CI-CD Deployment

Milan Ples - 20088120

01-01-2023

Contents

Abstract

The purpose of this project is to streamline the software development process through the use of a CI-CD workflow that automates the steps of building, testing, and deploying code changes. This is achieved through the utilization of git for version control, Github for code repository management, Jenkins for continuous integration, Docker for packaging code into containers, and AWS for deployment.

By implementing this workflow, developers can make code changes and have them automatically built, tested, and deployed, reducing the time and effort required to bring new features and updates to production. This enables teams to deliver software updates faster and with fewer errors, improving the overall efficiency and quality of the software development process.

Introduction

Continuous Integration-Continuous Deployment (CI-CD) is a software development approach that aims to automate the process of building, testing, and deploying code changes. This approach is designed to improve the efficiency and quality of the software development process by allowing teams to deliver updates and new features faster, with fewer errors.

One way to implement a CI-CD workflow is through the use of tools such as git for version control, Github for code repository management, Jenkins for continuous integration, Docker for packaging code into containers, and AWS for deployment. In this project, these tools are used together to create a streamlined process for building, testing, and deploying code changes. This enables developers to focus on writing code and adding new features, while the CI-CD pipeline handles the rest.

Purpose, Intended Use and Audience

The purpose of this CI-CD project is to automate the process of building, testing, and deploying code changes, in order to improve the efficiency and quality of the software development process. It is intended to be used by software development teams who want to streamline their workflow and deliver updates and new features to their users faster, with fewer errors.

The intended audience for this project is software developers and devops professionals who are responsible for building, testing, and deploying code changes. By implementing a CI-CD workflow, these individuals can focus on writing code and adding new features, while the pipeline handles the rest of the process. This allows them to work more efficiently and effectively, and helps to ensure that code changes are thoroughly tested and deployed in a timely manner.

Overall, the goal of this CI-CD project is to make the software development process more efficient, reliable, and scalable, enabling teams to deliver high-quality software updates to their users more quickly and with fewer errors.

Rationale for the Project

As a software developer, I have long recognized the importance of automation in streamlining the development process and reducing costs. When I was introduced to CI-CD during my internship, I was immediately struck by the potential of this approach to transform the way we build, test, and deploy code changes.

Through the use of tools such as git, Github, Jenkins, Docker, and AWS, a CI-CD pipeline can automate the process of building, testing, and deploying code changes, reducing the time and effort required to bring new features and updates to production. This not only improves the efficiency of the development process, but also helps to ensure the reliability and quality of the final product.

I have therefore made it a priority to advocate for the implementation of a CI-CD pipeline at my current organization. I am confident that this approach will greatly benefit our team and our users, and I am excited to see the positive impact it can have on our development process. By embracing automation and adopting a CI-CD workflow, I believe we can significantly improve the speed and reliability of our software development process, while also reducing costs and minimizing errors.

Goals and Requirements

Goals

Methodology

Development cycle

Planning

An Agile Approach with Kanban

Test-Driven-Development

Continuous Integration/Development

Technologies

AWS

Results

Discussion

Limitations

Conclusion

Bibliography

example: - Oakland, J. (1987). The Economics of Non-excludability. The Journal of Economic Perspectives, 1(1), 19-32. doi:10.1257/jep.1.1.19