**Practical 8: Continuous Integration**

Agile teams, because they are producing robust code each iteration, typically find that they are slowed down by the long diff-resolution and debugging sessions that often occur at the end of long integration cycles. The more programmers are sharing the code, the more problematic this is. For these reasons, agile teams often therefore choose to use Continuous Integration.

**Continuous Integration** (CI) involves producing a clean build of the system several times per day.**Agile** teams typically configure CI to include automated compilation, unit test execution, and source control **integration**.

**Integrate your CI server with Pivotal Tracker and CodeShip**

We use Pivotal Tracker and Basecamp to communicate the progress, Github for our code, Codeship for continuous integration and continuous deployment, HipChat for our internal communication, and finally CodeClimate to ensure our code quality.

**CodeShip**

Codeship is a hosted Continuous Delivery platform. It helps to release software quickly, automatically and multiple times a day. It shortens the development cycles thus reducing the risk of bugs and increasing innovation. It helps software companies to develop a better product faster by taking care of the testing and release process. Managing your own continuous deployment system takes time, specialized knowledge, custom solutions and dedicated, on-premise infrastructure. Codeship automates software deployment and all the necessary tasks involved with it.

**Travis CI**

1. To implement Travis CI we need to sign in to Travis CI with GitHub account.

2. Accept the GitHub access permissions confirmations in order to allow synchronization between GitHub and Travis CI.

3. Let’s say there’s small module done using Netbeans. (A sample is created for testing purpose)

package app1;

public class App1 {

public static void main (String[] args){

int a=3, b=2, c=1;

if (a>c){

a = b = c = 1;

System.out.println(b);

}

}

}

4. The project is required to first initialize with Git in order to commit and push. This is done by selecting Team>Git>Initialize then click “Ok” button.

5. Select Windows>Output>output to view the initialization of Git with the project.

6. Once initialization is done, go to Team>Commit. Enter the Commit message and click “ok”.

7. With Git commit, the changes will be recorded to the repository.

8. Login to GitHub account and add a new repository. Enter the repository name and description then click “Create Repository”.

9. The page will then display a link where it can be used for git push. Copy the link.

10. Back to NetBeans, go to Team>Remote>Push in order to upload the remote refs along with associated objects.

11. Paste the link at the “Repository URL” and enter username and password according to GitHub account then click “Next”.

12. In Local Branch tick “master->master” to indicate the root. Then click “Finish”.

13. In GitHub, click on the newest repository created and the files for the project will be created.

14. In order to allow Travis C! to understand the project type, we need to create a new file at the repository name “.travis.yml”. Enter the language and sudo of the project.

language:java

sudo : required

15. Confirm that “.travis.yml” file is successfully created.

16. Login to Travis and synchronize Travis CI and GitHub.

17. Then allow Travis CI to access to GitHub repository in order to allow execute testing

18. In GitHub, go to Settings>WebHooks & Services select “Travis CI” under Services and click “test Services”.

19. Doing so, test payload will be send to Travis CI and the testing will be executed.

20. If success, they will prompt success message and email to you indicating successful testing.

21. If not, they will prompt failed message and email to you indicating a failure testing.

**Resource**

https://code.tutsplus.com/tutorials/codeship-continuous-integration-and-delivery-made-simple--cms-23517