Continuous Integration and Travis

Davide Falessi

Continuous Integration

- CI is a software development practice where members of a team integrate their work **frequently**.
- Each integration is verified by an **automated build** (including test) to detect integration errors as quickly as possible.
- Many teams find that this approach leads to significantly reduced **integration problems** and allows a team to develop cohesive software more rapidly.

Example of workflow

1. Select the ticket to work on
2. Branch (via SVN)
3. “Develop” the ticket (via Eclipse)
4. Test the ticket locally (via Eclipse)
5. Commit the changes and check the remote successfulness of the built (via Travis)
6. Fix eventual problems
7. Update ticket description (via GitHub)

CI in practice

Practices of Continuous Integration

- **Maintain a Single Source Repository**
- **Automate the Build**
- **Make Your Build Self-Testing**
- **Everyone Commits to the Mainline Every Day (or feature)**
- **Every Commit Should Build the Mainline on an Integration Machine**
- **Fix Broken Builds Immediately**
- **Keep the Build Fast**
- **Test in a Clone of the Production Environment**

Travis: Keyterm

- *build* - a group of *jobs*. For example, a build might have two *jobs*, each of which tests a project with a different version of a programming language. A *build* finishes when all of its jobs are finished.
- *stage* - a group of *jobs* that run in parallel as part of sequential build process composed of multiple stages.
- *job* - an automated process that clones your repository into a virtual environment and then carries out a series of *phases* such as compiling your code, running tests, etc. A job fails if the return code of the script *phase* is non zero.
- *phase* - the sequential steps of a job. For example, the install phase, comes before the script phase, which comes before the optional deploy phase.

Travis: Broken Builds

- The build is considered *broken* when one or more of its jobs completes with a state that is not *passed*:
  - *errored* - a command in the before install, install, or before script phase returned a non-zero exit code. The job stops immediately.
  - *failed* - a command in the script phase returned a non-zero exit code. The job continues to run until it completes.
  - *canceled* - a user cancels the job before it completes.
- The Travis Common Builds Problems page is a good place to start troubleshooting why your build is broken.

Travis: How To

- Go to hello world example, export, ANT
  - Now you should have a file called build.xml
  - This is required to tell Travis how to compile the files.
- Create a file called ".travis.yml"

language: java

jdk:

- openjdk8

script: ant build

Travis: How To

- Create a new repo on GitHub e.g., "TestTravis"
  - Make sure you add a readme file so that you can use SVN
- Go to Travis CI .org
- Connect with your GitHub account
- Select in Travis the Github repos you want to connect with Travis (TestTravis).
- Go to Travis CI .org build status
- Create a working directory
- Move the example to your SVN working directory trunk and commit it.
- Observe Travis CI .org build status
  - If it fails: troubleshoot.
  - If it does not fail, insert a bug in the code and make sure it fails.

Lab

- Setup Travis to automatically build a "hello world" project in GitHub (as shown in the slides).

References

- **https://docs.travis-ci.com/user/sonarcloud/**
- https://www.thoughtworks.com/continuous-integration
- https://martinfowler.com/articles/continuousIntegration.html
- https://travis-ci.org/