⚙️ Aula Prática — Integração Contínua (CI) com GitHub Actions

Duração: 50–60 minutos
Pré-requisito: Aula anterior — uso básico de Git e GitHub.
Objetivo:

- Compreender o conceito de CI/CD.
- Criar um fluxo de integração contínua no GitHub Actions.
- Executar testes automatizados a cada push.

🧩 1. Introdução

📘 Conceitos Fundamentais

- **CI (Continuous Integration):** integrar código frequentemente e testar a cada push.
- **CD (Continuous Delivery/Deployment):** automatizar a entrega ou publicação do software.
- **GitHub Actions:** serviço do GitHub para automação via "workflows" (arquivos YAML).

Fluxo básico:

1. Desenvolvedor faz push
2. GitHub Actions detecta o evento
3. Executa o pipeline (ex: build, testes, deploy)

🧠 Perguntas rápidas

1. O que é CI/CD e por que é importante?
   👂*Resposta: CI: Integração contínua do código, CD: Entrega contínua do código. Pode-se afirmar que sua importância reside em uma automação e agilidade no processo de desenvolvimento de software, permitindo entregas mais rápidas, frequentes e confiáveis, além da redução de erros com a automação de testes.*
2. Em qual pasta os workflows do GitHub ficam armazenados?
   👂*Resposta: Os workflows ficam armazenados no diretório criado ".github"*

💻 2. Mão na Massa: Criando um Workflow

🧩 Etapa 1 – Criar repositório de prática

1. No GitHub, crie um repositório chamado **ci-cd-teste**.
2. Clone o repositório e crie um arquivo Python simples:

git clone https://github.com/<usuario>/ci-cd-teste.git

```
cd ci-cd-teste
echo "print('Hello CI/CD!')" > main.py
git add .
git commit -m "Arquivo inicial"
git push origin main
```

🧩 Etapa 2 – Criar workflow do GitHub Actions

Crie o diretório e o arquivo de workflow:

```
mkdir -p .github/workflows
nano .github/workflows/teste.yml
```

Cole o conteúdo abaixo:

```
name: Teste CI

on:
  push:
    branches: [ "main" ]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout do código
        uses: actions/checkout@v4

      - name: Configurar Python
        uses: actions/setup-python@v5
        with:
          python-version: "3.x"

      - name: Executar script
        run: python main.py
```

Faça commit e envie:

git add .github/workflows/teste.yml

git commit -m "Adiciona workflow de teste"

git push origin main

🧩 Etapa 3 – Verificando o pipeline

No GitHub:

- Vá em **Actions → Teste CI**
- Veja o workflow em execução após o push

Perguntas:

1. O que aparece no log do GitHub Actions após a execução?
   💡*Resposta: O log mostra, em tempo real, o que aconteceu em cada etapa do workflow.*
2. O que acontece se alterar o código e fizer novo push?
   💡*Resposta: o GitHub cria uma nova instância do workflow e começa a executartodo o processo novamente, e o log do novo workflow é atualizado.*

🧩 Etapa 4 – Introduzindo um teste automatizado (extra)

Crie um arquivo test_sample.py:

```
def test_soma():
  assert 1 + 1 == 2
```

Atualize o workflow para rodar testes com pytest:

```
- name: Instalar dependências
  run: pip install pytest


- name: Executar testes
  run: pytest -v
```

Faça commit e push novamente.
O GitHub Actions agora executará **testes automatizados** em cada push.

Pergunta extra:
O que acontece se um teste falhar?
💡*Resposta: Ao invés de aparecer um ícone verde de sucesso, aparece um ícone vermelho com X, indicando que o teste falhou*

📋 3. Entrega da Atividade

Tarefas obrigatórias:

1. Criar o workflow que executa main.py.
2. Fazer commit e push.
3. Mostrar o print da execução bem-sucedida no GitHub Actions.

Tarefa bônus:

- Adicionar mais testes automatizados com pytest.

🧠 4. Para finalizar

- Como o GitHub Actions ajuda a detectar erros cedo?

  O GitHub Actions executa testes automáticos toda vez que há uma mudança no código, por exemplo, quando acontece um push. Se algo falhar, ele mostra no log do workflow, e antes de fazer deploy, haverá a informação de que houve falha.

- Quais seriam exemplos reais de CI/CD em projetos web ou mobile?

  Em projetos web ou mobile, o Actions compila o app e toda testes automatizados, se tudo der certo, pode fazer upload automático em uma plataforma de hospedagem.

- Como o *deploy automático* poderia ser feito a partir deste pipeline?

  Poderia ser feito o deploy automático se todos os teste forem bem sucedidos. No contexto do GitHub Actions, isso ocorre de forma controlada, repetível e rastreável, tornando o processo de entrega de software mais ágil, seguro e confiável.

Print do Workflow - GitHub Actions