



# FLUXO DE TRABALHO NO GIT

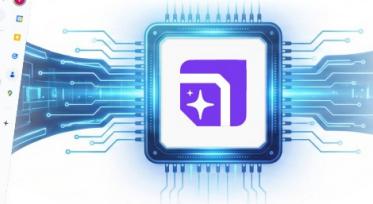
Nesse ebook você vai aprender como é o Fluxo de Trabalho no Git!

# Crie Ebooks técnicos incríveis em minutos com IA

Conheça a 1ª IA Especializada na criação de Ebooks **com código!**



Chega de formatar código no Google Docs



Deixe que nossa IA faça o trabalho pesado

 Syntax Highlight

 Adicione Banners Promocionais

 Edite em Markdown em Tempo Real

 Infográficos feitos por IA

**TESTE AGORA** 

Salve salve Pythonista!

Nesse post você vai aprender o fluxo de trabalho do **Git**!

Você verá:

- Como realizar a configuração inicial do Git com o comando `git init`
- Os estados do Fluxo de Trabalho do Git: *Working Directory*, *Staging* e *Repository*.
- Como verificar o estado da sua *branch* com `git status`.
- Adicionar arquivos à área de *Staging* com `git add`.
- Como criar commits com `git commit`.
- Como criar *branches* com `git branch`.
- Como mesclar códigos de diferentes *branches* com `git merge`.
- Como atualizar o repositório remoto com `git push`.

E **muito mais!**

Então vamos nessa!

## Configuração Inicial do Git

Após a instalação do `git`, você deve definir seu **usuário** e **email** para que esses dados sejam utilizados pelo git ao criar um commit.

Fazemos isso através dos seguintes comandos:

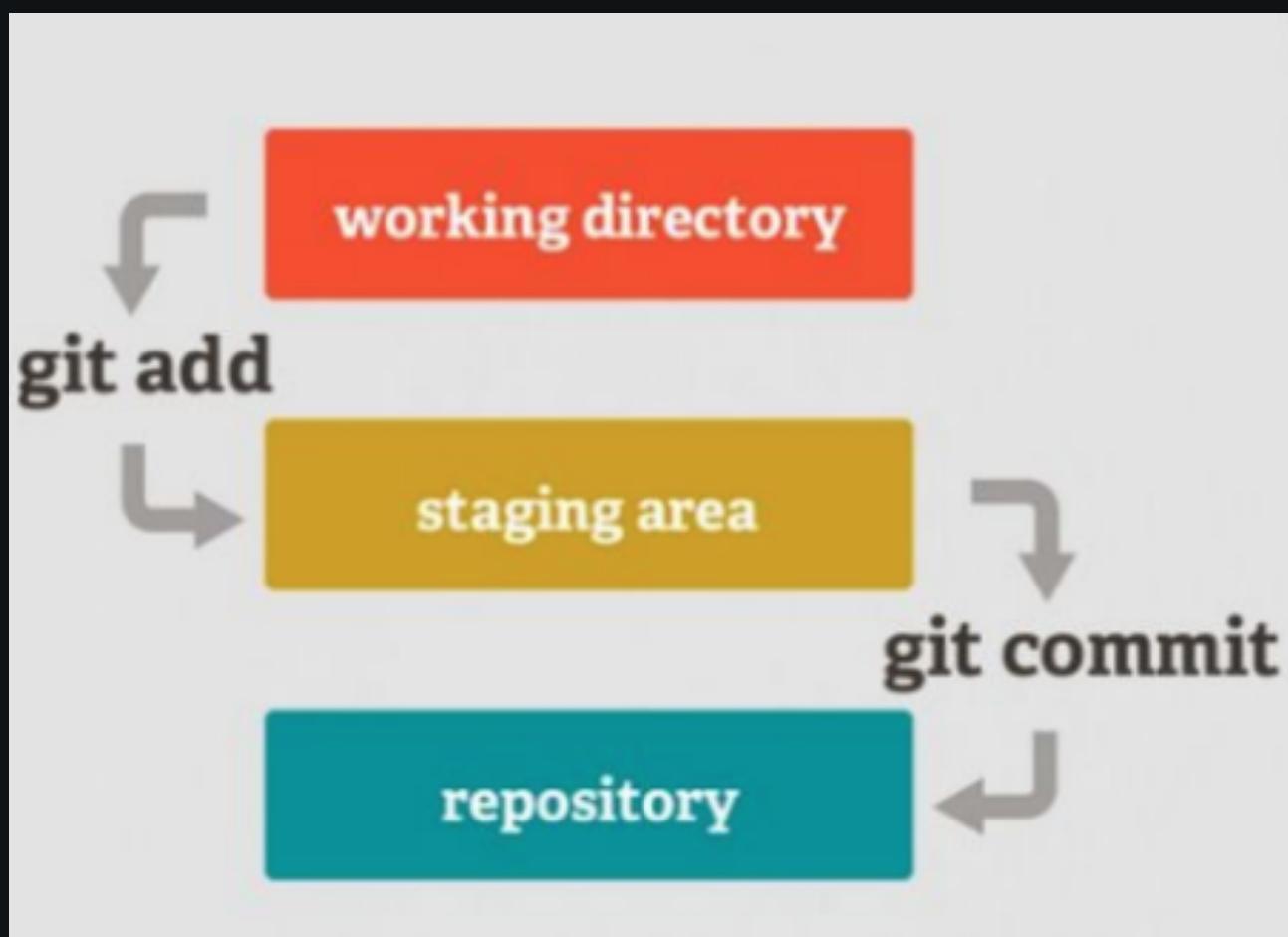
```
git config --global user.name "Seu nome"  
git config --global user.email seu.email@email.com
```

Após ter feito isso, vá a pasta do projeto e use o comando `git init` para inicializar um novo repositório:

```
git init
```

# Fluxo de trabalho no Git

Quando se está trabalhando localmente, o git representa seu repositório em três estados distintos.



O primeiro deles é a **Working Directory** que contém a versão atual dos arquivos do projeto.

O segundo é o **Staging (Index)**, que funciona como uma área temporária que diz ao git o que irá em seu próximo commit.

Por último, temos o **Repository (HEAD)** que aponta para o último commit feito no repositório remoto.

O principal comando para determinar quais os estados atuais dos arquivos do projeto é:

```
git status
```

Para mover apenas um arquivo da *Working Directory* para o *Staging*, utilizamos:

```
git add <arquivo>
```

Ou, para mover todos os arquivos que foram modificados:

```
git add *
```

E para mover um arquivo do *Staging* para o *Repository*, criando um commit:

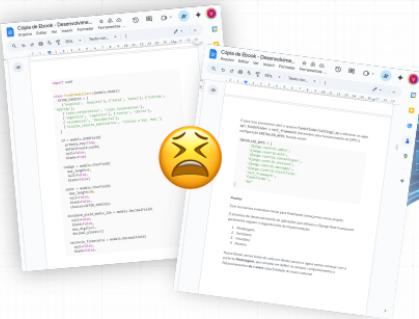
```
git commit -m "Mensagem do commit"
```



*Estou desenvolvendo o **DevBook**, uma plataforma que usa IA para gerar ebooks técnicos profissionais. Depois de ler, dá uma passada no site!*

## Crie Ebooks técnicos incríveis em minutos com IA

Conheça a 1ª IA Especializada na criação de Ebooks **com código!**



Chega de formatar código no Google Docs



Deixe que nossa IA faça o trabalho pesado

 Syntax Highlight

 Adicione Banners Promocionais

 Edite em Markdown em Tempo Real

 Infográficos feitos por IA

**TESTE AGORA! PRIMEIRO CAPÍTULO 100% GRÁTIS** 

## Ramificações do projeto Git

Utilizamos as chamadas *Branches* para desenvolver código isoladamente.

Isto é, ao se trabalhar em uma branch, podemos testar e experimentar à vontade, sem interferir no trabalho de outros, bastando retornar à branch original para se ter a versão anterior do código.

A branch `main` (antigamente chamada de `master`) é a única branch e a branch “padrão” quando se cria um novo repositório.

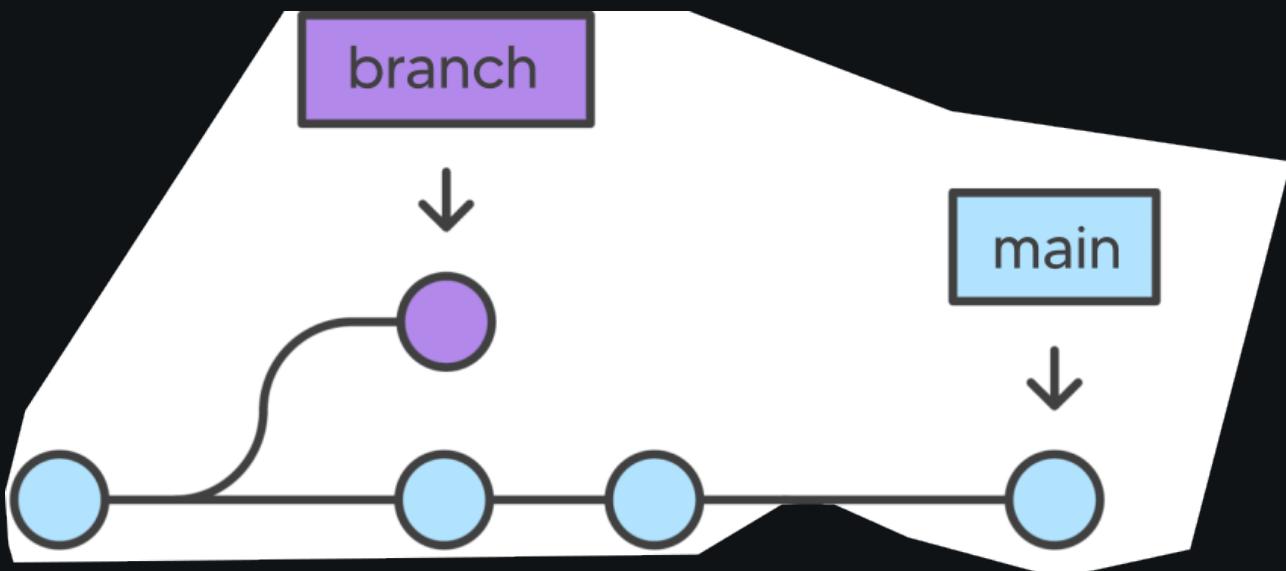
O comando para se criar uma nova branch é o:

```
git branch <nome da branch>
```

Você pode mudar de branch utilizando o comando:

```
git checkout <nome da branch>
```

Visualmente, esse é o funcionamento de uma branch - uma ramificação no código:



## Atualizando o servidor remoto Git

Ao finalizar o desenvolvimento de uma nova funcionalidade, mesclamos os códigos da branch que foi criada com a ramificação principal do projeto ( `main` ou `master` ).

Fazemos isso com o comando `merge` .

Esse comando pega as alterações de uma branch e as aplica à outra branch:

```
git merge <branch para mesclar>
```

Por exemplo, se estivermos na branch `adiciona-botao-login` e quisermos mesclar o código desta branch com o código desenvolvido na branch `main`, podemos executar o comando `merge` desta forma:

```
git merge main
```

Desta forma, as alterações da branch `main` serão mescladas com a branch atual `adiciona-botao-login`.

Feito isso, podemos atualizar o servidor remoto com o comando `git push`:

```
git push {nome servidor remoto} {branch}
```

Por padrão o servidor remoto se chamada `origin`.

Portanto, se quisermos atualizar a branch remota `adiciona-botao-login` podemos executar o comando desta forma:

```
git push origin adiciona-botao-login
```

## Conclusão

Nesse post você aprendeu como utilizar os principais comandos do dia a dia de trabalho com o Git!

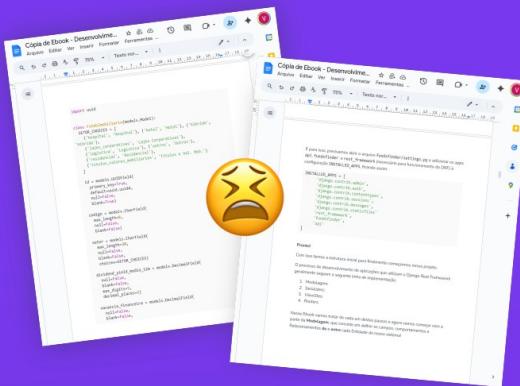
Saber utilizar o Git é crucial para sua carreira de Pythonista de sucesso!

Se ficou com alguma dúvida, fique à vontade para deixar um comentário no box aqui embaixo! Será um prazer te responder! 😊



# Crie Ebooks técnicos em minutos com IA

Conheça a 1ª IA Especializada na criação de Ebooks **com código!**



Chega de formatar código no Google Docs



**Arquitetura de Software Moderna**

A arquitetura de software alvo é profissional contendo o e-mail e produções de software para arquiteturas modernas. Oferece recursos como interface gráfica com interface de usuário.

```
import python
import python

class Arquitetura_de_Software_Moderna:
    ...
    def share(self):
        pass
    ...
    return "Arquitetura de NeXt", "arquitetura_moderna"
    ...

    def __init__(self):
        if user_authenticated():
            self.user_authenticated = user_authenticated()
            self.user_email = user_email()
            self.user_name = user_name()
        ...
        # Envie AI para gerar um código
        return type
    ...
    return self

    
```

**AI-generated system**

A arquitetura com propósito alvo é software amigável de usuários modernos. Seus recursos incluem conceitos de interface de usuário moderna, interface gráfica com interface de usuário.

**Clean layout**

Garantimos que o documento é organizado e fácil de ler. O layout é gerado automaticamente, criando um design visualmente agradável.



Infográficos feitos por IA

Syntax Highlight

Adicione Banners Promocionais

Deixe que nossa IA faça o trabalho pesado

Edite em Markdown em Tempo Real

**TESTE AGORA**



PRIMEIRO CAPÍTULO 100% GRÁTIS