Introdução ao git

Miguel Pinho

Janeiro 2020

Git?

Sistema de Controlo de Versões

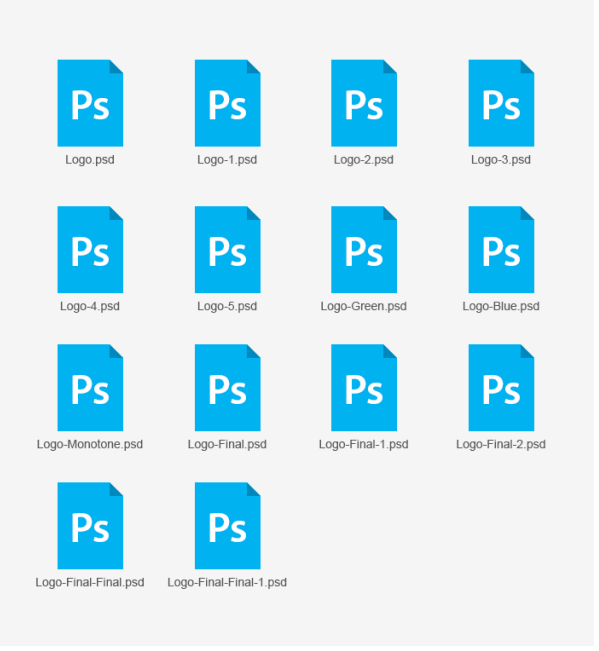
- Regista alterações dos ficheiros
- Permite recuperar o estado em momentos específicos
- Funciona com quase qualquer tipo de ficheiros (código, .pds, .pdf, .tex ...)

[1] https://git-scm.com/

Git?

Sistema de Controlo de Versões

Já devem ter tido uma pasta assim...

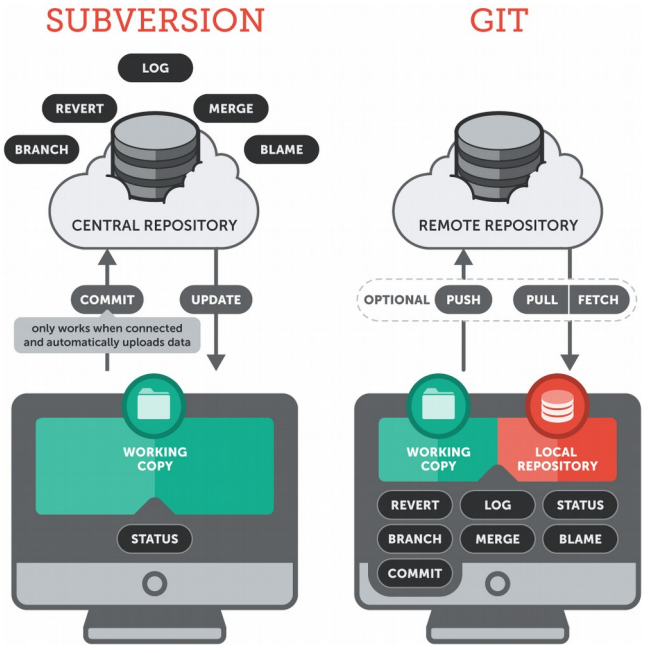


[2] https://uxdesign.cc/version-control-system-for-ui-designers-3022ae9c4753

Git?

Sistema Controlo de Versões Distribuído

- Podem existir várias cópias do projecto, em vários computadores (não é centralizado)
- Pode também funcionar apenas localmente
- Cada cópia guarda a história completa do repositório
- Permite sincronizar alterações entre nós



[3] https://scriptcrunch.com/295/

$_Terminal

1. Configurar user e email
2. Configurar cor no terminal

Comandos

```
$ git config --global user.name “Dave Null”
$ git config --global user.email dev@null.io
$ git config --global color.ui true
```

$_Terminal

1. Criar uma nova pasta e adicionar alguns ficheiros
2. Inicializar repositório

Comandos

```
$ git init
$ git status
```

$_Terminal

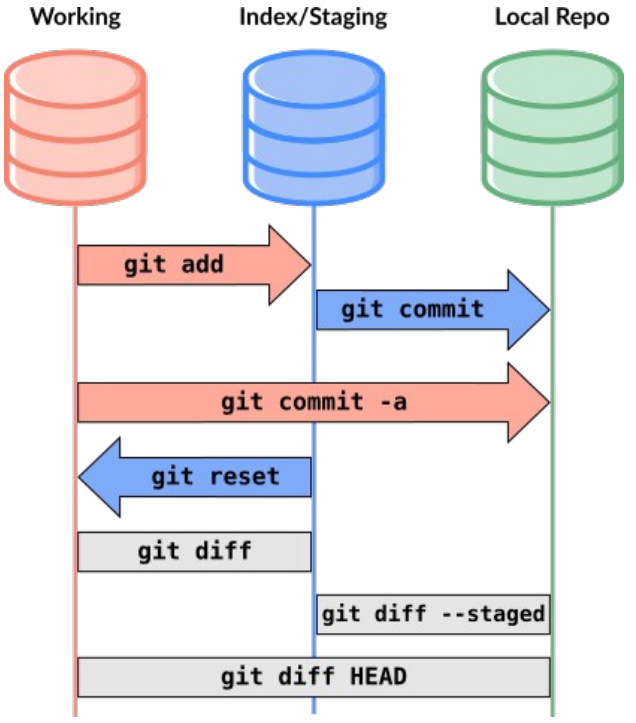
1. Registar as alterações (staging)
2. Adicionar as alterações à história (commit)

Comandos

```
$ git add -A
$ git commit
$ git log
```

Zonas do Repositório

- **working directory:** ficheiros que podem ser alterados diretamente
- **index/staging area:** zona intermédia onde está a ser preparado o próximo commit (.git)
- **local repo:** base de dados local, com toda a história do repositório



$_Terminal

1. Fazer mais alterações
2. Registar e reverter alterações
3. Adicionar apenas uma alteração à história

Comandos

```
$ git diff
$ git add [file]
$ git reset [file]
$ git diff --staged
$ git commit
```

$_Terminal

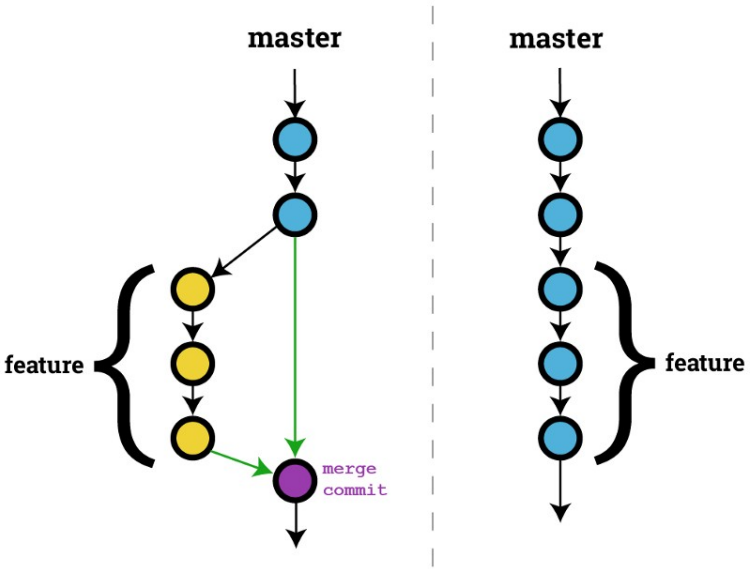
1. Reverter para um ponto anterior da história
2. Voltar ao ponto atual

Comandos

```
$ git log
$ git checkout HEAD~1
$ git checkout master
```

Branching

- O Git é permitir que existam vários estados do repositório em simultâneo: **branches**
- Exemplo: permite trabalhar e testar uma nova funcionalidade sem perturbar o ramo principal de desenvolvimento da aplicação
- As alterações efetuadas num ramo podem ser aplicadas noutros ramos: **merging**



[4] https://medium.com/@haydar_ai/learning-how-to-git-merging-branches-and-resolving-conflict-61652834d4b0

$_Terminal

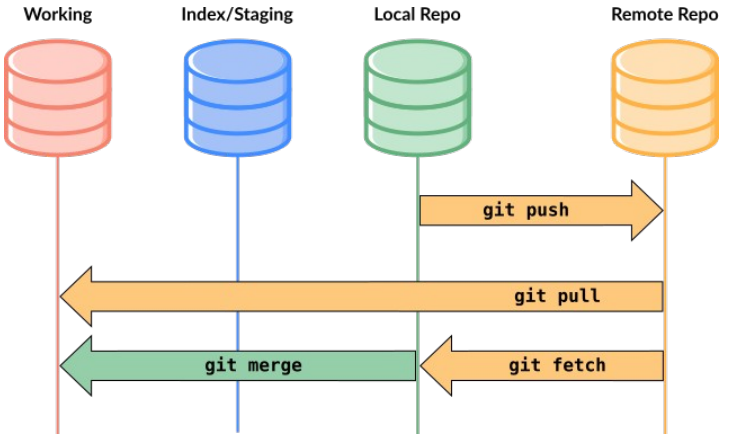
1. Criar uma ramificação (*branch)*
2. Avançar na história desse ramo
3. Regressar ao ramo original
4. Aplicar as alterações do ramificação no ramo original
5. Apagar a ramificação

Comandos

```
$ git checkout -b [ramo]
$ git branch
$ git checkout master
$ git show-branch
$ git merge [ramo]
$ git branch -d [ramo]
```

Repositórios Remotos

- Embora o git possa funcionar apenas localmente, grande parte do seu potencial está no trabalho colaborativo
- O estado de um repositório local pode ser sincronizado com um (ou mais) repositórios remotos
- Sites com o GitHub, o GitLab e o BitBucket permitem alojar facilmente repositórios na cloud



GitHub

GitLab

Bitbucket

$_Terminal

1. Criar um repositório vazio no GitHub
2. Guardar o estado do repositório local no repositório remoto

Comandos

```
$ git remote add origin [link]
$ git push -u origin master
$ git push origin --all
```

$_Terminal

1. Encontrar um repositório no GitHub (ou similar)
2. Clonar e explorar o repositório

Comandos

```
$ git clone [link]
```

Links

- **Resolver conflitos de merge:**
  - https://medium.com/@haydar_ai/learning-how-to-git-merging-branches-and-resolving-conflict-61652834d4b0
  - https://www.youtube.com/watch?v=1MVQYSlgXrI
- **Configurar chaves ssh:**
  - https://liyanxu.blog/2017/02/12/install-git-on-windows-and-set-up-ssh-keys/
  - https://help.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh
- **Tutorial de branching:**
  - https://learngitbranching.js.org/

FIM!

https://github.com/miguelpinho/git-lecture