PUC Minas
Virtual

Controle de Versão e CI

Renan Santos Mendes

PUC Minas
Virtual

Resumindo

Resumo

- **Controle de Versão:** é a prática de gerenciar código por meio de versões acompanhando as revisões e o histórico de alterações para facilitar a revisão e a recuperação do código.

- **Integração contínua:** é uma prática de desenvolvimento de software em que os desenvolvedores, com frequência, juntam suas alterações de código em um repositório central.

PUC Minas
Virtual

Git e GitHub

Instalação Git

Para continuarmos façam as seguintes instalações:

- Conheça e instale o Git - https://git-scm.com/
- Conheça e crie uma conta no GitHub - https://github.com/about

O que é Git?

Link para download

Imagem via git-scm.com

Características do Git

Sobre o Git:

- **Branching and Merging**;
- **Pequeno e rápido**;
- Distribuído;
- **Staging Area**;
- **Gratuito** e Aberto.

Características do GitHub

Sobre o GitHub:

- **Colaborativo;**
- **Automações e CI/CD;**
- Segurança;
- Gerenciamento de Projetos;
- Administração de Times;
- **Comunidade;**

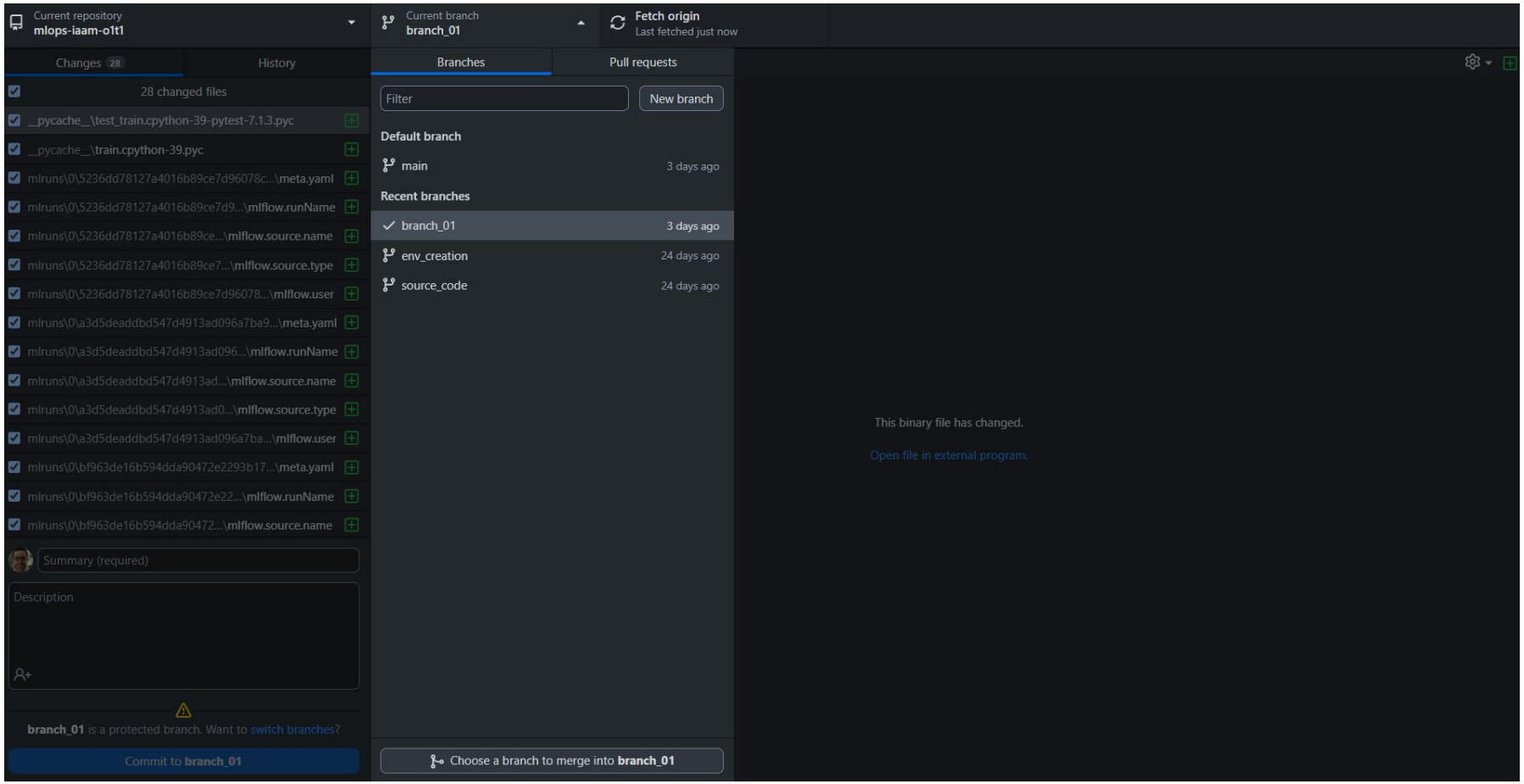
PUC Minas
Virtual

GitHub Desktop

GitHub Desktop

- Link para download;
- Interface **intuitiva;**
- Controle de versão **simplificado;**
- **Colaboração** eficiente;
- Gerenciamento de **conflitos**;
- Histórico e visualização de alterações;
- Disponibilidade **multiplataforma**.

GitHub Desktop



PUC Minas
Virtual

Funcionamento do Git

Funcionamento do Git

- Um esquema básico de funcionamento do Git

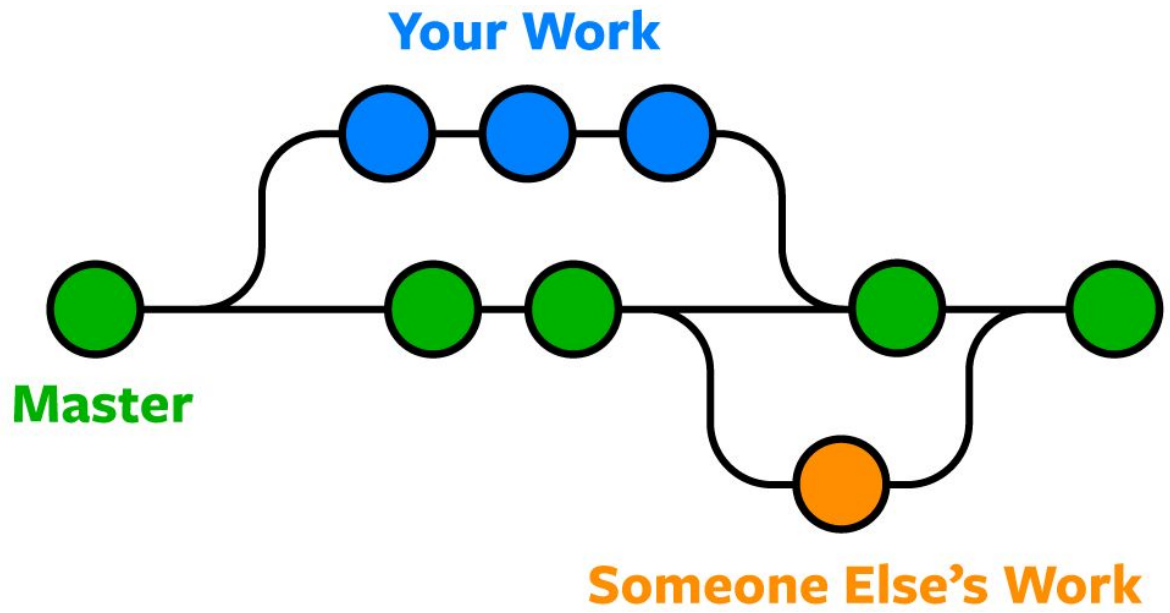


Imagem via git-scm.com

PUC Minas
Virtual

Adicionando alterações no repositório

Estrutura do Git

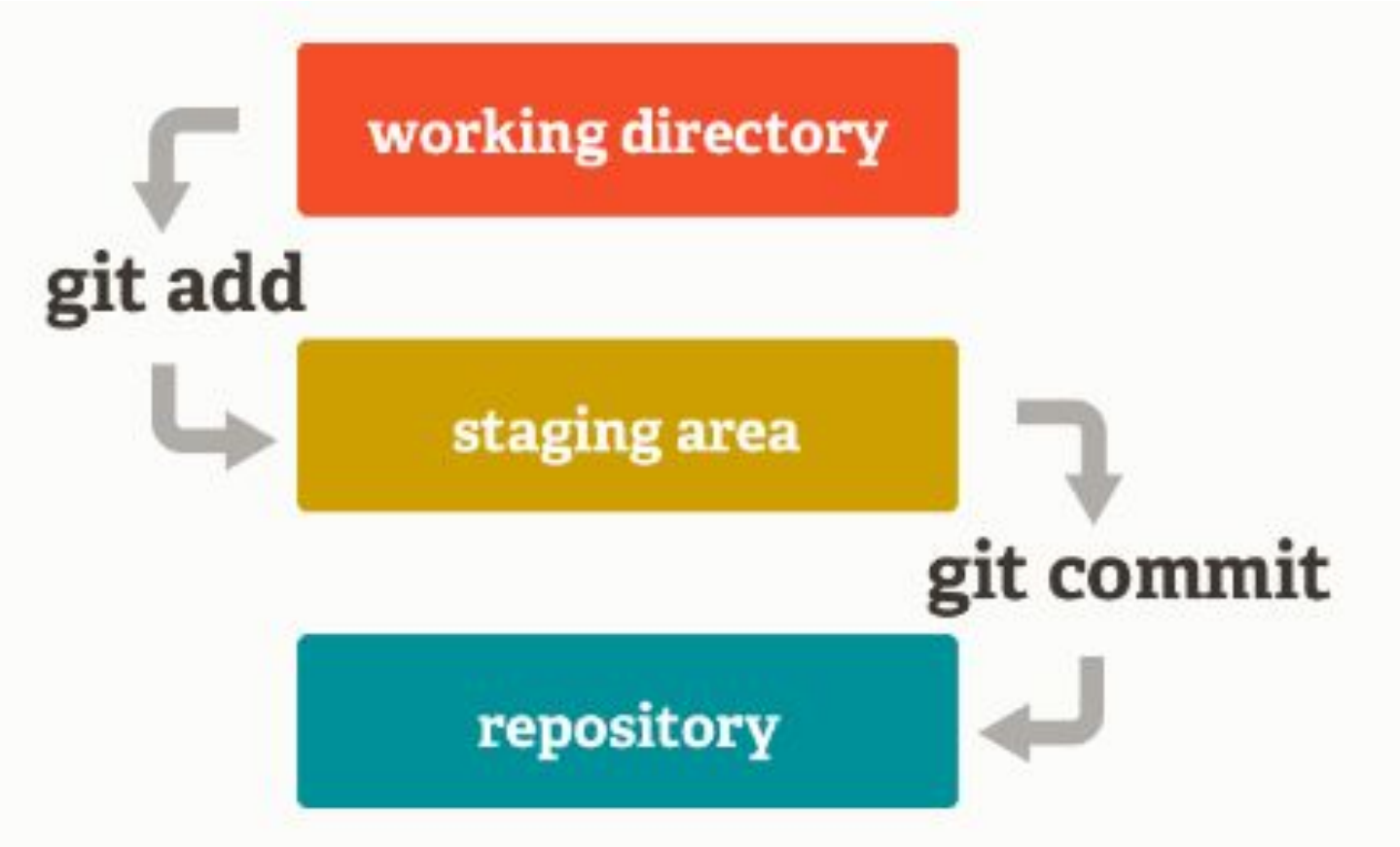


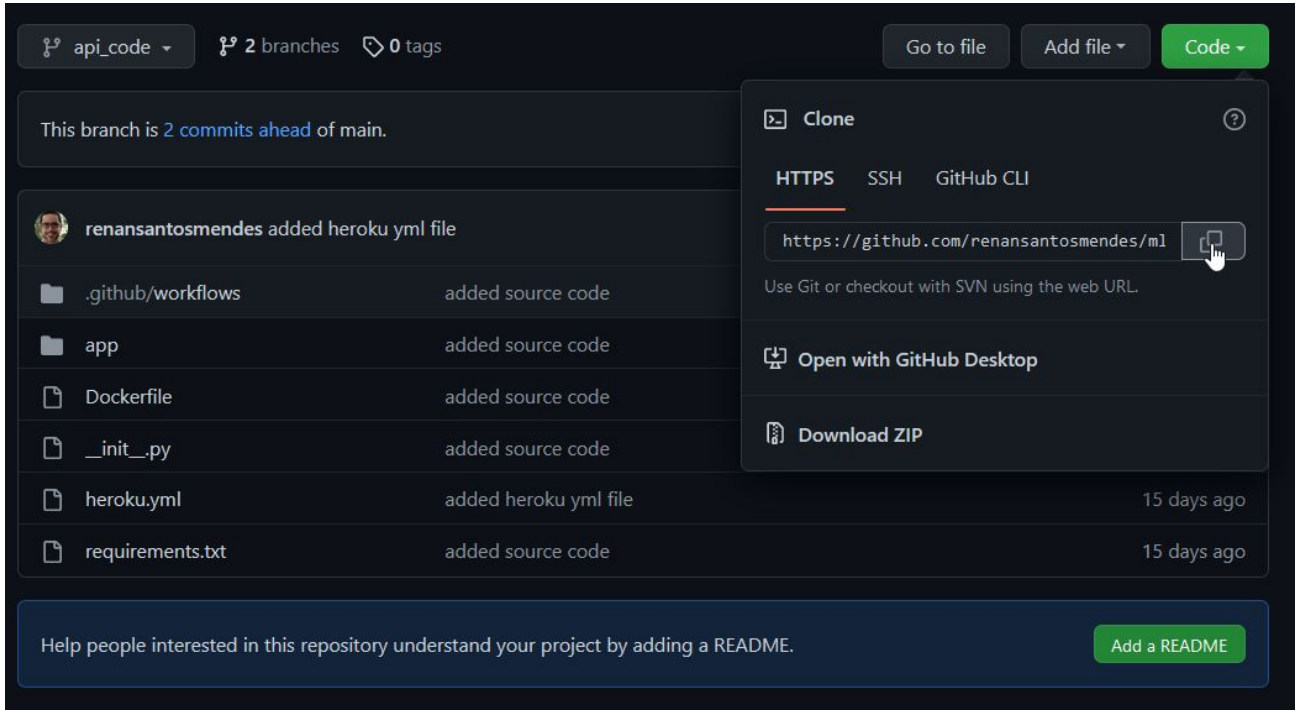
Imagem via git-scm.com

PUC Minas
Virtual

Alguns comandos no Git

git clone

git clone <url_do_repositorio>




```
PS C:\PUC\03 - MLOps\Aula 01> git clone https://github.com/renansantosmendes/mlopspuc.git
Cloning into 'mlopspuc'...
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 17 (delta 1), reused 12 (delta 1), pack-reused 0
Unpacking objects: 100% (17/17), 2.42 KiB | 70.00 KiB/s, done.
```

git checkout

git checkout -b <nome_do_ramo>

```
PS C:\01 - Estudo\PUC\MLOps\Aula01\puc_mlops> git checkout -b env_creation
Switched to a new branch 'env_creation'
PS C:\01 - Estudo\PUC\MLOps\Aula01\puc_mlops>
```

Adicionar alterações - git add + git commit

- git **status**
- git **add**
- git **commit** -m <mensagem>
- git **push**

```
PS C:\01 - Estudo\PUC\MLOps\Aula02\puc_mlops> git status
On branch env_creation

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        environment.yml
        requirements.txt

nothing added to commit but untracked files present (use "git add" to track)
PS C:\01 - Estudo\PUC\MLOps\Aula02\puc_mlops> git add .\environment.yml .\requirements.txt
PS C:\01 - Estudo\PUC\MLOps\Aula02\puc_mlops> git status
On branch env_creation

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   environment.yml
        new file:   requirements.txt
```

```
PS C:\01 - Estudo\PUC\MLOps\Aula02\puc_mlops> git commit -m 'added enviroment files'
[env_creation (root-commit) 3ecf0c0] added enviroment files
 2 files changed, 11 insertions(+)
 create mode 100644 environment.yml
 create mode 100644 requirements.txt
PS C:\01 - Estudo\PUC\MLOps\Aula02\puc_mlops> git push
fatal: The current branch env_creation has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin env_creation

PS C:\01 - Estudo\PUC\MLOps\Aula02\puc_mlops> git push --set-upstream origin env_creation
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 424 bytes | 424.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/renansantosmendes/puc_mlops.git
 * [new branch]      env_creation -> env_creation
Branch 'env_creation' set up to track remote branch 'env_creation' from 'origin'.
PS C:\01 - Estudo\PUC\MLOps\Aula02\puc_mlops>
```

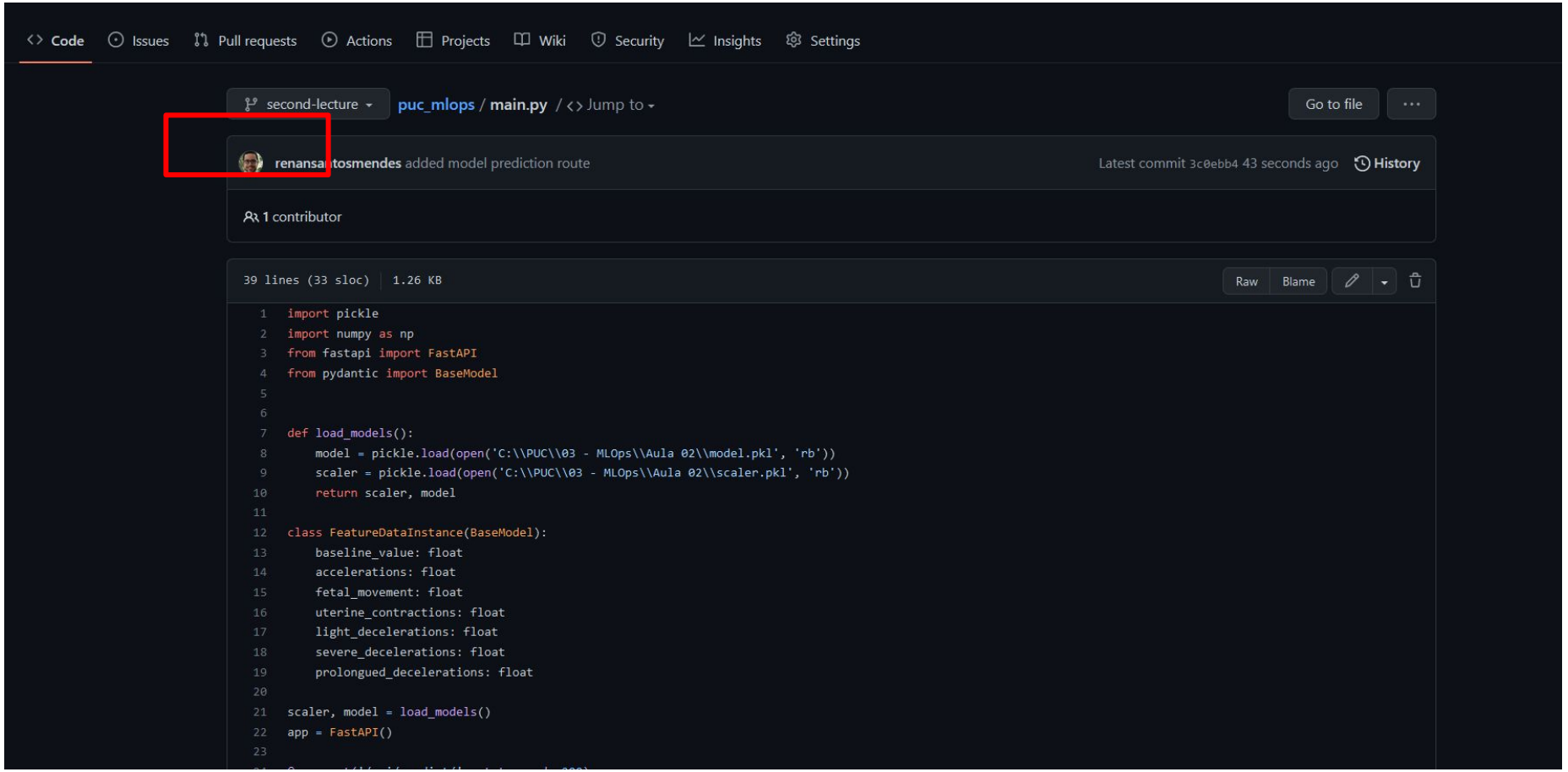
PUC Minas
Virtual

Abrindo um PR ou MR

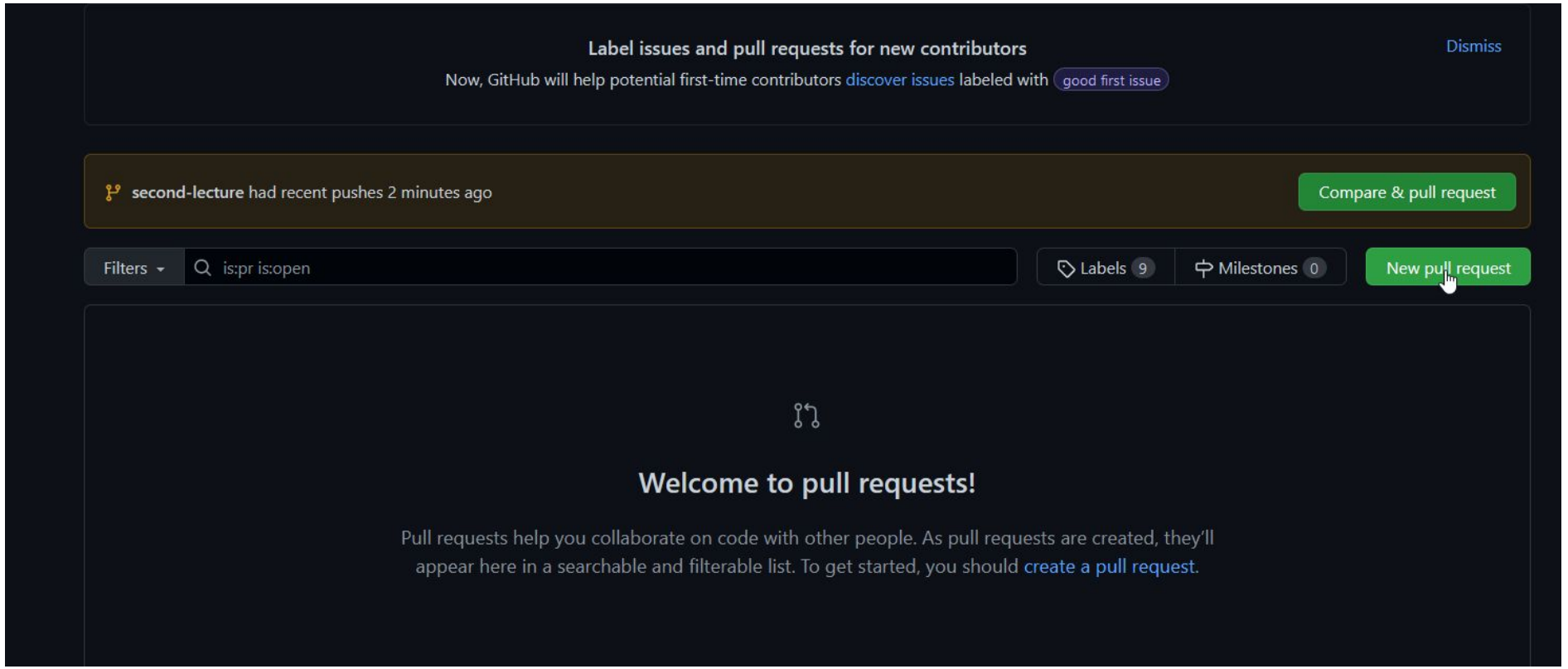
Abrindo um *Pull Request*

- O processo de enviar o **código de um ramo para o código principal** é denominado **Pull Request**, ou conhecido como **PR;**
- No Git pode ser feito de duas formas:
  - Comando git request-pull;
  - Pela interface;

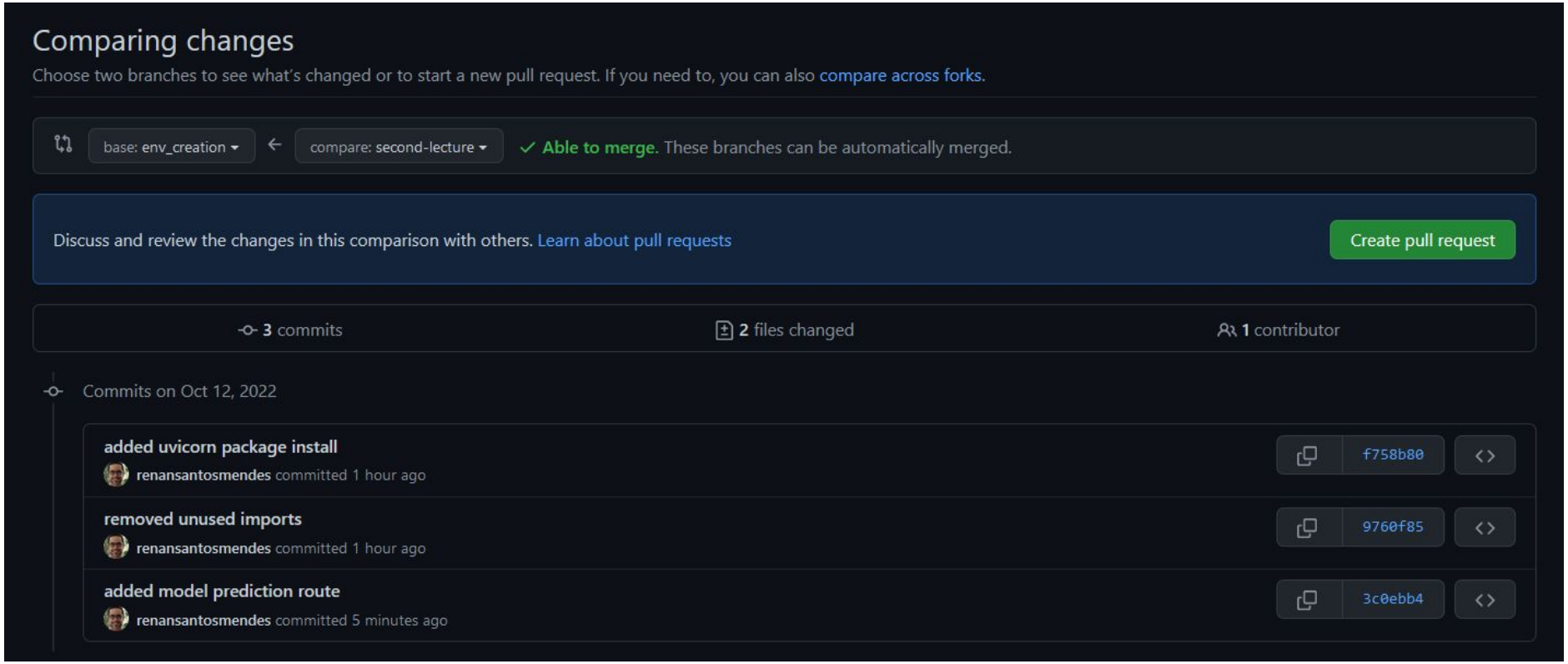
Abrindo um *Pull Request*



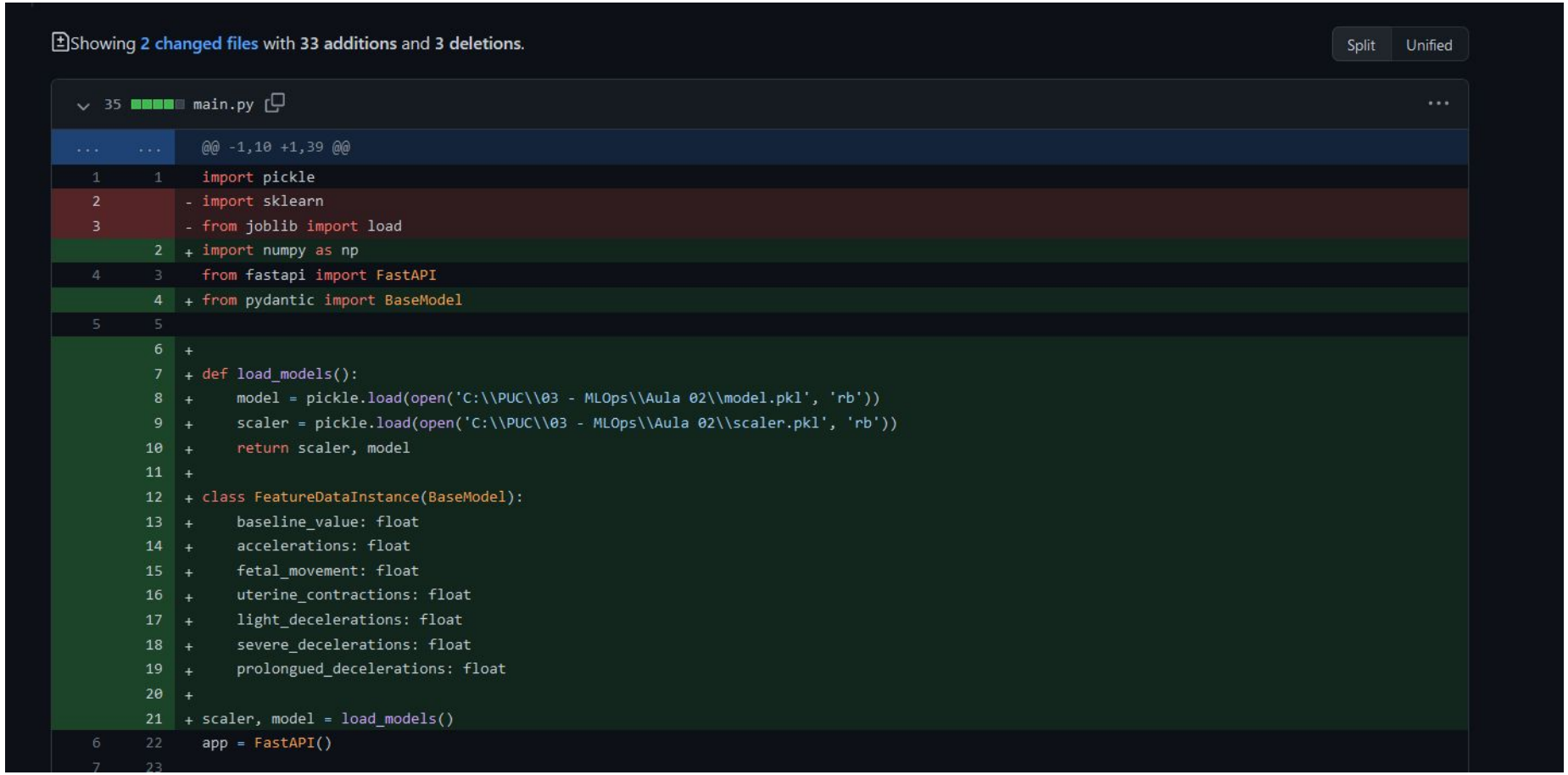
Abrindo um *Pull Request*



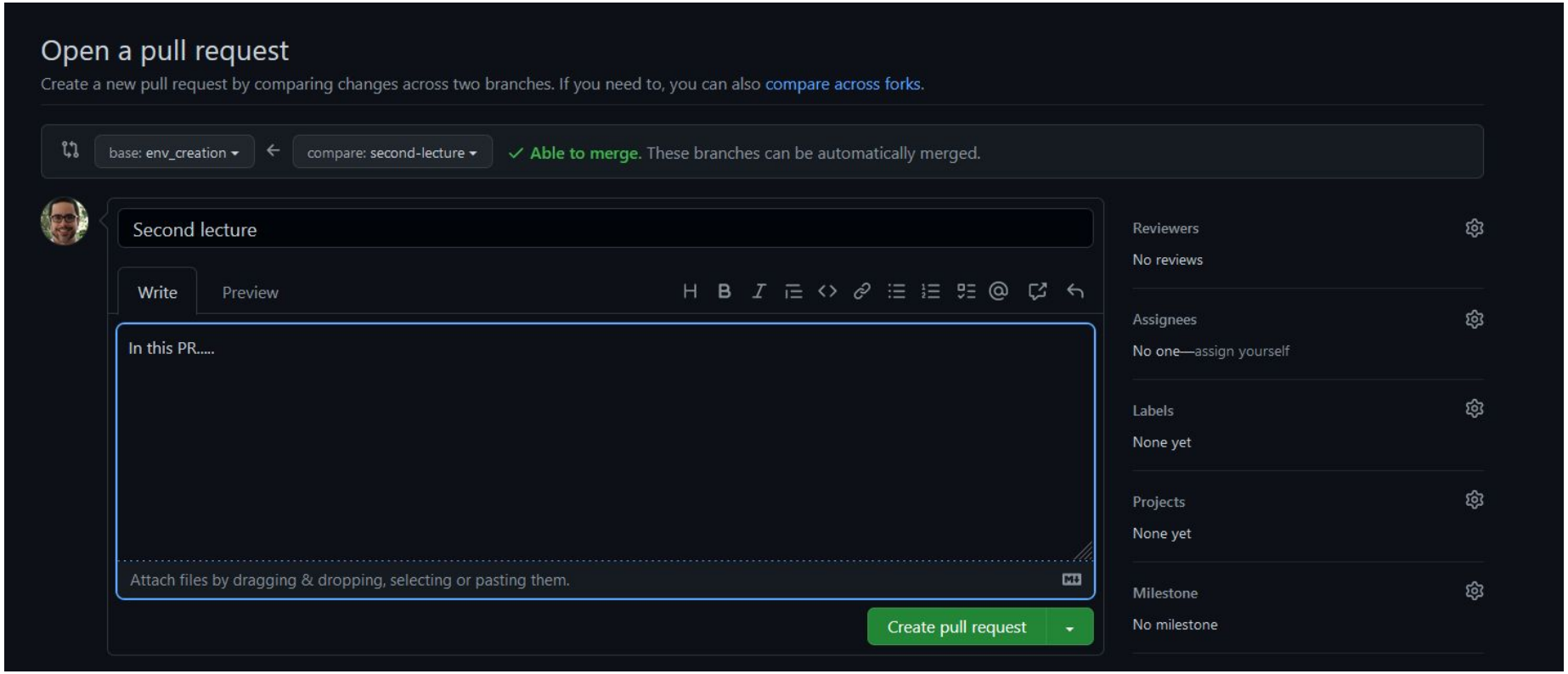
Abrindo um *Pull Request*



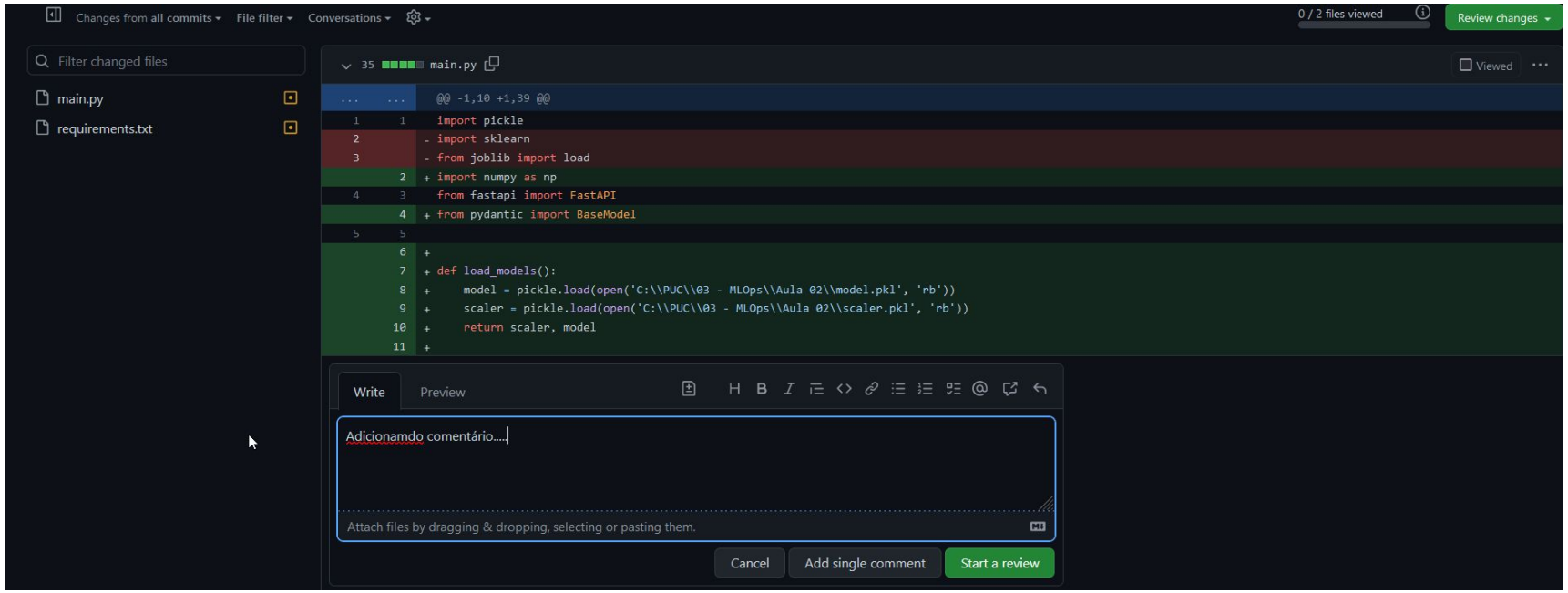
Abrindo um *Pull Request*



Abrindo um *Pull Request*



Abrindo um *Pull Request*



PUC Minas
Virtual

PUC Minas
Virtual

Ambientes de Desenvolvimento

Renan Santos Mendes

PUC Minas
Virtual

Anaconda e Python

Python e Anaconda

- **Python** é uma das linguagens de programação mais populares do mundo
  - Análise e Ciência de Dados
  - Inteligência Artificial e Aprendizado de Máquina
  - Desenvolvimento Web & Server

Python e Anaconda

- **Anaconda** é uma distribuição de Python para programação científica que facilita o gerenciamento de pacotes e a implantação em múltiplos sistemas operacionais.
  - Python
  - Conda https://www.anaconda.com/
- **PyCharm**: Ferramenta para desenvolvimento de código;

PUC Minas
Virtual

Arquivos de ambiente

Arquivos de ambiente

- Criação de ambientes permite o uso de diferentes versões de bibliotecas;
- A troca de ambientes pode ser feita de forma **rápida** e **simples;**
- Ambiente virtual = ambiente isolado;
- Existem diversas formas de criar os ambientes:
  - venv
  - pyenv
  - pipenv
  - **conda env**

Arquivos de ambiente

```yaml
name: mlops-env
dependencies:
  - python=3.9
  - pip
  - pip:
    - -r requirements.txt
```

```
fastapi==0.85.0
mlflow==1.29.0
scikit-learn==1.1.2
pandas==1.5.0
matplotlib==3.6.1
```

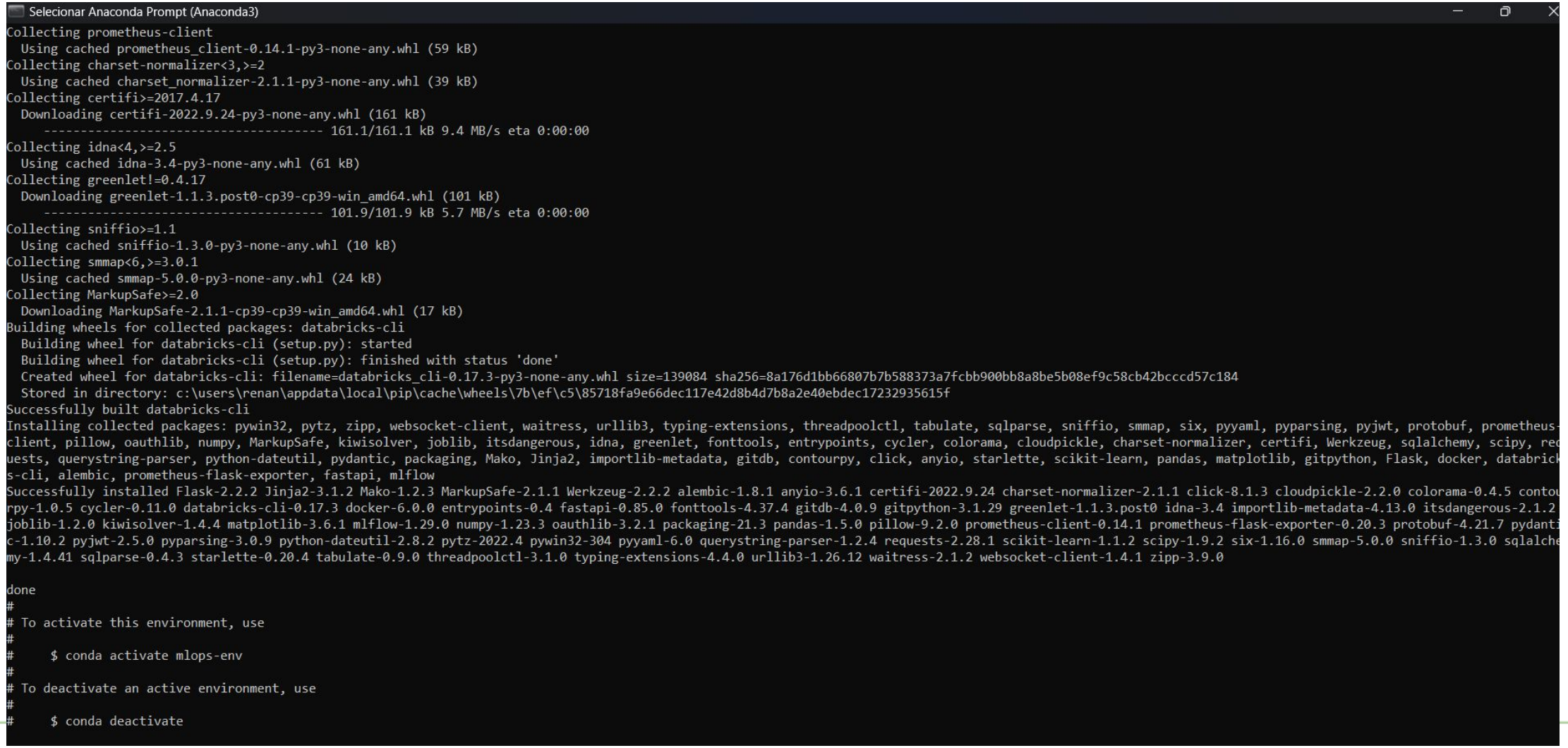
Arquivos de ambiente

Comandos para a criação e ativação de um **ambiente**:

- conda env create -f environment.yml
- conda activate mlops-env
- conda deactivate

Comandos para a criação de um **ambiente vazio**:

- conda create --name myenv python=3.9
- conda activate myenv
- Ex.: pip install "tensorflow<2.11"


Selecionar Anaconda Prompt (Anaconda3)
Collecting prometheus-client
Using cached prometheus_client-0.14.1-py3-none-any.whl (59 kB)
Collecting charset-normalizer<3,>=2
Using cached charset_normalizer-2.1.1-py3-none-any.whl (39 kB)
Collecting certifi>=2017.4.17
Downloading certifi-2022.9.24-py3-none-any.whl (161 kB)
161.1/161.1 kB 9.4 MB/s eta 0:00:00
Collecting idna<4,>=2.5
Using cached idna-3.4-py3-none-any.whl (61 kB)
Collecting greenlet!=0.4.17
Downloading greenlet-1.1.3.post0-cp39-cp39-win_amd64.whl (101 kB)
101.9/101.9 kB 5.7 MB/s eta 0:00:00
Collecting sniffio>=1.1
Using cached sniffio-1.3.0-py3-none-any.whl (10 kB)
Collecting smmap<6,>=3.0.1
Using cached smmap-5.0.0-py3-none-any.whl (24 kB)
Collecting MarkupSafe>=2.0
Downloading MarkupSafe-2.1.1-cp39-cp39-win_amd64.whl (17 kB)
Building wheels for collected packages: databricks-cli
Building wheel for databricks-cli (setup.py): started
Building wheel for databricks-cli (setup.py): finished with status 'done'
Created wheel for databricks-cli: filename=databricks_cli-0.17.3-py3-none-any.whl size=139084 sha256=8a176d1bb66807b7b588373a7fcbb900bb8a8be5b08ef9c58cb42bcccd57c184
Stored in directory: c:\users\renan\appdata\local\pip\cache\wheels\7b\ef\c5\85718fa9e66dec117e42d8b4d7b8a2e40ebdec17232935615f
Successfully built databricks-cli
Installing collected packages: pywin32, pytz, zipp, websocket-client, waitress, urllib3, typing-extensions, threadpoolctl, tabulate, sqlparse, sniffio, smmap, six, pyyaml, pyparsing, pyjwt, protobuf, prometheus-client, pillow, oauthlib, numpy, MarkupSafe, kiwisolver, joblib, itsdangerous, idna, greenlet, fonttools, entrypoints, cycler, colorama, cloudpickle, charset-normalizer, certifi, Werkzeug, sqlalchemy, scipy, requests, querystring-parser, python-dateutil, pydantic, packaging, Mako, Jinja2, importlib-metadata, gitdb, contourpy, click, anyio, starlette, scikit-learn, pandas, matplotlib, gitpython, Flask, docker, databricks-cli, alembic, prometheus-flask-exporter, fastapi, mlflow
Successfully installed Flask-2.2.2 Jinja2-3.1.2 Mako-1.2.3 MarkupSafe-2.1.1 Werkzeug-2.2.2 alembic-1.8.1 anyio-3.6.1 certifi-2022.9.24 charset-normalizer-2.1.1 click-8.1.3 cloudpickle-2.2.0 colorama-0.4.5 contourpy-1.0.5 cycler-0.11.0 databricks-cli-0.17.3 docker-6.0.0 entrypoints-0.4 fastapi-0.85.0 fonttools-4.37.4 gitdb-4.0.9 gitpython-3.1.29 greenlet-1.1.3.post0 idna-3.4 importlib-metadata-4.13.0 itsdangerous-2.1.2 joblib-1.2.0 kiwisolver-1.4.4 matplotlib-3.6.1 mlflow-1.29.0 numpy-1.23.3 oauthlib-3.2.1 packaging-21.3 pandas-1.5.0 pillow-9.2.0 prometheus-client-0.14.1 prometheus-flask-exporter-0.20.3 protobuf-4.21.7 pydantic-1.10.2 pyjwt-2.5.0 pyparsing-3.0.9 python-dateutil-2.8.2 pytz-2022.4 pywin32-304 pyyaml-6.0 querystring-parser-1.2.4 requests-2.28.1 scikit-learn-1.1.2 scipy-1.9.2 six-1.16.0 smmap-5.0.0 sniffio-1.3.0 sqlalchemy-1.4.41 sqlparse-0.4.3 starlette-0.20.4 tabulate-0.9.0 threadpoolctl-3.1.0 typing-extensions-4.4.0 urllib3-1.26.12 waitress-2.1.2 websocket-client-1.4.1 zipp-3.9.0
done
#
To activate this environment, use
#
$ conda activate mlops-env
#
To deactivate an active environment, use
#
$ conda deactivate

Arquivos de ambiente

```
(base) C:\01 - Estudo\PUC\MLOps\Aula02\puc_mlops>conda activate mlops-env

(mlops-env) C:\01 - Estudo\PUC\MLOps\Aula02\puc_mlops> conda deactivate

(base) C:\01 - Estudo\PUC\MLOps\Aula02\puc_mlops>
```

PUC Minas
Virtual

PUC Minas
Virtual

Pipelines para Treino

Renan Santos Mendes

PUC Minas
Virtual

Pipelines

Pipelines

- Uma pipeline é uma **sequência automatizada de processos** no desenvolvimento de software.
- A pipeline é composta por várias etapas;
- A pipeline **automatiza** e **facilita** o desenvolvimento e a entrega;
- Ela é altamente **personalizável** e pode ser adaptada às necessidades específicas de uma equipe ou projeto;
- As pipelines ajudam a garantir a **qualidade**, detectar e **corrigir problemas** e melhorar a **colaboração**.

PUC Minas
Virtual

GitHub Actions

GitHub Actions

- Uma pipeline no GitHub Actions é um **fluxo automatizado de etapas** (ou jobs)
- São executadas em resposta a eventos específicos no seu repositório do GitHub.
- Uma pipeline é definida em um arquivo chamado **workflow**, armazenado no diretório **.github/workflows** do seu repositório.

GitHub Actions

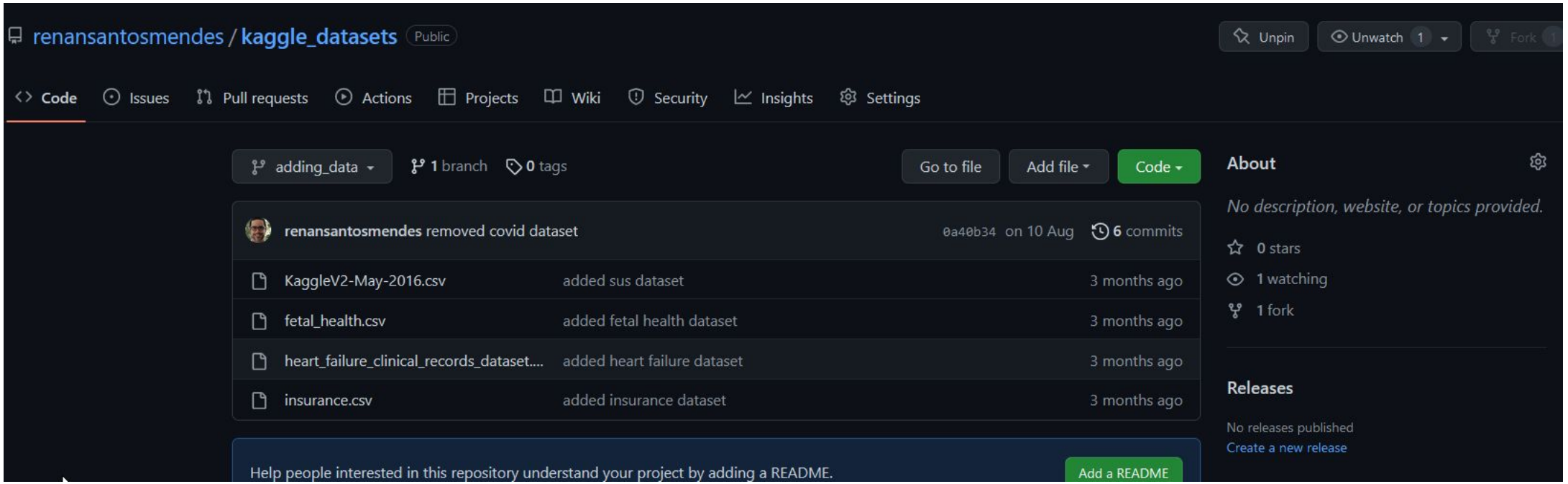
- O arquivo workflow contém instruções para o GitHub Actions executar as etapas desejadas em resposta a eventos como push de código ou abertura de pull request.
- A pipeline é composta por **um ou mais jobs**, que podem ser executados em **paralelo ou em sequência**, dependendo da configuração.

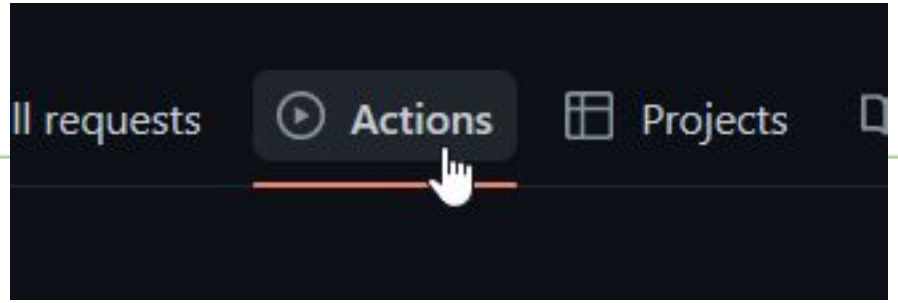
GitHub Actions

Vantagens

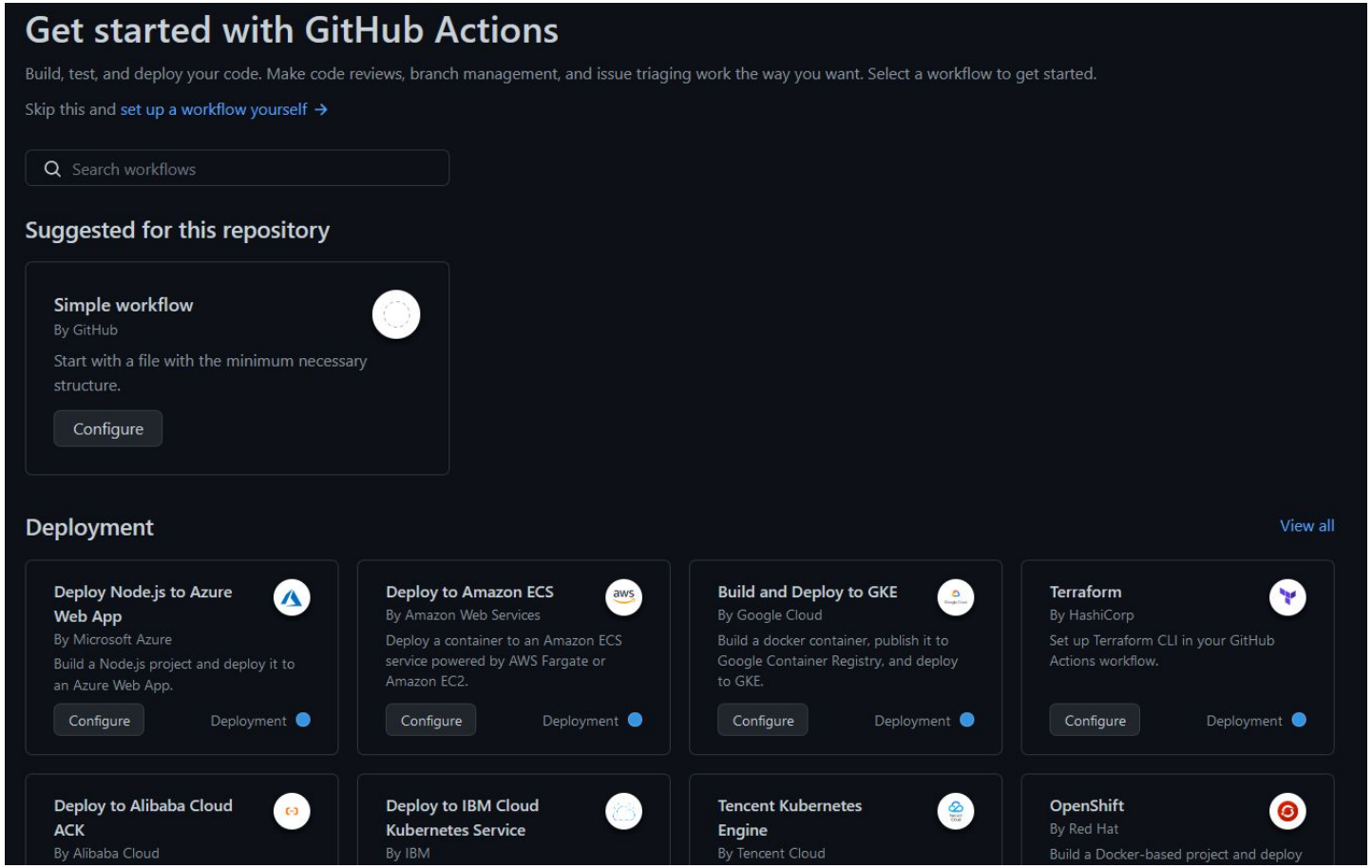
- Simples
- Integrado ao GitHub
- Funciona para CI & CD
- Flexível
- “Marketplace” de elementos prontos
- https://docs.github.com/en/actions

GitHub Actions

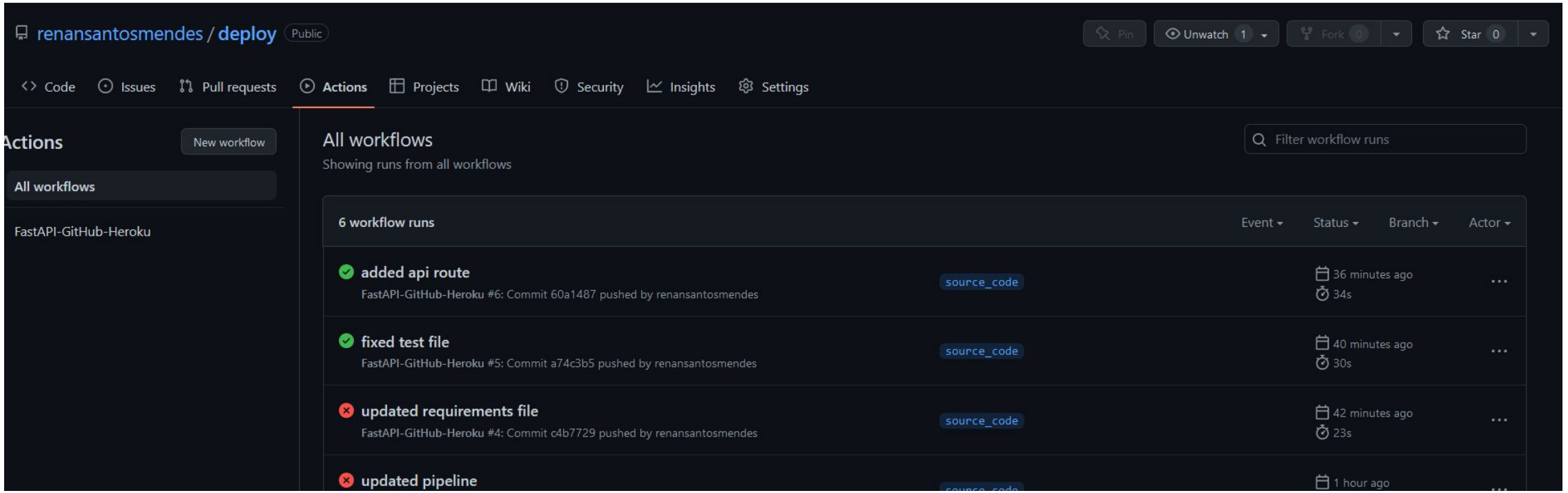




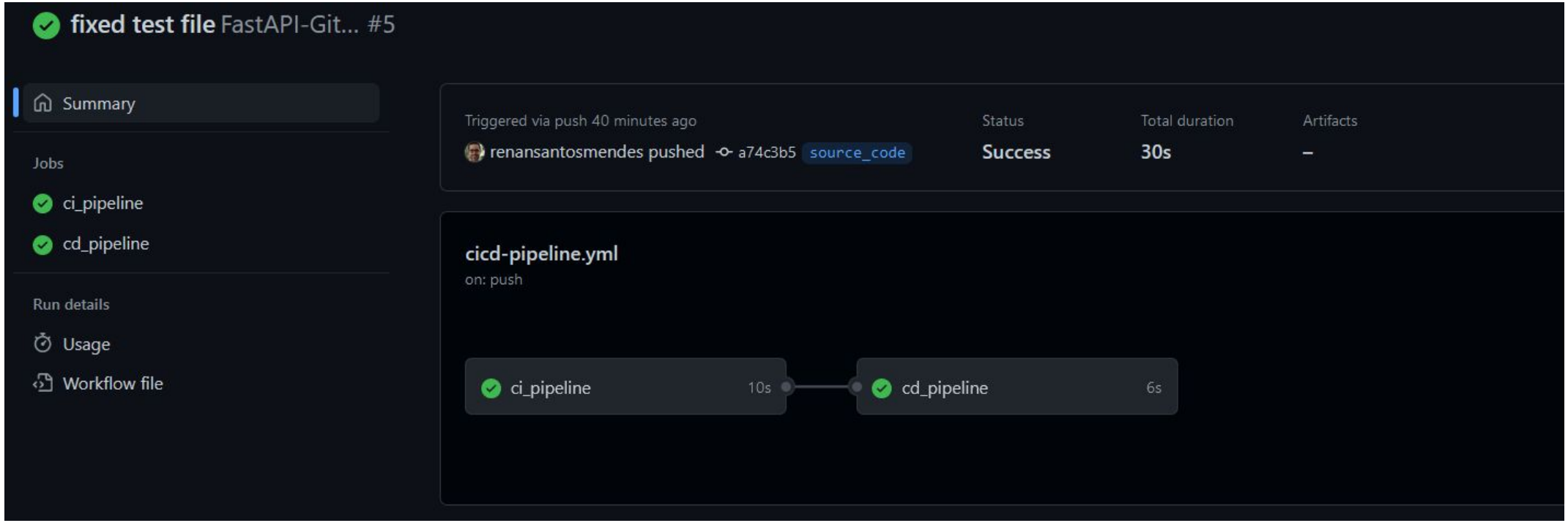
GitHub Actions



GitHub Actions



GitHub Actions



PUC Minas
Virtual

Criando uma pipeline manualmente

GitHub Actions

Para se criar um arquivo de workflow, um sequência de passos deve ser feita:

- Criar um arquivo .yml no subdiretório workflows dentro de uma pasta .github;
  - Ficaria da seguinte forma: Projeto/.github/workflows/file.yml
- Dentro do arquivo .yml devem ser criados pelos menos os seguintes passos:
  - Configuração inicial
  - Jobs
  - Steps dentro dos jobs

GitHub Actions

```yaml
name: FastAPI-GitHub-Heroku

env:
     HEROKU_API_TOKEN: ${{ secrets.HEROKU_API_TOKEN }}
     HEROKU_APP_NAME: ${{ secrets.HEROKU_APP_NAME }}

on:
  push:
    branches:
    - main
    - source_code
  pull_request:
    branches:
    - main

jobs:
  ci_pipeline:
     runs-on: ubuntu-latest
     steps:
         - uses: actions/checkout@v1
           with:
             fetch-depth: 0

         - name: Set up Python 3.9
           uses: actions/setup-python@v1
           with:
              python-version: 3.9
```

PUC Minas
Virtual