iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.82621

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

Simplifying Application Deployment Using a One Click Deployment Framework

Tejas Govind Gaikwad[1], Swami Narayan Kshatriya[2], Renuka Rajendra Joshi[3], Vijay Chinataman Deore[4], Bhavana A. Khivsara[5]
Department of Computer Engineering, SNJB's K.B. Jain College of Engineering, Nashik, 423101, India

Abstract: ***Application deployment is a vital part of modern software development, but it often involves complicated setup tasks, managing dependencies, and manual processes that need a lot of DevOps skills. Developers usually run into issues like mismatched environments, configuration mistakes, and slow deployment steps when moving apps from code repositories. While platforms such as Heroku, Netlify, and Vercel make deployment easier, they are usually tied to specific platforms and only work with certain types of apps. On the other hand, Infrastructure as Code (IaC) tools and CI/CD pipelines offer automation, but they require advanced technical skills to set up and keep running. To solve these problems, this paper introduces ClickToDeploy, a simple and platform-agnostic one-click deployment system that automates the process of deploying apps directly from a Git repository. The system uses Docker for containerization along with automated build, packaging, and deployment steps to make sure environments are the same across development and production setups. The architecture includes parts for connecting with repositories, automating builds and packaging, managing deployments, handling workflows, and monitoring to show real- time status and logs. By cutting down on manual work and mak- ing deployment processes simpler, the system allows developers, students, and small teams to quickly deploy apps without needing deep DevOps knowledge. Testing shows that this method makes deployment faster, cuts down on configuration errors, and gives a dependable and consistent setup for delivering applications, thus connecting academic development practices with real-world DevOps workflows.***
Index Terms: One-Click Deployment, Docker, CI/CD, DevOps, Containerization, Git, Infrastructure as Code, Automation.

I. INTRODUCTION

In today's software engineering world, quickly and reliably deploying software is a key need for companies that want to deliver applications efficiently. Traditional ways of deploying software involve many manual steps like setting up envi- ronments, installing dependencies, preparing infrastructure, and configuring the application. These steps take a lot of time and can lead to mistakes, which might cause problems when moving from development to production environments. As software systems become more complex, the need for automated and dependable deployment methods is growing.

To tackle these issues, DevOps practices have become com- mon in the software industry. DevOps brings together software development and IT operations to improve teamwork, auto- mate tasks, and speed up the delivery of software. Continuous

Integration (CI) and Continuous Deployment (CD) pipelines are key to automating the process of building, testing, and deploying apps, which helps developers work more efficiently and ensures that systems are reliable [1]. However, setting up CI/CD pipelines often requires expertise in managing and configuring infrastructure, which can be challenging for beginners, students, or small teams.

Infrastructure as Code (IaC) has become an important method for managing and setting up computing resources through configuration files that machines can read instead of manual processes. Tools like Terraform, Ansible, and Chef help automate infrastructure setup and maintain consistency across different environments [2]. While these tools improve automation and scalability, they often require specific knowledge and complex scripts, which can be hard for developers not familiar with DevOps concepts.

Containerization technologies like Docker have made deploying applications easier by packaging the app and its dependencies into lightweight, portable containers. Containers ensure that apps run the same way across different environ- ments, reducing issues like dependency conflicts and environment mismatches [3]. Despite these improvements, many deployment platforms like Heroku, Netlify, and Vercel are limited to certain frameworks or types of applications, making them less suitable for more general deployment needs.

To address these limitations, this research introduces *Click- ToDeploy*, a platform-independent one-click deployment sys- tem that simplifies deploying applications directly from a Git repository. The system automatically retrieves source code, builds Docker images, deploys containerized apps, and monitors deployment status through an automated pipeline. By reducing the need for manual setup and DevOps skills, the proposed system aims to make deployment faster, easier, and more accessible for developers, startups, and educational settings. The goal of this work is to connect academic software development practices with real-world DevOps workflows while offering a simple and effective automation solution.

A. Motivation

A common problem faced by students occurs when a project runs successfully on their local machine but fails when deployed on a remote server. For example, a Node.js application may run locally because the correct dependencies are installed globally. When deployed to a server, however, missing dependencies or incorrect environment configurations cause the application to fail.

This situation leads to the well-known “works on my machine” problem. Students may spend hours debugging environment issues rather than focusing on improving their application.

ClickToDeploy aims to eliminate these issues by automat- ically packaging applications and their dependencies inside Docker containers. By providing a one-click deployment interface, the system allows students to focus on software development rather than infrastructure management.

II. LITERATURE SURVEY

Several methods have been suggested to make software deployment and infrastructure management easier in today’s software development settings. Continuous delivery and de- ployment allow companies to release software more often and reliably by automating the building, testing, and deployment steps [1]. These practices cut down on manual work and make the software delivery process more consistent. However, to use continuous delivery effectively, it is important to connect development and operations workflows smoothly and include automated infrastructure management.

The use of Continuous Integration (CI) and Continuous Deployment (CD) pipelines has greatly boosted the efficiency of software development. These pipelines automatically inte- grate, test, and deploy apps, making the release process faster and helping to improve the quality of the final product [2]. Even though these pipelines offer many benefits, setting them up often needs knowledge of multiple tools and managing infrastructure resources.

DevOps practices help improve software development by fostering better teamwork between developers and operations staff. DevOps uses automation, monitoring, and infrastructure management methods to speed up the software release cycle while keeping the system running smoothly [3]. Studies show that adopting DevOps can lead to higher productivity and better scalability in modern software systems [4].

Another important method used in modern deployment systems is Infrastructure as Code (IaC). IaC lets developers au- tomatically set up and manage infrastructure resources through configuration scripts instead of manually configuring them. Tools such as Terraform and Ansible let developers control infrastructure using code, making the setup process repeatable and reducing errors. However, these tools need knowledge of scripting languages and cloud infrastructure management, which can be tough for new users and small teams [5].

Platform-as-a-Service (PaaS) platforms like Heroku and Render make app deployment simpler by hiding server setup and infrastructure management. These platforms allow de- velopers to quickly deploy applications without needing to worry about server maintenance or environment setup. While PaaS makes deployment easier, they often limit customization options and have specific rules that can make them unsuitable for more general deployments [6].

Containerization technologies, especially Docker, are now widely used to package apps and their dependencies into containers. Docker ensures that apps run the same way across different environments like development, testing, and produc- tion by including all necessary dependencies in the containers. This method helps avoid issues caused by different environ- ments and makes deployments more reliable. However, using Docker still needs some manual setup through Dockerfiles and command-line operations, which can be tricky for people who are not experienced with these tools [7].

Looking at these methods, there is a need for a simpler deployment system that brings together containerization and automation while making existing DevOps tools less compli- cated. The proposed ClickToDeploy system tackles this need by offering a one-click deployment feature. This system au- tomatically fetches code from Git repositories, builds Docker images, and deploys apps in a containerized environment. This approach simplifies the deployment process and makes modern DevOps practices easier to use for developers, students, and small teams.

A. *Literature Survey Comparison*

Table I summarizes the key approaches, advantages, and limitations of existing deployment methods compared to the proposed *ClickToDeploy* system.

TABLE I
COMPARISON OF EXISTING DEPLOYMENT APPROACHES

Approach	Key Idea	Advantages	Limitations
Continuous Delivery	Automated release pipeline	Faster and reliable releases	Complex pipeline setup
CI/CD Systems	Automated build, test, deploy	Improves productivity and code quality	Tool integration complexity
DevOps Practices	Dev & Ops collaboration	Faster development cycle	Requires expertise & infrastructure knowledge
IaC	Automated infrastructure provisioning	Reproducible infrastructure	Requires scripting knowledge
PaaS Platforms	Managed deployment environment	Easy deployment	Platform limitations
Containerization (Docker)	Portable application contain- ers	Environment consistency	Manual Docker configuration
ClickToDeploy	One-click automated deployment	Simple deployment workflow	Future scope: multi-cloud scaling

B. *Problems Identified from Literature Survey*

Based on the above analysis, the following limitations were identified in existing systems:

1) High Complexity of DevOps Tools: Many deployment automation systems require advanced knowledge of De- vOps tools and scripting languages.

2) Platform Dependency in Deployment Platforms: PaaS services often restrict deployment to specific environ- ments and frameworks.

3) Manual Configuration in Containerization: Container-based deployment requires manual Dockerfile creation and command execution.

4) Complex CI/CD Pipeline Setup: Configuring CI/CD pipelines requires integration of multiple tools and in- frastructure resources.

5) Limited Accessibility for Beginners: Students and beginner developers often face difficulties understand- ing and implementing DevOps-based deployment work- flows.

The proposed *ClickToDeploy* system aims to address these limitations by providing a simplified and automated deploy- ment framework that requires minimal configuration and tech- nical expertise.

III. PROPOSED SYSTEM — CLICKTODEPLOY

The proposed system, *ClickToDeploy*, is meant to make the application deployment process easier by offering a fully automated one-click deployment option. It uses container- ization and automation to deploy apps directly from a Git repository with very little user involvement. Unlike traditional deployment methods that need manual setup of infrastructure, installation of dependencies, and environment configuration, this framework handles all those tasks automatically through an integrated pipeline.

The main goal of the system is to simplify the DevOps- based deployment process while still keeping the reliability and consistency that come with containerized environments. All a user needs to do is provide a Git repository URL that contains the application's source code. The system then automatically gets the source code, builds the application environment using Docker, and deploys the app into a con- tainerized runtime environment.

The architecture of the system is made up of several modules that work together to automate the deployment work- flow. The first module is the Repository Integration Module, which takes the Git repository URL from the user and clones the repository into the deployment environment. The second module is the Build and Packaging Module. It looks at the project structure and uses Docker to create a container image that includes the app and all its dependencies. This ensures that the environment is consistent across different platforms.

The third module is the Deployment Module, which runs the generated Docker image as a container on the target server or cloud environment. This ensures the application becomes available to users once deployment is complete. The fourth module is the Automation Engine, which manages the entire workflow through an automated pipeline that triggers build and deployment tasks whenever a new repository is provided or updated.

In addition, the system includes a *Monitoring and Logging Module* that tracks deployment status, collects system logs, and gives real-time updates to the user. This helps developers spot any errors during deployment and ensures the app runs correctly after it is deployed.

By combining these modules, the *ClickToDeploy* system offers a platform-independent deployment framework that can work on any system that supports Docker. It reduces the need for manual configuration, avoids problems caused by mis- matched environments, and allows developers, students, and small teams to deploy applications quickly without needing deep DevOps expertise.

A. Key Features of Proposed System

1) One-Click Deployment: Deploy applications directly from a Git repository with a single command.
2) Automated CI/CD Workflow: Automatically builds and deploys applications without manual configuration.
3) Docker-Based Containerization: Ensures consistent en- vironments across development and production systems.
4) Platform Independence: Works on Linux, Windows, or cloud servers that support Docker.
5) Real-Time Monitoring: Provides logs and deployment status updates.

B. Advantages of Proposed System

1) Reduces deployment complexity.
2) Eliminates manual configuration steps.
3) Ensures environment consistency using containers.
4) Suitable for beginners and educational environments.
5) Faster deployment compared to traditional methods.

C. System Architecture of ClickToDeploy

The *ClickToDeploy* architecture is designed to automate the entire application deployment pipeline from a Git repository to a running containerized application. The system follows a six-layer modular architecture where each layer performs a specific role in the deployment workflow. The architecture integrates Git-based source control, Docker containerization, CI/CD automation, multi-cloud deployment, and real-time monitoring services, all coordinated through a central automa- tion engine.

D. System Modules

1) User Interface Module: The User Interface module acts as the entry point of the system. It provides a simple interface through which users can interact with the deployment platform. The interface can be implemented as a web-based dashboard or command-line interface (CLI).

Functions:

- Accept Git repository URL
- Trigger deployment process
- Display deployment status
- Show logs and errors

Input: Git repository link from the user.

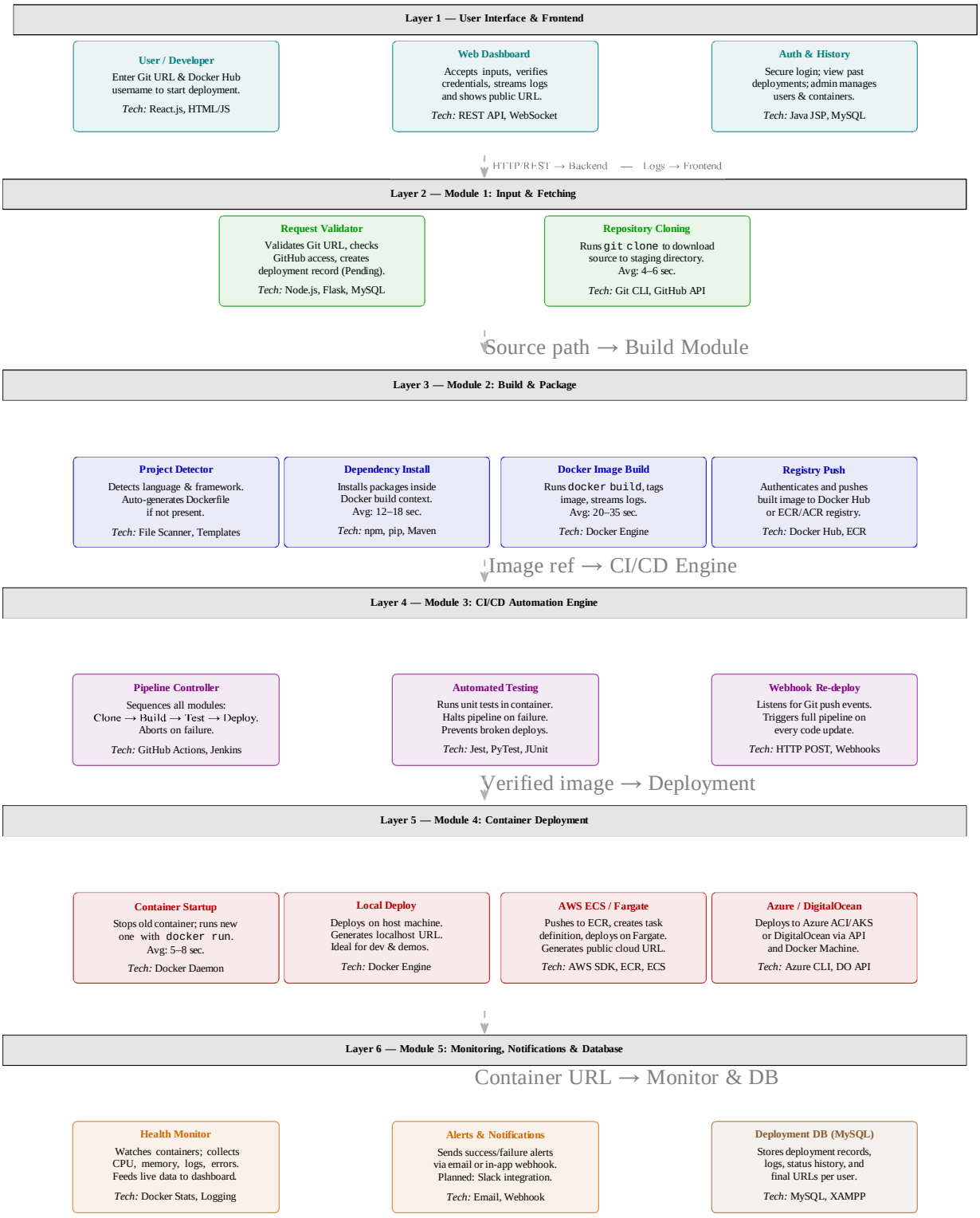


Fig. 1. Six-Layer System Architecture of ClickToDeploy

2) *Git Repository Input Module:* This module validates and processes the Git repository URL provided by the user. The system ensures that the repository exists and can be accessed.

Functions:
- Validate repository link
- Authenticate repository access if required
- Send repository details to the fetching module

Output: Valid repository URL for cloning.

3) Source Fetch Module (Git Clone): This module retrieves the application source code from the Git repository using Git commands.
Functions:
- Clone repository
- Download source code
- Prepare project files for build process

Technologies: Git, GitHub / GitLab APIs.
Output: Local copy of the application source code.

4) Build and Package Module: This module is responsible for transforming the source code into a deployable container image.
Functions:
- Read Dockerfile
- Install dependencies
- Build Docker image
- Package application into container

Technologies: Docker Engine, Dockerfile configuration.
Out- put: Docker image containing the application.

5) Deployment Engine: The deployment engine runs the generated Docker image as a container on the target environ- ment.
Functions:
- Create Docker container
- Configure networking and ports
- Start application services

Deployment Platforms: Local server, Cloud server, Container runtime environments.
Output: Running application container.

6) Monitoring and Logging Module: This module contin- uously tracks the deployed application and provides real-time feedback to the user.
Functions:
- Monitor container health
- Track CPU and memory usage
- Display deployment logs
- Report errors

Benefits: Helps developers detect deployment failures and improves system reliability.

C. Deployment Algorithm — ClickToDeploy
Input: Git Repository URL
Output: Live Application URL
1: Receive repository URL from user
2: Validate repository accessibility
3: Clone repository into temporary workspace
4: Detect project type (Node, Python, Java)
5: if Dockerfile does not exist then 6: Generate Dockerfile template 7: end if
8: Build Docker image using docker build
9: Run container using docker run

10: Monitor container status
11: Provide deployment logs to user interface
12: return Live Application URL

IV. RESULTS AND PERFORMANCE EVALUATION

A. Experimental Setup

The performance of the proposed *ClickToDeploy* system was evaluated in a controlled development environment to analyze its deployment efficiency, resource consumption, and scalability. The experiments were conducted on a desktop system equipped with an Intel Core i5 processor, 8 GB RAM, and Windows 11 operating system. The user interface of the system was accessed using Google Chrome (version 120 or above). The backend services of the system were deployed in a containerized environment to simulate real-world deployment scenarios. Network performance tests were conducted using a broadband connection with an average speed of 50 Mbps, and additional observations were taken over a 4G LTE network (~20 Mbps) to evaluate behavior under different network conditions. For testing purposes, several Git repositories containing different types of applications were used. These included Node.js web applications, static web projects, and container- ized backend services. Each test deployment followed the full automated pipeline, including repository retrieval, dependency setup, Docker image generation, container execution, and service exposure.

B. Deployment Time Analysis

The time required for each stage of the automated deploy- ment pipeline was measured to understand the efficiency of the proposed system. The results indicate that the majority of the deployment time occurs during the Docker image creation stage, while other operations such as repository cloning and container startup are comparatively faster. Fig. 2 shows the average deployment time per stage.

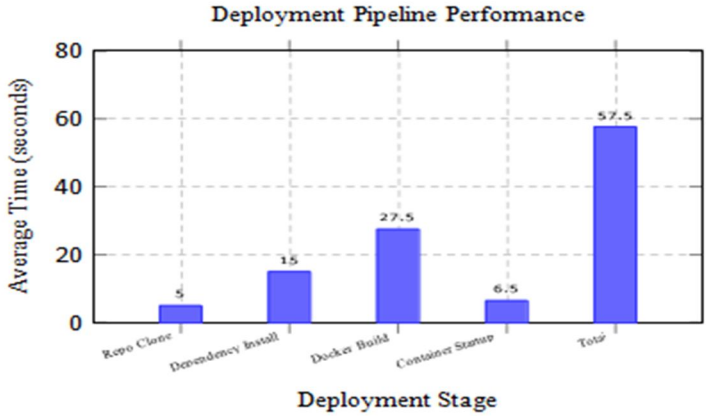


Fig. 2. Deployment Pipeline Stage-Wise Average Time (seconds)

Table II provides the full range of deployment times per stage.

TABLE II
DEPLOYMENT PIPELINE PERFORMANCE

Deployment Stage	Average Time (seconds)
Repository Cloning	4 – 6
Dependency Installation	12 – 18
Docker Image Build	20 – 35
Container Startup	5 – 8
Total Deployment Time	45 – 70

The findings show that the system can deploy small to medium-sized applications in approximately one minute, which is significantly faster compared to traditional manual deployment processes that often require several minutes of configuration and setup.

C. Resource Utilization

System resource usage was also monitored during the deployment process to evaluate efficiency and stability. The evaluation focused on CPU usage, memory consumption, and network usage during the automated deployment workflow. Fig. 3 illustrates the resource utilization metrics.

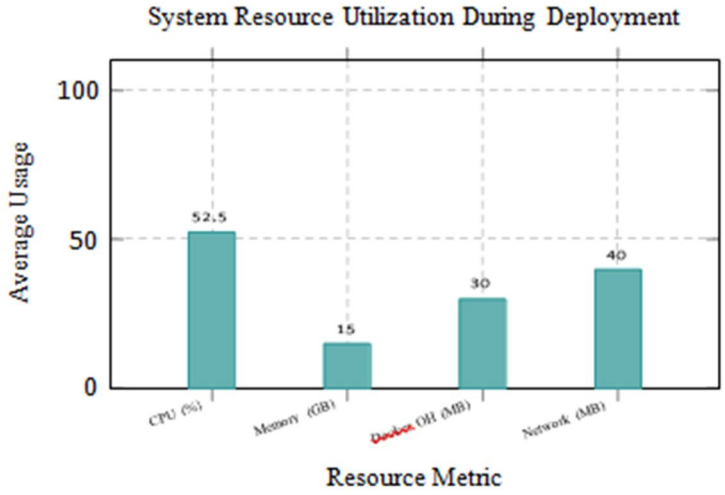


Fig. 3. System Resource Utilization During Deployment (scaled for visibility)

TABLE III
SYSTEM RESOURCE UTILIZATION DURING DEPLOYMENT

Resource Metric	Average Usage
CPU Utilization	45% – 60%
Memory Usage	1.2 – 1.8 GB
Docker Engine Overhead	~300 MB
Network Data Transfer	~40 MB per deployment

The observations indicate that the system operates within moderate hardware limits and does not require high-end moderate hardware limits and does not require high-end infrastructure. This makes the solution practical for both local development environments and cloud-based deployment platforms.

D. Scalability Evaluation

To analyze the scalability of the system, multiple deployment operations were executed simultaneously. The goal was to evaluate how the system behaves when handling several deployment requests at the same time. Fig. 4 illustrates how deployment time and success rate vary with concurrent deployments.

TABLE IV
SCALABILITY METRICS BY NUMBER OF DEPLOYMENTS

Concurrent Deployments	Avg. Deployment Time	Success Rate
1	50 sec	100%
3	60 sec	100%
5	75 sec	98%
10	95 sec	96%

The results show that the system maintains stable function- ality even when several deployment processes are executed concurrently. Although the deployment time slightly increases as the number of requests grows, the overall success rate remains high, demonstrating that the system can effectively manage multiple deployments.

E. Comparison with Traditional Deployment Methods

A comparison was performed between the proposed *Click- ToDeploy* system and conventional manual deployment practices to highlight the improvements provided by automation. Fig. 5 illustrates the key performance differences.

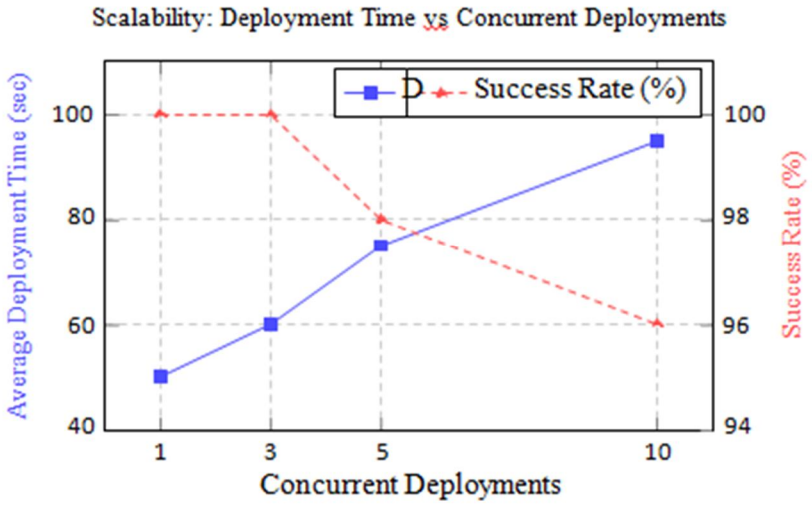


Fig. 4. Scalability: Deployment Time and Success Rate vs Concurrent Deployments

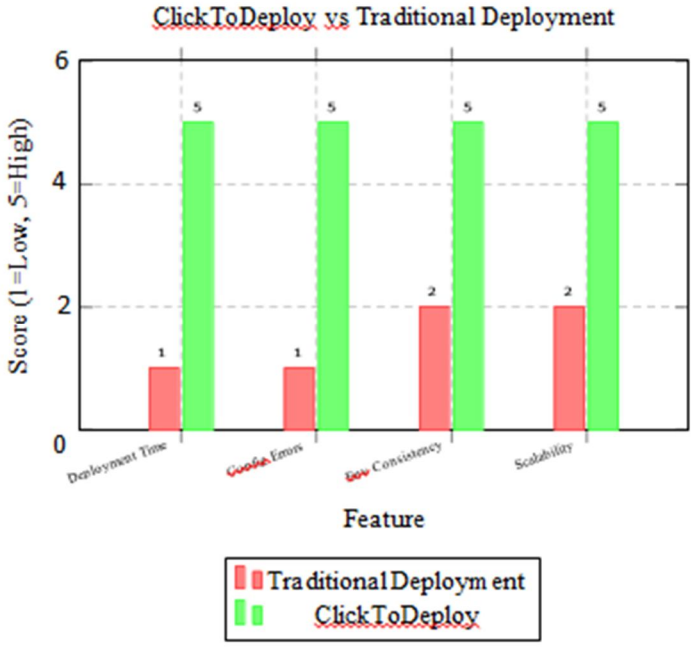


Fig. 5. Comparison: ClickToDeploy vs Traditional Deployment Methods

The comparison clearly demonstrates that the proposed system simplifies the deployment workflow by eliminating many manual steps and reducing the chances of configuration errors.

V. DISCUSSION

The evaluation results confirm that the *ClickToDeploy* plat- form significantly improves the application deployment pro- cess by integrating automation and containerization technologies. The system minimizes manual intervention and ensures consistent runtime environments through Dockerbased containers. Additionally, the modular design of the architecture allows the system to scale efficiently when multiple deployment requests are processed simultaneously. The automated

TABLE V
COMPARISON WITH TRADITIONAL DEPLOYMENT METHODS

Feature	Traditional Deploy-ment	ClickToDeploy
Configuration Process	Manual	Automated
Average Deploy-ment Time	10 – 20 minutes	Less than 1 minute
Risk of Configu-ration Errors	High	Low
Environment Consistency	Often inconsistent	Container-based con-sistency
Scalability	Limited	High

workflow, which integrates repository retrieval, build oper- ations, and container deployment, helps developers deploy applications quickly and reliably.

However, some limitations remain. Deployment speed may vary depending on the size of the repository, the number of dependencies, and the available network bandwidth. Fu- ture improvements will focus on implementing build caching mechanisms, integrating advanced CI/CD pipelines, and en- abling multi-cloud deployment support to further enhance the efficiency and flexibility of the system.

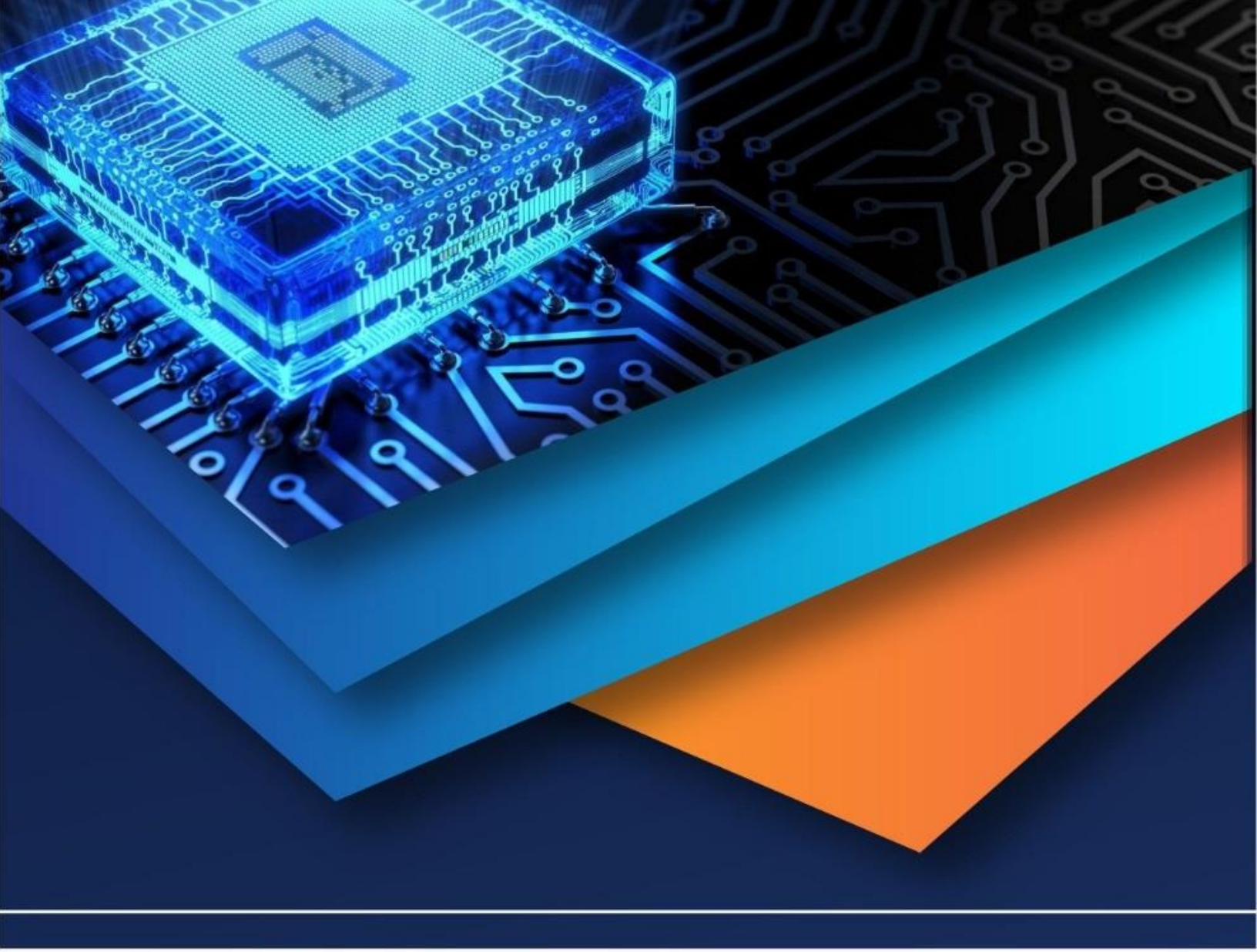
VI. CONCLUSION

This paper presented *ClickToDeploy*, a system designed to simplify and automate the software deployment process. The proposed platform integrates containerization and automated workflows to enable developers to deploy applications directly from a repository with minimal configuration. By reducing manual steps and environment-related issues, the system helps developers achieve faster and more consistent application deployment.

The evaluation of the system shows that *ClickToDeploy* improves deployment efficiency and reduces configuration complexity compared to traditional deployment methods. The modular architecture also allows the system to scale and adapt to different development environments. Future improvements may include integration with advanced CI/CD pipelines, multi- cloud deployment support, and enhanced monitoring and se- curity features.

REFERENCES

[1] J. Humble and D. Farley, Continuous Delivery: Reliable Soft- ware Releases Through Build, Test, and Deployment Automation. Boston: Addison-Wesley Professional, 2010. [Online]. Available: https://dl.acm.org/doi/10.5555/1869904
[2] M. Shahin, M. A. Babar, and L. Zhu, "Continuous integra- tion, delivery and deployment: A systematic review on ap- proaches, tools, challenges and practices," IEEE Transactions on Software Engineering, vol. 43, pp. 1–21, 2017. [Online]. Available: https://www.researchgate.net/publication/315381994
[3] L. E. Lwakatare, P. Kuvaja, and M. Oivo, "Relationship of DevOps to Agile, Lean and Continuous Deployment," in Proc. International Conference on Software Process Improvement (EuroSPI), 2016, pp. 399– 415.
[4] G. Kim, J. Humble, P. Debois, and J. Willis, The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technol- ogy Organizations. Portland: IT Revolution Press, 2016.
[5] M. Rahman, I. Mahmud, and A. K. Hassan, "A systematic analysis of Infrastructure as Code technologies and tools," Gazi University Journal of Science Part A: Engineering and Innovation, vol. 10, no. 4, pp. 1–16, 2023.
[6] D. Merkel, "Docker: Lightweight Linux containers for consistent de- velopment and deployment," Linux Journal, vol. 2014, no. 239, p. 2, 2014.
[7] C. Pahl, "Containerization and the PaaS cloud," IEEE Cloud Computing, vol. 2, no. 3, pp. 24–31, 2015.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY