How to Test Using Jenkins?

Raphaël Castanier
LTH
Email: raphael.castanier@grenoble-inp.org

Lars Gustafsson
LTH
Email: ada10lgu@student.lu.se

***Abstract*—Jenkins, the open source automation server, is used in junction with a multitude of plugins to support building, deploying and automating software projects. This useful tool has a large community-based support and is widely used in industry and research. Jenkins is mainly used in Continuous Integration/Continuous Distribution workflows, but can also be implemented for various other applications. Because Jenkins is highly modulable with its plugin system, it allows many automated process to be run automatically. We will focus in this study on Jenkins usage for software testing, both on test unit generation and automation. We implemented a Jenkins CI study case on LaTeXfile generation (*this paper*). Finally, we will introduce Jenkins Acceptance Test Harness (ATH).**

I. Introduction

Jenkins [1] is an automation software. This open-source project provides an environment to automate tasks on various projects. Its scalability and robustness makes it used in several industrial environments, university world and widely spread in the software development projects.

One of Jenkins strenght is its ability to be extended through plugins. Indeed, the widely community built around Jenkins developed a lot of plugins (more than 1500 during Dec. 2019) for various applications and answering spceific domain needs. Plugins are designed to automate specific tasks such as using Version Control System to retrieve source code, building environment for several platforms or perforing large number of unit tests.

We can talk about Unit Test Automation with Jenkins-CI tool [2]. Jenkins ability to automate tasks is really powerful to automate unit testing. It allows developers to run large number of various tests in different environments to ensure unit tests are correctly run. In Continuous-Integration (CI) processes, Jenkins automates build, testing and release steps, allowing developers to save their time from this redundent tasks and providing monitoring along software product life-cycle.

Jenkins is also studied in academic researches, presentig its strenghts and its weakness. Consequently, Jenkins has been improved for its own unit testing solution, the Acceptance Test Harness (ATH).

This paper will present a related work about Jenkins (Section II.), then will present Jenkins more in deep in Section III. We will talk about Coninuous-Integration benefits of Jenkins in Section IV. Section V. presents a study case we made on Jenkins fo build *this paper*. Section VI. introduces Acceptance Test Harness (ATH).

II. Related work

Since Jenkins introduction in 2004, some studies have tried to capitalize on its many possible applications.

Ahmed *et al.* implemented a unit test automation with Jenkins for a Robot Framework application [2]. We can quote Seth and Khare for their Jenkins application for Automated Coninuous Integration (ACI) in embedded software development [3]. Other Jenkins applications have been made by Arcuri, Campos and Fraser for Unit Test Generation (UTG) during software development, using EvoSuite plugins for Maven, IntelliJ and Jenkins [4]. Budiardja, Bouvet and Arnold exploited Jenkins properties to run application-level regression tests on High-Performance Computing systems (HPC) [5]

In the field of testing Jenkins itself, Munir and Runeson study on Acceptance Test Harness (ATH) shows contributions and weakness of ATH to Jenkins environment [6].

III. Jenkins

Jenkins is an open-source automation server [1]. It was first introduced in 2004 as Hudson, a solution to build repository Java code. In 2011 the project forked and was renamed Jenkins.

Jenkins supports automation of all development processes. Jenkins gives developers feedback on which the last commit is going to work. It suites for Continuous Integration and Continuous Development.

Jenkins is a server software, installed on a network computer and provides a web interface. The Jenkins instance administrator can install plugins to provide specific services. Then developers can create jobs, build process, pipelines...

A Jenkins plugin is a software extension for Jenkins providing support for specific task. The Jenkins community provides more than 1,500 plugins to automate any build task and is highly configurable [7]. With all available plugins, Jenkins can build projects, perform static analysis, deploy products and so on...Plugins are sorted in 5 categories: Platforms, User Interface, Administration, Source Code Management and Build Management.

A Platform plugin will provide a testing environment for your project. For example, the Android Emulator plugin [8] provides test suites for Android projects. A Source Code Management plugin can give Jenkins the ability to fetch GitHub, GitLab or Bitbucket repository code. Finally, a Build Management plugin will provide tools for build result analysis (such as the JUnit plugin [9]).

Because of automation process, Jenkins allows its users to reduce time spent on build and test tasks and allows automated regression testing. Add the Open-Source Software side of Jenkins (one can use Jenkins for free), the return on investment can be really beneficial [10].

IV. Continuous Integration

Continuous Integration workflows are software development practices where several redundant tasks are executed several times. In CI workflows, developers produce source code in a continuous manner by pushing new changes to the source repository (e.g., Git, Mercurial or SVN). Then product has to be tested in several level (Unit testing, Integration testing) and quality (Regression testing, Acceptance testing) before a build step, creating the final product.

All this steps are redundant and the non-manual tasks can be automatized.

For instance, Jenkins jobs can be triggered from different sources : source repository event, time based, on demand. Then, once an hour, a day or a week, the larger test suites and build process can be scheduled to provide building status monitoring, through the Jenkins notification ability.

Although Jenkins can be used for build processes automation, it can be used for Unit Test Automation [2]. As example, a commit on the source repository can trigger Jenkins test jobs to run the different testing levels with different test suites, in order to provide a quick report in case of defect introduction.

Through its several available plugins, Jenkins can run a lot of test unit suites for numerous test environment. One of Jenkins power is its ability to run tests for different testing environment, as different platforms (Windows, Linux, MacOS, Android, iOS…). Another good side of automation is the plugins ability to add Unit Test Generation (UTG) for newly added code, that can save a lot of time for test teams [4].

V. Study Case

As study case, we instanciated a Jenkins server to build *this paper*. The case is Jenkins usage for a build process, ie LaTeXfile generation from GitHub hosted code.

A. Jenkins installation and configuration

We first installed a Jenkins server from a Docker container on a private server, following steps from other papers [2]. Then we made this Jenkins instance public through Internet at http://jenkins.fivefactorial.se/ and created user accounts.

B. GitHub authentication

We created a private GitHub repository to host the `paper.tex` source file. We used our own credentials to allow Jenkins to access this repository (see picture 1). We were able there to run a job that was just checking out repository code, confirming that authentication was working correctly (see picture 2).

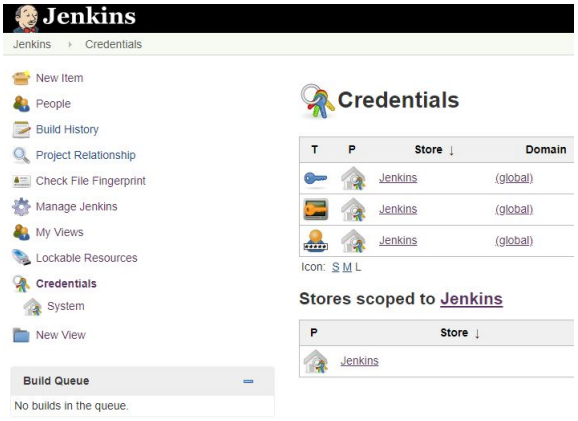

Fig. 1. Jenkins credentials page

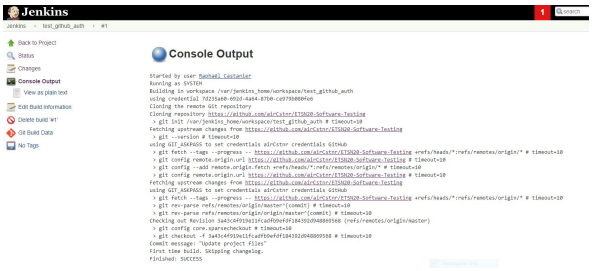

Fig. 2. GitHub authentication job output

C. Job creation

We created a new job to build the LaTeXfile into `PDF` format output. It builds each time a new commit appears on `master` branch and produces output file using `pdflatex` Linux command. See picture 3 for the job view.

We can see that Jenkins shows the last build workspace, the lase successful artifacts and the recent changes from VCS.

Now, each time a new commit appears on the remote repository, Jenkins tries to build the paper.

D. Unit Test

In this particular use case, the only relevant unit test is the fact that builds succeeds (no Error produced) or fails (one or more Errors appear, the job is canceled).

We could imagine another crietrion on number of LaTeXWarnings, thier severity or thier type but we didn't implemented it.

For example, some Errors like `Overfull hbox` and `Underfull hbox` warnings are indicating a potential risk of unsuccessful rendering, but they may not indicate a build failure.

E. Monitoring

We can follow build history through Jenkins view for the project (see picture 3) A blue colored ball means the build is successful and a red ball means the build failed.

There is also a weather icon the each job, indicating the success rate for a job (see picture 4). A sun means most of the previous builds succeeded.

Fig. 3. Project PDF job view

Fig. 4. Jenkins dashboard weather

VI. Acceptance Test Harness

We can briefly talk about the Jenkins Acceptance Test Harness (ATH), based on another study [6] and our sources.

Jenkins is an Open-Source software, maintened by a wide community of industrial and academic actors namely. The main development of Jenkins and its plugins is made by community members, but there is no test plan or test strategy for Jenkins Acceptance.

Although Jenkins is used to automate tasks in CI and Unit Testing, it seems to lack of testing of its own code. Manual tests are performed by developers but the test quality depends on tester thouroughness and are not necessarely relevant for Jenkins quality.

Furthermore, due to a large number of plugins, testing environments and configurations, it became tedious to test Jenkins and its plugins manually, using regression testing for example.

The ATH project [11] is born to automate End-to-end testing for Jenkins and its plugins in various conditions. According to the quoted study, it is a good start to improve Jenkins environment testing but its not an example for the field of software testing.

This ATH provides a testing framework to automate regressions tests on Jenkins and some plugins in various configurations.

VII. Conclusion

In this paper, we presented Jenkins, the automation server. Jenkins is a software running on server and executing various automated tasks on software. Due to its flexibility, Jenkins is able to adapt to many projects, in CI/CD proccess, for build, unit test and monitoring software projects. Jenkins is widely supported by an open-source community and is improved by plugins usage. Plugins are really numerous and fit to various project case automation.

We presented a study case on Jenkins we used to understand and practice its configuration. We were able to checkout a GitHub repository, build a LaTeXfile and monitor this job evolution.

We finally studied the ATH project, a test improvement initiative for Jenkins core and its plugins.

References

[1] "What is jenkins? the ci server explained," https://www.infoworld.com/article/3239666/what-is-jenkins-the-ci-server-explained.html.
[2] A. Ahmed *et al.*, "Unit test automation with jenkins-ci tool," 2015.
[3] N. Seth and R. Khare, "Aci (automated continuous integration) using jenkins: Key for successful embedded software development," in *2015 2nd International Conference on Recent Advances in Engineering & Computational Sciences (RAECS)*. IEEE, 2015, pp. 1–6.
[4] A. Arcuri, J. Campos, and G. Fraser, "Unit test generation during software development: Evosuite plugins for maven, intellij and jenkins," in *2016 IEEE International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2016, pp. 401–408.
[5] R. Budiardja, T. Bouvet, and G. Arnold, "Application-level regression testing framework using jenkins," *Concurrency and Computation: Practice and Experience*, vol. 30, no. 1, p. e4339, 2018, e4339 cpe.4339. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4339
[6] H. Munir and P. Runeson, "Software testing in open innovation: An exploratory case study of the acceptance test harness for jenkins," in *Proceedings of the 2015 International Conference on Software and System Process*, ser. ICSSP 2015. New York, NY, USA: ACM, 2015, pp. 187–191. [Online]. Available: http://doi.acm.org/10.1145/2785592.2795365
[7] "Jenkins plugins index," https://plugins.jenkins.io/.
[8] "Jenkins plugins andoid emulator," https://plugins.jenkins.io/android-emulator.
[9] "Jenkins plugins junit," https://plugins.jenkins.io/junit.
[10] "Jenkins - save money, time, and sanity." https://www.trustradius.com/reviews/jenkins-2016-08-25-11-21-32.
[11] "Jenkins github repository," https://github.com/jenkinsci/jenkins-test-harness.