Git e GitHub: Guia Rápido para Iniciantes

Qual a diferença entre Git e GitHub?

Pense da seguinte forma:

- **GitHub**: É como uma rede social ou uma "hospedagem" para seus projetos. É o **site** onde seus repositórios de código ficam armazenados na nuvem, permitindo que outras pessoas vejam seu trabalho e colaborem com você.
- **Git**: É a **ferramenta** que você instala no seu computador. Ele funciona como um "super-herói" do controle de versão, permitindo que você salve "fotos" (snapshots) do seu projeto ao longo do tempo, volte para versões antigas e gerencie todas as alterações de forma organizada.

Fluxo de Trabalho e Comandos Essenciais

Aqui estão os comandos mais importantes que você usará no dia a dia.

1. Baixando um Repositório do GitHub (Clonando)

Para começar a trabalhar em um projeto que já existe no GitHub, você precisa criar uma cópia dele na sua máquina local.

```
git clone <url_do_repositorio>
```

- **O que faz?** Baixa (clona) o projeto do GitHub para o seu computador.
- **Onde pegar a URL?** No GitHub, clique no botão verde "<> Code" e copie a URL HTTPS ou SSH.

2. Verificando o Status do seu Projeto

Depois de fazer alterações nos seus arquivos (adicionar, editar ou deletar código), este é o primeiro comando que você deve usar para ver o que mudou.

`git status`

- **O que faz?** Mostra quais arquivos foram modificados, quais são novos e quais estão prontos para serem salvos no histórico (commit). É o seu "GPS" para saber o estado atual do projeto.

3. Preparando suas Alterações (Adicionando à "Área de Preparo")

Antes de salvar suas modificações, você precisa dizer ao Git exatamente *quais* alterações você quer salvar. Isso é chamado de "adicionar à área de preparo" ou *staging*.

`git add <nome_do_arquivo>`

- **O que faz?** Adiciona um arquivo específico que você modificou à área de preparo.
- **Exemplo:** git add index.html

Se você quiser adicionar **TODOS** os arquivos que foram alterados de uma vez:

`git add .`

- **O que faz?** O . é um atalho que adiciona todas as modificações atuais à área de preparo. É muito útil, mas use com cuidado para não adicionar arquivos que não deveriam ir para o repositório.

4. Salvando suas Alterações (Fazendo um "Commit")

Um "commit" é como tirar uma foto do seu projeto no estado atual e salvá-la permanentemente no histórico. Cada commit tem uma mensagem que descreve o que foi

alterado.

```
git commit -m "Sua mensagem descritiva aqui"
```

- **O que faz?** Salva as alterações que estão na "área de preparo" (as que você usou git add) no histórico local do seu projeto.
- **Boas práticas para a mensagem (-m):** Seja claro e objetivo. Ex: "Adiciona formulário de contato" ou "Corrige bug no login".

5. Enviando suas Alterações para o GitHub (Upload)

Até agora, todas as suas alterações (commits) estão salvas apenas no seu computador. Para que elas apareçam no site do GitHub, você precisa enviá-las.

```
git push
```

- **O que faz?** Faz o "upload" de todos os seus commits locais para o repositório remoto no GitHub. Agora, suas alterações estão seguras na nuvem e visíveis para outros colaboradores.

6. Atualizando seu Projeto Local (Baixando alterações)

Se você estiver trabalhando em equipe, ou se fez uma alteração em outro computador, outras pessoas podem ter enviado atualizações para o repositório no GitHub. Para garantir que seu código local esteja atualizado, você usa o comando pull.

```
git pull
```

- **O que faz?** Verifica se há novas alterações no repositório do GitHub e as baixa para o seu projeto local. É uma boa prática usar este comando antes de começar a fazer novas

alterações.

Resumo do Fluxo Principal

1. **git status**: Onde estou? O que mudou?
2. **git add .**: Quero salvar TUDO isso que mudei.
3. **git commit -m "mensagem"**: Salvei essa "foto" do projeto com uma descrição.
4. **git push**: Enviando minhas "fotos" salvas para o GitHub.
5. **git pull**: Tem algo novo no GitHub? Deixa eu baixar.