Git & Github

Oque é o git?

Git é um **sistema de controle de versão distribuído** de código aberto, criado por Linus Torvalds, o mesmo criador do Linux. Mas o que isso realmente significa?

Pense no Git como uma máquina do tempo superpoderosa para seu código. Ele rastreia e salva cada alteração que você faz em seus arquivos, criando um histórico completo do projeto. Isso permite que você volte a versões anteriores, veja quem mudou o quê e quando, e até mesmo trabalhe em diferentes "linhas do tempo" do mesmo projeto simultaneamente.

Aqui estão os pontos principais que definem o que o Git é, de forma objetiva:

- **Sistema de Controle de Versão (VCS):** Ele gerencia as mudanças em um conjunto de arquivos ao longo do tempo. É a ferramenta que permite que você rastreie, compare e reverta versões.
- **Distribuído:** Essa é a parte crucial. Diferente de sistemas mais antigos que dependem de um servidor central, no Git, cada desenvolvedor tem uma cópia completa do histórico do projeto (o "repositório") em sua própria máquina. Isso significa que você pode trabalhar offline e, depois, sincronizar suas mudanças com o repositório central. Essa característica torna o trabalho em equipe mais flexível e resiliente.
- **Código Aberto:** O código-fonte do Git é público e gratuito para usar, modificar e distribuir.

Em resumo, Git é a ferramenta essencial para quem trabalha com desenvolvimento de software. Ele é a espinha dorsal de qualquer projeto colaborativo, garantindo que o histórico do código seja seguro e que múltiplas pessoas possam trabalhar juntas de forma organizada e eficiente.

Para que serve?

O Git serve para gerenciar e controlar as **versões do seu código**. Ele é uma ferramenta fundamental para qualquer pessoa que desenvolve software, desde programadores solo até grandes equipes.

Pense no Git como um sistema que:

1. **Cria um histórico completo do seu projeto:** O Git salva cada alteração que você faz nos seus arquivos. Em vez de ter várias pastas com nomes como "projeto_final_final" ou "código_novo_2", o Git armazena todas as versões em um único lugar, permitindo que você navegue pelo tempo e veja o que mudou, quando e por quem.
2. **Permite a colaboração em equipe:** O Git facilita que várias pessoas trabalhem no mesmo projeto ao mesmo tempo sem se atrapalharem. Ele gerencia as mudanças de cada desenvolvedor e ajuda a mesclá-las em uma única versão final do código. Isso evita conflitos e garante que todos estejam trabalhando na mesma página.
3. **Garante a segurança do seu trabalho:** Com o Git, você nunca perde o seu código. Se algo der errado, você pode facilmente reverter para uma versão anterior que funcionava. Isso elimina o medo de experimentar novas ideias, já que você sempre pode voltar ao ponto de partida.
4. **Facilita a experimentação:** O Git permite que você crie "ramificações" (branches), que são como linhas do tempo paralelas do seu projeto. Você pode usar uma branch para testar uma nova funcionalidade ou corrigir um bug sem afetar a versão principal do seu código. Quando a nova funcionalidade estiver pronta, você pode mesclá-la de volta ao projeto principal.

Em resumo, o Git serve para garantir que o desenvolvimento de software seja mais **organizado, seguro e colaborativo**. Sem ele, seria quase impossível gerenciar projetos complexos ou trabalhar em equipe de forma eficiente.

Como faço para utilizar o git?

A forma de atualizar o Git varia um pouco dependendo do seu sistema operacional. A maneira mais confiável e recomendada é usar o método de instalação nativo de cada sistema.

No Linux

A melhor forma de atualizar o Git no Linux é usando o gerenciador de pacotes da sua distribuição. Isso garante que a atualização seja segura e compatível com o resto do seu sistema.

- **Para distribuições baseadas em Debian/Ubuntu:**
  Abra o terminal e use os seguintes comandos:Bash

  ```
  sudo apt update
  sudo apt upgrade git
  ```

  O primeiro comando (`update`) atualiza a lista de pacotes disponíveis, e o segundo (`upgrade`) atualiza o Git para a versão mais recente que está nos repositórios do sistema.
- **Para distribuições baseadas em Fedora/CentOS/RHEL:**
  Abra o terminal e use o comando `dnf`:Bash

  ```
  sudo dnf update git
  ```
- **Para distribuições baseadas em Arch Linux:**
  Use o gerenciador de pacotes `pacman`:Bash

  ```
  sudo pacman -Syu git
  ```

No Windows

Existem algumas maneiras de atualizar o Git no Windows. A mais simples é usar a linha de comando do próprio Git.

- **Usando o Git Bash:**
  Abra o **Git Bash** (que é instalado junto com o Git for Windows) e execute o seguinte comando:Bash

  ```
  git update
  ```

  Esse comando chama o instalador do Git for Windows e o executa em modo de atualização, baixando a versão mais recente e instalando-a para você.
- **Reinstalando pelo Site Oficial:**
  Se o método acima não funcionar, a maneira mais garantida é simplesmente baixar a versão mais recente do Git no site oficial **git-scm.com/downloads** e executar o instalador. O instalador do Git for Windows detectará a versão antiga e a atualizará, mantendo todas as suas configurações.

Como faço para atualizar o git?

Para saber se o Git está instalado no seu computador, tanto no Linux quanto no Windows, você pode usar o terminal de comando. O processo é o mesmo para ambos os sistemas operacionais.

No Linux:

1. Abra o terminal (geralmente você pode encontrá-lo no menu de aplicativos ou usando um atalho como **Ctrl + Alt + T**).
2. Digite o seguinte comando e pressione **Enter**:Bash
   ```
   git --version
   ```
3. Se o Git estiver instalado, o terminal vai mostrar a versão do Git, como por exemplo:Bash
   ```
   git version 2.34.1
   ```
   Isso significa que o Git está pronto para ser usado.
4. Se o Git não estiver instalado, você verá uma mensagem de erro, algo como:Bash
   ```
   bash: git: comando não encontrado
   ```
   Nesse caso, você precisará instalá-lo.

No Windows:

1. Abra o **Prompt de Comando** ou o **PowerShell** (você pode procurar por "cmd" ou "powershell" no menu Iniciar).
2. Digite o mesmo comando e pressione **Enter**:Bash
   ```
   git --version
   ```
3. Se o Git estiver instalado, a resposta será a versão do Git, como:Bash
   ```
   git version 2.41.0.windows.1
   ```
4. Se o Git não estiver instalado, o terminal informará que o comando não é reconhecido, algo como:
   ```
   'git' não é reconhecido como um comando interno ou externo, um programa operáv
   ```

Como criar um repositório no github através do git instalado no meu computador

Primeiro vamos instalar o git

Para começar a usar o Git, você precisa seguir alguns passos básicos:

1. Inicializar um Repositório Git

A primeira coisa a fazer é transformar uma pasta do seu computador em um "repositório Git". Essa é a pasta onde o Git vai rastrear e gerenciar todas as alterações.

- Abra o terminal ou Git Bash.
- Navegue até a pasta do seu projeto. Se a pasta for `meu-projeto` na sua área de trabalho, você usaria:Bash
  `cd ~/Desktop/meu-projeto`
- Dentro da pasta, use o comando para inicializar o repositório:Bash
  `git init`
  Você verá uma mensagem como `Initialized empty Git repository in /caminho/para/o/seu/projeto/.git/`. Isso significa que o Git está pronto para trabalhar nessa pasta.

2. Adicionar os Arquivos para o "Staging Area"

O Git usa uma área intermediária chamada "staging area" (ou índice). Você precisa "adicionar" os arquivos que você modificou para essa área antes de registrar as mudanças permanentemente.

- Para adicionar um arquivo específico, por exemplo, `index.html`:Bash
  `git add index.html`
- Para adicionar todos os arquivos modificados na pasta de uma vez:Bash
  `git add .`

3. Fazer um "Commit" (Salvar as Mudanças)

O **commit** é o momento em que você salva as alterações que estão no "staging area" no histórico do seu projeto. Pense nisso como um ponto de salvamento.

- Para fazer um commit, use o seguinte comando, incluindo uma mensagem descritiva:Bash
  `git commit -m "Adiciona a página inicial do projeto"`
  É muito importante que a mensagem seja clara e resuma o que você fez nas mudanças.

4. Visualizar o Histórico

Para ver todos os commits que você fez, use o comando `log`:

- Para ver o histórico de commits:Bash

  `git log`

  Isso mostrará a lista de commits, quem os fez, a data e a mensagem. Para sair da tela de log, pressione a tecla `q`.

Resumo do Fluxo de Trabalho Básico:

1. **Crie/modifique** seus arquivos.
2. **Adicione** os arquivos modificados ao staging area com `git add .`
3. **Salve** as mudanças permanentemente com `git commit -m "sua mensagem"`.

Esse é o ciclo fundamental do Git. Repita os passos 2 e 3 sempre que você fizer um grupo de alterações que deseja salvar no histórico do projeto.

Conectar o Git ao GitHub

Para conectar um repositório Git local a um repositório online, você precisa de duas coisas: a URL do repositório remoto e o comando `git remote add`.

Passo 1: Obter a URL do Repositório Online

Primeiro, vá até o seu provedor de serviço Git (como **GitHub**, **GitLab** ou **Bitbucket**) e crie um novo repositório vazio.

Após a criação, a plataforma irá te fornecer uma URL. Ela geralmente tem um formato parecido com este:

`https://github.com/seu-usuario/nome-do-repositorio.git`

Copie essa URL, pois você irá precisar dela no próximo passo.

Passo 2: Conectar o Repositório Local ao Remoto

Abra o terminal na pasta do seu projeto local (o repositório onde você já usou o `git init`). Em seguida, use o comando `git remote add` para criar um link entre os dois repositórios.

Bash

`git remote add origin https://github.com/seu-usuario/nome-do-repositorio.git`

- `git remote add`: Este é o comando para adicionar um novo repositório remoto.

- `origin`: É um nome padrão para a sua conexão principal. Você pode usar outro nome, mas `origin` é a convenção mais comum.
- `https://...`: A URL que você copiou do seu provedor de serviço.

Passo 3: Enviar seu Código

Agora que a conexão está feita, você pode enviar seu código local para o repositório online. Para isso, use o comando `git push`.

Bash

```
git push -u origin main
```

- `u`: Cria uma ligação para que a sua branch local **main** (ou `master`) sempre se conecte à branch `main` no repositório `origin`. Depois de rodar este comando pela primeira vez, você poderá usar apenas `git push` nas próximas vezes.

Após o envio, seu repositório online no GitHub, GitLab, etc., estará atualizado com o seu código.