

Git

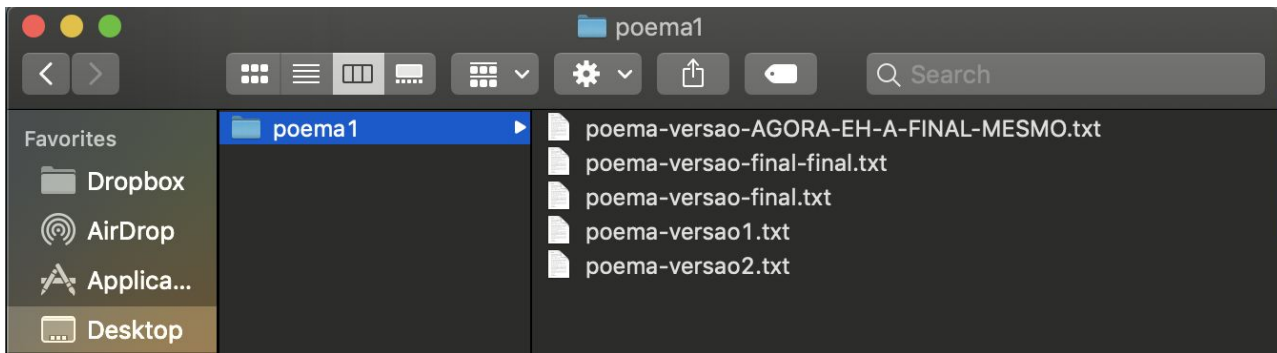
O que vamos ver hoje?

- Gerenciamento de código com Git
- Diferença entre Git e Github
- Comandos do Git

Motivação

Trabalhos e mais trabalhos

- Todos já tivemos que fazer vários trabalhos de escola
- Antes do surgimento de plataformas Cloud (como o Google Drive), tínhamos o costume de fazer assim:



Trabalhos e mais trabalhos

- Como fazíamos projetos em grupo?
 - Mandando os arquivos **separados** em um e-mail
 - E depois, alguém **sozinho** pegava o trabalho e formatava do jeito que tinha que ser

Trabalhos e mais trabalhos

- Por que guardamos versões dos nossos trabalhos?
 - Não perder **ideias antigas**
 - Poder **voltar atrás** em alguma decisão
 - Acompanhar **a evolução** que estamos fazendo
- O **git** é uma ferramenta que permite fazermos o **gerenciamento de versão** de nossos projetos (de programação ou não)

Trabalhos e mais trabalhos

- O **git** também facilita o trabalho **colaborativo**
- É muito fácil manter o **rastreamento** de arquivos que são alterados por duas pessoas ao mesmo tempo

Um pouco de história

Um pouco de história

- Este problema de versionamento é algo que já preocupava a comunidade científica (em especial, as pessoas desenvolvedoras) há bastante tempo
- Um dos primeiros sistemas a surgir foi o **bitkeeper**
- Mas o bitkeeper não era bom, e isso ficou evidente quando um dev começou a fazer um dos projetos open-source mais famosos da história...

Um pouco de história

- Em 1991, **Linus Torvalds** começou a elaborar o sistema operacional **Linux**
- A princípio, ele só **queria testar** seus conhecimentos de programação e criar o seu **próprio sistema operacional**
- Segundo Linus, seria "algo simples"

Um pouco de história

- O resultado é que o Linux se tornou o SO mais usado por pessoas desenvolvedoras no mundo
- Com o tempo, o projeto foi crescendo e se tornando cada vez mais importante
- Por ser um projeto **open-source**, qualquer pessoa poderia **contribuir** escrevendo código ou sugerindo funcionalidades e melhorias

Um pouco de história

- O **bitkeeper** começou a **não** ser mais o **suficiente**:
 - Ele era bastante lento e passou a ser pago
- Com isso, Linus e sua equipe decidiram criar o próprio **version control software** (VCS - software de controle de versão)
- Surge daí o "**git**", o software de controle de versão que ninguém do mundo da tecnologia imagina viver sem...

Git vs. Github

Git vs. Github

- O **git** é a **ferramenta** que gerencia as versões e colaborações em projetos
- O **Github** é um **serviço cloud** que permite armazenar os projetos
 - Existem outros, como Bitbucket e Gitlab. Todos usam a mesma ferramenta, o **git**.

Github

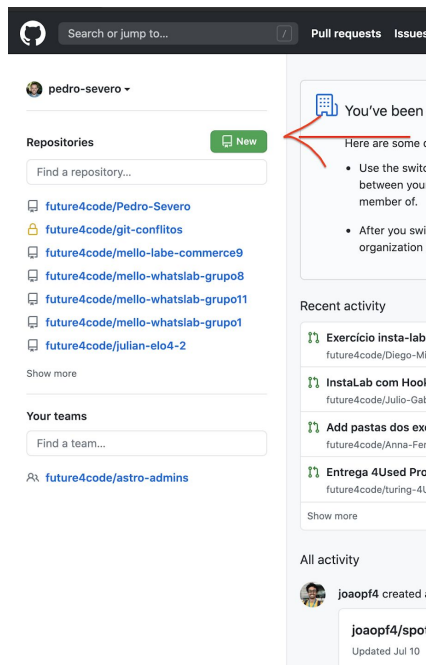
- O projeto que está na nossa máquina chamados de **repositório (ou *repo*) do git local**
- O projeto que está no github, chamados de **repositório (ou *repo*) do git remoto**

Comandos I

Começando o repositório

Começando o repositório

- Vamos começar **criando um repositório** no Github



Create a new repository

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

Owner



joaogolias

Repository name *



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

Description (optional)



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

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

☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer.

Add .gitignore: **None**

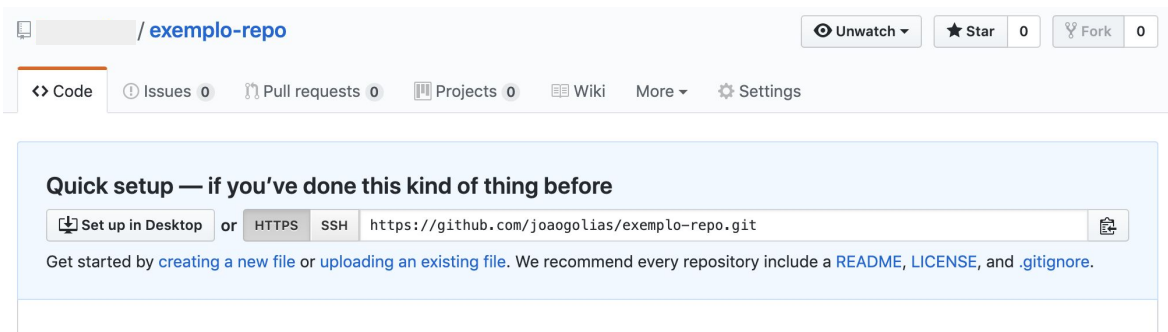
Add a license: **None** ⓘ

Create repository

Começando o repositório

- **git clone link-do-repo**

- É o comando que clona as informações do repositório remoto em uma pasta (repositório) na nossa máquina



Vamos ver na prática! 

Fixação

- O **git** surgiu como uma **ferramenta** que propõe facilitar o **versionamento** e a **colaboração** em qualquer tipo de projeto
- **Github** é a **plataforma** que guarda os repositórios na **nuvem**

Comandos II

Salvando Localmente

Salvando Localmente

- **git status**
 - Indica o status do repositório
 - Arquivos/pastas criados
 - Arquivos/pastas modificados
 - Arquivos/pastas removidos

Salvando Localmente

- **git add nome-do-arquivo**
 - Envia os arquivos modificados, removidos e criados para a Staging Area (que é local)
 - Também podemos utilizar a opção **git add --all** para adicionar todos os arquivos do repositório;
 - Ou a opção **git add .** para adicionar todos os arquivos da pasta onde você se encontra;

Salvando Localmente

- `git add .`



Salvando Localmente

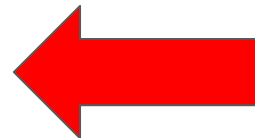
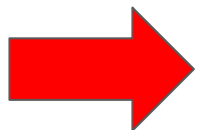
- **git commit -m "mensagem"**
 - Demarca uma versão do seu projeto com os arquivos que estiverem na Staging Area
 - A mensagem deve explicar as modificações, criações e deleções feitas

Salvando Localmente

- **git commit -m "mensagem"**
 - Não esquecer do -m
 - **Caso esqueça**, você vai entrar em uma parte do terminal, que, para sair, você deve digitar:
esc esc :q
 - Não esquecer das aspas ("")

Salvando Localmente

- **git commit -m "mensagem"**
 - REPETINDO PQ É MTO IMPORTANTE:
 - Não esquecer do -m



Salvando Localmente

- **git log**
 - Permite verificar o histórico de commits do projeto

Vamos ver na prática! 

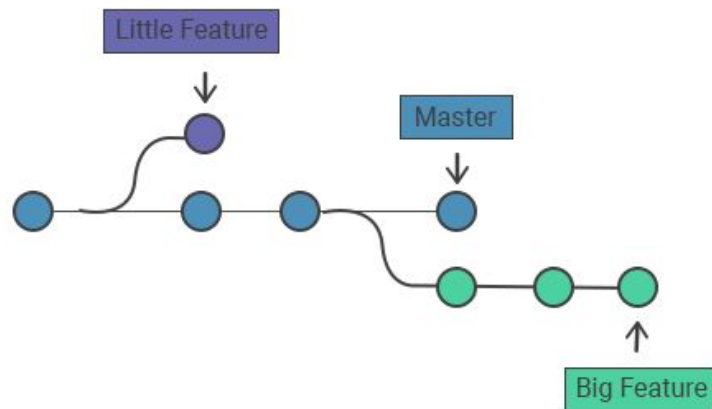
Comandos III

Dividindo o trabalho

Dividindo o Trabalho

- **git branch**

- Branch (ramo/galho) é uma ramificação do projeto principal



Dividindo o Trabalho

- **git branch**

- Este comando em si mostra a lista de branches que estão no seu repositório local
- A branch padrão se chama main* e, a princípio, apenas ela vai existir no seu repositório

* Anteriormente a branch padrão se chamava master, hoje em dia apenas repositórios antigos permanecem com esse nome.

Dividindo o Trabalho

- **git branch nome-da-branch**
 - Permite criar uma nova branch, com o nome que você escolheu

Dividindo o Trabalho

- **git checkout nome-da-branch**
 - Permite acessar uma branch que já foi criada (localmente ou remota)

Dividindo o Trabalho

- **git checkout -b nome-da-branch**
 - É uma junção dos comandos anteriores
 - Ele cria uma nova branch e já acessa diretamente

Vamos ver na prática! 

Fixação

- git clone
- git status
- git add nome-do-arquivo
- git commit -m "mensagem"
- git log
- git branch
- git branch nome-da-branch
- git checkout nome-da-branch
- git checkout -b nome-da-branch

Comandos IV

Salvando no Remoto

Salvando no Remoto

- **git push origin nome-da-branch**
 - Envia as suas alterações feitas para a branch no repositório remoto
 - Ele só envia as alterações que foram colocadas no commit

Vamos ver na prática! 

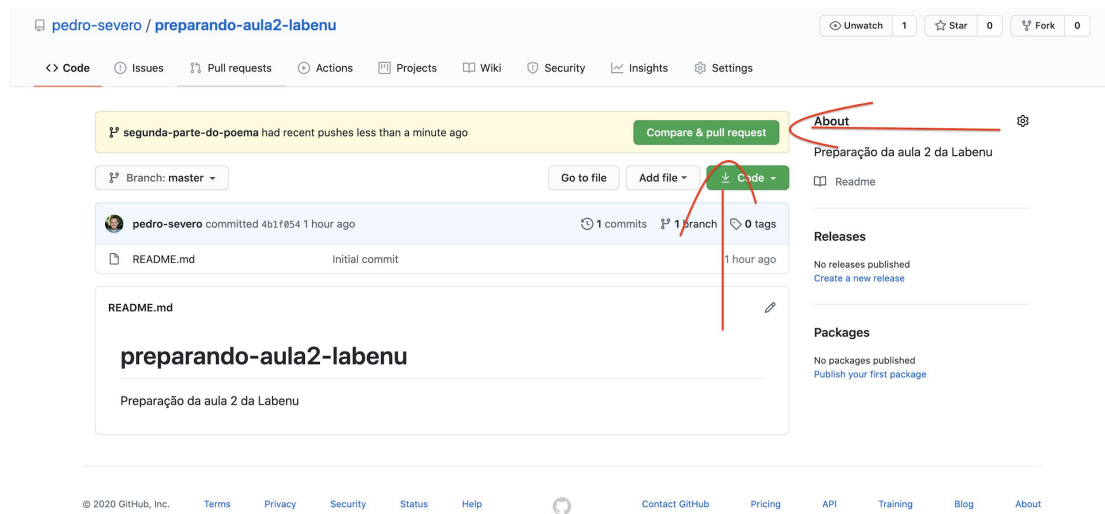
PR

Pull Request (PR) 🙄

- Depois de fazer todas as alterações na sua branch, você deve querer que elas sejam mescladas com a branch principal (a master)
- A esta **mesclagem**, damos o nome de **merge**

Pull Request (PR) 🙄

- Para fazer um merge no GitHub, nós devemos criar um **Pull Request** (ou PR) antes



<https://github.com/pedro-severo/preparando-aula2-labenu/pulls>

Pull Request (PR) 🙄

- Quando trabalhamos em equipe, os membros dela avaliam os nossos PRs
 - Pedindo correções no código
 - Sugerindo alterações
- Após o processo de **Code Review** (CR); e o seu código estiver **aprovado**, ele pode ser **mergeado** na main

Vamos ver na prática! 🧪

Comandos V

Atualizando o local

Atualizando o local

- **git pull origin nome-da-branch**
 - Atualiza a branch em questão no seu repositório local com as alterações commitadas na branch remota
 - Se você já estiver acessando a branch que deseja atualizar, o comando pode ser reduzido a git pull

Vamos ver na prática! 

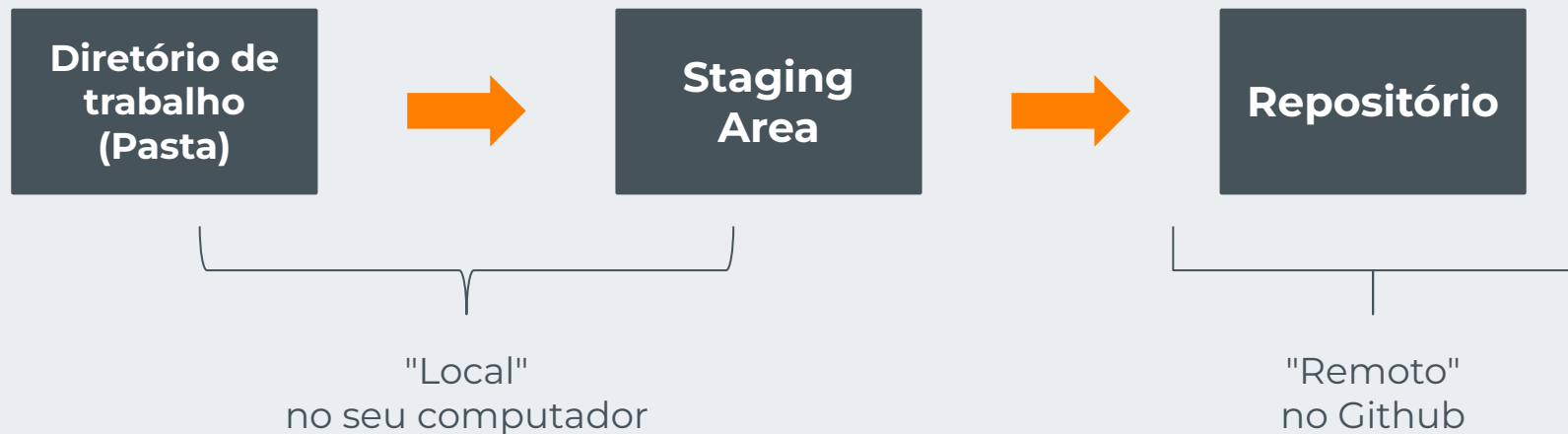
Resumo

Resumo

- O **git** é uma ferramenta que ajuda muito o dia a dia de desenvolvedora(e)s, porque:
 - Permite gerenciar várias **versões do código**
 - Facilita o trabalho colaborativo em equipes
- O **GitHub** é um sistema cloud que permite que guardemos os nossos repositórios remotos

Resumo

- **Staging area:**



Resumo

- **Começando o repositório**
 - git clone link-do-repo
- **Salvando localmente**
 - git status
 - git add nome-do-arquivo
 - git add .
 - git commit -m "mensagem"
 - git log

Resumo

- **Dividindo o Trabalho**

- git branch
- git branch nome-da-branch
- git checkout nome-da-branch
- git checkout -b nome-da-branch

- **Salvando no Remoto**

- git push origin nome-da-branch
- git pull origin nome-da-branch

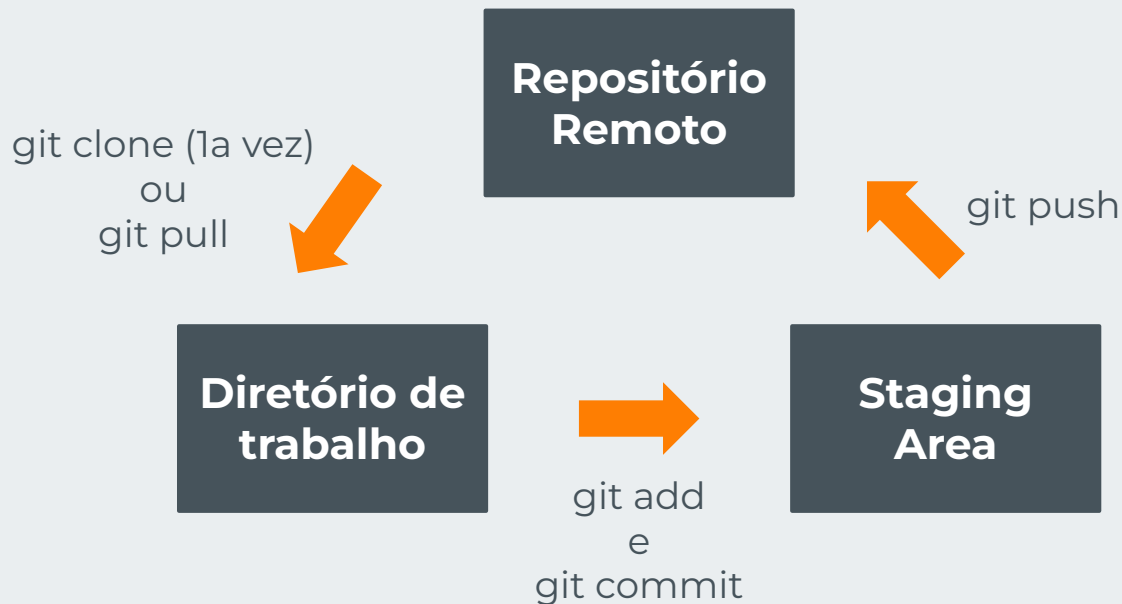
Resumo

- Sempre queremos que as alterações de uma branch nossa **sejam mescladas com as informações que já estão na master** (merge)
 - Para isso , devemos criar um PR
 - Solicitando aos nossos colegas de trabalho que **avaliem o nosso código**, dando sugestões de melhoria

Resumo

- Importante: comandos de git **não são** o mesmo que comandos do terminal!
 - **Ex:** git mkdir ❌
- Importante 2: **branch não é pasta!**

Resumo



Resumo



In case of fire



🔑 1. git commit



2. git push



3. leave building

Dúvidas?



Programa
3000 TALENTOS TI
Obrigado(a)!