Containers in Software Development: A Systematic Mapping Study

Mikael Koskinen[1][0000-0003-2880-2809], Tommi Mikkonen[2][0000-0002-8540-9918] and Pekka Abrahamsson[1] [0000-0002-4360-2226]
[1] Faculty of Information Technology, University of Jyväskylä, Jyväskylä, Finland
[2] Department of Computer Science, University of Helsinki, Helsinki, Finland
mikael.koskinen@student.jyu.fi, pekka.abrahamsson@jyu.fi, tommi.mikkonen@helsinki.fi

Abstract. Over the past decade, continuous software development has become a common place in the field of software engineering. Containers like Docker are a lightweight solution that developers can use to deploy and manage applications. Containers are used to build both component-based architectures and microservice architectures. Still, practitioners often view containers only as way to lower resource requirements compared to virtual machines. In this paper, we conducted a systematic mapping study to find information on what is known of how containers are used in software development**.** 56 primary studies were selected into this paper and they were categorized and mapped to identify the gaps in the current research. Based on the results containers are most often discussed in the context of cloud computing, performance and DevOps. We find that what is currently missing is more deeply focused research.

Keywords: Containers, Software Engineering, Systematic Mapping Studies

1 Introduction

Over the past decade, continuous software development has become a common place in the field of software engineering. New toolchains have emerged to manage the complexity in continuous deployment activity. Containers are a lightweight solution that developers can use to deploy and manage applications [1]. Containers are often seen as a more light-weight alternative to Virtual Machines (VMs) [2]. Virtual Machines include the operating system where containers don't, allowing the containers to provide system resource usage advantages when compared against VMs [3].

The usefulness of containers is not limited to them being a more lightweight version of Virtual Machines. One interesting feature of the containers is that they provide portability [1] and thus modularity, making them suitable for working as software components [4] or as autonomous microservices [5]. When software systems grow, they encounter three problems:

1. Maintaining the software becomes harder
2. Adding new features to the system slows down
3. The resource requirements for the software grow

One option to address these problems is to make systems modular [6]. In modular systems software is split into smaller modules and the full software systems are built by combining different modules [7]. Component-based software architecture and microservice architecture allow developers to build more modular software [7, 8]. In component-based architecture systems are created by connecting different software components [9]. Components are required when the system is compiled, and they are loaded when the system starts. Because of this, component-based systems don't help with the growing resource requirements, but it makes maintaining the software easier.

Similar to components, microservices are autonomous services that together fulfill a business requirement [5]. Also, like component-based architecture, each microservice is required for the system to be fully functional. Since containers are not compiled as part of the software system, they could be used as way to build plug-in based architecture where containers-based plugins could provide new functionality into existing software and they could be added and removed runtime [10-12]. Based on our observation, containers are used to build both component-based architectures and microservice architectures [1, 5]. Still, containers are often viewed as way to lower resource requirements compared to Virtual Machines [3].

As using containers in software development is a new research area, the need for a systematic mapping study is crucial in order to summarize the progress so far and identify the gaps and requirements for future studies. In this paper we present a systematic mapping study of how containers are used in software development. In this research, we conducted a systematic mapping study to find information for the key question: What is currently known of **how containers are used in software development.** This paper is the first part of a larger study. The aim of this study is to learn if containers are used mainly as a lightweight replacement for the virtual machines or if their portability and low resource usage is used to build container-based software components. Next parts of this study will include a multi-vocal study [13] and a case-study [14].

The rest of this paper is structured as follows. Section 2 introduces the research methodology. Section 3 presents our key results. Section 4 provides discussion based on the results. Section 5 presents threats to validity of this research. Section 6 draws conclusions.

2 Research Methodology

Systematic Mapping Study (SMS) [15] is used in this paper to identify the gaps in the literature and identify where new or better primary studies are needed for using containers. This paper follows systematic mapping guidelines provided by [15-18].

The process of systematic mapping study can be split into multiple phases:

1. Defining the research questions
2. Conducting search
3. Study selection (Screening the papers)
4. Defining the classification scheme
5. Data extraction

6. Systematic mapping of the data using the classification scheme

The following figure 1 illustrates the process of this systematic mapping study:

Fig. 1. Systematic Mapping Study Process

The following sections are used to describe SMS from this study's perspective.

2.1 Definition of research questions

First task was the definition of research question (RQ). The research questions are listed in Table 1.

Table 1. Research questions

RQ number	Question	Motivation
RQ 1	How Containers are used in Software Development?	The question allows us to get the what is known of how containers are used in software development. What technologies are used and what software development problems are containers used to tackle.
RQ 1.1	Are containers used to modularize software system, either through component-based architecture or through microservices architecture?	Based on our observation, containers could be used to architecture software systems. Still, the practitioners mostly seem to discuss containers as a technology for handling software's infrastructure.
RQ 1.2	Are containers used to provide plugin-support for software systems?	Based on our observation, containers could be used to extend existing plugin-architecture based software systems.

2.2 Conduct Search

After defining the research questions relevant search terms and data sources were defined.

Search terms. Without correct search terms correct literature and research cannot be found. Table 2 lists the search terms used in this study. The following steps were used to create the search terms, as defined in [19]:

- Derive major terms from the questions by identifying the population, intervention and outcome.
- Identify alternative spellings and synonyms for major terms.
- Check the keywords in any relevant papers we already have.
- Use the Boolean OR to incorporate alternative spellings and synonyms.
- Use the Boolean AND to link the major terms from population, intervention, and outcome.

Table 2. Search terms

"container" OR "containers" OR "docker" OR "Kubernetes"
AND
"software engineering" OR "software design"

Data sources and search criteria. For this research only formal data sources were considered. These included papers and journals from the four digital libraries: IEEExplorer, ScienceDirect, SpringerLink and ACM Digital Library. The reason for selecting these sources is that they are important sources of computer science related research. Search terms were matched against title, keywords and abstract.

The search was performed between 22th of May and 5th of June in 2019. In total 3504 results were found. Results were exported into bibtex-format and loaded into reference manager.

Table 3. Results before study selection process

Library	Results
IEEEXplorer	120
ScienceDirect	1095
SpringerLink	889
ACM Digital Library	1400

Study Selection. After finding the initial results, the next phase of the SMS was study selection. The main goal of the study selection is to find select relevant studies that properly address the research questions. As displayed in Table 4, in this study 5 inclusion criteria and 7 exclusion criteria were used.

Table 4. Study selection criteria

Inclusion criteria	Exclusion criteria
• Studies that are presented as full paper. • Studies that focus on using modern containers in software development.	• Studies that are duplicate • Studies that are presented as short paper. • Studies that do not provide abstract

• Studies that compare containers and virtualization. • Studies that are related to Docker. • Studies that are related to Kubernetes	• Studies that are not peer-reviewed • Studies that are not written in English • Studies that are not related to the software engineering. • Studies that are not related to modern Docker-style containers. For example, articles related to Java containers or Inversion of Control Containers.

Of the 3504 results, 60 were removed as duplicates. Two-step selection process was used to filter out the irrelevant studies for this paper. First of each study the title was reviewed using inclusion and exclusion criteria. Each excluded study was marked as such. After this step, 3308 studies were filtered out and the second step was applied to the remaining 136 studies. In this step of each study abstract was skimmed through. In this second step, 80 studies were excluded.

In total, 56 studies [20-75] were selected as the primary studies of this paper.

Classification schema. The selected primary studies and the research questions were used to create the classification scheme for this study. Based on a qualitative assessment, research classification approach from [76] was used to classify the papers. The classifications are listed in more detail in table 5.

Table 5. Research type facet adapted from [76]

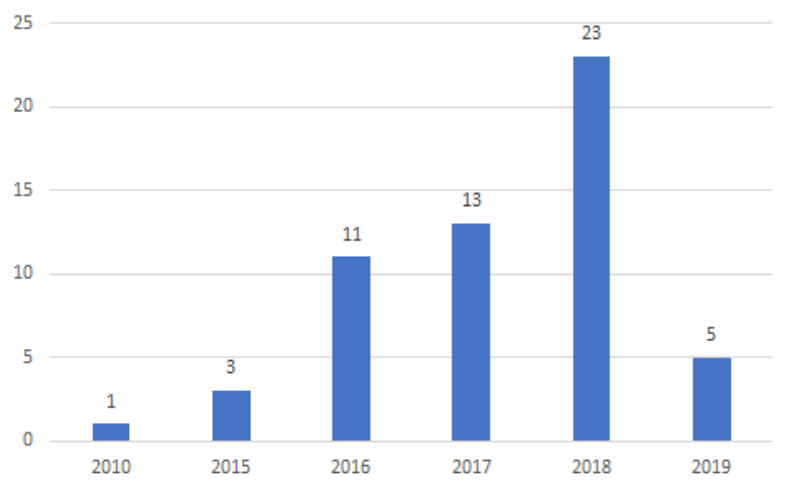
Research type	Description
Evaluation research	Type of paper which investigates a problem in practice.
Solution proposal	A paper which presents a solution for a problem. Benefits of the solution are described.
Validation research	Paper which investigates the properties of a solution that has not yet been implemented.
Experience report	Paper based on work done in practice. Describes what and how something has been done personally by the author.
Opinion	Paper based on the opinion of the author. Opinion articles do not rely on research methodology.

Data extraction. After using the primary studies and the research questions to create the classification schema, relevant data was extracted from the studies based on the classification schema. Title, author (first), year of publication, keywords, abstract and research type were extracted from each paper.

3 Results

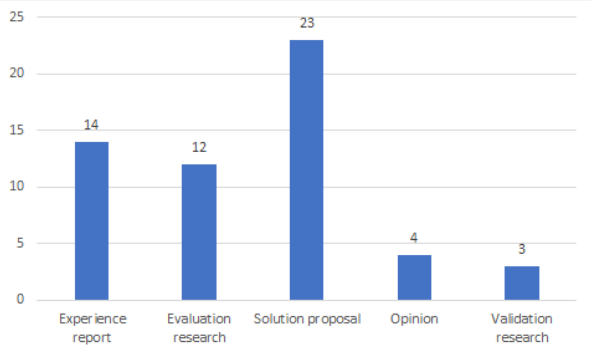
In this section the results are presented found in this mapping study are presented. Of the initial amount of 3504 papers, 56 were selected as the primary papers for this study.

Fig. 2. Articles by year



Papers were mapped into the classification schema presented earlier in this study. The results presented in Figure 3 of this mapping indicate that solution proposal is the most common paper when containers are discussed.

Fig. 3. Paper research types



Experience reports and evaluation research complete the top 3 of research types. Also, few validation research and opinion papers were found. Next, results are validated against the research questions.

3.1 RQ 1 How are Containers used in Software Development?

First research question was set to assess how containers are used in software development. The initial opinion of this study was that containers are often used as a lightweight alternative to virtual machines.

Keywords were extracted from each article's title and abstract. These keywords were then grouped together into different categories which were identified by generalizing the keywords. Table 6 presents the list of generalized categories. Each study belongs to one or more categories.

Table 6. Categories

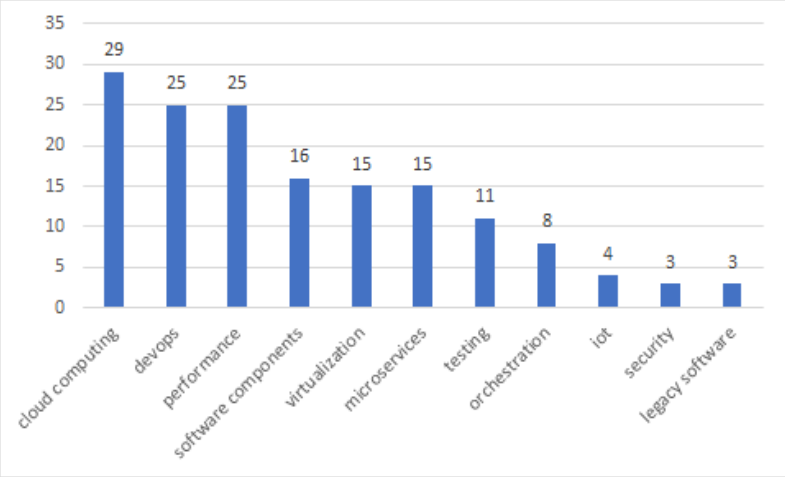
Focus	Keywords
Software Components	Modules, Packages, Artifacts, Bundle, Component
Cloud Computing	Cloud, PaaS, SaaS, Cloud Infrastructure, Cloud environment, Cloud platforms
DevOps	DevOps, CI, CD
Performance	Scalability, I/O, CPU, Scaling, Replication, resources, GPU, Resource contention, performance
Security	Security, Password, Secure
Microservices	Microservice-architecture, Microservices, Micro-service
Legacy Software	Modernization, Legacy
Orchestration	Orchestration, Docker Swarm, Kubernetes
Testing	Testing, Benchmark, Software Quality
IoT	IoT, Internet of Things
Plugin	Plugins, Addon, Extensions
Virtualization	Virtualization, Virtual Machine, VM

Based on the results, containers are most often discussed in relation to cloud computing, performance and devops (Figure 4). More than 50% of the papers discussed containers in context of cloud computing. Performance related aspects and devops discussed in 45% of the papers. Most of the papers do not focus on a single category. Instead, only 13 papers belong to a single category as shown in Table 7.

Table 7. Number of categories and number of papers

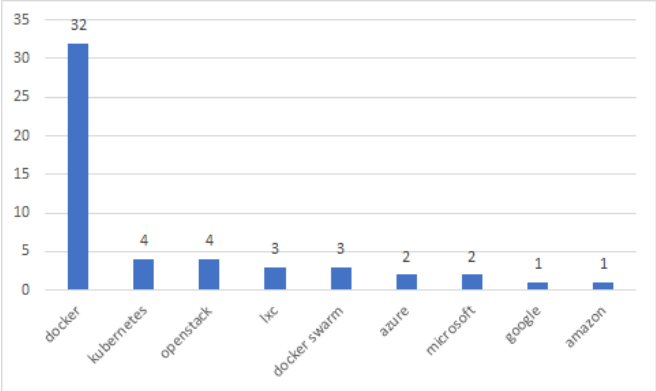
# of categories	# of papers
2	14
3	14
1	13
4	8
6	4
5	3

Fig. 4. Articles by categories



If we look at specific technologies (Figure 5) and companies discussed in the papers, we can see that Docker dominates the field. More than 57% of articles mention Docker in their abstract or in their title.

Fig. 5. Articles by Container Technology or Organization



3.2 RQ 1.1 Are containers used to modularize software system, either through component-based architecture or through microservices architecture?

The motivation of the first sub research question was to find out if containers are discussed in relation of software architecture. 16 of the 56 papers discuss containers from software component's point of view. Also, microservices are discussed in 15 papers (Figure 4). This clearly indicates that containers used to modularize software

system, either through component-based architecture or through microservices architecture.

3.3 RQ 1.2 Are containers used to provide plugin-support for software systems?

The motivation of the second sub research question was based on our observation that containers could be used to extend existing plugin-architecture based software systems. Even though 20% of the articles mentioned software components, we didn't find any indications that containers are used to create plugin-based software architecture.

4 Discussion

The implications of this systematic mapping study are described in the following sub sections.

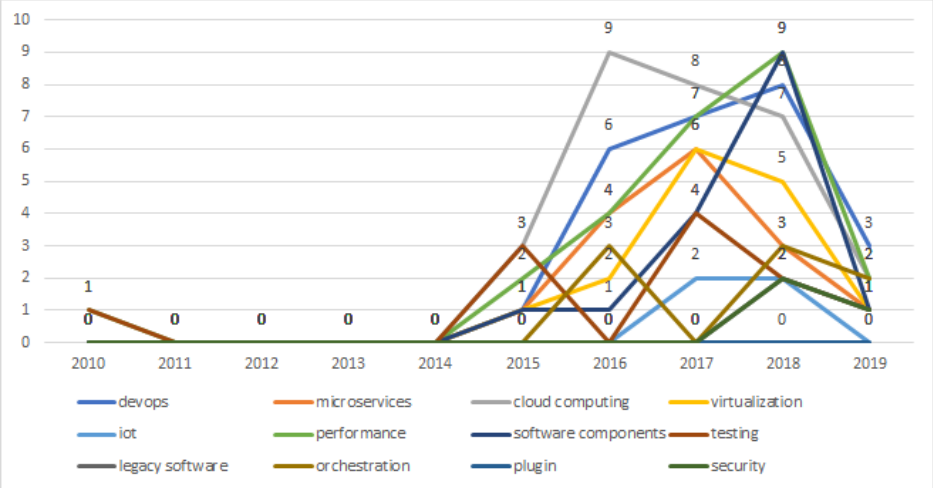
4.1 Research in using containers in software development

Results indicate that the number of container related articles is growing (Figure 2). 70% of the studies have been released between 2017 and 2019. There are multiple indicators that research on using containers in software development is a new research area:

1. First primary study found for this research is from 2010.
2. Number of research papers is rapidly growing.
3. Current research often covers multiple software development categories instead of focusing into a single category.
4. Research papers often start by describing what software containers are. This is an indication that the technology is seen as new by researchers and an introduction to the technology is required.
5. Most of the research focuses on a single container technology, Docker.

In summary it can be said that containers are a new research area. The amount of research has been growing steadily and there's no indication that in 2019 research related to containers is going to slow down.

Fig. 6. Trends of using containers in software development



4.2 More focused research

Only 13 of the selected 56 primary studies focus their research on one category. 52% of the primary studies are related to three or more categories. It's clear that there is room for more focused research. Many of the categories are large topics and instead of research covering multiple large categories, research could focus on a single category like container security, container performance and using containers for devops.

4.3 Potential research avenues

As seen in Figure 4, cloud computing, devops and performance related discussion are most common in current container research. There are multiple gaps or less-researched categories which provide potential research avenues:

- Container security
- Legacy applications and containers
- Container-based plugin technologies

Solution proposals, experience reports and evaluation research are currently the most popular research types. Together they make 88% of the primary studies selected for this research. This may indicate that containers are currently used to solve existing problems related to software development. The lack of validation research supports this as validation research could be used to test new ideas.

Figure 5 shows that Docker is the dominant technology used in research. Even though there are studies like [74] which compare Docker to other container technologies, there's room for more research. Best practices-based papers are helpful for the

industry: they help those organization who are already using containers and those who are just starting to use them. Only [35] provides best practices of using containers.

5 Threats to validity

In this section the threats of validity of this research are discussed. Also selected mitigation strategies are discussed. Three potential threats of validity were identified:

Search. This study is based on the search results provided by research databases and their search engines. Because of this, the results are subject to the limitations of the search engines. We mitigated this by using four different research databases.

The keywords selected for this study are subject to search term bias. Two different container related technologies were named in the search terms and this may have affected search results, causing these two technologies to be more prevalent in the search results. Search term bias was mitigated by including generic search terms.

Identification of the primary studies. The selected inclusion and exclusion criteria listed in Table 4 may have affected the identification of the primary studies. For example, only papers written in English were selected. Also, not all the studies related to containers in software development are available from the used research databases. Risk of excluding primary studies was mitigated by using multiple research databases.

Data extraction. Categories in chapter 7 were selected by the researcher after keywords were extracted. Researcher acknowledges that if there are errors in keyword extraction, this may invalidate the categorization of the keywords. To mitigate the keyword extraction and categorization, keywords were extracted multiple times and the selected categories were identified only after keyword extraction.

6 Conclusion

This paper is a part of larger study. The aim of the study is to learn if containers are used mainly as a lightweight replacement for the virtual machines or if their portability and low resource usage is used to build container-based software components. In this paper a systematic mapping was performed to examine what is known of how containers are used in software development. The next part of this research is a multivocal study. The research will conclude with a case study.

Four research databases were used to locate 3504 papers of which 56 were selected as the primary studies. The results indicate that cloud computing, devops and performance are the driving forces of container related discussion. Of the 56 primary studies 52% discussed cloud computing, 48% performance and 45% devops. Docker is currently the leading technology in container-based software development. 57% of the papers mentioned Docker in their title or in their abstract. Other container related technologies were mentioned at most in 7% of the papers.

As an answer to RQ 1.1, 55% of the primary studies mentioned software components or microservices. This clearly indicates that containers are used to modularize software system, either through component-based architecture or through microservices architecture. As the examination of RQ 1.2 indicated, no papers discussing the usage of containers for plugin-based architectures were found.

The findings of this paper indicate that using containers in software development is a new research area. Most of the studies don't focus on a single software development category. Instead, they often present introduction on what containers are, clearly indicating that software containers are seen as a new technology. Also, best practices-based research is not yet widely available.

References

1. Paraiso F, Challita S, Al-Dhuraibi Y et al (Jun 2016) Model-Driven Management of Docker Containers. In: Anonymous IEEE, p 718-725.
2. Dua R, Raja AR, Kakadia D (2014) Virtualization vs Containerization to Support PaaS. In: Anonymous IEEE Computer Society, Washington, DC, USA, p 610–614.
3. Hoenisch P, Weber I, Schulte S et al (2015) Four-fold auto-scaling on a contemporary deployment platform using docker containers.
4. Kung-Kiu Lau, Zheng Wang (2007) Software Component Models. TSE 33(10):709-724. doi:10.1109/TSE.2007.70726.
5. Jaramillo D, Nguyen DV, Smart R (Mar 2016) Leveraging microservices architecture by using Docker technology. In: Anonymous IEEE, p 1-5.
6. Woodfield SN, Dunsmore HE, Shen VY (1981) The Effect of Modularization and Comments on Program Comprehension. In: Anonymous IEEE Press, Piscataway, NJ, USA, p 215–223.
7. Card DN, Page GT, McGarry FE (1985) Criteria for Software Modularization. In: Anonymous IEEE Computer Society Press, Los Alamitos, CA, USA, p 372–377.
8. Völter M PluggableComponent – A Pattern for Interactive System Configuration.
9. Crnkovic (2003) Component-based software engineering ? new challenges in software development.
10. Birsan D (2005) On Plug-ins and Extensible Architectures. Queue 3(2):40–46. doi:10.1145/1053331.1053345.
11. Mayer J, Melzer I, Schweiggert F (2003) Lightweight Plug-In-Based Application Development.
12. Marquardt K (1999) Patterns for Plug-Ins. EuroPLoP.
13. Garousi V, Felderer M, Mäntylä MV (2019) Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. Information and Software Technology 106:101-121. doi:10.1016/j.infsof.2018.09.006.
14. Eisenhardt K (1989) Building Theory From Case Study Research. The Academy of Management Review 14:532-550. doi:10.2307/258557.
15. Petersen K, Feldt R, Mujtaba S et al (2008) Systematic Mapping Studies in Software Engineering. In: Anonymous BCS Learning & Development Ltd., Swindon, UK, p 68–77.
16. A. Kitchenham B (2007) Kitchenham, B.: Guidelines for performing Systematic Literature Reviews in software engineering. EBSE Technical Report EBSE-2007-01.
17. Kitchenham B, Charters S (2009) Systematic reviews. https://www.york.ac.uk/crd/guidance/.

18. Kitchenham B, Brereton P (2013) Using Mapping Studies in Software Engineering. Information and Software Technology 55(12):2049-2075. doi:10.1016/j.infsof.2013.07.010.
19. Kitchenham BA, Mendes E, Travassos GH (2007) Cross versus Within-Company Cost Estimation Studies: A Systematic Review. TSE 33(5):316-329. doi:10.1109/TSE.2007.1001.
20. Stillwell M, Coutinho JGF (2015) A DevOps approach to integration of software components in an EU research project. In: Anonymous Proceedings of the 1st International Workshop on Quality-Aware DevOps ACM, New York, NY, USA, p 1-6.
21. Tuo F, Bai Y, Long S et al (2018) A New Model of Docker-based E-learning in Hadoop. In: Anonymous Proceedings of the 2018 International Conference on Distance Education and Learning - ICDEL '18 ACM Press, New York, New York, USA, p 22-31.
22. Kozhirbayev Z, Sinnott RO (2017) A performance comparison of container-based technologies for the Cloud. Future Generation Comput Syst 68:175-182. doi:10.1016/j.future.2016.08.025.
23. Telschig K, Schonberger A, Knapp A (2018) A Real-Time Container Architecture for Dependable Distributed Embedded Applications. In: Anonymous 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE), aug, vol 2018-Augus. IEEE, p 1367-1374.
24. Syed MH, Fernandez EB (2018) A reference architecture for the container ecosystem. In: Anonymous Proceedings of the 13th International Conference on Availability, Reliability and Security ACM, New York, NY, USA, p 1-6.
25. Rahman M, Chen Z, Gao J (2015) A service framework for parallel test execution on a developer's local development workstation. In: Anonymous Proceedings - 9th IEEE International Symposium on Service-Oriented System Engineering, IEEE SOSE 2015, vol 30., p 153-160.
26. Kratzke N (2018) About the complexity to transfer cloud applications at runtime and how container platforms can contribute?. In: Ferguson D, Mu{ ~{n}}oz, V{ '{i}}ctor M{ '{e}}ndez, Cardoso J et al (eds) Communications in Computer and Information Science, vol 864. Springer International Publishing, Cham, p 19-45.
27. Song M, Zhang C, Haihong E (2018) An Auto Scaling System for API Gateway Based on Kubernetes. In: Anonymous 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), p 109-112.
28. Cito J, Schermann G, Wittern JE et al (2017) An Empirical Analysis of the Docker Container Ecosystem on GitHub. In: Anonymous IEEE International Working Conference on Mining Software Repositories IEEE Press, Piscataway, NJ, USA, p 323-333.
29. Zhang Y, Yin G, Wang T et al (2018) An Insight Into the Impact of Dockerfile Evolutionary Trajectories on Quality and Latency. In: Anonymous 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), jul, vol 1. IEEE, p 138-143.
30. Naughton T, Sorrillo L, Simpson A et al (2018) Balancing performance and portability with containers in HPC: An OpenSHMEM example. In: Gorentla Venkata Manjunath, Imam N, Pophale S (eds) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol 10679 LNCS. Springer International Publishing, Cham, p 130-142.
31. Naik N (2016) Building a virtual system of systems using docker swarm in multiple clouds. In: Anonymous ISSE 2016 - 2016 International Symposium on Systems Engineering - Proceedings Papers, p 1-3.
32. Shah J, Dubaria D (2019) Building Modern Clouds: Using Docker, Kubernetes \& Google Cloud Platform. In: Anonymous 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), jan IEEE, p 184.

33. Klinaku F, Frank M, Becker S (2018) CAUS: An Elasticity Controller for a Containerized Microservice. In: Anonymous Companion of the 2018 ACM/SPEC International Conference on Performance Engineering ACM, New York, NY, USA, p 93-98.
34. Kehrer S, Riebandt F, Blochinger W (2019) Container-based Module Isolation for Cloud Services. In: Anonymous 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE), p 177-186.
35. Berger C, Nguyen B, Benderius O (2017) Containerized development and microservices for self-driving vehicles: Experiences \& best practices. In: Anonymous Proceedings - 2017 IEEE International Conference on Software Architecture Workshops, ICSAW 2017: Side Track Proceedings, p 7-12.
36. Sharma P, Chaufournier L, Shenoy P et al (2016) Containers and Virtual Machines at Scale: A Comparative Study. In: Anonymous Proceedings of the 17th International Middleware Conference ACM, New York, NY, USA, p 1:13.
37. R 'ev 'esz, 'Ad 'am, Pataki N (2019) Continuous A/B Testing in Containers. In: Anonymous Proceedings of the 2019 2nd International Conference on Geoinformatics and Data Analysis - ICGDA 2019 ACM, New York, NY, USA, p 11-14.
38. Barna C, Khazaei H, Fokaefs M et al (2017) Delivering Elastic Containerized Cloud Applications to Enable DevOps. In: Anonymous Proceedings of the 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems IEEE Press, Piscataway, NJ, USA, p 65-75.
39. Bahadori K, Vardanega T (2019) Devops meets dynamic orchestration. In: Bruel J, Mazzara M, Meyer B (eds) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol 11350 LNCS. Springer International Publishing, Cham, p 142-154.
40. Dhakate S, Godbole A (2015) Distributed cloud monitoring using Docker as next generation container virtualization technology. In: Anonymous 2015 Annual IEEE India Conference (INDICON), p 1-5.
41. Naik N (2017) Docker container-based big data processing system in multiple clouds for everyone. In: Anonymous 2017 IEEE International Symposium on Systems Engineering, ISSE 2017 - Proceedings, p 1-7.
42. Martin A, Raponi S, Combe T et al (2018) Docker ecosystem – Vulnerability Analysis. Comput Commun 122:30-43. doi:10.1016/j.comcom.2018.03.011.
43. Nardelli M, Hochreiner C, Schulte S (2017) Elastic Provisioning of Virtual Machines for Container Deployment. In: Anonymous Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion ACM, New York, NY, USA, p 5-10.
44. Fokaefs M, Barna C, Veleda R et al (2016) Enabling Devops for Containerized Data-intensive Applications: An Exploratory Study. In: Anonymous Proceedings of the 26th Annual International Conference on Computer Science and Software Engineering IBM Corp, Riverton, NJ, USA, p 138-148.
45. Santos EA, McLean C, Solinas C et al (2018) How does docker affect energy consumption? Evaluating workloads in and out of Docker containers. J Syst Software 146:14-25. doi:10.1016/j.jss.2018.07.077.
46. Zhu H, Bayley I (2018) If Docker is the Answer, What is the Question?. In: Anonymous 2018 IEEE Symposium on Service-Oriented System Engineering (SOSE), mar IEEE, p 152-163.
47. Casalicchio E, Perciballi V (2017) Measuring Docker Performance: What a Mess!!!. In: Anonymous Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion ACM, New York, NY, USA, p 11-16.

48. Guo D, Wang W, Zeng G et al (2016) Microservices architecture based cloudware deployment platform for service computing. In: Anonymous Proceedings - 2016 IEEE Symposium on Service-Oriented System Engineering, SOSE 2016, p 358-364.
49. Shadija D, Rezai M, Hill R (2017) Microservices: Granularity vs. Performance. In: Anonymous Companion Proceedings of the10th International Conference on Utility and Cloud Computing ACM, New York, NY, USA, p 215-220.
50. Naik N (2016) Migrating from Virtualization to Dockerization in the Cloud: Simulation and Evaluation of Distributed Systems. In: Anonymous Proceedings - 2016 IEEE 10th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Environments, MESOCA 2016, p 1-8.
51. Balalaie A, Heydarnoori A, Jamshidi P (2016) Migrating to Cloud-Native Architectures Using Microservices: An Experience Report. In: Celesti A, Leitner P (eds) Advances in Service-Oriented and Cloud Computing Springer International Publishing, Cham, p 201-215.
52. Xu T, Marinov D (2018) Mining Container Image Repositories for Software Configuration and Beyond. In: Anonymous Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results ACM, New York, NY, USA, p 49-52.
53. Ferrer AJ, P 'erez, David Garc 'ia, Gonz 'alez, Rom 'an Sosa (2016) Multi-cloud Platform-as-a-service Model, Functionalities and Approaches. Procedia Computer Science 97:63-72. doi:10.1016/j.procs.2016.08.281.
54. Zhang Y, Vasilescu B, Wang H et al (2018) One size does not fit all: an empirical study of containerized continuous deployment workflows. In: Anonymous Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering ACM, New York, NY, USA, p 295-306.
55. Yarygina T, Bagge AH (2018) Overcoming Security Challenges in Microservice Architectures. In: Anonymous 2018 IEEE Symposium on Service-Oriented System Engineering (SOSE), mar IEEE, p 11-20.
56. Lv K, Zhao Z, Rao R et al (2016) PCCTE: A portable component conformance test environment based on container cloud for avionics software development. In: Anonymous 2016 International Conference on Progress in Informatics and Computing (PIC), p 664-668.
57. Wang B, Song Y, Cui X et al (2017) Performance comparison between hypervisor- and container-based virtualizations for cloud users. In: Anonymous 2017 4th International Conference on Systems and Informatics (ICSAI), nov IEEE, p 684-689.
58. Heinrich R, van Hoorn A', Knoche H et al (2017) Performance Engineering for Microservices: Research Challenges and Directions. In: Anonymous Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion ACM, New York, NY, USA, p 223-226.
59. Jindal A, Podolskiy V, Gerndt M (2019) Performance Modeling for Cloud Microservice Applications. In: Anonymous Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering ACM, New York, NY, USA, p 25-32.
60. Siami Namin Akbar, Sridharan M, Tomar P (2010) Predicting Multi-core Performance: A Case Study Using Solaris Containers. In: Anonymous Proceedings of the 3rd International Workshop on Multicore Software Engineering ACM, New York, NY, USA, p 18-25.
61. Hassan F, Rodriguez R, Wang X (2018) RUDSEA: Recommending Updates of Dockerfiles via Software Environment Analysis. In: Anonymous Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering ACM, New York, NY, USA, p 796-801.

62. Gogouvitis SV, Mueller H, Premnadh S et al (2018) Seamless computing in industrial systems using container orchestration. Future Generation Comput Syst. doi:10.1016/j.future.2018.07.033.
63. Goldschmidt T, Hauck-Stattelmann S (2016) Software Containers for Industrial Control. In: Anonymous Proceedings - 42nd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2016, p 258-265.
64. Yin K, Chen W, Zhou J et al (2018) STAR: A Specialized Tagging Approach for Docker Repositories. In: Anonymous 2018 25th Asia-Pacific Software Engineering Conference (APSEC), dec IEEE, p 426-435.
65. Benni B, Mosser S', Collet P et al (2018) Supporting Micro-services Deployment in a Safer Way: A Static Analysis and Automated Rewriting Approach. In: Anonymous Proceedings of the 33rd Annual ACM Symposium on Applied Computing ACM, New York, NY, USA, p 1706-1715.
66. Ye F, Jing Z, Huang Q et al (2018) The Research of a Lightweight Distributed Crawling System. In: Anonymous 2018 IEEE 16th International Conference on Software Engineering Research, Management and Applications (SERA), jun IEEE, p 200-204.
67. Oh J, Kim S, Kim Y (2018) Toward an Adaptive Fair GPU Sharing Scheme in Container-Based Clusters. In: Anonymous 2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W), p 79-85.
68. L 'opez, Manuel Ram$ backslash$'$ backslash$irez, Spillner J (2017) Towards Quantifiable Boundaries for Elastic Horizontal Scaling of Microservices. In: Anonymous Companion Proceedings of the10th International Conference on Utility and Cloud Computing ACM, New York, NY, USA, p 35-40.
69. Morris D, Voutsinas S, Hambly NC et al (2017) Use of Docker for deployment and testing of astronomy software. Astronomy and Computing 20:105-119. doi:10.1016/j.ascom.2017.07.004.
70. Punjabi R, Bajaj R (2016) User stories to user reality: A DevOps approach for the cloud. In: Anonymous 2016 IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT), p 658-662.
71. Senington R, Pataki B, Wang XV (2018) Using docker for factory system software management: Experience report. Procedia CIRP 72:659-664. doi:10.1016/j.procir.2018.03.173.
72. Knoche H, Eichelberger H (2018) Using the Raspberry Pi and Docker for Replicable Performance Experiments: Experience Paper. In: Anonymous Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering ACM, New York, NY, USA, p 305-316.
73. Morabito R (2017) Virtualization on internet of things edge devices with container technologies: A performance evaluation. IEEE Access 5:8835-8850. doi:10.1109/ACCESS.2017.2704444.
74. Tesfatsion SK, Klein C, Tordsson J (2018) Virtualization Techniques Compared: Performance, Resource, and Power Usage Overheads in Clouds. In: Anonymous Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering ACM, New York, NY, USA, p 145-156.
75. Ueda T, Nakaike T, Ohara M (2016) Workload characterization for microservices. In: Anonymous 2016 IEEE International Symposium on Workload Characterization (IISWC), sep IEEE, p 1-10.
76. Wieringa R, Maiden N, Mead N et al (2005) Requirements Engineering Paper Classification and Evaluation Criteria: A Proposal and a Discussion. Requir. Eng. 11(1):102?107. doi:10.1007/s00766-005-0021-6.