

Etapa 3

1. O log mostra cada "step" que definimos no .yml. Você verá o "Checkout", a "Configuração do Python" e, o mais importante, no "step" Executar script, aparecerá a saída Hello CI/CD!

2. O GitHub Actions detecta esse novo push automaticamente (por causa da configuração on: push: branches: ["main"]) e inicia uma nova execução de todo o workflow do zero.

Etapa 4

Pergunta extra: Se um assert falhar (ex: `assert 1 + 1 == 3`), o step "Executar testes" dará erro. Isso interrompe todo o workflow e o GitHub marca o "job" com um X vermelho. Isso é o objetivo da CI: impedir que código com bugs avance.

2 execuções de fluxo de trabalho				Evento ▾	Status ▾	Filial ▾	Ator ▾
✓	Adicionados testes automatizados com pytest	principal	6 minutos ago	...	12s		
Teste CI #2: Commit 0281776 enviado por Eduardo-Silva-TI							
✓	fluxo de trabalho de teste CI	principal	27 minutos ago	...	9s		
Teste CI #1: Commit 1c8aa3a enviado por Eduardo-Silva-TI							

Minhas UCs x Google Gemini x Adicionados testes automatizados com pytest x Aula_Pratica_CI_ x 127.0.0.1:60187/ x pedro leopoldo x Atividade do João x

github.com/Eduardo-Silva-TI/ci-cd-teste/actions/runs/19247714045/job/55025723496#step:1:1

Mineros acusados d... Nova pasta boruto ep 90 - Pesq... Gmail YouTube Maps Serviço Acompanhamento ASSASSINO DE ALU... Todos os marcadores

← Teste CI

✓ Adicionados testes automatizados com pytest #2

Executar novamente todos os trabalhos ...

Resumo

Empregos

construir

Detalhes da execução

Uso

Arquivo de fluxo de trabalho

construir

teve sucesso há 7 minutos em 8s

Registros de pesquisa

- ✓ Configurar tarefa 1s
- ✓ Confirma o código 0s
- ✓ Python 1s
- ✓ 2s
- ✓ Testes do Executor 1s
- ✓ Script executor 0s
- ✓ Postar Python 0s
- ✓ Pós-finalização da compra do código 0s
- ✓ Trabalho concluído 0s

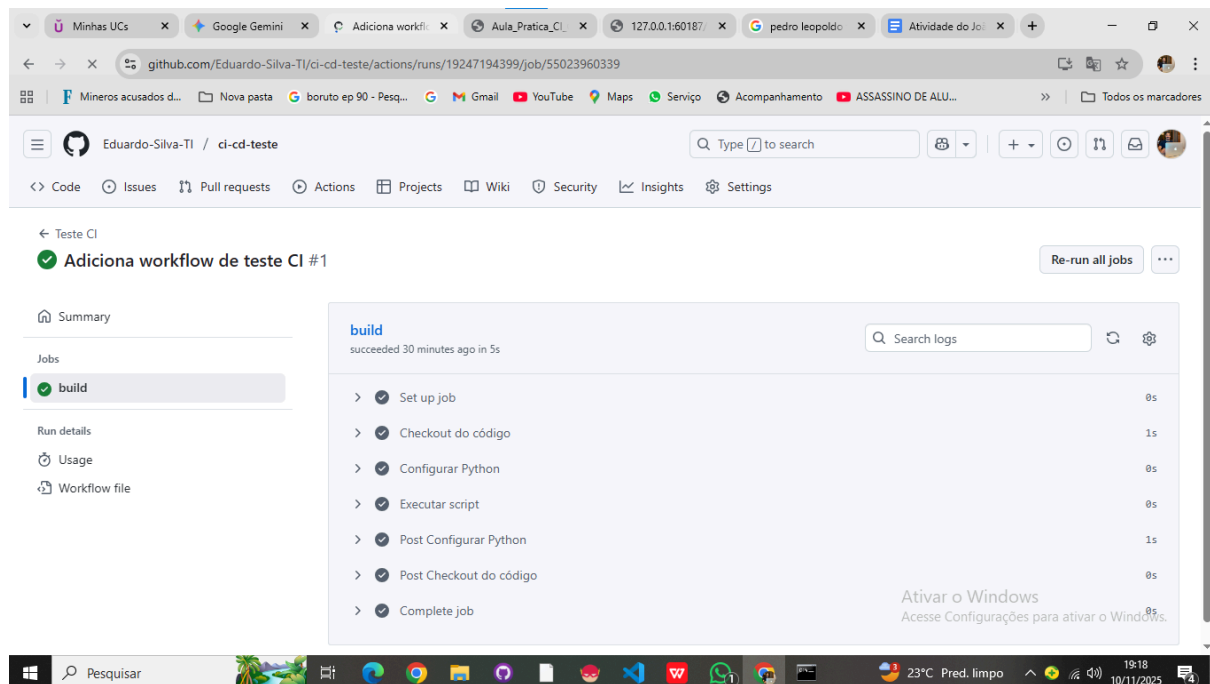
Ativar o Windows

Acesse Configurações para ativar o Windows.

Pesquisar

23°C

19:18 10/11/2025



4. Para finalizar:

- Ele **executa os testes automatizados (como o `pytest`) a cada push**. Se uma nova alteração de código quebrar um teste, o desenvolvedor (ou a equipe) é notificado imediatamente pelo "build" vermelho.

- **Web:** A cada push, o pipeline poderia rodar testes, fazer o *build* do projeto (ex: `npm run build` para React) e, se passar, fazer o *deploy* automático para um servidor (como Vercel, Netlify ou AWS).

Mobile: A cada push, o pipeline poderia compilar o app (APK/IPA) e enviá-lo para um serviço de testes internos (como TestFlight da Apple ou Google Play Internal Testing).

- Adicionando um **novo "step" no final** do nosso "job" *build*. Esse "step" só rodaria se os testes passassem. Ele usaria uma *action* específica (ex: `aws-s3-sync`) ou um script (ex: `scp/rsync`) para copiar os arquivos do projeto para o servidor de produção.

Link do GitHub:

<https://github.com/Eduardo-Silva-TI/ci-cd-teste/actions/runs/19247194399/job/55023960339>