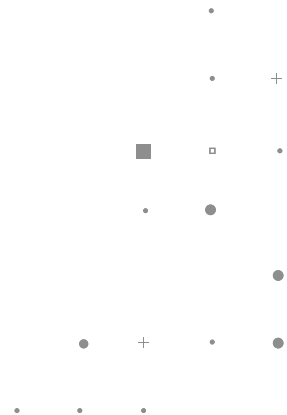




# FIAP

## GRADUAÇÃO



TDS

# Front-End Design Engineering

Prof. Alexandre Carlos - [profalexandre.jesus@fiap.com.br](mailto:profalexandre.jesus@fiap.com.br)

Prof. Luís Carlos - [lsilva@fiapcom.br](mailto:lsilva@fiapcom.br)

## Git e Github

- O que é controle de versão
- Git – principais comandos
- Github – repositório remoto



## O que é controle de versão?

O controle de versão é um sistema que é utilizado para gerenciar alterações em programas de computador, documentos, grandes aplicações web ou outros projetos.

Ele resolve problemas como:

- 1 – A necessidade de fazer cópias e mais cópias de um mesmo projeto** – Ter que fazer cópias do projeto a cada alteração importante, temendo ter a necessidade de voltar ao estado anterior.
- 2 – Acidentes durante o desenvolvimento** – Copiar, sobrepor ou até mesmo apagar um trecho de código, ou até mesmo um ou vários arquivos e não conseguir recuperar.



## O que é controle de versão?

No mercado temos basicamente dois tipos de versionadores:

**1 – Que verifica e salva os arquivos que apresentam diferenças em versões e não possibilita versões paralelas.**

Ex: CVS, Subversion, etc.

**2 – Git ele faz snapshots (como se fosse uma foto que registra o estado naquele momento).**

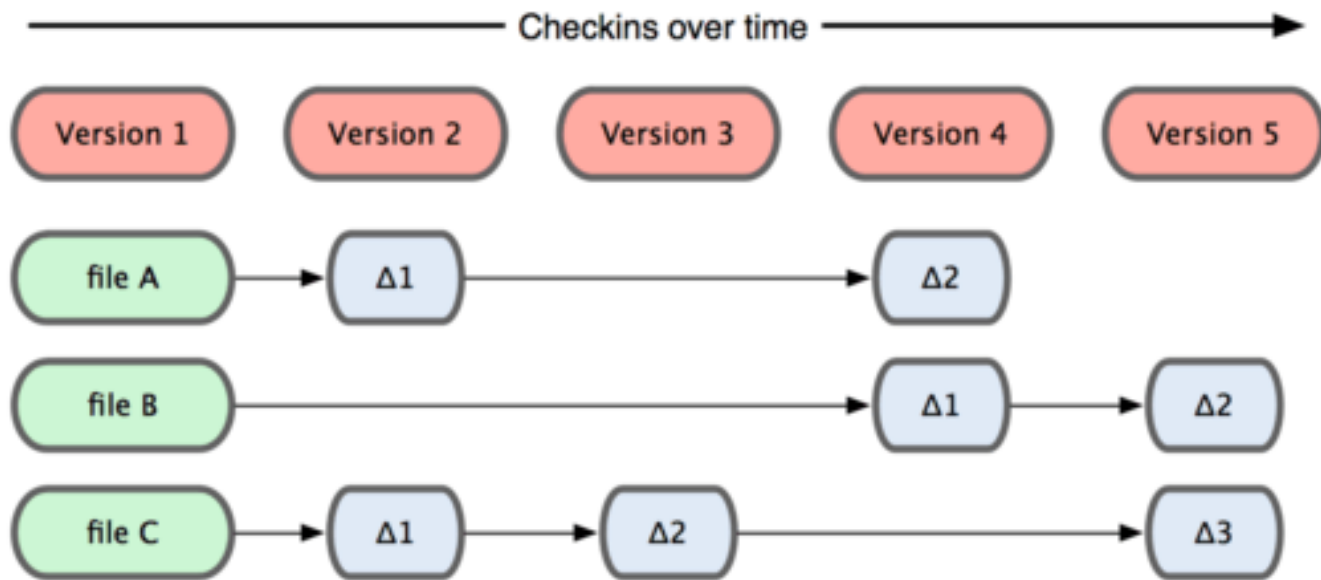
Ex: Git





# O que é controle de versão?

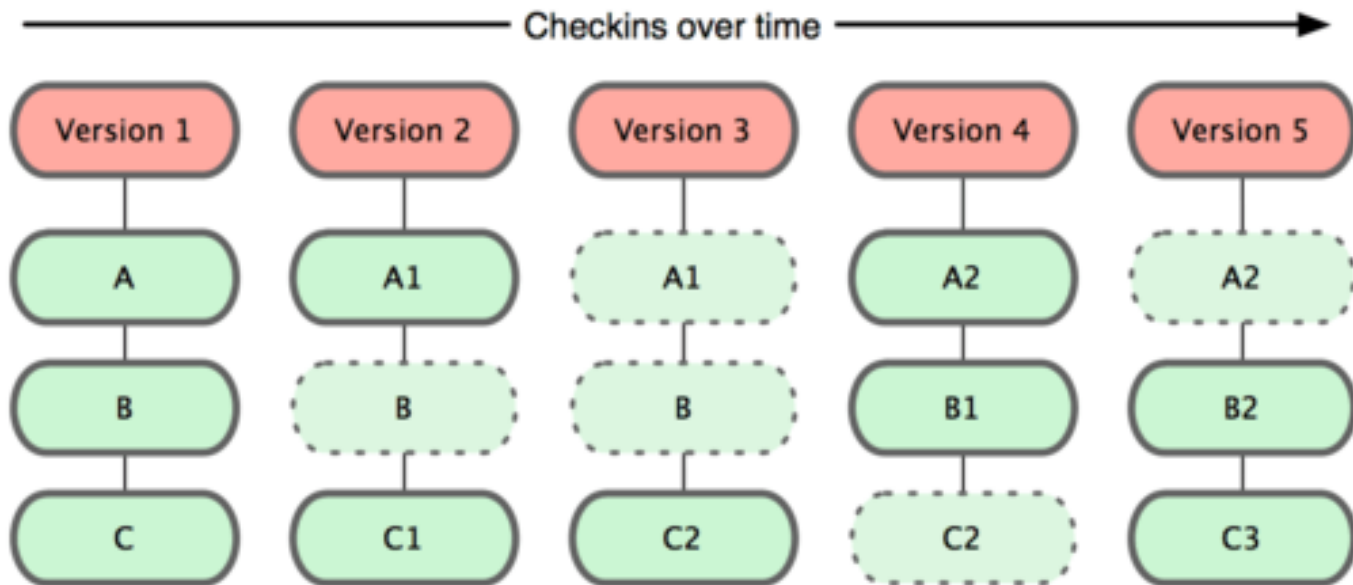
- Outros Sistemas:





# O que é controle de versão?

- Sistema Git:





## Um pouco sobre o Git

O Git foi criado por Linus Torvalds, o mesmo criador do sistema operacional Linux, depois de uma quebra de contrato com a BitKeeper, empresa que fornecia o sistema de versionamento para eles na época.

Insatisfeito com valores e desempenho da ferramenta, ele quebrou o contrato e decidiu criar o seu próprio versionador, trazendo melhorias como:

- Velocidade e espaço de armazenamento;
- Design Simples, mais fácil de utilizar;
- Robustez, permite a criação de milhares versionamentos paralelos;
- Capaz de lidar com grandes projetos





Importante!!!



git



github  
SOCIAL CODING

# O que é Github?



É um serviço web utilizado para compartilhar projetos que utilizam o Git para versionamento, se tornando assim uma rede social para desenvolvedores.

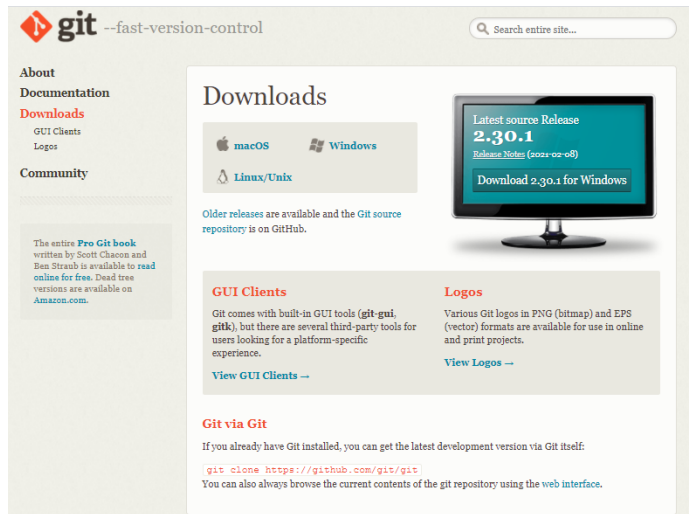
Sua função principal é armazenar projetos na web versionados pelo Git.



# Instalação do Git

A instalação do Git é muito simples, basicamente é só ir seguindo as orientações do aplicativo de instalação e ir dando Next.

Para fazer o download é só acessar o endereço: <https://git-scm.com/download>





## Configurações básicas do Git

Quando começamos a utilizar o Git pela primeira vez em nossa máquina ou vamos utilizar uma máquina compartilhada com outras pessoas, é importante fazer algumas configurações básicas como:

Nome do Usuário:

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho  
$ git config --global user.name "lcsilva76"
```

E-mail do Usuário:

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho  
$ git config --global user.email "lsilva@fiap.com.br"
```



## Configurações básicas do Git

Para verificar as configurações atuais use:

Nome do Usuário: `lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho`  
`$ git config user.name`

`lcsilva76`

Resposta

E-mail do Usuário: `lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho`  
`$ git config user.email`

`lsilva@fiap.com.br`

Resposta



## Repositórios

Para que o Git possa fazer o controle dos nossos projetos, precisamos identificar a sua pasta (diretório) como um repositório, assim ele sabe que deve monitorar e fazer o controle dela.

Para vermos como funciona, crie uma pasta no seu desktop chamada “Exemplo-Git”;

Você pode fazer direto de dentro do git bash (terminal) usando o comando:  
`mkdir exemplo-Git`

Acesse a pasta com o comando: `cd exemplo-Git`

Para inicializar a pasta como repositório digite: `git init`

Para visualizar o conteúdo do repositório digite: `ls -la`

OBS. Cuidado, só use o comando `git init` dentro das pasta, pois a partir dele o git tem o controle da pasta!!!



## Repositórios

Agora que já temos o nosso repositório configurado vamos criar o seu primeiro arquivo, na pasta Exemplo-Git. Crie um arquivo chamado “primeiro.txt”, você pode utilizar o bloco de notas para isso, escreva nele “Arquivo de teste” e salve o arquivo.

Agora no terminal digite ls, este comando lista do diretório.

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/Exemplo-Git (master)
$ ls
primeiro.txt
```



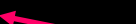
## Configurações básicas do Git

Outro ponto muito importante é que por algumas questões, o Git/Github então mudando o nome da ramificação principal do projeto, que antes era “master” e agora é utilizado o nome “main”. Para atualizar vamos utilizar o seguinte comando:

```
lcsi1@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/Exemplo-Git (master)
$ git config --global init.defaultBranch main
```

E para conferirmos se deu certo:

```
lcsi1@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/Exemplo-Git (master)
$ git config init.defaultBranch
main
```

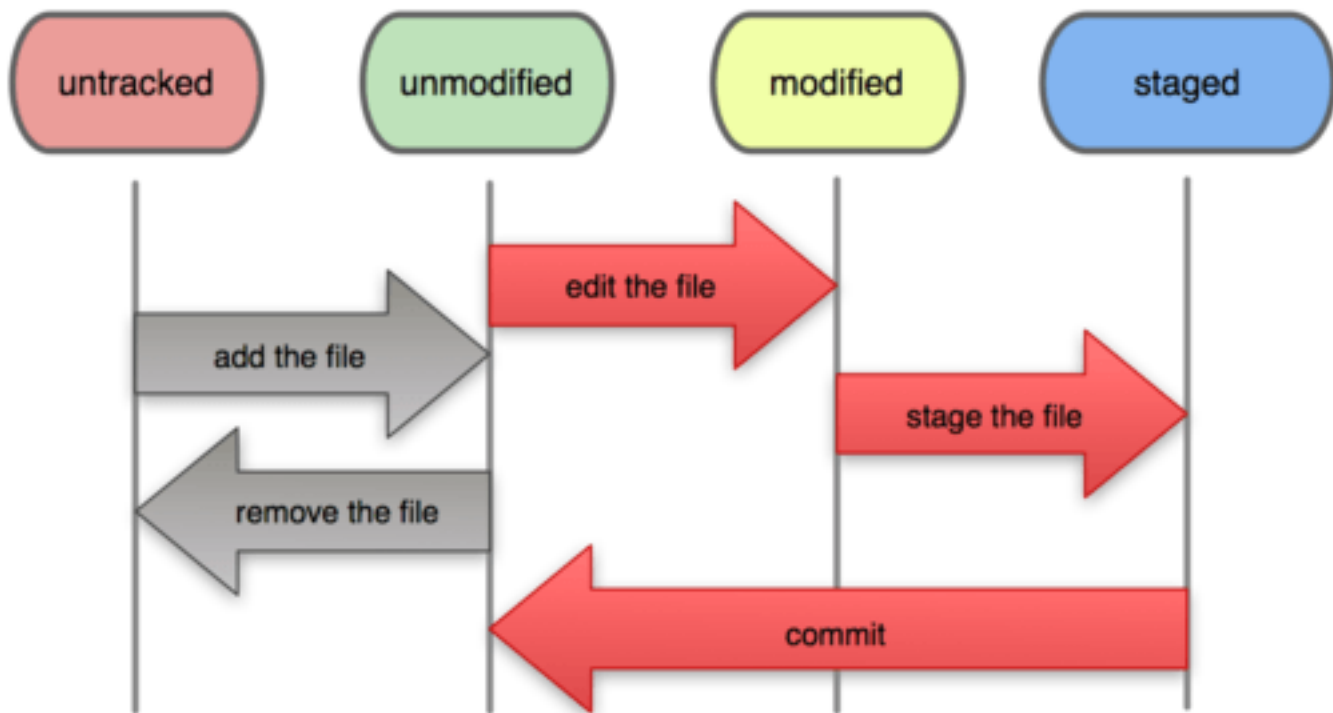
A pink arrow points from the right side towards the word "main" in the output of the command, indicating the new default branch name.





## Estados dos Arquivos

Para controlar o versionamento dos nossos arquivos o Git utiliza 4 Status diferentes em seu ciclo de vida:





## Estados dos Arquivos

Para sabermos qual o status dos arquivos use o comando: `git status`

```
lcsi1@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/exemplo-git (main)
$ git status
On branch main

No commits yet

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

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

Perceba que ele localizou o nosso arquivo e está avisando que ele ainda não está sendo rastreado pelo git “UNTRACKED”.



## Estados dos Arquivos

Agora vamos adicionar este arquivo ao controle de rastreamento do Git, digite:

**git add primeiro.txt**

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/exemplo-git (main)
$ git add primeiro.txt
```

Agora: **git status**

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/exemplo-git (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   primeiro.txt
```

Este é o estado de UNMODIFIED, ele está pronto para ser versionado.



## Estados dos Arquivos

Vamos fazer uma alteração no conteúdo do nosso arquivo, insira mais uma linha de texto, escreva por exemplo: Fiz uma alteração. Agora salve o arquivo e digite **git status** no terminal.

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/exemplo-git (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   primeiro.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:   primeiro.txt
```

Nosso arquivo continua sendo rastreado, mas agora ele diz que foi modificado “MODIFIED” e precisa ser adicionado novamente para ser versionado.



## Commit

Para fazer nosso primeiro commit (termo usado para versionamento) adicione novamente no modo STAGE usando o comando **git add primeiro.txt**.

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/exemplo-git (main)
$ git add primeiro.txt
```

Agora para realizarmos o commit use o comando: **git commit -m "Add primeiro.txt"**

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/exemplo-git (main)
$ git commit -m "Add primeiro.txt"
[main (root-commit) 0b6e532] Add primeiro.txt
1 file changed, 1 insertion(+)
create mode 100644 primeiro.txt
```

Vamos ver se funcionou? Digite **git status** novamente:

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/exemplo-git (main)
$ git status
On branch main
nothing to commit, working tree clean
```



## Commit

Usamos o comando **git log** para visualizar os commits feitos em nosso repositório.

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/exemplo-git (main)
$ git log
commit 0b6e532fac373ef58b667a292ea2cfe6e7b193c3 (HEAD -> main)
Author: lcsilva76 <lcsilva@fiap.com.br>
Date:   Sun Mar 10 20:48:42 2024 -0300

    Add primeiro.txt
```

Nele podemos visualizar todos os commits feitos e suas informações principais.

Utilizando a hash (código de identificação do commit), podemos ver as alterações feitas naquele commit, digite **git show 1a80a56f9c0d5c0b102fb8154330a548144170c7**

```
diff --git a/primeiro.txt b/primeiro.txt
new file mode 100644
index 0000000..20fbfbc
--- /dev/null
+++ b/primeiro.txt
@@ -0,0 +1 @@
+Primeira alteração.
\ No newline at end of file
```



## Commit

Muitas vezes podemos nos arrepender de alterações que fizemos em algum arquivo e querer se desfazer delas antes de fazer o commit. Para ver o que foi alterado em um arquivo desde o último commit pode usar o comando **git diff**

```
diff --git a/primeiro.txt b/primeiro.txt
index 20fbfbc..2dad78a 100644
--- a/primeiro.txt
+++ b/primeiro.txt
@@ -1,2 @@
-Primeira alteração.
\ No newline at end of file
+Primeira alteração.
+Segunda alteração.
\ No newline at end of file
```

Agora que temos certeza de que foi alterado, vamos desfazer usando o comando **git checkout primeiro.txt**

```
lcsil@LAPTOP-5HOML5IO MINGW64 ~/OneDrive/Área de Trabalho/exemplo-git (main)
$ git checkout primeiro.txt
Updated 1 path from the index
```

Se fecharmos e abrirmos novamente o nosso arquivo primeiro.txt você verá que as mudanças foram desfeitas



## Commit

Outra coisa que podemos querer desfazer é mandar o arquivo para o stage, para fazermos isso podemos usar o comando **git reset HEAD primeiro.txt**.

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/exemplo-git (main)
$ git reset HEAD primeiro.txt
Unstaged changes after reset:
M   primeiro.txt
```

Então agora, se usarmos o **git diff** novamente as alterações irão aparecer.

```
diff --git a/primeiro.txt b/primeiro.txt
index 20fbfbc..2dad78a 100644
--- a/primeiro.txt
+++ b/primeiro.txt
@@ -1,2 @@
-Primeira alteração.
\ No newline at end of file
+Primeira alteração.
+Segunda alteração.
\ No newline at end of file
```





## Commit

E se precisarmos voltar um commit?

Para esse problema temos 3 alternativas:

**Git reset --soft (hash)** = ele volta o commit, mas os arquivos continuam prontos para serem comitados novamente;

**Git reset --mixed (hash)** = ele volta o commit, mas os arquivos voltam para antes do staged;

**Git reset --hard (hash)** = ele volta o commit, também exclui todas as alterações;

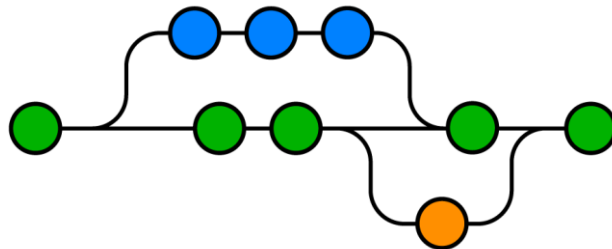
**Obs.** A hash deve ser a do commit para que teremos retornar



## O que é uma Branch?

É uma duplicação do projeto, gerando uma ramificação do projeto principal, permitindo ter um ou mais pessoas trabalhando nesses ramos. Assim no momento oportuno podem juntar suas contribuições ao projeto principal.

- Assim podemos:
- Trabalhar no projeto sem afetar o principal;
- Você pode apagar sem comprometer o projeto;
- Várias pessoas trabalhando
- Gerenciar os conflitos





## Como criar uma Branch?

Vamos para nosso projeto Exemplo-Git e criar nossa primeira branch:

- Criando a branch: **git checkout -b novaBranch**

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/Exemplo-Git (main)
$ git checkout -b novaBranch
Switched to a new branch 'novaBranch'
```

- Digite **git branch**, ele irá mostrar as branches que você possui. A que está com asterisco é a atual.

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/Exemplo-Git (novaBranch)
$ git branch
  main
* novaBranch
```



## Deletando ou mudando de Branch

Temos agora a **branch main** e uma ramificação a **novaBranch**, para navegarmos entre elas usamos o comando: **git checkout main** (repare que para navegar não usamos o **-b**)

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/Exemplo-Git (novaBranch)
$ git checkout main
Switched to branch 'main'
```

Para apagarmos uma branch usamos o comando: **git branch -D novaBranch** (você não pode estar na pasta dela no momento).

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/Exemplo-Git (main)
$ git branch -d novaBranch
Deleted branch novaBranch (was 5907b31).
```



## MERGE – Juntando branches

Com os projetos sendo trabalhados paralelamente, chega a hora de juntar trabalhos feitos no projeto, para isso usamos o “Merge”.

**1 -** Vamos criar uma nova branch chamada branch2: **git checkout -b branch2**

**2 -** Agora crie um arquivo chamado segundo.txt e escreva algo nele.

**3 -** Quando adicionarmos e comitarmos essa alteração nossa branch2 terá conteúdo diferente da master: **git add segundo.txt** e logo após **commit -am “Atualizacao da branch2”**.

**4 -** Vamos voltar para master: **git checkout master** e agora altere o arquivo segundo.txt e comite as alterações: **git commit -am “Atualizacao da master”**.



## MERGE – Juntando branches

- 5 – Ainda dentro da master digite o comando: **git merge branch2 -m "Junção das branches"**

```
lcsi1@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/Exemplo-Git (main)
$ git merge branch2 -m "Junção das branches"
Merge made by the 'ort' strategy.
segundo.txt | 1 +
1 file changed, 1 insertion(+)
create mode 100644 segundo.txt
```

- 6 – Agora use o comando: **git log --graph**  
(vai mostrar as ramificações)

```
$ git log --graph
* commit 8f00f5d095e9abbb92d567d33b9abc0ef6aa3e59 (HEAD -> main)
  Merge: 0c000fe 0ed1183
  Author: lcsi1va76 <lcsi1va@fiap.com.br>
  Date: Sun Mar 17 21:47:05 2024 -0300

    Junção das branches

* commit 0ed1183f2708e05591362a050fc0bbaca97ec295 (branch2)
  Author: lcsi1va76 <lcsi1va@fiap.com.br>
  Date: Sun Mar 17 21:42:48 2024 -0300

    Atualização branch2

* commit 0c000fe365f67eb1a7437a20bbd80897f12c763a
  Author: lcsi1va76 <lcsi1va@fiap.com.br>
  Date: Sun Mar 17 21:43:56 2024 -0300

    terceira alteração

* commit 5907b310f5b8c3e0ec0e644950b3056d6cd726509
  Author: lcsi1va76 <lcsi1va@fiap.com.br>
  Date: Sun Mar 17 21:18:31 2024 -0300

    Segundo commit

* commit 0b6e532fac373ef58b667a292ea2cfe6e7b193c3
  Author: lcsi1va76 <lcsi1va@fiap.com.br>
  Date: Sun Mar 10 20:48:42 2024 -0300

    Add primeiro.txt
```

# Github – Repositório Remoto

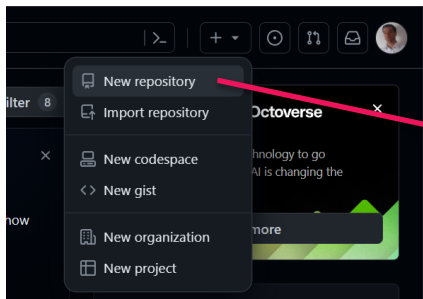




# Criando o Primeiro Repositório Remoto

Agora vamos criar um repositório remoto para guardar o nosso repositório local

Na barra de Menu do Github no canto superior direito, ao lado da sua foto, tem um símbolo de mais. Clique nele e escolha a opção **New Repository**.







# Criando o Primeiro Repositório Remoto

Agora temos que colocar as seguintes informações:

O nome do repositório remoto;


Uma descrição sobre o repositório, Algo breve, mas que identifique a razão dele;

Se ele será público ou privado, vamos trabalhar com repositórios públicos.

**Create a new repository**

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


Required fields are marked with an asterisk (\*).


Owner \*  lcsilva76

Repository name  exemplo-git is available.

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

Description (optional)

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

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

**Initialize this repository with:**

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

**Add .gitignore**

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

**Choose a license**

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

① You are creating a public repository in your personal account.

[Create repository](#)



# Criando o Primeiro Repositório Remoto

Agora que já temos o nosso repositório remoto, vamos subir nosso projeto para ele:

Esta é a opção para subirmos nosso projeto, copie a primeira linha e Execute em seu terminal.

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH `https://github.com/lcsilva76/exemplo-git.git`

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

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

```
echo "# exemplo-git" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/lcsilva76/exemplo-git.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/lcsilva76/exemplo-git.git
git branch -M main
git push -u origin main
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/Exemplo-Git (main)
$ git remote add origin https://github.com/lcsilva76/exemplo-git.git
```

Ainda no terminal digite **git remote** para verificar se já estão ligados. Deverá aparecer a resposta **origin**.



## Criando o Primeiro Repositório Remoto

Para subir nosso projeto digite a última no terminal:

**git push -u origin master**

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/Exemplo-Git (main)
$ git push -u origin main
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 8 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (14/14), 1.20 KiB | 246.00 KiB/s, done.
Total 14 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/lcsilva76/exemplo-git.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```



# Criando o Primeiro Repositório Remoto

Pronto, nosso projeto já está em um repositório remoto!!!

The screenshot shows the GitHub interface for a repository named 'exemplo-git'. The repository is public and has 1 branch (main) and 0 tags. The user 'lcsilva76' is the owner. The repository contains two files: 'primeiro.txt' and 'segundo.txt'. The 'primeiro.txt' file is highlighted with a red arrow. The repository also has a README section with the text 'Add a README' and a green button 'Add a README'. The right sidebar shows the repository's activity, including 0 stars, 1 watching, and 0 forks. The 'About' section mentions 'Exemplo da primeira aula de Github'.

exemplo-git Public

Pin Unwatch 1 Fork 0 Star 0

main 1 Branch 0 Tags

Go to file Add file Code

lcsilva76 Junção das branches 8f00f5d · 20 minutes ago 5 Commits

primeiro.txt terceira alteração 23 minutes ago

segundo.txt Atualização branch2 24 minutes ago

README

Add a README

Help people interested in this repository understand your project by adding a README.

Add a README

About

Exemplo da primeira aula de Github

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package



## Subindo Modificações no Repositório Remoto

Para esta parte, vamos criar um novo arquivo no nosso repositório chamado **terceiro.txt**, logo após digite algo dentro e salve.

Agora adicione todos para o stage, digitando **git add .** (o ponto significa todos)

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/Exemplo-Git (main)
$ git add .
```

Temos que fazer o commit para atualizar as informações do repositório: **git commit -m "Add terceiro.txt"**.

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/Exemplo-Git (main)
$ git commit -m "Adicionando terceiro.txt"
[main fb8ed34] Adicionando terceiro.txt
1 file changed, 1 insertion(+)
create mode 100644 terceiro.txt.txt
```



## Subindo Modificações no Repositório Remoto

Agora para subir use o comando: `git push origin main`

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/Exemplo-Git (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 338 bytes | 169.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/lcsilva76/exemplo-git.git
8f00f5d..fb8ed34  main -> main
```

Pronto, é só dar um refresh na página e ver o nosso arquivo `terceiro.txt` adicionado!

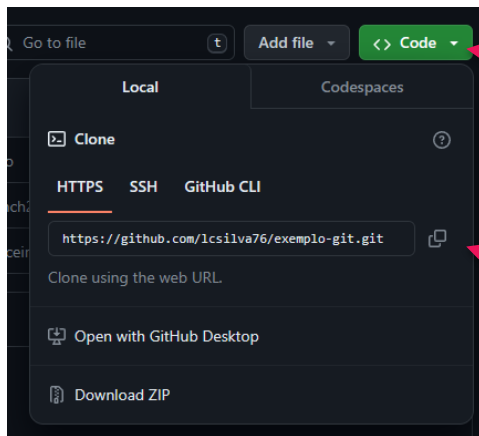
 <b>lcsilva76</b> Adicionando terceiro.txt		fb8ed34 · 2 minutes ago	 <b>6 Commits</b>
 primeiro.txt	terceira alteração	29 minutes ago	
 segundo.txt	Atualização branch2	30 minutes ago	
 terceiro.txt.txt	Adicionando terceiro.txt	2 minutes ago	



## Clonando um Repositório Remoto

É muito comum termos que acessar e trabalhar com nosso projeto em outras máquinas, para isso precisamos clonar (fazer uma cópia) do projeto em nossa máquina como se estivéssemos em outra, siga os passo:

**1** – No nosso repositório do github clique no botão **Code** e em seguida, **copie a URI** que está nele, é a URI do seu projeto.





## Clonando um Repositório Remoto

2 – Voltando para o nosso terminal, vamos sair da pasta atual e criar outra para nosso clone.

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/Exemplo-Git (main)
$ cd .. ← Sai da pasta atual
```

3 – Para fazer o clone na nova pasta digite: **git clone (URI copiada) Exemplo-Git-Clone**

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho
$ git clone https://github.com/lcsilva76/exemplo-git.git Exemplo-Git-Clone
Cloning into 'Exemplo-Git-Clone'...
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 17 (delta 2), reused 16 (delta 1), pack-reused 0
Receiving objects: 100% (17/17), done.
Resolving deltas: 100% (2/2), done.
```

4 – Agora abra a pasta com o comando **cd Exemplo-Git-Clone** e use o comando **ls** para conferir se os arquivos foram clonados.

```
lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho
$ cd exemplo-git-clone

lcsil@LAPTOP-5H0ML5IO MINGW64 ~/OneDrive/Área de Trabalho/exemplo-git-clone (main)
$ ls
primeiro.txt  segundo.txt  terceiro.txt.txt
```





## Praticando!!!

- Agora que conhecemos um pouco de Git e Github vamos praticar fazendo os seguintes exercícios:
  - 1 – Crie uma pasta no seu Desktop chamada **Exercicio-Git** e inicie ela no git;
  - 2 – Crie um arquivo chamado `exercicio1.txt`, escreva uma frase nele e salve.
  - 3 – Adicione o arquivo e faça o commit dele.
  - 4 – Crie uma branch chamada **branchExercicio**, crie um novo arquivo chamado `exercicio2.txt`, escreva uma frase, salve, adicione ele e faça o commit.
  - 5 – Volte para branch **master**, crie um arquivo chamado `exercicio3.txt`, escreva uma frase, salve, adicione ele e faça o commit.
  - 6 – Faça um merge da branch **branchExercicio** e visualize a junção com o **git log --graph**.
  - 7 – Crie um repositório remoto no github e faça um push da branch master do projeto.



FIAP

