**Nome do aluno:** Mateus França Briet **Nome da instituição:** Anhanguera **Curso:** Engenharia de Software **Disciplina:** Infraestrutura Ágil

**Título do trabalho:** Relatório de Aula Prática **Cidade – Estado:** SP **Ano:** Set/2025

**Link do repositório no GitHub:**

Relatório Completo: Análise de Aula Prática - Infraestrutura Ágil

1. Introdução

Este relatório tem como objetivo apresentar uma análise detalhada do "Relatório de Aula Prática - Infraestrutura Ágil" fornecido, destacando as aplicações e ferramentas utilizadas, os procedimentos realizados e as necessidades para a execução da atividade. O documento original descreve uma aula prática focada na simulação do monitoramento de um processo de pipeline de entrega utilizando o Git, com ênfase em conceitos de Integração Contínua (CI/CD).

2. Análise do Relatório Original: "Roteiro Aula Prática - Infraestrutura Ágil"

O roteiro da aula prática estabelece como objetivo principal a simulação do monitoramento de processo de pipeline de entrega, utilizando o GIT. Para isso, são delineados os seguintes aspectos:

2.1. Infraestrutura Necessária

Para a execução da aula prática, a infraestrutura listada inclui:

* Instalações: GIT.
* Materiais de Consumo: Computador (1 por aluno).
* Software: GIT (freeware).
* Equipamento de Proteção Individual (EPI): NSA (Não especificado, mas presumivelmente não aplicável ou um erro de digitação, pois EPIs não são comuns em atividades de infraestrutura de software).

2.2. Procedimentos Práticos

Os procedimentos práticos envolvem a realização do monitoramento do processo de pipeline de entrega com o Git. A atividade proposta visa:

* Entender o funcionamento do script para a realização da Integração Contínua.
* Criar um relatório ao final da atividade (o que este documento se propõe a fazer).

2.3. Procedimentos para a Realização da Atividade

O roteiro detalha a criação de um script para Integração Contínua, comparando o processo com o GitLab CI/CD. Os pontos chave incluem:

* GitLab CI/CD: O pipeline é definido em um arquivo .gitlab-ci.yml, que utiliza a linguagem YAML para estruturar jobs e a ordem de execução do pipeline.
* Jobs: São os elementos básicos do .gitlab-ci.yml, definidos com restrições e contendo a cláusula script.
* Loja Virtual: O projeto da aula prática é uma "Loja Virtual", para a qual o pipeline será definido.
* Contas Necessárias: É explicitada a necessidade de criar contas no GitLab e no Docker Hub.
* Arquivos do Projeto: O projeto "devops-master" é descompactado, e arquivos específicos são utilizados, como gitlab-ci.yml, Dockerfile, notificacaoFalha.sh, notificacaoSucesso.sh, plot\_simples\_grafico.py e relogio.py.
* Repositório: O projeto UNOPAR\_CI\_CD é mencionado como o repositório onde os arquivos serão "puxados" para o GitLab.
* Docker Hub: Utilizado para baixar imagens para construir a aplicação, sendo o padrão para busca e registro de imagens.

2.4. Checklist e Resultados

A checklist da atividade inclui:

* Instalar o sistema GIT.
* Simular se houve sucesso ou falha na execução do job.

Os resultados esperados da aula prática consistem na elaboração de um relatório contendo introdução, métodos, resultados e conclusão sobre o pipeline, visando a compreensão de seu funcionamento para a construção automática de aplicações.

3. Aplicações e Ferramentas Utilizadas e Necessárias

Com base na análise do relatório, as seguintes aplicações e ferramentas são essenciais para a execução da atividade:

3.1. GIT

* Descrição: Sistema de controle de versões distribuído, fundamental para o gerenciamento do código-fonte e o rastreamento de alterações em projetos de software [1].
* Função na Atividade: É a ferramenta central para o controle de versão do projeto e para a simulação do monitoramento do pipeline de entrega. Sua instalação é um pré-requisito.
* Recurso: Download e instalação: <https://git-scm.com/downloads>

3.2. GitLab CI/CD

* Descrição: Uma plataforma de Integração Contínua e Entrega Contínua (CI/CD) integrada ao GitLab, que permite automatizar as etapas de desenvolvimento, teste e implantação de software [2].
* Função na Atividade: Utilizado para definir o pipeline de entrega através do arquivo .gitlab-ci.yml, que orquestra os "jobs" (tarefas) a serem executados, como a construção e o teste da aplicação.
* Recurso: Criação de conta: <https://gitlab.com/users/>

3.3. Docker Hub

* Descrição: Um serviço de registro de imagens de contêineres que permite aos desenvolvedores armazenar, compartilhar e gerenciar imagens Docker. É um repositório público e privado para imagens de contêineres [3].
* Função na Atividade: Essencial para a busca e o registro de imagens Docker que serão utilizadas na construção da aplicação dentro do pipeline de CI/CD. As imagens são a base para a execução dos ambientes isolados dos jobs.
* Recurso: Criação de conta: <https://hub.docker.com/>

4. Processo de Integração Contínua na Aula Prática

O processo de Integração Contínua descrito na aula prática segue os seguintes passos:

1. Criação do Script Inicial: Desenvolvimento de um script que possibilita a realização da Integração Contínua.
2. Definição do Pipeline com **.gitlab-ci.yml**: Configuração do pipeline de CI/CD no GitLab utilizando um arquivo .gitlab-ci.yml. Este arquivo define os estágios (stages) e as tarefas (jobs) que compõem o pipeline.
3. Estrutura do **.gitlab-ci.yml**: O arquivo é composto por jobs, que são as unidades básicas de execução. Cada job pode ter restrições de execução e deve conter uma cláusula script com os comandos a serem executados.
4. Uso de Imagens Docker: O script de Integração Contínua faz uso de imagens Docker, que são baixadas do Docker Hub. Essas imagens fornecem o ambiente necessário para construir e testar a aplicação.
5. Execução e Monitoramento: O pipeline executa os jobs definidos. Ao final da execução, o sucesso ou falha do job é determinado, e o pipeline acusa uma falha caso o script não seja concluído com êxito.
6. Repositório **UNOPAR\_CI\_CD**: Os arquivos do projeto são "puxados" para este repositório no GitLab, onde o pipeline é então acionado.

5. Conclusão

A aula prática de "Infraestrutura Ágil" demonstra a aplicação de conceitos fundamentais de DevOps, como controle de versão com Git e automação de pipeline com GitLab CI/CD e Docker. A atividade visa proporcionar uma compreensão prática de como essas ferramentas se integram para criar um fluxo de entrega contínua eficiente e monitorável. A elaboração de um relatório final, conforme solicitado, consolida o aprendizado e a documentação das etapas e ferramentas envolvidas.

6. Referências

[1] Git. (n.d.). About Git. Retrieved from <https://git-scm.com/about> [2] GitLab. (n.d.). What is GitLab CI/CD?. Retrieved from <https://docs.gitlab.com/ee/ci/> [3] Docker. (n.d.). Docker Hub. Retrieved from <https://hub.docker.com/>