Understanding DevOps practices through software repositories

***Abstract*—DevOps is a combination of tools and methodologies that aims at improving the software development, build, and deployment processes by shortening this 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 can be used to extract data regarding various software practices, tools, and uses. This data can then be used to gather various insights that can inform both the technical and academic decision processes. 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 do that by collecting an array of 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.

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

I. Introduction

In 2022 the Information technology (IT) market accounted for $9,325.69 billion dollars worldwide [1]. It has been shown that 50% of IT costs are related to the maintenance and upkeep of existing systems and from those 50% correspond to emergencies, unplanned work, and changes [2]. This happens partially because in most organizations IT and development teams' goals often conflict since 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, while the IT teams role is to provide the customer with a secure, reliable and stable experience. This dynamic leads to "technical debt", that is decisions made over time lead to increasingly harder 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 lead 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 the worlds of development (Dev) and IT operations (Ops), it is a means for software development and delivery by using various tools and techniques that integrate this two worlds. The integration of these two fields is achieved through 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 problems faced by developers are in 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, it is used by 94 million developers, and 90% of fortune 500 companies, and in that year had 413 million open sources contributions [1].

With this work, our goal is to gain insights into how several tools are used for different DevOps phases. We plan on doing so by collecting information from public software repositories on GitHub. With this work we aim at answering the following research questions:

- RQ1 What is the evolution of DevOps tools and projects? With RQ1 our goal is to mine various GitHub repositories and find each of the studied tools and how many repositories are using it. Our goal is also to find how the tools used varied over time.
- RQ2 How are DevOps tools used? With RQ2 our aim is 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.

The rest of the document is organized as follows:

In Section II we present several works related to our own, those works focus on mining information related to several aspects of DevOps in various software repositories.

In Section IV we present the methodology for this work. This 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 conclusions for this work. The conclusions include limitations of our work and possible improvements.

[1] https://octoverse.github.com/

II. Related Work

In this Section we introduce our related work, this paper related work focuses on several that mine software repositories to understand and improve aspects of software development, these works also include.

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

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 [5].

M.Zahedi et al, present an empirical study aimed at exploring continuous software engineering from the practitioners' perspective by mining discussions from Q&A websites. The authors analyze 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 discuss identify and discuss key challenges. [6].

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 [7].

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

III. The DevOps lifecycle

The DevOps process is made up of different phases comprising its lifecycle, these faces 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, that is the whole software gets broken down into smaller pieces, and software 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 the integration of 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 of the application, including potential errors such as high memory usage and networking problems. This monitoring allows development teams to find and correct errors earlier and substantially reduce IT costs.
- **Feedback** - This phase comprises the gathering, analysis, and use of feedback coming from the client's end, this information includes performance and errors as well as customer feedback [9].
- **Deployment** - Continuous deployment affects the process of deploying new code to the production environment. This process is made automatically and reduces the need of having manual and planned releases since new changes to the source code are put into production after the process of integration and testing. This phase makes extensive use of container technology such as Docker to make use of standardized environments [9].
- **Monitoring** - This phase entails performance monitoring and recording of the application, including potential errors such as high memory usage and networking problems. This monitoring allows development teams to find and correct errors earlier and substantially reduce IT costs.
- **Operations** - This phase includes the automation of all operations processes that help developers automate releases, detect issues faster, and improve software products.

IV. Research Design

To achieve our goals 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].
- 'A Survey of DevOps Concepts and Challenges' published in ACM Computing Surveys [13].

From the list of tools, we use the GitHub search API [2] to

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

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	Building
Mesos	Building

gather data about repositories that contain the name of any of the tools in its name, the description, or the README of the repository. This API searches original GitHub repositories and allows for different queries. It returns the total number of matches for the query and also returns information of up to 100 repositories that matched the query, it is possible to collect information for the first 1000 by making several calls, however, this is not enough to collect all the repositories. Our strategy was due to this limitation to do various queries and to 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:
  - The number of repositories for each of the tools. We organize the data in a graphic containing the number of repositories found for each of the tools using a base 10 logarithmic scale, this graph can be seen in Figure 1.
  - The number of repositories created each year from 2010 to 2022 for each of the tools. We organized this information in a graphic containing for each of the years the number of repositories created for each of the tools in Figure 2.
- To answer RQ2 'How are DevOps tools used?' we collect the following information:
  - The number of stars for the repositories for each tool. To do so we queried the API for repositories containing the name of each of the tools and the number of stars in the given ranges. We organized the information in a graphic containing for each of the tools the percentages of repositories with stars in each of the given ranges in Figure 3.
  - The number of forks for the repositories for each tool. To do so we queried the API for repositories containing the name of each of the tools and the number of forks in the given ranges We organize the information for this in a graphic containing for each tool the percentage of repositories with the forks in a given range in Figure 7.
  - The number of archived repositories for each of the tools. To do so for each of the tools we queried the API for the repositories that were archived or not archived. With this, we created a graphic containing for each tool the percentage of archived and unarchived repositories in Figure 4
  - The number of followers for each of the tools repositories. To collect the information we queried the API for the repositories containing the names of each of the tools 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.
  - The size of the repositories for each tool. To collect the information we queried the API for the repositories containing the names of each of the tools and the size of the repository in a given range. We organize this information in a graphic containing for each of the tools the percentage of repositories within a given size range in Figure 5.

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 as well as when the repositories referencing each tool were created. From Figure 1 we can see Docker is the more popular tool by far. We can also see that tools such as Maven, Jenkins, Ansible, Terraform, Selenium, and Kubernetes are also 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, with exception of 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 observe that repositories containing Docker comprise a considerable of the total amount of repositories from 2014 onwards. Repositories containing Docker, Jenkins, Kubernetes, Maven, Ansible, Selenium and Terraform make

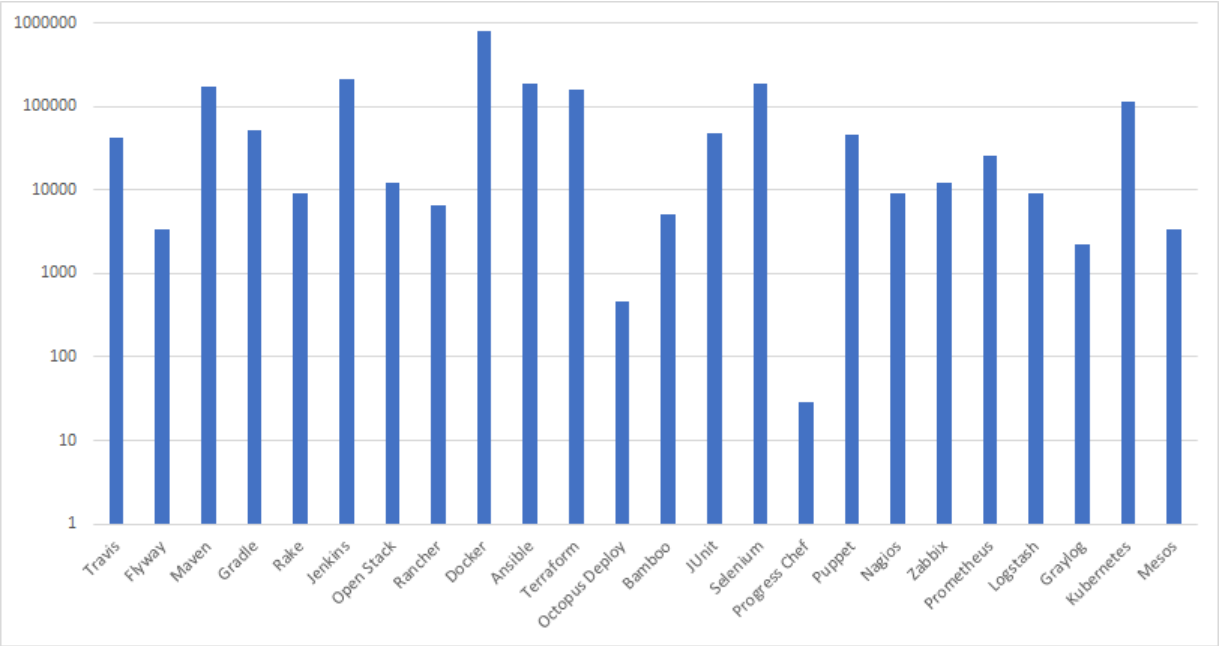

Fig. 1. Number of repositories found for each tool in logarithmic scale

up the vast majority of repositories found from 2014 onwards. Most of those tools are once again used in the Deployment phase.

To answer RQ2 'How are DevOps tools used?' we can look at Figure 4, Figure 6, Figure 7, Figure 5 and Figure 3 to analyze information pertaining to 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 percentage of repositories with lower impact when compared with other tools. We can see similar results by looking at Figure 6 and Figure 7 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 percentage 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 4 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 5 that Mesos and Bamboo are the tools with the highest-sized repositories.

VI. Concluding Remarks and Future Work

In this work we collect data concerning software repositories containing several DevOps tools, we collected the data using the GitHub search API and then proceeded to organize and analyze it. From the data, we reached several conclusions concerning the present and past use of several tools. We also gained insights into how the characteristics of the software repositories using the tools. We believe that these results can be used to guide future work concerning several aspects of DevOps.

Some aspects of our work could be improved, it is possible that we could find more repositories by using the GitHub code search API to find scripts associated with each of the tools, also there is a small risk of false positives by querying for repositories with the tool name in the title, README, and description, especially with tools that have common names. We could also extract more information from the repositories by downloading the information from the repositories and doing local processing, this information includes which tools are used together.

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*

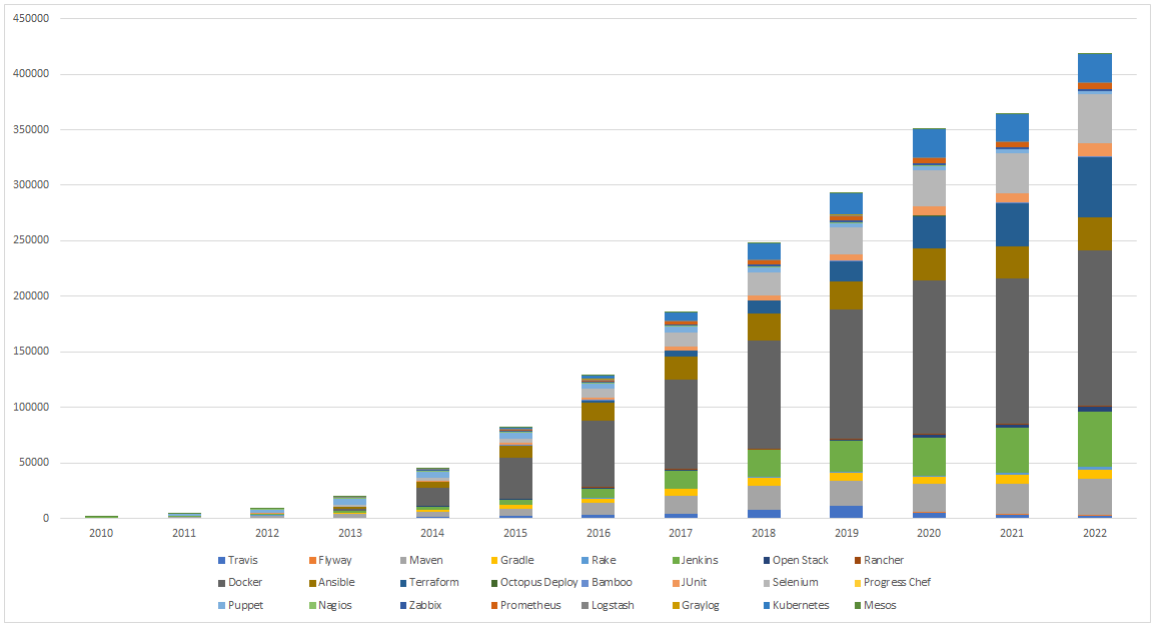

Fig. 2. Number of repositories created from 2010 to 2022 for each tool

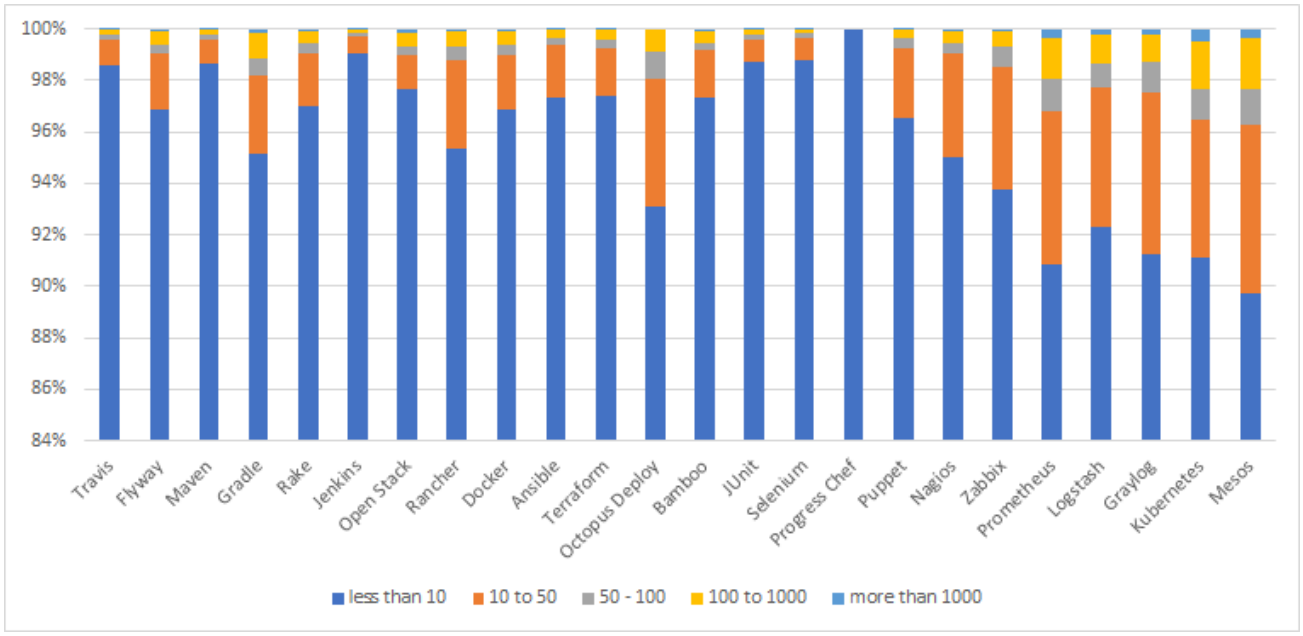

Fig. 3. Percentage of repositories with stars in a given range for each tool

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

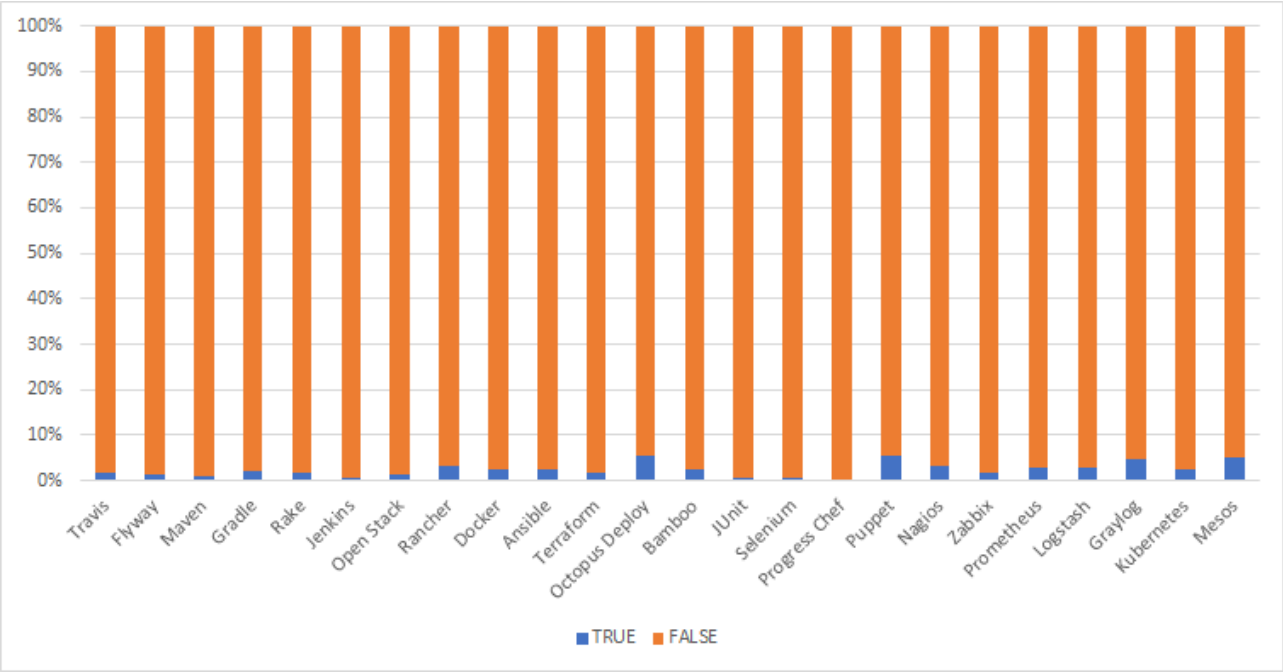

Fig. 4. Percentage of archived and unarchived repositories for each of the tools.

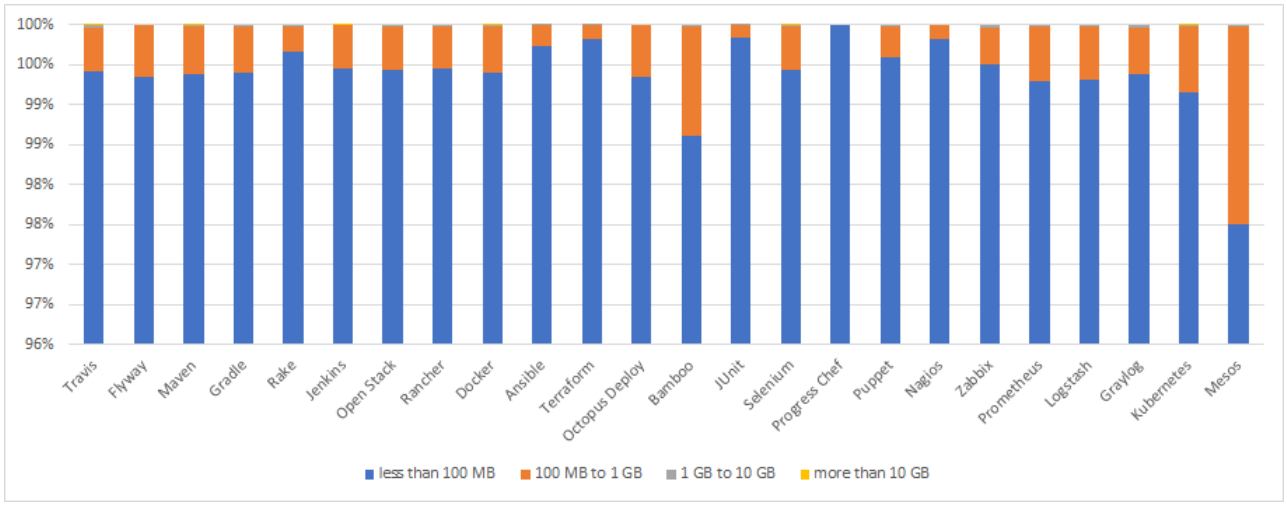

Fig. 5. Percentage of repositories within a given size range for each tool

[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

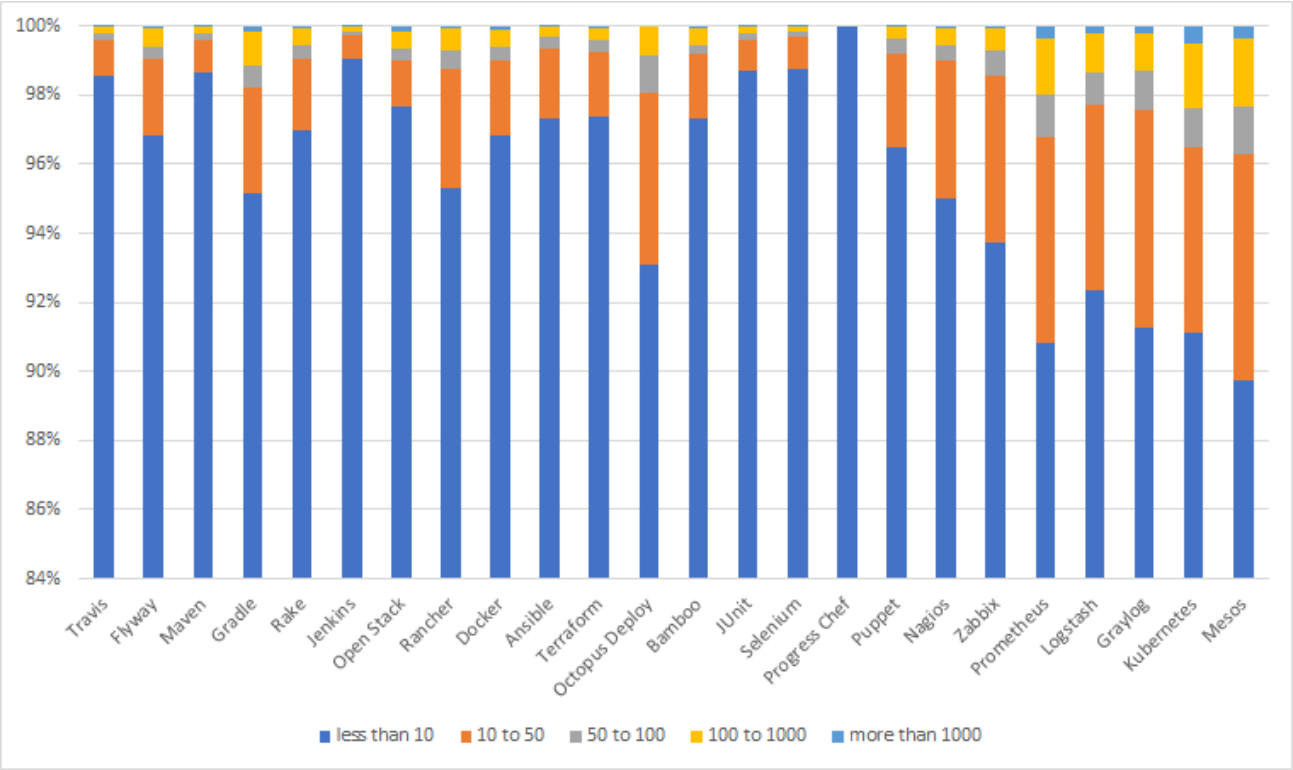

Fig. 6. Percentage of repositories for each tool with the number of followers in a given range

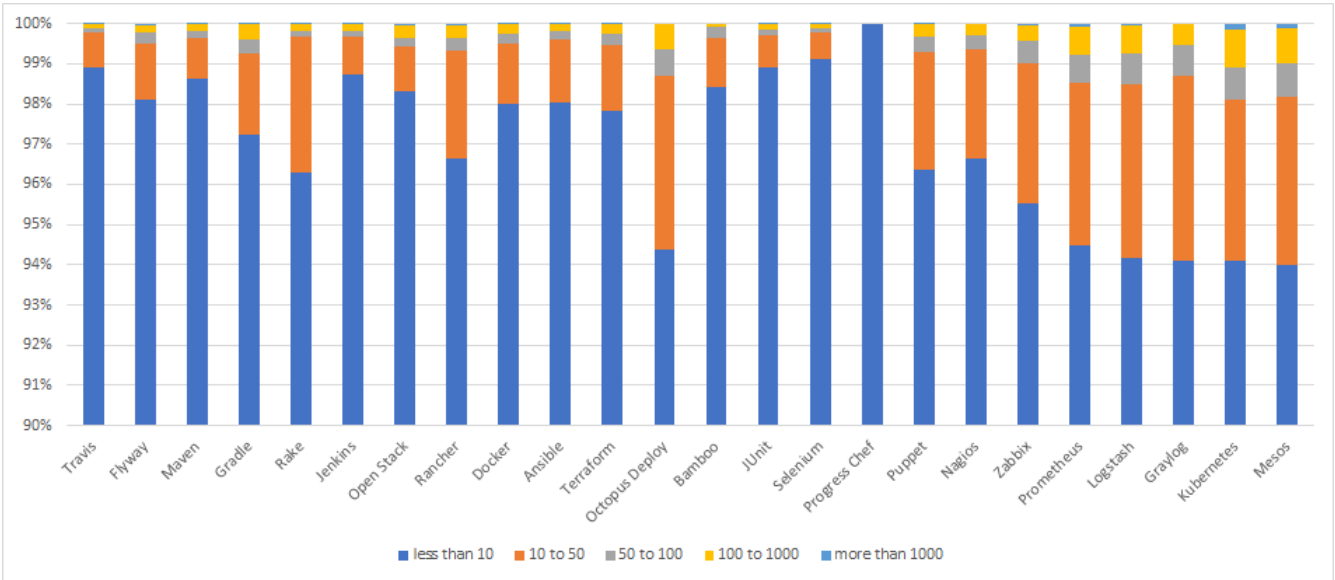

Fig. 7. Percentage of repositories for each tool with the number of forks in a given range