IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 12, Issue 1, January 2024

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.379

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201073|

A Comprehensive Empirical Study Determining Practitioners' Views on Docker Development Difficulties: Stack Overflow Analysis

Varun Kumar Tambi[1], Nishan Singh[2]

Vice President (Software Engineer, Product Manager), JPMorgan Chase & Co., United States of America[1]

Consultant, EXL SERVICE COM INDIA PVT LTD, India[2]

ABSTRACT: Through an examination of Stack Overflow posts, this study explores Docker-related subjects and difficulties in order to pinpoint common problems and patterns in Docker development. We created an extensive dataset of Docker conversations by utilising both tag-based and content-based filtering techniques. These conversations were categorised using Latent Dirichlet Allocation (LDA) topic modelling, which showed that the most common theme is Application Development, which includes topics like Framework Management, Coding Problems, Data Transfer, and Docker-specific frameworks. Because of Docker's strong market presence, developers are very interested in using it to build and manage apps, as evidenced by this dominant category. The study highlights important areas of interest and difficulty within the Docker community and offers insightful information on the many technical and operational problems faced by Docker developers. These findings offer guidance for future research, tool development, and educational resources, aiming to address the evolving needs of Docker practitioners and enhance the overall development experience.

KEYWORDS: Docker, Stack, Overflow, framework, Learning Curve, Development Efficiency, Compatibility Issues, Tool Development.

I. INTRODUCTION

The adoption of containerization technologies has transformed the landscape of software development, with Docker emerging as one of the most popular and widely used tools for building, shipping, and running applications. Docker provides a lightweight, consistent environment across various stages of development, from testing to production, enabling developers to manage dependencies, streamline workflows, and enhance the overall efficiency of software delivery [1]. Despite its advantages, the complexities and challenges inherent in Docker development are significant, especially as projects scale in size and complexity. These challenges encompass a wide array of technical, operational, and organizational issues, ranging from the intricacies of container orchestration and networking to security vulnerabilities and the steep learning curve for practitioners new to containerization.

To better understand these challenges from the perspective of those directly engaged in Docker development, it is essential to conduct a large-scale empirical study that delves into the real-world experiences of practitioners. This study leverages data from Stack Overflow, one of the most comprehensive and widely-used platforms where developers seek help, share knowledge, and discuss problems related to software development. By analyzing the questions, discussions, and solutions presented on Stack Overflow, this research aims to identify the recurring challenges and pain points that practitioners encounter in Docker development [15]. The platform's vast repository of community-driven content provides a rich dataset for understanding the practical difficulties developers face, how they resolve these issues, and what common themes emerge across different domains and use cases.

The focus of this research is not only on identifying the technical challenges but also on uncovering the broader implications of these difficulties on software development practices. For instance, issues related to Docker's integration with continuous integration/continuous deployment (CI/CD) pipelines, managing stateful applications, and ensuring compatibility across diverse environments are all critical aspects that can influence project success or failure [5]. Additionally, this study aims to explore how the community's collective knowledge evolves over time, reflecting the dynamic nature of technology adoption and the ongoing learning process within the developer community.

Through a detailed analysis of the data gathered from Stack Overflow, this study seeks to provide a comprehensive overview of the challenges that developers face when working with Docker at scale. By doing so, it will contribute to

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201073|

the broader understanding of containerization's impact on software development, offering valuable insights for practitioners, educators, and tool developers. The findings could inform best practices, guide the development of more robust tools and frameworks, and ultimately help to mitigate the challenges associated with Docker, enabling smoother, more efficient development workflows in the future [8].

Each year, Stack Overflow (SoF) conducts a survey that gathers insights from developers on their preferred technologies and job preferences. In the 2019 survey, Docker stood out significantly: it was ranked first in the "Most Wanted Platform," second in the "Most Loved Platform," and third in the "Platform In Use" categories. This was Docker's debut in the survey, highlighting the strong interest developers have in container technologies.

The field of software engineering is evolving, with a growing emphasis on DevOps practices. DevOps aims to streamline the software development lifecycle by fostering close collaboration between development and operations teams. One key aspect of this approach is the use of containers and microservices, which help in creating modular, reusable components that can be easily developed and deployed [10]. Docker has been a leader in this area, providing a popular solution for packaging and shipping software efficiently.

The widespread adoption of Docker has led to a vibrant community of developers who discuss, troubleshoot, and share their experiences on platforms like Stack Overflow. This community-generated content, including questions and answers about Docker, serves as a valuable resource for understanding common issues and best practices. However, with the rapid growth of container technologies, developers face new challenges. Mastery of various domains such as networking, operating systems, cloud computing, and software engineering becomes crucial. By analyzing the discussions and questions on Stack Overflow related to Docker, we can identify the most common difficulties developers encounter and areas where they seek help [11-12]. This understanding can guide both practitioners in addressing their challenges and researchers in focusing their studies, ultimately benefiting the broader developer community.

II. LITERATURE REVIEW

The literature surrounding Docker development and containerization technologies is rich with studies that explore various facets of their adoption, challenges, and impacts on software engineering practices. Docker, introduced in 2013, revolutionized the software development landscape by offering a solution to the age-old problem of "it works on my machine" through the encapsulation of applications and their dependencies into containers. This innovation has spurred a significant body of research that examines the technical, operational, and organizational implications of containerization, particularly as it has gained widespread acceptance in both industry and academia.

DevOps builds on lean and agile methodologies by aiming for full automation throughout the software development and delivery process. Instead of following a strict, one-size-fits-all guide, which isn't practical, DevOps encourages developers to integrate and streamline the traditionally separate areas of development and operations. By doing so, it connects these previously isolated teams more effectively. There are many tools available that can support this integration, making it easier for developers to automate their workflows and improve efficiency [14].

One major stream of research has focused on the technical challenges associated with Docker and container orchestration. Studies have highlighted issues such as the complexity of container orchestration, especially when managing large-scale distributed systems. Tools like Kubernetes, which have been developed to address these challenges, introduce their own set of complexities, including steep learning curves and difficulties in configuring and optimizing deployments. Research by Nirmata (2017) points out that while orchestration platforms simplify the deployment of containers, they also require a deep understanding of distributed systems, which can be a significant barrier for developers. Moreover, issues related to network configuration, security vulnerabilities, and resource management are recurrent themes in the literature. For instance, the work of Grigoriu et al. (2019) delves into the security aspects of Docker, emphasizing the risks associated with container breakout attacks and the importance of implementing robust security measures.

Another important aspect of the literature explores the performance trade-offs of using Docker containers. Several studies have examined the overhead introduced by containerization, particularly in comparison to traditional virtualization techniques. Felter et al. (2015) conducted one of the seminal studies in this area, comparing the performance of Docker containers with virtual machines (VMs). Their findings indicate that Docker containers offer near-native performance, with significantly lower overhead compared to VMs. However, subsequent research has

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201073|

nuanced this understanding by exploring specific use cases where Docker's performance may degrade, such as in high I/O operations or in scenarios requiring fine-grained resource allocation. These studies suggest that while Docker is generally efficient, developers need to carefully consider the specific requirements of their applications when deciding whether to containerize.

The literature also extensively discusses the integration of Docker into continuous integration/continuous deployment (CI/CD) pipelines, which has become a cornerstone of modern DevOps practices. Studies have shown that Docker can significantly accelerate CI/CD workflows by providing consistent environments across development, testing, and production stages. However, challenges remain, particularly in managing stateful applications and ensuring data persistence across containerized environments. Research by Leitner et al. (2019) highlights the difficulties of maintaining state in containerized applications, which traditionally thrive in stateless environments. This has led to the development of new patterns and practices, such as using external databases or distributed file systems, to manage state outside of containers.

From an organizational perspective, the literature reveals a spectrum of challenges related to the adoption of Docker at scale. Transitioning from traditional deployment methods to containerized environments often requires significant changes in organizational culture, workflows, and tooling. Studies by Pahl (2015) and Mieso et al. (2018) discuss the adoption curve of Docker in enterprises, noting that while the benefits of faster deployments, scalability, and improved resource utilization are clear, the transition can be fraught with difficulties. These include the need for retraining staff, re-architecting legacy systems, and managing the increased complexity of containerized microservices architectures. Furthermore, the adoption of Docker is often accompanied by the need for new governance and compliance strategies, particularly in industries with stringent regulatory requirements.

In addition to empirical studies, there is a growing body of work that provides theoretical frameworks for understanding the implications of Docker and containerization on software architecture and engineering practices. Researchers like Baldini et al. (2017) have proposed models for assessing the impact of containerization on system architecture, emphasizing the shift towards microservices and serverless computing models. These frameworks help in understanding how containerization aligns with broader trends in software architecture, such as the move towards more modular, scalable, and resilient systems.

The literature on Docker development also reflects the evolving nature of the technology itself. As Docker and related tools continue to evolve, so too do the challenges and best practices associated with their use. This dynamic nature is evident in the continuous updates to the Docker platform, the emergence of new tools and frameworks for managing containers, and the ongoing discussions within the developer community, particularly on platforms like Stack Overflow. This evolving landscape suggests that the challenges identified in current literature may change as the technology matures and as practitioners develop more sophisticated approaches to managing containerized environments.

The literature on Docker development is extensive and multifaceted, covering technical, operational, and organizational challenges. It provides valuable insights into the benefits and limitations of containerization, offering guidance for both practitioners and researchers interested in understanding and optimizing the use of Docker in modern software development. The ongoing evolution of Docker and related technologies ensures that this field will continue to be a rich area for research and discussion, with new challenges and solutions emerging as the technology and its use cases expand.

Cloud computing has become a highly important and profitable area in the tech industry. Organizations of all sizes are moving to cloud platforms for a variety of reasons. While cloud providers are working hard to make this transition as smooth as possible, we haven't fully perfected the process yet. However, by using the right tools and best practices, we can make the move to the cloud more efficient and effective [16]. The success of cloud automation and platform orchestration relies on several key layers working together seamlessly. This paper offers an overview of cloud technology and outlines how it's changing industrial automation. It highlights how cloud computing can boost productivity and optimize costs. Additionally, the paper discusses strategies for deploying and maintaining cloud resources efficiently, ensuring they can be managed, provisioned quickly, and released with minimal administrative effort.

The Mann-Kendall Trend Test, often referred to as the MK test, is a statistical tool used to determine whether a time series data exhibits a consistent upward or downward trend over time. This test is non-parametric, meaning it doesn’t rely on assumptions about the data's distribution, such as normality, which makes it versatile for different types of data.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201073|

However, the test assumes that the data does not exhibit serial correlation, which means that the value at one point in time should not be dependent on previous values. If there is serial correlation, it can distort the significance level of the test and lead to incorrect conclusions.

To address issues with serial correlation and improve accuracy, several variations of the Mann-Kendall test have been developed. For example, the Hamed and Rao Modified MK Test and the Yue and Wang Modified MK Test are designed to correct for serial correlation. Another adaptation, the Modified MK Test using the Pre-Whitening method, also helps in dealing with serial correlation. Additionally, the Seasonal Mann-Kendall test has been developed to account for seasonal effects in the data.

The Mann-Kendall test is highly regarded for its ability to detect trends, but its applications extend beyond basic trend analysis. Other modified versions, such as the Multivariate MK Test, Regional MK Test, Correlated MK Test, and Partial MK Test, have been developed to handle specific conditions related to spatial data [18].

Mesos is a platform designed to efficiently manage and share resources across different cluster computing frameworks, such as Hadoop and MPI. Traditional cluster computing often involves separate clusters for each framework, which can lead to underutilization of resources and redundant data storage. Mesos addresses these issues by allowing multiple frameworks to share a single set of commodity hardware. This sharing improves overall cluster utilization and reduces the need to duplicate data across different frameworks. To support the sophisticated scheduling needs of modern frameworks, Mesos uses a distributed two-level scheduling system known as resource offers. In this system, Mesos first decides how many resources to allocate and then offers these resources to different frameworks. Each framework can then choose which resources to accept and determine what tasks or computations to perform with them. This mechanism allows frameworks to optimize their resource usage based on their specific needs [19]. The effectiveness of Mesos is demonstrated by its ability to achieve near-optimal data locality even when multiple frameworks share the same cluster. It has been tested to scale up to 50,000 nodes in emulated environments and has shown resilience to system failures. This scalability and robustness make Mesos a powerful tool for managing large and diverse computing environments efficiently.

III. OBJECTIVE OF THE STUDY

The objective of the study is to:

1. Classify and analyze the types of issues developers encounter when using Docker, focusing on technical and operational challenges across various aspects of Docker application development.
2. Explore the trends in Docker-related discussions to highlight areas of growing interest and emerging difficulties, providing a clearer understanding of the evolving landscape of Docker technology.

IV. METHODOLOGY

In the initial phase of our analysis, we utilized the Stack Overflow (SoF) dataset, which is accessible through the SOTorrent [10] database. This dataset comprises a comprehensive collection of questions and answers, each accompanied by a variety of data points. For each post, the dataset includes details such as the post's unique identifier, type (whether it's a question or an answer), title, body content, associated tags, creation date, view count, score, number of favorites, and the identifier of the accepted answer if applicable. A question's answer is deemed accepted when the original poster marks it as such. Additionally, each question can be associated with a minimum of one and a maximum of five tags. To extract a substantial portion of this dataset, we used Google's BigQuery [19] platform, retrieving a total of 46,947,633 posts, which includes both questions and answers, spanning over 11 years, from August 2008 to December 2019. These posts were contributed by 4,533,602 developers. Of the total posts, 18,597,996 (39.61%) are questions, while 28,248,207 (60.16%) are answers. Notably, 9,731,117 answers (20.72%) have been marked as accepted, indicating an acceptance rate of 34.45%.

4.1 Constructing a Docker-Focused Dataset

To begin addressing our research questions, we first identified a subset of SoF questions that specifically pertain to Docker-related topics.

Tag-Based Filtering

In this analytical step, we developed a set of Docker-related tags, denoted as T, to isolate and extract Docker-related questions from the SoF dataset. We initiated the process with an initial set of Docker tags, referred to as $T0$, which

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201073|

included the tag 'docker.' Next, we extracted the subset of questions P from our overall dataset S, where the tags matched those in $T0$. We then created a set of candidate tags $T1$ by identifying the tags associated with these Docker-related questions. Finally, we refined the tag set T by retaining only those tags that were highly relevant to Docker, excluding any that were not. To assess the significance and relevance of each tag within the Docker tag set T, we applied two heuristics, α and β.

4.2 Content-Based Filtering

In some cases, Docker-related posts on Stack Overflow might not include any of the identified Docker tags. For instance, the post with question ID 26787241 discusses issues related to Docker container linking but lacks any Docker-specific tags. This gap exists because there is no standardized method for assigning tags to Stack Overflow posts, and some Docker-related discussions may not be properly tagged. This observation led us to analyze the content of the posts, in addition to the tags, to ensure we captured all relevant Docker discussions.

For content-based filtering, we followed the methodology outlined by Le et al.. We began by compiling a list of Docker-related keywords, drawing inspiration from the approach used by Pletea et al.. To minimize false positives, we employed subword matching for keywords longer than three characters, as recommended. We then calculated two parameters: the ratio of Docker-related words ($kwratio$) and the total count of Docker keywords present ($kwcount$). These parameters were used to broaden the scope of our Docker-related post selection, helping to filter out irrelevant discussions. For example, the post with ID 56408913 includes the term 'docker,' but the main focus of the discussion is on the installation of Yourls, a URL shortening service.

To refine our filtering, we established thresholds for $kwratio$ and $kwcount$, determined as $x = 0.051$ and $y = 4$, respectively, based on unclosed Docker-related posts ($Posttag$). Since closed posts may be duplicates or irrelevant to the discussion, we focused on unclosed posts to ensure they were genuinely relevant to Docker. Using this content-based filtering approach, we selected Stack Overflow posts that met or exceeded the thresholds for $kwratio$ and $kwcount$, thereby building a dataset ($Postcontent$) of Docker-related discussions that might not have been tagged appropriately but still focused on Docker topics.

V. RESULT AND DISCUSSION

RQ1: Docker Topics – What Topics Do Developers Ask About?

Docker development spans a wide range of software engineering disciplines, requiring developers to be knowledgeable in areas like cloud computing, operating systems, and distributed networking, which aren't always necessary in more traditional software development roles. This diverse knowledge requirement means that the challenges Docker developers face are likely different from those encountered in other areas of software development. Given that developers often turn to Q&A websites like Stack Overflow (SoF) to seek solutions and share problems, this research question aims to identify the most common topics related to Docker that developers are discussing. Understanding these topics is crucial, as it helps to pinpoint which areas of Docker are generating the most interest or are particularly challenging. Additionally, identifying these topics is a key first step in highlighting trends in Docker's popularity and areas where developers may need more support or resources. Docker has garnered significant attention from the developer community, with many eager to build applications using the platform, making it essential to delve deeper into the topics they are discussing.

To identify the Docker topics that developers are asking about on SoF, we used Latent Dirichlet Allocation (LDA) topic modeling and topic labeling, as outlined in our methodology. This approach allows us to systematically categorize and analyze the discussions surrounding Docker on the platform.

The findings are presented in Table 1, which lists the Docker-related topics, their associated keywords, and their broader categories. Additionally, Figure 1 illustrates the percentage distribution of these topics and how they are categorized. From Figure 1, it is evident that the most frequently asked questions revolve around "Basic Concepts" of Docker. Other prominent topics include Framework Management, Data Transfer, Docker Networking, Commanding and Scripting, Environment Modification, File Management, Docker Desktop, Repository, and Continuous Integration. Each of these topics has a higher representation than the average percentage of 3.3% across all topics.

These 30 distinct topics have been grouped into 13 main categories for better clarity. In the following sections, we will delve into each of these categories, providing explanations, examples of related posts from SoF, and discussing their

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201073|

relevance to previous research. This will offer a comprehensive view of the key issues and areas of interest within the Docker community.

5.1 Application Development

The Application Development category encompasses topics like Framework Management, Coding Issues, Compilation, Data Transfer, Web Browsers, Python, and Framework Development. This category is the largest, accounting for approximately 21% of all topics, highlighting the significant interest developers have in using Docker for application development. This finding aligns which discusses developers' perspectives on leveraging Docker for various applications. One reason for this strong focus on Application Development could be Docker's substantial market presence, with a reported market share of nearly 83%. Within this category, posts typically relate to using Docker in the development of applications, including how frameworks are utilized, managed, and implemented. For example, developers may seek advice on using Docker to develop a web application (e.g., post ID: 48788271, titled "How to add Docker support for a web application developed using .NET Core and Angular 5?"), troubleshoot issues when building a Docker image (e.g., post ID: 48762638, titled "Error while building Docker container with .NET Core"), or address challenges in running an application with Docker (e.g., post ID: 37263261, titled "Docker web application image needs to keep running").

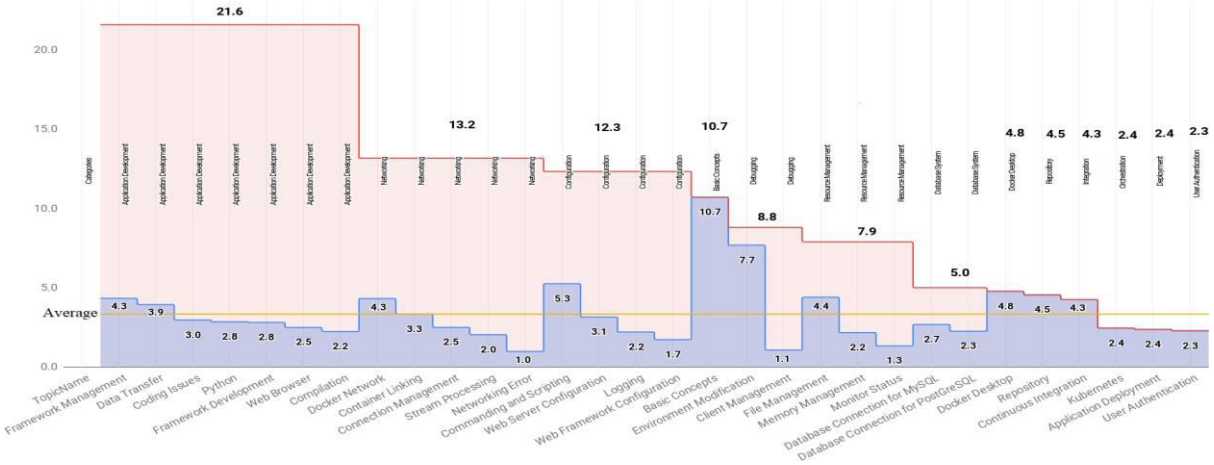

Figure 1: Distribution of Docker-related topics and their higher level categorization (in percentage).

Table 1: Names, categories (separated by tab) and top 10 words (stemmed topic words) for our Docker topics of SoF

Topic_No	Topic Name	Topic words	Categories
1	Stream Processing	node, cluster, kafka, connect, master, rabbitmq, messag, zookeep, worker, spark	Networking
2	Container Linking	volum, network, container_nam, redi, environ, mongo, mongodb, link, depends_on, restart_alway	Networking
3	Logging	info, log, debug, elasticsearch, warn, info_main, configur, fail, pid, messag	Configuration
4	File Management	volum, directori, mount, folder, data, path, copi, insid, local, share	Resource Management
5	Basic Concepts	applic, deploy, app, differ, environ, develop, possibl, base, exampl, make	Basic Concepts
6	Memory Management	memori, process, time, size, limit, set, data, cpu, number, thread	Resource Management
7	Networking Error	peer, network, order, debu, chaincod, note_innodb, innodb, fail, channel, utc	Networking
8	Framework Management	copi, project, java, applic, step, app, add, workdir, directori, expos	Application Development
9	Docker Desktop	window, machin, vm, linux, ssh, default, ubuntu, connect, set, mac	Docker Desktop
10	Web Server Configuration	nginx, proxi, request, locat, configur, http, root, set, app, url	Configuration
11	Monitor Status	id, latest, status, ps, minut_ago, hour_ago, minut, pull_complet, pull, second_ago	Resource Management
12	Database Connection for PostGreSQL	databas, postgr, connect, db, postgresql, user, web_1, data, migrat, log	Database System
13	Commanding and Scripting	script, echo, bash, execut, exec, set, cmd, entrypoint, variabl, env	Configuration
14	Connection Management	connect, fail, client, connect_refus, curl, tcp, localhost, socket, request, listen	Networking
15	Kubernetes	pod, kubernet, kubectl, node, deploy, cluster, spec, kubelet, apivers_kind, minikub	Orchestration
16	Database Connection for MySQL	mysql, db, databas, connect, volum, link, wordpress, root, environ, user	Database System
17	Coding Issues	string, return, true, null, function, valu, code, fals, var, id	Application Development
18	Repository	registri, pull, push, tag, repositori, login, certif, authent, password, user	Repository
19	Environment Modification	chang, issu, problem, updat, restart, time, remov, stop, found, solut	Debugging
20	Compilation	compil, librari, packag, make, modul, tensorflow, load, devic, model, fail	Application Development
21	Data Transfer	updat, packag, env, copi, add, echo, rm, sudo, curl, mkdir	Application Development
22	Web Framework Configuration	php, apach, compos, configur, directori, nginx, set, extens, exec, laravel	Configuration
23	Continuous Integration	jenkin, test, stage, script, pipelin, plugin, job, step, git, gitlab	Integration
24	User Authentication	user, root, sudo, fail, root_root, permiss, group, daemon, directori, process	User Authentication
25	Application Deployment	instanc, deploy, aw, task, ec, app, set, configur, azur, cluster	Deployment
26	Web Browser	test, app, browser, web, chrome, page, url, selenium, heroku, remot	Application Development
27	Docker Network	network, ip, access, ip_address, connect, node, swarm, address, default, expos	Networking
28	Python	line, python, import, pip_instal, return, django, app, flask, code, print	Application Development
29	Client Management	fals, client, api, git_commit, info, kernel, id, true, debug_mode, default	Debugging
30	Framework Development	npm, node, app, copi, npm_err, nodej, yarn, depend, code, directori	Application Development

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201073|

VI. CONCLUSION

In this study, we conducted a comprehensive analysis of Docker-related posts on Stack Overflow to understand the topics that developers frequently discuss and the challenges they encounter. By employing a combination of tag-based and content-based filtering, we were able to create a robust dataset of Docker discussions, ensuring that we captured both well-tagged and more obscure posts relevant to Docker. Our analysis, particularly through LDA topic modeling, revealed that Application Development is the most dominant category, reflecting developers' strong focus on using Docker for building, managing, and deploying applications.

The significant interest in Application Development underscores the importance of Docker in modern software engineering, where developers are keen to leverage Docker's capabilities for creating scalable, portable, and efficient applications. The findings also highlight the diverse range of skills required for Docker development, which spans multiple disciplines, including cloud computing, networking, and software engineering.

Our research contributes to the understanding of Docker's role in the developer community, offering insights into the common issues and areas of interest that are likely to shape future developments and discussions in Docker technology. These insights can help guide future research, tool development, and educational efforts aimed at addressing the challenges faced by Docker developers, ultimately contributing to the continued growth and improvement of Docker as a pivotal technology in the software industry.

REFERENCES

1. Hervé Abdi. 2007. The Kendall rank correlation coefficient. Encyclopedia of Measurement and Statistics. Sage, Thousand Oaks, CA (2007), 508–510.
2. Amritanshu Agrawal, Wei Fu, and Tim Menzies. 2018. What is wrong with topic modeling? and how to fix it using search-based software engineering. Information and Software Technology 98 (2018), 74–88.
3. Syed Ahmed and Mehdi Bagherzadeh. 2018. What do concurrency developers ask about? a large-scale study using stack overflow. In Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). 1–10.
4. Moayad Alshangiti, Hitesh Sapkota, Pradeep K Murukannaiah, Xumin Liu, and Qi Yu. 2019. Why is Developing Machine Learning Applications Challenging? A Study on Stack Overflow Posts. In 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). IEEE, 1–11.
5. C. Anderson. 2015. Docker [Software engineering]. IEEE Software 32, 3 (2015), 102–c3.
6. Muhammad Asaduzzaman, Ahmed Shah Mashiyat, Chanchal K Roy, and Kevin A Schneider. 2013. Answering questions about unanswered questions of stack overflow. In 2013 10th Working Conference on Mining Software Repositories (MSR). IEEE, 97–100.
7. Mehdi Bagherzadeh and Raffi Khatchadourian. 2019. Going big: a large-scale study on what big data developers ask. In Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 432–442.
8. Kartik Bajaj, Karthik Pattabiraman, and Ali Mesbah. 2014. Mining questions asked by web developers. In Proceedings of the 11th Working Conference on Mining Software Repositories. 112–121.
9. Sebastian Baltes, Lorik Dumani, Christoph Treude, and Stephan Diehl. 2018. SOTorrent: reconstructing and analyzing the evolution of stack overflow posts. In Proceedings of the 15th International Conference on Mining Software Repositories, MSR 2018, Gothenburg, Sweden, May 28-29, 2018. ACM, 319–330. https://doi.org/10.1145/3196398.3196430
10. Abdul Ali Bangash, Hareem Sahar, Shaiful Chowdhury, Alexander William Wong, Abram Hindle, and Karim Ali. 2019. What do developers know about machine learning: a study of ML discussions on StackOverflow. In 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR). IEEE, 260– 264.
11. Anton Barua, Stephen W Thomas, and Ahmed E Hassan. 2014. What are devel- opers talking about? an analysis of topics and trends in stack overflow. Empirical Software Engineering 19, 3 (2014), 619–654.
12. David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. Journal of machine Learning research 3, Jan (2003), 993–1022.
13. J. Cito, G. Schermann, J. E. Wittern, P. Leitner, S. Zumberi, and H. C. Gall. 2017. An Empirical Analysis of the Docker Container Ecosystem on GitHub. In 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR). 323–333.
14. Christof Ebert, Gorka Gallardo, Josune Hernantes, and Nicolas Serrano. 2016. DevOps. IEEE Software 33, 3 (2016), 94–100.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 | A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 1, January 2024 ||

| DOI: 10.15680/IJIRCCE.2024.1201073|

15. Sally Fincher and Josh Tenenberg. 2005. Making sense of card sorting data. Expert Systems 22, 3 (2005), 89–93.
16. Somya Garg and Satvik Garg. 2019. Automated Cloud Infrastructure, Continu- ous Integration and Continuous Delivery using Docker with Robust Container Security. In 2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR). IEEE, 467–470.
17. Gensim. 2020. gensim: Topic modelling for humans. Retrieved April 7, 2020 from https://radimrehurek.com/gensim/
18. Md. Hussain and Ishtiak Mahmud. 2019. pyMannKendall: a python package for non parametric Mann Kendall family of trend tests. Journal of Open
19. Google. 2020. Big Query. Retrieved April 7, 2020 from https://cloud.google.com/ bigquery
20. Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D Joseph, Randy H Katz, Scott Shenker, and Ion Stoica. 2011. Mesos: A platform for fine-grained resource sharing in the data center.. In NSDI, Vol. 11. 22–22.

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor: 8.379

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details