Git *“under the hood”*

Matheus Tavares

matheustavares.gitlab.io

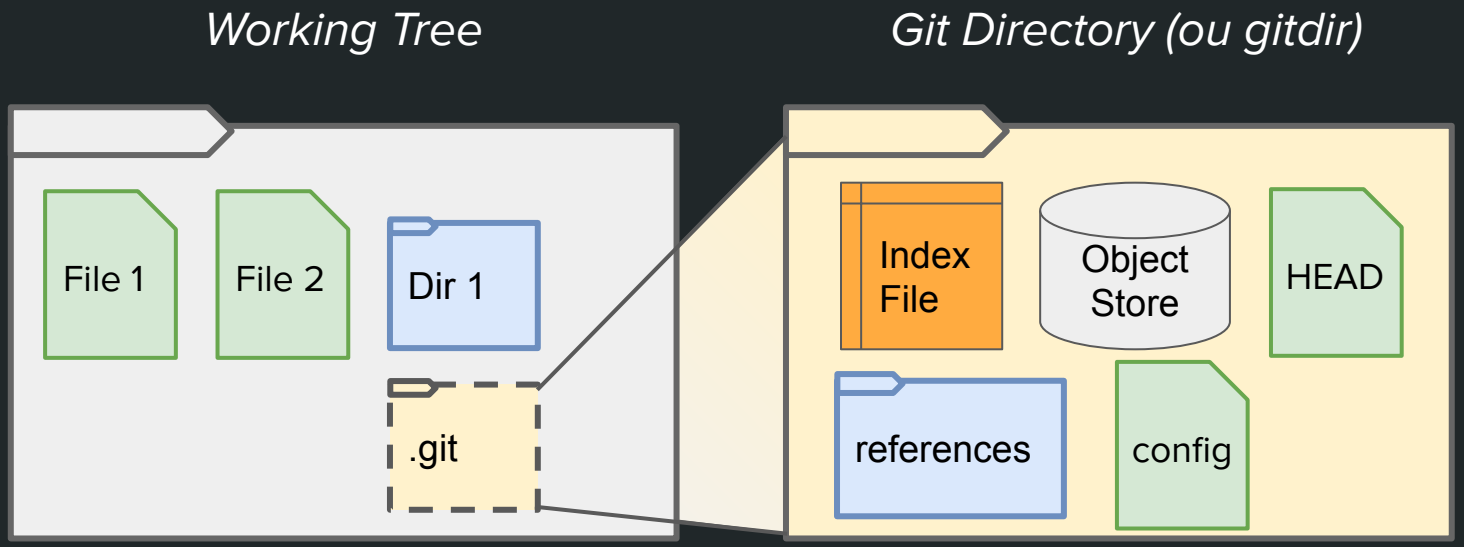
Inspirado em: **The Zen of Git**, de Tianyu Pu:
https://speakerdeck.com/tianyupu/the-zen-of-git

http://www.quickmeme.com/meme/3t6lr9

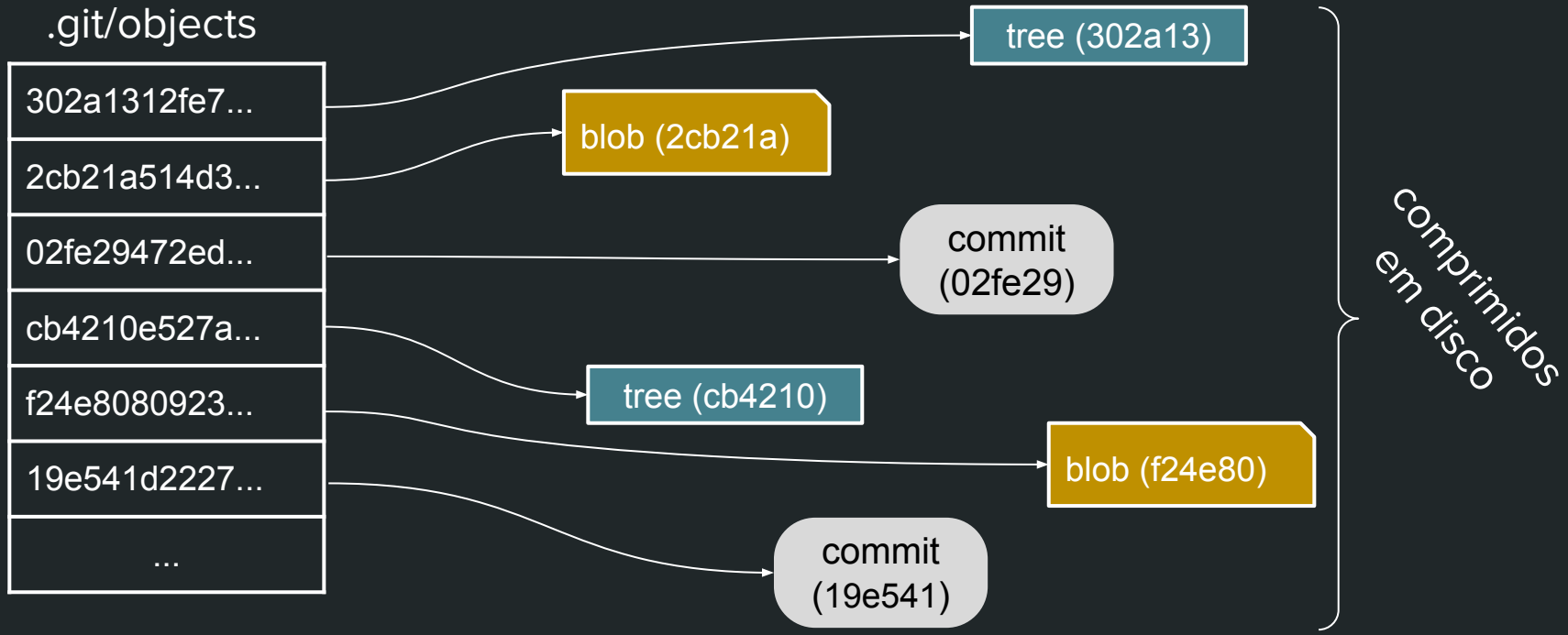
https://xkcd.com/1597/

Um pouco de como o Git funciona

Layout de Repositório

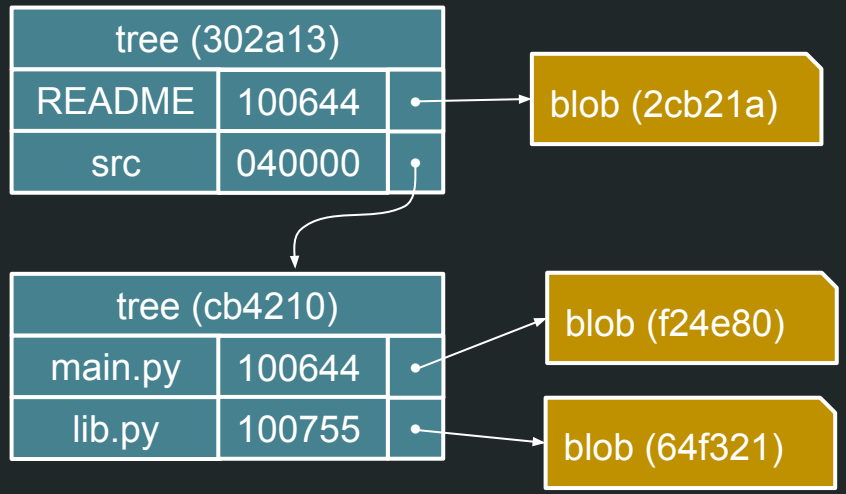


Um HashSet


.git/objects
302a1312fe7...
2cb21a514d3...
02fe29472ed...
cb4210e527a...
f24e8080923...
19e541d2227...
...
tree (302a13)
blob (2cb21a)
commit (02fe29)
tree (cb4210)
blob (f24e80)
commit (19e541)
comprimidos em disco

Relações entre os objetos

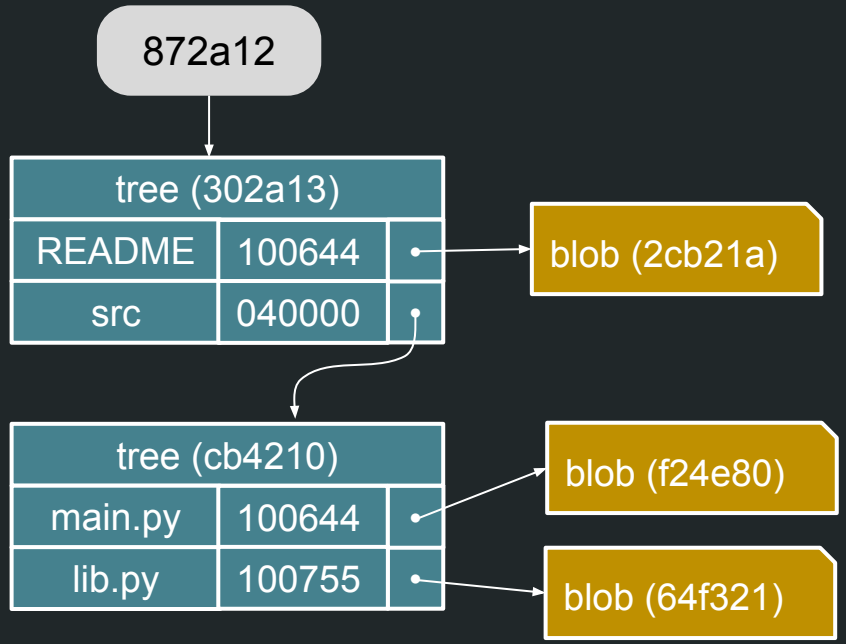
```
$ tree repo
repo
├── README
└── src
    ├── main.py
    └── lib.py
```



Relações entre os objetos

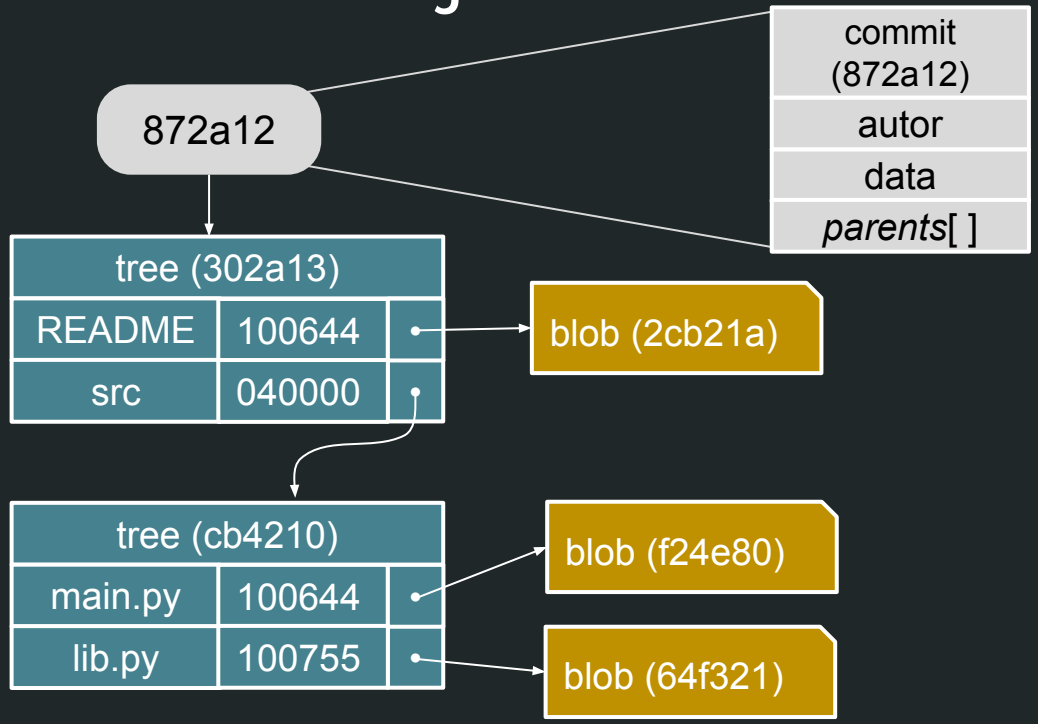
Um *commit* guarda um estado do projeto

(uma *tree*, não um *diff!*)



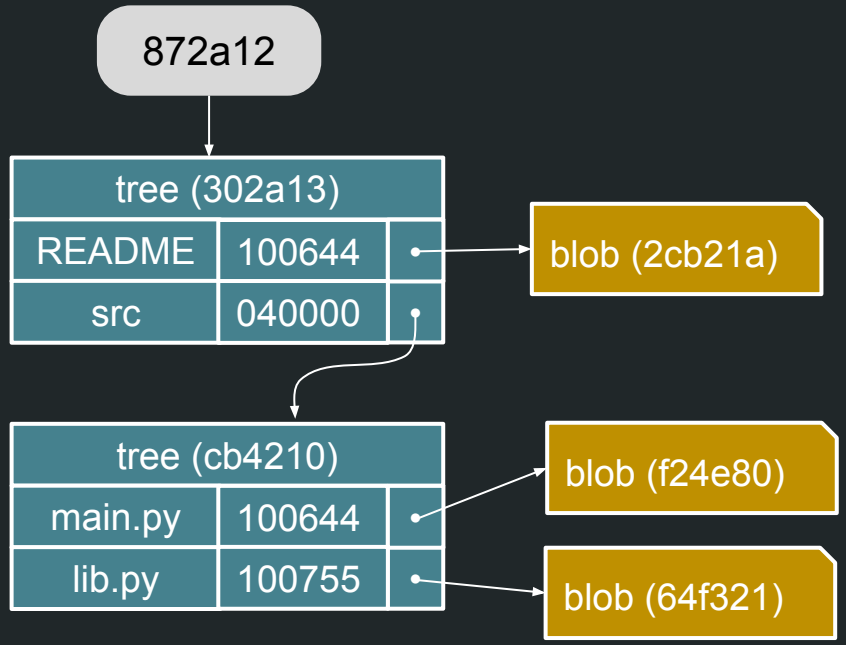
Relações entre os objetos

Commits também contém um conjunto de metadados.



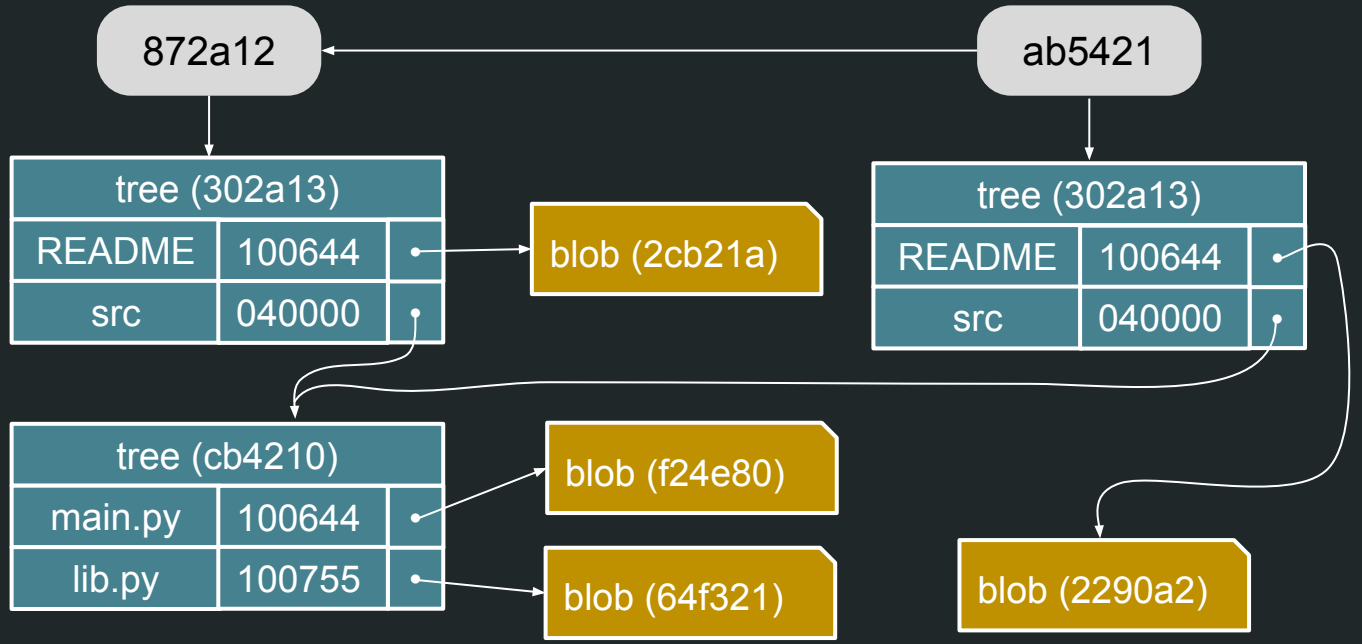
Relações entre os objetos

$ vim README
$ git commit



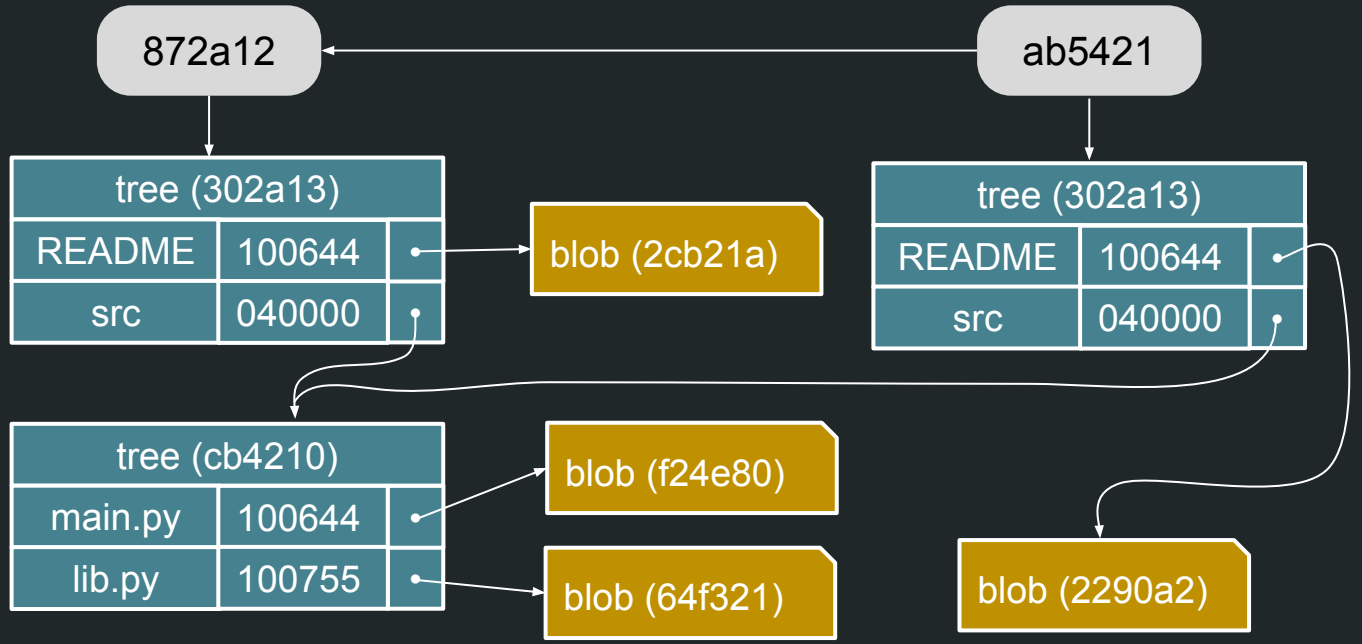
Relações entre os objetos

$ vim README
$ git commit



Grafo Acíclico Dirigido

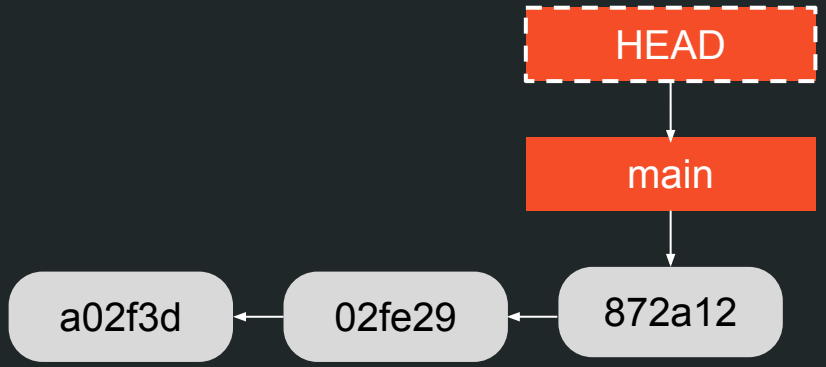
$ vim README
$ git commit



Referências

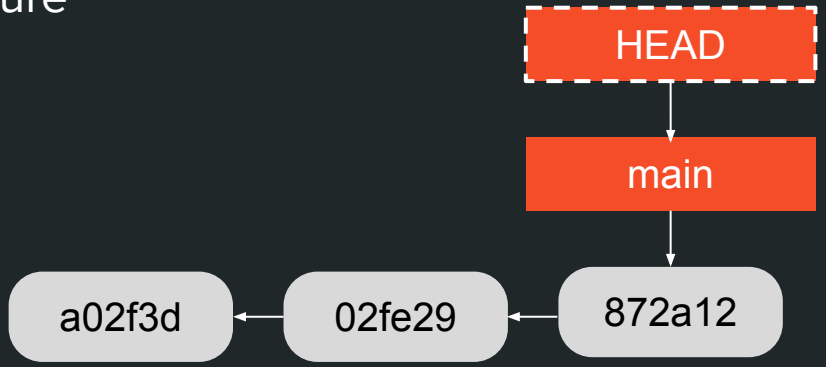
Referências

- branches
- tags
- HEAD



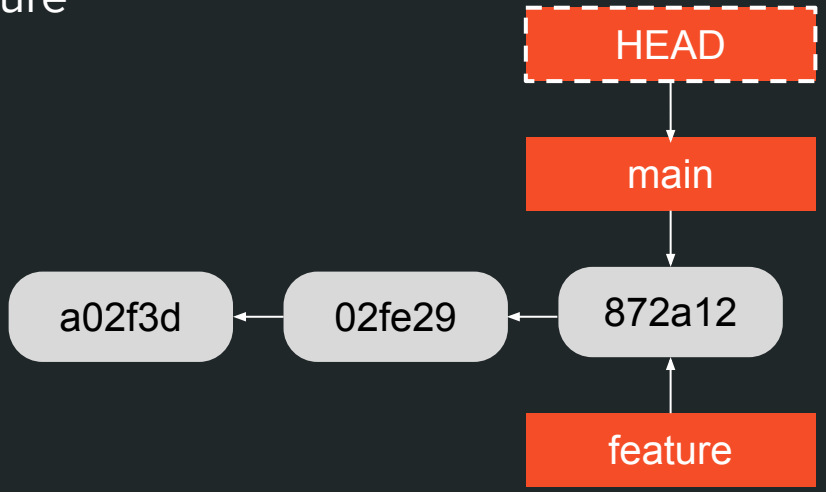
Referências: branches

$ git branch feature



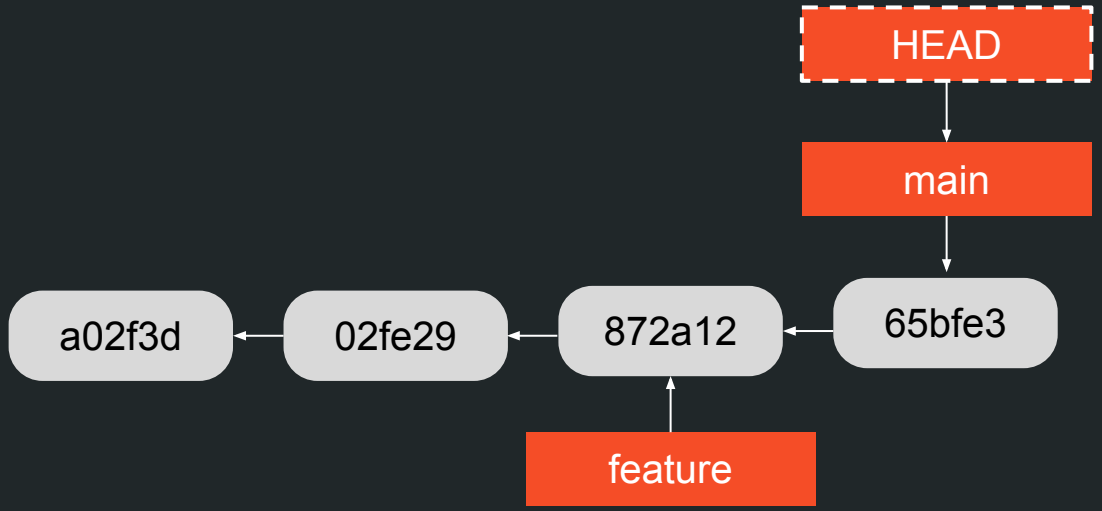
Referências: branches

$ git branch feature



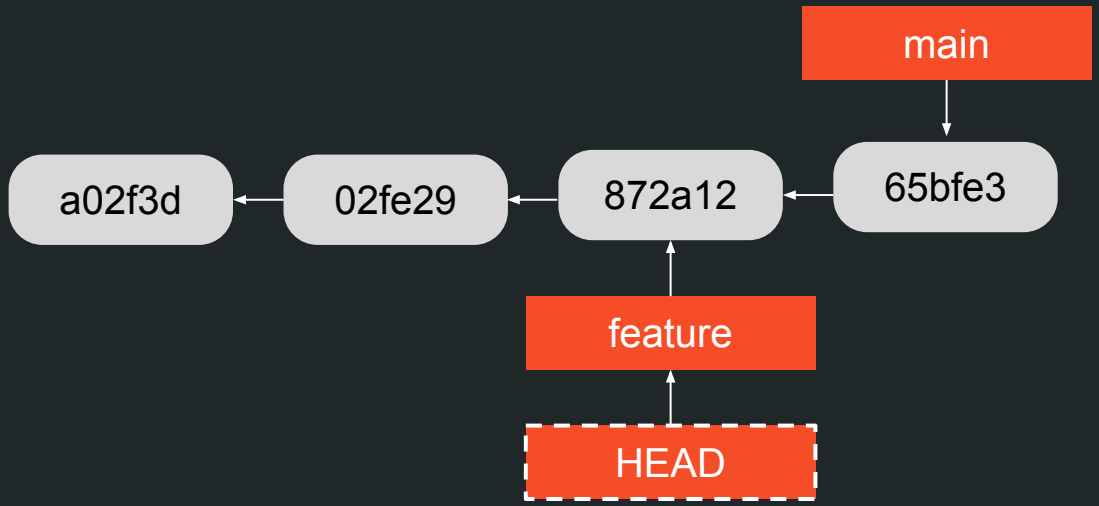
Referências: branches

$ git commit



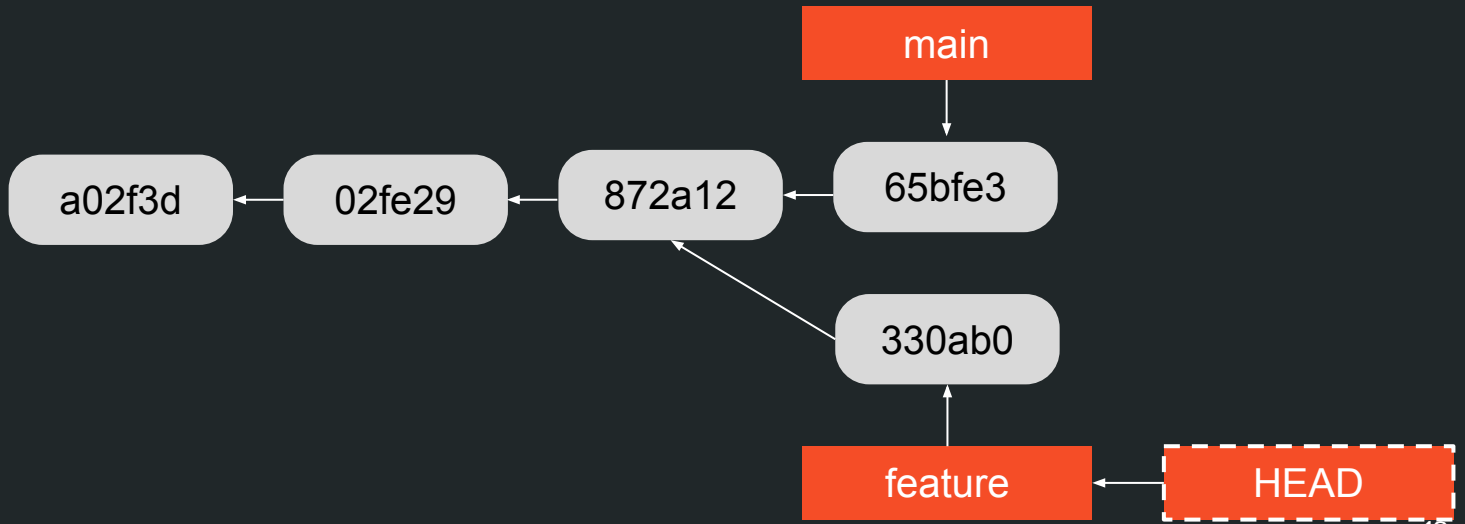
Referências: branches

$ git checkout feature



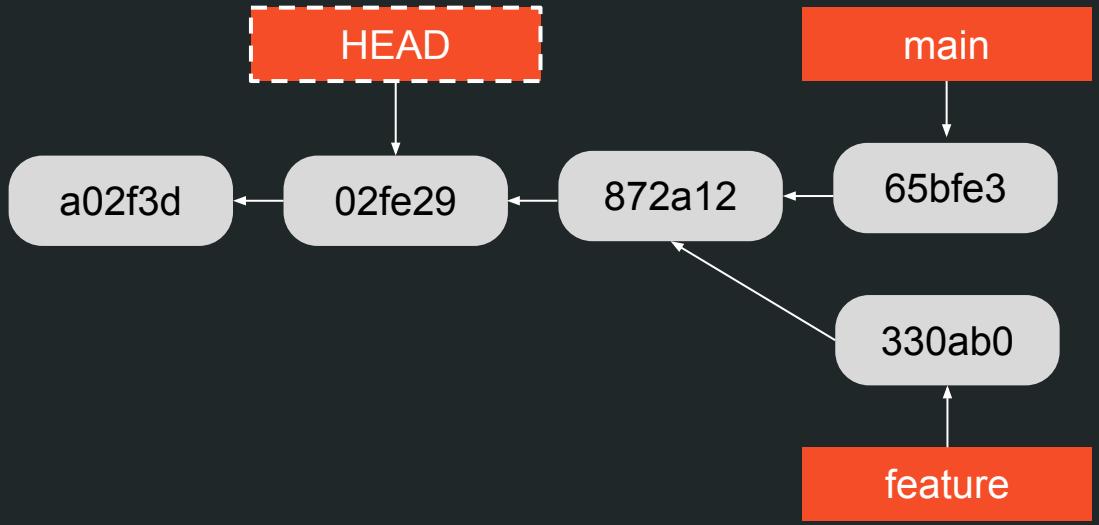
Referências: branches

$ git commit



Referências: branches

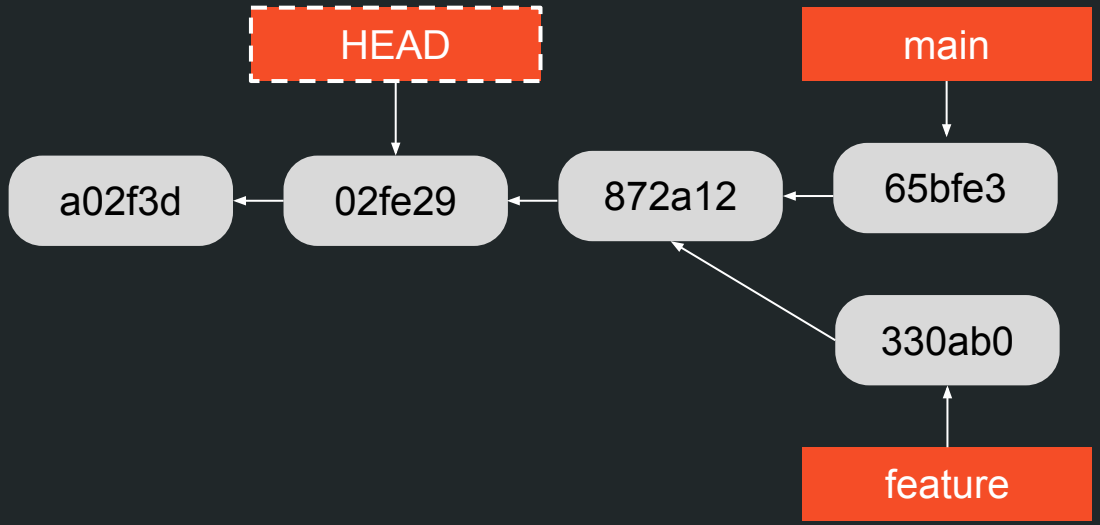
$ git checkout 02fe29



Referências: branches

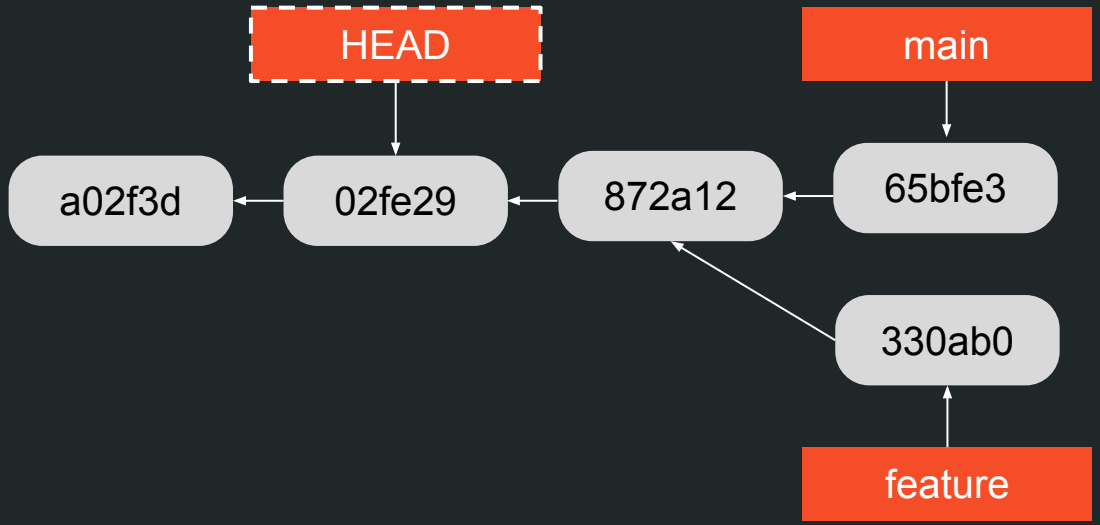
$ git checkout 02fe29

You are in “detached HEAD” state.



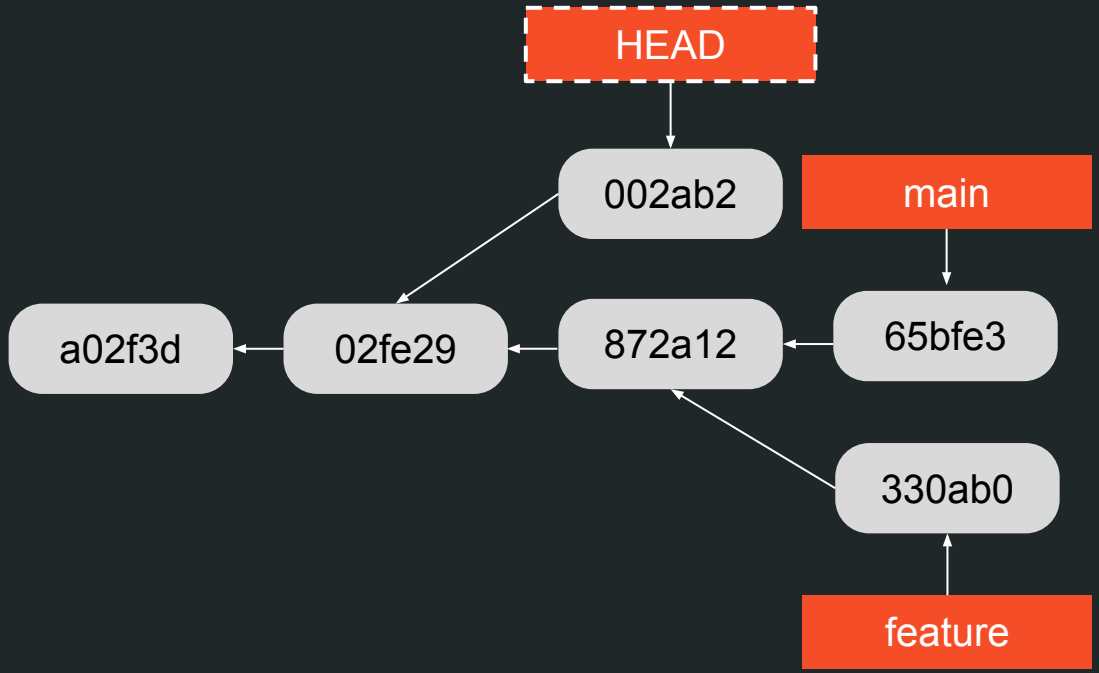
Referências: branches

$ git commit



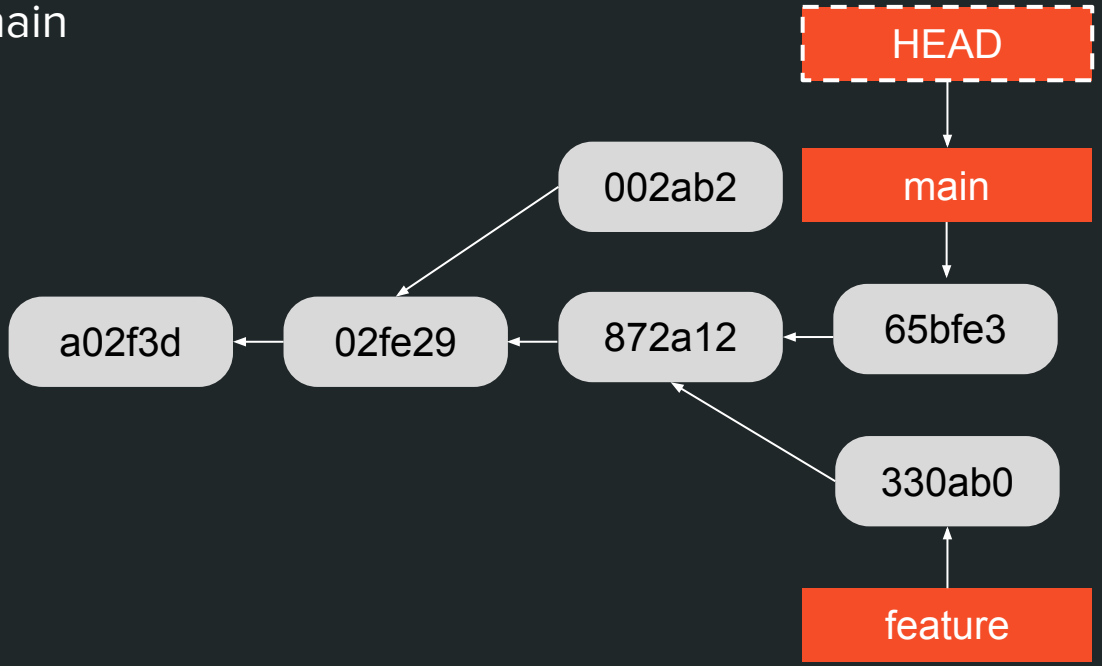
Referências: branches

$ git commit



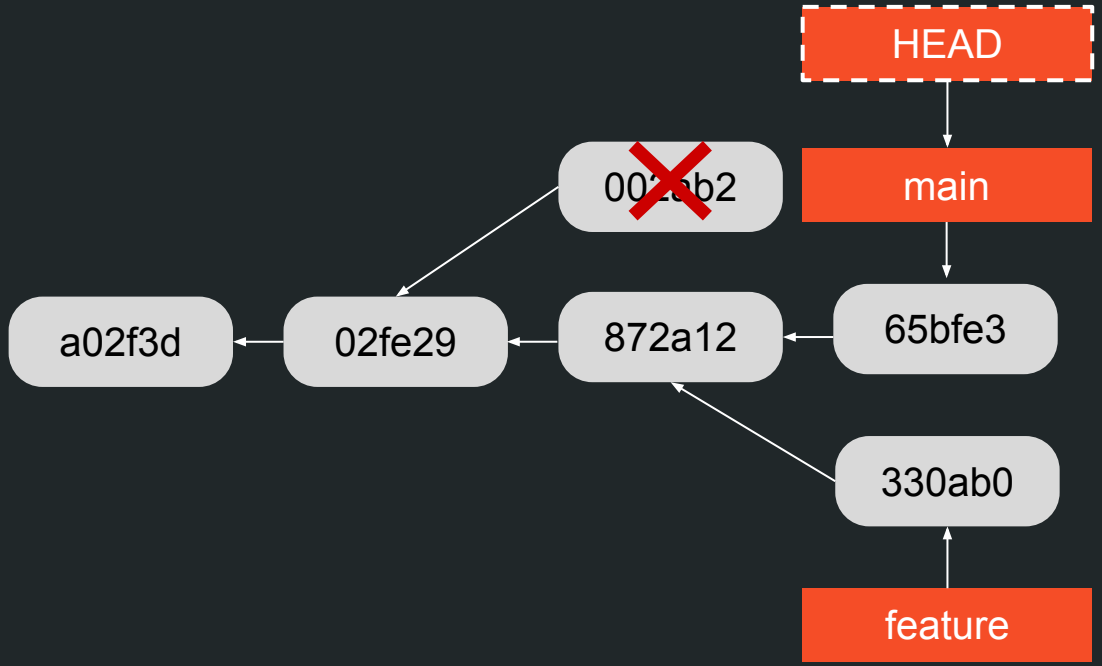
Referências: branches

$ git checkout main



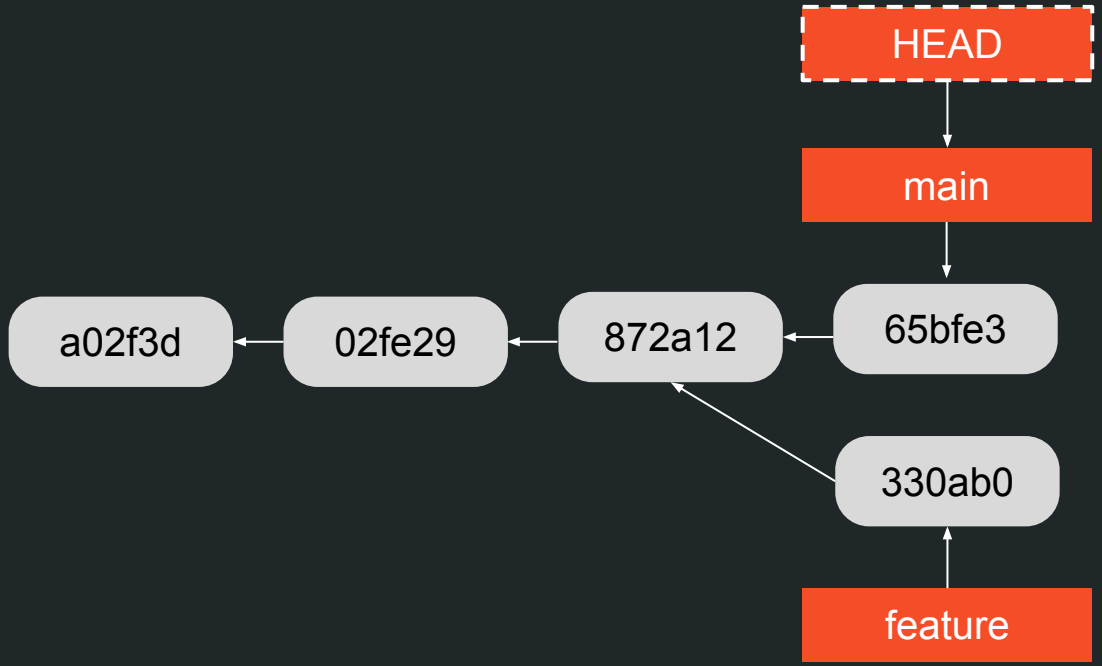
Referências: branches

$ git gc --auto



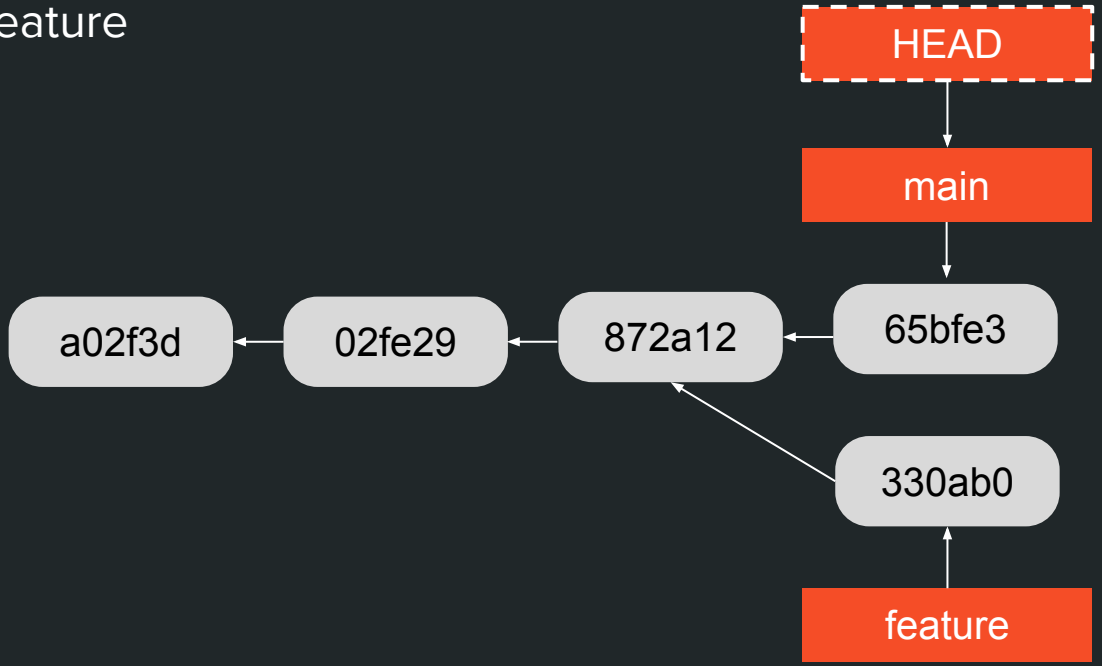
Referências: branches

$ git gc --auto



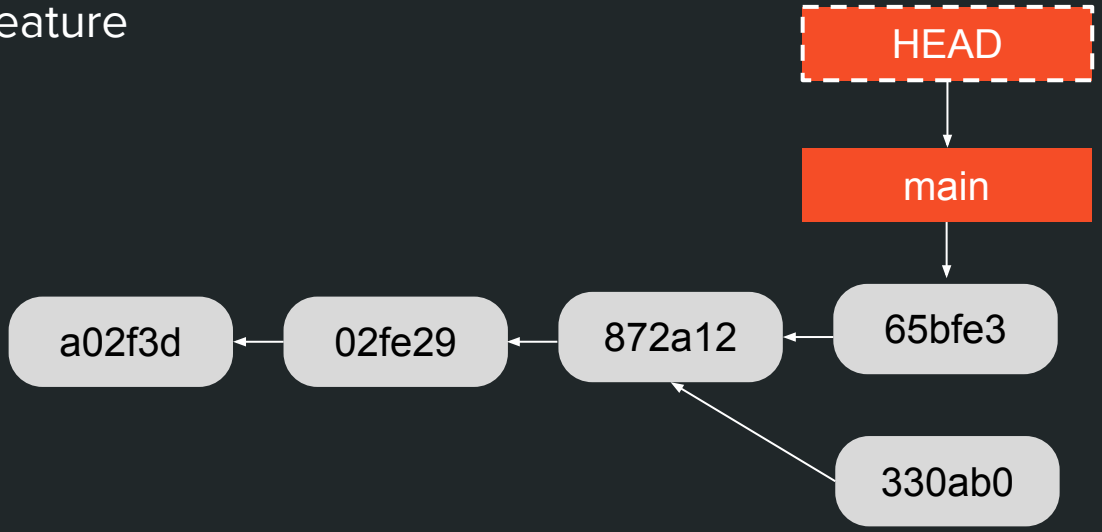
Referências: branches

$ git branch -D feature



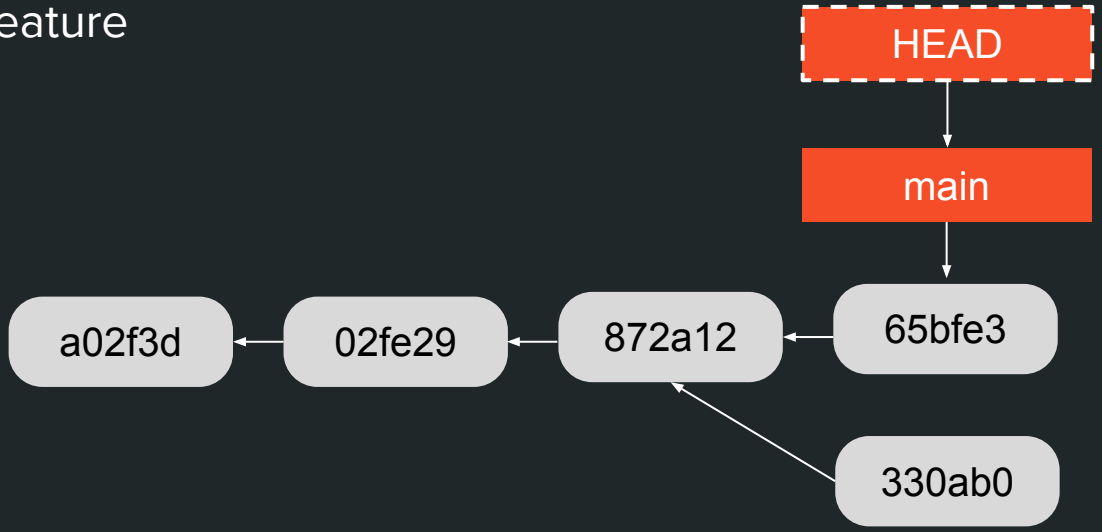
Referências: branches

$ git branch -D feature



Referências: branches

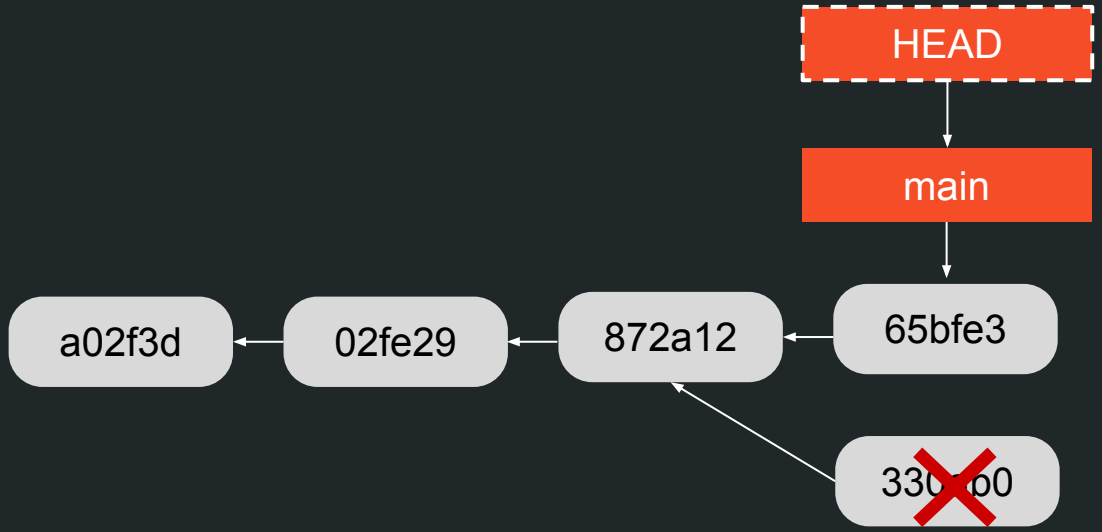
$ git branch -D feature



Ainda é recuperável*: reflog*

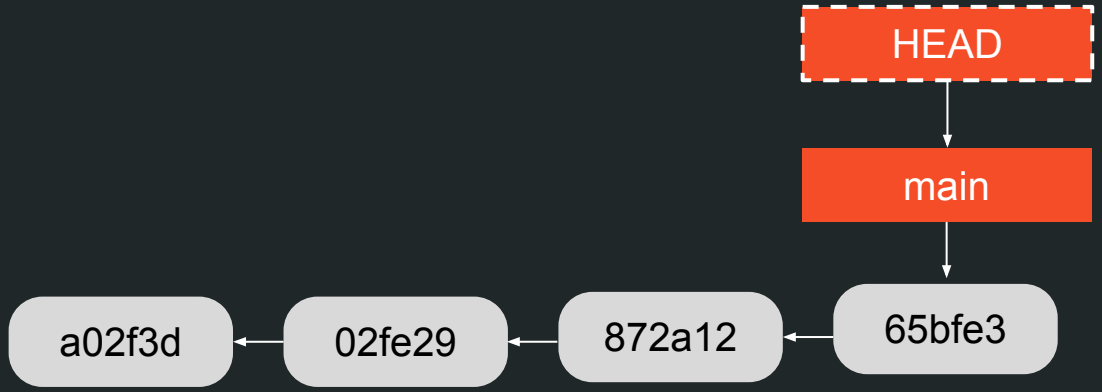
Referências: branches

$ git gc --auto



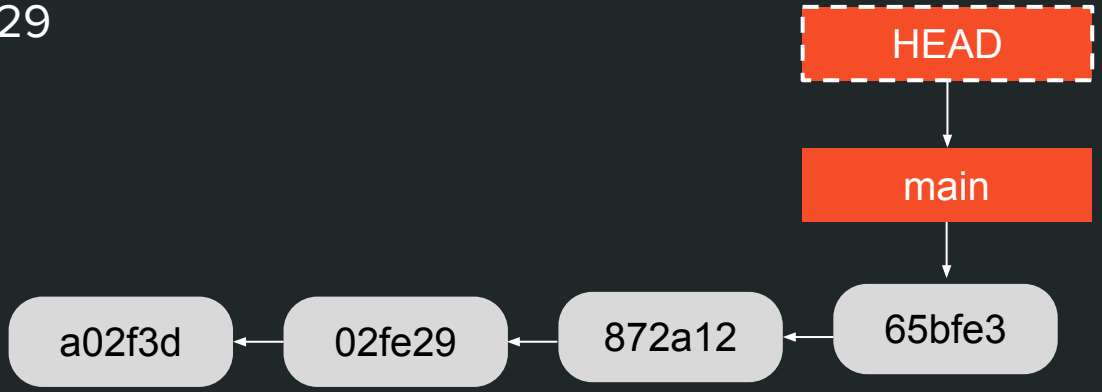
Referências: branches

$ git gc --auto



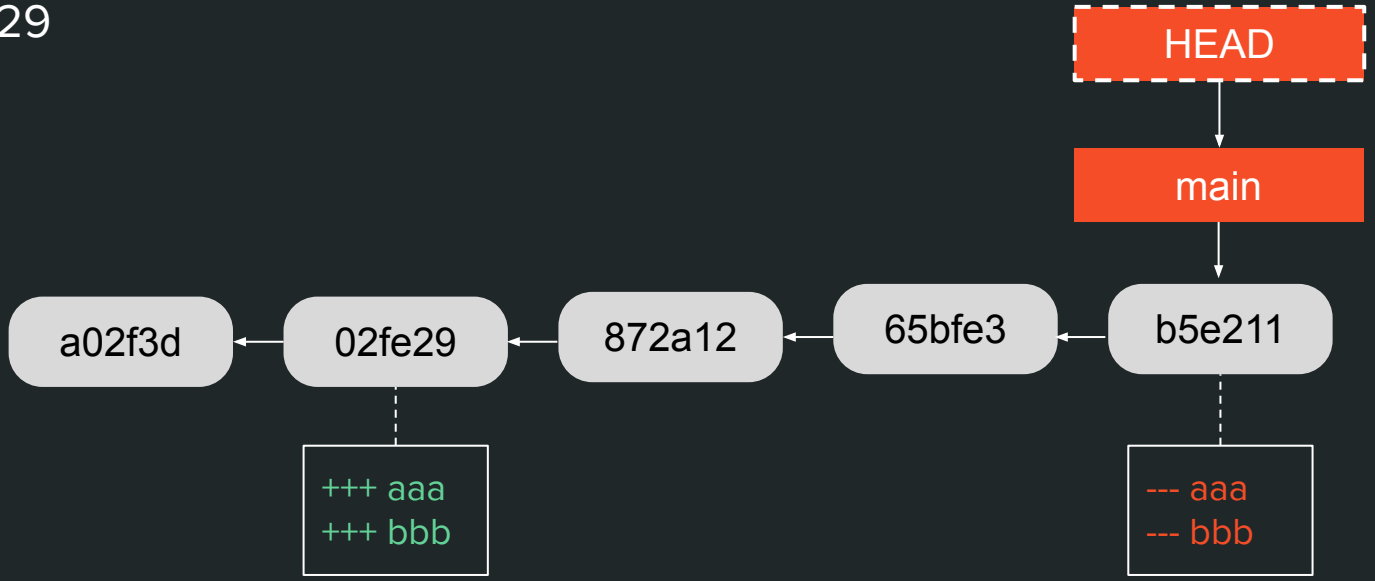
Referências: branches

$ git revert 02fe29



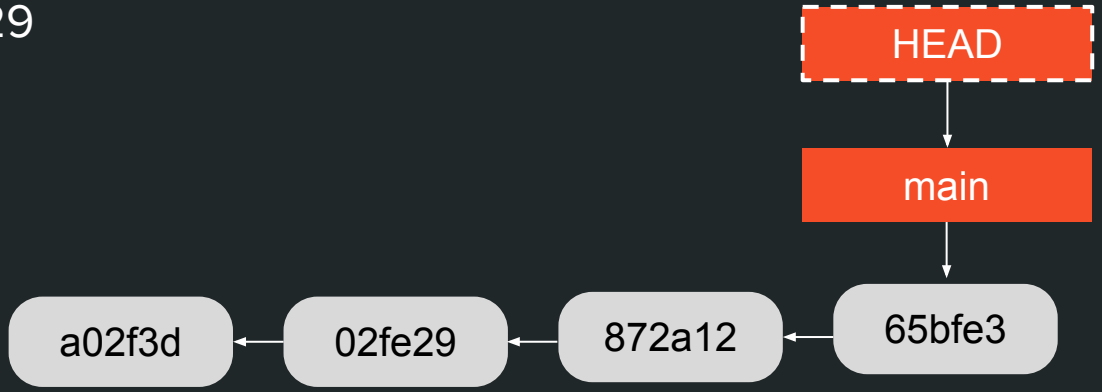
Referências: branches

$ git revert 02fe29



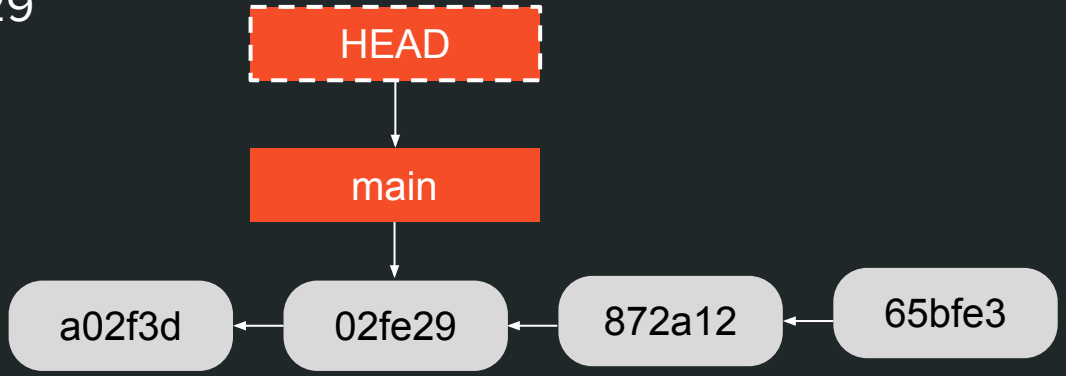
Referências: branches

$ git reset 02fe29



Referências: branches

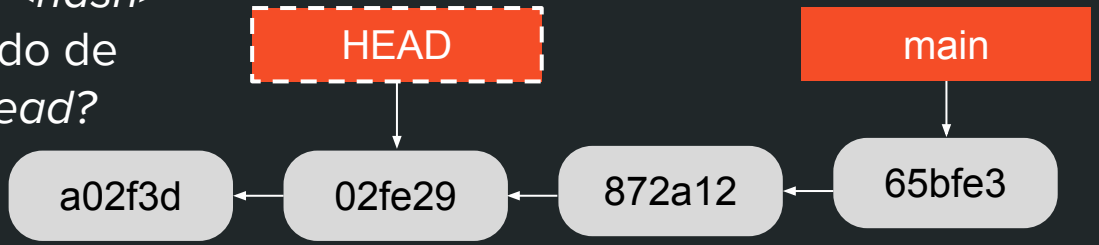
$ git reset 02fe29



Referências: branches

Exercício:

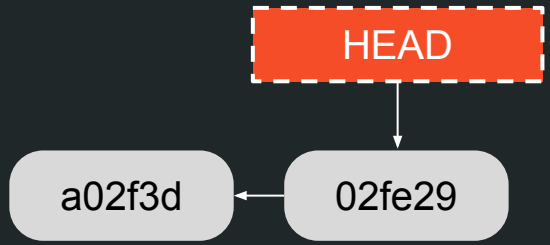
- Qual o resultado de um *git reset <hash>* em um estado de *detached head?*



Referências: branches

Reset:

- --soft: HEAD
- --mixed: HEAD, index
- --hard: HEAD, index, working tree



echo a >a
git add a
git commit

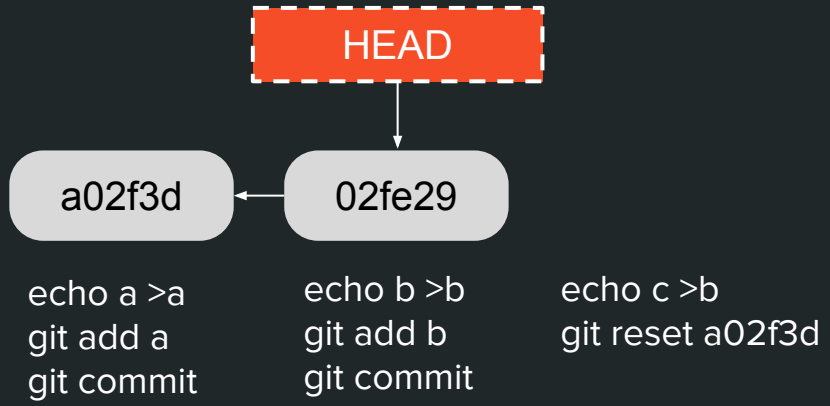
echo b >b
git add b
git commit

echo c >b
git reset a02f3d

Referências: branches

Reset:

- --soft:

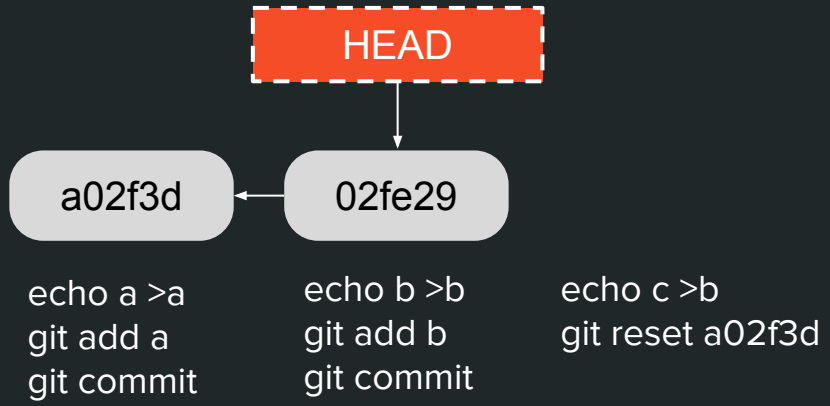


Referências: branches

Reset:

- --soft:

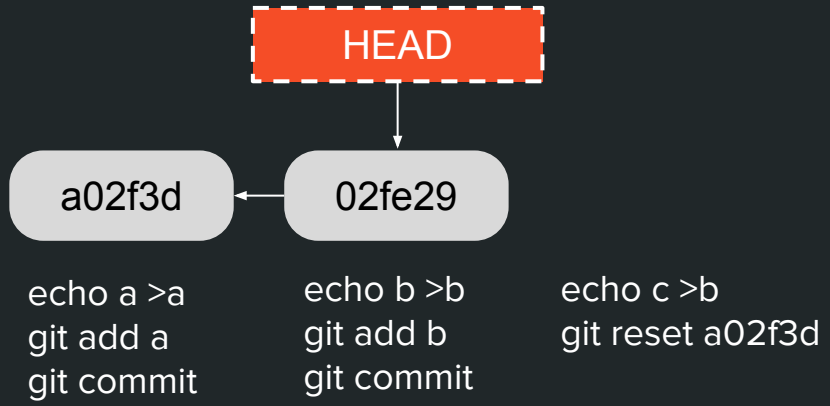
Changes to be committed:
new file: b
Changes not staged for commit:
modified: b



Referências: branches

Reset:

- --mixed: *(padrão)*

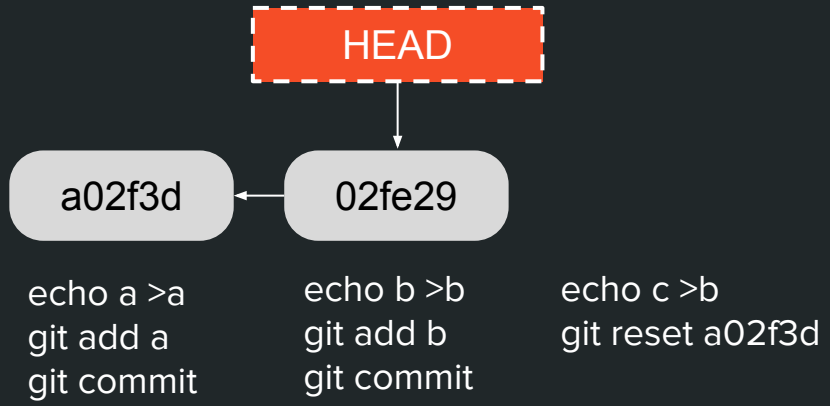


Referências: branches

Reset:

- --mixed: *(padrão)*

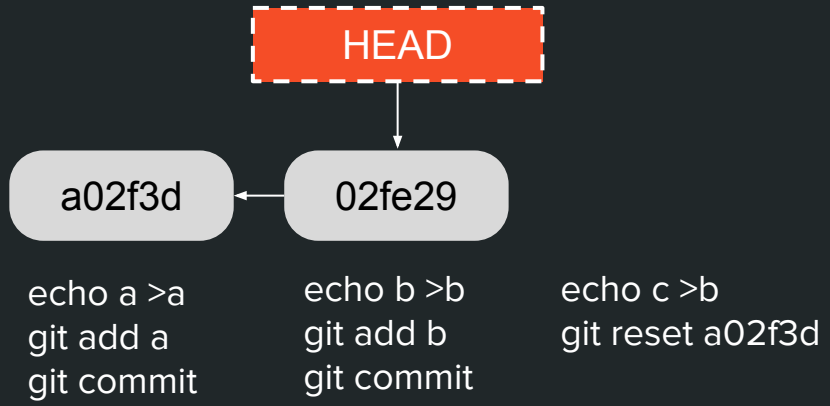
  Untracked files:
  b



Referências: branches

Reset:

- --hard:



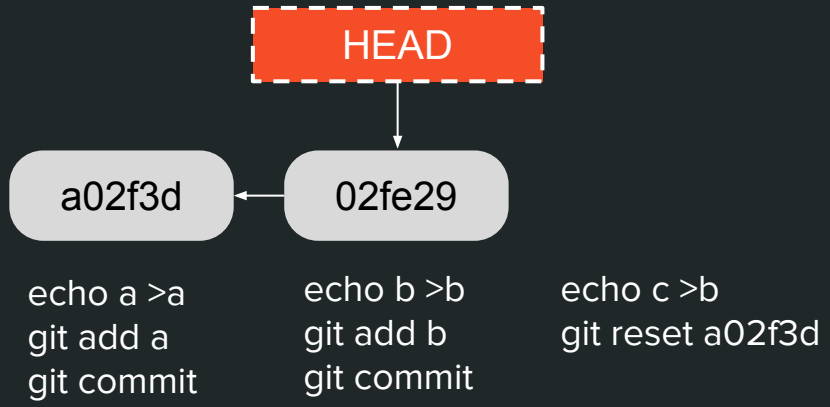
Referências: branches

Reset:

- --hard:

  On branch main
  nothing to commit, working tree clean

  $ ls
  a

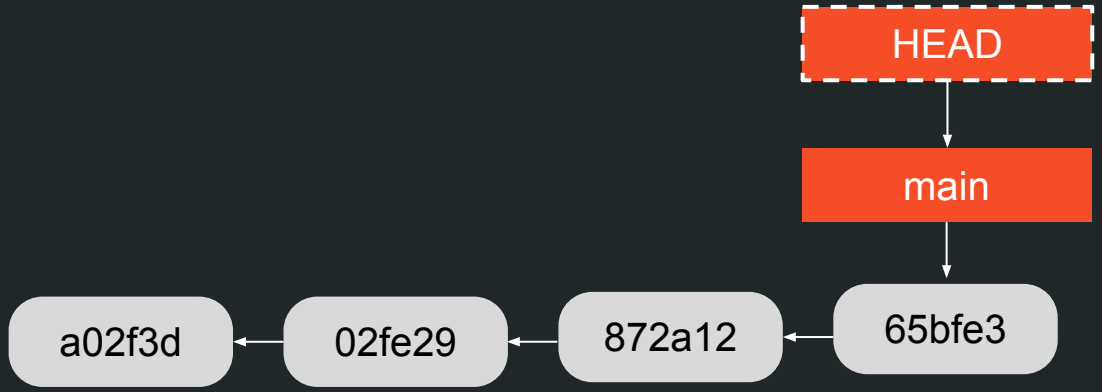


Referências: branches

Cuidado com *reset --hard*:

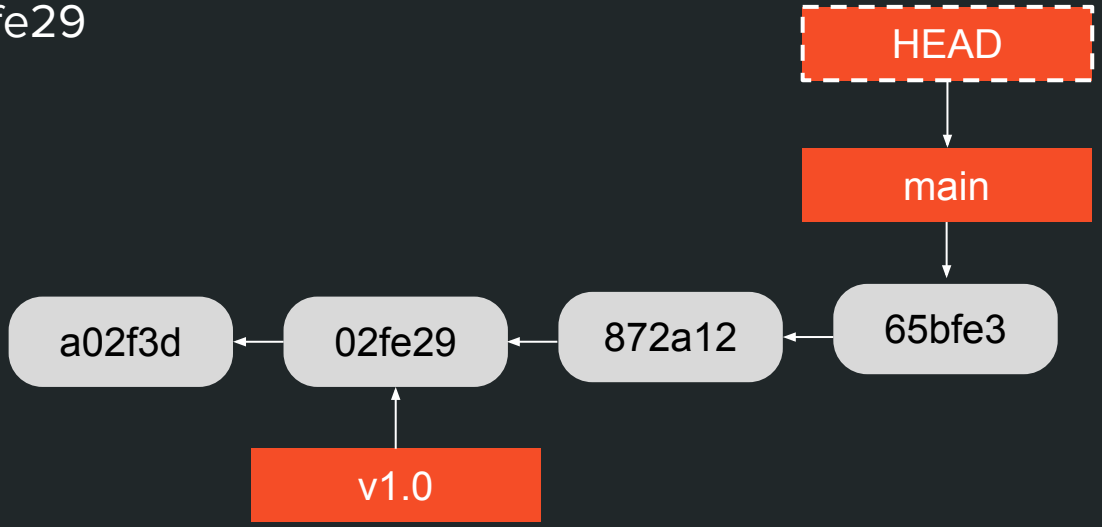
“Any changes to tracked files in the working tree since <commit> are discarded. Any untracked files or directories in the way of writing any tracked files are simply deleted”

Referências: tags



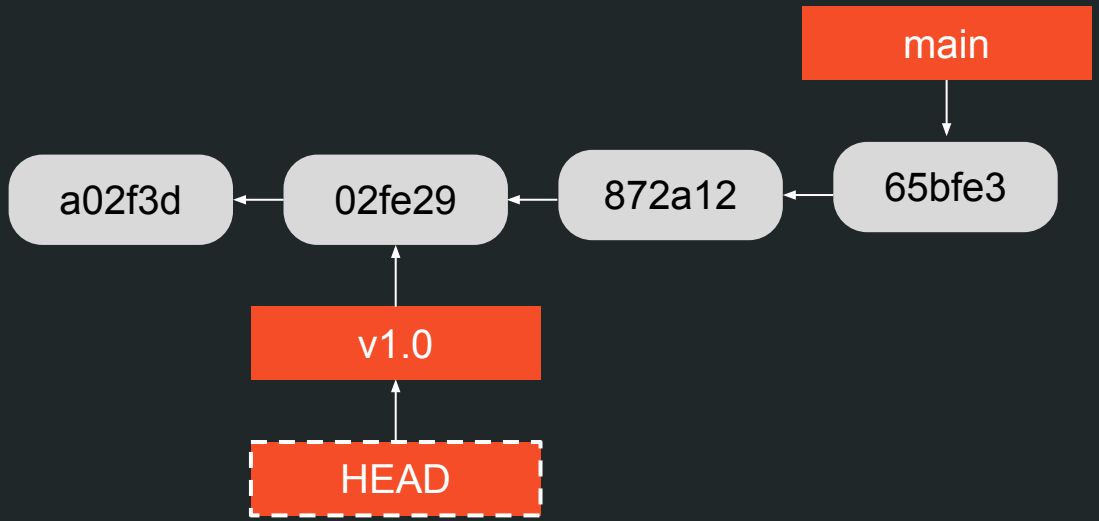
Referências: tags

$ git tag v1.0 02fe29



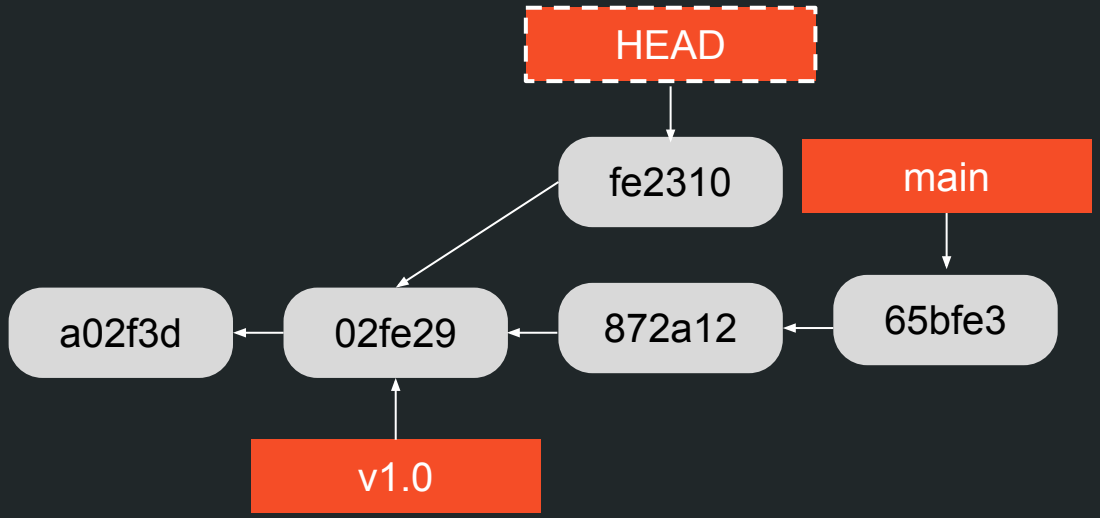
Referências: tags

$ git checkout v1.0



Referências: tags

$ git commit

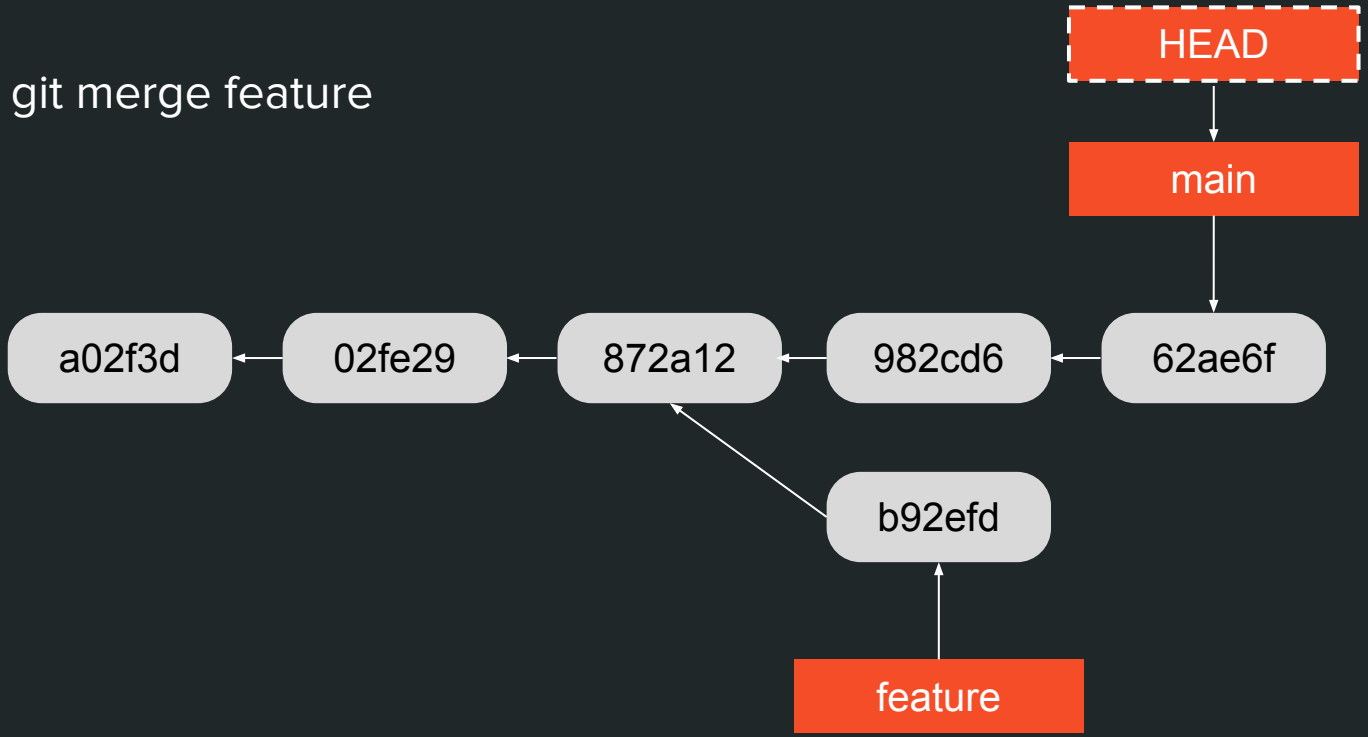


Merging

https://giphy.com/gifs/git-merge-cFkiFMDg3iFoI

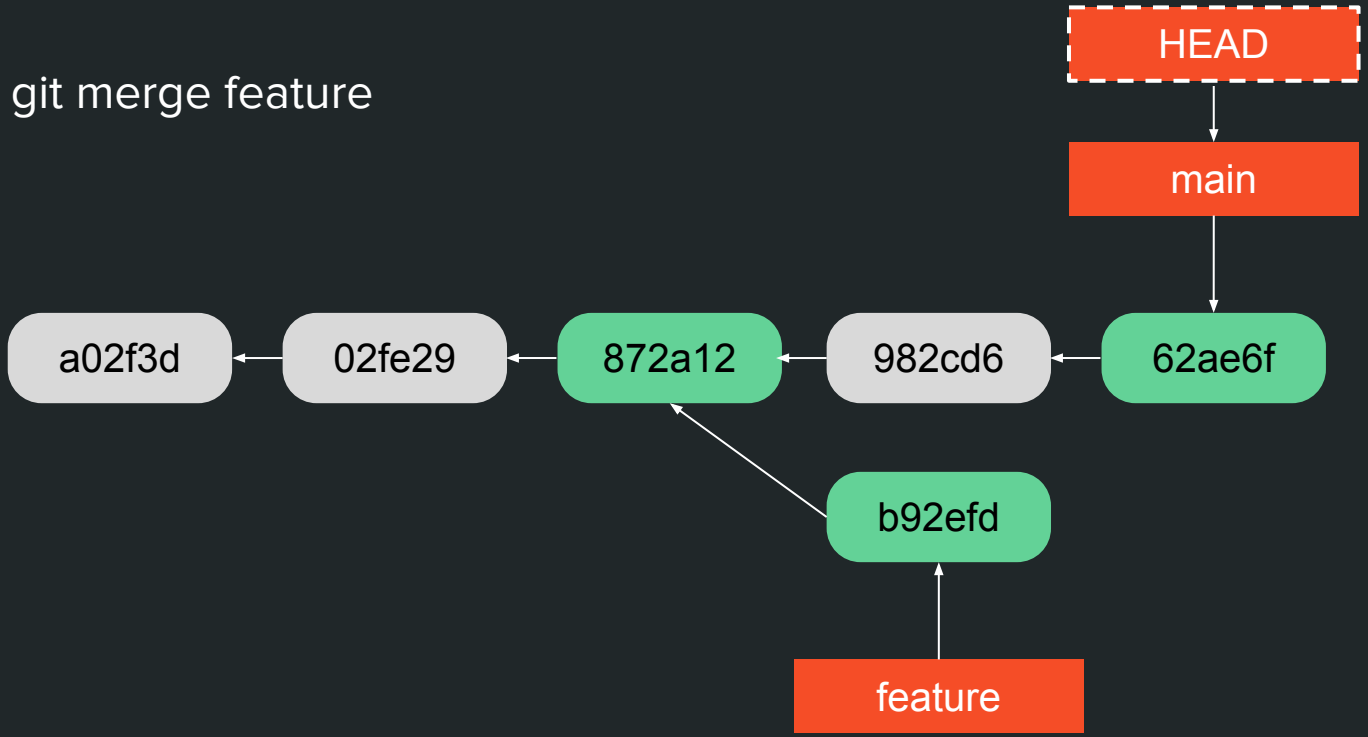
Merging

$ git merge feature



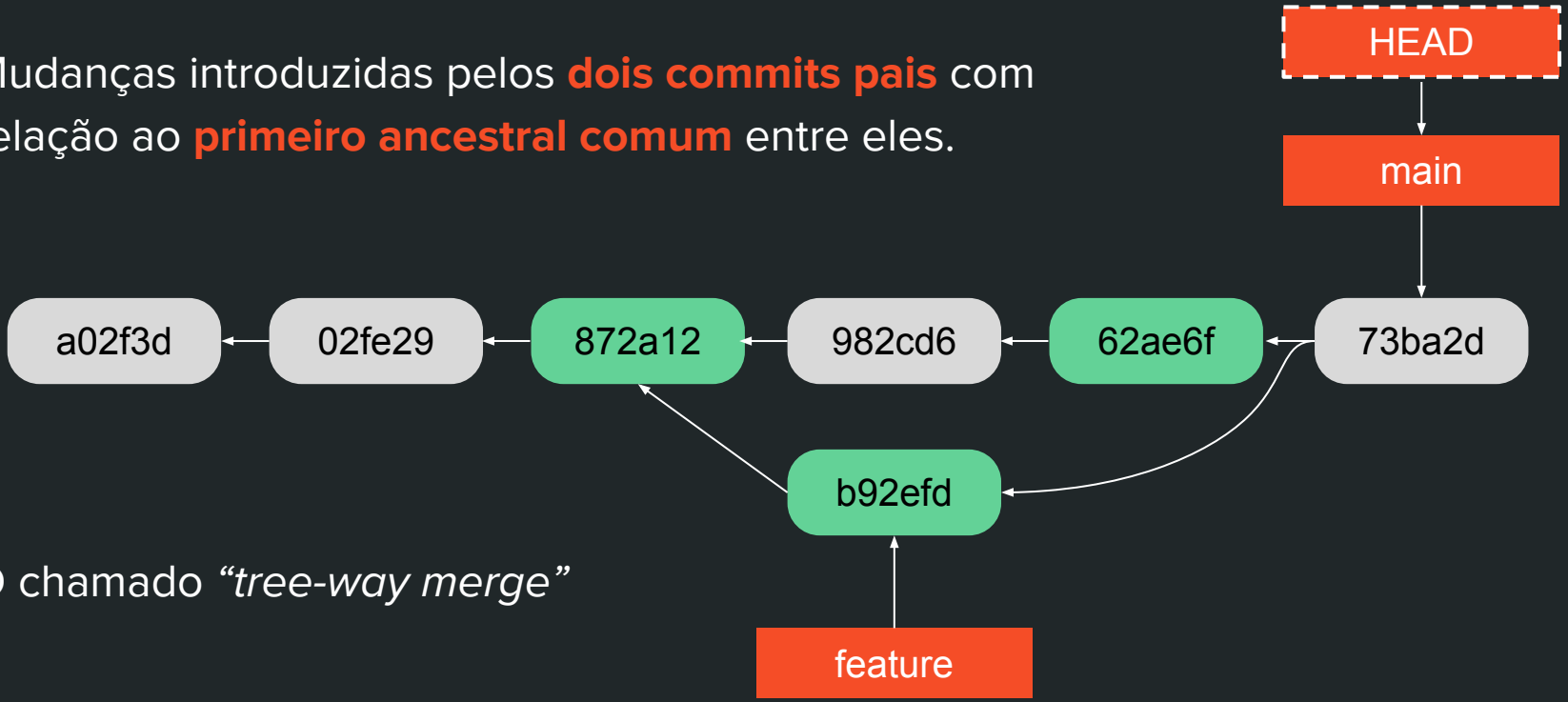
Merging

$ git merge feature



Merging

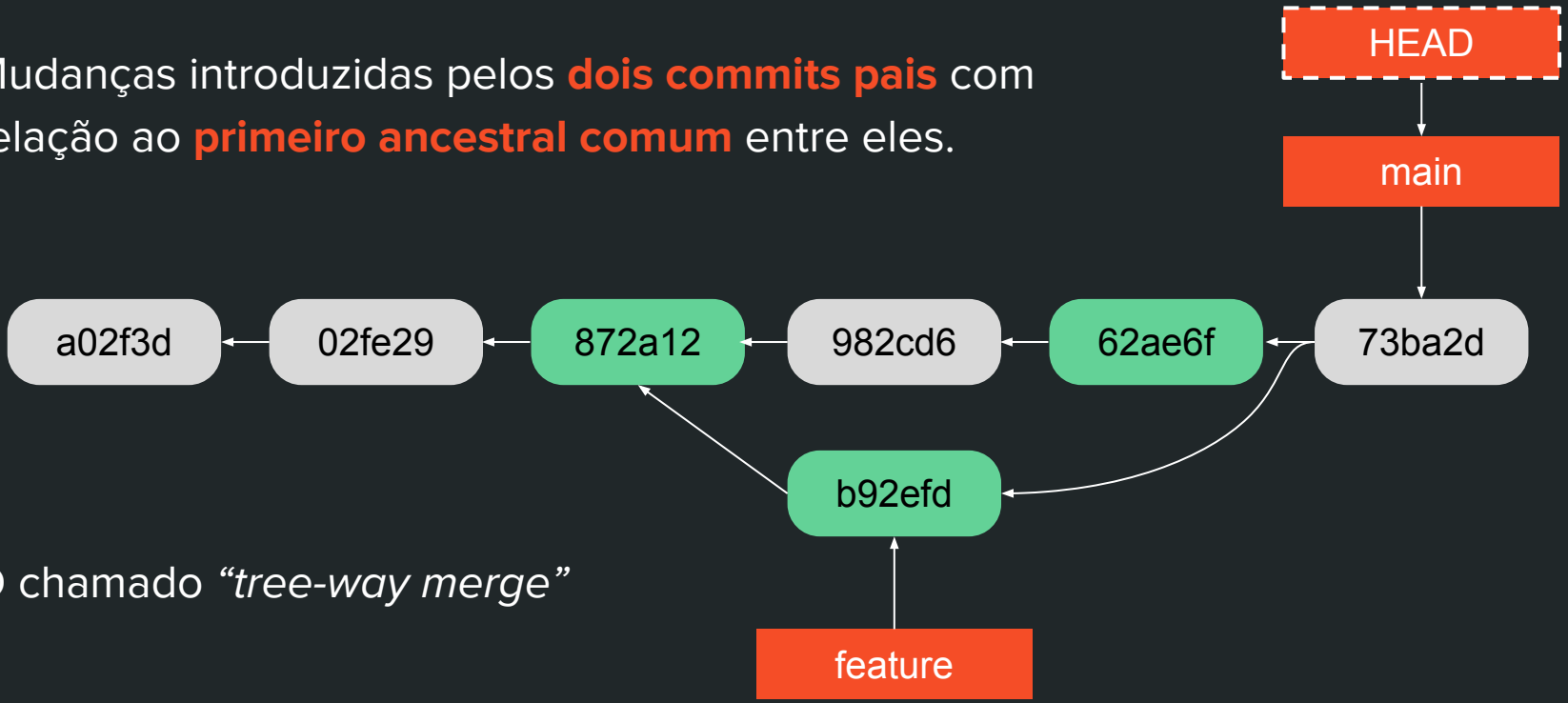
Mudanças introduzidas pelos **dois commits pais** com relação ao **primeiro ancestral comum** entre eles.



O chamado *“tree-way merge”*

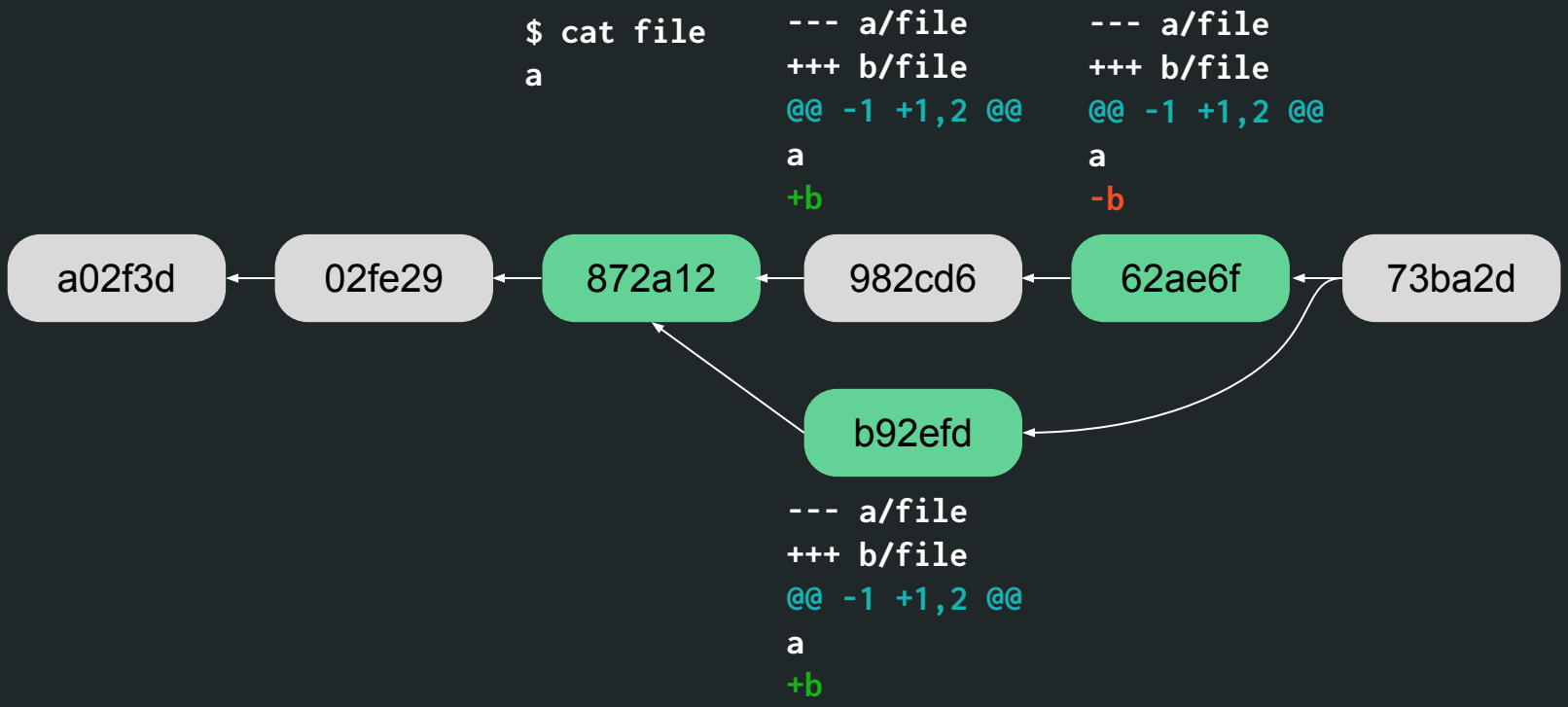
Merging

Mudanças introduzidas pelos **dois commits pais** com relação ao **primeiro ancestral comum** entre eles.

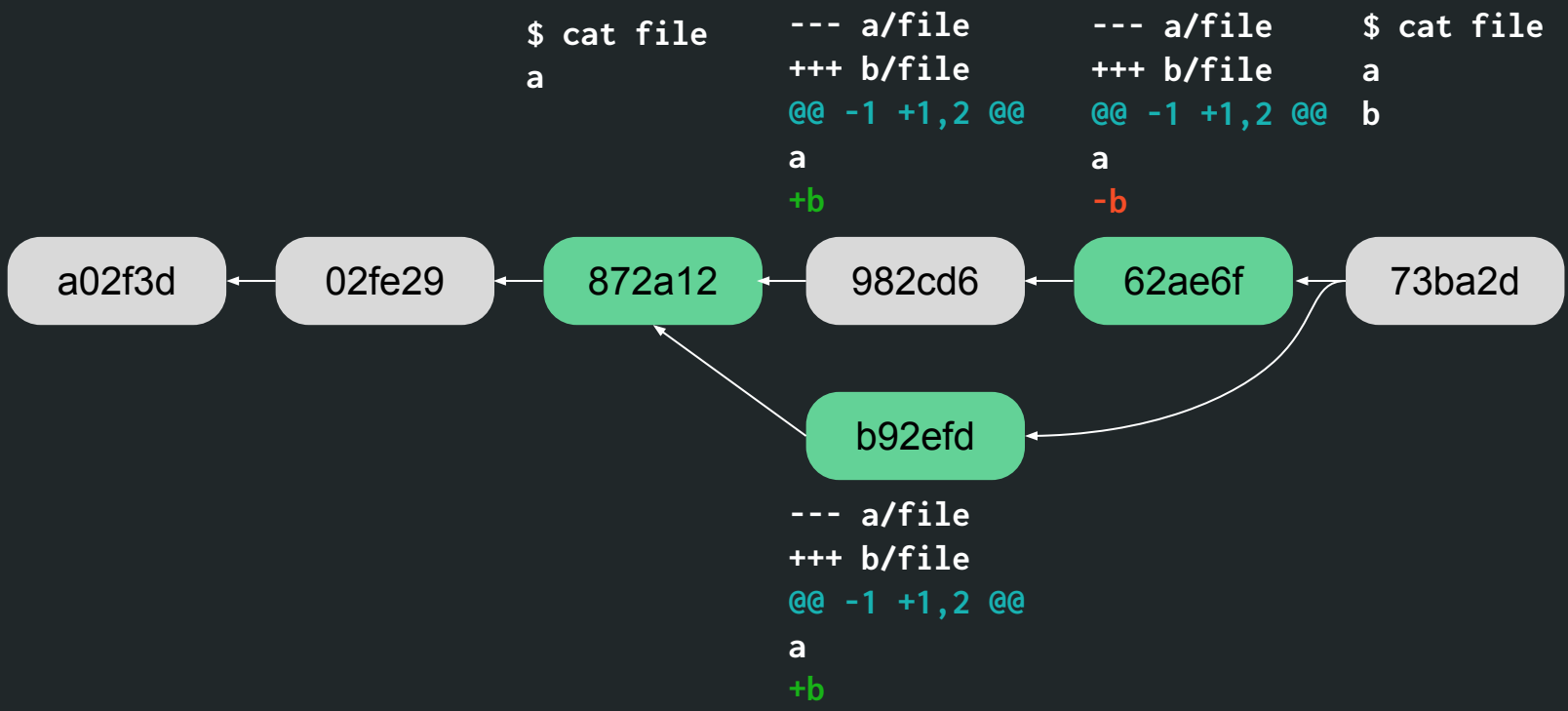


O chamado *“tree-way merge”*

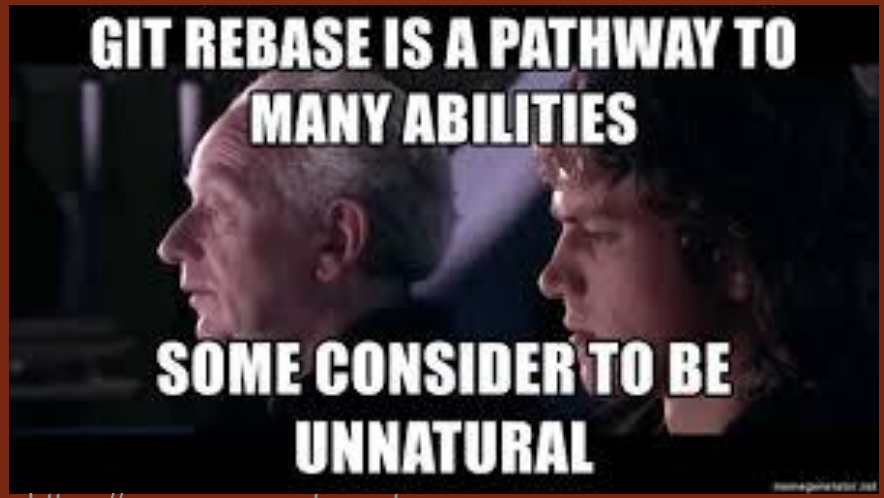
Merging


$ cat file
a
--- a/file
+++ b/file
@@ -1 +1,2 @@
a
+b
--- a/file
+++ b/file
@@ -1 +1,2 @@
a
-b
a02f3d
02fe29
872a12
982cd6
62ae6f
73ba2d
b92efd
--- a/file
+++ b/file
@@ -1 +1,2 @@
a
+b

Merging

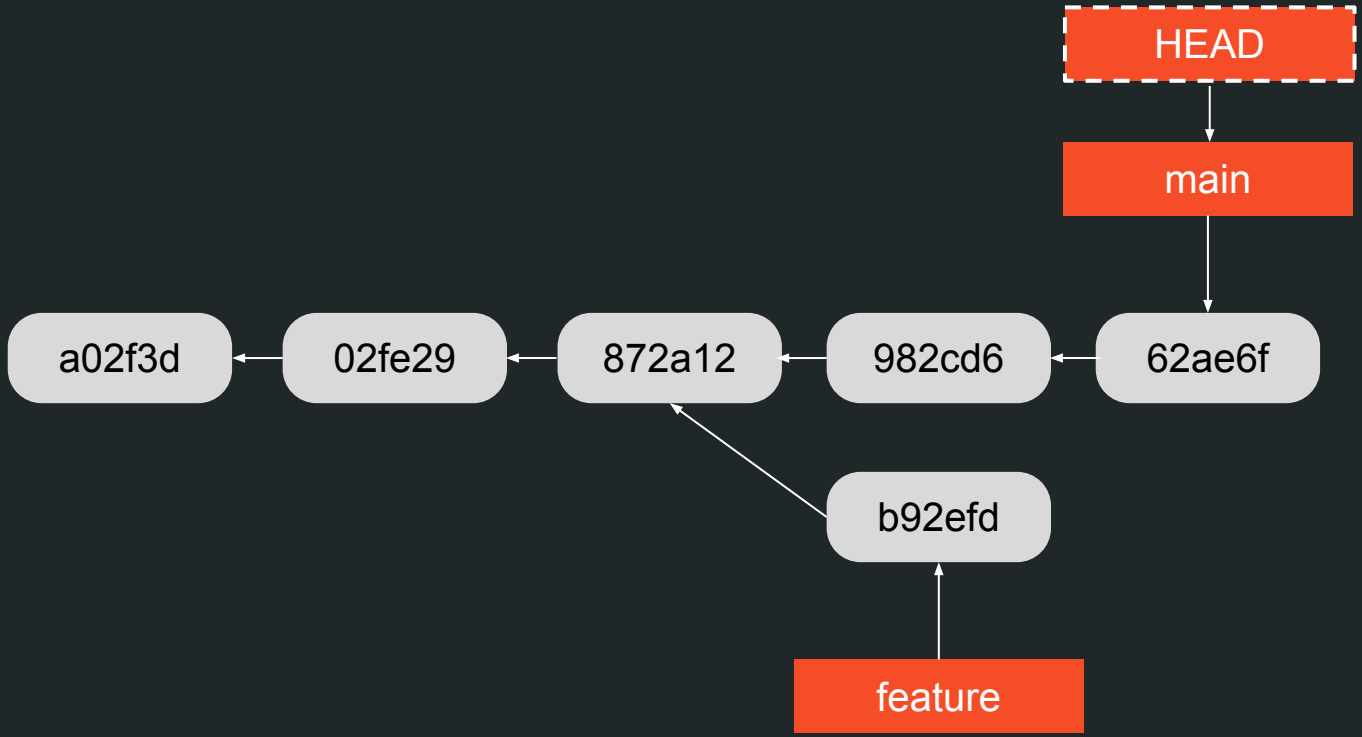

$ cat file
a
--- a/file
+++ b/file
@@ -1 +1,2 @@
a
+b
--- a/file
+++ b/file
@@ -1 +1,2 @@
a
-b
$ cat file
a
b
a02f3d
02fe29
872a12
982cd6
62ae6f
73ba2d
b92efd
--- a/file
+++ b/file
@@ -1 +1,2 @@
a
+b

Rebase: alterando o passado



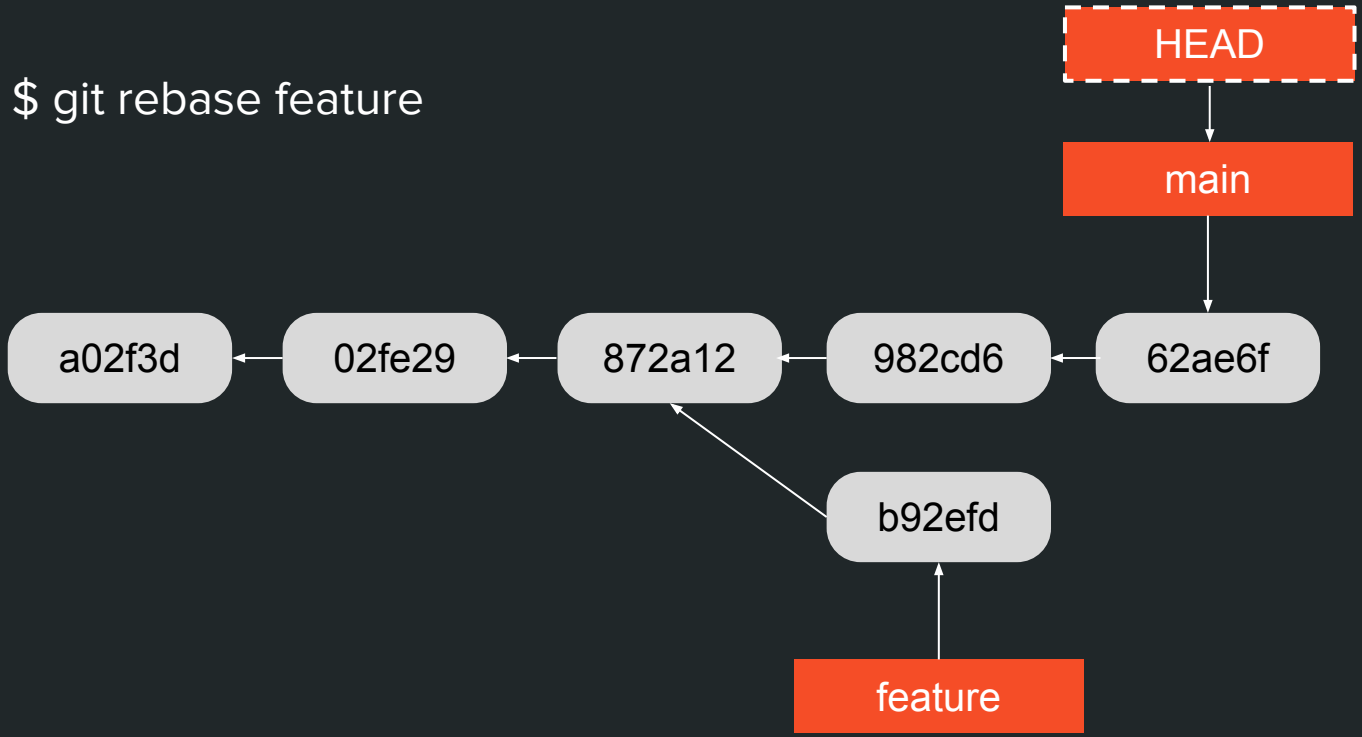
https://memegenerator.net

Rebase


HEAD
main
a02f3d
02fe29
872a12
982cd6
62ae6f
b92efd
feature

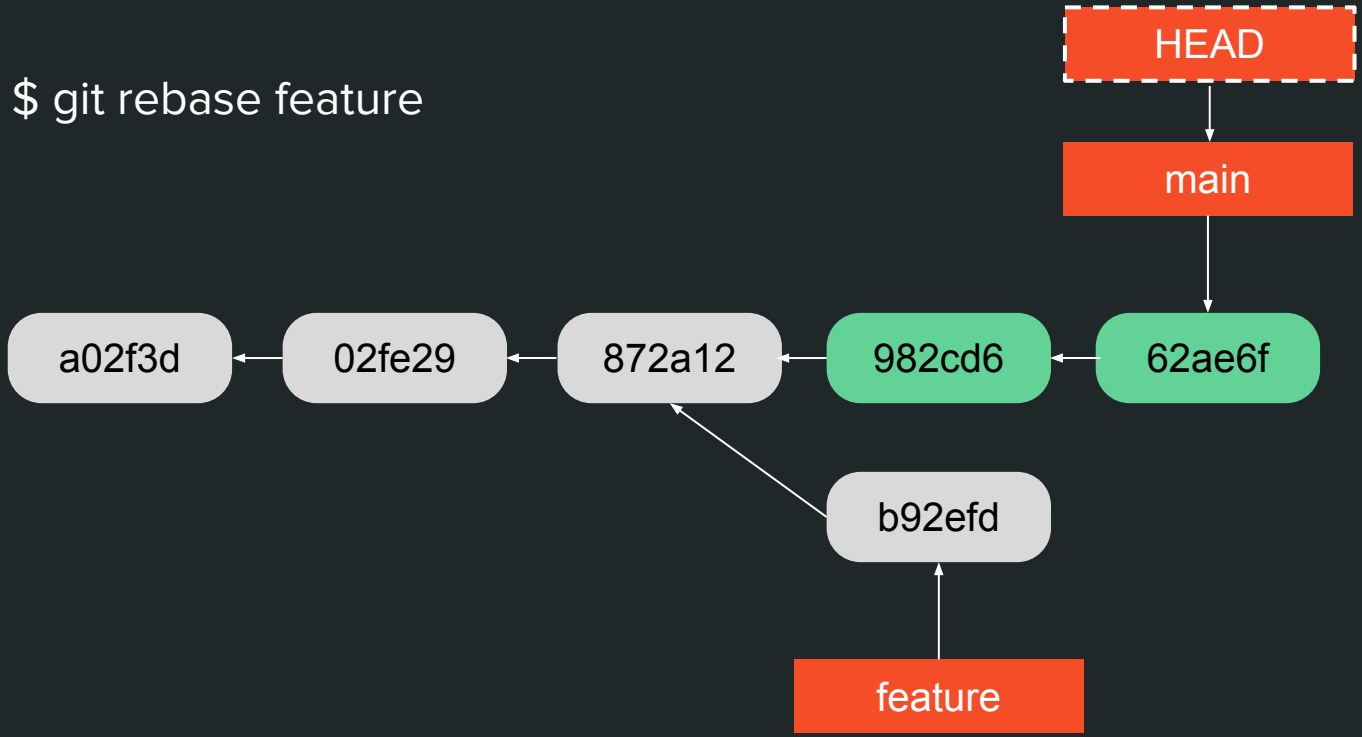
Rebase

$ git rebase feature



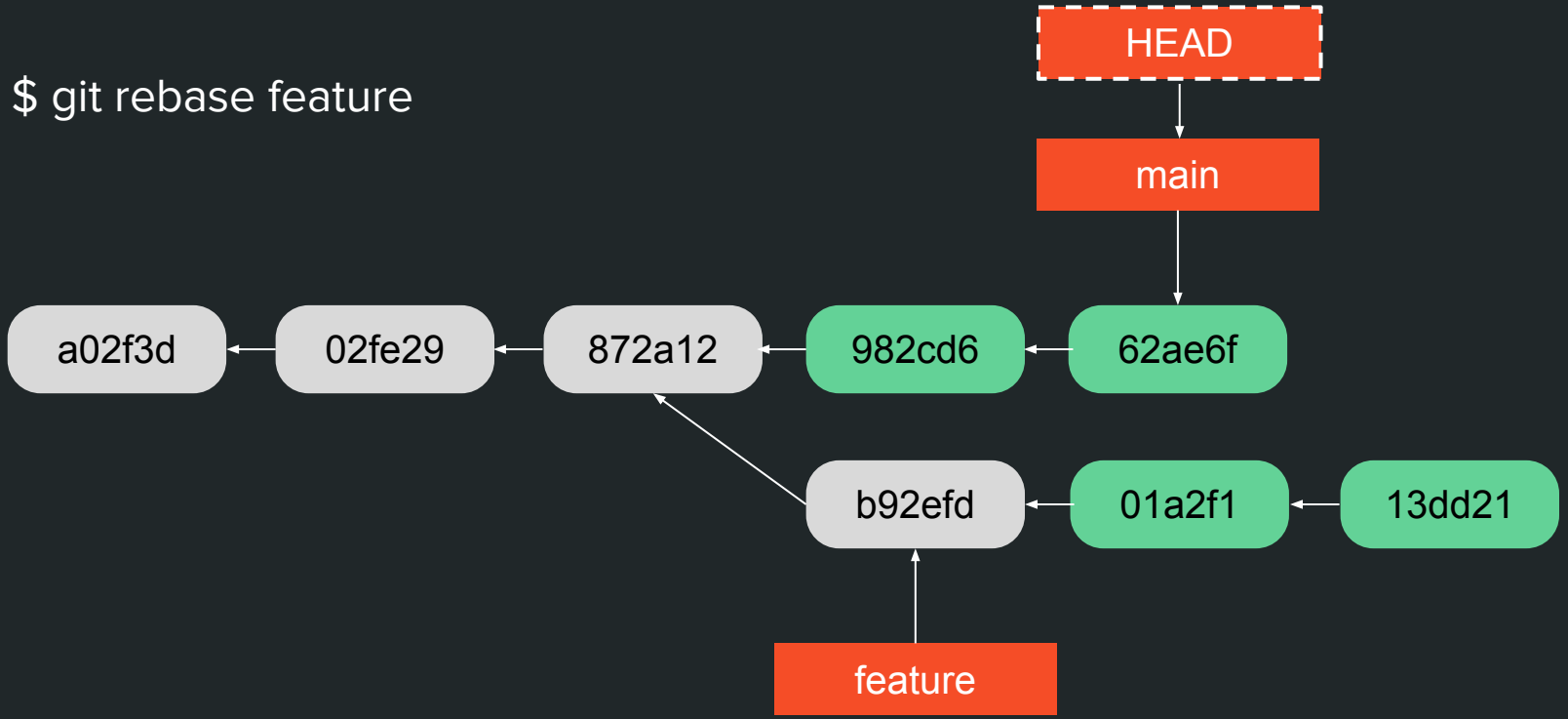
Rebase

$ git rebase feature



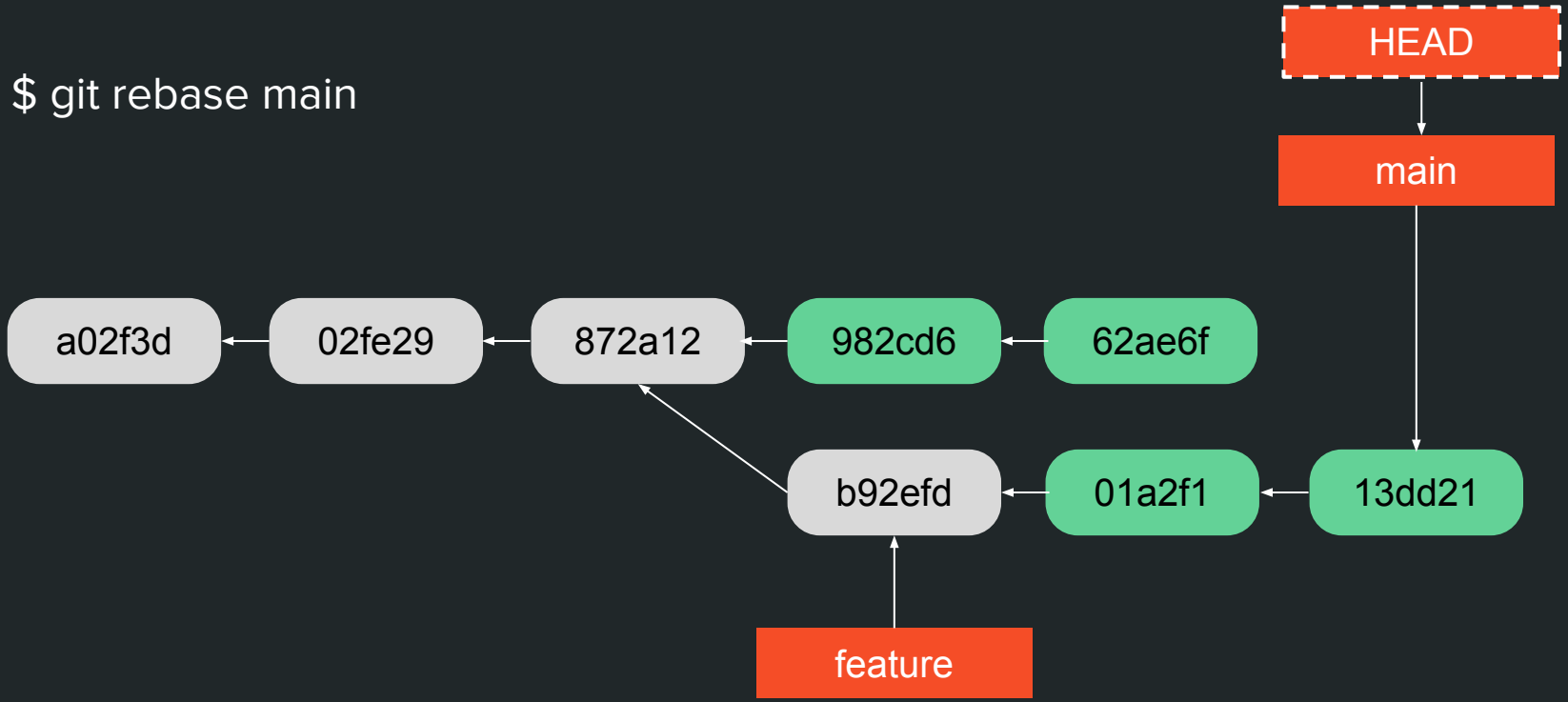
Rebase

$ git rebase feature



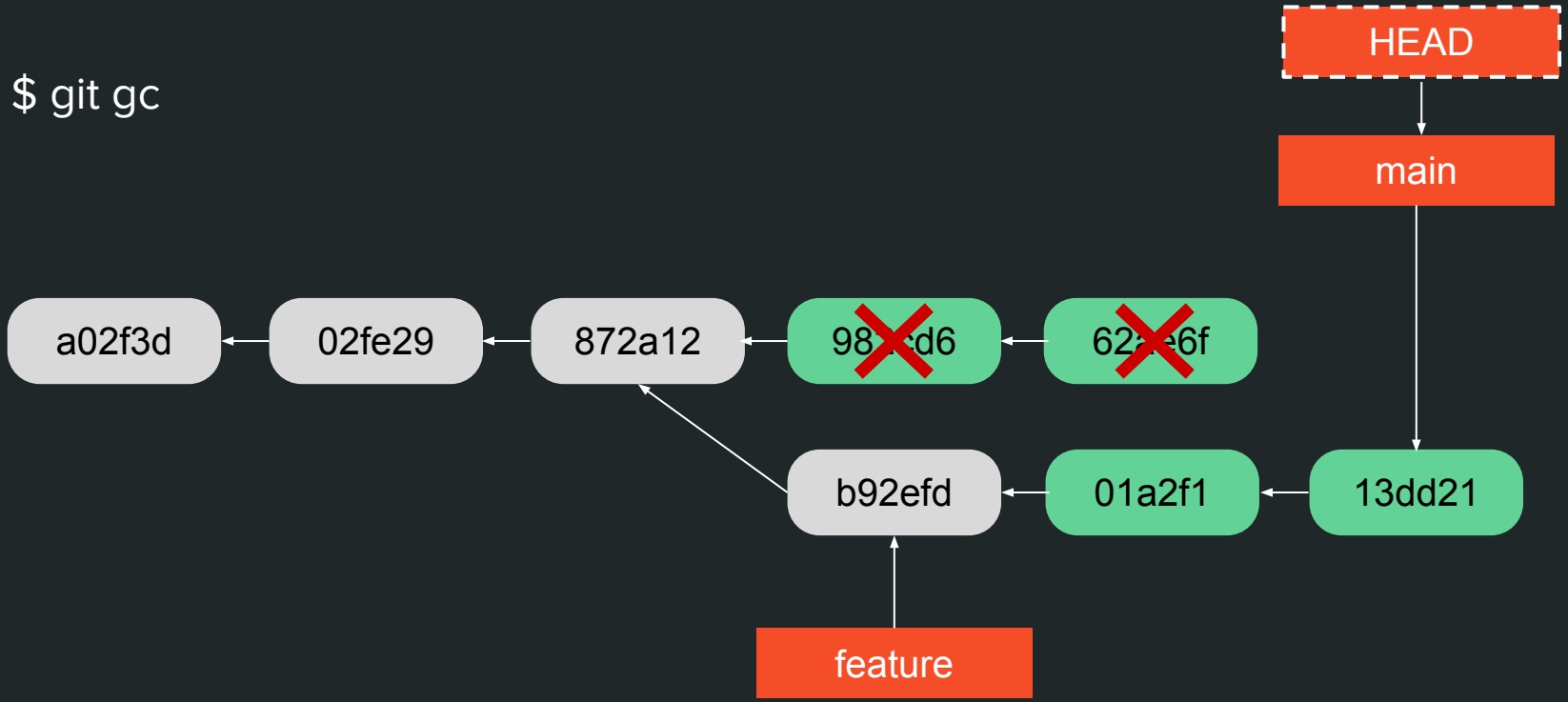
Rebase

$ git rebase main



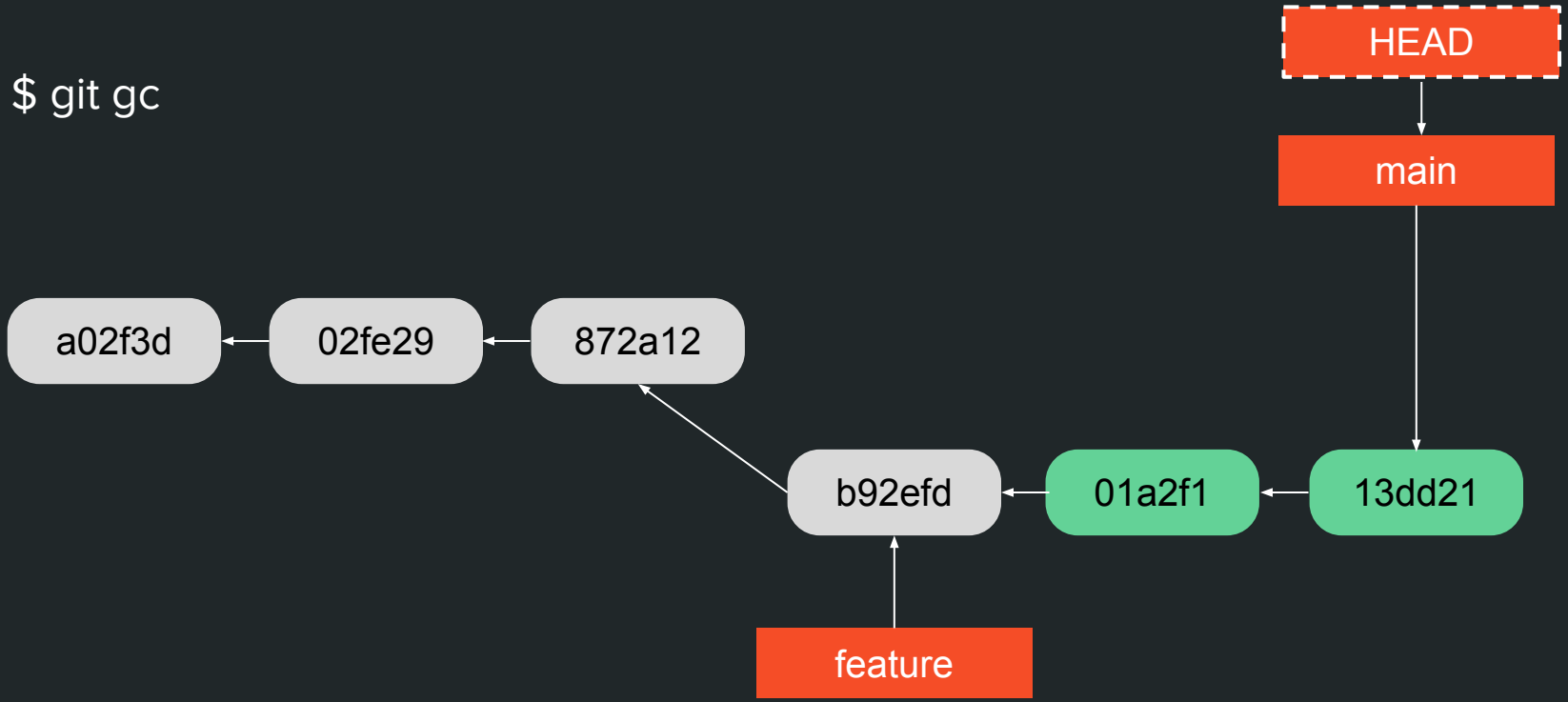
Rebase

$ git gc



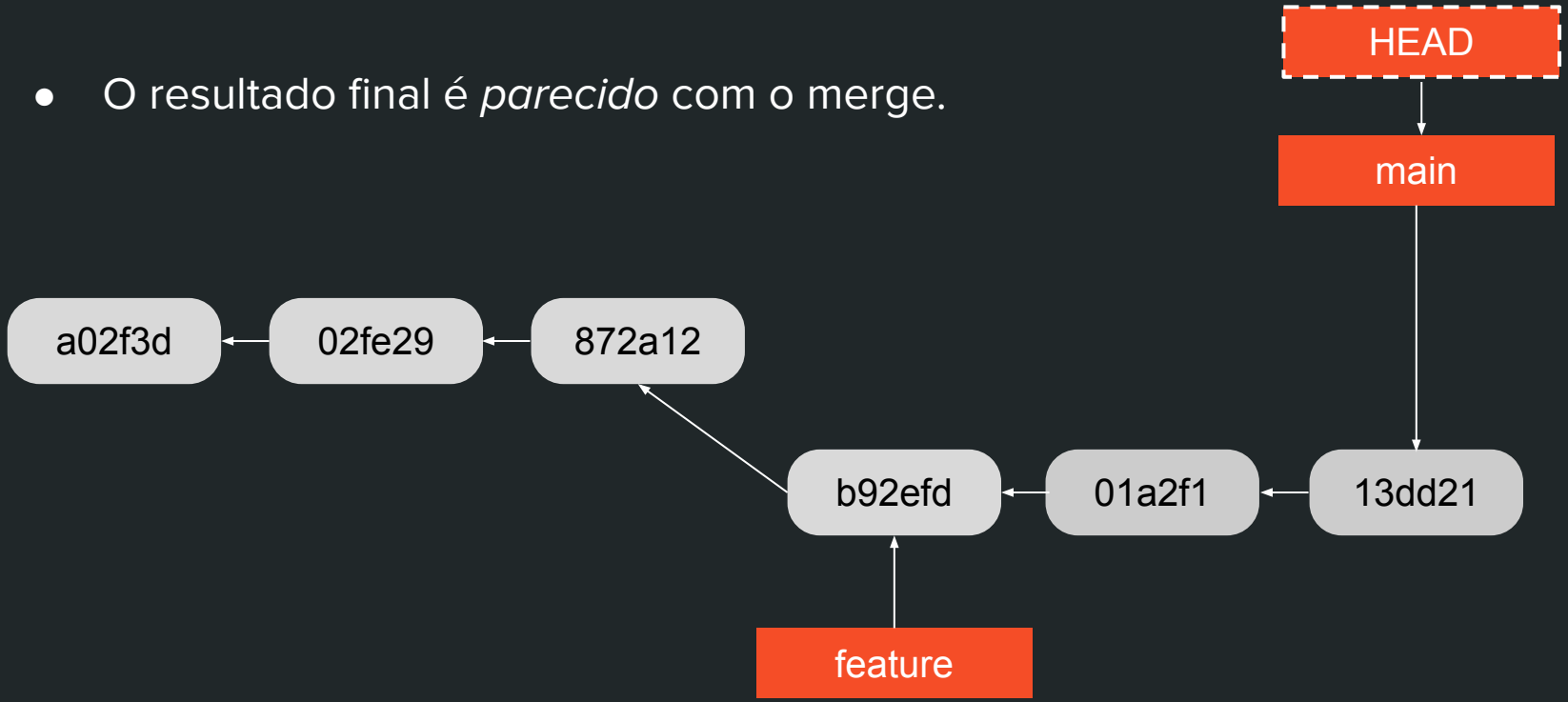
Rebase

$ git gc



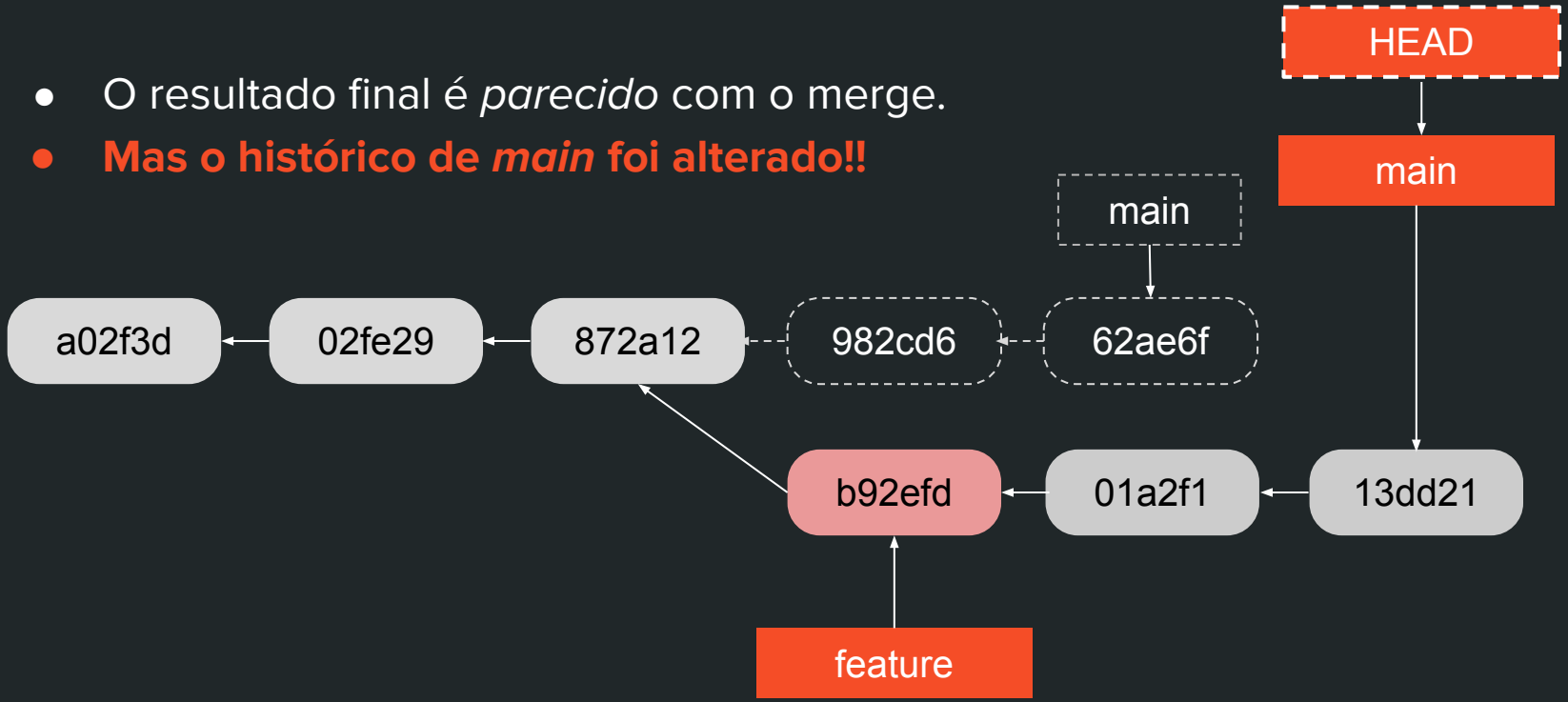
Rebase

- O resultado final é *parecido* com o merge.



Rebase

- O resultado final é *parecido* com o merge.
- **Mas o histórico de *main* foi alterado!!**



Regra de ouro do Rebase

Bom:

- Alterar o histórico de **uma branch pessoal** *(e.g. para organizar seus commits antes de um PR)*.

Possivelmente ruim:

- Alterar o histórico de **uma branch pública** (utilizada por outros).

http://jdduarte.com/git-training/#/

*Leia mais **aqui** e **aqui**.*

$ git rebase -i

- Adicionar
- Remover
- Reordenar
- Editar
- Juntar dois ou mais commits
- Executar um comando para cada commit

Some Git Tips

- *$ git help glossary*
- *$ git help revisions*
- *$ git reflog*
- “Git archeology”*: $ git blame / git log -S*
- Finding bugs or behavior changes*: $ git bisect*
- *$ git clone --depth=1 [--filter=blob:none]*
- *$git config commit.verbose true*

Obrigado!

https://matheustavares.gitlab.io

Referências

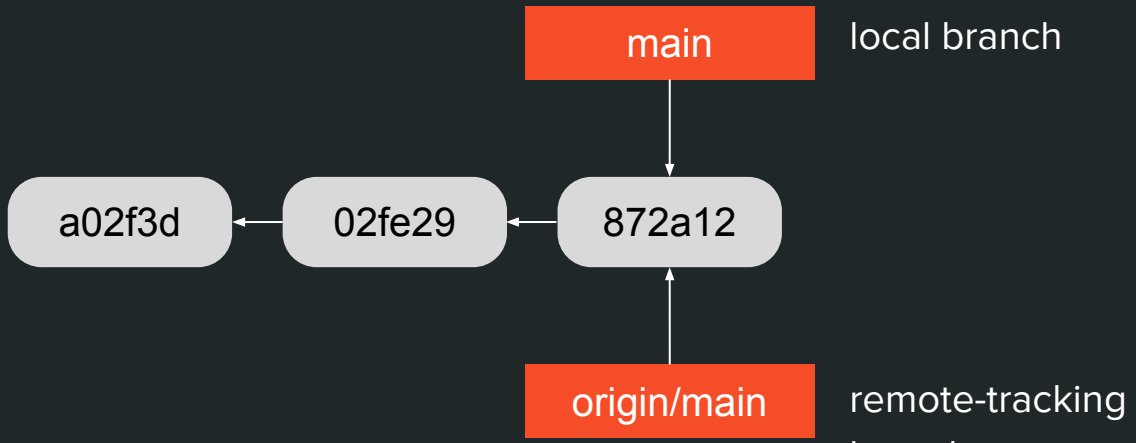
1. **The Zen of Git**, Tianyu Pu: https://speakerdeck.com/tianyupu/the-zen-of-git
2. **Pro Git**, Scott Chacon and Ben Straub: https://git-scm.com/book/en/v2
3. **Git Docs:** https://git-scm.com/docs/
4. **Git For Computer Scientists**, Tommi Virtanen: https://eagain.net/articles/git-for-computer-scientists/

https://matheustavares.gitlab.io

Extra: Remotes

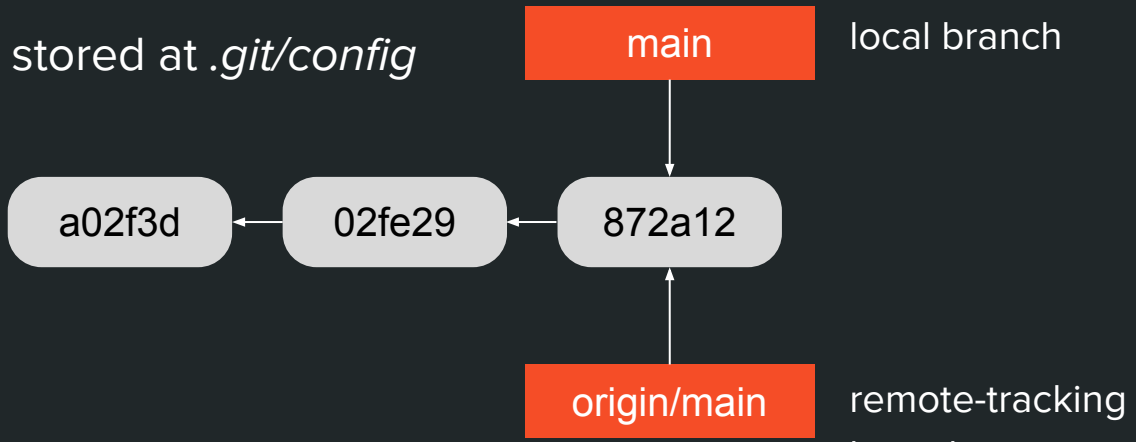
Remote-tracking Branches

$ git clone



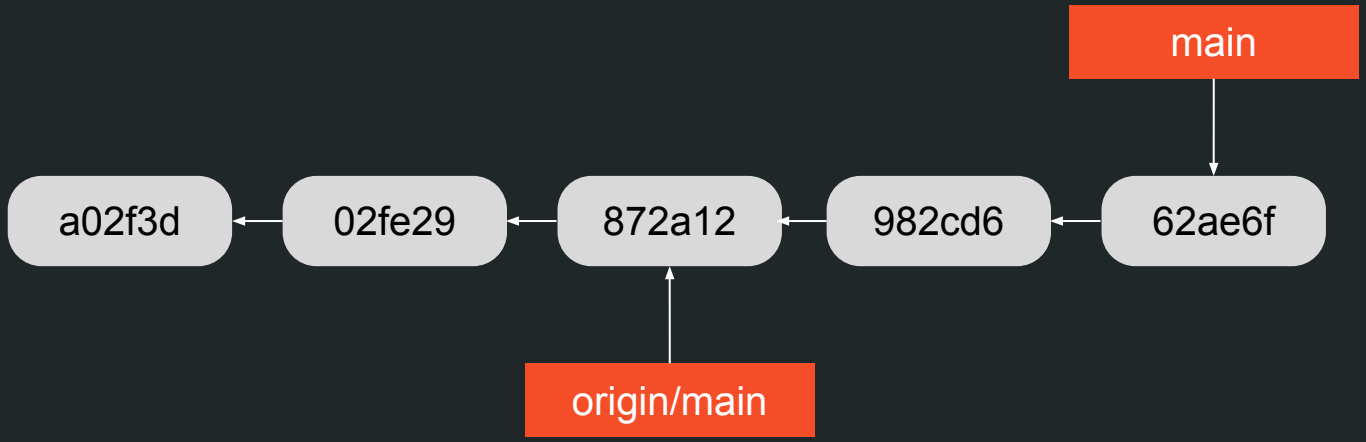
Remote-tracking Branches

The *tracking info* is stored at *.git/config*



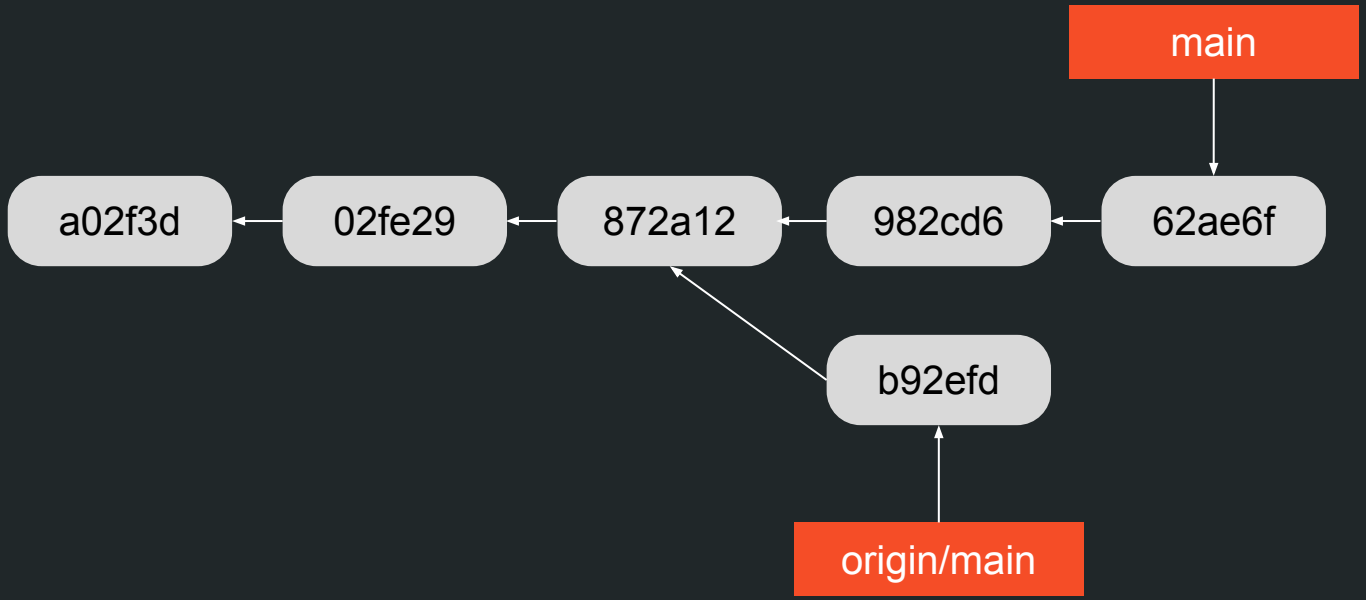
Remote-tracking Branches

$ git commit (2x)



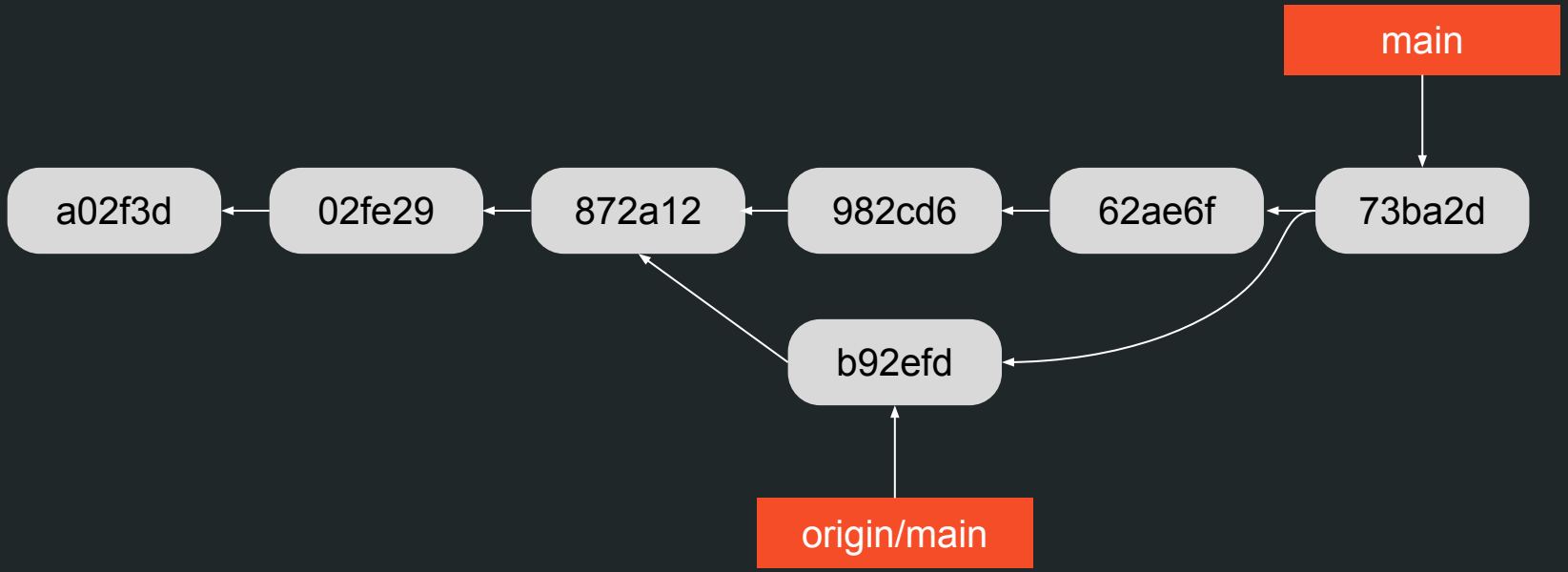
Remote-tracking Branches

$ git fetch origin



Remote-tracking Branches

$ git merge origin/main



Remote-tracking Branches

- Acabamos de replicar um “git pull”