

# Github Actions

Live de Python #170

# Quem sou eu?

Willian / Will / Wilzão

Fatec-AM - Segurança da informação

Engenheiro de qualidade

@williangl / @willianglxd



# GitHub Actions

GH Actions é uma ferramenta para automatizar tarefas durante o desenvolvimento de software.

O que isso significa?

Que com base em eventos do GitHub, conseguimos fazer com que uma série de comandos (tarefas) seja executada.

Eventos do GitHub? Calma lá que já vamos falar sobre isso

# Tarefas durante o desenvolvimento

Quando estamos desenvolvendo utilizamos algumas ferramentas de análise de código para garantir que estamos seguindo os padrões de desenvolvimento.

# Tarefas durante o desenvolvimento

Black → para estilização de código (linter)

```
→ black app --check  
would reformat app/__init__.py  
Oh no! 💥 💔 💥  
1 file would be reformatted, 1 file would be left unchanged.
```

# Tarefas durante o desenvolvimento

**isort** → para verificação dos imports (PEP-8)

```
✦ → isort --check app/  
ERROR: /home/willian/git/github_actions_live_de_python/app/__init__.py  
Imports are incorrectly sorted and/or formatted.
```

# Tarefas durante o desenvolvimento

`pydocstyle` → para verificação das docstrings (PEP-257)

```
✦ → pydocstyle app  
app/calc.py:1 at module level:  
    D100: Missing docstring in public module  
app/calc.py:1 in public function `soma`:  
    D103: Missing docstring in public function  
app/__init__.py:1 at module level:  
    D104: Missing docstring in public package  
app/__init__.py:5 in public function `create_app`:  
    D103: Missing docstring in public function
```

# Tarefas durante o desenvolvimento

pytest → para verificação dos testes unitários

```
→ pytest
```

```
===== test session starts =====  
platform linux -- Python 3.9.5, pytest-6.2.4, py-1.10.0, pluggy-0.13.1  
rootdir: /home/willian/git/github_actions_live_de_python, configfile: pyproject.toml  
collected 2 items  
  
tests/test_soma.py .. [100%]  
  
===== 2 passed in 5.10s =====
```



# GitHub Actions

Como o GitHub Actions tem o objetivo de automatizar essas tarefas repetidas, podemos utilizá-lo para fazer esse trabalho para nós.

*O formato do arquivo de configuração do GitHub Actions é o YAML que trabalha com indentação de 2 espaços.*

*Os arquivos devem ser armazenados na pasta **.github/workflows/***

# GitHub Actions

Seguindo a ordem com que executamos as validações este seria nosso arquivo YAML de configuração.

```
name: Integração Contínua

on: push

jobs:
  my_test:
    runs-on: ubuntu-latest
    steps:
      - name: Action Checkout
        uses: actions/checkout@v2

      - name: Instala o Python 3.9
        uses: actions/setup-python@v2
        with:
          python-version: 3.9

      - name: Instala o Poetry
        uses: Gr1N/setup-poetry@v4

      - name: Instala as Dependências
        run: poetry install

      - name: Executa o Black
        run: poetry run black app --check

      - name: Executa o Isort
        run: poetry run isort --check app/

      - name: Executa o Pydocstyle
        run: poetry run pydocstyle app

      - name: Executa Testes Unitários
        run: poetry run pytest
```

# GitHub Actions

Vamos ver como fica!

# Anatomia GitHub Actions

Name → Nome do Workflow

*\*Workflows são os procedimentos automatizados adicionados ao repositório.*

*São constituídos por um ou mais trabalhos (jobs) e podem ser ativados por um evento.\**



```
name: Integração Contínua

on: push

jobs:
  my_test:
    runs-on: ubuntu-latest
    steps:
      - name: Action Checkout
        uses: actions/checkout@v2

      - name: Instala o Python 3.9
        uses: actions/setup-python@v2
        with:
          python-version: 3.9
```

# Anatomia GitHub Actions

On → Evento que aciona o Workflow

*\*Eventos são atividades específicas que acionam o Workflow\**



```
name: Integração Contínua

on: push

jobs:
  my_test:
    runs-on: ubuntu-latest
    steps:
      - name: Action Checkout
        uses: actions/checkout@v2

      - name: Instala o Python 3.9
        uses: actions/setup-python@v2
        with:
          python-version: 3.9
```

**Eventos??**

Como assim eventos?

# Gitflow

Quando trabalhamos em times, o projeto recebe alteração de tempos em tempos de pessoas diferentes com códigos diferentes.

O gitflow entra para nos auxiliar em problemas que poderiam ocorrer se não houvesse uma organização em como realizar as modificações.

Com o gitflow definimos nossa branch principal, criamos uma branch de desenvolvimento e a partir dessa branch criamos a branch onde iremos trabalhar.

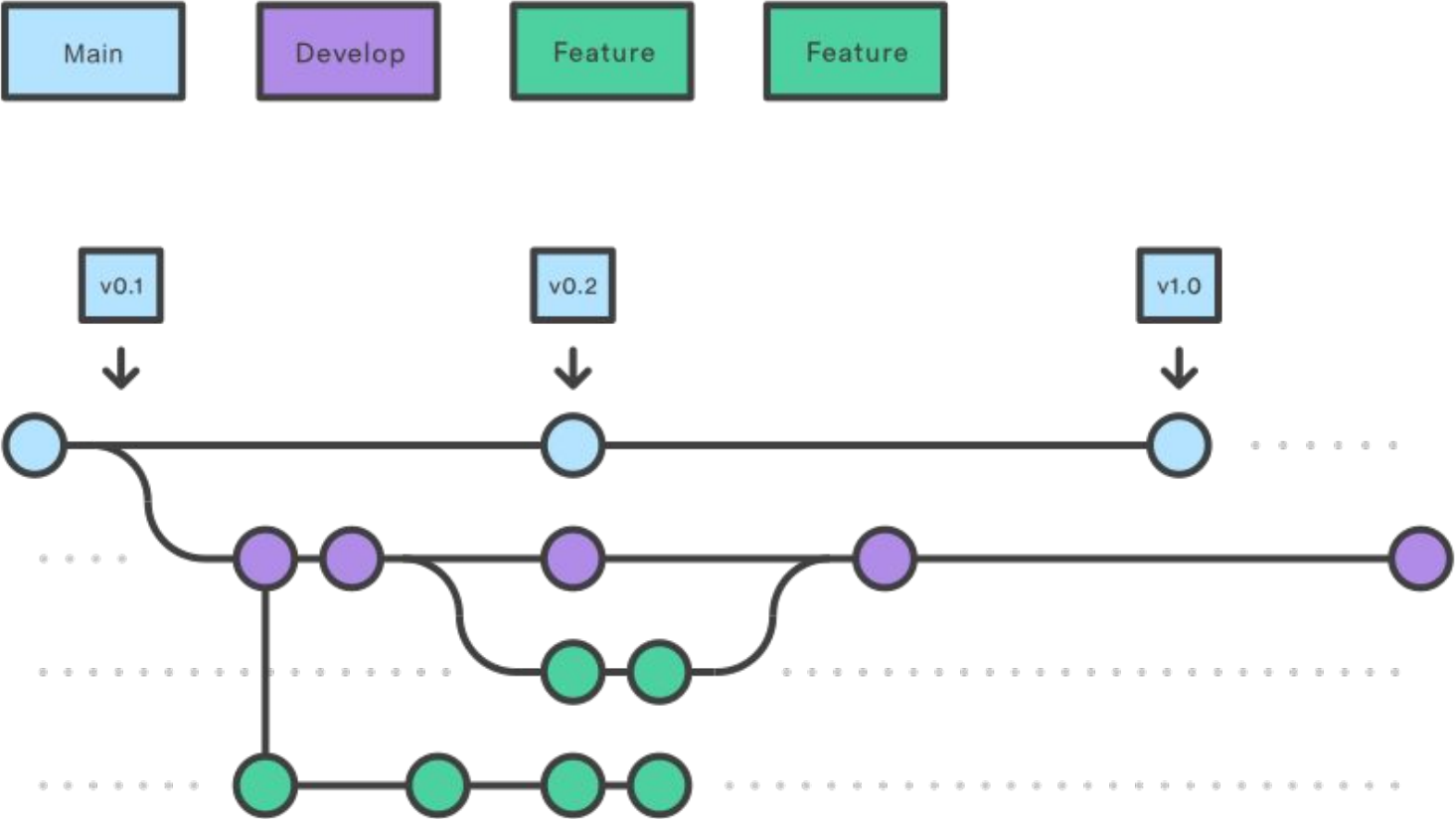
# Gitflow

Depois de finalizado o trabalho, fazemos o push para nosso repositório remoto, abrimos um pull request para ser revisado, testado e mergeado na nossa branch de desenvolvimento e quando todas as modificações entrarem na branch de desenvolvimento podemos fechar a versão para mergearmos na branch principal.

Como podemos observar na imagem a seguir.



# Gitflow



## Eventos??

Quando falamos em fazer um ***push*** ou um ***pull request*** estamos falando de 2 tipos de eventos que poderiam acionar nosso Workflow.

Detalhe!! Esses eventos podem ser combinados e configurados por branches

Além desses 2 existem diversos outros, porém não iremos abordá-los hoje.

Você pode encontrar essa informação aqui:

<https://docs.github.com/pt/actions/reference/events-that-trigger-workflows>

Eventos??



```
name: Integração Contínua
```

```
on:
```

```
  pull_request:
```

```
    branches:
```

```
      - main
```

```
  push:
```

```
jobs:
```

```
  ...
```

# Anatomia GitHub Actions

Jobs → Trabalhos do Workflow

***\*Trabalhos são um conjunto de etapas executadas no mesmo Workflow.\****



```
name: Integração Contínua

on: push

jobs:
  my_test:
    runs-on: ubuntu-latest
    steps:
      - name: Action Checkout
        uses: actions/checkout@v2

      - name: Instala o Python 3.9
        uses: actions/setup-python@v2
        with:
          python-version: 3.9
```

# Anatomia GitHub Actions

My\_test → Nome do job

*\*Jobs por padrão podem ser executados em paralelos, então cada job necessita de um nome\**



```
name: Integração Contínua

on: push

jobs:
  my_test:
    runs-on: ubuntu-latest
    steps:
      - name: Action Checkout
        uses: actions/checkout@v2

      - name: Instala o Python 3.9
        uses: actions/setup-python@v2
        with:
          python-version: 3.9
```

## Job em paralelo??

Quando criamos mais de um job no nosso Workflow estes jobs são executados em paralelo por default, mas também pode ser executado em série (caso queira)



**name:** Integração Contínua com jobs paralelos

**on:** push

**jobs:**

**executa\_linter:**

**runs-on:** ubuntu-latest

**steps:**

- **name:** Action Checkout  
**uses:** actions/checkout@v2
- **name:** Instala o Python 3.9  
**uses:** actions/setup-python@v2  
**with:**  
**python-version:** 3.9
- **name:** Instala o Poetry  
**uses:** Gr1N/setup-poetry@v4
- **name:** Instala as Dependências  
**run:** poetry install
- **name:** Executa o Black  
**run:** poetry run black app --check

**executa\_isort:**

**runs-on:** ubuntu-latest

**steps:**

- **name:** Action Checkout  
**uses:** actions/checkout@v2
- **name:** Instala o Python 3.9  
**uses:** actions/setup-python@v2  
**with:**  
**python-version:** 3.9
- **name:** Instala o Poetry  
**uses:** Gr1N/setup-poetry@v4
- **name:** Instala as Dependências  
**run:** poetry install
- **name:** Executa o Isort  
**run:** poetry run isort --check app/

# Anatomia GitHub Actions

Runs-on → Executor

*\*Executores são basicamente o Sistema Operacional onde cada tarefa será executada.*

*Atualmente o GitHub suporta:*

- **Ubuntu**
- **Windows**
- **MacOS**
- **Self-hosted\***



```
name: Integração Contínua
```

```
on: push
```

```
jobs:
```

```
  my_test:
```

```
    runs-on: ubuntu-latest
```

```
    steps:
```

```
      - name: Action Checkout
```

```
        uses: actions/checkout@v2
```

```
      - name: Instala o Python 3.9
```

```
        uses: actions/setup-python@v2
```

```
        with:
```

```
          python-version: 3.9
```



# Anatomia GitHub Actions

Steps → Etapa

*\*Etapa é uma tarefa individual que executa comandos em um job\**



```
name: Integração Contínua

on: push

jobs:
  my_test:
    runs-on: ubuntu-latest
    steps:
      - name: Action Checkout
        uses: actions/checkout@v2

      - name: Instala o Python 3.9
        uses: actions/setup-python@v2
        with:
          python-version: 3.9
```

# Anatomia GitHub Actions

Uses/Run → Ações

*\*Ações são comandos sozinhos que combinados em steps criam um job.\**

```
- name: Instala o Python 3.9
  uses: actions/setup-python@v2
  with:
    python-version: 3.9

- name: Instala o Poetry
  uses: Gr1N/setup-poetry@v4

- name: Instala as Dependências
  run: poetry install

- name: Executa o Black
  run: poetry run black app --check

- name: Executa o Isort
  run: poetry run isort --check app/

- name: Executa o Pydocstyle
  run: poetry run pydocstyle app

- name: Executa Testes Unitários
  run: poetry run pytest
```

# Integração Contínua - CI

Vocês já ouviram a palavra do CI hoje?

Basicamente esses fluxos que automatizamos são nossa bateria do CI

# Integração Contínua - CI

É uma prática no desenvolvimento de software onde as pessoas do time integram seu trabalho diariamente, cada uma delas.

Essas integrações precisam passar por validações para garantir que nada esteja quebrado.

O que significa? Bateria de validações da integridade do projeto, feedback rápido.

- Análise sintática (lint)
- Complexidade Ciclomática
- Testes unitários (coverage)
- Testes de integração
- Testes end-to-end (e2e)\*

# GitHub Actions

Top! Agora nós sabemos uma das funções do GitHub Actions.

Como agora toda nossa integração está sendo validada a todo instante talvez possamos fazer o deploy dela também, já que há uma massa de teste garantindo que tudo está funcionando.

# GitHub Actions

Com o GitHub Actions também conseguimos automatizar o deploy da nossa aplicação.

```
name: Deploy Automático

on:
  workflow_run:
    workflows: ["Integração Contínua"]
    branches: [main]
    types:
      - completed

jobs:
  deploy_app:
    runs-on: ubuntu-latest

    steps:
      - name: Make checkout
        uses: actions/checkout@v2

      - name: Deploy no Heroku
        uses: akhileshns/heroku-deploy@v3.12.12
        with:
          heroku_api_key: ${secrets.HEROKU_API_KEY}
          heroku_app_name: "todo-list-live-de-python"
          heroku_email: "williangldzn@gmail.com"
```

# GitHub Actions

Mas também é possível deixá-lo automatizado com trigger manual

```
name: Deploy Manual

on: workflow_dispatch

jobs:

  deploy_app:
    runs-on: ubuntu-latest

    steps:
      - name: Make checkout
        uses: actions/checkout@v2

      - name: Deploy no Heroku
        uses: akhileshns/heroku-deploy@v3.12.12
        with:
          heroku_api_key: ${secrets.HEROKU_API_KEY}
          heroku_app_name: "app-calculadora-gh-actions"
          heroku_email: "williangldzn@gmail.com"
```

# GitHub Actions

Assim como vimos que com o GH Actions conseguimos executar uma bateria de validação e demos o nome de CI, agora que também sabemos que o deploy pode ser realizado com ele, podemos falar sobre deploy contínuo.



# Deploy Contínuo - CD

É um conceito vindo do Extreme Programming, onde se todas as mudanças feitas passaram na bateria de testes então essas alterações estão prontas para subir para produção (ser deployada).

# Entrega Contínua - Continuous Delivery

É a disciplina do desenvolvimento de software que constitui na construção de software que pode ser lançado para produção a qualquer momento.

Estamos fazendo Continuous Delivery quando:








- O software é deployavel
- As novas alterações são feitas de uma forma que mantém o software deployavel
- Feedbacks rápidos em produção de qualquer mudança realizada por alguém
- É possível realizar o deploy de qualquer versão em qualquer ambiente através de um aperto de botão

# Entrega Contínua - Continuous Delivery

Vale ressaltar que a entrega pode acontecer em qualquer ambiente, seja ele produção ou de teste.

O tempo em que as entregas vão acontecer ficam a critério do projeto, podendo ser diariamente, semanalmente e assim por diante.

# Ferramentas de CI/CD

- GitLab-ci  / 
- Jenkins 
- Bamboo (Atlassian) 
- **Github Actions**  /  (repo público)
- Team City (JetBrains) 

# Act

Act é uma ferramenta muito útil que permite a execução do Workflow sem a necessidade de commitar as alterações.

Com o Act conseguimos executar localmente (através do docker) o workflow que criamos e assim testamos se nossa configuração está correta.

O Act pode ser encontrado aqui:

<https://github.com/nektos/act>

Isso é tudo pessoal!!

# Referências

## GitHub Actions:

- <https://docs.github.com/pt/actions/learn-github-actions/introduction-to-github-actions>
- <https://docs.github.com/pt/actions/reference/events-that-trigger-workflows>

## Gitflow:

- <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>
- <https://git-scm.com/docs/gitworkflows>
- <https://nvie.com/posts/a-successful-git-branching-model/>
- <https://guides.github.com/introduction/flow/>

# Referências

## Integração Contínua:

- Jez Humble, David Farley - Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. 2010, Addison-Wesley Professional
- <https://www.atlassian.com/continuous-delivery/continuous-integration>
- Live de Python #106 - CI - Integração contínua
- <https://www.jamesshore.com/v2/blog/2006/continuous-integration-on-a-dollar-a-day>
- <https://www.jamesshore.com/v2/blog/2005/continuous-integration-is-an-attitude>
- <https://www.martinfowler.com/articles/continuousIntegration.html>

## Deploy Contínuo:

- Jez Humble, David Farley - Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. 2010, Addison-Wesley Professional
- <https://www.atlassian.com/continuous-delivery/continuous-deployment>



# Referências

## Entrega Contínua:

- Jez Humble, David Farley - Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. 2010, Addison-Wesley Professional
- <https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment>
- <https://www.martinfowler.com/bliki/ContinuousDelivery.html>

## Act:

- <https://github.com/nektos/act>