

# Conceitos e Práticas de Git

#ninhodaaguia2022

# 1. Introdução

Neste ebook, você irá aprender sobre o Git e seus principais conceitos, além dos comandos mais utilizados, como o **pull**, **push**, **checkout**, **commit**, **init**, **diff**, **status** e **restore**. Ao final, você será capaz de aplicá-los na prática e usar essa poderosa ferramenta de versionamento de código.

## 2. Guia de instalação do Git

Há diversas maneiras de instalar o Git, aqui apresentamos as mais comuns nos 3 sistemas operacionais mais utilizados.

### Instalação do Git no Windows

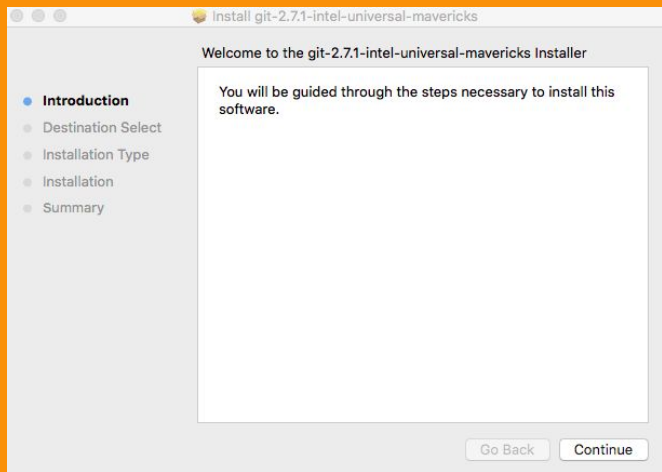
1. Acesse o site <https://gitforwindows.org/> e realize o download do instalador;
2. Realize o duplo clique no arquivo baixado e inicie o assistente de instalação;
3. Siga as instruções na tela e clique em *next*;
4. Clique em *finish* para concluir a instalação;
5. Abra o prompt de comando e digite:

```
git config - - global user.name "seu_usuario"
```

```
git config - - global user.email "seu_email"
```

## 2. Guia de instalação do Git

### Interface do instalador



### Instalação do Git no MacOS

1. Acesse o site:  
<https://sourceforge.net/projects/git-osx-installer/files/> e realize o download do instalador
2. Siga as instruções na tela e clique em *continue*
3. Verifique a instalação através do comando:  
`git -- version`
4. Execute:  
`git config - - global user.name "seu_usuario"`  
`git config - - global user.email "seu_email"`

## 2. Guia de instalação do Git

Para outras versões do Linux, acesse:

<https://bityli.com/qpOb>

### Instalação do Git no Debian/Ubuntu

1. Abra o terminal e digite:

```
sudo dnf install git
```

```
sudo yum install git
```

2. Verifique o sucesso da instalação através do comando: *git -- version*

3. Execute:

```
git config - - global user.name "seu_usuario"
```

```
git config - - global user.email "seu_email"
```

### 3. O que é o Git?

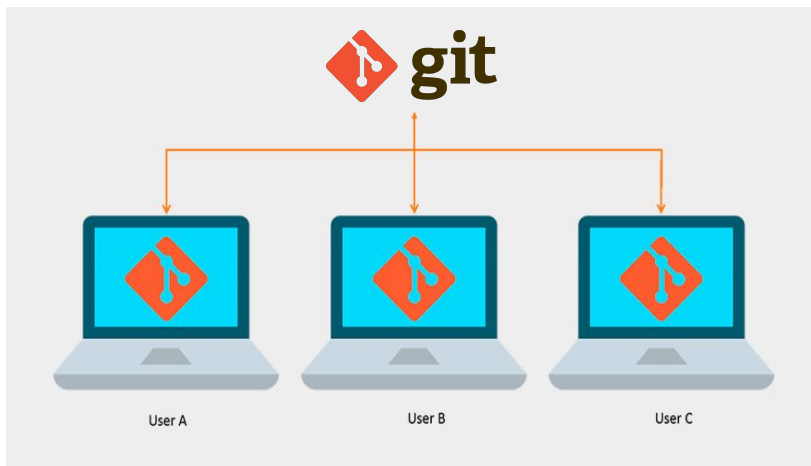
O Git é um sistema de controle de versão muito usado no desenvolvimento de softwares, mas podendo facilmente ser aplicado em outras áreas.

Ele foi projetado e desenvolvido por Linus Torvalds, com o objetivo de desenvolvimento do kernel (componente central) do Linux.

→ Também conhecido como controle de fonte, é a prática de rastrear e gerenciar as alterações em um código de software. Ele ajuda as equipes de desenvolvimento a gerenciar as alterações no código-fonte ao longo do tempo, permitindo um desenvolvimento mais rápido e inteligente.

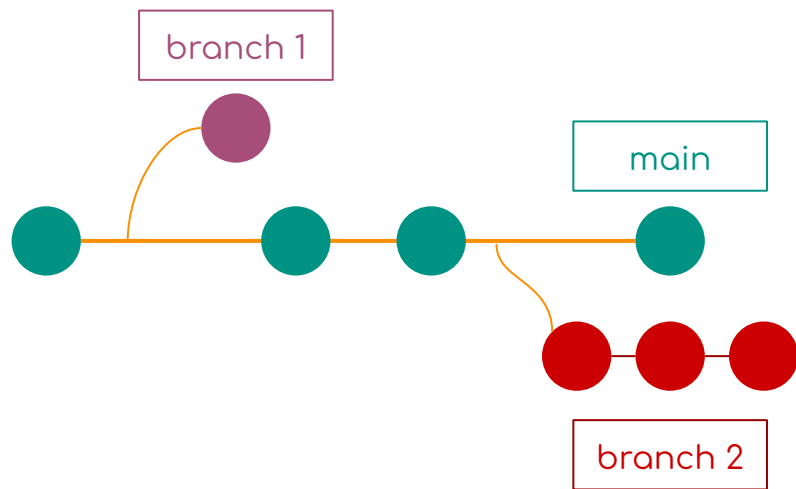


### 3. O que é o Git?



O Git mantém o registro de todas as modificações no código em um tipo especial de banco de dados. Desta forma, caso um erro seja cometido, os desenvolvedores podem “voltar no tempo” e comparar as versões anteriores do código para entender o que aconteceu e efetuar as correções necessárias.

## 4. O que são branches?



Branch significa ramo, e é exatamente isso o que elas são, ramificações do seu código. Imaginemos que você tem um software em desenvolvimento, mas que já está funcionando, e seu cliente solicita uma nova funcionalidade (que também pode ser chamada de *feature*). Você não pode alterar direto no código original. Ao invés disso, você pode criar uma ramificação, uma branch, onde você pega o estado atual daquele código e cria um novo ambiente de desenvolvimento, sem alterar ou atrapalhar o funcionamento do software original.



# 5. Principais comandos do Git

## commit

São as marcações que fazemos em um código ao longo de seu desenvolvimento. Um commit guarda as alterações, o usuário responsável por elas e uma mensagem de identificação. Cada commit possui um código único, chamado de hash (SHA1), que pode ser usado para voltar em alguma alteração específica ou algum ponto no "tempo" do código, e isso ajuda em alterações ou correções

`git commit -m "mensagem_do_usuario"`

## status

Mostra em qual status você está, como por exemplo, se existem commits a serem enviados ao repositório remoto, em qual branch está sendo realizado o trabalho e se existem arquivos não monitorados pelo Git.

`git status`

## diff

Caso o comando `status` não traga as informações que você necessita, como os arquivos que foram alterados e qual tipo de alteração, o `diff` será uma opção para verificar exatamente as linhas de código que houveram mudanças.

`git diff`

# 5. Principais comandos do Git

## pull

Incorpora as alterações de um projeto remoto na branch local. Com ele, você pega todas as alterações que vem sendo feitas e *commitadas* no git e traz para o seu computador, assim consegue continuar editando os códigos localmente.

```
git pull origin [nome_da_branch_local]
```

## push

Transfere as alterações locais para o repositório remoto. E isso só pode ser feito após o commit, onde você descreve tudo que alterou no código.

```
git push origin [nome_do_repositorio] [nome_da_branch_local]
```

## checkout


Realiza troca entre as suas branches. Caso ainda não possua uma branch secundária, você pode utilizar “-b” para criar uma.

```
git checkout -b {nome_da_branch} (para branch nova)  
git checkout master (para troca da branch secundária para principal)
```

## 5. Principais comandos do Git

### restore

Comando que serve para restaurar algum arquivo ou projeto por completo. Para isso, precisamos usar o ID do commit como valor de ponto de fonte de restauração



```
git restore - source ID_COMMIT
```

Para mais comandos no Git, consulte a documentação oficial:

<https://git-scm.com/docs>

## Autores:



Allan Quiterio



Carlos Eduardo



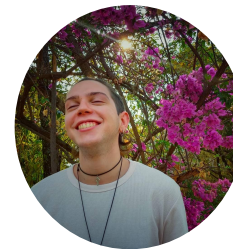
Gustavo Mendes



Pedro Florencio



Priscila Oliveira



Yan Stivaletti