## Laboratório Calculadora Simples

**Projeto:** Uma calculadora modular e orientada a objetos com funcionalidades crescentes.

**Objetivo didático:** Mostrar como o uso do Prefect traz:

* Clareza no versionamento e evolução de código.
* Automação do processo de CI/CD (push → deploy).
* Observabilidade (logs, histórico, status, falhas).
* Controle granular com retries, agendamento, dependências.
* Profissionalismo no ciclo de entrega contínua.

**🚀 Estrutura do novo laboratório**

| **Fase** | **Funcionalidade** | **Justificativa técnica** |
| --- | --- | --- |
| V0 | Adição e subtração | MVP básico, execução local |
| V1 | + Multiplicação | Atualização simples + novo deploy |
| V2 | + Divisão | Introduz tratamento de exceções |
| V3 | + Porcentagem | Função que simula cálculo financeiro simples |
| V4 | CLI simples ou Web API (opcional) | Interface de uso real do sistema |
| VFinal | Observabilidade via Prefect | Histórico, versionamento, controle de execução |

**📁 Estrutura sugerida do projeto**

bash

CopiarEditar

calculator\_prefect/

├── calculator/

│ ├── \_\_init\_\_.py

│ ├── base.py # classe abstrata / interface

│ ├── operations.py # implementação OO das operações

├── flows/

│ └── run\_calculator.py # fluxo Prefect que orquestra chamadas

├── tests/

│ └── test\_operations.py

├── .github/workflows/

│ └── deploy.yml

├── prefect.yaml

├── requirements.txt

└── README.md

**🧪 Diferenciais didáticos**

* Mostra evolução real de um projeto de software.
* Permite aplicar CI/CD a um sistema **que cresce com as entregas**.
* Pode ser integrado com testes, cobertura de código e até Sentry.
* Dá espaço para introduzir conceitos como: modularização, boas práticas, automação e versionamento.

**📝 E no relatório?**

Posso incluir:

* ✅ Nova seção com objetivo e plano do novo laboratório.
* ✅ Descrição detalhada de cada etapa (do V0 ao VFinal).
* ✅ Explicação técnica sobre o valor do Prefect em cada evolução.
* ✅ Código comentado das versões incrementais.

Se estiver de acordo, posso:

1. Criar o esqueleto da estrutura do projeto.
2. Escrever o prefect.yaml inicial.
3. Gerar o deploy.yml do GitHub.
4. Adicionar a nova seção no relatório .docx como parte do conteúdo didático.

**✅ Confirmação da Estrutura Didática do Novo Laboratório**

**🎯 Dois eixos centrais:**

**1. Orquestração com Prefect — O "Por quê"**

Mostrar aos alunos que **orquestrar não é apenas automatizar**, mas sim:

* **Controlar, versionar, auditar e escalar** workflows.
* Garantir **resiliência com retries**, **condições de execução**, agendamentos e visualização clara do pipeline.
* Evitar que falhas silenciosas prejudiquem a produção (monitoramento e logs).
* Preparar o projeto para integração com ferramentas como S3, bancos de dados, serviços externos, ML pipelines, etc.

**2. Engenharia de Software Moderna — O "Como"**

Construir um projeto:

* Modular.
* Escalável.
* Orientado a objetos.
* Com documentação, testes e CI/CD.

A proposta da **calculadora evolutiva** (com adição → subtração → multiplicação → divisão → porcentagem) **é perfeita** para ilustrar isso em pequenas doses funcionais, com cada versão gerando um novo deployment automatizado.

**✍️ Ação Imediata**

Proponho criar um novo documento com o seguinte conteúdo:

**🧩 Nome: Calculadora Orquestrada - Projeto Didático com Prefect**

E nele vamos estruturar:

1. **Objetivo pedagógico**: detalhar esses dois eixos.
2. **Etapas do projeto**:
   * V0 (adição e subtração)
   * V1 (+ multiplicação)
   * V2 (+ divisão com tratamento de erro)
   * V3 (+ porcentagem, simulação financeira)
   * V4 (CLI simples)
3. **Como o Prefect entra em cada etapa**.
4. **Pipeline de automação GitHub Actions** e gatilho por push.
5. **Estrutura modular do código (orientada a objetos)**.
6. **Visão final do ganho de orquestrar e versionar projetos reais.**

# Laboratório Didático: Orquestração com Prefect aplicada ao Desenvolvimento de uma Calculadora Modular

**🎯 Objetivo**

Desenvolver um laboratório didático que evidencie a importância da orquestração de tarefas com Prefect, aliado às boas práticas de Engenharia de Software. O laboratório deverá mostrar o ciclo de vida completo de um projeto de software: desde a fase de projeto (fundamental e anterior à codificação), passando pelo desenvolvimento modular orientado a objetos, até a automação de deploys e execuções usando GitHub Actions e Prefect Cloud. A proposta também busca destacar a importância da reprodutibilidade, escalabilidade, organização de sistemas reais e a necessidade de um bom projeto da solução antes de iniciar a codificação.

**🧩 Justificativa**

Alunos e profissionais iniciantes geralmente subestimam a importância de ferramentas de orquestração, acreditando que apenas "rodar um script" seja suficiente. Este laboratório irá demonstrar, de maneira prática e visual, os reais ganhos de controle, rastreabilidade, automação e confiabilidade trazidos por uma ferramenta como o Prefect. Simultaneamente, a proposta reforça a importância do projeto da solução e da aplicação das melhores práticas de Engenharia de Software desde o início do desenvolvimento, evidenciando o papel das etapas de análise, modelagem e planejamento no sucesso da implementação.

**🏗️ Estrutura Didática (Etapas)**

**✅ Etapa 0 — Preparação do Ambiente**

* Instalação e configuração do ambiente virtual, dependências e Prefect.
* Criação do repositório no GitHub com estrutura inicial.
* Testes de conectividade com Prefect Cloud.

**🔸 Etapa 1 — Projeto da Solução e Estrutura Base**

* Definição do escopo inicial: Calculadora com operações básicas (adição e subtração).
* Uso de ferramentas de modelagem (Canvas, Lucidchart, draw.io).
* Estrutura modular e orientada a objetos.
* Criação da estrutura de diretórios e arquivos principais.
* Documentação do projeto com UML simples (classes e fluxo).

**🔸 Etapa 2 — Primeiro Deploy (Adição e Subtração)**

* Implementação inicial das operações básicas.
* Testes locais com Prefect Flow.
* Deploy da primeira versão com GitHub Actions.

**🔸 Etapa 3 — Ampliação Funcional (Multiplicação e Divisão)**

* Expansão incremental do projeto.
* Refatorações, modularização e tratamento de erros.
* Novo commit + push → novo deploy automatizado.

**🔸 Etapa 4 — Finalização de Funcionalidades (Porcentagem)**

* Adição da última funcionalidade planejada.
* Validação final do comportamento.
* Push final e execução orquestrada via Prefect Cloud.

**🔸 Etapa 5 — Integração com Prefect e Visualização**

* Conversão dos scripts em flows.
* Observação da execução e logs no painel Prefect.
* Avaliação dos ganhos e comparação com execução local/manual.

**🔍 Metodologia**

* **Projeto da solução antes do código**, com apoio de ferramentas de modelagem (Canvas, Lucidchart, draw.io).
* Aplicação de princípios de modularidade, encapsulamento e orientação a objetos.
* Uso de diagramas UML e documentação progressiva.
* Git e GitHub como base de versionamento e integração.
* Prefect para controle de execução e rastreabilidade.
* GitHub Actions para automação do deploy e execução.
* Avaliação visual e técnica dos resultados com e sem orquestração.
* Reflexão crítica ao fim de cada etapa sobre os ganhos obtidos.

**🧭 Resultado Esperado**

* Um projeto de software simples, mas tecnicamente bem estruturado, escalável e modular.
* Orquestração eficaz com Prefect, evidenciando benefícios de automação e rastreabilidade.
* CI/CD funcional com deploy automático a partir do GitHub.
* Relatório técnico final detalhando todo o processo, decisões e aprendizados.

**✅ Pontos-Chave**

* Prefect é mais do que um agendador: é uma ferramenta de orquestração e confiabilidade.
* Engenharia de Software começa com **projeto da solução**, não com código.
* Boas práticas geram código reusável, manutenível e escalável.
* CI/CD não é um luxo — é um diferencial competitivo real.

**🧠 Observação Didática**

Este laboratório poderá ser adaptado para diferentes níveis (graduação, pós) ajustando o grau de complexidade da calculadora e os requisitos de automação. Também poderá ser expandido com integração a banco de dados, autenticação ou visualização via Dash/Streamlit em etapas futuras.

## Estrutura Didática do Repositório (com explicações)

### ✅ Etapa 0 — Preparação do Ambiente e Documentação Inicial

🧭 Contexto

Estamos iniciando o projeto calculadora\_simples, cujo objetivo final é demonstrar a importância da orquestração com Prefect, aliada às boas práticas de Engenharia de Software.

O projeto será construído incrementalmente, com deploys progressivos após cada nova funcionalidade, simulando um ciclo real de evolução de sistemas.

### 🔹 Ambiente de Desenvolvimento

* 📁 Repositório: calculadora\_simples
* 💻 Ambiente virtual: DEV
* 🐍 Python: 3.12.5
* 📂 Diretório atual: /Users/accol/Library/Mobile Documents/com~apple~CloudDocs/UNIVERSIDADES/UFF/CARREIRA\_POS/PPGEET/ORQUESTRACAO/LABORATORIO/calculadora\_simples

### 📁 Estrutura Inicial do Projeto

Texto

O conteúdo gerado por IA pode estar incorreto.

**✅** Ações obrigatórias nesta etapa

#### 1. Criar o arquivo docs/readme.txt

Este arquivo conterá a **especificação da versão inicial da calculadora**, com foco apenas em **adição e subtração**. Deve conter:

* Nome do sistema
* Objetivo
* Funcionalidades mínimas
* Público-alvo
* Forma de uso esperada

#### 2. Gerar requirements.txt

Com os pacotes usados no projeto e versões fixas. Incluir:

prefect==2.16.4

griffe==0.35.2

Outros pacotes serão adicionados conforme a evolução.

#### 3. Configurar infra/prefect.yaml

Este arquivo será responsável por indicar como o Prefect deve executar os flows. Terá a configuração git\_clone para repositório GitHub.

#### 4. Criar .gitignore e .prefectignore

Para evitar que arquivos locais/ocultos sejam versionados:

Texto

O conteúdo gerado por IA pode estar incorreto.

**5. Visibilidade no VSCode (macOS)**

Para visualizar arquivos ocultos como .gitignore e .prefectignore, usar atalho:



Só vale no terminal externo – fora do VsCode ok.