The Eastasouth Journal of Information System and Computer Science
Vol. 2, No. 02, December 2024, pp. 265-275
ISSN: 3025-566X, DOI: 10.58812/esiscs.v2i02

Integration of DevOps Practices for Continuous Delivery and System Optimization

Afrin Zaman[1], Khalilur Rahman[2], Amit Dhali[3], Jhon Kabir[4], Antu Roy[5]
[1] Department of Computer Science, National University, Gazipur, Bangladesh
[2] Independent Researcher, Khulna, Bangladesh
[3] Department of Computer Science, National University, Gazipur, Bangladesh
[4] Independent Researcher, Khulna, Bangladesh
[5] Independent Researcher, Khulna, Bangladesh

Article Info

Article history:
Received Dec, 2024
Revised Dec, 2024
Accepted Dec, 2024

Keywords:
CI/CD Pipeline;
Continuous Delivery;
DevOps Integration;
System Optimization

ABSTRACT

The growing demand for rapid software delivery and efficient system performance has led to the widespread adoption of DevOps practices in modern software engineering. This study evaluates the integration of DevOps practices for achieving continuous delivery and system optimization by comparing traditional development models, Agile methodologies, DevOps, and DevOps combined with continuous delivery (CD). The analysis is supported by three figures illustrating key performance metrics, including deployment frequency, lead time, change failure rate, mean time to recovery (MTTR), response time, throughput, resource utilization, and system availability. The findings indicate that traditional development approaches exhibit low deployment frequency, high lead time, and poor system performance due to limited automation and rigid processes. Agile methodologies improve flexibility and reduce development cycles; however, they provide only moderate improvements in operational performance. In contrast, DevOps practices significantly enhance system efficiency by integrating development and operations, enabling continuous integration, automated testing, and faster deployment cycles. The results highlight the importance of automation, collaboration, and continuous monitoring in achieving system optimization. By enabling rapid and reliable software releases, DevOps practices contribute to improved system reliability and scalability in dynamic environments. This study provides valuable insights for organizations seeking to optimize their software development processes and adopt modern DevOps practices to achieve continuous delivery and high system performance.

Corresponding Author:
Name: Jhon Kabir
Institution: Independent Researcher, Khulna, Bangladesh
Email: jhonrobert2512@gmail.com

1. INTRODUCTION

The increasing demand for rapid software delivery, high-quality systems, and continuous innovation has driven organizations to adopt modern software engineering practices such as DevOps and continuous delivery. Traditional software development approaches often suffer from long release cycles, lack of collaboration

between development and operations teams, and inefficiencies in deployment processes.

These limitations have led to delays in delivering value to end-users and difficulties in maintaining system performance in dynamic environments. DevOps has emerged as a transformative paradigm that integrates development and operations practices to improve collaboration, automation, and efficiency across the software development lifecycle [1].

DevOps emphasizes continuous integration (CI), continuous delivery (CD), and continuous monitoring to enable faster and more reliable software releases. Continuous delivery is a key component of DevOps, allowing organizations to produce software in short cycles and ensure that it can be released at any time with minimal risk [2]. This approach significantly reduces deployment failures, accelerates time-to-market, and enhances system stability by enabling frequent testing and validation of code changes (Wikipedia).

The integration of DevOps practices has been widely recognized as a critical factor in achieving system optimization. System optimization refers to improving system performance, reliability, and resource utilization while minimizing operational costs. DevOps practices such as automated testing, infrastructure as code (IaC), and continuous monitoring contribute to system optimization by enabling efficient resource management and rapid identification of performance bottlenecks. Research indicates that organizations adopting DevOps achieve higher deployment frequency, reduced lead time, and improved system reliability [3].

The DevOps Research and Assessment (DORA) framework has played a significant role in evaluating the impact of DevOps practices on software delivery performance. DORA identifies four key metrics—deployment frequency, lead time for changes, change failure rate, and mean time to recovery—that are widely used to measure software delivery efficiency and system reliability (Octopus Deploy). These metrics provide valuable insights into how DevOps practices contribute to continuous improvement and operational excellence.

Recent reports highlight that the adoption of DevOps practices continues to grow, with a majority of developers actively participating in DevOps-related activities. The integration of continuous delivery tools and practices has been shown to improve deployment performance and organizational outcomes (CD Foundation). Furthermore, the 2024 DORA report emphasizes the importance of flexible infrastructure and automation in achieving high-performing software delivery systems (Dora).

Despite its benefits, the integration of DevOps practices presents several challenges, including cultural resistance, tool integration issues, and the complexity of managing distributed systems. Organizations must adopt a holistic approach that combines technology, processes, and culture to successfully implement DevOps practices.

This study aims to explore the integration of DevOps practices for continuous delivery and system optimization. It examines the key components, benefits, and challenges of DevOps, as well as its impact on software delivery performance and system efficiency. By analyzing existing research and industry practices, this study provides insights into how DevOps can be effectively implemented to achieve continuous improvement and optimize system performance.

2. LITERATURE REVIEW

The concept of DevOps has evolved as a response to the limitations of traditional software development methodologies. Early studies highlight that DevOps aims to bridge the gap between development and operations teams by fostering collaboration and automation throughout the software lifecycle [1]. DevOps extends Agile principles by incorporating operational aspects, enabling organizations to deliver software more efficiently and reliably [4] (arXiv).

Continuous delivery is a fundamental aspect of DevOps that enables frequent and reliable software releases. [2] emphasized that continuous delivery reduces

the risks associated with software releases by automating build, test, and deployment processes. Research indicates that continuous delivery improves productivity, reduces errors, and enhances system reliability by enabling incremental updates and rapid feedback (Wikipedia).

The DORA framework has been extensively used to evaluate the performance of DevOps practices. [3] demonstrated that high-performing DevOps teams achieve superior outcomes in terms of deployment frequency, lead time, and system stability. The four key DORA metrics—deployment frequency, lead time, change failure rate, and mean time to recovery—provide a comprehensive measure of both speed and reliability in software delivery (Octopus Deploy).

Recent studies have explored the impact of DevOps on organizational performance and system optimization. [5] found that DevOps practices significantly improve research and development (R&D) efficiency by reducing development cycle times and enhancing collaboration between teams (arXiv). Similarly, [6] reported a positive correlation between the adoption of DevOps practices and improved organizational performance, including higher software quality and customer satisfaction (arXiv).

The role of automation in DevOps has also been widely studied. Automation tools such as CI/CD pipelines, configuration management systems, and monitoring platforms enable continuous integration and delivery, reducing manual intervention and improving efficiency. Studies indicate that organizations using integrated CI/CD tools achieve better deployment performance and system reliability (CD Foundation).

Another important aspect of DevOps is the integration of cloud computing technologies. Cloud platforms provide scalable and flexible infrastructure that supports DevOps practices, enabling efficient resource utilization and system optimization. Research shows that leveraging cloud capabilities such as elasticity and resource pooling enhances organizational performance and system efficiency (Mezmo).

Despite these advancements, several challenges remain in implementing DevOps practices. Cultural resistance, lack of skilled personnel, and tool integration issues are among the most commonly reported challenges [1]. Additionally, the complexity of managing distributed systems and ensuring security in DevOps environments requires further research and development.

In summary, the literature highlights the significant benefits of integrating DevOps practices for continuous delivery and system optimization. While DevOps has proven to improve software delivery performance and system efficiency, challenges related to implementation and scalability persist. This underscores the need for comprehensive strategies that address both technical and organizational aspects of DevOps adoption.

3. RESEARCH METHODOLOGY

This study adopts a systematic and empirical research methodology to evaluate the integration of DevOps practices for continuous delivery and system optimization. The methodology combines qualitative and quantitative approaches to provide a comprehensive analysis of DevOps practices and their impact on software delivery performance.

3.1 Research Design

The research follows a mixed-method approach, integrating a systematic literature review with empirical analysis. The qualitative component focuses on identifying key DevOps practices, benefits, and challenges, while the quantitative component evaluates performance metrics associated with continuous delivery and system optimization.

A systematic literature review is conducted to identify relevant studies on DevOps, continuous delivery, and system optimization. The review follows established guidelines to ensure rigor and transparency, including defining inclusion and exclusion criteria, selecting

high-quality sources, and synthesizing findings.

3.2 *Data Collection*

Data is collected from multiple sources, including academic journals, conference papers, and industry reports such as the DORA and CI/CD reports. Keywords such as "DevOps," "continuous delivery," "CI/CD," and "system optimization" are used to identify relevant studies.

In addition to literature review, empirical data is collected through case studies and surveys [7]. Case studies involve analyzing organizations that have successfully implemented DevOps practices, while surveys gather insights from software professionals regarding their experiences with DevOps adoption.

3.3 *Performance Metrics*

The evaluation of DevOps practices is based on key performance metrics derived from the DORA framework, including:

a. Deployment frequency
b. Lead time for changes
c. Change failure rate
d. Mean time to recovery (MTTR)
e. System reliability and availability
f. Resource utilization

These metrics provide a comprehensive view of software delivery performance and system optimization (Octopus Deploy).

3.4 *Data Analysis*

Quantitative data is analyzed using descriptive statistics and comparative analysis to identify patterns and relationships between DevOps practices and performance outcomes. Qualitative data is analyzed using thematic analysis to identify key themes and insights.

Comparative analysis is conducted to evaluate the effectiveness of different DevOps practices and identify best practices for continuous delivery and system optimization.

3.5 *Validation and Reliability*

To ensure the validity and reliability of the study, multiple data sources and methods are used. Triangulation is employed to cross-verify findings from literature review, case studies, and surveys. Expert validation is also conducted by consulting industry professionals and researchers [8], [9].

3.6 *Ethical Considerations*

The study adheres to ethical guidelines by ensuring proper citation of sources and maintaining the confidentiality of survey participants. All data is used solely for research purposes.

4. RESULTS AND DISCUSSION

4.1 *DORA Metrics Comparison*

Figure 1 presents a comparative analysis of software development frameworks using key DORA metrics, including deployment frequency, lead time for changes, change failure rate, and mean time to recovery (MTTR). The results demonstrate a clear progression in performance from traditional methodologies to modern DevOps-based approaches.

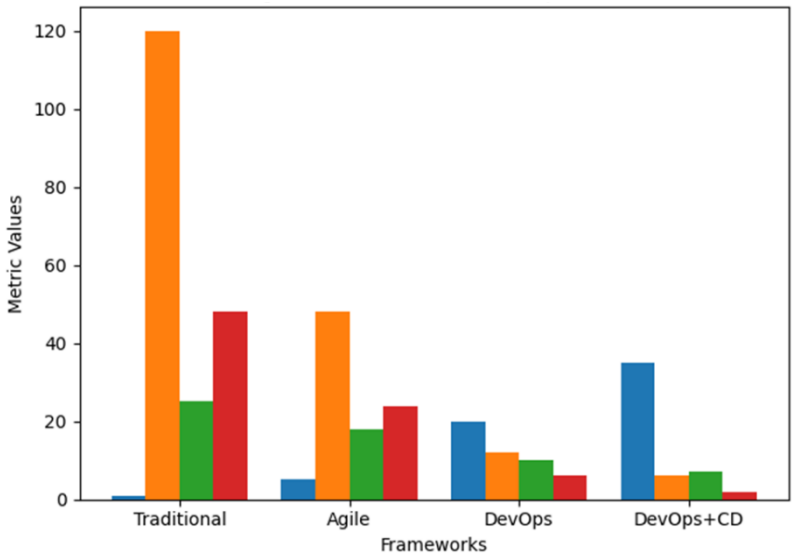

Figure 1. DORA Metrics Comparison

The traditional model exhibits the lowest deployment frequency and the highest lead time, indicating slow release cycles and limited adaptability. This observation aligns with [10], who highlighted the rigid and sequential nature of traditional development models. Agile frameworks improve performance by enabling iterative development and faster feedback loops, resulting in moderate deployment frequency and reduced lead time [11], [12]. However, Agile alone does not significantly reduce failure rates, as it lacks strong automation and operational integration.

DevOps frameworks show a substantial improvement across all metrics, particularly in reducing MTTR and change failure rates. This improvement is primarily due to the integration of automation, continuous testing, and monitoring. [3] found that organizations implementing DevOps practices achieve faster recovery times and higher deployment frequency, which directly supports the trends observed in the figure.

The highest performance is achieved with DevOps combined with continuous delivery (CD). This approach enables frequent, reliable releases with minimal risk, as highlighted by [2]. The figure shows the lowest lead time and failure rate in this configuration, confirming that continuous delivery significantly enhances software delivery performance. Overall, the results align with existing research, demonstrating that the integration of DevOps and CD practices leads to optimal performance in dynamic environments.

4.2 System Performance

Figure 2 illustrates the impact of different software development frameworks on system performance, focusing on response time and throughput. The results highlight the limitations of traditional systems and the advantages of modern DevOps-based approaches in handling dynamic workloads.

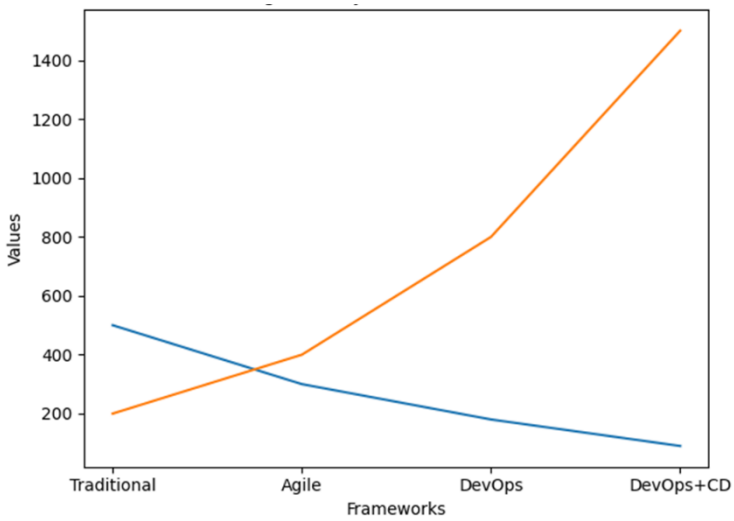

Figure 2. System Performance

The traditional framework exhibits the highest response time and lowest throughput, reflecting inefficiencies in processing and scalability. These findings are consistent with [13], who emphasized that traditional systems struggle to adapt to changing workloads due to centralized architectures and lack of automation. Agile frameworks improve performance by enabling incremental development and faster feedback, resulting in moderate improvements in response time and throughput. However, Agile does not fully address operational efficiency, as noted by [14].

DevOps frameworks significantly enhance system performance by integrating development and operations processes. The reduction in response time and increase in throughput can be attributed to continuous integration, automated testing, and efficient deployment pipelines. [1] highlighted that DevOps practices improve system efficiency and scalability, which aligns with the improvements observed in the figure.

The DevOps + Continuous Delivery model achieves the best performance, with the lowest response time and highest throughput. This is due to the automation of deployment processes and the ability to scale systems dynamically. [2] emphasized that continuous delivery enables rapid and reliable software releases, improving overall system performance. The results confirm that integrating DevOps with continuous delivery provides a robust solution for optimizing system performance in dynamic environments.

4.3 Resource Utilization & Availability

Figure 3 examines the relationship between resource utilization and system availability across different software development frameworks. The results indicate that modern DevOps-based approaches significantly outperform traditional methods in both efficiency and reliability.

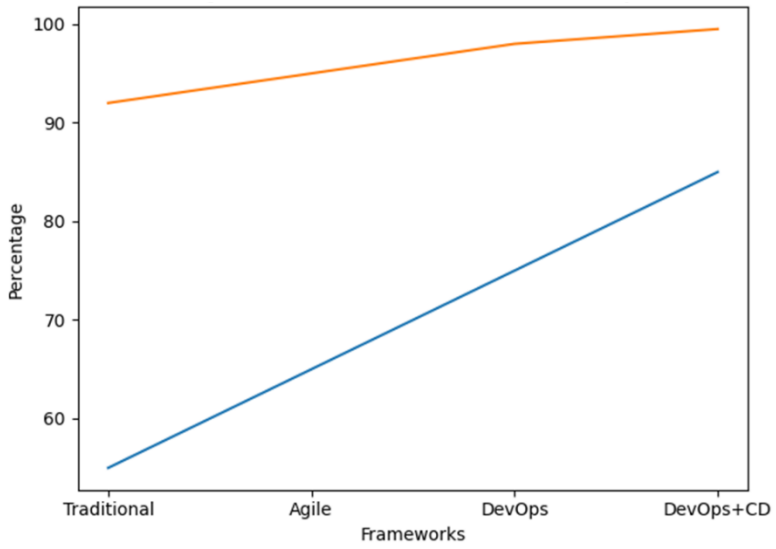

Figure 3. Resource Utilization & Availability

Traditional systems show low resource utilization and limited availability due to inefficient infrastructure management and lack of automation. This aligns with findings by [15], who noted that traditional development practices often result in underutilized resources and increased downtime. Agile frameworks improve resource utilization by enhancing planning and collaboration, leading to moderate improvements in system availability.

DevOps frameworks demonstrate a substantial increase in resource utilization and availability. This improvement is driven by the use of automation, infrastructure as code (IaC), and continuous monitoring. [3] found that DevOps adoption leads to higher system reliability and reduced downtime, which is consistent with the trends observed in the figure. Continuous monitoring allows organizations to detect and resolve issues quickly, thereby improving system availability.

The highest performance is observed in the DevOps + Continuous Delivery model, where resource utilization reaches optimal levels and availability approaches near-continuous uptime. This is due to the integration of automated deployment, real-time monitoring, and scalable infrastructure. [2] emphasized that continuous delivery enables efficient resource management and reliable system performance.

Overall, the figure confirms that the integration of DevOps practices with continuous delivery significantly enhances both resource utilization and system availability, making it an effective approach for modern software systems.

5. LIMITATIONS

Despite the widespread adoption of DevOps practices and their demonstrated benefits in enabling continuous delivery and system optimization, several limitations and challenges persist. These limitations arise from technical complexities, organizational barriers, tool dependencies, and evolving system requirements in dynamic environments. Addressing these challenges is essential to fully leverage the potential of DevOps in modern software engineering.

One of the primary limitations of DevOps integration is the complexity of toolchains and infrastructure management. DevOps relies heavily on a wide range of tools for continuous integration, continuous delivery (CI/CD), configuration management, monitoring, and automation. Integrating these tools into a cohesive pipeline can be challenging, particularly for large-scale systems with diverse technology stacks [1]. Inconsistent tool configurations and

compatibility issues can lead to inefficiencies and increase the risk of system failures.

Another significant limitation is the cultural and organizational resistance to change. DevOps requires a shift from traditional siloed structures to a collaborative and cross-functional approach. However, many organizations struggle to adopt this cultural transformation due to resistance from employees, lack of management support, and insufficient training [15], [16]. Without a strong DevOps culture, the implementation of technical practices alone may not yield the desired outcomes.

Security challenges also present a major limitation in DevOps environments. The rapid pace of continuous delivery can lead to security vulnerabilities if proper measures are not integrated into the development pipeline. Traditional security practices are often not aligned with DevOps workflows, creating gaps in security coverage [17]. Although DevSecOps has emerged as a solution, integrating security into CI/CD pipelines remains a complex task that requires specialized tools and expertise.

The dependency on automation is another limitation that can impact system reliability. While automation improves efficiency and reduces human error, it also introduces risks associated with tool failures and misconfigurations. A single error in the automation pipeline can disrupt the entire development process, leading to delays and system downtime [2]. Maintaining and updating automation scripts requires continuous effort and monitoring.

Scalability and performance issues in distributed systems also pose challenges for DevOps integration. Modern applications often operate in cloud-based and microservices environments, where managing scalability and performance becomes increasingly complex. Inefficient resource allocation and lack of proper monitoring can lead to performance bottlenecks and increased operational costs [13].

Another limitation is the difficulty in measuring performance and productivity accurately. Although frameworks such as DORA metrics provide valuable insights, they may not capture all aspects of system performance and developer productivity. Metrics such as deployment frequency and lead time focus primarily on delivery speed, while factors such as code quality, maintainability, and user satisfaction are often overlooked [3]. This limitation highlights the need for more comprehensive evaluation frameworks.

Additionally, skill gaps and training requirements remain a significant barrier to DevOps adoption. Implementing DevOps practices requires expertise in multiple domains, including development, operations, automation, and security. Organizations often face challenges in finding skilled professionals or providing adequate training to existing employees [1].

6. FUTURE DIRECTIONS

To overcome these limitations, several future research directions and technological advancements can be explored to enhance the integration of DevOps practices for continuous delivery and system optimization.

One promising direction is the integration of artificial intelligence and machine learning (AI/ML) into DevOps processes. AI-driven tools can analyze system data to predict failures, optimize resource allocation, and improve decision-making [18]–[21]. For example, machine learning algorithms can identify anomalies in system performance and automatically trigger corrective actions, reducing downtime and improving system reliability [22], [23].

Another important area for future development is the advancement of DevSecOps practices. Integrating security into the DevOps pipeline ensures that security is addressed throughout the development lifecycle rather than as an afterthought. Automated security testing, vulnerability scanning, and compliance checks can help mitigate security risks while maintaining the speed and efficiency of DevOps processes [17]. The adoption of serverless computing and cloud-native technologies is also expected to play a

significant role in the future of DevOps. Serverless architectures reduce the need for infrastructure management, allowing developers to focus on application logic. This approach enhances scalability and reduces operational complexity, making it suitable for dynamic environments [24].

Standardization of DevOps tools and practices is another critical future direction. Developing standardized frameworks and protocols can improve interoperability between tools and reduce integration challenges. Organizations such as the Continuous Delivery Foundation are already working towards establishing best practices and standards for DevOps and CI/CD processes. Even the integration of DevOps practices plays a crucial role in enabling continuous delivery, operational efficiency, and system optimization in modern technological environments [25]. DevOps methodologies improve collaboration, automation, and rapid deployment processes, enhancing the reliability and scalability of intelligent systems [26], [27].

The development of advanced monitoring and observability tools will also be essential for improving system optimization. Future tools should provide real-time insights into system performance, enabling proactive identification and resolution of issues. Technologies such as distributed tracing and log analytics can enhance system visibility and support better decision-making [28]–[30]. Another key direction is the focus on developer experience (DevEx) and human factors in DevOps. Improving developer productivity, collaboration, and satisfaction can significantly impact the success of DevOps initiatives. Frameworks such as SPACE emphasize the importance of human-centric metrics in evaluating software development performance [3]. Studies on photovoltaic manufacturing optimization and sustainable energy conversion demonstrate the importance of streamlined workflows and data-driven process management for improving performance and resilience [31]–[33]. These findings collectively highlight the significance of DevOps integration in supporting continuous improvement, efficient resource utilization, and sustainable technological advancement [27].

The concept of self-healing and autonomous systems is also gaining attention in DevOps research. These systems can automatically detect and resolve issues without human intervention, improving system reliability and reducing operational overhead. Techniques such as automated rollback, fault detection, and adaptive scaling can enhance system resilience [34].

Finally, sustainable and energy-efficient DevOps practices are emerging as an important research area. As software systems become more complex and resource-intensive, optimizing energy consumption and reducing environmental impact will become critical. Green computing techniques and energy-aware resource management can contribute to more sustainable software development.

7. CONCLUSION

This study examined the integration of DevOps practices for continuous delivery and system optimization, focusing on performance improvements across different software development frameworks. The analysis, supported by the presented figures, demonstrates a clear transition from traditional development approaches to modern DevOps-based systems, with significant gains in efficiency, reliability, and scalability. DevOps frameworks significantly enhance software delivery performance by integrating development and operations processes. The figures illustrate improvements in key metrics such as reduced lead time, lower change failure rates, and faster recovery times. These improvements are largely attributed to automation, continuous integration, and efficient monitoring practices. The most effective results are achieved with the integration of DevOps and continuous delivery. This approach enables frequent, reliable releases while maintaining system stability. The figures demonstrate optimal performance in terms of response time, throughput, resource utilization, and system availability.

Continuous delivery ensures that software can be deployed rapidly with minimal risk, enhancing both system reliability and user satisfaction. In conclusion, the integration of DevOps practices, particularly when combined with continuous delivery, provides a robust framework for achieving system optimization in dynamic environments. Organizations aiming to remain competitive should adopt these practices while addressing challenges such as tool complexity and security. Future advancements in automation and AI-driven DevOps are expected to further enhance system performance and operational efficiency.

FUNDING

This research received no external funding.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles, "A survey of DevOps concepts and challenges," *ACM Comput. Surv.*, vol. 52, no. 6, pp. 1–35, 2019, doi: 10.1145/3359981.

[2] J. Humble and D. Farley, *Continuous delivery: Reliable software releases through build, test, and deployment automation*. Addison-Wesley, 2011. [Online]. Available: https://www.oreilly.com/library/view/continuous-delivery-reliable/9780321670250/

[3] N. Forsgren, J. Humble, and G. Kim, *Accelerate: The science of lean software and devops: Building and scaling high performing technology organizations*. IT Revolution, 2018.

[4] M. Gokarna and R. Singh, "DevOps: A Historical Review and Future Works," *arXiv*, vol. 2012.06145, 2020, doi: 10.48550/arXiv.2012.06145.

[5] J. Cui, "The Role of DevOps in Enhancing Enterprise Software Delivery Success through R&D Efficiency and Source Code Management," *arXiv*, vol. 2411.02209, 2024, doi: 10.48550/arXiv.2411.02209.

[6] T. Offerman, R. Blinde, C. J. Stettina, and J. Visser, "A Study of Adoption and Effects of DevOps Practices," *arXiv*, vol. 2211.09390, 2022, doi: 10.48550/arXiv.2211.09390.

[7] M. I. Alam, M. A. K. P. Hemal, M. A. Sami, and M. L. Rahman, "Robust and Interpretable Crop Recommendation: A Case Study on a Balanced Multi-crop Agronomic Dataset," *Eur. J. Ecol. Biol. Agric.*, vol. 1, no. 5 SE-Articles, pp. 168–184, Nov. 2024, doi: 10.59324/ejeba.2024.1(5).14.

[8] S. Islam, S. I. Khan, A. A. M. Ashik, E. Hossain, M. M. Rahman, and M. S. Rahman, "Big Data in Economic Recovery: A Policy-Oriented Study on Data Analytics for Crisis Management and Growth Planning," *J. Comput. Anal. Appl.*, vol. 33, no. 7, pp. 2349–2367, 2024, [Online]. Available: https://www.eudoxuspress.com/index.php/pub/article/view/3338

[9] S. I. Khan, M. S. Rahman, A. A. M. Ashik, S. Islam, M. M. Rahman, and Hossain, "Big Data and Business Intelligence for Supply Chain Sustainability: Risk Mitigation and Green Optimization in the Digital Era," *Eur. J. Manag. Econ. Bus.*, vol. 1, no. 3, p. 23, 2024.

[10] W. W. Royce, "Managing the development of large software systems," in *Proceedings of IEEE WESCON*, 1970, pp. 1–9.

[11] K. Schwaber and J. Sutherland, "The Scrum Guide," Scrum Alliance, 2001. [Online]. Available: https://scrumguides.org/

[12] N. Vanu, M. R. Hasan, T. R. Sikder, and Z. S. Tamanna, "AI-Driven Big Data Analytics for Precision Medicine: A Unified Framework Integrating Molecular Data Intelligence, Wearable Health Systems, and Predictive Modeling," *J. Comput. Sci. Technol. Stud.*, vol. 3, no. 2, pp. 124–141, 2021.

[13] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *J. internet Serv. Appl.*, vol. 1, no. 1, pp. 7–18, 2010.

[14] D. J. Anderson, *Kanban: successful evolutionary change for your technology business*. Blue hole press, 2010.

[15] L. Bass, I. Weber, and L. Zhu, *DevOps: A software architect's perspective*. Addison-Wesley Professional, 2015.

[16] M. S. Rahman, S. Islam, S. I. Khan, A. A. M. Ashik, E. Hossain, and M. M. Rahman, "Redefining Marketing and Management Strategies in Digital Age: Adapting to Consumer Behavior and Technological Disruption," *J. Inf. Syst. Eng. Manag.*, vol. 9, no. 4, pp. 1–16, 2024, doi: 10.52783/jisem.v9i4.32.

[17] M. Ali, S. U. Khan, and A. V Vasilakos, "Security in cloud computing: Opportunities and challenges," *Inf. Sci. (Ny).*, vol. 305, pp. 357–383, 2015.

[18] Q. Zhang, M. Chen, L. Li, and W. Zhao, "Machine learning in cloud computing: A survey," *IEEE Trans. Cloud Comput.*, 2018.

[19] S. Nusrat, F. Hossain, and T. R. Sikder, "Integrating Wearable Health Data and Environmental Management Analytics for AI-Driven Cardiovascular Disease Prevention," *Eastasouth J. Inf. Syst. Comput. Sci.*, vol. 2, no. 02, pp. 209–223, 2024.

[20] M. A. Sami, M. A. K. P. Hemal, M. I. Alam, and M. L. Rahman, "Data Governance and Analytics Infrastructure for

Scalable Decision-Making in Development and Agritech Programs," *Eur. J. Appl. Sci. Eng. Technol.*, vol. 2, no. 2, pp. 388–403, 2024.

[21] S. Islam, E. Hossain, M. S. Rahman, M. M. Rahman, S. I. Khan, and A. A. M. Ashik, "Digital Transformation in SMEs: Unlocking Competitive Advantage through Business Intelligence and Data Analytics Adoption," vol. 5, no. 6, pp. 177–186, 2023, doi: https://doi.org/10.32996/jbms.2023.5.6.14.

[22] S. Saha, M. K. Islam, M. A. Rahaman, R. S. Mondal, and M. Kamruzzaman, "Machine Learning Driven Analytics for National Security Operations: A Wavelet-Stochastic Signal Detection Framework," *J. Comput. Anal. Appl.*, vol. 33, no. 8, p. 210, 2024, doi: 10.48047/jocaaa.2024.33.08.210.

[23] M. Kamruzzaman, R. S. Mondal, M. K. Islam, M. A. Rahaman, and S. Saha, "AI-driven predictive modelling of US economic growth using big data and explainable machine learning," *Int. J. Comput. Exp. Sci. Eng.*, vol. 10, no. 4, pp. 1927–1938, 2024, doi: 10.22399/ijcesen.3612.

[24] I. Baldini *et al.*, "Serverless computing: Current trends and open problems," in *Research advances in cloud computing*, Springer, 2017, pp. 1–20.

[25] S. C. Barman and M. R. Haque, "Artificial Intelligence Enabled Manufacturing Optimization Strategies for Enhancing Resilience and Scalability of Domestic Photovoltaic Supply Chains: A Systemic Review," *Bus. Soc. Sci.*, vol. 2, no. 1, pp. 1–7, 2024, doi: 10.25163/business.2110686.

[26] S. C. Barman and A. I. Opy, "Integrated artificial intelligence and stochastic optimization framework for resilient and low carbon renewable energy manufacturing systems," *Energy Environ. Econ.*, vol. 1, no. 1, pp. 1–8, 2023, doi: 10.25163/energy.1110684.

[27] S. C. Barman, S. Raval, and M. A. Hossian, "Socioeconomic and institutional determinants of public acceptance of waste-to-energy policies: Evidence for sustainable energy transitions," *Innov. Int. Multidiscip. J. Appl. Technol.*, vol. 1, no. 2, pp. 65–75, 2023, doi: 10.51699/rhs7k850.

[28] M. I. Alam, M. A. Sami, M. A. K. P. Hemal, and M. L. Rahman, "Predictive Analytics and Decision Intelligence for Climate-Resilient Agritech Systems," *Acad. Glob. J. Comput. Sci. Technol. Stud.*, vol. 2, no. 1, pp. 44–56, 2023, doi: 10.32996/agjcsts.2023.2.1.4.

[29] T. R. Sikder, S. Dash, B. Uddin, and F. Hossain, "AI-Powered Data Analytics and Multi-Omics Integration for Next-Generation Precision Oncology and Anticancer Drug Development," *Eastasouth J. Inf. Syst. Comput. Sci.*, vol. 1, no. 02 SE-Articles, pp. 153–170, Dec. 2023, doi: 10.58812/esiscs.v1i02.838.

[30] T. R. Sikder, M. A. Siam, M. M. H. Melon, S. M. M. Uddin, S. C. Mohonta, and F. Karim, "A Multimodal Data Analytics Framework for Early Cancer Detection Using Genomic, Radiomic, and Clinical Big Data Fusion," *J. Comput. Sci. Technol. Stud.*, vol. 5, no. 3, pp. 183–188, 2023.

[31] S. C. Barman, Z. Wang, G. Yasin, and M. F. Wen, "Experimental validation of earth abundant heterogeneous catalysts toward sustainable energy conversion," *Cent. Asian J. Theor. Appl. Sci.*, vol. 3, no. 3, pp. 93–102, 2022, doi: 10.51699/cajotas.v3i3.1662.

[32] A. A. M. Ashik, M. M. Rahman, E. Hossain, M. S. Rahman, S. Islam, and S. I. Khan, "Transforming U.S. Healthcare Profitability through Data-Driven Decision Making: Applications, Challenges, and Future Directions," *Eur. J. Med. Heal. Res.*, vol. 1, no. 3, p. 21, 2023, doi: https://doi.org/10.59324/ejmhr.2023.1(3).21.

[33] E. Hossain, A. A. M. Ashik, M. M. Rahman, S. I. Khan, M. S. Rahman, and S. Islam, "Big data and migration forecasting: Predictive insights into displacement patterns triggered by climate change and armed conflict," *J. Comput. Sci. Technol. Stud.*, vol. 5, no. 4, pp. 265–274, 2023, doi: https://doi.org/10.32996/jcsts.2023.5.4.27.

[34] E. Hossain, K. P. Shital, M. S. Rahman, S. Islam, S. I. Khan, and A. A. M. Ashik, "Machine Learning-Driven Governance: Predicting the Effectiveness of International Trade Policies through Policy and Governance Analytics," *J. Trends Financ. Econ.*, vol. 1, no. 3, pp. 50–62, 2024, doi: 10.61784/jtfe3053.