Gjøvik University College

Software Security

Group Project

Continuous Build System

Authors:

Victor Rudolfsson - 120912

Halvor M. Thoresen - 120915

Tommy Ingdal - 120913

November 5, 2014

Contents

1 Introduction

1.1 Task

For this assignment we have been given the following task.

Sketch a continuous build system using a private/hybrid cloud for HiG/IMT.
Consider the following functional requirements:

- a) Students submit code to a common repository
- b) Build system compiles source to binaries
- c) Teachers retrieve source code+binaries
- d) Deployment to web servers and app stores
- e) Automated functional testing

What are the legal, organizational, and technical security requirements in software integration?

- a) Requirements of students?
- b) Requirements of teachers?
- c) Requirements of system administrators?
- d) Requirements of the institution?

1.2 Workload

We have divided the work over the three members of the group. Tommy and Halvor worked together on the *Legal, Organizational and Technical Security requirements*, with Tommy focusing on the *Technical Security Requirements*, and Halvor on the *Legal and Organizational* aspects, while Victor wrote the *Architecture* part.

2 Architecture

For the build system, we have chosen two different software solutions to be integrated with one another, and configured with Puppet for easy administration. These will then be monitored using a combination of Check_MK and Nagios, called Open Monitoring Distribution [1].

These 4 softwares will run on a few different servers: *GitLab* and *Jenkins CI* are kept on different machines, and OMD is set up on both. This will reduce workload from the Jenkins CI machine building and testing software from affecting the reliability of the version control system (*GitLab*) running on the repository server. Furthermore, Jenkins will use two separate machines: one to build and one to run tests. This network can be set up as an internal subnetwork at HiG, accessible via VPN. [2]

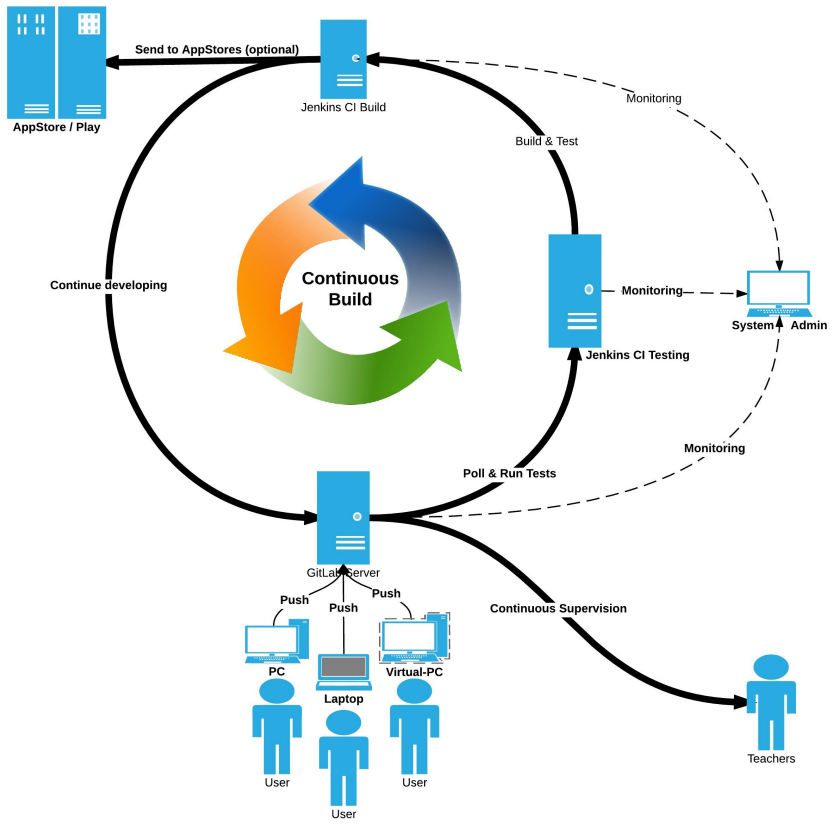

Figure 1: Continous Build Diagram

2.1 GitLab

GitLab, a self-hosted GitHub solution, is set up on one machine with teachers corresponding to administrators. As Administrators, they are allowed to set up Groups using the code for each Course and the year (e.g IMT101-2014) as naming conventions for the groups.

Now, with each Group corresponding to the current instance of the class, every member of this group is allowed to browse and create projects, so long as they're set as Masters. Students will thus be set as Masters, and allowed to create their own Projects, in which one or multiple students may collaborate. [3]

This allows teachers to easily find students in the courses they're teaching, and students to create and manage their projects independently while being supervised by the group's owner (Teacher).

2.2 Jenkins CI

Jenkins CI will be set up on a separate server, and configured to monitor repositories in the groups it has been given on the GitLab server. When a new *Group* (and/or repositories in this group) is created, it will be added to *Jenkins CI* using *System hooks*. [4] [5]

Jenkins monitors repositories for changes, continuously runs tests and builds the projects, while marking the success of these in the projects using Jenkins Build Status plugin [6]. This allows the status of Builds and Tests to be shown in the README of each project.

With plugins for Google Play & AppStore, Jenkins CI may be configured to automagically push a project to the AppStore or Google Play if tests and build passes. [**?**]

2.3 Open Monitoring Distribution

To keep an eye on these servers, *Open Monitoring Distribution* (containing Nagios, nsca, check_nrpe, Icinga, Shinken, NagVis, pnp4nagios, rrdtool, Check_MK, MK Lives-

tatus, Multisite, Dokuwiki, Thruk, Mod-Gearman, check_logfiles, check_mysql_health, jmx4perl, check_webinject and check_multi) will be deployed on both servers. [1]

This allows the system administrators to monitor both servers easily.

3 Legal, Organizational And Technical Security Requirements

3.1 Legal

HIG as a public institution has a responsibility to follow all legal requirements related to the system they run. Failing to do so can result in legal actions against, loss of revenue and loss of repitation. Most legal issues related to a development and distribution system are based around privacy or copyright laws.

3.1.1 Copyright

- The original users or developers should keep ownership and copyright of all creations. This is to prevent lecturers, admins or others with access, to take credit for the work. [7]
- The developer should always be in control over what happends to the creation. If the creation is to be published, the original creator or developer should be given full credit.

3.1.2 Privacy

- Only the users and the lecturer or admin should have access to the users files until they are published.
- The users should be able to control who has access to all their own files.

3.2 Organizational

The organization should have stong and detailed policies to avoid conflicts, issues and to support the students, teachers and admins.

- As the organization will be hosting the system on their own servers, an availability policy is needed to unsure that the system is running uninterrupted and is

accessible for the users as much as possible.

- A storage policy is required to define for how long information will be stored in the organization's systems.
- A policy to define what the users can submit to the servers should be implemented to ensure stability and ease of maintenance for the system admins.

3.3 Security

When designing a system where different people can upload code and get it compiled, you really need to think about the security requirements of the system. You should implement some sort of mechanism which protects the users from security breaches (e.g malware) and also encrypts the communication with the repository.

3.3.1 Authentication

When you use the HiG network you have to use your studentID and password. This mechanism should also be implemented in this build system such that the administrators (i.e teachers) have control over who can upload code for compilation.
Since HiG already have a good authentication protocol the build system can just implement this existing solution.

3.3.2 Malware

Whenever a user can upload code that's compiled on the fly, you have to consider the fact that the code may be malicious. The build system should implement a feature that scans the code *before* it's compiled, in order to remove potential malware (i.e trojans, worms, viruses etc).
If you compile and test a program without doing some sort of scanning of the binary, any user who uploads code can easily own the server and thus compromise the confidentiality and integrity of other users.

3.3.3 SSL Protocol

SSL should be implemented and enforced whenever a user is communicating with the repository (i.e build system), such that the communications can't be intercepted by a third-party.
If the users send everything over HTTP (clear-text) this information can potentially be intercepted and altered by a MiTM attack.

3.3.4 Redundancy

In a continuous build system you should implement some sort of redundancy. If you have one main server and this server goes offline for some reason, users can no longer get their code to compile. This obviously is not a good idea, since an important project may get delayed.
If you instead distribute the build system over several servers and implement a redundancy plan, you will potentially increase the total uptime and make sure that the users can work without problems.

4 Conclusion

In this paper we have proposed a solution to a continuous build system using Gitlab, Jenkins and Open Monitoring Distribution, with Puppet for easy deployment.
We have also disussed the legal and organizational requirements related to this solution.

We believe that the proposed solution, using available technologies, will give the users and administrator great control of their source code, as well as the compiled binaries/applications.

References

[1] OpenMonitoringDistribution. OMD - The Open Monitoring Distribution. http://omdistro.org/, 2014. [Online; accessed 02-November-2014].

[2] Jenkins CI. Distributed Builds. https://wiki.jenkins-ci.org/display/JENKINS/Distributed+builds, 2014. [Online; accessed 05-November-2014].

[3] Ben Bodenmiller. GitLab Permissions. https://gitlab.com/gitlab-org/gitlab-ce/blob/master/doc/permissions/permissions.md, 2014. [Online; accessed 05-November-2014].

[4] Vanja Radovanović. Gitlab Hook Plugin. https://wiki.jenkins-ci.org/display/JENKINS/Gitlab+Hook+Plugin, 2014. [Online; accessed 05-November-2014].

[5] GitLab. GitLab System Hooks. http://doc.gitlab.com/ce/system_hooks/system_hooks.html, 2014. [Online; accessed 05-November-2014].

[6] Dieter De Meyer. Jenkins Embeddable Build Status plugin. https://wiki.jenkins-ci.org/display/JENKINS/Embeddable+Build+Status+Plugin, 2014. [Online; accessed 05-November-2014].

[7] Norway. personopplysningsloven. https://lovdata.no/dokument/NL/lov/2000-04-14-31, 2000. [Online; accessed 05-November-2014].