CI/CD Build Analysis and Interactive Dashboard: A Data-Driven Approach to DevOps Optimization

DevOps Analytics Done

By:- **MIHIR MILIND UGHADE**

Abstract

This report presents a data-driven solution to enhance Continuous Integration and Continuous Deployment (CI/CD) pipeline performance through an interactive dashboard. By analyzing key metrics such as build times, resource utilization, and success rates, the project aims to identify bottlenecks, predict failures, and optimize DevOps processes. Using a simulated dataset of 200 records, we developed a framework for visualizing and interpreting CI/CD data, with potential for integration into real-world pipelines. The resulting dashboard empowers teams to make informed decisions, improving build reliability and resource efficiency.

1 Introduction

Modern software development relies heavily on efficient CI/CD pipelines to deliver high-quality code rapidly. However, the complexity and volume of pipeline data often obscure actionable insights. This project introduces an interactive dashboard that visualizes critical CI/CD metrics, enabling teams to monitor performance, detect inefficiencies, and enhance deployment strategies. The solution leverages Python-based analytics and interactive visualizations to provide a scalable framework for DevOps optimization.

2 Problem Statement

CI/CD pipelines generate extensive data, including build times, code complexity, and resource usage. Without effective analysis, teams struggle to:

- Identify bottlenecks causing prolonged build times.
- Predict and prevent build failures.
- Optimize resource allocation for cost and performance efficiency.

This project addresses these challenges by providing a clear, interactive interface to analyze pipeline metrics and support data-driven decision-making.

3 Methodology

The project was executed in five phases:

3.1 Data Simulation and Preparation

A synthetic dataset of 200 records was generated, including features such as author, deployment strategy, build time, and resource usage. Derived metrics (e.g., files per test, lines per file) were calculated to enrich analysis. Data validation ensured completeness and consistency.

3.2 Exploratory Data Analysis

We analyzed distributions of build times, success rates, and resource usage, identifying correlations between code complexity, test coverage, and build outcomes. Comparisons across authors and deployment strategies highlighted performance variations.

3.3 Interactive Dashboard Development

Using Plotly Express and ipywidgets, we created an interactive dashboard featuring:

- Scatter plots for build time versus predicted success probability.
- Bar charts comparing success rates by deployment strategy.
- Histograms of files changed and resource efficiency.
- Filters for author and deployment strategy.

3.4 Insights and Analysis

The dashboard revealed high-success deployment strategies, identified resource-heavy builds, and highlighted trade-offs between code complexity and reliability, supporting strategic decision-making.

3.5 Future Enhancements

Future work includes integrating real CI/CD logs, implementing machine learning for predictive analytics, and deploying the dashboard as a web application for team-wide access.

4 Technologies Used

The project utilized:

- Python: Core programming language.
- Pandas and NumPy: Data manipulation and analysis.
- Plotly Express: Interactive visualizations.
- ipywidgets: Dynamic filtering in Jupyter Notebook.
- Jupyter Notebook/Google Colab: Development environment.

5 Outcomes

The project delivered:

- An interactive dashboard for real-time CI/CD performance monitoring.
- Early identification of potential build failures through predictive metrics.
- Insights into resource utilization, enabling optimization.
- A reusable framework adaptable to real-world CI/CD data.

6 Conclusion

The CI/CD Build Analysis Dashboard provides a powerful tool for DevOps teams to enhance pipeline efficiency and reliability. By visualizing key metrics and enabling interactive exploration, it bridges the gap between raw data and actionable insights. Future enhancements will further integrate real-world data and advanced analytics, making this a cornerstone for data-driven DevOps.

7 References

- Project documentation, simulated dataset, and analysis notebooks, available at https://github.com/mihir0804/AI_DevOps_Pipeline_Optimize (includes Jupyter notebooks for data analysis and interactive dashboard, with instructions for generating the simulated dataset).
- Plotly Express documentation: https://plotly.com/python/plotly-express/.
- ipywidgets documentation: https://ipywidgets.readthedocs.io/.