Continuous Integration Processes for Modern Client-Side Web Applications

Ratha Tim
Sansiri Tanachutiwat

The Sirindhorn International Thai-German Graduate School of Engineering
King Mongkut's University of Technology North Bangkok
Thailand
Email: tim.r-sse2013@tggs-bangkok.org
sansiri.t@tggs.kmutnb.ac.th

Marko Vukadinovic
Heinz-Josef Schlebusch

KISTERS AG
Germany
Email: marko.vukadinovic@kisters.de
schlebusch@kisters.de

Horst Lichter

RWTH Aachen University
Germany
Email: lichter@swc.rwth-aachen.de

***Abstract*— Continuous Integration (CI) is very useful for applications that involve many files and multiple developers. Unfortunately, not all types of applications can easily apply this approach. Apparently, CI does not gain a lot of attention with Modern Client-Side Web Application (MCSWA) because it requires complicated testing, i.e. the running environments are browsers. There is no compiler or error warning when a developer writes bad code and the build behavior in the usual CI practice is different from the build process in MCSWA. If the integration process is done manually by an integration expert, unexpected errors are found not only while integrating but also when performing manual tests to verify it. Moreover, it is problematic and elaborate if a developer needs to test the features by clicking around with repeated user-interaction in different browsers; especially, he might create human errors or miss some steps. This indicates that manual processes consume a lot of time and they are stressful. Therefore, the intent of this paper is to demonstrate a new approach for a CI process which specifically applies to MCSWA. It also provides a precise cycle for web development teams on how these repeated processes are important to run automatically with effective expected outcomes.**

***Keywords*— client-side; web application; build; continuous integration;**

I. INTRODUCTION

In the area of modern client-side web development, there are lots of communities and organizations which have offered JavaScript frameworks and plugins on the internet. Those frameworks and plugins have many useful modules and functionalities that help to speed up the development process, yet user' requirements still keep changing over time (e.g. better interactivity, better performance or better security). These points are constantly increasing the complexity of application development. This is a reason why developers have to rely on more than one framework or a plugin within an application which of course often leads to conflicts or even defects. Usually, defects are caught during the integration process and it is very hard to fix them quickly because web applications need to be tested on many different environments including multiple operating systems and even more browsers.

The issues described above are demanding a proper solution, because unexpected and unforeseeable problems might affect project deadlines. This might result in unsolvable bugs for more complex applications. To improve this situation and enhance development quality and efficiency, Continuous Integration (CI) [1]–[5] is a very promising approach to apply.

Hence, to reduce the number of defects during the integration process, some managing processes are required to work automatically. This paper will illustrate a solution by using our proposed CI process that can be perfectly adapted with web development environments. Moreover, we will also introduce a way to allow tests be run on a Cloud Services in order to minimize time and cost compared to an expensive local installations of many different test systems.

II. LITERATURE REVIEW

Continuous Integration (CI) is a software practice that requires software to be integrated in a shared repository up to several times a day. Each integration is verified by an automated build, which includes testing to detect integration errors as quickly as possible [6], [7]. The first step in a CI Process is developers committing their code to a Version Control Systems (VCS). A CI server detects changes in the repository (e.g. by polling every few minutes). When a change is detected the project is pulled into the CI server itself. This will trigger a build process which builds the software. Then the built-result report will be generated by CI server and sent to project manager (or members). The CI server continues to check for further changes and repeat this cycle [7], [8].

Since Martin Fowler wrote an article about CI practices in 2006 [9], there are lots of organizations that have already applied this approach to their development and build pipeline.

2017 5th International Electrical Engineering Congress,
Pattaya, Thailand, 8-10 Marchpp, DOI: 10.1109/IEECON.2017.8075805

For example, the popular tools for CI are Jenkins, Hudson, Travis CI or TeamCity. Common build tools like Ant, Rake, MSBuild, or Maven are often used to create a build process [10]–[13]. However, these tools are mostly used by java, ruby, or .net development teams. Currently, there is still little information about CI with modern client-side web applications. This is mainly due to its system testing environment complexity, cost and time consumption. Especially, it is even more troublesome if the interface might usually change which causes unit and functional tests change. That is why in practical work, there are still many web development teams that decide to integrate their applications manually.

Our goal is to convert from manual integration to automated integration. Yet, common CI is just a basic concept which still needs to be customized in order to smoothly adapt with modern client-side web application's testing systems. Hence, we will illustrate our CI approach in the next chapter.

III. Overview And Detail CI Process

A. Overview CI Process

Figure 1 presents the system overview and the components of our proposed CI process that are involved in the whole CI development. The main actors are: developers, a Version Control System (VCS), an on demand cloud testing service, and a testing server. The process is consistent with the common CI approach, yet there are some specific tasks that might work differently. For example, in the CI server program contains four mandatory processes which need to be run sequentially. The four processes are: "Test before build", "Build", "Test after Build", and "Deploy". And, the "Test after Build" is selected to perform all tests on a Cloud Service.

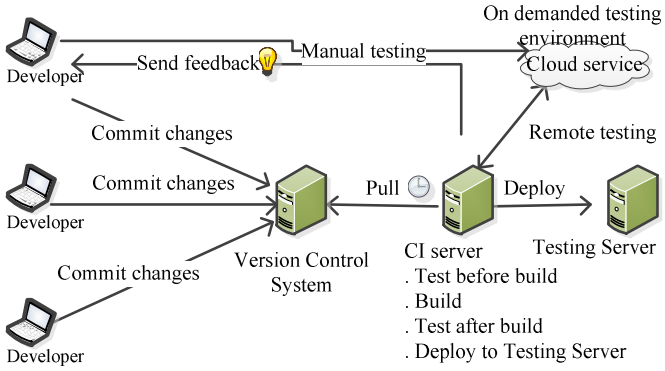

Fig. 1. A CI components in modern client-side web application

B. Detail CI Process

1) Development and Manual Testing

Local development is an important part in the CI approach. Every new commit from a local development is a key signal which triggers the CI server to execute its automation processes (i.e. test, build, and etc.). A well-defined and executed local development helps CI approach to work effectively. The developers need to be aware of the basic practical flow during their development.

This process starts with developers checking out the project from VCS, so that they can work on the assigned tasks. The specific tasks of developers consist of maintaining source code and tests (unit test and functional test). Unit tests [14] are simply used to verify the behavior of single elements (or units) in a software system. Those are, most of the time, single methods (functions) or classes. It is a type of test which works by executing a piece of code directly and expecting a specific result. On the other hand, Functional tests [7] work by issuing the commands to a device/browser that mimic actual user interaction. Functional tests do not always call other APIs directly; hence mockup data needs to be provided to test such behavior.

Every day, developers also perform manual tests on their local machine in order to make sure that all tests are passed and the source code work correctly. Then at the end of this cycle developers commit their source code and tests to the VCS.

2) Automated Process on a CI Server

Setting up CI server can be done by manual or using its tool like CruiseControl [15] or CruiseControl.NET [16]. Due to the fact that web application's behavior works different from other applications written by java, ruby or .net, we decided to set up CI server by ourselves. Our CI server is responsible for the four mandatory automated processes. Hence, this server will automatically check new commit from VCS according to our pre-defined period of time, and once a change is detected, the processes will be executed as consequence.

a) Test before Build

The aim of "Test before Build" is to catch the errors as soon as a new commit is made. Fortunately, with modern web applications we do not need to compile our code before running it. It means browsers act as the platform to interpret JavaScript code. So, if any test fails after running in the browsers, the project is fragged, indicating to the developers that it contains errors.

We know that the CI server is running on a predefined time-based trigger. When the trigger goes off, the CI program checks-out the project from the VCS. After that, the program CI determines if there is a new commit? If so, the CI server will automatically run "Test before Build" script, otherwise it will do nothing in this cycle. The "Test before Build" script will invoke browsers (local browsers which are installed in CI server) and perform all tests. When it finishes, it will summarize the result. Then, the CI program will determine if the result contains errors. If so, the program will send an email to notify the project manager (or a team member) with a log file. Otherwise, the CI program ends this process and the CI server will wait for the next trigger event. The Figure 2 shows the process of the CI program.

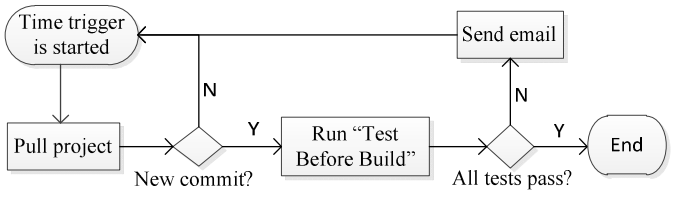

Fig. 2. CI program on "Test before Build" process

2017 5th International Electrical Engineering Congress,
Pattaya, Thailand, 8-10 Marchpp, DOI: 10.1109/IEECON.2017.8075805

b) Build

Usually, the word "build" in software development landscape is used for compiling from source code to machine code. However in web application development, it means source code minification and source code obfuscation. Minification is a process of combining multiple project files into few combined files then removing all unnecessary characters or dead-code without changing its functionality. Notwithstanding, the obfuscation process is used to hide the actual code intention. So that, the process will rename variable and function names to meaningless names, and remove all extra white space and line breaks.

In our approach, the build process is performed automatically on the CI server, after a successful "Test before Build". Then after the build process finishes, the CI program finds the error in log file. If there is any error, the program will send an email to the project manager with log file. Otherwise, the CI program ends the build process and the CI server will wait for the next trigger event.

c) Test after Build

Normally, a web application might run after the build process is finished, but it does not mean it does all right thing. In a real practice, the errors might encounter. Therefore, performing automated test is the best way to detect and announce the errors quickly. And this process is named "Test after Build" and it will be executed after the previous build process is done without any errors.

In our approach, the "Test after Build" process is worked with external platform. It means all tests will be run on a cloud-based service. The cloud-based service allow us to perform the tests on a number of different environments, i.e. operating systems, browsers, system resources, and etc., which are not existed on our local. This is very helpful because performing tests on a local machine is very costly to include many environments.

Recently, using cloud services has become common in the software development. The testing process is controlled by the CI server. The CI server opens the connection to the cloud-based service. Then, it sends following commands to the cloud service: authentication, initialize the operating systems with the corresponding browsers that we want to run our tests. The cloud-based service listens to the commands and works accordingly. Once it finishes it will send the result back to the CI server, and then the result will be saved in a log file. The whole process can be easily illustrated in the Figure 3.

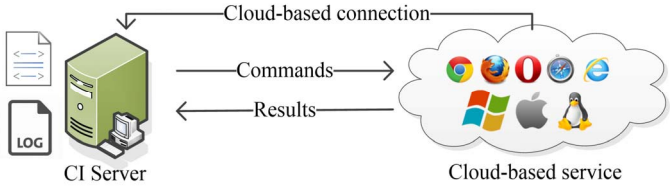

Fig. 3. Simple Cloud-based process

d) Deploy

Deploying a web application on a test server ensures that the application can be manually tested. Since the programmers deploy multiple times during the development process, it is preferred to automate this job. As a consequence, the deploying process will be started once the "Test after Build" process finishes without any defects.

The process is such a good approach for testers (QA, other developers, project manager, and etc.). They could perform manual testing on the latest version of the application as fast as the whole CI cycle is done. Especially, with modern client-side web applications, this can be easily done by copying all your project files to web directory of the test server. After the automated deployment process is finished, we successfully completed one CI cycle and the next cycle will start by the time trigger.

IV. Implementation And Outcome

We need various tools and necessary procedures to achieve this CI approach. These tools are required to adapt to our current development environment. Finally, we chose Intern [17] as the JavaScript testing framework, Selenium [18] as the web testing environment, SauceLabs [19] as the cloud-based service and Grunt as the JavaScript task runner.

A. Implementation

All sample tests are following Intern structure and its interfaces. To run these JavaScript tests we need a platform to execute it. With Intern, there are two ways of running the tests, i.e. via browser and command line. Selenium is a tool to automate web drivers which works according to Intern's commands. Intern and Selenium can let us achieve an automated testing in a local environment. Therefore, we use SauceLabs service for our cloud-based testing. We also need a task runner in order to invoke aforementioned tools. For this purpose we use Grunt task runner.

The whole implementation is divided into five sub tasks. Firstly, we developed a local testing environment which automates local browsers and runs on our CI server. This task, which is referred to the "Test before Build" process, is done by Intern and Selenium. Secondly, we developed a build script for the "Build" process by using Grunt. Thirdly, we defined an automated cloud testing process which is done by Intern and SauceLabs service for the "Test after Build" process (SourceLabs uses a Selenium grid [20] that can provide a large number of operating systems and browsers). Fourthly, we developed a deployment script for the "Deploy" process. Lastly, we created script to ensure the whole process will run successfully and perform error handling tasks.

Figure 4, shows system template of CI program which is set up on CI server. We developed four main grunt tasks: Task1, Task2, Task3, and Task4. Task1 is runTBB means "run test before build", Task2 is runBuild means "run build", Task3 is runTAB means "run test after build" and Task4 is runDeploy means "run deploy". The runTBB is connected to local brewers to perform local testing. While the runTAB is connected to SauceLabs to perform cloud testing. Finally, the runDeploy is connected to testing server.

2017 5th International Electrical Engineering Congress,
Pattaya, Thailand, 8-10 Marchpp, DOI: 10.1109/IEECON.2017.8075805

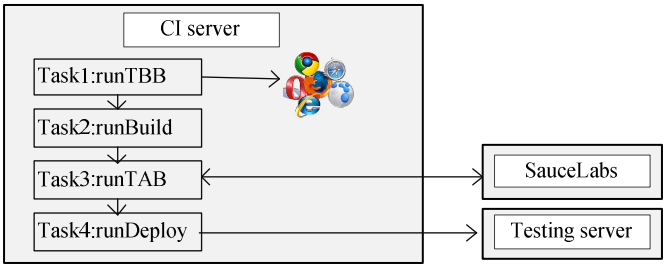

Fig. 4. System template of CI program

B. Outcomes

We faced some problems before achieving this system. The resulting approach will enhance our development quality in following aspects:

- Errors could be detected from the early stage because every new commit will be verified by the CI server.
- It improved working collaboration within the development team.
- Developers have more confidence in their code because they have more time to concentrate on their tasks. Moreover, writing tests is a process to improve source code quality.
- When software quality is improved this means Quality Assurance (QA) is improved accordingly.
- With this system we could replace a lot of manual tasks, including verbal communication between developers and QA.

V. Conclusion and Future Work

From our proposed conceptual processes to actual implementation, it is shown that applying a CI approach with MCSWA is not complicated. It can be done by following our processes step by step individually. Then combining each step and setting up the condition on how you want each process connect to each other. Therefore, CI could work efficiently when developers commit great code and tests. So, it can help the automation processes detect unpredictable and unforeseeable errors.

Finally, our experiment shows that the CI approach could also work with MCWA. However, according to our research timeline we did not manage to develop our system to work perfectly. Our CI is only capable to run a single project at a time. So, if we have multiple projects they must work in series. Moreover, we use a time trigger for scheduling our CI server to check for changes. This can be improved by triggering the process from the VCS when a change happens. Hence, setting the CI server to work with multiple projects at the same time and using "git hooks" to detect new commit automatically can be our future work.

Acknowledgement

I would like to pay special thankfulness to my professors: Prof. Dr. rer. nat. Horst Lichter and Dr. Sansiri Tanachutiwat for the support and assistance. Their encouragement made it possible to achieve the research's goal. Furthermore, I am thankful to Marko Vukadinovic for the technical support; without him I would have not solved my critical issues. Last but not least, I also would like to show my gratitude to Dr. Heinz-Josef Schlebusch who provided a good collaboration between my university and the company for this research.

References

[1] S. T. Lai and F. Y. Leu, "Applying Continuous Integration for Reducing Web Applications Development Risks," in 2015 10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA), 2015, pp. 386–391.

[2] F. A. Abdul and M. C. S. Fhang, "Implementing Continuous Integration towards rapid application development," in 2012 International Conference on Innovation Management and Technology Research, 2012, pp. 118–123.

[3] J. Lu, Z. Yang, and J. Qian, "Implementation of continuous integration and automated testing in software development of smart grid scheduling support system," in 2014 International Conference on Power System Technology (POWERCON), 2014, pp. 2441–2446.

[4] A. Bhattacharya, "Impact of Continuous Integration on Software Quality and Productivity," The Ohio State University, 2014.

[5] "Continuous Integration Essentials |," Continuous Integration Essentials | Codeship. [Online]. Available: https://codeship.com/continuous-integration-essentials.

[6] "Continuous Integration | ThoughtWorks." [Online]. Available: https://www.thoughtworks.com/continuous-integration.

[7] P. M. D. S. Matyas Andrew Glover, Continuous Integration: Improving Software Quality and Reducing Risk 1st Edition, 1st ed. Addison-Wesley Professional; 1 edition (July 9, 2007), 2007.

[8] "Benefits of Continuous Integration - CodeProject." [Online]. Available: http://www.codeproject.com/Tips/859451/Benefits-of-Continuous-Integration.

[9] "Continuous Integration," martinfowler.com. [Online]. Available: http://martinfowler.com/articles/continuousIntegration.html.

[10] M. Meyer, "Continuous Integration and Its Tools," IEEE Softw., vol. 31, no. 3, pp. 14–16, May 2014.

[11] M. D. Paul, "Continuous Integration," presented at the Continuous IntegrationImproving Software Quality with Continuous Integration, 2007.

[12] Q. Gilles, "Beginners guide to continuous integration," presented at the PROGRESS EXCHANGE 2013.

[13] 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), 2015, pp. 1–6.

[14] Hazem Saleh, JavaScript Unit Testing. Packt Publishing; 1st New edition edition (January 14, 2013), 2013.

[15] "CruiseControl Home." [Online]. Available: http://cruisecontrol.sourceforge.net/.

[16] "CruiseControl.NET - Overview - CruiseControl.NET." [Online]. Available: http://www.cruisecontrolnet.org/projects/ccnet.

[17] "Intern: The user guide: What is Intern?" [Online]. Available: https://theintern.github.io/intern/#what-is-intern.

[18] "Selenium Documentation — Selenium Documentation." [Online]. Available: http://www.seleniumhq.org/docs/.

[19] "Cross Browser Testing, Selenium Testing, and Mobile Testing | Sauce Labs." [Online]. Available: https://saucelabs.com/.

[20] "Selenium Grid." [Online]. Available: http://www.seleniumhq.org/projects/grid/.