

Ciclo do Software e Versionamento

Professor: Henrique Delegrego

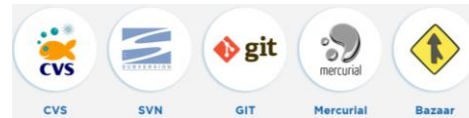


Ciclo do Software e Versionamento



- É uma ferramenta de versionamento

Versionamento

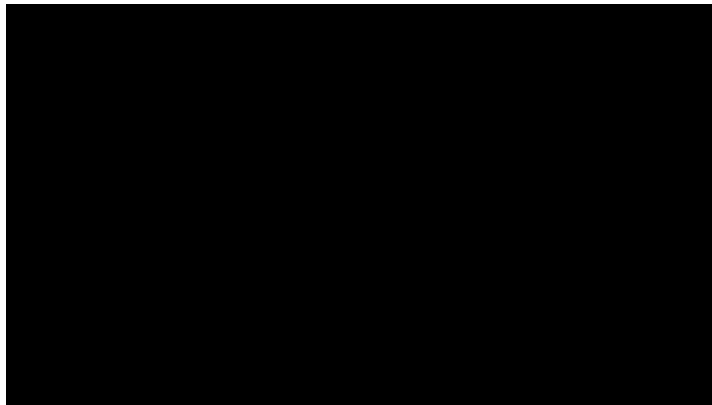


- Responsável por gerenciar alterações em programas
- À medida que as equipes desenvolvem, é comum que várias versões do mesmo software sejam implementadas em locais diferentes
- Permite que várias pessoas trabalhem em um mesmo projeto

Ciclo do Software e Versionamento



- É a ferramenta de versionamento mais usada no mundo
- Tem concorrentes como o GitLab, Atlassian, BitBucket e vários outros
- **Agora:** Vamos criar uma conta no GitHub, personalizar o nosso perfil e fazer o push de um arquivo



Ciclo do Software e Versionamento

Palavras chaves de versionamento

- Repositório
- README
- Staging Area
- Commit
- Push / Pull
- Branch
- Pull Request / Merge Request (PR / MR)
- Merge
- Clone
- Fork



Ciclo do Software e Versionamento

Repositório (Repo)

- Local onde são armazenados arquivos e pastas relacionados a um projeto específico
- Armazena o histórico completo do projeto, incluindo **commits**, **branches** e **tags**
- Permite reverter para versões anteriores



Ciclo do Software e Versionamento

README

- Arquivo de texto que fornece informações sobre o repositório
- Serve como ponto de entrada principal para usuários e colaboradores, oferecendo uma visão geral do projeto, além de instruções sobre como utilizá-lo, contribuir ou configurá-lo

README.md

Bootstrap Ruby Gem build passing gem v4.1.1

Bootstrap 4 ruby gem for Ruby on Rails (Sprockets) and Hanami (formerly Lotus).

For Sass versions of Bootstrap 3 and 2 see [bootstrap-sass](#) instead.

Installation

Please see the appropriate guide for your environment of choice:

- [Ruby on Rails 4+](#) or other Sprockets environment.
- [Other Ruby frameworks](#) not on Rails.

a. Ruby on Rails

Add `bootstrap` to your Gemfile:

```
gem 'bootstrap', '~> 4.1.1'
```

Ensure that `sprockets-rails` is at least v2.3.2.

`bundle install` and restart your server to make the files available through the pipeline.

Ciclo do Software e Versionamento

Staging Area

- Funciona como um espaço intermediário entre o diretório de trabalho e o repositório onde você escolhe as alterações que deseja incluir no próximo commit
- Permite selecionar com precisão o que será incluído no próximo commit
- Facilita a organização das mudanças, especialmente em projetos grandes ou com muitas alterações simultâneas

Ciclo do Software e Versionamento

Commit

- Salva as mudanças feitas no projeto, seja adições, edições, exclusões ou movimentações de arquivos
- Cada commit é acompanhado por uma mensagem de commit, essa mensagem fornece uma descrição concisa do que foi alterado ou adicionado no commit
- A convenção é usar:
- **Inglês**
 - Imperativo
 - Ex: “fix bug”
- **Português**
 - Presente do indicativo
 - Ex: “conserta bug”

Ciclo do Software e Versionamento

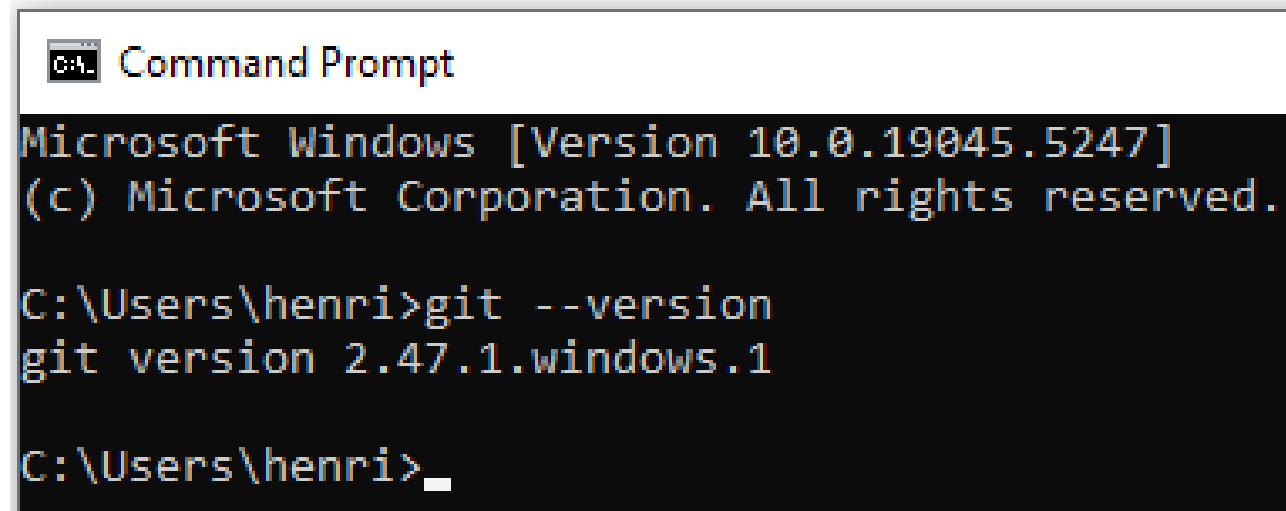
Push / Pull

- **Push**
 - É a ação de fazer upload da sua versão local para um repositório, disponibilizando suas alterações para outros
- **Pull**
 - Processo de fazer download das alterações de um repositório para a sua versão
 - Usado para buscar e integrar mudanças do repositório remoto para o seu branch local

Ciclo do Software e Versionamento

Instalando Git

- Para checar se você tem o Git instalado na sua máquina, abra o cmd e digite: **git --version**



```
C:\> Command Prompt
Microsoft Windows [Version 10.0.19045.5247]
(c) Microsoft Corporation. All rights reserved.

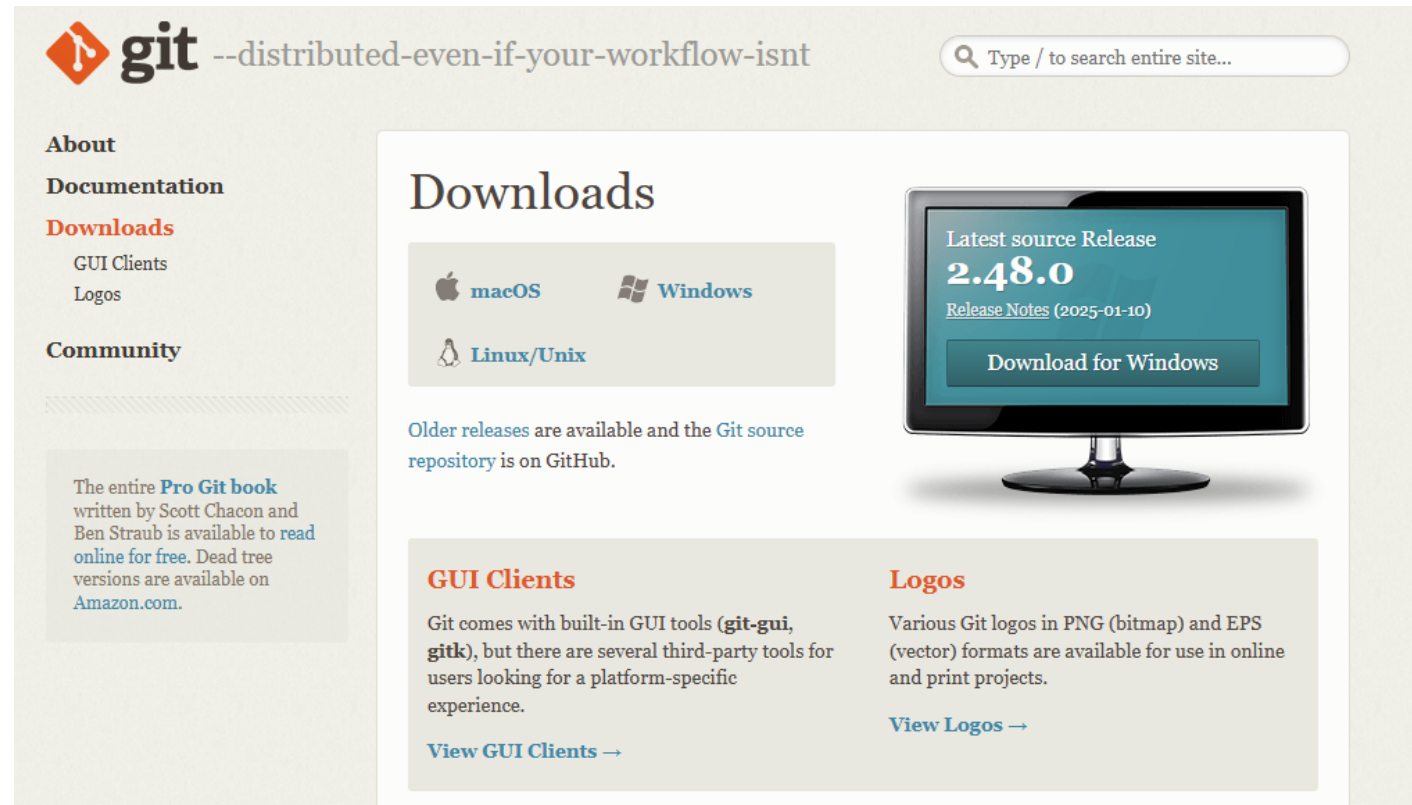
C:\Users\henri>git --version
git version 2.47.1.windows.1

C:\Users\henri>_
```

Ciclo do Software e Versionamento

Instalando Git

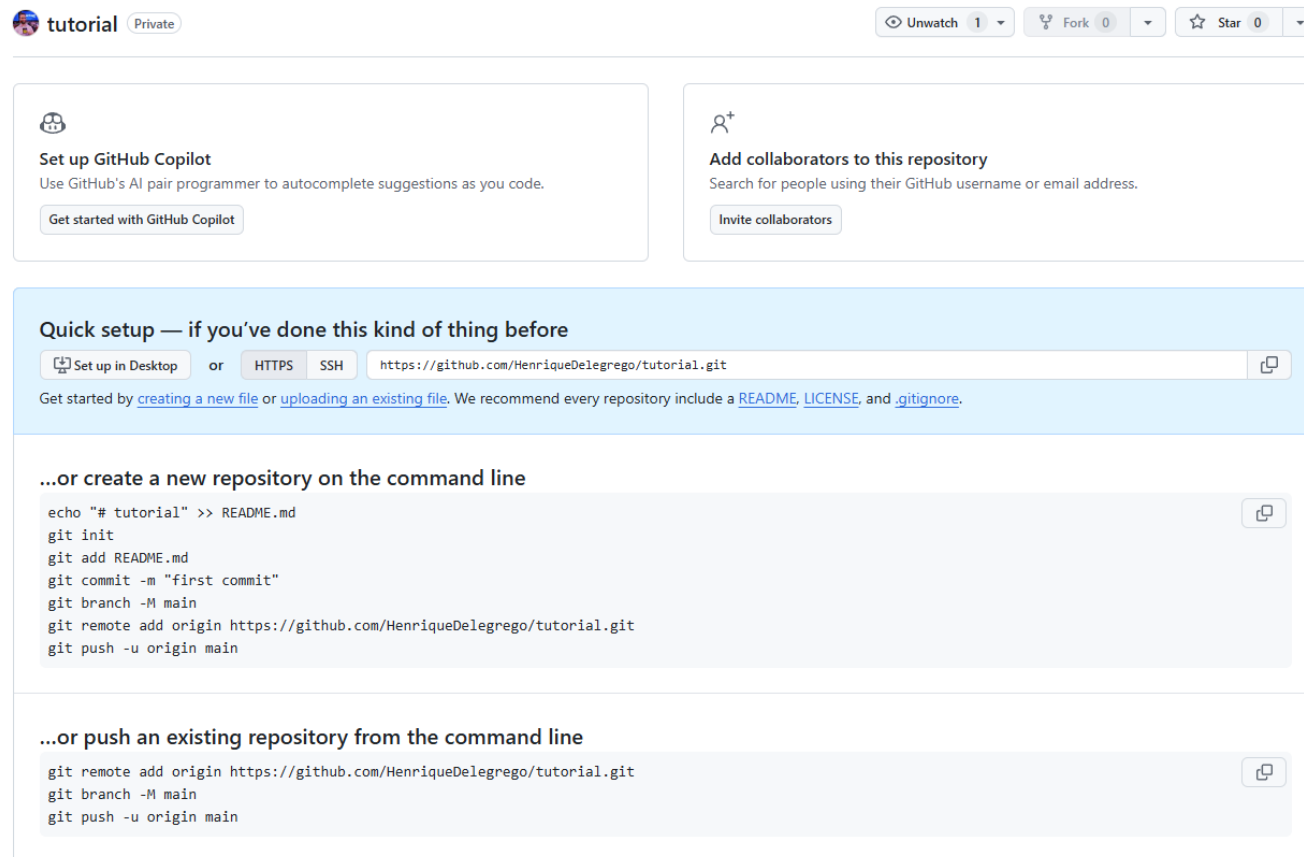
- Caso você não tenha instalado, vá no site do Git para baixar



Ciclo do Software e Versionamento

Iniciando um repositório

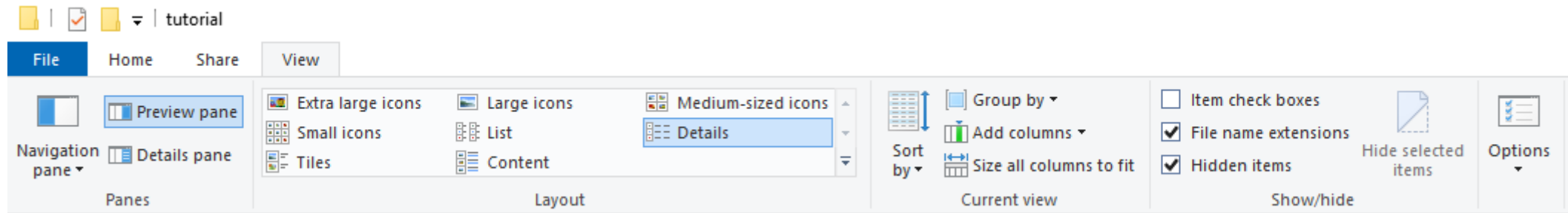
- Assim que um repositório no GitHub é criado, essa tela aparece



Ciclo do Software e Versionamento

Iniciando um repositório

- Habilite a visualização da extensão dos arquivos e pastas ocultas



Ciclo do Software e Versionamento

Iniciando um repositório

- Crie uma pasta no seu computador
- Crie os arquivos de programação
- Abra algum terminal no caminho do diretório
- Execute os comandos do Git

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

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

Ciclo do Software e Versionamento

Iniciando um repositório

- **echo "# tutorial" >> README.md** Escreve no arquivo README.md
- **git init** Inicializa um repositório na pasta
- **git add .** Adiciona todos os arquivos na Staging Area
- **git commit -m "first commit"** Faz o commit dos arquivos com uma mensagem
- **git branch -M main** Renomeia a branch principal para main (nomenclatura moderna)
- **git remote add...** Conecta o repositório local ao seu repositório no GitHub
- **git push -u origin main** Faz o push do commit para o GitHub

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

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


Ciclo do Software e Versionamento

Push e Pull no terminal

- Para fazer um push em um repositório do GitHub
 - `git add .`
 - `git commit -m "mensagem de commit"`
 - `git push -u origin main`
- Para fazer um pull de um repositório do GitHub
 - `git pull`

```
Henrique@Henrique-PC MINGW64 ~/Desktop/Git/tutorial (main)
$ git add .

Henrique@Henrique-PC MINGW64 ~/Desktop/Git/tutorial (main)
$ git commit -m "adiciona index.html"
[main 1716a2e] adiciona index.html
1 file changed, 11 insertions(+)
create mode 100644 index.html

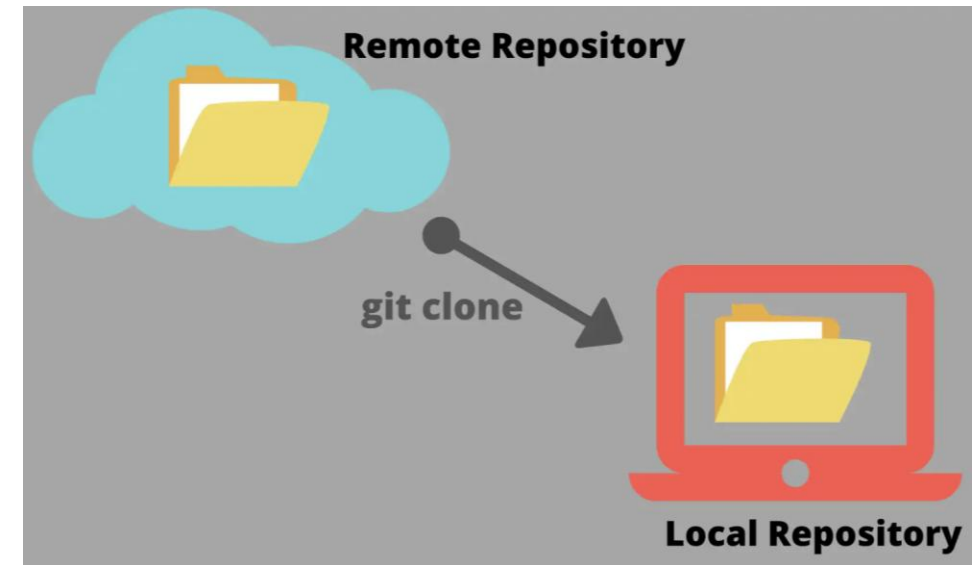
Henrique@Henrique-PC MINGW64 ~/Desktop/Git/tutorial (main)
$ git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 502 bytes | 502.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/HenriqueDelegregio/tutorial.git
 4c643a0..1716a2e  main -> main
branch 'main' set up to track 'origin/main'.
```

```
Henrique@Henrique-PC MINGW64 ~/Desktop/Git/tutorial (main)
$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 1003 bytes | 250.00 KiB/s, done.
From https://github.com/HenriqueDelegregio/tutorial
 1716a2e..3a0d6a3  main      -> origin/main
Updating 1716a2e..3a0d6a3
Fast-forward
 arquivo-novo.txt | 1 +
1 file changed, 1 insertion(+)
create mode 100644 arquivo-novo.txt
```

Ciclo do Software e Versionamento

Clone

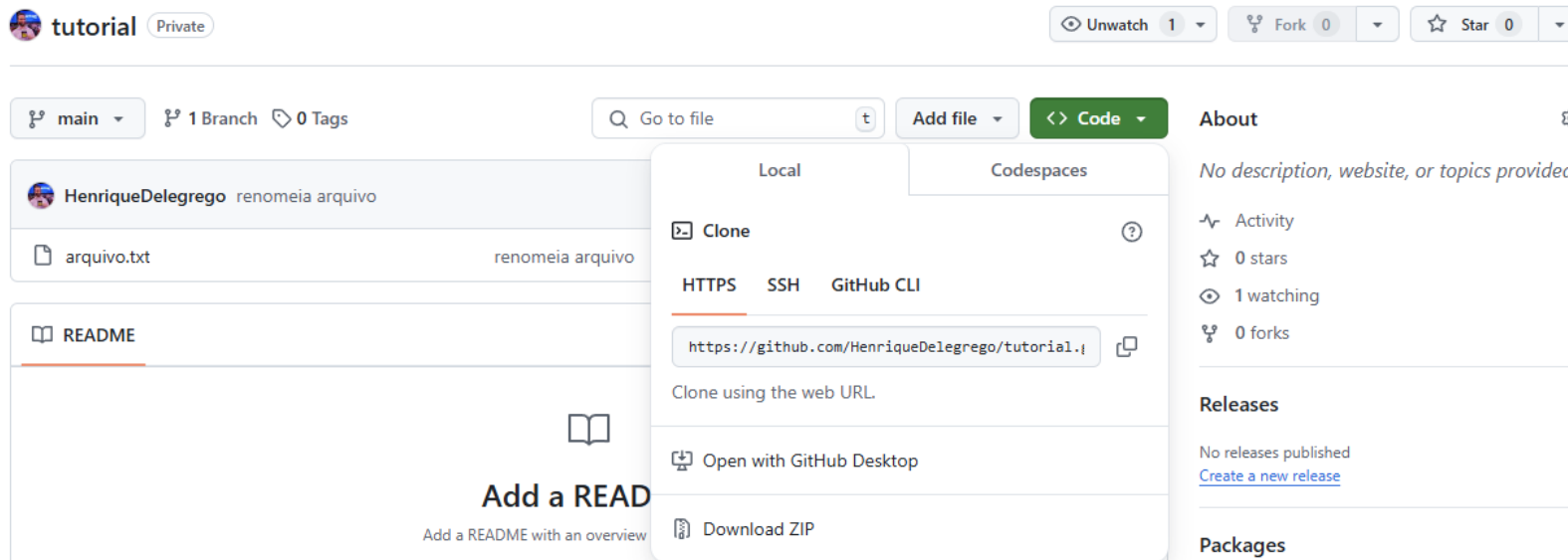
- Cria uma cópia local de um repositório existente
- Isso permite que você trabalhe e faça alterações no projeto usando
- A cópia local é totalmente funcional, o que significa que você pode editar arquivos e fazer commits



Ciclo do Software e Versionamento

Clonando um repositório

- Para clonar um repositório do GitHub em um repositório local
 - Vá no repositório do GitHub e pegue o link dele
 - Digite no terminal dentro da pasta do repositório: `git clone <link>`
 - Ex: `git clone https://github.com/HenriqueDelegrego/tutorial.git`
 - Isso vai fazer com que o repositório do GitHub se conecte ao seu repositório local



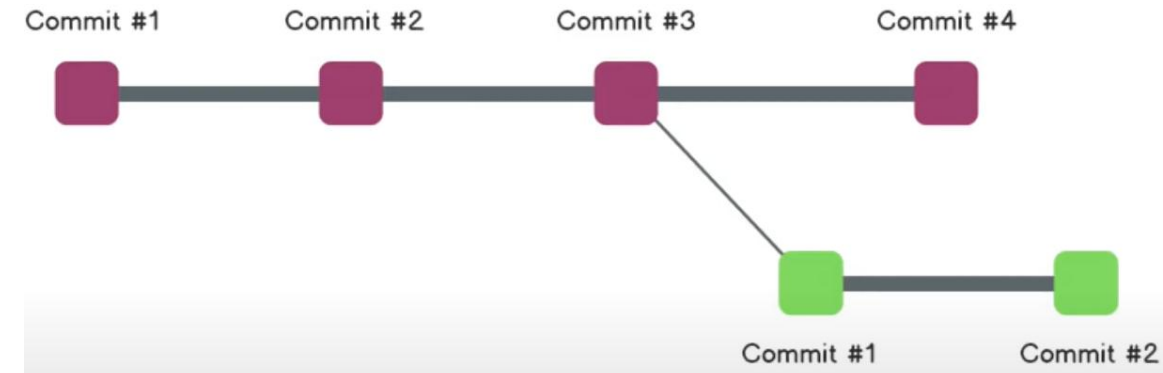
Agora:

- Fazer exercício 1 e 2

Ciclo do Software e Versionamento

Branch

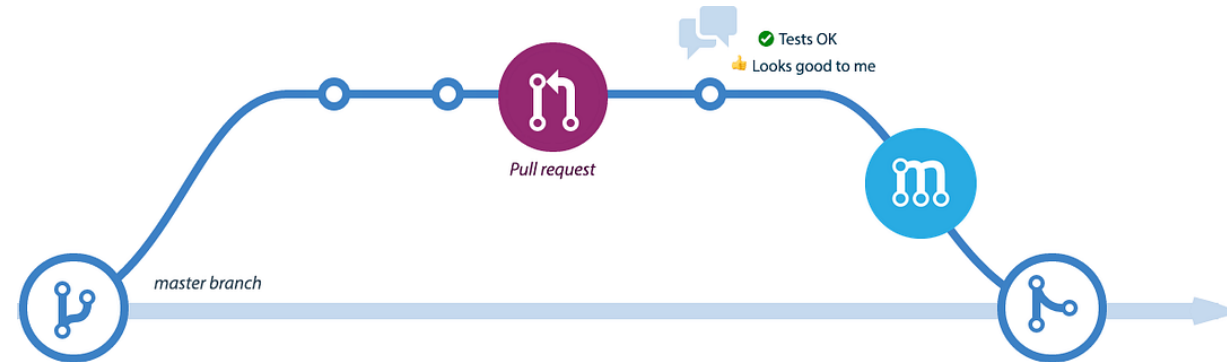
- Linha de desenvolvimento separada
- Erros ou funcionalidades do software muitas vezes só estão presentes em determinadas versões
- Possibilita desenvolver duas versões do software simultaneamente, uma versão tem erros corrigidos, mas não possui novas funcionalidades, e outra versão onde novas funcionalidades estão sendo trabalhadas



Ciclo do Software e Versionamento

Pull Request / Merge Request (PR / MR)

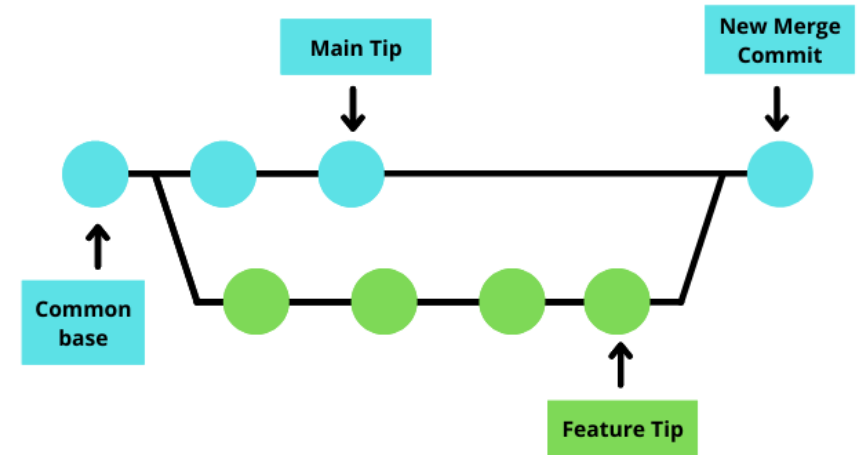
- Usado para contribuir com alterações feitas de um branch para outro
- Permite que outros desenvolvedores revisem, discutam e sugiram modificações antes de as alterações serem mescladas
- Uma vez que o PR foi revisado e aprovado pelos membros necessários da equipe, pode ser feito o **merge**
- Caso as mudanças não forem necessárias ou soluções melhores terem sido encontradas, um PR pode não ser aceito e pode ser fechado sem ser feito o merge



Ciclo do Software e Versionamento

Merge

- Processo de combinar as alterações de uma branch em outro
- É tipicamente usado para integrar o trabalho de uma branch de funcionalidade ou bug de volta na branch principal
- Um **conflito de merge (merge conflict)** acontece quando o Git não consegue mesclar automaticamente as alterações porque a mesma parte do código foi editada de maneiras diferentes em duas branches
- A dica é sempre trabalhar com a versão atualizada dos arquivos
- Merge conflicts podem ocorrer por:
 - Edições simultâneas: Duas pessoas modificam as mesmas linhas em um arquivo
 - Exclusão vs. Modificação: Uma pessoa exclui um arquivo enquanto outra o edita



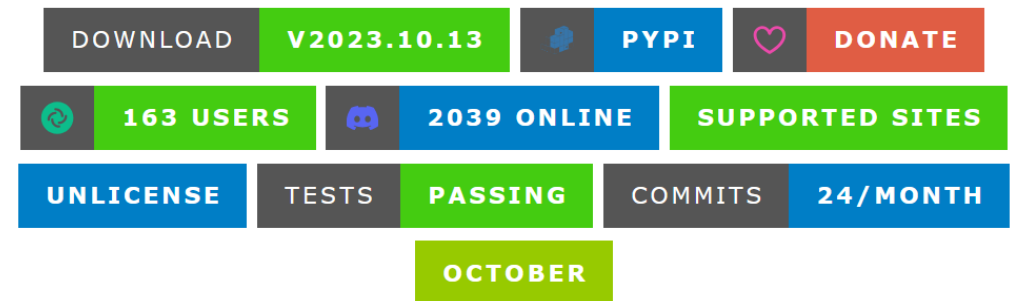
Ciclo do Software e Versionamento

Fork

- Processo de criar uma cópia pessoal de um repositório, tipicamente de projetos de código aberto
- Feito para implementar novas funcionalidades ou consertar bugs
- Tem a opção de contribuir com o projeto original a partir de um Pull Request



YT-DLP *A youtube-dl fork with additional features and fixes*



yt-dlp is a [youtube-dl](#) fork based on the now inactive [youtube-dlc](#). The main focus of this project is adding new features and patches while also keeping up to date with the original project

Ciclo do Software e Versionamento

Criando outra branch

- Para criar uma outra branch:
 - Digite no terminal dentro da pasta do repositório: **git checkout -b <nome da branch>**
 - Para se locomover entre branches já existentes, repita o código omitindo o <-b>
 - Mencione o nome da branch no push. Ex: **git push -u origin outra-branch**

```
Henrique@Henrique-PC MINGW64 ~/Desktop/Merge/teste-branch (main)
$ git checkout -b outra-branch
Switched to a new branch 'outra-branch'

Henrique@Henrique-PC MINGW64 ~/Desktop/Merge/teste-branch (outra-branch)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Henrique@Henrique-PC MINGW64 ~/Desktop/Merge/teste-branch (main)
$
```

Agora:

- Fazer exercício 3 e 4

Ciclo do Software e Versionamento

GitHub Desktop

- Aplicativo de interface gráfica (GUI) para interação com repositórios do GitHub

