



# Controle de Versão

Configuração, Utilização e Principais Comandos

## Conteúdo:

- Conceitos iniciais
- Instalação do Git
- Repositórios
- Repositório Local
- Repositório Remoto
- Enviando Códigos
- Descobrimo o que falta enviar
- Recebendo Códigos
- Lista de Comandos
- Como criar um bom README.md

## 01 Conceitos

- Vamos desenvolver um *software* de *e-Commerce* com uma equipe de 6 desenvolvedores; Como todos estarão trabalhando simultaneamente no desenvolvimento do projeto, como trabalhar em conjunto?
- Precisamos de um Sistema de Controle de Versão!
  - Atualmente, a melhor ferramenta, adotada pela ampla maioria das empresas e desenvolvedores é o **Git**.

## 02 Instalação do Git

- O primeiro passo é instalar o Git na sua estação de trabalho. Para isto, acesse o endereço <https://git-scm.com/downloads>
  - Existe versões para Windows, Linux e MacOS
- Uma vez instalado, acesse o *prompt* de comando e digite o comando **git --version**:

```
Command Prompt
(c) Microsoft Corporation. All rights reserved.
C:\Users\teste config>git --version
git version 2.35.1.windows.2
C:\Users\teste config>
```

- Funcionou? Agora precisamos configurar o seu nome e e-mail utilizando os dois comandos a seguir.
  - Por que isto é necessário? Todas as vezes que você enviar um código para o repositório, seu nome e e-mails vão ser registrados.
  - Essa configuração é realizada uma única vez.

```
git config --global user.name "Daniel Abella"
git config --global user.email daniel@daniel-abella.com
```

- Caso você necessite saber quais as configurações de nome e e-mail estão vigentes no computador, basta executar o comando apresentado a seguir.

```
git config --list
```

## 03 Repositórios e Tipos

- Um repositório contém todos os arquivos do seu projeto, bem como o histórico de revisão de cada arquivo.
- Existem dois tipos de repositórios, repositório local e repositório remoto (*remote*).
  - Na seção 04 descreveremos como utilizar um repositório *remoto*, utilizando o Github, que é um dos serviços gratuitos para repositórios com Git.
  - Na seção 05 descreveremos como utilizar um repositório local, utilizando a sua própria máquina como repositório.

## 04 Repositório Local

- Um repositório local é basicamente uma pasta onde todos os seus arquivos estarão guardados e versionados pelo Git
  - A palavra versionados, significa que, toda a evolução (isto é, o histórico) do artefato.
  - Para criar um novo repositório, o comando é o **git init**
  - O comando **git init** cria um novo repositório do Git. Ele pode ser usado para converter um projeto existente e não versionado em um repositório do Git ou inicializar um novo repositório vazio. máquina como repositório.

```
Command Prompt
C:\Users\teste config\Downloads\curso-git>git init
Initialized empty Git repository in C:/Users/teste config/Downloads/curso-git/.git/
C:\Users\teste config\Downloads\curso-git>
```

- No exemplo acima, criamos um repositório dentro da pasta chamada **curso-git**. Ou seja, antes de criar o repositório, certifique-se em qual pasta você está.

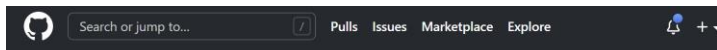
## 05 Repositório Remoto (GitHub)

- O GitHub é um dos repositórios Git remotos mais utilizados no mundo.
  - Quando uso o termo *remoto*, refiro-me ao fato que, o código fonte do seu projeto estará protegido e armazenado em um servidor remoto.
  - Concorrentes ao GitHub, temos o GitLab e BitBucket.

## 05 Repositório Remoto (GitHub)

(Continuação)

- Para utilizarmos o GitHub, acesse o endereço [www.github.com](https://www.github.com) e, o primeiro passo é criar uma conta por meio do botão **Sign up**. Caso já tenha conta, clique em **Sign in**.
- O segundo passo é criar um repositório remoto. Para isto, clique no botão **New**, que a tela a seguir será apresentada.



### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?  
[Import a repository.](#)

Owner \*  Repository name \*

Great repository names are short and memorable. Need inspiration? How about [didactic-robot?](#)

Description (optional)

☒ Public  
Anyone on the internet can see this repository. You choose who can commit.

☐ Private  
You choose who can see and commit to this repository.

- Na tela acima, temos que preencher os seguintes campos:
  - Repository Name:** nome do repositório
  - Description:** Descrição do repositório (é opcional)
  - Tipo de Repositório:** *Public* (Público) ou *Private* (Privado)
    - Um repositório *Public*, todo mundo pode visualizar o seu código fonte, porém apenas as pessoas autorizadas por você que podem contribuir
    - Um repositório *Private*, apenas pessoas autorizadas podem visualizar o código fonte, bem como contribuir
- Ao fim, clique no botão **Create repository** e seu repositório estará criado na nuvem! Agora, para passar a trabalhar neste repositório, acesse o prompt de comando e digite os 2 comandos a seguir:

#### Comandos

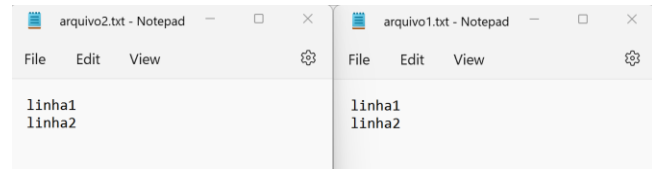
```
git clone https://github.com/daniel-abella/curso-git.git  
cd curso-git
```

- daniel-abella** se refere ao meu usuário no Github
- curso-git** se refere ao meu repositório criado no Github
- Agora, na pasta **curso-git** você terá uma cópia do repositório que está no Github. E, oportunamente, você e seus colegas podem enviar seus códigos para o seu repositório do Github.

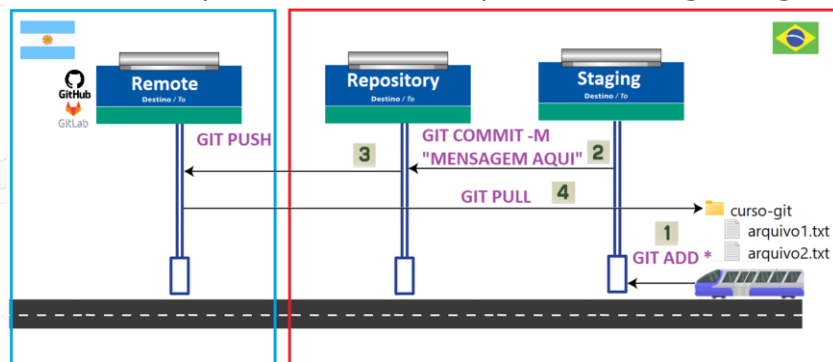
## 06 Enviando Códigos

- Se você chegou aqui, você criou seu repositório de uma das duas formas: repositório local (etapa 4) ou repositório no Github (etapa 5). Agora, apresentaremos como enviar seus códigos, bem como receber código dos seus colegas.

- Agora, para aprender como enviar seus códigos, vamos fazer uma simulação. Crie, dentro da pasta **curso-git** dois arquivos (**arquivo1.txt** e **arquivo2.txt**). Para cada um dos arquivos, vamos criar duas linhas de texto (linha1 e linha2), conforme exemplificado na imagem a seguir.



- Como enviar nossos arquivos para o repositório do Github?
- Os passos necessários estão representados na imagem a seguir.



- A analogia acima se baseia em uma viagem de metrô, na qual temos 3 paradas, descritas a seguir.
- Lembram que criamos 2 arquivos anteriormente na pasta local **curso-git**? O primeiro passo é dizer quem você quer levar no metrô, ou seja, informar qual dos 2 arquivos eu quero que sejam armazenados no repositório.
  - Para isso, pode-se fazer algum dos comandos a seguir. O primeiro deles (com \*), sinaliza que **todos** os arquivos novos ou modificados serão enviados ao repositório. Por outro lado, o segundo comando (com **arquivo1.txt**) sinaliza que, apenas o **arquivo1.txt** será enviado ao repositório.

#### Comandos para Sinalizar Códigos que Quero Enviar

```
git add *
```

```
git add arquivo1.txt
```

- O segundo passo é armazenar localmente o(s) arquivo(s) no servidor local. Para isto, executa-se o seguinte comando. A mensagem que está entre aspas detalha a sua contribuição e é muito importante que esteja bem escrita.

#### Comandos para Enviar para o Servidor LOCAL

```
git commit -m "Enviando o arquivo1.txt"
```

- Agora que o arquivo está localmente, se necessário, você pode remeter ao servidor remoto (como Github), para isto, execute o comando a seguir.

#### Comandos para Enviar para o Servidor REMOTO

```
git push
```

- Como discutimos anteriormente, o comando `git commit -m "Mensagem"` envia os códigos marcados no `git add` para o servidor LOCAL. Enquanto que, o `git push` envia ao servidor REMOTO.
- Posso enviar direto para o servidor REMOTO? Não, precisa enviar localmente (`git commit`) e depois remotamente (`git push`)
- Imagine-se em um incêndio e você precisa enviar o seu código para um servidor externo, porque em alguns minutos sua máquina pegará fogo.
- Quais os comandos necessários? Na imagem a seguir, temos um excelente exemplo de como proceder.



- Com base na imagem acima, inicialmente você faz um `git commit` para enviar o seu código para um repositório local (presume-se que você tenha feito o `git add` antes)
- Agora que enviado localmente, você pode enviar remotamente por meio do comando `git push`.
- Como o código já está nas nuvens (foi enviado para o servidor remoto), agora podemos deixar o prédio (passo 3).

## 07 Descobrindo o que falta enviar

- Como vimos, no comando `git add *` ou `git add arquivo1.txt` sinalizamos ao `git` o(s) arquivo(s) que queremos remeter ao servidor local e possivelmente ao servidor remoto.
- Entretanto, na maioria das vezes não saberemos quais o(s) arquivo(s) alteramos e estão passíveis de serem enviados. Para conhecer o(s) arquivo(s) nesta situação, execute o comando a seguir.

### Comandos p/ Descobrir o que Pode Enviar ao Servidor

`git status`

## 08 Recebendo Códigos

- Agora vamos fazer o inverso da seção 6. Imagine que, outros colegas de trabalho tenham enviado código ao repositório remoto e você queira receber estas alterações. O comando é o apresentado a seguir.

### Comandos p/ Receber Novidades do Servidor Remoto

`git pull`

## 09 Lista dos Comandos

### Comando

### Descrição

`git clone`  
<https://github.com/daniel-abella/curso-git>

Clona tudo que está no servidor remoto para minha máquina, onde: **daniel-abella** se refere ao meu usuário no Github e **curso-git** se refere ao meu repositório criado no Github

`git status`

O que falta enviar para o repositório local

`git add *`

Sinaliza que todos os arquivos novos ou modificados devem ser enviados ao repositório local

`git add arquivo1.txt`

Sinaliza que apenas o arquivo1.txt deve ser enviado ao repositório local, demais arquivos são ignorados

`git commit -m "Msg"`

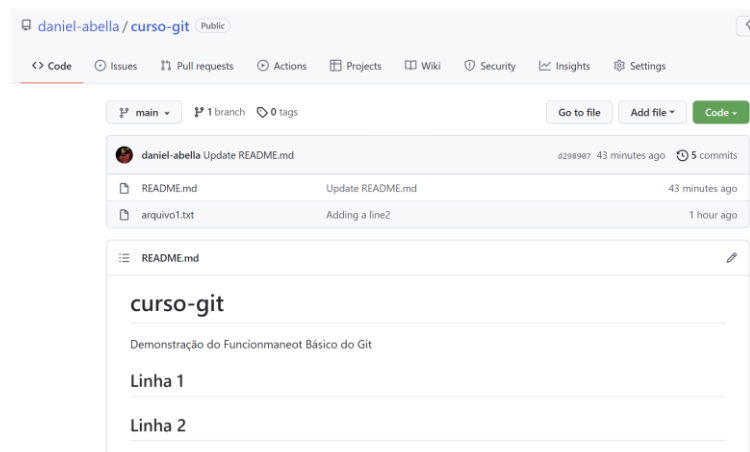
Envia para o repositório local os arquivos sinalizados e enviado a "Msg" como mensagem descrevendo

`git push`

Envia para o repositório remoto o que você tinha enviado localmente

## 10 Como criar um bom README.md

- Ao criar o repositório, sugere-se criar automaticamente o arquivo README.MD, que é um arquivo que é aberto ao acessarem o repositório. Este arquivo é como se fosse a porta de entrada do repositório e deve ser apresentável e explicar bem o projeto que está armazenado no repositório.



- O arquivo README.MD é escrito em uma linguagem chamada Markdown (por isso a extensão do arquivo é MD). Caso você queira usar um editor para criar um README bonito, acesse o endereço <https://dillinger.io/>