

# Aula 05

Controle de versão  
Introdução

[emerson@paduan.pro.br](mailto:emerson@paduan.pro.br)

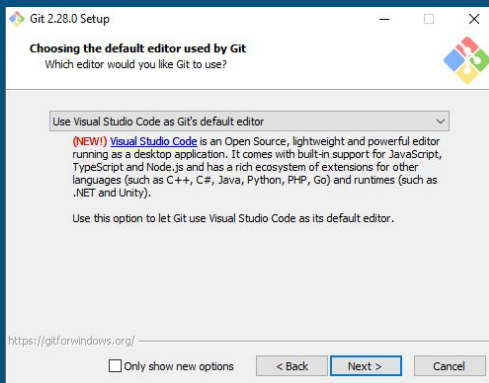
Antes de  
iniciar...

Dúvidas ???



# Git

<https://git-scm.com/downloads>



Controle de versão

[emerson@paduan.pro.br](mailto:emerson@paduan.pro.br)

## Contexto

Já passou por isso?

```
trabalho.doc  
trabalho-v02.doc  
trabalho-v03.doc  
...etc..  
trabalho-vfinal.doc  
trabalho-vfinal-ultima.doc  
trabalho-vfinal-ultima-mesmo.doc  
trabalho-vfinal-ultima-mesmo-entregue.doc
```

[emerson@paduan.pro.br](mailto:emerson@paduan.pro.br)

# O que é?

---

O que é controle de versão?

- Um sistema que mantém um registro das alterações realizadas, permitindo saber quais foram as alterações realizadas (histórico).
- Permite saber quem fez e quando fez as alterações
- Permite REVERTER as alterações feitas
- Permite o desenvolvimento colaborativo

[emerson@paduan.pro.br](mailto:emerson@paduan.pro.br)

# Comparando com outros

---

Vantagens?

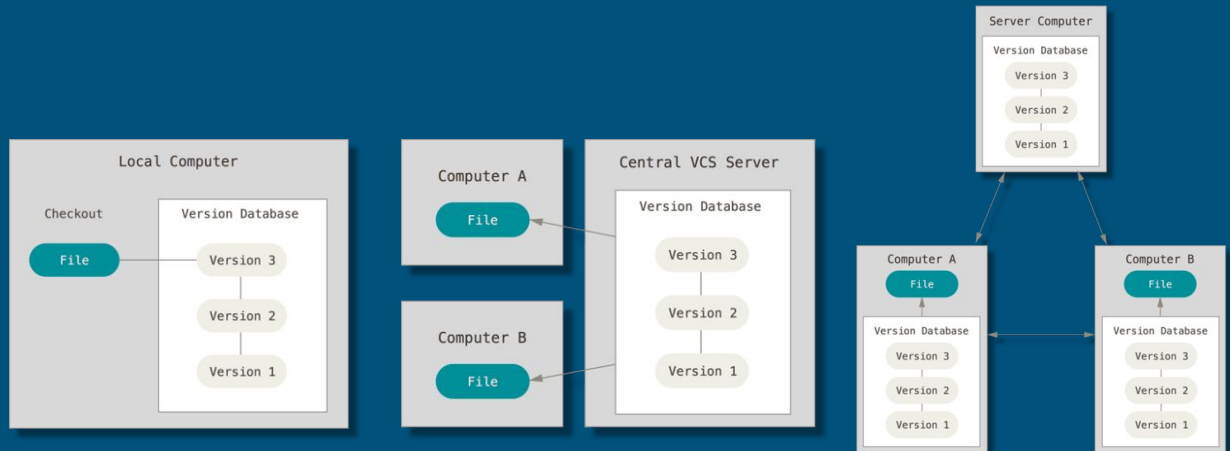
- Faz o controle de versão distribuída
- Usuários mantêm uma cópia do código completo em sua máquina local
- Desempenho
- Uma das mais utilizadas pelos desenvolvedores

[emerson@paduan.pro.br](mailto:emerson@paduan.pro.br)

# Repositórios

Conjunto de arquivos e o histórico de alterações daqueles arquivos.

Normalmente trata-se de todos os Snapshots de um projeto.



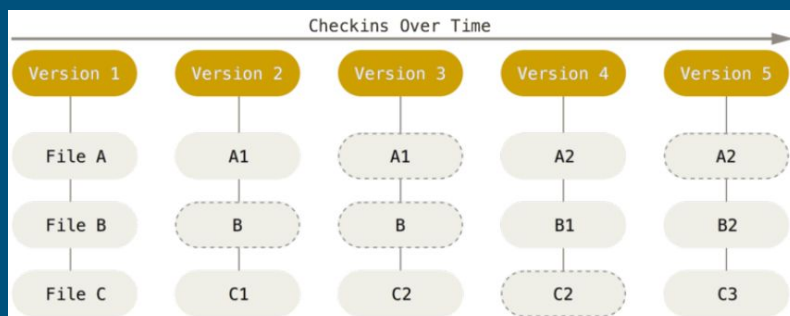
emerson@paduan.pro.br

## Snapshots



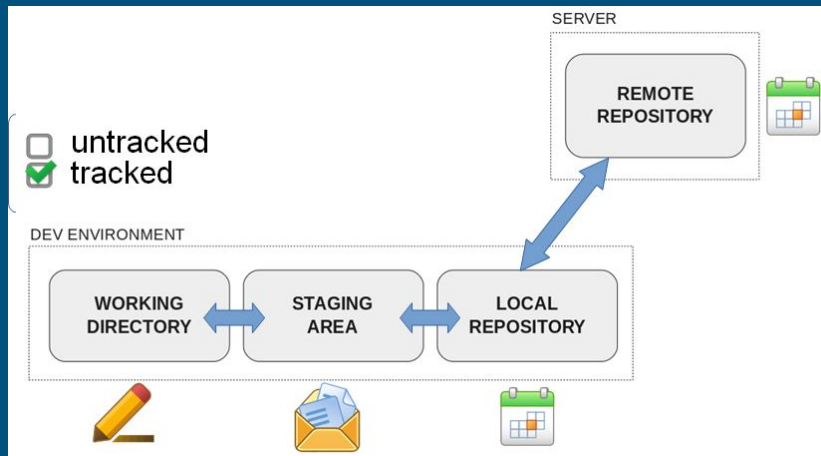
Maneira como o Git mantém um histórico do seu código.

Basicamente ele grava como estava o código em um determinado ponto (tempo).



emerson@paduan.pro.br

# Áreas do Git



emerson@paduan.pro.br

## Iniciando...

\$> `git init` [diretório]

Cria um repositório GIT no diretório indicado, ou no diretório atual.

emerson@paduan.pro.br

# Configurações

---

Caso não estejam aparecendo as cores no terminal

```
$> git config --global color.ui auto
```

Configurando o usuário:

```
$> git config [--global] user.name seuNomeDeUsuário
```

```
$> git config [--global] user.email seuE-mail
```

emerson@paduan.pro.br

# Status

---

```
$> git status
```

Lista como estão os arquivos untracked, staged, unstaged.

emerson@paduan.pro.br

# Adicionando ao stage

```
$> git add <arquivo(s)>
```

```
$> git add * / git add .
```

Adiciona dos arquivos *modificados* para o stage.

emerson@paduan.pro.br

# Commit

```
$> git commit -m "mensagem do commit"
```

Cria um commit (snapshot) dos arquivos que estão no stage.

Um commit é identificado por um **texto** informado pelo programador e um **hash code**.

Pode aparecer como substantivo  
ou verbo:

“comitei” o código 🤖  
= fiz um commit

emerson@paduan.pro.br

# Log

---

\$> `git log`

Mostra o histórico dos commits.

emerson@paduan.pro.br

# .gitignore

---

Arquivo texto que indica arquivos e/ou pastas que devem ser ignorados pelo git.

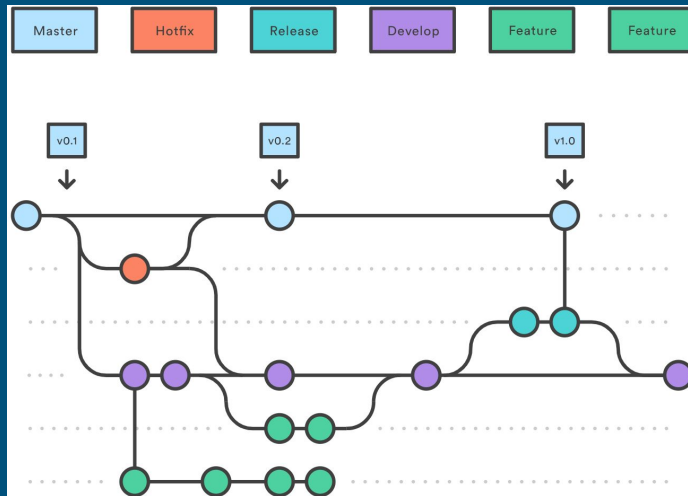
Podem ser usados padrões de nomes como `"*.class"`, por exemplo

Diretórios são indicados usando `"/"` no final como: `dir/`

emerson@paduan.pro.br



# Branches



# Branches

\$> git branch <nome da branch>  
Cria uma nova branch (ramificação).

\$> git branch

Exibe todas as branches. A atual é mostrada com um \*

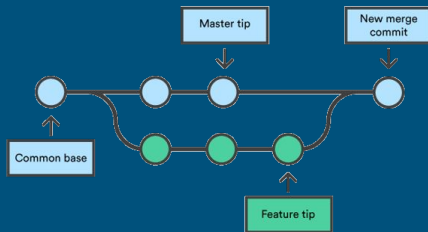
\$> **git checkout** <nome da branch>  
Alterna para uma outra branch.

\$> **git branch -d** <nome da branch>  
Apaga uma branch.

# Merge

```
$> git merge <branch1> <branch2>
```

Junção de duas branches.



emerson@paduan.pro.br

# Repositórios Git



## GitHub

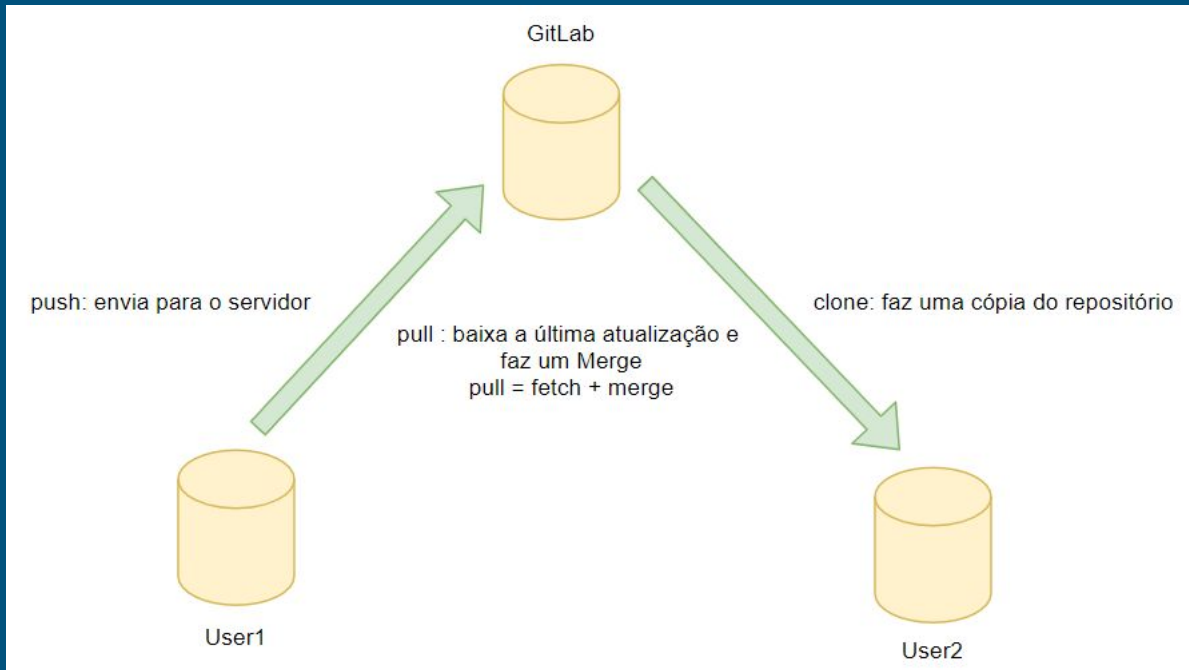
<https://github.com/>



## GitLab

<https://gitlab.com/>

emerson@paduan.pro.br



emerson@paduan.pro.br

## Remote

\$> **git clone** <url do repositório remoto>

Clona na máquina local um repositório remoto

\$> **git remote add origin** <url remoto>

Adiciona (configura) um repositório remoto para executar o push.

\$> **git push** [origin] [branch]

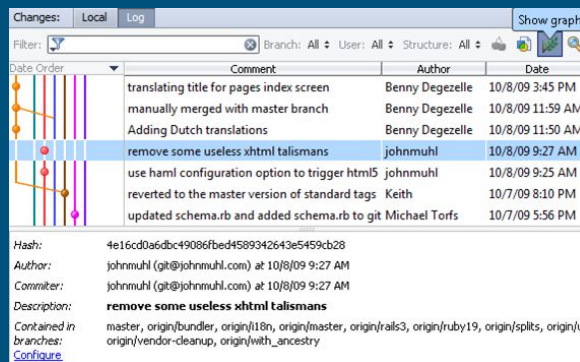
Envia o commit branch para o origin remoto.

\$ **git pull** <remote>

Faz uma cópia da branch atual do repositório remoto, para o repositório local fazendo um merge na cópia local.

emerson@paduan.pro.br

# Ferramentas visuais



emerson@paduan.pro.br

## Letras no VS Code

A - Added (This is a new file that has been added to the repository)

M - Modified (An existing file has been changed)

D - Deleted (a file has been deleted)

U - Untracked (The file is new or has been changed but has not been added to the repository yet)

C - Conflict (There is a conflict in the file)

R - Renamed (The file has been renamed)

emerson@paduan.pro.br