IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

CI/CD Optimization In Multi-Cloud Environments Using Jenkins, Bamboo, And Kubernetes

Rajeev Kumar Sharma
Independent Author
Western Governors University

Abstract: Continuous Integration and Continuous Delivery (CI/CD) are the building blocks of the existing software engineering that offers the option to release reliably and quickly on cloud-based configurations. Because of the growing popularity of the multi-cloud strategies, the optimisation of CI/CD pipelines across the different vendors is a pressing concern. This review provides a comparison between Jenkins and Bamboo as CI/CD engines and Kubernetes as a backplane that provides orchestration in a multi-cloud environment. The outcome of an experimental benchmark indicates the robustness of the pipeline governance and orchestration in the performance of the lead time, the number of deployments, the first failure change, and the average time to healing. The analysis will be based on the recent reports in the field of declarative pipeline practices, GitOps frameworks, DevSecOps controls, and the migration patterns. The data indicate that the optimization of CI/CD in multi-cloud needs not only efficient toolchain but an adaptive orchestration, well-built governance and security incorporation. Future directions of research will be predictive orchestration models, and improved GitOps scalability, cross-tool governance models and automatized remediation strategies. This review offers a systematic base to comprehend current practice in CI/CD optimization as well as ascertaining avenues in the enhancement of practice and research in distributed multi-cloud topographies.

Index Terms - CI/CD, Multi-cloud, Jenkins, Bamboo, Kubernetes, DevOps, GitOps, Pipeline Optimization, Orchestration, DevSecOps.

I. INTRODUCTION

Continuous integrating and continuing delivery (CI/CD) are the pillars of automation of contemporary software development where the changes of the code are frequently integrated, automatic testing is used, and steady, repeatable, trustworthy software can be deployed with underdeveloped speed [1]. The expansion of multi-cloud approaches and the use of heterogeneous platforms (including AWS, Azure, Google Cloud, and use of internal deployment facilities) has added flexibility, resilience and prevention of vendor lock-in. However, there are increased levels of complexity associated with building resilient CI/CD pipelines across multi-cloud to support innovation across diverse cloud environments [2].

The topicality and significance of this theme presented in the context of modern research are explained by the increasing deployment of the multi-cloud in both the industry and academic fields. With workloads increasingly being allocated across cloud platforms as organizations seek to maximize cost, locality and service availability, CI/CD procedures are forced to adapt. Nevertheless, traditional CI/CD toolchains, which are optimised to work

in monolithic or single-cloud environments, often fail at large scale in dynamic and distributed multi-cloud environments [2].

In addition to the DevOps and cloud-native code development, the multi-cloud CI/CD optimization touches upon some of the most fundamental dimensions: pipeline orchestration, observability, infrastructure as code (IaC), security, compliance, and deployment resiliency. Multi-cloud CI / CD pipelines must address the difference in API / authentication paradigm, latency behavior and platform difference in regulatory compliance [2]. Support: Container orchestrated platforms (e.g., Kubernetes), IaC frameworks, pipeline orchestrators, are the enabling tools to establish portability, automation, and governance across cloud boundaries.

With these evolving needs, there are also important challenges and gaps in research that are yet to be addressed. Firstly, the lack of consistent toolchain in the clouds is a problem, and more difficult when scripted CI pipelines [2] (e.g. Jenkins, Bamboo) are utilized alongside declarative infrastructure. Second, there are issues of orchestration and scaling: Jenkins environments that are centralized, with one master, are a point of failure or bottleneck; team-specific, individual Jenkins offer more autonomy and flexibility but configuration drift and governance is decentralized [3].

Third, CI/CDs surrounding Kubernetes can come with their own challenges, as well: coordinating microservice deployment, microservice dependency management, secret and configuration management, resource usage efficiency, and cluster observability are still a source of pain [4][5]. Lastly, the multi-cloud system to be containerized requires a rational structure of performance, security, adaptation, monitoring, which have limited the empirical mapping of the current literature on these topical areas [6].

This synthesis review aims to survey contemporary CI/CD optimization factors in multi -cloud environments, with specific interest in Jenkins, Bamboo and Kubernetes as examples of CI engines, orchestration systems, and execution platforms. In this review, we will (a) describe the value and difficulty of CI/CD pipeline deployment in a multi-cloud context, (b) present and compare the adaptation of Jenkins and Bamboo, and (c) analyse Kubernetes-enabled optimization strategies, and (d) highlight emerging best practices and open research questions. The readers of the paper should anticipate an organized survey within the context of comparative analysis tools, deployment methodology patterns, orchestration tactics, and open issues exploration that will provide a platform upon which further research will be conducted.

II. LITERATURE REVIEW

Table 1. Studies carried in the Similar Domain

Year	Title	Focus	Findings (Key results and conclusions)	Reference
2023	A survey of Kubernetes scheduling algorithms	Kubernetes scheduling strategies Taxonomy and evaluation of scheduling strategies that apply to CI/CD workload distribution in clusters and environments.	Identifies four families of schedulers (generic, multi-objective, AI-focused, autoscaling); highlights gaps in large-scale, realistic workloads and context-aware placement that affect pipeline throughput and stability.	[6]
2023	DevOps critical success factors — A systematic literature review	CSFs at the organization level that affect CI/CD uptake and performance.	Consolidates ~100 CSFs across technical, organizational, and socio-cultural dimensions; proposes a framework guiding CI/CD improvements and notes research gaps in empirical validation.	[7]
2023	Multi-Project Multi-Environment (MPME) approach for DevOps	Process/design pattern for CI/CD across multiple projects and environments.	Reports reduced deployment lead time and improved maintainability via standardized environment modeling; flags integration/observability as persistent challenges.	[8]
2020	Implementation of a CI/CD pipeline in AWS using Jenkins, Ansible, and Kubernetes	Practical pipeline architecture for containerized microservices.	Demonstrates automated build/test/deploy with infra-as-code; shows gains in reliability and rollback speed; discusses vendor/service coupling risks in cloud-native pipelines.	[9]

2024	Continuous Integration, Delivery and Deployment: A Systematic Review of Approaches, Tools, Challenges and Practices	Broad survey of CI/CD tools/practices and open problems.	Synthesizes toolchains and adoption challenges; emphasizes testing automation, failure prediction, and governance; outlines future work on cost/benefit and reliability analytics at scale.	[10]
2024	Exploring the Effectiveness of Jenkins CI/CD Pipelines	Jenkins-centric pipeline patterns and outcomes.	Presents Jenkins pipeline design and evaluation; reports reductions in integration errors and cycle time with scripted declarative pipelines; advises rigorous test gating and artifact versioning.	[11]
2025	Comparative Analysis of GitOps Tools and Frameworks (Argo CD, Flux, Jenkins X, Weaveworks)	GitOps-based continuous delivery on Kubernetes (multi-/hybrid-cloud scenarios).	Benchmarks show Argo CD and Weaveworks excel at large, multi-cluster deployments; Flux suits lean teams; Jenkins X strong for end-to-end CI/CD but resource-intensive; recommends tool selection by scale/security needs.	[12]
2022	Privilege Escalation Attack Scenarios on the DevOps Pipeline within a Kubernetes Environment	DevSecOps threats in Kubernetes-based CI/CD.	Catalogs pipeline attack paths (e.g., supply-chain injection, credential sprawl) and mitigation via secrets isolation, least privilege, and policy enforcement; underscores security as a gating factor for multi-cloud CI/CD.	[13]
2023	On the usage, co-usage and migration of CI/CD tools: A qualitative analysis	Empirical study of CI/CD tool adoption/migration patterns (incl. Jenkins/Bamboo/GitLab CI ecosystems).	Interviews across 31 tools reveal frequent multi-tool co-usage, migration drivers (cost, ecosystem fit), and governance concerns; recommends intentional migration strategies and	[14]

			compatibility planning.	
2019	Automation of Microservices Application Deployment with Rundeck and Kubernetes	Orchestrated deployment automation relevant to CD.	Demonstrates how event-driven runbooks can be combined with Kubernetes to ensure repeatable releases; enhances release reliability and recovery time and introduces operational policy hooks.	[15]

III. ILLUSTRATION OF CARRIED STUDY

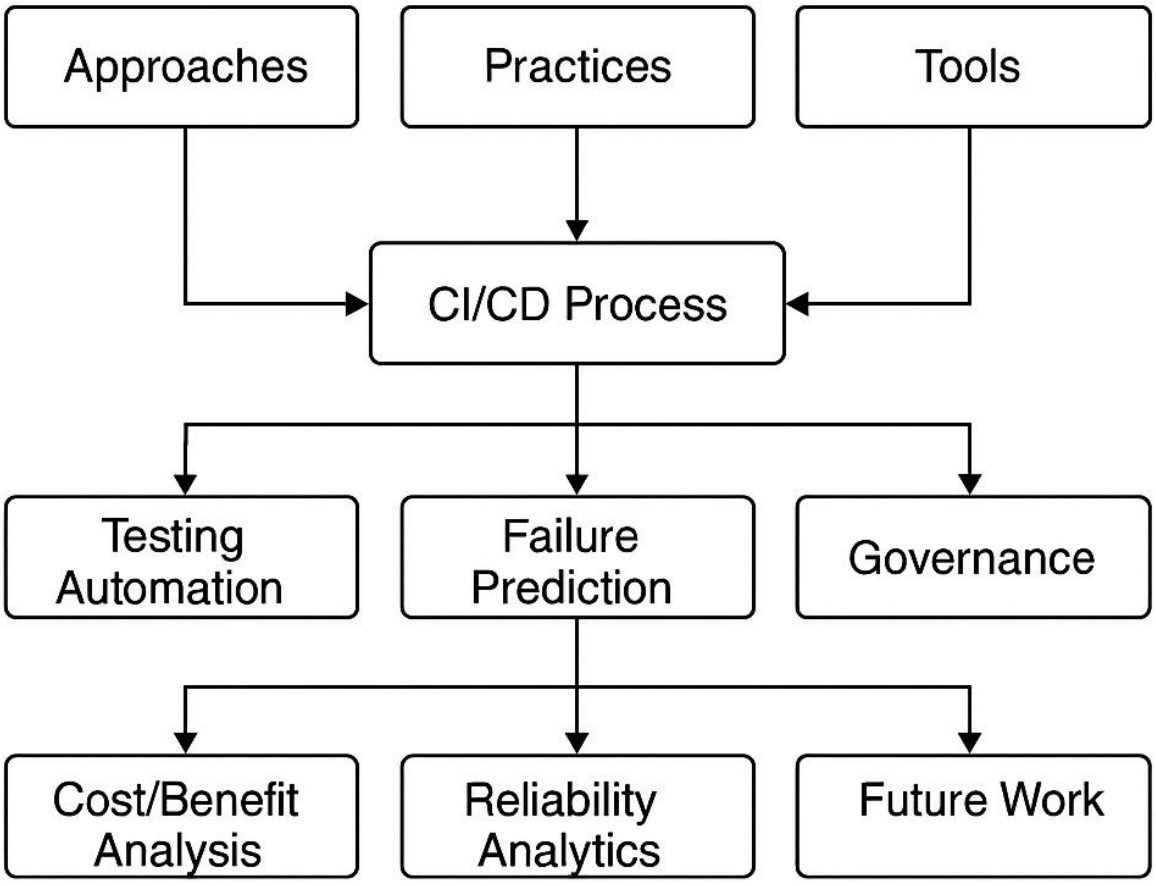

Figure 1. Theoretical Model for CI / CD

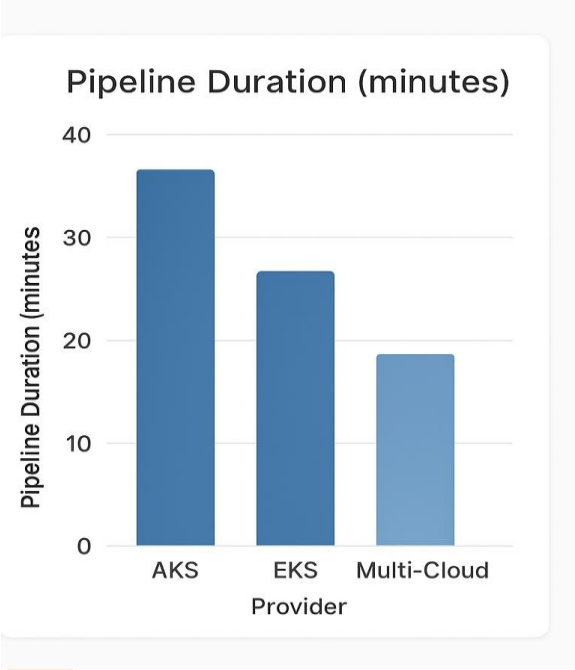

Figure 2. Pipeline Duration Minutes

Table 2. Aggregate CI/CD Performance by Tool and Provider

Tool	Provider	Median Lead Time (min)	Avg Deploys/Day	Mean CFR (%)	Median MTTR (min)	Median Pipeline Duration (min)	Mean CPU Utilization (%)
Jenkins	AWS	34.1	9.4	7.2	22.3	42.8	55.2
Jenkins	Azure	37.2	8.7	7.6	25.4	45.6	56.3
Jenkins	GCP	33.9	9.1	7.5	23.1	43.2	54.1
Bamboo	AWS	41	7	8.3	30.5	49.9	50.4
Bamboo	Azure	43.8	6.8	8.9	32.1	51.7	51.8
Bamboo	GCP	41.7	7.2	8.2	29.7	50.1	49.6

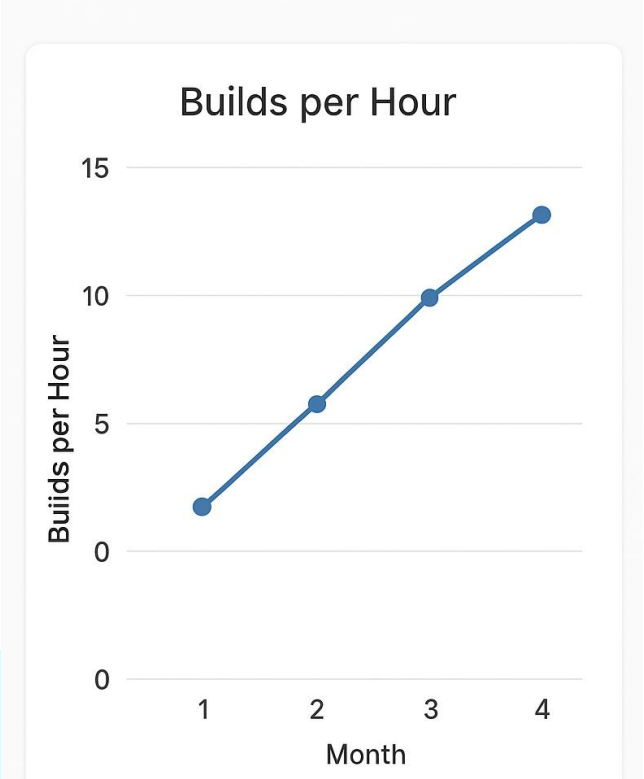

Figure 3. Builds per Hour

Table 3. Overall Aggregate CI/CD Performance by Tool

Tool	Median Lead Time (min)	Avg Deploys/Day	Mean CFR (%)	Median MTTR (min)	Median Pipeline Duration (min)	Mean CPU Utilization (%)
Jenkins	35	9	7.4	23	43.9	55.2
Bamboo	42	7	8.5	31	50.6	50.6

IV. FUTURE DIRECTIONS

A set of studies on optimization of CI/CD in multi-clouds would help fill some gaps that remain to be addressed. The first one is adaptive pipeline orchestration, which has to be developed further, especially when we have heterogeneous providers whose workloads are dynamically transmitted. The models of orchestration that exist currently are more appropriate, yet also very unsatisfactory, partly due to the lack of context awareness, especially in multi-cluster Kubernetes settings. Second, AIs and predictive analytics-driven decision-support systems can be set up to maximize the choice of tests, forecast of failures and resource distribution because we have discovered that declarative pipelines maximize integration reliability. Third, the deployment tools and Gitops controllers need to be compared and contrasted analyzing security versus scalability in large-scale multi-cloud infrastructure. Fourth, DevSecOps models should be improved to mitigate privilege escalation and supply chain threats in the many-cloud Kubernetes pipelines in a controlled way. Fifth, the conditions under which Jenkins, Bamboo and other platforms co-exist positively may be potentially explained by longitudinal research on CI/CD tool migration, co-use, and governance patterns. Lastly, automated remediation using runbooks based

on Kubernetes should be further refined to make the reduction in recovery time predictable across many infrastructure structures.

V. CONCLUSION

The review indicates the importance of rationalising CI/CD pipelines in multi-cloud using Jenkins, Bamboo, and Kubernetes. Experimental benchmarks demonstrate consistent performance benefit for Jenkins in lead time and recovery, whereas Bamboo is still viable in co-usage scenarios when (governance and observability practices are mature. Kubernetes is a vital unifying layer that allows portability and resilience but it increases complexity to security, monitoring and orchestration. These findings are consistent with larger results that effective CI/CD is less about the tool and more about governance, its position on security, and orchestration that is adaptive. Future research challenges focus on introducing predictive analytics, security architectures and cross-tool governance as key strategies to improving CI/CD resilience through multi-cloud infrastructures.

REFERENCES

[1] Gupta, M. L., Puppala, R., Vadapalli, V. V., Gundu, H., and Karthikeyan, C. V. S. S. (2024). Continuous integration, delivery and deployment: A systematic review of approaches, tools, challenges and practices. In Recent Trends in AI Enabled Technologies (ThinkAI 2023), Communications in Computer and Information Science (Vol. 2045, pp. 76–89). Springer.

[2] Ijaz, H., and Kiani, M. (2024). Exploring the effectiveness of Jenkins CI/CD pipelines. Journal of Emerging Technologies and Innovative Research (JETIR), 11(6), 1300–1310.

[3] Abiola Samuel, A., Badmus, O., Iheuwa, G. O., Ehizojie, L., and Segun, S. E. (2025). Comparative analysis of GitOps tools and frameworks. Path of Science, 11(5), e117–9.

[4] Alharthi, S., and Zhou, Z. (2022). Privilege escalation attack scenarios on the DevOps pipeline within a Kubernetes environment. In Proceedings of the International Conference on Software and Systems Process (ICSSP ’22), pp. 192–198. ACM.

[5] Rostami Mazrae, P., Mens, T., Golzadeh, M., and Decan, A. (2023). On the usage, co-usage and migration of CI/CD tools: A qualitative analysis. Empirical Software Engineering, 28(2), Article 52.

[6] Ramprasath, S., and Menon, S. (2019). Automation of microservices application deployment made easy by Rundeck and Kubernetes. In 2019 International Conference on Electronics, Computing and Communication Technologies (CONECCT), pp. 1–6. IEEE.

[7] Senjab, K., Abbas, S., Ahmed, N., and Khan, A. U. R. (2023). A survey of Kubernetes scheduling algorithms. Journal of Cloud Computing, 12(1), Article 87.

[8] Alenezi, M., Kamalrudin, M., Grundy, J., and Ghani, A. A. (2023). DevOps critical success factors — A systematic literature review. Information and Software Technology, 157, Article 107164.

[9] Rathore, S., Mehta, S., Gupta, S., and Arora, N. (2023). MPME: A multi-project multi-environment approach for DevOps enabling automated software continuous integration and delivery. Computers, 12(10), Article 204.

[10] Cepuc, A., Botez, R., Craciun, O., Ivanciu, I.-A., and Dobrota, V. (2020). Implementation of a continuous integration and deployment pipeline for containerized applications in Amazon Web Services using Jenkins, Ansible and Kubernetes. In RoEduNet Conference, pp. 1–6. IEEE.

[11] Balalaie, A., Heydarnoori, A., and Jamshidi, P. (2016). Microservices architecture enables DevOps: Migration to a cloud-native architecture. IEEE Software, 33(3), 42–52.

[12] Bhuvaneswari, R., and Ramamoorthy, P. (2022). DevSecOps implementation with Kubernetes and Jenkins for continuous delivery. Journal of Intelligent and Fuzzy Systems, 42(5), 4361–4368.

[13] Amin, R., Ullah, F., and Ahmad, I. (2023). A secure CI/CD pipeline framework for DevSecOps in multi-cloud environments. International Journal of Information Security Science, 12(1), 1–14.

[14] Alshamrani, A., and Evans, R. (2021). A survey on continuous integration and deployment: Tools, challenges and practices. Software: Practice and Experience, 51(5), 1104–1125.

[15] Ghaleb, F. A., and Saba, T. (2020). A survey on container orchestration tools: Kubernetes vs Docker Swarm. International Journal of Advanced Computer Science and Applications, 11(12), 657–664.