

5ª EDIÇÃO

Git

Sistema de controle
de versão

Luiza
<CODE>

Luizalabs
magalu

Ementa:

1. Versionamento e Git
2. GitHub
3. Iniciando o Git
4. Clone
5. Comandos básicos git: add, commit, status, log, push, pull
6. Branch
7. Merge
8. Pull request
9. Conflitos
10. Diff
11. Stash
12. Reset, Revert, Checkout
13. Rebase
14. Tags

1. Versionamento

1. Versionamento

1.1 O que é versionamento?

O versionamento é o gerenciamento de versões diferentes de um documento de texto qualquer.

O versionamento é controlado pelo o que chamamos de **sistema de controle de versões**. Normalmente, esses sistemas são utilizados no desenvolvimento de software para controlar as diferentes versões e histórico de desenvolvimento do código.

1.2 O que é Git?

O Git é uma **ferramenta de versionamento não centralizado** muito poderosa que permite que desenvolvedores colaborem entre si de forma organizada na construção de um projeto que envolva código. Se você ainda não o conhece, não tem problema, vamos a um overview dos principais conceitos envolvidos.

Foi criado em 2005 por Linus Torvalds (criador do sistema operacional Linux).

1. Versionamento



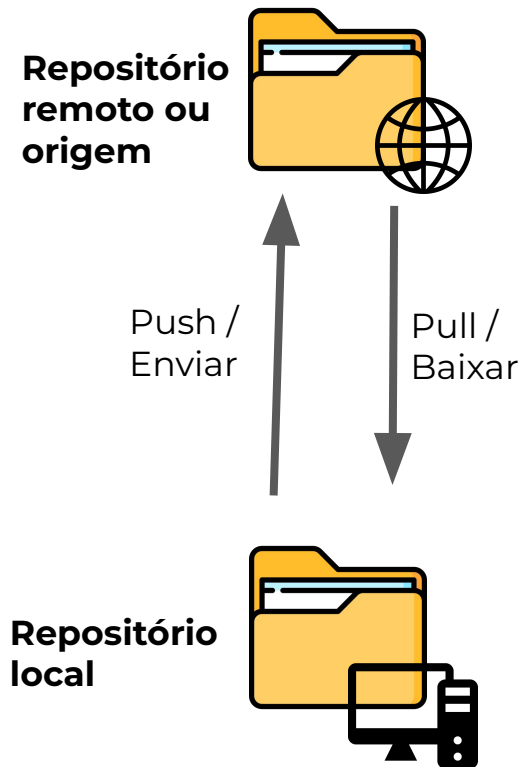
Versionamento conta uma história

- what - o que foi feito?
- why - por que foi feito?
- who - quem fez?
- when - quando foi feito?
- where - aonde foi alterado?
- how - como foi feito?

1. Versionamento

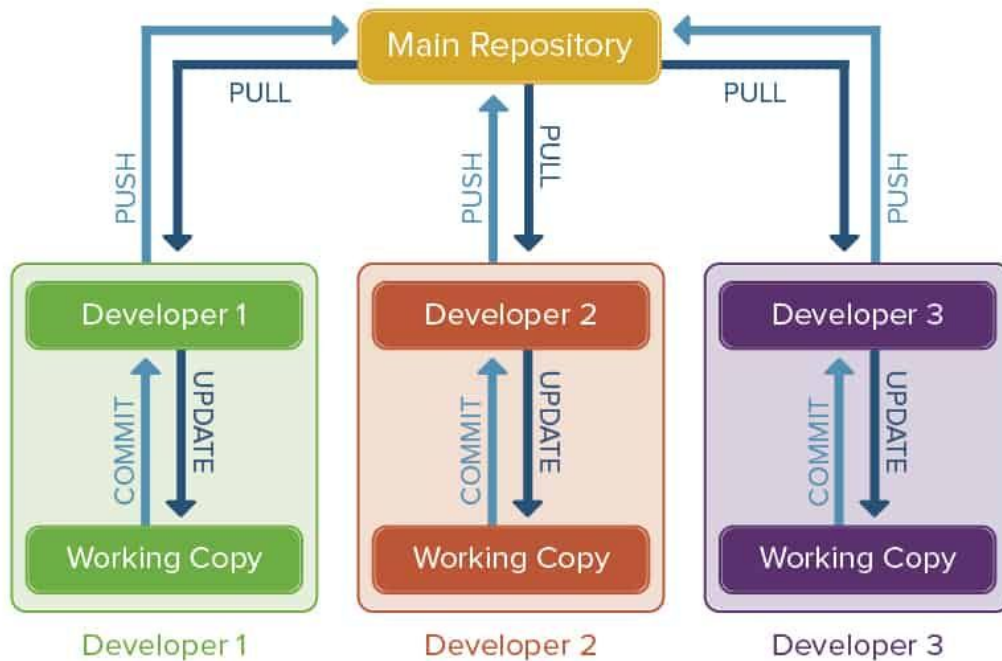
1.4 Termos

- **Repositórios:** refere-se a um local onde são **armazenados** os códigos do nosso software
 - **Repositório Origin:** é o nome dado por padrão ao repositório **remoto** ao qual nosso repositório local está vinculado, hospedadas na **internet** ou em uma rede.
 - **Repositórios locais:** que armazenam os arquivos de um dado projeto em **nossa máquina**, atualizando sempre com a versão origin.



1. Versionamento

Distributed Version Control



1. Versionamento

1.4 Termos

Branches: É uma versão, como o nome diz uma **ramificação**, uma “linha do tempo” da versão original. Cada branch tem um nome unico.

Nomes de branches:

- **main:** é a branch principal, aonde deve ter apenas códigos testados e revisados, pronto para ser usado no sistema.
- **develop:** branch usada para ambientes de teste
- **minha-atividade:** toda atividade pode ter um nome unico de branch, descrevendo brevemente a que se refere.

Commits: é um marco em uma branch, grava as alterações em um ou mais arquivos.

Merge: ao terminar os commits em minha branch, eu faço merge entre a minha branch e a branch main

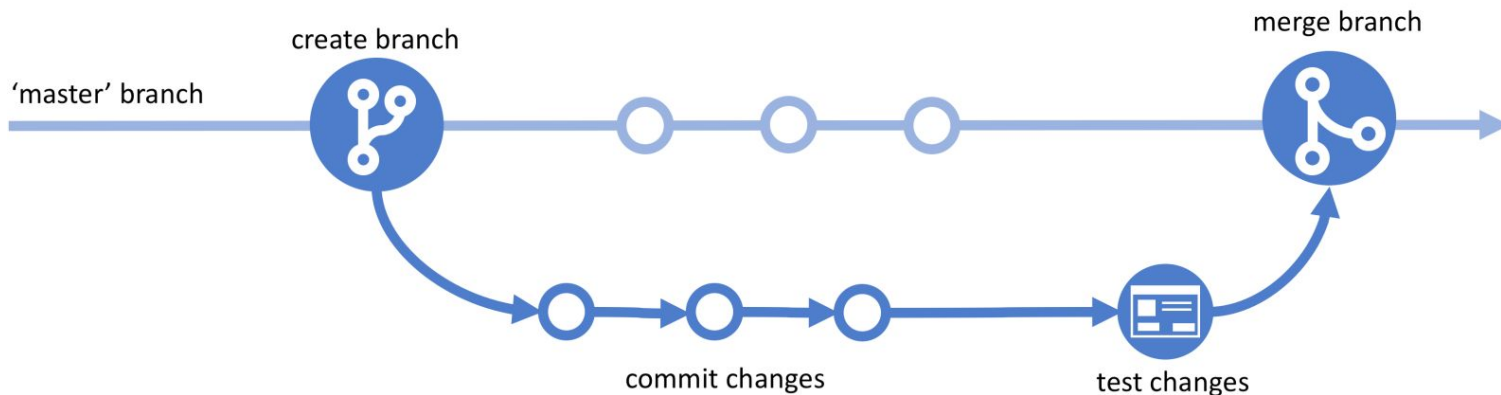
Branch: develop ▾

Commits on Nov 27, 2018

Merge branch 'develop' of https://github.com/devinehowest/COD1819_WN_08... robbevrhst committed 31 minutes ago	ca47e08	<>
cleanup robbevrhst committed 31 minutes ago	74f2a76	<>
createCharacter in player class vitaliethrill committed 38 minutes ago	c04e3fa	<>

1. Versionamento

Simplified Git Flow



1. Versionamento

1.3 Vantagens

1. **Controle de histórico:** É possível visualizar todo o **histórico** de desenvolvimento e voltar para **versões anteriores**;
1. **Trabalho em equipe:** Permite que **várias pessoas trabalhem** no mesmo conjunto de arquivos (**repositórios**) ao mesmo tempo **evitando conflitos** entre as **alterações**;
1. **Marcação e resgate de versões estáveis:** Por meio de **padrões**, é possível **identificar** quais **versões** do código estão **estáveis** e quais estão em **desenvolvimento**;
1. **Ramificação de projeto:** É possível ter várias **linhas** de **desenvolvimento paralelas** sem que uma interfira na outra;
1. **Segurança:** Possuem recursos para evitar invasões de agentes infecciosos nos arquivos;
1. **Confiança:** Pode ser usado como *backup*.

2. GitHub

2. GitHub

2.1 O que é GitHub?

GitHub é uma **plataforma** de **hospedagem** de **código-fonte** e **arquivos** com **controle de versão** usando o **Git**. Ele permite que programadores, utilitários ou qualquer usuário cadastrado na plataforma contribuam em projetos privados e/ou Open Source de qualquer lugar do mundo. GitHub é amplamente utilizado por programadores para **divulgação** de seus **trabalhos** ou para que outros programadores **contribuam** com o **projeto**, além de promover fácil comunicação através de **recursos** que **relatam problemas** ou **mesclam repositórios remotos** (issues, pull request).



Ícone GitHub

2. GitHub

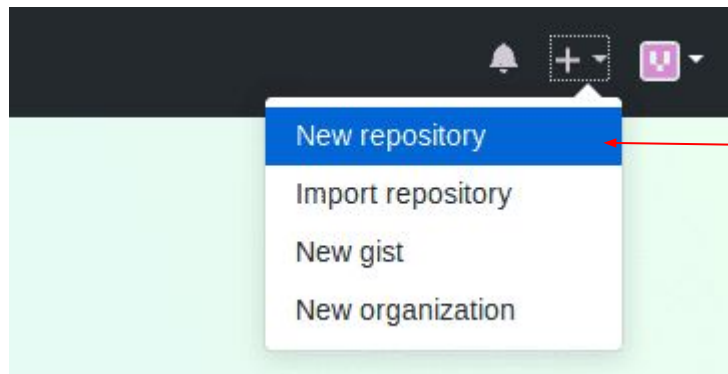
2.2 Iniciando com o GitHub

Passo 1: Acessar o site <https://github.com/>

Passo 2: Clicar em *Sign Up* e criar uma conta com seu e-mail e senha

Passo 3: Fazer o login na plataforma do github

Passo 4: Clique em criar novo repositório



2. GitHub

2.2 Iniciando com o GitHub

Passo 5: Criar o repositório conforme a imagem ao lado

Create a new repository

Owner *
meu-usuario ▼

Repository name *
meu-repositorio ✓

Description (optional)
meu primeiro repositório

☒
Public
Anyone on the internet can see this

☐
Private
You choose who can see and comm

Initialize this repository with:
Skip this step if you're importing an existing repository.

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None ▼

Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

License: None ▼

Create repository

2. GitHub

2.3 Entendendo arquivos padrões de um projeto

README.md - é utilizado para **descrever** seu **projeto**, **explicar** as características principais, **como** fazer para **funcionar** (uma espécie de resumo) onde você pode acrescentar diversas informações que você considere útil, ele é escrito em **Markdown** que é uma linguagem de marcação utilizada para converter o texto em um HTML válido.

O que pode ter no seu readme:

- O que é este esse projeto / o que esse projeto faz
- Como executar e testar o projeto/código
- Como consumir essa aplicação/essa API
- Quem fez esse projeto e porquê
- Quais são os próximos passos a implementar

.gitignore - Um arquivo **Git Ignore** especifica quais **arquivos** e **pastas** não devem ser **monitorados** em um determinado **código-fonte** pelo git, ao adicionar algum **nome** de um arquivo X dentro desse arquivo, o git irá ignorá-lo, mesmo que você mude ou delete o arquivo X.



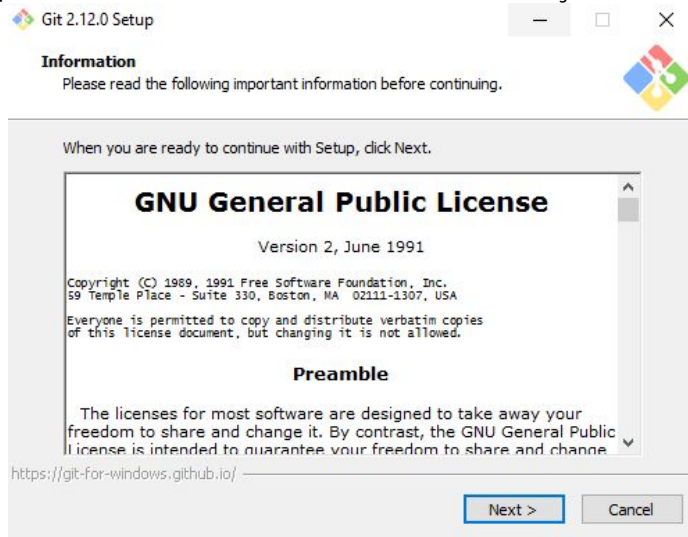
3. Iniciando o Git

3. Iniciando o Git

3.1 Instalando o Git

3.1.1 Instalar o GIT no Windows:

- Acesse [o site oficial](#) e faça o **download** do **instalador** do **GIT** para **Windows**. Depois de baixado, **clique duas vezes** no arquivo para iniciar o assistente de instalação. Basta seguir as instruções na tela, clicando em Next. Ao término, clique em Finish para concluir com êxito a instalação.

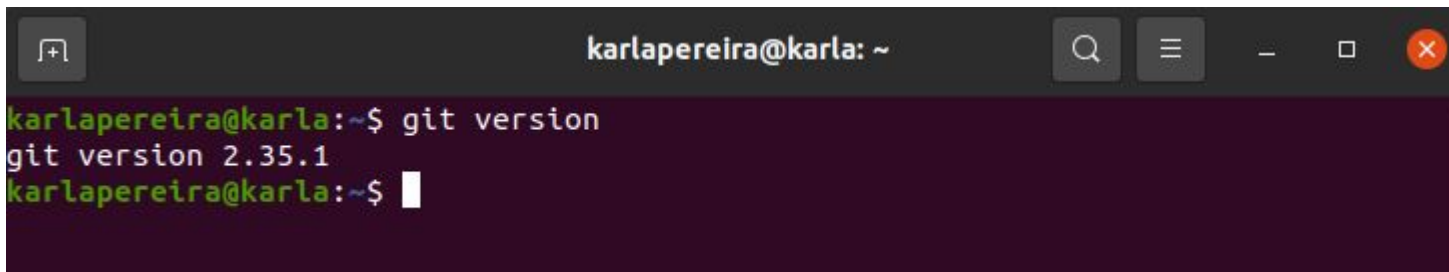


3. Iniciando o Git

3.1 Instalando o Git

3.1.1 Instalar o GIT no Linux:

- Abra o terminal e execute os seguintes comandos:
sudo apt-get update
sudo apt-get install git
- Verifique se a instalação ocorreu com sucesso usando *git --version*.



```
karlapereira@karla: ~  
karlapereira@karla:~$ git version  
git version 2.35.1  
karlapereira@karla:~$
```

3. Iniciando o Git

3.1 Instalando o Git

3.2 Configurar o Git

- Abra o prompt de comando e digite os seguintes comandos no terminal:

```
git config --global user.name "Meu nome"
```

```
git config --global user.email "exemplo@seuemail.com.br"
```

- Verificar as configurações

```
git config user.name
```

```
git config user.email
```

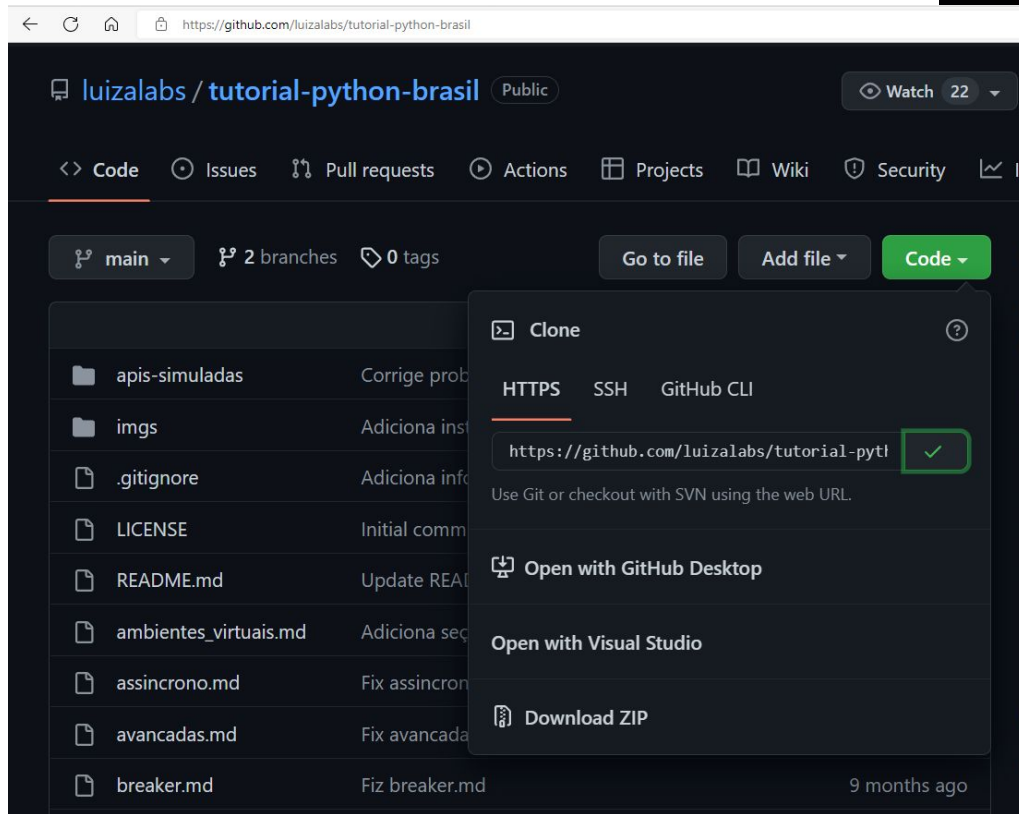
4. Clone

4. Clone

4.1 O que é git clone?

O **git clone** é um utilitário de linha de **comando** que é usado para selecionar um repositório existente e criar um **clone** ou cópia do repositório de destino.

4.2 Copiar *URL* do repositório destino

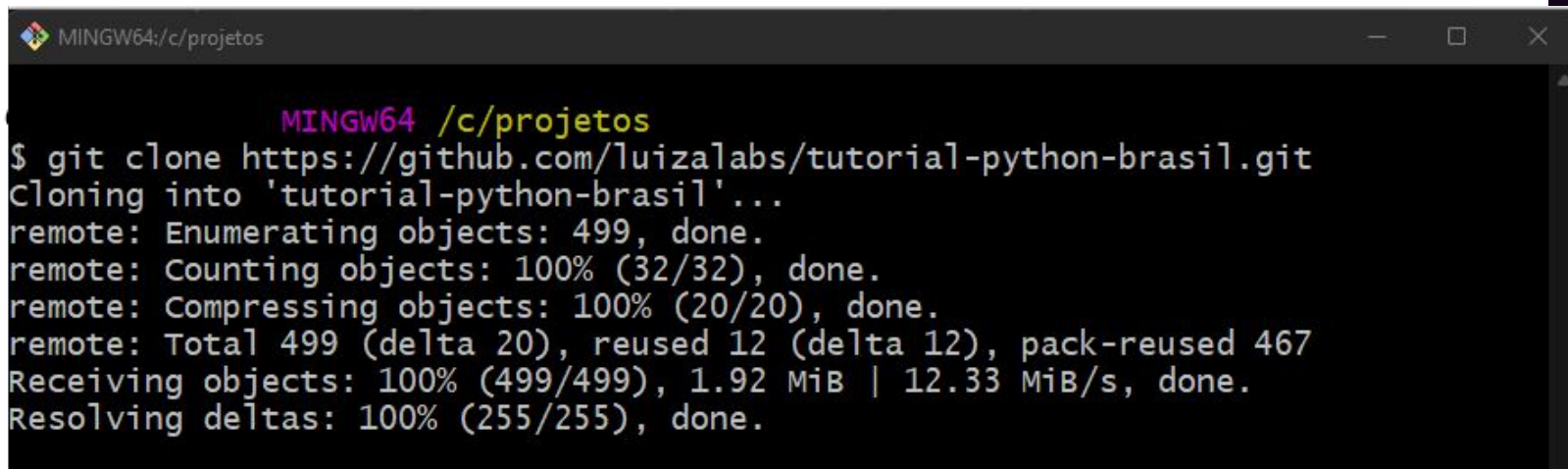


4. Clone

4.3 Clonando um repositório remoto

Escolha um diretório onde deseja deixar os códigos e arquivos do repositório remoto, em seguida, dentro do repositório abra o prompt de comando e digite:

git clone "URL copiada"

A screenshot of a terminal window titled 'MINGW64:/c/projetos'. The prompt is 'MINGW64 /c/projetos'. The command entered is '\$ git clone https://github.com/luizalabs/tutorial-python-brasil.git'. The output shows the cloning process: 'Cloning into 'tutorial-python-brasil'...', 'remote: Enumerating objects: 499, done.', 'remote: Counting objects: 100% (32/32), done.', 'remote: Compressing objects: 100% (20/20), done.', 'remote: Total 499 (delta 20), reused 12 (delta 12), pack-reused 467', 'Receiving objects: 100% (499/499), 1.92 MiB | 12.33 MiB/s, done.', and 'Resolving deltas: 100% (255/255), done.'

```
MINGW64:/c/projetos
MINGW64 /c/projetos
$ git clone https://github.com/luizalabs/tutorial-python-brasil.git
Cloning into 'tutorial-python-brasil'...
remote: Enumerating objects: 499, done.
remote: Counting objects: 100% (32/32), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 499 (delta 20), reused 12 (delta 12), pack-reused 467
Receiving objects: 100% (499/499), 1.92 MiB | 12.33 MiB/s, done.
Resolving deltas: 100% (255/255), done.
```


5. Comandos básicos git

5. Comandos básicos git

5.1 - O que é o git status?

O comando **git status** **exibe** as **condições** do **diretório** de **trabalho** e da **área de staging**. Ele permite que você veja quais alterações foram **despreparadas**, quais não foram e quais arquivos não estão sendo **monitorados** pelo Git.

git status

```
MINGW64: c:/projetos/tutorial-python-brasil
MINGW64 /c/projetos/tutorial-python-brasil (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

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

no changes added to commit (use "git add" and/or "git commit -a")
```

5.2 O que é git log?

O comando **git log** **exibe** instantâneos que receberam commit. Ele permite que você **liste** e **filtre** o **histórico do projeto** e pesquise alterações específicas.

5. Comandos básicos git

5.3 O que é git add?

O comando **git add** adiciona uma **alteração** no diretório ativo à área de **staging**. Ele diz ao **Git** que você quer incluir atualizações a um arquivo específico no próximo commit. No entanto, **git add** não tem efeito real e significativo no repositório — as alterações não são gravadas mesmo até você executar **git commit**.

Adicionar todas as alterações para área de staging:

git add .

Adicionar todos arquivos alterados de uma pasta:

git add nomeDaPasta

Adicionar um arquivo específico:

git add meuarquivo.py

git add nomeDaPasta/meuarquivo.py

```
MINGW64 /c/projetos/tutorial-python-brasil (main)
git add novodoc.txt

MINGW64 /c/projetos/tutorial-python-brasil (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   novodoc.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   README.md
```

5. Comandos básicos git

5.4 - O que é git commit?

O comando **git commit** **captura** um **instantâneo** das **mudanças** preparadas do **projeto** no momento. Os instantâneos com commit podem ser considerados versões "seguras" de um projeto, o Git nunca os altera, a menos que você peça a ele. Antes da execução de git commit, o comando git add é usado para promover ou "preparar" mudanças no projeto que são armazenadas em um commit. Estes dois comandos - git commit e git add - estão entre os mais usados.

git commit -m "comentário resumido das mudanças"

```
MINGW64 /c/projetos/tutorial-python-brasil (main)
$ git commit -m "meu primeiro commit"
[main 39a18ac] meu primeiro commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 novodoc.txt
```

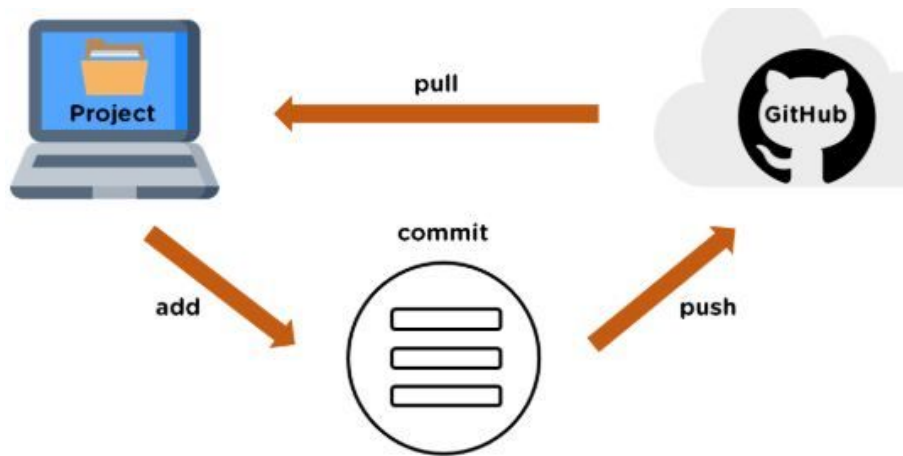
5. Comandos básicos git

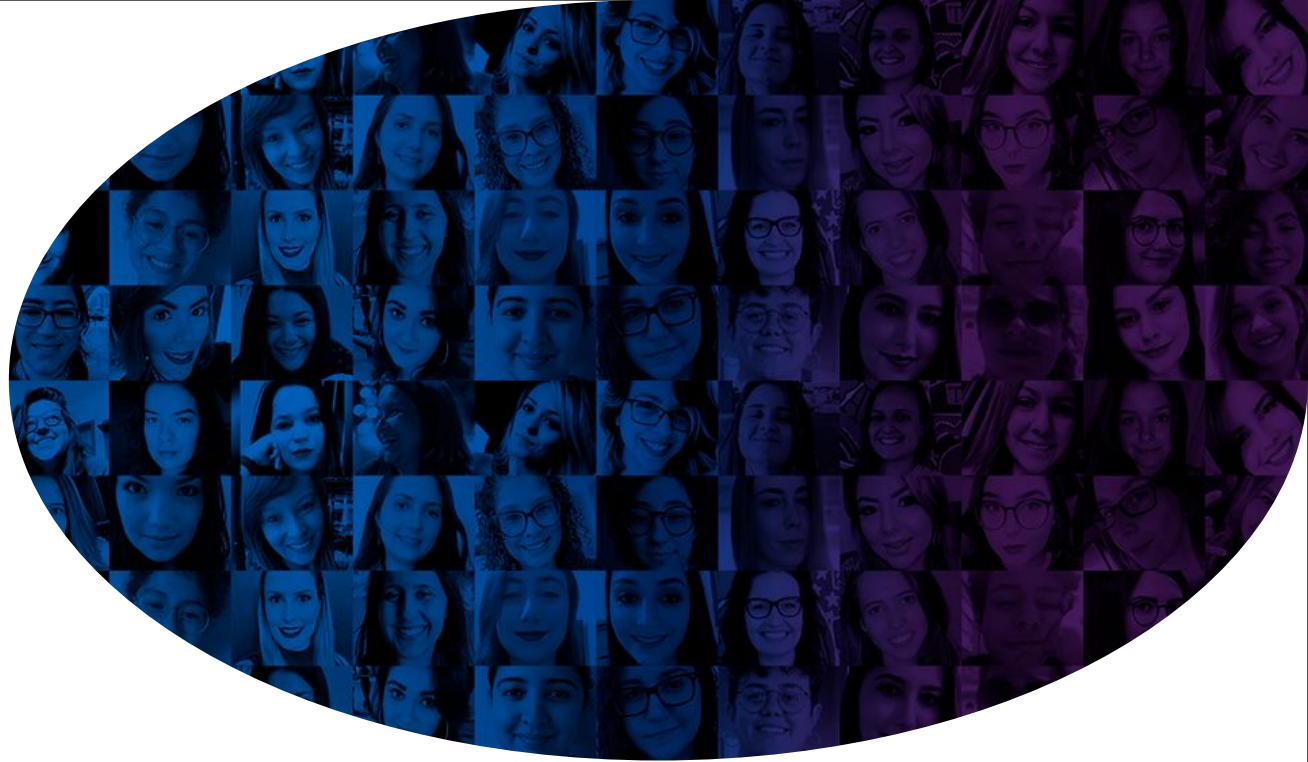
5.5 - O que é o git push?

O comando **git push** é usado para **enviar** o **conteúdo** do **repositório local** para um **repositório remoto**. O comando push transfere commits do repositório local a um repositório remoto, ou seja, o comando push exporta commits para branches remotos. Os branches remotos são configurados usando o comando git remote.

5.6 - O que é git pull?

O comando **git pull** é usado para **buscar** e **baixar conteúdo** de **repositórios remotos** e fazer a **atualização imediata** ao **repositório local** para que os conteúdos sejam iguais.





Perguntas?

Magalu



#VemSerFeliz