Using Docker Containers to Improve Reproducibility in Software and Web Engineering Research

Jürgen Cito[1(✉)], Vincenzo Ferme[2], and Harald C. Gall[1]

[1] University of Zurich, Zurich, Switzerland
{cito,gall}@ifi.uzh.ch
[2] University of Lugano (USI), Lugano, Switzerland
vincenzo.ferme@usi.ch

Abstract. The ability to replicate and reproduce scientific results has become an increasingly important topic for many academic disciplines. In computer science and, more specifically, software and web engineering, contributions of scientific work rely on developed algorithms, tools and prototypes, quantitative evaluations, and other computational analyses. Published code and data come with many undocumented assumptions, dependencies, and configurations that are internal knowledge and make reproducibility hard to achieve. This tutorial presents how Docker containers can overcome these issues and aid the reproducibility of research artifacts in software and web engineering and discusses their applications in the field.

Keywords: Reproducibility · Containers · Cloud

1 Motivation

Reproducibility can be described as the repeatability of a certain process in order to establish a fact or the conditions under which we are able to observe the same fact [1]. The ability to replicate and reproduce scientific results has become an increasingly important topic for many academic disciplines. In computer science and, more specifically, software and web engineering (SE/WE), contributions of scientific work rely on developed algorithms, tools and prototypes, quantitative evaluations, and other computational analyses.

However, even if code and data are published alongside the paper as open source artifacts, they come with many undocumented assumptions, dependencies, and configurations that make reproducibility hard to achieve [2]. Reproduction of results often requires internal knowledge that is missing from the published manuscript.

Docker container [3] is an open source technology that can address the issues of reproducibility in SE/WE research. Containers can be seen as lightweight virtual machines that allow to set up a computational environment, including all necessary dependencies (e.g., libraries), configuration, code and data needed,

A. Bozzon et al. (Eds.): ICWE 2016, LNCS 9671, pp. 609–612, 2016.
DOI: 10.1007/978-3-319-38791-8_58

within a single unit (called *image*). The steps necessary to achieve the state in such an image are documented within a *Dockerfile*, a script that holds all infrastructure configuration and commands. Images can be distributed publicly and seamlessly run on Linux, and also have support for major operating systems through Docker machine. The major difference to virtual machines is that Docker images share the kernel with the underlying host machine, which enables much smaller image sizes and higher performance. This has made Docker particularly attractive to industry and has thus seen a steep rise in adoption of the technology [4,5].

Containers address the shortcomings of previous approaches (e.g., open sourcing) and make artifacts in SE/WE research immediately usable to reviewers, interested readers, and future researchers and improves dissemination of scientific results.

This tutorial aims on giving a hands-on introduction to Docker, and show how researchers can package an existing research project in the SE/WE community within a Docker container.

2 Importance to the Web Engineering Community

In recent years, software and web engineering conferences have started to encourage the submission of artifacts that support replication (e.g., replication packages at FSE[1], data showcase at MSR[2]), signaling the importance of reproducibility in the field.

Reproducibility can be further improved if all artifacts belonging to a paper are packaged and documented in Docker containers. This allows others to immediately make use of the package without the need of internal knowledge and without dependency issues.

This tutorial will offer an opportunity to familiarize the audience with how Docker containers work and how SE/WE researchers can leverage this technology to provide a reproducible package to their own research. More specifically, it will give a hands-on tutorial on how existing prototypes can be packaged to form a reproducible entity.

3 Outline

The tutorial is supposed to take half a day (3 h). It will first introduce the basics of container technology, how it differs to virtual machines, and why it has gained widespread attraction in industry. It will then convey the basic building blocks of how an image can be constructed. In addition, it will give guidance on how to best produce a Dockerfile out of working containers. It will then continue to apply these basic techniques to a specific use case in the Web Engineering domain. The tutorial will conclude with a discussion on the advantages, challenges, and

[1] http://esec-fse15.dei.polimi.it/replicationPack.html.
[2] http://2016.msrconf.org/#/data.

limitations of the use of containers to enable reproducibility in SE/WE research. The detailed outline of the tutorial is described in the following.

1. Introduction to Containers and Reproducibility of SE/WE Research.
The tutorial will start by introducing the term reproducibility in relation to SE/WE research. It will continue to introduce container technologies and how they can help with reproducibility.

2. Docker Container Basics.
The tutorial will cover a short overview of the Docker ecosystem and will introduce the basic building blocks and its tooling. This block in the tutorial will also walk through the process and concrete instructions necessary to build an initial container.

3. Web Engineering Use Case.
This part of the tutorial will walk through a concrete use case that could be found in web engineering. The use case is based on a distributed, real-time node.js application, realized by multiple services. The concrete instructions to construct the Docker image will be elaborated along the way.

4. Open Challenges and Limitations.
We conclude the tutorial with a discussion on the open challenges that still remain in the area of reproducibility, what kind of limitations exist.

All materials covered in this tutorial, including all scripts and resulting artifacts, will be made available online at:
http://www.ifi.uzh.ch/seal/people/cito.html.

4 Target Audience

This tutorial is suitable for both academic researchers and industry professionals that want to learn more about Docker containers and reproducibility in general. No prior knowledge of Docker or any other container technology is necessary. To follow along with the instructions, we assume basic skills in working with the Linux console (e.g., `bash`). The audience will be pointed to further material, for those who want to learn more about container technologies.

5 About the Organizers

The material to be included in the tutorial is authored by Jürgen Cito, Vincenzo Ferme, and Harald C. Gall.

Jürgen Cito is a Ph.D. candidate at the University of Zurich, Switzerland. In his research, he investigates the intersection between software engineering and cloud computing. In the summer of 2015, he was a research intern at the IBM T.J. Watson Research Center in New York, where he worked on cloud analytics

based on Docker containers. That year he also won the local Docker Hackathon in New York City with the project `docker-record`[3].

More information is available at: http://www.ifi.uzh.ch/seal/people/cito.html.

Vincenzo Ferme is a Ph.D. candidate at the University of Lugano (USI), Switzerland. In his research, he is involved in the BenchFlow Project. The goal of the project is to design the first benchmark for assessing and comparing the performance of workflow management systems. In the context of the project, he is developing a framework for automated software performance benchmarking that largely relies on Docker[4].

More information is available at: http://www.vincenzoferme.it.

Harald C. Gall is a professor of software engineering in the Department of Informatics at the University of Zurich, Switzerland. His research interests include software engineering, focusing on software evolution, software quality analysis, software architecture, reengineering, collaborative software engineering, and service centric software systems. He was the program chair of the European Software Engineering Conference and the ACM SIGSOFT ESEC-FSE in 2005 and the program co-chair of ICSE 2011.

More information is available at: http://www.ifi.uzh.ch/seal/people/gall.html.

References

1. Mockus, A., Anda, B., Sjøberg, D.I.: Experiences from replicating a case study to investigate reproducibility of software development
2. Boettiger, C.: An introduction to docker for reproducible research. ACM SIGOPS Oper. Syst. Rev. **49**(1), 71–79 (2015)
3. Merkel, D.: Docker: lightweight linux containers for consistent development and deployment. Linux J. **2014**(239), 2 (2014)
4. Gerber, A.: The state of containers and the docker ecosystem: 2015. Technical report, White paper
5. Cito, J., Leitner, P., Fritz, T., Gall, H.C.: The making of cloud applications: an empirical study on software development for the cloud. In: Proceedings of the 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, pp. 393–403. ACM, New York (2015)

[3] https://github.com/citostyle/docker-record.
[4] https://github.com/benchflow.