Maturity Increment and Innovation Report

Reese Gunardi - 12109627 ​

Pierce Kinghorn -12091381 ​

Myles Renaud - 12096375 ​

Jack Rolfe - 12017976​

Nick Kelly – 0194134

Table of Contents

[1 Introduction 2](#_Toc84171485)

[2 Body 3](#_Toc84171486)

[2.1 Initial Development Process 3](#_Toc84171487)

[2.2 Current Development Process 3](#_Toc84171488)

[2.3 Docker 4](#_Toc84171489)

[2.3.1 Overview 4](#_Toc84171490)

[2.3.2 Implementation 4](#_Toc84171491)

[2.3.3 Issues Encountered 4](#_Toc84171492)

[2.3.4 Results 4](#_Toc84171493)

[2.3.5 Industry Literature 4](#_Toc84171494)

[2.3.6 Research Literature 5](#_Toc84171495)

[2.4 Continuous Integration and Continuous Development (CI/CD) 6](#_Toc84171496)

[2.4.1 Overview 6](#_Toc84171497)

[2.4.2 Implementation 6](#_Toc84171498)

[2.4.3 Issues Encountered 6](#_Toc84171499)

[2.4.4 Results 6](#_Toc84171500)

[2.4.5 Industry Literature 6](#_Toc84171501)

[2.4.6 Research Literature 6](#_Toc84171502)

[3 Conclusion 7](#_Toc84171503)

[4 References 8](#_Toc84171504)

# Introduction

The application development cycle in any team is vital for ensuring that the product is delivered successfully, on-time, and to specification. It is not uncommon for technologies and techniques to change over the course of the cycle.

The development team strove to implement the project in with a technology stack matching what the client currently uses.

The specific stack was:

* Linux operating system
* CakePHP framework
* PostgreSQL database
* Apache Webhost

Lack of technical knowledge and time constraints meant that the MySQL was used as a temporary database during development.

Time constraints also led to the adoption of the Materialise CSS framework. The client has applications which use this framework, but they do not consider it part of their core stack. During development, the framework conflicted with the CakePHP framework and was removed. It was decided that a mixture of custom CSS and extending the CakePHP codebase would be utilised.

The remainder of this report will first highlight the changes made to the development process as the project progressed and the two major innovations of Docker and Continuous Integration and Continuous Development (CI/CD). For each innovation the report will surface industry and research studies which highlight the benefits, show evidence of benefits from the application implementation, and highlight some challenges faced.

# Body

## Initial Development Process

The initial development process required some manual configuration for each developer, where all of the programs were downloaded and installed. After this, the program XAMPP was utilised to quickly start the MySQL and Apache resources. This allowed for rapid prototyping and quick start-ups.

When the project began, the plan was to develop the application on the local machines and host locally using XAMPP. Issues were encountered when deploying because the application was developed using the Window’s operating system when the server ran a Linux environment. The deployment introduced environment specific errors so that the application would work locally while creating errors on the server. Errors related to case sensitivity were often an issue because development on a Windows operating system is case insensitive, while a Linux system is case sensitive. This meant that the development and deployment processes consisted of hours of work and were prone to errors. For these reasons, it was decided that the software Docker would be utilised in conjunction with a CI/CD pipeline.

## Current Development Process

Once the Docker and CI/CD technologies were implemented into the system the deployment process became simpler and less prone to error. It should be noted that the development process required more time due to docker building and rebuilding an environment on start-up.

The current process is to build the docker containers with the “docker compose up” command. This builds the images on the local machine and runs the application and database in their specific containers. A developer can access the website to verify their changes by accessing the site on the “localhost” URL.

For this process to work, one must first install Docker. Installation instructions are contained on the project Read Me to help onboard any new developers.

Docker compose allows for the building and linking of multiple containers in one command. It is like a docker container “dockerfile” that manages a series of docker containers in the same way that a regular “dockerfile” manages a series of programs. This process is known as container orchestration.

Multiple containers were used to allow for data persistence and decoupling of functions. The database and application are decoupled. This allows for scalability and reduces the time taken to complete development because the database image does not need to be re-built with every change. Only the application image does.

Finally, on merge to the main branch the CI/CD pipeline deploys the application to the server and builds the image. This is done securely via the GitHub Actions tool which is available for free to public GitHub repositories.

## Docker

### Overview

Docker is a tool which is used for running a standardised virtual environment across operating systems. The virtual environment is configured by a code file which specifies which packages and software to install at build. The benefit of Docker is that it allows for standardised functionality across environments, meaning that changes are reflected exactly from the local environment to the production environment.

### Implementation

Docker was implemented into the application by extending the PHP and Apache image which can be used as a base for PHP projects. CakePHP was installed into the container along with the PHP dependency manager Composer.

### Issues Encountered

When programming in a group setting, it is desirable to start with Docker at the onset of the project. The change of technologies during a project can cause issues for the team. For example, during the creation of this application the team had to spend time configuring XAMPP initially and learning how it works. Furthermore, the late-stage introduction contains risk for server deployments. During the integration of Docker for this application, the server had to be deleted and re-imaged because of some conflicts Docker caused.

Finally, using different technologies introduces different expectations for the development team. For example, rapid prototyping was extremely simple and fast in XAMPP because the server could remain running while the site was reloaded. With docker the entire container had to be re-built for each code change.

### Results

Once implemented Docker provided a seamless integration between our Docker localhost and the Amazon Lightsail server instance that was used for the production deployment.

New features would be updated on the live server with little resistance or unexpected architectural issues because developers could accurately test and bug-fix on their local environments.

### Industry Literature

#### Docker Industry Literature 1

Amazon AWS is a leading cloud provider and was used for the deployment of the application production server. Amazon recommends that Docker is used for applications which need to work on any environment in a highly reliable and low-cost manner. Furthermore, they have collaborated with Docker to make their Lightsail, ECS, and Fargate work with Docker in an easier manner. (Amazon Web Services 2021)

#### Docker Industry Literature 2

Gartner performs high quality industry research into emerging technologies to assist their clients with making the best technology implementation decisions possible. They recommend Docker for the same reasons Amazon does but add the example of reducing errors during the CI/CD process by standardising the installed software programs on the target machine, suiting the application well because CI/CD was utilised (Watson 2021).

### Research Literature

#### Docker Research Literature 1

Research was conducted into supporting technologies which could help the application perform better. The research found was published to the IEEE and shows that Docker is slightly less performant than a native OS deployment for common software tasks such as I/O. The author notes that it is a trade between hardware costs as a detriment when compared to the saving of developer time and the reduction of errors as the benefit. The author concludes that as the Docker technologies improve, and hardware costs decrease the determinants will become almost non-existent (Zhao et al. 2021).

#### Docker Research Literature 2

This performance comparison between virtual machines and docker containers studies the evaluation and practices between methods and the methodology used to test. Providing details on Docker containers rapid execution times in comparison to a virtual machine lowered performance (Yadav, Sousa & Callou 2018).

## Continuous Integration and Continuous Development (CI/CD)

### Overview

CI/CD is a combination of continuous integration and continuous deployment allowing a direct pipeline to manage and update live server applications and database from the development environment. This provides a process and system for repaid deployment and flexibility to respond to changing environments in application development. CI/CD supports team-based automation, building, testing and deployment of application development.

### Implementation

The CI/CD process was implemented by using GitHub Actions as a pipeline manager. The manager is configured using the YAML language and is free for public repositories. GitHub Actions allows for certain information (such as SSH passphrases) to be converted into environment Secrets which are stored in a secure location to prevent security breaches.

### Issues Encountered

Deployment of new seeding scripts and database changes were limiting factors of CI/CD prior to Docker implementation, updates, and changes to the database via a seeding script were unable to be automated and required extensive resetting and restarting of server infrastructure. CI/CD issues occurred during some updates with differing architecture between the LightSail server and CI/CD pipeline in comparison with localhost testing.

### Results

The CI/CD pipeline provided the system and processes for the team to self-manage live deployments that supported individuals in a free-flowing environment. This drastically reduced deployment speeds and human resources required to develop, merge, and deploy contributions and features from all team members in parallel.

### Industry Literature

CI/CD Industry literature 1

Amazon documentation provides insight and knowledge using CI/CD on Amazon Web Services. These benefits, challenges, features and outcomes are explained in detail with supporting information required to tool a software development environment using CI/CD and DevOps (Stacy, et al. 2017).

CI/CD Industry Literature 2

This article provides a deeper look at DevOps methodologies and approaches to management using a container approach system. The article shows that CI/CD can decrease teamwork issues which are caused from commits with errors (Vedagiri and Rajamani 2017).

### Research Literature

#### CI/CD Research Literature 1

This study (Gallaba 2019) seeks to analyse and improve upon existing CI/CD systems. In this article various limitations are listed regarding use of CI/CD. Using the information provided by this article significantly hastened the workflow cycle.

#### CI/CD Research Literature 2

A peer reviewed article Continuous Delivery: Huge benefits, but challenges too (Chen 2015) covers’ the process of CD with a code, build, acceptance test, performance test, manual test, production approach.

# Conclusion

In conclusion, the reduction of errors and the reduced manual workload offset the increased build time when developing the application.

The initial development processes the team used was prone to errors and dramatic workloads when deploying changes to the server. Furthermore, individual installations of all software programs introduced the risk of using incompatible or outdated program versions.

The current process has seen a reduction in errors and deployment times, but an increase in the time taken to develop on a local machine.

Docker was introduced to prevent the “works on my machine” issue where developers report that the problem only occurs on the server, but not on their local environment. It was seen that the Windows to Linux errors were eliminated along with the file-path errors which occur when using XAMPP on localhost compared to a production instance of Apache. One must note that Docker alternatives are available and should be considered. The key takeaway should be that containerisation technology can reduce the errors in deployment.

Additionally, CI/CD was introduced to assist with deployment. When combined with Docker, the deployment process became almost error free and simple. The only action needed was a merge to the Main branch then the application would deploy without error.

It is this report’s recommendation that Docker and a CI/CD pipeline are used from the start of the project with a project key performance indicator being to optimise the process flow. It is estimated that with proper Docker Image build optimisation and more effective CI/CD methods that the speed of development could almost match that of using XAMPP while the speed of deployment could be almost instantaneous and result in zero-downtime deployments.

# References

Amazon Web Services, Inc 2021, What is Docker?, viewed 1 October 2021, https://aws.amazon.com/docker/

Chen, L 2015, 'Continuous delivery: huge benefits, but challenges too', IEEE Software, vol. 32, no. 2, pp. 50-54.

Gallaba, K 2019, 'Improving the robustness and efficiency of continuous integration and deployment', 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 619-623.

Watson, R 2021, What you need to know about Docker and containers, viewed 1 October 2021, https://www.gartner.com/en/documents/3512817/what-you-need-to-know-about-docker-and-containers

Stacy, D, Prudnikov, M, Khan, A & Shen, X 2017, *Practicing continuous integration and continuous delivery on AWS*, Amazon Web Service, Inc.

Vedagiri, K & Rajamani, R, 2017, *Using containers to more effectively manage DevOps continuous integration*, Cognizant.

Yadav, RR, Sousa, ETG & Callou GRA 2018, 'Performance comparison between virtual machines and Docker containers', IEEE Latin America Transactions, vol. 16, no. 8, pp. 2282-2288, viewed 1 October 2021.

Zhao, N, Tarasov, V, Albahar, H, Anwar, A, Rupprecht, L, Skourtis, D, Paul, AK, Chen, K & Butt, AR 2021, 'Large-scale analysis of Docker images and performance implications for container storage systems', IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 4, pp. 918-930, viewed 1 October 2021.