International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 03 | Mar 2026 www.irjet.net p-ISSN: 2395-0072

Enhancing Frontend Deployment Using AWS CloudFront- A Vercel-Inspired Platform

Manvvi Cilagani[1], K. Sai Kusumitha[2], K. Srikar Reddy[3], K. Naveen[4], L. Nagender Kumar[5]

[1]*Frontend and Backend Developer*
[2]*Frontend Developer*
[3]*Backend Developer*
[4]*Frontend Developer*
[5]*Team Mentor, Dept. of Computer Science & Engineering, Joginpally B R Engineering College, Telangana, India.*

Abstract - *The increasing reliance on web applications has created a strong demand for efficient and scalable frontend deployment systems. Traditional hosting approaches often require manual configuration, resulting in delays and potential errors. Modern platforms simplify deployment but introduce limitations in terms of cost, flexibility, and infrastructure control.*

This paper presents a system titled "Enhancing Frontend Deployment using AWS CloudFront – A Vercel-Inspired Platform", which focuses on automating frontend deployment using cloud-based services. The proposed solution integrates GitHub repositories, performs automated build processes, and distributes static content through AWS CloudFront for optimized global delivery.

The system is designed to minimize deployment time, improve performance using CDN-based caching, and provide a cost-effective alternative for students and developers. By leveraging AWS S3 for storage and CloudFront for content delivery, the platform ensures high availability, scalability, and low latency.

The results demonstrate that the proposed system significantly improves deployment efficiency while maintaining simplicity and affordability.

Key Words: **Frontend Deployment, AWS CloudFront, CDN, CI/CD, GitHub Integration, Web Hosting, Automation**

1. INTRODUCTION

The rapid growth of modern web technologies has increased the need for efficient deployment systems that can deliver applications quickly and reliably. Developers require platforms that not only automate the deployment process but also ensure high performance and scalability.

Traditional deployment methods involve manually uploading files, configuring servers, and managing hosting environments. These processes are time-consuming and prone to human error. Modern solutions like automated deployment platforms have simplified this workflow, but they often come with limitations such as restricted customization and higher costs.

This project proposes a deployment platform inspired by modern tools, focusing on automation, performance, and affordability. The system integrates GitHub repositories, builds frontend applications automatically, and deploys them using cloud infrastructure. By utilizing AWS CloudFront, the system ensures faster content delivery through globally distributed edge locations.

2. LITERATURE REVIEW

Recent advancements in cloud computing have significantly improved frontend deployment practices. Many platforms now offer automated deployment pipelines integrated with version control systems. These systems reduce manual effort and improve development efficiency.

However, existing solutions often impose limitations such as pricing constraints, lack of infrastructure control, and dependency on proprietary systems. While CDN-based delivery improves performance, it is not always accessible to all users due to cost barriers.

This study identifies the need for a system that combines automation, scalability, and affordability. By analyzing existing platforms and their drawbacks, a more flexible and cost-efficient solution can be developed using cloud services.

3. SYSTEM ANALYSIS

3.1 Functional Requirements

The system is designed to automate and simplify the frontend deployment process by integrating multiple functionalities into a unified workflow. It enables users to connect their GitHub repositories seamlessly, allowing the platform to fetch source code directly. Once the repository is linked, the system automatically installs dependencies and builds the frontend application without manual intervention. The generated build files are then deployed to cloud storage, ensuring secure and scalable hosting. Additionally, the platform generates a live deployment URL, which allows users to instantly access and share their application. Throughout the process, the system provides real-time deployment status updates, improving transparency and enhancing the overall user experience.

3.2 Non-Functional Requirements

The system is designed to meet essential non-functional requirements that ensure quality and reliability. It delivers high performance with minimal latency by leveraging efficient processing and content delivery mechanisms. The architecture supports scalability, allowing multiple users and deployments to be handled simultaneously without performance degradation. In addition, the system ensures reliable uptime and consistent availability, making deployed applications accessible at all times. Security is also a key consideration, with appropriate measures implemented to protect user data and ensure safe handling of information throughout the deployment process.

4. PROPOSED SYSTEM

The proposed system introduces an automated frontend deployment platform designed to enhance efficiency while minimizing operational costs. It integrates GitHub to fetch project source code directly, enabling seamless interaction with modern development workflows. The system automates the build process by executing required commands without manual intervention, thereby reducing errors and saving time. Generated files are stored in AWS S3, ensuring secure and scalable storage, while AWS CloudFront is utilized to deliver content globally with low latency through CDN technology. Additionally, the platform provides real-time status tracking, allowing users to monitor deployment progress at each stage. This integrated approach results in faster deployment, reduced manual effort, and significantly improved overall performance.

5. SYSTEM ARCHITECHTURE

The system follows a distributed architecture composed of four key components that work together to enable efficient and scalable frontend deployment.

5.1 Frontend Layer

The frontend layer serves as the user interaction interface. It allows users to submit GitHub repository URLs, initiate deployments, and track the status of their applications. This layer is designed to be simple and user-friendly, ensuring ease of use for both beginners and developers.

5.2 Backend Server

The backend server acts as the core processing unit of the system. It handles important operations such as cloning the repository from GitHub, installing dependencies, executing the build process, and managing deployment workflows. It also maintains deployment status and communicates with other system components.

5.3 Storage Layer (AWS S3)

The storage layer is responsible for storing the static build files generated after the deployment process. AWS S3 is used due to its high reliability, scalability, and secure storage capabilities, ensuring that application files are safely maintained and easily accessible.

5.4 CDN Layer (AWS CloudFront)

The CDN layer distributes the stored content globally using AWS CloudFront. It caches data at multiple edge locations and serves users from the nearest server, which reduces latency and significantly improves application load time and performance.

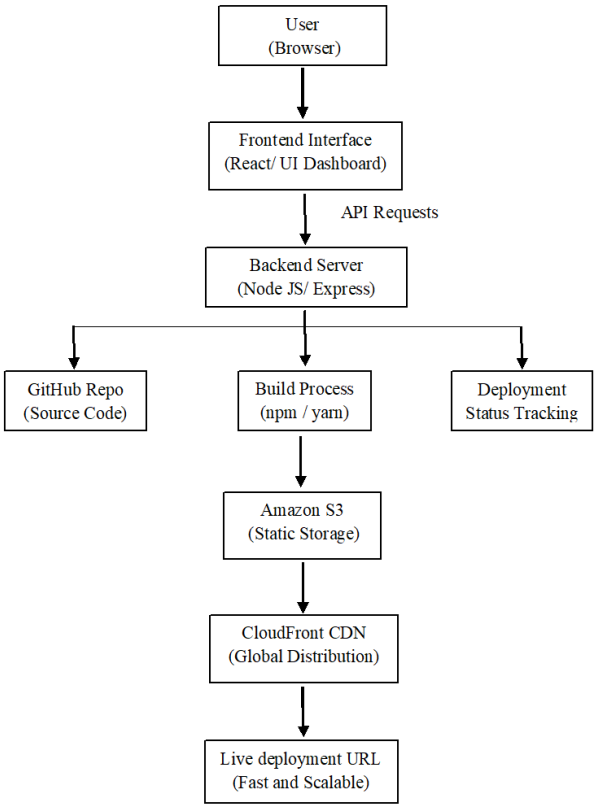

Fig1: System Architecture

6. METHODOLOGY

The system follows a well-defined and fully automated workflow designed to streamline the frontend deployment process and minimize manual intervention. The process begins when the user submits a GitHub repository URL through the platform's interface. The backend server then retrieves the source code directly from the repository, ensuring seamless integration with version control systems.

Once the code is fetched, the system automatically installs all required dependencies, preparing the environment for execution.

After setting up the environment, the build process is initiated, where the application is compiled into optimized, production-ready static files. These generated files are then securely uploaded to AWS S3, which provides reliable and scalable storage. Following this, AWS CloudFront takes over the delivery process by distributing the content across multiple global edge locations. This ensures that users can access the application from the nearest server, significantly reducing latency and improving load times.

Finally, the system generates a unique deployment URL, which is shared with the user for instant access to the live application. Throughout this process, the system operates automatically and efficiently, ensuring consistency, reducing errors, and enabling a smooth and reliable deployment pipeline.

7. METHODOLOGY

The system is implemented using a modular approach, where each module is responsible for a specific functionality in the deployment pipeline. This design improves maintainability, scalability, and clarity of the overall system.

7.1 Authentication Module

The Authentication Module is responsible for managing user access and ensuring system security. It allows users to register and log in using their credentials, thereby providing controlled access to the platform. The module uses token-based authentication mechanisms, such as JSON Web Tokens (JWT), to maintain secure user sessions. Once authenticated, users can access their respective projects and deployment details without repeated login requests. This module also ensures that sensitive data is protected and unauthorized access is prevented through secure validation techniques.

7.2 Repository Integration Module

The Repository Integration Module enables seamless interaction with GitHub repositories. It allows users to provide a repository URL, which is then validated by the system to ensure correctness and accessibility. After validation, the backend clones the repository into the server environment using Git commands. This module ensures that the latest version of the source code is fetched accurately and prepared for further processing. It plays a crucial role in connecting version control systems with the deployment pipeline, thereby enabling automated workflows.

7.3 Deployment Engine

The Deployment Engine is the core component of the system that handles the build and deployment process. Once the repository is cloned, this module installs all required dependencies and executes the necessary build commands to generate production-ready files. The generated files are then uploaded to AWS S3 for storage. After successful upload, the module triggers AWS CloudFront to distribute the content globally through its CDN network. Additionally, the deployment engine manages error handling, logs build outputs, and updates deployment status, ensuring a reliable and efficient deployment process.

8. RESULTS

The implemented system successfully achieves automated frontend deployment while providing a fast and reliable hosting solution. The integration of cloud services and automation significantly enhances the overall efficiency of the deployment process. The platform was tested with multiple frontend applications, and the results indicate consistent performance and reliability across different scenarios.

One of the major improvements observed is the reduction in deployment time, as the automated workflow eliminates manual steps such as configuration and file uploads. Additionally, the use of CDN technology through AWS CloudFront enables faster content delivery by serving users from the nearest edge locations, thereby reducing latency and improving load times.

The system also demonstrates strong scalability, as it can handle multiple deployments simultaneously without significant performance degradation. This makes it suitable for use by multiple users or projects at the same time. Furthermore, the overall solution proves to be cost-effective, as it leverages cloud services efficiently, minimizing infrastructure and maintenance expenses.

During testing, the platform maintained stable performance even under multiple deployment requests, confirming its reliability and robustness as a frontend deployment solution.

9. CONCLUSION AND FUTURE ENHANCEMENTS

9.1 Conclusion

The proposed system presents an efficient and scalable solution for modern frontend deployment challenges. By integrating automation with cloud-based services, the platform significantly simplifies the deployment process, eliminating the need for manual configuration and reducing the chances of human error. The use of automated workflows ensures consistency and speeds up the entire deployment cycle, making it highly suitable for developers and students.

Furthermore, the system effectively leverages cloud infrastructure to optimize performance while maintaining cost efficiency. The integration of AWS S3 for storage and AWS CloudFront for content delivery ensures high availability, reliability, and scalability. In particular, the use

of AWS CloudFront plays a crucial role in enhancing application performance by distributing content across global edge locations, resulting in reduced latency and faster load times. Overall, the system delivers a practical and affordable alternative to existing deployment platforms while maintaining strong performance and usability.

9.2 Future Enhancements

Although the current system provides a strong foundation, several enhancements can further improve its functionality and user experience. One important improvement is the integration of custom domain support, allowing users to deploy applications with personalized domain names instead of default URLs. Additionally, automating SSL certificate generation and management would enhance security by enabling HTTPS for all deployed applications without manual configuration.

Another valuable enhancement is the inclusion of real-time deployment logs, which would allow users to monitor the build and deployment process in detail and quickly identify any errors. Finally, introducing multi-user collaboration features would enable teams to work together on projects, share deployments, and manage applications more efficiently. These enhancements would make the system more robust, user-friendly, and aligned with modern industry standards.

REFERENCES

[1] Amazon Web Services Documentation – Cloud Front and S3

[2]Git Hub APID documentation

[3] Research on Content Delivery Networks and Web Performance

[4] Studies on CI/CD Pipelines in Web Development