A STUDY ON EFFECTIVENESS IN USING CONTINUOUS INTEGRATION AND DEPLOYMENT FOR SOFTWARE DEVELOPMENT: A STATE OF THE ART REVIEW

ADIS NABAWI AZIZAN [1]*, ASADULLAH SHAH [2],

[1,2]Kulliyyah of Information and Communication Technology, International Islamic University, Kuala Lumpur, Malaysia

**Corresponding author*: adisazizan@gmail.com

(Received: 3rd Jan 2019; Accepted: 26th Oct 2020; Published on-line: 30th Nov 2020)

ABSTRACT: Objective - The purpose of this paper is to help in providing insight on the effectiveness in using continuous integration in software development. The aim of the study is achieved using the help of existing journal and article, theories and literature review of effectiveness in implementing the continuous integration. Design Approach - A total of 17 journals and articles including literature review have been analysed using state of the art literature review methodology. Search keywords that have been used are continuous integration, continuous deployment, software development + Jenkins. The main databases that we use are mostly IEEE (ieee.org) and articles of webpages. We focus on software development and include all the related based on software development. Currently the databases are based on the access provided by the university so we excluded all the paid databases that university does not have access to. Findings – Due to limited resources and cost, the review is limited to software development but provides possibilities for future research in different sectors such as industries that required automation in their work. We also found out that continuous delivery became popular in the current year as there is a rise of software and applications due to the booming mobile technology. Problem Statement – Even with the growth of technology, some organisations still depend on the traditional method. This is because of the lack of expertise and no budget. Continuous integration already exists in over a decade, but the implementation is not widely introduced.

KEY WORDS: *Continuous integration, Continuous deployment, Software Development, Jenkins, Github*

1. INTRODUCTION

This paper is focused on literature review to study on the effectiveness of continuous integration and deployment. There has been systematic review by Shahin, Ali Babar and Zhu (2017), but it focused on the tools and challenges instead of the benefit and effectiveness in using Continuous Integration (CI) and Continuous Deployment (CD). In our review, we focus on the effectiveness and the benefit of using CI and CD in work environment especially in software development. Continuous integration is a practice in software development where developers work together and integrate their work, at least once a day. This is done by using an automated tool (Eddy et al., 2017). Continuous integration helps to have a shorter time and life cycle including increase of software quality and productivity of the team (Shahin et al., 2017).

Continuous integration is used widely for computer and mobile development work. Developers often work hard to put into action testing into their projects and connect the different kinds of tools into an integrated testing process (Penson et al., 2017). To keep on building high quality codes and to lower the time and work wanted for testing is to focus on Continuous integration (Eddy et al., 2017).

To make sure CI is effective, it must allow for a seamless back and forth between development, testing, and deployment but it is riddled with people factor. As an example, working in large increments might not only lead to more meaningful change sets, but might also complicate synchronization between team members and, if necessary, revert changes (Davis, Serebrenik, & Filkov, 2000).

Dependencies between developer and code components can be increased when the size of the team and the project is large and be a trigger to the build failure. Identifying any relationship between the failures of build and the team's size including the project will be helpful for developers to identify the issues and able to prevent build failures when size of the team increases (Islam & Zibran, 2017). It is important to find the problem in the development process and subsequently increase development efficiency by understanding the build field and the factor that impact the outcomes (Rausch, Hummer, Leitner, & Schulte, 2017).

Continuous integration has become popular in current times. This evidence can be found from published works in the reference. Examples of the practice of continuous integration expected to have some benefits are the rapid and lots of feedback from customers and development team. Secondly, reliability and frequent releases improve customer satisfaction and quality of products. Lastly, manual implementation can be eliminated and relationship between teams can be strengthened (Shahin et al., 2017). There are many repositories developed by developers, and frameworks and plugins for web development. All those frameworks and plugins have many useful modules and functionalities that help to speed up the development process. These points are constantly increasing the complexity of application development. This is a reason why developers must rely on more than one framework or a plugin within an application which of course often leads to conflicts or even defects.

2. CONTINUOUS INTEGRATION: AN OVERVIEW

Continuous integration also includes continuous deployment and continuous delivery. In this literature review, we will explain the three types of continuous integration. Since its realization in 1991 and distribution spread as part of Microsoft's and Extreme Programming's development practices, Continuous Integration (CI) has become a global Software Engineering phenomenon (Beller, Gousios, & Zaidman, 2017). It is a set of standard that is relevant to the daily movement of work of development team (Meyer, 2014).

Continuous integration and automated testing are valuable in finding regression errors and ensuring the build is always green. Automated testing helps tests to be conducted more frequently and expedites the testing process (Penson et al., 2017). There are also some studies that focused on continuous delivery in industry use and talked about continuous delivery being implemented successfully with the benefits of continuous delivery. They also have quick feedback from users, bugs are found faster

due to quicker release cycles, and features do not need to wait for lengthy periods of time to be implemented (Eddy et al., 2017).

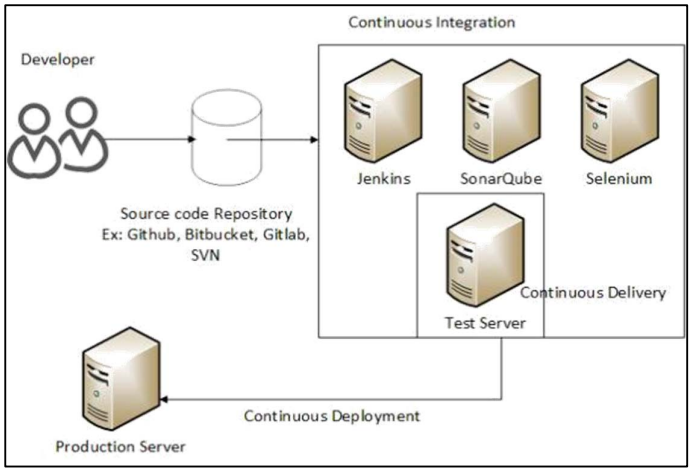

Fig. 1. The overview of continuous integration (Matthies et al., 2017; Beller et al., 2017)

CONTINUOUS INTEGRATION

Continuous Integration (CI) is a practice where the incremental changes to a software project are automatically built, tested for regression, and checked for common quality problems as they push to the project repository such as Github, Gitlab, Bitbucket or SVN (Gupta, Khan, Gallaba, & McIntosh, 2017). One of the cloud-based providers that stands out as provider of CI services is TRAVIS CI. The impact to the developer attraction and retention is less known in the adoption of the CI (Gupta et al., 2017). Thus, this study should be conducted to find the data as what has been questioned by many.

CONTINUOUS DEVELOPMENT

Continuous Deployment (CD) automatically and continuously deploys the application to live production environments, and what differentiates continuous deployment from continuous delivery is the state of production environment which is for the actual customers (Shahin et al., 2017).This can be done using a tool called Jenkins to automatically run a script and deploy to the different server at the same time.

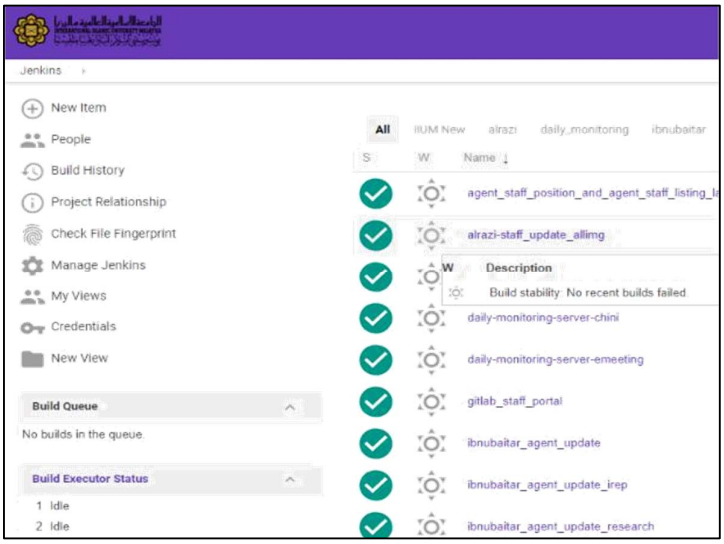

Fig. 2. Jenkin dashboard by International Islamic University Malaysia

The above figure shows the Jenkin dashboard where the look and feel have been customised. The green icon means that it has success build.

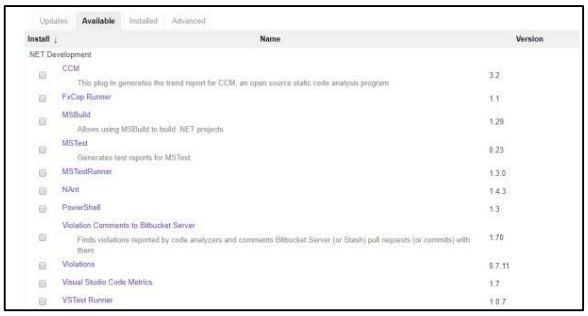

Fig.3. Plugins in Jenkins that are available for download and use

The above figure shows plugin that can be used and installed without the need of developer to create their own plugin. This saves time when you want to setup your own Jenkins server.

CONTINUOUS DELIVERY

Continuous delivery is a process that implements the analysis and testing steps of continuous integration, it also improves the delivery with notification mechanism to the staging environment for quality assurance (QA) testers (Eddy et al., 2017). Web application will run after the build process is finished, but it might not be the right thing. In a good practice, there might be errors so performing automated test is the best way to detect and inform the errors quickly. This process is named “Test after Build” and it will be executed after the previous build process is done without any errors (Tim &

Lichter, 2017). The tester will write the script, so the code can be tested right away once the code is pushed to the repository, Github for example.

The main requirements for a continuous delivery pipeline are as follows:

i. Automated testing
ii. An archiving system for versioning and performing roll backs when necessary.
iii. A deployment mechanism when build is passed to a staging environment before it is approved to be shipped off to production.
iv. A notification mechanism to inform the developers about the build.
v. A feedback mechanism for QA testers to discuss changes that may need to be implemented before push to production.

The important requirement for a CI is VM configuration where machine can have different parameters: as type, product type, product version, network configuration and it is very hard to support a specific software product (Zaslavskiy et al., 2017). Thus, testing is required in the continuous delivery.

3. CHALLENGES IN ADAPTING CONTINUOUS INTEGRATION

There was also a case study on the Effectiveness of Test-Driven Development and Continuous Integration where a Dutch small-to-medium enterprise (SME) implemented test-driven development and continuous integration and discovered many defects compared to existing methodology; this also improved the focus on quality and applications test (Amrit & Meijberg, 2018). But there are still challenges when adopting the continuous integration in the teams that are used to the existing methodology.

In their papers, Bougouffa, Diehm, Schwarz, and Vogel-Heuser (2017), and Mårtensson, Ståhl, and Bosch (2017) stated that the use of CI depends on the developer's thinking. From the paper (Bougouffa et al., 2017), the results of a survey of 61 industrial companies (machine and plant manufacturing) showed that the percentages of CI usage are as follows: 52% of the companies do not use Continuous Integration, 33% use Continuous Integration only to a certain extent, and 15% use Continuous Integration by default and it is embedded in their current methodology.

There were also several procurements of server etc to make sure that the CI environment could be setup (Tim & Lichter, 2017). This also required the cloud to be setup, so we can test the viability on the cloud-based system with the CI server. Tim and Lichter (2017) used Intern as their base for JavaScript framework and Selenium as the web testing environment for their testing to make sure all tests were executed.

4. RESEARCH PROBLEM

Even with the growth of technology some organisations still depend on the traditional method. This is because of the lack of expertise and no budget. Continuous integration already exists in over a decade, but the implementation is not widely introduced.

Some developers also find that the setup of continuous integration and continuous deployment is a hassle as they need to setup everything before they can begin coding. This also why the use of the continuous is so unpopular among the developers who are already used to the traditional method.

To measure the effectiveness of continuous integration, Amrit and Meijberg (2018) considered three things:

i. Reduction of defect.
ii. Throughput and the defect lead.
iii. Development productivity.

Reduction of defect is whether CI helped in reducing the number of defects which include the pre-release defect level and post-release defect. Throughput and the defect lead check on whether CI helped in reducing the time to find and fix the defects. Development productivity is the productivity in closing the defect.

Thus, the problem has been to identify the effectiveness and the benefit of the reduction of defect, throughput and the defect lead and also the development productivity.

By having an appropriate tool in place does not determine the project, continuous integration is about fostering responsibility and providing customer value (Meyer, 2014). Meyer (2014) also mentioned that when latest changes are continuously merged into the mainline branch, there is a deeper responsibility for everyone on the team to make sure that those changes not only pass the build but that they also have minimal impact on the production environment. Then, another research problem is, the effectiveness of the continuous integration and deployment must be minimal impact and pass the build with ease and clarity to show the effectiveness of continuous integration and deployment.

5. METHODOLOGY FOR RESERCH

There are 3 methodologies that can be adopted in the research namely action research, design-based research and observation study. These 3 methodologies will help in finding the effectiveness of continuous integration and deployment.

ACTION RESEARCH

This action research can be done in industries that never use continuous integration and deployment in their developments. This action research is where we implement the continuous integration and deployment in our current projects and set aside the differences before and after the use this method. From the article of Fowler (2006), he said that people find out as they try it and that it makes a huge difference to development. Some developers are still not using the continuous integration as part of their work and we can differentiate it with action research by comparing the before and after the implementation of continuous integration. There should be interviews with the developers on whether there are not comfortable in changing the way that they had done before.

DESIGN – BASED RESEARCH

Some people are still biased on using the continuous integration. So, our next venture is to evaluate the performance of people that use the continuous integration with the people that do not use. In Rahman and Roy (2017), there was a paper that measure the impact on the code review by exploring two distinct aspects of continuous integration which are automated build status and automated build frequency, and investigating how the former aspects might affect the later. Thus, the interviews from the developers are important.

OBSERVATION STUDY

For observation study, we can detect how much time needed to finish developing simple application and deploy the application including testing. We then can compare between two groups; one that implements continuous integration with the one that does not implement. Beller et al. (2017) used TRAVISTORRENT to investigate the CI testing and they came up with the following questions:

i. Do the Continuous Integration lead to the better product which less bugs etc.?
ii. Do the Continuous Integration lead to the less of regression test.
iii. What are Continuous Integration best practices that successful projects employ (build more often, fail more often, have more tests in the CI)?
iv. Do Continuous Integration-enabled projects switch to Continuous Delivery?
v. How long may a Continuous Integration build run to still be considered helpful?
vi. Does a broken build negatively affect other developers' productivity, as is often claimed?
vii. Two development models compete over how to use Continuous Integration? Should developers keep on successful build or fix the build often?
viii. Do a failed build lead to less outside contributions?
ix. Do multiple integration environments lead to fewer defects?

All these questions were already answered in their paper. Their research brought insight on the CI capability in the modern age and the use of TravisTorrent which was in the prototype phase at that time. TravisTorrent aborts or skips the build once it detects some issues in their log analysis. Once the build is skipped, developer can check through the log where the issues come from. This enables quick fixing and less bug on the way to delivery.

Fig.4. Travis Torrent logo. The alternative of Jenkins

6. CONCLUSION

Today's world requires effective and fast delivery process for software development. Due to the disrupt of technology for example smartphone, the application system became popular to download and used by everyday consumer. It is not tied to the use of agencies or department anymore and user wants it fast and efficient as it will help them in everyday life.

The use of code versioning like Github or Gitlab and deployment tool like Jenkins will help developers to have easier integration during the software development phase. These tools will help developers and beat the manual ways of deployment and integration thus the product output will be better in quality.

Fig.5. Tool's to be use in continuous deployment

Developers need to deploy their application to the production regularly and to test them with different environment at the same time. This requires a technology to help developer and tester integrate their project and deliver them to the user instantly. Developers can use Github, Bitbucket or Gitlab to manage their codes and Jenkins as a CI to integrate with others for example SonarQube (Kwon & Ko, 2017), Selenium, HP ALM. The SonarQube for example helps to check the quality of the codes that have been written (Kwon & Ko, 2017). Then this should be automatically deployed to several production servers at the same time once the test is successful.

A study to find the effectiveness of using Continuous Integration and its components is important as the fast pace of software development is the trend nowadays. Continuous Integration approach could also work with modern Client-side Web Applications. There is also research that provides a characterization of open source software projects that follow continuous integration practices. The research characterized those software projects based on their activity, popularity, size, testing, and stability.

This topic has become popular lately and a lot of research papers have been written in 2016 and 2017. All these papers complimented the use of continuous integration and the analysis of it.

REFERENCES

Amrit, C., & Meijberg, Y. (2018). Effectiveness of Test-Driven Development and Continuous Integration: A Case Study. IT Professional, IT Prof., 20(1), 27–35. https://doi.org/10.1109/MITP.2018.014121554

Beller, M., Gousios, G., & Zaidman, A. (2017). TravisTorrent: Synthesizing Travis CI and GitHub for Full-Stack Research on Continuous Integration. IEEE International Working Conference on Mining Software Repositories, 447–450. https://doi.org/10.1109/MSR.2017.24

Bougouffa, S., Diehm, S., Schwarz, M., & Vogel-Heuser, B. (2017). Scalable cloud based semantic code analysis to support continuous integration of industrial PLC code. Proceedings - 2017 IEEE 15th International Conference on Industrial Informatics, INDIN 2017, 621–627. https://doi.org/10.1109/INDIN.2017.8104843

Davis, U. C., Serebrenik, A., & Filkov, V. (2000). The Impact of Continuous Integration on Other Software Development Practices : A Large-Scale Empirical Study. (CI), 60–71.

Eddy, B. P., Wilde, N., Cooper, N. A., Mishra, B., Gamboa, V. S., Shah, K. M., … Shields, N. A. (2017). A Pilot Study on Introducing Continuous Integration and Delivery into Undergraduate Software Engineering Courses. 2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T), 47–56. https://doi.org/10.1109/CSEET.2017.18

Fowler, M. (2006). Continuous Integration.

Gupta, Y., Khan, Y., Gallaba, K., & McIntosh, S. (2017). The impact of the adoption of continuous integration on developer attraction and retention. IEEE International Working Conference on Mining Software Repositories, 491–494. https://doi.org/10.1109/MSR.2017.37

Islam, M. R., & Zibran, M. F. (2017). Insights into Continuous Integration Build Failures. IEEE International Working Conference on Mining Software Repositories, 467–470. https://doi.org/10.1109/MSR.2017.30

Kwon, J.-H., & Ko, I.-Y. (2017). Cost-Effective Regression Testing Using Bloom Filters in Continuous Integration Development Environments. 2017 24th Asia-Pacific Software Engineering Conference (APSEC), 160–168. https://doi.org/10.1109/APSEC.2017.22

Mårtensson, T., Ståhl, D., & Bosch, J. (2017). Continuous Integration Impediments in Large-Scale Industry Projects. Proceedings - 2017 IEEE International Conference on Software

Architecture, ICSA 2017, 169–178. https://doi.org/10.1109/ICSA.2017.11 Meyer, M. (2014). Continuous integration and its tools. IEEE Software, 31(3), 14–16.

https://doi.org/10.1109/MS.2014.58

Penson, W., Huang, E., Klamut, D., Wardle, E., Douglas, G., Fazackerley, S., & Lawrence, R. (2017). Continuous integration platform for Arduino embedded software. Canadian Conference on Electrical and Computer Engineering. https://doi.org/10.1109/CCECE.2017.7946696

Rahman, M. M., & Roy, C. K. (2017). Impact of continuous integration on code reviews. IEEE International Working Conference on Mining Software Repositories, 499–502. https://doi.org/10.1109/MSR.2017.39

Rausch, T., Hummer, W., Leitner, P., & Schulte, S. (2017). An Empirical Analysis of Build Failures in the Continuous Integration Workflows of Java-Based Open-Source Software. IEEE International Working Conference on Mining Software Repositories, 345–355. https://doi.org/10.1109/MSR.2017.54

Shahin, M., Ali Babar, M., & Zhu, L. (2017). Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices. IEEE Access, 5(Ci), 3909–3943. https://doi.org/10.1109/ACCESS.2017.2685629

Tim, R., & Lichter, H. (2017). Continuous Integration Processes for Modern Client-Side Web Applications. (March), 8–10. https://doi.org/10.1109/IEECON.2017.8075805

Zaslavskiy, M., Kaluzhniy, A., Berlenko, T., Kinyaev, I., Krinkin, K., & Turenko, T. (2017). Full automated continuous integration and testing infrastructure for Maxscale and MariaDB. Conference of Open Innovation Association, FRUCT, 273–278. https://doi.org/10.23919/FRUCT.2016.7892211