Understanding DevOps practices through software repositories

Hugo da Gião*
* Department of Informatics Engineering, Faculty of Engineering, University of Porto & HASLab/INESC TEC, Portugal
up202111338@up.pt

***Abstract*—DevOps is a combination of tools and methodologies that improves the software development, build, and deployment processes by shortening its lifecycle and improving software quality. Despite its many benefits, it still presents many challenges regarding its ease of use and accessibility. One of the reasons is the tremendous proliferation of different tools, languages, and syntaxes, which makes the field quite challenging to learn and keep up to date.**

Software repositories contain data regarding various software practices, tools, and uses. This data can help gather multiple insights that inform technical and academic decision-making. GitHub is currently the most popular software hosting platform and provides a search API that lets users query its repositories.

Our goal with this paper is to gain insights into the tools used for DevOps by developers by analyzing GitHub repositories. We collect various tools from surveys, reports, and blog posts. After gathering the tools, we use the GitHub search API to find repositories using each of these tools. We also use the API to extract various insights regarding those repositories. We then organize and analyze the data collected.

In our analysis, we found several insights concerning the tools studied. According to our analysis, these insights include that Docker is the most popular tool. We also found that the tools used for the Deployment phase are some of the most popular. We also found that, on average, repositories associated with more popular tools had lower interaction.

***Index Terms*—DevOps, mining repositories**

I. Introduction

In 2022 the Information technology (IT) market accounted for $9,325.69 billion dollars worldwide [1]. 50% of IT costs are related to the maintenance and upkeep of existing systems, and of those, 50% correspond to emergencies, unplanned work, and changes [2]. This cost happens partially because the IT and development team's goals often conflict in most organizations. Instead of working for a common goal, most organizations' development teams are responsible for reacting to and answering rapid changes in markets and customer needs. In contrast, IT teams provide customers with a secure, reliable, and stable experience. This dynamic leads to technical debt, meaning decisions made over time lead to increasingly more complex problems and slower operations and thus impede the achievement of the organization's goals [2].

Inspired by the Lean practices used in the manufacturing industry, which significantly increased the number of orders fulfilled on time and led to companies adopting those practices dominating the market, DevOps comes as a solution to the problems of the IT industry. This methodology comes from the union of development (Dev) and IT operations (Ops). It is a means for software development and delivery using various tools and techniques that integrate these two worlds. Integrating those fields involves automated development, build, deployment, and monitoring. One of the goals is to achieve reliable, secure, fast, and continuous delivery of software [2].

However, organizations and developers face many obstacles when adopting DevOps, including changes in the architectural organization, dealing with problems in older infrastructure, and the integration of different DevOps tools [3]. Other issues developers face are learning and using the specific DevOps tools, which can be overwhelming and can decrease the improvements expected with the adoption of DevOps [4].

GitHub is a widely used version control and software hosting service. As of 2022, 94 million developers and 90% of fortune 500 companies use it, and in that year, had 413 million open source contributions [1]. GitHub also provides an API to collect information retaining to its software repositories. With this API, we can search repositories inside GitHub using parameters such as keywords in their name and README, size, number of stars, followers, and forks [2]. The insights collected using the API help understand various developer behaviors.

With this work, we aim to gain insights into how developers use several tools for different DevOps phases. The information collected can guide future research on possible improvements to the DevOps process. We plan on collecting information from public software repositories on GitHub. With this work, we aim to answer the following research questions:

RQ1 What is the evolution of DevOps tools and projects?
With RQ1, we want to mine GitHub repositories and find how many repositories use each tool. Our goal is also to discover how the tools used varied over time.

RQ2 How are DevOps tools used?
With RQ2, we aim to gather information concerning the repositories using each of the tools studied. This information includes the number of forks, stars, followers, the percentage of archived repositories, and the size of the repositories.

We organize the rest of the document as follows:

Section II presents several works related to our own. Those works focus on mining information about several DevOps

[1]https://octoverse.github.com/
[2]https://docs.github.com/en/rest/search

aspects in various software repositories.

In Section III, we present and describe the several phases of the DevOps lifecycle.

In Section IV, we present the methodology used for this work. This methodology includes how we found the tool for this study, what information we collected from the repositories and how it related to our research questions, how we used the GitHub API to collect information from the repositories, and how we organized the data collected from said repositories.

In Section V, we showcase and analyze the results from our work and use them to answer our research questions.

Finally, in Section VI, we present our findings for this work. This Section also includes limitations of our work and possible improvements.

II. Related Work

In this Section, we introduce our related work. This paper's related work focuses on several articles that mine software repositories to understand and improve aspects of software development. Similarly to these works, we mine repositories related to the DevOps process. However, in our work, we do not search for specific scripts and identify the repositories using keywords in their title, description, and README. We also do not analyze specific files but study the repositories using the GitHub search API data. We also analyze repositories that contain several tools with different purposes and uses.

Reference [5] by T. Xu et al., this paper introduces the idea of mining container image repositories for configuration and other deployment information of software systems. The authors also showcase the opportunities based on concrete software engineering tasks that can benefit from mining image repositories. They also e summarize the challenges of analyzing image repositories and the approaches that can address these challenges.

Reference [6] by M. Zahedi et al. present an empirical study exploring continuous software engineering from the practitioners' perspective by mining discussions from Q&A websites. The authors analyzed 12,989 questions and answers posted on Stack Overflow. The authors then used topic modeling to derive the dominant topics in this domain. They then identify and discuss key challenges.

Reference [7] by Y. WU et al. present a preliminary study on 857,086 Docker builds from 3,828 open-source projects hosted on GitHub. Using the Docker build data, the authors measure the frequency of broken builds and report their fix times. they also explore the evolution of Docker build failures across time.

Reference [8] by J. Henkel et al. introduce a binnacle toolset that enables the authors to ingest 900,000 GitHub repositories. Focusing on Docker, they extracted approximately 178,000 unique Dockerfiles.

III. The DevOps lifecycle

The DevOps process is made up of different phases comprising its lifecycle. These phases are necessary to guarantee the full range of benefits associated with the use of DevOps, such as improved speed, software quality, and more dynamic teams, the phases comprising this lifecycle are as follows [9]:

- **Development** - The phase of continuous development involves the tasks of planning and developing software. This process is an application of the agile methodology to the world of software development. The whole software gets broken down into smaller pieces and is updated in smaller batches after being tested [9].
- **Integration** - This phase is at the core of the DevOps lifecycle and affects the process of committing new changes in the source code. When doing continuous integration, developers are forced to do more frequent software releases, and these more frequent releases, in conjunction with the use of various testing techniques such as unit testing and integration testing and other quality assurance practices such as code reviews, allow them to be more secure and stable [9].
- **Testing** - Continuous testing involves integrating automatic testing tools into the software development pipeline. This process saves organizations time and increases productivity [9].
- **Monitoring** - This phase entails performance monitoring and recording the application, including potential errors such as high memory usage and networking problems. This monitoring allows development teams to find and correct mistakes earlier and substantially reduce IT costs [9].
- **Feedback** - This phase comprises the gathering, analyzing, and using feedback from the client's end. This information includes performance and errors and customer feedback [9].
- **Deployment** - Continuous deployment affects deploying new code to the production environment. This process is made automatically and reduces the need for manual and planned releases since recent changes to the source code are put into production after integration and testing. This phase extensively uses container technology such as Docker to make use of standardized environments [9].
- **Operations** - This phase includes the automation of all operations processes that help developers automate releases, detect issues faster, and improve software products [9].

IV. Research Design

Our first step was to collect a list of several DevOps tools. To do so, we found several blogs written by companies specialized in DevOps, reports about the state of DevOps, and surveys that listed DevOps tools used by developers. We then compiled and organized the tools according to the DevOps phases for which they are used in Table I. Our sources for gathering the tools are the following:

- "Comprehensive List of DevOps Tools 2023" from Qentelli [10].
- "DevOps Tools" from Atlassian [11].
- "The State of DevOps Tools" from DevOps.com [12].

TABLE I
PUBLICATIONS ORGANIZED ACCORDING TO THEIR DOMAINS AND DEVOPS LIFECYCLE PHASES.

Tool	Phase
Travis	Integration
Flyway	Integration
Maven	Integration Deployment
Gradle	Deployment
Rake	Deployment
Jenkins	Deployment Integration Operations
Open Stack	Deployment
Rancher	Deployment
Docker	Deployment
Ansible	Deployment
Terraform	Deployment
Octopus Deploy	Deployment
Bamboo	Deployment
JUnit	Testing
Selenium	Testing
Progress Chef	Operations
Puppet	Operations
Nagios	Monitoring
Zabbix	Monitoring
Prometheus	Monitorin
Logstash	Monitoring
Graylog	Monitoring
Kubernetes	Deployment
Mesos	Deployment

- "A Survey of DevOps Concepts and Challenges" published in ACM Computing Surveys [13].

From the contents of Table I, we use the GitHub search API [3] to gather data about repositories. We collected repositories with the name of any of the tools in their title, description, or README. This API searches original GitHub repositories and allows for different queries. It returns the total number of matches for the query and information of up to 100 repositories matching it. Collecting data for the first 1000 by making several calls is possible. However, this is not enough to collect all the repositories. Due to this limitation, our strategy was to do various queries and collect the number of matches for different scenarios instead of collecting the information for each of the tool's repositories and then doing a local analysis. We use this API to collect the following information about the repositories:

- To answer RQ1 'What is the evolution of DevOps tools and projects?' we collected the following information:
  - Repositories for each of the tools. We organize the data in a graphic containing the number of repositories found for each tool using a base ten logarithmic scale. This is the graphic in Figure 1.
  - Repositories created each year from 2010 to 2022 for each tool. We organized this information in a graphic containing each year's number of created repositories for each tool. This is the graphic in Figure 2.

[3]https://docs.github.com/en/rest/search

- To answer RQ2 'How are DevOps tools used?' we collect the following information:
  - Stars for each tool. We queried the API for repositories containing each tool and having a number of stars in a given range. We organized the information in a graphic containing for each tool the percentages of repositories with stars in each of the given ranges in Figure 3.
  - Forks for the repositories for each tool. We queried the API for repositories containing each tool with forks in the given ranges. We organize the information in a graphic containing for each tool the percentage of repositories with the forks in a given range in Figure 4.
  - Archived repositories for each of the tools. For each tool, we queried the API for archived or not archived repositories. With this, we created a graphic containing for each tool the percentage of archived and unarchived repositories in Figure 5
  - Number of followers for each of the tools repositories. We queried the API for the repositories containing the names of each tool and the number of followers in a given range. We organize this information in a graphic containing for each tool the percentage of repositories with several followers in a given range in Figure 6.
  - Size of each tool's repositories. We queried the API for the repositories containing each tool with a size in a given range rep. We organize this information in a graphic containing the percentage of repositories within a given size range for each of the tools in Figure 7.

V. RESULTS

To answer RQ1 'What is the evolution of DevOps tools and projects?' we can look at Figure 1 and Figure 2. From these Figures, we can see which tools are referenced in more GitHub repositories and when the repositories referencing each tool were created. From Figure 1, we can notice Docker is the more popular tool by far. We can also see that tools such as Maven, Jenkins, Ansible, Terraform, Selenium, and Kubernetes are some of the most used. On the other hand, Progress Chef is the least used of the tools studied. Other tools, such as Flyway, Mesos, Graylog, and Rancher, are some of the least used. We can see that the most used tools are all used for the Deployment phase of DevOps, except Selenium which is used for testing. Of the least used tools, only Rancher is used for the Deployment phase.

From Figure 2, we can observe a considerable increase in the projects using the tools in this study up to 2020. From the Figure, we can also see that repositories containing Docker comprise a considerable amount of the total repositories from 2014 onwards. Repositories containing Docker, Jenkins, Kubernetes, Maven, Ansible, Selenium, and Terraform make up most of the repositories from 2014 onwards. Most of those tools are once again used in the Deployment phase.

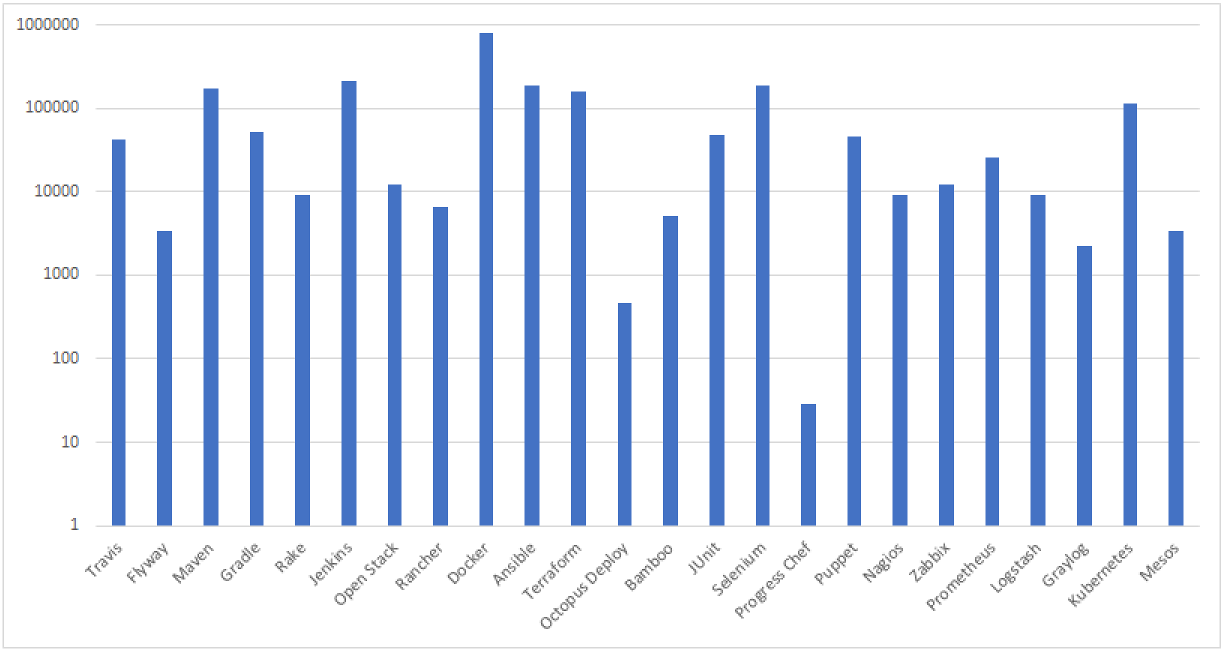

Fig. 1. Number of repositories found for each tool in logarithmic scale

To answer RQ2 ‘How are DevOps tools used?’ we can look at Figure 5, Figure 6, Figure 4, Figure 7 and Figure 3 to analyze information about the collected repositories. From Figure 3, we can conclude that tools such as Kubernetes, Mesos, Prometheus, Logstach, Graylog, Zabbix, and Nagios have a higher percentage of higher impact repositories. On the other hand, tools such as JUnit, Selenium, and Maven have a higher rate of repositories with lower impact when compared with other tools. We can see similar results by looking at Figure 6 and Figure 4, where we can see a correlation between the number of stars, forks, and followers of a repository. We can see that the tools with a higher percentage of higher-impact repositories are used for different stages of the DevOps process and that some of the tools with a higher rate of lower-impact repositories are widely used tools, which indicates that these results are affected by the high number of repositories using those tools.

From Figure 5, we can observe that for most of the tools, most of the repositories are not archived, Puppet and Octopus Deploy have some of the highest ratios of archived projects, but even those are very low. Finally, we can observe from Figure 7 that Mesos and Bamboo are the tools with the bigger repositories.

VI. Concluding Remarks and Future Work

This work collects data concerning software repositories containing several DevOps tools. We collected the data using the GitHub search API and then organized and analyzed it. From the data, we reached several conclusions concerning the current and past use of several tools. We also gained insights into how the characteristics of the software repositories using the tools. These results can be used to guide future work concerning several aspects of DevOps.

Some aspects of our work could be improved. We could find more repositories by using the GitHub code search API to find scripts associated with each tool. Doing so would entail analyzing scripts for each of the collected tools and finding patterns that would help identify them. Currently, the code search API limits the search of repositories within a given organization or list of organizations. An alternative to the code search API would be to have an approach similar to J. Henkel et al [8] and download the data for all repositories with stars and creation data within a threshold and do a mix of local processing to find files with matching extensions and download those files for analysis of its contents.

Using either of these approaches would vastly mitigate the risk of false positives that happens by querying for repositories with the tool name in the title, README, and description, especially with tools whose names are commonly used words. These approaches would also increase the number of repositories analyzed and allow for a deeper analysis and insights, such as which tools are used together. However, when compared to ours, these approaches would require more processing time and capabilities.

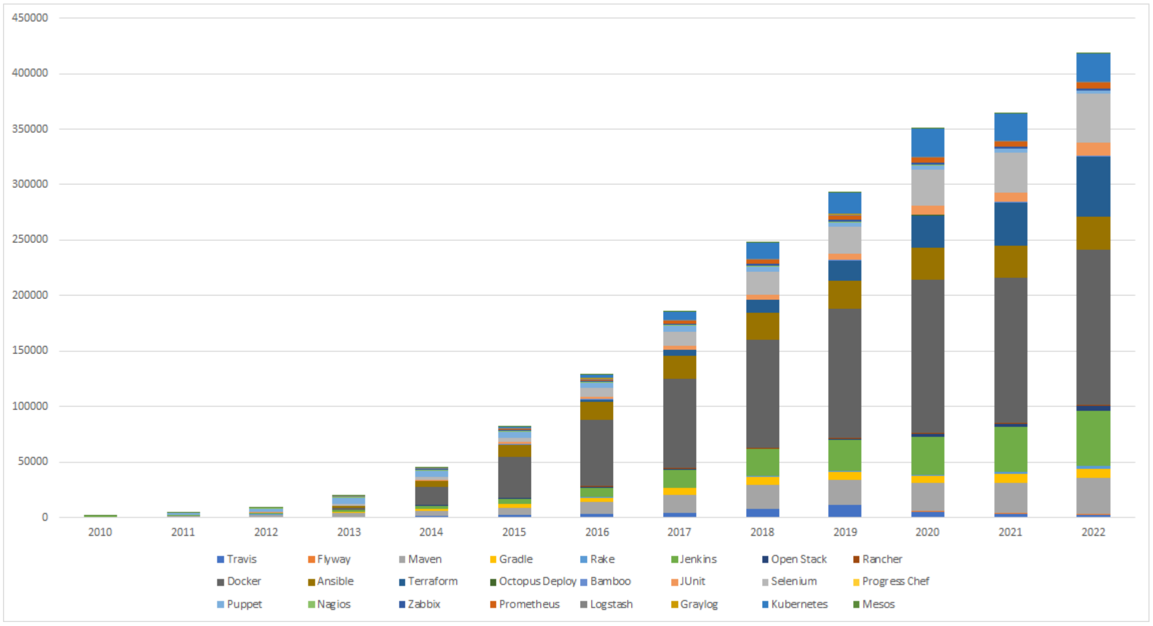

Fig. 2. Number of repositories created from 2010 to 2022 for each tool

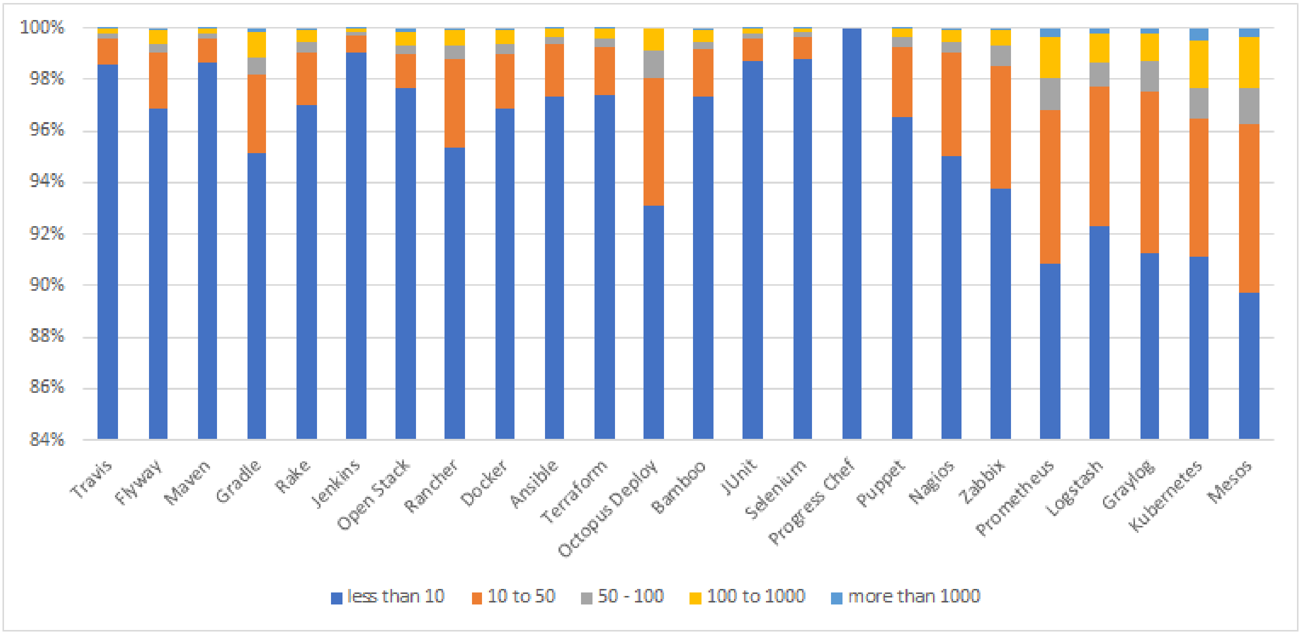

Fig. 3. Percentage of repositories with stars in a given range for each tool

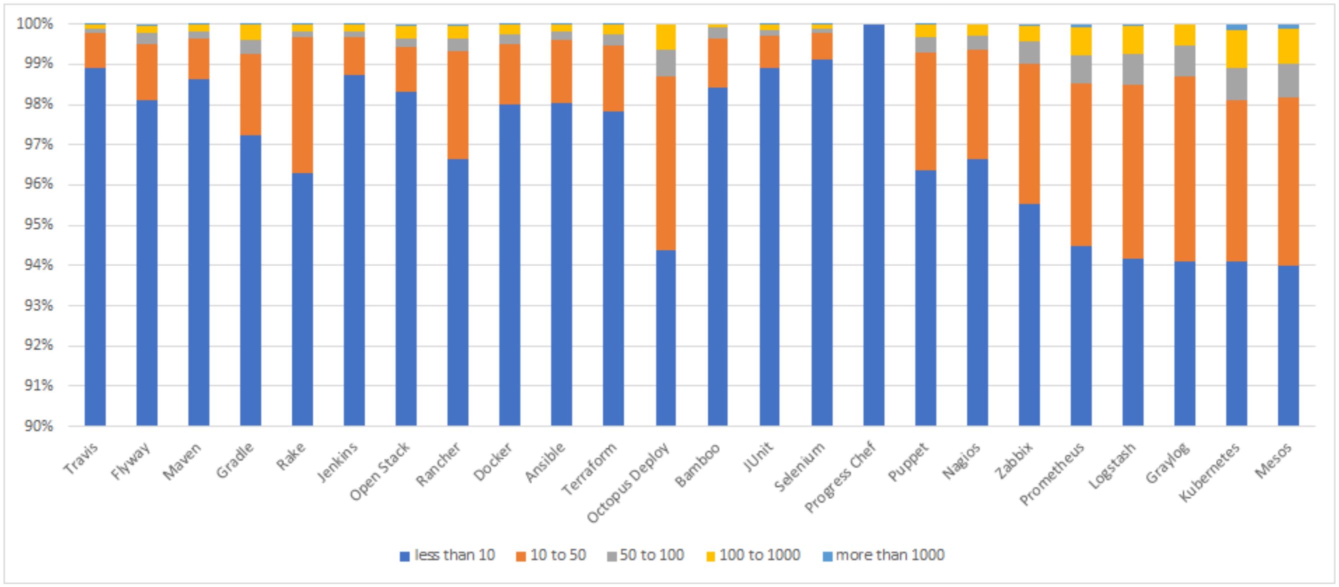

Fig. 4. Percentage of repositories for each tool with the number of forks in a given range

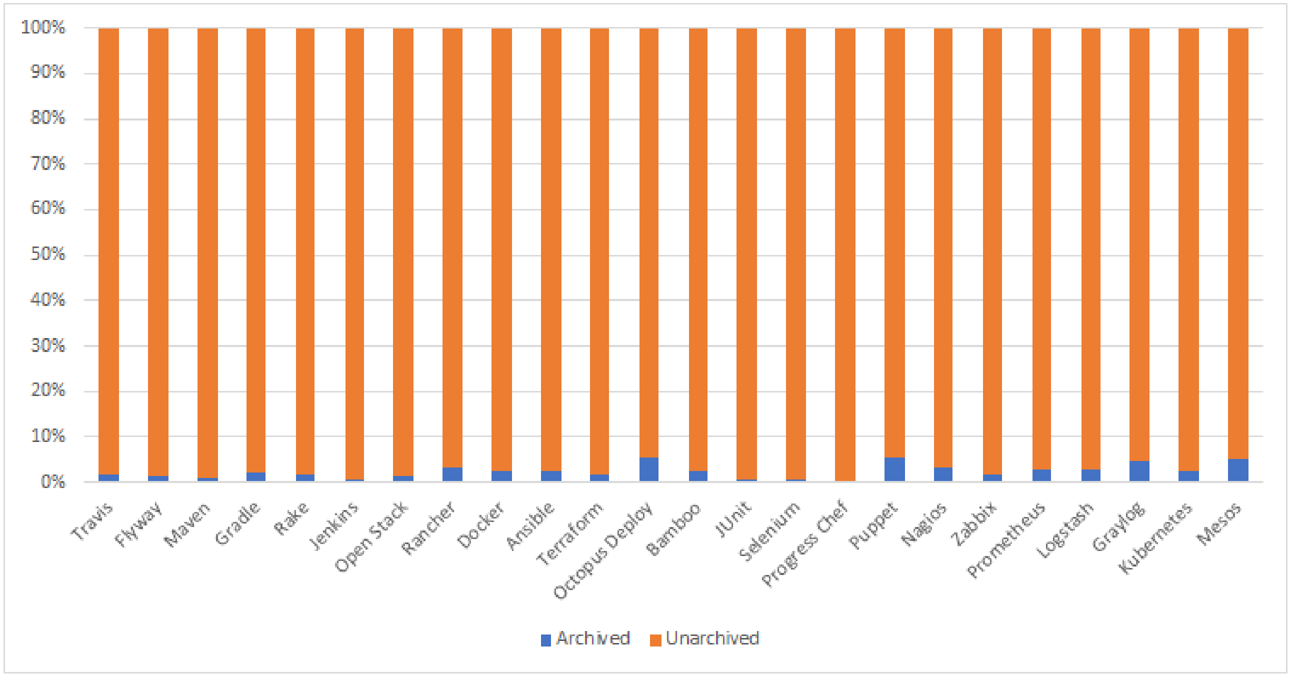

Fig. 5. Percentage of archived and unarchived repositories for each tool.

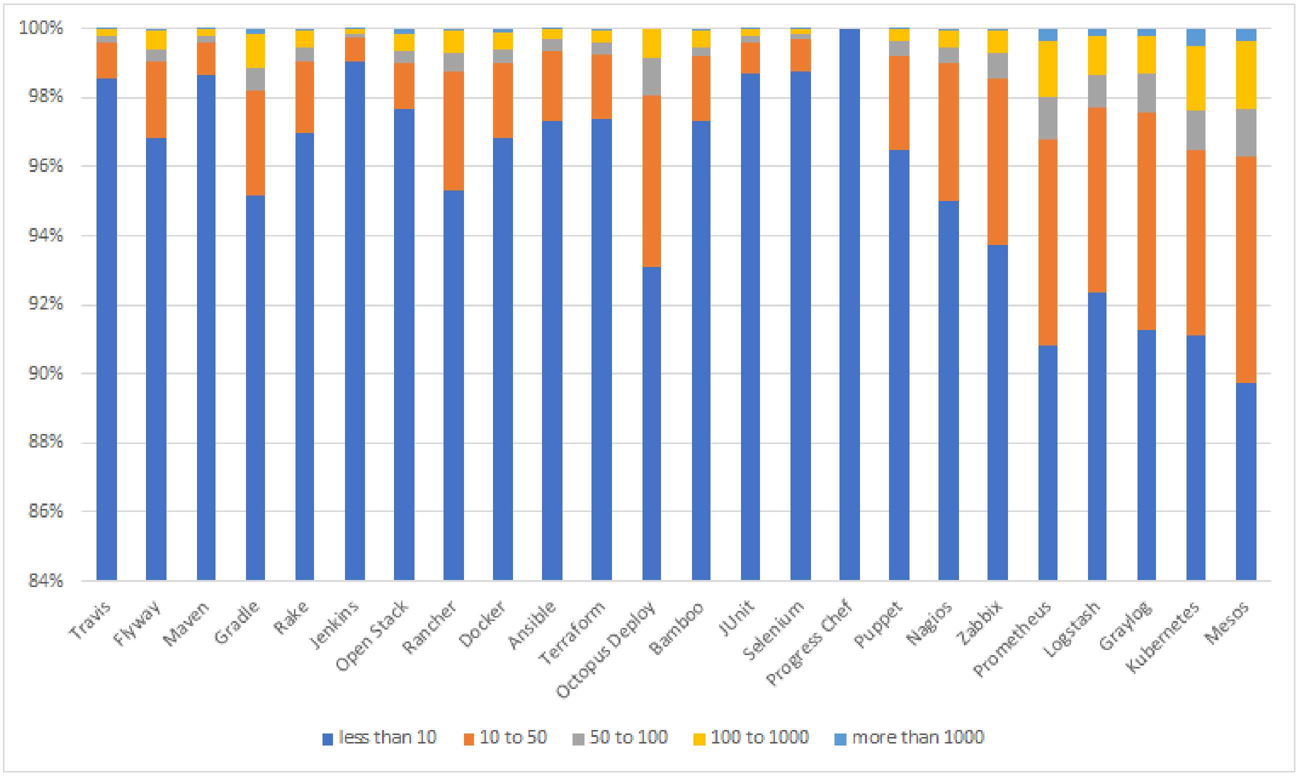

Fig. 6. Percentage of repositories for each tool with the number of followers in a given range

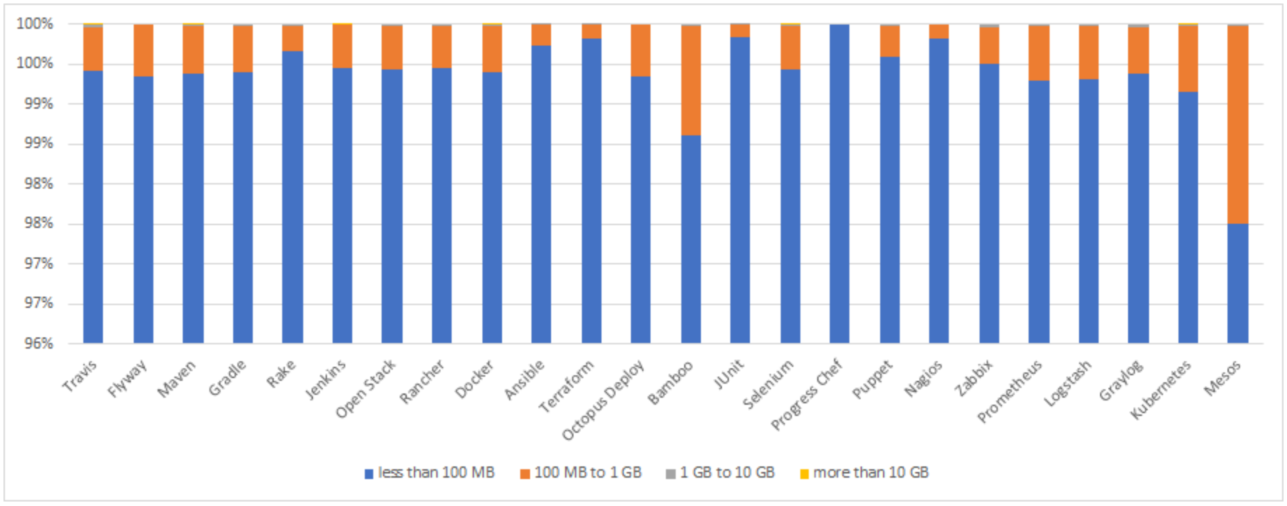

Fig. 7. Percentage of repositories within a given size range for each tool

References

[1] T. B. R. Company, "Information technology global market report 2022, by type, organization size, end user industry," Mar 2022. [Online]. Available: https://www.researchandmarkets.com/reports/5561700/information-technology-global-market-report-2022

[2] G. Kim, J. Humble, P. Debois, and J. Willis, *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*, ser. ITpro collection. IT Revolution Press, 2016. [Online]. Available: https://books.google.pt/books?id=ui8hDgAAQBAJ

[3] G. B. Ghantous and A. Q. Gill, "Devops: Concepts, practices, tools, benefits and challenges," in *PACIS*, 2017.

[4] M. U. Haque, L. H. Iwaya, and M. A. Babar, "Challenges in docker development: A large-scale study using stack overflow," in *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, ser. ESEM '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3382494.3410693

[5] T. Xu and D. Marinov, "Mining container image repositories for software configuration and beyond," in *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results*, ser. ICSE-NIER '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 49–52. [Online]. Available: https://doi.org/10.1145/3183399.3183403

[6] M. Zahedi, R. N. Rajapakse, and M. A. Babar, "Mining questions asked about continuous software engineering: A case study of stack overflow," in *Proceedings of the Evaluation and Assessment in Software Engineering*, ser. EASE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 41–50. [Online]. Available: https://doi.org/10.1145/3383219.3383224

[7] Y. Wu, Y. Zhang, T. Wang, and H. Wang, "An empirical study of build failures in the docker context," in *Proceedings of the 17th International Conference on Mining Software Repositories*, ser. MSR '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 76–80. [Online]. Available: https://doi.org/10.1145/3379597.3387483

[8] J. Henkel, C. Bird, S. K. Lahiri, and T. Reps, "Learning from, understanding, and supporting devops artifacts for docker," in *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, 2020, pp. 38–49.

[9] L. Zhu, L. Bass, and G. Champlin-Scharff, "Devops and its practices," *IEEE Software*, vol. 33, no. 03, pp. 32–34, may 2016.

[10] A. S. B. B. R. Badam, B. S. Bhanda, S. Bhanda, B. R. Badam, and R. Badam, "Comprehensive list of devops tools 2023." [Online]. Available: https://www.qentelli.com/thought-leadership/insights/devops-tools

[11] Atlassian, "Devops tools for each phase of the devops lifecycle." [Online]. Available: https://www.atlassian.com/devops/devops-tools

[12] D. McVittie and A. Shimel, *The State of DevOps Tools*, 2018. [Online]. Available: https://devops.com/wp-content/uploads/2018/03/StateOfDevOpsTools_v13.pdf

[13] L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles, "A survey of devops concepts and challenges," *ACM Comput. Surv.*, vol. 52, no. 6, nov 2019. [Online]. Available: https://doi.org/10.1145/3359981