Git e Github

Programa
3000 TALENTOS TI

O que vamos ver hoje?

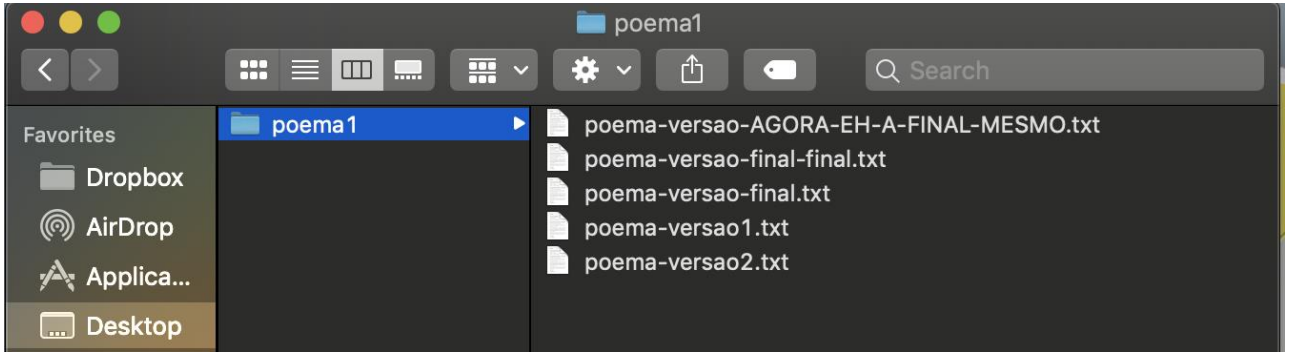
- Gerenciamento de código com Git
- Diferença entre Git e Github
- Comandos do Git

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

Problemas de versionamento referem-se a **desafios que surgem quando se lida com diferentes versões de software, código-fonte ou documentos**.

Esses problemas podem ocorrer em várias situações e **podem causar conflitos, perda de dados ou dificuldade na coordenação do trabalho entre diferentes desenvolvedores ou equipes.**

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"

Trabalhos e mais trabalhos

- Já em 2005 criou o Git. **Ele queria um sistema de controle de versão distribuído que fosse rápido e eficiente para lidar com o grande volume de contribuições ao projeto Linux**.

- **Git: G**lobal **I**nformation **T**racker | Rastreador de informações globais.

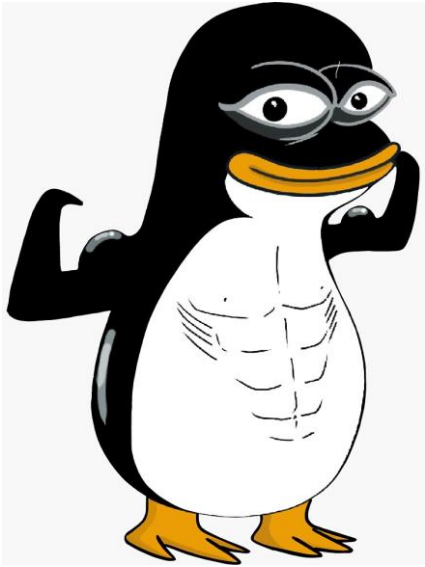
Linus Torvalds

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*** *(Código aberto)*, 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...

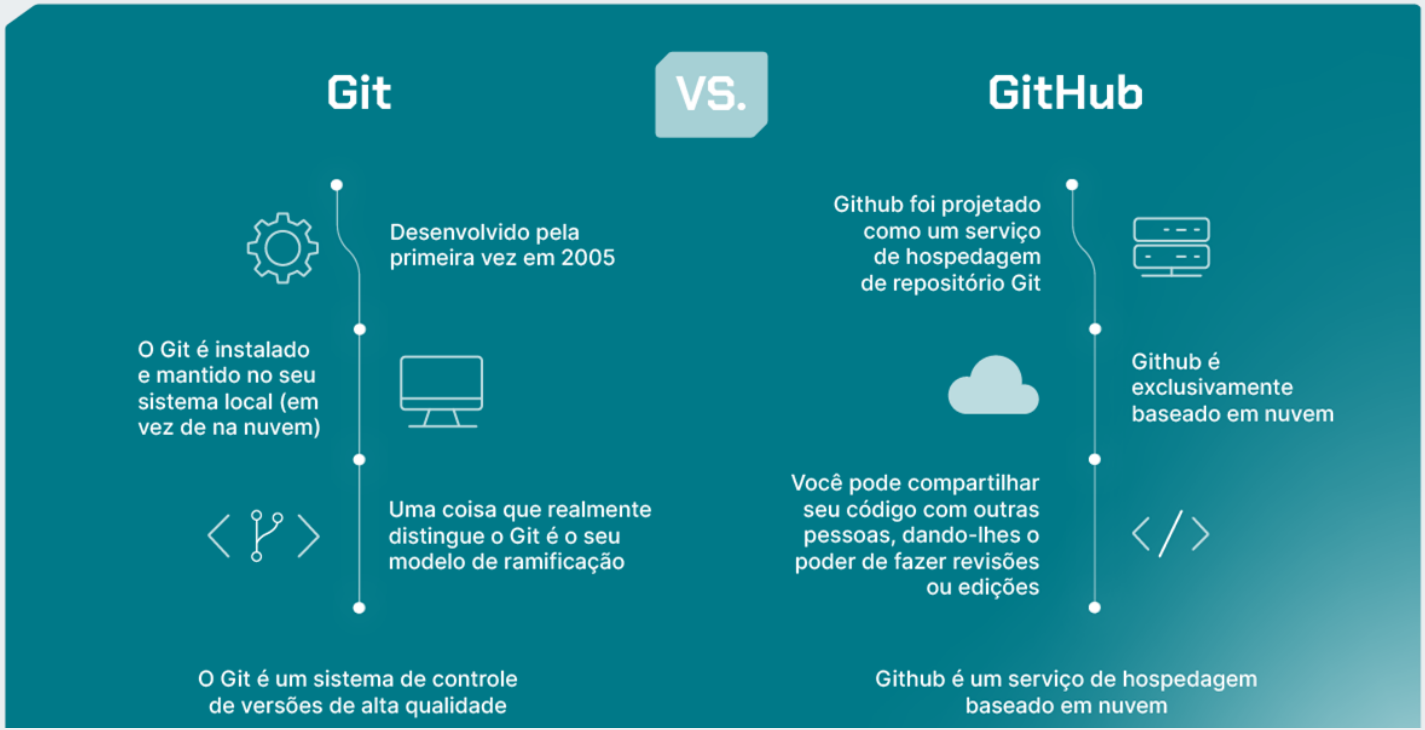
Git vs. Github

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**.

Git vs. Github



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**

Github

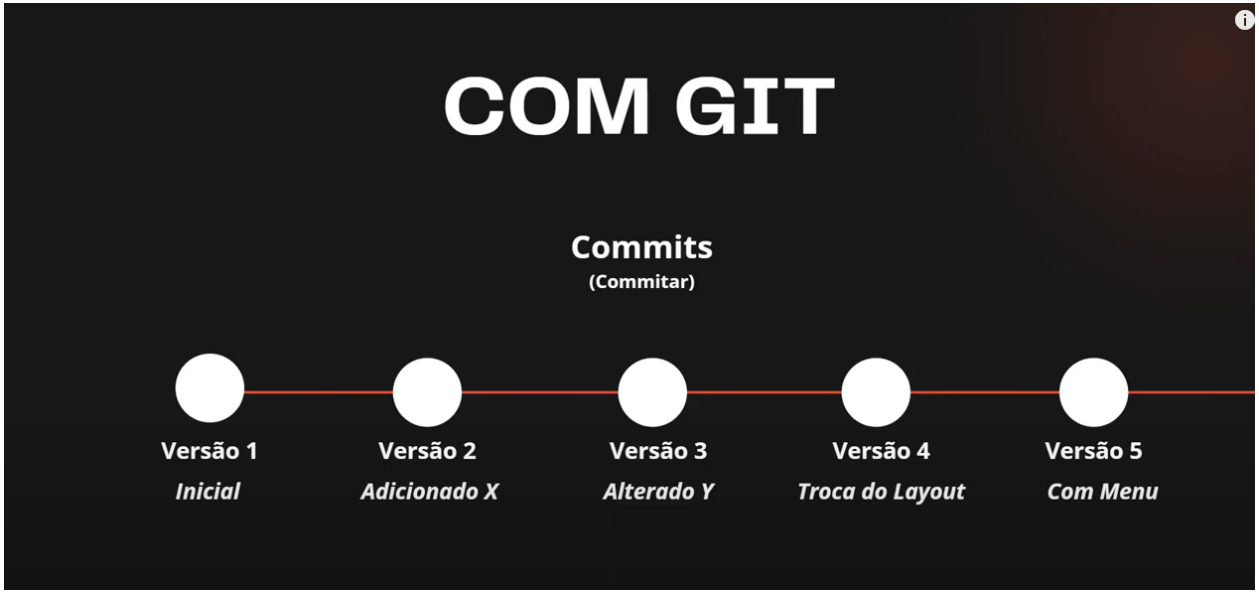
- O **Octocat** é uma criatura híbrida, misturando um gato e um polvo. Ele simboliza a flexibilidade, adaptabilidade e a natureza colaborativa do **GitHub**.

Github

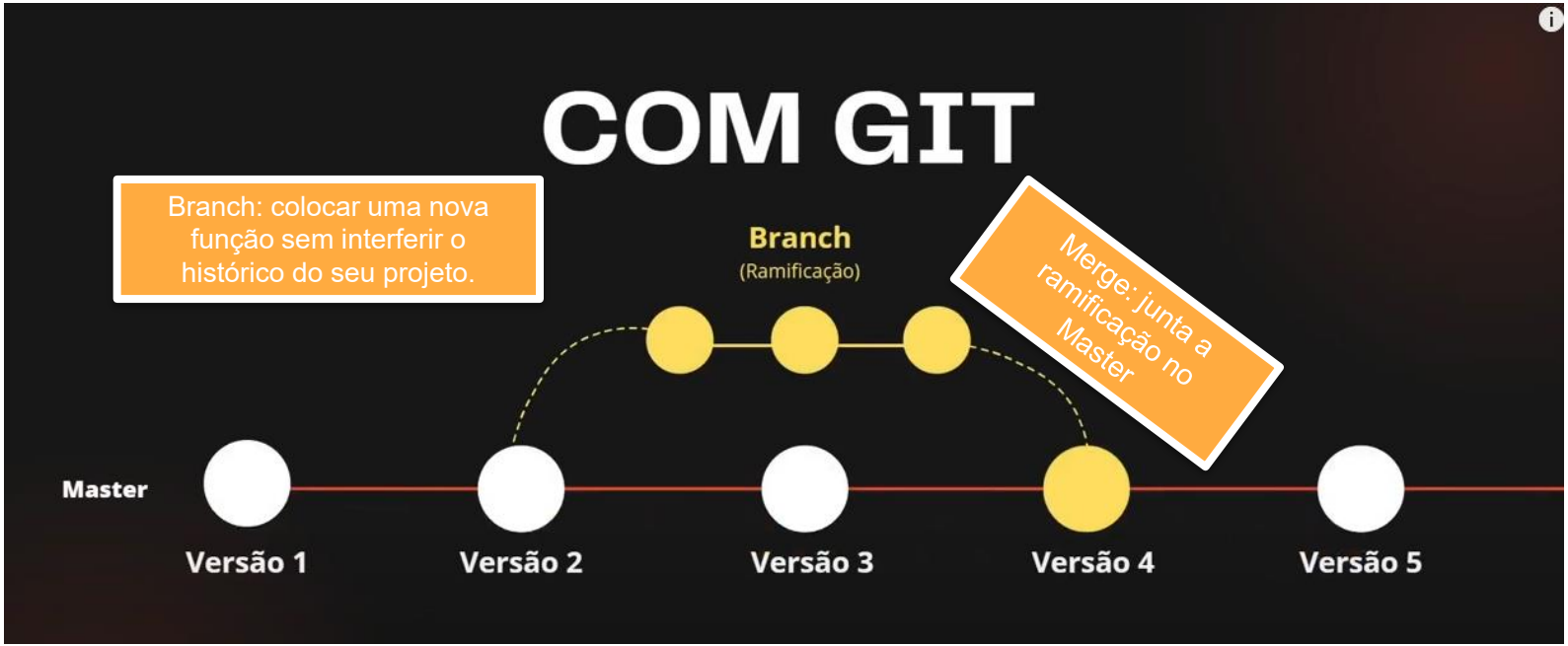
Polvos são conhecidos por seus múltiplos tentáculos, que podem ser vistos como uma metáfora para a capacidade de gerenciar múltiplos projetos e colaborar em várias frentes ao mesmo tempo.

Gatos são frequentemente associados à curiosidade e à exploração, refletindo a natureza exploratória e inovadora dos desenvolvedores que usam GitHub.

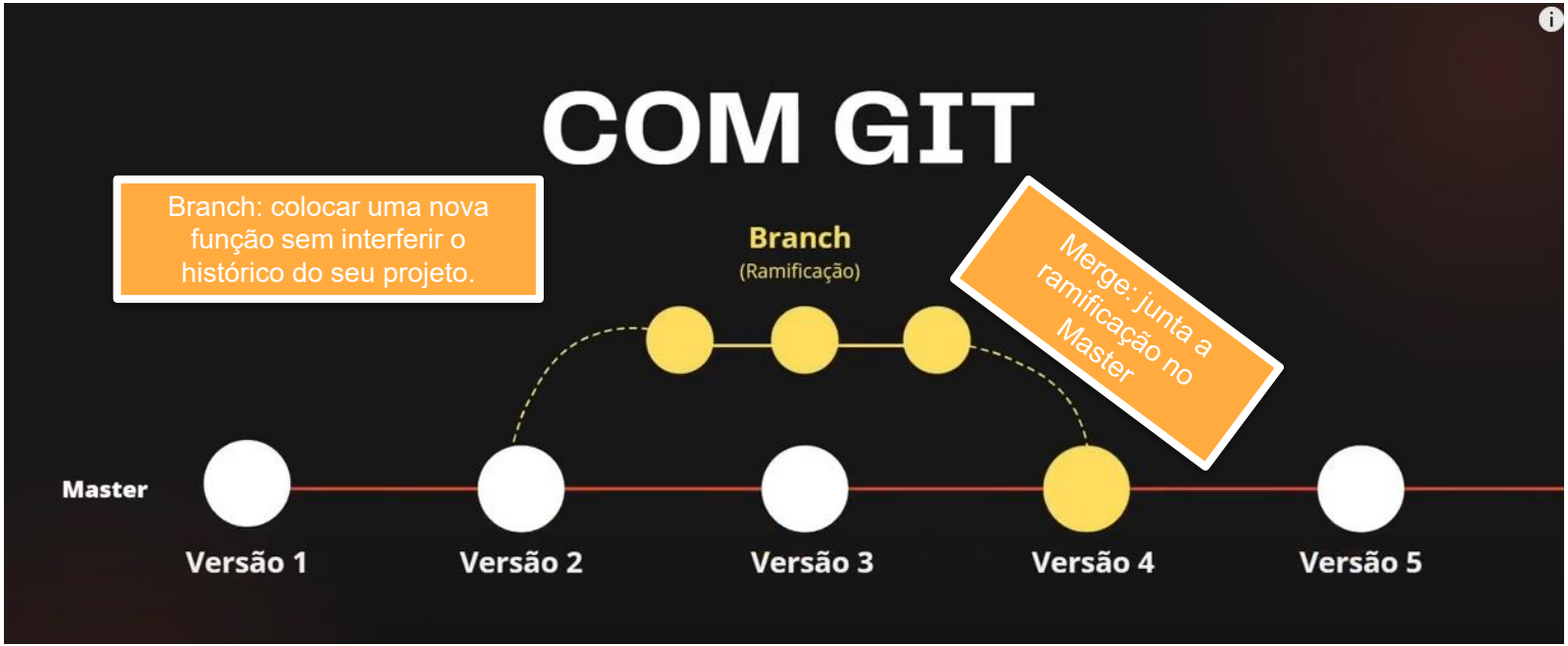
Git vs. Github



Git vs. Github



Git vs. Github



Git e VSCode

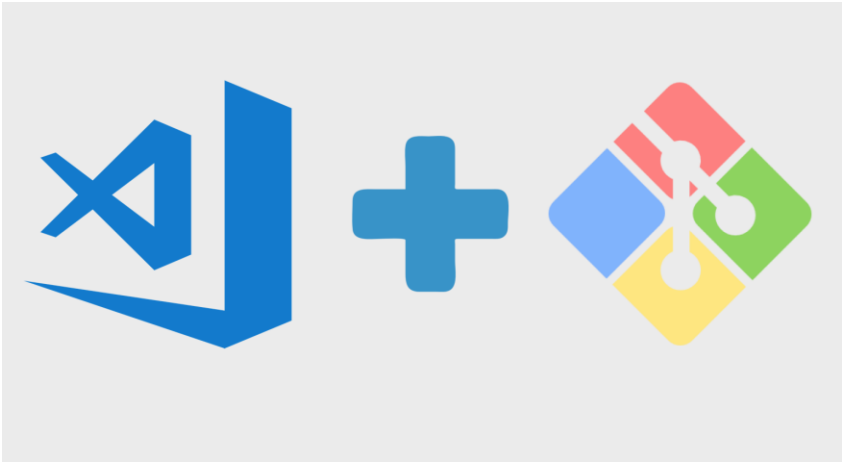
Ok. Estamos falando sobre repositórios e espaços para armazenamentos... Mas o que vou guardar lá?

Git e VSCode

Precisamos ter alguma coisa para guardar github.

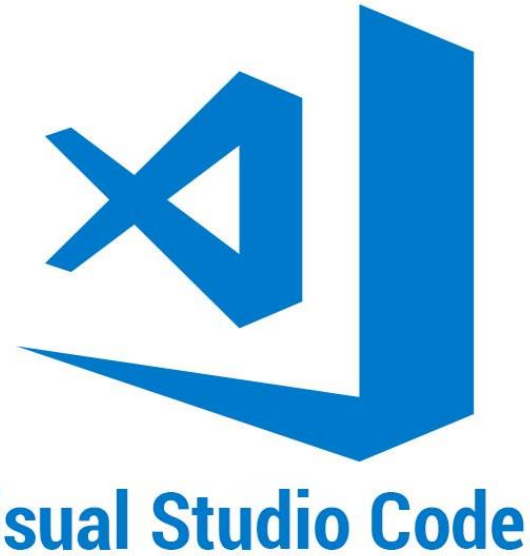
Git e VSCode

Git e Visual Studio Code (VSCode) trabalham muito bem juntos, proporcionando uma experiência de desenvolvimento integrada e eficiente. Aqui estão algumas das maneiras como você pode usar Git dentro do VSCode e os benefícios dessa integração:



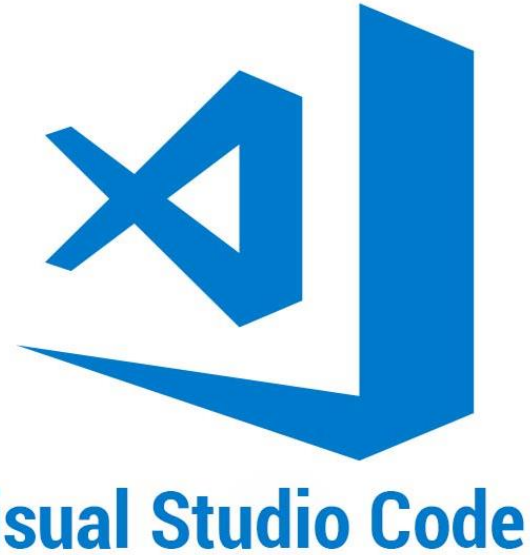
Git e VSCode

O **Visual Studio Code (VSCode)** é um editor de código desenvolvido pela Microsoft. É amplamente utilizado por desenvolvedores devido às ***suas funcionalidades poderosas, flexibilidade e suporte a uma ampla gama de linguagens e tecnologias***.



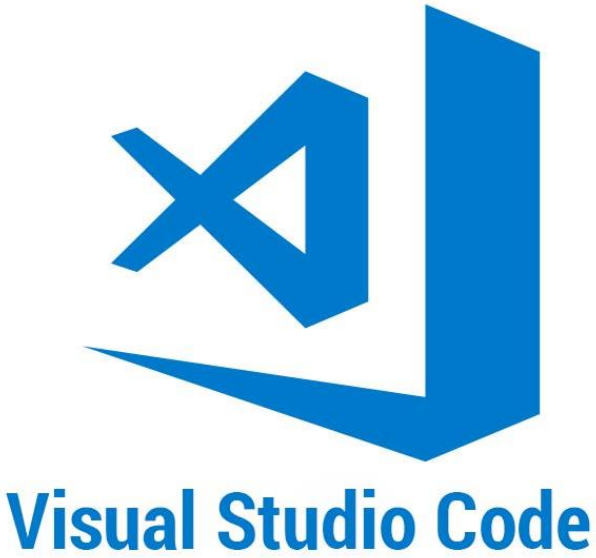
Git e VSCode

- **Principais Características do VSCode**
- **Editor de Texto Avançado**:
  - **O que é?**: É um lugar onde você pode escrever e editar código, que são as instruções que um computador segue para realizar tarefas.
  - **Por que é útil?**: Ele oferece funcionalidades especiais que ajudam a escrever código mais rapidamente e com menos erros.



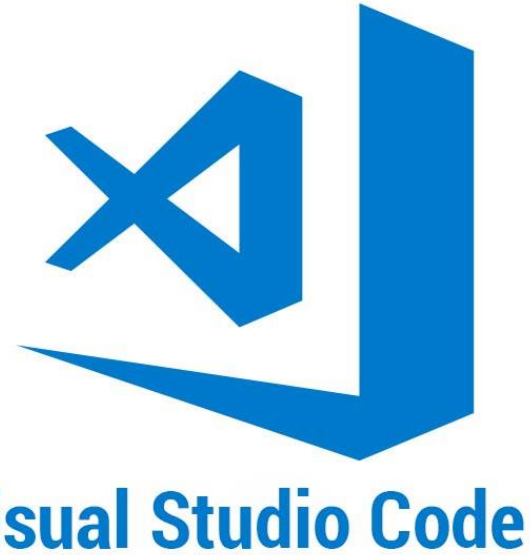
Git e VSCode

- **Sugestões e Autocompletar**:
  - **O que é?**: Quando você está digitando, o VSCode sugere palavras ou comandos que você pode usar, com base no que está escrevendo.
  - **Por que é útil?**: Ajuda a evitar erros e acelera o processo de codificação.



Git e VSCode

•**Depuração**:
•**O que é?**: Ferramentas que permitem testar o código para encontrar e corrigir problemas (ou "bugs").
•**Por que é útil?**: Facilita encontrar e corrigir erros, garantindo que o software funcione corretamente.

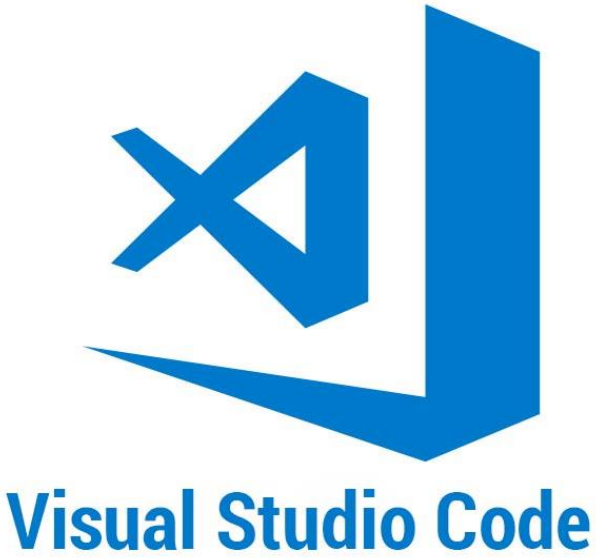


Git e VSCode

Terminal Integrado:

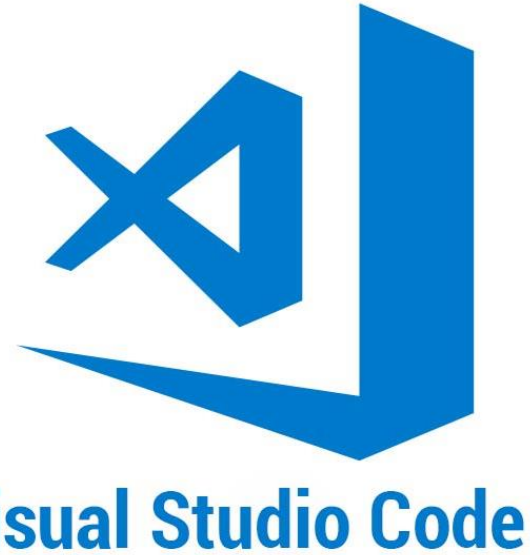
•**O que é?**: Uma área dentro do VSCode onde você pode executar comandos, como se estivesse usando um console.

•**Por que é útil?**: Permite fazer várias tarefas sem sair do editor, como compilar e executar programas.



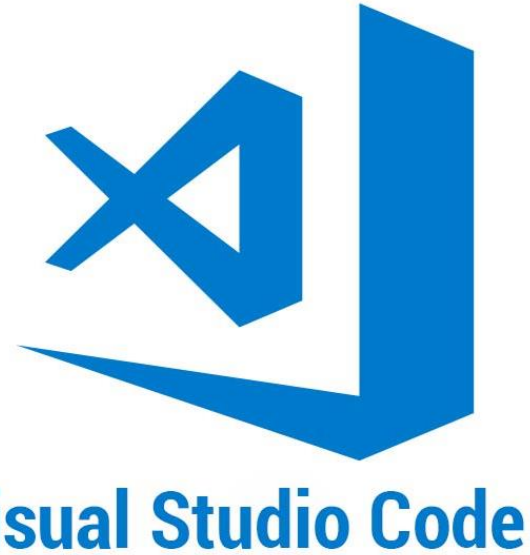
Git e VSCode

Controle de Versão:
•**O que é?**: Uma funcionalidade para acompanhar e gerenciar mudanças no código ao longo do tempo.
•**Por que é útil?**: Ajuda a reverter alterações indesejadas e a colaborar com outras pessoas em projetos de código.



Git e VSCode

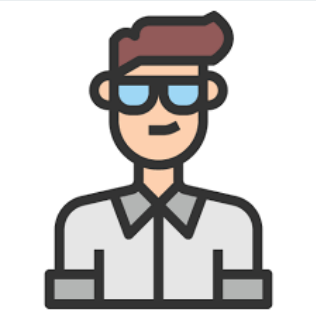
Extensões:
•**O que é?**: Pequenos complementos que você pode adicionar ao VSCode para torná-lo ainda mais poderoso.
•**Por que é útil?**: Permite personalizar o editor para atender às suas necessidades específicas, como suporte a novas linguagens de programação ou ferramentas adicionais.



Git e VSCode

Responsável pelo conteúdo

HTML



Responsável por deixar bonito

CSS

Responsável pela organização e ação de vender o livro

JavaScript

Git e VSCode

Ok. Estamos falando sobre repositórios e espaços para armazenamentos... Mas o que vou guardar lá?

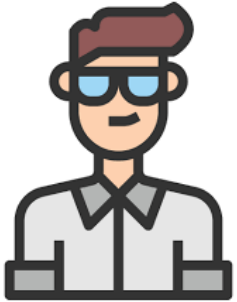
Git e VSCode

Precisamos ter alguma coisa para guardar github.

Git e VSCode

Responsável pelo conteúdo

HTML



Responsável por deixar bonito

CSS

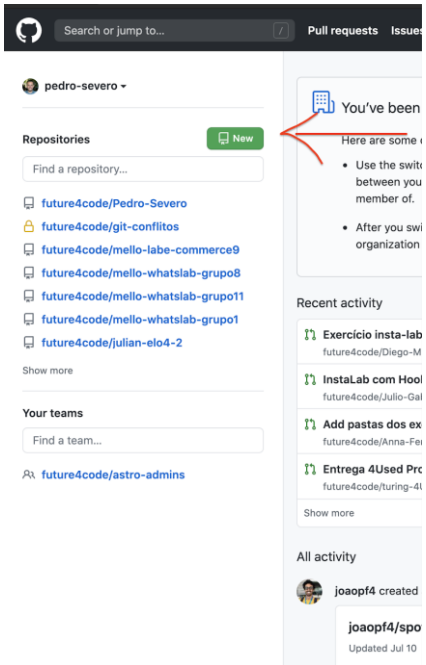
Responsável pela organização e ação de vender o livro

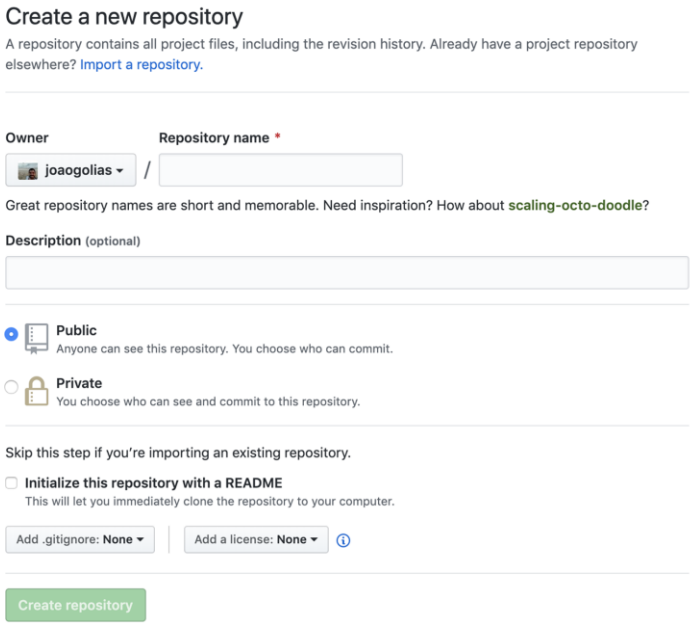
JavaScript

Comandos I
Começando o repositório

Começando o repositório 🎁

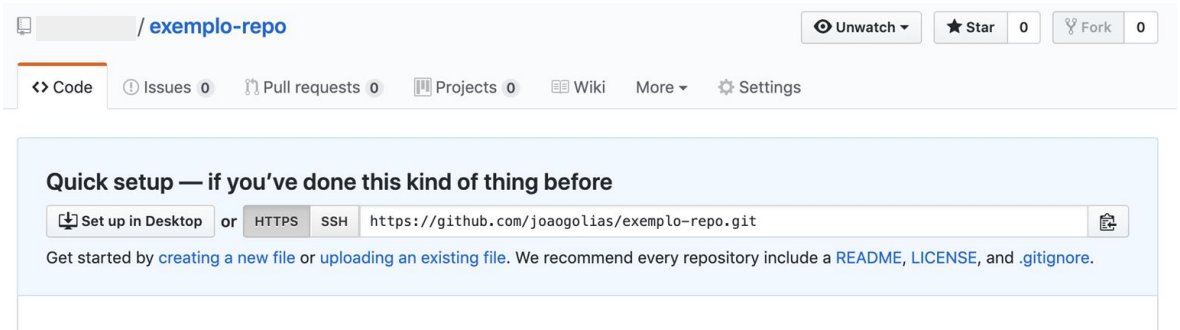
- Vamos começar **criando um repositório** no Github





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! 🔬

Programa
3000 TALENTOS TI

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

Programa
3000 TALENTOS TI

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 

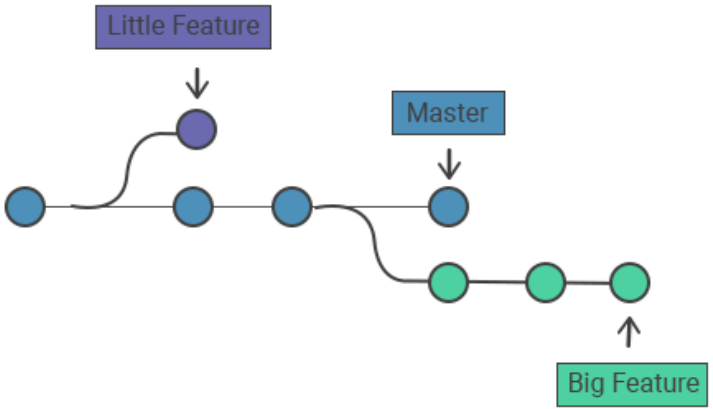
- **git log**
  - Permite verificar o histórico de commits do projeto

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! 

Programa
3000 TALENTOS TI

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

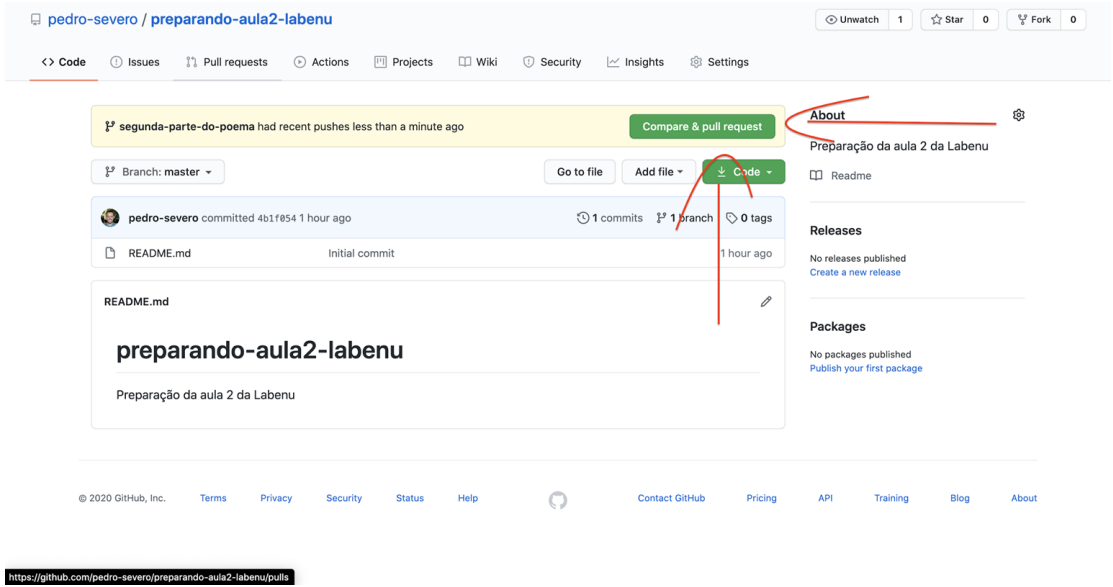
Pull Request (PR) 🥺

- 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



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

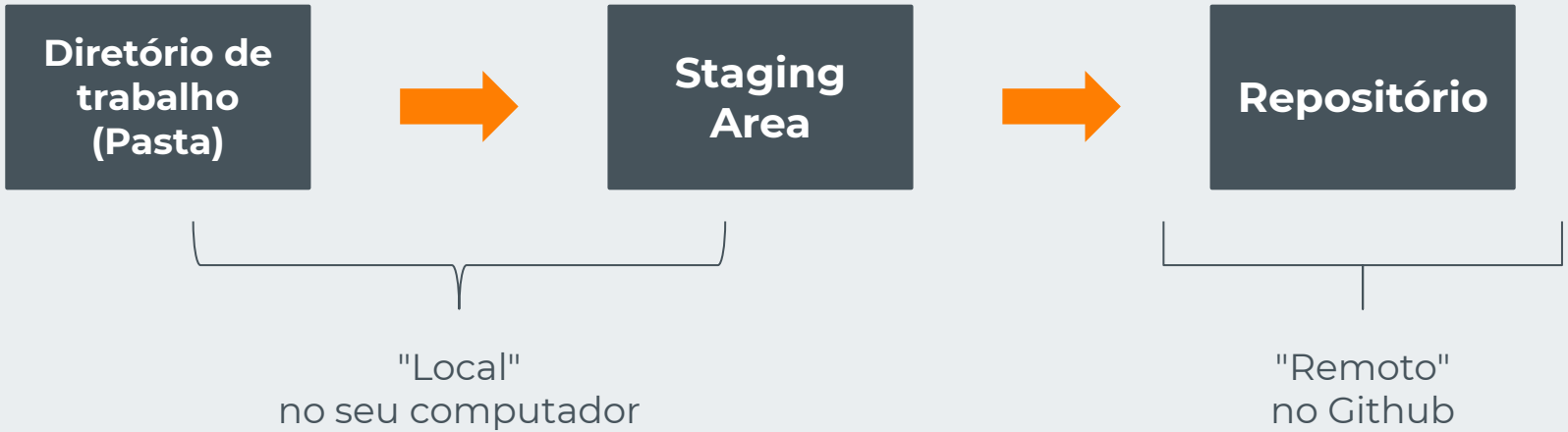
Resumo 

- 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 

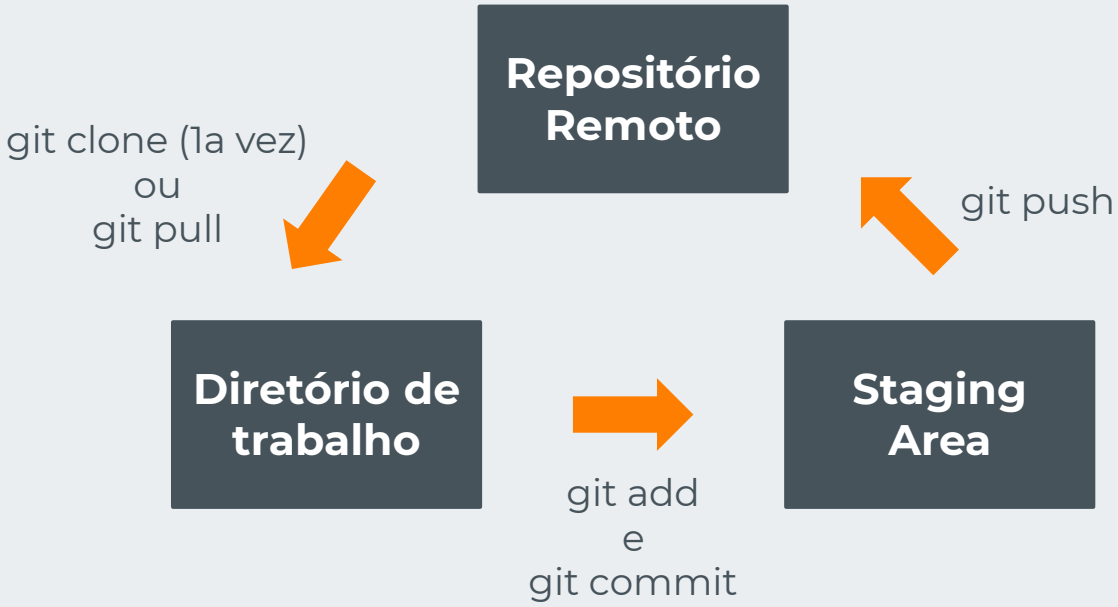
- Sempre queremos que as alterações de uma branch nossa **sejam mescladas com as informações que já estão na master** (merge)
    - Para isso , devemos criar um PR
    - Solicitando aos nossos colegas de trabalho que **avaliem o nosso código**, dando sugestões de melhoria

Resumo 

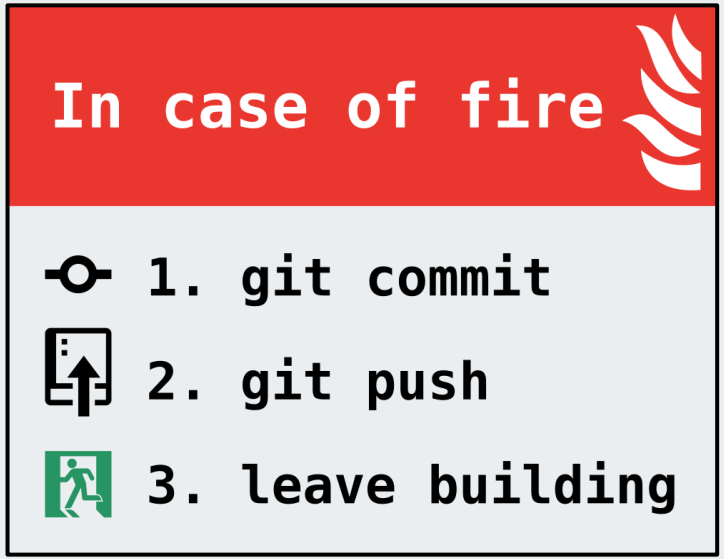
- Importante: comandos de git **não são** o mesmo que comandos do terminal!
  - **Ex:** git mkdir ❌

- Importante 2: **branch não é pasta!**

Resumo


Repositório Remoto
git clone (1a vez)
ou
git pull
git push
Diretório de trabalho
git add
e
git commit
Staging Area

Resumo



Dúvidas? 

Programa

3000 TALENTOS TI

Obrigado(a)!