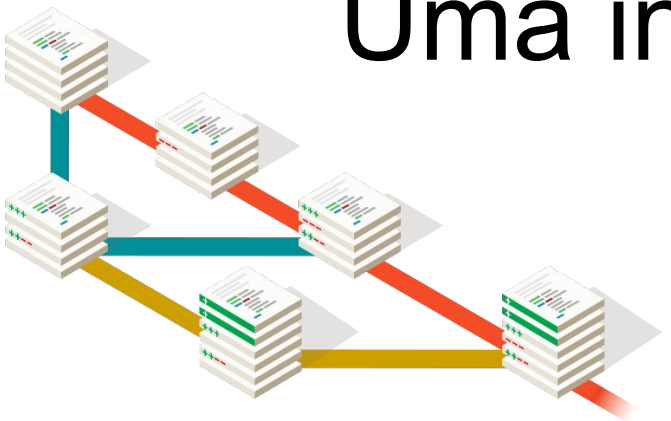
Sistema de controle de versão: Uma introdução ao Git

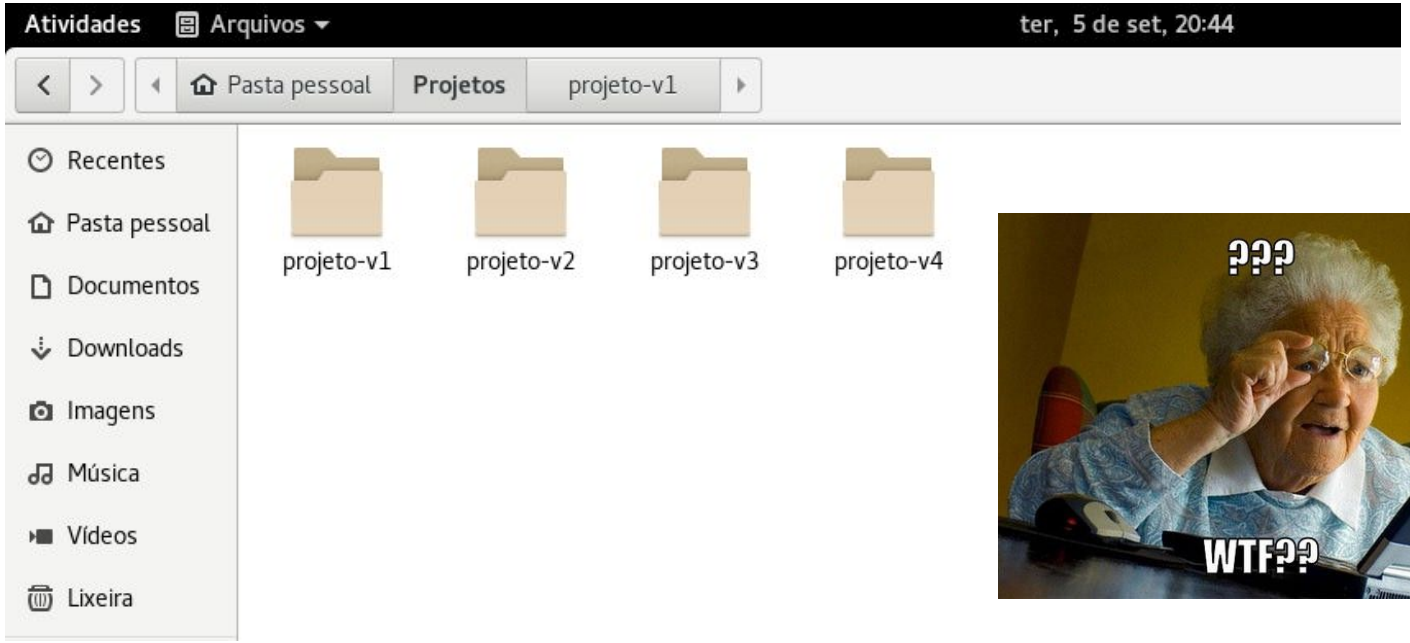


git

https://goo.gl/Gwxx4j

Índice

Sistema de controle de versão?



Sistema de controle de versão?

É um sistema que registra as mudanças feitas em um arquivo ou um conjunto de arquivos ao longo do tempo de forma que você possa recuperar versões específicas.

O que é Git?

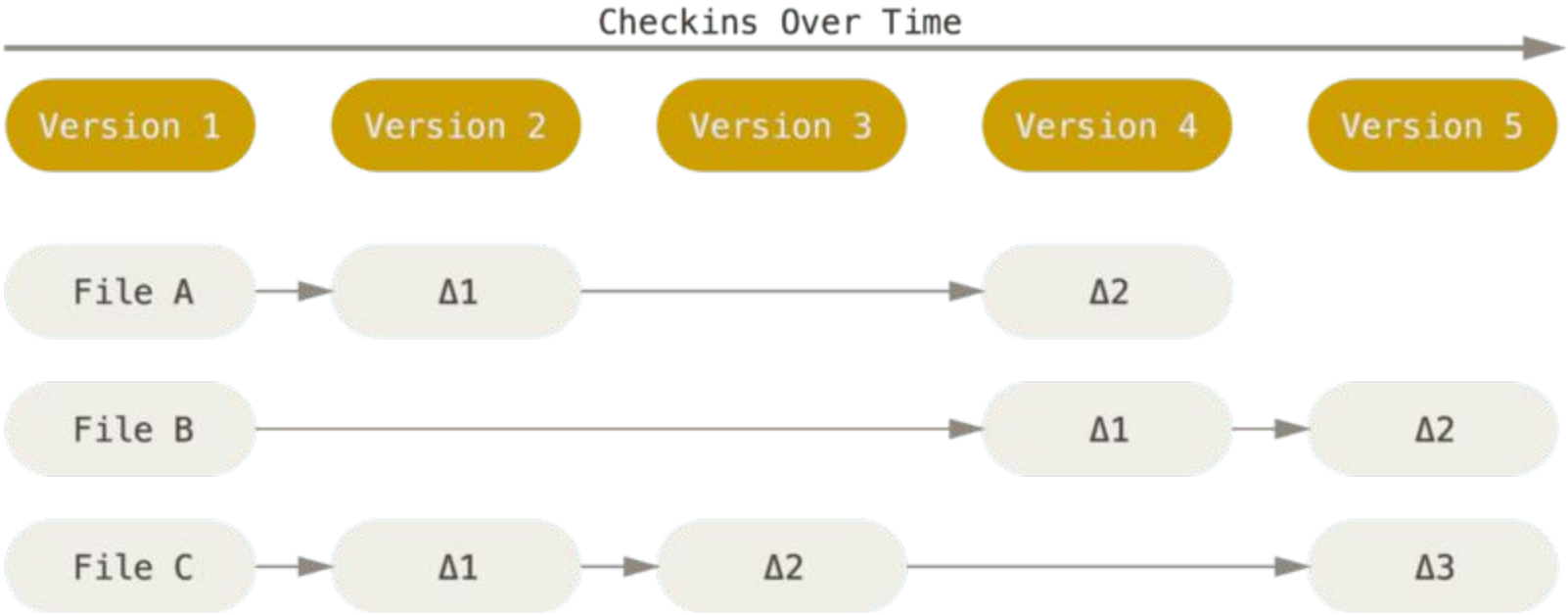
O Git é um sistema de controle de versão distribuído livre e de código aberto projetado para lidar com tudo, desde projetos pequenos a muito grandes com velocidade e eficiência.

Quem usa?

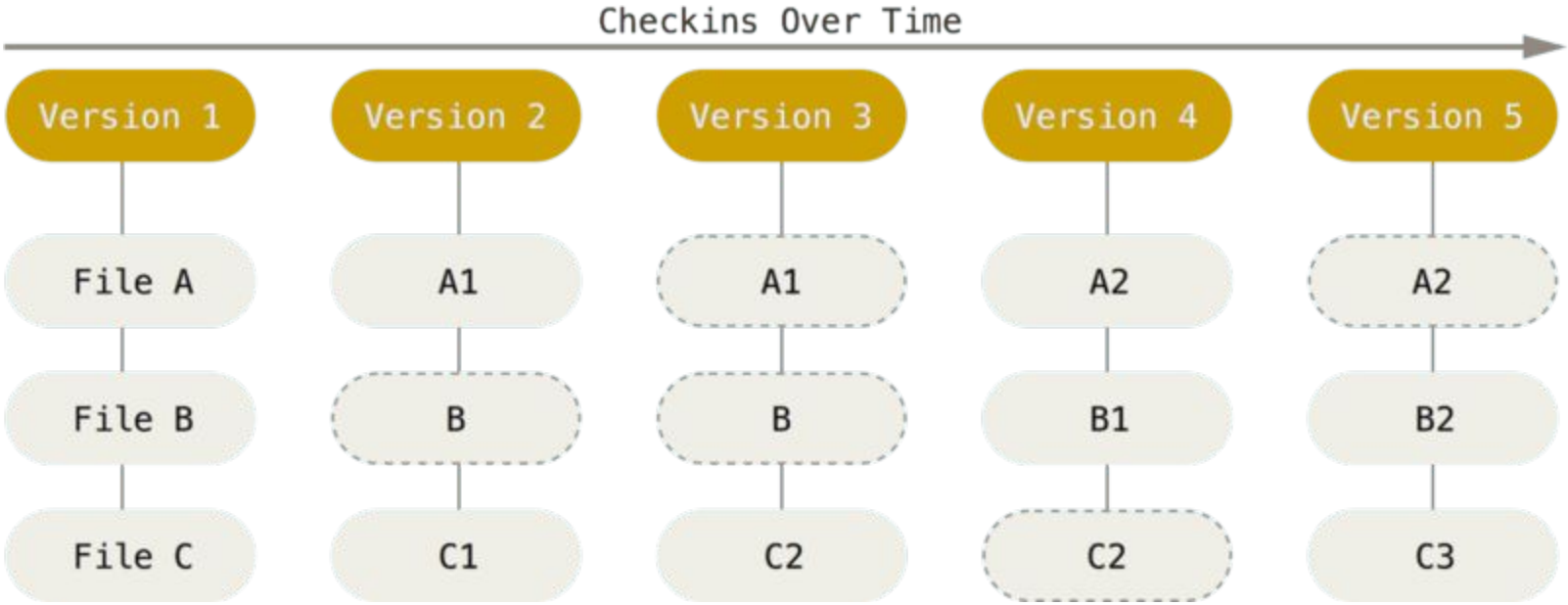

Companies & Projects Using Git
Google
facebook
Microsoft
twitter
LinkedIn
NETFLIX
PostgreSQL
android
RAILS
Qt
GNOME
eclipse

git

Outros sistemas



Sistema Git


Checkins Over Time
Version 1
Version 2
Version 3
Version 4
Version 5
File A
File B
File C
A1
B
C1
A1
B
C2
A2
B1
C2
A2
B2
C3


git

Instalação

Download for Linux and Unix

It is easiest to install Git on Linux using the preferred package manager of your Linux distribution. If you prefer to build from source, you can find the tarballs on kernel.org.

Debian/Ubuntu

```
# apt-get install git
```

Fedora

```
# yum install git
```
(up to Fedora 21)

```
# dnf install git
```
(Fedora 22 and later)

Gentoo

```
# emerge --ask --verbose dev-vcs/git
```

Arch Linux

```
# pacman -S git
```

openSUSE

```
# zypper install git
```

Principais comandos

Git essencial

Configuração

Git tem uma ferramenta chamada ***git config*** que permite a você ler e definir variáveis de configuração que controlam todos os aspectos do Git.

Configuração

Essas variáveis podem ser armazenadas em três lugares diferentes:

```
/etc/gitconfig (--system)
~/.gitconfig (--global)
.git/config (--local)
```

Configuração inicial (identidade)

Configurando o nome do usuário:

```
git config --global user.name "NOME"
```

Definindo o e-mail:

```
git config --global user.email "email@email.com"
```

Configuração inicial

Seu editor:

```
git config --global core.editor vim
```

Sua ferramenta de diff:

```
git config --global merge.tool vimdiff
```

Configuração inicial

Verificando suas configurações:

```
git config --list
```

```
git config user.name
```

```
git config user.email
```

Inicializando um novo repositório

```
git init
```

```
[douglas@inspiron GitHub]$ mkdir oficina-git
[douglas@inspiron GitHub]$ cd oficina-git/
[douglas@inspiron oficina-git]$ git init
Initialized empty Git repository in /home/douglas/GitHub/oficina-git/.git/
[douglas@inspiron oficina-git]$
```

Clonando um repositório

```
git clone /caminho/para/o/repositório
```

Exemplos:

```
git clone https://github.com/torvalds/linux.git
```

```
git clone https://github.com/torvalds/linux.git pasta
```

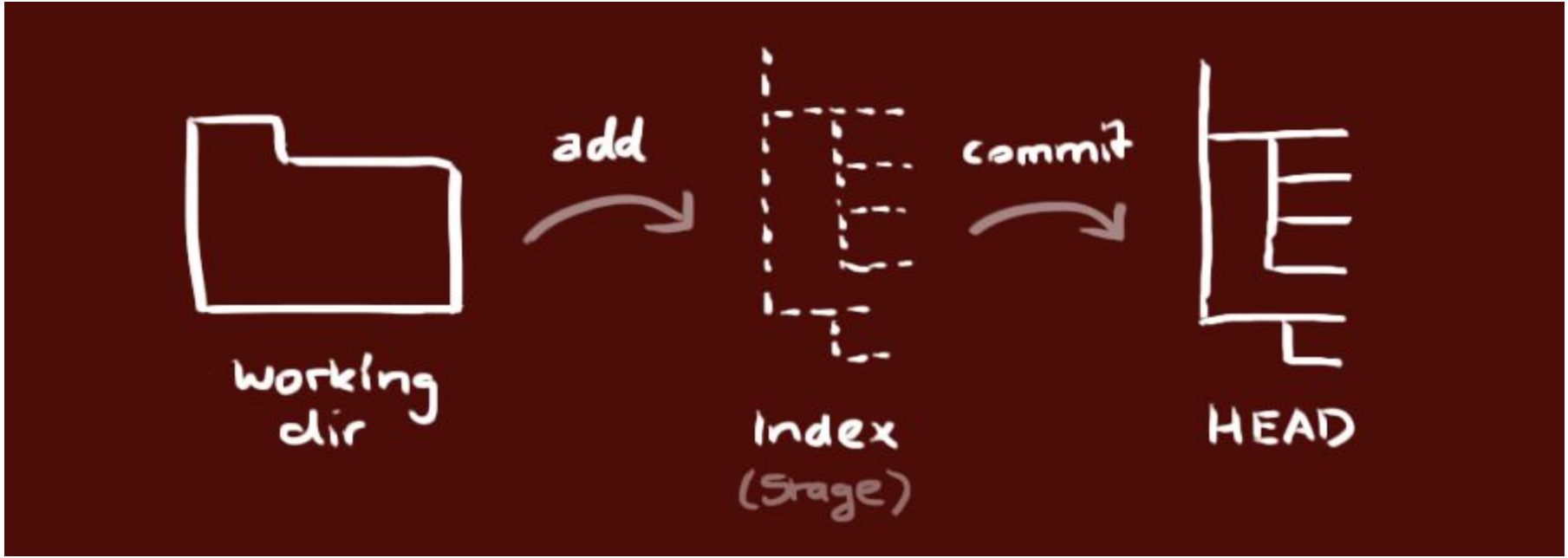
Adicionando

```
git add <arquivo>
```

```
git add *
```

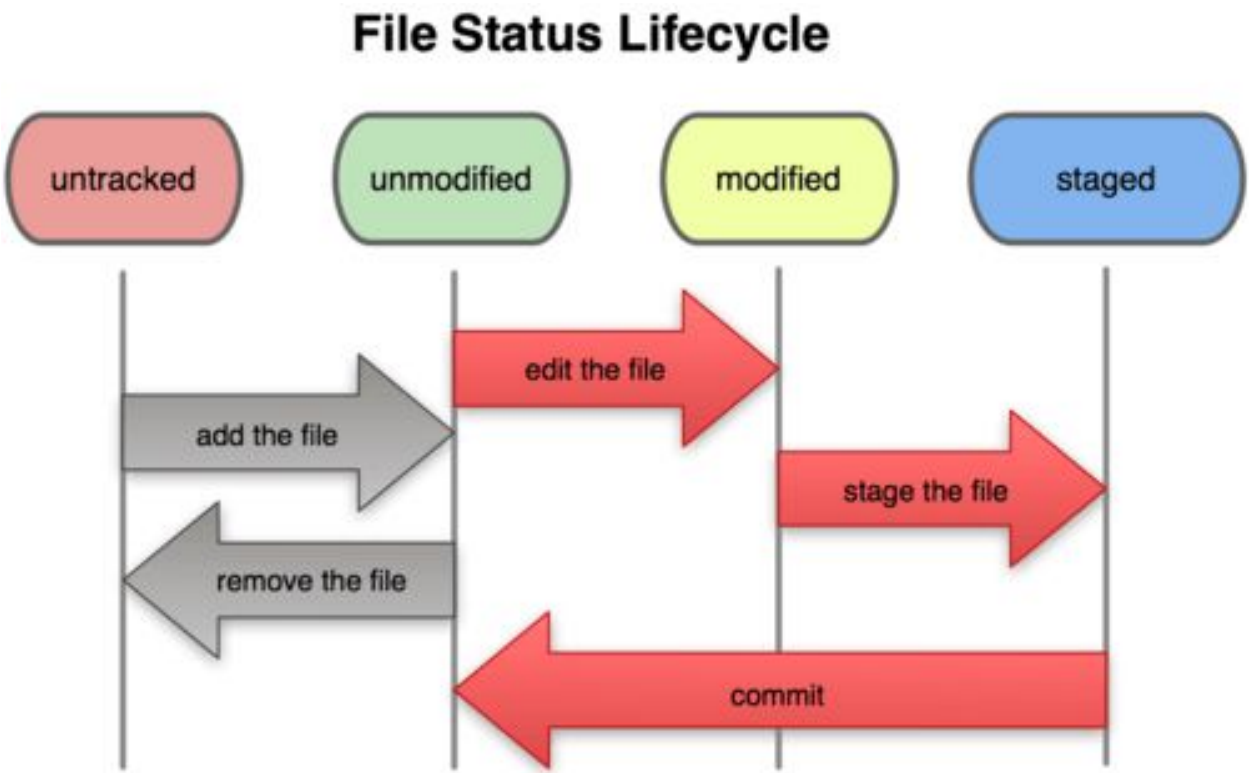
```
git add *.c
```

git



git

Ciclo de vida dos arquivos



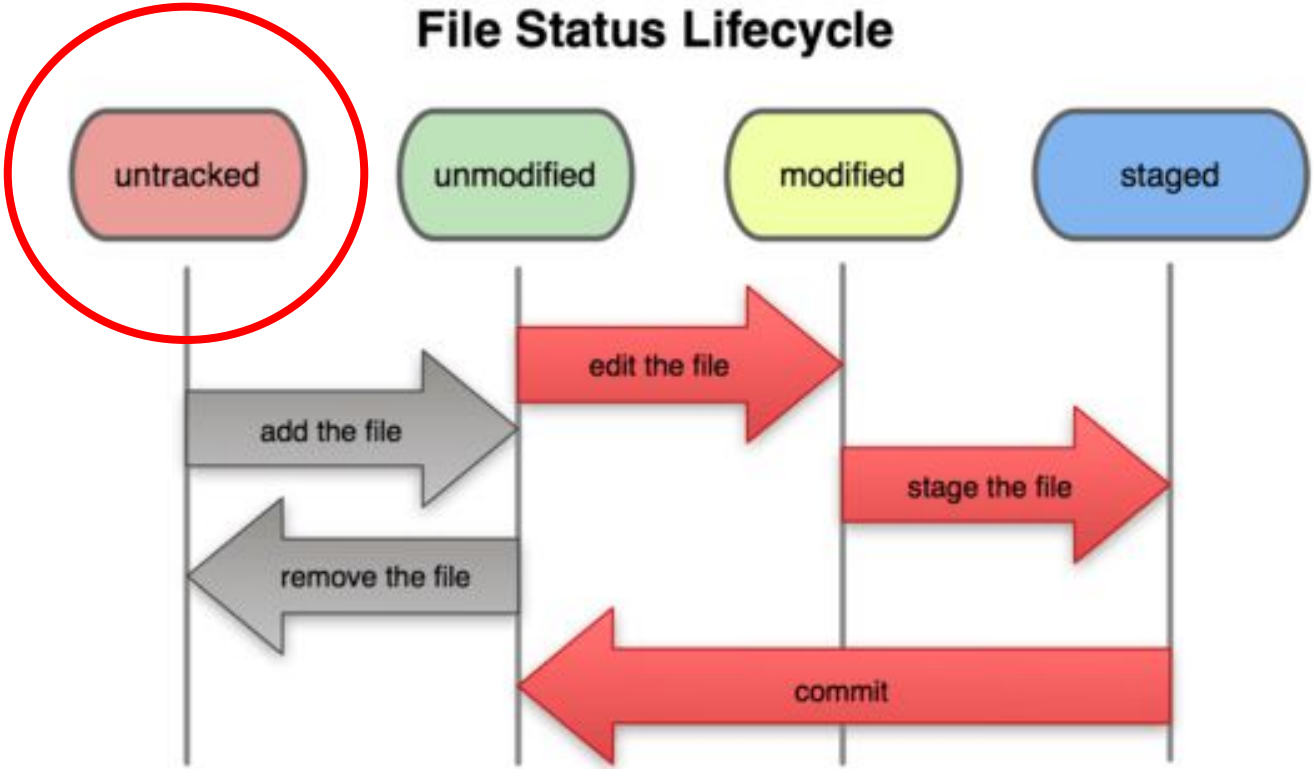
Ciclo de vida dos arquivos

Verificando o estado atual dos arquivos:

```
git status
```

```
[douglas@inspiron oficina-git]$ touch READ.txt
[douglas@inspiron oficina-git]$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        READ.txt

nothing added to commit but untracked files present (use "git add" to track)
[douglas@inspiron oficina-git]$
```

Ciclo de vida dos arquivos



git

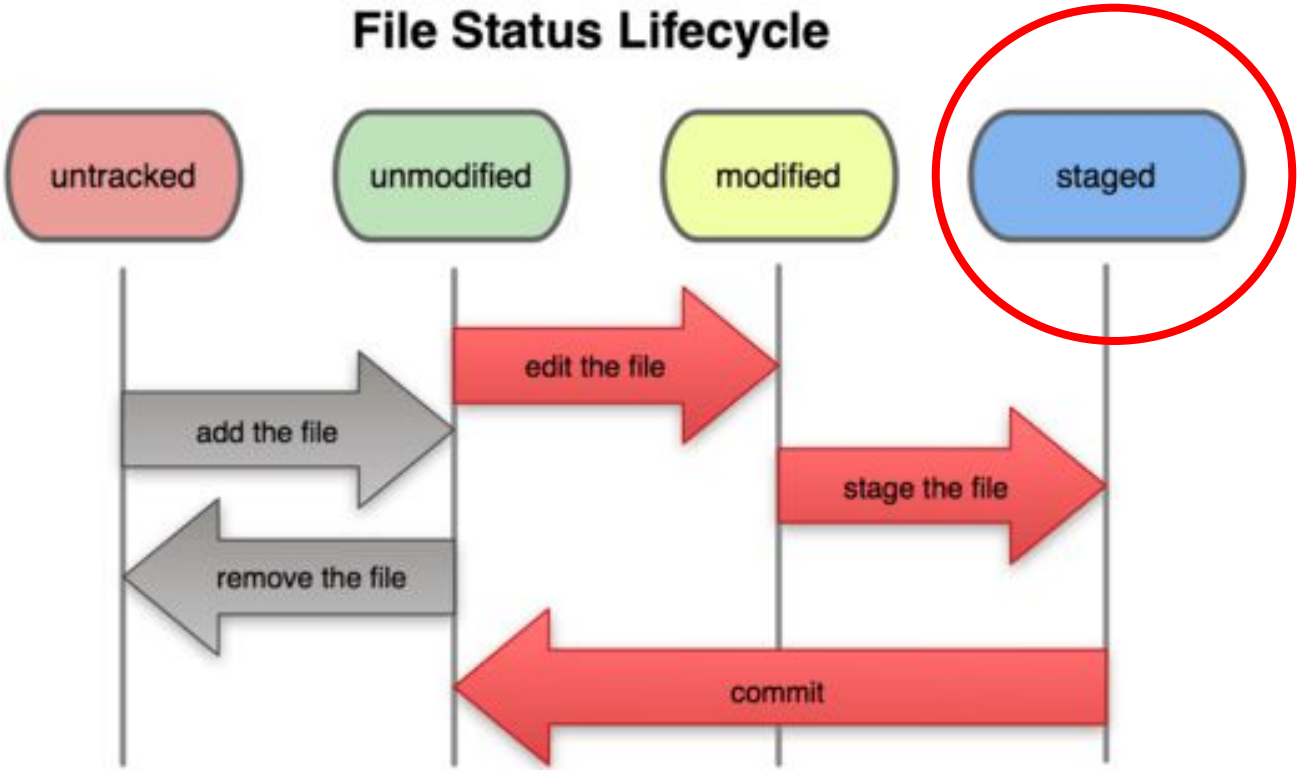
Ciclo de vida dos arquivos

Verificando o estado atual dos arquivos:

```
git status
```

```
[douglas@inspiron oficina-git]$ git add READ.txt
[douglas@inspiron oficina-git]$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   READ.txt

[douglas@inspiron oficina-git]$
```

Ciclo de vida dos arquivos



git

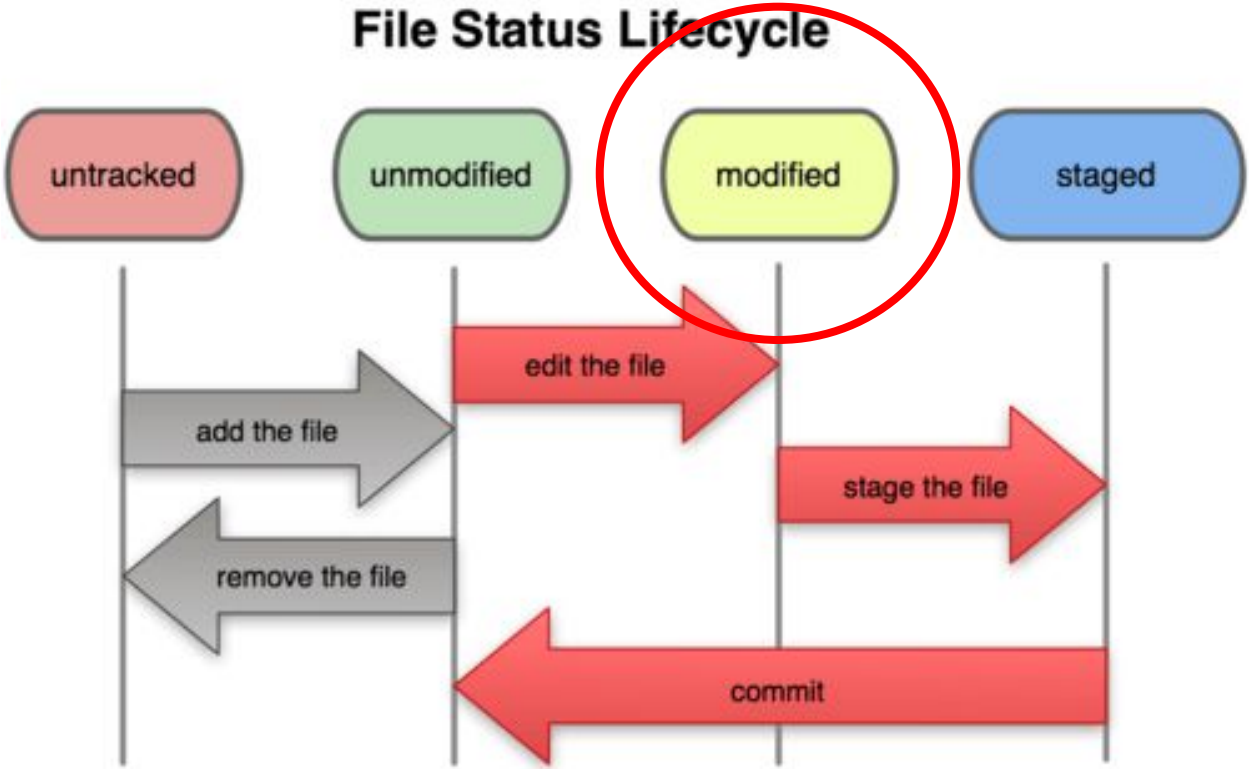
Ciclo de vida dos arquivos

Verificando o estado atual dos arquivos:

`git status`

```
[douglas@inspiron oficina-git]$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   READ.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   READ.txt

[douglas@inspiron oficina-git]$
```

Ciclo de vida dos arquivos

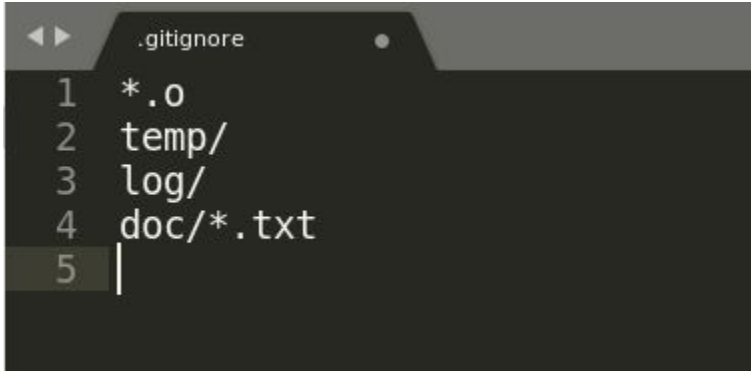


Ignorando arquivos

Muitas vezes, você terá uma classe de arquivos que não quer que o Git automaticamente adicione ou mostre como arquivos não monitorados.

Ignorando arquivos (exemplo)

Neste caso, podemos criar um arquivo contendo uma lista de padrões a serem checados chamado *“**.gitignore**”*



Fazendo commit das mudanças

```
git commit
```

```
git commit -m "MENSAGEM"
```

git

Visualizando o histórico

Depois que você tiver criado vários *commits*, ou se clonou um repositório com um histórico de commits existente, você provavelmente vai querer ver o que aconteceu.

```
git log
```

Visualizando o histórico

Exemplos:

```
git shortlog
```

```
git log --decorate
```

```
git shortlog -sn
```

```
git log --stat
```

```
git show <hash commit>
```

```
git log --author <BUSCA>
```

Desfazendo coisas

É bem possível que em algum momento você queira desfazer algo.

Antes do *“**git add**”*

```
git checkout --README.txt
```

Depois do *“**git add**”* (*staged*)

```
git reset HEAD README.txt
```

Desfazendo coisas

```
git reset [--soft | --mixed | --hard]
```

soft - Mata o *commit* porém os arquivos modificados retornam para o estado de *staged*, prontos para serem comitados novamente.

Desfazendo coisas

```
git reset [--soft | --mixed | --hard]
```

mixed - Mata o *commit* porém os arquivos modificados retornam para o estado de *modified*.

Desfazendo coisas

```
git reset [--soft | --mixed | --hard]
```

hard - Mata brutalmente o *commit*.

Exemplo:

```
git reset --hard 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
```

Repositórios remotos

Exibindo:

```
git remote
```

```
git remote -v
```

Inspecionando:

```
git remote show [nome-remoto]
```

Repositórios remotos

Adicionando um remoto:

```
git remote add [nome-remoto] [url]
```

Renomeando:

```
git remote rename [nome-atual] [novo-nome]
```

Removendo:

```
git remote rm [nome-remoto]
```

Repositórios remotos

Para pegar dados dos seus projetos remotos:

```
git fetch [nome-remoto]
```

```
git pull
```

Para enviar os dados para a fonte:

```
git push [nome-remoto] [branch]
```

Assuntos não abordados

- *Branch* (ramificação);
- *Merge;*
- *Rebase;*
- *Tags;*
- Git remoto com chave SSH.

Referências

- **git - guia prático**. <http://rogerdudler.github.io/git-guide/>
- **Git Tutorial - Try Git**. <https://try.github.io>
- **Pro Git**. <https://git-scm.com/book/pt-br/v1>
- **Git e Github para iniciantes**.

  <https://www.udemy.com/git-e-github-para-iniciantes/>