Micah Gabriel Hobby (17027531)
BSc (Hons) Computer Science, Placement Student

Analysis of Docker and Contenerisation for an organisational workflow

20/01/20 – UFCFE6-15-3 Professional Experience

UWE Bristol | University of the West of England

Table of Contents

1. Introduction

In the highly digital world we live in an online presence is vital for any organisation looking to expand its customer base. With approximately 93% of the U.K's population having access to the internet (O'Dea, 2018). There are approximately 175 billion euros worth of transactions and this is consistently growing every year, predicted to reach 200 billion in 2019 (Sabanoglu, 2019). With traffic moving from the high-street to the internet (Butler, 2018), organisations are looking for complete development/hosting service packages especially if they don't have an in-house development team or the budget for several new hires required.

Circle Interactive provides hosting, maintenance, security audits and development as a simple and effective solution to clients looking to improve member and customers online experience. They provide non-technical organisations with the opportunity to develop an online presence and services.

Over the last few months Circle Interactive has faced increased demand from new clients. Since a large portion of work involved with setup and software upgrades follows a predetermined workflow it has caused a bottleneck. This is due to Circle Interactive currently only having 2 System Administrators, therefore a push has been made to re-prioritize system admin time. To achieve this they are training developers to perform more server related tasks to allow them more time to focus on client responsibilities.

Currently at the organisation most developers are more focused on implementing software solutions, with only the occasional server related tasks. With sysadmins required to perform setup, maintenance and upgrade tasks that junior developers could easily cover with some training. Since the influx of new clients, and due to there being more developers than sysadmins, simplifying setup and including developers in server maintenance and setup is crucial. Part of the new initiative is a new solution that has been proposed that would allow developers to work more closely with sysadmins on client projects.

By implementing Docker and using container technology, therefore allowing for enhanced benefits in version control, easier backups and upgrades. This would potentially free up sysadmins time and provide developers with additional tools and knowledge when developing for a client.

This paper intends to cover the issues related to implementing docker into the organisational workflow. Covering related concerns about security, energy usage and efficiency around this upcoming technology. It aims to understand the benefits of implementing Docker by looking at previous research conducted around containerisation. It also covers academic journals on the benefits/drawbacks of its use, and analyses whether it should be implemented at Circle Interactive.

2. Literature Review

Container based applications have been described by Bashari Rad et al. (2017) as having the advantage of 'speed, portability, scalability, rapid delivery, and density'. They have the advantage of being very small and are therefore very quick to build, develop, test and deploy. This paper measures and compares the performance of Docker and Virtual Machines. It further outlines the advantages of using docker containers over traditional bare-metal systems. It overall provides a performance based overview for each technology, and performs a broad analysis of the various systems.

For comparison of actual system performance Felter et al. (2015), contrasted CPU, memory and IO performance of virtual machines and containers. The evaluation was carried out using MySQL as the benchmark, it's objective was to find what performance overhead if any, is caused for I/O intensive workloads. This paper was highly quantitative and technical, it explored claims of definitive evidence of performance overhead.

Another study by Boettiger (2014) has analysed containerisation focusing on four technical challenges faced by containers. This paper was more qualitative in nature and provides a more organisational and human view with challenges and benefits of containerisation such as:

1. Dependency Hell, where less than 50% of software studied could be successfully built or installed. Since Researchers were unable to recreate the original software conditions, they were unable to run it. This was due to outdated or unmet software dependencies. This is a challenged currently faced by Circle Interactive, due to older versions which clients have not paid to be maintained, rendering them useless once they break.
2. Imprecise Documentation, on how to reproduce the conditions to run the code. This was seen as a major barrier to entry and prevents researches from being able to even build the software. This problem affected around 70% of the projects covered in the study. Circle Interactive faces a similar

challenged due to tight development cycles, with incomplete documentation that sometimes hinders progress, especially on legacy software developed by members who have left the team.

3. Code Rot. Changes in a projects dependencies such as bugfixes or new development can change results of the applications being tested. It is important that any solution can be proven to be robust against such changes, especially when there are updates that improve a software's API, and possibly deprecate functionality.
4. Barriers to adoption & re-use. The study found that available solutions such as workflow software, virtual machines, continuous integration services, create a barrier to entry since they require additional effort to overcome the learning curve of the service. This causes additional and unnecessary effort and might discourage contributing to a project. If it is not easy to make code available for extension less people are likely to do it.Circle Interactive could benefit from making solutions implemented easier to re-use on other projects.

The paper focuses on alternative solutions to both academic software and software development in general. It outlines how Docker and containerisation solves these issues and what benefits it provides. It explored evidence to demonstrate how the technology could integrate with Circle Interactive.

As for research into energy consumption and energy efficiency, research by Santos et al. (2018) was made into energy consumption of docker container workloads compared to bare-metal systems. Raun et al. (2016) tested file mount implementation and port mapping, research was performed on available cloud containers from Amazon & Google. Both researchers intended to summarise the quantitative metrics of docker performance.

And finally, Bui (2015) conducted research into the security of Docker. This is in comparison to bare-metal systems and virtual machines, it analyses the internal security of Docker and how Docker interacts with security features of the Linux kernel. This paper lacked analysis of real world threat models and focused more on analysis of existing security.

3. Methodology

This topic initially arose as part of the companies yearly review, as an attempt at streamlining the organisations operations. To ensure the solution was relevant, conversations were held with the Head sysadmin Robert, as well as the Head Developer and the Student Manager. From these conversations it was clear, docker offered a clear separation of responsibilities and allowed for developers to perform more tasks therefore freeing up sysadmins time. It was referred to as an 'organisational goal' by the Student Manager.

From there research was conducted into the feasibility of containers. While informing the research for this paper, a variety of sources were employed including:

- The UWE library and Google Scholar to locate relevant literature and research papers.
- Academic Journals.
- Docker's whitepaper and organisational case studies.
- Official company statistics and journal information to inform comparisons.

After the initial research, more investigation was conducted into areas of concern to Circle Interactive. Particular interest was shown in regard to current workflow, which employ VMs and bare-metal (no virtualisation or containerisation) systems. Therefore, analysis into performance, energy efficiency and security were needed.

4. Findings

From all the research conducted there appear to be 4 separate challenges when implementing docker in DevOps.

- Security: From analysis conducted by Bui (2015), it was determined that Docker provides a high level of isolation and resource limiting, through the use of cgroups, copy-on-write file system, and name spaces. They stated docker containers are secure out of the box and can be hardened by running them as "non-privileged". It is also stated that using docker is comparable in safety to using a VM and offers similar security mechanisms. (see Appendix 1)
- Performance: Findings from studies by Santos et al. (2018) indicate that Containerisation is of low overhead compared with virtualization from Virtual Machines. It is a lightweight alternative with better performance than a VM, as it offers isolation and resource control without the overhead of an additional kernel for each instance of a container. They found that docker is now displacing virtual machines as the virtualization solution of choice.

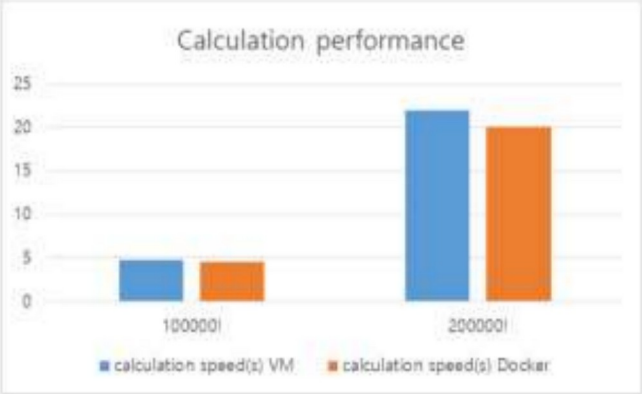

Figure 1: CPU Calculation Performance (Bashari Rad et al. 2017)

Docker allows for hundreds of containers on a single system making it efficient. In a similar set of tests by Felter et al. (2015), comparing memory, IO and network performance they found Docker outperforms KVM. On similar Redis and MySQL databases, Docker performs just as well as bare-metal Linux after configuration.

A third set of experiments conducted by Ruan et al. (2016) found that dockers file mount implementation and port mapping cause higher I/O latency. The experiment performed research on available cloud containers from Amazon & Google and concluded that containerisation is only feasible on bare-metal, and not inside VM's. Overall the research indicates containers are more efficient with less performance overhead than VM's. However, compared to bare-metal that can be up to a 5% decrease in I/O operations performance

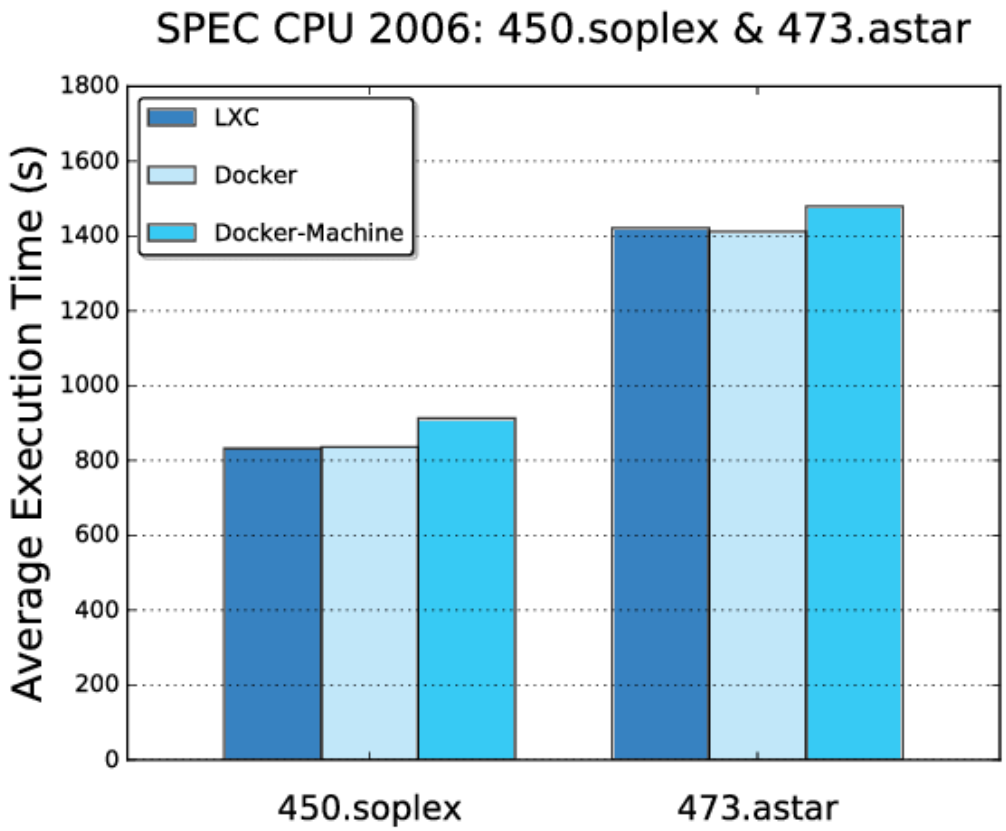

Figure 2: Execution time of Docker against LXC (operating-system-level virtualization) Lower is better. (Ruan et al., 2016).

- Energy Efficiency: Santos et al. (2018) studied energy consumption of docker containers compared to bare-metal systems and found that when running similar processes on both systems containers show an increase in average power consumption of around 2W.

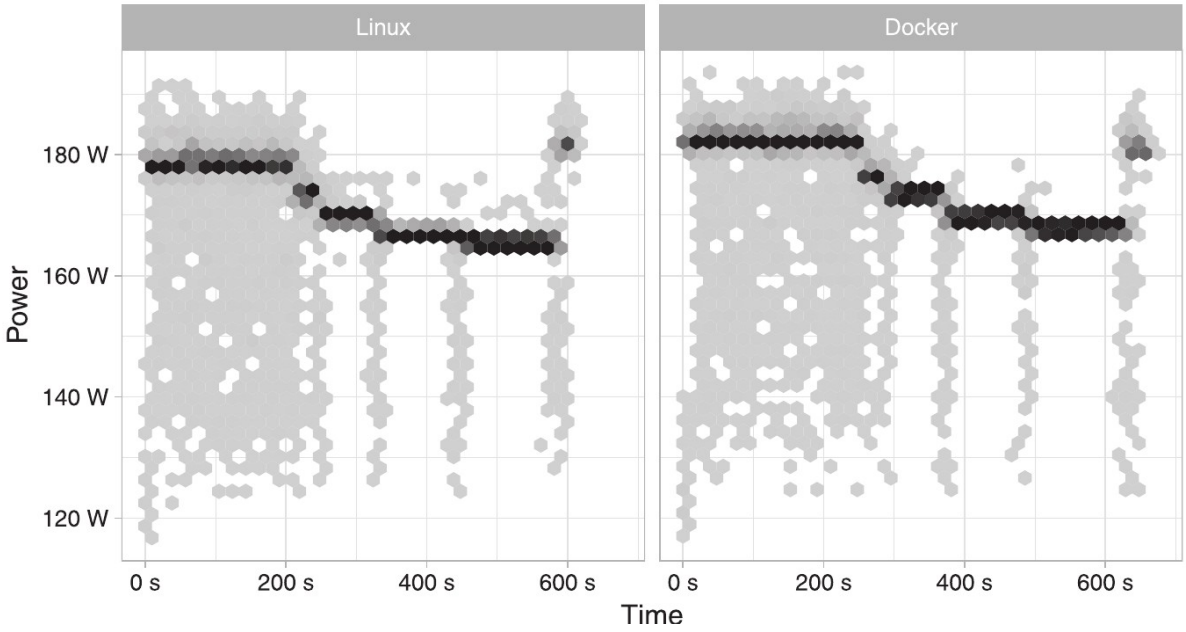

Figure 2: Density plot of wattage measurements across all Redis Benchmark runs over time (Santos et al. 2018).

- Ease of Use and re-usability: The use of containers allows for effective version control since Dockerfiles are small and plaintext, ideal for git. As research by Boettiger (2014) indicates Docker also handles the packaging and execution of a container so that it works identically across different machines. This lowers the barrier to entry and increases opportunities for reuse and quick deployment. Containerisation allows for a program to work identically across different platforms and environments.

It is also good to note that according to Santos et al. (2018) containerisation and docker are being adopted at a blistering pace, and have become an integral part of DevOps. Since it runs on GNU/Linux based systems it can be deployed to around 67% of server side devices (Finley et al., 2016).

5. Discussion

Docker provides a complete binary image where all the software including dependencies has been configured and tested and is ready for easy distribution and setup. It immediately improves a projects imprecise documentation, by providing a simple script that includes everything required, allowing for efficient version control and reducing setup time on different machines and systems.

Using containerisation also tackles the issue of code-rot, which occur when changes to dependencies are introduced with new versions. These can be caused by security fixes, new features or fixes to deprecation. Without docker and containerisation it would be difficult to track when code-rot happens.

And finally Docker lowers the barrier of entry for utilizing and further extending software. Especially in the context of open source software, such as that used at Circle Interactive, it is important that code re-use and low barriers to entry are present.

In terms of security Docker and containerised applications have adequate security compared to bare metal solutions. With performance there is a slight decrease in raw performance when compared to bare-metal. Due to the previously mentioned slight disadvantages an organisation might be discouraged about the overall value docker provides.

However, these disadvantages are more than made up for by the value of containerisation and the benefits provided in the forms of time & cost savings. In energy efficiency, the increased 2W in power consumption can be offset by dynamic power consumption, only using what is needed and scaling down in periods of low demand.

Currently Circle Interactive's workflow includes 3 working environments: 'develop', 'test', and 'production'. After a weekly development sprint work must be ported to a test environment or to production. This requires a non-trivial amount of time to properly port new changes and merge environments for deployment.

With a Docker setup Circle Interactive could benefit from containerisation, allowing near instant pack, port and unpack to the new environment, while still maintaining the necessary version control and redundancy. It is important to note that docker would allow Circle Interactive to handle an increased customer influx, as it would increase productivity for sysadmins and developers, and reduce the time spent on version control and setup.

6. Conclusion

Implementing Docker at Circle Interactive would provide a number of clear benefits:

- It would improve the quality of the current documentation, by providing up to date and complete information on the current installation. This includes dependencies, current software version and clear instructions to reproduce it. It would also be extremely beneficial during software releases as it would provide an easy pathway to upgrade.

- Docker containers allow for the separation of inside and outside of the container. It allows for system admins to deploy and maintain the server with containers, and for the developer to look after the applications inside the docker container. This would shift the current responsibility of sysadmins to maintain both inside and outside and allow developers to manage the whole installation and development process.
- It would allow containers to work in every environment with dependencies embedded within the applications. Therefore, providing a reliable, consistent and improved environment allowing for predictable results. It allows for more containers to be run on a single host, therefore offering higher density and no overhead wastage of resources.
- Circle Interactive would benefit from the increased simplicity of deploying new changes and could improve it's version control.
- Overall it would allow the organisation to provide better service to the client and to operate more efficiently as an organisation. It would also strengthen their commitments to contributing to the community, by lowering the barrier to entry and encouraging re-use.

In conclusion the benefits of docker far outweigh the negatives. Therefore, Circle Interactive would benefit from implementing the technology at every stage of production, deployment and maintenance on their clients software.

After conducting this research it is very clear why docker and containerisation are becoming so widely used, from industries such as online-gaming to e-commerce. It is clear that Docker will continue to grow in popularity and adoption, it has a clear place in DevOps and in Circle Interactive.

7. Recommendations

After analysing Circle Interactive's requirements and organisational workflow, it is clear that Circle Interactive should continue to implement a stronger policy to include the use of docker, into any future client work.

To ensure that legacy software does not become unusable, they should also offer already existing clients the opportunity to upgrade to Docker at a slightly subsidised rate. This would benefit both the client and Circle Interactive, with shorter development, backup and troubleshooting times.

Due to its wide adoption in the industry it is very likely clients would approve the adoption and would desire their software in a container. Circle should continue their current path to adoption and should continue to ensure their clients up to date reliable software.

Word count: 2,485

8. Appendix 1

Available Container Security Features, Requirements and Defaults			
Security Feature	**LXC 2.0**	**Docker 1.11**	**CoreOS Rkt 1.3**
User Namespaces	Default	Optional	Experimental
Root Capability Dropping	Weak Defaults	Strong Defaults	Weak Defaults
Procfs and Sysfs Limits	Default	Default	Weak Defaults
Cgroup Defaults	Default	Default	Weak Defaults
Seccomp Filtering	Weak Defaults	Strong Defaults	Optional
Custom Seccomp Filters	Optional	Optional	Optional
Bridge Networking	Default	Default	Default
Hypervisor Isolation	Coming Soon	Coming Soon	Optional
MAC: AppArmor	Strong Defaults	Strong Defaults	Not Possible
MAC: SELinux	Optional	Optional	Optional
No New Privileges	Not Possible	Optional	Not Possible
Container Image Signing	Default	Strong Defaults	Default
Root Interation Optional	True	False	Mostly False

Appendix 1: ‘Understanding and Hardening Linux Containers’ from Introduction to Container Security (Anon, 2016).

9. References

Anon (2016) *Introduction to Container Security* [online]. Available from: https://www.docker.com/sites/default/files/WP_IntrotoContainerSecurity_08.19.2016.pdf [Accessed 20 January 2020].

Bashari Rad, B., Bhatti, H. and Ahmadi, M. (2017) An Introduction to Docker and Analysis of its Performance. *IJCSNS International Journal of Computer Science and Network Security*. 173 pp. 8.

Boettiger, C. (2014) An introduction to Docker for reproducible research, with examples from the R environment. *ACM SIGOPS Oper. Syst. Rev.* [online]. 49 . Available from: https://www.researchgate.net/publication/266477818_An_introduction_to_Docker_for_reproducible_research_with_examples_from_the_R_environmentdoi:10.1145/2723872.2723882.

Bui, T. (2015) *Analysis of Docker Security*. Available from: https://www.researchgate.net/publication/270906436_Analysis_of_Docker_Security.

Butler, S. (2018) *6,000 shops close in tough year for UK's high streets* [online].

Felter, W., Ferreira, A., Rajamony, R. and Rubio, J. (2015) An updated performance comparison of virtual machines and Linux containers. In: *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)* [online]. March 2015 pp. 171–172. Available from: https://www.researchgate.net/publication/295899644_An_updated_performance_comparison_of_virtual_machines_and_Linux_containersdoi:10.1109/ISPASS.2015.7095802.

Finley, K., Finley, K., Edelman, G., Simonite, T., Knight, W., Thompson, C. and Raviv, S. (2016) *Linux Took Over the Web. Now, It's Taking Over the World* [online].

John, W. (2015) *Docker and the Three Ways of DevOps* [online]. Available from: https://www.docker.com/sites/default/files/WP_Docker and the 3 ways devops_07.31.2015 (1).pdf [Accessed 20 January 2020].

O'Dea, S. (2018) *Topic: Internet usage in the UK* [online].

Ruan, B., Huang, H., Wu, S. and Jin, H. (2016) *A Performance Study of Containers in Cloud Environment*. In: [online]. 2016 pp. 343–356. Available from: https://www.researchgate.net/publication/310514065_A_Performance_Study_of_Containers_in_Cloud_Environmentdoi:10.1007/978-3-319-49178-3_27.

Sabanoglu, T. (2019) *E-commerce sales in the UK 2014-2018 Statista* [online].

Santos, E.A., McLean, C., Solinas, C. and Hindle, A. (2018) How does docker affect energy consumption? Evaluating workloads in and out of Docker containers. *Journal of Systems and Software*. 146 pp. 14 – 25. doi:https://doi.org/10.1016/j.jss.2018.07.077.