GIT GUIA PARA INICIANTES >_

Commits, branches e fluxo de trabalho — passo a passo, com exemplos

Prof. Gustavo Franz (Science/Biology)
Python Developer in Progress
Building Educational Solutions with Python

GitHub: github.com/GustaFranz

O que e Git?

Git e um **sistema de controle de versao**. Ele registra o historico do seu projeto, permite trabalhar em equipe, experimentar sem medo e voltar atras quando necessario.

- ✓ Salva o historico do projeto (commits).
- ✓ Permite trabalhar em varias linhas (branches).
- ✓ Facilita colaboracao e revisao de codigo.
- ✓ Protege contra perda de codigo.
- ✓ Da liberdade para testar sem medo de errar.

O fluxo basico do Git

Arquivo alterado -> (git add) -> **Stage** -> (git commit) -> **Repositorio local** -> (git push) -> **GitHub (remoto)**.

- ✓ Voce altera arquivos no projeto.
- ✓ **git add** coloca no stage (voce escolhe o que sera salvo).
- ✓ **git commit** salva no historico.
- ✓ **git push** envia para o GitHub.

1. Stage (area de preparacao)

O stage e uma area intermediaria entre suas alteracoes e o commit. Voce escolhe exatamente o que entra no proximo commit.

Comando	O que faz	Exemplo
`git add arquivo.py`	Coloca um arquivo especifico no stage.	`git add app.py`
`git add .`	Coloca todos os arquivos no stage.	`git add .`
`git status`	Mostra o que mudou e o que esta no stage.	`git status`
`git restore --staged x`	Tira do stage sem apagar as alteracoes.	`git restore --staged app.py`

Dica

Sempre use git status antes de commitar. Ele mostra exatamente o que esta pronto para ser salvo.

2. Commits (salvando alteracoes)

O commit e um 'ponto de salvamento' no historico. Cada commit deve ter uma mensagem clara sobre o que foi feito.

Comando	O que faz	Exemplo
`git commit -m "msg"`	Salva as mudancas do stage no historico.	`git commit -m "cria login"`
`git log`	Mostra o historico de commits.	`git log`
`git log --oneline`	Mostra o historico resumido (1 linha).	`git log --oneline`
`git diff`	Mostra o que foi alterado e ainda nao adicionado.	`git diff`

Exemplo de historico (git log --oneline)

```
1ab32cd  cria sistema de login
f4d9ab6  corrige botao de acesso
98f736e  ajusta layout da tela inicial
```

3. Branches (linhas de trabalho)

Branch e uma linha paralela de desenvolvimento. Permite criar funcionalidades sem mexer na branch principal (geralmente main).

Comando	O que faz	Exemplo
`git branch`	Lista todas as branches.	`git branch`
`git branch nome`	Cria uma nova branch.	`git branch tela-login`
`git checkout nome`	Muda para outra branch.	`git checkout tela-login`
`git checkout -b nome`	Cria e ja muda para a nova branch.	`git checkout -b pagamento`
`git switch nome`	Muda de branch (alternativa ao checkout).	`git switch main`
`git switch -c nome`	Cria e ja muda (alternativa).	`git switch -c nova-feature`

4. Mesclando (merge)

Quando a funcionalidade da branch estiver pronta, juntamos (merge) na branch principal.

Comando	O que faz	Exemplo
`git checkout main`	Vai para a branch principal.	`git checkout main`
`git merge nome`	Mescla a branch informada na atual.	`git merge tela-login`
`git push`	Envia as mudancas para o GitHub.	`git push`

Dica

Sempre teste a branch antes do merge. Depois, voce pode excluir a branch que nao precisa mais.

5. Envio e atualizacao (GitHub)

Comando	O que faz	Exemplo
`git push -u origin nome`	Envia a branch pela primeira vez.	`git push -u origin feature`
`git push`	Envia as mudancas da branch atual.	`git push`
`git pull`	Baixa e aplica as alteracoes do GitHub.	`git pull`
`git fetch`	Baixa as alteracoes sem aplicar (so informa).	`git fetch`

Dica

Sempre de git pull antes de comecar a trabalhar, para evitar conflitos.

6. Fluxo de trabalho recomendado

Siga esta ordem sempre que for criar uma nova funcionalidade:

Ordem dos comandos

```
git pull  # atualiza a branch principal
git checkout -b nova-funcionalidade  # cria e entra na branch
# ... faca suas alteracoes no codigo ...
git add .  # coloca tudo no stage
git commit -m "mensagem clara"  # salva no historico
git push -u origin nova-funcionalidade  # envia para o GitHub
# abra um Pull Request, faca o merge na main e exclua a branch
```

Conceitos importantes

Termo	Significado
`Repositorio`	Projeto onde o codigo e gerenciado.
`Commit`	Salvamento de um ponto da alteracao.
`Branch`	Linha paralela de desenvolvimento.
`Merge`	Juncao de uma branch em outra.
`Stage`	Area de preparacao do commit.
`Push`	Enviar alteracoes para o GitHub.
`Pull`	Baixar alteracoes do GitHub.
`Remote`	Repositorio na nuvem (ex.: GitHub).

Comandos de emergencia (use com cuidado)

Comando	O que faz	Exemplo
`git restore arquivo`	Descarta alteracoes (volta ao ultimo commit).	`git restore app.py`

Comando	O que faz	Exemplo
`git reset --soft HEAD~1`	Desfaz o ultimo commit, mantendo as alteracoes.	`git reset --soft HEAD~1`
`git reset --hard HEAD~1`	Desfaz o commit e APAGA as alteracoes (perigoso).	`git reset --hard HEAD~1`

Resumo rapido + boas praticas

- ✓ Comandos do dia a dia: status, add ., commit -m, push, pull.
- ✓ Faca commits pequenos e com mensagens claras e objetivas.
- ✓ Use uma branch para cada funcionalidade ou correcao.
- ✓ Sempre teste antes do merge e mantenha a main estavel.
- ✓ Pratique e erre sem medo: com o tempo, vira natural.

Prof. Gustavo Franz (Science/Biology)
Python Developer in Progress
Building Educational Solutions with Python

Professor de Ciencias e Biologia desde 2013 — atualmente estudando Programacao.

Eu crio estes resumos para organizar os assuntos e estudar de forma clara e objetiva. Estou compartilhando porque eles tambem podem ajudar outros desenvolvedores que estao comecando. Bons estudos — vamos aprender juntos!

GitHub: github.com/GustaFranz

Exercicios Python: github.com/GustaFranz/exercicios_python
Para quem quiser ver a resolucao dos meus exercicios em Python.

EasyAnsi: github.com/GustaFranz/easyansi
Para quem quiser codificar personalizando com cores e estilos de forma mais facil, pratica e intuitiva.