Oblig #5 - Continuous Build System

120505	-	Jan Petter Berg Nilsen
120914	-	Jannis Schaefer
120917	-	Eirik Vestreng Solberg
120920	-	Brage Celius

November 5, 2014

Abstract

In this report we sketch a continuous build system using a private/hybrid cloud for HIG/IMT. The report will cover various topics such as system-, tecnical-, legal-, and organizational requirements. As a conclusion, we will end up with a system architecture to be used for the continuous build system.

Contents

1. Introduction

1.1 Task

We have been assigned the task of sketching a continuous build system using a private/hybrid cloud for HIG/IMT.

1.2 Partition of Work

As instructed, we will provide a list describing the Partition of workload. It is however important to note that the group has worked together for the entire project and the list should be regarded as guiding rather than definite. For the "chapter" tasks, "members" refers to the one who wrote down that part of the article.

Task	**Members**
Discuss and get to know the task	Everyone
Discuss and Draft all requirements of the system.	Everyone
Search for similar software	Everyone
Discuss the system Architecture	Everyone
Chapter 1	Brage Celius
Chapter 2.1	Jannis Schaefer
Chapter 2.2	Brage Celius
Chapter 2.3	Jan Petter Berg Nilsen
Chapter 3	Eirik Vestreng Solberg

Table 1.1: Partition of Workload

2. Technical, Legal and organizational requirements

2.1 Technical Requirements

The system has a range of technical security requirements which can be categorized into *confidentiality*, *integrity* and *availability*. They are based on the principle that the system should continue to perform as expected.

2.1.1 Malware

The system needs to be protected from malware by quick detection and removal of such. Infection of a file may spread to any other files and corrupt or at least disrupt many projects. Malware threatens the confidentiality, integrity and availability of the system, as many alternations are able to access files throughout the system, reading and possibly modifying them.

2.1.2 Secure Authentication

Users and systems must be ensured to possess the identity they claim to have. A lack of proper authentication opens up to impersonation, which results in loss of confidentiality and aids loss of integrity and availability.

2.1.3 Encryption of communications

Any communication (e.g. authentication, file transfer) must be protected to ensure confidentiality and integrity of the information transferred. Unencrypted communication may be read and modified by an attacker, opening to a range of security problems including, but not limited to: Introduction of malware to the system, falsification of authentication and information loss.

2.1.4 Backup

Any interruptions of service have a great likelihood to occur in such a way that information may be lost. It is possible that files may be corrupted or damaged in other ways. Further may physical components be impaired as well, prompting replacement.

In such cases it is imperative to have redundant copies of files and possibly equipment to ensure the smallest loss of availability possible within the constraints of cost versus benefit.

2.2 Legal Requirements

When designing the system, there are several legal requirements that must be fulfilled. The most important of these are protecting the privacy and copyright of users in accordance with the Norwegian Laws. [2] [3] Protecting the privacy and copyright of the users is extremely important because neglecting to do so may result in legal actions against the system provider.

2.2.1 Privacy

- User information should be protected and stored securely.
- Only the users themselves should have access to their private repository.
- Users should be able to share their documents with a select group of students or teachers.

2.2.2 Copyright

- Users, or a group of users, should have copyright ownership for the documents they create.
- Only the copyright owner(s), or HIG as an organization, should be able to publish a document.
- When sharing or publishing documents, they should be marked with information regarding who is the copyright owner.

2.3 Organizational Requirements

When using this type of system there are some organizational requirements which support the students, teachers and the institution. These need to be fulfilled. It is important for the organisation to maintain both the user and system security rights and ensure their confidentiality, integrity, and availability. The most important recruitments revolve around policies that the organisation has made, so that users have a defined set of rules and guidelines regarding usage of the system. These policies could have different sub categories, such as privacy and copyright.

If the organization does not develop and follow these types of requirements, it can result in big consequences for both the users and the institution. In the worst case scenario results could incorporate legal action that could cost the institution both money and reputation.

Organizational requirements for our system

When we look at the security requirements which are needed for this type of system from an organizational perspective, they are:

- The policy should have rules on who can access what on the system, so that every user has its own repository where only the right users have read and write access. This could help to give privacy to students who submit materials to the repository. It would also prevent other users from taking credit for work they haven't done, so that they as a student cant cheat and copy of other students.

- The policy should have a guideline for what a user can submit to a repository or not. It could also be a requirement to check signatures of materials before they get uploaded to the repository, so that it would be less possible for malware and copyrighted materials get uploaded. A possibly prevented situation could be that a student submits malware to the repository or copyrighted materials. This action could lead to the loss of the institutional license to app-stores [1] [5] or other liabilities.

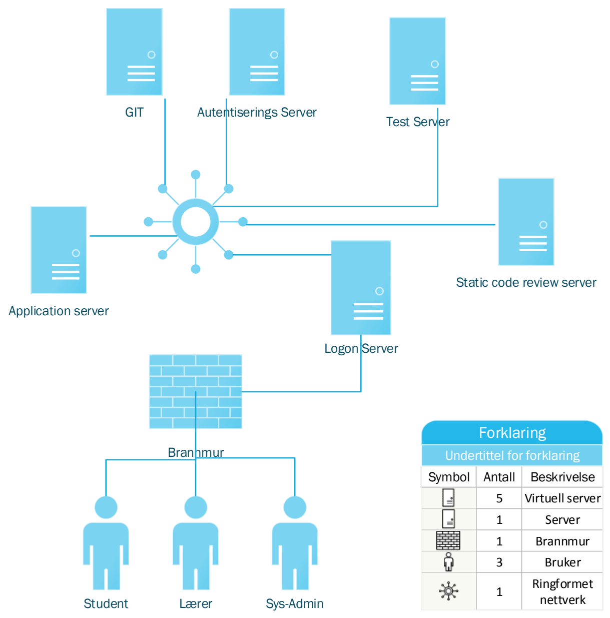
- For the system to use, maintain and upgrade the different types of security mechanisms, is it necessary for the organization to provide the right amount of specific documentation to the system administrator. This will benefit the institution, because it will offer a clear guideline, so that the different security mechanisms will be implemented and maintained correctly, and maintain their purpose as long as they will be used.

3. System architecture

3.1 Architecture introduction

I will in this part describe the system architecture and some thoughts around the design choices we have made. there will be section for both network topology and use case diagrams for the logical architecture and sequence diagrams to explain the communication between the components of the system.

3.2 Network topology



Forklaring

Undertittel for forklaring

Symbol	Antall	Beskrivelse
▯	5	Virtuell server
▯	1	Server
▦	1	Brannmur
👤	3	Bruker
☼	1	Ringformet nettverk

Figure 3.1:

As seen in figure[3.1] it's a simple network topology builds on a proxy architecture. This is to insure the security of the system. behind the login server an internal switch to the virtual machines which each represent a service in the cloud system.

The application server purpose is to store the binaries of running applications published in app-stores and orchestrate updates and bugfixes.

The Git server handles version control of both published and unpublished projects. This is thought as a replacement to the SVN server and is easier to integrate with binaries.

The test servers purpose is to create a test environment to test application and programs before release to simulate real life software development and to catch eventual errors or vulnerabilities before publishing. For this server the student/coordinator will have to write test cases themselves as in test driven development.

Static code review server runs a static code review software with the ability to review all of the languages GUC teaches/publishes. The sofware we have found to be a good alternative is the DMS® Software Reengineering Toolkit^{TM}[4]. Teacher will with this tool be able to decide which rules it enforce and even add own rules if there are special restrictions or obligations on the students applications.

Authentication server is pretty much self explanatory and I will not go into any details in this assignment.

3.3 Use Cases

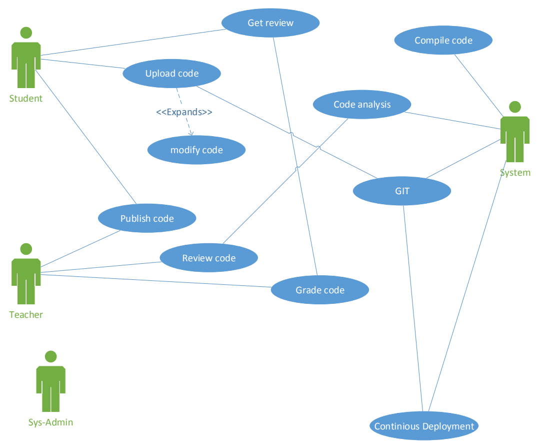


Figure 3.2:

The figure [3.2] shows the most usual traffic/use of the planned architecture. The student use of the system is the basic user and all other roles have the same rights only extended.

3.3.1 User Stories

I
- Student writes an assignment using GUC's Git server as version control.
- Student sets the project as deliverance on Fronter by supplying an ID to the project in Git.
- The System runs static code review and compiles the code before giving the source, binaries and a review report to the teacher
- Teacher download the project folder and reviews/grades the assignment and uploads the grade.
- Student gets notification that the assignment is finished reviewed and finds logs in and find the grade and code review from teacher.

II
- Student works on a project and wants to make a review of the code and uploads the code in Git and sends signal to the code review server with the ID of the project.
- The code review server reviews the code and sends the report back to the repository and notifies the student.

III
- Teacher likes an application made by a student and wants to upload it to a app store and notifies student and asks for permission.
- Student gets question and either grants permission to upload the binaries or not
- If permission granted binary moved to the application server and pushed up to the app-store. and test enviroment for updates is created

IV
- Student has an app in the app-store and wants to update it with new functionality. After writing the extension compile it and put it in the test server
- The test server runs the test-cases uploaded with it and returns a report and crash reports if there are any. Repeats until satisfied with results
- Student notifies teacher/coordinator to review changes and approve changes before pushing the new patch to the application server and to the app-store

V
- Student upload malware
- Code review server flags it as malicious code and notifies Teacher and Sys-Admin
- Teacher/Sys-Admin reviews the flagged content and determines if it was a false positive or malware
- The faculty is notified and reacts after procedure

3.4 System Communication

The system communicate through a virtual network as all of the servers are virtual. In this section i will go trough how the use cases is seen as communication.

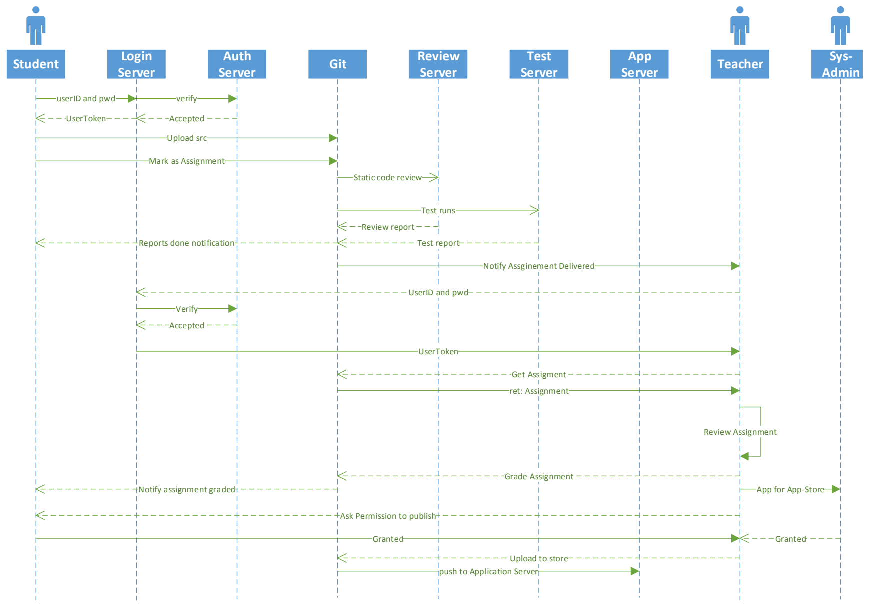


Figure 3.3: Communication between modules in the system

This model shows everything happening from the student uploads source code till its reviewed graded and uploaded to app-store. the update use-case has the same kind of communication and I will therefore not have a new figure to illustrate it. In The event of malware upload the sequence diagram will look something like this figure: 3.4

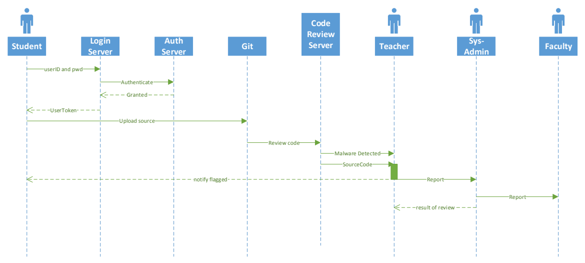


Figure 3.4: Sequence if Student uploads malware

4. Conclusion

In our report we proposed a possible solution which we believe to accommodate the presented use-cases. We identified technical, legal and organizational security requirements which have to be taken into consideration when implementing our solution.

Bibliography

[1] Apple. Apple developer terms of agreement. https://developer.apple.com/programs/terms/registered_apple_developer_20100301.pdf, November 2014.

[2] Norway Department of Justice. Personopplysningsloven. https://lovdata.no/dokument/NL/lov/2000-04-14-31, 06 2013.

[3] Norway Department of Justice. Åndsverksloven. https://lovdata.no/dokument/NL/lov/1961-05-12-2, 07 2014.

[4] Semantic Designs. Dms® software reengineering toolkit™. http://www.semdesigns.com/Products/DMS/DMSToolkit.html, October 2014.

[5] Google. Android developer terms of agreement. https://developer.android.com/preview/license.html, November 2014.