UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

MURILO NUNES MARÇAL

COMPARATIVO DE FERRAMENTAS PARA INTEGRAÇÃO CONTÍNUA: ANÁLISE DE COMPLEXIDADE DE USO

GUARAPUAVA

2024

MURILO NUNES MARÇAL

COMPARATIVO DE FERRAMENTAS PARA INTEGRAÇÃO CONTÍNUA: ANÁLISE DE COMPLEXIDADE DE USO

Comparison of Continuous Integration Tools: Analysis of Complexity of Use

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Sitemas para Internet do Curso Tecnólogo em Sitemas para Internet da Universidade Tecnológica Federal do Paraná.

Orientador: Hermano Pereira

GUARAPUAVA

2024

RESUMO

Este trabalho apresenta um estudo comparativo entre três ferramentas de integração contínua: GitHub Actions, GitLab CI/CD e Jenkins. O objetivo é analisar a complexidade de uso de cada ferramenta, buscando identificar suas vantagens e desvantagens no contexto de desenvolvimento de software. A justificativa para este estudo está na crescente adoção de metodologias ágeis e a necessidade de soluções eficientes para automatizar fluxos de integração e entrega contínua. A metodologia deste trabalho consiste na criação de esteiras em cada uma das plataformas, seguido de uma avaliação de facilidade de configuração, curva de aprendizado e suporte a plugins. Para garantir a consistência e a confiabilidade dos resultados, os testes serão realizados em um ambiente controlado, onde as variáveis de infraestrutura permanecerão constantes em todas as ferramentas. Dessa forma, será possível obter uma comparação precisa das capacidades de cada plataforma no contexto de automação de fluxos de desenvolvimento e entrega de software.

Palavras-chave: entrega contínua; integração contínua; análise de performance; desenvolvimento e operações; .

ABSTRACT

This paper presents a comparative analysis of three continuous integration tools: GitHub Actions, GitLab CI/CD, and Jenkins. The objective is to evaluate the complexity of each tool to identify their respective advantages and disadvantages in the software development context. This study is motivated by the growing adoption of agile methodologies, which emphasize the need for efficient solutions to automate continuous integration and delivery workflows. The methodology consists of creating pipelines on each platform, followed by an assessment of configuration ease, learning curve, and plugin support. To ensure consistency and reliability of the results, tests will be conducted in a controlled environment, where infrastructure variables remain constant across all tools. This approach aims to provide an accurate comparison of each platform's capabilities in automating development and delivery workflows.

Keywords: continuous integration; continuous delivery ; benchmarking ; devops ; .

LISTA DE ABREVIATURAS E SIGLAS

Abreviaturas

art.	Artigo
cap.	Capítulo
sec.	Seção

Siglas

CI/CD	Continuous Integration / Continuous Deployment (Integração Contínua / Entrega Contínua)
DevOps	Development and Operations (Integração entre desenvolvimento e operações)
VPS	Virtual Private Server (Servidor Virtual Privado)

SUMÁRIO

1 INTRODUÇÃO

A automação de processos no desenvolvimento de software, através de ferramentas de integração contínua (CI/CD), é uma prática essencial no contexto de DevOps e desenvolvimento ágil, facilitando a entrega rápida e confiável de software. À medida que a complexidade dos projetos aumenta, ferramentas como GitHub Actions[1], GitLab CI/CD[2]e Jenkins[3] surgem como soluções importantes para automatizar processos e garantir qualidade e eficiência no ciclo de desenvolvimento. Segundo (DUVALL PAUL M.; GLOVER, 2007), a integração contínua reduz riscos ao identificar erros de forma antecipada e melhora a qualidade do software, permitindo uma validação constante a cada commit. A entrega contínua, por sua vez, conforme argumentado por (HUMBLE; FARLEY, 2010), assegura que o software esteja sempre em um estado pronto para a implantação, o que é essencial para atender às exigências de um mercado dinâmico. Este trabalho busca realizar um comparativo entre estas três ferramentas para CI/CD, com foco em suas funcionalidades, facilidade de uso e curva de aprendizado.

1.1 Considerações iniciais

A integração contínua (CI) e a entrega contínua (CD) são práticas fundamentais no desenvolvimento de software moderno, especialmente em ambientes que adotam metodologias ágeis e DevOps. Essas práticas são implementadas por meio de esteiras automatizadas, conhecidas como *pipelines*, para organizar e executar as etapas de compilação, testes, integração, revisão e implantação, permitindo um ciclo de desenvolvimento acelerado e com maior confiabilidade no processo de entrega de software (FOWLER, 2006). De acordo com Humble e Farley (HUMBLE; FARLEY, 2010), a integração e entrega contínua são essenciais para reduzir o tempo entre o desenvolvimento de uma funcionalidade e sua disponibilização, além de mitigar os riscos de falhas em produção.

Cada uma dessas plataformas apresenta suas características particulares, vantagens e desafios que podem impactar diretamente a eficiência de equipes de desenvolvimento. A escolha de uma ferramenta adequada pode afetar desde a facilidade de implementação das esteiras até a eficiência geral da equipe e do sistema (DYCK; PENNERS; LICHTER, 2013). Como afirmado por Duvall (DUVALL PAUL M.; GLOVER, 2007), o sucesso na adoção de CI/CD depende não apenas da escolha das ferramentas, mas também da integração dessas com as práticas da equipe e do projeto.

GitHub Actions e GitLab oferecem integração nativa com o controle de versão, o que proporciona uma experiência mais fluida para desenvolvedores que já utilizam essas ferramentas. Por exemplo, o GitHub Actions, integrado ao ecossistema do GitHub, permite automação

[1] https://github.com/
[2] https://about.gitlab.com/
[3] https://www.jenkins.io/

de *workflows* diretamente no repositório com *templates* de CI/CD prontos e integrações de ferramentas de terceiros pelo GitHub Marketplace[4] (GitHub, Inc., 2024), (GitLab, Inc., 2024).

Jenkins, por outro lado, como uma ferramenta open-source altamente configurável, permite uma flexibilidade maior em termos de customização e uso de plugins (Jenkins, 2024a). Entretanto, essa flexibilidade vem acompanhada de uma complexidade maior de configuração e uma curva de aprendizado mais acentuada, Segundo Riungu-Kalliosaari (RIUNGU-KALLIOSAARI; MäKINEN; TYRVäINEN, 2016), ferramentas como Jenkins, apesar de seu potencial de flexibilidade, podem ser mais adequadas para equipes com maior maturidade técnica, sendo preferido em cenários que exigem customizações específicas e fluxos de trabalho complexos.

Neste cenário, existe uma demanda crescente por estudos que avaliem de forma comparativa essas plataformas de CI/CD, fornecendo diretrizes claras para diferentes contextos de uso (SHAHIN; BABAR; ZHU, 2017). Este trabalho insere-se nesse contexto ao realizar uma análise comparativa, considerando aspectos como facilidade de uso, complexidade de configuração e desempenho de execução dos pipelines.

1.2 Objetivos

Os objetivos deste trabalho consistem em realizar uma análise comparativa entre três ferramentas de integração e entrega contínua. De forma que os resultados alcançados possam auxiliar desenvolvedores e equipes de T.I na escolha da ferramenta mais adequada para seus projetos.

1.2.1 Objetivo geral

Avaliar e comparar a complexidade de uso, curva de aprendizado e desempenho das ferramentas de integração e entrega contínua GitHub, GitLab e Jenkins.

1.2.2 Objetivos específicos

Avaliar a facilidade de uso e configuração das ferramentas GitHub, GitLab e Jenkins no contexto de pipelines de CI/CD. Analisar o desempenho dessas ferramentas em diferentes cenários, utilizando uma rotina padronizada para medir o tempo de configuração das pipelines. Identificar as principais características e limitações de cada ferramenta, comparando suas funcionalidades e eficácia. Oferecer recomendações práticas para a escolha da ferramenta mais adequada a diferentes contextos de desenvolvimento, baseadas nos resultados obtidos.

[4] https://github.com/marketplace

1.3 Justificativa

A justificativa para este trabalho reside na necessidade de aprofundar o entendimento das equipes de desenvolvimento sobre práticas de Integração Contínua (CI) e Entrega Contínua (CD) no desenvolvimento de software, práticas fundamentais para a automação de processos e aumento de eficiência no desenvolvimento de software (FOWLER, 2006).

A escolha das plataformas GitHub, GitLab e Jenkins não é apenas técnica; ela reflete a busca por soluções que atendam a diferentes necessidades das equipes. GitHub e GitLab, por exemplo, oferecem interfaces acessíveis e integrações nativas com sistemas de controle de versão (GitHub, Inc., 2024), (GitLab, Inc., 2024). E o Jenkins, embora mais complexo, oferece flexibilidade de configuração, com uma vasta quantidade de plugins, que possibilitam um alto nível de customização (Jenkins, 2024b).

A análise aqui proposta avalia fatores como facilidade de uso, complexidade de configuração das esteiras, fornecendo tanto uma contribuição acadêmica quanto um guia prático. Além disso, o estudo também contribui para a literatura acadêmica e prática, auxiliando equipes de desenvolvimento e engenheiros de *software* na escolha da ferramenta mais apropriada para suas necessidades específicas, contribuição importante para o mercado de trabalho, onde a adoção de práticas CI/CD é cada vez mais indispensável para a manutenção da qualidade do *software* e do ciclo de entrega (RIUNGU-KALLIOSAARI; MäKINEN; TYRVäINEN, 2016).

REFERÊNCIAS

DUVALL PAUL M., M. S.; GLOVER, A. **Continuous Integration: Improving Software Quality and Reducing Risk**. [*S.l.*]: Addison-Wesley Professional, 2007.

DYCK, A.; PENNERS, R.; LICHTER, H. Towards definitions for release engineering and devops. *In*: **Proceedings of the 3rd International Workshop on Release Engineering (RELENG '13)**. [*S.l.*]: IEEE, 2013.

FOWLER, M. Continuous integration. **https://martinfowler.com/articles/continuousIntegration.html**, 2006.

GitHub, Inc. **GitHub Actions Overview**. [*S.l.*], 2024. Acessado em 7 de novembro de 2024. Disponível em: https://resources.github.com/devops/tools/automation/actions/.

GitLab, Inc. **Get started with GitLab CI/CD**. [*S.l.*], 2024. Acessado em 3 de novembro de 2024. Disponível em: https://docs.gitlab.com/ee/ci/.

HUMBLE, J.; FARLEY, D. **Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation**. [*S.l.*]: Addison-Wesley Professional, 2010.

Jenkins. **Documentação do Jenkins**. [*S.l.*], 2024. Acessado em 12 de novembro de 2024. Disponível em: https://www.jenkins.io/doc/.

Jenkins. **Suporte de Plugins do Jenkins**. [*S.l.*], 2024. Acessado em 10 de outubro de 2024. Disponível em: https://plugins.jenkins.io/.

RIUNGU-KALLIOSAARI, L.; MäKINEN, S.; TYRVäINEN, P. Devops adoption benefits and challenges in practice: A case study. **Software: Practice and Experience**, v. 46, n. 6, p. 779–799, 2016.

SHAHIN, M.; BABAR, M. A.; ZHU, L. Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. **IEEE Access**, v. 5, p. 3909–3943, 2017.