Doce Git

↳ por Gabriel G. de Brito

= Universidade Federal do Paraná =

Git é um sistema de versionamento distribuído criado para as necessidades do kernel Linux. Atualmente é o padrão de facto para versionamento de código. Estima-se que seja utilizado por mais de 90% dos desenvolvedores [Stack Overflow, 2022].

O git é organizado por meio de diffs. Ou seja,
o histórico é guardado se guardando as alterações
feitas em dado momento. As alterações estão
organizadas em um grafo semelhante à uma merkle tree.

`git status`

```
* 80eb24d78ccb433cb67f4f2816951c804ff134ce (tag: v2.3.0) Pumped version to 2.3.0
*   6d254514772ee0056ad7b8f9863bcde8bff241a5 Merge pull request #18 from gboncoffee/internationalization
|\
| * 1335aac22a7017e667af658553750b73f81992d7 Added internationalization to architectures
* | c6b24a3c4db7487ad6228a4837304a8e9b60a916 Merge pull request #16 from gboncoffee/internationalization
|\|
| *   0d11d8c19ba5993be9fcd147699499aec6c604b2 Merge branch 'master' into internationalization
| |\
| |/
|/|
* |   dd19bb7a33919c967541577e2bddf4363dc9de54 Merge pull request #17 from gboncoffee/go22
|\ \
| * | 45535ea099195dc59317e6e5eb358e3b051c709d Pumped go version to 1.22
|/ /
| * ec2ca2380bb599c74aa04f31de9a3f65194c0c21 Added internationalization support with intergo
|/
* 1c05f30751957f0c70854db55cd535e5a6d7ebb0 Better portuguese cheatsheet for RISC-V
* 456decc596bfd113cbdf9fe0e16921bafc862600 (tag: v2.2.0) Update version to 2.2.0
```

`git log --graph --oneline --all`

Vamos focar no workflow em time com git/GitHub.

A ação mais comum vai ser invariavelmente...

`git add .`

`git commit`

`git push`

Podemos pensar em repositórios remotos como outras branches, da qual fazemos merge e (possivelmente) rebase.

Assim, um workflow comum seria cada desenvolvedor ter um `fork` do repositório principal. Que opera como se fosse uma branch.

Dessa maneira, cada um tem a liberdade de manipular o histórico como quiser antes de mergear no principal. Além disso, podemos usar ferramentas de PR do GitHub.

`git branch <nova branch>`

`git branch -D <branch>`

`git checkout <branch>`

OBS: checkout funciona para commits também!

Como trabalhamos em máquinas locais e mandamos o código para o GitHub, efetivamente cada um tem duas branches: o fork no GitHub e a cópia local.

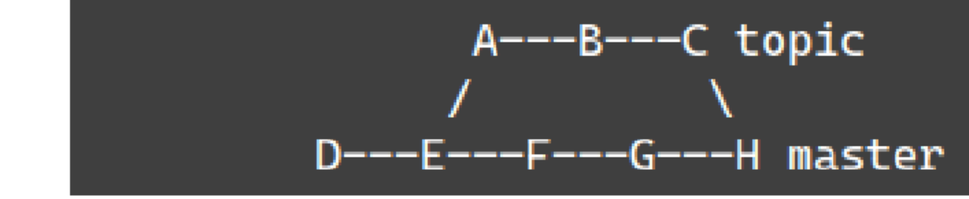
Em comandos que operam em branches remotas podemos usar `--set-upstream` para configurar uma branch padrão.

Ex: `git push --set-upstream origin master` fará com que rodar somente `git push` seja equivalente à `git push origin master`.

Lidando com branches, precisamos entender merging e rebasing. Pull é equivalente a `git fetch` seguido de merge ou rebase.

```
Assume the following history exists and the current branch is "topic":

             A---B---C topic
            /
       D---E---F---G master

From this point, the result of either of the following commands:

    git rebase master
    git rebase master topic

would be:

                     A'--B'--C' topic
                    /
       D---E---F---G master
```

Then "**git merge topic**" will replay the changes made on the **topic** branch since it diverged from **master** (i.e., **E**) until its current commit (**C**) on top of **master**, and record the result in a new commit along with the names of the two parent commits and a log message from the user describing the changes.



Trabalhamos em repositórios remotos via SSH ou HTTP.
SSH geralmente é preferido.

`ssh-keygen -t ed25519 -C "user@mail.com"`

Aceite as opções padrão.

Encontre nas configurações do GitHub a área para adicionar novas chaves. Cole o conteúdo da chave.

Tarefa: gerar chave ssh e configurar no GitHub.

Tarefa: criar fork do repositório "gboncoffee/doce-git".

Tarefa: clonar seu fork, adicionar seu nome e o seu doce preferido, e mandar uma PR para o "upstream".