

# GIT E GITHUB



## UM GUIA BÁSICO E DESCOMPLICADO

Aldo Tovo Neto

# Introdução

Este guia foi desenvolvido como projeto para o bootcamp “Caixa – IA Generativa com Microsoft Copilot”.

O Git é um sistema de controle de versão amplamente utilizado para gerenciar o código-fonte de projetos de software. Já o GitHub é uma plataforma baseada na nuvem que permite hospedar repositórios Git, facilitando a colaboração entre desenvolvedores. Este eBook apresenta os principais comandos básicos do Git, juntamente com exemplos práticos de uso.



01

# CONFIGURAÇÕES

---

# Configuração Inicial do Git

Antes de começar a usar o Git, é importante configurá-lo com seu nome e e-mail. Esses dados serão usados para identificar as alterações feitas por você.

```
1  # Configurar o nome do usuário
2  git config --global user.name "Seu Nome"
3
4  # Configurar o e-mail do usuário
5  git config --global user.email "seuemail@example.com"
6
7  # Verificar as configurações do Git
8  git config --list
```

# 02

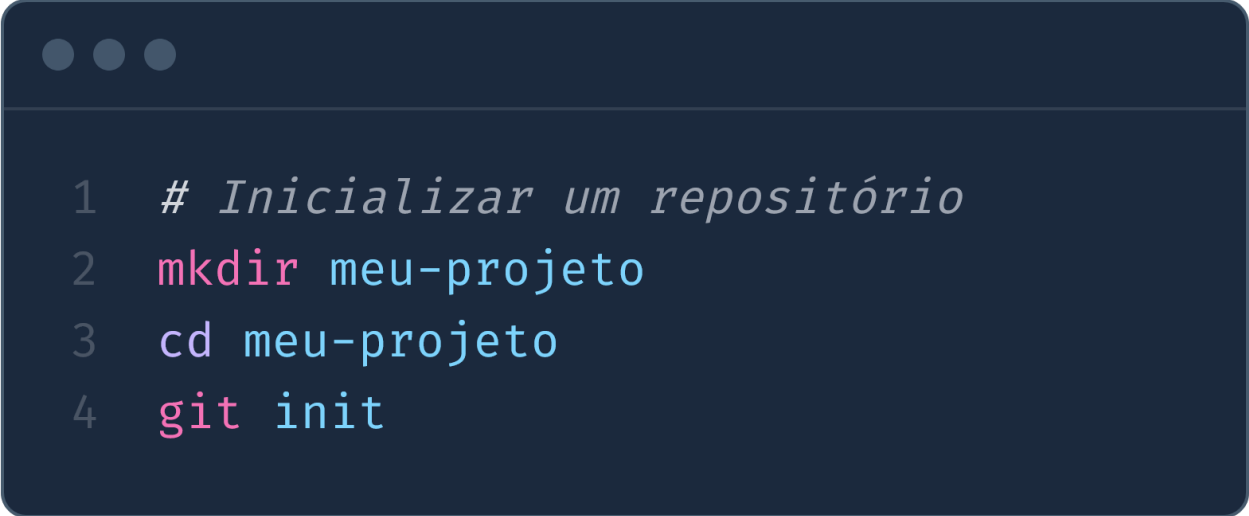
## COMANDOS

---

# Principais comandos do Git

## 2.1 - Inicializar um Repositório


O comando `git init` cria um novo repositório Git no diretório atual.

A terminal window with a dark blue background and three light blue window control buttons in the top-left corner. It contains four lines of text: a comment, a directory creation command, a directory change command, and the Git initialization command.

```
1  # Inicializar um repositório
2  mkdir meu-projeto
3  cd meu-projeto
4  git init
```

## 2.2 - Clonar um Repositório Existente

Use `git clone` para copiar um repositório remoto para sua máquina local.

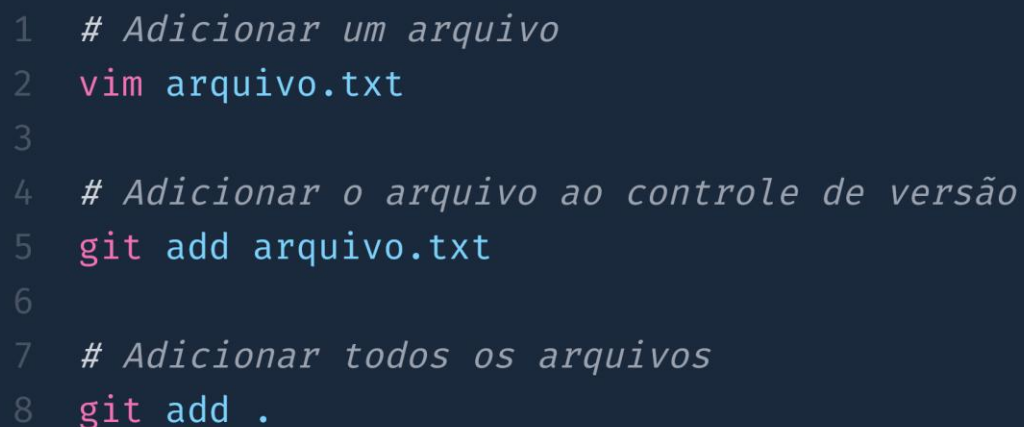
A terminal window with a dark blue background and three light blue window control buttons in the top-left corner. It contains two lines of text: a comment and the Git clone command with a placeholder URL.

```
1  # Clonar um repositório remoto
2  git clone https://github.com/usuario/repo.git
```

# Principais comandos do Git

## 2.3 - Adicionar Arquivos ao Controle de Versão


O comando `git add` adiciona arquivos ao índice (*staging area*).



```
1  # Adicionar um arquivo
2  vim arquivo.txt
3
4  # Adicionar o arquivo ao controle de versão
5  git add arquivo.txt
6
7  # Adicionar todos os arquivos
8  git add .
```

## 2.4 - Criar um Commit

O comando `git commit` salva as alterações no histórico do repositório.

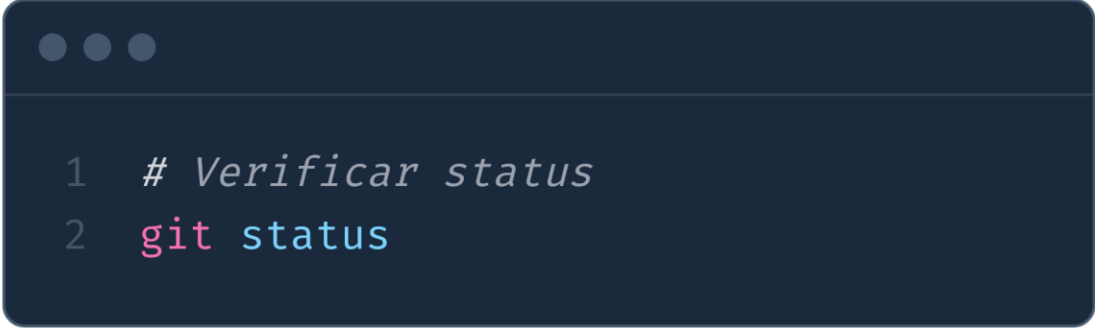


```
1  # Criar um commit
2  git commit -m "Mensagem do commit"
```

# Principais comandos do Git

## 2.5 - Verificar o Status do Repositório

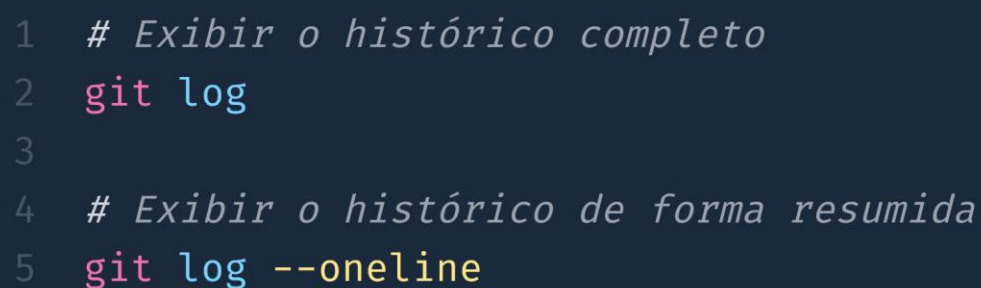
Use `git status` para visualizar o estado atual do repositório.

A terminal window with a dark blue background and three light blue window control buttons in the top-left corner. It contains two lines of text: a comment and a command.

```
1  # Verificar status
2  git status
```

## 2.6 - Visualizar o Histórico de Commits

Com o comando `git log`, você pode verificar o histórico de commits.

A terminal window with a dark blue background and three light blue window control buttons in the top-left corner. It contains five lines of text: two comments and three commands.

```
1  # Exibir o histórico completo
2  git log
3
4  # Exibir o histórico de forma resumida
5  git log --oneline
```



# Principais comandos do Git

## 2.7 - Sincronizar com um Repositório Remoto

Os comandos `git push` e `git pull` permitem enviar e buscar alterações entre o repositório local e o remoto.

```
1  # Enviar alterações para o repositório remoto
2  git push origin main
3
4  # Buscar alterações do repositório remoto
5  git pull origin main
```

## 2.8 - Criar e Alternar Entre Branches

Branches permitem trabalhar em diferentes funcionalidades ou correções sem afetar a branch principal.

```
1  # Criar uma nova branch
2  git branch minha-branch
3
4  # Alternar para a nova branch
5  git checkout minha-branch
6
7  # Criar e alternar em um único comando
8  git checkout -b minha-branch
```

# Principais comandos do Git

## 2.9 - Mesclar Branches

Use `git merge` para unir alterações de uma branch em outra.

```
1  # Alternar para a branch principal
2  git checkout main
3
4  # Mesclar outra branch na branch atual
5  git merge minha-branch
```

## 2.10 - Resolver Conflitos

Quando há conflitos durante um merge, o Git marca os arquivos afetados para edição manual. Depois de resolver os conflitos, use os comandos abaixo:

```
1  # Adicionar os arquivos corrigidos
2  git add arquivo.txt
3
4  # Finalizar o merge
5  git commit -m "Resolver conflitos e finalizar merge"
```

# Principais comandos do Git

## 2.11 - Remover Arquivos

Você pode remover arquivos do controle de versão com `git rm`.

```
1  # Remover um arquivo e adicioná-lo ao próximo commit
2  git rm arquivo.txt
```

## 2.12 - Ignorar Arquivos

Crie um arquivo `.gitignore` para especificar quais arquivos ou pastas devem ser ignorados pelo Git.

Exemplo de um `.gitignore`:

```
1  # Ignorar arquivos temporários
2  *.log
3  *.tmp
4
5  # Ignorar a pasta build
6  build/
```

03

# TRABALHANDO COM O GITHUB

---

# Trabalhando com o GitHub

## 3.1 - Criar um Repositório Remoto

No GitHub, você pode criar um novo repositório diretamente pela interface da plataforma.

Basta acessar: `<https://github.com/>`

## 3.2 - Conectar o Repositório Local ao Remoto

Use o comando `git remote` para conectar seu repositório local ao GitHub.



```
1  # Adicionar um repositório remoto
2  git remote add origin https://github.com/usuario/repo.git
3
4  # Verificar os repositórios remotos configurados
5  git remote -v
```

# Trabalhando com o GitHub

## 3.3 - Trabalhar com Pull Requests

No GitHub, Pull Requests (PRs) são uma maneira de propor mudanças em um projeto. Você pode usar um PR para mostrar suas alterações, pedir revisões ou discutir ideias com outros colaboradores.

Mas, como isso funciona na prática?

### 3.3.1 - Crie uma branch nova para sua tarefa

Sempre que começar algo novo, crie uma branch. Por exemplo, se você está corrigindo um bug:

A terminal window with a dark blue background and three light blue window control buttons in the top left corner. It contains a single line of text: "1 git checkout -b correcao-bug". The text is color-coded: "1" is light blue, "git" is pink, "checkout" is light blue, "-b" is yellow, and "correcao-bug" is light blue.

```
1 git checkout -b correcao-bug
```

# Trabalhando com o GitHub

## 3.1.2 - Faça suas alterações

Edite os arquivos necessários e salve as mudanças. Quando terminar, adicione essas alterações ao Git:

A terminal window with a dark blue background and three light blue window control buttons (minimize, maximize, close) in the top-left corner. It contains two lines of text: a line number '1' followed by the command 'git add .' and a line number '2' followed by the command 'git commit -m "Corrige bug no sistema de login"'.

```
1  git add .  
2  git commit -m "Corrige bug no sistema de login"
```

## 3.1.3 - Envie sua branch para o GitHub

Use o comando abaixo para subir sua branch:

A terminal window with a dark blue background and three light blue window control buttons (minimize, maximize, close) in the top-left corner. It contains one line of text: a line number '1' followed by the command 'git push origin correcao-bug'.

```
1  git push origin correcao-bug
```

# Trabalhando com o GitHub

## 3.1.4 - Crie o Pull Request no GitHub

- Vá até o repositório no GitHub.
- Clique na aba "Pull Requests" e depois em "New Pull Request".
- Selecione sua branch (no exemplo, "correcao-bug") e a branch principal (geralmente "main") para comparação.
- Descreva suas mudanças de forma simples e envie o PR.





# Trabalhando com o GitHub

## 3.1.5 - Revisão e Aprovação

Outros membros do projeto podem revisar o código, deixar comentários ou solicitar ajustes. Quando estiver tudo certo, o PR pode ser aprovado e "mesclado" ao projeto principal.

## 3.1.6 - Dica Extra

Após o merge, limpe suas branches antigas:

A dark-themed terminal window with three window control buttons (red, yellow, green) in the top-left corner. It contains a single line of text: 

```
1  git branch -d correcao-bug
```

```
1  git branch -d correcao-bug
```

Com o tempo, trabalhar com PRs ficará mais natural. Eles ajudam a organizar e revisar o trabalho, garantindo que todos os colaboradores estejam alinhados.

# 04

## CONCLUSÃO

---

# Conclusão

Este guia cobriu os principais comandos básicos do Git e como usá-los em conjunto com o GitHub. Com a prática, você se familiarizará com o fluxo de trabalho e poderá aproveitar ao máximo essa poderosa ferramenta de controle de versão



# Agradecimentos

Obrigado por dedicar seu tempo para aprender com este eBook. Esperamos que ele tenha sido útil em sua jornada no Git e GitHub. Boa codificação!

