**O que é Git?**

O Git é um sistema de controle de versão distribuído de código aberto e gratuito utilizado pela maioria dos bons desenvolvedores, com ele é possível criar um histórico de alterações do código que está sendo escrito em um projeto e também auxilia a controlar o fluxo de alterações e novas funcionalidades quando se trabalha com vários desenvolvedores no mesmo projeto.

**Para que serve o Git?**

Através de comandos específicos do git podemos fazer o versionamento do nosso código, com isso podemos "voltar no tempo" do nosso projeto e restaurar versões anteriores, e no caso podemos ter uma cópia de segurança do nosso código que pode ficar guardado em um repositório como o **GitHub**, além disso o Git é excelente para trabalho em equipe, pois cada integrante trabalha em uma parte do projeto separadamente e depois tudo é reunido para a contrução do projeto em si.

**Como podemos utilizar o Git para versionar nossos projetos?**

Através dos **commits** principalmente, claro que existem outros diversos comandos como veremos adiante, mas os commits fazem a maior parte da mágica acontecer.

Antes de tudo, precisamos instalar o Git em nossa máquina, mas como podemos fazer isso? [**Pelo site oficial do Git**](https://git-scm.com/)**,**lá há como fazer o [**download do git**](https://git-scm.com/download/win) e em seguida a instalação em nossa máquina, é um processo bem tranquilo e simples.

Para verificar a instalação podemos abrir o terminal e executar o comando **git --version**ou algo do tipo dependendo do seu sistema operacional, mas isso é tranquilo de consultar dentro da própria página do Git.

Em seguida podemos criar uma identificação pessoal no Git para assinar nossos commits da seguinte forma:

git config --global user.name "Seu nome"

git config --global user.email "seu-email@example.com"

**Iniciando um repositório local**

Dentro de uma pasta podemos executar o comando **git init** para indicar que usaremos o Git nesse projeto, com isso será criada uma pasta oculta com nome de **.git**, o retorno desse comando será algo como o seguinte:

git init

Initialized empty Git repository in C:/nomequalquer/.git/

Agora podemos usar o comando **git add**, ele serve para que os arquivos ou pastas sejam adicionados em um **commit**, esse que por sua vez é dado pelo comando **git commit**, vejamos a seguir:

git add .

em seguida:

git commit -m "mensagem do commit"

uma mensagem parecida com a abaixo será mostrada:

[master (root-commit) 8cb43a6] mensagem do commit

1 file changed, 15 insertions(+)

create mode 100644 index.html

A partir desse ponto já fizemos um checkpoint do que escrevemos, se fizermos mudanças nos arquivos ou adicionarmos novas linhas ou arquivos vamos seguir o mesmo processor, fazer a adição com o **git add**e em seguida fazer o **commit**com **git commit**.

**O que são Branches (Galhos)?**

O sistema de versionamento Git, trabalha com uma estrutura de branches, que são ramificações do projeto em questão. Cada branch é responsável por gerir e versionar uma determinada parte ou função de um projeto, evitando problemas de superposição de código, caso um ou mais desenvolvedores estejam trabalhando em um mesmo projeto ao mesmo tempo.

Imagine que você e mais dois colegas estejam trabalhando em um mesmo projeto, caso um de vocês altere algum arquivo do projeto, terão trabalho em identificar os arquivos que foram alterados, e unificar os três projetos quando forem entregar o trabalho final, para resolver este problema, o sistema Git utiliza os branches para anotar e monitorar os arquivos que foram alterados em cada uma das ramificações do projeto, dessa forma, você e seus colegas podem trabalhar no mesmo projeto e o Git fará o trabalho de identificar quais arquivos foram alterados em cada uma das branches.

Agora que entendemos o que são branches, o proprio GitHub(próxima aula) nos da o comando para criar a branch principal:

git branch -M main

[24. Criando um Branch](https://githowto.com/pt-BR/creating_a_branch)

[Aprender como criar um branch local no repositório É hora de fazer o nosso hello world ser mais expressivo. Como isso pode demorar um pouco, é melhor mover essas modificações para um novo branch, isolando-as das modificações do branch master. Vamos nomear o nosso novo branch como "style".](https://githowto.com/pt-BR/creating_a_branch)

[[](https://githowto.com/pt-BR/creating_a_branch)https://githowto.com/pt-BR/creating\_a\_branch](https://githowto.com/pt-BR/creating_a_branch)

**Repositório Remoto x Repositório Local:**

Quando falamos de diferentes repositórios podemos ter os locais, e os remotos, cada qual tem suas vantagens e suas desvantagens, vamos abordar aqui algumas dessas qualidade de cada um dos tipos de repositórios:

* **Repositório Local:**

**Vantagens**

* 1. Dados armazenados localmente.
  2. Provavelmente já terá funcionalidades armazenadas como o node\_modules e outras dependências da aplicação.
  3. É só ligar o Computador e acessar os dados.

**Desvantagens**

* 1. Ocupa grande espaço na memória caso tenha dependências ou pastas grandes.
  2. Caso seu local de armazenamento do computador ou servidor pare de funcionar, provavelmente você irá perder seu projeto inteiro.
  3. Localmente só uma pessoa trabalha no projeto por vez
* **Repositório Remoto:**

**Vantagens**

* 1. Dados armazenados em Nuvem.
  2. Caso seu local de Armazenamento local pare de funcionar, você conseguirá acessar os dados de um projeto de outro computador.
  3. Varias pessoas conseguem acessar este projeto ao mesmo tempo.
  4. Não ocupam espaço de armazenamento local

**Desvantagens**

* 1. Caso o servidor da Nuvem estiver offline, você provavelmente não irá conseguir acessar este projeto.
  2. Você pode ter seus dados da aplicação roubados caso um Hacker invada o servidor.
  3. Erros de versionamento são mais comuns que deveriam ser, seja por uma atualização que não foi enviada ou por consumo errôneo dos dados.