🚀 Aula Prática — Entrega Contínua (CD) com GitHub Actions

Duração: 50–60 minutos
Pré-requisito: Ter concluído a atividade de Integração Contínua (CI).
Objetivo:

- Entender o conceito de **Entrega Contínua (CD)**.
- Automatizar a **implantação (deploy)** de uma aplicação simples após testes bem-sucedidos.
- Publicar o resultado automaticamente no GitHub Pages.

🧩 1. Introdução

📘 Conceitos Fundamentais

- **CI (Continuous Integration):** Testa e valida o código a cada push.
- **CD (Continuous Delivery/Deployment):** Automatiza a entrega ou publicação após os testes passarem.
- **GitHub Actions:** Serviço do GitHub que permite criar *pipelines* de automação em arquivos YAML.

Fluxo básico:

1. Desenvolvedor faz push
2. CI executa build e testes
3. Se os testes passarem, CD realiza o deploy automaticamente

---

🧠 Perguntas rápidas

1. O que é CD e qual sua relação com CI?
   💡 *Resposta: CD significa entrega contínua. Sua relação com CI (entrega contínua) é de dependência, pois sem sucesso nos testes requisitados no código, a entrega não é realizada.*
2. Quais são os benefícios da entrega contínua?
   💡 *Resposta: Ciclos de lançamento mais rápidos, qualidade de código aprimorada com testes automatizados, redução de custos com a dominuição da necessidade de trabalho manual, maior colaboração entre equipes de diferentes áreas.*

💻 2. Mão na Massa: Criando o Workflow de CD

🧩 Etapa 1 – Reaproveitar o repositório anterior

Use o mesmo repositório da atividade de CI: **ci-cd-teste**.

Crie um arquivo simples de página web:

```
echo "<h1>Aplicação implantada via GitHub Actions</h1>" > index.html
git add index.html
git commit -m "Adiciona página inicial"
git push origin main
```

🧩 Etapa 2 – Criar o workflow de deploy

Crie um novo arquivo de workflow:

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

Cole o conteúdo a seguir:

```
name: Deploy Automático

on:
  push:
    branches: [ "main" ]

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout do código
        uses: actions/checkout@v4

      - name: Instalar dependências (opcional)
        run: |
          echo "Nenhuma dependência necessária para HTML puro"

      - name: Publicar no GitHub Pages
        uses: peaceiris/actions-gh-pages@v4
```

with:

github_token: ${{ secrets.GITHUB_TOKEN }}

publish_dir: ./

Para autorizar o GitHub Actions a realizar o *push* da branch gh-pages, no repositório **ci-cd-teste** no github:

1. Vá em **Settings → Actions → General**.
2. Role até **Workflow permissions**.
3. Marque a opção **"Read and write permissions"**.
4. Clique em **Save**.

Faça o commit:

git add .github/workflows/deploy.yml

git commit -m "Adiciona workflow de deploy automático"

git push origin main

🧩 Etapa 3 – Permissões e Configurações do GitHub Pages

1. Vá em **Settings → Pages**
2. Em "Source", selecione a branch gh-pages
3. Após o workflow ser executado, acesse:

https://<seu-usuário>.github.io/ci-cd-teste/

Você verá sua página publicada automaticamente 🚀

4. Atualize a página index.html:
5. echo " Deploy automático conforme .github/workflows/deploy.yml " >> index.html
6. git add index.html
7. git commit -m "Atualiza página inicial"

git push origin main

Veja se a sua página foi atualizada e publicada automaticamente no github pages🚀

🧩 Etapa 4 – Tornando o deploy condicional (opcional)

Atualize o workflow para só fazer deploy se os testes passarem:

```yaml
name: CI/CD - Teste e Deploy

on:
  push:
    branches: [ "main" ]   # Dispara o workflow apenas em pushs para a branch main

jobs:
  # Etapa de Testes
  test:
    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 testes
        run: pytest -v || echo "Sem testes definidos"
        # Se não houver testes, não falha o pipeline — apenas imprime a mensagem

  # Etapa de Deploy (só roda se os testes passarem)
  deploy:
    runs-on: ubuntu-latest
    needs: test       # Espera o job 'test' terminar antes de começar
    if: success()     # Só executa se o job anterior tiver sido bem-sucedido

    steps:
```

```
    - name: Checkout do código
      uses: actions/checkout@v4

    - name: Instalar dependências (opcional)
      run: |
        echo "Nenhuma dependência necessária para HTML puro"

    - name: Publicar no GitHub Pages
      uses: peaceiris/actions-gh-pages@v4
      with:
        github_token: ${{ secrets.GITHUB_TOKEN }}
        publish_dir: ./   # Diretório raiz do seu site
```

📋 3. Entrega da Atividade

Tarefas obrigatórias:

1. Criar o workflow deploy.yml.
2. Fazer commit e push.
3. Verificar se o deploy foi executado com sucesso no GitHub Actions.
4. Enviar o link do site publicado no GitHub Pages.

Tarefa bônus:

- Adicionar uma etapa condicional que só faz deploy se os testes passarem.

🧠 4. Para finalizar

- Qual é a principal diferença prática entre CI e CD?
- O que aconteceria se o teste falhasse antes do deploy?
- Como a entrega contínua aumenta a confiança do time no processo?

💡 *Respostas:*

A principal diferença prática entre CI e CD é que o CI se concentra em integrar e testar continuamente o código para garantir que ele permaneça estável, enquanto o CD automatiza a entrega desse código já validado para o ambiente final, tornando a publicação mais rápida e confiável.

Se um teste falhar antes do deploy, o processo simplesmente é interrompido e o código não é publicado, evitando que uma versão com erro chegue ao usuário ou ao ambiente de produção.

A entrega contínua aumenta a confiança do time porque automatiza e padroniza o processo, garantindo que somente versões testadas e estáveis sejam entregues, eliminando etapas manuais e reduzindo significativamente o risco de erros.
