

Apostila

VERSIONAMENTO DE PROGRAMAS

José Marcelo Traina Chacon

Sérgio Luiz Moral Marques

2024

O que é o versionamento de software?	3
Como armazenar versões?	3
Definição de Sistema de controle de versões.....	4
Por que usar um Sistema de Gerenciamento de Versões?.....	4
Soluções Comerciais.....	4
Soluções Open-source.....	5
GIT	5
Um Pouco de História do Git	5
Diferença de GIT e GITHUB.....	6
GITHUB	6
Funcionamento de GIT e GITHUB.....	7
Instalando o GIT - Roteiro.....	7
Criar uma conta no repositório GITHUB	12
Duas formas para criar um repositório e trabalhar com GIT e GITHUB.....	13
Criando um repositório no GITHUB.....	13
Clonar um projeto do GITHUB para o repositório Local GIT.....	15
Manipulando arquivos no repositório GIT.....	19
Postar projeto do repositório privado GIT na plataforma de hospedagem GITHUB	23
Alterando um arquivo na plataforma GITHUB e transferindo para o repositório GIT	25
Criando um repositório no GIT	27
Criando uma TAG para controle dos versionamentos	33
Verificar alterações realizadas nos arquivos do diretório de trabalho – git diff	35
Restaurando uma versão	38
Criando uma ramificação no projeto Branch	40
Mesclando branchs	47
Sugestão de fluxo de trabalho	49
Continue estudando GIT e GITHUB	53

O que é o versionamento de software?

O que é o versionamento de software?

Podemos definir o versionamento como uma metodologia de classificação adotada por programadores com o intuito de controlar e acompanhar o histórico de alterações em um software.

Esse sistema permite distinguir as mudanças realizadas, facilitando a identificação de cada uma delas.

Como armazenar versões?

Não se preocupar: Pior opção é a não controlar versionamento de softwares



Acumular arquivos ZIP numerados (ou com data da versão): Dificuldade em verificar as mudanças feitas em cada versão, que devem ser feitas manualmente.



Sistema de Gerenciamento de Versões

Facilidade para verificar as mudanças no código entre versões

Facilidade em restaurar versões para testes



Definição de Sistema de controle de versões

Um sistema de controle de versões (ou versionamento), na função prática de Sistemas da Informação e da Engenharia de Software, é um software que tem a finalidade de gerenciar diferentes versões no desenvolvimento de um documento qualquer. Esses sistemas são comumente utilizados no desenvolvimento de software para controlar as diferentes versões — histórico e desenvolvimento — dos códigos-fontes e também da documentação.

Por que usar um Sistema de Gerenciamento de Versões?

Histórico das alterações: Responsável pelas alterações (quem quebrou o build)

Facilidade para voltar atrás e restaurar o código

Não precisa se preocupar em estragar um código que funciona

Permite juntar alterações feitas por diversos desenvolvedores

Tipos de Sistemas de Gerenciamento de Versões

Soluções Comerciais

Microsoft Visual SourceSafe (VSS)



Rational ClearCase



Borland StarTeam



Soluções Open-source

Concurrent Version System (CVS)



Subversion (SVN)



Git – Mercurial



GIT

Git é um sistema de controle de versão distribuída, rápido e escalável (tradução do manual)

Basicamente é um versionador de arquivos, é utilizado principalmente para gerenciar versões de softwares desenvolvidos por um ou mais desenvolvedores, com ele podemos implementar novas funcionalidades e tudo é registrado em um histórico, o qual podemos retroceder sempre que necessário, os integrantes de um projeto podem enviar correções, atualizações, etc. As alterações enviadas para o projeto principal não comprometem o mesmo pois cabe ao dono do projeto a inclusão ou não das alterações efetuadas.

Um Pouco de História do Git

O GIT foi desenvolvido inicialmente por Linus Torvalds (Criador do Linux), pela necessidade de ter um software capaz de controlar a versão do kernel do Linux.

Diferença de GIT e GITHUB

GIT

Software de Controle de Versão

Versionamento

GITHUB

Plataforma de rede social para programadores

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.

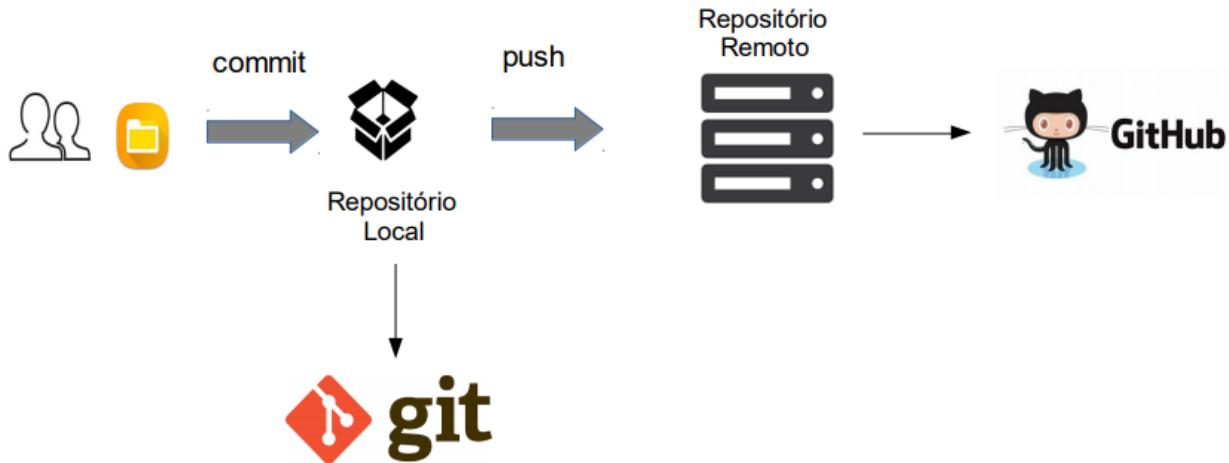
Funcionamento de GIT e GITHUB

GIT – Versionamento

Usuário cria ou altera o seu arquivo e o envia (Commit) para o Repositório Local

GITHUB - Rede social para programadores

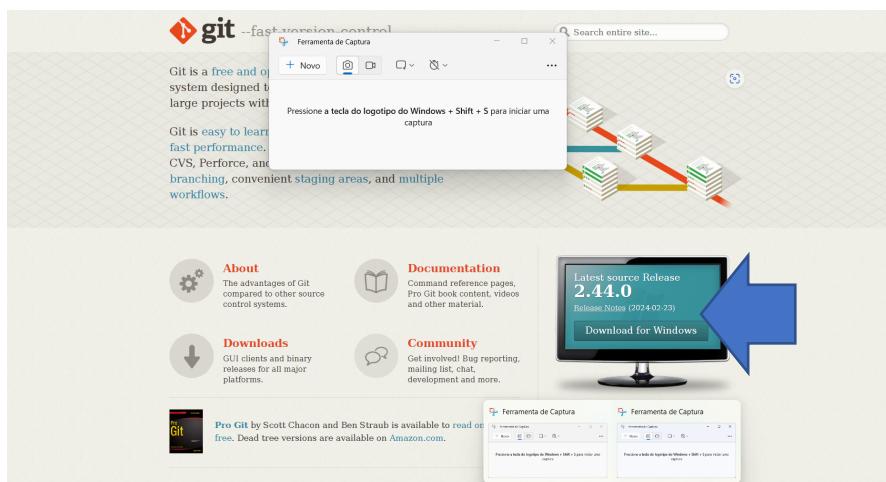
O programador disponibiliza o seu código (push) em um repositório Remoto



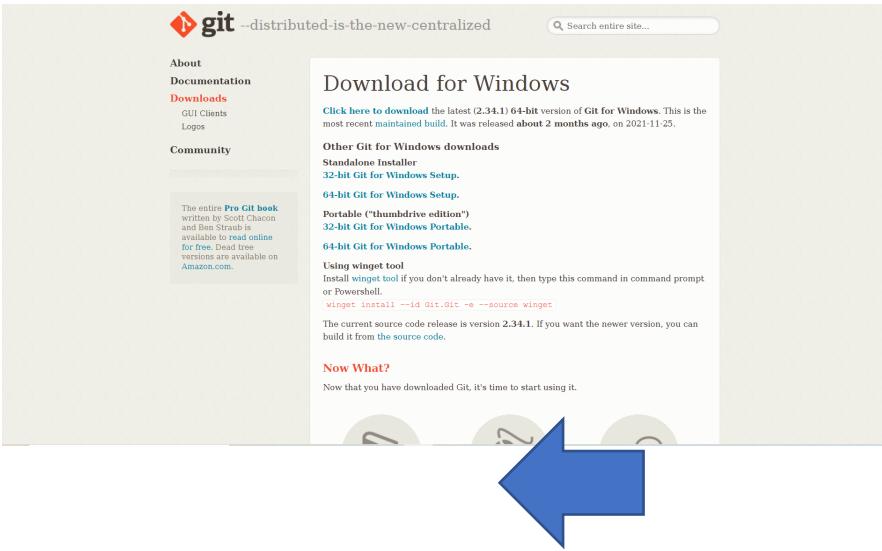
Instalando o GIT- Roteiro

Abra o site: git-scm.com

Clique em Download



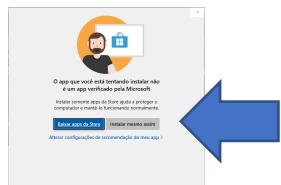
Escolha o tipo de sistema



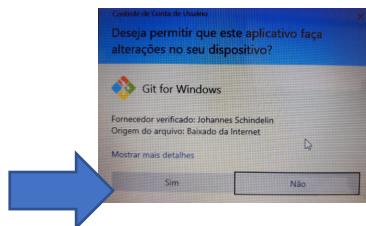
Iniciará o download na barra de tarefas. Clique no botão para instalar o programa.



Se aparecer a janela de mensagem do APP Store clique em Instalar mesmo assim



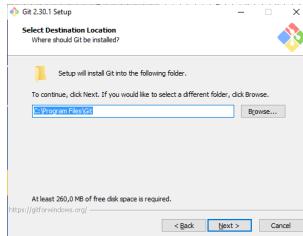
Se aparecer a janela controle de usuário clique em sim



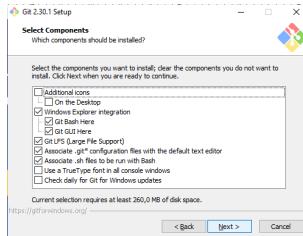
Clique em Next



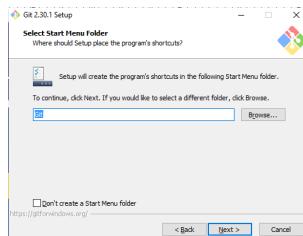
Clique em Next



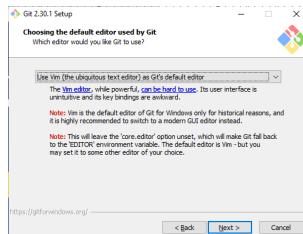
Clique em Next



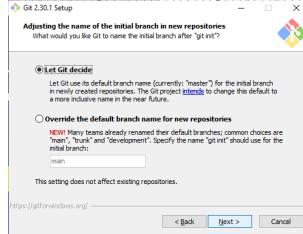
Clique em Next



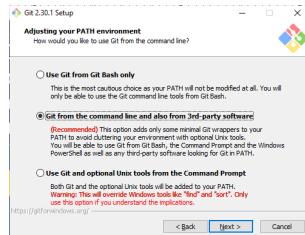
Clique em Next



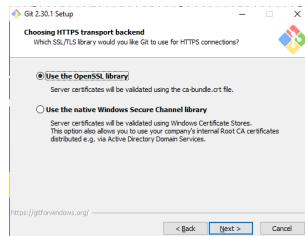
Clique em Next



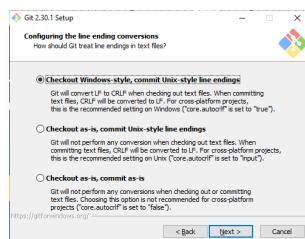
Clique em Next



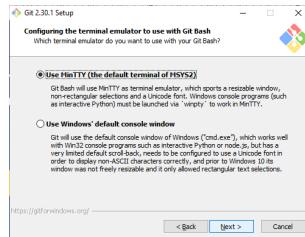
Clique em Next



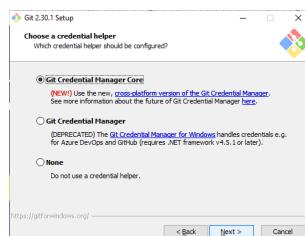
Clique em Next



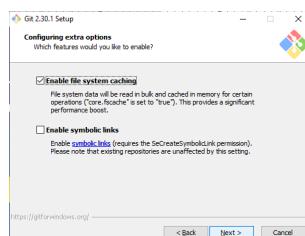
Clique em Next



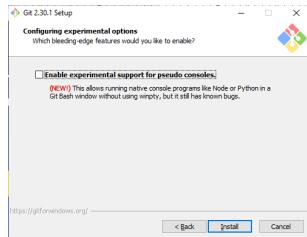
Clique em Next



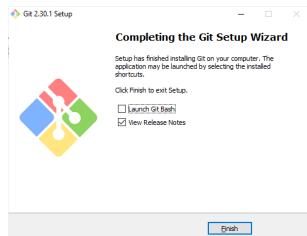
Clique em Next



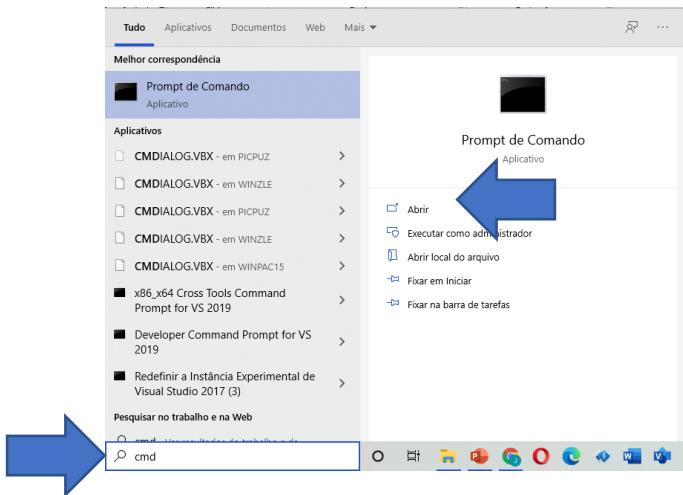
Clique em Install



Clique em Finish



Digite cmd na pesquisa do Windows e abra o Prompt de Comando



Configurar os seus dados

- git config --global user.name "João Silva"
- git config --global user.email "exemplo@seuemail.com.br"

Nota: Lembre de substituir **João Silva** e **exemplo@seuemail.com.br** com seus dados. Qualquer *commit* criado posteriormente será associado à esses dados.

```
C:\Users\jmtch>git config --global user.name "Marcelo Chacon"
C:\Users\jmtch>git config --global user.email "chacon@puc-campinas.edu.br"
```

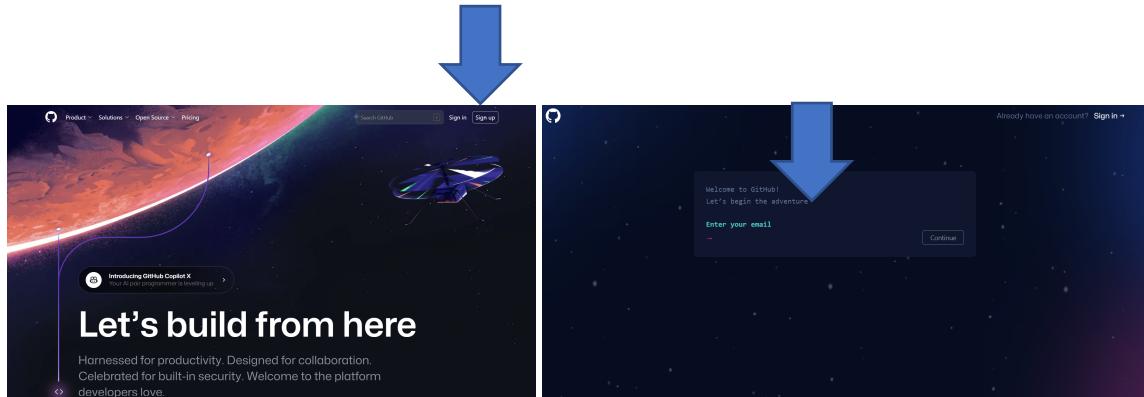
Criar uma conta no repositório GITHUB

Abra o site: github.com/

Clique em Criar uma conta

Efetue seu cadastro

Confirme o cadastro no seu email



Instalação concluída

Crie seu primeiro projeto
Pronto para começar a construir? Crie um repositório para uma nova ideia ou traga um repositório existente para continuar contribuindo com ela.

Trabalhando em equipe?
O GitHub foi desenvolvido para colaboração. Configure uma organização para melhorar a maneira como sua equipe trabalha em conjunto e obtenha acesso a mais recursos.

Aprenda Git e GitHub sem nenhum código!
Usando o guia Hello World, você criará um repositório, iniciará um branch, escreverá comentários e abrirá uma solicitação pull.

Descubra projetos e pessoas interessantes para preencher seu feed de notícias pessoais.
Seu feed de notícias o ajuda a acompanhar as atividades recentes nos repositórios que você [assiste](#) e nas pessoas que [segue](#)

Duas formas para criar um repositório e trabalhar com GIT e GITHUB

Temos duas formas possíveis para trabalhar os repositórios no GIT e GITHUB

1. Criar o repositório do GITHUB para o computador (GIT)
2. Criar um repositório do computador (GIT) para o GITHUB

Criando um repositório no GITHUB

ETAPA 1. Criar um Re却itório

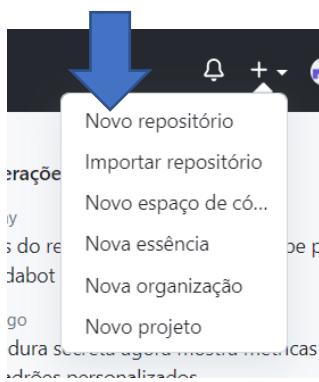
O que é um repositório

Um repositório geralmente é usado para organizar um único projeto. Os repositórios podem conter pastas e arquivos, imagens, vídeos, planilhas e conjuntos de dados - tudo o que seu projeto precisar.

Recomenda-se incluir um README ou um arquivo com informações sobre seu projeto. O GitHub facilita a adição de um ao mesmo tempo em que você cria seu novo repositório. Ele também oferece outras opções comuns, como um arquivo de licença.

Para criar um novo repositório

No canto superior direito, próximo ao seu avatar ou ícone de identidade, clique em + e selecione Novo repositório.

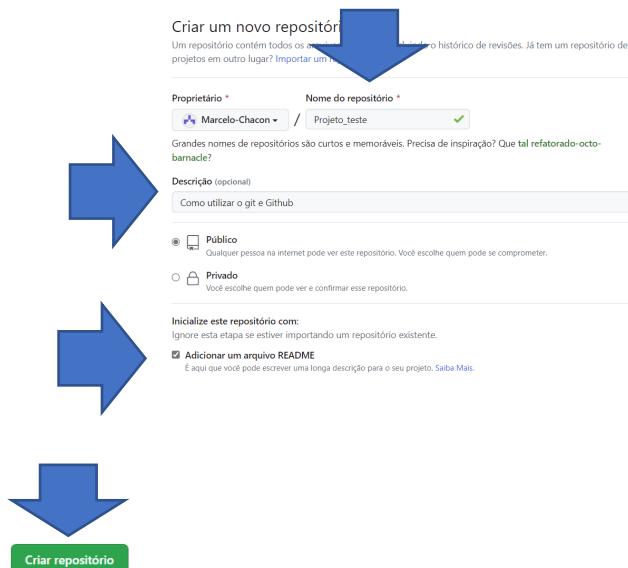


Nomeie seu repositório Projeto_teste.

Escreva uma breve descrição.

Selecione Inicializar este repositório com um README .

Clique em Criar Repositório



O Re却torio estaria criado no GITHUB

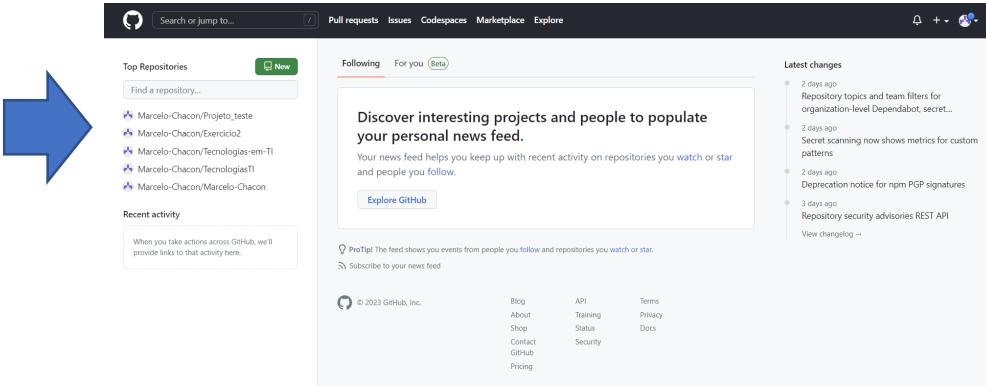
The screenshot shows the GitHub repository page for 'Marcelo-Chacon / Projeto_teste'. The repository has one commit by 'Marcelo-Chacon' titled 'Initial commit' made 1 minute ago. The commit message is 'Como utilizar o git e Github'. The repository has 0 stars, 0 forks, and 0 releases. The footer includes standard GitHub links like Terms, Privacy, Security, Status, Docs, Contact GitHub, Pricing, API, Training, Blog, and About.

Clonar um projeto do GITHUB para o repositório Local GIT

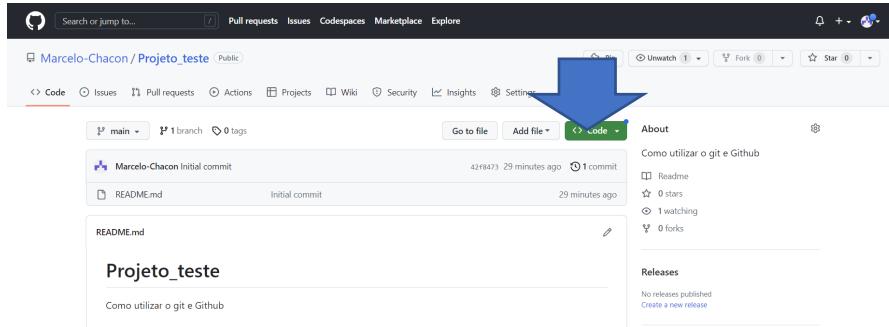
ETAPA 2. Clonar um Projeto para o repositório local GIT

Clonar um Projeto do Plataforma de hospedagem GitHub no seu repositório Local GIT.

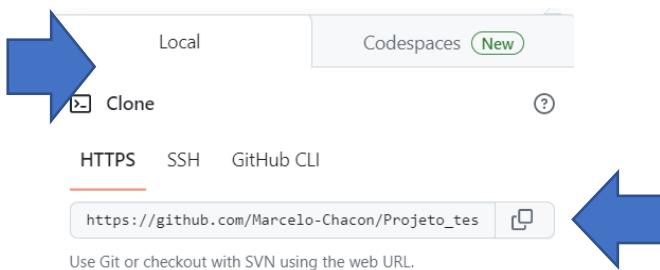
Abra o repositório criado no GITHUB



Baixar o projeto do GITHUB no repositório local GIT, clique no botão Code.



Copie o endereço URL para utilizar no GIT

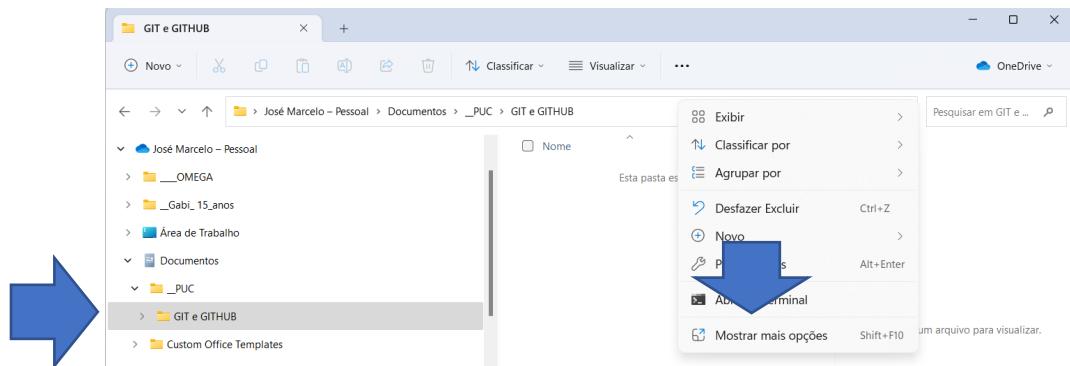


Abra o Explorador de arquivos

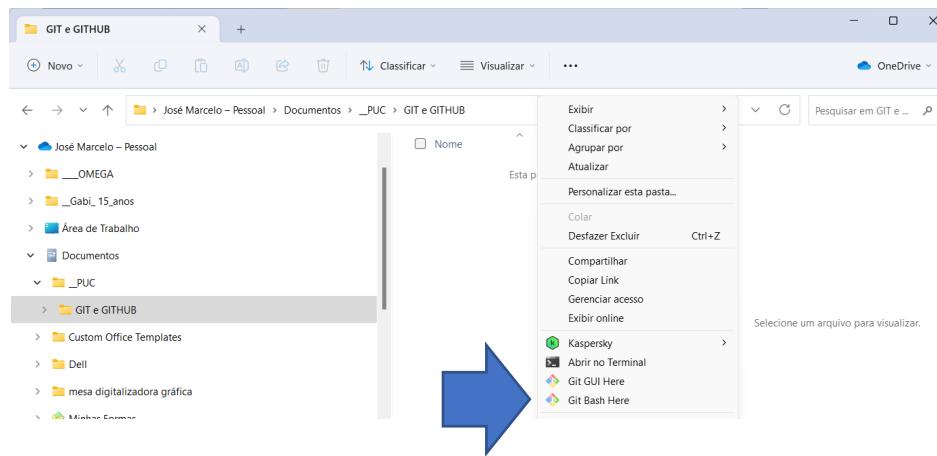


Crie ou escolha uma pasta que será o repositório local de seu projeto

Clique com o botão direito do mouse na visualizador de pastas e arquivos, clique em mais opções



Clique em Git Bash Here



Será aberto o terminal do git já na pasta selecionada para criar o repositório

Clone o projeto no GIT, digitando git clone e colando o link copiado do GITHUB (SHIFT + INSERT), e pressione Enter

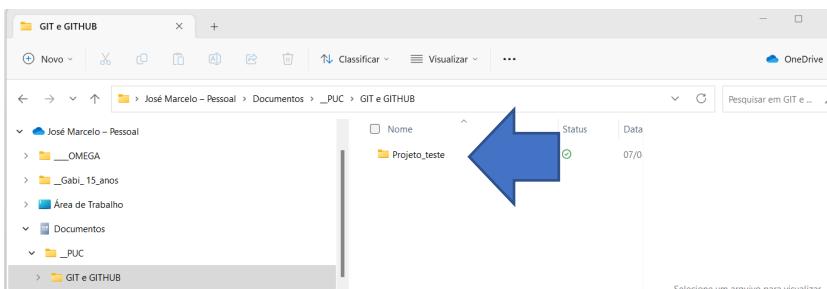
```
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB
$ git clone https://github.com/Marcelo-Chacon/Projeto_teste.git
Cloning into 'Projeto_teste'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB
$
```

O repositório local do projeto foi criado em um diretório local em sua máquina

Digite dir no Terminal Git , e pressione Enter

```
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB
$ dir
Projeto_teste
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB
$
```

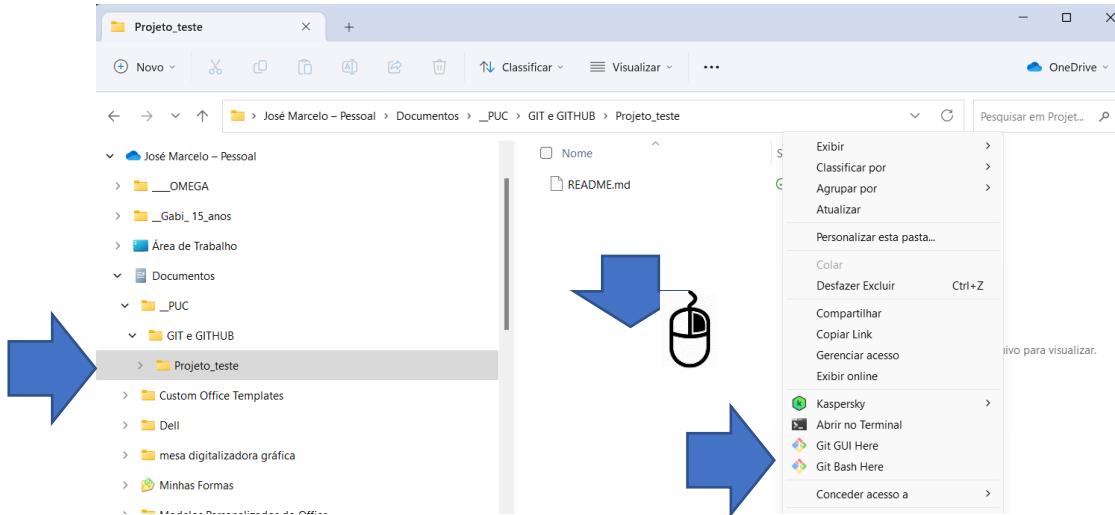
Ou abra o navegador e encontre a pasta criada em seu usuário local



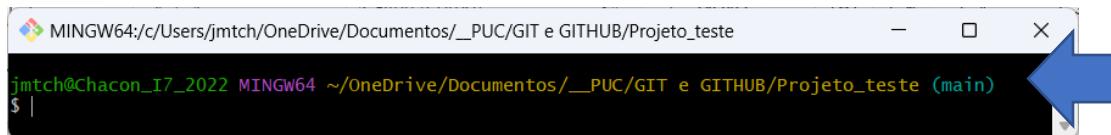
Entre na sua pasta de repositório local de seu projeto

Clique com o botão direito do mouse no visualizador de pastas e arquivos, clique em mais opções

Clique em Git Bash Here



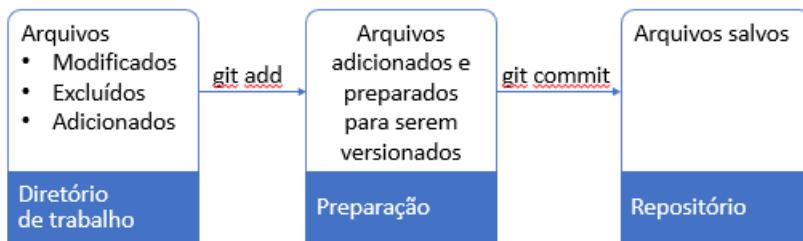
Será mostrado que você está na Branch main



Manipulando arquivos no repositório GIT

ETAPA 3. Manipulando arquivos no repositório Git

Como funcionam as áreas do Git



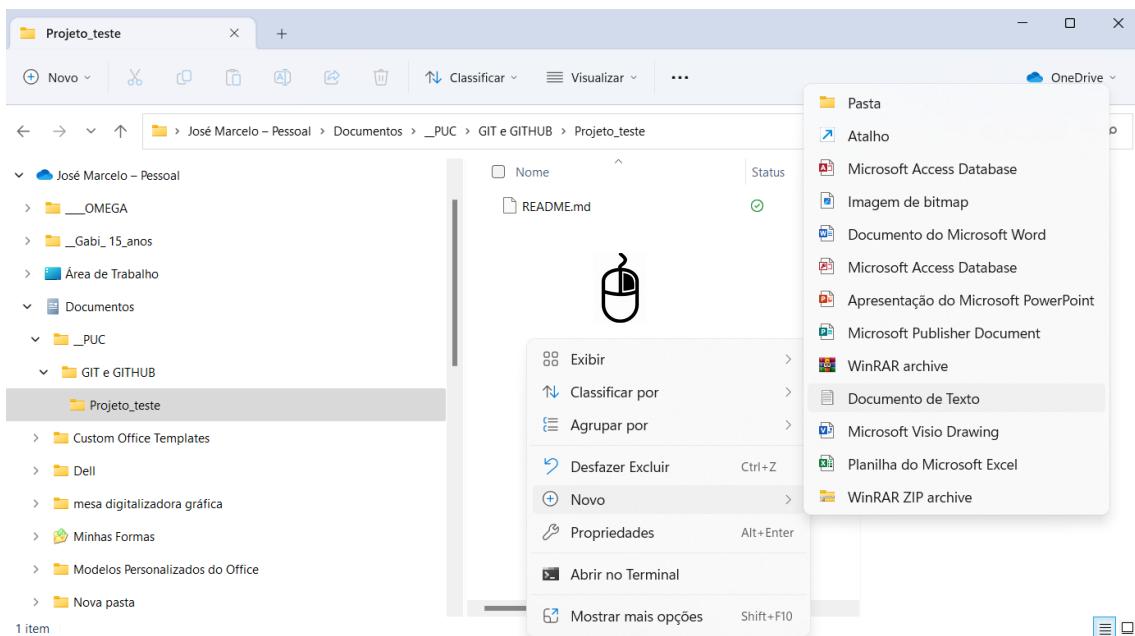
Diretório de trabalho: Local onde foram modificados, excluídos ou adicionados arquivos no projeto.

Preparação: Local onde os arquivos são preparados para serem versionados no Git, isto é feito através do comando `git add`

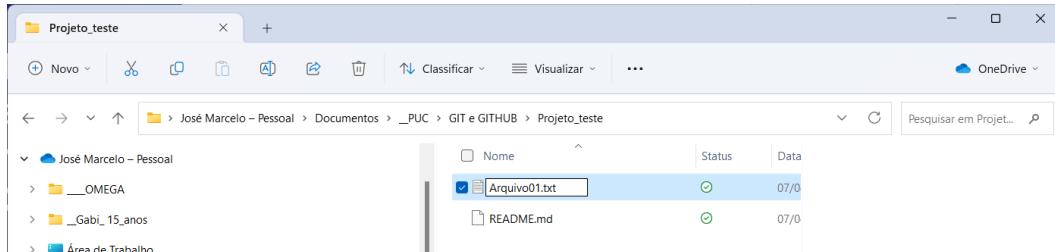
Repositório: Arquivos salvos com versionamento do git, isto é feito através do comando `git commit`

Copiar um criar um arquivo no repositório Git

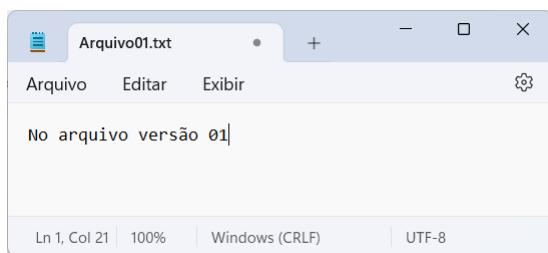
Criar um arquivo TXT no repositório Git



Renomear o arquivo para Arquivo01.txt e pressionar Enter

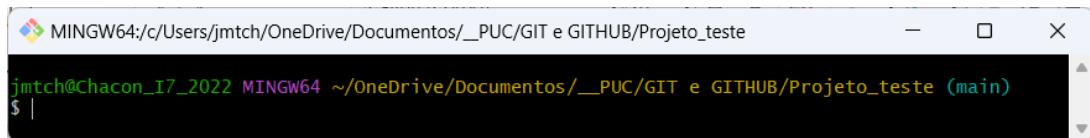


Abra o Arquivo01.txt e digite o texto



Salve o Arquivo01.txt

Abra o Git Bash Here do repositório Git



Mostrar como está o nosso repositório Git

Digite o comando **git status**

```
MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/GIT e GITHUB/Projeto_teste
jmtch@Chacon_17_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB/Projeto_teste (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Arquivo01.txt

nothing added to commit but untracked files present (use "git add" to track)
jmtch@Chacon_17_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB/Projeto_teste (main)
$
```

Foi incluído um arquivo na Branch Main

Necessita adicionar este arquivo para fazer o Commit

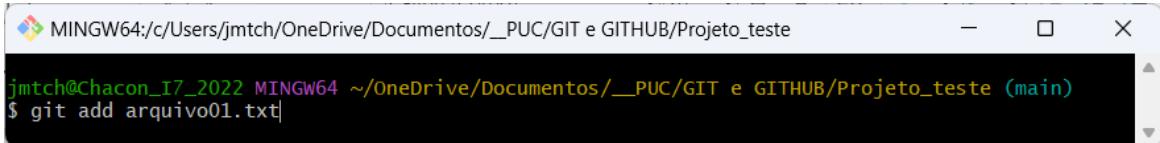
Isto significa que o arquivo foi criado na pasta, mas não está adicionado no repositório Git.



Adicionando arquivos na área de Preparação do Git

Você pode adicionar arquivos na área de preparação de duas maneiras:

Adicionar cada arquivo do diretório de trabalho isoladamente para a área de preparação: **git add <nome do arquivo>**

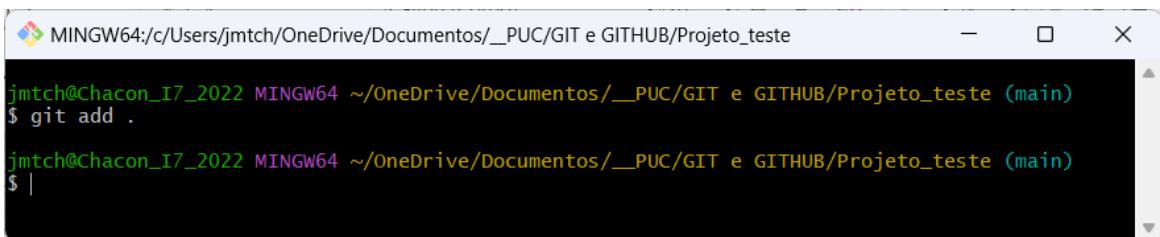


```
MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/GIT e GITHUB/Projeto_teste
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB/Projeto_teste (main)
$ git add arquivo01.txt
```

Não é muito prático, pois tem que acionar arquivo por arquivo para a área de preparação

Adicionar todos os arquivos do diretório de trabalho para a área de preparação:

git add .



```
MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/GIT e GITHUB/Projeto_teste
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB/Projeto_teste (main)
$ git add .
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB/Projeto_teste (main)
$ |
```

Adicionou todos os arquivos da pasta de trabalho para a área de preparação do Git

Verificar que os arquivos foram adicionados na área de preparação: **git status**



```
MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/GIT e GITHUB/Projeto_teste
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB/Projeto_teste (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:   Arquivo01.txt

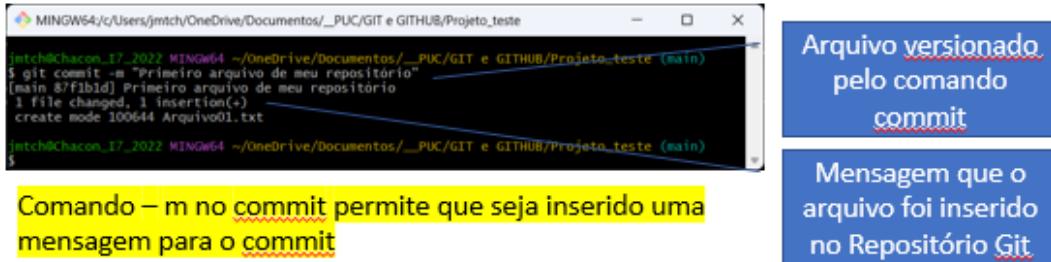
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB/Projeto_teste (main)
$ |
```

Novo arquivo na área de preparação



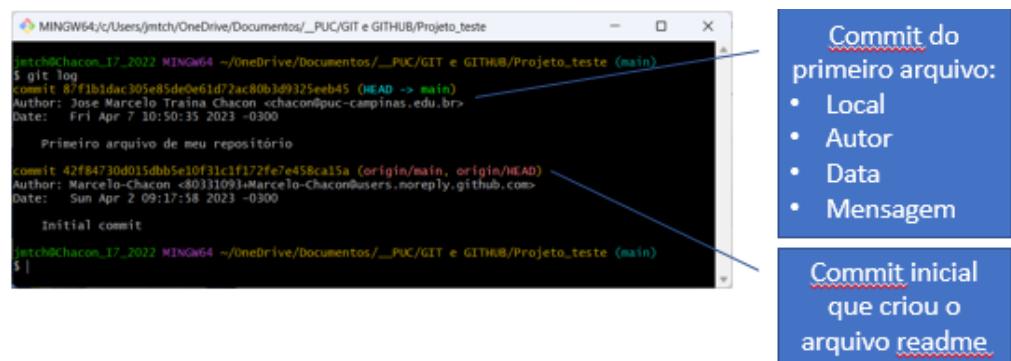
ETAPA 4. Versionando arquivos no GIT

Criando o versionamento do projeto no Git: **git commit –m “Mensagem”**



Visualizando os históricos dos commits (Log)

Visualizar os logs do commit: **git log**



Postar projeto do repositório privado GIT na plataforma de hospedagem GITHUB

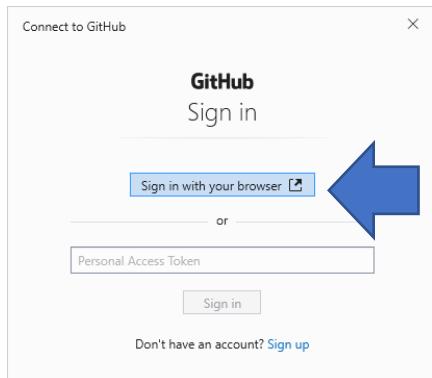
ETAPA 5. Fazer um push para o GITHUB

Para transferir o seu projeto do repositório local (GIT) para a plataforma de hospedagem (GITHUB) digite no prompt de comando:
git push origin main e pressione **Enter**



```
MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/GIT e GITHUB/Projeto_teste
jmtch@Chacon_I7_2022 MINGW64 ~ % $ git push origin main
```

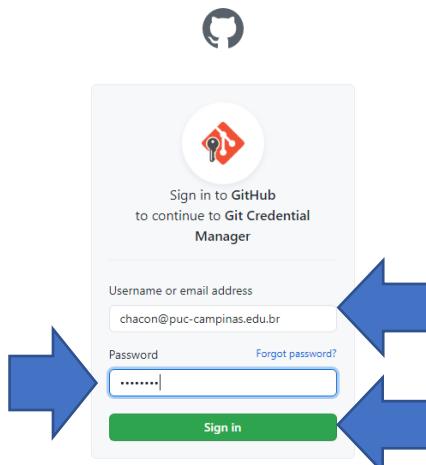
Se solicitado faça o login no GITHUB, faça o login na sua conta.



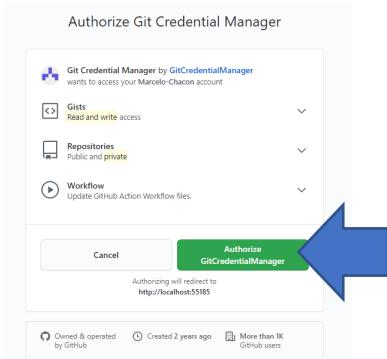
Digite o nome

Digite a senha

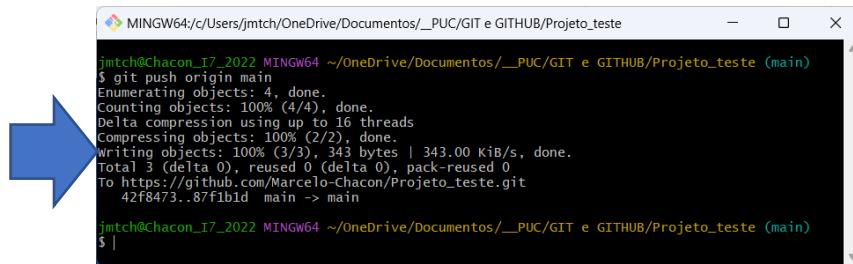
Clique em Sign in



Clique em Authorize GitCredentialManager

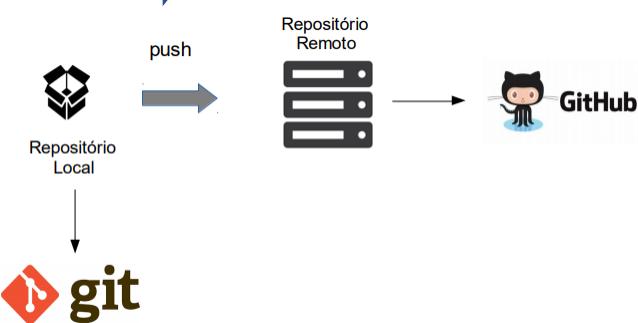
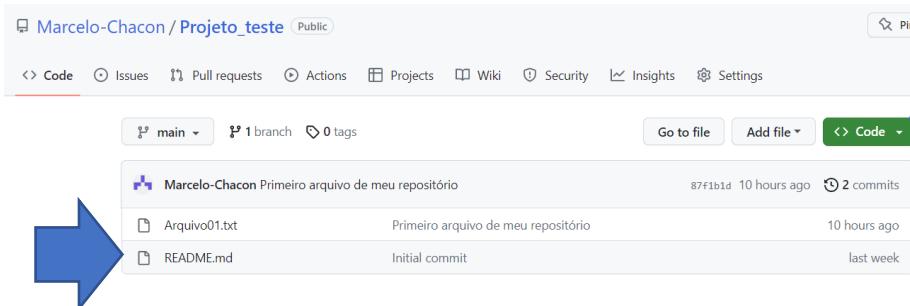


Pronto o conteúdo de seu repositório local foi enviado ao GITHUB



Pode ser confirmado no GITHUB (não esqueça de atualizar a página do GITHUB

- F5)

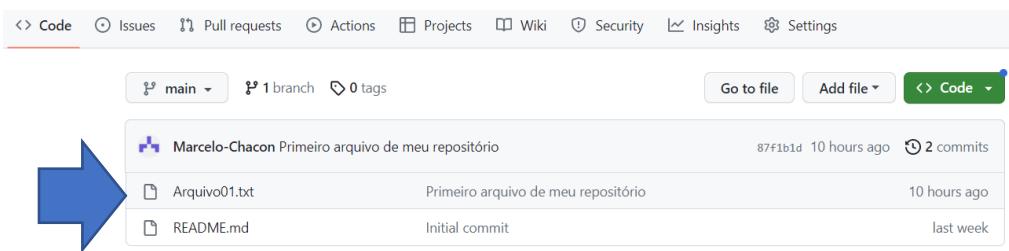


Alterando um arquivo na plataforma GITHUB e transferindo para o repositório GIT

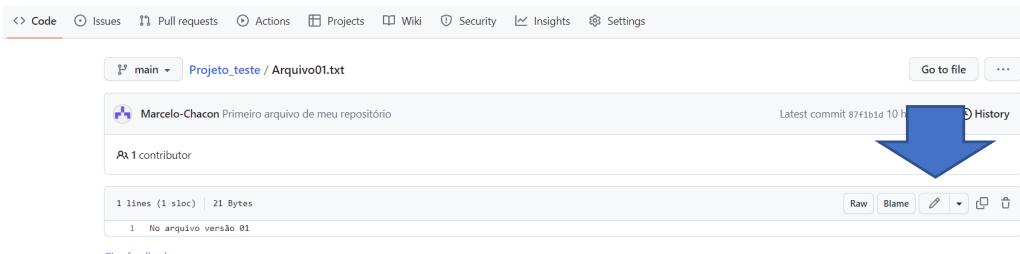
No **GitHub**, as alterações salvas são chamadas de **commits**. Cada **commit** tem uma mensagem de **commit** associada, que é uma descrição que explica por que uma mudança específica foi feita. As mensagens de **commit** capturam o histórico de suas alterações, para que outros colaboradores possam entender o que você fez e por quê.

Faça as alterações e commit

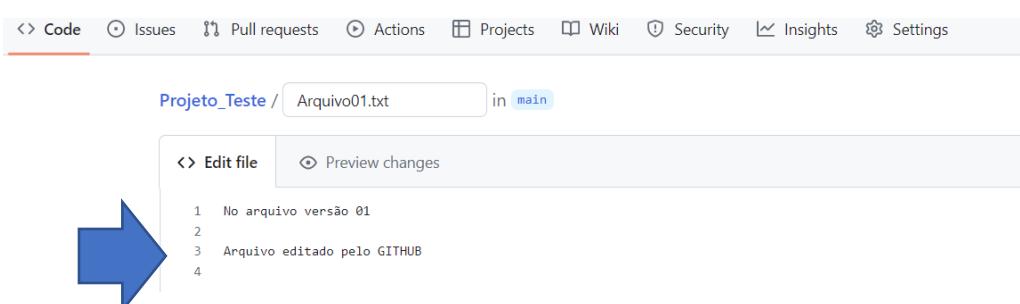
Clique no arquivo **Arquivo01.txt**



Clique no ícone de lápis no canto superior direito da visualização do arquivo a ser editado.



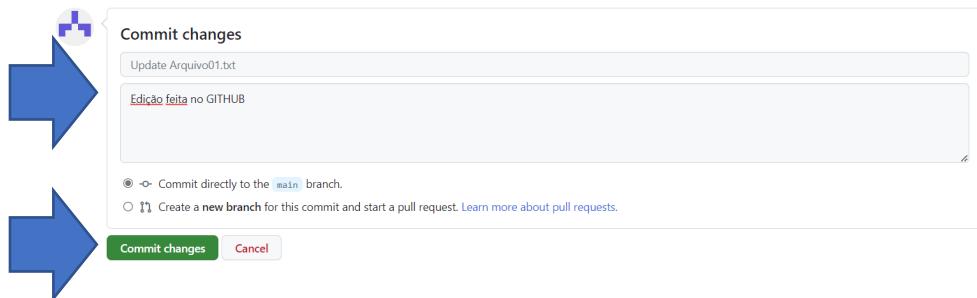
No editor, escreva o texto abaixo.



Faça as alterações e commit

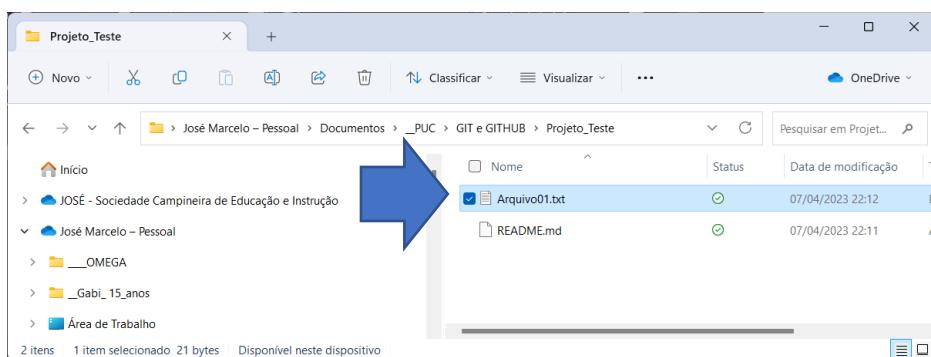
Escreva uma mensagem de confirmação que descreva suas mudanças.

Clique no botão Confirmar alterações.

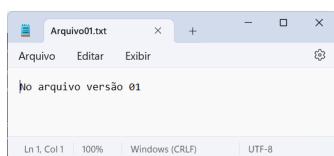


As alterações feitas no GITHUB não estarão atualizadas no GIT

Abrir o Arquivo01.txt no Explorador de arquivos



Veja que o conteúdo no repositório local não foi alterado



Carregando o repositório GITHUB atualizado para o GIT

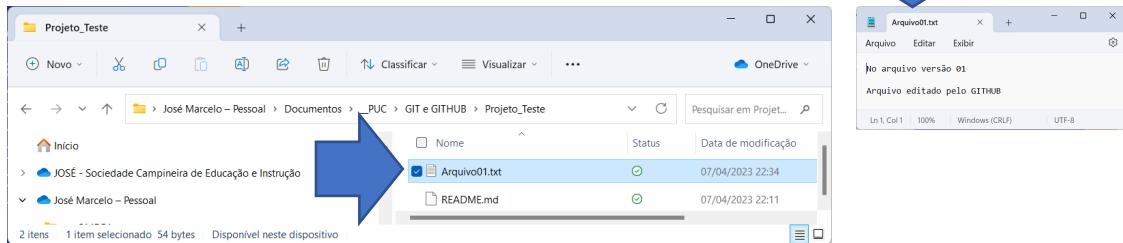
Digite o comando git pull origin main

```
MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/GIT e GITHUB/Projeto_Teste
jmtch@Chacon_17_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB/Projeto_Teste (main)
$ git pull origin main
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 727 bytes | 145.00 KiB/s, done.
From https://github.com/Marcelo-Chacon/Projeto_Teste
 * branch      main      -> FETCH_HEAD
 * branch      main      -> origin/main
Updating 7d00c08..b570de4
Fast-forward
  Arquivo01.txt | 4 +++
  1 file changed, 3 insertions(+), 1 deletion(-)

jmtch@Chacon_17_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB/Projeto_Teste (main)
```

Abrir o Arquivo01.txt no Explorador de arquivos

O Arquivo01.txt estará atualizado



Ver o log: git log

```
MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/GIT e GITHUB/Projeto_Test
commit b570de4067e0315dda017d409d1cc7f87ea31ed7 (HEAD -> main, origin/main, origin/HEAD)
Author: Marcelo-Chacon <80331093+Marcelo-Chacon@users.noreply.github.com>
Date:   Fri Apr 7 22:17:09 2023 -0300

    Update Arquivo01.txt
    Editado pelo GITHUB

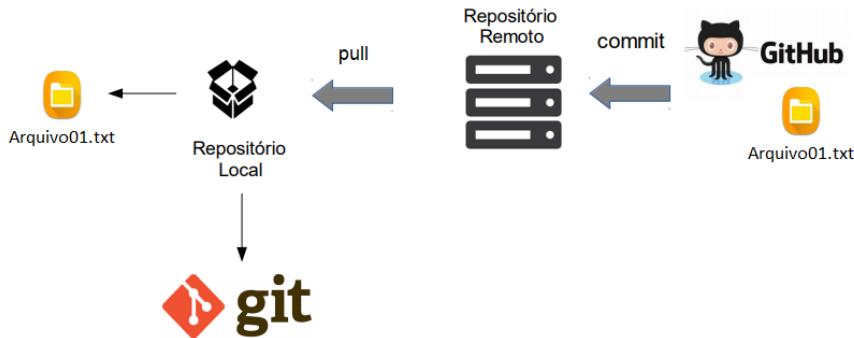
commit 7d0b0c863a5c84a71ddf932da759c7529ddf137c
Author: Jose Marcelo Traina Chacon <chacon@puc-campinas.edu.br>
Date:   Fri Apr 7 22:13:59 2023 -0300

    Primeiro arquivo do meu repositorio

commit 27872fc0f738472b0068fc54a8753ddc2ab51cbd
Author: Marcelo-Chacon <80331093+Marcelo-Chacon@users.noreply.github.com>
Date:   Fri Apr 7 22:09:30 2023 -0300

:
```

É possível ver o commit realizado pelo update do Arquivo01.txt e sua mensagem



Criando um repositório no GIT

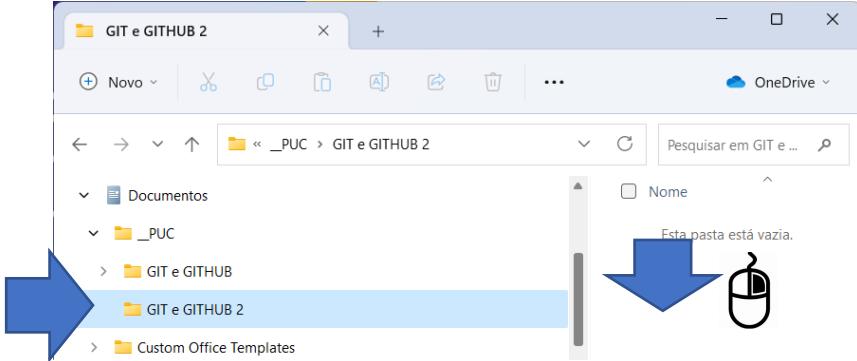
ETAPA 1. Criar um Repositório no GIT

Abra o Explorador de arquivos

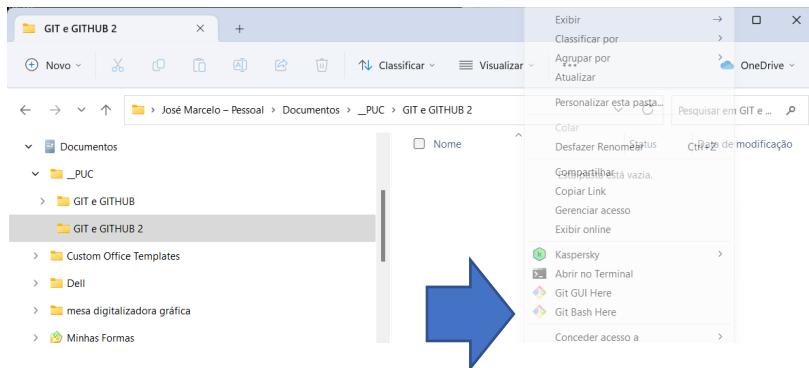


Crie ou escolha uma pasta que será o repositório local de seu projeto

Clique com o botão direito do mouse na visualizador de pastas e arquivos, clique em mais opções



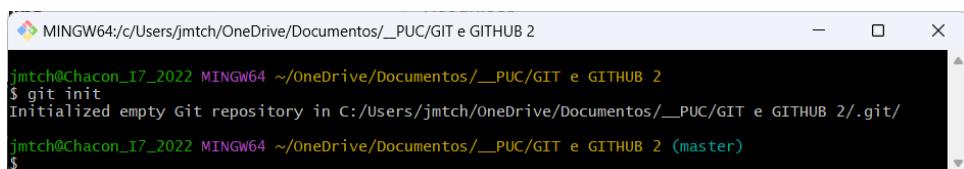
Clique em Git Bash Here



Será aberto o terminal do git já na pasta selecionada para criar o repositório



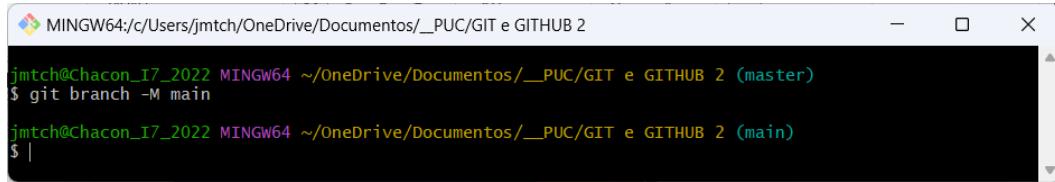
Crie um repositório GIT digitando: git init, e pressione Enter



Reparam que o repositório foi criado e está mostrando que estamos na Branch master. Quando um repositório é criado no GITHUB é criado a Branch main e no GIT é

criado a Branch master. Vamos renomear a Branch master para main para ficar igual a nomenclatura do GITHUB.

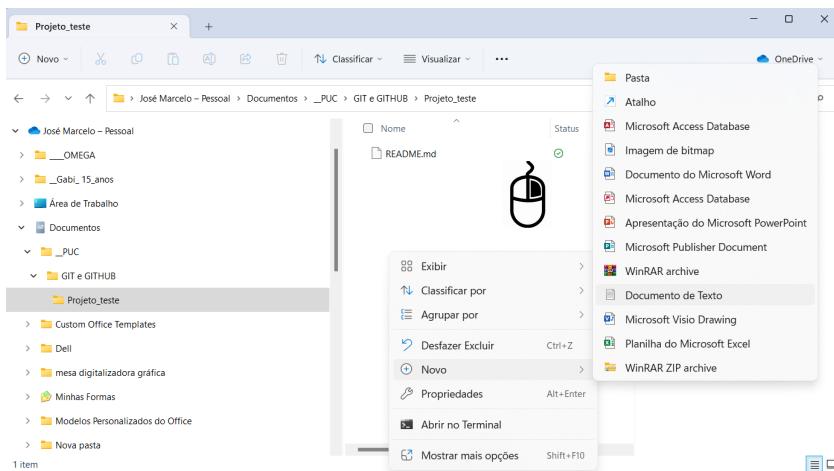
Renomear a Branch master para main digite: **git branch -M main**



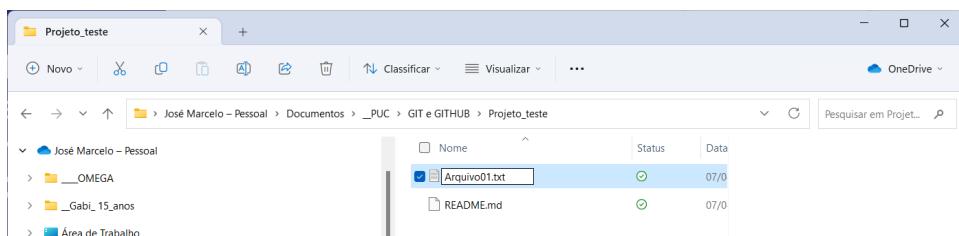
```
MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/GIT e GITHUB 2
jmtch@Chacon_17_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (master)
$ git branch -M main
jmtch@Chacon_17_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ |
```

Copiar um criar um arquivo no repositório Git

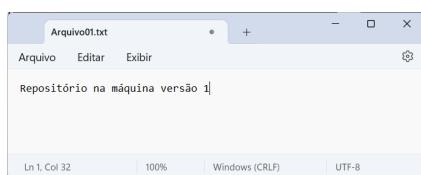
Criar um arquivo TXT no repositório Git



Renomear o arquivo para Arquivo01.txt e pressionar Enter



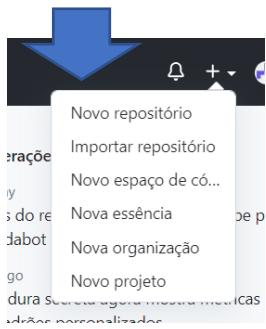
Abra o Arquivo01.txt e digite o texto



Salve o Arquivo01.txt

ETAPA 2. Criar um Repositório no GITHUB

No canto superior direito, próximo ao seu avatar ou ícone de identidade, clique em + e selecione Novo repositório.



Nomeie seu repositório Projeto_teste2.

Escreva uma breve descrição.

Não selecione Inicializar este repositório com um README .

Clique em Criar Repositório

Create a new repository

A repository contains all project files and history. Already have a project repository elsewhere? Import a repository.

Owner * Marcelo-Chacon / Projeto_teste2 ✓

Repository name * Projeto_teste2

Great repository names are short and memorable. Need inspiration? How about didactic-octo-lamp?

Description (optional) Como utilizar git e github

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.

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.

Criar repositório

Após criado o repositório no GITHUB aparecerá as seguintes mensagens:

The screenshot shows a GitHub repository page for 'Marcelo-Chacon / Projeto_teste2'. At the top, there are buttons for Pin, Unwatch, Fork, and Star. Below the header, there's a section titled 'Quick setup — if you've done this kind of thing before' with three options:

- Set up in Desktop** or **HTTPS** **SSH** https://github.com/Marcelo-Chacon/Projeto_teste2.git
- ...or create a new repository on the command line**
- ...or push an existing repository from the command line**
- ...or import code from another repository**

Each option includes a code snippet for the corresponding command-line operation.

O repositório foi criado mas não foi associado com o repositório do git

Devemos então seguir os passos para criar um novo repositório com linhas de comando

Criar um novo repositório com linhas de comando

...or create a new repository on the command line

```
echo "# Projeto_teste2" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Marcelo-Chacon/Projeto_teste2.git
git push -u origin main
```

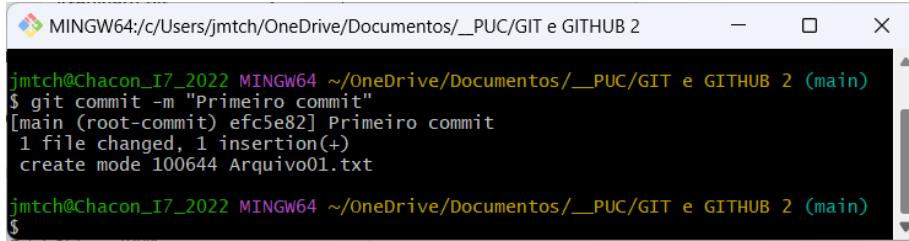
O comando git init já foi realizado anteriormente

O comando git add README.md será substituído por **git add .**

The terminal window shows the following session:

```
MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/GIT e GITHUB 2
jmtch@Chacon_17_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ git add .
jmtch@Chacon_17_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$
```

O comando `git commit -m "first commit"` será substituído por `git commit -m "Primeiro commit"`



```
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ git commit -m "Primeiro commit"
[main (root-commit) efc5e82] Primeiro commit
 1 file changed, 1 insertion(+)
 create mode 100644 Arquivo01.txt

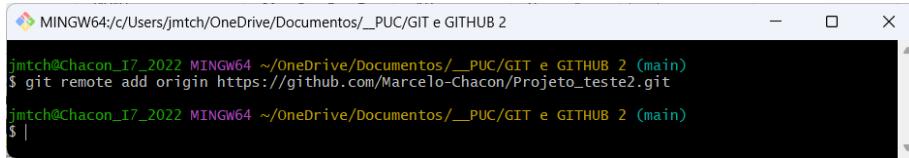
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$
```

O comando `git branch -M main` já foi realizado anteriormente

Copiar o comando do GITHUB:

git remote add origin https://github.com/Marcelo-Chacon/Projeto_teste2.git

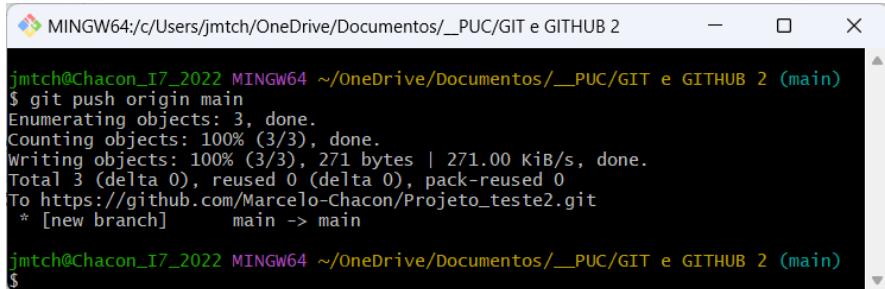
Colar o comando no git (Shift + Insert):



```
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ git remote add origin https://github.com/Marcelo-Chacon/Projeto_teste2.git

jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ |
```

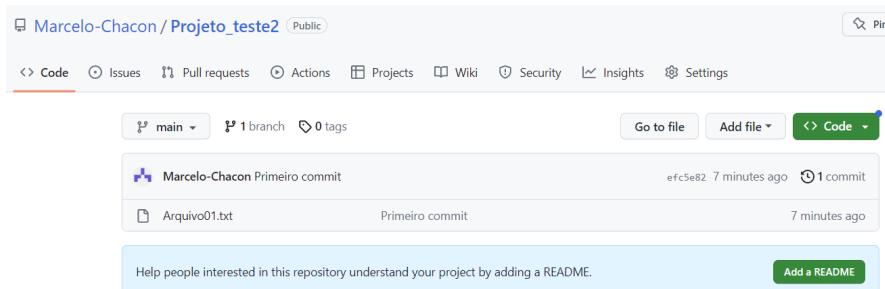
O comando `git push -u origin main` será substituído por `git push origin main`



```
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 271 bytes | 271.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Marcelo-Chacon/Projeto_teste2.git
 * [new branch]      main -> main

jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$
```

Ir para o repositório GITHUB e atualize a página (F5)



Marcelo-Chacon / Projeto_teste2 (Public)

- Code
- Issues
- Pull requests
- Actions
- Projects
- Wiki
- Security
- Insights
- Settings

main 1 branch 0 tags

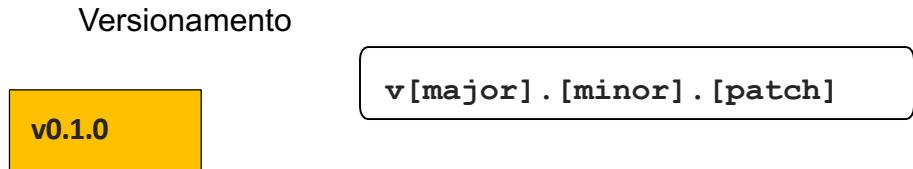
Marcelo-Chacon Primeiro commit
efc5e82 7 minutes ago 1 commit

Arquivo01.txt Primeiro commit 7 minutes ago

Add a README

Criando uma TAG para controle dos versionamentos

O Git tem a capacidade de marcar pontos específicos no histórico de um repositório como sendo importantes. Normalmente, as pessoas usam essa funcionalidade para marcar versões do programa.



[patch]: correção de *bugs*.

[minor]: incrementos de funcionalidades compatíveis com versões anteriores.

[major]: incrementos de funcionalidades incompatíveis com versões anteriores.

Versões teste: alpha (a), beta (b)

Ex: v0.1.9 > v0.1.10 > v0.2.0a > v0.2.0b > v0.2.0

Criando uma tag

Criar uma tag anotada no Git é simples. A maneira mais fácil é especificar quando você executa o comando:

git tag -a v0.1.0 -m "Minha versão 0.1.0"

Onde: -a = cria uma tag para o último commit

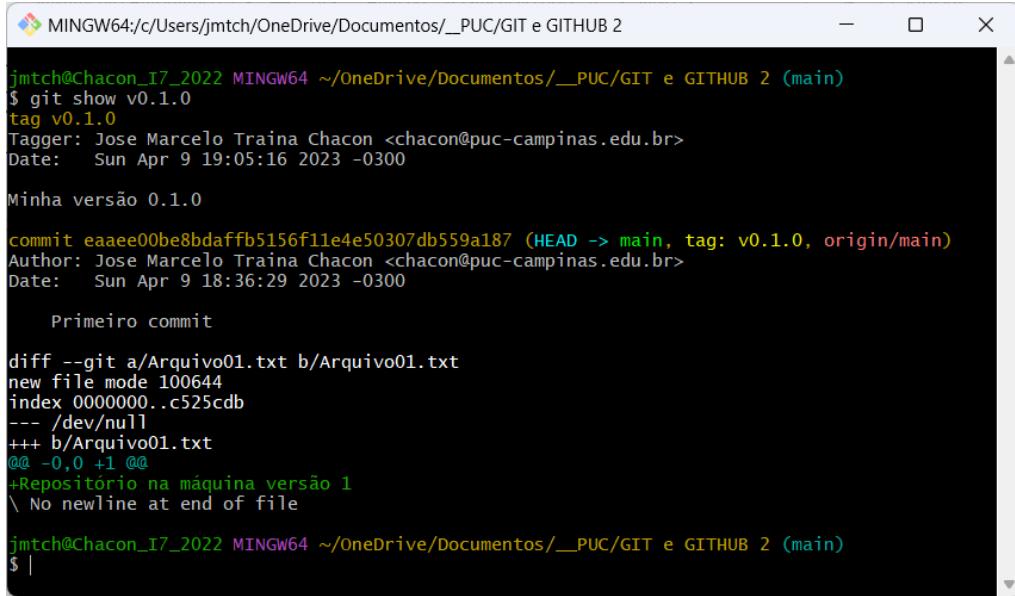
 - m = insere uma mensagem para a tag

```
MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/GIT e GITHUB 2
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ git tag -a v0.1.0 -m "Minha versão 0.1.0"
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ |
```

Para ver as tags existentes, digite: **git tag**

```
MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/GIT e GITHUB 2
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ git tag
v0.1.0
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ |
```

Você pode ver os dados da tag junto com a confirmação que foi comutada usando o comando: `git show v0.1.0`



```

MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/GIT e GITHUB 2
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ git show v0.1.0
tag v0.1.0
Tagger: Jose Marcelo Traina Chacon <chacon@puc-campinas.edu.br>
Date:   Sun Apr 9 19:05:16 2023 -0300

Minha versão 0.1.0

commit eaeee00be8bdaffb5156f11e4e50307db559a187 (HEAD -> main, tag: v0.1.0, origin/main)
Author: Jose Marcelo Traina Chacon <chacon@puc-campinas.edu.br>
Date:   Sun Apr 9 18:36:29 2023 -0300

    Primeiro commit

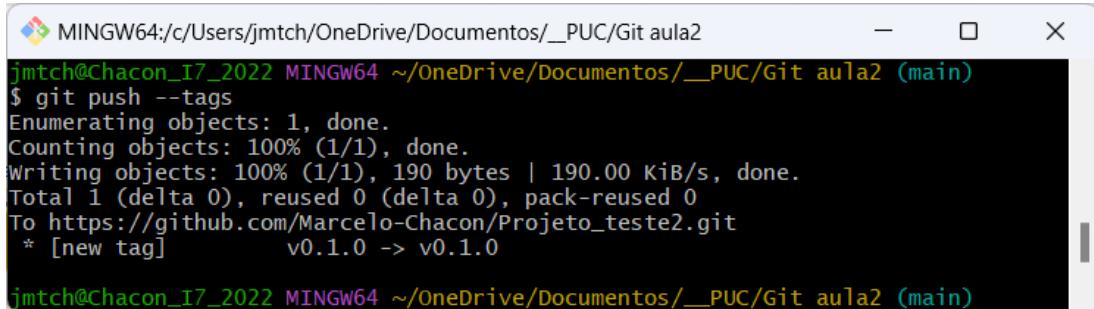
diff --git a/Arquivo01.txt b/Arquivo01.txt
new file mode 100644
index 000000..c525cdb
--- /dev/null
+++ b/Arquivo01.txt
@@ -0,0 +1 @@
+Repositório na máquina versão 1
\\ No newline at end of file

jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ |

```

Compartilhar as TAGS com o repositório remoto GITHUB, utilize o comando:

`git push --tags`



```

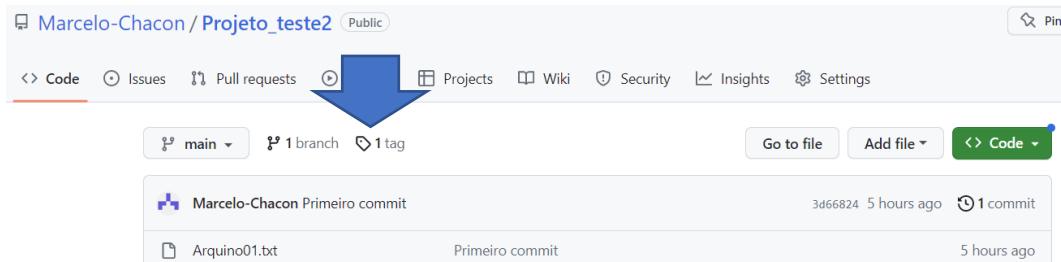
MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/Git aula2
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/Git aula2 (main)
$ git push --tags
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 190 bytes | 190.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Marcelo-Chacon/Projeto_teste2.git
 * [new tag]           v0.1.0 -> v0.1.0

jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/Git aula2 (main)

```

Ir para o repositório GITHUB e atualize a página (F5)

Clique em **tags**



Marcelo-Chacon / Projeto_teste2 Public

Code Issues Pull requests Projects Wiki Security Insights Settings

main · 1 branch · 1 tag

Marcelo-Chacon Primeiro commit 3d66824 5 hours ago 1 commit

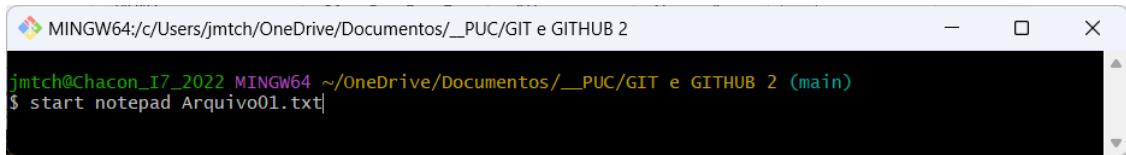
Arquino01.txt Primeiro commit 5 hours ago

Visualização das tags no GITHUB

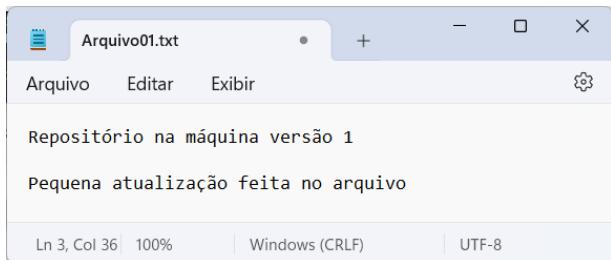
The screenshot shows a GitHub repository interface. At the top, there's a header with the repository name 'Marcelo-Chacon / Projeto_teste2' and various navigation links like 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. Below this, a tab bar has 'Releases' and 'Tags' buttons; 'Tags' is currently selected. Under the 'Tags' section, there's a list with one item: 'v0.1.0'. Below the tag name, it says '5 hours ago' and shows download links for 'zip' and 'tar.gz'.

Verificar alterações realizadas nos arquivos do diretório de trabalho – git diff

Abra o Arquivo01.txt

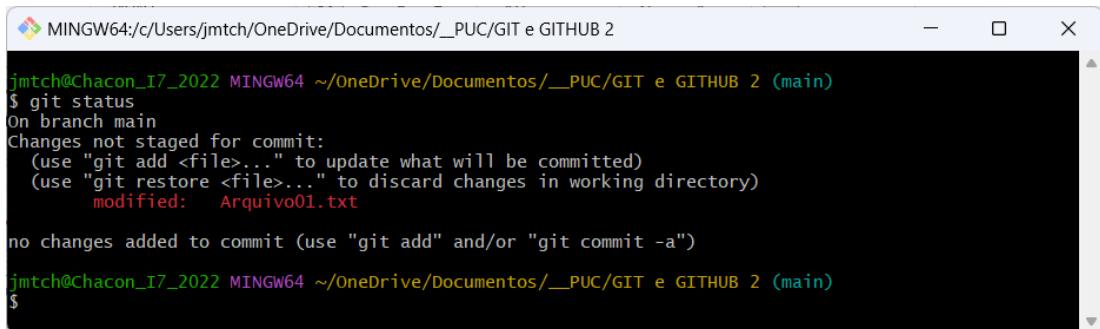


Acrescente a frase: Pequena atualização feita no arquivo

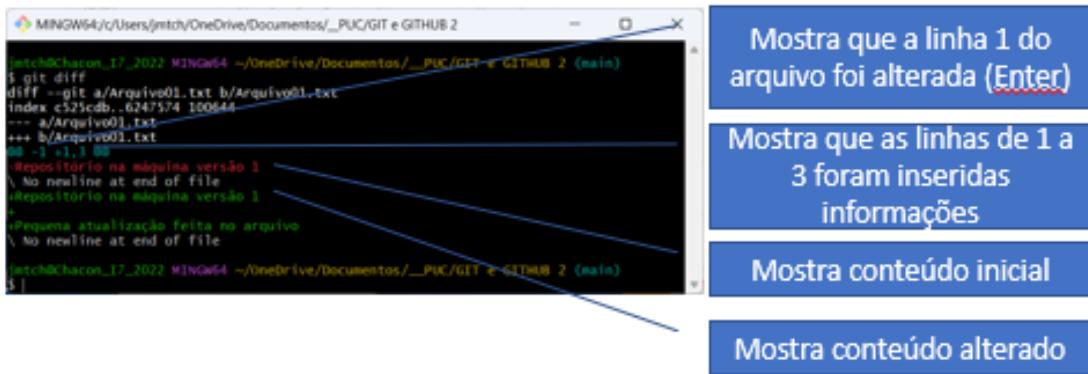


Salve o arquivo

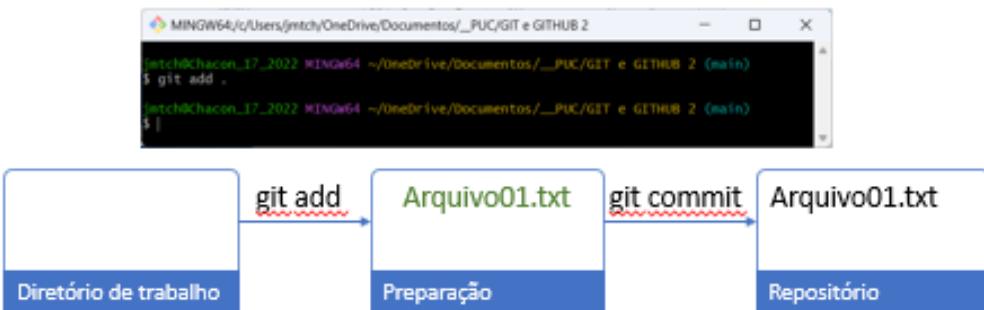
Veja o status: git status



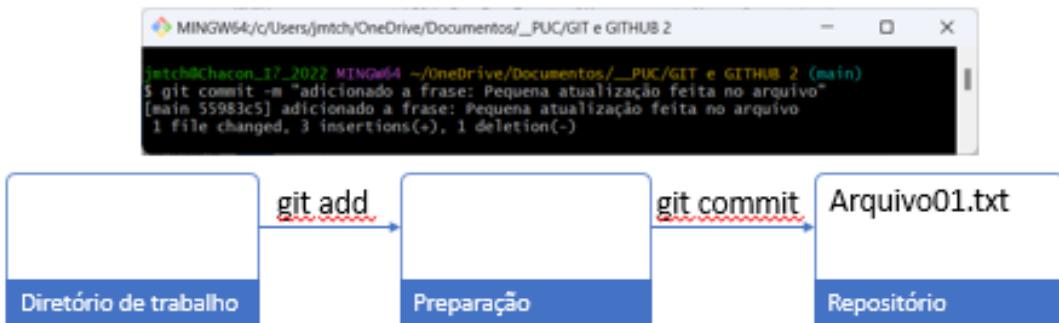
Digite **git diff** para comparar o arquivo do Diretório de trabalho com o do Repositório



Adicionar o arquivo alterado para a área de preparação: **git add .**



Transferir o arquivo para o repositório local:
git commit – m “adicionado a frase: Pequena atualização feita no arquivo”



Atualizando o tag do repositório:

git tag -a v0.1.1 -m "Minha versão 0.1.1"

```
MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/GIT e GITHUB 2
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ git tag -a v0.1.1 -m "Minha versão 0.1.1"
```

Verificar as tags do Arquivo01.txt: **git tag**

```
MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/GIT e GITHUB 2
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ git tag
v0.1.0
v0.1.1
```

Salvando as alterações para o repositório remoto:

git push origin main e a: git push --tags

```
MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/GIT e GITHUB 2
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 374 bytes | 374.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Marcelo-Chacon/Projeto_teste2.git
  ff66b75..55983c5 main -> main

MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/Git aula2
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/Git aula2 (main)
$ git push --tags
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 190 bytes | 190.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Marcelo-Chacon/Projeto_teste2.git
 * [new tag]          v0.1.1 -> v0.1.1

jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/Git aula2 (main)
```

Pode ser confirmado no GITHUB (não esqueça de atualizar a página do GITHUB

– F5)

Marcelo-Chacon / Projeto_teste2 (Public)

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 2 tags

Go to file Add file <> Code

Marcelo-Chacon adicionado a frase: Pequena atualização feita no arquivo 55983c5 2 hours ago 2 commits

Arquivo01.txt adicionado a frase: Pequena atualização feita no arquivo 2 hours ago

Restaurando uma versão

O comando **git revert** é uma operação de desfazer avançada que oferece um método seguro de desfazer alterações.



1. Para restaurar a versão v 0.1.0, digite: **git revert v0.1.0**

```
MINGW64S:\Users\mitch\OneDrive\Documentos\_Puc\GIT e GITHUB 2 - x
1.git revert v0.1.0
CONFLICT (modify/delete): Archivo01.txt deleted in (empty tree) and modified in HEAD
error: Version of 'Archivo01.txt' on disk is newer than the version in HEAD
error: Could not merge 'Archivo01.txt'. Please, commit
hint: After resolving the conflicts, mark them with
hint:   "git add/rm <pathspec>"; then run
hint:   "git commit --fixup <commit>" to skip this
hint: or you can instead skip this commit with "git revert --skip".
hint: To abort and get back to the state before "git revert",
hint: run "git revert --abort".

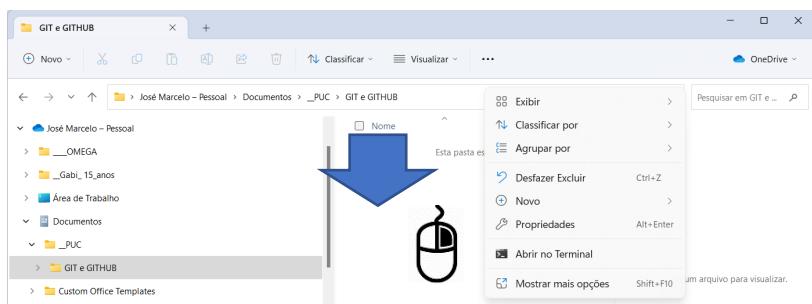
Archivos:charon_17_2022 MINGW64 ~\OneDrive\Documentos\_Puc\GIT e GITHUB 2 (main) [REVIS]
```



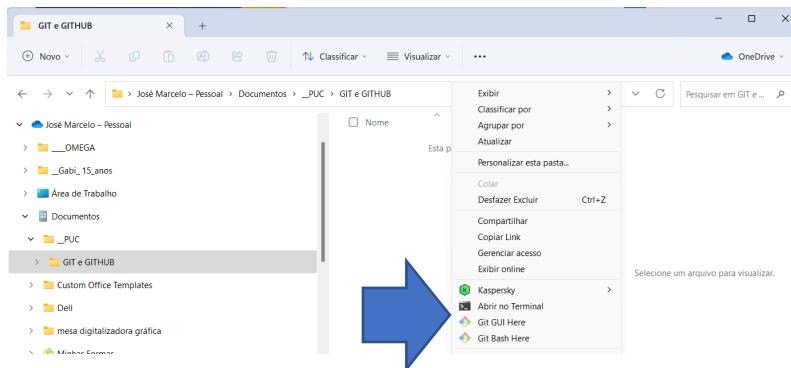
Conflitos aparecem, pois na versão 0.1.1 existem modificações feitas no arquivo 01.txt

Para resolver o conflito, iremos abrir o GIT GUI here

Clique com o botão direito do mouse no visualizador de pastas e arquivos, clique em mais opções



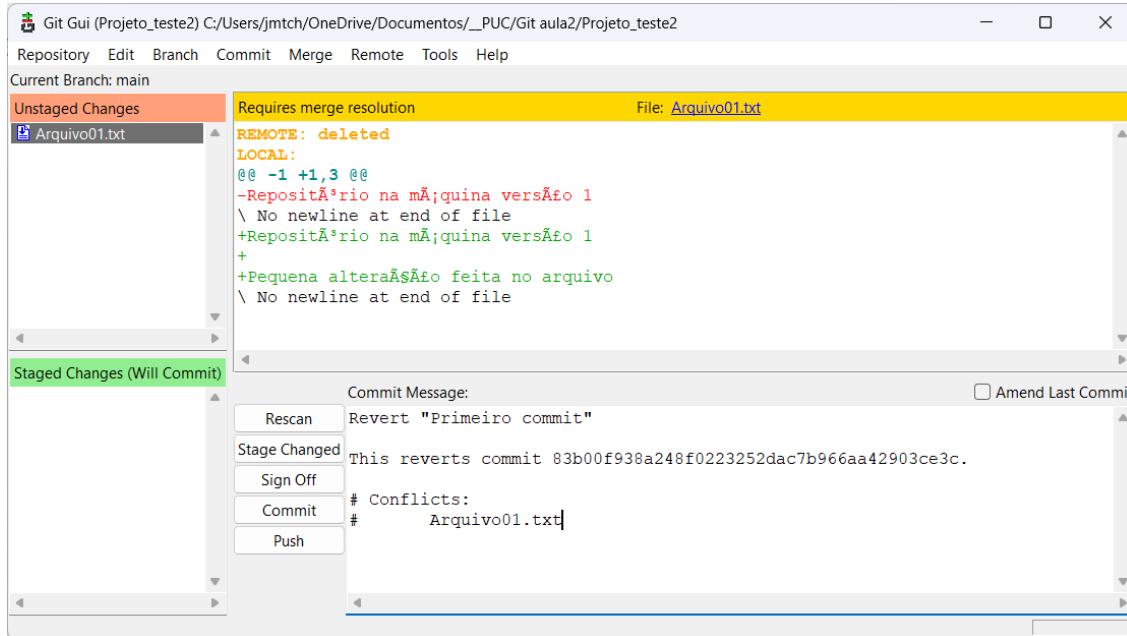
Clique em Git GUI Here



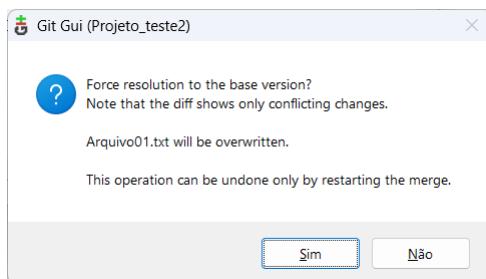
Há um conflito entre as versões do arquivo01.txt

Clique com o botão direito na área de Requires merge resolution e escolha a opção

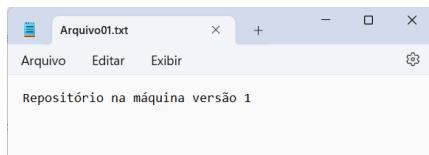
Revert To Base



Clique em sim e feche o GIT GUI



Abra o arquivo01.txt e verifique que a versão foi restaurada



Volte para o GIT Bach Here e digite: **git revert --continue**

```
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/Git aula2/projeto_teste2 (main|REVERTING)
$ git revert --continue
[main 60e3cc0] Revert "Primeiro commit"
1 file changed, 1 insertion(+), 3 deletions(-)

jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/Git aula2/projeto_teste2 (main)
$
```

Digite git log para ver o commit realizado

```
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/Git aula2/projeto_teste2 (main)
$ git log
commit 60e3cc077097088144439f26a878ae23a9b41962 (HEAD -> main)
Author: Jose Marcelo Traina Chacon <chacon@puc-campinas.edu.br>
Date:   Wed Apr 19 19:18:50 2023 -0300

    Revert "Primeiro commit"

    This reverts commit 83b00f938a248f0223252dac7b966aa42903ce3c.

commit 59bfd46caabd36c912ed6780bbcd092ce6797635 (tag: v0.1.1, origin/main, origin/HEAD)
Author: Jose Marcelo Traina Chacon <chacon@puc-campinas.edu.br>
Date:   Mon Apr 17 19:27:21 2023 -0300

    adicionando a frase: Pequena atualização feita no arquivo

commit 83b00f938a248f0223252dac7b966aa42903ce3c (tag: v0.1.0)
Author: Jose Marcelo Traina Chacon <chacon@puc-campinas.edu.br>
Date:   Mon Apr 17 19:21:14 2023 -0300

    Primeiro commit

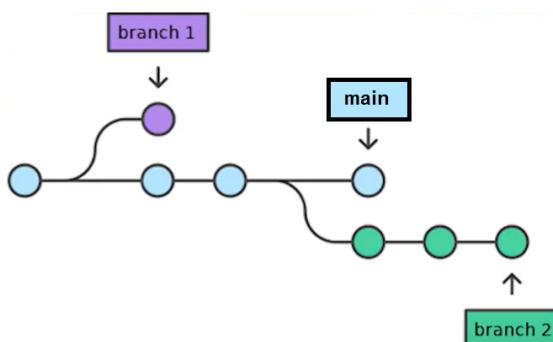
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/Git aula2/projeto_teste2 (main)
$ |
```

Criando uma ramificação no projeto Branch

Ramificação é a maneira de trabalhar em diferentes versões de um repositório ao mesmo tempo.

Por padrão, seu repositório tem um branch denominado **main** que é considerado o branch definitivo. Usamos branches para experimentar e fazer edições antes de enviá-los a **main**.

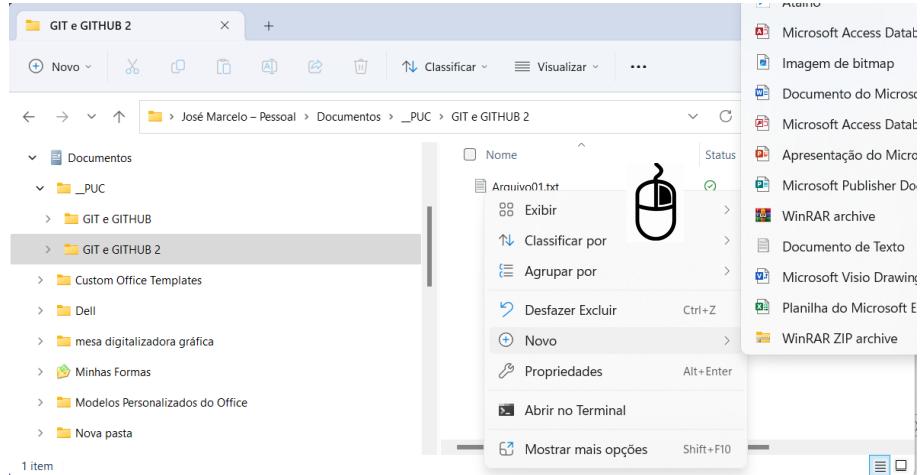
Ao criar um branch fora do branch **main**, você está fazendo uma cópia, ou **snapshot**, de **main** como era naquele momento. Se outra pessoa fez alterações no branch **main**, enquanto você estava trabalhando em seu branch, você poderia puxar essas atualizações.



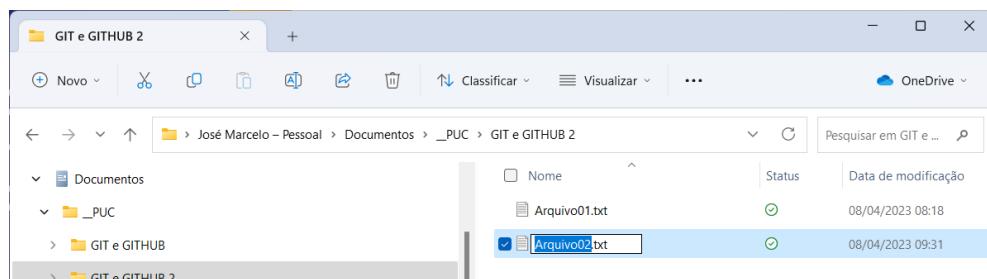
Vamos utilizar o repositório Projeto_teste2 para criar as ramificações

ETAPA 1. Criar um arquivo no repositório GIT

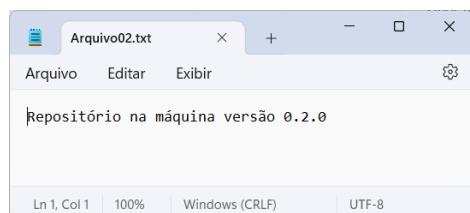
Criar um arquivo TXT no repositório Git



Renomear o arquivo para Arquivo02.txt e pressionar Enter



Abra o Arquivo02.txt e digite o texto



Salve o Arquivo02.txt



ETAPA 2. Criar uma ramificação

Para criar a ramificação digite o comando: **git branch ramo01**

```
MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/GIT e GITHUB 2
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ git branch ramo01

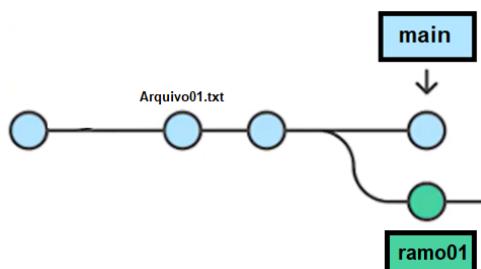
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ |
```

Verificar as ramificações existentes no repositório, digite o comando: **git branch**

```
MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/GIT e GITHUB 2
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ git branch
* main
  ramo01

jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ |
```

O git mostra que existe 2 branches e mostra com * e escrito em verde a branch ativa, ou seja a branch main

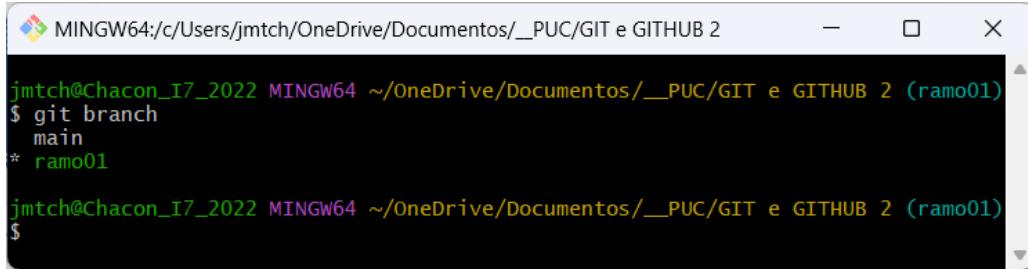


Para entrar na branch ramo01, digite o comando:
git checkout ramo01

```
MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/GIT e GITHUB 2
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ git checkout ramo01
Switched to branch 'ramo01'

jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (ramo01)
$ |
```

Verificar as ramificações existentes no repositório, digite o comando: **git branch**



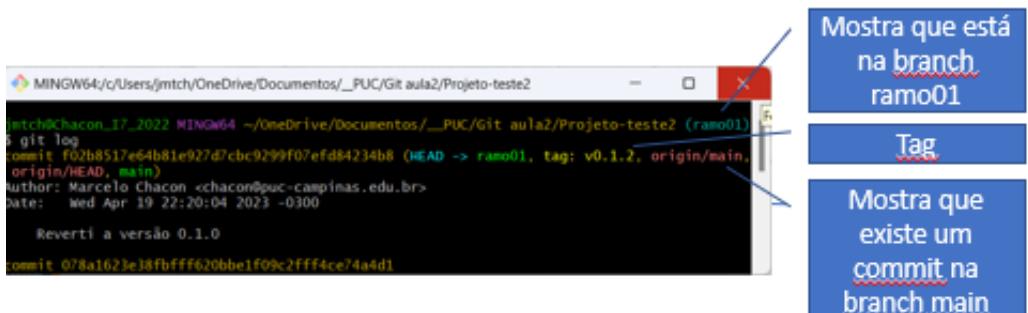
```
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (ramo01)
$ git branch
  main
* ramo01

jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (ramo01)
$
```

O git mostra que existe 2 branchs e mostra com * e escrito em verde a branch ativa, ou seja a branch ramo01

ETAPA 3. Commitar o arquivo para o ramo01

Verificar o log do ramo01, digite o comando: **git log**



```
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/Git aula2/Projeto-teste2 (ramo01)
$ git log
commit f02b8517e64b81e927d7cbc9299f07efd84234b8 (HEAD -> ramo01, tag: v0.1.2, origin/main, origin/HEAD, main)
Author: Marcelo Chacon <chacon@puc-campinas.edu.br>
Date:   Wed Apr 19 22:20:04 2023 -300

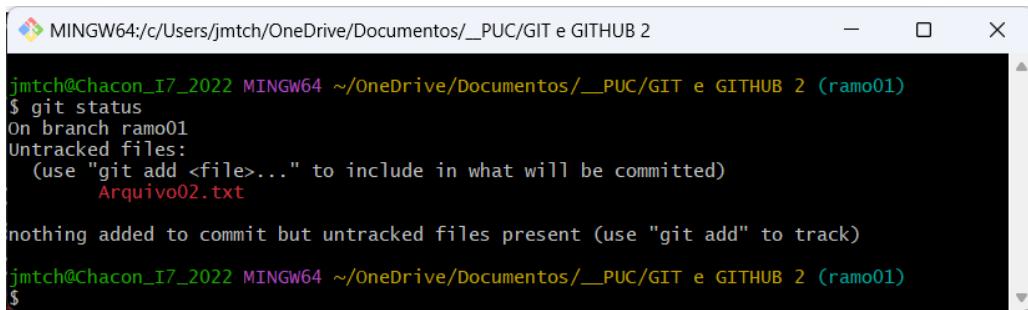
    Reverti a versão 0.1.0
commit 078a1623e38fbfff620bbe1f09c2fff4ce74a4d1
```

Mostra que está na branch ramo01

Tag

Mostra que existe um commit na branch main

Verificar o status da branch, digite **git status**



```
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (ramo01)
$ git status
On branch ramo01
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Arquivo02.txt

nothing added to commit but untracked files present (use "git add" to track)

jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (ramo01)
$
```

Mostra que tem um arquivo não commitado no repositório

Adicionar o Arquivo02.txt para a área de Preparação, digite o comando: **git add .**

```
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (ramo01)
$ git add .

jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (ramo01)
$ |
```

Verifique o status do repositório, digite: **git status**

```
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (ramo01)
$ git status
On branch ramo01
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   Arquivo02.txt

jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (ramo01)
$ |
```



Criando o versionamento do projeto no Git, digite o comando:
git commit -m “Versão 0.2.0, ramo01”

```
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (ramo01)
$ git commit -m "Versão 0.2.0, ramo01"
[ramo01 1cddb64] Versão 0.2.0, ramo01
 1 file changed, 1 insertion(+)
 create mode 100644 Arquivo02.txt

jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (ramo01)
$ |
```

Para verificar os log de commits, digite: **git log**

```
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (ramo01)
$ git log
commit 3cd8d495e4d321e7481ed728815ac1b2fc7566 (HEAD -> ramo01)
Author: Jose Marcelo Tráina Chacon <chacon@puc-campinas.edu.br>
Date:  Sun Apr 9 19:00:46 2023 -0300
  versão 0.2.0, ramo01

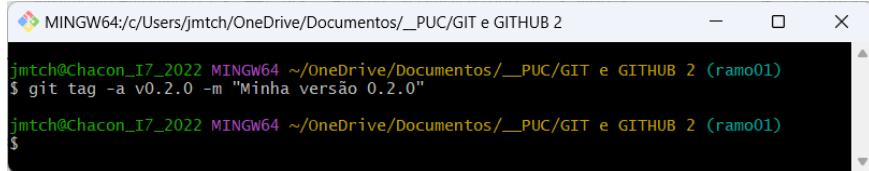
commit eace00be06d0ff5156f11e4e50307db519a187 (tag: v0.1.0, origin/main, main)
Author: Jose Marcelo Tráina Chacon <chacon@puc-campinas.edu.br>
Date:  Sun Apr 9 18:36:29 2023 -0300
  Primeiro commit

jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (ramo01)
$ |
```

ETAPA 3. Controlar o versionamento

Criar uma tag para a versão V0.2.0 da branch ramo01, com o comando:

git tag -a v0.2.0 -m "Minha versão 0.2.0"



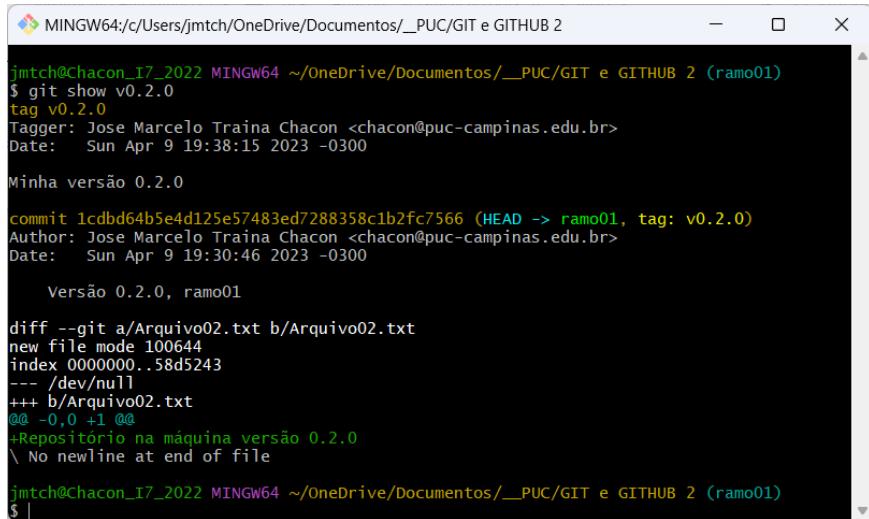
```
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (ramo01)
$ git tag -a v0.2.0 -m "Minha versão 0.2.0"
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (ramo01)
$
```

Ver as tags existentes: **git tag**



```
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/Git aula2/Projeto-teste2 (ramo01)
$ git tag
v0.1.0
v0.1.1
v0.2.0
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/Git aula2/Projeto-teste2 (ramo01)
$
```

Confirmação da tag comutada: **git show v0.2.0**



```
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (ramo01)
$ git show v0.2.0
tag v0.2.0
Tagger: Jose Marcelo Traina Chacon <chacon@puc-campinas.edu.br>
Date:   Sun Apr 9 19:38:15 2023 -0300

Minha versão 0.2.0

commit 1cdbd64b5e4d125e57483ed7288358c1b2fc7566 (HEAD -> ramo01, tag: v0.2.0)
Author: Jose Marcelo Traina Chacon <chacon@puc-campinas.edu.br>
Date:   Sun Apr 9 19:30:46 2023 -0300

    Versão 0.2.0, ramo01

diff --git a/Arquivo02.txt b/Arquivo02.txt
new file mode 100644
index 0000000..58d5243
--- /dev/null
+++ b/Arquivo02.txt
@@ -0,0 +1 @@
+Repositório na máquina versão 0.2.0
\ No newline at end of file
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (ramo01)
$ |
```

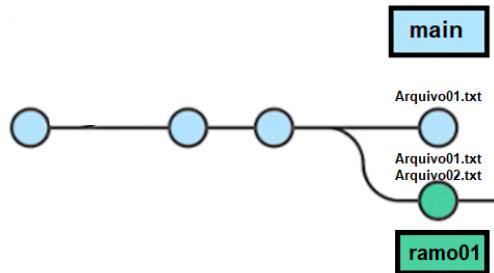
ETAPA 4. Enviar o repositório local GIT para o repositório GITHUB

Enviar a ramificação ramo01 para o repositório GITHUB, utilize o comando:

`git push origin ramo01`

```

MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/GIT e GITHUB 2
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (ramo01)
$ git push origin ramo01
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 335 bytes | 335.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'ramo01' on GitHub by visiting:
remote:   https://github.com/Marcelo-Chacon/Projeto_teste2/pull/new/ramo01
remote:
To https://github.com/Marcelo-Chacon/Projeto_teste2.git
 * [new branch]      ramo01 -> ramo01
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (ramo01)
$ 
```



Verificando as branches no GITHUB

Atualize a página do GITHUB (F5)

Mostra que houve um push recente no ramo01

Mostra que existe 2 branches no repositório

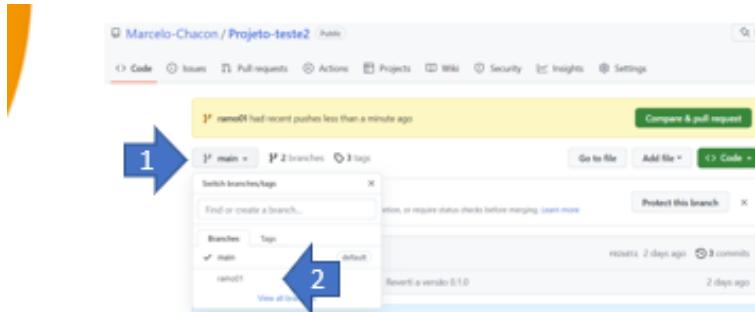
Mostra que está na branch main

Arquivo da branch main

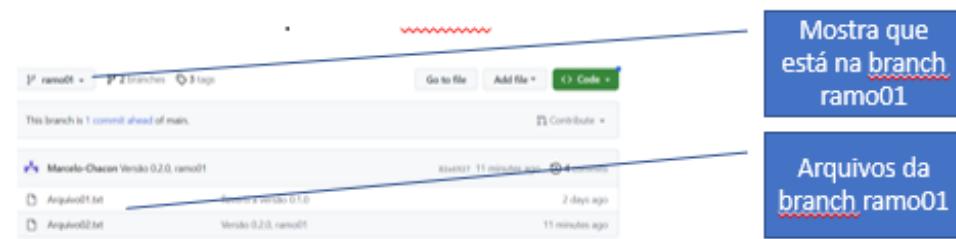
Verificando as branches no GITHUB

Mudar para a branch ramo01, clique em caixa de seleção de branch

Escolha a branch ramo01



Visualiza os arquivos da branch ramo01



Mesclando branches

Após concluir o desenvolvimento de novas funções para o sistema na branch ramo01, iremos incorporá-la na branch main.

Selecione a branch onde será incorporada as modificações:
git checkout main

```
MINGW64:/c/Users/jmtch/OneDrive/Documentos/_PUC/GIT e GITHUB 2
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (ramo01)
$ git checkout main
Switched to branch 'main'

jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$
```

Incorpore as modificações realizadas na branch ramo01 para a branch main:

git merge ramo01

```
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ git merge ramo01
Updating eaaee00..1cdbd64
Fast-forward
 Arquivo02.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 Arquivo02.txt

jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$
```

Verificar o status da branch: **git status**

```
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ git status
On branch main
nothing to commit, working tree clean

jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$
```

Verifique que não há commits a serem realizados nesta branch, pois eles já foram realizados na branch ramo01

Enviar as alterações para o repositório remoto GITHUB:
git push origin main

```
jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$ git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Marcelo-Chacon/Projeto teste2.git
 eaaee00..1cdbd64 main -> main

jmtch@Chacon_I7_2022 MINGW64 ~/OneDrive/Documentos/_PUC/GIT e GITHUB 2 (main)
$
```

Verificando as branches no GITHUB

Atualize a página do GITHUB (F5)

Marcelo-Chacon / Projeto-teste2 · Public

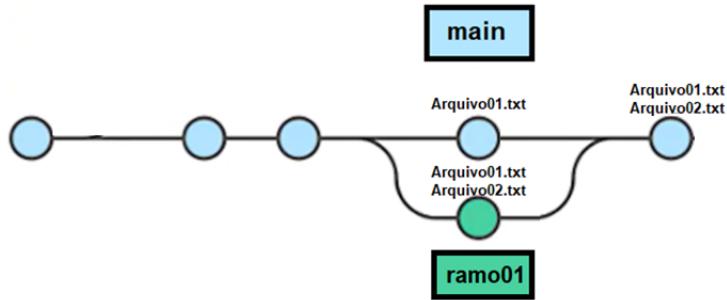
Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 2 branches 3 tags

Your main branch isn't protected
Protect this branch

Author	Commit Message	Date	Commits
Marcelo-Chacon	Versão 0.2.0, ramo01	82e6927 14 minutes ago	4 commits
	Arquivo01.txt	Reverti a versão 0.1.0	2 days ago
	Arquivo02.txt	Versão 0.2.0, ramo01	14 minutes ago

Branch main foi atualizada, mostrando os arquivos Arquivo01.txt e Arquivo02.txt



Sugestão de fluxo de trabalho

master ou main – versão estável.

hotfix - Correção de bugs da versão estável.

release - Teste e correções de versões.

develop - Desenvolvimento

feature - Implementação de funcionalidades.



master – versão estável.

hotfix - Correção de bugs da versão estável.

release -Teste e correções de versões.

develop - Desenvolvimento

feature - Implementação de funcionalidades.



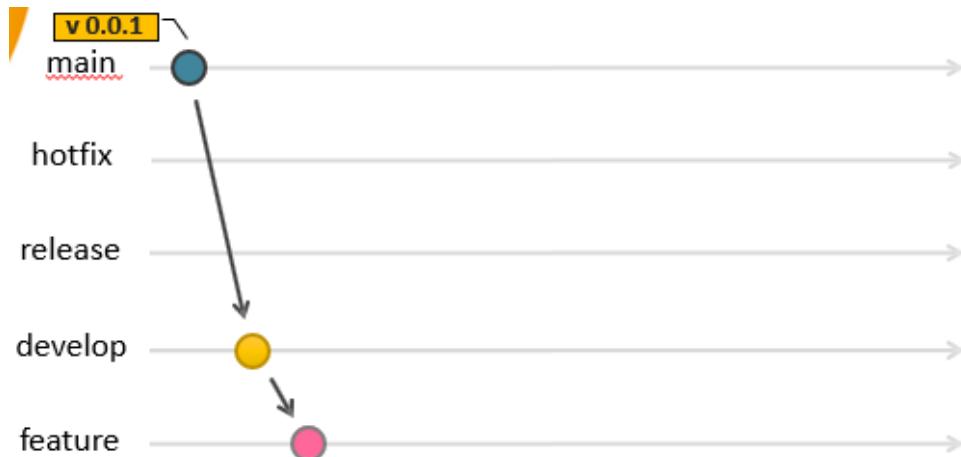
master – versão estável.

hotfix - Correção de bugs da versão estável.

release -Teste e correções de versões.

develop - Desenvolvimento

feature - Implementação de funcionalidades.



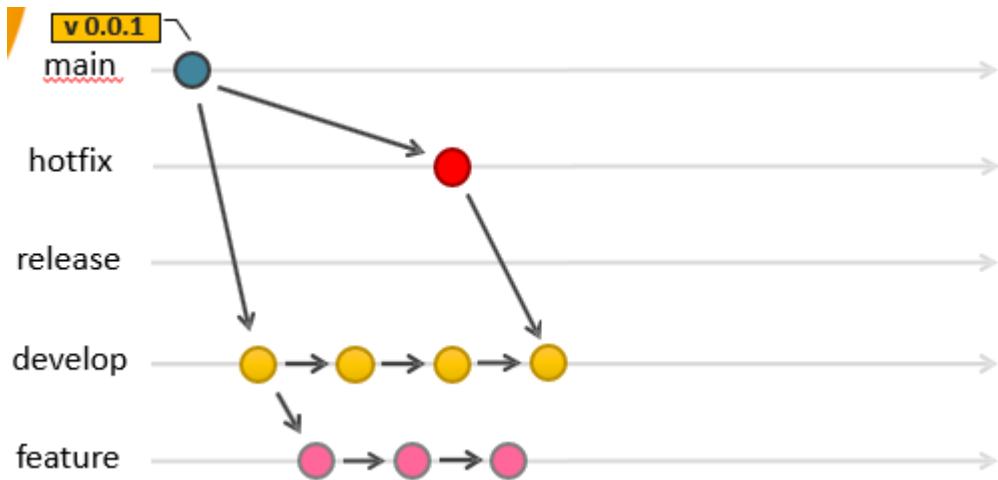
master – versão estável.

hotfix - Correção de bugs da versão estável.

release - Teste e correções de versões.

develop - Desenvolvimento

feature - Implementação de funcionalidades.



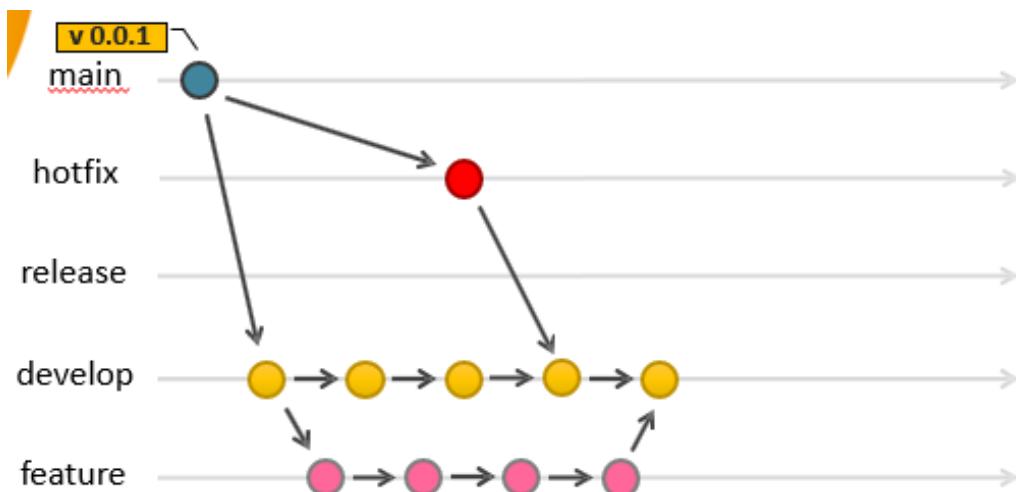
master – versão estável.

hotfix - Correção de bugs da versão estável.

release - Teste e correções de versões.

develop - Desenvolvimento

feature - Implementação de funcionalidades.



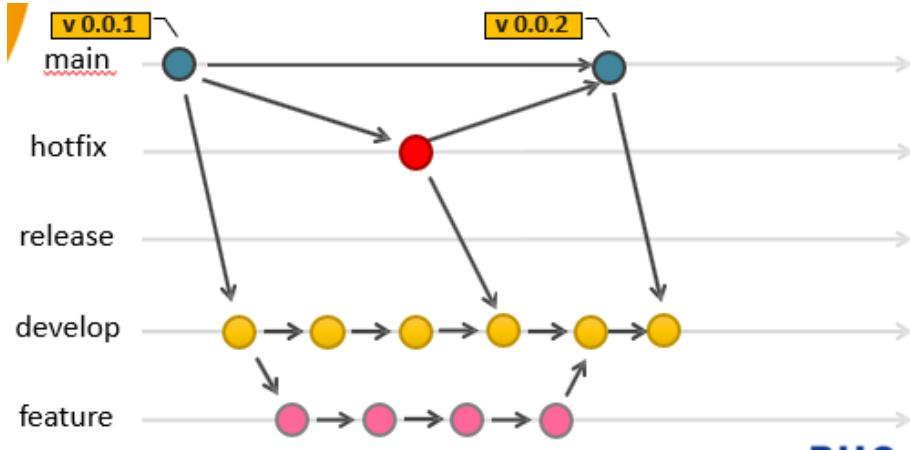
master – versão estável.

hotfix - Correção de bugs da versão estável.

release - Teste e correções de versões.

develop - Desenvolvimento

feature - Implementação de funcionalidades.



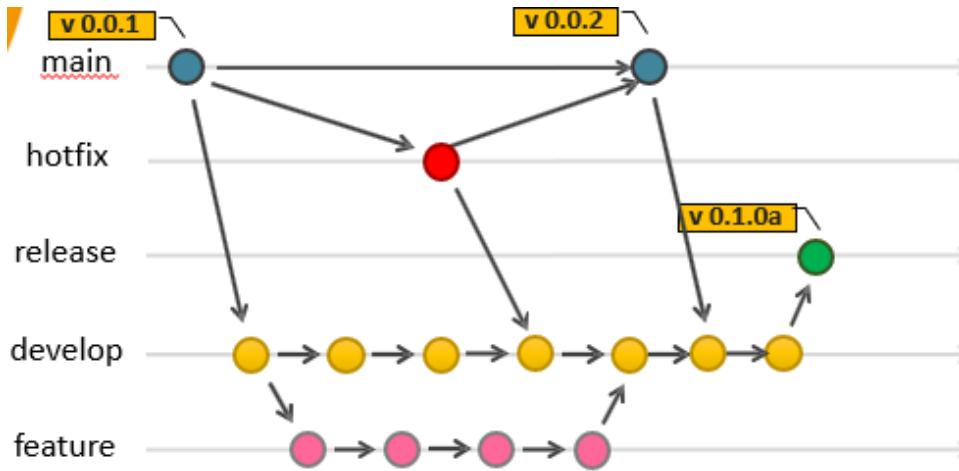
master – versão estável.

hotfix - Correção de bugs da versão estável.

release - Teste e correções de versões.

develop - Desenvolvimento

feature - Implementação de funcionalidades.



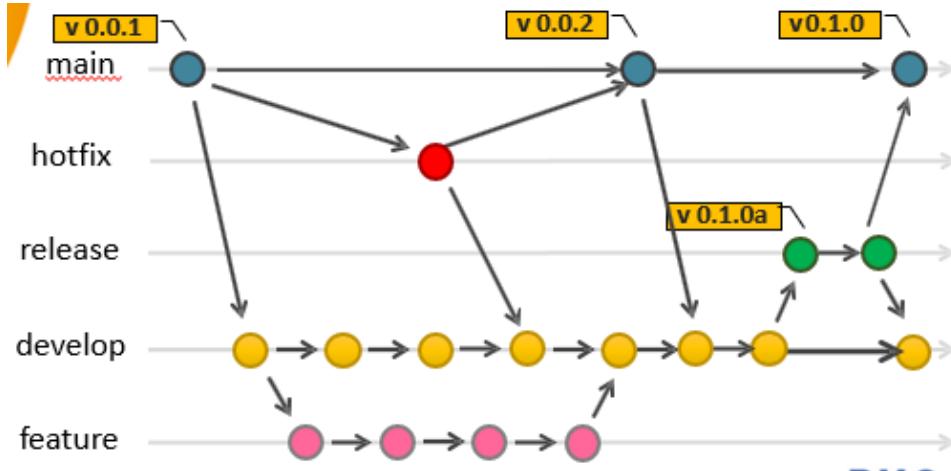
master – versão estável.

hotfix - Correção de bugs da versão estável.

release - Teste e correções de versões.

develop - Desenvolvimento

feature - Implementação de funcionalidades.



Continue estudando GIT e GITHUB

Referências do git

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

Livro Pro git

- <https://git-scm.com/book/pt-br/v2>

Trabalhando em equipe com Git Flow

- <https://www.youtube.com/watch?v=394mc6PV8t8&list=PLR8JXremim5BNbLUpzYEi3Xnb790ttsE1&index=7>

Como trabalhar com Git e GitHub no VsCode | Tutorial

- <https://www.youtube.com/watch?v=HlqyLRKy-YE&list=PLR8JXremim5BNbLUpzYEi3Xnb790ttsE1&index=4>

Referência Bibliográfica

CONSERVANCY, Software Freedom. **Guia de Referência GIT**. Disponível em: <https://git-scm.com/site>. Acesso em: 01 fev. 2024.

BAFFA, Augusto. **INF 1771 – Inteligência Artificial**: aula xx - projetos e controle de versões. Rio de Janeiro: Pontifícia Universidade Católica do Rio de Janeiro, 2024. 32 slides, color. Disponível em: http://augustobaffa.pro.br/site/wp-content/uploads/2016/09/IA_Aula_xx_Controle_de_versoes_2015.2.pdf. Acesso em: 01 fev. 2024.

IPSENSE. **Entenda a importância do versionamento de software**. 2020. Disponível em: <https://www.ipsense.com.br/amazon-web-services/entenda-a-importancia-do-versionamento-de-software/>. Acesso em: 01 fev. 2024.