

Tutorial GitHub

Principais comandos do git:

git config --list » Lista as configurações do Git, se estiver dentro do repositório, lista mais itens

git config --global user.name "Meu Nome" » Define o nome de usuário para o Git

git config --global user.email "email@dominio.com" » Define o e-mail de usuário para o Git (tem de ser o cadastrado no GitHub)

git config --global core.editor vim » Define o editor de texto padrão para abrir automaticamente arquivos informados pelo Git

git init » Inicializa um repositório Git

git status » Vê o estado atual do projeto

git add arquivo.txt » Adiciona o arquivo arquivo.txt ao projeto

git commit -m "Minhas mudanças efetuadas" » Armazena as mudanças efetuadas e descreve o que foi alterado

git log » Mostra todas as mudanças que já foram efetuadas: commit, autor e data

git push -u origin master » Envia todos os arquivos modificados e “commitados” para o repositório no github

-u - faz com que o Git armazene esse comando e da próxima vez basta utilizarmos **git push**

origin- diz que o repositório no github possui o mesmo nome do projeto/diretório que você está enviando

master - é o nome da *branch*.

git pull origin master » Verifica as mudanças efetuadas por outros colaboradores do projeto

git diff HEAD » Verifica as partes dos arquivos alterados no último commit.

git reset arquivo.txt » Remove um arquivo do projeto

git checkout – arquivo.txt » Desfaz a última alteração feita num arquivo

git rm "*.txt" » Remove 1 ou mais arquivos utilizando “curinga”

git clone https://github.com/user/project.git » Copia um projeto pro seu PC

info git » Obtém a Documentação do git

man git » Obtém o Manual do git

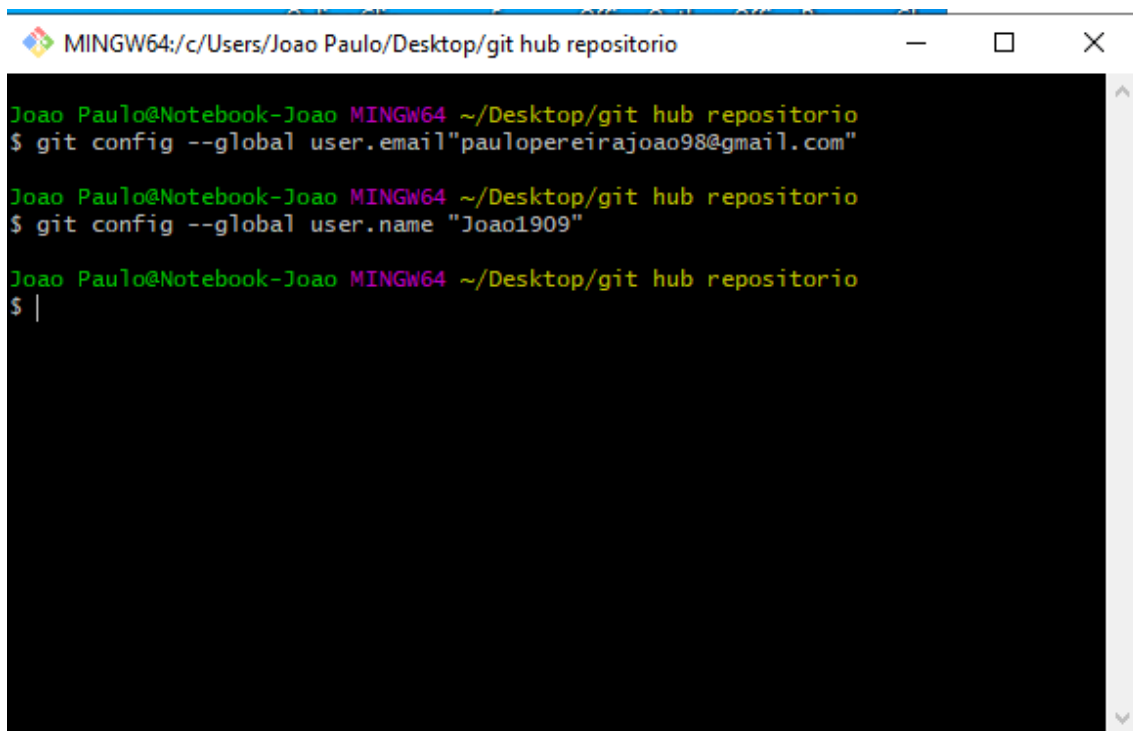
Configurando o ambiente de acesso

Deve possuir o git instalado em seu computador e uma conta no github. Para baixar o git basta ir no site oficial selecionar o sistema operacional, baixar e seguir o guia de instalação.

Após a instalação é necessário configurar seu nome e email que serão exibidos em seus commits, para isso basta digitar no terminal os comandos abaixo:

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

git config --global user.email "email@xxxx.com"

A screenshot of a terminal window titled "MINGW64: c:/Users/Joao Paulo/Desktop/git hub repositório". The terminal shows the following commands and output:

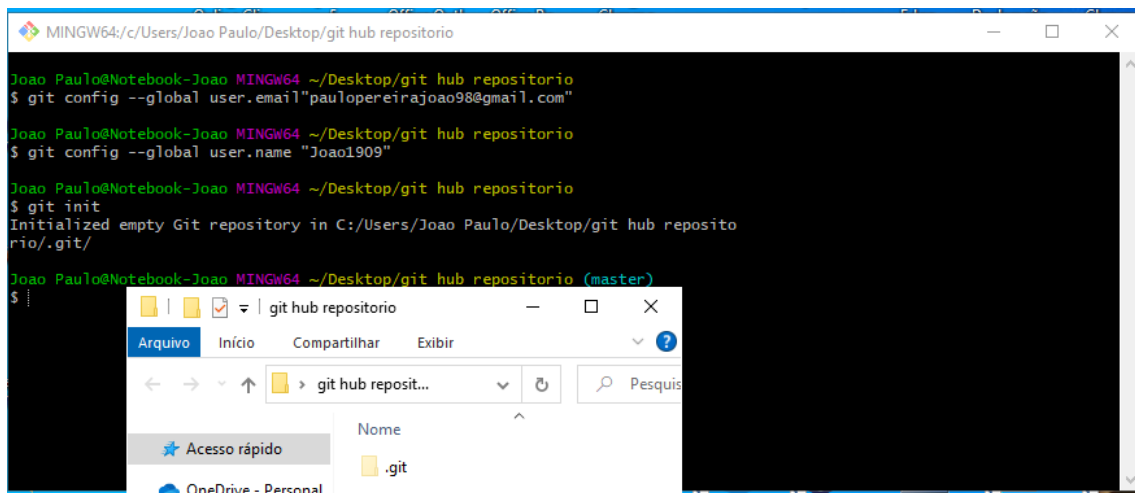
```
Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/git hub repositório
$ git config --global user.email "paulopereirajoao98@gmail.com"

Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/git hub repositório
$ git config --global user.name "Joao1909"

Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/git hub repositório
$ |
```

Criando um novo repositório

Dentro da aplicação crie uma nova pasta e utilize o comando **git init** para criar um novo repositório.

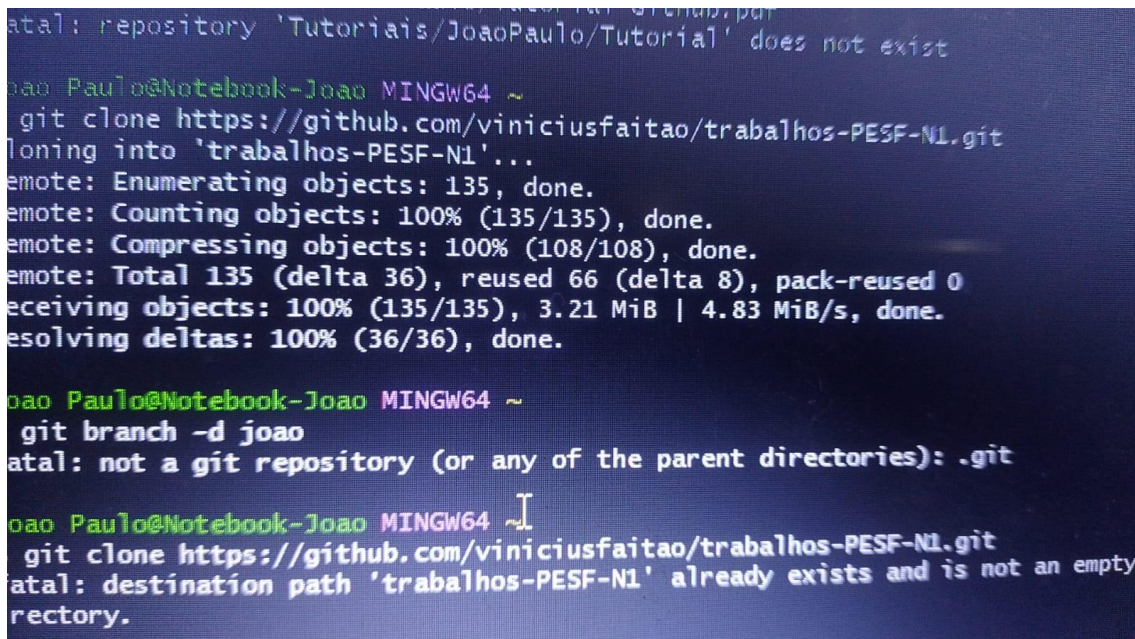


Como obter um repositório

Crie uma cópia de trabalho em um repositório local executando o comando no terminal: **git clone /caminho/do/repositório**

Caso utilize um servidor remoto utilize o comando:

git clone user@servidor /caminho/do/repositório



Fluxo de trabalho

Seus repositórios locais consistem em três "árvores" mantidas pelo git. A primeira delas é sua Working Directory que contém os arquivos vigentes. A segunda Index que funciona como uma área temporária e finalmente a terceira que é HEAD que aponta para o último commit que você fez.

Comandos adicionar & confirmar

Você pode propor mudanças (adicioná-las ao Index) usando **git add <arquivo>** ou **git add ***.

Adicionando 1 arquivo: git add “nome do arquivo”

```
Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/git hub repositorio (joao)
$ git status
On branch joao

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        txt teste.txt

nothing added to commit but untracked files present (use "git add" to track)
Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/git hub repositorio (joao)
$ git add "txt teste.txt"
Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/git hub repositorio (joao)
```

Adicionando vários arquivos: git add .

```
Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/git hub repositorio (joao)
$ git add .
Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/git hub repositorio (joao)
$ git status
On branch joao

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   teste 2.xlsx
        new file:   teste3.bmp
        new file:   txt teste.txt
```

Este é o primeiro passo no fluxo de trabalho básico do git. Para realmente confirmar estas mudanças (isto é, fazer um *commit*), use

git commit -m "comentários das alterações".

Agora o arquivo é enviado para o HEAD, mas ainda não para o repositório remoto.

```
Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/git hub repositorio (joao)
$ git commit -m "commit inicial 1"
[joao (root-commit) f8155a7] commit inicial 1
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 teste 2.xlsx
create mode 100644 teste3.bmp
create mode 100644 txt teste.txt
```

Enviando alterações para o repositório

Suas alterações agora estão no HEAD da sua cópia de trabalho local. Para enviar estas alterações ao seu repositório remoto, execute

git push origin máster.

Altere master para qualquer ramo (branch) desejado, enviando suas alterações para ele.

Git push na branch João para o repositório no github

The screenshot displays a GitHub repository interface on the left and a terminal window on the right. The GitHub page shows a commit by 'Joao1909' with files README.md, teste2.xlsx, teste3.bmp, and txt teste.txt. The terminal window shows the commands to commit and push these files to a remote repository named 'joao'.

```
Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/teste-2-github (joao)
$ git commit -m "commit 1"
[joao 610141b] commit 1
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 teste2.xlsx
create mode 100644 teste3.bmp
create mode 100644 txt teste.txt

Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/teste-2-github (joao)
$ git push joao
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 5.41 KiB | 1.35 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Joao1909/teste-2-github.git
 e01798a..610141b joao -> joao

Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/teste-2-github (joao)
$ |
```

Se você não clonou um repositório existente e quer conectar seu repositório a um servidor remoto, você deve adicioná-lo com o comando:

git remote add origin <servidor>

Adicionando a branch ao repositório

```
Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/teste-2-github (joao)
$ git remote add joao https://github.com/Joao1909/teste-2-github.git
```

Agora você é capaz de enviar suas alterações para o servidor remoto que desejar.

```
Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/git hub repositorio (joao)
$ git remote add joao https://github.com/Joao1909/git-hub-repositorio.git

Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/git hub repositorio (joao)
$ git push joao
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 5.35 KiB | 913.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'joao' on GitHub by visiting:
remote:   https://github.com/Joao1909/git-hub-repositorio/pull/new/joao
remote:
To https://github.com/Joao1909/git-hub-repositorio.git
 * [new branch]      joao -> joao
```

Como fazer Ramificando

Branches são usados para desenvolver funcionalidades isoladas umas das outras. O branch master é o branch "padrão" quando você cria um repositório. Use outros branches para desenvolver e mesclê-os ao branch master após a sua conclusão.

Crie um novo branch chamado "funcionalidade_x" e selecione-o usando **git checkout -b funci_x**, retorne para o master usando **git checkout máster** e remova o branch da seguinte forma **git branch -d funci_x** um branch não está disponível a outros a menos que você envie o branch para seu repositório remoto **git push origin <funci_x>**

```
Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/git hub repositorio (joao)
$ git remote add joao https://github.com/Joao1909/git-hub-repositorio.git

Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/git hub repositorio (joao)
$ git push joao
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 5.35 KiB | 913.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'joao' on GitHub by visiting:
remote:   https://github.com/Joao1909/git-hub-repositorio/pull/new/joao
remote:
To https://github.com/Joao1909/git-hub-repositorio.git
 * [new branch]      joao -> joao
```

Criando branch João e trocando para ela

Podemos ver que só tem a branch main, estarei criando a branch João e trocando para ela.

```
Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/teste-2-github (main)
$ git branch
* main

Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/teste-2-github (main)
$ git branch joao

Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/teste-2-github (main)
$ git checkout joao
Switched to branch 'joao'
```

Como atualizar & mesclar

Para atualizar seu repositório local com a mais nova versão, execute **git pull** na sua pasta de trabalho para *obter e fazer merge* (mesclar) alterações remotas.

Para fazer merge de um outro branch ao seu branch ativo (ex. master), use **git merge <branch>** em ambos os casos o git tenta fazer o merge das alterações automaticamente. Infelizmente, isto nem sempre é possível e resulta em *conflitos*. Você é responsável por fazer o merge destes conflitos manualmente editando os arquivos exibidos pelo git. Depois de alterar, você precisa marcá-los como merged com **git add <arquivo>** antes de fazer o merge das alterações, você pode também pré-visualizá-las usando **git diff <branch origem> <branch destino>**.

```

Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/teste-2-github (joao)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/teste-2-github (main)
$ git merge joao
Updating e01798a..610141b
Fast-forward
 teste 2.xlsx | Bin 0 -> 8081 bytes
 teste3.bmp   | 0
 txt teste.txt | 0
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 teste 2.xlsx
create mode 100644 teste3.bmp
create mode 100644 txt teste.txt

Joao Paulo@Notebook-Joao MINGW64 ~/Desktop/teste-2-github (main)
$ git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
to https://github.com/Joao1909/teste-2-github.git
 e01798a..610141b main -> main

```

Como fazer rotulação

É recomendado criar rótulos para releases de software. Este é um conhecido conceito, que também existe no SVN. Você pode criar um novo rótulo chamado **1.0.0** executando o comando **git tag 1.0.0 1b2e1d63ff** o **1b2e1d63ff** representa os 10 primeiros caracteres do id de commit que você quer referenciar com seu rótulo. Você pode obter o id de commit com **git log** você pode também usar menos caracteres do id de commit, ele somente precisa ser único.

Como sobrescrever alterações locais

No caso de você ter feito algo errado (que seguramente nunca acontece ;)) você pode sobrescrever as alterações locais usando o commando **git checkout -- <arquivo>** isto substitui as alterações na sua árvore de trabalho com o conteúdo mais recente no HEAD. Alterações já adicionadas ao index, bem como novos arquivos serão mantidos. Se ao invés disso você deseja remover todas as alterações e commits locais, recupere o histórico mais recente do servidor e aponte para seu branch master local desta forma **git fetch origin** e **git reset --hard origin/master**