JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Survey on Dockerfile and Docker-Compose Development Challenges

[1]Thara D K,[2]Bhoomika T N

[1]Professor and Head,Department of ISE ,CIT,Gubbi,Tumakuru
[2]Student,Department of ISE , CIT, Gubbi, Tumakuru

Abstract :**The advent of cloud computing and Infrastructure-as-Code enabled with open-source technologies like Docker has revolutionized the world of software development. Still, the human-centric aspect of Docker development is understudied. In this work, we investigated the experiences and difficulties of developers as they compose Dockerfiles and docker-compose.yml files. By conducting an online survey among 68 graduate students and 120 professional developers, we learned that most of the developers consider Dockerfile composition as time-consuming with a higher burden for beginners. Solutions to problems are often by tweaking and testing, and many developers do not use supportive tools. These findings urge for consideration of human factors in Docker development and manifest potential directions for improving supportive tools. Fathoming developer perspectives may result in a more seamless user adoption and utilization of containerization technologies in software development.**

IndexTerms **-Docker, docker-compose, orchestration, cloud computing, survey.**

I. INTRODUCTION

The scene of program advancement has been revolutionarily modified amid later a long time due to the expansion of containerization innovations, counting Docker, LXC, and Kubernetes. These advances drop beneath the scope of Foundation as Code and have started to shape hones of advanced advancement as the dominance of cloud computing has expanded and collaboration between improvement and operations groups was considered significant. Docker, as one of these advances, has overseen to set up itself as the standard of containerization and advanced a unused worldview by intensely affecting the concept of virtualization with a more lightweight and dexterous elective of a containerized environment. The center include that characterizes Docker is the utilization of domain-specific dialects for characterizing an environment encapsuled inside a holder. This gives a entire modern meaning to virtualization, turning the prepare much more light and effective. And what is most imperative, Docker empowers extraordinary dialects to characterize these holders, conveying an fabulous opportunity to make, convey, and oversee applications all through differing situations. Which is truly huge since it gives the makers a chance to squander less time on compatibility issues and offer assistance create cool things. In any case, in spite of the ubiquity of Docker and the accessibility of numerous apparatuses that offer assistance with all infrastructure-related stuff, there is small observational prove as to how these apparatuses offer assistance. And that is the major objective of our think about. We in fact need to discover out how program experts utilize Docker and its mate, Docker-Compose.

II. RELATED WORK

In this ocean we embark on a journey into the uncharted territories of ship stories and their musicians, where empirical studies are still scarce, and solutions to developmental challenges are few Together we explore some projects that it's wonderfully illuminating on these issues, and we take a much broader approach to examining them. Guerriero et al. [8] conducted survey interviews with 44 practitioners, revealing both effective and conflicting practices in infrastructure-as-code (IaC) specification and found a lack of resources for continuous quality improvement, a testing and understanding which often leads to difficulties. Dalla Palma and others. [13] introduced 46 metrics to assess the quality of IaC specifications, suggesting future empirical research to validate these metrics. opportunities and so on. [14] analyzed Stack Overflow data, revealed the most common Docker-related questions clustered around application development, configuration, and debugging, and revealed areas of struggle for developers Sito and others. [15] analyzed Dockerfiles from GitHub, revealed common quality issues such as build errors and insufficient definitions, and identified a plethora of challenges in Dockerfile development Rahman and others. [16] analyzed IaC texts in organizations, identified syntax and design errors, and highlighted common pitfalls in IaC development, despite repeated contradictory models Ibrahim [19] analyzed the implementation of Docker-Compose, revealing that it is not very efficient in its implementation, many projects did not use the advanced features properly. Collectively, these studies highlight the need to examine technical challenges in container orchestration specification development, providing valuable insights into uncharted territories.

III. GOALS OF THE STUDY

We recommend that developers often take a cyclical path of trial and error when setting up a Docker environment. This includes modifying Dockerfiles and other specifications, building them into images, adding containers, and troubleshooting when things don't go as planned. The process is repetitive, with developers frequently revisiting and revising specifics. Similarly, there are research efforts to configure multiple containers with tools like docker-compose. Therefore, we believe that there is a need to improve the effectiveness of these development activities.

The goal of our survey is to unpack the complexities of identifying Docker containers and orchestras, shedding light on the most challenging aspects. We try to understand time-consuming projects and reveal the practical challenges faced by practitioners. Additionally, we aim to identify the tools and techniques developers use to find common and effective issues.
We pose three main research questions:
1. What activities are particularly time-consuming, identifying areas ripe for performance improvement?
2. What strategies do manufacturers use to troubleshoot and resolve problems while providing insights into potential support improvements or alternatives?
3. How do supporting software tools contribute to the development process, highlight gaps in platform support, and inform future tooling needs?

IV. METHODLOGY

We reached out to software engineers who had some familiarity with Docker technology through an online poll.We were able to collect data from a large number of developers through the survey, something that would be challenging to achieve with other exploratory research techniques like interviews, which demand a significant time commitment from the researchers. The Molléri et al. [22] checklist served as a reference and evaluation tool for the survey's design, implementation, and reporting.

A.Sampling and Recruitment process:

We searched forums and groups related to programming, DevOps, Docker, and general programming to get replies from developers who were familiar with Docker technology. Although practical, this method could have encouraged respondents to distribute the survey to others, which could have resulted in referral-chain sampling. We promoted the poll on a number of platforms, including Slack, Discord, Reddit, LinkedIn, Twitter, Facebook, and at the XP conference—a gathering that is well-liked by experts in Agile Software Development—in order to guarantee diversity. 120 answers came from these attempts. After gathering background data on the participants, we concentrated on evaluating replies related to Dockerfiles and docker-compose.yml files, limited to those who had prior expertise with these file formats. This yielded 119 and 107 useful responses, respectively. We gave early access to survey findings and a short film outlining the importance of the study as incentives for participation.

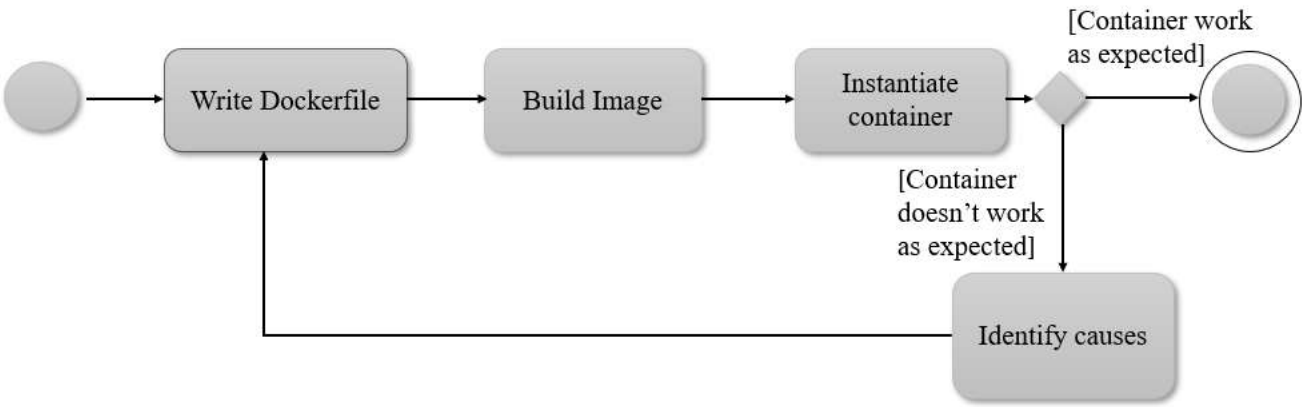

Fig1: Motivational example of a workflow when developing a Dockerfile.

B.INSTRUMENT DESIGN

The online form used to create the questionnaire was optimized to allow for a maximum response time of five minutes. In order to ensure objectivity and specificity, questions were carefully crafted and mostly used closed-ended forms such as five-point Likert scales or numerical inputs [24]. Open-ended inquiries were sporadically used to capture unanticipated problems or strategies.
It was divided into four primary elements: participant characterisation, identification of existing challenges, identification of current methods, and utilization of supplementary tools. Each of these areas was directly related to the study objectives presented in Section III.Two researchers with experience in Docker technologies piloted the poll before it was sent to professional developers in order to find any possible usability problems or ambiguities in the questions. The questionnaire's original and updated versions are both freely accessible to the public under a CC.

C. RESEARCH VARIABLES

In order to statistically investigate developer views on the amount of time spent on various development tasks, we in our study analyzed answers to closed-ended questions. We are able to obtain insights into how challenges and techniques differ with competence by segmenting the data according to the years of experience developers have with Docker and Docker-Compose. Years

of Docker technology experience are the independent factors. In order to address our main research issue, we want to determine which tasks take the most time. We also investigate open-ended topics to find new approaches and resources that developers employ to solve problems. These replies offer insights on problem-solving strategies and the tools developers use while developing new software, which aids in addressing secondary research issues.

V. DATA HANDLING

We managed the data by using Google Forms for the questionnaire; the answers were kept in a spreadsheet that the researchers could view. Names and other personal information were not required of responders, but if they wanted further research specifics, they may give their email addresses. We used the email addresses only to notify responders of research findings, completely according to data protection requirements. To ensure openness and support future study, we have also made the dataset and analytic scripts accessible as part of a replication package.

VI.ANALYSIS OF THE RESULT

A. PRELIMINARY RESULTS WITH STUDENTS.

We tested the study in advance using University of Porto graduate students pursuing an MSc in Informatics and Computing Engineering. 68 people responded to this round, most of them were novices with little knowledge of Dockerfile standards. According to preliminary results, just 4.4% of respondents said they used supplementary tools when working with Docker-Compose, however a sizable chunk of respondents (54%), spent a lot of effort figuring out why their Docker containers did not operate as planned. We were able to obtain important information for our research topics thanks to this early stage, which also allowed us prepare for the main study with specialists and improve our questionnaire.

B.PROFESSIONALEXPERIENCE

We administered three questions twice, once for Dockerfiles and once for docker-compose.yml files, in order to gauge the participants' professional knowledge with Dockerfile and docker-compose.yml development. The following inquiries reflected the independent variables in our analysis (see Section IV-C):

1.How much experience (in years) do you have working on projects that had a [Dockerfile/docker-compose.yml file]?

2.How much experience (in years) do you have working on projects where you have used [Dockerfiles/docker-compose.yml files] created by others (colleagues or third parties)?

3.How much experience (in years) do you have workingon projects where you created/updated a[Dockerfile/docker-compose.ymlfile]?

C.TIME-CONSUMING ACTIVITIES IN Dockerfile DEVELOPMENT

The creation of Dockerfiles entails a number of tasks, each of which developers estimate to take varying amounts of time. There are several hurdles associated with these tasks, ranging from diagnosing container misbehavior to reading Docker documentation. Finding system dependencies, verifying container operation, deciphering container misbehavior, and rebuilding containers following changes are among the tasks that respondents to our poll said took the longest. It's interesting to note that less experienced developers often find most tasks to take longer than their more experienced peers. Not all tasks, nevertheless, exhibit appreciable progress with practice. Even with increasing experience, tasks like determining system dependencies, verifying the functioning of containers, and recreating containers are still viewed as time-consuming.These observations assist in answering our initial study question by highlighting common problems and directions for future Dockerfile development. improvements to the tasks involved in troubleshooting malfunctioning

D.TIME-CONSUMING ACTIVITIES IN docker-compose.yml DEVELOPMENT

The study examined laborious tasks in the construction of Docker-compose.yml from the standpoints of writing and reading. Reading documentation, determining required keys and images, troubleshooting services, establishing properties, dependencies, volumes, networks, configurations, and secrets are some examples of writing-related duties. Understanding services, dependencies, volumes, and networks is necessary for reading activities. Overall, the results showed a neutral opinion, with a little lean towards agreement on tasks linked to debugging. Most jobs were viewed as taking less time by experienced developers than by novice ones. Debugging, documentation, and setup activities were particularly time-consuming for novice engineers. Tests of statistical significance verified notable variations according to expertise levels. Overall, setting and debugging tasks proved more difficult for less experienced engineers, indicating that these areas may use significant improvement. This is consistent with the knowledge that beginners frequently depend on trial-and-error techniques.

E. APPROACHES FOR DIAGNOSING AND CORRECTING PROBLEMS

Regarding Dockerfiles and Docker-Compose, the poll found no discernible differences in the approaches taken by seasoned engineers and less experienced developers to solve problems. Trial and error, web research, and manual command execution within containers were common methods for handling Dockerfiles. A few embraced organized approaches like as end-to-end testing and continuous integration. Respondents used comparable strategies for Docker-Compose, such as looking at output logs and running commands within containers. A few employed methodical techniques such as service isolation and dependency testing. These methods often need stepping outside of the writing context in order to obtain feedback, indicating possible advantages of techniques that allow feedback to be collected in the same setting without interfering with the process of creating specifications.

F. ANCILLARY SOFTWARE TOOLS

Regarding the use of tools or plugins other than general-purpose IDEs for developing Dockerfiles and docker-compose.yml, participants were questioned. It's interesting to note that replies from experienced and novice developers were comparable, suggesting that expertise levels had no effect on tool adoption. Most said they didn't use any auxiliary tools (78% for Dockerfiles, 83% for docker-compose.yml files). Those that did cited linters, syntax highlighters, and tools for managing containers and local images. These results imply that current technologies, which provide fundamental functions based on static analysis, are underused. There is a need for more advanced tools since specification creation is seen to be time-consuming, particularly for Dockerfiles. But in order to address this demand, technologies might need to go beyond static analysis's constraints.

VII.LIMITATIONSAND THREATS TO VALIDITY

1. Design of Questionnaire: Online surveys work well for simple inquiries, but they might not allow for thorough explanations. We included open-ended questions to address this, allowing for more in-depth answers without overwhelming participants.
2. Questionnaire Clarity: Measures were taken to guarantee the questionnaire's clarity, including testing it on researchers and students. Even yet, a tiny percentage of the sample said that they were confused by some of the questions.
3. Response Integrity: Respondents may provide false information or provide the same answer more than once. The majority of replies are thought to be sincere, even though no incentives were provided, therefore this hazard cannot entirely be eradicated.
4. Evaluating Developer Experience: Relying only on years of experience may not adequately represent skill. Nonetheless, reliable associations between experience-relatedinquiries and others point to a mostly accurate conclusion.
5. The Sample's Representativeness: Although a wide range of people could participate in the online survey, it could not accurately reflect all developers. Therefore, while broad findings probably represent the state of practice today, precise point estimates might not be entirely representative.
6. Extension to Additional IaCPlatforms:Because of the emphasis on Docker and Docker-Compose, the findings are not as broadly applicable to other Infrastructure as Code (IaC) systems. Docker-Compose is less common for large-scale deployments than Docker, even though Docker is extensively utilized, which might restrict the applicability of conclusions about docker-compose.yml files.

VIII. CONCLUSION

With an emphasis on Docker and Docker-Compose, the study described in this article attempts to provide light on the evolution of container and orchestration specifications. By covering particular uses of these technologies, it enhances earlier Infrastructure as Code (IaC) interviews. Our research reveals difficult parts of developing Dockerfiles; developers believe that debugging and dependency discovery take a lot of effort. Likewise, debugging becomes an issue while developing Docker-Compose, particularly for novice engineers. It's interesting to note that very few engineers integrate supplementary tools into their process. Subsequent studies could explore these results in greater detail using usability assessments and suggest innovative methods or settings to optimize the development process. These might make use of ideas like liveness and live software development, as well as model-driven engineering and visual programming, to help developers.

REFERENCES

[1] D. Reis, ‘‘Live Docker containers,’’ M.S. thesis, Dept. Eng., Univ. Porto, Porto, Portugal, 2020.
[2] B. Piedade, ‘‘Visual programming language for orchestration with Docker,’’ M.S. thesis, Dept. Eng., Univ. Porto, Porto, Portugal, 2020.
[3] M. Artac, T. Borovssak, E. Di Nitto, M. Guerriero, and D. A. Tamburri,‘‘DevOps: Introducing infrastructure-as-code,’’ in Proc. IEEE/ACM 39th Int. Conf. Softw. Eng. Companion (ICSE-C), May 2017, pp. 497–498.
[4] D. Bernstein, ‘‘Containers and cloud: From LXC to Docker to kubernetes,’’IEEE Cloud Comput., vol. 1, no. 3, pp. 81–84, Sep. 2014.
[5] J. Cito, P. Leitner, T. Fritz, and H. C. Gall, ‘‘The making of cloud applications: An empirical study on software development for the cloud,’’ in Proc. 10th Joint Meeting Found. Softw. Eng., Aug. 2015, pp. 393–403.
[6] Y. Zhang, G. Yin, T. Wang, Y. Yu, and H. Wang, ‘‘An insight into the impactofdockerfile evolutionary trajectories on quality and latency,’’ in Proc.IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC), Jul. 2018,pp. 138–143.
[7] R. Smith, Docker Orchestration. Birmingham, U.K.: Packt, 2017.
[8] M. Guerriero, M. Garriga, D. A. Tamburri, and F. Palomba, ‘‘Adoption,support, and challenges of infrastructure-as-code: Insights from industry,’’in Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME), Sep. 2019,pp. 580–589.
[9] A. Rahman, R. Mahdavi-Hezaveh, and L. Williams, ‘‘A systematicmapping study of infrastructure as code research,’’ Inf. Softw. Technol.,vol. 108, pp. 65–77, Apr. 2019.
[10] D. Weerasiri, M. C. Barukh, B. Benatallah, and C. Jian, ‘‘CloudMap:A visual notation for representing and managing cloud resources,’’ in Proc.28th Int. Conf. (CAiSE), 2016, pp. 427–443.
[11] P. Lourenço, J. Dias, A. Aguiar, and H. Ferreira, ‘‘CloudCity: A liveenvironment for the management of cloud infrastructures,’’ in Proc. 14thInt. Conf. Eval. Novel Approaches Softw. Eng., 2019, pp. 27–36.
[12] B. Piedade, J. P. Dias, and F. F. Correia, ‘‘An empirical study on visualprogramming Docker compose configurations,’’ in Proc. 23rd ACM/IEEEInt. Conf. Model Driven Eng. Lang. Syst., Companion, Oct. 2020,pp. 24–34.
[13] S. Dalla Palma, D. Di Nucci, F. Palomba, and D. A. Tamburri, ‘‘Toward a catalog of software quality metrics for infrastructure code,’’ J. Syst. Softw.,vol. 170, Dec. 2020, Art. no. 110726.
[14] M. U. Haque, L. H. Iwaya, and M. A. Babar, ‘‘Challenges in Dockerdevelopment: A large-scale study using stack overflow,’’ in Proc. 14thACM/IEEE Int. Symp. Empirical Softw. Eng. Meas. (ESEM), Oct. 2020,pp. 1–11.
[15] J. Cito, G. Schermann, J. E. Wittern, P. Leitner, S. Zumberi, and H. C. Gall,‘‘An empirical analysis of the Docker container ecosystem on GitHub,’’in Proc. IEEE/ACM 14th Int. Conf. Mining Softw. Repositories (MSR),May 2017, pp. 323–333.
[16] A. Rahman, S. Elder, F. H. Shezan, V. Frost, J. Stallings, and L. Williams,‘‘Bugs in infrastructure as code,’’ 2018, arXiv:1809.07937.

[17] A. Rahman, E. Farhana, C. Parnin, and L. Williams, ‘‘Gang of eight:A defect taxonomy for infrastructure as code scripts,’’ in Proc. ACM/IEEE42nd Int. Conf. Softw. Eng., vol. 20, Jun. 2020, pp. 752–764.
[18] A. Rahman, E. Farhana, and L. Williams, ‘‘The ‘as code’ activities:Development anti-patterns for infrastructure as code,’’ Empirical Softw.Eng., vol. 25, no. 5, pp. 3430–3467, Sep. 2020.
[19] H. Ibrahim, ‘‘A study of the use of Docker compose and dockerhubimages,’’ M.S. thesis, School Comput., Queen’s Univ., Kingston, ON,Canada, 2019.
[20] Thara D. K., et al. "EEG Forecasting With Univariate and Multivariate Time Series Using Windowing and Baseline Method." *IJEHMC* vol.13, no.5 2022: pp.1-13. http://doi.org/10.4018/IJEHMC.315731
[21] T. Harter, B. Salmon, R. Liu, A. C. Arpaci-Dusseau, andR. H. Arpaci-Dusseau, ‘‘Slacker: Fast distribution with lazy Docker containers,’’ in Proc. 14th USENIX Conf. File Storage Technol. (FAST),2016, pp. 181–195.
[22] Z. Huang, S. Wu, S. Jiang, and H. Jin, ‘‘FastBuild: Accelerating Dockerimage building for efficient development and deployment of container,’’in Proc. 35th Symp. Mass Storage Syst. Technol. (MSST), May 2019,pp. 28–37.
[23] J. S. Molléri, K. Petersen, and E. Mendes, ‘‘An empirically evaluated checklist for surveys in software engineering,’’ Inf. Softw. Technol.,vol. 119, Mar. 2020, Art. no. 106240.
[24] S. Baltes and P. Ralph, ‘‘Sampling in software engineering research:A critical review and guidelines,’’ 2020, arXiv:2002.07764.
[25] Thara, D. K., and H. A. Vidya. "Detecting Insurance Fraud: A Study on Field Fires with Computer Vision and IoT." *International Journal of Advanced Scientific Innovation* 5.7 (2023).
[26] S. Jamieson, ‘‘Likert scales: How to (ab)use them,’’ Med. Educ., vol. 38,no. 12, pp. 1217–1218, Dec. 2004.
[27] D. Reis and B. Piedade, Davidreis97/Challenges_With_Docker: Replica-tion Package. Zenodo, Apr. 2021.
[28] H. B. Mann and D. R. Whitney, ‘‘On a test of whether one of two randomvariables is stochastically larger than the other,’’ Ann. Math. Statist.,vol. 18, no. 1, pp. 50–60, Mar. 1947.
[29] A. Luxton-Reilly, E. Mcmillan, E. Stevenson, E. Tempero, and P. Denny,‘‘Ladebug: An online tool to help novice programmers improve their debugging skills,’’ in Proc. 23rd Annu. ACM Conf. Innov. Technol.Comput. Sci. Educ., New York, NY, USA, Jul. 2018, pp. 159–164, doi:10.1145/3197091.3197098.
[30] S. L. Tanimoto, ‘‘A perspective on the evolution of live programming,’’ inProc. 1st Int. Workshop Live Program. (LIVE), May 2013, pp. 31–34.
[31] A. Aguiar, A. Restivo, F. F. Correia, H. S. Ferreira, and J. P. Dias, ‘‘Livesoftware development: Tightening the feedback loops,’’ in Proc. Conf. Companion 3rd Int. Conf. Art, Sci., Eng. Program., Apr. 2019, pp. 1–6
[32] D. Amaral, G. Domingues, J. P. Dias, H. S. Ferreira, A. Aguiar, R. Nóbrega,and F. F. Correia, ‘‘Live software development environment using virtual reality: A prototype and experiment,’’ in Proc. Int. Conf. Eval. Novel Approaches Softw. Eng. Cham, Switzerland: Springer, 2019, pp. 83–107.
[33] A. Bhattacharjee, Y. Barve, A. Gokhale, and T. Kuroda, ‘‘CloudCAMP: Automating the deployment and management of cloud services,’’ in Proc.IEEE Int. Conf. Services Comput. (SCC), Jul. 2018, pp. 237–240.
[34] P. B. G, T. D. K and T. K. N, "ML based methods XGBoost and Random Forest for Crop and Fertilizer Prediction," *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)*, Al-Khobar, Saudi Arabia, 2022, pp. 492-497, doi: 10.1109/CICN56167.2022.10008234.