Continuous integration build system for HiG

Mandatory Assignment 5

Per Christian Kofstad, 120916
Anna Kaardal, 120918
Synne Gran Østern 120922,
Joachim Hansen 120483,

1.1 What is continuous integration?

The world is changing, and fast. The plan driven approaches to software development often fell short to accommodate the business needs for fast delivery. Agile processes were created as a reaction for these shortcomings. One development practice that the agile processes adopted (one of the development practises of Extreme Programming) was continuous integration [1].

Here I will present an example that most students can relate to, that emphasizes the difference between integration and continuous integration. Imagine that your course teacher have given you a mandatory assignment. You and 3 other people group up and divide the tasks to work on them individually. Issues arise when members of the group might decide to write first and find sources later; write on the same topics, using conflicting methodology, using a different type of language (e.g. technical and natural) and so on. When the group finally meets and tries to get all the **components** of the report to work together as a whole, a lot of effort need to be put in by all members to fix these issues (none continuous Integration). Alternatively if the students all worked on the same document (version); wrote their task complete with spelling and sources and checked (tested) to see if this task (component) fits into the current report. Then delivering a draft to the teacher within a short time frame would be a non-issue (continuous Integration). You can draw parallel between this example and a software developing team integrating components into the whole system as a last activity and not as an ongoing process.[2] [3] [4]

---

[1] Ian Sommerville, *Software Engineering* (9th Edition): 57-59,65-66

[2] AgileAllience, *Continous Integration*, http://guide.agilealliance.org/guide/ci.html Timestamp: 04.11.2014

[3] AgileAllience, *Integration*, http://guide.agilealliance.org/guide/integration.html, Timestamp: 04.11.2014

[4] Martin Fowler, *Continuous Integration*, http://www.martinfowler.com/articles/continuousIntegration.html: 2066.05.01

1.2 What enables continuous integration for software development?

Developers/programmers use version control software (e.g. SVN and GIT), programs that runs automated test (unit testing, acceptance testing, system testing) and programs that can automate the build the process, as part of their toolbox to enable continuous integration. Automating tasks that are time consuming and having a shared repository that you can build from makes the effort of integrating less painful . [5] [6]

Benefits and shortcomings of continuous integration:

+ Enables frequent deployment

+ Reduced risk for failing to deliver, do to having a better understanding for how long time the integration process will take.

+ By introducing good automated regression tests, you can discover more security flaws and bugs.

+ Version control software makes it possible to compare versions and by doing so you can more easily find the root of problems and alternatively go back to a previous version. By connecting all tests and documentation to each version it should be easier to search for the root of the problems, rather than only search through source code versions.

- Integration challenges may occur when the average time for the automated test take longer than the average time between commits.
- Cost of hardware and software.
- Another system to maintain for the IT department.
- Another system to learn for the teachers and students.

[7] [8]

1.3 What similar solutions exist today?

It seems to be many solutions for continuous build systems available for anyone to buy or use. You find both paid alternatives and open source alternatives, and apparently they all do

---

[5] AgileAllience, *Continous Integration,* http://guide.agilealliance.org/guide/ci.html Timestamp: 04.11.2014

[6] Martin Fowler, *Continuous Integration,* http://www.martinfowler.com/articles/continuousIntegration.html#PracticesOfContinuousIntegration: 2006.05.01

[7] Same as fotnote above

[8] Rick Kitts, *This is Rediculous Continuous Integration Hell*, http://www.artima.com/weblogs/viewpost.jsp?thread=30031: 2004 .01.22

the job. These are some examples of systems that is high rated, and found when doing web-search for continuous build systems.

TeamCity - Free version, Enterprise Server version. You need to install it separately.
Bamboo - Free trial - Paid option, everything happens on their servers.
Buildbot - Standalone - Free to use installation.
Hudson - Old open source solution, integrates well with eclipse.
Team Foundation Build - Microsoft Team Foundation Server.
Jenkins - Open Source solution.

We found that there are both simple stripped-down systems and heavy integrated systems that contains a lot of advanced functionality. It is both paid and open source versions, and there is free to use, paid support financed versions.

2 Continuous build system for IMT department at HIG

The system which we will be sketching in this assignment, is supposed to be used by HiG/IMT students to commit their workings to a common repository, where version control is performed. The system will be used for project work, other forms of group works, obligatory assignments, and exams. Workings can be committed to the repository both as individual and in groups. Automated build and testing will be performed on regular basis, making sure that the code still works as intended. The teacher will be able to retrieve the source code and binaries, and will be able to comment on, grade and give feedback on the work to all the members of the group. In the end, programs will have the ability to be deployed to web servers and app stores.

2.1 Our solution

We do in the following section present our solution for the continuous build system.
Our system uses the Jenkins open source application as a tool for continuous integration.[9] In addition to this, the system is adapted to the requirement of the HiG/IMT.

[9] Jenkins CI, https://wiki.jenkins-ci.org/display/JENKINS/Home. 2014.07.22

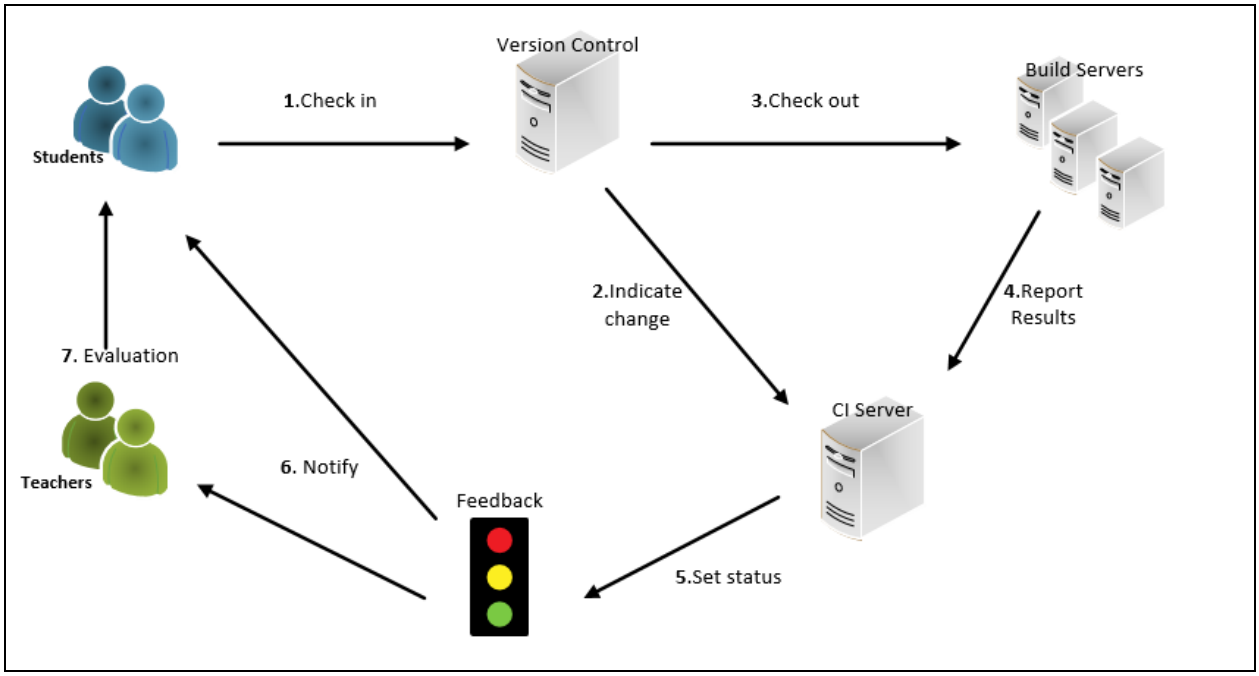

Figure 1: A graphic presentation of the system, based on the figure at
http://www.programmingresearch.com/products/continuous-integration-jenkins-plug-in/

Description of Figure 1:

1. **Check in:**
   A student checks in his files on a regular base. Subversion is then performing version control on it.
2. **Indicate change:**
   After the version control is performed, source code changes are sent to the CI server.
3. **Check out:**
   After the version control, the new version is checked out. For every check out, a build is triggered. We have chosen Jenkins to perform contiguous builds.
4. **Report results:**
   The results of the build (successful/ failed) are reported to the CI server.
5. **Set status:**
   Status from the build is set, indicating whether it was successful or not.
6. **Notify:**
   Based on the status, feedback from the build is generated, and the users are notified.
7. **Evaluation:**
   The teacher is able to evaluate his student`s workings, decide whether it meets the requirements, grade, comment, and give feedback to all the students who are members of the group. After the evaluation, the students can choose if they want to deploy the project to web servers and app stores.

2.1.1 The different elements in figure 1:

Students:

Students are in this system both the developers and testers. They work locally at their own host, and uploads the code to the software through version control. Students can also act as a oversee and directly active in the development.

Teachers:

Teachers can retrieve and see the source code and the binary code, however, how active the teacher is in each project depends on the teacher and what kind of project. E.g. the teacher can function as a manager or even a "customer" in some projects, whereas in others he/she functions as a overviewer and does not take any active role. But in additional to this "traditional" role, the teacher can also act as a developer, and publish code that the students are going to build upon.

Version control:

This system uses SVN (Subversion) integrated to the Jenkins solution. This makes sure the students (if they use it correctly) gets the latest changes, and can add their own code into the software without corrupting the existing code. Additionally it also provides with a history where both teachers and students can see the actions over time.

Build Servers:

The builds gets triggered each time a student do a check out. Jenkins do provide a periodically build (e.g. each night),[10] ( ). But as students often have irregular work hours and want to have the possibility to work at any time they like without getting disturbed by a build, we have chosen this system to only perform triggered builds.

CI server:

CI server (continuous integration server) is in many ways the heart of the system. The CI system collects information from the version control server, and the builds from the builds servers. The CI server may then run analysis tools such as static code testing, developer premade testing of the code (such as junit tests), testing tools for code quality (such as SonarQube [11]). Then all the information related to a submitted version in the repository is present in a structural way. This makes it easier for developers, testers, team leaders, managers and clients to get a status on what state the code is in, and whether or not the project is on schedule.

Feedback:

The system will generate feedback to the users (both teacher and the group members/ students) concerning the build and whether there are conflicts according to the earlier version. In this way, the users know if the project is still able to be built after the revision, and whether the revision led to any conflicts.

---

[10] Kohsuke Kawaguchi, *Building a software project* https://wiki.jenkins-ci.org/display/JENKINS/Building+a+software+project , 2014.05.21

[11] SonarQube, http://www.sonarqube.org/, Timestamp: 2014.11.04

2.1.2 Automated functional testing

The automated testing happens at the CI server in figure 1. Jenkins uses a harness based on JUnit. This is to make the test development simpler. [12] As long as the developer implements JUnit-tests in the code correctly, these will be runned automatically and generate reports.

2.1.3 Deployment to web servers and app stores

When (or if) the project reach an acceptable level according to the requirements, and the teacher have approved the software for deployment, the student can (or if required in the project), deploy the software. The figure 1 does not show this process in detail, but the CI server deploy the software when this is required. However, the deployment can not happen before the testing is complete and approved. After a software has been deployed, it is still possible to continue with the development.

2.2 Legal, organizational and technical security requirement

In this section, we will define the organizational and and technical security requirements for the system.

2.2.1 The software

The software shall not be a platform to distribute intellectual work that is protected by the Norwegian copyright law. Distribution should only occur when the owner have given permission for this use. Computer software is explicitly covered by “åndsverkloven” §1 (12 and distribution is deemed illegal by § 53 a) b) - åndsverkloven. The teachers have more freedom to use Copyright material for educational purposes. The course teacher can therefore distribute material used for this purpose to his students § 13 - åndsverkloven. [13]

If the integration software is sold as a solution, then it is regulated by the Norwegian law “forbrukerkjøpsloven” § 1. Given that a contract between buyer and seller have been successfully negotiated (e.g. about software performance and availability) then the software should stay true to the contract and to what a user can normally expect to a software of this type §15 a) - “forbrukerkjøpsloven” [14]

---

[12] Kohsuke Kawaguchi, *Unit Test,* https://wiki.jenkins-ci.org/display/JENKINS/Unit+Test , 2014.06.11
[13] *Åndsverkloven* - åvl, http://lovdata.no/dokument/NL/lov/1961-05-12-2: 2014.07.01
[14] *Forbrukerkjøpsloven,* http://lovdata.no/dokument/NL/lov/2002-06-21-34/KAPITTEL_9#KAPITTEL_9: 2002.06.21

HIG as an academic institution is required by law to make sure that their researchers (e.g. Phd students, master students and teachers) follows recognized ethical norms §1 - Forskningsetikkloven. The system needs a way to check whether their user have copied text from a online resource and presenting that as their own work. [15]

2.2.2 Requirements of users

Version control has to be performed, so the users can make sure that their code are not overwritten. They do also have to be able to go back to earlier versions of the code when necessary. Others than member of the group shall not have access to the work.
The teachers has to be able to retrieve their student`s workings as binary and source code.
The system has to be available for the users with minimal down time.
Like mentioned in the section above, the system needs some kind of plagiarism check, so the teacher can make sure that the students complies with HiG`s provisions for plagiarism.

2.2.3 Requirements of system administrators

Legal requirements for the administrator role is that the system policy must specify what access the administrator has to the repositories content. Access rights for the administrator role must also be clearly defined in the user policy signed by teachers and students as users of the system. The administrator should have access to enough information to be able to see system faults and system problems, so that it is possible for the administrator to handle the incidents and operate the system in the best possible way.

Technical requirements for administrator role could be divided into sub-roles, one for software and one for hardware. Hardware for enabling infrastructure to run the system, possibly as an integrated part of the rest of HiG's IT systems, we will not focus on the hardware role and requirements, since we see this as the same as existing requirements as of the current IT systems on HiG.

Requirements for software administrators of the systems should be to have enough knowledge of the system to be able to run the system in order to meet the requirements of uptime during work hours. All administrators should accept electronically or sign manually a terms and conditions for the administrator role of the system.

As an organizational security measure, the organization must always ensure that there is always an acceptable level of knowledge about the system among the technical staff, and users.

---

[15] *Forskningsetikkloven*, http://lovdata.no/dokument/NL/lov/2006-06-30-56 :2006.06.30

2.2.4 Requirements of the institution

The technical requirements for the institution is much the same as the requirements for system administrators. As of specific requirements for the institution in terms of technical and organisational on a security perspective there should be pointed out a person or a role that is responsible for the continuous build system system as a whole.

Also there should be a policy of how to handle updates and maintenance of the system, to ensure both stability and security. If there is a risk analysis of the system, there should be a periodic revision of this risk analysis.

According to “Copyright Act” where the legal requirements for copyright and usage is specified.[16] The institution as a whole, HiG will stand responsible for any violation of the user agreements, terms and conditions. This is especially important when it comes to the agreements according to protection of the code made by students and teachers. If a teacher or a student do not allow the code and compiled versions to be available for open distribution then the system should handle this without any possibility of workarounds.

2.3 Trust

As Ken Thompson says in Reflections on Trusting Trust[17] “You can't trust code that you did not totally create yourself.“ He argues that unless you as a programmer has been writing the whole system yourself you can not be sure that anyone else has written bug free code. Also he argues, if you yourself has written the code for the whole system, you are also likely to have written some bugs in the program. In other words, it is really hard to create the perfect secure, bug-free system. There has to be a trust in the system in many ways when you have an continuous build system. As a student using the system, you must trust that the system protects the code according to the agreed terms and conditions, also that the system uses a compiler that is as trustworthy as expected from a compiler compiling student work. Both students and teachers must trust that the compiler and server is clean so that bugs and malware is not compiled into the code and makes programs malicious or bad.

The question of who decides what is trustworthy or not, is a hard question, can be discussed in many ways. In the matter of what compiler and testing software is good enough for this solution the answer might be, as long as it is similar to what students would have used on their private computers, that is good enough.

---

[16] Same as fotnote 13

[17] K. Thompson (1984). *Reflections on trusting trust. Communications of the ACM 27(8)*:761-763 http://dx.doi.org/10.1145/358198.358210

We could in addition consider Martin Naedele and Thomas E. Koch article Trust and Tamper-Proof Software Delivery. Here the "Trust model" is presented as a way to evaluate the trust of the software. The different types are: Liability, Reputation, Strong interest, Weak interest, proven in use, directive, blind and idealism. [18] In this case the key element are: Reputation; as a well known information security center, HIG stands out, and will therefore lose some of its well earned reputation as a highly regarded educational institute. Strong interest also have a significant value here. The students who develops, will most likely have strong motivation to make a software that the users will be satisfied to use. Many students are quite fresh in the IT-field and will want to prove to others at HiG and potential employer that they are capable to produce software of good quality.

2.4 The ICT Supply Chain

A supply chain is referred to as "a system of organizations, people, technology, activities, information and resources involved in moving a product or service from supplier (producer) to customer[19]. The ICT Supply Chain concerning supply chains in the Information and Communication Technology.

Most systems of today obtain software components and other artifacts from different suppliers[20], the system we are sketching in this assignment is no exception, because it uses Jenkins, SVN and SonarQube. These components might again rely on other components from other suppliers, from all over the world. On the way to a finished product, a supply chain is formed. You can never be completely sure that the components provided in the different parts of the supply chain are secure. Code might for instance contain malicious functionality or vulnerabilities, either intentionally or by a mistake. The question is; to which extent can you trust the suppliers and the services they provide, and how can you know their trustworthiness?

To avoid vulnerabilities from entering the supply chain, it is important that the different parts are following standards. Many ICT standards exists, note that none of them are actually specific to supply chain, but still relevant. Some examples is the ISO/IEC 27036 standard which contains guidelines for security of outsourcing, and ISO TC247 which concerns fraud controls and countermeasures[21]. These might both be important to ensure trustworthiness of the different suppliers.

---

[18] Martin Naedele and Thomas E. Koch, *Trust and Tamper-Proof Software Delivery*. http://dl.acm.org/citation.cfm?doid=1137627.1137636 2006.
[19] ENISA, *Supply Chain Integrity,* https://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/ , 2012.11.15
[20] Martin Naedele and Thomas E. Koch, *Trust and Tamper-Proof Software Delivery*, http://dx.doi.org/10.1145/1137627.1137636, 2006
[21] ENISA, *Supply Chain Integrity,* https://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/ , 2012.11.15

If we evaluate the Jenkin application with the “Trust Model”[22] and the type of rationales, it is here relevant to consider the “Proven in use” element. Referring to the Jenkins page “Who are using Jenkins”[23]. We see that it is a widely used software, and can because of that be considered as proven in use. With so many users, if responsible for harmful artifacts, would be severe for the company`s reputation.

2.5 Code signing

To ensure that the code and compiled program is actually from the user that claims to have made it, the CI system should have some sort of signing mechanism. Preferably both on the code and on the compiled package. As a user of the code, in this case, a student that is viewing example code of a program or a teacher that is viewing student assignments. Both parts will gain on the assurance that the code or program they are testing is the program claimed to be made by the specific user. The easiest way might be to sign the code, then all parts just have to trust that the system do not mess anything up during the automate build process. A student signs the code with a hash of the code and private key, then sends the certificate with the code into the system. Then the teacher, reviewing the code can verify that it is infact made by the student that signed it. If the certificate matches against the student`s public key and the hash of the version of the code that the teacher has.

We do signing in order to achieve two things; authenticity and integrity. Authenticity in order to confirm where the code came from, who has made it. Integrity in order to ensure that the code has not been tampered with.

3 Conclusion

In this assignment, we have sketched a continuous build system for HiG/IMT. We have looked at security requirements for the system, in user- , institution- and administration perspective.

There is not many directly legal requirement regarding the system itself. But both the configuration and policy have to be implemented in a way that satisfies legally.
The system will use services from different suppliers (Subversion, Jenkins, SonarQube). We have evaluated what can be done ensuring trust on general basis, and discussed the challengers the ICT supply chain meets. We have also discussed the trust factor both when it comes to the developers (students) and the suppliers to the system. And finally we discussed the code signing solution.

It is not easy to define what a continuous build system should consist of in terms of security

[22] See footnote 20
[23] Michael Prokop, *Who is using Jenkins?,* https://wiki.jenkins-ci.org/pages/viewpage.action?pageId=58001258, 2014, 10. 31

measures and policies. It is also hard to define what is good enough for HiG as an institution, and in order to implement a continuous build system, we recommend to define what it should be used for and by whom. Then we would recommend doing a risk analysis to find what measures would be the most effective ones.

However, our impression is that a continuous build system for educational purposes, will bring many benefits for both students and teachers and might be quite realistic and relevant regarding students future career.

How we worked

We used a lot of time, researching and reading about continuous build systems. After this was done we discussed different solutions, and divided up in different parts of the assignment to work on. When we worked with the assignment we sat together and worked parallel with different parts of the document. We helped each other, and commented, rewrote each others text during the process.

Anna
Worked on ICT Supply Chain, and the introduction to Continuous build system for IMT department at HIG. I did also work on our solution for the continuous build system, and a little at requirements.

Synne
I have designed the figure and the system based on Jenkins' design. Additionally I have described the objects in the system. Also the testing and deployment part, and the Trust model argumentation.

Joachim
Worked on introducing the user to continuous integration ("What is continuous integration?","What enables continuous integration for software development?") and worked on legal requirements for the software integration.

Per Christian
Worked a lot with trying to map what systems is popular and widely used today, and shortly about what is the difference between different types systems. I wrote shortly about my findings in "What similar solutions exist today?". Also I worked with the requirements for system administrators and institutions, trust and code signing.