

Dia 2 - Comandos Básicos do Git

Teoria (30min)

O Fluxo Básico do Git: add → commit → push

O Git trabalha com três áreas principais:

1. **Working Directory (Diretório de Trabalho)** - Onde você edita seus arquivos
2. **Staging Area (Área de Preparação)** - Onde você prepara as mudanças para commit
3. **Repository (Repositório)** - Onde ficam armazenados os commits

Fluxo de Trabalho Detalhado:

Working Directory → [git add] → Staging Area → [git commit] → Local Repository → [git push] → Remote Repository

1. Working Directory (Diretório de Trabalho)

- **O que é:** A pasta onde você trabalha normalmente, editando arquivos
- **Analogia:** É como sua mesa de trabalho onde você escreve e edita documentos
- **Estado dos arquivos:** Modificados, mas ainda não "preparados" para serem salvos
- **Exemplo:** Você edita `index.html` no VS Code

```
bash
# Você modifica arquivos aqui
# Git detecta: "Hmm, algo mudou!"
git status # Mostra: "modified: index.html"
```

2. Staging Area (Área de Preparação)

- **O que é:** Uma área intermediária onde você "prepara" as mudanças antes de salvá-las definitivamente
- **Analogia:** É como uma bandeja onde você coloca documentos que quer enviar pelo correio
- **Estado dos arquivos:** "Staged" - prontos para commit, mas ainda não commitados
- **Comando:** `git add nome-arquivo` ou `git add .`

```
bash
git add index.html # Move index.html para staging area
git status # Mostra: "Changes to be committed: modified: index.html"
```

Por que existe essa etapa?

- Permite escolher **exatamente** quais mudanças você quer salvar
- Você pode ter editado 5 arquivos, mas commitar apenas 2
- Dá controle fino sobre o que vai ser versionado

3. Local Repository (Repositório Local)

- **O que é:** O histórico completo do projeto na sua máquina
- **Analogia:** É como um arquivo permanente onde ficam todas as versões dos seus documentos
- **Estado dos arquivos:** Commitados - salvos como uma "foto" do projeto naquele momento
- **Comando:** `git commit -m "mensagem"`

bash

```
git commit -m "Adiciona navbar ao site"
# Cria um "snapshot" permanente das mudanças
# Gera um ID único (hash) para este commit
```

4. Remote Repository (Repositório Remoto - GitHub)

- **O que é:** Uma cópia do projeto armazenada na nuvem (GitHub, GitLab, etc.)
- **Analogia:** É como enviar uma cópia dos documentos para um cofre na nuvem
- **Estado dos arquivos:** Disponíveis para outros colaboradores
- **Comando:** `git push origin main`

bash

```
git push origin main
# Envia todos os commits locais para o GitHub
# Outros podem ver e baixar suas mudanças
```

Exemplo Prático Completo:

Situação: Você quer adicionar um botão no seu site

bash

1. *WORKING DIRECTORY* - Você edita o arquivo

Abre `index.html` e adiciona: `<button>Clique aqui</button>`

2. *Verifica o que mudou*

`git status`

Output: *modified: index.html (arquivo está no working directory)*

3. *STAGING AREA* - Prepara as mudanças

`git add index.html`

`git status`

Output: *Changes to be committed: modified: index.html*

4. *LOCAL REPOSITORY* - Salva definitivamente

`git commit -m "Adiciona botão de call-to-action"`

`git status`

Output: *nothing to commit, working tree clean*

5. *REMOTE REPOSITORY* - Envia para a nuvem

`git push origin main`

Suas mudanças agora estão no GitHub!

😞 Por que esse fluxo existe?

1. **Controle:** Você decide exatamente o que commitar
2. **Organização:** Pode agrupar mudanças relacionadas em um commit
3. **Segurança:** Mudanças ficam salvas localmente antes de ir para a nuvem
4. **Colaboração:** Outros podem ver seu trabalho quando você faz push

📁 Estados dos Arquivos:

Untracked... → Arquivo novo que o Git não conhece

Modified... → Arquivo conhecido que foi alterado (Working Directory)

Staged... → Arquivo preparado para commit (Staging Area)

Committed... → Arquivo salvo no repositório local

Pushed... → Arquivo enviado para repositório remoto

1. `git add` - Adicionando arquivos ao staging

- **Propósito:** Move arquivos do working directory para a staging area
- **Sintaxe básica:**

bash

```
git add nome-do-arquivo.txt # Adiciona um arquivo específico
git add . # Adiciona todos os arquivos modificados
git add *.js # Adiciona todos os arquivos .js
```

2. `git commit` - Criando um snapshot

- **Propósito:** Salva as mudanças da staging area no repositório local
- **Sintaxe básica:**

```
bash

git commit -m "Mensagem descritiva"
git commit -am "Adiciona e commita arquivos já rastreados"
```

3. `git push` - Enviando para o repositório remoto

- **Propósito:** Envia commits do repositório local para o remoto
- **Sintaxe básica:**

```
bash

git push origin main # Push para a branch main
git push # Push para branch atual (se configurada)
```

Comandos de Verificação:

- `git status` - Mostra o estado atual dos arquivos
- `git log` - Histórico de commits
- `git diff` - Diferenças entre working directory e staging area

Prática (1h)

Exercício 1: Configuração Inicial (10min)

1. Navegue até seu repositório:

```
bash

cd caminho/para/seu/repositorio
```

2. Verifique o status atual:

```
bash

git status
```

Exercício 2: Modificando o README.md (15min)

1. **Abra o README.md** no seu editor de texto favorito
2. **Adicione o seguinte conteúdo** (ou personalize):

```
markdown
```

```
# Meu Projeto de Aprendizado Git
```

```
Este repositório documenta minha jornada aprendendo Git e GitHub.
```

```
## Progresso
```

-  Dia 1: Conceitos fundamentais
-  Dia 2: Comandos básicos (em andamento)

```
## Tecnologias
```

- Git
- GitHub
- Markdown

```
## Autor
```

```
[Seu Nome]
```

3. **Salve o arquivo**

Exercício 3: Praticando o Fluxo add → commit → push (20min)

1. **Verifique as mudanças:**

```
bash
```

```
git status
```

```
git diff
```

2. **Adicione o arquivo modificado:**

```
bash
```

```
git add README.md
```

3. **Verifique o status novamente:**

```
bash
```

```
git status
```

4. **Faça o commit:**

```
bash
```

```
git commit -m "Atualiza README.md com informações do projeto"
```

5. **Envie para o repositório remoto:**

```
bash
```

```
git push origin main
```

Exercício 4: Criando o arquivo de documentação (15min)

1. Crie o arquivo `dia-02-comandos-git.md`:

```
bash
```

```
touch dia-02-comandos-git.md
```

2. Abra o arquivo e adicione sua documentação:

```
markdown
```

```
# Dia 02 - Comandos Básicos do Git
```

```
## O que aprendi hoje:
```

```
#### Comandos principais:
```

- ``git add``: Move arquivos para staging area
- ``git commit``: Cria um snapshot das mudanças
- ``git push``: Envia commits para repositório remoto

```
#### Fluxo de trabalho:
```

1. Edito arquivos no working directory
2. Uso ``git add`` para preparar mudanças
3. Uso ``git commit`` para salvar mudanças
4. Uso ``git push`` para enviar ao GitHub

```
#### Comandos úteis para verificação:
```

- ``git status``: Ver estado atual
- ``git diff``: Ver diferenças
- ``git log``: Ver histórico

```
## Exercícios realizados:
```

- [x] Modifiquei README.md
- [x] Pratiquei fluxo add → commit → push
- [x] Criei este arquivo de documentação

```
## Dúvidas/Observações:
```

- [Anote aqui suas dúvidas ou observações]

```
## Próximos passos:
```

- Dia 3: Trabalhando com branches

3. Salve o arquivo

4. Pratique o fluxo novamente:

```
bash
```

```
git add dia-02-comandos-git.md
```

```
git commit -m "Adiciona documentação do Dia 2"
```

```
git push origin main
```

Desafio Extra



Pratique mais algumas vezes o fluxo básico:

1. **Faça uma pequena alteração no README.md**
2. Use `git status` para ver as mudanças
3. Use `git diff` para ver exatamente o que mudou
4. **Adicione, commite e faça push da mudança**

Checklist do Dia 2

- ☐ Entendi o fluxo add → commit → push
- ☐ Pratiquei `git add` com diferentes opções
- ☐ Criei commits com mensagens descritivas
- ☐ Fiz push das mudanças para o GitHub
- ☐ Modifiquei o README.md
- ☐ Criei o arquivo `dia-02-comandos-git.md`
- ☐ Documentei meu aprendizado

Dicas Importantes

1. **Mensagens de commit:** Sejam descritivas e no presente do indicativo
 -  "Adiciona funcionalidade de login"
 -  "mudanças"
2. **Frequência de commits:** Faça commits pequenos e frequentes
3. **Sempre verifique:** Use `git status` antes e depois dos comandos
4. **Versionamento:** Cada commit é uma versão do seu projeto

Recursos Adicionais

- [Documentação oficial do Git](#)
- [GitHub Guides](#)

- [Interactive Git Tutorial](#)
-

Parabéns! 🎉 Você completou o Dia 2 e já domina o fluxo básico do Git!