

# Versionamento com GIT

Rodrigo Attique

# O que é versionamento?

- GIT é um sistema de controle de versão, também conhecido pela sigla VCS – version control system.
- Os VCS são uma ótima forma de otimizar o trabalho ao analisar as alterações feitas em um código de projeto compartilhado, além de ser um componente fundamental do sistema de gerenciamento de configuração de software que cuida das mudanças que precisam ser feitas em um projeto.
- As alterações/revisões/ atualizações feitas são identificáveis através de códigos de letra ou números. Informações como data e a identidade do autor da alteração também são mantidas.

# Como surgiu?

- Em 2005, Linus Torvalds (o homem conhecido por criar o núcleo, ou kernel, do SO Linux) desenvolveu o GIT, que desde então tem sido ativamente mantido por Junio Hamano, um engenheiro de software japonês.
- Atualmente, o GIT é um dos mais famosos sistemas de controle de versão de código aberto e milhões de projetos no mundo inteiro o utilizam para seu controle de versão (incluindo projetos comerciais e de código aberto).

# Como surgiu?

O Git é totalmente grátis e pode ser instalado em Mac, Linux, Windows e Solaris diretamente do site oficial. Algumas das características essenciais do GIT são:

1- Um sistema de controle de versão distribuído, o GIT segue uma abordagem peer to peer, contrário de outros como o Subversion (SVN) que segue um modelo baseado em cliente-servidor.

# Como surgiu?

2- GIT permite aos desenvolvedores ter uma infinidade de ramos de código completamente independente. Criação, exclusão e fusão desses ramos é simples e não leva tempo.

3- No GIT, todas as operações são atômicas. Isso significa que uma ação pode ter sucesso ou falhar (sem fazer nenhuma alteração). Isso é importante porque em alguns sistemas de controle de versão (como o CVS) onde as operações não são atômicas, se uma operação de repositório é suspensa, ela pode deixar o repositório em um estado instável.

3- No GIT, tudo é armazenado dentro da pasta `.git`. Isso não é o mesmo em outros VCS como SVN e CVS onde os metadados de arquivos são armazenados em pastas ocultas (por exemplo, `.cvs`, `.svn`, etc.)

# Como surgiu?

5- GIT usa um modelo de dados que ajuda a garantir a integridade criptográfica de qualquer coisa presente dentro de um repositório. Cada vez que um arquivo é adicionado ou um commit é feito, suas somas de verificação são geradas. Da mesma forma, eles são recuperados através de suas somas de verificação também.

6- Outra característica presente no GIT é sua área de teste ou índice. Na área de preparação, os desenvolvedores podem formatar commits e receber feedback antes de aplicá-los.

# Como surgiu?

5- GIT usa um modelo de dados que ajuda a garantir a integridade criptográfica de qualquer coisa presente dentro de um repositório. Cada vez que um arquivo é adicionado ou um commit é feito, suas somas de verificação são geradas. Da mesma forma, eles são recuperados através de suas somas de verificação também.

6- Outra característica presente no GIT é sua área de teste ou índice. Na área de preparação, os desenvolvedores podem formatar commits e receber feedback antes de aplicá-los.

# Versatilidade

- O GIT é consideravelmente simples de usar. Para começar, você pode criar um repositório ou conferir um já existente. Após a instalação, um simples git-init irá deixar tudo pronto. Da mesma maneira, o comando git clone pode criar uma cópia de um repositório local para um usuário.





# 1º Passo – Instalar o GIT

# Instalar o GIT no Windows:

Instalar o GIT no Windows é tão simples como baixar um instalador e executá-lo. Execute os seguintes passos para instalar o GIT no Windows:

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

### Information

Please read the following important information before continuing.



When you are ready to continue with Setup, click Next.

## GNU General Public License

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.  
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies  
of this license document, but changing it is not allowed.

### Preamble

The licenses for most software are designed to take away your  
freedom to share and change it. By contrast, the GNU General Public  
License is intended to guarantee your freedom to share and change

<https://git-for-windows.github.io/>

Next >

Cancel

# Configurando...

- Abra o [prompt de comando](#) e digite os seguintes comandos no terminal:

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



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

GIT instalado no Windows com sucesso!

# Instalar o GIT no Linux:

- Se você usa Linux em qualquer sistema (ou num servidor VPS), então você deve estar acostumado com instalar programas e pacotes em seu computador usando comandos de instalação apt-get ou yum. Instalar o GIT não é diferente:

Para usuários Debian/Ubuntu (apt-get):

1. Abra o terminal e execute os seguintes comandos:

```
sudo apt-get update  
sudo apt-get install git
```

2. Verifique se a instalação ocorreu com sucesso usando `git --version`.

# Instalar o GIT no Linux:

3. Execute os seguintes comandos no terminal para configurar seu e-mail e nome de usuário que serão associados à sua conta GIT:

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



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

# 2º Passo – Como Usar o GIT

Agora que o GIT está instalado e configurado no seu dispositivo, vamos explorar os conceitos básicos do GIT e como começar a usar o GIT.

# Ferramentas de Versionamento



# Criar/configurar/verificar um repositório

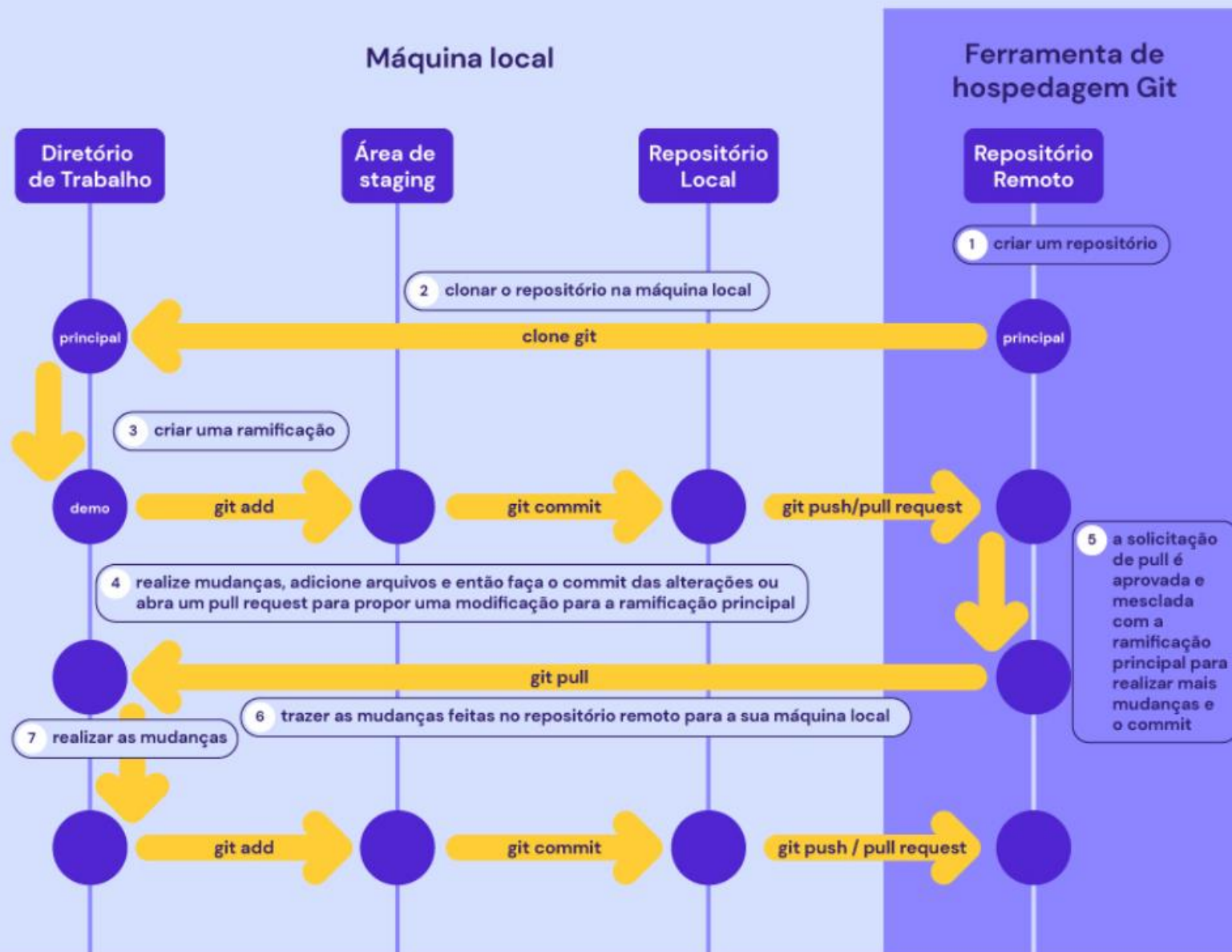
- Um repositório é o maior bem de qualquer projeto controlado por versão. Para transformar qualquer diretório em um repositório GIT, o simples comando `git init <directory>` pode ser utilizado. Uma pasta chamada `.git` também deve começar a existir no diretório em que o comando foi executado.
- Por outro lado, se você já tem um diretório e deseja verificar (clone-lo), você pode usar o comando `git clone`. Se você estiver tentando verificar um repositório local, use o seguinte comando:

# Criar/configurar/verificar um repositório

```
git clone /path/to/local/repository
```

Se você pretende verificar um repositório armazenado remotamente, use:

```
git clone user.name@host:/path/to/remote/repository
```



# Fluxo de Trabalho

- Agora que o repositório está pronto, vamos falar sobre a estrutura que é mantida pelo GIT.
- Cada repositório local consiste em três árvores: o diretório de trabalho que contém os arquivos reais; O índice que desempenha o papel de uma área de teste e o HEAD que é um ponteiro para o último commit feito pelo usuário.
- Então, é assim que o fluxo de trabalho pode ser explicado: o usuário adiciona um arquivo ou alterações do diretório de trabalho para o índice (a área de teste) e uma vez revistos, o arquivo ou as alterações são finalmente comprometidos com o HEAD.

# Os comandos Add e Commit:

Alterações ou adições de arquivos propostas são adicionadas ao índice usando o comando *add*. Para adicionar qualquer arquivo, o comando é:

```
git add <nome_do_arquivo>
```

Se você está realmente confiante o suficiente para fazer essas mudanças no **HEAD**, então você pode usar o comando *commit*.

```
git commit -m "Adicionar qualquer mensagem sobre o commit aqui"
```



**Importante!** Nota: Uma vez que o comando *commit* é executado (a partir do diretório de trabalho), o arquivo fica comprometido com o **HEAD**, mas ainda não é enviado para o repositório remoto.

# Dando Continuidade às Mudanças

- Depois de confirmar as alterações (e acreditar que elas estão prontas para serem enviadas para o repositório original), você pode usar o comando push.
- Uma vez que o `git push origin master` é executado de dentro do diretório de trabalho, as mudanças presentes no HEAD são enviadas para o repositório remoto. No comando acima mencionado, o master pode ser alterado para o nome do branch ao qual você deseja que as alterações sejam comprometidas.
- Se, no entanto, um repositório existente ainda não tiver sido clonado e você pretende estabelecer uma ligação entre o seu repositório e um servidor remoto, execute o seguinte comando:

# Dando Continuidade às Mudanças

```
git remote add origin <servidor>
```



**Importante!** Nota: Substitua pelo endereço do servidor remoto.

Uma vez clonado, quaisquer alterações feitas serão aplicadas para o servidor pertinente.

# Branches

- Outra característica brilhante (mas avançada) do GIT é sua capacidade de permitir que desenvolvedores e gerentes de projeto criem vários ramos (branches) independentes dentro de um único projeto.
- O objetivo principal de um branch é desenvolver novas funcionalidades, mantendo-os isolados uns dos outros. O branch padrão em qualquer projeto é sempre o master branch. Tantos branches quanto necessários podem ser criados e eventualmente mesclados ao master branch.
- Um novo branch pode ser criado usando o seguinte comando:



# Branches

```
git checkout -b feature_n *
```

`feature_n` é o nome do *branch*.

Se você deseja retornar ao *master* branch, o seguinte comando pode ser usado:

```
git checkout master
```

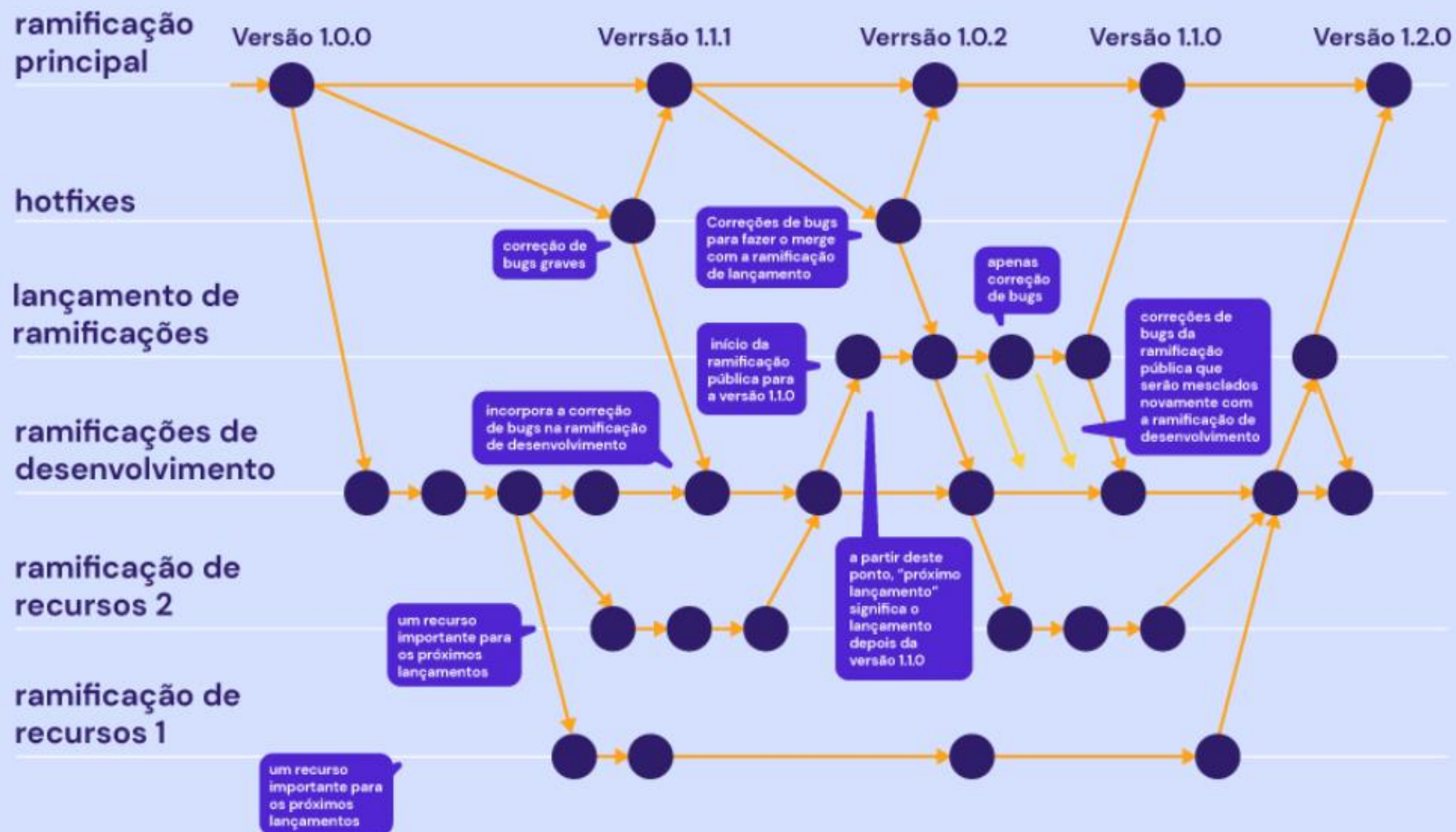
Qualquer *branch* pode ser excluído usando o seguinte comando:

```
git checkout -b feature_n
```

# Branches

Para tornar o *branch* disponível para outros usuários, você terá que movê-lo para o repositório remoto. Para fazer isso, use o seguinte comando:

```
git push origin feature_n
```



# Atualizando e Dando Merge

- Caso você queira atualizar seu diretório de trabalho local para o mais recente do repositório remoto, o simples comando `git pull` pode ser usado.
- Para mesclar outro branch (dar um merge) no atualmente ativo, use: `git merge feature_n`.
- Se você der um merge ou pull, o GIT sempre tenta lidar com os conflitos por conta própria, mas as vezes não consegue. Em caso de falha devido a conflitos, o usuário tem que resolver os conflitos manualmente. Depois de editar os arquivos (para erradicar conflitos), marque-os como merged usando:

# Atualizando e Dando Merge

```
git add <nome.arquivo>
```

Se antes do *merge* você desejar visualizar as alterações, o seguinte comando pode ser executado:

```
git diff <nome_branch_origem> <nome_branch_alvo>
```

# Tagging

Antes de lançar atualizações/alterações de software, é sempre recomendado criar tags. Para fazer isso no GIT, use o seguinte comando:

```
git tag 1.1.0 1c2d2d56fa
```

O **1c2d2d56fa** no comando acima refere-se aos primeiros 10 caracteres do **commit-id** que é referenciado com a tag. O ID de *commit* pode ser encontrado no log.

# Log

O histórico do repositório pode ser estudado através do log. O comando `git log` recupera as informações. Para recuperar os *commits* feitos por um único usuário, você pode usar:

```
git log --author =Smith
```

Uma versão compactada do log (um *commit* por linha) pode ser visualizada usando:

```
git log --pretty=oneline
```

Para exibir somente os arquivos que foram alterados:

```
git log --name-status
```

# Substituindo Alterações Locais

- Se você acabou fazendo bagunça e precisa reverter as alterações feitas em qualquer arquivo, faça isso usando o seguinte comando:

\$ git checkout -- <nomedoarquivo>

- Isso substituirá as alterações da árvore de trabalho pelos últimos dados presentes no HEAD. Quaisquer alterações que já tenham sido adicionadas ao índice não serão prejudicadas.



# Substituindo Alterações Locais

- Isso substituirá as alterações da árvore de trabalho pelos últimos dados presentes no HEAD. Quaisquer alterações que já tenham sido adicionadas ao índice não serão prejudicadas.
- Por outro lado, se todas as alterações/commits locais devem ser eliminados e o master branch local for necessário para apontar para o histórico mais recente do servidor, execute os seguintes comandos:

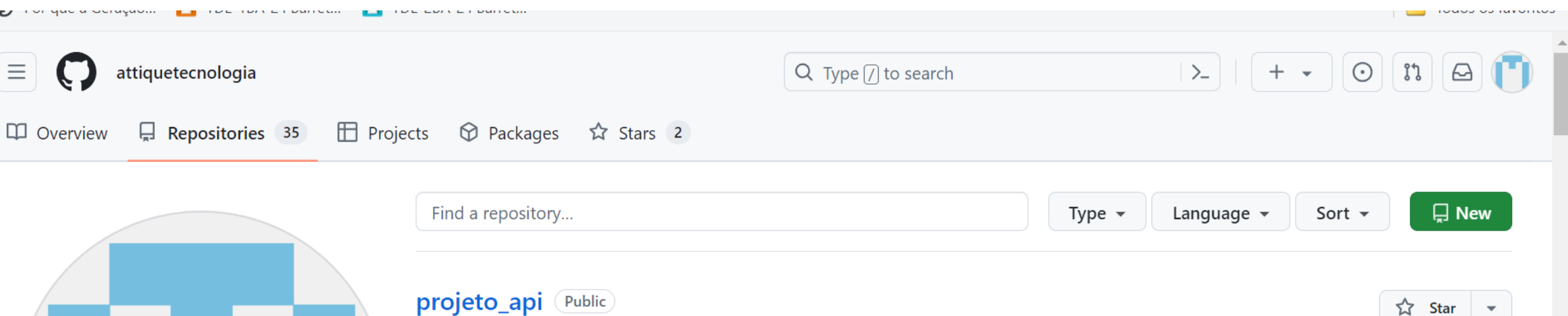
```
$ git fetch origin
```

```
$ git reset --hard origin/master
```

# GITHUB.com

# Criando um repositório no github.com

- Na pagina repositórios clique no botão New (botão verde).



# Atribua um nome para o repositório


## 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 \*

 attiquetecnologia ▾

Repository name \*

/ algoritmos1ba24

✔ algoritmos1ba24 is available.



Great repository names are short and memorable. Need inspiration? How about **vigilant-goggles** ?

Description (optional)

Todos os códigos da aula de lógica de programação que é muito legal!

 Public

# Marque as opções abaixo

- 
- ☒  **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

.gitignore template: None ▼

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

Choose a license

License: None ▼

# Por fim clique no botão verde Create repository

---

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

## Choose a license

License: None ▼

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

This will set  main as the default branch. Change the default name in your [settings](#).

---

 You are creating a public repository in your personal account.

---



Create repository

# Se tudo estiver ok, você verá esta tela

The screenshot shows a GitHub repository interface. At the top, there's a navigation bar with 'main' branch selected, '1 Branch', and '0 Tags'. A search bar 'Go to file' and buttons 'Add file' and 'Code' are also present. Below this, a commit history section shows an 'Initial commit' by 'attiquetecnologia' 1 minute ago. A file list shows 'README.md' as the only file. The main content area displays the 'README' file with the title 'algoritmos1ba24' and the description 'Todos os códigos da aula de lógica de programação que é muito legal!'. On the right sidebar, the 'About' section repeats the description, and there are links for 'Readme', 'Activity', '0 stars', '1 watching', and '0 forks'. Below that, the 'Releases' section states 'No releases published' with a link to 'Create a new release'. The 'Packages' section also states 'No packages published' with a link to 'Publish your first package'.

main 1 Branch 0 Tags

Go to file Add file Code

attiquetecnologia Initial commit 716958a · 1 minute ago 1 Commits

README.md Initial commit 1 minute ago

README

## algoritmos1ba24

Todos os códigos da aula de lógica de programação que é muito legal!

About

Todos os códigos da aula de lógica de programação que é muito legal!

Readme Activity 0 stars 1 watching 0 forks

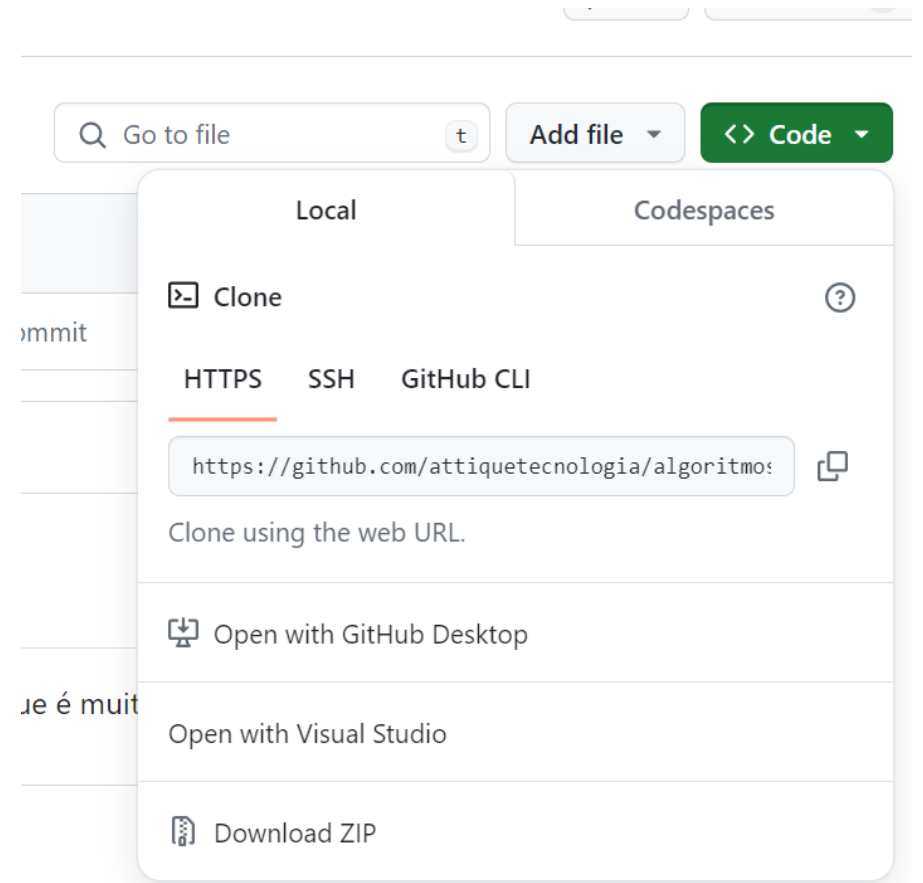
Releases

No releases published  
[Create a new release](#)

Packages

No packages published  
[Publish your first package](#)

# Clique no botão Code e depois copie o endereço do repositório





# Clonando Repositórios

---

# Git Clone

- Clonar é o termo usado para “sincronizar” um repositório remoto com uma máquina local, [conforme imagem](#).
- Um repositório remoto pode ser clonado em diversas máquinas em diferentes lugares com o comando git clone.
- \$ git clone https://.... nome\_do\_repositório
- O endereço do repositório pode ser obtido por exemplo no site do github conforme [mostrado aqui](#).