LAMK Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

Implementation of Continuous Integration and Continuous Delivery in Scrum

Case study: Food ‘N Stuff and WebRTC Applications

LAHTI UNIVERSITY OF APPLIED SCIENCES LTD
Degree programme in Business Information Technology
Bachelor’s Thesis
Autumn 2016
Phan Dinh Huy Trinh
Minh Hieu Doan

Lahti University of Applied Sciences
Degree Programme in Business Information Technology

TRINH, HUY:	Title: Implementing Continuous Integration and Continuous Delivery with Scrum
DOAN, HIEU	Case study: Food 'N Stuff and WebRTC Application
Bachelor's Thesis in Business	
Information Technology	72 pages, 3 pages of appendices
Autumn 2016	

ABSTRACT

Scrum, which is the most popular practice of Agile Methodology, has been increasingly growing in popularity over the past decade. The implementation of Scrum brings various positive benefits to software development teams and enable them to adapt quickly to changes in the requirements and business environment. However, there are always opportunities for further improvement.

The goal of this study is to identify the benefits adapting Continuous Integration and Continuous Delivery in Scrum by comparing the development processes of two artefacts, which are small-scale projects conducted at Lahti University of Applied Sciences. The first is a food management application and social platform managed only by applying Scrum. The second one is a real-time web communication application implemented with Scrum and the support of Continuous Integration and Continuous Delivery.

The results of the study revealed that the adaptation of Continuous Integration and Continuous Delivery offers three major improvements in terms of time, quality and portability. By implementing these into Scrum during the development process of the second application, the authors were able to save time, improve the application's quality throughout the development process and improve the portability of the application.

Keywords: Agile Software Development Methodologies, Scrum, Continuous Integration, Continuous Delivery, Docker.

CONTENTS

LIST OF ABBREVIATIONS

API	Application Programming Interface
ARN	Amazon Resource Name
AWS	Amazon Web Services
CD	Continuous Delivery
CI	Continuous Integration
CVCS	Centralized Version Control System
CVS	Concurrent Version System
DVCS	Distributed Version Control System
EC2	Elastic Compute Cloud
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IAM	Identity and Access Management
IP	Internet Protocol
Lahti UAS	Lahti University of Applied Sciences
LXC	Linux Containers
MVC	Model-View-Controller
NAT	Network Address Translation
SSH	Secure Shell
SSL	Secure Sockets Layer

SVN	Subversion
TFS	Team Foundation Server
URL	Uniform Resource Locator
VCS	Version Control System
WebRTC	Web Real-Time Communication

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Background

According to the State of Agile survey in 2015, Agile Software Development Methodologies have been growing in popularity and implementation. Out of the 3,880 survey respondents 95 percent stated that their organization has adopted agile methodologies in its business. In addition, the survey also pointed out that Agile Methodologies are being implemented especially in enterprises and large organizations. In 2006, when the survey was first carried out, two thirds of the responses stated that their organization has less than 100 employees. However, according to the 2015 survey, 31% of the respondents worked in organizations that have over 1,000 employees. Agile Methodologies include different forms and practices, which are Scrum, Lean and Kanban, Extreme Programming, Crystal, Feature-Driven Development (McLaughlin 2016). According to Versionone.com's survey, 70% of the respondents stated that the Agile practice which they have been using is Scrum or a hybrid version of Scrum. Clearly then, Scrum is the most popular of the Agile Software Development methods. (VERSIONONE.com 2015, 2.)

Scrum and other Agile Methodologies, in general, have several limitations. One of them is the low frequency of code integration of the project. The same is true with other traditional approaches: developers practice Agile methods separately for a long period of time without being aware of the increasing number of conflicts as time goes by. Therefore, after each merge and build of the application, developers often spend more time on debugging new bugs and conflicts, which costs unnecessary extra effort and time. (Continuous Integration in Agile Software Development 2016)

Another limitation of Scrum is the release of the software. The release phase of a modern software development cycle is still similar to traditional software development methods. The preparation and work for the release are postponed until the very end of the project and it creates an overwhelming tension during that period. A great number of critical risks are created since a lot of things can go wrong and it is extremely difficult or even impossible to fix any problems that pop up in this phase due to the pressure of the deadline. (Humble & Farley 2011, 5-6.)

1.2 Motivation of the Study

After having worked in various software development projects and applied Scrum as the main project management approach, the authors have realized and experienced the existing limitations of Scrum Methodology and set out to find a solution.

During the research process, the authors discovered Continuous Integration (CI) and Continuous Delivery (CD) and noticed that the implementation of CI and CD in Scrum projects can bring many benefits that can improve the software development process

With this objective in mind, the authors decided to carry out further research and came up with the following research question:

What are the benefits of implementing CI and CD in Scrum-related projects, and how can CI and CD improve these projects?

1.3 Thesis Structure

The thesis includes eight chapters. The first chapter includes a short introduction to the topic, and introduces the reason for conducting the study and the research question. The second chapter demonstrates the design of the research process. It includes research methods, data collection and data analysis of the research.

The third chapter provides a theoretical background. It covers the theory of all three software development methods, which are Scrum, CI and CD, and related technologies, which are Version Control, Docker, Web Real-Time Communication (WebRTC), MEAN stack and Amazon Web Services (AWS).

The fourth and fifth chapter demonstrate development methods and the development process of two artefacts. Chapter four describes the development process of an application which only adopts Scrum. The fifth chapter proceeds to describe the development process of another application which was developed by adopting Scrum, CI and CD. Next, chapter six, summarizes the differences between the project using only Scrum and the project using Scrum with the support of CI and CD.

Chapter seven introduces conclusions. It summarizes the study and provides answers to the research question. Finally, chapter eight discusses the limitations of the study, its reliability and validity, and presents suggestions for further research and development.

2 RESEARCH DESIGN

Research design covers the research method and approach for this study. Moreover, it also discusses the way data is collected and analyzed.

2.1 Research Method

Design science research or design research is the method applied in this study. According to Barab and Squire (2004, 18), the purpose of design science research originated from the need for a change towards something better. A study applying the design science research method contains a strict process of constructing artefacts to undertake the mentioned problems, measuring the pattern and explaining the project results to the pointed audience (Peffers, Tuunanen, Rothenberger & Chatterjee 2008, 6).

Development work aimed at improving a product, service, process or action, and doing research represent the two processes related to design research. (Kananen 2013, 50). Figure 1 shows the relationship between these two processes.

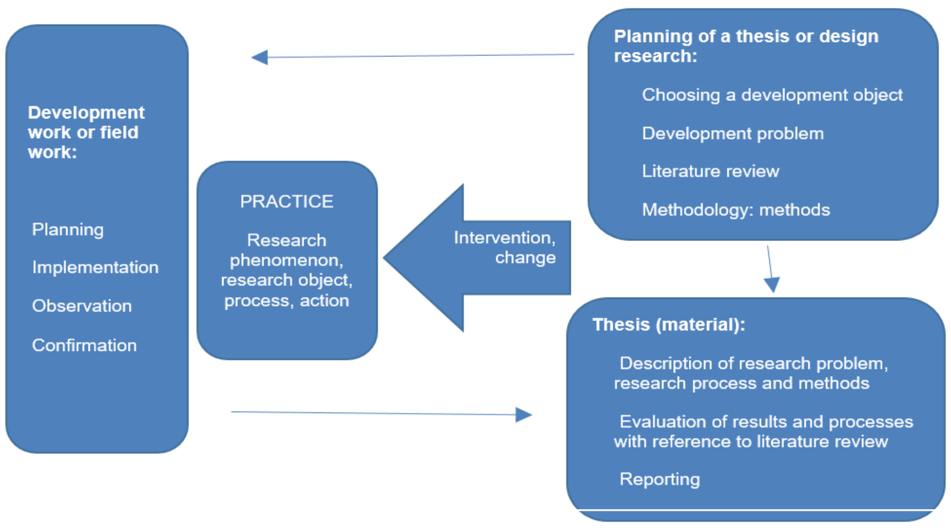

Development work or field work:
Planning
Implementation
Observation
Confirmation
PRACTICE
Research phenomenon, research object, process, action
Intervention, change
Planning of a thesis or design research:
Choosing a development object
Development problem
Literature review
Methodology: methods
Thesis (material):
Description of research problem, research process and methods
Evaluation of results and processes with reference to literature review
Reporting

Figure 1 Relationship between development work and thesis (Kananen 2013)

Deductive and inductive are usually referred to as the two distinct research approaches. According to Blackstone (2012, 19-20), the inductive approach starts from specific observation and proceeds to broader generalizations and theories, and is informally called as the “hill-climbing” approach. On the other hand, the deductive approach applies theory to a particular situation to acquire evidence that is utilized for a specific conclusion. This method can also be called the “waterfall" approach. Figure 2 describes how the deductive approach works according to Burney (2008):

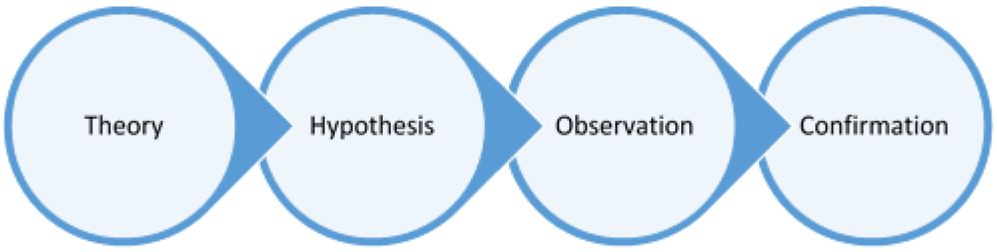


Figure 2 Deductive Research Approach (Burney 2008)

Developing new concepts or theories is not in the scope of this study since this would require too much time and knowledge. Moreover, the authors’ purpose is to do research based on an existing solution, Agile Methodology, and apply knowledge from experience and observation ~~into~~ to the development process in order to finally solve the research problems and answer the research question. Hence, this study applies the deductive research approach.

In addition, when using the design research approach as a research framework, a qualitative research approach appears to be the most suitable approach since it is required at all phases of this framework (Kananen 2013, 102). Additionally, Rogelberg (2004, 162-164) also states that the qualitative approach helps to get a deep understanding of a phenomenon and inclusive knowledge on processes. Therefore, the authors decide to apply qualitative approach in this study.

According to Kananen (2013, 102), data collection methods and data analysis are combined when using qualitative research methods. He also states that in order to analyze material using qualitative analysis methods, data must be collected by using qualitative data collecting methods. Data collection and analysis are discussed next.

2.2 Data Collection and Analysis

According to Kananen (2013, 103), observation, theme interview, and literature references are three most crucial data collecting methods of qualitative research. They can be used depend on the kind of phenomenon subject to research, its distinction and the information accuracy and authenticity.

This is a practice-oriented study that aims to explore how to improve Scrum by comparing and studying two application development processes. The study applies observation as its data collecting method because it is a suitable method when collecting data on processes (Kananen 2013, 104).

Research diary is a basic method used during the research stage in order to record the observations regarding a phenomenon (Kananen 2013, 108). The authors decided to write a research diary during the development tasks since this is helps in answering two crucial questions that can be used to evaluate the results of this study: what kind of information does the material provide and what findings can be made from it (Kananen 2013, 108).

The project applying the Scrum method will be described in detail in chapter 4 in order to explain the related drawbacks. As a result, these drawbacks are then used to establish the evaluation criteria that helps the authors to focus on what should be noted during the development process of the proposal solution. The solution be described in chapter 5. The criteria will be described in chapter 6.

After finishing the project with the proposal solution, the authors collect all the notes and discussion reports for content analysis. The purpose of ~~the~~ content analysis is to acknowledge the core of the text and make a concise statement out of it (Kananen 2013, 128). The criteria drawn based on the drawbacks of the project that only applied Scrum will be used to compared with the new development process. The aim is to find improvement suggestions. Furthermore, other benefits and weaknesses of each technology used during the development process will also be described in chapter 5. Finally, the authors provide a conclusion to summarize the thesis and answer the research question arisen in the introduction. Figure 3 illustrates the diagram of this process, which is adapted for qualitative research by Kananen (2013, 103).

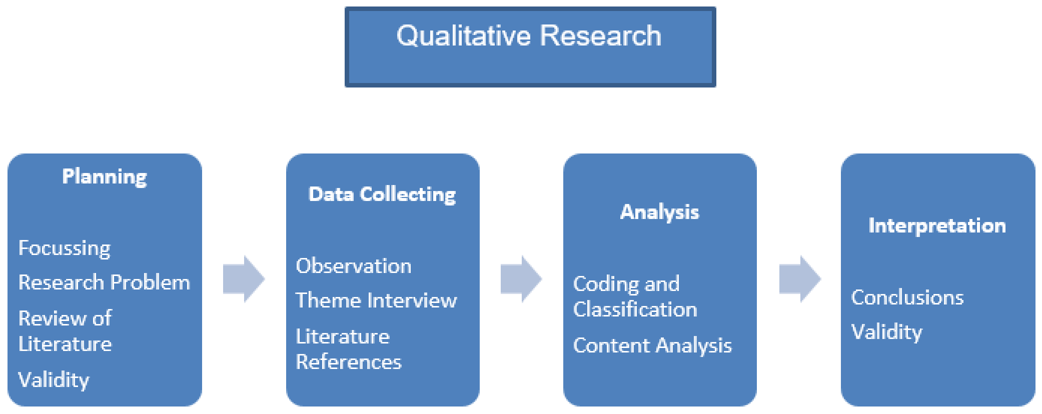


Figure 3 Process diagram for qualitative research (Kananen 2013)

3 THEORETICAL BACKGROUND

The theoretical background gives an insight into what methodologies and technologies will be applied in this study and during the development process.

3.1 Methodologies

This subchapter discusses the following three project-management methods: Scrum, CI and CD.

3.1.1 Agile and Scrum Methodology

Agile Software Development Method is a modern approach to project management. It pays attention to collaborating closely with the customer, continuous development and improvement throughout the project, and quickly response to changes in the requirements of the customer. (Rico, Sayani & Sone 2009, 1-2.)

Agile Manifesto and the Declaration of Interdependence

It is certain that Agile Manifesto and the Declaration of Interdependence are the basis of ~~the~~ Agile Software Development and modern project management. While Agile Manifesto establishes a list of values and principles for software development, the Declaration of Interdependence provides a collection of results that the user of Agile Software Management can expect if the adaptation of Agile is implemented successfully. (Pham & Pham 2012, 3.)

Agile Manifesto was first drafted in 2001 by a team of experts in software engineering. The ambition of the team was to create a better software development process After the meeting, four values for the modern software development were established:

- Individuals and interactions over processes and tools
- Working with software over comprehensive documentation

- Customer collaboration over contract negotiation
- Responding to change over following plan

(Pham & Pham 2012, 3.)

Furthermore, Agile Manifesto also includes twelve principles:

1. Creating customer's satisfaction through early and continuous delivery of the software is the most important aspect
2. Improving customer's competitive edge by quickly adapt to changes in every stage of the development process
3. Providing the customers with working software within a short period of time, which varies from two weeks to one month
4. Collaboration between the business team and technical team is vital during the whole development process
5. Motivating and trusting each member to provide great results for their work
6. Encouraging face-to-face communication within the team
7. The measurement of progress is how well the software functions
8. The necessary of thinking about maintainability during development process in Agile
9. Paying attention to the quality of technical and design continuously improves agility
10. Simplicity is a vital aspect
11. Self-organizing team provides the best idea of architecture, requirements, and designs
12. Self-reflection during each interval is essential since it provides solution to become more effective

(Pham & Pham 2012, 4.)

Then, in 2005, a group of project managers established six expected result of a successful implementation of Scrum and how to achieve them. The list was called Declaration of Interdependence:

- Increase return on investment by making continuous flow of value

- Deliver reliable results by engaging customers in frequent interactions and shared ownership
- Expect uncertainty and manage for it through iterations, anticipation, and adaptation
- Unleash creativity and innovation by recognizing that individuals are the ultimate source of value, and creating an environment where they can make a difference
- Boost performance through group accountability for results and shared responsibility for team effectiveness
- Improve effectiveness and reliability through situationally specific strategies, processes, and practices.

(Pham & Pham 2012, 4-5.)

Current situation and history of Scrum

The term “Scrum” was first mentioned in an article named “The New New Product Development Game” by Hirotaka Takeuchi and Ikujiro Nonaka in 1986 (Takeuchi & Nonaka 1986, 4). In their article, the two authors provided a new approach to project management. In this approach, the team is small but each member of the team possesses a unique skill set and the team shares the same goal. The term and approach were originated from rugby, which the authors used as a comparison. Then, in the next decade, Jeff Sutherland, VP of Engineering at Easel, Inc, and Ken Schwaber, who is working for Advanced Development Methods, continued to work on the improvement of their project management methods and productivity in the own company based on the initial theory by Hirotaka Takeuchi and Ikujiro Nonaka. In 1995, Sutherland and Schwaber teamed up and created a new methodology based on their findings and successful projects and named it Scrum. (Pham & Pham 2012, 5-6.)

Scrum in Practice

Firstly, in term of roles and responsibilities, a basic Scrum team consists of three main roles: Scrum Master, Product Owner, and the development team. The development team is small but cross-functional including coders, designers, and testers. (Pham & Pham 2012, 6.)

A Scrum project is usually started with a meeting between the Product Owner and the stakeholders. During this meeting, the Product Owner's responsibility is to receive ideas from the clients and put them in a product backlog. This product backlog is a collection of requirements that have been prioritized based on the negotiation between the Product Owner and the customer. In this backlog, the requirement is ranging from business to technology requirements to simply a bug fix to changes in the design. Furthermore, the requirement is often rewritten into short user stories and is vital to be ready before the release and sprint planning meeting. (Pham & Pham 2012, 7.)

Next, the whole team organizes a release and sprint planning meeting. Although the release meeting is not compulsory at first, it is proven extremely helpful and essential since it enables not only the Product Owner to understand more about the products for later meetings with the customer but also the team to establish an estimated delivery schedule of the software. Furthermore, it is vital for the team to conduct a sprint planning meeting. The best practice for the sprint meeting is to divide it into two parts: "What" and "How". During the first part, the development team and the Scrum Master go through all the existing requirements in the product backlog with the Product Owner. The aim is to decide which requirements will be carried out in which sprints and what the goals of that sprint are. Later, during the "How" part, the team estimates how much it will take to complete each task and put all that information into a tracking system, which can be a project management software or a simple whiteboard, before working in the actual sprint. (Pham & Pham 2012, 8.)

In most cases, a sprint's duration ranges from one to four weeks. In addition, the requirement in the Sprint backlog will be kept as a constant. Changing the Sprint backlog is extremely rare and negative. Then, a daily stand-up is carried out during every sprint for the Scrum Master to keep track of the progress of the sprint. When the sprint is about to end, the team has a Sprint review for inspection of the previous sprint and further adaption and improvement for next sprint. In this meeting, the team discusses the completed task as well as uncompleted tasks. The Product Owner gains an overview of the status of software and provides feedbacks for the team. Furthermore, the product owner can provide additional information about the new changes in requirements and direction of the market. (Pham & Pham 2012, 8-10.)

Lastly, a retrospective meeting is carried out in the very end of the sprint. The team discusses what went well and what went wrong in the previous sprint in order to find a solution to improve the productivity of team before next sprint. (Pham & Pham 2012, 11.)

These sprints are executed repetitively until the end of the project. The goal of Scrum and Agile, in general, is to improve the effectiveness of the development process by constantly reviewing and satisfying the customer by offering an adaptive version of the product as fast as possible.

Benefits of Scrum

Although the implementation and practice of Scrum seem complicated and difficult to carry out, it is proven that a successful adaptation of Scrum leads to various advantages. The outcome includes

- A risk reduction system
- A lean software development cycle
- An adaptive project management process
- A framework based on team self-organization, motivation, ownership and pride

(Pham & Pham 2012, 15.)

Possibility for improvement

At the moment, one of the biggest setbacks of Scrum is the interval between each release and delivery of the software to the customer. Due to how the team is organized in Scrum, it is possible for a team to take from one to four weeks to gather every member’s work and put them into a release version. It results in the fact that it is unavoidable for the clients or the end users to wait for that period to see a new and working version of the application. Although it has been a huge improvement comparing to the traditional Waterfall Project Management Methods, which the customers only see the products in the last stage of the project, shortening the time between each release is still necessary. The top priorities of Agile Software Development have always been to satisfy the customer through the early and continuous delivery of well-functional software and quick adaption to changes and problems.

Then, another issue is the conflicts after each integration by developers in a group. It is common for the developers to work on their own, do the unit test, build the application locally and see that the application works completely as expected. However, when the developers start to submit the changes with the rest of the team, unexpected conflicts and problems may occur. In addition, the longer the developers work separately, the more complex the problem gets and the more difficult it is for them to be solved.

However, these problems can be solved by adapting of CI and CD into Scrum.

3.1.2 Continuous Integration

In this part, the authors provide the audience with the definition, evolution, benefits of CI as well as how CI is implemented in theory. In the end, the authors explain how the adaptation of CI support and improve Scrum.

Definition

CI is a development approach where each developer of a team commits the code for integration daily. During the integration, the software goes through an automated test and build. After that, the system gives immediate feedback about the state of the software. Thanks to this process, the team is capable of detecting bugs or defects and dealing with them quickly. (Duvall, Matyas, & Glover 2007, xx.)

Evolution of CI

Firstly, it is essential to be clear that CI is not a completely new software development approach but only an advancement in the development of integration process. The practice of executing builds frequently has always been considered as one of the best practices for years and been mentioned in several books about software development processes. Moreover, the development and raising the popularity of CI is also closely related to the development of Agile Methods. Thanks to Agile Methods, software teams has started to pay more attention to continuous builds. (Duvall et al 2007, 36.)

How CI works and key features

In order to fully understand CI, its processes, and features, the authors figure that the best approach is to go through every step in a CI process and explain the details and a related component of each step. Here is what a basic CI process looks like:

- A developer completes a small task in his local machine and commits the changes to the version control repository
- The CI system picks up the last changes in the repository as soon as possible. The maximum advisable time is a few minutes.
- The CI system carries out a build of the software based on a build script.
- After the build, the CI system provides the developer the information of the person who has made the changes and the result of the build immediately.

(Duvall et al 2007, 5.)

Next, the authors explain each step of the process in details for better understanding.

In the first step, the three most important factors are the frequency and the size of the commit as well as the existence of version control. Up until this point, the application of version control repository is undeniably vital for every software development. It is essential for programmers to keep a record of the changes of the source code and other assets. At the moment, there is a few number of existing version control systems supporting CI, such as CVS, Subversion, and Git. For the both projects, Git is chosen. Description and the reason why the authors choose Git among other components are provided later in the chapter 3. (Duvall et al 2007, 7-8.).

For the frequency and the size of commit, it is recommended that the programmer commits small tasks as early and frequent as possible. The reason behind this practice is to make it easier and more convenient for developers themselves to keep track of the problems in case of a failed build. The more complex each change is, the more difficult it is for the developers to identify the problems and deal with it. (Duvall et al 2007, 39-40.)

After the developers have committed their works to version control repository, the rest is up to the CI system. The role of the CI system is to get the latest copy of the repository and build the application. The developer is capable of setting the server to scan the repository for changes every few minutes or even every hour, although this practice will not be considered as CI. (Duvall et al 2007, 8-9.) Nowadays, there are many free and open source CI systems such as CircleCI, TravisCI, and Jenkins. Based on the authors’ experience and research, TravisCI has been chosen for the CI systems.

After retrieving all the changes in the repository, the system builds the test based on build scripts. The build script can be only one or a collection of scripts. Their function is not only to compile but also to test, inspect and deploy the software. Although it is a common practice for developers to build their software on their local machine and integrated development environment (IDE), it has not been strongly recommended since a build is only appropriate if the same build can be executed without the IDE. That is the reason why the use of CI and build script is necessary for the quality management of the software. (Duvall et al 2007, 10.)

The last and most vital step in every CI system is the feedback process. Nowadays, email is the most common practice for feedback. After building and testing the software, the system will generate the result of the build as well as feedback and send it the developer via email. The key factor in the feedback mechanism is time. It is essential that the developers receive the outcome of the latest build as soon as possible since the earlier a problem is identified, the easier and quicker the programmer can fix it. (Duvall et al 2007, 10.)

The Benefits of CI

If CI is correctly utilized and implemented, it provides the development team with the following benefits:

- Reduction of risks
- Automate repetitive manual process
- The software is deployable anywhere and anytime
- Better project visibility
- Greater confidence from the team about the quality of the software.

(Duvall et al 2007, 29.)

Firstly, the most popular benefit that CI offers is the risks reduction. By making the software integrate many times per day, the developer can identify and fix problems rapidly and the team is able to keep track of the quality of the application frequently. It also avoids assumption by the developers by providing real-time and accurate feedbacks after each integration. (Duvall et al 2007, 30.)

Then, the automation of the repetitive processes is also a major benefit. The automated system and processes save time, costs and effort. Thus, it enables the team to concentrate and work on other activities. (Duvall et al 2007, 30-31.)

Next, from the customer’s point of view, the fact that software is always available for deployment is the greatest benefit. Thanks to CI, the software can be easily updated with small changes and ready to deploy immediately. The changes range from bug fix to a new feature and the customer will be delighted by how a recent requested small change is quickly dealt with. (Duvall et al 2007, 31.)

Furthermore, CI provides developers with greater visibility of the project as a whole. In the past, in order to make a decision about innovating an improvement, the developer is required to collect the data manually or even worse, make a guess. As a result, it takes a lot of time as well as energy and the required information might not be collected at all. However, thanks to the application of CI, the process has become a lot easier. Continuous System provides real-time data about the latest build such as the defect percentage and completion progress. In addition, the trends of the software involving the success rates of total builds or overall quality of the software is accessible thanks to the system. (Duvall et al, 2007, 31.)

Lastly, since the system runs the software through a thorough series of automated test and inspection, the team is certain that the quality of the product is qualified for deployment if the build was successful. Besides, if the build failed, the developer is immediately informed of the problems and where the causes are. (Duvall et al 2007, 32.)

How CI improves Scrum

CI supports Scrum in two aspects: the quality of the software after each integration and the possibility to deploy the application anytime and anywhere. Thanks to the automated testing and automated build, the developers are encouraged to integrate their works as many times per day as possible and they are provided with feedback immediately by the system. This process reduces risks of conflicts when too many developers are working on the same project separately. Furthermore, CI enables the possibility to deploy the software anytime and anywhere. This feature is extremely valuable for the customer since it leads to the fact that a working and quality version of the product is ready to be deployed and delivered to them.

Now, the only issue left is how to deliver this quality version of the product as quick as possible to the customer. Fortunately for the development team, CD provides the solution to the problem.

3.1.3 Continuous Delivery

Definition and the relationship between CD and CI

CD is a software development practice that enables software to be ready for deployment into production anytime and anywhere (Fowler 2013). The main goals of CD are to implement the build, test, deploy and release process of the application automatically. This implementation is called deployment pipeline which is illustrated in figure 4. (Humble & Farley 2011, 3.)

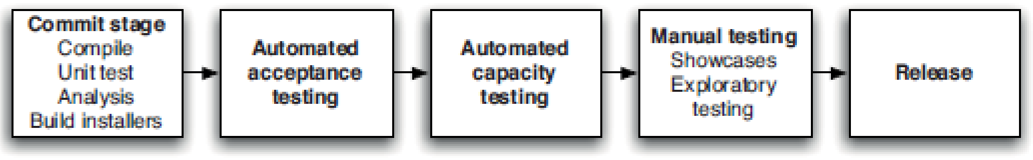


Figure 4 Deployment Pipeline (Humble & Farley 2011, 4)

According to the above figure, it is clear that some of the processes are similar to CI. It is because CI is a part of CD. In CD, the process includes frequently automated builds of the software, automated testing, and inspections in every build and a feedback system which enables developers to detect and deal with bugs as soon as possible. Then, the last step is to release the software into a production-like environment. The idea of CD is to also automate the release process and this step is the one that turns CI into CD. (Fowler 2013.)

How CD works in the release phase

Due to common project management methods and software development practices, the release day of an application is usually stressful and intensive. First, the host environment is required to be prepared. Then, a third-party software is required to be installed. Next, the configuration files are also required to be created and copied to the host server and if the project includes a database system, all of them is needed to be transferred to the production. In short, there is a lot of vital activities needed to be carried out on the release day. However, they are usually done manually, which makes the whole process is prone to human's error, and one mistake can cause the software to not function as expectation although it had passed all the testing before releasing. (Humble & Farley 2011, 5.)

However, by applying CD, the development team is able to improve the release phase of the application by automating all of the required activities during delivery. During the release phase, these automation processes include dependencies and production environment managements. First is to automate the management system of dependencies. A dependency appears when a part of an application is relied on another software to function properly. A common type of dependencies is third-party libraries. It is a piece of software that is developed outside the organization and is rarely updated. In order to manage the supported libraries automatically, using version control or declaring them and using tools like Maven and Ruby to download them from the internet are the two main practices at the moment. Using version control, which is to pushing every library along with the software into the repository, is the simple approach and is recommended for small projects. It ensures that the team is using the exact same pieces of code from the libraries. However, if the size and number of libraries grow, the size of the repository also increases and it makes the build-time during integration longer. Therefore, it is considered preferable to use dependencies management tools like Maven and Ruby. The tool enables developer their required libraries and their version. The tool will then download that version of libraries whenever there is a build or a deployment. The drawback of this practice is that developers are required to configure both the tool and the libraries carefully in order to execute the repeatable build. However, the effort is worthy because it maintains the size of the software and makes sure that the team is working on the same code simultaneously. (Humble & Farley 2011, 351 – 355.)

Then, automating the configuration and creation of a production-like environment is another essential step in CD. The application's environment, which includes hardware, software, infrastructure and external system, is important for an application's functionality and quality. One change in one of these things might cause failure of the product. Nowadays, the common practice for this process is to install every piece of software and rewrite the configuration files manually. Although the practice is simple and straightforward, there are too many problems and risks that go along with it. These risks are:

- One change is able to break everything
- The amount of configuration information is too great to manage and handle
- Difficulty in recreates the environment for testing purpose
- Identifying and fixing a problem in the environment is difficult and time-consuming

Therefore, it is recommended to automate these processes in order to save time, reduce risk and make the environment establishment be easy and convenient. In CD, the common practice is to apply version control to the configuration files and take advantage of environment management tools like Puppet to create the environment based on the configuration files. It is essential to keep the files in version control because it allows the team to keep off the changes of the configuration. Whenever a build based on a new configuration goes wrong, version control enables the team to revert to a previous and working configuration. Then, same as the management system for dependencies, the management system for environment allows developers to declare all the related information about the operating system and software, which is the purpose of all the configuration files. Based on the information provided in the configuration, the system is able to download and install the needed dependencies. This process enables the development team to recreate the environment quickly for both testing and deploying purpose. (Humble & Farley 2011, 49-53.)

The benefits of CD

As mentioned above, since CI is a part of CD, CD inherits all the benefits of CI. These advantages include the decrease in risks, the increase in the team's level of confidence and deployment flexibility. (Humble & Farley 2011, 17-21.)

Further, CD also offers the development another benefit, which is stress reduction. By automating every repeated and manual processes, the team avoids errors and is always certain that the software and the production environment is stable, well-configured and ready for deployment and delivery. It results in the release day being much less tense and overwhelming for developers. (Humble & Farley 2011, 20.)

Lastly, since the release process become smoother and more convenient for developers, it enables the team to deliver the software to the customer faster. As mentioned in the CI section, the question is how to deliver the quality and deployable version of the software to the customer immediately after integration. CD solves the problem by automatically configuring the production environment and it ensures that the software is truly ready to be deploy anywhere and anytime. This practice is extremely supportive in Agile in general and Scrum in particular because based on the Agile Manifesto principles, delivering an early version of the product frequently is always the top prior of every Agile development team.

3.2 Technologies

Technologies covers the general definition of concepts and tools used for developing the web application. Furthermore, the explanation as to why those technologies are chosen is also included in this chapter.

3.2.1 Version Control

Loeliger and McCullough define Version Control System (VCS) (2012, 1) as “a tool that manages and tracks different versions of software or other contents”. They also address some of the critical roles of VCS such as developing and preserving a repository of content, offering admission to historic versions of each datum, and storing all modifications in a log. In short, the purpose of VCS is that whenever developers would like to make an abundant of modifications to the existing content, they can perform an action as marking those changes as a stage. This stage could be seen as a failsafe mechanism in case that things go wrong and are not as they expected, they can easily revert to the states that things were still in control. Somasundaram (2013) discusses this sort of issue through the demonstration of the content creation with and without a VCS. Figure 5 and 6 show his demonstrations.

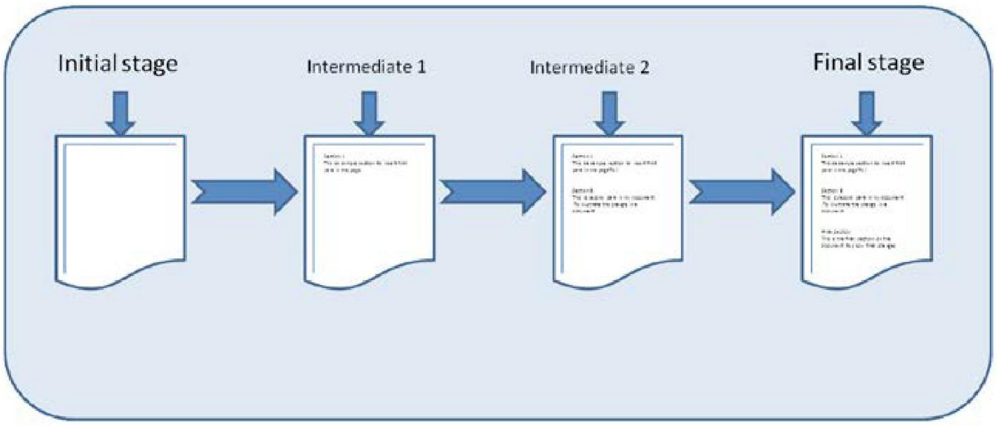


Figure 5 Workflow without Version Control System (Somasundaram 2013)

The flow of the above figure is one-way, which means that developers cannot go back to any previous phases from the final stage to create a new direction with new purpose without any data loss. The only alternative approach developers could do in order to preserve data is utilizing the “save as” option which is giving the file a different name and starting any modification from that file.

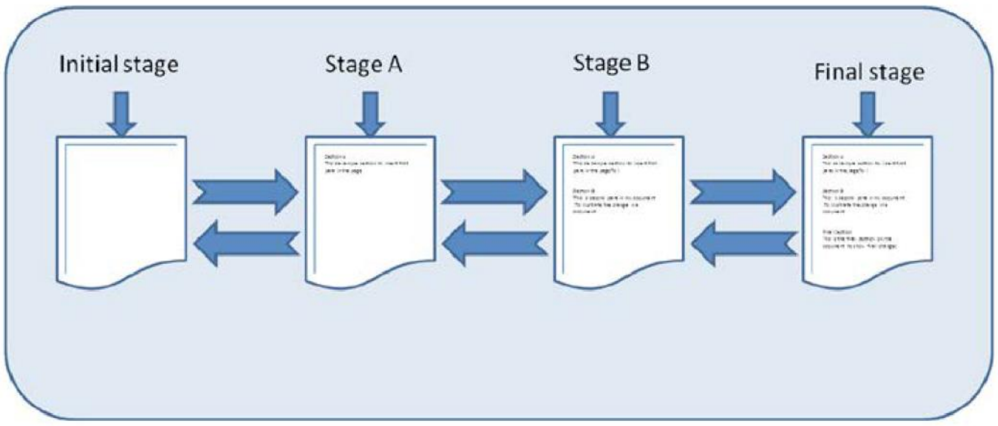


Figure 6 Workflow with Version Control System (Somasundaram 2013)

In contrast, utilizing a VCS can make the workflow look like the previous figure. Developers now can mark each and any alteration that they think important as a new stage and go on with further development. They now can go back to any previous stages that they have marked whenever they want.

Santacroce (2015, 1) points out two different types of VCS, which are Centralized Version Control System (CVCS) and Distributed Version Control System (DVCS). Table 1 compares their differences and some examples for each VCS.

Table 1 Differences between Centralized VCS and Distributed VCS

	Centralized VCS	Distributed VCS
Description	Developers can access the files that are kept on a remote server from their local machines.	Developers can have or not a single server or more and they can work offline despite internet connection

	Possibility of losing works since there is only 1 single unit	
Examples	Concurrent Version System (CVS), Subversion (SVN), Team Foundation Server (TFS)	Bazaar, Mercurial, Git.

Git

Git is an open-source, high-performance, flexible and hard-to-corrupt DVCS utilized in the Linux kernel project (McQuaid 2014, 3). It was created by Linus Torvalds in 2005 and now is maintained by Junio C. Hamano (Gajda 2013, 1). There are several factors that make Git become more and more popular and set Git apart from other VCSs. First, atomicity guarantees the avoidance of partial completions when handling content with Git, which ensures there is no data loss or version disparity occurring. Second, unlike other VCSs which storing different files among versions of each file, Git focuses on the file relation and takes a snapshot of the entire set of files whenever a version is created. Therefore, instead of storing multiple versions of the same file, Git only refers to the previous snapshot if there is no modification in the file content. This makes the operation performed using Git takes only a couple of seconds even when handling a lot of files. Last, Git provides a strict security for developers' file content by using SHA-1 hash which performs a checksum before those files are stored, which means that no one can possibly modify the contents of any folder or file without Git's notice. (Somasundaram 2013, 16-18.)

GitHub

GitHub is a service that offers Git repository hosting, issue trackers, and several other tools for collaboration teams (McQuaid 2014, 224). According to Westby (2015, 211), GitHub is the most popular hosted VCS for open source projects with more than nine million users in 2015. Thus, the authors decide to use Git and GitHub for version controlling in this project.

3.2.2 Docker

In order to understand Docker, the definition of containers should be mentioned first. Mouat (2016, 3) defines containers as “an encapsulation of an application with its dependencies.” While they are considered mistakenly as virtual machines at first glance due to the fact that they also contain a separate instance of an operating system which can be used to install and run applications, there are several benefits that developers can only find when they are working with containers instead of virtual machines. First, the elimination of environment transition could help developers avoiding lots of errors and bugs thanks to the portability of containers. Next, the lightweight attribute can ensure that a single host machine now can run multiple containers simultaneously in order to imitate a production-ready distributed system. Last but not least, containers deliver simplicity to end-users, which means now users can avoid the complex configuration and installation when downloading and running applications that cost them a huge amount of time before. (Mouat 2016, 3-4.)

Docker is a container management system that assists developers in managing Linux containers (LXC) in a straightforward and comprehensive way (Gallagher 2015, 3). In more practical sense, Docker wraps and extends LXC in numerous approaches, such as portable images and user-friendly interface, in order to construct an entire solution for container foundation and allocation (Mouat 2016, 6). In the scope of this study, the purpose of the authors is not covering all the Docker concepts, but only involving the crucial components and their brief definition for the software development process related to this project:

- Docker images: templates for Docker containers, which are made up of multiple layers of the read-only filesystem.

- Dockerfile: the basic building block of Docker containers, which contains an instruction that can be used to create a Docker image.
- Docker Hub: the default service for hosting and distributing images, it can also be considered a community where developers can share, find and extend Docker images.

(Mouat 2016, 24-28.)

- Docker Hub Automated Build: Docker Hub‘s special feature which offers developers to utilize build clusters in order to create images automatically from a repository (GitHub or Bitbucket) containing a Dockerfile. There are several advantages of Automated Build compared to ordinary image creation such as images are built exactly as indicated, the availability of Dockerfile for users and up-to-date repository. (Docker 2016.)
- Docker Cloud: Software-as-a-Service hosted by Docker, which provides developers the ability to manage, deploy and scale their applications in any environment. (Docker 2016)

Docker Hub Automated Build and Docker Cloud play important roles in the CI and CD respectively, and their utilization with related concepts and features will be described in details in chapter 4.

3.2.3 Web Real-Time Communication

In May 2011, an open-source project for web-based real-time communication known as WebRTC was published and released by Google (w3.org 2011). Ericsson Labs first created a pre-standard concept, Connection Peer Application Programming Interface (API) in January 2011, and then has expanded to the point that there are now advanced implementations in certain modern web browsers. Browsers now are able to communicate and exchange real-time media with other web browsers in a peer-to-peer fashion, without any third-party software (Loreto & Romano 2014, 1.). Ristic (2015, 2) points out the difficulty of developing a real-time communication application without WebRTC, which is having to import a vast of frameworks and libraries to handle connection dropping, data loss or Network Address Translation (NAT) traversal. And with the assistant of WebRTC, all of those libraries or frameworks are now built into the web browser API, making the implementation details easier. For more details, Sergiienko (2014, 4-5.) makes a list of advantages of using WebRTC in business in general level, which also are criteria that help the author to consolidate the decision of choosing WebRTC to be the protocol for developing the desired application instead of others. They are:

- Cost Reduction: No deployment software for the customers and since WebRTC is a free and open source technology, business no longer is required to pay for complex solutions or IT support.
- Plugin-free: Formerly, building interactive media web-based application required users to install or use several solutions such as Flash or Java Applets, which also led to paying attention to the distinctness among operating systems and platforms.
- Peer-to-peer communication: No middle point server is needed since the communication now will be established straightforwardly between two or more endpoints.
- Simplicity: WebRTC functionality can be simply implemented into the web services or applications by using JavaScript API and other increasing developing frameworks, which means that there is no longer the need for professional developers or specific knowledge.
- Cross-platform: Every operating system with a web browser can run WebRTC application
- Open source: New errors and bugs can be discovered and solved effectively and quickly by a growing community.

With the purpose of focusing more on the software development than on the technology, the authors choose to use simpleWebRTC, which is a bundle of libraries wrapping WebRTC APIs and providing code snippets, client components, and server implementation to simplify the process of developing WebRTC application. SimpleWebRTC introduction and implementation will be described in more details later in chapter 4.

3.2.4 MEAN Stack

According to Haviv (2014, 7), MEAN stack is defined as follows

> The MEAN stack is a powerful, full-stack JavaScript solution that comprises four major building blocks: MongoDB as the database, Express as the web server framework, AngularJS as the web client framework, and Node.js as the server platform.

MongoDB

MongoDB is a document database, which stores documents as binary JSON, or BSON. Unlike relational databases, which holds the concept of columns defining the name or type of data and rows defining entry, document databases present the concept of rows in which each row is a document which defining and holding the data. The following snippet demonstrates a simple MongoDB document.

```
{
        "firstName" : "Simon",
        "lastName" : "Holmes",
        _id : ObjectId("52279effc62ca8b0c1000007")
}
```

(Holmes 2013, 12.)

Express

Express is a web application framework for Node.js, which aiming to establish a web server capable of listening to incoming requests and returning appropriate responses. It also abstracts away some of the complexities related to routing Uniform Resource Locator (URL), building

Hypertext Markup Language (HTML) pages and using sessions. (Holmes 2013, 10-11.) In short, Express makes the developing of web application on top of Node.js more rapid and straightforward.

AngularJS

Over 25 years of web development, three-tier architecture has been utilized in various technology stacks for building web-based applications. This architecture includes three crucial layers: database, server, and client and can be also known as Model-View-Controller (MVC) architectural pattern (Ramirez 2000). AngularJS is a front-end JavaScript framework designed to utilize the MVC architectural pattern for building single-page applications or SPAs. In order to extend the functionality of HTML, which is its fundamental purpose, AngularJS applies specific attributes that connect JavaScript business logic and HTML elements. This allows two-way data binding between models and views and makes DOM manipulation cleaner. Moreover, with the utilization of MVC architecture and dependency injection, code structure and testability are also improved significantly. (Haviv 2014, 162.)

Consequently, the authors aim to use MEAN stack for developing the video conferencing application with WebRTC. Each component in this stack has many concepts and advanced techniques that will not be covered in this study. The purpose of the authors in this chapter is only presenting a basic definition of each technology and their advantages. Some additional technologies and how to connect the components together in order to build a complete application will be defined and described later in chapter 4.

Node.js

Node.js is a software platform, which contains a built-in Hypertext Transfer Protocol (HTTP) server library, allows developers to create their own web server and build web applications above it (Holmes 2013, 6). Haviv (2014, 30) also defines Node.js as an uncomplicated, highly effective, and simply scalable platform with the capability of running complex applications.

3.2.5 Amazon Web Services

AWS is a cloud computing service by Amazon, which offers a reliable, scalable and low-cost cloud infrastructure to businesses (AWS 2016). AWS provides a wide collection of global cloud-based services such as storage, compute, backup, analytics, developer and management tools, which assist organizations in lowering IT cost, scaling and as a result, moving faster (AWS 2016). In this study and in the development process, in particular, the authors merely choose to use several services of AWS, which are Elastic Compute Cloud and Identity and Access Management.

Elastic Compute Cloud (EC2) is a web service of AWS which offers cloud-based resizable compute capacity. Users can create and control instances easily with web service APIs, which also provide the possibility to customize the instances with any type, operating system, and software packages. (AWS 2016)

Identity and Access Management (IAM) is an AWS web tool designed to enable developers to control securely the access to AWS services as well as resources for their users. IAM assists in creating and managing users and groups, creating roles and permissions to allow or deny the access to AWS resources. (AWS 2016)

4 DEVELOPMENT PROCESS OF THE FOOD MANAGEMENT APPLICATION

In this chapter, the author describes the software development process and management of the first artefact. The management method using in this artefact is only Scrum Methodology.

4.1 Introduction

In fall of 2015, the authors and three other students studied a course called Agile Web Application Development Project. In this course, the students were expected to develop a quality and functional web application called Food N' Stuff using the knowledge that they have learned in three other courses, which are Agile Software Development Methods, Web Application Development, and Testing.

4.2 Project Goals and Software Specification

The project contains a few number of goals. The first is to create a functional web application by applying the knowledge from the Web Development Application and Testing courses. Then, it is compulsory for the team to practice Scrum during the development process since it has been taught in the Agile Software Development Methods course.

For the specifications of the application, it is given by the stakeholders, which is the teacher of the course. It should allow the users to register, login, browse the community's recipes, select favorite, create a new one and use them to create shopping plan.

4.3 Development Team

The team consists of three main roles, which are Scrum Master, Project Owner, and the development team. One author was the Product Owner and the other was the Scrum Master. The other students form the development team with three different roles, which are developer, designer, and tester.

4.4 Project Coordination

For this project, the team executed Scrum closely to the theory. The project was carried out in five sprints which each last for two weeks. In a sprint, the team organized a planning meeting at the first day of the sprint, a retrospective meeting at the last day of sprint and daily check-ups throughout the whole development process.

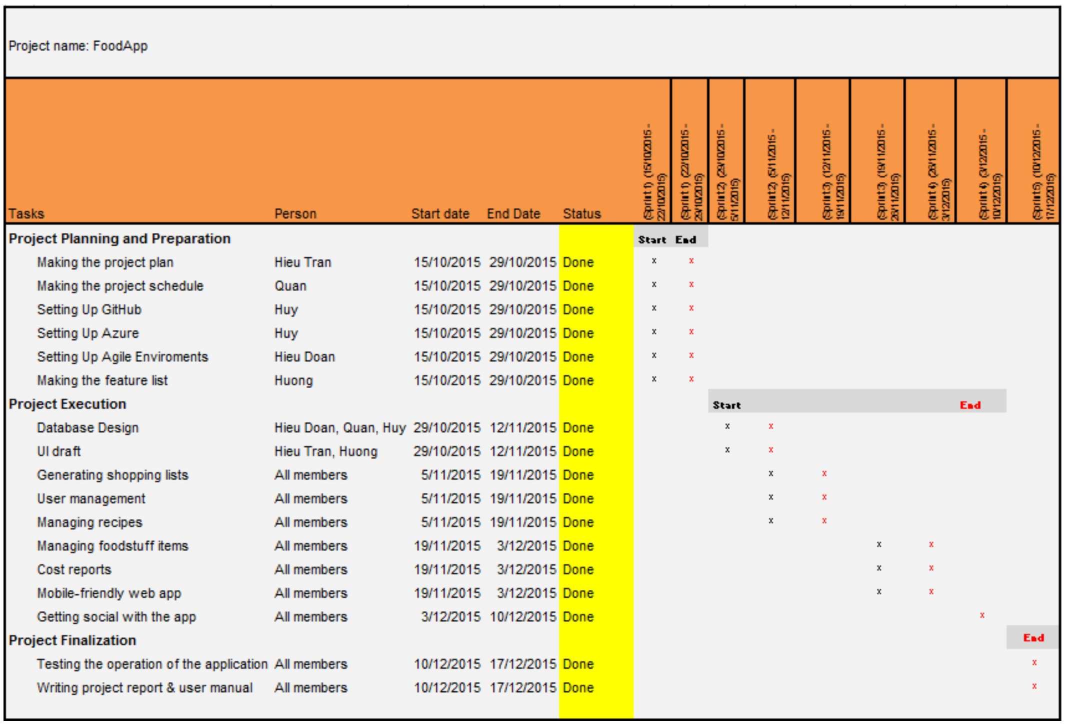
In the planning meeting, the Product Owner prioritized the tasks in the backlogs and the Scrum Master reviewed them and divided the task to the development team.

Then, during the development process, the development team worked on the tasks that were given to them by the Scrum Master in the planning meeting. These tasks varied from new feature to bug fix to testing. In daily check-ups, the Scrum Master carried out daily check-ups properly in order to keep track with every member's working progress. It enabled the Scrum Master to react quickly to changes in the risk of the difficulties and complexity of the tasks or sickness of the members. During that time, the Product Owner's role was to collaborate closely with the stakeholders, who is the teachers of the course, to collect changes, feedback, and new requirements.

Lastly, in the retrospective meeting, the team reviewed all the works that had been done or not done in the previous two weeks. In that same meeting, the team also met up with the stakeholders, showed them the current states of the application and received feedbacks and recommendation for changes and bug fixings from them.

4.5 Development Phases

Figure 7 illustrates the development phases of the Food ‘N Stuff application that the authors have already developed earlier, including the divided tasks for 5 developers, the start and end date as well as the status of each task.



Project name: FoodApp

Tasks	Person	Start date	End Date	Status	(Sprint 1) (15/10/2015 - 22/10/2015)	(Sprint 1) (22/10/2015 - 29/10/2015)	(Sprint 2) (29/10/2015 - 5/11/2015)	(Sprint 2) (5/11/2015 - 12/11/2015)	(Sprint 3) (12/11/2015 - 19/11/2015)	(Sprint 3) (19/11/2015 - 26/11/2015)	(Sprint 4) (26/11/2015 - 3/12/2015)	(Sprint 4) (3/12/2015 - 10/12/2015)	(Sprint 5) (10/12/2015 - 17/12/2015)
Project Planning and Preparation					**Start**	**End**							
Making the project plan	Hieu Tran	15/10/2015	29/10/2015	Done	x	x							
Making the project schedule	Quan	15/10/2015	29/10/2015	Done	x	x							
Setting Up GitHub	Huy	15/10/2015	29/10/2015	Done	x	x							
Setting Up Azure	Huy	15/10/2015	29/10/2015	Done	x	x							
Setting Up Agile Enviroments	Hieu Doan	15/10/2015	29/10/2015	Done	x	x							
Making the feature list	Huong	15/10/2015	29/10/2015	Done	x	x							
Project Execution							**Start**					**End**	
Database Design	Hieu Doan, Quan, Huy	29/10/2015	12/11/2015	Done			x	x					
UI draft	Hieu Tran, Huong	29/10/2015	12/11/2015	Done			x	x					
Generating shopping lists	All members	5/11/2015	19/11/2015	Done				x	x				
User management	All members	5/11/2015	19/11/2015	Done				x	x				
Managing recipes	All members	5/11/2015	19/11/2015	Done				x	x				
Managing foodstuff items	All members	19/11/2015	3/12/2015	Done						x	x		
Cost reports	All members	19/11/2015	3/12/2015	Done						x	x		
Mobile-friendly web app	All members	19/11/2015	3/12/2015	Done						x	x		
Getting social with the app	All members	3/12/2015	10/12/2015	Done								x	
Project Finalization													**End**
Testing the operation of the application	All members	10/12/2015	17/12/2015	Done									x
Writing project report & user manual	All members	10/12/2015	17/12/2015	Done									x

Figure 7 Development phases of Food Management Application

4.6 Development Process

According to the project coordination and development phases, the Food ‘N Stuff project has been completed with the management of Scrum method. The development process including installation, setting up the project as well as developing application features is described in the development phases. The authors do not discuss these tasks in details due to two reasons. First, the development of this project has happened since fall of 2015 hence guiding audience through each step and taking a snapshot for each task again for illustration are extremely difficult and troublesome. Second, the authors have already taken notes during the development process of this project and conclude them to find out the results including drawbacks and limitation, which will be described next. These results, as stated before, are the most important components to establish the criteria needed for the comparison with the new proposal solution. Therefore, the authors consider the development process of this project is not necessary to be mentioned and covered.

4.7 Results

In the end, the project has been successful. The team has delivered a fully functional and well-designed web application in the agreement time. Moreover, the stakeholders have been satisfied with the team as well as the development process of the project due the team has always delivered the software that contains new features and adapts to every change in the requirements well in each retrospective meeting. The software is now published and available on GitHub as Open Source for further development.

4.8 Drawbacks and Limitation

Although the project has been considered successful, the authors, who were the Product Owner and Scrum Master, has identified several limitations of Scrum during the development process of the application.

The first aspect is the interval time between each built and its quality. During the development process, the developers have worked separately and independently from each other. Then, since there has been no agreement on how frequent all the developers should integrate their code to the master branch, the development of the software has gone on differently and separately for days. Thus, after each merge from every member’s work, the team has had to deal with various conflicts and spent extra time on debugging and solving the difference. Moreover, the longer the interval has been and the more code developers have shared, the more problematic the situation has been. Although the team has been able to solve the problems and deliver the project on time, the team has noticed that this kind of problem should not occur and should be dealt with entirely.

Secondly, the availability of the software has been limited. Due to the limitation of the team’s knowledge, the software has merely been tested and deployed locally. It has led to the fact that the team has been only capable of delivering the software to the stakeholder during the retrospective meeting which only has happened each couple of weeks. During that time, there has been always changes in requirements and feedbacks that the Product Owner found that it could be found out sooner if the team could deliver the application on a more frequently basis. Moreover, the earlier the recognition of changes, the better it is for the team to adapt as well as save a great amount of time and effort. Then, whenever the team has been required to deploy the application to a new host server such as Microsoft Azure or LAMK’s virtual machines, there has been extra time and effort spending on setting up the new hosting environment since each of them requires a different kind of configuration and installation.

5 DEVELOPMENT PROCESS OF THE WEBRTC APPLICATION AND THE IMPLEMENTATION OF CI AND CD

This chapter provides the readers with the development process of the second artefact. The software development management method of the artefact is Scrum Methodology with the support of CI and CD. Moreover, the authors also describe the implementation and adaptation of CI and CD in details.

5.1 Introduction

The LamkRTC is an open source project developed by the authors. The aim of the project is to create a WebRTC application for teamwork in LUAS and to see the benefits of applying CI and CD along with Scrum. The project is divided into two main phases: setting up and development. The project has started in August of 2016 and finished within the same month.

5.2 Project Goals

The goal of this project is not only to complete a quality application within a scheduled timetable, meeting all the project specification, but also to apply CI and CD in Scrum while developing the software.

In the end, the final goal of this application is to see how CI and CD can improve the Scrum Methodology.

5.3 Software Specifications

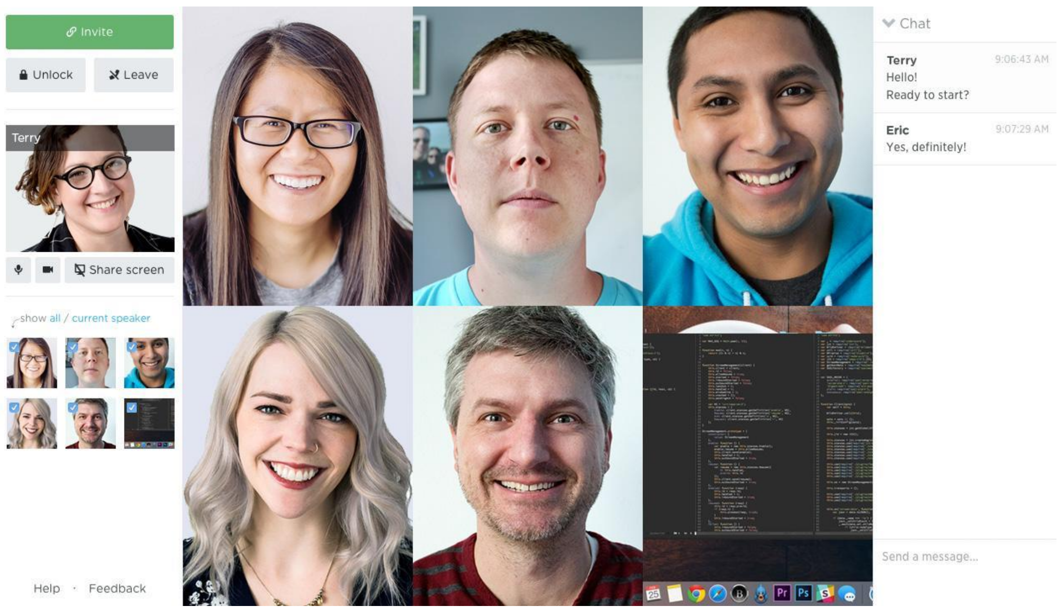
Since the development of the application is the author’s own idea in order to learn about CI and CD as well as create a uniform communication system for the school, the specification of the application is decided by the authors.

For the functionalities of the application, the aim of the project is to create a real-time communication application which meets all these following specifications. These specifications include creating room for teamwork, group conversation via video, audio, and chat, disabling and enabling video and audio stream, inviting others to the room and locking the room to limit the number of participants. In order to accomplish these goals, the authors use the WebRTC APIs and SimpleWebRTC library.

SimpleWebRTC and Talky.io

SimpleWebRTC is a JavaScript library which was developed by Henrik Joreteg and is being preserved by &yet, an IT company located in the US (&yet 2016.). The official site of SimpleWebRTC (https://simplewebrtc.com/) provides the reader with demos and guideline of the library. Moreover, the website introduces Talky.io, a WebRTC application built based on SimpleWebRTC.

Talky.io (https://talky.io/) is a real-time communication application developed by &yet. The application is built with the help of SimpleWebRTC library and provides the users with a great list of features. These features include creating group video conservation for up to 15 people, screen sharing and locking the room. Moreover, the Talky.io is also available for iOS devices such as iPhone and iPad. (&yet 2016.)



Terry 9:06:43 AM
Hello!
Ready to start?

Eric 9:07:29 AM
Yes, definitely!

Figure 8 A screenshot of Talky.io and its features

Software requirements specification is a descriptive list of the goal and expectation of the software. It is the first process of the whole development cycle and it enables the authors to focus on achieving the desired features and goal of the software. (Rouse 2007.)

Talky.io is used as a model for creating the software specification of the application. However, due to the limitation of time and ability of the authors, the application does not have the same features as Talky.io.

After discussion, the authors decide to leave out two features of Talky.io, which are screen sharing and iOS compatibility. The screen sharing feature requires the developers to create a separate browser extension, which leads to extra time. Then, due to the lack of iOS development knowledge, the authors decide not to develop the application on iOS. However, the remaining features still fulfill the original goal of the authors for the application. Here is the list of all the features and requirement of the application:

- Room creation: The users can create a room and name it by themselves.
- Video/Audio chat: The users can communicate with other, who is also in the room, in real-time with video images and sounds.
- Instant messaging: The users can send messages to everybody in the room via the chat box.
- File sharing: The users can share digital files with others using chat box.
- Disable/Enable Video/Audio: The users can turn off and turn on the webcam as well as the audio input and output if they feel necessary.
- Invite people: The user can invite people to join them by copying the URL and send it to others.
- Lock room: The admin of the room, who created the room, can lock the room by himself or herself in order to limit the participants of the conversation.

- Leave room: The users can leave their current room and back to the room creation view.
- Notification system: Inform users when user joining or leaving the room and when the user changes their username.

5.4 Development Team

Since the application is a personal idea of the developers, the developers are the Product Owner, Scrum Master and Stakeholders of the software at the same time.

For the development of the application, the roles are divided into two main categories and each developer takes one:

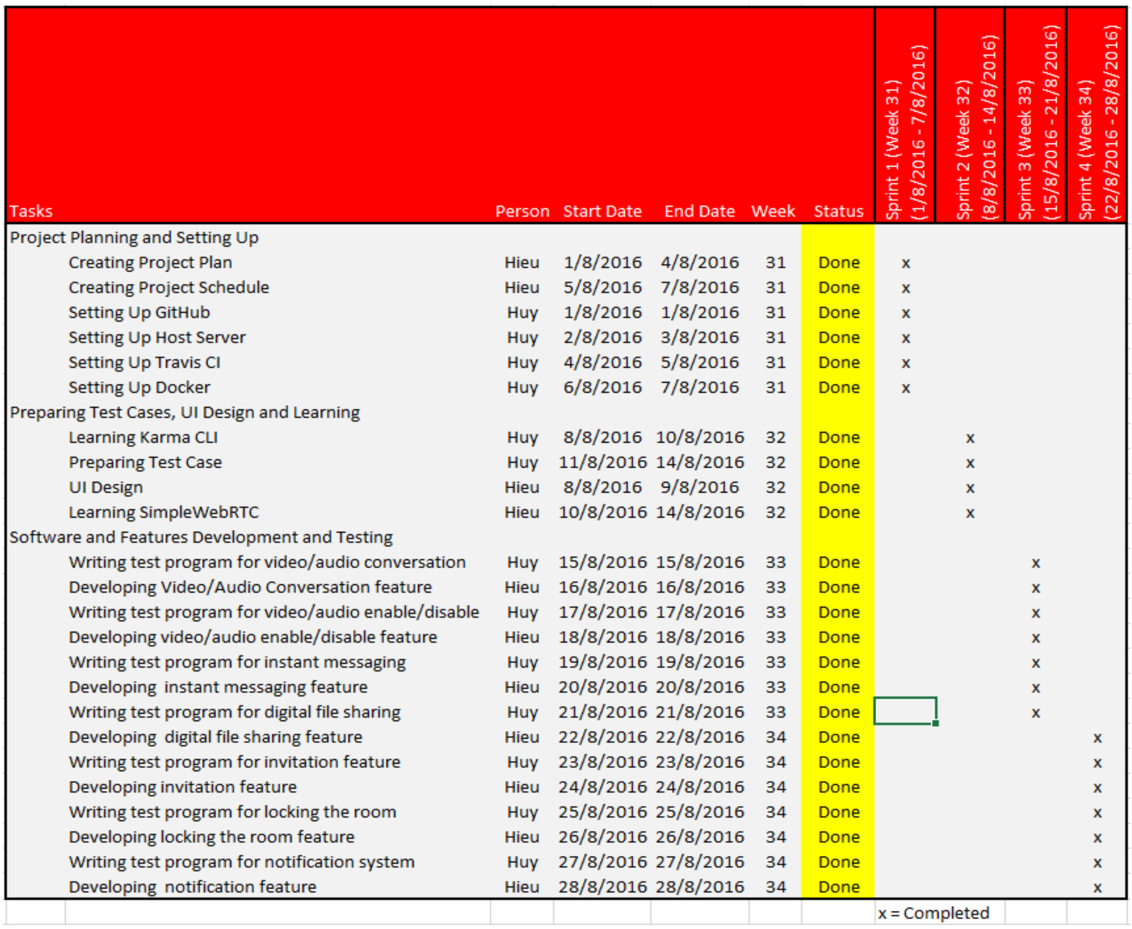
- Front-end developer and designer
  - The role of this position is to design the user interface of the application and develop the functional features of the application.
- Back-end developer and tester
  - Responsibilities of this role are to develop the server for signaling and writing test cases for automation test for CI and CD.

5.5 Project Coordination

During the development process of this application, the authors have carried out the same practices as the food application system with a few differences. This project only lasts for four sprints and each sprint is one week. Then, since the authors have lived in different cities during the time of development, the daily check-ups have been replaced by constantly updating via instant messaging service including Facebook Messenger. Other than that, the project is coordinated in the same way as the food management application project.

5.6 Development Phases

Figure 9 summarizes the development phases of the WebRTC project, including the divided tasks for each developer, the start and end date as well as the current status of each task.



Tasks	Person	Start Date	End Date	Week	Status	Sprint 1 (Week 31) (1/8/2016 - 7/8/2016)	Sprint 2 (Week 32) (8/8/2016 - 14/8/2016)	Sprint 3 (Week 33) (15/8/2016 - 21/8/2016)	Sprint 4 (Week 34) (22/8/2016 - 28/8/2016)
Project Planning and Setting Up									
Creating Project Plan	Hieu	1/8/2016	4/8/2016	31	Done	x			
Creating Project Schedule	Hieu	5/8/2016	7/8/2016	31	Done	x			
Setting Up GitHub	Huy	1/8/2016	1/8/2016	31	Done	x			
Setting Up Host Server	Huy	2/8/2016	3/8/2016	31	Done	x			
Setting Up Travis CI	Huy	4/8/2016	5/8/2016	31	Done	x			
Setting Up Docker	Huy	6/8/2016	7/8/2016	31	Done	x			
Preparing Test Cases, UI Design and Learning									
Learning Karma CLI	Huy	8/8/2016	10/8/2016	32	Done		x		
Preparing Test Case	Huy	11/8/2016	14/8/2016	32	Done		x		
UI Design	Hieu	8/8/2016	9/8/2016	32	Done		x		
Learning SimpleWebRTC	Hieu	10/8/2016	14/8/2016	32	Done		x		
Software and Features Development and Testing									
Writing test program for video/audio conversation	Huy	15/8/2016	15/8/2016	33	Done			x	
Developing Video/Audio Conversation feature	Hieu	16/8/2016	16/8/2016	33	Done			x	
Writing test program for video/audio enable/disable	Huy	17/8/2016	17/8/2016	33	Done			x	
Developing video/audio enable/disable feature	Hieu	18/8/2016	18/8/2016	33	Done			x	
Writing test program for instant messaging	Huy	19/8/2016	19/8/2016	33	Done			x	
Developing instant messaging feature	Hieu	20/8/2016	20/8/2016	33	Done			x	
Writing test program for digital file sharing	Huy	21/8/2016	21/8/2016	33	Done			x	
Developing digital file sharing feature	Hieu	22/8/2016	22/8/2016	34	Done				x
Writing test program for invitation feature	Huy	23/8/2016	23/8/2016	34	Done				x
Developing invitation feature	Hieu	24/8/2016	24/8/2016	34	Done				x
Writing test program for locking the room	Huy	25/8/2016	25/8/2016	34	Done				x
Developing locking the room feature	Hieu	26/8/2016	26/8/2016	34	Done				x
Writing test program for notification system	Huy	27/8/2016	27/8/2016	34	Done				x
Developing notification feature	Hieu	28/8/2016	28/8/2016	34	Done				x

x = Completed

Figure 9 Development phases for the WebRTC project

5.7 Development Process

5.7.1 Installation, Setting Up and Implementation of CI and CD

In this part, the authors explain how to setup the necessary environment for the project, CI, and CD.

Setting up the signaling server

In order for the peers to find, exchange details of the contact, define the session and finally connect to each other with the direct peer-to-peer communication, there should be a signaling server in between for the handshake or coordination. (Manson 2013, 15.) Signalmaster is a simple signaling server for doing signaling for WebRTC, which was also created by SimpleWebRTC team, is the obvious choice to be the signaling server for the web application. There are three tasks required in order to implement this server, they are:

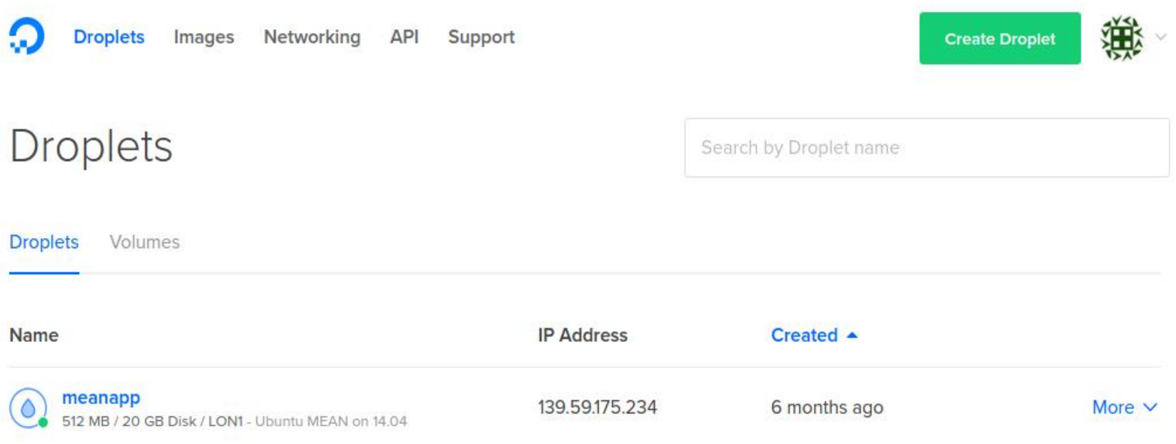
1. Creating droplet on DigitalOcean for production environment
2. Creating Secure Sockets Layer (SSL) certificate for web server
3. Using Express to create the signaling server

Creating droplet on DigitalOcean for production environment

In this situation, the signaling server requires a production environment in order to play its role in coordinating the peers. Therefore, the author spins up a new Droplet, which is a virtual private server on DigitalOcean, to put the server up, running and widely accessible by the peers.

The first step of creating the droplet is to access the DigitalOcean Control Panel and press the Create Droplet button in order to access the Create Droplets page. In the Droplet Creation page, there are several options for developers to choose for their droplet depends on what type of server they want. In the image categories, the authors choose the pre-installed MEAN on 14.04 app since it meets the criteria for building a signaling server. There is a wide range of prices, capacity and storage option for the droplet’s size, and considering the need for just only a small web server to handle the traffic of a small division in LUAS, the cheapest droplet size, which is 5$/month, is chosen. Next, the London data center region is also picked since it is the nearest place to Finland. The authors choose to use the provided root password from DigitalOcean sent via email instead of generating new Secure Shell (SSH) key due to the simplicity. Last, the droplet is named meanapp and created with those above information.

Figure 10 shows the information of the droplet that the authors have created along with its Internet Protocol (IP) Address.

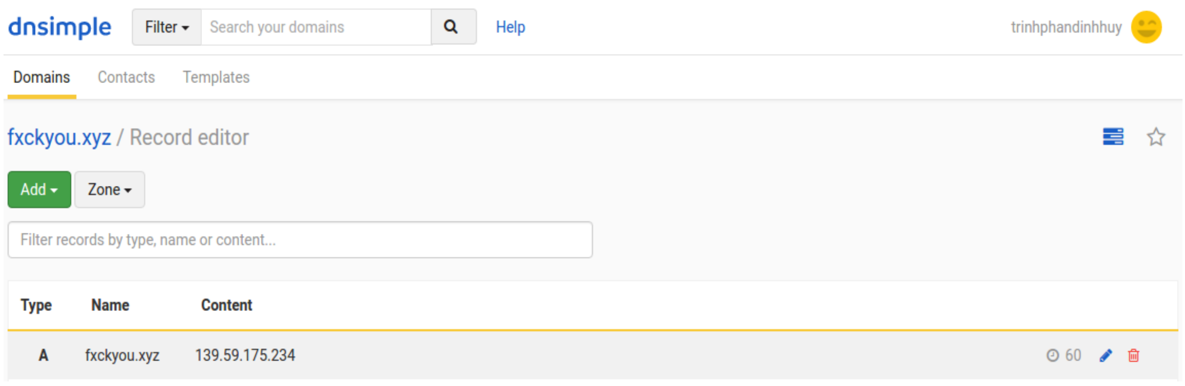


Name	IP Address	Created	
meanapp 512 MB / 20 GB Disk / LON1 - Ubuntu MEAN on 14.04	139.59.175.234	6 months ago	More

Figure 10 Droplet information on DigitalOcean

Creating SSL certificate for the web server

Before creating the signaling server, SSL certificate for the web server must be installed in order to secure the connection between clients and server since this is required for all data channels (Rescorla 2014, 16). There are many providers which can provide SSL certificates and Let’s Encrypt is one of the most popular certificate authority, which can enable Hypertext Transfer Protocol Secure (HTTPS) on websites or web apps without any cost. Moreover, this service provider was chosen because they provide powerful client software, which can automate the installation and issuance, work on many operating systems and ease the configuration for end-users. (Let’s Encrypt Getting Started 2016.) In order to implement the SSL certificate for the signaling server, there are two steps required. First of all, a registered domain should be owned and controlled to associate with the certificate. This domain should also point to the server’s IP address in order for Let’s Encrypt to validate the ownership of the domain it is issuing a certificate for. Figure 11 shows the creation of an A record for resolving the domain that the authors have purchased earlier to the public IP address of the droplet that the server is put on.



Type	Name	Content	
A	fxckyou.xyz	139.59.175.234	60

Figure 11 Creating an A record on DNSimple

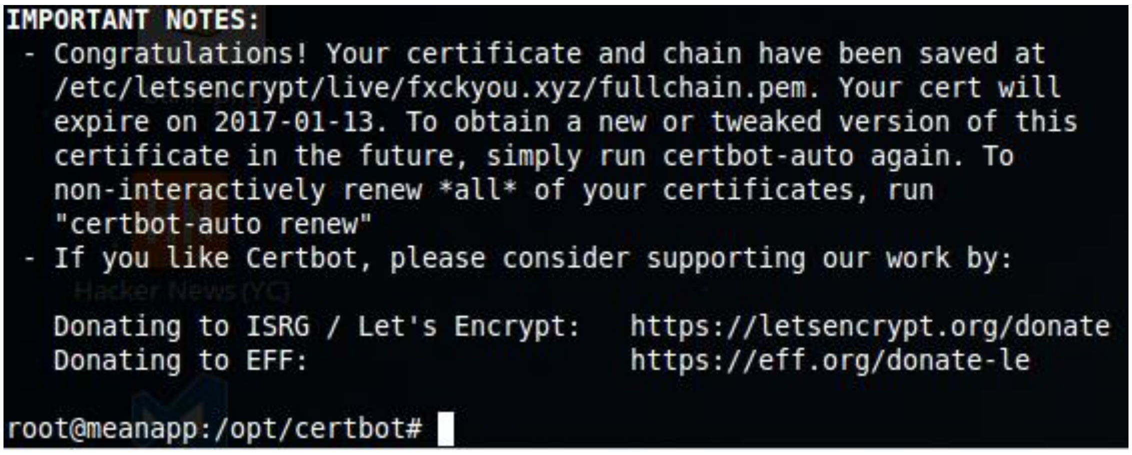
Next, the server needs to be configured in order to install Let’s Encrypt‘s client, which is Certbot for issuing the certificate. The server on DigitalOcean's droplet can be accessed via SSH with a simple command:

```
$ ssh user@ipaddress
```

The user in this situation is root, and the IP Address is the public IP provided by DigitalOcean for the droplet. After executing the command, there is a prompt asking for the user’s password. The password here is the one sent via email when the authors have created the droplet. Once the password is entered properly, a secure connection via SSH with the server is set and the server now can be configured through command line interface. There are numerous ways to install Let’s Encrypt client, Certbot, to the server depends on what kind of system the server is running. The authors’ choice is pulling the package from GitHub and utilizing the direct execution file of the software in order to avoid the complexity of installation. The following commands are executed line by line to first pulling the repository from GitHub to the server, then accessing the directory and running the execution file. The email using for registration is one of the authors' emails and the domain associated with the certificate is also provided.

```
$ git clone https://github.com/certbot/certbot.git
$ cd certbot/

$ ./certbot-auto certonly --email trinhphandinhhuy@gmail.com
-d fxckyou.xyz
```

After executing the last command, there is an interactive Graphical User Interface guiding through the process of obtaining and installing the certificate. After all, the authors get the congratulation prompt notifying that the SSL certificate has been issued for the server and saved at `/etc/letsencrypt/live/fxckyou.xyz/fullchain.pem`, which is showed in figure 12.



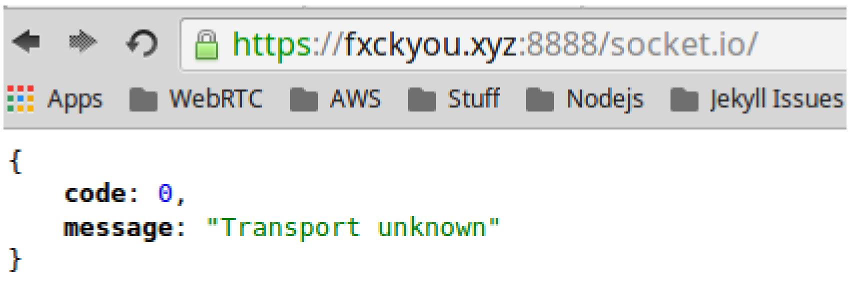
```
IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at
   /etc/letsencrypt/live/fxckyou.xyz/fullchain.pem. Your cert will
   expire on 2017-01-13. To obtain a new or tweaked version of this
   certificate in the future, simply run certbot-auto again. To
   non-interactively renew *all* of your certificates, run
   "certbot-auto renew"
 - If you like Certbot, please consider supporting our work by:

   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le

root@meanapp:/opt/certbot#
```

Figure 12 Prompt with SSL certificate issued successfully

Once the SSL certificated is obtained, the signaling server is ready to be built. The signalmaster, which is a simple server for WebRTC signaling from andyet, abstracts all the complexities of WebRTC API, and when combining with Express, the solution for developing a functional signaling server only requires developers a little effort, which providing the location of the private key and the certificate generated from Let’s Encrypt. After the server configuration is finished, the following command is executed inside the same directory in order to ensure the signaling server run in Production mode.

```
$ NODE_ENV=production node server.js
```

Once having executed, the signaling server is up and running properly when accessing the domain and the port indicated in the code, which is fxckyou.xyz and 8888, respectively. Figure 13 shows the results in the web browser when opening a web browser to the specified domain and port.



```
{
    code: 0,
    message: "Transport unknown"
}
```

Figure 13 Result when accessing https://fxckyou.xyz:8888/socket.io/

Setting up the web application and workflow

MEAN application foundation

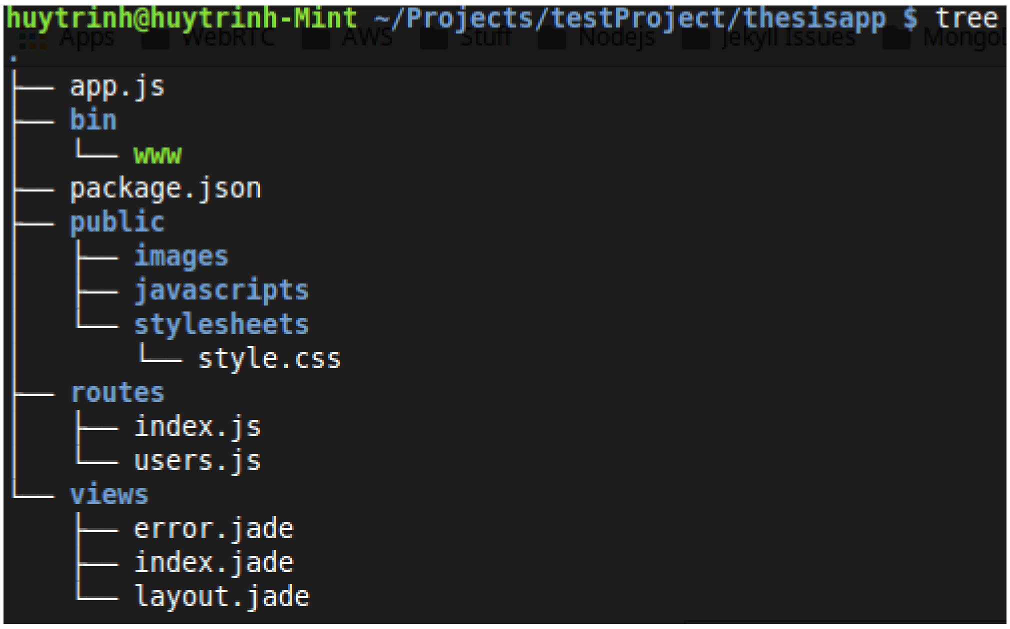
In order for demonstrating the workflow when implementing Docker, CI, and CD, first of all, a foundation of the application needs to be created. The web application actually is combined by two parts: the back-end server for serving static files and the front-end code for the actual application. This is because one of the WebRTC APIs, getUserMedia will not work by just simply open an HTML in the browser. With the purpose of involving MEAN stack into the web development, the authors have utilized Express for the back-end server. First, a boilerplate of Express has been created by using the following commands

```
$ express thesisapp
$ cd thesisapp
$ npm install
```

In Express main file, which is app.js, the following line of code is added in order to provide the functionality of serving static files, which contain the code of the actual application, in the client folder.

```
app.use(express.static(path.join(__dirname, '/../client')));
```

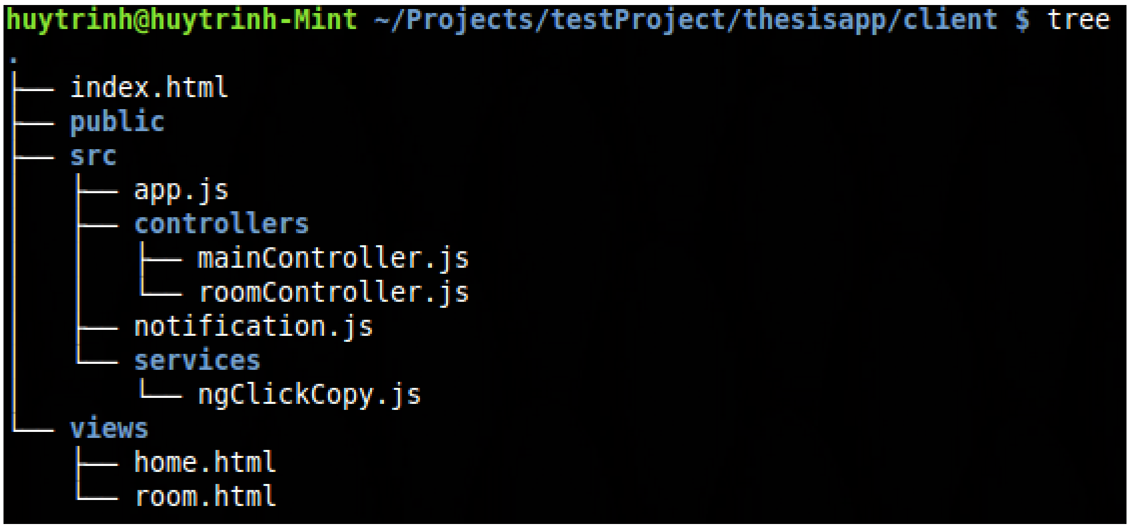
The following figure shows the directory structure of the server folder.



```
huytrinh@huytrinh-Mint ~/Projects/testProject/thesisapp $ tree
.
├── app.js
├── bin
│   └── www
├── package.json
├── public
│   ├── images
│   ├── javascripts
│   └── stylesheets
│       └── style.css
├── routes
│   ├── index.js
│   └── users.js
└── views
    ├── error.jade
    ├── index.jade
    └── layout.jade
```

Figure 14 Directory structure of the server folder

Next, the actual application has been implemented with AngularJS. There are two views for the user interface: the room creation view and the main view for video conferencing. Therefore, there are two controllers associating with two above views: mainController for handling the room creation view and roomController for handling the functionalities in the video conferencing. The application has two services which are ClickCopy and Notifications. The directory structure of the front-end is illustrated in figure 15.



```
huytrinh@huytrinh-Mint ~/Projects/testProject/thesisapp/client $ tree
.
├── index.html
├── public
├── src
│   ├── app.js
│   ├── controllers
│   │   ├── mainController.js
│   │   └── roomController.js
│   ├── notification.js
│   └── services
│       └── ngClickCopy.js
└── views
    ├── home.html
    └── room.html
```

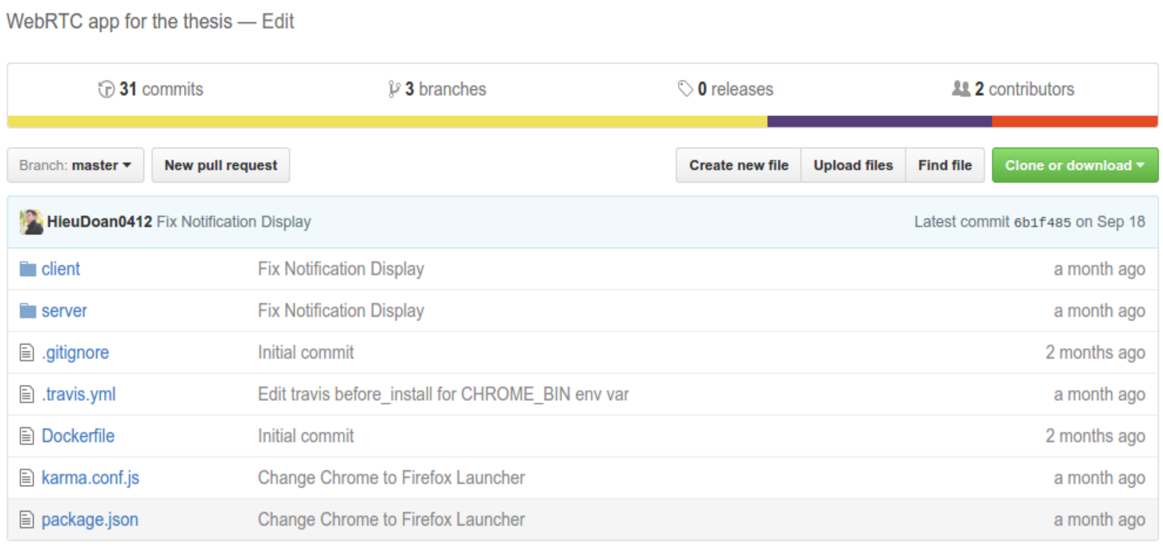
Figure 15 Directory structure of the client folder

Creating GitHub repository

Once the application has been set up in one of the authors' computers, which is localhost, a repository has been also created on GitHub in order for storing the whole project in a safe place, enabling collaboration between two developers and creating the first component in the automated development workflow. The following commands has been executed line by line in order to initiate a Git repository in localhost, add all the directories and files within the project for committing, perform the first commit, add the origin remote in the central server and push the first commit to the master branch in the created repository on GitHub.

```
$ git init
$ git add -A
$ git commit -m "first commit"
$ git remote add origin
https://github.com/trinhphandinhhuy/thesisapp.git
$ git push -u origin master
```

Figure 16 shows the GitHub repository for the application. There are several files which have not been described yet since they are created for other purposes which will be defined later in this chapter. “2 contributors” icon indicates there are two developers in this project and “3 branches” icon indicates 3 branches, which are master, testing and design that the authors have created for different intention.



WebRTC app for the thesis — Edit

31 commits | **3** branches | **0** releases | **2** contributors

Branch: **master** | **New pull request** | **Create new file** | **Upload files** | **Find file** | **Clone or download**

HieuDoan0412 Fix Notification Display — Latest commit 6b1f485 on Sep 18

client	Fix Notification Display	a month ago
server	Fix Notification Display	a month ago
.gitignore	Initial commit	2 months ago
.travis.yml	Edit travis before_install for CHROME_BIN env var	a month ago
Dockerfile	Initial commit	2 months ago
karma.conf.js	Change Chrome to Firefox Launcher	a month ago
package.json	Change Chrome to Firefox Launcher	a month ago

Figure 16 GitHub repository for the WebRTC application

Setting up Testing platform and TravisCI for CI

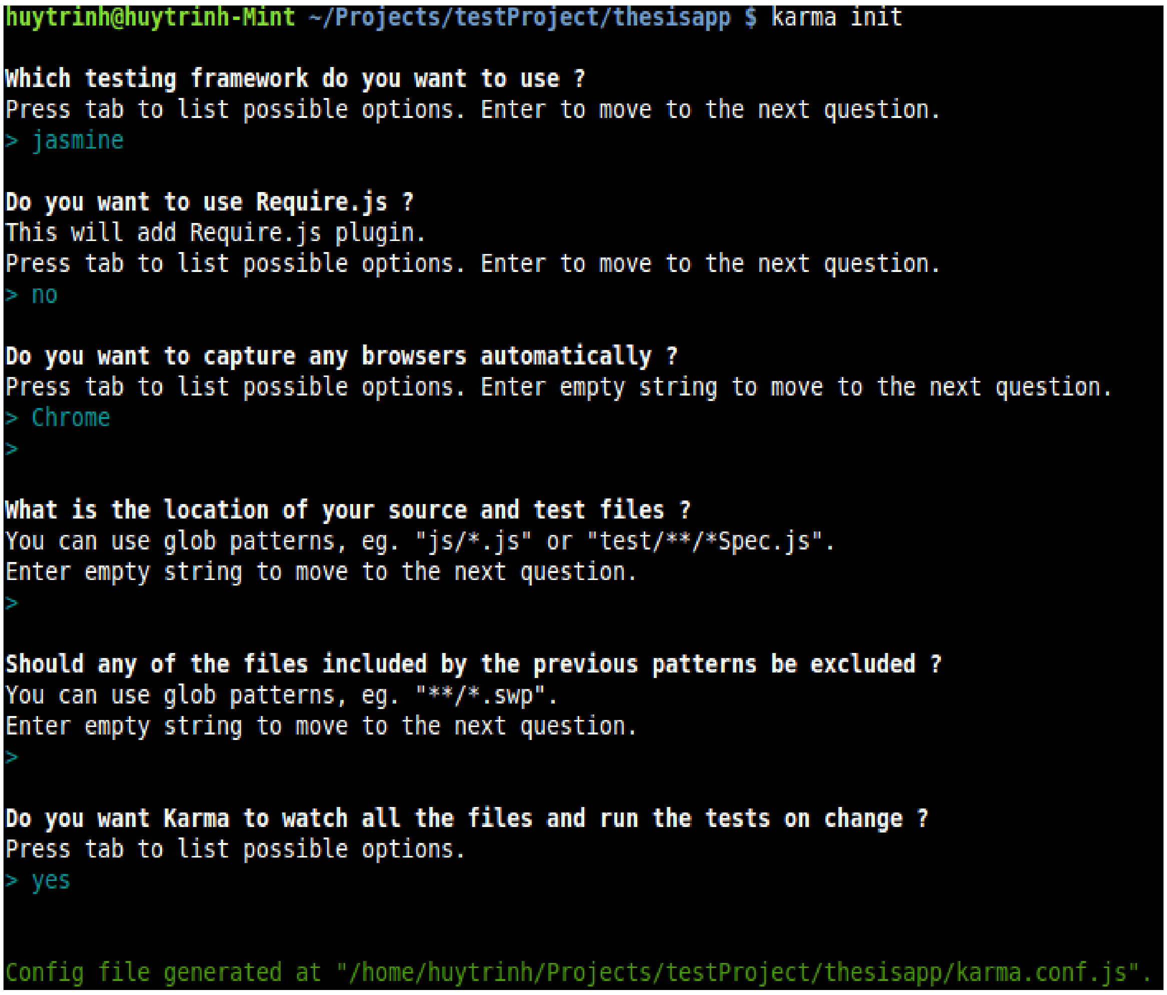
Two components for a testing platform that has been chosen for this project are Jasmine and Karma since the authors have had experience with both of them in previous projects. Jasmine is a behavior-driven development framework for JavaScript code testing (Jasmine 2016). Karma, which runs on Node.js, is a JavaScript test runner from AngularJS team (Bielski 2016). The details including comprehensive definition and implementation of Karma and Jasmine is not defined in this study because the purpose of this subchapter is showing how two testing components fit into this phase of the workflow, not studying about any particular testing frameworks since they can be replaced easily with numerous other ones.

In order to implement Jasmine and Karma into the project, they first have been installed along with their dependencies. The following commands have executed installation of Karma Command Line Interface for configuring Karma and installation of packages needed to run Jasmine and Karma.

```
$ npm install -g karma-cli
```

```
$ npm install karma karma-jasmine jasmine-core karma-chrome-
launcher angular-mocks --save-dev
```

After installation, Jasmine and Karma can be configured using the simple command karma init. The following figure shows a series of questions prompting for helping setting up the Karma configuration file.



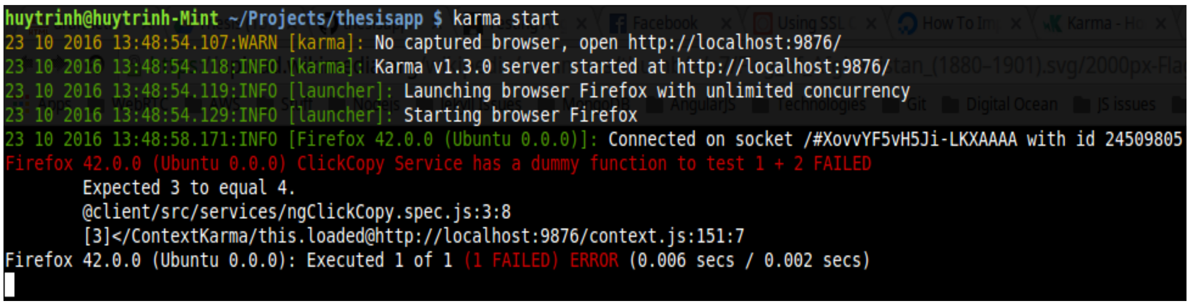
```
huytrinh@huytrinh-Mint ~/Projects/testProject/thesisapp $ karma init

Which testing framework do you want to use ?
Press tab to list possible options. Enter to move to the next question.
> jasmine

Do you want to use Require.js ?
This will add Require.js plugin.
Press tab to list possible options. Enter to move to the next question.
> no

Do you want to capture any browsers automatically ?
Press tab to list possible options. Enter empty string to move to the next question.
> Chrome
>

What is the location of your source and test files ?
You can use glob patterns, eg. "js/*.js" or "test/**/*Spec.js".
Enter empty string to move to the next question.
>

Should any of the files included by the previous patterns be excluded ?
You can use glob patterns, eg. "**/*.swp".
Enter empty string to move to the next question.
>

Do you want Karma to watch all the files and run the tests on change ?
Press tab to list possible options.
> yes


Config file generated at "/home/huytrinh/Projects/testProject/thesisapp/karma.conf.js".
```

Figure 17 Question prompting to configure Karma

After all the configuration has been done, a simple test case has been included in a test file to see whether the whole testing platform works properly, which is showed below

```
describe('ClickCopy Service', function () {
    it('has a dummy function to test 1 + 2', function () {
       expect(1 + 2).toEqual(4);
    });}
```

This sample code has been placed in the ClickCopy service, which will be developed later, is only for illustration of the test structure. This test contains a function to check the result of 1 + 2 is equal to 4, and since 1 + 2 = 3, this test definitely fails. In order to see the result of this test whether it passes or fails, first of all, this test file has been included in the karma configuration file. Next, in the terminal, the command `karma start` has been executed.



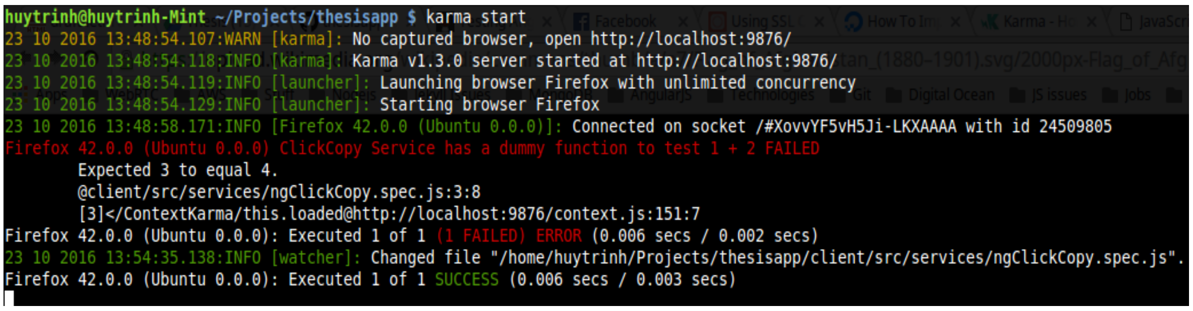
```
huytrinh@huytrinh-Mint ~/Projects/thesisapp $ karma start
23 10 2016 13:48:54.107:WARN [karma]: No captured browser, open http://localhost:9876/
23 10 2016 13:48:54.118:INFO [karma]: Karma v1.3.0 server started at http://localhost:9876/
23 10 2016 13:48:54.119:INFO [launcher]: Launching browser Firefox with unlimited concurrency
23 10 2016 13:48:54.129:INFO [launcher]: Starting browser Firefox
23 10 2016 13:48:58.171:INFO [Firefox 42.0.0 (Ubuntu 0.0.0)]: Connected on socket /#XovvYF5vH5Ji-LKXAAAA with id 24509805
Firefox 42.0.0 (Ubuntu 0.0.0) ClickCopy Service has a dummy function to test 1 + 2 FAILED
        Expected 3 to equal 4.
        @client/src/services/ngClickCopy.spec.js:3:8
        [3]</ContextKarma/this.loaded@http://localhost:9876/context.js:151:7
Firefox 42.0.0 (Ubuntu 0.0.0): Executed 1 of 1 (1 FAILED) ERROR (0.006 secs / 0.002 secs)
```

Figure 18 The result when starting Karma with the failed test case

The result shows that the test has failed. In order to make the test pass, the result in toEqual() function has been corrected as follows:

```
expect(1 + 2).toEqual(3);
```

The nature of Karma allows the test automatically starts again without the need to execute the command karma start. The test result has shown the success message in the following figure.



```
huytrinh@huytrinh-Mint ~/Projects/thesisapp $ karma start
23 10 2016 13:48:54.107:WARN [karma]: No captured browser, open http://localhost:9876/
23 10 2016 13:48:54.118:INFO [karma]: Karma v1.3.0 server started at http://localhost:9876/
23 10 2016 13:48:54.119:INFO [launcher]: Launching browser Firefox with unlimited concurrency
23 10 2016 13:48:54.129:INFO [launcher]: Starting browser Firefox
23 10 2016 13:48:58.171:INFO [Firefox 42.0.0 (Ubuntu 0.0.0)]: Connected on socket /#XovvYF5vH5Ji-LKXAAAA with id 24509805
Firefox 42.0.0 (Ubuntu 0.0.0) ClickCopy Service has a dummy function to test 1 + 2 FAILED
        Expected 3 to equal 4.
        @client/src/services/ngClickCopy.spec.js:3:8
        [3]</ContextKarma/this.loaded@http://localhost:9876/context.js:151:7
Firefox 42.0.0 (Ubuntu 0.0.0): Executed 1 of 1 (1 FAILED) ERROR (0.006 secs / 0.002 secs)
23 10 2016 13:54:35.138:INFO [watcher]: Changed file "/home/huytrinh/Projects/thesisapp/client/src/services/ngClickCopy.spec.js".
Firefox 42.0.0 (Ubuntu 0.0.0): Executed 1 of 1 SUCCESS (0.006 secs / 0.003 secs)
```

Figure 19 The result when starting Karma with the correct test case

Finally, the foundation of the testing platform in the local machine has been completely set up. Next step is to involve one of the CI services for running the test code whenever it is pushed to GitHub repository. TravisCI is a popular CI service with can be integrated with both Karma and GitHub to perform this task. There are two tasks needed to be done in order to integrate TravisCI into the workflow. The first task is to enable TravisCI for the repository with the test needs to be built in TravisCI dashboard. The second one is to add a .travis.yml file to the repository to indicate what tasks the developer wants Travis to do. (TravisCI 2016)

The configuration in .travis.yml is quite descriptive. The language and version for this project are Node.js 6.0. There are several commands needed to be executed before the installation and script test for the environment variables and dependency installation. Next, it indicates TravisCI to run the test runner Karma with the provided configuration file. TravisCI also is noticed that it only builds the test if the code is pushed to branch master of the GitHub repository. The after_success option will execute the command after the test passed. The CI part for this project has been integrated successfully and more details about how this part works will be described later in the demonstration.

Setting up Docker Automated Build

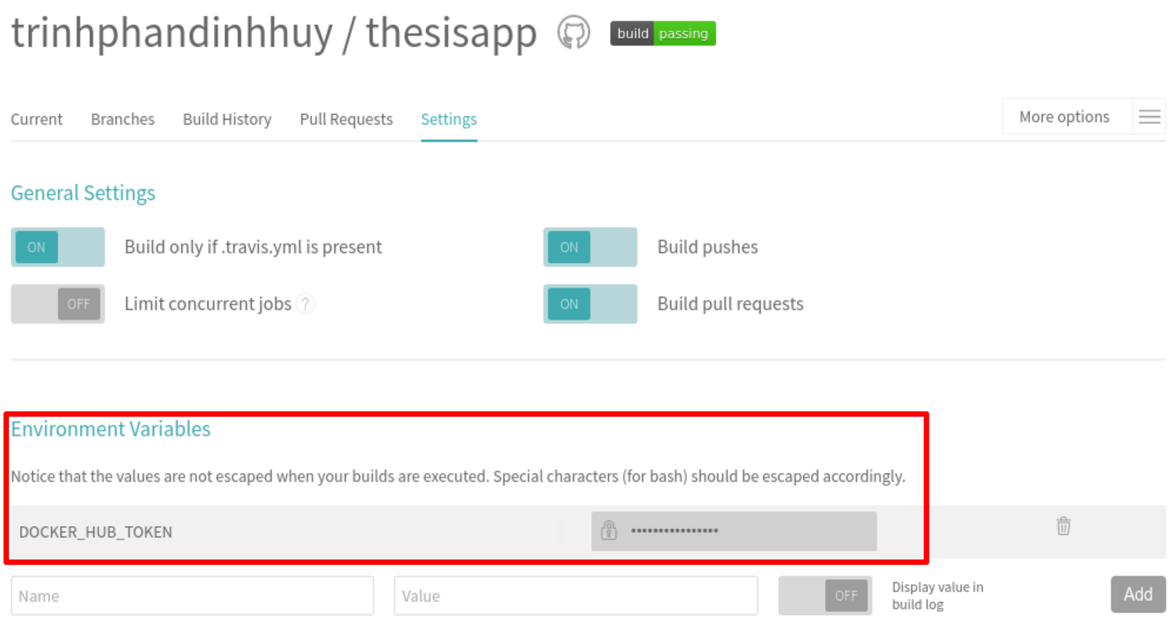
The next component in the automated development workflow is Docker Automated Build. This feature of Docker Hub can automatically build a specified image based on the provided Dockerfile in the project when receiving a remote build trigger from a CI, which is TravisCI in this case.

The Dockerfile starts with the image is built from the latest version of Node.js, which is a base image. Next, a folder is created and all the directories and files in the project are copied to this folder. The npm install command then is executed to ensure all the dependencies for the project are properly installed. Lastly, port 3000 is exposed for connections and the command for running the application is executed. After the Dockerfile is specified in the project, the Automated Build is ready to be implemented. First of all, an account on Docker Hub has been created and linked to a hosted repository provider, which is GitHub in this case. In the Docker Hub Dashboard, the Select dropdown has been pressed and the Create Automated Build field has been chosen. The system has shown the list of repositories of the GitHub account associating with Docker Hub account in the first place. The thesisapp project has been picked and the system has continued to display the Create Automated Build dialog. This dialog has shown several options involving repository name, visibility, short description, code branches, and tags. All the options have been then left as defaults and the Create button has been pressed in order to create the repository on Docker Hub.

After the Automated Build for the project has been created, the first image is built and the detail of the build is shown on the Build Details page. Although there are numerous configurations for existing build setting, the authors only have focused on the remote build triggering and webhooks. The remote build trigger can be found in the Build Triggers section in Build Details page. After pressing the Activate Triggers button, a trigger token and a trigger URL have been supplied. This trigger token and URL can be used in a CI for triggering automated image build. As mentioned earlier, after the test build succeeds TravisCI provides the functionality of executing a command, which can be used to perform the curl command for sending a POST request to Docker Hub for triggering the image build. The curl command with after_success option has been added at the end of the .travis.yml file as follows.

```
after_success:
  - |
        curl -H "Content-Type: application/json" --data
'{"build": true}' -X POST
https://registry.hub.docker.com/u/huytrinh/thesisapp/trigger
/$DOCKER_HUB_TOKEN/
```

In the command above, the $DOCKER_HUB_TOKEN variable has been substituted for the actual token for security reason. This variable can be specified in the Environment Variables section the TravisCI, which is shown in figure 20.



trinhphandinhhuy / thesisapp

General Settings

Build only if .travis.yml is present: ON

Limit concurrent jobs: OFF

Build pushes: ON

Build pull requests: ON

Environment Variables

Notice that the values are not escaped when your builds are executed. Special characters (for bash) should be escaped accordingly.

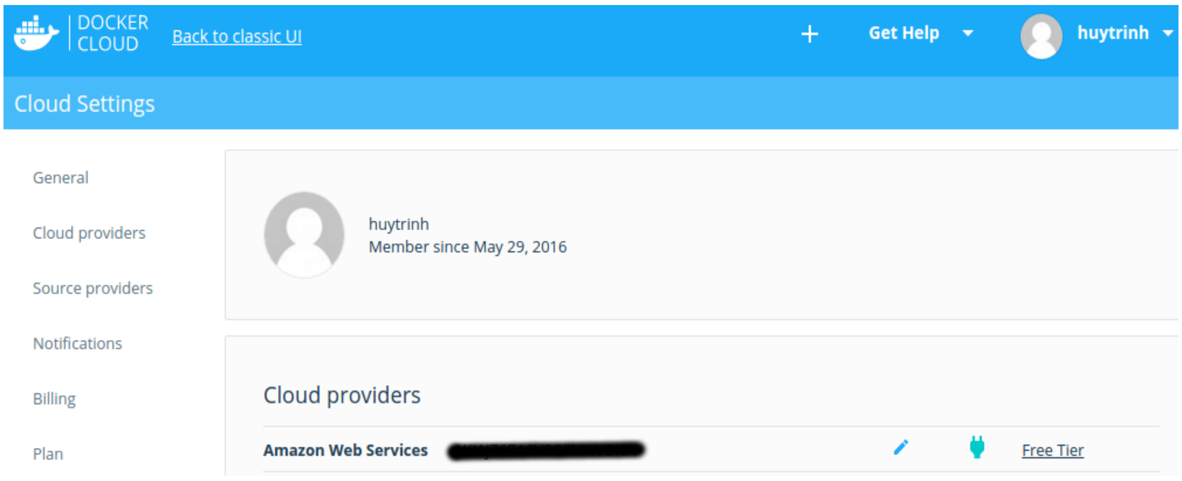
DOCKER_HUB_TOKEN ••••••••••••••••

Figure 20 Specify environment variables in TravisCI

Once the .travis.yml file has been fully configured with the token for triggering image build, the link between TravisCI and DockerHub has been connected properly.

Setting up AWS for linking to Docker Cloud

Before moving on to the last piece of the automated workflow, which is Docker Cloud, AWS IAM should have been configured, which involving creating the policy and attaching it to a new role allowing Docker Cloud to accommodate and manage resources. The detail of AWS IAM configurations is not covered in this subchapter since it is available on Docker Cloud Documentation, which can be found at https://docs.docker.com/docker-cloud/infrastructure/link-aws/ . The authors have used this article as a reference for configuring AWS IAMS as well and the outcome of this tutorial is the Role Amazon Resource Name (ARN) of the created role, which can be used to connect AWS with Docker Cloud. The installation continues with the Cloud Settings page of Docker Cloud, the system has provided the list of cloud infrastructure providers associating with Docker. Next, the plug icon of AWS field has been clicked to connect this provider with Docker Cloud. A dialog then has been prompted asking the Role ARN and after this name has bene provided and the Save button has been clicked, the connection between AWS and Docker Cloud has been established and Docker Cloud can use AWS resources to create node clusters and nodes, which will be described later. Figure 21 shows the Cloud Settings page with AWS connection.



huytrinh
Member since May 29, 2016

Cloud providers

Amazon Web Services [redacted] Free Tier

Figure 21 Joining AWS with Docker Cloud

Configuring Docker Cloud for Continuous Delivery and Deployment

In order for Docker Cloud to build application images from Docker Hub and automate deploying these images to AWS, there are several tasks needed to be done. First of all, a node cluster and at least one node have been created to be ready for the deployment. This task has created an AWS EC2 instance according to the provided options from Docker Cloud and all the services needed for the application is deployed in this instance. There is a clear instruction in Docker Cloud documentation mentioning creating this first node cluster and node, which the authors do not discuss in this study. The article can be found at https://docs.docker.com/docker-cloud/getting-started/your_first_node/ . Secondly, a service has been needed to be deployed in order to create a container that runs the web application. A service can be easily created and deployed by a helpful wizard which guiding developers through several configuration steps. However, in this case, the full web application has needed more than 1 service to run properly. In order to run in HTTPS mode, which is critical for browsers to access camera and microphone as mentioned earlier in this chapter, the web application has to run along with a proxy server which supporting SSL Termination. Therefore, at least 2 services should run simultaneously and this could be done easier than using the wizard with Docker Cloud stack. A stack is a convenient way to deploy automatically a collection of services which are linked to each other, without the need of separate service definition (Docker 2016). The authors have used a stack file, which is instruction for creating multiple services, to create one service for the proxy server and another service for running the web application. In order to create a stack file, the Create button in the Stacks page has been clicked and a text area for writing service create instruction has been then displayed.

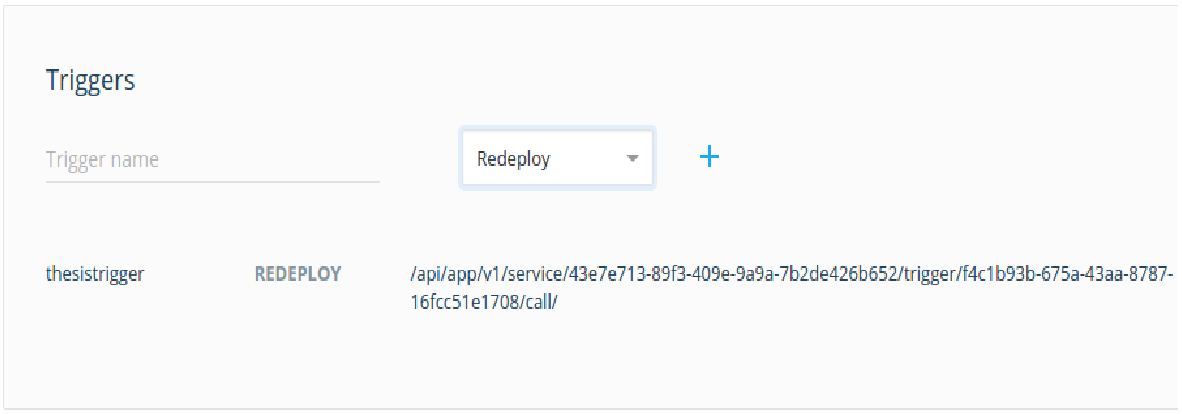
The instruction in the stack file can be described as follows:

- Create a lb service which stands for load balancing

  - The environment option contains the SSL certificate that is obtained earlier for the application.
  - The image for this service is HAProxy, which is a load balancer that supports SSL Termination
  - This service will link to the web service to provide SSL Termination function.
  - Port 80 and 443, which are used by HTTP and HTTPS respectively are exposed.
- Create a web service
  - The image for this service is the web application image that already is built in Docker Hub earlier.
  - When triggered, this service will be re-deployed automatically.

Once the Create & Deploy button has been clicked, the whole stack containing two services has been deployed and the web application can have been accessed with the domain associating with the SSL certificate, which is https://thesisapp.ontheroof.top/.

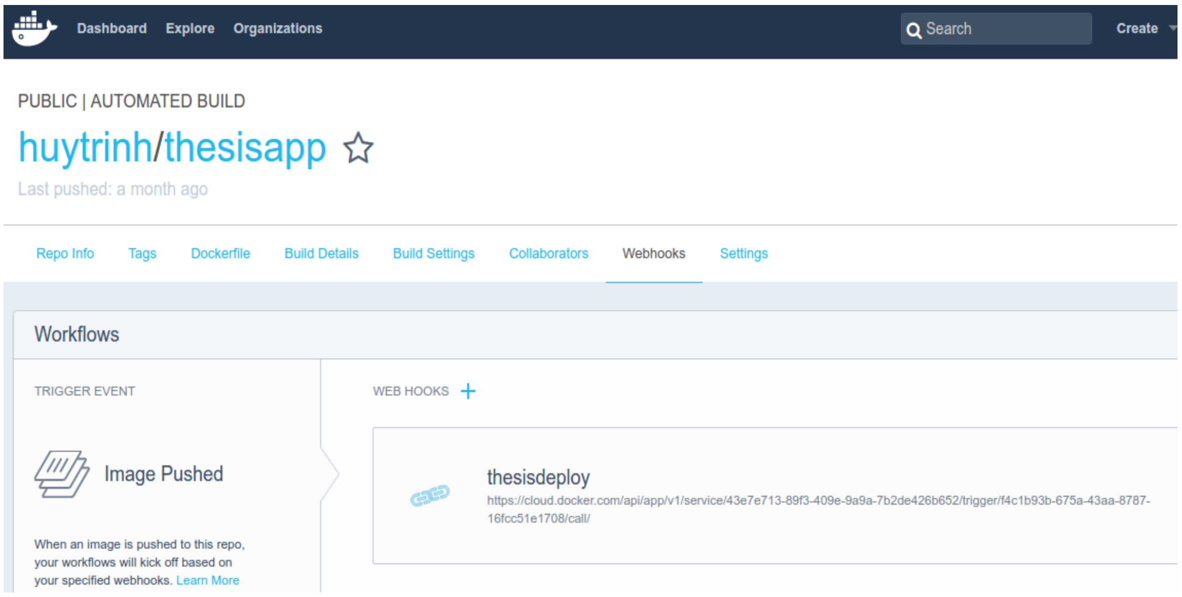
Lastly, the redeploy trigger in the web service has been activated to enable the redeploy function when a new image from Docker Hub is built. Figure 22 shows the section where this trigger can be activated.



thesistrigger REDEPLOY /api/app/v1/service/43e7e713-89f3-409e-9a9a-7b2de426b652/trigger/f4c1b93b-675a-43aa-8787-16fcc51e1708/call/

Figure 22 Redeploy trigger activation

After activating the redeploy function and receiving the webhook URL from Docker Cloud, the automated workflow has been completed with putting this URL in the WebHook service of Docker Hub, which can be seen in figure 23.



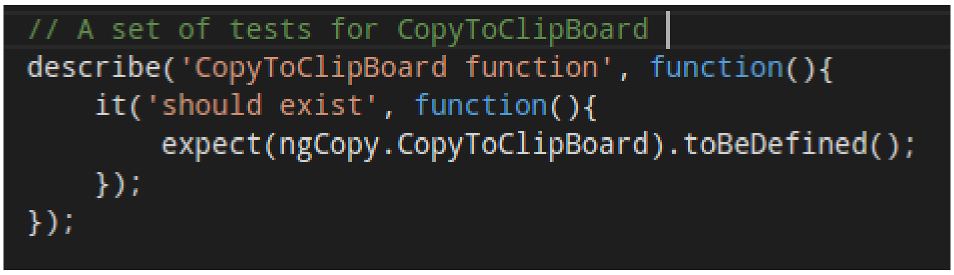
When an image is pushed to this repo, your workflows will kick off based on your specified webhooks. Learn More

thesisdeploy
https://cloud.docker.com/api/app/v1/service/43e7e713-89f3-409e-9a9a-7b2de426b652/trigger/f4c1b93b-675a-43aa-8787-16fcc51e1708/call/

Figure 23 Adding trigger URL to Docker Hub Webhooks page

5.7.2 Development Workflow Demonstration

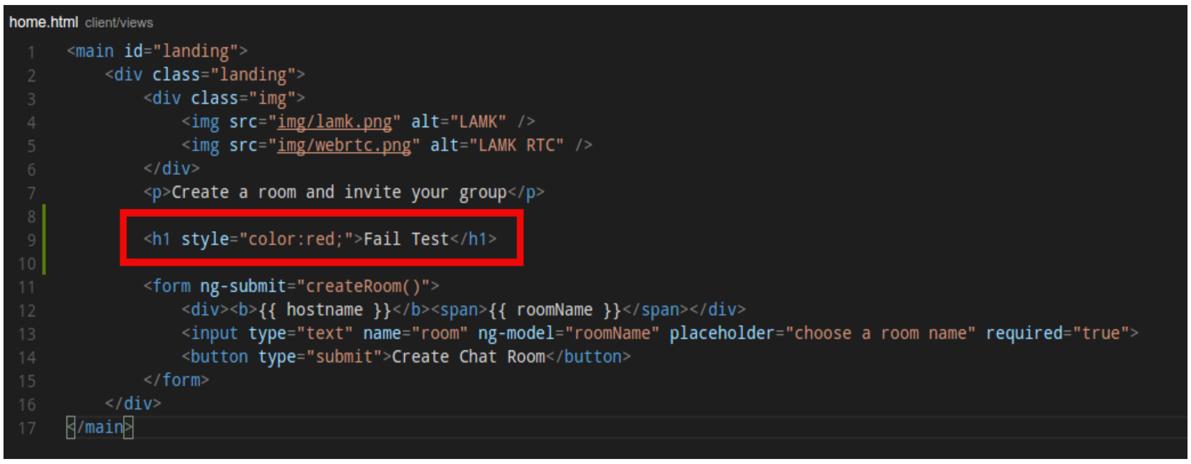
This subchapter demonstrates how all the components described above connect together to automate the software development process, from testing the code pushed on GitHub to deploying the containerized application to AWS, which is one of the crucial results of this study. This has started with developing a function, which is CopyToClipBoard - a chosen function for this demonstration, and writing a test case for checking whether this function exists. This test case is extremely simple compared to other test cases of this project, but it is enough for the testing purpose. The CopyToClipBoard is a small function utilized to copy the link of a created conference room to the clipboard in order for sharing to other peers to connect. This function first has been deleted or commented out of ngClickCopy service to make the test, which is shown in figure 24, fails.



```
// A set of tests for CopyToClipBoard
describe('CopyToClipBoard function', function(){
    it('should exist', function(){
        expect(ngCopy.CopyToClipBoard).toBeDefined();
    });
});
```

Figure 24 Test case to check the existence of CopyToClipBoard function

In order to prove the benefit of CI system, an H1 tag has been included in the home.html file, which is the HTML file representing the home view.



home.html client/views

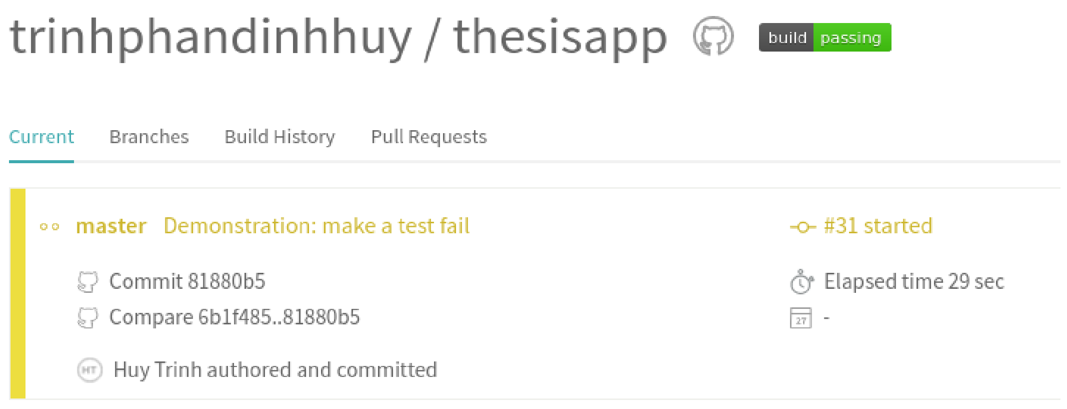
```
<main id="landing">
    <div class="landing">
        <div class="img">
            <img src="img/lamk.png" alt="LAMK" />
            <img src="img/webrtc.png" alt="LAMK RTC" />
        </div>
        <p>Create a room and invite your group</p>

        <h1 style="color:red;">Fail Test</h1>

        <form ng-submit="createRoom()">
            <div><b>{{ hostname }}</b><span>{{ roomName }}</span></div>
            <input type="text" name="room" ng-model="roomName" placeholder="choose a room name" required="true">
            <button type="submit">Create Chat Room</button>
        </form>
    </div>
</main>
```

Figure 25 H1 tag is added for verification

Once the test and the modification have been completed, the code has been pushed to the master branch on GitHub repository with the following commands.

```
$ git add –A

$ git commit –m “Demonstration: make a test fail”
$ git push origin master
```

TravisCI has got the test build trigger from GitHub and has begun to build the test. The yellow color with 2 small circle animation indicates that the test is being built.



trinhphandinhhuy / thesisapp build passing

Current Branches Build History Pull Requests

master Demonstration: make a test fail #31 started

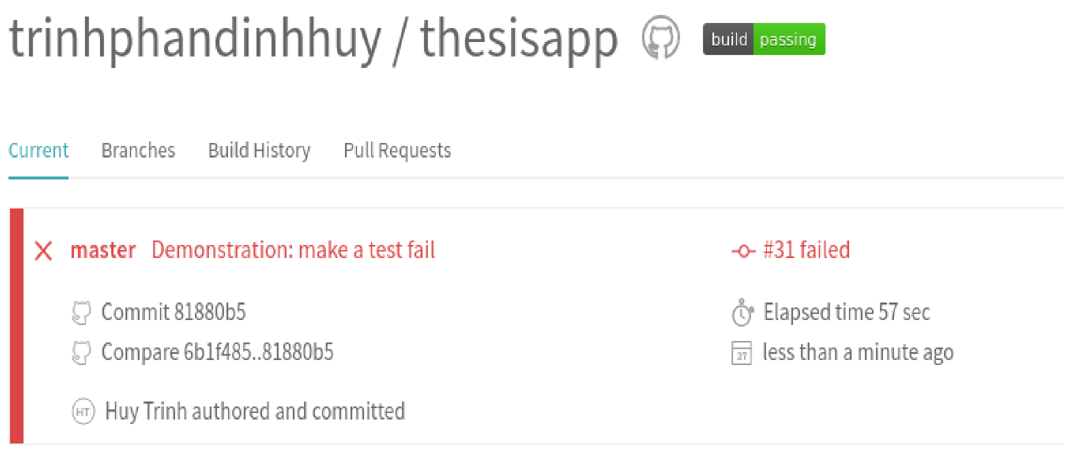
Commit 81880b5 Elapsed time 29 sec

Compare 6b1f485..81880b5 -

Huy Trinh authored and committed

Figure 26 Test is being built in TravisCI

After a few seconds to a minute, TravisCI has completed the build with the results showing that the build has been broken due to the failure of the test.



trinhphandinhhuy / thesisapp build passing

Current Branches Build History Pull Requests

master Demonstration: make a test fail #31 failed

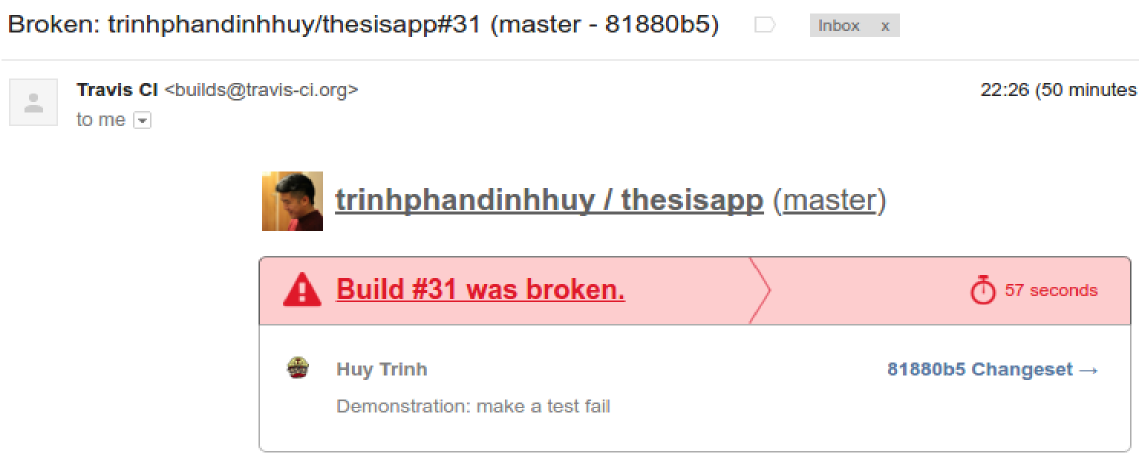
Commit 81880b5 Elapsed time 57 sec

Compare 6b1f485..81880b5 less than a minute ago

Huy Trinh authored and committed

Figure 27 Test fails in TravisCI

An email with more details also has been sent to the developer’s inbox for notifying the result of the test build, which is a remarkable feature of TravisCI.


Broken: trinhphandinhhuy/thesisapp#31 (master - 81880b5)

Inbox x

Travis CI <builds@travis-ci.org>
to me

22:26 (50 minutes

trinhphandinhhuy / thesisapp (master)

Build #31 was broken.

57 seconds

Huy Trinh

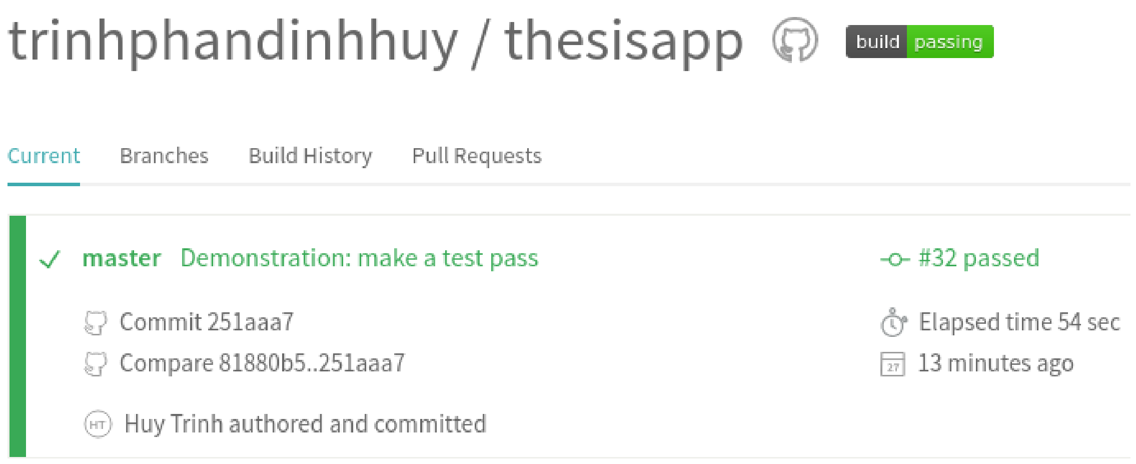
Demonstration: make a test fail

81880b5 Changeset →

Figure 28 Email sent for build notification

In order to inspect that Docker has not been triggered to build an image from GitHub because of the failed test, the Build Details of Docker Hub has been examined and there has been no image building indeed. Finally, the web application has been checked to see whether it shows the H1 text included in the failed version of the application. Figure shows the front page of the web application with no sight of the H1 text, which proves that Docker does not build a new image when the test fails, and the working version of the web application has been still preserved, which means that users can still be able to use the software normally with the CopyToClipboard and other functions.

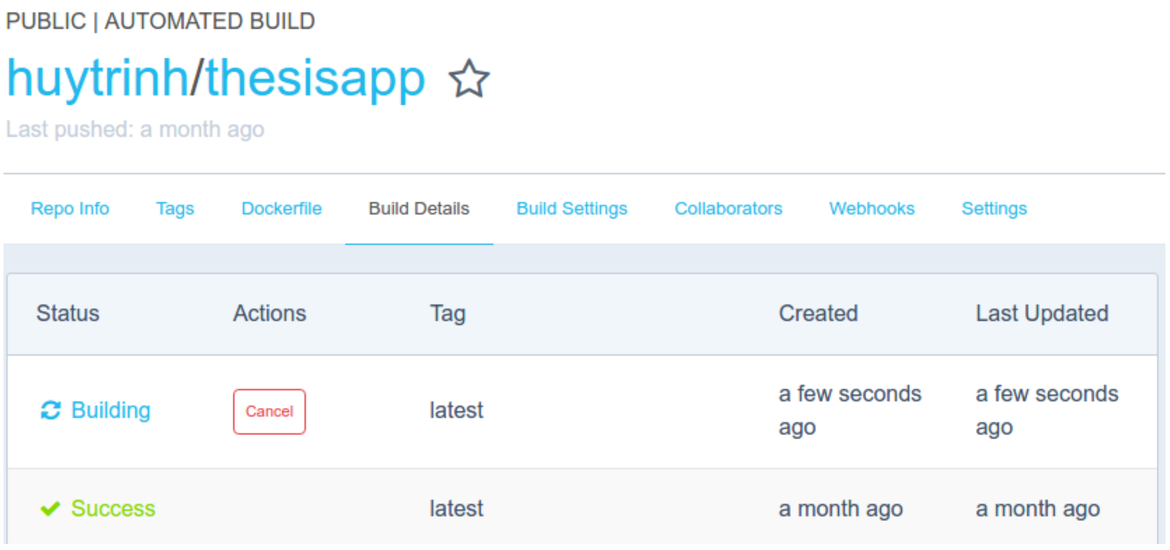
In order to make the test passes, the CopyToClipboard function has been added again and instead of placing a big, red ‘Fail Test’ text on the front page, the content in H1 tag has been replaced with ‘Pass Test’ with green color. The code has been again pushed to GitHub repository, but this time, the test has passed. Figure 29 shows the result of TravisCI for this test build.


trinhphandinhhuy / thesisapp build passing

✓ master Demonstration: make a test pass -o- #32 passed
Commit 251aaa7
Compare 81880b5..251aaa7
Elapsed time 54 sec
13 minutes ago
Huy Trinh authored and committed

Figure 29 Test passes in TravisCI

Because the test now has passed, the after_success option in the .travis.yml file has been called and has executed the POST request to trigger Docker Hub to build the application image from GitHub repository. Figure 30 shows the build details of this image a few seconds after getting a passed test from TravisCI.


PUBLIC | AUTOMATED BUILD
huytrinh/thesisapp
Last pushed: a month ago

Status	Actions	Tag	Created	Last Updated
Building	Cancel	latest	a few seconds ago	a few seconds ago
✓ Success		latest	a month ago	a month ago

Figure 30 Image building in Docker Hub triggered by TravisCI

After a few minutes, the image build has been successful and Docker Hub has been able to operate Webhook to trigger Docker Cloud for redeploying this image to AWS. Figure 31 indicates that the web service containing the image in Docker Cloud has been being redeployed with a blinking small gray square.

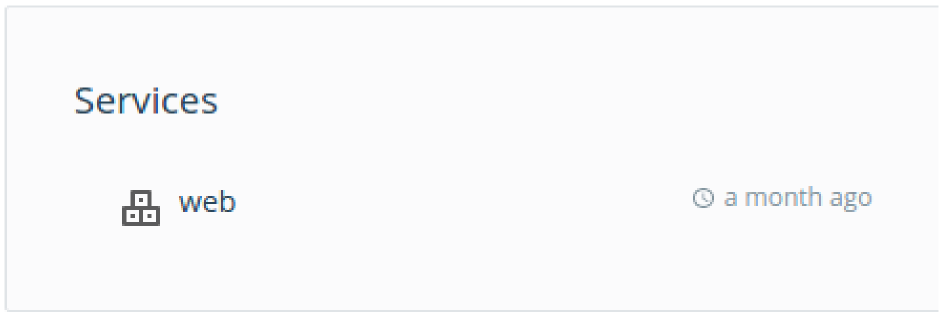


Figure 31 Web service is being redeployed

And figure 32 indicates that the web service has been redeployed favorably with a static small blue square.

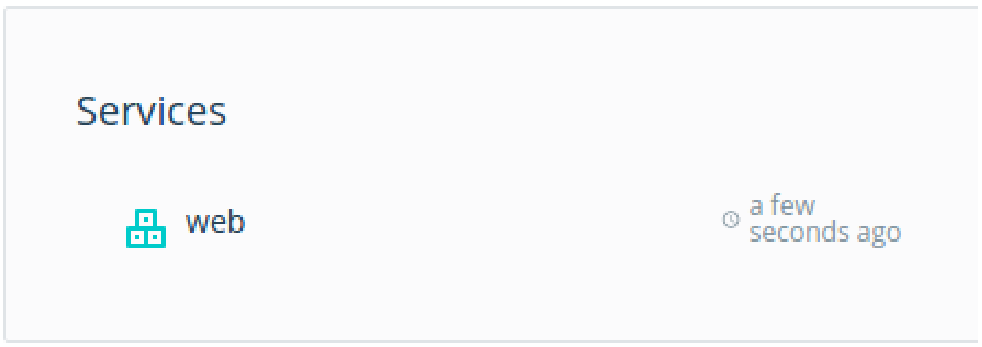


Figure 32 Web service is redeployed successfully

Lastly, a new green text with the phrase ‘Pass Test’ has been shown on the front page of the web application, which is displayed in figure 33, proves that the working code developed on the local machine has been successfully deployed in the host.

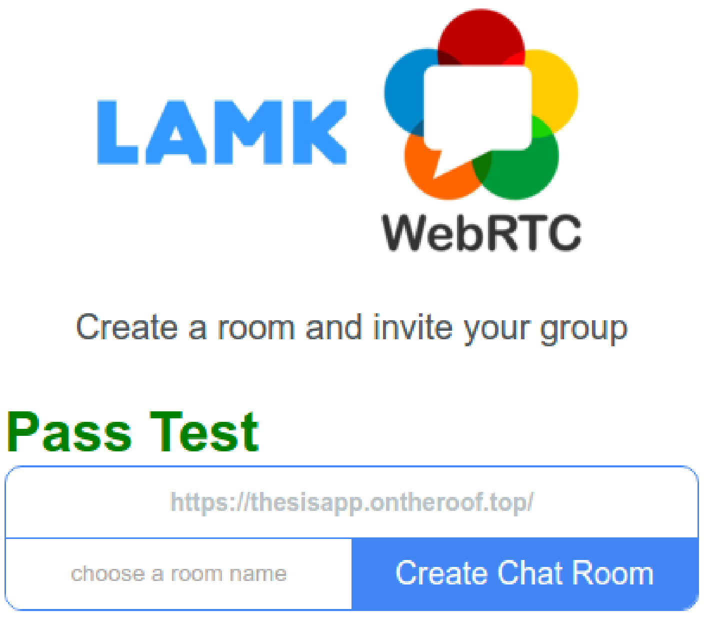
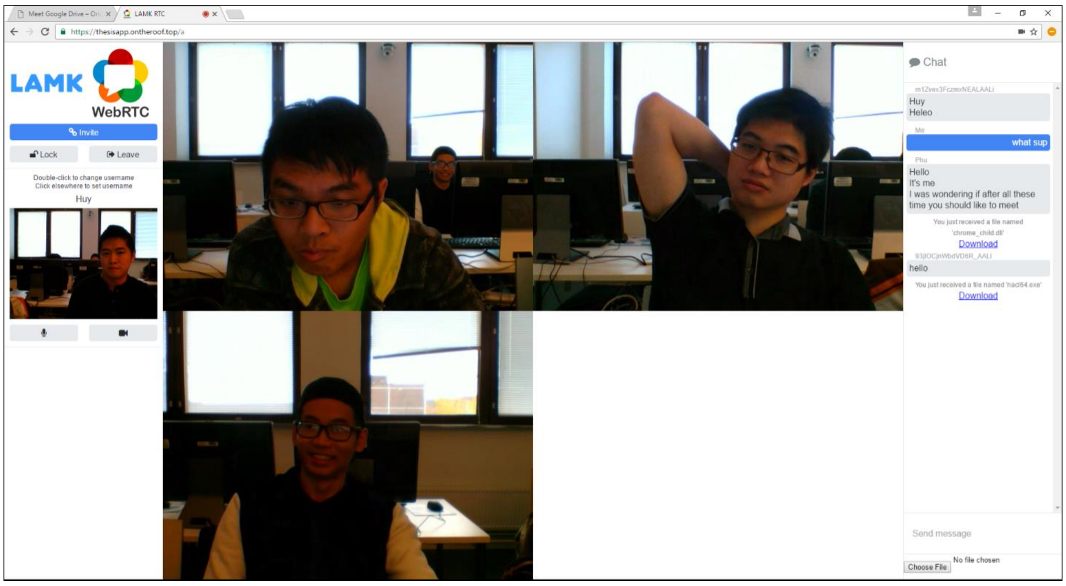

LAMK
WebRTC
Create a room and invite your group
Pass Test
https://thesisapp.ontheroof.top/
choose a room name
Create Chat Room

Figure 33 Front page shows 'Pass Test' indicating the workflow with CI and CD works effectively

5.8 Result

Figure 34 shows a demo video conferencing among 4 participants utilized the WebRTC application developed by the authors.



m12vex3FczmxNEALAALi
Huy
Heleo

Me
what sup

Phu
Hello
It's me
I was wondering if after all these time you should like to meet

You just received a file named 'chrome_child.dll'
Download

93jIOCjnWbdVD6R_AALI
hello

You just received a file named 'nacl64.exe'
Download

Figure 34 Video Conferencing Demo of the WebRTC application

The project was successful. The authors were able to meet all the specification of the software in the expected period of time. Moreover, the authors succeeded in implementing CI and CD along with Scrum. As a result, the authors identified some major improvements during the development process. These improvements are discussed and demonstrated further in the next chapter.

6 THE STUDY

This section provides the results of the study by comparing and evaluating the development process of the two artefacts.

6.1 Evaluation Criteria

Since both projects were small and developed by the same group of developers, it is justified to compare the two development processes. The comparison is between the Food n' Stuff application development process, which only applies Scrum methodologies, and the real-time communication application development process, which implements Scrum with the support of CI and CD. The evaluation is based on the three main aspects identified during the development process of the second application. The three aspects are: time, quality and portability of the application.

6.2 Comparison and Evaluation

6.2.1 Time

In the end of the WebRTC project, the authors agree that time is the most influential factor that CI and CD bring to Scrum. Since the system enables developers to submit their code daily and includes automated testing during each integration, it is quick and convenient for the developers to identify a bug or an error. In Food n' Stuff's development process, it has been common for the team to take a few days to identify a problem and then has spent another few days to fix them. In addition, the longer the time is, the greater the problem is. However, it is solved with the adaptation of CI. In the WebRTC project, by committing their work every day, the developers have been able to identify the bug immediately and deal with it soon before it creates other problems.

Then, after the application has passed all the tests, the new changes and features in the WebRTC project have been immediately deployed and available to deliver to the customer. Although the Food ‘N Stuff project is quite small, it has taken the stakeholders two weeks to be able to see the changes and give their feedbacks. Whereas it merely has taken one day in the case of the WebRTC project to carry out the exact same actions. Therefore, it is undeniable that shortening the time of delivery from two weeks to one day is a significant improvement that CD has brought to Scrum.

6.2.2 Quality

Thanks to the nature of CI and CD, the quality of the software is easily maintained and managed. The automated testing in the CI ensures that a defective software is not going to be deployed and delivered to the customer. In Food ‘N Stuff case, a bug has been sometimes identified by the stakeholders during the delivery of the software at the end of each sprint despite the thorough testing of the team. Although all the bugs have been minor and dealt with before the next delivery, it is unacceptable to let the clients use a defective version of the product despite all the adequate and better features. The case is obviously different in the WebRTC application. If the automated testing system, which is carried out daily, identifies a defect in the software, the CI and CD are going to keep the software in the safe state and not apply the changes at all. It ensures the users always receive a stable and sufficient version of the product.

6.2.3 Portability

Deployment is always one of the hardest parts to be dealt with in the software development process. This phase requires the developers to register for the hosting server, setup the virtual machines, install all the dependencies and environment runtime needed to run the application and keep it running more than 99% of the time to serve the customer. Obviously, performing all these tasks take a great deal of time and effort, let alone the developers have to have proficient knowledge of each hosting server configuration. Therefore, in the Food ‘N Stuff project, the team has had a very hard time trying to deal with the deployment phase. However, in the WebRTC project, these issues have been solved with the implementation of CD and Docker, which has been described earlier in chapter 5. While CD ensures that the application is in the flawless state and ready to be deployed, Docker brings the portability to the application. By applying them into the development process, the application has been able to be deployed anywhere and delivered to the customer at any desirable moment.

6.3 Benefits and Drawbacks during the Development Process

While developing the WebRTC application using the proposal solution which implements CI and CD, the authors have encountered several problems and discover some benefits that worth mentioning in this study.

The most troublesome situation of the development process is the coverage of the test cases. Due to the limited time and testing knowledge, the developers have been only able to write the most basic test cases for the application. These test cases have not ensured that the behavior of each function can be checked comprehensively. Therefore, the automated testing and building of the application have not been able to demonstrate the full ability of CI.

Although the cycle time, which is the time started from the creation of user story to the time that the customers have an adequate version of the software, has been quite short, the setup and installation have taken a huge amount of time.

Despite several disadvantages, the authors have gained many things during the development process. First, the authors have an opportunity to carry out a complete project, from planning, setting up to developing features. This experience plays a very important role for the authors' further study as well as profiling for the jobs. Second, the authors have learned lots of concepts and technologies besides CI, CD, and Docker.

This is considered as another achievement along with the results of this study.

7 CONCLUSIONS

7.1 Summary of the Thesis

The goal of this study was to show the benefits of CI and CD in Agile Software Development Methods, more specifically Scrum. To do this, the authors created a new artefact by applying Scrum with CI and CD. The new artefact was then compared with an application they had created earlier by applying only Scrum.

Chapter 3 provided detailed information about project management methodologies and the related technologies. The project management methods include Scrum, CI, CD and the technologies are Version Control, Docker, WebRTC, and MEAN stack. Chapter 4 and 5 described two projects and their development in order to point out differences in the processes based on three criteria: time, quality and the portability of the application.

Comparing the two development processes allows to answer the research question of this study: What are the benefits of implementing CI and CD in Scrum-related projects, and how can CI and CD improve these projects?

7.2 Answering Research Question

In short, integrating CI and CD into Scrum improves a software development process significantly. As already stated, the three criteria this affects are time, quality and portability of the application.

In terms of time, the developers are able to identify existing bugs immediately after the daily integration and deliver the product in just one day. During the previous project, the duration for identifying the bugs were more than three days and it was two weeks for the development team to deliver the software to the customer.

Next, the quality and stability of the application are maintained thanks to the automated testing process in the CI. Thanks to CI and CD, the software will never be deployed to the production environment if it is defective.

Lastly, Docker brings the portability to the application. The developers now are able to deploy the application to any hosting servers which support Docker and do not have to worry about the environment runtime setup or dependencies installation.

8 DISCUSSION

8.1 Limitations

The criteria used to compare the two processes is the first limitation of this study. Based on the development process of the Food ‘N Stuff application, only three criteria were set. Time, quality and portability may not be enough to compare two processes, and therefore the results of the comparison may not reflect all the differences between two processes. The improvements, therefore, of the new solution compared to the old one are not fully covered.

Secondly, since the WebRTC project was carried out by two IT students, and the authors’ report and tutorial writing skills are rather restricted, some steps in the development process may have accidentally been ignored or missed. The description of the CI and CD implementation then may not be precise and as comprehensive as the authors would want it to be.

Lastly, CI and CD were chosen only because of the authors’ work experience and because the proved to work sufficiently well in the WebRTC project. However, other systems may achieve the same goal with better performance and shorter time.

8.2 Reliability and Validity

Technologies change and innovate rapidly. At the time of writing this thesis, Docker Cloud was still in beta version and the MEAN stack was introduced just two years ago. In the future, this quick development can affect the implementation of CI and CD. The reliability and validity of this study are measured by the results of the evaluation process between two projects. In addition to the implementation of CI and CD, there might be some factors affecting the results such as the skills of the authors, the size of the project, the size of the team, the chosen technology stack and the time period that the project is carried out.

8.3 Further Research

Further study may involve a larger team with different positions to utilize other workflows in the development process for exploring more advantages as well as disadvantages of the proposal solution.

Further research should focus on Docker. In this study, the authors mention only the basic usage of Docker when integrating with CI and CD. There is a great deal of Docker benefits not only for CI and CD but also for the software development process in general that are needed to be explored.

REFERENCES

Written References

Barab, S. & Squire, K. 2004. Design-based Research: Putting a Stake in the Ground. The Journal of the Learning Sciences, 13(1), 1 – 14.

Duvall, P., Matyas, S. & Glover A. 2007. Continuous Integration: Improving Software Quality and Reducing Risk. Boston: Pearson Education, Inc.

Gallagher, S. 2015. Mastering Docker. Birmingham: Packt Publishing Ltd.

Haviv A.Q. 2014. MEAN Web Development. Birmingham: Packt Publishing Ltd.

Humble, J. & Farley D. 2011. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Boston: Pearson Education, Inc.

Kananen, J. 2013. Design Research as Thesis Research (Applied Action Research). Jyväskylä: JAMK University of Applied Sciences.

Loeliger, J. & McCullough, M. 2012. Version Control with Git, Second Edition. Sebastopol: O’Reilly Media, Inc.

Lorento, S. & Romano, S.P. 2014. Real-Time Communication with WebRTC. Sebastopol: O’Reilly Media, Inc.

McQuaid, M. 2015. Git in Practice. Greenwich: Manning Publications Co.

Mouat, A. 2016. Using Docker. Sebastopol: O’Reilly Media, Inc.

Peffers, K., Tuunanen, T., A.Rothenberger, M. & Chatterjee, S., 2008. A Design Science Research Methodology for Information Systems Research. Journal of Management Information Systems. Abingdon: M.E. Sharpe, Inc.

Pham, A & Pham P.V. 2011. Scrum in Action: Agile Software Project Management and Development. Boston: Course Technology

Rico D., Sayani H. & Sone S. 2009. The business Value of Agile Software Methods: Maximizing ROI with Just-in-Time Processes and Documentation. Florida: J. Ross Publishing

Ristic, D. 2015. Learning WebRTC: Develop interactive real-time communication applications with WebRTC. Birmingham: Packt Publishing Ltd.

Rogelberg, S. 2004. Handbook of Research Methods in Industrial and Organizational Psychology. Malden: Blackwell Publishing Ltd.

Santacroce, F. 2015. Git Essentials. Birmingham: Packt Publishing Ltd.

Sergiienko, A. 2014. WebRTC Blueprints: Develop your very own media applications and services using WebRTC. Birmingham: Packt Publishing Ltd.

Somasundaram, R. 2013. Git: Version Control for Everyone. Birmingham: Packt Publishing Ltd.

Tekeuchi, H. & Nonaka, I, 1986. The New New Product Development Game. Brighton: Harvard Business Review.

Westby, E.J.H. 2015. Git for Teams: A User-Centered Approach to Creating Efficient Workflows in Git. Sebastopol: O'Reilly Media, Inc.

Electronic References

&yet. SimpleWebRTC.js from &yet [accessed 1 Oct 2016]. Available at: https://simplewebrtc.com

&yet. Talky [accessed 1 Oct 2016]. Available at: https://about.talky.io

Alvestrand, H. 2011. Google release of WebRTC source code [email message]. Recipient public-webrtc@w3.org. Sent 1 June 2011 [referenced 8 October 2016.2016]

AWS 2016. About AWS [accessed 28 Oct 2016]. Available at: https://aws.amazon.com/about-aws/

AWS 2016. Amazon EC2 - Virtual Server Hosting [accessed 28 Oct 2016]. Available at: https://aws.amazon.com/ec2/?nc2=h_m1

AWS 2016. AWS Identity and Access Management (IAM) [accessed 28 Oct 2016]. Available at: https://aws.amazon.com/iam/?nc2=h_m1

AWS 2016. Cloud Products [accessed 28 Oct 2016]. Available at: https://aws.amazon.com/products/

Bielski, G. M. 2016. Karma - a Javascript Test Runner [accessed 30 Oct 2016]. Available at: http://www.methodsandtools.com/tools/karma.php

Blackstone, A., 2012. Sociological Inquiry Principles: Qualitative and Quantitative Methods. Washington: Flat World Education, Inc. Available at: https://open.umn.edu/opentextbooks/BookDetail.aspx?bookId=139

Burney, A. 2008. Inductive & Deductive Research Approach. University of Karachi. Available at: http://www.drburney.net/INDUCTIVE%20&%20DEDUCTIVE%20RESEARCH%20APPROACH%2006032008.pdf

Docker 2016. Automated Builds on Docker Hub [accessed 08 Oct 2016]. Available at: https://docs.docker.com/docker-hub/builds/

Docker 2016. Automated Builds on Docker Hub [accessed 08 Oct 2016]. Available at: https://docs.docker.com/docker-hub/builds/

Docker 2016. Delivering Containers-as-a-Service (CaaS) with the Docker Cloud [accessed 08 Oct 2016]. Available at: https://www.docker.com/sites/default/files/DK_DockerSubscription_05112016_0.pdf

Docker 2016. Manage service stacks [accessed 08 Oct 2016]. Available at: https://docs.docker.com/docker-cloud/apps/stacks/

Fowler, M. 2013. Continuous Delivery [accessed 10 Oct 2016]. Available at: http://martinfowler.com/bliki/ContinuousDelivery.html

Jasmine 2016. Jasmine Introduction [accessed 30 Oct 2016]. Available at: http://jasmine.github.io/1.3/introduction

Let’s Encrypt. Getting Started - Let's Encrypt - Free SSL/TLS Certificates. Website [accessed 5 Oct 2016]. Available at: https://letsencrypt.org/getting-started/

McLaughlin, M. What Is Agile Methodology? [accessed 10 Oct 2016]. Available at: https://www.versionone.com/agile-101/agile-methodologies/

Ramirez, A.O. 2000. Three-Tier Architecture [accessed 02 Oct 2016]. Available at: http://www.linuxjournal.com/article/3508

Rouse, M. 2007. Software Requirements Specification (SRS). Article [accessed 1 Oct 2016]. Available at: http://searchsoftwarequality.techtarget.com/definition/software-requirements-specification

TravisCI 2016. TravisCI Getting Started [accessed 30 Oct 2016]. Available at: https://docs.travis-ci.com/user/getting-started/

VERSIONONE 2015. State of Agile Report [accessed 28 Sep 2016]. Available at: http://www.agile247.pl/wp-content/uploads/2016/04/VersionOne-10th-Annual-State-of-Agile-Report.pdf

VERSIONONE 2016. Continuous Integration in Agile Software Development [accessed 10 Oct 2016]. Available at: https://www.versionone.com/agile-101/agile-software-programming-best-practices/continuous-integration/

APPENDICES

Appendix 1. Stackfile for Docker Cloud

```
lb:

  environment:

    - "SSL_CERT=-----BEGIN PRIVATE KEY-----\\n PRIVATE KEY 
HERE\\n-----END CERTIFICATE-----\\n"

  image: 'dockercloud/haproxy:latest'

  links:

    - web

  ports:

    - '80:80'

    - '443:443'

  roles:

    - global

web:

  autoredeploy: true

  image: 'huytrinh/thesisapp:latest'
```

Appendix 2. .travis.yml file for TravisCI configuration

```
language: node_js

sudo: false

node_js:

  - "6"

before_install:

  - "export DISPLAY=:99.0"

  - "sh -e /etc/init.d/xvfb start"

before_script:

  - npm install

  - export CHROME_BIN=/usr/local/bin/my-chrome-build

script: node_modules/karma/bin/karma start karma.conf.js --
single-run

branches:

  only:

  - master

after_success:

  - |

        curl -H "Content-Type: application/json" --data
'{"build": true}' -X POST
https://registry.hub.docker.com/u/huytrinh/thesisapp/trigger
/$DOCKER_HUB_TOKEN/
```

Appendix 3. Dockerfile for Docker Automated Build

```
FROM node:latest

MAINTAINER HuyTrinh trinhphandinhhuy@gmail.com

RUN mkdir -p /usr/src/app

WORKDIR /usr/src/app

COPY . /usr/src/app

RUN cd /usr/src/app

RUN npm install

EXPOSE 3000

CMD ["node", "/usr/src/app/server/bin/www"]
```