# 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:
3. pip install prefect==2.16.4 griffe==0.35.2
4. Criar estrutura de pastas:
5. calculator\_prefect/
6. ├── src/
7. │ └── calculadora/
8. │ ├── \_\_init\_\_.py
9. │ └── operacoes.py
10. ├── flows/
11. ├── tests/
12. ├── infra/
13. ├── docs/
14. └── .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 no Prefect Cloud.

### 🚀 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 e no Prefect Cloud.

### ✅ 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 e Prefect Cloud.

### ✅ 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:
3. ./infra/run\_server.sh
4. Registrar flow localmente:
5. prefect deploy -n hello-deploy
6. Iniciar o worker:
7. ./infra/run\_worker.sh
8. Executar via Prefect UI: [http://localhost:4200](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:
5. @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