josif 2025
17ª Jornada Científica e Tecnológica
14º Simpósio de Pós-Graduação
do IFSULDEMINAS
ISSN: 2319-0124
REALIZAÇÃO
INSTITUTO FEDERAL
Sul de Minas Gerais
APOIO
CNPq
FAPEMIG

OTIMIZAÇÃO DE PIPELINES CI/CD NO GITLAB: Uma Abordagem Utilizando Aprendizado de Máquina para Caching Preditivo

Diogo H. da SILVA[1]; Paulo C. Dos SANTOS[2]

RESUMO

A otimização de pipelines de Integração e Entrega Contínua (CI/CD) é um desafio central em ambientes DevOps, onde o tempo de build de imagens Docker in Docker (DinD) representa um gargalo crítico. A gestão de cache, é uma das principais estratégias para mitigar a lentidão, porém, é tipicamente manual, reativa. A proposta deste trabalho é analisar a viabilidade da aplicação de Aprendizado de Máquina para transformar a decisão baseada em suposições em uma ação preditiva e otimizada. A abordagem consiste em desenvolver um modelo capaz de decidir, com base nos metadados de uma alteração de código, se a estratégia mais eficiente é utilizar o cache existente ou realizar um build limpo. Para validar a abordagem, foi construído um laboratório experimental com GitLab e Docker, onde um pipeline automatizado executa builds em ambos os cenários e coleta métricas detalhadas. Como resultado principal, foi gerado um dataset estruturado, validando a viabilidade da coleta e estabelecendo a base para o futuro treinamento de um modelo de classificação para otimização de pipelines.

Palavras-chave:
DevOps; CI/CD; GitLab; Aprendizado de Máquina; Otimização de Cache.

1. INTRODUÇÃO

A crescente adoção de metodologias ágeis no cenário atual de desenvolvimento de software vem impulsionando a competitividade. Diante desta necessidade, as organizações adotaram a cultura DevOps que utiliza pipelines de CI/CD para acelerar a entrega de software (RED HAT, 2022).

A conteinerização com Docker é padrão, mas o tempo de build de imagens é um gargalo. O uso de cache é a principal técnica para mitigar a lentidão (DOCKER, 2025), mas sua gestão manual é ineficiente, podendo levar a falhas ou ao uso de dependências desatualizadas (SANTOS, 2023). A proposta é investigar o uso de Aprendizado de Máquina para otimizar essa decisão, uma abordagem corroborada por estudos que demonstram a eficácia de modelos preditivos em pipelines (DILEEPKUMAR; MATHEW, 2025).

3. MATERIAL E MÉTODOS

A pesquisa foi um estudo experimental para avaliar a viabilidade da coleta de dados em um ambiente CI/CD controlado. O laboratório foi implementado com GitLab, GitLab Runner e um servidor DNS em máquinas virtuais. Um pipeline (.gitlab-ci.yml) foi desenvolvido para um projeto de API em Python (Flask), executando, a cada commit, dois jobs de build em paralelo: build_no_cache e build_with_cache . Um script Shell extraiu métricas do Git, GitLab e Docker. Os dados foram consolidados em um arquivo CSV (training_dataset.csv), salvo como artefato do

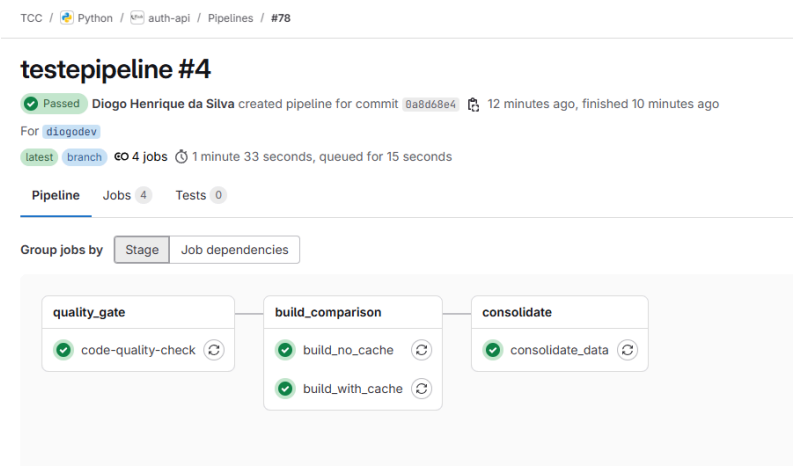
pipeline.

4. RESULTADOS E DISCUSSÃO

A fase preparatória do projeto foi concluída, validando a metodologia de coleta de dados que é crucial para a sustentação dos modelos preditivos. O principal resultado foi a implementação de um pipeline de CI/CD funcional, projetado para a coleta automatizada e comparativa de métricas.

A Figura 1 exibe o fluxo de execução deste pipeline no GitLab. Após um commit, o pipeline executa uma sequência de estágios: primeiro o quality_gate para análise estática de código, seguido pelo build_comparison, que executa em paralelo os builds com e sem cache. Por fim, o estágio consolidate unifica os dados coletados. O status de sucesso em todos os jobs, como observado na figura, confirma a operacionalidade do fluxo e a capacidade de gerar os dados necessários para a análise.

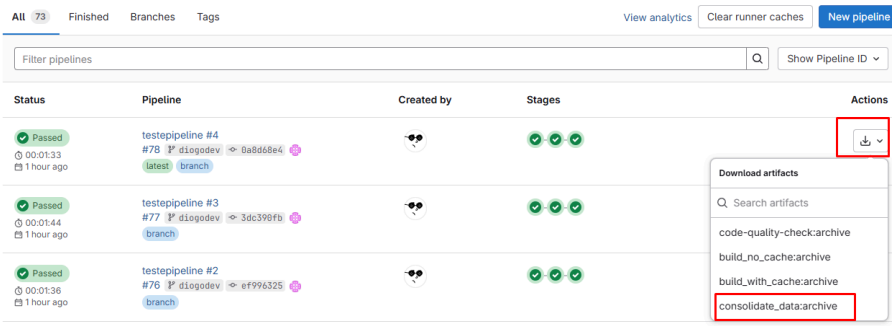
Figura 1: Fluxo de execução da pipeline no GitLab.



Fonte: dos autores.

Ao final de cada execução, o pipeline armazena os artefatos gerados, como ilustrado na Figura 2. O artefato consolidate_data:archive contém o training_dataset.csv, que é o produto final desta fase. Cada linha do dataset representa uma execução, correlacionando as características de uma alteração no código com as métricas de performance de cada estratégia de cache (com e sem). A coleta padronizada assegura que as diferenças observadas refletem apenas o impacto do uso do cache, eliminando vieses do ambiente.

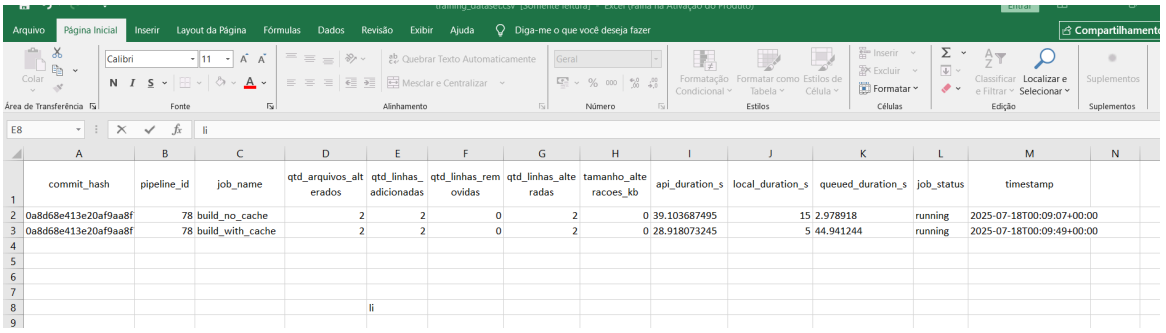
Figura 2: Download dos artefatos gerados pelo pipeline, incluindo o dataset consolidado.



Fonte: dos autores.

Esses resultados são fundamentais, pois fornecem a base de dados necessária para a próxima fase do trabalho: o treinamento e a validação de modelos de classificação (como XGBoost e Random Forest) para a tomada de decisão preditiva.

Figura 3: Dataset gerado pelo pipeline do Gitlab.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	commit_hash	pipeline_id	job_name	qtd_arquivos_alterados	qtd_linhas_adicionadas	qtd_linhas_removidas	qtd_linhas_alteradas	tamanho_alteracoes_kb	api_duration_s	local_duration_s	queued_duration_s	job_status	timestamp	
2	0a8d68e413e20af9aa8f	78	build_no_cache	2	2	0	2	0	39.103687495	15	2.978918	running	2025-07-18T00:09:07+00:00	
3	0a8d68e413e20af9aa8f	78	build_with_cache	2	2	0	2	0	28.918073245	5	44.941244	running	2025-07-18T00:09:49+00:00	
4														
5														
6														
7														
8					li									
9														

A Figura 3 demonstra o impacto do cache, com uma economia de 10.19s na duração da API e 10s na duração local do build, validando o potencial da abordagem.

5. CONCLUSÃO

Este trabalho demonstrou a viabilidade de construir um sistema automatizado para coleta de dados de performance em pipelines CI/CD, resolvendo o desafio inicial de obter um dataset confiável para análise. A infraestrutura e o pipeline desenvolvidos se mostraram eficazes, culminando na geração de um conjunto de dados estruturado e pronto para ser utilizado no treinamento de modelos de aprendizado de máquina.

Conclui-se que a abordagem experimental foi bem-sucedida e constitui um passo essencial para a criação de um sistema de caching preditivo. O trabalho futuro se concentrará na aplicação de algoritmos de ML sobre os dados coletados para desenvolver e integrar um modelo de classificação funcional ao pipeline, com o objetivo final de validar seu impacto real na otimização do tempo de build

REFERÊNCIAS

DILEEPKUMAR, S.R.; MATHEW, J. Optimizing Continuous Integration and Continuous Deployment Pipelines with Machine Learning: Enhancing Performance and Predicting Failures. Advances in Science and Technology Research Journal, [S. l.], 2025. Disponível em: https://www.astrj.com/pdf-197406-120644?filename=Optimizing%20continuous.pdf. Acesso em: 10 jun. 2025.

DOCKER. Using the build cache. Docker Docs, 2025. Disponível em: https://docs.docker.com/get-started/docker-concepts/building-images/using-the-build-cache/. Acesso em: 08 jun. 2025.

RED HAT. O que é CI/CD?. Red Hat, 18 out. 2022. Disponível em: https://www.redhat.com/pt-br/topics/devops/what-is-ci-cd. Acesso em: 10 jun. 2025.

SANTOS, Bruno Martins. Segurança em Pipelines de CI/CD: Análise de Riscos, Detecção de Anomalias e Notificações através de um Chatbot Inteligente. 2023. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Universidade Federal de Pernambuco, Recife, 2023. Disponível em: https://repositorio.ufpe.br/bitstream/123456789/53196/1/TCC%20Bruno%20Martins%20Santos.pdf. Acesso em: 10 jun. 2025.