www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 03, March 2025, pp : 15-20

e-ISSN : 2583-1062
Impact Factor : 7.001

STREAMLINING DEPLOYMENT VIA CLOUD SERVICES

Prof. G.L. Girhe[*1], Amol Harinkhede[*2], Harsh Pakhale[*3]

[*1]Head Of Department Computer Engineering Department, SRPCE, Nagpur, Maharashtra, India.

[*2,3]UG Student, Computer Engineering Department, SRPCE, Nagpur, Maharashtra, India.

DOI: https://www.doi.org/10.58257/IJPREMS38765

ABSTRACT

Our Cloud-Based Quick Deployment Service makes it easy to deploy, test, and monitor applications in the cloud by using an asset-based, actor-centric approach. Users can quickly deploy their code by providing a GitHub repository link, after which the system automatically sets up the environment, assigns a unique identifier, and runs the code on the cloud platform. This removes the hassle of manual setup, making development faster and more efficient. The service also improves security by allowing quick deployment of monitoring tools to detect threats and protect cloud assets. With features like automated environment management, real-time debugging, and built-in security checks, developers can focus on building applications instead of fixing configuration issues.

Keywords: VPC (Virtual Private Cloud), S3 Bucket, SQS (Simple Queuing Service), EC2 Instance, Redis.

1. INTRODUCTION

Building a cloud-based quick deployment Service transforms the way modern applications are developed and deployed. This platform simplifies the deployment process, allowing developers to focus more on creating innovative solutions rather than dealing with complex setup issues. By simply providing a GitHub repository link, users can have their environment automatically configured, including the installation of dependencies and necessary settings. Once the setup is complete, the service deploys and runs the code, offering a temporary domain for real-time testing

2. RELATED WORK

Cloud deployment has been a focus of various studies, exploring public, private, hybrid, and multi-cloud models. Tools like Kubernetes for orchestration and dynamic resource allocation for cost efficiency have been widely analyzed. Research highlights persistent challenges in security, compliance, and scalability, alongside emerging trends such as serverless and edge computing. This study builds on prior work to address gaps in interoperability, real-time scaling, and multi-tenant security, advancing cloud deployment practices.

3. METHODOLOGY

This section introduces a cloud-based deployment service designed to enable rapid and efficient application deployment directly from GitHub. The system automates the entire environment setup, dependency installation, and configuration process, ensuring that applications are deployed with minimal manual effort. By integrating with version control systems, the service detects changes in the repository, automatically triggers builds, and deploys the latest version of the application. To streamline the deployment workflow, the system leverages containerization and serverless computing, allowing applications to be deployed in isolated environments that ensure consistency across different cloud infrastructures. Automated provisioning allocates resources dynamically based on workload

4. PROBLEM STATEMENT

Deploying web applications at scale presents multiple challenges, requiring developers to handle code builds, infrastructure provisioning, cloud service configurations, and application scaling. These processes are often complex, time-consuming, and prone to misconfigurations, leading to increased costs, longer development cycles, and reduced application reliability. Developers, particularly those with limited cloud expertise, may struggle with optimizing performance, managing security risks, and ensuring seamless scalability across cloud environments

5. ARCHITECTURAL DESIGN

The architecture of the existing system is designed to facilitate the seamless deployment of web applications from GitHub repositories to the cloud. It encompasses three main phases: Uploading, Deployment, and Request Handling. Each phase is powered by dedicated services that work collaboratively to ensure efficient and smooth operations.

Uploading Phase

Users submit GitHub repository URLs via a simple interface. The upload service clones the repository and stores the project files securely in an AWS S3 bucket for processing.

www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 03, March 2025, pp : 10-14

e-ISSN : 2583-1062
Impact Factor : 7.001

Deployment Phase

The deployment service processes the uploaded files, building assets for frameworks like React by transforming JSX into static files (HTML, CSS, JavaScript). AWS auto-scaling, using SQS and EC2, manages resource scaling dynamically. Built assets are stored back in S3 for efficient distribution.

Request Handling Phase

An HTTP server handles user interactions, serving application files stored in S3. It supports file retrieval, caching strategies, and global delivery through CDNs to ensure fast, reliable access. Status checks and robust error handling enhance reliability and user experience.

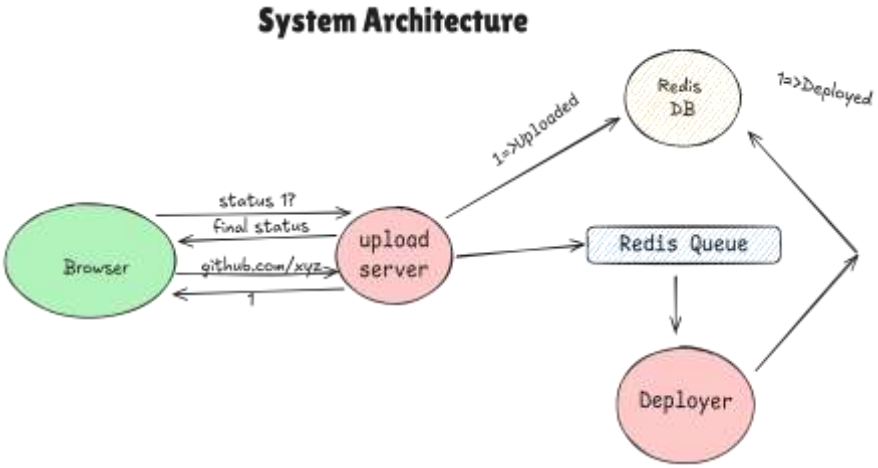

Fig 1: System Architecture

This design delivers an automated, scalable, and efficient workflow for deploying web applications from source code to live environments.

UPLOADING PHASE

The process starts with a user-friendly interface that allows users to submit their GitHub repository URLs. This action activates the upload service, which performs the following tasks:

Cloning the Repository: The upload service clones the repository from GitHub to capture the latest version of the code.

Uploading to S3: It subsequently uploads the project files to an AWS S3 bucket, offering a secure storage solution for the source code, which is crucial for further processing

When you upload a file to Amazon S3, it is stored as an S3 object. Objects consist of the file data and metadata that describes the object. You can have an unlimited number of objects in a bucket. Before you can upload files to an Amazon S3 bucket, you need write permissions for the bucket. For more information about access permissions, see Identity and Access Management for Amazon S3.

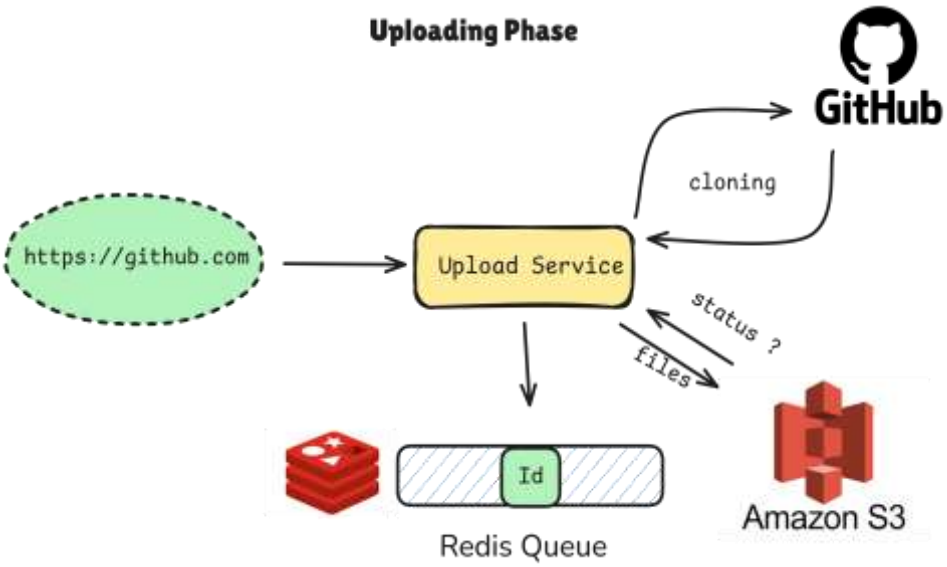

Fig 2: Upload Service

www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 03, March 2025, pp : 10-14

e-ISSN : 2583-1062
Impact Factor : 7.001

The initial phase involves uploading your project to the deployment service. This is where the system interacts with the GitHub repository to get the code that needs to be deployed. A persistent queue (often backed by Redis) is utilized to manage deployment tasks. This means that uploads can be processed in an orderly fashion, preventing server overload and ensuring that requests can be handled one at a time if necessary.

DEPLOYMENT PHASE

Once the upload is complete, the system signals the deployment service to initiate the transformation of the raw source code into deployable assets. Key responsibilities of the deployment service include:

Build Process**:** For projects built with React, the service transforms JSX and other components into static HTML, CSS, and JavaScript files.

Auto-Scaling: Utilizing AWS's auto-scaling features, including Amazon SQS for task queuing and EC2 or Faregate for dynamically adjusting computing resources, this service guarantees optimal performance during fluctuations in demand.

Once the build process is complete, the resulting assets are stored back in S3, ready for user access.

The task of designing a scalable, efficient, and cost-effective deployment solution should not be limited to how you will update your application version, but should also consider how you will manage supporting infrastructure throughout the complete application

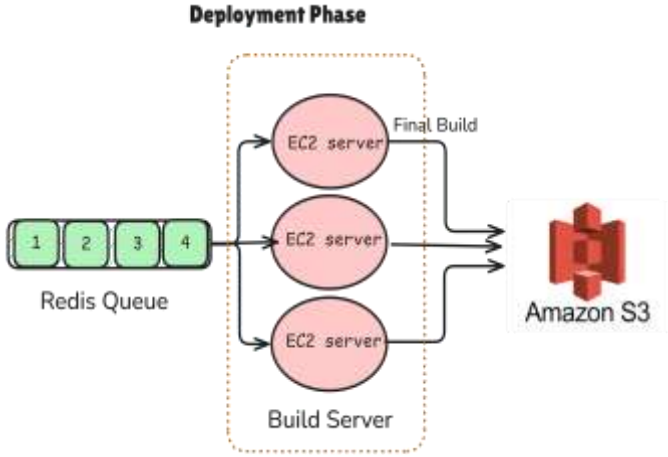

Fig 3: Deployment Service

REQUEST HANDLING PHASE

The request phase in a Cloud-Based Rapid Deployment Service (CBRDS) is critical as it manages how clients interact with the deployed applications. This phase involves receiving and processing requests from users, serving the appropriate application files, and handling application status checks. Below is a detailed breakdown of the entire request phase process:

- Receiving HTTP Requests

The first step in the request phase is to set up a robust HTTP server capable of handling requests from users. Utilizing the Express framework, we can create a server that listens for incoming requests on specified routes.

- Common request types include:
- GET requests to retrieve the application files.
- GET requests to check the deployment status.
- Each request is routed to appropriate handler functions based on the endpoint.

Fetching Application Files from Storage

Once the deployment ID is obtained, the system needs to search for the associated application files stored in an object storage service like AWS S3. This involves constructing a request to retrieve the relevant files based on the ID.

Using the deployment ID to retrieve the associated files from the storage service (e.g., AWS S3).

Construct an S3 get Object request to retrieve the files.

Serving the Application Files: After successfully retrieving the application files, the service must prepare to serve these files back to the client. This involves sending the appropriate files as a response to the user's request.

Depending on the nature of the request (whether it's for HTML content, CSS, or JavaScript), the server should send back the corresponding file with.

www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 03, March 2025, pp : 10-14

e-ISSN : 2583-1062
Impact Factor : 7.001

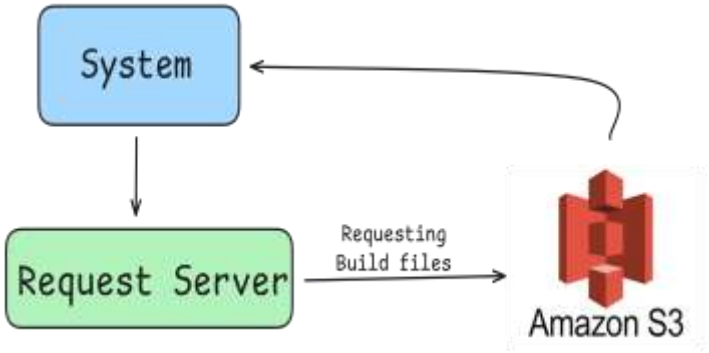

Fig 4: Request Phase

The request phase in a Cloud-Based Rapid Deployment Service (CBRDS) is a multifaceted process that manages user interactions and resource retrieval efficiently. By establishing a clear structure for receiving requests, unpackaging parameters, fetching application files, serving responses, handling status checks, and implementing robust error handling and logging, the service ensures a seamless and user-friendly experience.

6. RESULTS AND DISCUSSIONS

EC2 INSTANCE MONITORING

The percentage of physical CPU time that Amazon EC2 uses to run the EC2 instance, which includes time spent to run both the user code and the Amazon EC2 code. At a very high level, CPU-Utilization is the sum of guest CPU Utilization and hypervisor CPU Utilization. Tools in your operating system can show a different percentage than CloudWatch due to factors such as legacy device simulation, configuration of non-legacy devices, interrupt-heavy workloads, live migration, and live update.

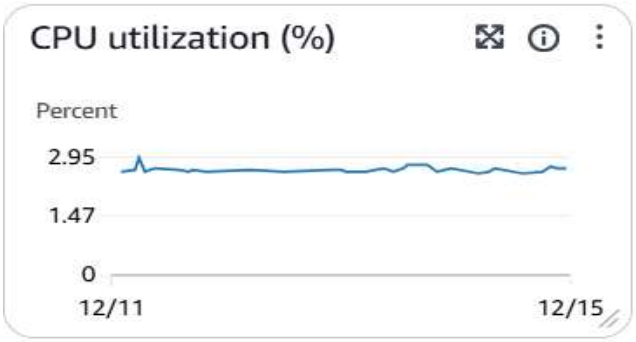

Fig 5: CPU Utilization

The number of packets sent out by the instance on all network interfaces. This metric identifies the volume of outgoing traffic in terms of the number of packets on a single instance.

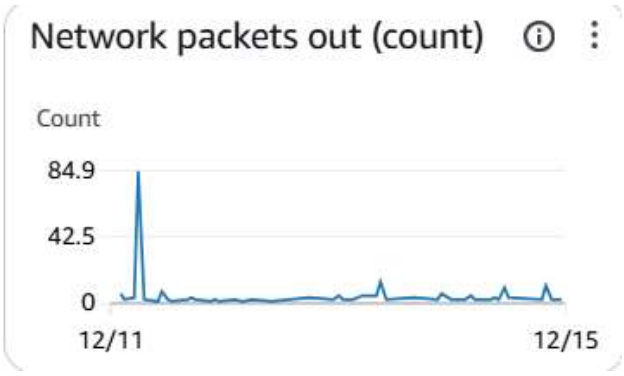

Fig 6: Network Packets out (count)

This metric is available for basic monitoring only (5-minute periods). To calculate the number of packets per second (PPS) your instance sent for the 5 minutes, divide the Sum statistic value by 300. You can also use the CloudWatch metric math function DIFF_TIME to find the packets per second. For example, if you have graphed Network Packets Out in CloudWatch as m1, metric math formula m1/(DIFF_TIME(m1)) returns the metric in packets/second. For more information about DIFF_TIME and other metric math functions.

www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 03, March 2025, pp : 10-14

e-ISSN : 2583-1062
Impact Factor : 7.001

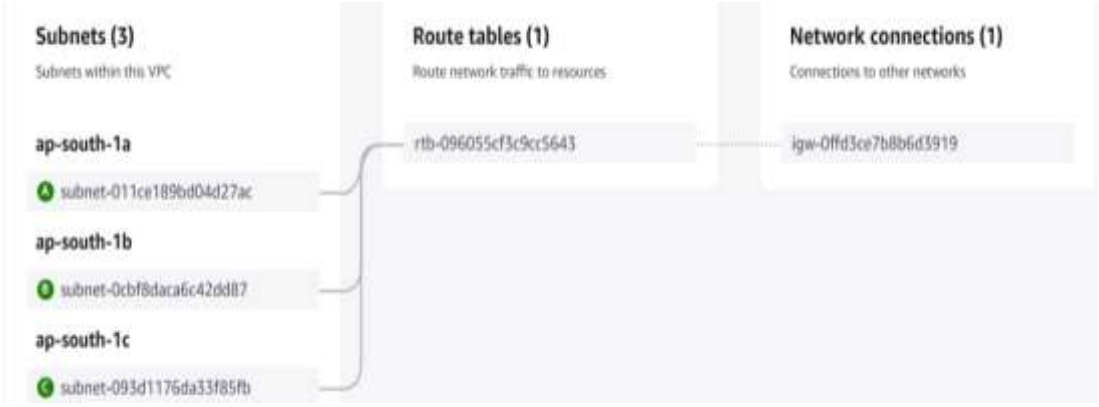

Fig 7: VPC (Virtual Private Cloud)

The resource map shows relationships between resources inside a VPC and how traffic flows from subnets to NAT gateways, internet gateway and gateway endpoints.

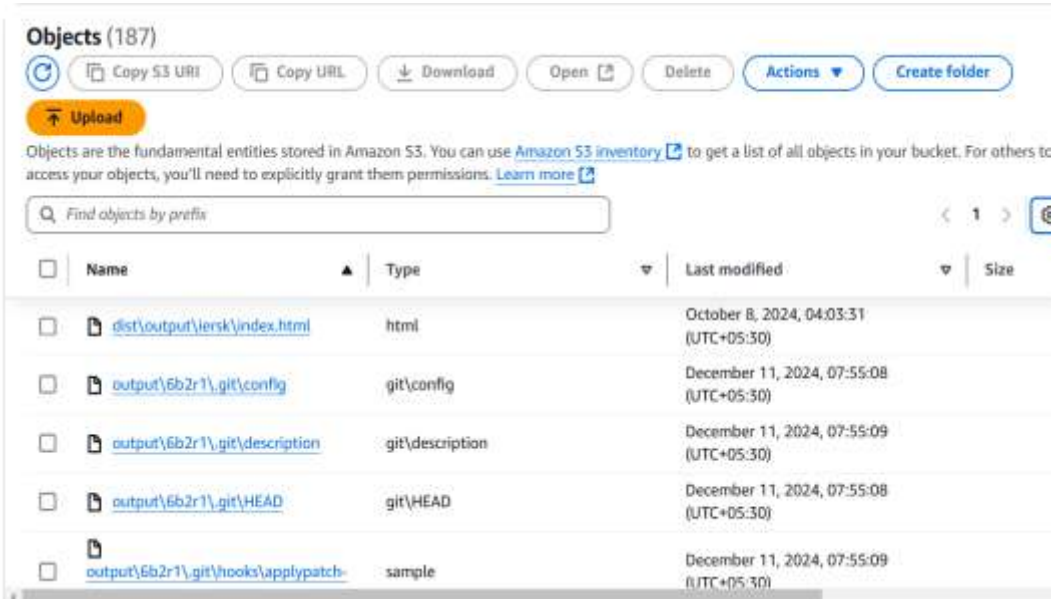

Fig 8: AWS S3

Amazon Simple Storage Service (Amazon S3) is an object storage service offering industry-leading scalability, data availability, security, and performance. Millions of customers of all sizes and industries store, manage, analyze, and protect any amount of data for virtually any use case, such as data lakes, cloud-native applications, and mobile apps. With cost-effective storage classes and easy-to-use management features.

7. CONCLUSION

The existing cloud-based rapid deployment service offers a streamlined and efficient solution for deploying web applications from GitHub repositories to the cloud. By leveraging robust services like AWS S3 for secure storage, auto-scaling with AWS EC2 and Fargate, and effective request handling through caching and global distribution, the system ensures scalability, reliability, and high performance. Its structured phases—uploading, deployment, and request handling—simplify complex deployment workflows, enabling seamless user interactions and quick application delivery. This architecture paves the way for a responsive, user-friendly platform that meets the growing demands for rapid and scalable web application deployment

8. FUTURE SCOPE

The future of deployment services has great potential for further development and scaling Integration with More Cloud Providers. Expanding beyond AWS to include Google Cloud Platform (GCP) and Microsoft Azure for more flexibility. Advanced Auto-Scaling and Load Balancing: Utilizing machine learning to predict traffic patterns and adjust server capacity dynamically. Support for Multiple Frameworks: Extending support to Angular, Vue.js, and full-stack applications like Node.js or Django. CI/CD Pipeline Integration: Evolving into a comprehensive CI/CD platform with automated testing, quality assurance, and rollback mechanisms.

9. REFERENCES

[1] Durner, D., Leis, V., & Neumann, T. (2024). Exploiting Cloud Object Storage for High-Performance Analytics Proceedings of the VLDB Endowment, 16(11), 2769-2782 3.

[2] Prof. G. L. Girhe, Amol Harinkhede, Harsh Pakhale , Deployment Through Cloud Services, Vol. 04, Issue 09, September 2024, pp : 929-931

[3] Prof. G. L. Girhe , Aditya Awathare, Shailesh Madankar, Quick Deployment Service, Vol. 04, Issue 09, September 2024, pp : 939-941

[4] Patel, S., et al.**,** Review of PaaS offerings such as AWS Elastic Beanstalk, Google App Engine, and Azure App Service for fast, scalable deployment of web applications. International Journal of Cloud Computing and Services Engineering, 2023. 10(2): p. 178-192..

[5] Li, P., et al., Comparing deployment speed, cost-efficiency, and cloud platform suitability for applications, focusing on Amazon Web Services and Microsoft Azure for enterprise solutions. IEEE Cloud Computing, 2023. 6(4): p. 115-128..

[6] Tan, Y., et al., Examining the role of automated scaling in cloud-based deployment services, improving application uptime and resource utilization with dynamic scaling. Cloud Computing Review, 2023. 14(3): p. 67-82.

[7] Chen, W., et al., Exploring various cloud service models (IaaS, PaaS, SaaS) for rapid application deployment, evaluating operational overhead and efficiency in cloud environments. Journal of Computing and Cloud Technology, 2023. 9(1): p. 32-44.

[8] Reddy, A., et al.**,** Evaluating multi-region deployment services in cloud platforms for global application performance, focusing on load balancing, redundancy, and failover strategies. Journal of Cloud Infrastructure, 2022. 18(7): p. 230-243.

[9] Platforms like AWS Lambda for rapid deployment, highlighting cost reductions, scalability, and simplified infrastructure management. Cloud Computing Advances, 2022. 10(3): p. 102-113

[10] Singh, K., et al., Review of serverless computing. Huang, L., et al., Optimizing deployment of high-performance web applications using cloud resources, focusing on latency reduction and efficient resource

[11] Zhang, T., et al., Investigating the security challenges in cloud-based rapid deployment services, focusing on encryption, authentication, and data privacy. IEEE Transactions on Cloud Computing, 2022. 8(6): p. 926-939.

[12] Brown, M., & Lee, S. “Building Custom Deployment Pipelines with AWS Services,” AWS Summit Proceedings, 2021.

[13] Duvvuri, K., & Prathibha, S. “A Study on CI/CD Pipeline Automation,” International Journal of Engineering and Advanced Technology, 2021.

[14] Liang, Q., et al., Surveying cloud platforms for continuous integration and delivery (CI/CD), evaluating tools and techniques for rapid application deployment. International Journal of Web and Cloud Computing, 2021. 10(1): p. 89-101..