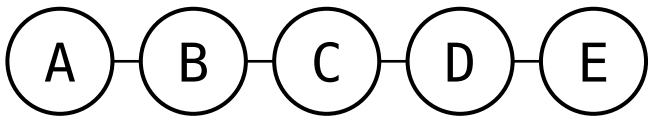
Rebase after base

Learn how to master `git rebase`

Hugo ARNAL

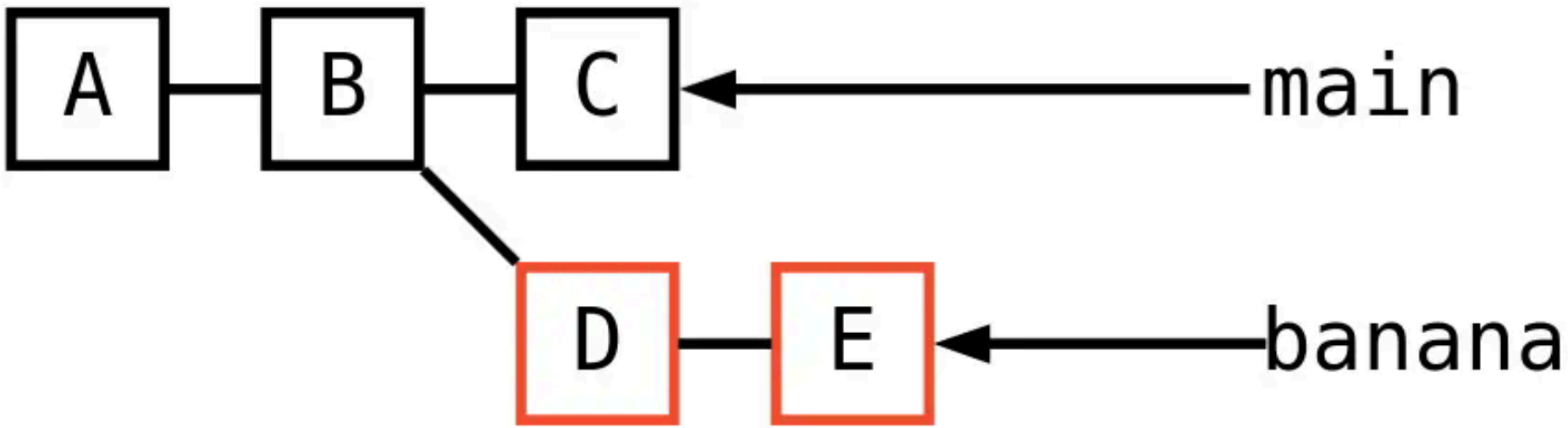
What is a rebase?

Commits

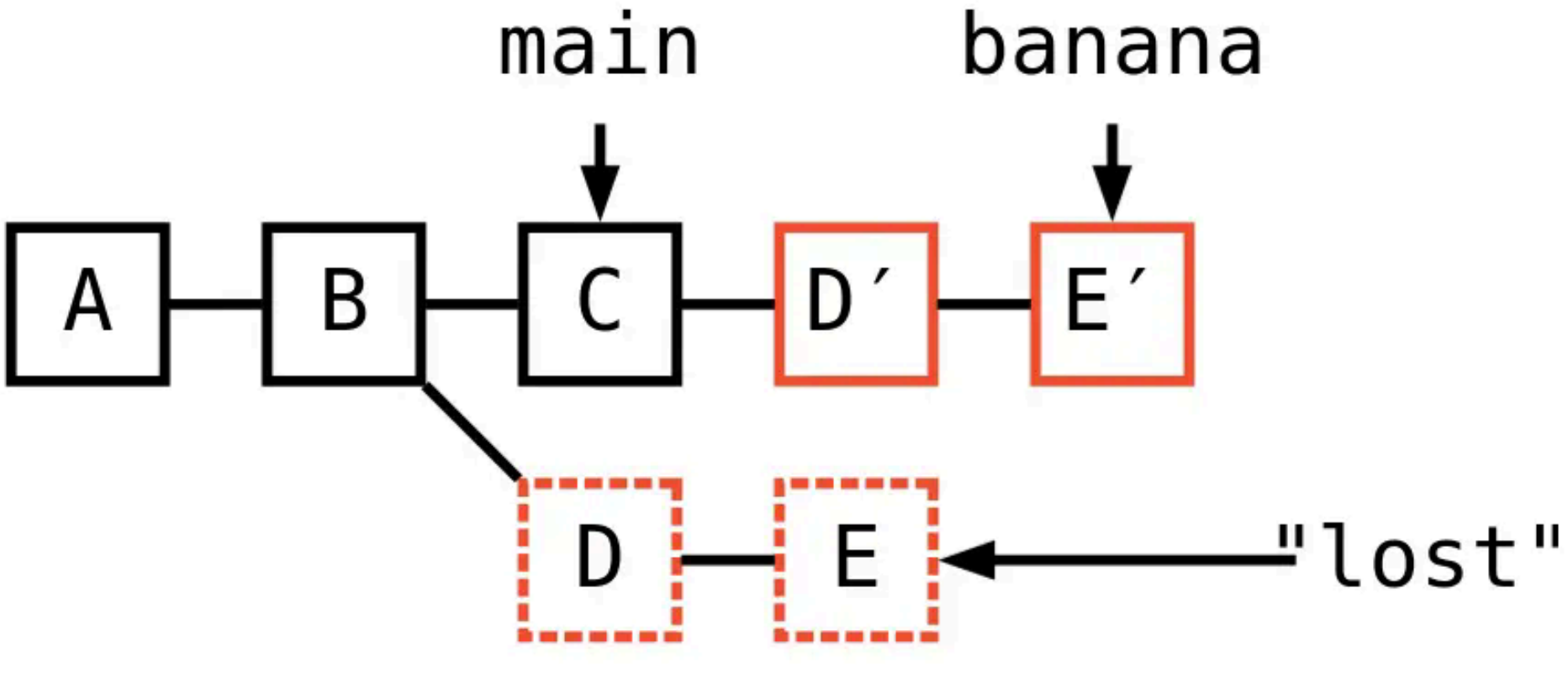


From left → right

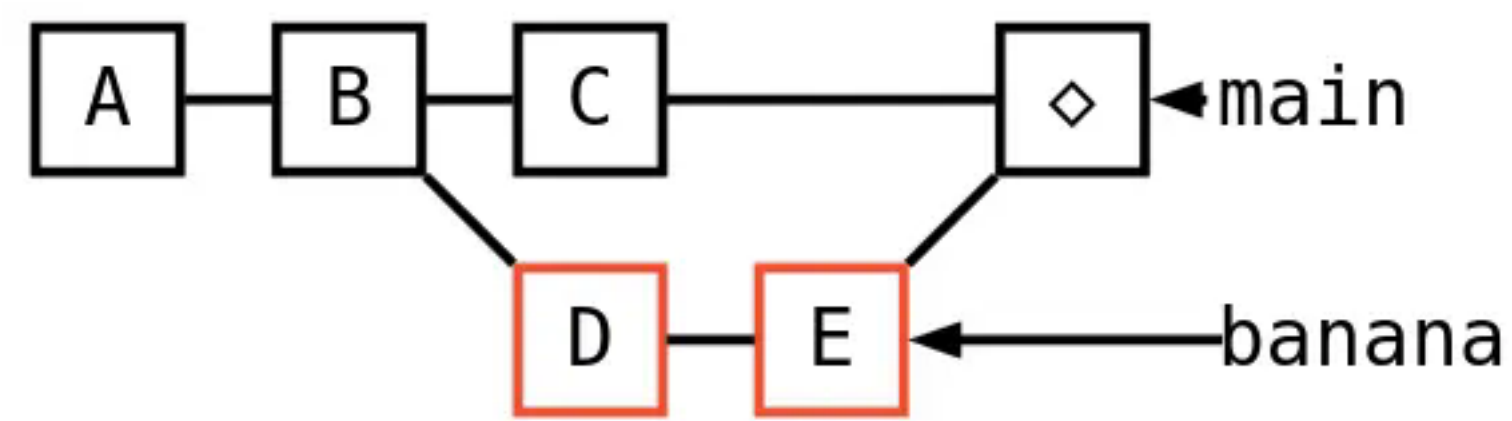
Branches



Rebase



Merging

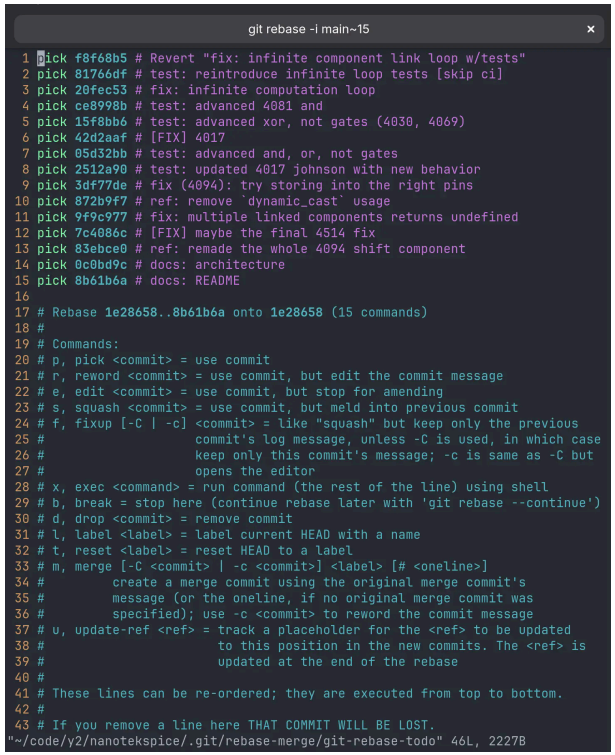


⚠ Warning ⚠

Using `git rebase` will lead to use `git push --force` which is EXTREMELY dangerous.

Full control with interactive mode

`git rebase -i main~50` (50 last commits since `main`)

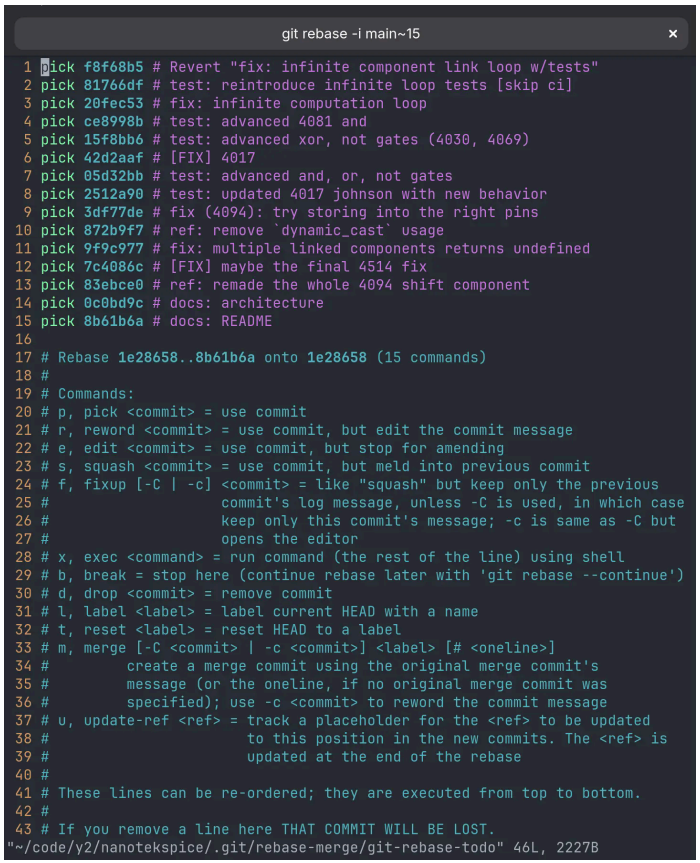


Full control with interactive mode

- Step 1: do not `panic!()`
- Step 2: `git config --global core.editor vim`
- Step 3: let’s analyse this

Full control with interactive mode

- Your commits are listed from oldest to newest
- By default, they’re all in “pick” mode: essentially does not touch the commit.



Full control with interactive mode

- Change the content of a commit by replacing “pick” with “edit”

```
hugo@chicago ~/code/y2/nanotekspice > git rebase -i main~15
Stopped at 81766df...  # test: reintroduce infinite loop tests [skip ci]
You can amend the commit now, with

  git commit --amend

Once you are satisfied with your changes, run

  git rebase --continue
hugo@chicago ~/code/y2/nanotekspice > gst
interactive rebase in progress; onto 1e28658
Last commands done (2 commands done):
   pick f8f68b5 # Revert "fix: infinite component link loop w/tests"
   edit 81766df # test: reintroduce infinite loop tests [skip ci]
Next commands to do (13 remaining commands):
   pick 20fec53 # fix: infinite computation loop
   pick ce8998b # test: advanced 4081 and
  (use "git rebase --edit-todo" to view and edit)
You are currently editing a commit while rebasing branch 'main' on '1e28658'.
  (use "git commit --amend" to amend the current commit)
  (use "git rebase --continue" once you are satisfied with your changes)

nothing to commit, working tree clean
hugo@chicago ~/code/y2/nanotekspice >
```

```
hugo@chicago ~/code/y2/nanotekspice > gst
interactive rebase in progress; onto 1e28658
Last commands done (2 commands done):
   pick f8f68b5 # Revert "fix: infinite component link loop w/tests"
   edit 81766df # test: reintroduce infinite loop tests [skip ci]
Next commands to do (13 remaining commands):
   pick 20fec53 # fix: infinite computation loop
   pick ce8998b # test: advanced 4081 and
  (use "git rebase --edit-todo" to view and edit)
You are currently editing a commit while rebasing branch 'main' on '1e28658'.
  (use "git commit --amend" to amend the current commit)
  (use "git rebase --continue" once you are satisfied with your changes)

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

no changes added to commit (use "git add" and/or "git commit -a")
hugo@chicago ~/code/y2/nanotekspice > git add .
hugo@chicago ~/code/y2/nanotekspice > git commit --amend -m "modified commit"
[detached HEAD 3eb0e14] modified commit
 Date: Mon Mar 2 13:13:31 2026 +0100
 3 files changed, 9 insertions(+)
 create mode 100644 tests/snes/files/other/simple_infinite_loop.nts
hugo@chicago ~/code/y2/nanotekspice > git rebase --continue
Successfully rebased and updated refs/heads/main.
```

Success!

```
commit 3eb0e14da2ec258754076d7112f204f1a3ff9c55
Author: Hugo ARNAL <hugo@hugoarnal.com>
Date:   2026-03-02 13:13:31 +0100

    modified commit

commit f8f68b592518907f61d4bdcd30ed5076edc0b86e
Author: Hugo ARNAL <hugo@hugoarnal.com>
Date:   2026-03-02 13:03:52 +0100

    Revert "fix: infinite component link loop w/tests"

    This reverts commit 5874cfe0b4794e22a2ad2d91f2779b17d9237b95.

commit 1e28658a92f00254b12beae2784eaa1f9d52442d
Author: Hugo ARNAL <hugo@hugoarnal.com>
Date:   2026-03-03 11:08:17 +0100

    test: updated 4514 decoder
```

Full control with interactive mode

- Squash commits to clean your history

```
git checkout -b squash
for c in H e l l o ' ' w o r l d; do
    echo "$c" >> squash.txt
    git add squash.txt
    git commit -m "Add '$c' to squash.txt"
done
```

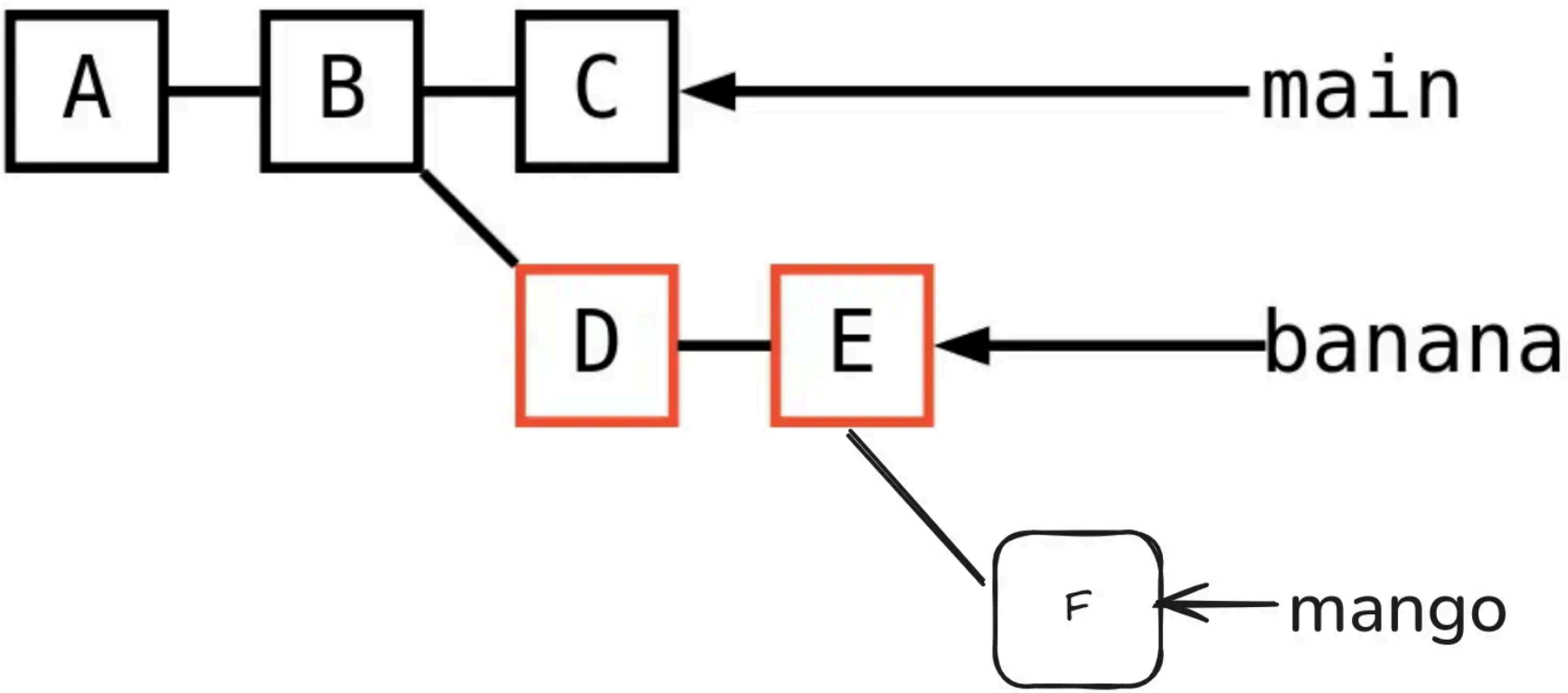
Source: git-rebase.io

- Replacing all picks (except the first one) with squash

```
pick 1e85199 Add 'H' to squash.txt
squash fff6631 Add 'e' to squash.txt
squash b354c74 Add 'l' to squash.txt
squash 04aaf74 Add 'l' to squash.txt
squash 9b0f720 Add 'o' to squash.txt
squash dc158cd Add ' ' to squash.txt
squash dfcf9d6 Add 'w' to squash.txt
squash 7a85f34 Add 'o' to squash.txt
squash c275c27 Add 'r' to squash.txt
squash a513fd1 Add 'l' to squash.txt
squash 6b608ae Add 'd' to squash.txt
```

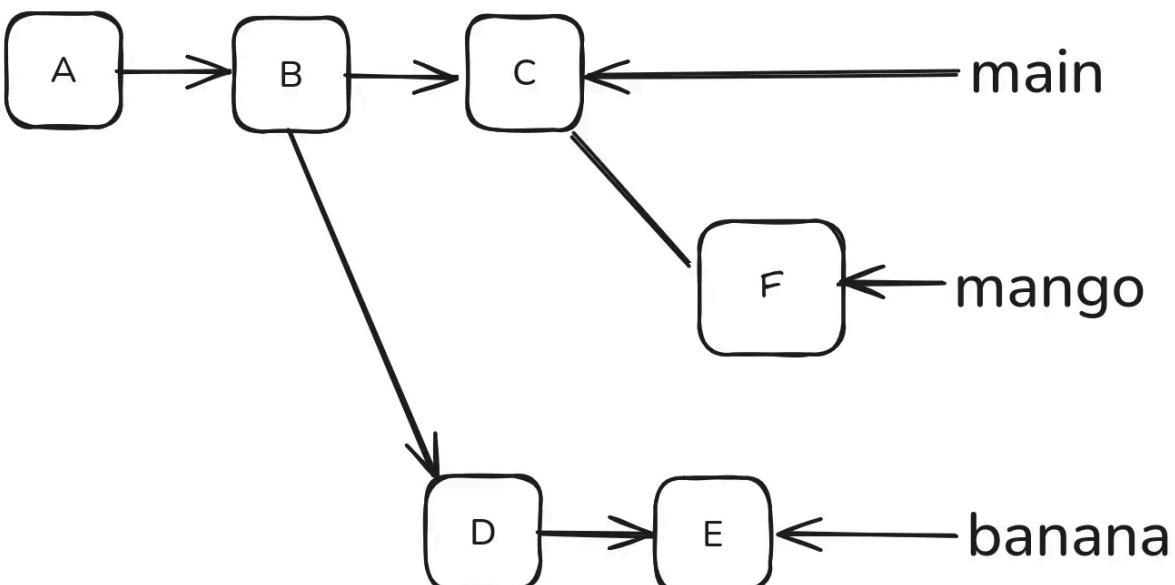
- This will create a singular commit with all the content from the other commits.

Rebasing to… rebase!



Rebasing to… rebase!

Let’s say mango doesn’t need to depend on banana to be merged in main.

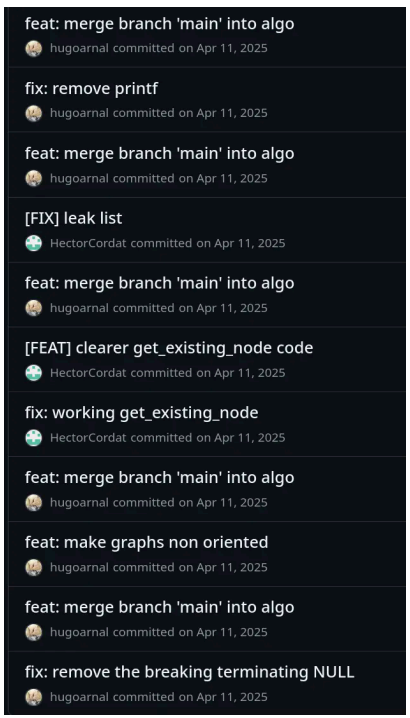


Bunch of other features

- Using `git rebase --autosquash`
- Splitting one commit into several
- Reordering commits
- Resolving conflicts

Why use git rebase?

- Cleaner git history (prevent merge commits)



Why use `git rebase`?

- More control over your git history

Editing and controlling the git history of your branch is usually essential when working on open source projects.

Why use git rebase?

- Features are clearer when getting reviewed

Question: as a PR reviewer, which one do you find clearer?

1.

- feat: new feature
- fix: feature
- ref: change how feature works
- imp: respect coding style
- Merge branch “main”
- fix: new changes with main branch
- feat: another important feature

2.

- feat: new feature
- feat: another important feature

When you should NOT use `git rebase`

- Two people working on the same branch

Pushing with `--force` might delete some work

An alternative to it is `git push --force-with-lease` but retains dangerous

When you should NOT use `git rebase`

- Rewriting the history of an already pushed branch

I recommend to prevent this by setting rules on your branches (unfortunately can’t be done on Epitech repos, except for mirrors).

In resume

- Use `git rebase` to maintain a clean history

Cleaner history makes code reviews easier in general.

- Do not use rebases when working with other people on the same branch (usually).

Because you will use `git push --force` which is ⚠ DANGEROUS ⚠

Going further?

`git history` landing in Git 2.54!

A safer way to just modify commit messages, without rebasing.

See an example

Sources

Useful tutorials:

- https://git-rebase.io
- https://learngitbranching.js.org

- https://git-scm.com/cheat-sheet.pdf
- https://jvns.ca/blog/2023/11/06/rebasing-what-can-go-wrong-
- https://github.blog/open-source/git/highlights-from-git-2-54/
- https://git-scm.com/docs/git-history/2.54.0
- https://front-end.social/@stefan/116441339289854563

Slides

https://github.com/hugoarnal/talks