



# Técnicas de Programação



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano



## Git

---

Git é considerado o arroz com feijão do desenvolvedor. Não importa qual sua especialidade, você vai precisar dele e é importante que saiba utilizar do jeito certo.





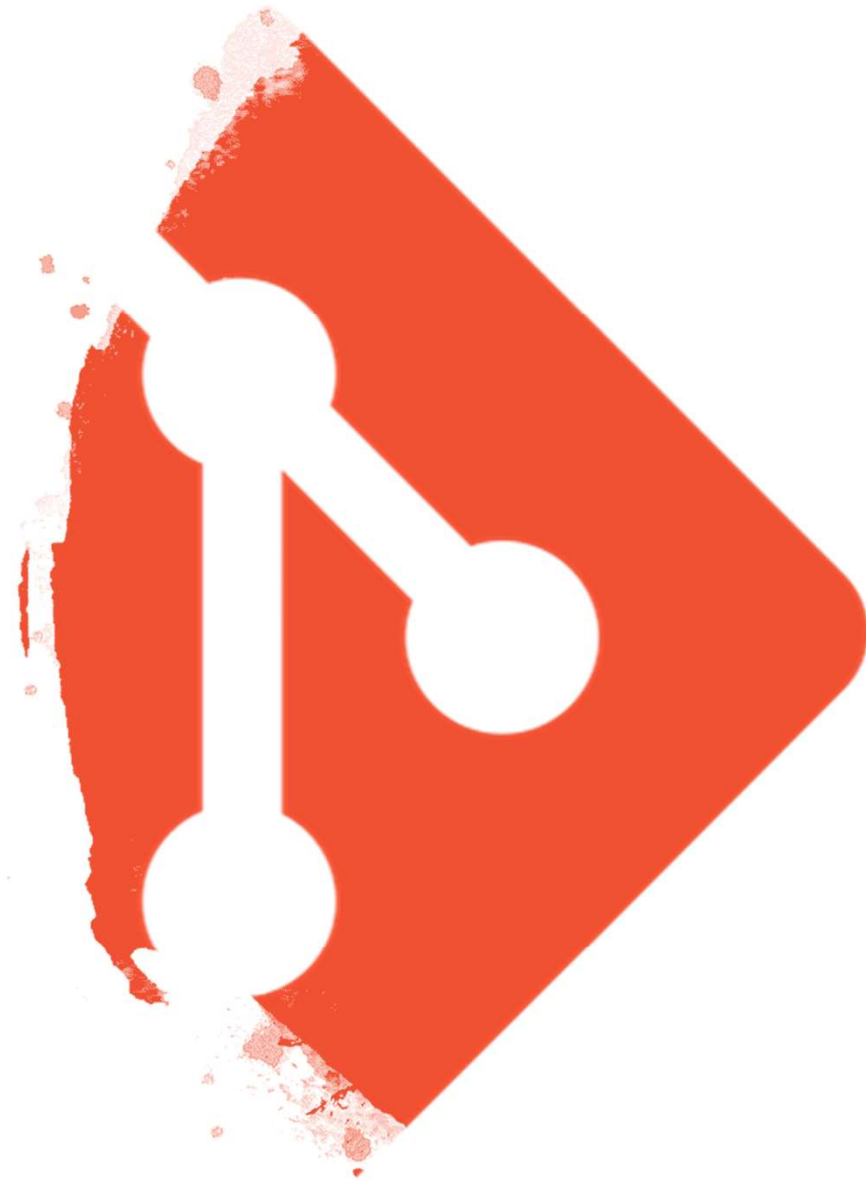
# Git

Pela documentação oficial, Git é um sistema de controle de versão distribuído de código aberto e gratuito, projetado para lidar com tudo, de projetos pequenos a grandes. O que isso significa? Significa que com o Git é possível manter um histórico das alterações dos seus arquivos, sabendo **quem**, **por que** e **quando** um arquivo foi editado.



# Principal funcionalidade

A principal funcionalidade do Git, que o faz ser amplamente utilizado em projetos de desenvolvimento de *software*, é a **possibilidade de fazer o controle de versões de modo colaborativo**, ou seja, é possível que o mesmo arquivo seja modificado ao mesmo tempo por dois desenvolvedores diferentes, e que ambas as alterações sejam salvas sem que nenhum código seja sobrescrito.



## Principal funcionalidade

Para permitir o modo colaborativo, o Git utiliza o conceito de ramificação ou *branch*, onde cada *branch* é uma linha do tempo que possui marcos ou *commits*, e nesse *branch* os arquivos podem ser alterados livremente sem impactar outras ramificações.



# Explicação do versionamento





## Github e Gitlab

Github e Gitlab são plataformas de hospedagem de código-fonte. Elas **permitem que os desenvolvedores contribuam em projetos privados ou abertos** (mais conhecidos como projetos *open source*).



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano





# Github e Gitlab

Nessas plataformas, cada projeto contendo um código-fonte é considerado um repositório. Por exemplo, se você participa de projeto em que é desenvolvido o site de um e-commerce, e o código do frontend é desenvolvido separadamente do código [backend](#), cada um desses códigos-fontes serão hospedados como repositórios separados.



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano





# GitLab

## Funcionalidade

Uma das principais funcionalidades que difere uma plataforma da outra, é o foco que o Gitlab vem dando à integração com ferramentas de [DevOps](#). O Gitlab proporciona, nativamente, ferramentas de integração e entrega contínua ou *CI/CD*, além de métricas para acompanhamento de qualidade de código, performance e teste de usabilidade.

E como tudo isso se relaciona com o Git? Ambas fazem o controle de versão dos projetos hospedados utilizando o Git. Desse modo, quando você utiliza o Git no seu projeto, você pode acompanhar o versionamento do seu repositório em uma dessas plataformas.

No geral, as duas plataformas são muito boas para hospedagem de código, e cabe a você decidir qual delas faz mais sentido utilizar no seu projeto, uma vez que as duas utilizam Git como controle de versão.



Sign in - GitLab

gitlab.com/users/sign\_in?redirect\_to\_referer=yes&\_cd\_chljshl\_k\_=19727b6cd900ce0abab1c6dad4464548ddd33474-160694877...

## GitLab.com

GitLab.com offers free unlimited (private) repositories and unlimited collaborators.

- [Explore projects on GitLab.com](#) (no login needed)
- [More information about GitLab.com](#)
- [GitLab Community Forum](#)
- [GitLab Homepage](#)

By signing up for and by signing in to this service you accept our:

- [Privacy policy](#)
- [GitLab.com Terms](#).

Username or email

Password

☐ Remember me [Forgot your password?](#)

Sign in

Don't have an account yet? [Register now](#)

Sign in with

☐ Remember me

GitLab

By registering for and using GitLab services, I agree to the [Terms of Service](#) and [Privacy Policy](#)

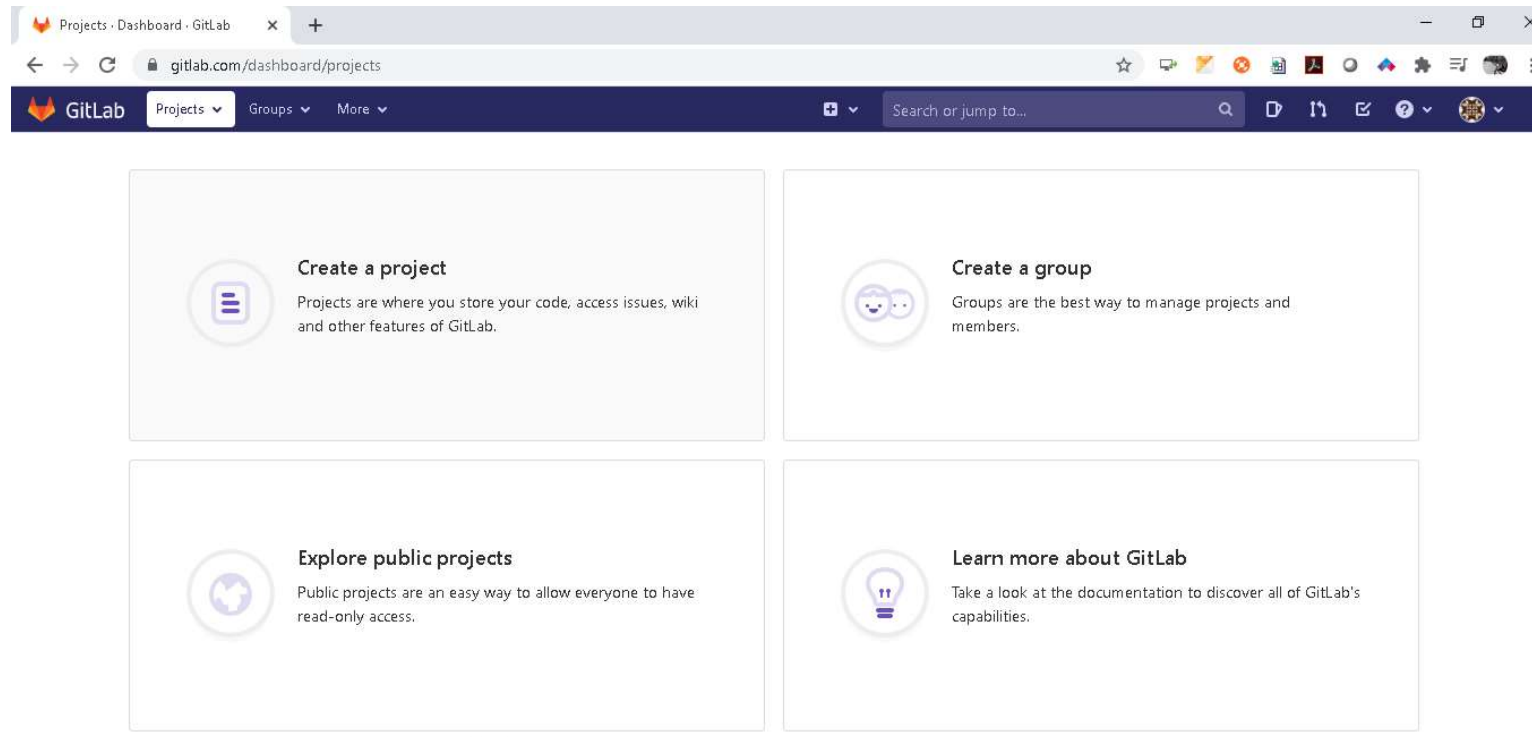
(If you do not agree with these terms you may contact [support@gitlab.com](mailto:support@gitlab.com) for assistance with your existing account.)

# Gitlab



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano

# Gitlab



# Gitlab

New Project · GitLab

gitlab.com/projects/new#blank\_project

GitLab Projects Groups More

Search or jump to...

+ Create blank project

Create blank project

Create a blank project to house your files, plan your work, and collaborate on code, among other things.

New project → Create blank project

Project name

AprendendoGitLab

Project URL

https://gitlab.com/alexandergobbato/

Project slug

aprendendogitlab

Want to house several dependent projects under the same namespace? [Create a group.](#)

Project description (optional)

Criando o projeto no gitlab para aprendizagem do aluno

Visibility Level ?

☒ Private  
Project access must be granted explicitly to each user. If this project is part of a group, access will be granted to members of the group.

☐ Public  
The project can be accessed without any authentication.

☒ Initialize repository with a README  
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

Create project

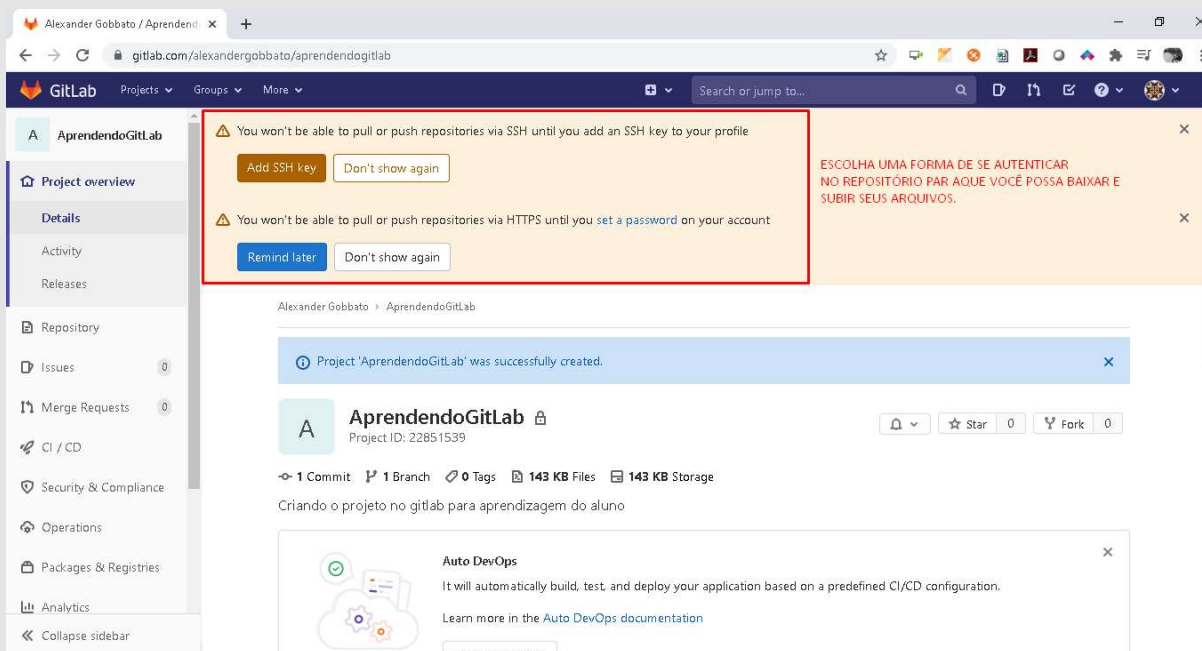
Cancel



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano

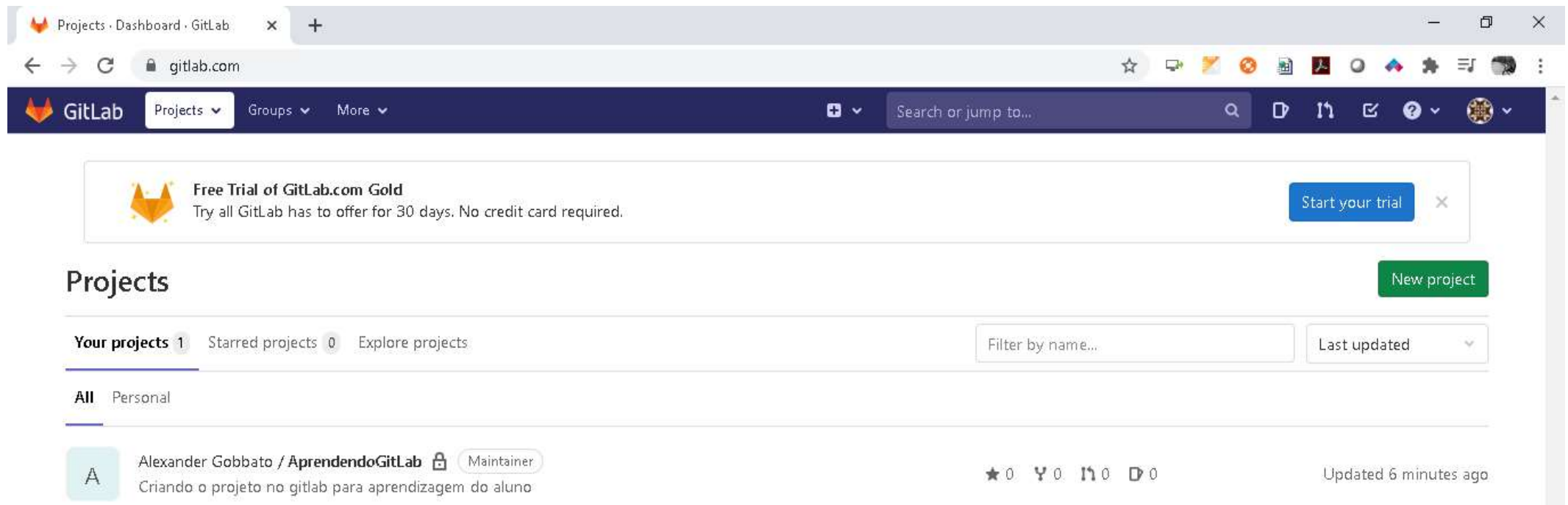
\*\*\*\*\* IMPORTANTE \*\*\*\*\*

Para deixar  
esse acesso  
e não ter  
que criar um  
arquivo SSH,  
optaremos  
pelos https.



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano

# Gitlab



The screenshot shows the GitLab web interface. At the top, there's a dark blue header with the GitLab logo, navigation links (Projects, Groups, More), a search bar, and user profile icons. Below the header is a promotional banner for 'Free Trial of GitLab.com Gold'. The main section is titled 'Projects' and includes a 'New project' button. Under 'Your projects', there's a list of projects. The first project is 'Alexander Gobbato / AprendendoGitLab', which is locked and has a 'Maintainer' role. It has 0 stars, 0 forks, 0 merges, and 0 deployments. The project description is 'Criando o projeto no gitlab para aprendizagem do aluno'. It was updated 6 minutes ago.

Projects · Dashboard · GitLab

gitlab.com

GitLab Projects Groups More

Search or jump to...

Free Trial of GitLab.com Gold  
Try all GitLab has to offer for 30 days. No credit card required.

Start your trial

## Projects

New project

Your projects 1 Starred projects 0 Explore projects

Filter by name... Last updated

All Personal

A Alexander Gobbato / **AprendendoGitLab** Maintainer

★ 0 🍴 0 🔀 0 📦 0

Criando o projeto no gitlab para aprendizagem do aluno

Updated 6 minutes ago



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano



GitHub: Where the world builds

github.com

Why GitHub? Team Enterprise Explore Marketplace Pricing

Search GitHub

Sign in Sign up

# Built for developers

GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 50 million developers.

**Username**

**Email**

**Password**

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

**Sign up for GitHub**

By clicking "Sign up for GitHub", you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account related emails.

GitHub



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano



Join GitHub · GitHub

github.com/join

Why GitHub? Team Enterprise Explore Marketplace Pricing

Search GitHub Sign in

Join GitHub

## Create your account

There were problems creating your account.

**Username \***

Username can't be blank

Email can't be blank

Password can't be blank. 8 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

**Email preferences**

☐ Send me occasional product updates, announcements, and offers.

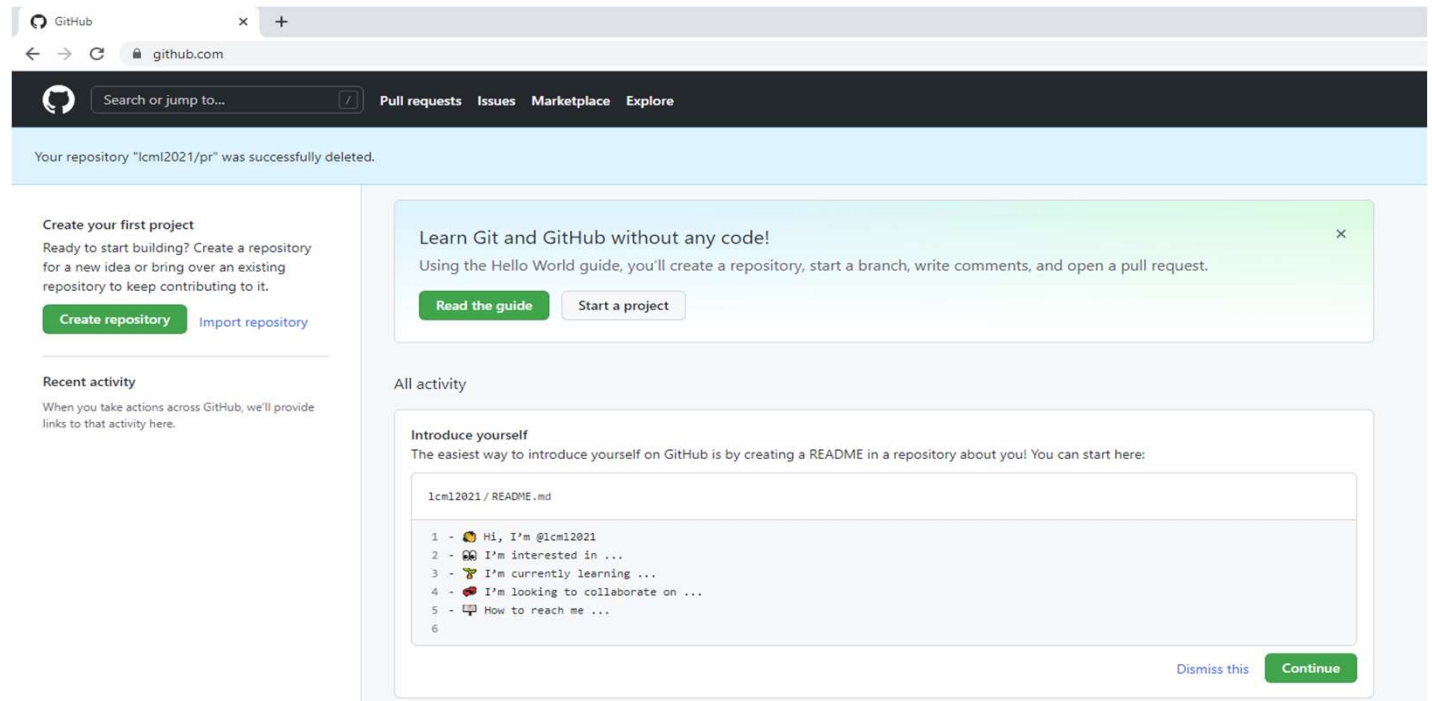
**Verify your account**

Github



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano

# Github



Clique em create repository



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano

## Create a new repository

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

Owner \* Repository name \*

 lcml2021 /  

Great repository names are short and memorable. Need inspiration? How about [cautious-octo-umbrella?](#)

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:

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

- ☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)
- ☐ **Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)
- ☐ **Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

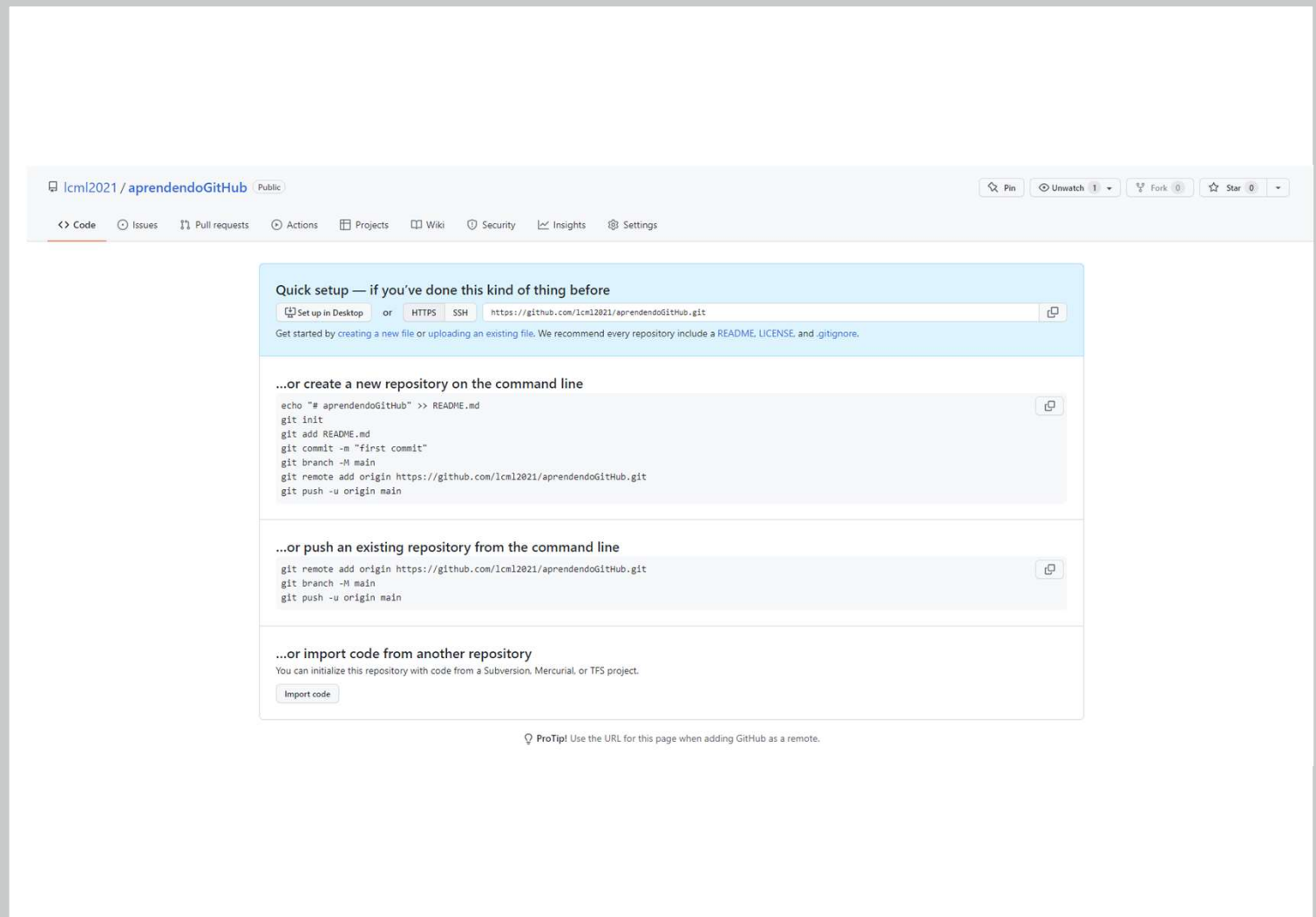
Create repository

# Github – Create Repository



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano

# Github



The screenshot shows the GitHub repository page for 'lcm12021/aprendendoGitHub'. The repository is public. The page displays the 'Quick setup' section with instructions for cloning the repository. The repository URL is 'https://github.com/lcm12021/aprendendoGitHub.git'. The instructions include a 'Quick setup' section with a 'Set up in Desktop' button, an 'or' separator, and 'HTTPS' and 'SSH' options. Below this, there are three sections: '...or create a new repository on the command line', '...or push an existing repository from the command line', and '...or import code from another repository'. The 'Import code' button is visible under the third section. A 'ProTip!' note at the bottom suggests using the URL for adding GitHub as a remote.

lcm12021 / **aprendendoGitHub** Public

Pin Unwatch Fork Star

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

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

Set up in Desktop or HTTPS SSH <https://github.com/lcm12021/aprendendoGitHub.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 "# aprendendoGitHub" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/lcm12021/aprendendoGitHub.git
git push -u origin main
```

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

```
git remote add origin https://github.com/lcm12021/aprendendoGitHub.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

**ProTip!** Use the URL for this page when adding GitHub as a remote.



Técnicas de Programação. - Prof. Msc. Luiz C M Lozano



# GIT – Instalação no Windows

Primeiro devemos testar se o computador a ser utilizado possui o GIT instalado. Para isso abra o **prompt** (CMD) no Windows e digite:

- `git --version`

Se o resultado for parecido com:

- `git version 2.30.1.windows.1`  
Nada mais precisa ser feito.

Caso algo parecido com o seguinte apareça:

- 'git' não é reconhecido como um comando interno ou externo, um programa operável ou um arquivo em lotes. Entre em <https://git-scm.com/downloads> e baixe a versão compatível com o seu Windows.





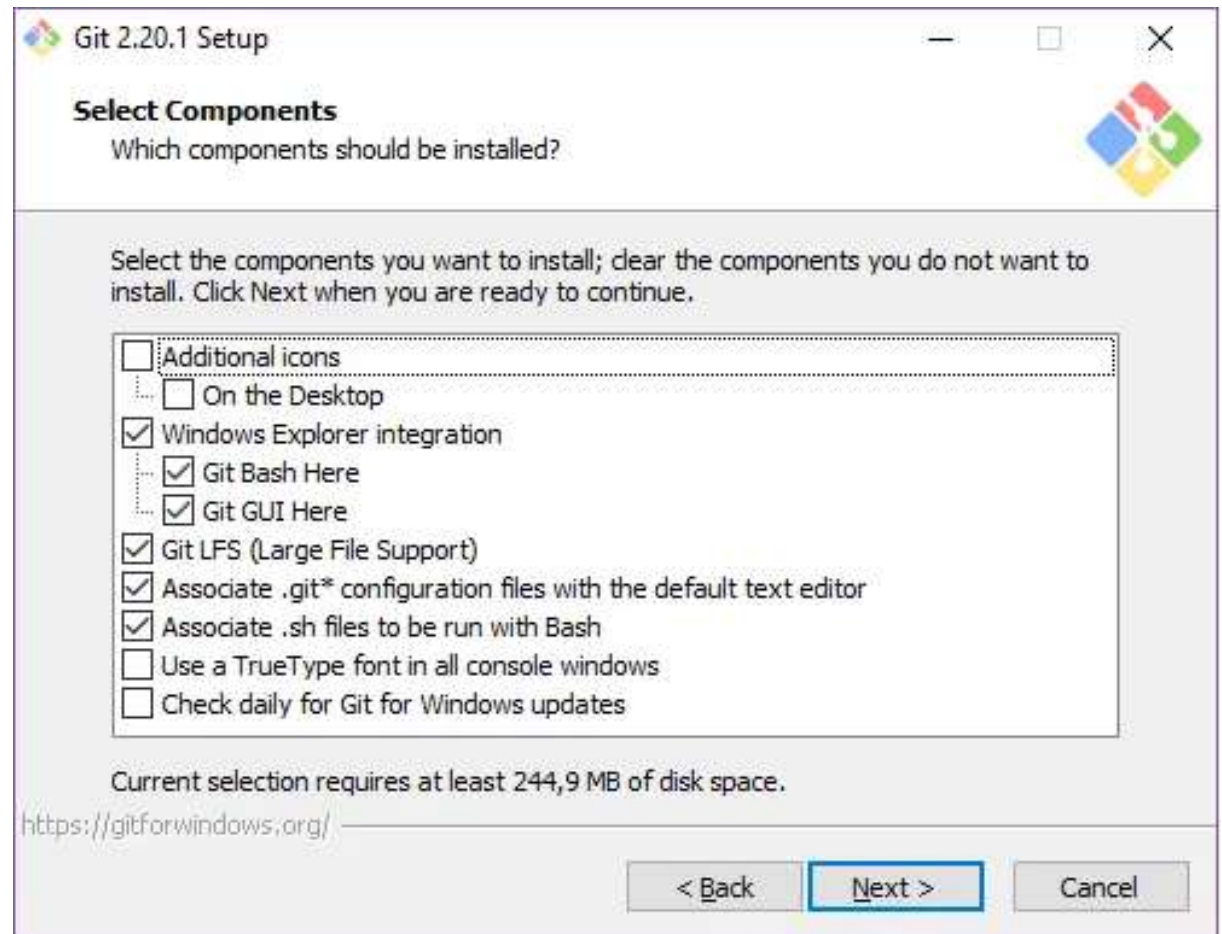
# GIT – Instalação no Windows

- Clique no botão **Next >** para concordar com os termos de licença.
- Na próxima tela salve o Git na pasta default C:\Programs Files\Git clicando no botão **Next>**
- Após escolher o diretório, você deve escolher os componentes que deseja instalar.



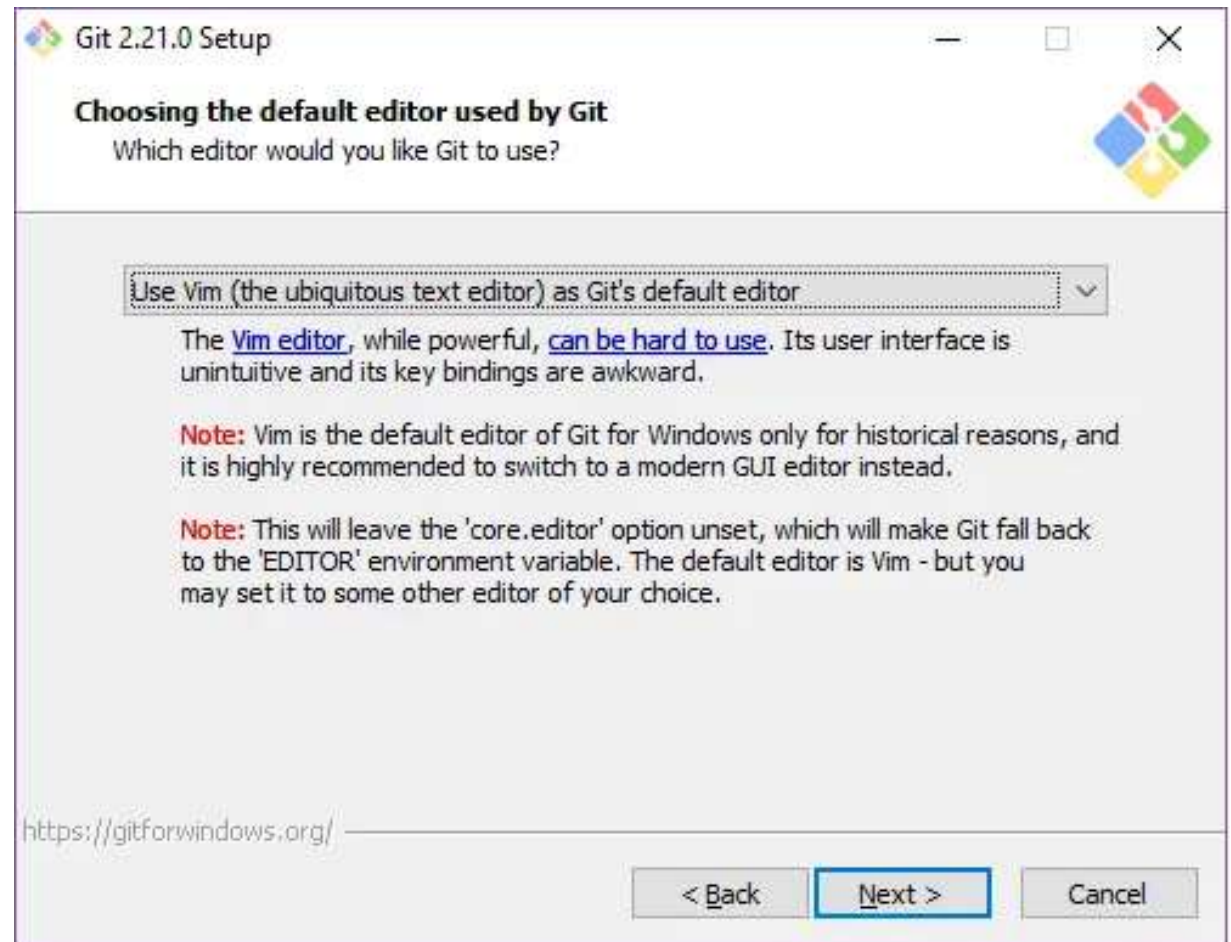
# GIT – Instalação no Windows

Na imagem é mostrado uma sugestão do que selecionar durante a instalação.



# GIT – Instalação no Windows

- Agora podemos escolher em qual editor default vamos utilizar o Git. Podemos escolher VSCode que utilizaremos durante nossas aulas, mas podemos utilizar outros.



# GIT – Instalação no Windows

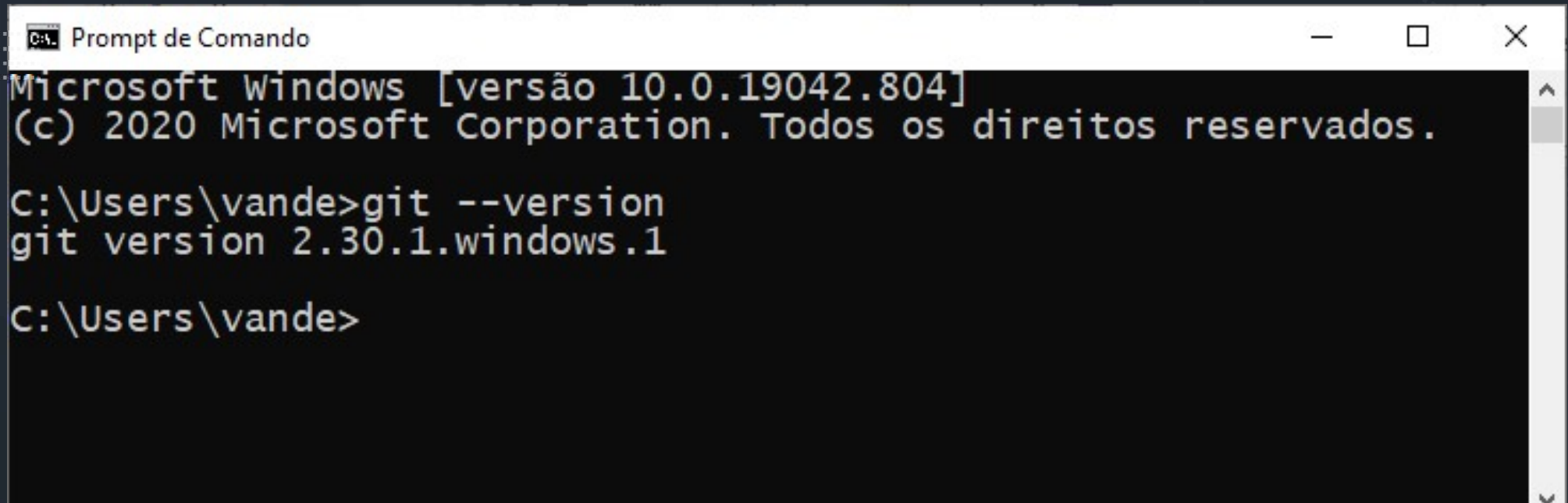
- Agora vamos escolher como utilizar o Git através da linha de comando.
- Sugiro escolher a segunda opção assim você poderá usar o Git dentro do VSCode, no GitBash no CMD e afins.



# GIT – Instalação Finalizando

- Para finalizar, basta aceitar todos os valores padrões do instalador até o fim. Quando acabar a instalação, teste novamente usando o CMD (não esqueça de abrir um novo).

`git --version`



```
Microsoft Windows [versão 10.0.19042.804]
(c) 2020 Microsoft Corporation. Todos os direitos reservados.

C:\Users\vande>git --version
git version 2.30.1.windows.1

C:\Users\vande>
```



# GIT – Instalação Finalizando

Pronto, agora você tem o Git instalado no seu computador. Se você não gosta de usar a linha de comando, existem vários clientes de Git para o Windows.

Agora precisamos da configuração básica. Você deve informar pelo menos seu e-mail e seu nome para o git.

- `git config --global user.email "seu-email@example.com"`
- `git config --global user.name "Seu Usuário"`

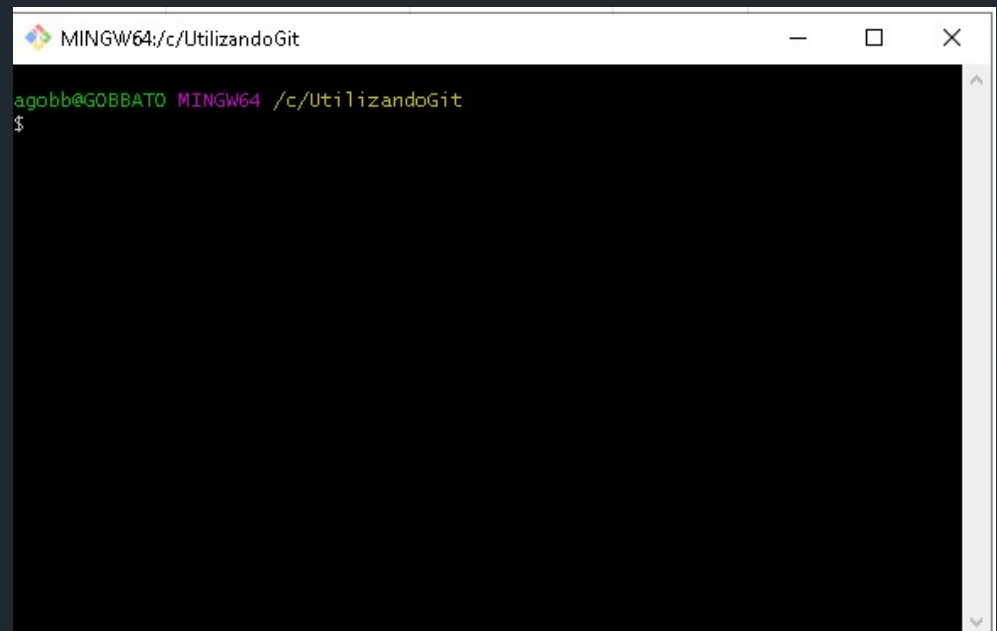
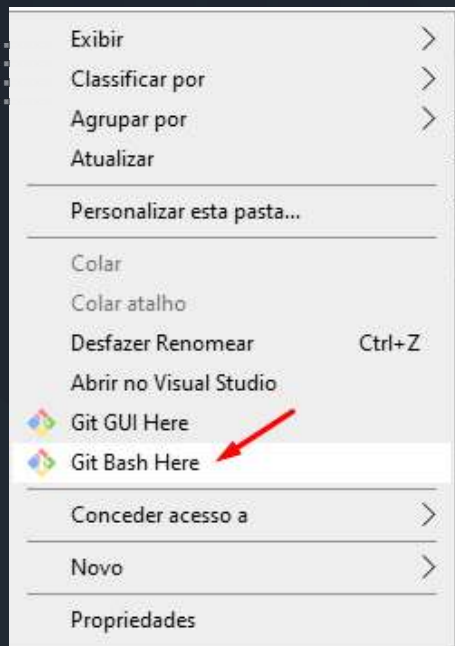
Podemos utilizar o Git integrado com o VSCode, mas existem outros softwares que possam interessar como: Github Desktop, SourceTree, TortoiseGIT.





# Utilizando comando do Git

- Selecionamos a pasta onde iremos clonar e com o botão direito inicializamos o git.
- Seguir os padrões do projeto publico dentro do GitHub



# Controle de versionamento de código com o Git



# Git

Como funciona o gerenciamento dos arquivos?

Todos os arquivos gerenciado pelo Git são armazenados como um blob numa estrutura de pastas localizado em “/.git/objects”.

```
PS C:\dev\example_git> tree .git
Listagem de caminhos de pasta
O número de série do volume é 0000022E 18CB:85F2
C:\DEV\EXAMPLE_GIT\.GIT
|_ hooks
|_ info
|_ logs
|   |_ refs
|       |_ heads
|_ objects
|   |_ 4b
|   |_ 61
|   |_ 78
|   |_ a6
|   |_ f4
|   |_ info
|   |_ pack
|_ refs
|   |_ heads
|   |_ tags
```



# Git

Como funciona o gerenciamento dos arquivos?

Os arquivos são organizados em commits que possui a uma estrutura com uma mensagem e uma tree (conjunto de chave e valor)

Essas estruturas (commit e tree) são armazenados em formato de blob no mesmo diretório dos arquivos.

```
01609280520@DPB0AVBP153818 MINGW64 /c/dev/example_git (master)
$ git commit -m 'oi'
[master (root-commit) a64b067] oi
2 files changed, 2 insertions(+)
create mode 100644 a.txt
create mode 100644 b.txt
```

```
PS C:\dev\example_git> tree .git
Listagem de caminhos de pasta
O número de série do volume é 0000022E 18CB:85F2
C:\DEV\EXAMPLE_GIT\.GIT
├── hooks
├── info
├── logs
│   └── refs
│       └── heads
├── objects
│   ├── 4b
│   ├── 61
│   ├── 78
│   └── a6 ←
```



# Tags



# Tags

## O que são?

Tags são marcações na árvore de commits que apontam para um commit específico. Criando uma espécie de checkpoint.





# Tags

## Quando devem ser criadas?

Uma Tag deve ser criada sempre que se desejar adicionar um ponto de recuperação de um código estável em uma determinada branch.

No nosso contexto, uma tag deve ser criada todas as vezes que um novo pacote for publicado em produção e apenas na branch master dos projetos.



# Tags

## Processo de criação de um tag via linha de comando

A partir da branch que se deseja criar a tag, no nosso contexto seria a branch master após um deploy em produção, digite o comando a seguir:

- `git tag <nome_da_tag> -m '<mensagem_descritiva>'`

Esse comando criará um tag no seu repositório local.

\* Na mensagem descritiva, deve incluir a data do deploy em produção



# Tags

## Processo de criação de um tag via linha de comando

Para enviar essa tag para o repositório remoto, digite o comando a seguir:

- `git push origin <nome_da_tag>`

Esse comando empurrará a tag do seu repositório local para o remoto.



# Tags

## Padronização na nomenclatura das tags

Com a finalidade de facilitar a visualização e compreensão das tags, decidimos adotar o seguinte padrão para todas as tags.

- V.<MAJOR>.<MINIOR>.<PATH>

Que seria basicamente a versão que essa tag está marcando, e como citado anteriormente, toda tag deve conter um mensagem com um resumo dessa nova versão.



# Branch



# Branch

## O que é?

Branchs são muitos similares a tags, o que diferencia um do outro é que uma tag ele sempre vai apontar para um commit, independente a evolução do software(commits posteriores), enquanto a branch sempre aponta para o último commit feito naquela “tag”, criando uma linha alternativa à branch da qual ela foi derivada.



# Branch

## O ciclo de vida de uma branch

- Branchs são criadas para um finalidade especifica e deve-se excluí-la sempre que a razão pela qual ela foi criada for concluída.
- No nosso contexto, sempre que uma funcionalidade (razão pela qual a branch é criada) for publicada em produção (com a criação de uma tag), os dados dessa branch deve ser mesclada com a branch master e a branch da funcionalidade deve ser excluída.





# Branch

**Qual deve ser a origem de uma branch?**

- De qualquer branch que se faz necessário criar uma bifurcação para adição de novas funcionalidades ou correção de bugs.
- No nosso contexto, em 99% dos casos, deve ser originada da branch master.





# Branch

## Processo de criação de uma branch via linha de commando

- A partir da branch que se deseja criar um bifurcação, digite os comandos a seguir:
- `git branch <nome_da_branch>`
- Após esse comando digite a linha abaixo para selecionar a branch como o local de trabalho.
- `git checkout <nome_da_branch>`



# Branch

## Processo de criação de uma branch via linha de comando

- Ou podemos utilizar esse atalho do git para executar os dois passos do slide anterior de uma única vez.
- A partir da branch que se deseja criar um bifurcação, digite o comando a seguir:
- `git checkout -b <nome_da_branch>`
- Por trás do atalho ele executa os mesmo dois passos anteriormente citados.



# Branch

## Processo de criação de uma branch via linha de comando

- Para enviar a branch para o repositório remoto.
- Da nova branch criada, digite o comando a seguir:
- `git push -u origin <nome_da_branch_remota>`



# Branch

## Padronização na nomenclatura das branches

A partir de hoje teremos os seguintes padrões para o nome das branches criadas, além da branch master que reflete o código que está executando em produção.



# Branch

## Padronização na nomenclatura das branches

- Teremos uma branch chamada **release\_<num\_serv>**.
- A finalidade dessa branch é disponibilizar as funcionalidades e correções de bug num ambiente para testes, sejam eles de desenvolvimento ou homologação.
- Essa branch poderá ser apagada, sempre que for necessário, portanto não façam commits nela.



# Branch

## Padronização na nomenclatura das branches

- Para todas as novas funcionalidades deve-se criar uma branch com o seguinte padrão **feature\_<nome\_func>**.
- Como citado anteriormente, essa branch deve ser retirada da **master** em 99% dos casos e assim que essa funcionalidade for publicada em produção, o conteúdo dela deve ser 'mergeado' com a master e, por fim, deve ser excluída.



# Branch

## Padronização na nomenclatura das branches

- Sempre que tivermos erros críticos em produção no qual necessitamos publicar uma nova versão o mais rápido possível deve-se criar uma branch com o seguinte padrão **hotfix\_<desc\_func\_bug>**.
- Como citado anteriormente, essa branch deve sempre ser retirada da **master** e assim que essa funcionalidade for publicada em produção, o conteúdo dela deve ser 'mergeado' com a master e, por fim, deve ser excluída.



# Branch

## Padronização na nomenclatura das branches

- Sempre que tivermos erros em produção no qual não necessitamos publicar uma nova versão o mais rápido possível deve-se criar uma branch com o seguinte padrão **bugfix\_<desc\_func\_bug>**.
- Como citado anteriormente, essa branch deve sempre ser retirada da **master** e assim que essa funcionalidade for publicada em produção, o conteúdo dela deve ser 'mergeado' com a master e, por fim, deve ser excluída.





# Porque não utilizar o GitHub Desktop?

- É mais fácil de utilizar por possuir aquela interface gráfica, porém, existem ambientes que não se encaixam por não possuírem interface gráfica. Utilizamos o GIT BASH exatamente pelo fato dele ser por linha de comando. Todos os computadores conseguimos utilizar linha de comando mas não são todos que podemos usar interface gráfica. Com GIT Bash atingimos a maior parte do nosso publico

# Vamos agora colocar as mãos na massa!!!



## Créditos:

Material elaborado pelos  
Professores:

Alexander Gobbato P.  
Albuquerque

Vanderson Bossi

e adaptado pelo Prof. Luiz  
Lozano

