<https://helda.helsinki.fi>

Teaching Container-Based DevOps Practices

Kousa, Jami

Springer

2020-06-10

Kousa , J , Ihantola , P , Hellas , A & Luukkainen , M 2020 , Teaching Container-Based DevOps Practices . in M Bielikova , T Mikkonen & C Pautasso (eds) , Web Engineering : ICWE 2020 . , 34 , Lecture Notes in Computer Science , vol. 12128 , Springer , Cham , pp. 494-502 , International conference on web engineering , Helsinki , Finland , 09/06/2020 . https://doi.org/10.1007/978-3-030-50578-3_34

<http://hdl.handle.net/10138/330789>

https://doi.org/10.1007/978-3-030-50578-3_34

acceptedVersion

Downloaded from Helda, University of Helsinki institutional repository.

This is an electronic reprint of the original article.

This reprint may differ from the original in pagination and typographic detail.

Please cite the original version.

Teaching Container-based DevOps Practices

Jami Kousa^{1*}[0000–0002–4862–8062], Petri Ihantola¹[0000–0003–1197–7266],
Arto Hellas²[0000–0001–6502–209X], and Matti Luukkainen¹[0000–0003–4779–4998]

¹ University of Helsinki, Helsinki, Finland
jami.kousa@helsinki.fi

² Aalto University, Espoo, Finland

Abstract. We present the design of a online course that focuses on container-based virtualization as part of the DevOps toolchain. In addition, we outline the professional background of participants taking the course, and describe how this affects perceived previous knowledge of DevOps. We found out that the self-evaluated conceptual understanding of DevOps topics is nearly equal regardless of the participants professional identity (e.g., student or developer). However, there are significant differences in how much participants have used tools like Docker before. We conclude that there is a clear need for lifelong learning among software engineering professionals as (future) developers often struggle in operations related skills such as command line or networking.

Keywords: DevOps · Education · Lifelong learning

1 Introduction

Traditionally the development and operation of software have been separated. However, bringing these functions together to provide an optimal flow of software from ideas to production is getting mainstream in software engineering [3]. The joint operation of developers and operations people is referred to with the term DevOps. DevOps has numerous definitions, which show different views of the term and the field. A systematic mapping study by Jabbari et al. combines them by stating [6]: “*DevOps is a development methodology aimed at bridging the gap between Development and Operations, emphasizing communication and collaboration, continuous integration, quality assurance, and delivery with automated deployment utilizing a set of development practices.*”

Based on our observations, the operations (and consequently DevOps) have at most a lightweight role in many CS or even Software Engineering programs. Deployment, operation, and maintenance are part of in ACM Curricula recommendation for CS but are not apparent in the course examples [1]. DevOps-related skills were poorly trained also at our university a few years ago [9].

We argue that integrating DevOps into the university curricula can be used to introduce the operations to students as well as rehearse and strengthen existing knowledge in software development in general. However, teaching DevOps can be challenging as it crosses boundaries of traditional computer science courses [4]. The work presented in this article emerges from our personal experiences of teaching DevOps practices at the University of Helsinki.

2 Background

A major effort for DevOps to succeed is to make the overall software delivery flow continuous and using a meaningful set of tools [11]. Containers are a concrete example of this. They are operating-system-level virtualization designed to be a standard unit of software that packages up code and all its dependencies, so the application runs reliably from one computing environment to another. Containers facilitate DevOps by allowing the use of code to perform infrastructure management tasks automatically (i.e, Infrastructure-as-Code) [13] and by removing the need to manually install software when setting up servers [12]. Moreover, containers enable fast deployment and recovery, supporting continuous deployment [7]. Docker is an open software for containerization that has been adopted as the de-facto method for virtualization in the industry [14].

One of the most comprehensive reports on teaching DevOps was published in 2016 by Christensen [4], who describes the challenges and proposes teaching methods for DevOps. Discussion in the paper is done in relation to a course in which students developed web software that scales to tens of thousands of concurrent users. In conclusion, they recommend emphasizing the programming process and realistic context for teaching DevOps skills.

In a similar vein, Pengxiang and Leong [10] published a study on teaching DevOps related hybrid-skills in the cloud computing environment. They analyzed the skill-set wanted by the industry, and identified a gap in a current polytechnic cloud computing curriculum. Their proposed an approach where students deployed web software for other students to submit their learning diaries.

A recent publication by Kuusinen and Albertsen [8] explores university-company collaboration in teaching Continuous Delivery and DevOps in a two weeks intensive course. The first week was theoretical, while the second was practical. Authors found the approach where industry partners up-to-date technical knowledge was integrated into university education successful.

Finally, Christensen et al. acknowledge the scarcity of research done on teaching DevOps, especially in teaching methodology and culture of DevOps. Some DevOps practices are introduced in other courses to provide some experience of their usage. Version control management systems, which may be classified as DevOps tools, such as git, can be introduced in many courses as Haaranen and Lehtinen describe [5]. Other tools, such as continuous integration using CircleCI³, are integrated into software engineering projects. These kinds of integrations make it possible for students to learn DevOps tools and practices during courses that do not focus on DevOps as a methodology.

3 Research Questions and Methodology

Based on the previous research, we have developed course that introduces DevOps via relevant tools. In this study, we investigate how our approach works. In addition, we are interested in what kinds of students come to the course,

³ <https://circleci.com/>

and how the background is linked to the participants previous knowledge. The concrete research questions we will answer are:

- RQ1** What is the professional background of participants taking the course?
RQ2 How does the professional background affect participants' perceived previous understanding of DevOps?
RQ3 What are the core challenges for students when learning DevOps?

3.1 DevOps with Docker Course

Our DevOps with Docker course is based on study material first used in the internal training of an industry partner. The material was then tailored to create an introductory course to containerization for students at the university level. So far the *DevOps with Docker* course have been held three times. The latest instance of the course was held in Autumn 2019 as a MOOC for anyone. In the following, we will briefly introduce the course design. (see <https://devopswithdocker.com/>).

The course consists of four parts. The first part focus on setting up the environment by installing docker and testing it works. The next two parts are an introduction to containerization with Docker and container orchestration with docker-compose. More precisely, this is about containerization of an existing web application (Figure 1, items 1 & 2) that was created for the purposes of this course. In the third part, the application is integrated with other services; cache (3), database (4) and a reverse proxy (5). In the fourth part, students still work with the same web application but the tasks are related to improving the build process and security. The course size is 1-3 ECTS points (depending on how many sections students will complete).

The main application containerized and configured throughout the course is a simple web service implementing the automated assessment and thus providing immediate feedback for students. The service returns a web page with with a single button. Pressing the button returns either an ok message (if tasks are correctly solved) or an error message relates to one of the services. After students have completed a session, they were requested to push their progress to GitHub and submit a link to their GitHub repository.

Fig. 1. Multiple services were required in the configuration to complete the exercises. The order of the exercises are numbered from 1 to 5.

3.2 Participants and Measures

We will answer our research questions based on the data collected from the last iteration of the course. At the beginning of the first section of the course, students were asked about their background, previous experiences in DevOps and Containers, and personal learning objectives. The detailed items related to the background were BG1: I have a general understanding of containers (1=strongly disagree, ... 5=strongly agree), BG2: I have understanding of problems containers solve (1=strongly disagree, ... 5=strongly agree), BG3: I have used Docker before (1=none, ... 5=daily), BG4: I'm familiar with DevOps or have general understanding what it is (1=strongly disagree, ... 5=strongly agree), and BG5: My background is (student, developer, data scientist, DevOps engineer, other).

At the end of the course (i.e., at the end of Section 2, 3, or 4), there was a feedback questionnaire where we asked, e.g., *Which exercise did you consider to be the hardest and why?* In addition, as the pre-course survey was anonymous, the question about student professional background (BG5) was repeated at the end of the course. The initial survey was answered by 651 participants. Phases 1, 2, 3, and 4 of the course were submitted by 219, 175, 142 and 98 participants, respectively. Post-course feedback was provided by 140 participants.

Table 1. (Spearman) correlations between items related to previous knowledge, their count (N), means (M) and standard deviation (SD). * $p < .05$, ** $p < .01$, and *** $p < .001$

	2.	3.	4.	N	M	SD
1. BG1	0.49***	0.13**	0.17***	632	2.67	1.24
2. BG2		0.19***	0.16***	632	2.60	1.16
3. BG3			0.03	630	1.91	1.05
4. BG4				627	2.78	1.10

Related to RQ1, participants of the pre-survey were grouped based on professional identity (BG5). Moreover, while answering RQ2, non-parametric analyses of variance (Kruskal-Wallis) were used to investigate group differences in the previous knowledge and skills (BG1 - BG4). In the case of significant differences were observed, pairwise Man-Whitney tests were conducted with p-values adjusted by using the Benjamini-Hochberg method. The effect sized were estimated with Cliff's delta. Although all the items were related to the previous understanding of DevOps, they were not combined into a single factor because of relatively small correlations (Table 1). RQ3 was answered based on the open-ended answers in the post questionnaire that were coded manually by one of the authors. Coding was verified by an another person. Course staff was interviewed to triangulated the observations.

4 Results

4.1 Background and previous knowledge

629 participants answered something to the question about their professional status. The biggest groups were students (43%) and Developers/Software Engi-

neers (35%). Sizes of other groups are provided in Table 2. Professions in the category other included answers such as system administrator, consultant, network engineer, manager, and software tester. Group-wise medians and median absolute deviations of previous knowledge are also provided in Table 2.

Table 2. Sized of professional identity groups together with medians and median absolute deviations of background variables per groups

	n	BG1		BG2		BG3		BG4	
		med	mad	med	mad	med	mad	med	mad
Student	270	2.00	1.48	2.00	1.48	1.00	0.00	3.00	1.48
Developer	218	2.00	0.74	2.00	0.00	2.00	1.48	2.00	0.74
DevOps engineer	34	3.00	1.48	2.00	0.00	3.00	1.48	2.50	1.48
Data scientist	24	2.00	0.00	2.00	0.00	2.00	0.00	2.00	0.00
Other	83	2.00	1.48	2.00	1.48	2.00	1.48	2.00	1.48

Table 3. Dunn’s post-hoc (pairwise Mann-Whitney) analysis and possible effect sizes.

Comparison	BG1			BG2			BG3		
	Z	P.adj	d	Z	P.adj	d	Z	P.adj	d
Data scientist - DevOps eng	-2.756	.015	-0.38	-1.340	.360	-	-2.131	.041	-0.41
Data scientist - Other	-0.695	.696	-	-0.071	.943	-	1.549	.135	-
DevOps eng - Other	2.818	.024	0.31	1.673	.314	-	4.553	.000	0.52
Data scientist - Software eng	-1.026	.610	-	-0.434	.738	-	-0.288	.774	-
DevOps eng - Software eng	2.789	.018	0.28	1.431	.381	-	2.744	.010	0.29
Other - Software eng	-0.461	.716	-	-0.596	.788	-	-3.259	.003	-0.23
Data scientist - Student	-0.634	.658	-	0.549	.729	-	2.998	.005	0.38
DevOps eng - Student	3.297	.010	0.33	2.606	.092	-	6.632	.000	0.62
Other - Student	0.208	.835	-	1.063	.480	-	2.228	.037	0.16
Software eng - Student	0.941	.578	-	2.309	0.105	-	7.674	.000	0.38

Professional background affects general understand of containers - BG1 ($H(4)=12.03$, $p=.017$), understanding of problems that containers solve - BG2 ($H(4)=9.94$, $p=.041$), and whether authors have used the docker or not - BG3 ($H(4)=86.78$, $p<.000$). However, a professional background does not affect how familiar participants are with DevOps or whether they have a general understanding of what DevOps is - BG4 ($H(4)=6.00$, $p=.20$). The results of the post-hoc analysis, together with the effect sizes (d), are illustrated in Table 3.

4.2 Completers and Challenges

The most difficult topics mentioned in the feedback were networking (37 times), volumes (23), application frameworks (14), Nginx (16), Postgres (8), and CORS (8). Based on the feedback collected from the course staff, the usage of operating systems, namely Linux, was challenging at the beginning of the course as many students were not familiar with the usage of the command-line interface and, for example, creating small script files using bash. In addition, the students were requesting more realistic, production-like, environments, and tool sets to

practice. There was a demand for an additional part about cloud services such as AWS or Google Cloud and advanced tools such as Kubernetes. Distribution of the professional profiles at the end of the course is similar to the beginning of the course, and illustrated in Table 4.

Table 4. Professional identity of those who started vs. those completed the course

	Data scientist	DevOps eng	Other	Software eng	Student
pre-course	3.8%	5.4%	13.2%	34.7%	43.0%
post-course	1.4%	5.0%	8.6%	33.6%	51.4%

5 Discussion

One of the contributions of this paper is to introduce the DevOps with Docker course design. While DevOps imposes cultural changes to software development and organizational structure, we decided to approach the topic from the tools perspective. The decision is supported by the fact that main effort for DevOps to succeed is to make the overall process continuous [11]. We recognize that containers themselves do not offer a complete picture of DevOps, lacking key subjects such as monitoring. However, containers offer the means against a subset of the presented problems, such as "works on my machine". This divide between the development and production environments is a significant hurdle as the classic "works on my machine" divide opposes DevOps principles [2]. Moreover, the teaching of methods to solve this problem offers students the possibility to design development pipelines when confronted with such a task.

There were practically no differences between professional identity groups in self evaluated conceptual understanding of DevOps (BG4) or container (BG1 and BG2). Even those who identified themselves as DevOps engineers stood out from the crowds only in one questions (BG1) where the effect sizes were small. Surprisingly, students and software developers reported similar amounts of knowledge on containers. This similarity between the two groups indicates that there is a need for life long learning courses for software industry. It is unclear if software engineers attending the course are representative sample of developers in general. It is possible to argue that only developers lacking DevOps skills would take the course in the first place. However, even those who identified themselves as DevOps engineers (and were more skilled) came in the course.

One possible explanation for only small differences is that more you know, the better you understand what you don't know. This is supported by the fact that the differences in the more concretely framed question about how often participants have used Docker were pretty clear. As assumed, DevOps engineers had most experience in using the Docker, while students were the least experienced. The effect size between DevOps engineers and both software developers and data scientists is medium, no differences were observed between developers and data scientists.

Networking was the most commonly reported challenge by the participants. In contrast to the networking course where students learn, e.g., how the internet

works and the layers in the OSI model, the course we offered focused mostly on the transport layer with port configuration between applications and containers. This was challenging despite the theoretical understanding of the transport layer most university student attendees had from a mandatory networking course.

The perceived challenge in volume usage, the mapping of data in the container to the host, may also be partially explained by lack of experience with filesystems - linking it into the challenge of operating systems and CLI mentioned in other feedback. The third challenge was adapting to changing tools, services and application frameworks used as part of the exercises. To proceed in the challenging exercises attendees did not have to program in a new language but had to read the documentation, often a readme file, on what the exercise specific services required, such as open port or environmental values designating an API endpoint. For some attendees, this was a challenge as the errors encountered during these exercises were often unfamiliar at first but contained the necessary information to continue.

In the end, students requested advanced tools and more realistic context. Deployment in real cloud environments is different from simulating the same with containers in a local environment. The limited amount of realistic context in the course was identified as an area of improvement, and recommended earlier by Christensen as well [4]. Moreover, we believe that in advanced topics such as DevOps, authentic environments may be easier to master.

The threats to validity are the selection, maturation and attrition of the group. The course material was open to anyone regardless of signing up for the course and the surveys were an optional exercise during the course and as such the answers include attendees that did not sign up for the course as well as miss students that did take the course. Selecting only attendees who wanted to complete the optional survey may have an effect on the results. As attendees were able to progress the course at any pace, it was possible for them develop their skills independently of the course, for up to 9 months between the start of the course and the end. This may limit the amount of feedback on challenges of the earlier sections of the course. The attrition influences the number of answers between the beginning and ending surveys, but the distribution of professional identity remained closely the same between the surveys.

6 Conclusions

When answering our first research question (*what is the professional background of participants taking the course*), we found out that the DevOps course attracted especially those who identified themselves as Students or Software Engineers. While looking at the second question (*how does the professional background affect participants' perceived previous understanding of DevOps?*), we observed that self reported previous skills of these two groups were surprisingly similar.

Based on this, we conclude that there is a demand for lifelong learning around DevOps practices for software engineers as the relatively new methodology interests and challenges professionals working in the industry. For the last research

question (*the core challenges for students when learning DevOps*) multiple topics related to administration were identified. The multidisciplinary skills emphasis of DevOps requires an introduction in a more practical approach for otherwise familiar concepts. We believe that such skills are best learned in as authentic environments as possible.

References

1. ACM Computing Curricula Task Force (ed.): Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. <https://doi.org/10.1145/2534860>
2. Anderson, C.: Docker [Software engineering]. IEEE Software **32**(3), 102–c3 (May 2015). <https://doi.org/10.1109/MS.2015.62>
3. Bolscher, R., Daneva, M.: Designing software architecture to support continuous delivery and devops: A systematic literature review. In: 14th Int. Conf. Software Technologies (May 2019). <https://doi.org/10.5220/0007837000270039>
4. Christensen, H.B.: Teaching devops and cloud computing using a cognitive apprenticeship and story-telling approach. In: ACM Conf. Innovation and Technology in CS Education. p. 174–179 (2016). <https://doi.org/10.1145/2899415.2899426>
5. Haaranen, L., Lehtinen, T.: Teaching git on the side: Version control system as a course platform. In: Proc. ACM Conf. Innovation and Technology in Computer Science Education. pp. 87–92 (2015). <https://doi.org/10.1145/2729094.2742608>
6. Jabbari, R., bin Ali, N., Petersen, K., Tanveer, B.: What is devops? a systematic mapping study on definitions and practices. In: Proc. Scientific Workshop Proceedings of XP2016. ACM (2016). <https://doi.org/10.1145/2962695.2962707>
7. Kang, H., Le, M., Tao, S.: Container and Microservice Driven Design for Cloud Infrastructure DevOps. In: IEEE Int. Conf. Cloud Engineering (IC2E). pp. 202–211 (2016). <https://doi.org/10.1109/IC2E.2016.26>
8. Kuusinen, K., Albertsen, S.: Industry-Academy Collaboration in Teaching DevOps and Continuous Delivery to Software Engineering Students: Towards Improved Industrial Relevance in Higher Education. In: 41st Int. Conf. Software Engineering: Software Engineering Education and Training (2019). <https://doi.org/10.1109/ICSE-SEET.2019.00011>
9. Luukkainen, M., Vihavainen, A., Vikberg, T.: Three years of design-based research to reform a software engineering curriculum. In: Proc. 13th Conf. Information Technology Education. pp. 209–214 (2012). <https://doi.org/10.1145/2380552.2380613>
10. Pengxiang, J., Leong, P.: Teaching Work-Ready Cloud Computing using the DevOps approach. In: Int. Symposium Advances in Technology Education (2014)
11. Rajkumar, M., Pole, A.K., Adige, V.S., Mahanta, P.: DevOps culture and its impact on cloud delivery and software development. In: Int. Conf. Advances in Computing, Communication, Automation. pp. 1–6 (2016). <https://doi.org/10.1109/ICACCA.2016.7578902>
12. Spinellis, D.: Don't Install Software by Hand. IEEE Software **29**(4), 86–87 (2012). <https://doi.org/10.1109/MS.2012.85>
13. Syed, M.H., Fernandez, E.B.: A reference architecture for the container ecosystem. In: Proc. 13th Int. Conf. Availability, Reliability and Security. pp. 1–6. ACM (2018). <https://doi.org/10.1145/3230833.3232854>
14. Zhang, Y., Vasilescu, B., Wang, H., Filkov, V.: One size does not fit all: An empirical study of containerized continuous deployment workflows. In: 26th ACM Joint Meeting on European Software Engineering Conf. and Symposium on the Foundations of Software Engineering (2018). <https://doi.org/10.1145/3236024.3236033>