

Derrubei Todo o Sistema da Empresa Com um Comentário

Passo a passo de como derrubar um projeto inteiro com uma linha de código

ESSA FOI A MINHA PRIMEIRA M**
COMO ESTAGIÁRIO**



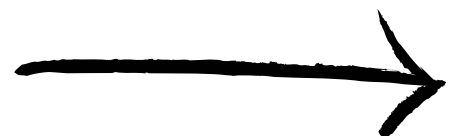
GitHub Actions

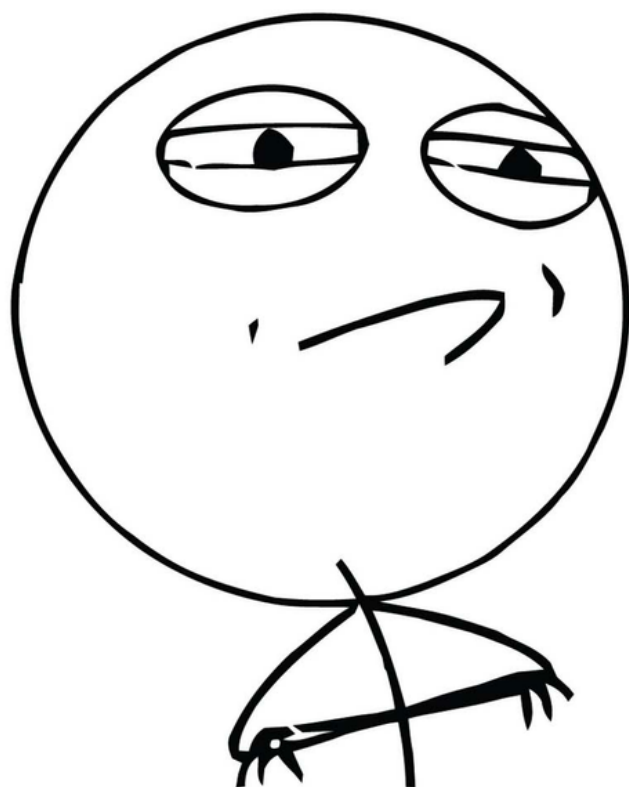


**Sextou! Dia de Deploy
sem CI/CD em Prod!**

Convenhamos,
**É impossível derrubar um
projeto inteiro com uma
linha de comentário...**

Não é?





CHALLENGE ACCEPTED



O projeto

```
from django.http import HttpResponse
import platform
import django

def hello_world(request):
    python_version = platform.python_version()
    django_version = django.get_version()

    return HttpResponse("Hello, World!<br>Tudo funcionando bem<br>"
                        "--Python Version: {}<br>"
                        "--Django Version: {}".format(python_version,
                                                        django_version))
```

(O código)

```
Hello, World!
Tudo funcionando bem
--Python Version: 2.7.18
--Django Version: 1.4
```

(A aplicação rodando em Prod)



A Suposição Enganosa

Estagiário - "Foi só um comentário, não precisa testar, **vou fazer minha PR**, to cheio de coisa para fazer."

```
from django.http import HttpResponse
import platform
import django

def hello_world(request):
    python_version = platform.python_version()
    django_version = django.get_version()

    return HttpResponse("Hello, World!<br>Tudo funcionando bem até  
agora<br>"
                        "--Python Version: {}<br>"
                        "--Django Version: {}".format(python_version,
                                                        django_version))

git init
```

Contexto:

- Sistema legado
- Python 2.7
- Django 1.4
- Sem CI/CD



A Realidade dos PRs

Dev sênior - "Foi só um comentário, não preciso testar, **vou aprovar**, to cheio de coisa para fazer."



(Obs: Você saberia identificar o erro?)

1 minuto depois...



ERROR

SyntaxError at /hello/

Non-ASCII character '\xc3' in file /app/hello/views.py on line 9, but no encoding declared; see <http://python.org/dev/peps/pep-0263/> for details (views.py, line 9)

Request Method: GET

Request URL: http://0.0.0.0:8000/hello/

Django Version: 1.4

Exception Type: SyntaxError

Exception Value: Non-ASCII character '\xc3' in file /app/hello/views.py on line 9, but no encoding declared; see <http://python.org/dev/peps/pep-0263/> for details (views.py, line 9)

Exception Location: /app/hello/urls.py in <module>, line 2

Python Executable: /usr/local/bin/python

Python Version: 2.7.18

Python Path: ['/app',
'/usr/local/lib/python2.7.zip',
'/usr/local/lib/python2.7',
'/usr/local/lib/python2.7/plat-linux2',
'/usr/local/lib/python2.7/lib-tk',
'/usr/local/lib/python2.7/lib-old',
'/usr/local/lib/python2.7/lib-dynload',
'/usr/local/lib/python2.7/site-packages']

Server time: Fri, 18 Aug 2023 04:46:01 -0500

Você saberia dizer **o que quebrou** a aplicação? Sim, aquela linha quebrou o app e **TODOS** os clientes vão ver!

(Obs: O primeiro comentário é o meu Github com o projeto, caso você não acredite que é possível quebrar um projeto com uma linha de comentário!)



Contexto do caso

Esse post teve como motivação uma publicação do r/brdev no reddit.

Foi um caso real onde o usuário compartilhou sua experiência.

Basicamente era um **sistema legado**, com uma versão de **Python sem suporte** e um **Framework desatualizado**, sem **CI/CD**, sem processo de **Review/PR**. Ou seja,

Igual a 80% dos projetos!



Um comentário

À primeira vista, pode parecer um mero comentário. Mas na realidade...



```
@@ -6,6 +6,6 @@ def hello_world(request):
6     python_version = platform.python_version()
7     django_version = django.get_version()
8
9 -     return HttpResponse("Hello, World!<br>Tudo
   +     return HttpResponse("Hello, World!<br>Tudo
      funcionando bem<br>"
10                                "--Python Version: {}<br>"
11                                "--Django Version:
   +                                "--Python Version: {}<br>"
   +                                "--Django Version:
      {}".format(python_version, django_version))
```

O Python 2.7 não suporta caracteres com acento. Esse '**funcionando bem até agora**' transformou um comentário inofensivo em uma armadilha letal.



A realidade

Surpreendentemente, isso é a **norma em muitos lugares.**

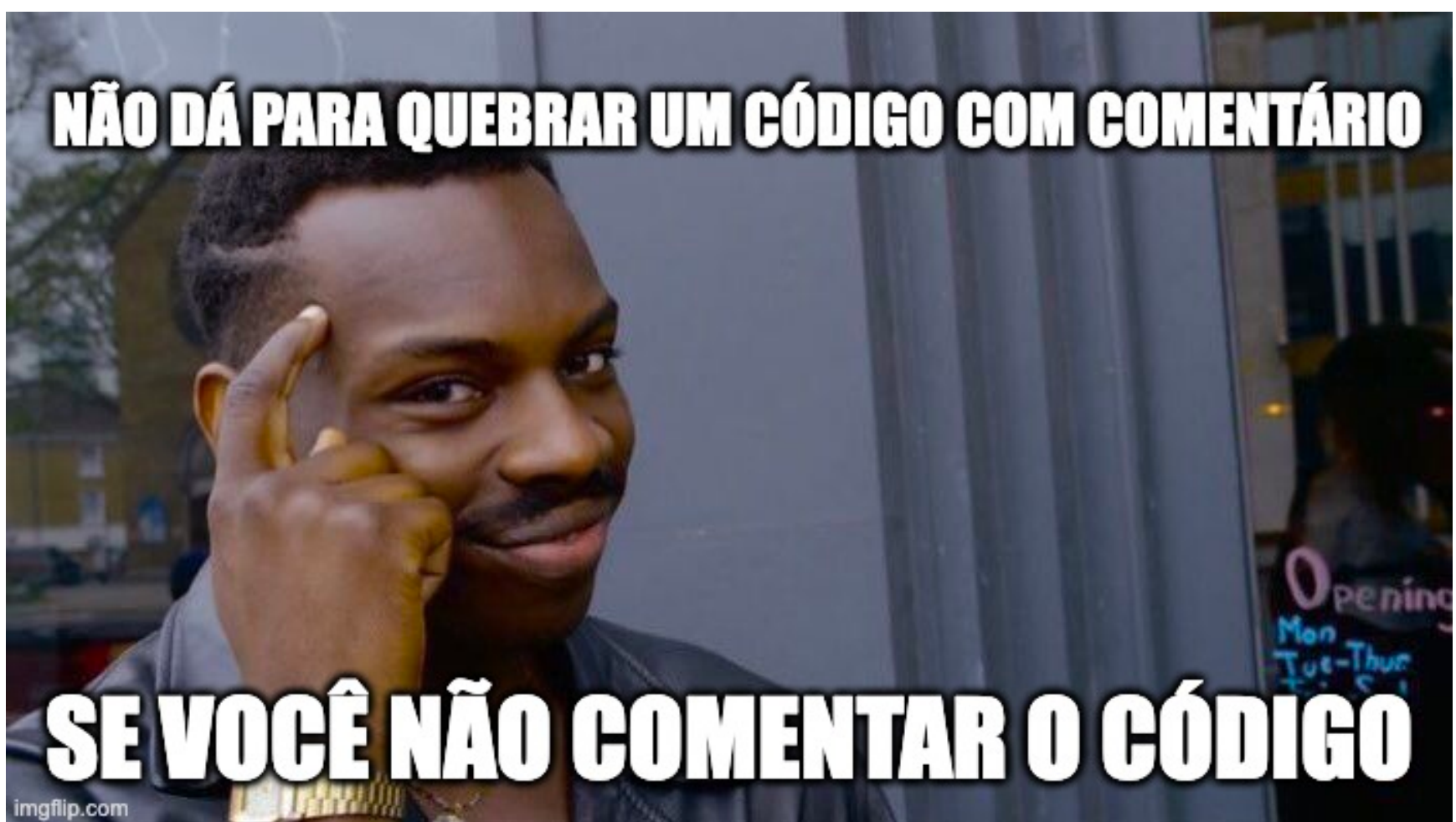
Não é uma questão de '**se**' um problema surgirá em projetos assim, mas '**quando**'.

Tudo é apenas uma questão de tempo. Simplesmente não há tempo de testar **tudo manualmente o tempo todo.**

E como evitar isso?



A sacada



kkk,

Não, **você deve comentar!**



O que é CI?

Definição: Processo **automatizado** em que as alterações do código são **verificadas e testadas!**

Processo Típico:

1. Desenvolvedor envia código para o repositório.
2. Sistema de CI inicia testes automatizados.
3. Feedback é fornecido ao desenvolvedor.
4. Correções são feitas, se necessário.



Exemplo de um CI

```
name: Django CI UTF

on: [push, pull_request]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2

      - name: Manually install Python 2.7 and pip
        run: |
          sudo apt-get update
          sudo apt-get install -y python2.7 python-pip

      - name: Verify Python version
        run: python2.7 -V

      - name: Install dependencies
        run: |
          python2.7 -m pip install --upgrade pip
          python2.7 -m pip install -r requirements.txt

      - name: Check for non-ASCII characters
        run: |
          for file in $(find . -name "*.py"); do
            if grep -Pn "[^[:ascii:]]" $file; then
              echo "Non-ASCII characters found in $file!"
              exit 1
            fi
          done
          echo "No non-ASCII characters detected."
```

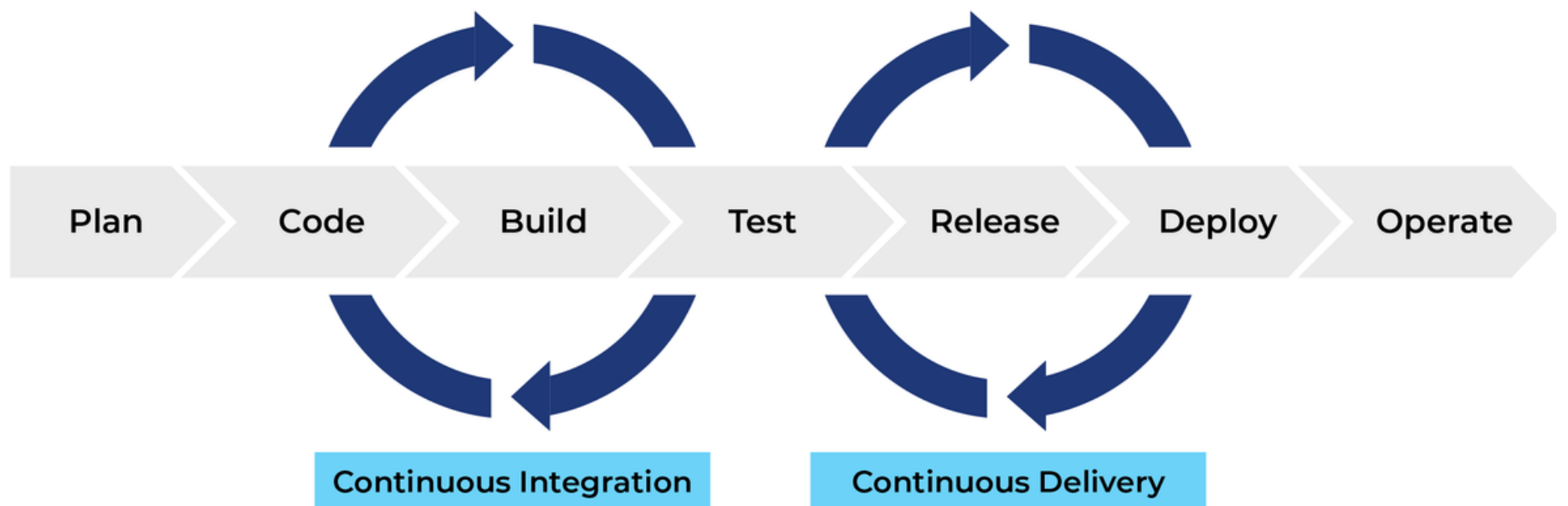
Cria um ambiente espelho

Realiza os testes, nesse caso buscando non-ASCII caracteres.

Sim, com umas 20 linhas de código você evitaria tudo isso.



Por que CI/CD?



**Pq aumenta a eficiência,
melhora a colaboração e ,
principalmente, evita que o
software exploda!**



Prevenção do CI

O CI é como um **QA 24/7** e que **testa tudo sempre**.

Sempre que houver **qualquer modificação** no projeto ele irá testar **TUDO novamente**.

```
name: Django CI UTF
on: [push, pull_request]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2

      - name: Manually install Python 2.7 and pip
        run: |
          sudo apt-get update
          sudo apt-get install -y python2.7 python-pip

      - name: Verify Python version
        run: python2.7 -V

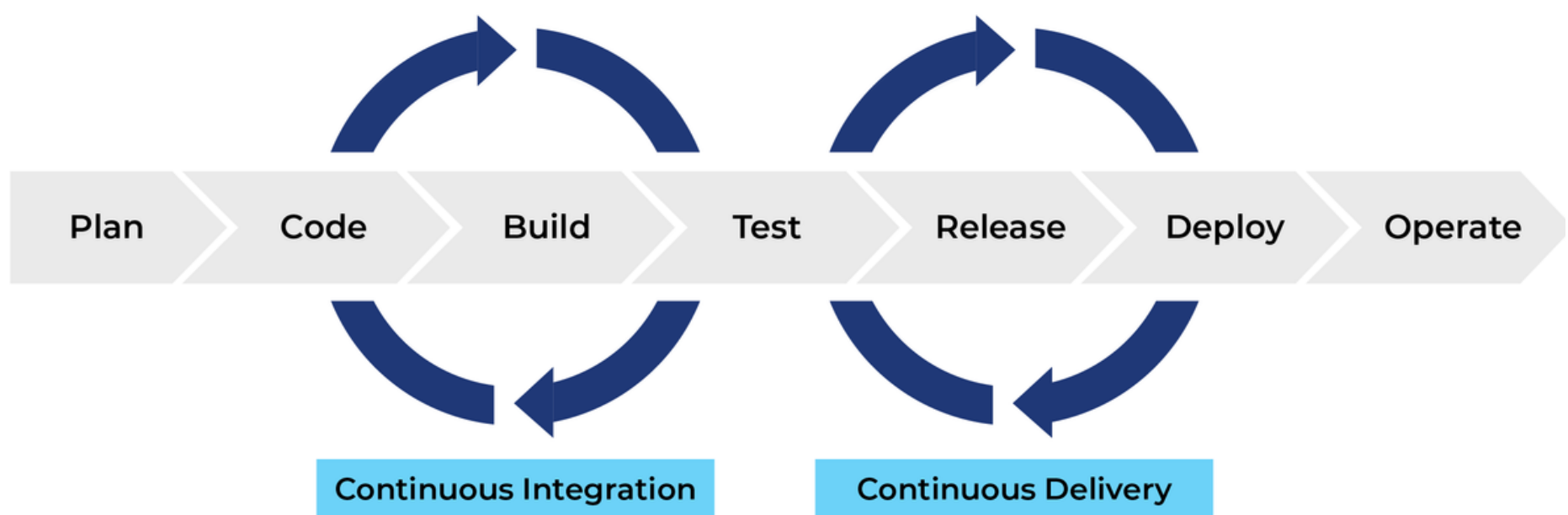
      - name: Install dependencies
        run: |
          python2.7 -m pip install --upgrade pip
          python2.7 -m pip install -r requirements.txt

      - name: Check for non-ASCII characters
        run: |
          for file in $(find . -name "*.py"); do
            if grep -Pn "[^[:ascii:]]" $file; then
              echo "Non-ASCII characters found in $file!"
              exit 1
            fi
          done
          echo "No non-ASCII characters detected."
```

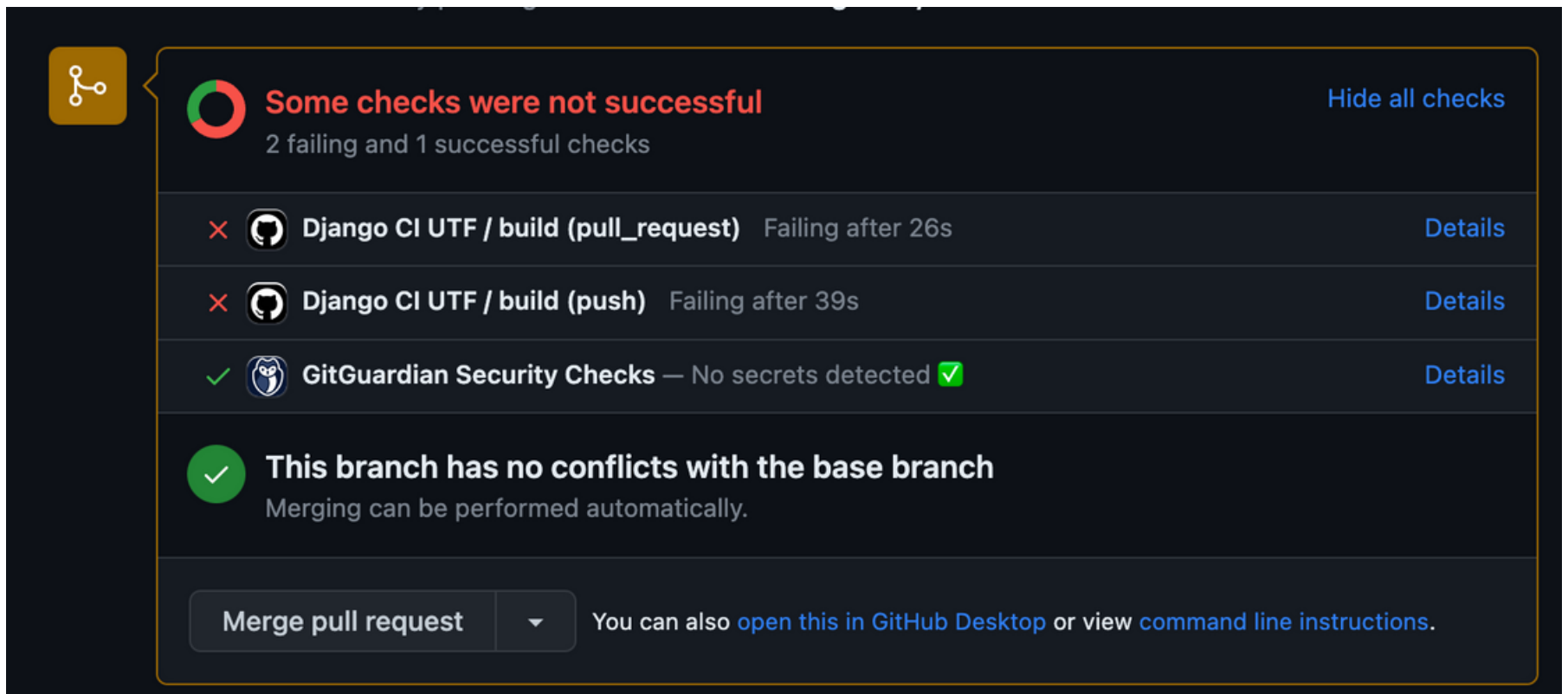
Cria um ambiente espelho

Realiza os testes, nesse caso buscando non-ASCII caracteres.

O código só vai para deploy (chegar no cliente) se passar nos testes!



Exemplo da CI rodando



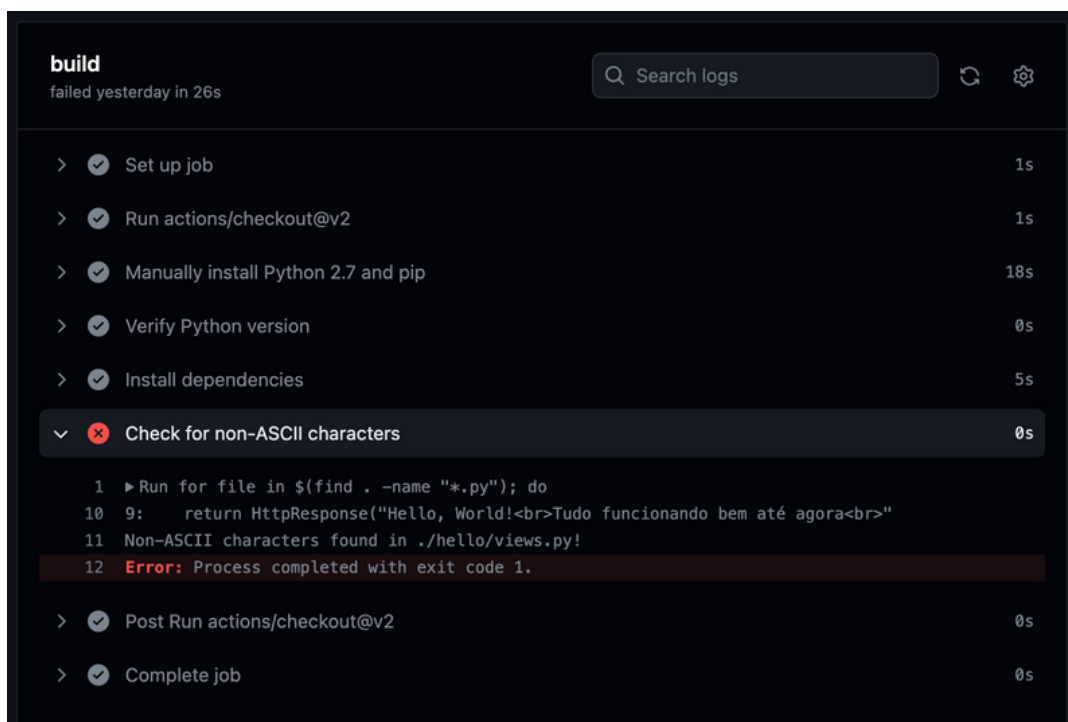
The image shows a GitHub Actions summary view. At the top, a red circle icon indicates that some checks were not successful. Below this, a list of checks is shown: two failing checks (Django CI UTF / build (pull_request) and Django CI UTF / build (push)) and one successful check (GitGuardian Security Checks). A green checkmark icon indicates that the branch has no conflicts with the base branch. At the bottom, there is a button labeled "Merge pull request" and a link to view command line instructions.

Some checks were not successful [Hide all checks](#)
2 failing and 1 successful checks

- ✗ **Django CI UTF / build (pull_request)** Failing after 26s [Details](#)
- ✗ **Django CI UTF / build (push)** Failing after 39s [Details](#)
- ✓ **GitGuardian Security Checks** — No secrets detected ✓ [Details](#)

✓ **This branch has no conflicts with the base branch**
Merging can be performed automatically.

[Merge pull request](#) You can also [open this in GitHub Desktop](#) or view [command line instructions](#).



The image shows a GitHub Actions build log. The build failed yesterday in 26s. The log shows a list of steps: Set up job, Run actions/checkout@v2, Manually install Python 2.7 and pip, Verify Python version, Install dependencies, and Check for non-ASCII characters. The last step, "Check for non-ASCII characters", failed with an error message: "Error: Process completed with exit code 1." The log also shows the output of the script, which includes a non-ASCII character.

build
failed yesterday in 26s

Search logs

- > ✓ Set up job 1s
- > ✓ Run actions/checkout@v2 1s
- > ✓ Manually install Python 2.7 and pip 18s
- > ✓ Verify Python version 0s
- > ✓ Install dependencies 5s
- > ✗ **Check for non-ASCII characters** 0s

```
1 ▶Run for file in $(find . -name "*.py"); do
10 9:   return HttpResponse("Hello, World!<br>Tudo funcionando bem até agora<br>")
11 Non-ASCII characters found in ./hello/views.py!
12 Error: Process completed with exit code 1.
```

- > ✓ Post Run actions/checkout@v2 0s
- > ✓ Complete job 0s

Olha o meu QA 24/7 por aqui. Pegou o meu erro e não deixou eu fazer merge. Ou seja, o projeto não explodiu! Vou ver onde errei, corrigir e enviar novamente.



O que aprendemos?

1) A Importância da Atualização:

Manter seu código e suas ferramentas atualizadas é fundamental. **Pare de usar versão sem suporte!**

2) A Delicadeza do Código:

Um simples comentário pode ter implicações enormes.

3) A Necessidade de CI/CD:

Os processos de Integração Contínua e Entrega Contínua são essenciais para evitar que erros!



Para onde vamos agora?

1) **Aprofunde-se em CI/CD:**

Se ainda não adotou, agora é a hora. Ferramentas como GitHub Actions e Jenkins estão aí para nos ajudar!

2) **Educação Contínua:**

O mundo da tecnologia está sempre evoluindo. Mantenha-se atualizado, participe de comunidades, compartilhe conhecimento e nunca pare de aprender.

3) **Encorajamento:**

Erros acontecem. Em vez de atribuir culpas, use-os como oportunidades de aprendizado!



TL;DR

- 1) **Atenção aos Detalhes**
- 2) **Mantenha-se Atualizado**
- 3) **Adote CI/CD**
- 4) **Revisão de Código**
- 5) **Sempre Aprenda**

Conclusão:

A prevenção é sempre melhor que a correção. Ferramentas e práticas estão disponíveis para nos ajudar a evitar erros. Use-as!



Obrigado!

Se essa postagem te ajudou, curte e comente! **Sua interação pode ajudar outras pessoas** a se beneficiarem desse conhecimento.

Siga-me para receber dicas, insights e tutoriais de problemas que já passei muita noite virado para resolver!

(e espero que você não passe!)

