

BuildSphere - Unified CI/CD Operations Platform

**Dhiraj Chaudhari, Tanmay Zile, Taterao Dongardive, Ayush kohale, Pranav Kute, Sanchit Sarode,
Ankita .M .Bhoyar**

Department of Computer Science & Engineering

P. R. Pote Patil College of Engineering & Management, Amravati-444602, Maharashtra, India

Abstract— Modern software development relies on multiple DevOps tools for source-code management, continuous integration, containerization, deployment, and application hosting. However, using separate tools can make the software delivery workflow difficult to monitor and manage from a single place. BuildSphere is proposed as a unified CI/CD operations platform that brings important software-delivery activities into a centralized web-based environment. The project uses GitHub for source-code management, Jenkins for continuous integration and automated pipeline execution, Docker for containerization, Render for application deployment, and Vercel for frontend hosting. The platform is designed to provide a structured view of build and deployment activities and reduce unnecessary switching between development and deployment environments. It also follows a role-based architecture and provides a foundation for monitoring, analytics, notifications, and AI-assisted DevOps features. The proposed approach demonstrates how commonly used DevOps services can be connected into one operational workflow to improve visibility, automation, and software delivery efficiency.

Keywords— DevOps, CI/CD, GitHub, Jenkins, Docker, Render, Vercel, Continuous Integration, Continuous Deployment, Software Delivery, Build Monitoring.

I. INTRODUCTION

Modern software development depends on multiple tools to manage different stages of the Software Development Life Cycle (SDLC). Teams commonly use GitHub for source-code management and collaboration, Jenkins for Continuous Integration (CI) to automate builds and testing, Docker for creating consistent containerized environments, and cloud platforms such as Render and Vercel for application deployment and hosting. These tools help developers deliver software faster, improve code quality, and reduce manual work through automation.

Despite these advantages, each tool usually operates through its own dashboard and interface. Developers often need to switch between GitHub to review commits, Jenkins to check build results, Docker to inspect containers, and cloud platforms to verify deployments. This scattered workflow can make it difficult to monitor the complete CI/CD pipeline, quickly identify failures, and understand the overall status of a project, especially when multiple repositories and team members are involved.

BuildSphere is proposed as a centralized CI/CD operations platform that combines information from different DevOps tools into a single, easy-to-use dashboard. Instead of replacing existing tools, the platform is designed to integrate with them and provide a unified view of repository activity, build execution, deployment status, and operational metrics. This centralized approach helps developers, administrators, and project managers monitor software delivery more efficiently while reducing the need to navigate multiple platforms.

The system is developed as a web-based application using React for the frontend, Spring Boot for the backend, and PostgreSQL for data management, as described in the project synopsis. It includes features such as role-based access control, CI/CD monitoring, analytics, and notifications, while its scalable architecture allows future enhancements such as AI-assisted DevOps insights, predictive monitoring, and intelligent build-failure analysis. BuildSphere aims to improve visibility, coordination, and operational efficiency throughout the software delivery process.

II. LITERATURE REVIEW

A. Background / Basic Concepts

Continuous Integration (CI) is a software development practice in which developers regularly merge their code changes into a shared repository, allowing automated systems to build and test the application whenever new changes are introduced. This helps identify errors early, reduces integration conflicts, and improves software quality. Continuous Deployment (CD) extends this process by automatically deploying successfully tested applications to a hosting or production environment, reducing manual deployment effort and enabling faster software delivery. Together, CI/CD practices create a more efficient development workflow by shortening feedback cycles, improving collaboration, and ensuring that software updates can be released more reliably.

Git-based version control forms the foundation of collaborative software development by maintaining a complete history of code changes and supporting branch-based workflows for multiple developers. Tools such as Jenkins automate build and testing pipelines after code is committed, while Docker packages applications and their dependencies into portable containers that run consistently across different environments. Cloud platforms like Render simplify hosting for backend and containerized services, whereas Vercel provides fast and efficient deployment for modern frontend applications. Since these tools often operate through separate interfaces, a unified operations platform such as BuildSphere can integrate information from each stage and display repository activity, build status, container details, and deployment updates through a single centralized dashboard.

B. Existing Methods / Techniques

Existing DevOps practices generally treat source management, build automation, containerization, deployment, and monitoring as interconnected but separately managed activities. Continuous delivery research established automated build, test, and deployment practices, while DORA metrics introduced measures such as deployment frequency, lead time, mean time to recovery, and change failure rate. Modern cloud-native approaches additionally use containers and orchestration to improve portability and scalability. The BuildSphere approach builds on these practices by focusing on operational visibility across multiple tools. Instead of replacing GitHub, Jenkins, Docker, Render, or Vercel, the proposed platform treats them as components of one software-delivery workflow and provides a centralized interface for observing relevant activities.

C. Comparison of Existing Studies

Table 1. summarizes representative DevOps concepts and the gap addressed by BuildSphere.

Ref.	Author(s) & Year	Method / Technique	Application	Key Finding	Research Gap
[1]	Mowad et al. (2022)	CI/CD deployment practices	DevOps software delivery	CI/CD reduces the gap between development and operations	Does not define a unified project-specific dashboard
[2]	Mohammed et al. (2024)	AI-driven CI/CD	Software engineering pipelines	AI can support automation and decision-making in CI/CD	AI capabilities are not combined with a complete multi-tool dashboard
[3]	Arachchi & Perera (2018)	Pipeline automation	Agile software projects	Automated CI/CD supports repeatable delivery	Limited centralized operational visibility
[4]	Forsgren, Humble & Kim (2018)	DORA metrics	DevOps performance	Defines important software-delivery performance measures	Metrics require implementation and integration in an operational platform
[5]	Gupta et al. (2021)	Role-based access control	DevOps platforms	RBAC supports controlled access to shared systems	Not focused on end-to-end CI/CD monitoring
[6]	Turnbull (2014)	Monitoring practices	Infrastructure/software systems	Monitoring improves operational awareness	Does not provide the BuildSphere multi-tool integration model

III. RESEARCH GAP / LIMITATIONS

Existing DevOps tools such as GitHub, Jenkins, Docker, SonarQube, Render, and Vercel effectively automate individual stages of the software development lifecycle. However, these tools operate through separate interfaces, requiring developers to switch between multiple dashboards to track code changes, build status, code-quality reports, deployments, and notifications. This fragmented workflow makes it difficult to obtain a complete view of the CI/CD pipeline and can increase the time required to identify operational issues.

BuildSphere addresses this research gap by providing a centralized, role-based platform that integrates information from multiple DevOps tools into a single dashboard. The platform is designed to present build, deployment, code-quality, infrastructure, and notification updates through one interface, improving visibility and collaboration. Its scalable architecture also supports future AI-assisted features such as build-failure prediction, deployment-risk estimation, and build-log summarization, making it a foundation for smarter DevOps operations.

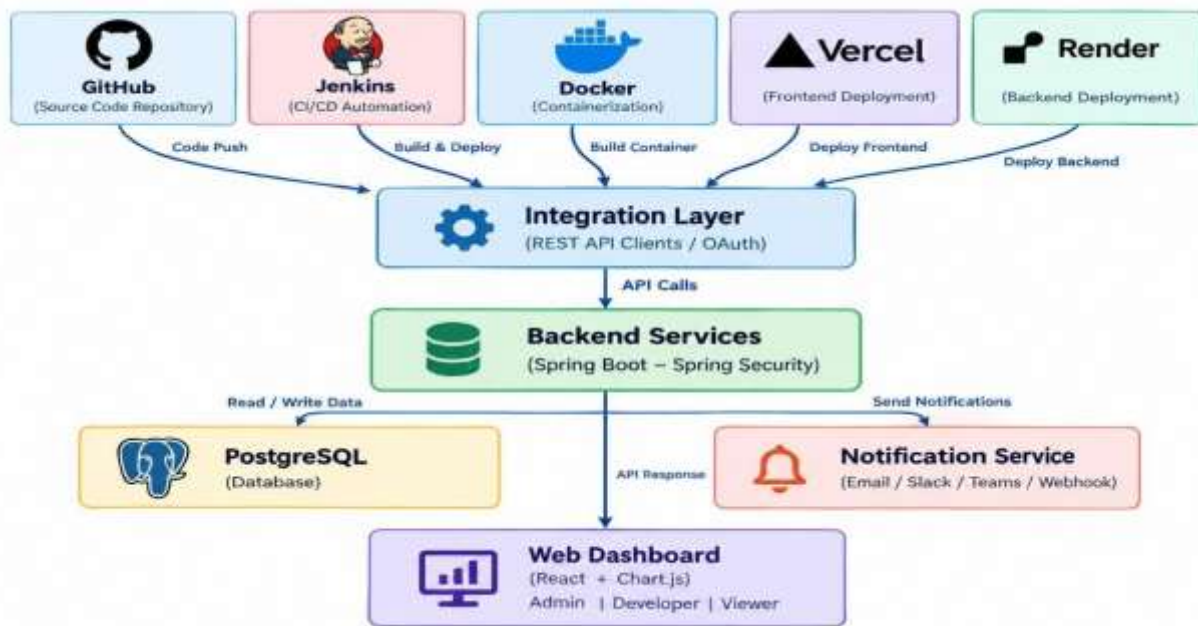
IV. PROPOSED METHODOLOGY

The proposed system follows an applied system-design methodology based on development and functional testing of an experimental prototype. The architecture is organized into an integration layer, backend services, database, notification service, and web dashboard. External DevOps services are connected through their available APIs or deployment mechanisms.

GitHub acts as the source-code repository and collaboration layer. Jenkins is responsible for continuous integration and pipeline execution. Docker is used to package the application and its dependencies into reproducible containers. Render is used as a cloud deployment environment for the application service, while Vercel is used for frontend deployment. The React frontend communicates with Spring Boot backend services, and persistent project information is stored in PostgreSQL as described in the project synopsis.

The proposed workflow begins when a developer pushes code to GitHub. The change can trigger a Jenkins pipeline, where the application is built and tested. Docker is then used to create the required container image or deployment package. The resulting service can be deployed through Render, while the React-based frontend can be delivered through Vercel. Build and deployment information can subsequently be represented in the BuildSphere dashboard. methodology is enough or to nedd more discsriptive or other.

Figure 1. High-Level System Architecture of BuildSphere.



• BuildSphere CI/CD Workflow

GitHub → Jenkins CI → Docker → Render (Backend/Service) → Vercel (Frontend) → BuildSphere Dashboard

A. System Components

Component	Role in BuildSphere
GitHub	Source-code repository, version control, branch management, and collaborative development.
Jenkins	Continuous integration server used for automated build, test, and pipeline execution.
Docker	Containerization layer used to package applications and dependencies consistently.
Render	Cloud deployment platform used to host application services.
Vercel	Frontend deployment and hosting platform for the React application.
React	Web frontend used to provide the BuildSphere user interface and dashboard.
Spring Boot	Backend framework used to implement REST services and application logic.
PostgreSQL	Persistent data store for build, deployment, and operational information.

B. Research and Evaluation Approach

- Overall research methodology : applied/system-design research based on an experimental prototype.
- Research design : development and functional testing of an integrated CI/CD monitoring dashboard.
- Integration approach : connect software-delivery stages using GitHub, Jenkins, Docker, Render, and Vercel.
- Performance indicators: build success/failure rate, average build duration, mean time to recovery (MTTR), lead time, and prediction accuracy where the corresponding modules are implemented.
- Validation approach: unit testing, integration testing of APIs, deployment testing, and role-based functional testing.
- Data collection approach: Collection of repository, build, deployment, and notification information from integrated DevOps services through available APIs and platform integrations.
- System architecture approach: Implementation of a layered architecture consisting of an integration layer, Spring Boot backend, PostgreSQL database, notification service, and React-based web dashboard.

V. IMPLEMENTATION AND DEPLOYMENT

The implementation of BuildSphere follows a practical DevOps workflow that connects source-code management, continuous integration, containerization, and cloud deployment into a unified process. Developers manage the project

through GitHub, where code changes are committed using a version-control workflow. These updates are detected by Jenkins, which executes the configured CI pipeline to build and validate the application through automated processes.

After a successful build, Docker packages the application into a consistent and reusable runtime environment, ensuring that it can run reliably across different deployment environments. The Spring Boot backend is deployed on Render, while the React frontend is deployed on Vercel. This separation allows frontend and backend services to be managed independently while remaining part of the same application architecture.

The BuildSphere dashboard acts as the central monitoring interface by bringing together information from different stages of the DevOps pipeline. The platform is designed with role-based access for Admin, Developer, and Viewer users, allowing each role to access relevant project information. The dashboard presents repository activity, pipeline status, deployment updates, and operational analytics in one place, reducing the need to switch between multiple DevOps tools and improving overall visibility throughout the software delivery process.

VI. CHALLENGES / LIMITATIONS

- Third-party integration: reliable communication with external APIs and services can be affected by authentication requirements, rate limits, and service availability.
- Deployment dependency: the project depends on external cloud and CI/CD services, so changes in their configurations or policies may affect the workflow.
- Security: API credentials, access tokens, repository information, and deployment secrets must be protected using secure authentication and authorization mechanisms.
- Resource requirements: continuous monitoring, data collection, container builds, and AI-assisted log processing may require additional computational resources.
- Project time constraints: integrating and testing several independent DevOps services within an academic project schedule is challenging.

VII. EXPECTED OUTCOMES

The proposed BuildSphere platform is expected to deliver a functional centralized DevOps dashboard that improves visibility across the software development and deployment lifecycle. By integrating tools such as GitHub, Jenkins, Docker, Render, and Vercel into a single workflow, the platform aims to provide developers and administrators with a unified view of repository activity, build execution, deployment status, and system updates. This centralized approach is intended to reduce the need for frequent switching between multiple dashboards, making CI/CD pipeline information easier to access and understand.

The platform is also expected to improve operational efficiency by enabling faster identification of build failures, deployment issues, and workflow interruptions. Instead of manually checking each DevOps tool separately, users can monitor key activities from a common interface, helping reduce repetitive tasks and improve response time during software delivery. Features such as role-based access control, notifications, and activity monitoring are expected to support better coordination between developers, administrators, and project teams while encouraging a more organized development process.

Beyond its immediate functionality, BuildSphere is designed as a scalable and reusable architecture that can support future enhancements. The modular design of the React frontend, Spring Boot backend, and PostgreSQL database allows additional DevOps integrations and monitoring features to be added without major architectural changes. In future versions, the platform can be extended with AI-assisted DevOps capabilities, including intelligent build-failure analysis, predictive monitoring, deployment recommendations, and automated operational insights, making BuildSphere a stronger decision-support platform for modern software development teams.

VIII. INNOVATION / PRACTICAL RELEVANCE

The main innovation of BuildSphere is its ability to consolidate multiple stages of the software delivery lifecycle into a single operational view. Instead of replacing existing DevOps tools, the platform integrates information from services such as GitHub, Jenkins, Docker, Render, and Vercel and presents it through a centralized dashboard. This allows users to monitor repository activity, build status, deployment progress, and operational updates from one interface, reducing the need to switch between multiple platforms.

Another important aspect of BuildSphere is its role-based design, which provides different levels of access for administrators, developers, and viewers. The platform is built on a scalable architecture using React, Spring Boot, and PostgreSQL, making it suitable for future enhancements. This approach makes BuildSphere practically relevant for development teams that rely on multiple DevOps services while creating a foundation for future AI-assisted monitoring and analytics features.

IX. FUTURE SCOPE

BuildSphere can be further enhanced by integrating Kubernetes for container orchestration and expanding support for additional cloud platforms beyond Render and Vercel. Future versions can include broader DevOps integrations,

allowing the platform to monitor more tools and provide a centralized view of increasingly complex software delivery environments.

The platform can also be extended with AI-assisted DevOps features, such as build-failure prediction, deployment-risk estimation, intelligent build-log summarization, and automated alert prioritization. Additional improvements may include richer DORA metrics and analytics, support for more notification channels, deeper historical trend analysis, and enhanced security and audit capabilities, making BuildSphere a more comprehensive platform for modern software development teams.

X. CONCLUSION

BuildSphere proposes a unified CI/CD operations platform for reducing the fragmentation that occurs when software delivery activities are distributed across multiple DevOps tools. The project combines GitHub for source control, Jenkins for continuous integration, Docker for containerization, Render for service deployment, and Vercel for frontend hosting within a structured application workflow.

The proposed architecture provides a foundation for centralized monitoring, role-based access, analytics, notifications, and AI-assisted DevOps features. By bringing these activities into one platform, the project aims to improve operational visibility and support a more manageable software-delivery process. The implementation and evaluation of the prototype can provide a practical basis for further development of integrated DevOps monitoring systems.

REFERENCES

- [1] A. M. Mowad, H. Fawareh, and M. A. Hassan, "Effect of using continuous integration (CI) and continuous delivery (CD) deployment in DevOps to reduce the gap between developer and operation," in Proc. 2022 International Arab Conference on Information Technology (ACIT), 2022, pp. 1–8.
- [2] A. S. Mohammed, V. R. Saddi, S. K. Gopal, S. Dhanasekaran, and M. S. Naruka, "AI-Driven Continuous Integration and Continuous Deployment in Software Engineering," in Proc. 2024 2nd International Conference on Disruptive Technologies (ICDT), 2024, pp. 531–536.
- [3] S. A. Arachchi and I. Perera, "Continuous integration and continuous delivery pipeline automation for agile software project management," in Proc. 2018 Moratuwa Engineering Research Conference (MERCon), 2018, pp. 156–161.
- [4] A. Verma and S. Rao, "Kubernetes-based container monitoring for cloud-native applications," in Proc. International Conference on Cloud Computing, 2020, pp. 112–119.
- [5] R. K. Gupta, M. Venkatachalapathy, and F. K. Jeberla, "Challenges in adopting continuous delivery and DevOps in a globally distributed product team: A case study of a healthcare organization," in Proc. 2019 ACM/IEEE International Conference on Global Software Engineering (ICGSE), 2019, pp. 30–34.
- [6] G. A. Doukoure and E. Mnkandla, "Facilitating the management of agile and DevOps activities: Implementation of a data consolidator," in Proc. 2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD), 2018.
- [7] R. Salama, F. Al-Turjman, P. Chaudhary, and S. P. Yadav, "Benefits of Internet of Things (IoT) Applications in Health care—An Overview," in Proc. 2023 International Conference on Computational Intelligence, Communication Technology and Networking (CICTN), 2023, pp. 778–784.
- [8] K. Patel and N. Joshi, "Alert management strategies in DevOps environments," *International Journal of Computer Applications*, vol. 45, no. 6, pp. 23–29, 2019.
- [9] E. van Eyk, A. Iosup, S. Seif, and M. Thömmes, "A SPEC RG cloud group's vision on the performance challenges of FaaS cloud architectures," arXiv:2010.12667, 2020.
- [10] T. Chen, Y. Liu, and H. Wang, "Machine learning approaches for build failure prediction," *IEEE Transactions on Software Engineering*, vol. 49, no. 4, pp. 210–221, 2023.
- [11] M. Renieres and S. P. Reiss, "Fault localization with nearest neighbor queries," in Proc. 18th IEEE International Conference on Automated Software Engineering, 2003, pp. 30–39.
- [12] B. Turhan, T. Menzies, A. B. Bener, and D. Stefano, "On the relative value of software testing in lifecycle development," in Proc. ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, 2009, pp. 391–400.
- [13] A. E. Hassan, "Predicting faults using the complexity of code changes," *IEEE Transactions on Software Engineering*, vol. 35, no. 2, pp. 181–197, 2009.
- [14] S. Gupta and P. Nair, "Role-based access control models for multi-tenant DevOps platforms," *International Journal of Information Security*, vol. 15, no. 1, pp. 33–41, 2021.
- [15] D. F. Ferraiolo, J. Cugini, and D. R. Kuhn, "Role-based access control (RBAC): Features and motivations," in Proc. 11th Annual Computer Security Applications Conference, 1995, pp. 241–248.