ROTEIRO LABORATÓRIO

LABORATÓRIO DIDÁTICO: CALCULADORA_SIMPLES COM PREFECT

🌟 OBJETIVO GERAL

Desenvolver uma aplicação de calculadora modular e orientada a objetos, com entregas incrementais e deploys automatizados utilizando Prefect 2.x e GitHub Actions. O laboratório demonstra na prática os princípios de orquestração, CI/CD, testes automatizados e boas práticas de Engenharia de Software.

✅ ETAPA 0 — PREPARAÇÃO DO AMBIENTE

📌 *OBJETIVO:*

Criar a fundação para o projeto, configurar o ambiente e garantir que tudo esteja limpo para início seguro.

🔧 AÇÕES:

1. Criar ambiente virtual `DEV` com Python 3.12.5.
2. Instalar dependências iniciais:

```
pip install prefect==2.16.4 griffe==0.35.2
```

4. Criar estrutura de pastas:

```
calculator_prefect/
├── src/
│   └── calculadora/
│       ├── __init__.py
│       └── operacoes.py
├── flows/
├── tests/
├── infra/
├── docs/
└── .github/workflows/
```

15. Inicializar repositório Git, conectar ao GitHub.
16. Criar arquivos:
    - `.gitignore`, `.prefectignore`
    - `prefect.yaml`
    - `requirements.txt`
    - `docs/readme.txt`: escopo inicial com soma e subtração.

✅ ETAPA 1 — MODELAGEM DA SOLUÇÃO E PROJETO OO

📌 OBJETIVO:

Modelar a arquitetura da calculadora antes de codificar, com base em princípios de OO e modularização.

🛠️ AÇÕES:

1. Criar **Canvas do Projeto** (docs/canvas.md)
2. Criar **UML de Classes** com foco em `Calculadora` e `Operacoes`.
3. Criar `src/calculadora/operacoes.py`: implementar `somar` e `subtrair`.
4. Criar `flows/flow_versao1.py`: flow básico Prefect.
5. Criar `tests/test_operacoes.py` com `unittest`.
6. Configurar `prefect.yaml` com o entrypoint correto.
7. Criar workflow `.github/workflows/prefect-deploy.yml`.
8. Criar secrets no GitHub: `PREFECT_API_KEY`, `PREFECT_API_URL`.

✅ ETAPA 2 — PRIMEIRO DEPLOY: SOMA E SUBTRAÇÃO

📌 OBJETIVO:

Concluir a versão V0 com deploy automatizado.

🚀 AÇÕES:

1. Testar localmente com:
2. `python -m unittest discover -s tests`
3. Executar:
4. `prefect deploy -n hello-deploy`
5. Confirmar deploy no GitHub Actions.

✅ ETAPA 3 — AMPLIAÇÃO FUNCIONAL: MULTIPLICAÇÃO E DIVISÃO

📌 OBJETIVO:

Expandir funcionalidades e evidenciar a evolução do projeto com versionamento e deploy incremental.

➕ AÇÕES:

1. Atualizar `operacoes.py` com `multiplicar` e `dividir` (com `try/except`).

2. Adicionar novos testes.
3. Atualizar `flow_versao1.py` com chamadas às novas funções.
4. Executar testes, fazer commit e push.
5. Verificar execução no GitHub Actions.

✅ ETAPA 4 — FINALIZAÇÃO COM DOCKER E PREFECT SERVER LOCAL

📌 OBJETIVO:

Executar localmente o Prefect Server com Docker e registrar o deploy off-cloud.

🐳 AÇÕES:

1. Criar scripts:
   - `infra/run_server.sh`
   - `infra/run_worker.sh` Ambos com permissão de execução (`chmod +x`).
2. Subir servidor Prefect:

```
./infra/run_server.sh
```

4. Registrar flow localmente:

```
prefect deploy -n hello-deploy
```

6. Iniciar o worker:

```
./infra/run_worker.sh
```

8. Executar via Prefect UI: http://localhost:4200

✅ ETAPA 5 — REORGANIZAÇÃO COM SRC/ E INTEGRAÇÃO DOS TESTES

📌 OBJETIVO:

Refinar arquitetura com src/, adicionar % e consolidar CI/CD com testes embutidos no flow Prefect.

⌨ AÇÕES:

1. Garantir que os imports referenciem `from calculadora.operacoes import ...`
2. Atualizar `PYTHONPATH` no VSCode e GitHub Actions
3. Adicionar função `calcular_porcentagem`
4. Refatorar `flow_versao1.py` para incluir:

```
@task
```

def executar_testes(): subprocess.run(["python", "-m", "unittest", "discover", "-s", "tests"], check=True)

5. Garantir falha automática no deploy se testes falharem
6. Validar execução via CLI e Prefect UI

📚 Conclusão Didática

Este laboratório ensina, com rigor técnico e progressão prática:

- Como orquestrar aplicações com Prefect
- Como organizar código real com boas práticas
- Como evoluir sistemas com controle e rastreabilidade
- Como integrar testes e CI/CD de forma confiável