Chronicles of CI/CD: A Deep Dive into its Usage Over Time

Anonymous Author(s)

ABSTRACT

DevOps is a combination of methodologies and tools that improves the software development, build, deployment, and monitoring processes by shortening its lifecycle and improving software quality. Part of this process is CI/CD, which embodies mostly the first parts, right up to the deployment. Despite the many benefits of DevOps and CI/CD, it still presents many challenges promoted by 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 technologies developers use for CI/CD by analyzing GitHub repositories. Using a list of the state-of-the-art CI/CD technologies, we use the GitHub search API to find repositories using each of these technologies. We also use the API to extract various insights regarding those repositories. We then organize and analyze the data collected.

From our analysis, we provide an overview of the use of CI/CD technologies in our days, but also what happened in the last 12 years. We also show developers use several technologies simultaneously in the same project and that the change between technologies is quite common. From these insights, we find several research paths, from how to support the use of multiple technologies, both in terms of techniques, but also in terms of human-computer interaction, to aiding developers in evolving their CI/CD pipelines, again considering the various dimensions of the problem.

CCS CONCEPTS

• **Software and its engineering** → **Software libraries and repositories**; • **Applied computing** → *Document analysis*; *Document searching*; • **Information systems** → **Data mining**.

KEYWORDS

CI/CD, DevOps, Mining Repositories, Technology Evolution

ACM Reference Format:

Anonymous Author(s). 2018. Chronicles of CI/CD: A Deep Dive into its Usage Over Time. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In 2022 the Information Technology (IT) market accounted for $9,325.69 billion dollars worldwide [12]. Half of IT costs are related to the maintenance and upkeep of existing systems, and of those, 50% correspond to emergencies, unplanned work, and changes [20]. 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 [20].

In recent years, DevOps, that is, the integration of development and IT teams seamlessly, has emerged to enable organizations to respond to market demands with unparalleled agility hoping to address the aforementioned problems. In particular, methodologies and technologies for Continuous Integration, Continuous Delivery, and Continuous Deployment, or simply CI/CD, have emerged as a means for organizations to achieve rapid and frequent delivery of changes. A CI/CD pipeline encompasses a series of essential steps involved in integrating and deploying codebase changes [20]. This process involves the regular integration of new code changes, which undergo automated building and testing procedures. Subsequently, the validated code is deployed into production through the CD process [20].

Organization have the option to employ various technologies like GitHub Actions[1], GitLab CI/CD[2], Travis CI[3], CircleCI[4] or Jenkins[5]. The wide range of existing technologies makes it harder for researchers and developers to study and improve the CI/CD process. Indeed, there is no view of the usage of CI/CD technologies in a global sense, e.g., most used technologies, their evolution over time, which kinds of projects most rely on CI/CD, and so on.

GitHub is a widely used version control and software hosting service. As of 2023, more than 100 million developers used GitHub, and the platform hosted over 284 million public repositories [22]. GitHub also provides an API to collect information from 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[6]. The insights collected using the API help understand various developer behaviors and have been used in several studies [11, 15–17, 24]. Indeed, in this work, we build on GitHub to gain insights into how developers use several CI/CD technologies. The information collected can guide future research on possible improvements to the DevOps

[1] https://github.com/features/actions
[2] https://docs.gitlab.com/ee/ci/
[3] https://www.travis-ci.com/
[4] https://www.jetbrains.com/teamcity
[5] https://www.jenkins.io/
[6] https://docs.github.com/en/rest/search

process and in particular for CI/CD. With this work, we thus aim to answer the following research questions:

RQ1 What characterizes the current landscape of CI/CD?
In addressing Research Question 1 (RQ1), our objective is to gain a comprehensive understanding of the prevailing utilization of diverse CI/CD technologies. Our aim is to identify patterns that can serve as catalysts for future (research) endeavors, both within industrial and academic contexts.
From our sample of repositories, containing 612557 individuals, we know that about 32.7% include some CI/CD technology. In Section 4 we explore the most used technologies and their relationship with programming languages. Indeed, projects in some languages do not rely often on CI/CD which deserves further investigation and possibly better solutions for those developers.

RQ2 Can a single CI/CD technology adequately meet the needs of a specific project?
With RQ2, our goal is to understand to which extent the technologies can serve the projects' needs.
Indeed, we have discovered that 30271 projects use at least two technologies and that some (few cases) use up to 13 different technologies at the same time. This motivates further investigation on the interoperability of the different technologies and the support developers have to work with multiple CI/CD technologies at the same time.

RQ3 What is the evolution of CI/CD usage over time?
With this research question, our goal is to find how projects change over time regarding the CI/CD technologies used. Our goal is also to get an overview of changes in pipelines over time.
Over the years, there have been two major technologies used. Moreover, it is clear that developers often change technologies. Since currently there is little support for these evolutions, this paves a very interesting path for future research.

We organize the rest of the document as follows: Section 2 presents several works related to our own. Those works focus on mining information about several DevOps aspects in various software repositories. In Section 3, we present the methodology used for this work. This methodology includes how we found the CI/CD technologies for this study, what information we have collected from the repositories and how it is 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 the said repositories. In Sections 4, 5, and 6, we showcase and analyze the data for each research question. In Section 7 we answer our RQs and provide several future research paths based of this work. Section 8 discusses the threats to the validity of our work. Finally, Section 9 presents our conclusions and future work.

2 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 CI/CD.

M. Golzadeh et al. undertook a longitudinal empirical study spanning nine years, aiming to gain deeper insights into the rapidly evolving CI landscape. Their analysis delved into the development history of 91,810 GitHub repositories hosting active npm packages employing at least one CI service. The authors scrutinized the frequency of various CI/CD tools, their combinations, and the occurrence of tool transitions.

Our research shares a similar objective, albeit expanding beyond the scope of the referenced paper. We conducted an analysis using a larger dataset comprising 612,557 repositories sourced from diverse origins, encompassing a wider array of project types and programming languages. Moreover, our study was conducted more recently.

While some of our findings align with the conclusions drawn by the authors, such as the prevalence and growth of GitHub Actions, as well as the occurrence of multiple CI/CD tool usage, we also observed disparities. Notably, we found that the number of repositories utilizing GitHub Actions was nearly double those employing Travis CI. Additionally, we identified a significant decline in Travis CI usage from 2019 onward.

In contrast to the authors' approach of providing insights into total migrations between different technologies, we focused specifically on the case of Travis CI due to its substantial usage and provided insights into its decline since 2019. Our analysis revealed that 53.7% of repositories using Travis CI in 2019 had migrated to GitHub Actions by 2023. Furthermore, we presented the percentage of technological changes per year.

Additionally, we explored the usage of CI/CD tools across different programming languages, offering insights such as the number of repositories employing CI/CD per language, the distribution of tools used per language, and CI/CD utilization per language..

Xu et al. [31] introduce 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 summarize the challenges of analyzing image repositories and the approaches that can address these challenges. These authors focus their work on technologies for deployment, while with our work we give a broader overview of the usage of CI/CD technologies.

Wu et al. [30] 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. This work is focused on a particular technology, whilst ours covers many.

Zahedi et al. [32] 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. Although the studied topic is related to CI/CD, we analyzed concrete software projects.

Mazrae et al. [27] present a qualitative study of CI/CD technologies usage, co-usage, and migration based on in-depth interviews. They identify reasons for the use of specific technologies, reasons

for co-usage of CI/CD technologies in the same project, and migrations executed by the interviewees. Their study reveals a clear trend in migration from Travis to GitHub Actions. We have used a very different source of data, but some of the conclusions of our study are in line with Mazrae et al.

Liu et al. [24] mine 84,475 open-source Android applications from GitHub, Bitbucket, and GitLab to search for CI/CD adoption. They find only around 10% applications leverage CI/CD technologies, a small number of applications (291) adopt multiple CI/CD technologies, nearly half of the applications that adopt CI/CD technologies do not really use them, and CI/CD technologies are useful to improve project popularity. Their approach is similar to ours however, our analysis is done with a greater sample of 612557 repositories and we don't limit ourselves to Android applications.

Calefato et al. [11] study MLOps (that is, DevOps but focused on machine learning projects) practices in GitHub repositories, focusing on GitHub Actions and CML. Their preliminary results suggest that the adoption of MLOps workflows is rather limited. On the other hand, we have found that, indeed, many projects rely on CI/CD pipelines.

Kumar et al. [21] assess the maturity of DevOps practices in the software industry. To this end, they analyze the HELENA2 dataset (an international survey aiming to collect data regarding the common use of software and systems in practice). They rank organizations by DevOps maturity. The authors conducted a user survey while we analyzed software repositories.

G. Destro et al [14]. developed Garimpeiro, a web application designed to assess the implementation of CI/CD practices in open-source repositories hosted on GitHub. Utilizing their tool, the authors conducted a comparative analysis by extracting data from two repositories, and examining the toolchains across various stages of the deployment pipeline.

In contrast, our approach diverges from the authors' as our objective is not solely to assist users in analyzing various software repositories based on their CI/CD utilization. Instead, we aim to offer insights into the contemporary landscape of CI/CD adoption across open-source software repositories.

J. Bernardo et al. [9] conducted an analysis of 162,653 pull requests originating from 87 GitHub projects, spanning across 5 distinct programming languages. Their empirical study aimed to explore the influence of CI adoption on the time required to merge pull requests. Interestingly, the findings revealed that only 51.3% of the projects exhibited faster merging times after implementing CI. Additionally, the authors highlighted a significant surge in pull request submissions following the adoption of CI, which they identified as a primary factor contributing to improved project outcomes.

In contrast to evaluating the advantages of CI/CD across software projects, our focus lies in comprehending the present status of CI/CD. Our objective is to provide insights that can aid developers and researchers in making informed decisions for the future.

M. Manglaviti et al [25]. examined 1,279 Travis CI GitHub repositories, focusing on the human resources involved in CI system development and maintenance. Their analysis revealed that the CI development team constitutes a minority within the overall developer team for each project. They noted a decrease in the proportion of CI developers as the number of total contributors increased. Additionally, the authors concluded that while there are development costs associated with CI adoption, the reported benefits outweigh these expenses.

In contrast to this study, our objective is not to assess the cost-effectiveness of CI/CD. Instead, our aim is to gain insights into the current landscape of CI/CD.

M. Rahman et al [26]. conducted an exploratory study utilizing 578K automated build entries to examine the influence of automated builds on code reviews. Their findings indicated that projects with frequent builds tend to maintain a consistent level of reviewing activities over time, contrasting with those with infrequent builds.

In contrast to Rahman et al.'s focus on the benefits of CI/CD, our work centers on providing an overview of its current status.

F. Zampetti et al [33]. developed 34 CI/CD pipelines by manually analyzing 615 pipeline configuration change commits. These taxonomies categorize actions that either enhance extra-functional properties or modify the behavior of the pipeline. Subsequently, the authors analyzed 4,644 Travis CI projects. Their analysis revealed that certain pipeline components, such as jobs and steps, undergo more frequent changes than others, and highlighted an increasing adoption of Docker over time.

In contrast to focusing on the evolution of pipeline changes over time, our objective is to examine the present utilization of CI/CD tools and their historical development.

Hilton et al. [18] studied CI by mining 34,544 OSS projects on GitHub and surveying 442 developers. They found many OSS teams that don't use CI. Of the ones that use CI, 90% used Travis. They find popular projects are more likely to use CI and that the median time for CI adoption is one year. Hilton et al.'s study was published in 2016, so its results are not representative of the current state of CI/CD.

Beller et al. [8], through an analysis of GitHub, found that Travis had seen a sharp increase in usage up to 2017, being used by one-third of popular projects. Just as Hilton et al.'s, Beller et al.'s study is not representative of the current state of CI/CD.

Decan et al. [13] study GHA use in nearly 70,000 GitHub repositories. They find 43.9% of repositories use GHA workflows. They also characterize these repositories according to which jobs, steps, and reusable Actions are used and how.

Vasilescu et al. [29] studied 1,884 GitHub projects in 2014. They found Travis usage in 918 repositories (48.7%).

Lamba et al. [23] study the spread of CI/CD in Node Package Manager (NPM) package repositories. Their analysis is done through repository badges, a recent innovation on code hosting platforms. They search for 12 CI/CD technologies in 168,510 NPM package repositories hosted on GitHub. Their study focuses on how CI/CD technologies gain market share.

JetBrains [] provides results from yearly surveys of developers about the developer ecosystem from 2017 to 2023. Their results show Jenkins as the most popular CI/CD technology until 2022 when GHA takes over. They also reveal the increasing relevance of CI/CD.

The Continuous Delivery Foundation, a project of the Linux Foundation, [2] provides a report on the state of CD. They find that 84% of developers participated in DevOps-related activities as of Q1 2023 and that the average number of DevOps-related technologies

used by developers concurrently remained stable from 2019 to 2023 at 4.5.

Stack Overflow's annual developer surveys [1] show an increase in CI/CD and DevOps usage year-over-year.

3 DATA COLLECTION

This section details the process we followed to collect the data for our analysis. Fig. 1 presents an overall view that we detail in the following paragraphs.

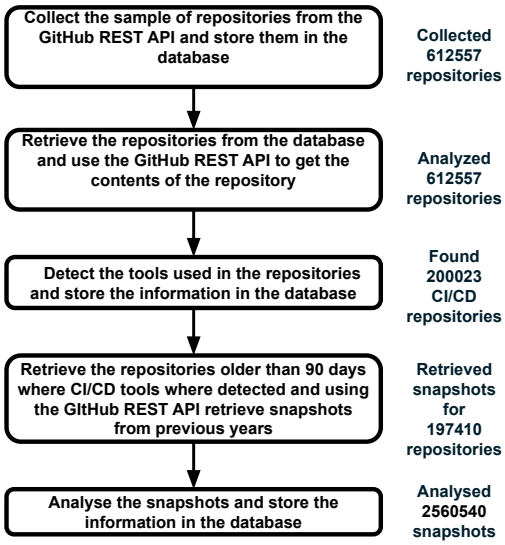

Figure 1: Data collection process.

3.1 Repository Sampling

The initial data collection phase involved assembling a representative sample of GitHub repositories, reflective of real-world projects. To achieve this, we excluded school projects and smaller repositories, employing a methodology inspired by prior research projects. That is, we focused on repositories attaining a certain level of popularity, ultimately choosing to collect only those with more than ten stars, accounting for a mere 5% of all GitHub repositories.

The sampling process encompassed repositories from the beginning of 2012 to October 18, 2023. This timeframe was selected because the concept of DevOps gained prominence around that period. This assertion is supported by the emergence of the first comprehensive survey on the state of DevOps by Puppet Labs in the same year [10].

Leveraging the GitHub REST API, we devised a methodology to identify the top one thousand repositories with the highest star count each week. That is, we queried GitHub and got one thousand repositories every week in the time range. From these, we excluded the ones with fewer than ten stars. This approach yielded a comprehensive dataset, totaling 612,557 repositories, spanning the specified date range. The entire dataset of repositories is provided for reference [6]. The code used to create this dataset is also available [3].

3.2 CI/CD Technologies

Following the collection phase, our objective was to discern the technologies employed in each repository. We used a dataset of 61 CI/CD technologies curated by the Cloud Native Computing Foundation [28], established by the Linux Foundation in 2015. We identified pertinent artifacts and patterns for each technology in the dataset, enabling us to automatically recognize repositories utilizing those technologies. This process involved analyzing file trees accessed through the GitHub REST API and scrutinizing file contents with extensions compatible with the respective technologies. Each technology was identified using one of two heuristics: *i)* some technologies use files with a particular extension; *ii)* for the others, we inspected specific types of files (e.g., YAML files) searching for content specific to the underlying technology. From this process, we divided the initial 61 technologies into different categories: *i)* 39 technologies that we could identify (Table 1); *ii)* 10 which we could not identify since there was not a clear artifact that we could use (Table 2); *iii)* 4 technologies that required the use of specific code to be identified as they are libraries embedded in the code (Table 3); and *iv)* 1 deprecated technology with no current documentation that made it impossible to recognize (Table 4).

Table 1: CI/CD technologies we can identify and analyze.

Agola	AppVeyor
ArgoCD	Bytebase
Cartographer	CircleCI
Cloud 66 Skycap	Cloudbees Codeship
Devtron	Flipt
GitLab	Google Cloud Build
Helmwave	Travis
Jenkins	JenkinsX
Keptn	Liquibase
Mergify	OctopusDeploy
OpenKruise	OpsMx
Ortelius	Screwdriver
Semaphore	TeamCity
werf	Woodpecker CI
GitHubActions	Codefresh
XL Deploy	Drone
Flagger	Harness.io
Flux	GoCD
Concourse	Kubernetes
AWS CodePipeline	

Table 2: CI/CD technologies that we cannot identify due to the lack of clearly identifiable artifacts.

Akuity	Bamboo
Buildkite	Bunnyshell
CAEPE	Keploy
Northflank	OpenGitOps
Ozone	Spacelift

Table 3: Libraries introducing unnecessary complexity.

Brigade	k6
OpenFeature	Unleash

Table 4: Deprecated technologies lacking documentation.

D2iQ Dispatch

3.3 The Repositories with CI/CD Technology

Our analysis focused on the sample acquired earlier, specifically the contents of the latest commit to the main branch of each repository. To ensure reproducibility, we recorded the SHA of the file tree (available in the database of our dataset). A total of 200,023 repositories were identified as utilizing one or more of the CI/CD technologies identified in the previous section. The comprehensive dataset containing the repositories and the corresponding technologies is also made available for further examination [5]. We use this dataset to analyze the state of CI/CD according to the most recent snapshots. The code used to create this dataset and the figures to support our answers to the research questions are also available [3].

To examine the past state of CI/CD, we retrieved snapshots of repositories over time and ran the same CI/CD technologies analysis on each snapshot. The comprehensive dataset containing the repositories and the corresponding snapshots and technologies is also made available for further examination [7]. The code used to create this dataset and the figures to support our answers to the research question are also available [4].

We ran a test on the 197410 repositories to choose the time interval for the analysis. Snapshots were retrieved at 90-day, 180-day, and 365-day intervals for each repository. For each snapshot, we then ran CI/CD technologies analysis. Our goal was to determine the number of changes in the CI/CD tech stack we would lose by increasing the retrieval interval, a change being any difference in the CI/CD stack compared to the previous snapshot. For the sample, we found 274606 stack changes at a 90-day sampling interval, 233115 changes at a 180-day sampling interval, and 195292 changes at a 365-day sampling interval. From a 90-day to a 180-day rate, there was a 15.1% decrease in the detected changes, and from a 180-day to a 365-day rate, there was a 16.2% decrease in detected changes. A lower sampling interval, or retrieving all commits for each repository, was not feasible due to GitHub API rate limits and the time for the study. Based on these results, we determined to run our analysis at a 90-day sampling rate.

A subset of the previously selected repositories was analyzed. The inclusion criteria were repositories where CI/CD technologies were detected in the latest commit and created before July 16, 2023 (so they were at least 90 days old). This resulted in a subset containing 197504 repositories.

For each selected repository, the first commit was retrieved. A sequential iterative process was employed, where the latest commit (if one existed) was retrieved for each 90-day interval starting from either January 1, 2012 or the repository's first commit date, whichever was later. This process continued until the date of the repository's last update at the time of retrieval. All commits were retrieved from the default branch of the repository. The git trees of the snapshot commits were then examined for CI/CD technology presence using the previously described methodology.

After all repositories were processed, we cleaned the retrieved data. Any snapshots from before January 1, 2012, were discarded, and the last snapshot of each repository was set to the one used in the previous analysis. For some snapshots, the GitHub API could not return a git tree. We attempted to process these snapshots again to eliminate any momentary API malfunction. The original detection method for GitHub Actions could lead to false detection when other technologies were present in the snapshot, so we reprocessed all snapshots that had more than one technology detected and GitHub Actions present. Lastly, we checked each snapshot's date and detected technologies against the detected technologies' launch dates and removed snapshots where a technology was detected before it was launched. If, at the end of these cleaning steps, a repository was left without snapshots, it was discarded.

From an initial 197504 repositories selected for temporal analysis, we retrieved the CI/CD technology use history of 197410. For the 94 repositories whose CI/CD technology history could not be retrieved, the reasons are as follows: the 19 repositories could not be processed in the initial snapshot retrieval because they had either been deleted or gone private, in the data cleaning step, another 39 repositories could not be processed because they had either been deleted or gone private, and 36 were discarded because they had no snapshots at the end of the cleaning steps.

4 RQ1: WHAT CHARACTERIZES THE CURRENT LANDSCAPE OF CI/CD?

By answering this RQ we intend to provide a clear and accurate overview of the current usage of CI/CD technologies in the open-source realm.

CI/CD usage

We collected 612,557 created between 2012 and 2023, that is, between the beginning of the establishment of DevOps and now. From these, 200,023, 32.7%, contain at least one CI/CD technology. This means about one third of the repositories in the last twelve years have, or had, some CI/CD support.

In Fig. 2, we depict the distribution of technologies within CI/CD projects, exclusively focusing on technologies whose usage exceeds 1% of the total repositories containing CI/CD technologies. In the online appendix of this paper[7], we present a chart with all the technologies. In fact, we present several other charts in this appendix. The figure indicates that GitHub Actions significantly dominates the landscape, being present in more than half (57.8%) of the repositories, with Travis trailing as the second most utilized technology (38.8%). In comparison, the majority of other technologies exhibit significantly lower usage rates in contrast to these leading two. Notably, among the 39 technologies analyzed, only ten surpass the 1% usage threshold across CI/CD repositories.

The predominance of GitHub Actions seems to indicate that repository platforms have quite some influence, at least in the

[7] https://sites.google.com/view/msr2024

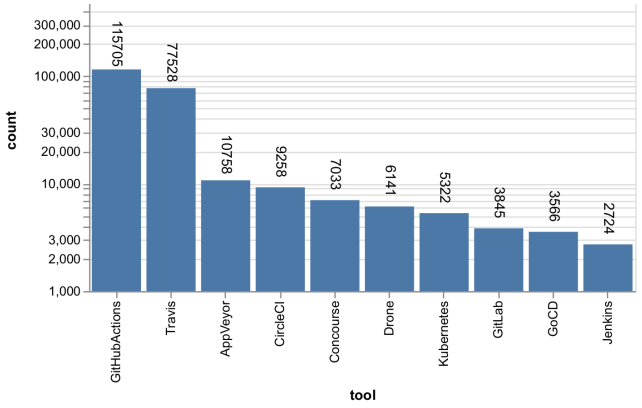

Figure 2: Distribution of technologies across CI/CD repositories (in a logarithmic scale).

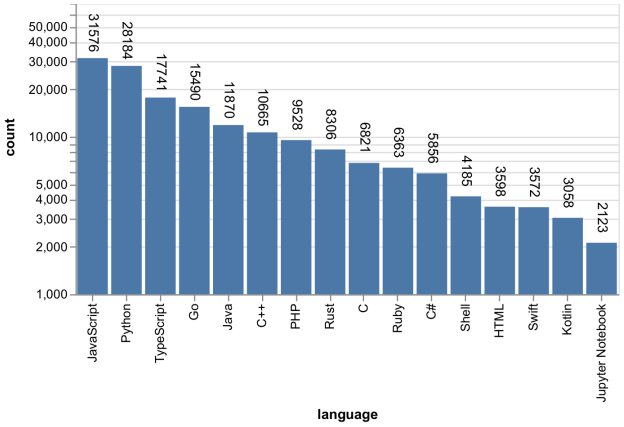

Figure 3: Programming languages in repositories containing CI/CD (logarithmic scale).

projects they hold. Thus, a possible conclusion is that, when considering the usage of CI/CD technology, developers also consider where the code will be managed.

CI/CD and programming languages

Fig. 3 illustrates the number of repositories of each programming language with a usage rate exceeding 1% in repositories employing at least one CI/CD technology. We use the GitHub API to get the main programming language of each repository (which does not mean the project does not use other languages). Notably, based on this depiction, JavaScript and Python emerge as the most extensively utilized languages. In absolute terms, these languages are the ones where more projects include CI/CD technologies. A comparison of the languages highlighted in the IEEE Spectrum's top programming languages of 2023 [19] reveals noteworthy disparities. While SQL and R claim top positions in IEEE Spectrum's ranking, they are absent from our list. This discrepancy may be attributed to SQL often being embedded within other programming languages and R being predominantly utilized by less technically inclined users, with less emphasis on system development. Additionally, Jupyter Notebooks appear on our list, albeit not reaching that popularity compared to IEEE Spectrum's rankings.

The fact that projects using languages such as R, a language so popular among users without a computer science (CS) background (e.g., researchers, mathematicians), do not usually include CI/CD technologies, seems to indicate CI/CD is of difficult adoption by users without a technical (CS) background.

Fig. 4 displays the prevalence of languages in CI/CD repositories, highlighting the percentage of sample repositories utilizing each language with and without CI/CD technologies. The figure indicates that languages like Go, Rust, and TypeScript exhibit the highest prevalence of CI/CD use within their repositories. Notably, despite Python and JavaScript having a substantial presence in CI/CD repositories, as illustrated in Fig. 3, they show a lower overall percentage of repositories utilizing CI/CD technologies.

This data seems to indicate more recent languages (e.g., Go, Kotlin, Rust, TypeScript) tend to include CI/CD in their projects, in some cases having more than 60% of all projects some technology.

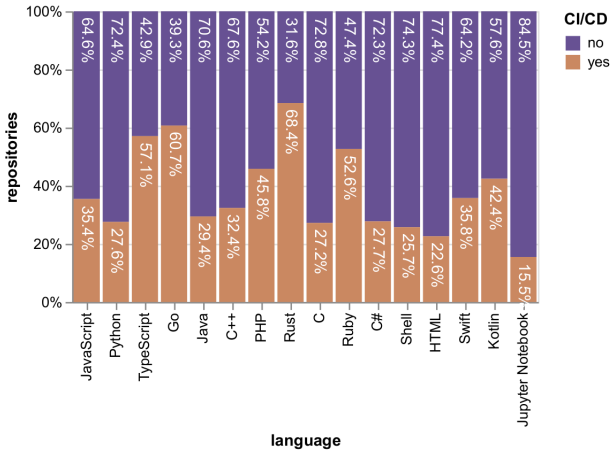

Figure 4: Percentage of CI/CD usage per language.

On the other hand, more classic languages (e.g., C, C#, HTML, Shell) seem to have less support from CI/CD technologies. From this, a possible conclusion is that CI/CD technologies find a better fit in modern languages and not so much in pre-existing ones.

Fig. 5 reveals that GitHub Actions and Travis stand out as the predominant technologies utilized across various programming languages. Notably, Travis demonstrates higher popularity in JavaScript projects, while GitHub Actions is more prominently employed in Python projects. This trend may be attributed to the synchronized growth of GitHub Actions and the increasing popularity of Python. Furthermore, GitHub Actions emerges as the preferred technology across most programming languages.

This analysis suggests that programming languages experiencing heightened popularity in recent years, like Python and TypeScript, are more commonly associated with GitHub Actions. Conversely, languages such as JavaScript are more prevalent in projects utilizing Travis. This observation implies that newer projects tend

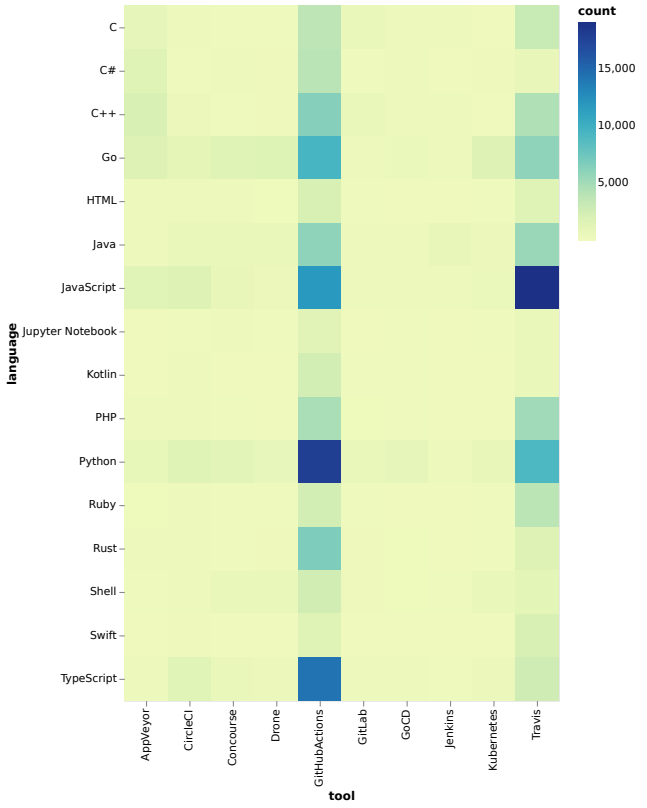

Figure 5: Heat map of technology use per programming language across CI/CD repositories.

to favor GitHub Actions, indicating a potential trend of migration from Travis to GitHub Actions, particularly in numerous JavaScript projects.

5 RQ2: CAN A SINGLE CI/CD TECHNOLOGY ADEQUATELY MEET THE NEEDS OF A SPECIFIC PROJECT?

With this RQ we intend to understand the co-usage of several CI/CD technologies among the projects.

Fig. 6 displays the logarithmic distribution of the number of CI/CD technologies employed across repositories with at least one such technology. A notable observation from the figure is that more than three quarters of projects (84.9%) opt for a single CI/CD technology. Nevertheless, 30271 repositories, constituting 15.1% of all repositories with CI/CD technologies, utilize more than one technology. This suggests that while a substantial portion of projects operates effectively with a single CI/CD technology, there exists a subset of projects that necessitate the utilization of multiple technologies. This implies that certain technologies may not be adequate or suitable for specific workloads, prompting the adoption of a diverse CI/CD technology stack in these instances.

The fact that many projects require using two or more CI/CD technologies raises several concerns. To the best of our knowledge, the interoperability of these technologies has not been properly

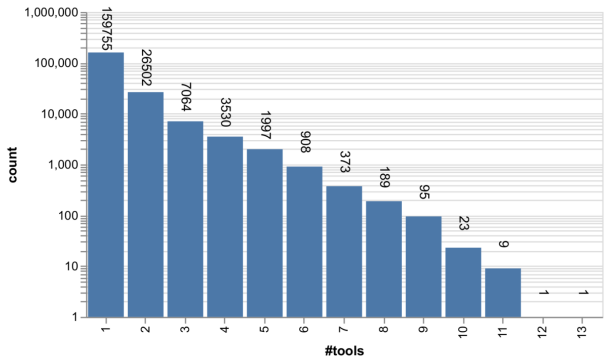

Figure 6: Distribution of the number of technologies in repositories containing CI/CD technologies (logarithmic scale).

studied. This deserves an investigation of its own. Moreover, CI/CD supporting tools (e.g., editors) are not tailored for co-usage, which makes their co-usage more difficult for developers. Thus, to study how to aid developers in these projects becomes an interesting research path.

In Fig. 7 and Fig. 8, we present the number and percentage of repositories using (at most) a certain number of technologies. Some repositories have no single technology because, at a certain point (most likely, in their beginning), they did not use CI/CD. In the first few years, one can see that the vast majority used just one technology, but that number quickly rose. In the last 4 years, at least 23,000 repositories (per year) use more than one technology (Fig. 9). Note this is simultaneous usage, not different usage over time (which we will address in RQ3). This finding agrees with both Mazrae et al. [27] and Goldazeh et al. [15] studies of CI usage.

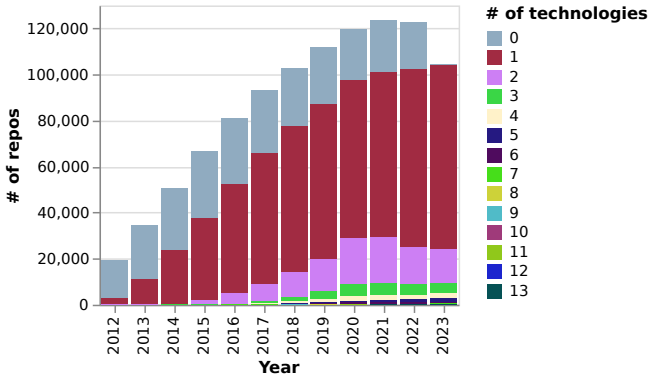

Figure 7: Number of repositories using at most a given number of technologies in a given year, by year.

Although in the last few years there seems to be a decrease in the use of multiple technologies, clearly, there are many repositories still using them that the research community should try to support.

In Fig. 10 we present a study of the co-usage of two technologies. Not surprisingly, GitHub Actions is used in combination with all

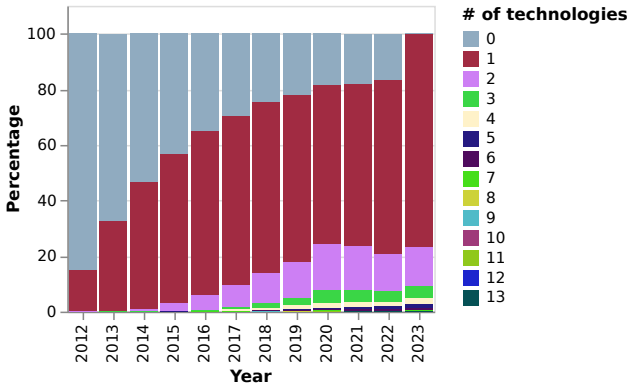

Figure 8: Percentage of repositories using at most a given number of technologies in a given year by year

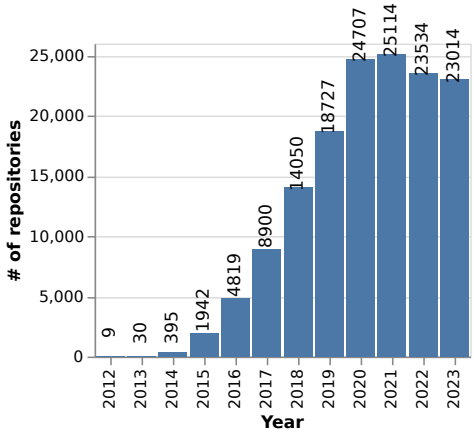

Figure 9: Number of repositories using more than one technology simultaneously in a given year, by year.

the other technologies, but most significantly with Travis. Travis itself is quite used with AppVeyor and Drone with Kubernetes.

The stronger combinations deserve more attention as they represent more use cases and in particular more users the research community may help.

Finally, in Fig. 5, we present the average number of CI/CD technologies used across different programming languages (we use the same set of languages as in RQ1). As can be seen, all the projects written in these languages use, on average, more than one CI/CD technology.

This reinforces our previous conclusions: the usage of multiple technologies is quite present and deserves further study and support.

6 RQ3: WHAT IS THE EVOLUTION OF CI/CD USAGE OVER TIME?

With this RQ, we intend to understand the usage of CI/CD over the last 12 years.

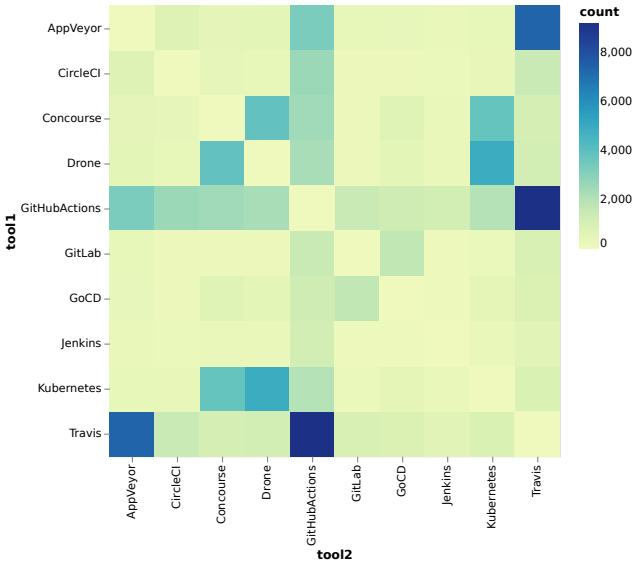

Figure 10: Heat map with the combination of technologies co-usage.

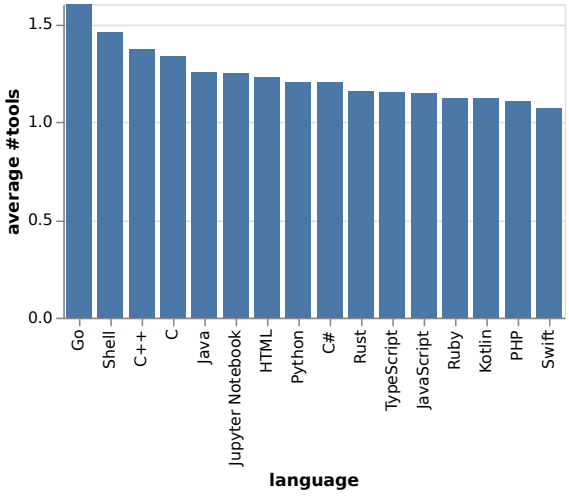

Figure 11: Average number of tools used per programming language repository.

We start by analyzing the usage of CI/CD starting in 2012 until now. For Fig. 12, we count the number of repositories created each year where, in their last commit, there is the presence of CI/CD technologies. As can be seen, even in repositories from 2012, almost one quarter included at least one technology; the percentage then increased until 2015 to more than 37%, and has been since then in a gentle descending/stable curve. The drop in the last year is justified because many projects start without CI/CD, and only later, it is introduced.

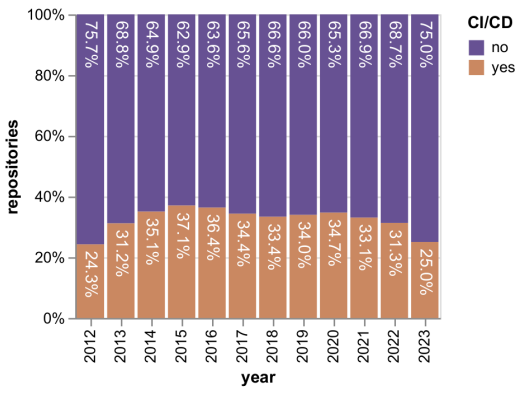

Figure 12: The usage of CI/CD per year.

As seen in Fig. 13, CI/CD technologies are being integrated into the development workflow sooner as time goes on. CI/CD's increasing popularity is due to older, possibly more complex, projects' adoption and new, perhaps simpler, projects that see value in continuous practices. The sharp decreases in 2022 and 2023 come from all analyzed repositories having at least one CI/CD technology in 2023.

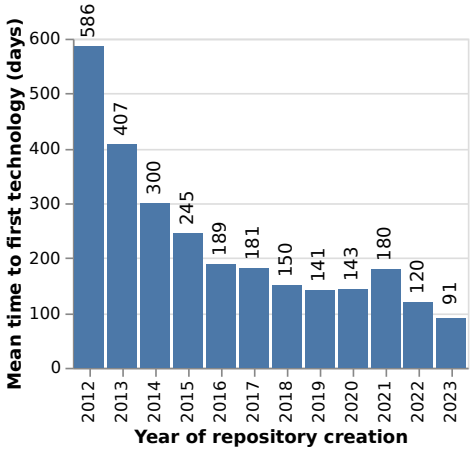

Figure 13: Mean time in days to first CI/CD technology detection by repository creation year.

Usage of different technologies

A repository may use more than one technology at a time, as seen in Fig. 14. The technologies used in a repository in a given year are the union of the technologies of the snapshots retrieved in that year. In cases where a repository does not have any snapshots in a given year but has activity following that year, the technologies used for that year are the ones in the most recent snapshot up to that point in time, i.e., if a repository has snapshots in 2016 and 2018, but not in 2017, its used technologies in 2017 are the ones from the last snapshot from 2016. If a repository has no activity following a given year, its technology usage stops being considered. This methodology applies to all charts showing technology usage over time.

Fig. 14 shows two significant trends in CI/CD, Travis and GitHub Actions. Travis usage steadily increased from 2012 until it peaked in 2019 with 73284 repositories (36.6%). Since 2019, Travis's usage has been declining. This coincides with the rapid adoption of GitHub Actions; from 2019 to 2020, there was a 502.8% growth in the number of repositories using GitHub Actions, and from 2020 to 2021, there was an 86.6% growth. Of the 36587 repositories that used Travis in 2019 and were still active in 2023, 59.7% were using GitHub Actions and not Travis in 2023, 21.1% used Travis and not GitHub Actions, and 16.9% used both. Of the 87582 repositories using GitHub Actions in 2023, 45.4% had no snapshots from before 2020. The exodus from Travis and the influx from newer repositories have been the main drivers for GitHub Actions growth.

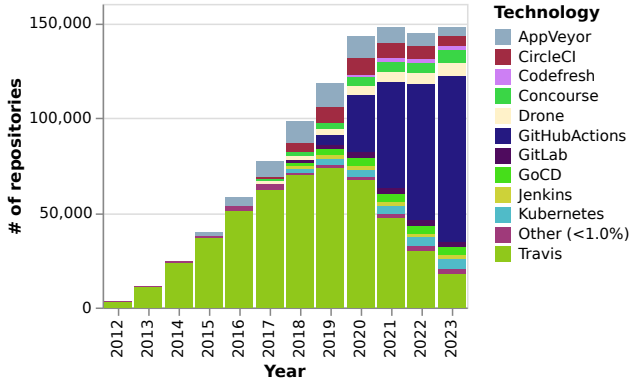

Figure 14: Number of repositories where each technology was detected by year.

From this data, we can see a shift in the CI/CD technologies used over the years. This could mean developers tend to change their technologies and may need support for doing so. We thus investigate this shift further on.

Fig. 15 shows the top 10 CI/CD stack transitions from repositories that solely used Travis in 2019. While many stop being active, 22.9% moved from Travis to GitHub Actions. If we consider only the ones active, this 53.7% of all active Travis projects moved to GitHub Actions while.

From this information, we can conclude that developers actually change CI/CD technologies. However, there is little support for this kind of change. Thus, the research community has quite an interesting path to explore.

How often do projects change CI/CD technologies?

As Fig. 16 shows, the percentage of snapshots with CI/CD changes compared to the previous snapshot grows steadily from 2013 (2.3%) to 2019 (6.9%) and peaks in 2020 (12.2%) and 2021 (12.6%), coinciding with GitHub Actions's explosive growth phase. Since 2021, this number has remained stable at almost 8%.

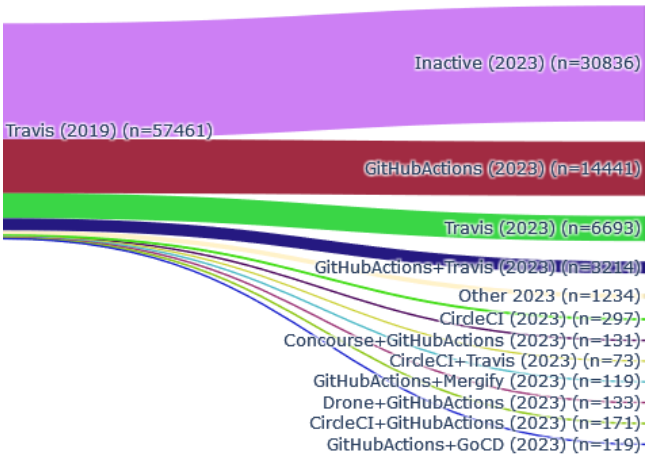

Figure 15: CI CI/CD technology stack transitions from repositories solely using Travis CI in 2019.

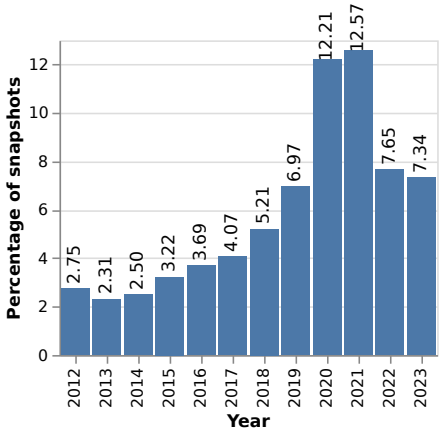

Figure 16: Percentage of snapshots with changes in the CI/CD technology stack from the previous snapshot by year (all repositories).

This is a very significant result since it shows that every year, between 2.3% (in 2013) and 12.6% (in 2021) of all snapshots include some kind of change in the CI/CD technologies used. This represents a very significant amount of technologies' shift, with all the known issues with that. Moreover, it is important to notice that this constant over time and there is no reason to think this may change in a near future. This means the research community can give a significant contribution to aid all these teams when they migrate and evolve their systems.

If we limit this analysis to a subset of active repositories from 2012 to 2023 (Fig. 17), we find similar trends but higher change percentages over time. For this analysis, we discarded all snapshots for each repository before the first, where we detected CI/CD technologies. This was done because we were not interested in a project's first choice of technologies.

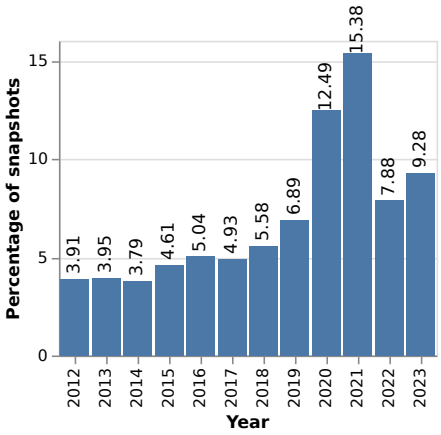

Figure 17: Percentage of snapshots with changes in the CI/CD technology stack from the previous snapshot by year for the set of repositories active from 2012 to 2023 (n=8296).

The percentage of CI/CD technology stack changes is significant and is consistently higher in long-running projects, meaning projects continuously look for technologies that better fit their workflow. Although one could expect some stability over the years for mature projects, this is not the case as can been seen in the data. This reinforces the need for the research community to provide support for these changes.

7 DISCUSSION

We have analyzed the 12 years of the usage of CI/CD. Answering RQ1, from the more than 600 thousand repositories collected, almost one third, 32.7%, includes CI/CD technologies. This number is expected to grow as many projects start without the usage of CI/CD, which is added in later phases. We can also see one of the current technologies dominates the market, with 57.8% of the projects using GitHub Actions. This number may be biased as we used GitHub as our source of projects, but, if this is the case, it means the source code manager has a significant impact in the choice of CI/CD technologies.

Regarding the relationship with programming languages, it is possible to conclude some languages are more related to the usage of CI/CD than others. In particular, more recent languages, such as TypeScript, Go, or Rust have a higher percentage of projects with CI/CD than languages such as C, Java, or C#. This may indicate a technological-generational gap between languages and CI/CD technologies, which should further be studied aiming to improve the adoption of CI/CD by older languages.

Another interesting aspect is the fact that top-used languages such as R [19] do not have projects that include CI/CD in a significant amount. This may indicate this kind of technology is not of easy adoption by less technically skilled users, as R is often used by researchers and mathematicians.

A surprising aspect we discovered during our work related to RQ2 is that many projects use more than one technology for CI/CD. In fact, up to 13 technologies may be used at the same time. In the last 4 years, each year, more than 23,000 projects include more than

one technology, which accounts for about 20% of all projects. From this, is possible to answer RQ2 negatively, that is, one technology is often not sufficient to cope with all the requirements of some projects. This raises several future work directions. A first path is to endeavor a deeper investigation is required to understand exactly why this happens. Moreover, it is necessary to investigate the interoperability of the technologies, how they work together, and how to provide the proper support for these users, as existing technologies tend to have tools (e.g., editors, interpreters) designed to be used independently of others.

In the analysis for RQ3, we can see about a third of the repositories have CI/CD, a number that is relatively stable over the years. However, we have shown the time between the creation of the repository and the adoption of CI/CD has been decreasing substantially. Regarding the usage over the years, with RQ3 we discovered developers tend to change quite often CI/CD technologies. Indeed, there has been a massive change from Travis to GHA, but changes in technology are quite common. Indeed, in the last two years, more than 7% of all snapshots include some kind of change in the CI/CD technological stack. If we consider long-run projects (projects running from 2012 up until now), this number is even greater. This means developers need support to be successful in these endeavors, which currently is mostly non-existent. This opens quite promising research paths, including techniques to support the evolution of this kind of artifacts (e.g., model-driven approaches, languages-based), but also for the human-computer interaction community on how to aid these developers being more effective and efficient.

8 THREATS TO VALIDITY

There are multiple threats to the validity of our study, which we address in the following paragraphs.

Our study focuses on open-source software and, in particular, on projects hosted on GitHub. Thus, our sampling does not include other kinds of software (e.g., propriety). Thus, we cannot generalize our conclusions to these other kinds of software projects. Nevertheless, many companies also have their software on GitHub, and one may expect workers from these companies to use similar technologies in other projects. Moreover, others have reached similar conclusions by interviewing developers [15].

Since we only used GitHub, we cannot say these results apply to projects in other code repository services. However, there is no reason to consider projects hosted on GitHub to be significantly different from other projects in other repository services.

Still regarding the use of GitHub as the source of the software projects we analyzed, we could observe a predominance of GitHub Actions. One of the main reasons for this may in fact be related to the use of GitHub as the source of projects. However, we also found GitLab Actions, the CI/CD technology used by another repository service (GitLab).

We collected our sample repositories by getting the 1,000 results sent by the GitHub API, doing it for every week in our time frame. This gave us more than 600,000 repositories, from which more than 200,000 have CI/CD. Although we could have collected more repositories, this would increase the time to retrieve them in a way that would make our work unfeasible. Moreover, the query did not impose any restriction on the results, except for the 10 starts we used to have some kind of "quality" metric for the projects. Thus, the repositories retrieved should not be biased in any other way.

We considered only technologies that we could identify through files in the repository. Indeed, from the 61 technologies identified by the Cloud Native Computing Foundation, we could not identify 14 technologies (plus 1 deprecated). Nevertheless, we were able to identify 64% of all technologies.

Some technologies are detected through file contents and we cannot guarantee a random file would not have a certain string inside that matches. However, we defined content that would only make sense in the technology context, this probably did not happen. In any case, if it happened, was for a very small number of files that should not change the overall conclusions of our work.

We assumed that the presence of CI/CD artifacts (e.g., configuration files) means the underlying project is using such a technology. However, this may not be the case as some artifacts may be left forgotten from older usages.

9 CONCLUSIONS

In this work, we have investigated more than 600,000 software projects, from which more than 200,000 include diverse CI/CD technologies. We have characterized the current usage of CI/CD, related the technologies with programming languages, discovered that many projects use several technologies at the same time, and that projects tend to change their CI/CD technologies frequently. From this, several research paths can be seen, from better understanding why developers need to use several technologies simultaneously and to provide the proper support to them, to how to aid when developers want to evolve their technologies.

As future work, we plan to extend this work to other repository services such as GitLab, or Bitbucket. Another interesting research direction is to replicate this work in an industrial setting. We will also extend our work to consider the complete DevOps cycle.

REFERENCES

[1] [n. d.]. *Stack Overflow Insights - Developer Hiring, Marketing, and User Research.* https://survey.stackoverflow.co/

[2] [n. d.]. *State of Continuous Delivery Report 2023: The Evolution of Software Delivery Performance.* https://cd.foundation/state-of-cd-2023/

[3] Anonymous. 2023. https://anonymous.4open.science/r/DevOps-Repositories-122E

[4] Anonymous. 2023. https://anonymous.4open.science/r/github-devops-mining-C0D7

[5] Anonymous Author. 2023. CI/CD repos with tools. https://doi.org/10.6084/m9.figshare.24578740.v2

[6] Anonymous Author. 2023. CI/CD repositories from GitHub. https://doi.org/10.6084/m9.figshare.24578746.v1

[7] Anonymous Author. 2023. CI/CD repositories with tool history. https://doi.org/10.6084/m9.figshare.24578752.v1

[8] Moritz Beller, Georgios Gousios, and Andy Zaidman. [n. d.]. Oops, My Tests Broke the Build: An Explorative Analysis of Travis CI with GitHub. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)* (Buenos Aires, Argentina, 2017-05). IEEE, 356–367. https://doi.org/10.1109/MSR.2017.62

[9] João Helis Bernardo, Daniel Alencar da Costa, and Uirá Kulesza. 2018. Studying the impact of adopting continuous integration on the delivery time of pull requests. In *Proceedings of the 15th International Conference on Mining Software Repositories* (Gothenburg, Sweden) *(MSR '18)*. Association for Computing Machinery, New York, NY, USA, 131–141. https://doi.org/10.1145/3196398.3196421

[10] Puppet by Perforce. 2023. https://www.puppet.com/resources/history-of-devops-reports accessed: 16/11/2023.

[11] Fabio Calefato, Filippo Lanubile, and Luigi Quaranta. 2022. A Preliminary Investigation of MLOps Practices in GitHub. In *Proceedings of the 16th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*

(Helsinki, Finland) *(ESEM '22)*. Association for Computing Machinery, New York, NY, USA, 283–288. https://doi.org/10.1145/3544902.3546636

[12] The Business Research Company. 2022. Information Technology Global Market Report 2022, By Type, Organization Size, End User Industry. https://www.researchandmarkets.com/reports/5561700/information-technology-global-market-report-2022

[13] Alexandre Decan, Tom Mens, Pooya Rostami Mazrae, and Mehdi Golzadeh. 2022. On the Use of GitHub Actions in Software Development Repositories. In *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (Limassol, Cyprus, 2022-10). IEEE, 235–245. https://doi.org/10.1109/ICSME55016.2022.00029

[14] Gabriel Augusto Destro and Breno Bernard Nicolau de França. 2020. Mining Software Repositories for the Characterization of Continuous Integration and Delivery. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering* (, Natal, Brazil,) *(SBES '20)*. Association for Computing Machinery, New York, NY, USA, 664–669. https://doi.org/10.1145/3422392.3422503

[15] Mehdi Golzadeh, Alexandre Decan, and Tom Mens. 2022. On the rise and fall of CI services in GitHub. In *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)* (Honolulu, HI, USA, 2022-03). IEEE, New York, New York, United States, 662–672. https://doi.org/10.1109/SANER53432.2022.00084

[16] Mubin Ul Haque, Leonardo Horn Iwaya, and M. Ali Babar. 2020. 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)* (Bari, Italy) *(ESEM '20)*. Association for Computing Machinery, New York, NY, USA, Article 7, 11 pages. https://doi.org/10.1145/3382494.3410693

[17] Jordan Henkel, Christian Bird, Shuvendu K. Lahiri, and Thomas Reps. 2020. Learning from, Understanding, and Supporting DevOps Artifacts for Docker. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. Association for Computing Machinery, New York, NY, United States, 38–49. https://doi.org/10.1145/3377811.3380406

[18] Michael Hilton, Timothy Tunnell, Kai Huang, Darko Marinov, and Danny Dig. [n. d.]. Usage, costs, and benefits of continuous integration in open-source projects. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering* (Singapore Singapore, 2016-08-25). ACM, 426–437. https://doi.org/10.1145/2970276.2970358

[19] IEEE. 2023. https://spectrum.ieee.org/the-top-programming-languages-2023 accessed: 17/11/2023.

[20] G. Kim, J. Humble, P. Debois, and J. Willis. 2016. *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution Press, Portland, Oregon, United States. https://books.google.pt/books?id=ui8hDgAAQBAJ

[21] Ankur Kumar, Mohammad Nadeem, and Mohammad Shameem. 2022. Assessing the Maturity of DevOps Practices in Software Industry: An Empirical Study of HELENA2 Dataset. In *Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering* (Gothenburg, Sweden) *(EASE '22)*. Association for Computing Machinery, New York, NY, USA, 428–432. https://doi.org/10.1145/3530019.3531335

[22] GitHub Staff Kyle Daigle. 2023. Octoverse: The State of Open Source and rise of AI in 2023. https://github.blog/2023-11-08-the-state-of-open-source-and-ai/

[23] Hemank Lamba, Asher Trockman, Daniel Armanios, Christian Kästner, Heather Miller, and Bogdan Vasilescu. 2020. Heard it through the Gitvine: an empirical study of tool diffusion across the npm ecosystem. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Virtual Event USA, 2020-11-08). ACM, 505–517. https://doi.org/10.1145/3368089.3409705

[24] Pei Liu, Xiaoyu Sun, Yanjie Zhao, Yonghui Liu, John Grundy, and Li Li. 2023. A First Look at CI/CD Adoptions in Open-Source Android Apps. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering* (Rochester, MI, USA) *(ASE '22)*. Association for Computing Machinery, New York, NY, USA, Article 201, 6 pages. https://doi.org/10.1145/3551349.3561341

[25] Marco Manglaviti, Eduardo Coronado-Montoya, Keheliya Gallaba, and Shane McIntosh. 2017. An Empirical Study of the Personnel Overhead of Continuous Integration. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. 471–474. https://doi.org/10.1109/MSR.2017.31

[26] Mohammad Masudur Rahman and Chanchal K. Roy. 2017. Impact of Continuous Integration on Code Reviews. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. 499–502. https://doi.org/10.1109/MSR.2017.39

[27] Pooya Rostami Mazrae, Tom Mens, Mehdi Golzadeh, and Alexandre Decan. 2023. On the usage, co-usage and migration of CI/CD tools: A qualitative analysis. *Empirical Software Engineering* 28, 2 (2023), 52. https://doi.org/10.1007/s10664-022-10285-5

[28] Case Study. 2023. Cloud native computing foundation. https://www.cncf.io/

[29] Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. 2015. Quality and productivity outcomes relating to continuous integration in GitHub. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering* (Bergamo Italy, 2015-08-30). ACM, 805–816. https://doi.org/10.1145/2786805.2786850

[30] Yiwen Wu, Yang Zhang, Tao Wang, and Huaimin Wang. 2020. An Empirical Study of Build Failures in the Docker Context. In *Proceedings of the 17th International Conference on Mining Software Repositories* (Seoul, Republic of Korea) *(MSR '20)*. Association for Computing Machinery, New York, NY, USA, 76–80. https://doi.org/10.1145/3379597.3387483

[31] Tianyin Xu and Darko Marinov. 2018. Mining Container Image Repositories for Software Configuration and Beyond. In *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results* (Gothenburg, Sweden) *(ICSE-NIER '18)*. Association for Computing Machinery, New York, NY, USA, 49–52. https://doi.org/10.1145/3183399.3183403

[32] Mansooreh Zahedi, Roshan Namal Rajapakse, and Muhammad Ali Babar. 2020. Mining Questions Asked about Continuous Software Engineering: A Case Study of Stack Overflow. In *Proceedings of the Evaluation and Assessment in Software Engineering* (Trondheim, Norway) *(EASE '20)*. Association for Computing Machinery, New York, NY, USA, 41–50. https://doi.org/10.1145/3383219.3383224

[33] Fiorella Zampetti, Salvatore Geremia, Gabriele Bavota, and Massimiliano Di Penta. 2021. CI/CD Pipelines Evolution and Restructuring: A Qualitative and Quantitative Study. In *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 471–482. https://doi.org/10.1109/ICSME52107.2021.00048

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009