Understand what we're doing on Git

Link to slides & Exercises

Maxime Fabre

https://github.com/fabremx/git-formation

Sources and inspirations

http://blog.schauderhaft.de/gitkata/

https://github.com/praqma-training/git-katas

https://learngitbranching.js.org/

TABLE OF CONTENT

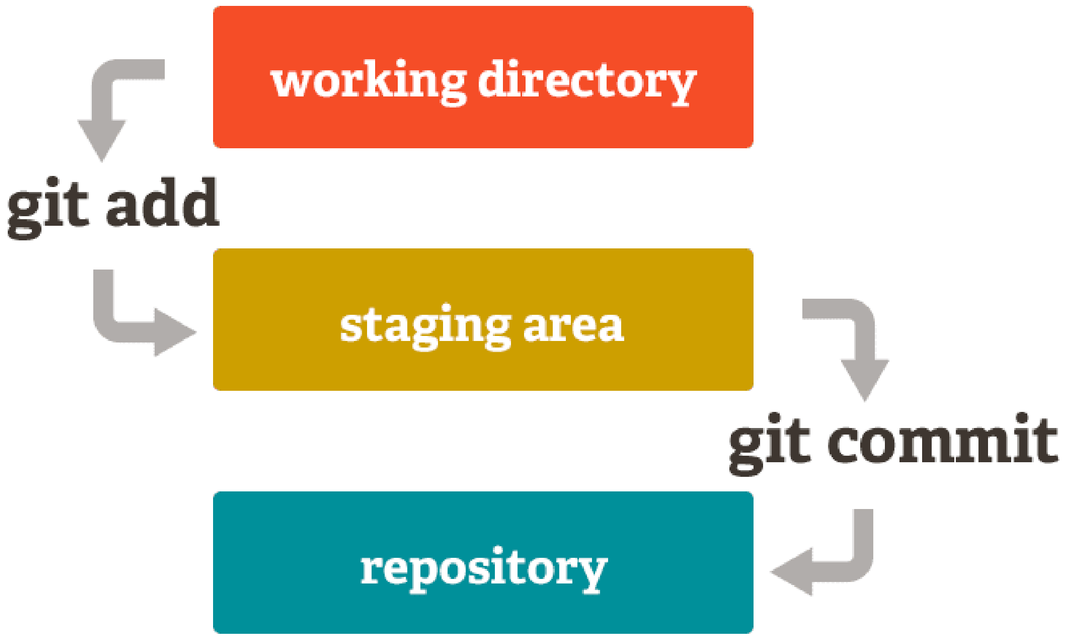
What we’re going to do

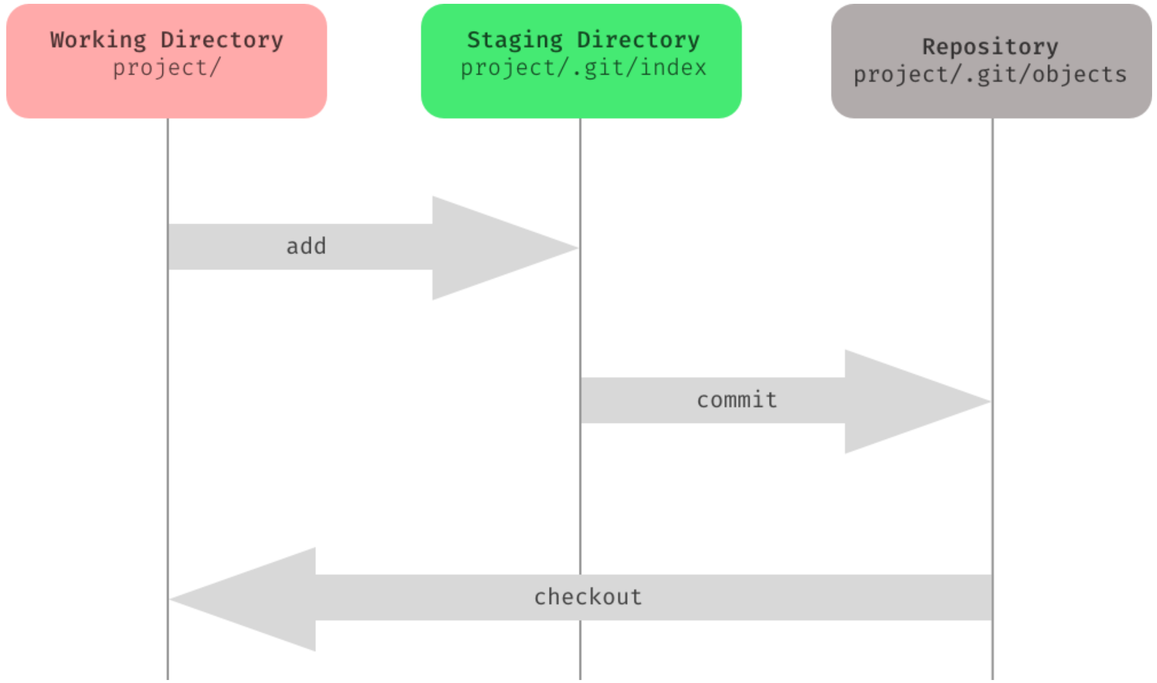
1. Git stages
2. Commits and branches
3. Merge strategy
4. Move into commits
5. Revert vs Reset
6. Rewrite history
7. Origin and remote branches
8. Fetch vs pull
9. Tracking branches
10. Good practices
11. Let’s practice with scenarised exercises

1. Git stages

Git stages

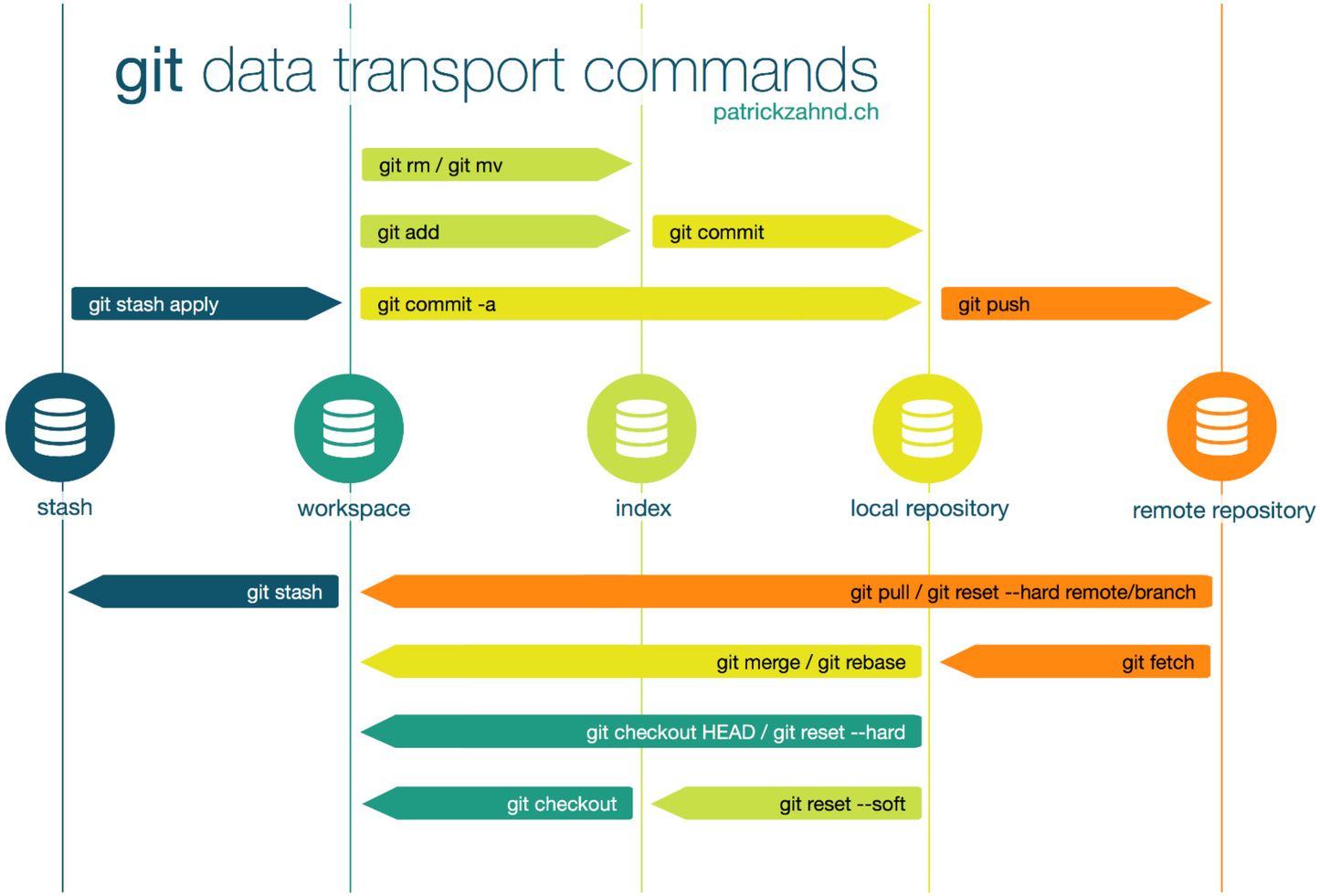
Stages of local repository





Git stages

Local + Remote repositories



Let’s Practice

Basic staging + Basic Stashing

https://github.com/fabremx/git-formation/tree/master/exercises/basic-staging

https://github.com/fabremx/git-formation/tree/master/exercises/basic-stashing

2. Commits and branches

Commits and branches

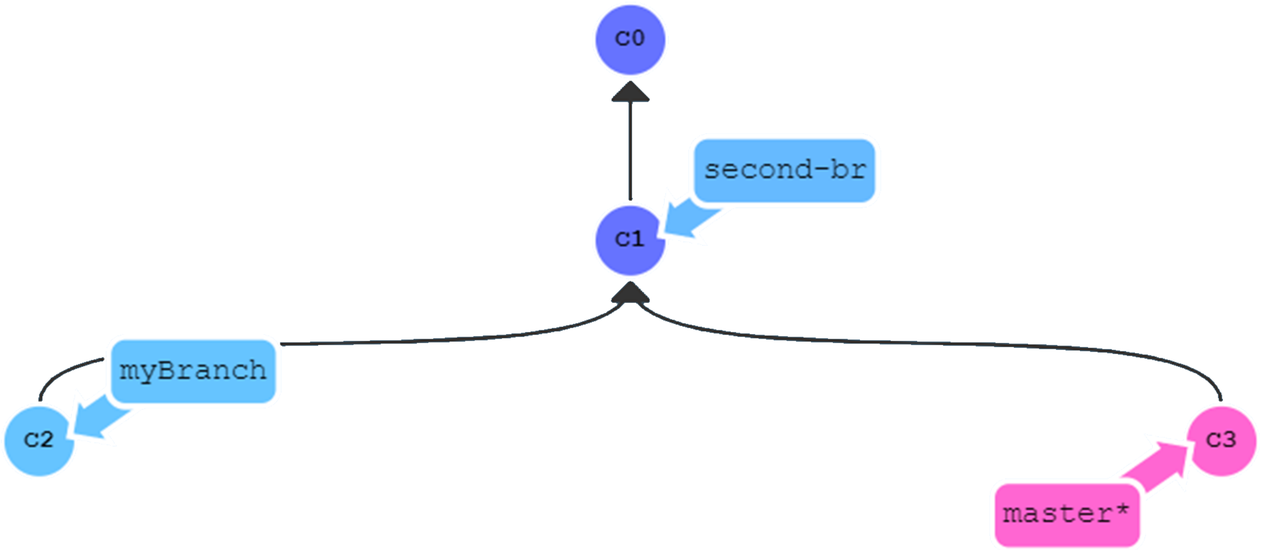
What is a commit ?

- Git commit -> Record any changes ("delta") since the previous version of the repository.
- Commits are linked to one or more parents

Commits and branches

What is a branch ?

- A commit reference
- Each branch have his own staged files



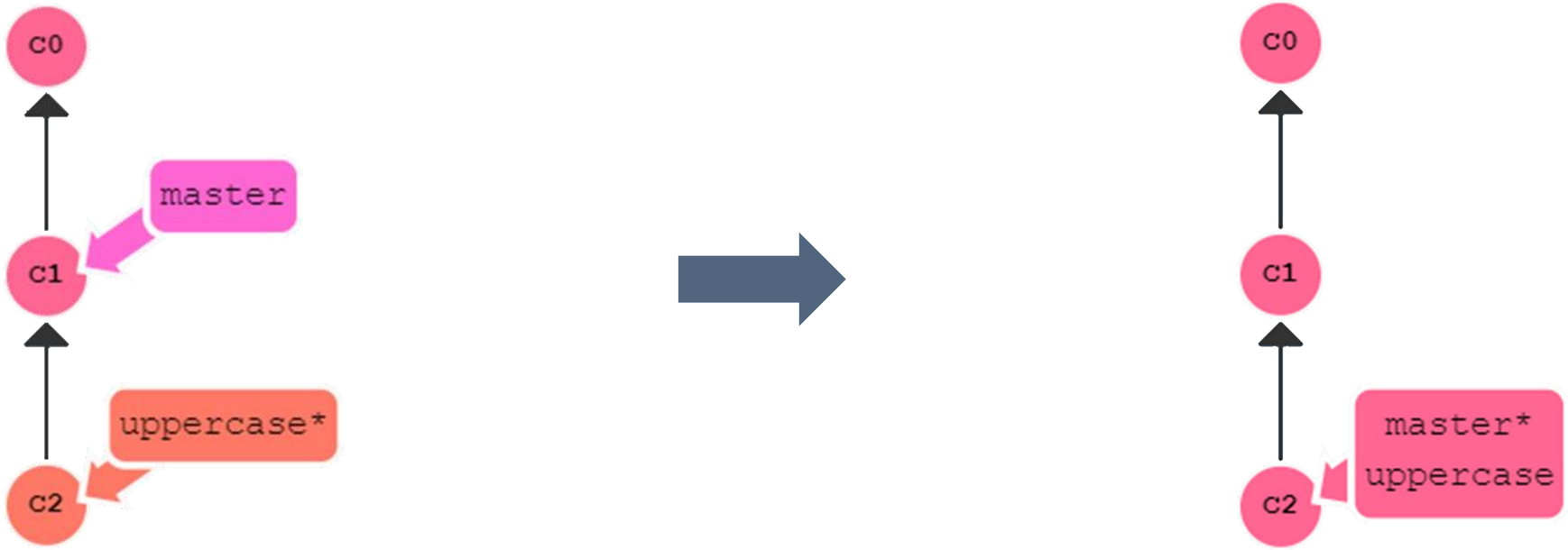

```
* 215c055 (HEAD -> master) Add file2.txt
| * 94e5294 (myBranch) Add file1.txt
|/
* a86667e (second-branch) dummy commit
```

3. Merge strategy

Merge strategy

Fast forward merge

Exemple: Merge uppercase branch into master




```
* 53941b7 (HEAD -> uppercase) Add uppercase greeting
* 258d2ec (master) Add content to greeting.txt
* 8b1fa91 Add file greeting.txt
```

```
* 53941b7 (HEAD -> master, uppercase) Add uppercase greeting
* 258d2ec Add content to greeting.txt
* 8b1fa91 Add file greeting.txt
```

Merge strategy

Merge without fast foward

1. Let’s take our previous example
2. Create a no-ff branch

```
* 1d77627 (no-ff) Add something
* 53941b7 (HEAD -> master, uppercase) Add uppercase greeting
* 258d2ec Add content to greeting.txt
* 8b1fa91 Add file greeting.txt
```

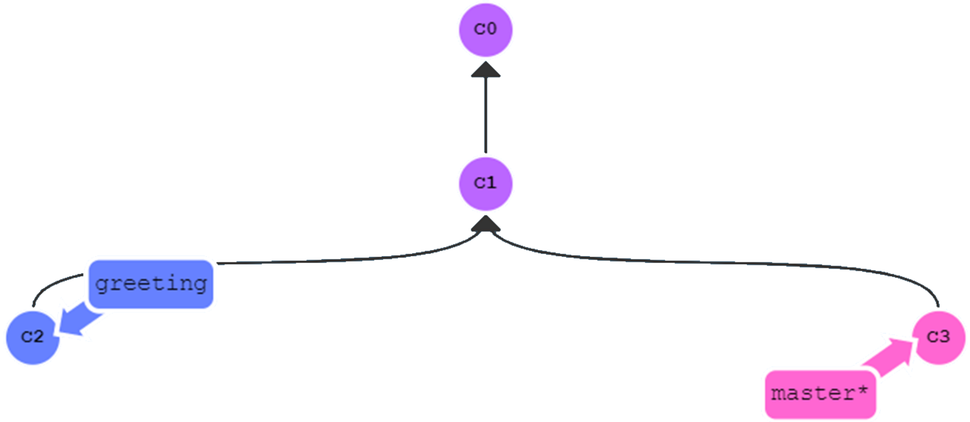
git merge –no-ff no-ff

```
*   18298df (HEAD -> master) Merge branch 'no-ff'
|\
| * 1d77627 (no-ff) Add something
|/
* 53941b7 (uppercase) Add uppercase greeting
* 258d2ec Add content to greeting.txt
* 8b1fa91 Add file greeting.txt
```

Merge strategy

Git merge

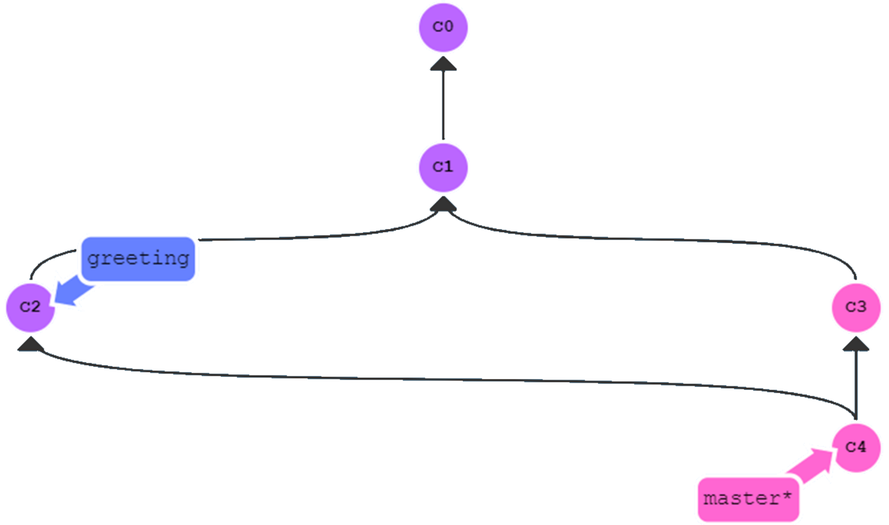
```
* 48689e4 (HEAD -> master) Add README
| * 41d240c (greeting) Add greeting
|/
* 32f6a20 Add content to greeting.txt
* 259b794 Add file greeting.txt
```



In **master** branch :

- *git merge greeting*

```
*   cb0fd78 (HEAD -> master) Merge branch 'greeting'
|\
| * 41d240c (greeting) Add greeting
* | 48689e4 Add README
|/
* 32f6a20 Add content to greeting.txt
* 259b794 Add file greeting.txt
```



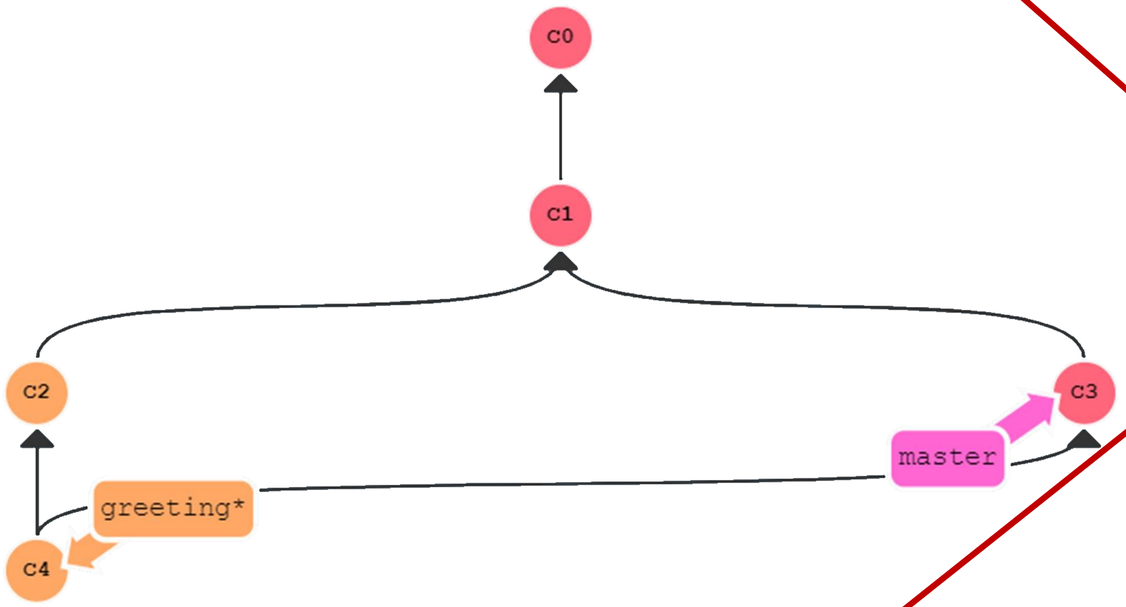
Merge strategy

Git merge – DO NOT

DO NOT merge master into private branch

Dans **greeting**

- *git merge master*

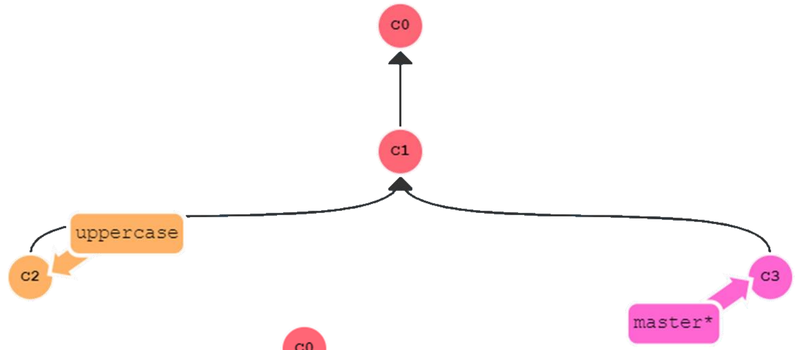



```
*   14b9b84 (HEAD -> greeting) Merge branch 'master' into greeting
|\
| * 48689e4 (master) Add README
* | 41d240c Add greeting
|/
* 32f6a20 Add content to greeting.txt
* 259b794 Add file greeting.txt
```

Merge strategy

Git rebase

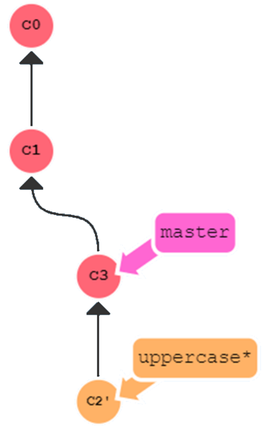
```
* 4cdfbaa (HEAD -> master) Add readme
| * b5d74f0 (uppercase) Change greeting to uppercase
|/
* 29abf6c Add content to greeting.txt
* b735996 Add file greeting.txt
```



git rebase master uppercase

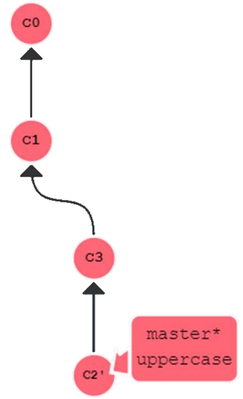
OR

git checkout uppercase; git rebase master




```
* 79ea525 (HEAD -> uppercase) Change greeting to uppercase
* 4cdfbaa (master) Add readme
* 29abf6c Add content to greeting.txt
* b735996 Add file greeting.txt
```

git checkout master; git merge uppercase

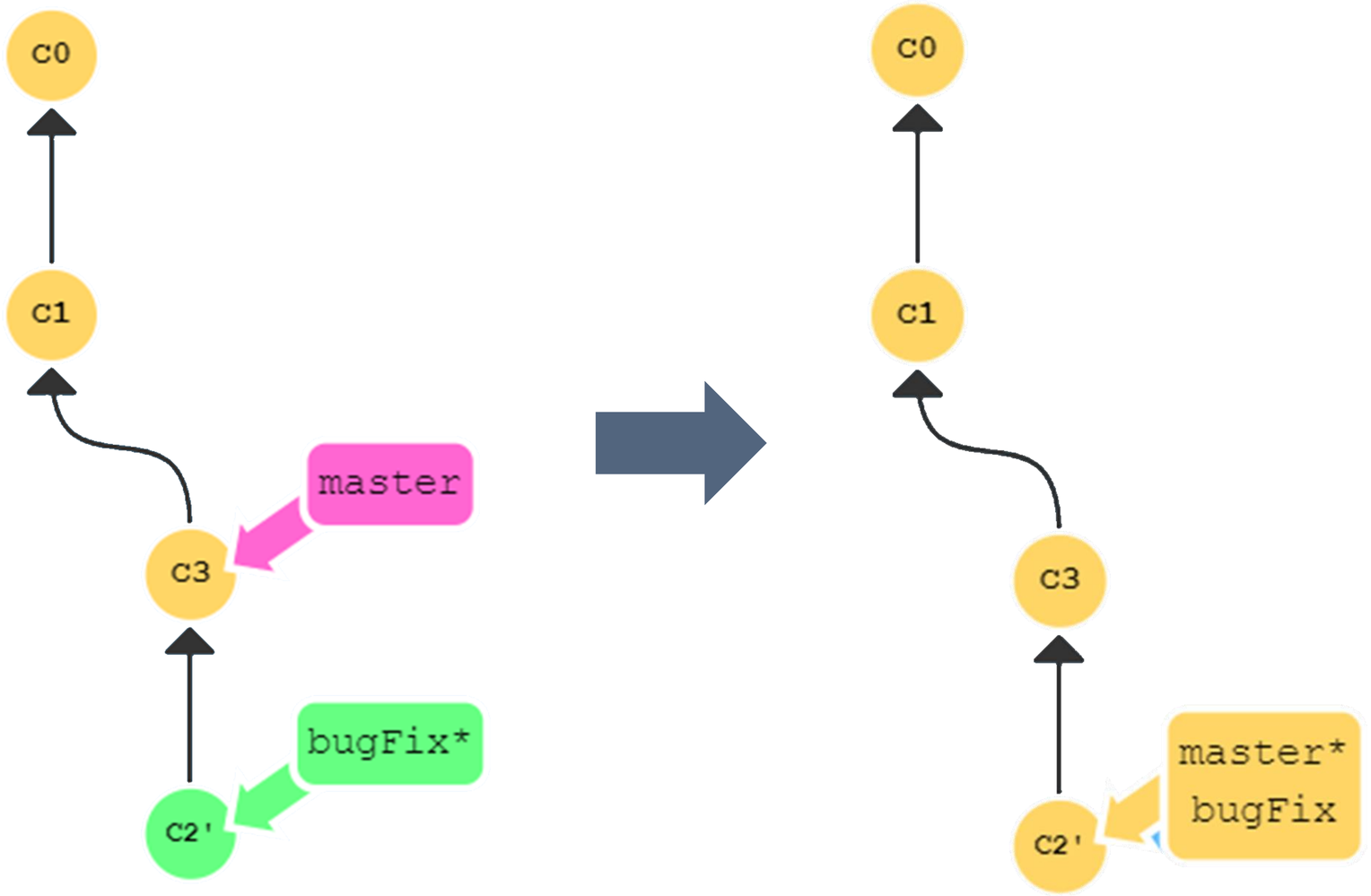



```
* 79ea525 (HEAD -> master, uppercase) Change greeting to uppercase
* 4cdfbaa Add readme
* 29abf6c Add content to greeting.txt
* b735996 Add file greeting.txt
```

Merge strategy

Rebase – update master

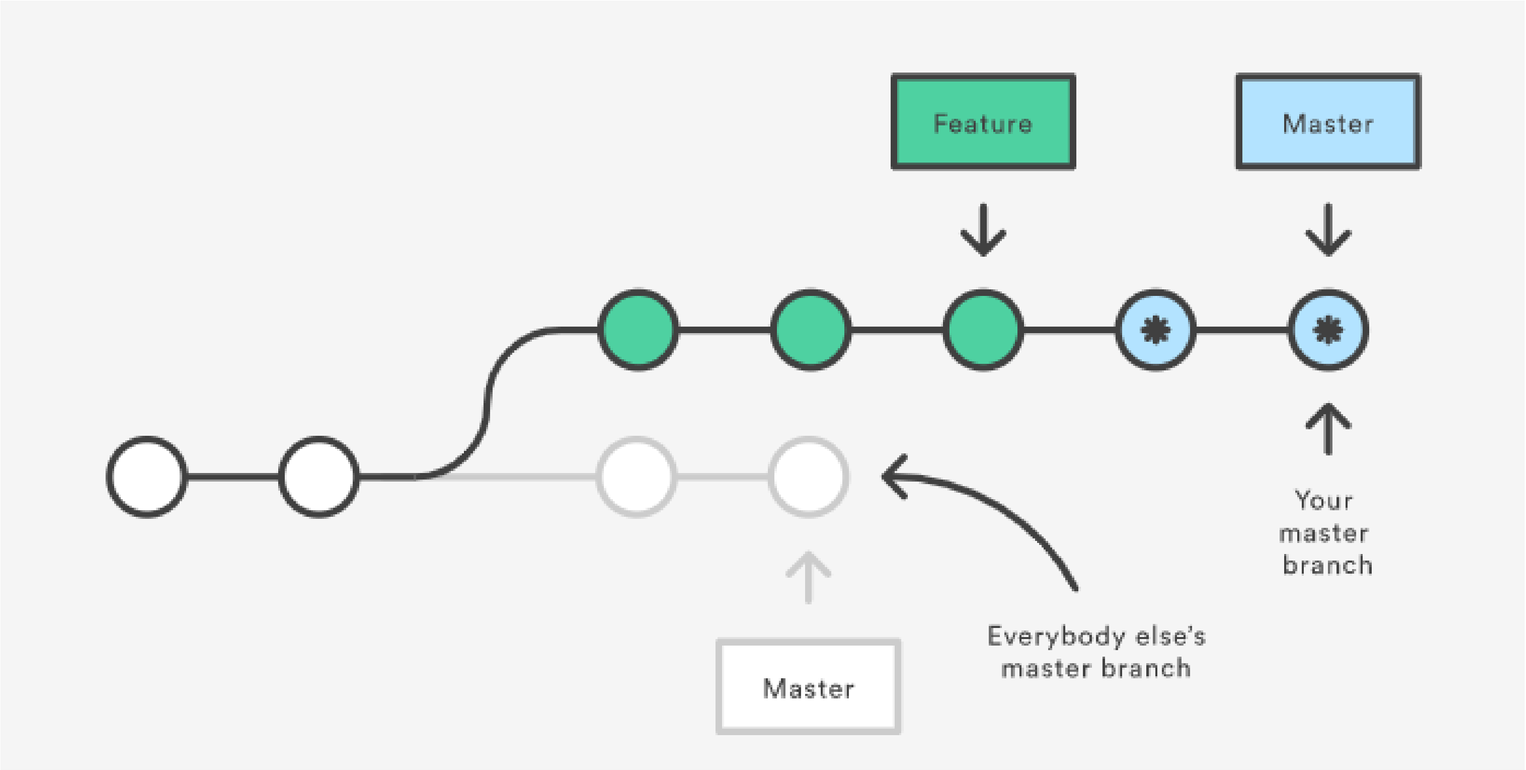
- In **master** branch
  - ➢ *Git merge bugFix*
  - ➢ *Git rebase bugFix*
- In **bugFix** branch
  - ➢ *Git branch –f master*
- Anywhere
  - ➢ *Git rebase bugFix master*



Merge strategy

Rebase – DO NOT

DO NOT rebase strategy in public branches



Merge strategy

Merge VS Rebase

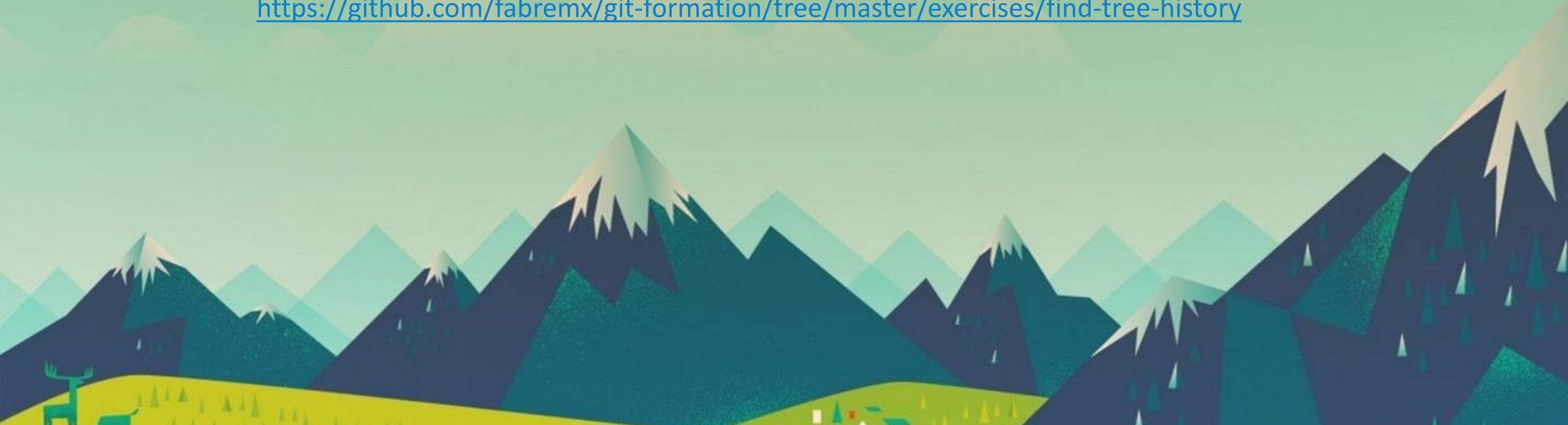
Preferences differ by developers

- **Rebase**: modifies the history but ensures a linear and clean commits tree
- **Merge**: keep the history but create a merge commit and don't guarantee a clean commits tree

Let’s Practice

Find tree history

https://github.com/fabremx/git-formation/tree/master/exercises/find-tree-history



4. Move into commits

Move into commits

What is HEAD ?

- HEAD is the symbolic name for the commit where we are currently on
- HEAD always points to the most recent commit in the commits tree.

Move into commits

Relative reference

- Relative references: Allows you to move easily from one commit to another commit (git checkout)

➢ Come back to a previous commit with ^
  ❖Exemple: master^^

➢ Come back to several commit with ~<num>
  ❖Exemple: master~3

Move into commits

Move branch reference

Git branch –f <nomDeLaBranche>

- Move branch on HEAD

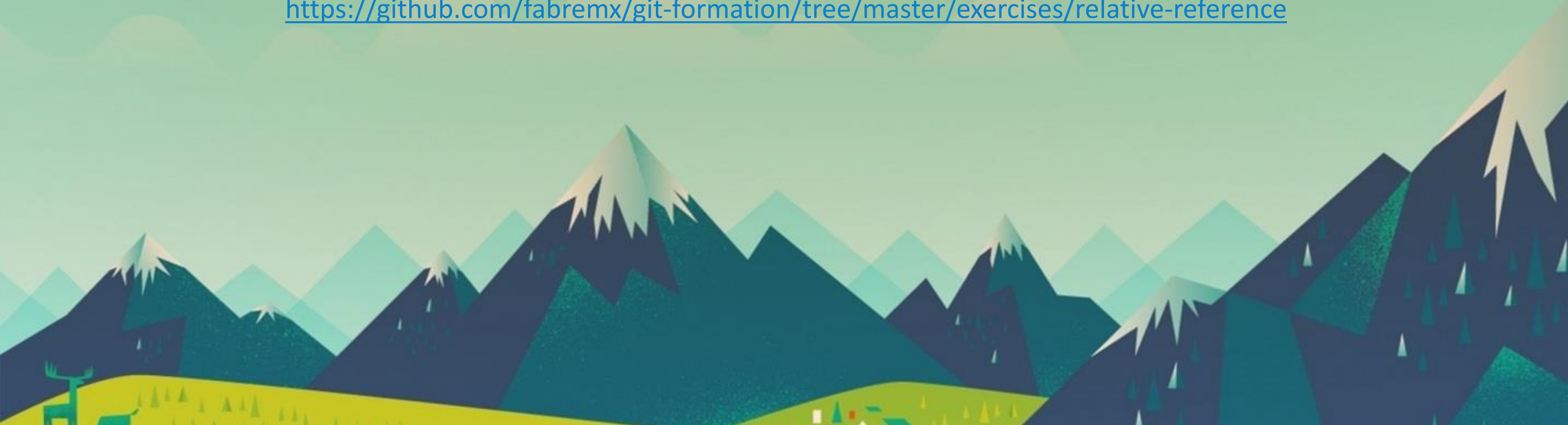
Git branch –f <NomDeLaBranche> <NomDuCommit>

- Move branch on commit

Let’s Practice

Relative reference

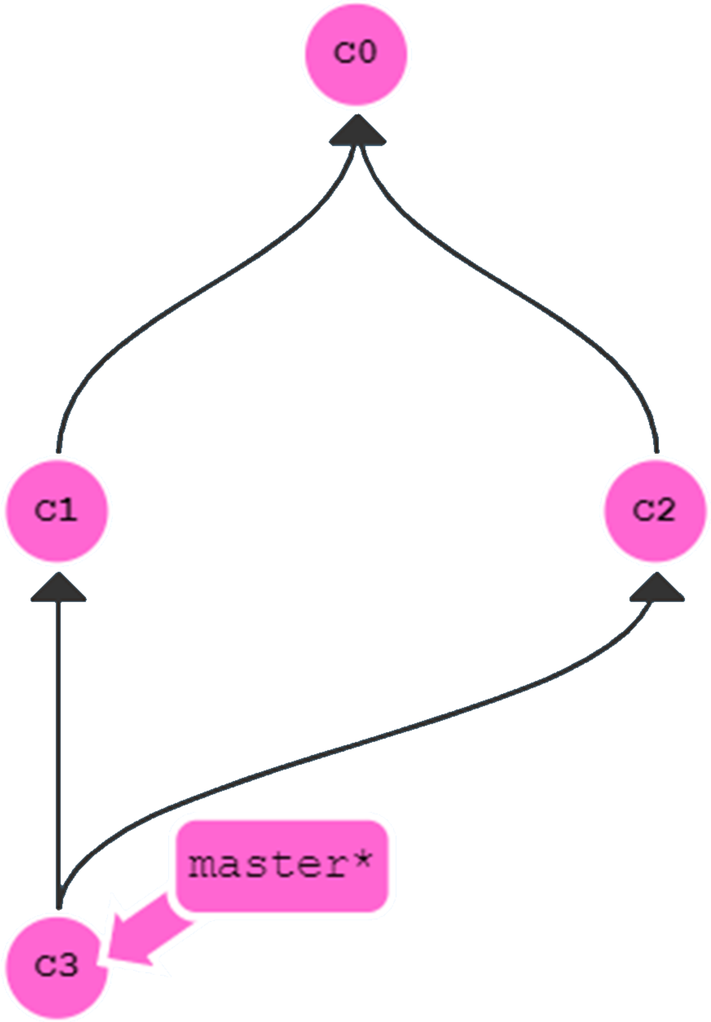
https://github.com/fabremx/git-formation/tree/master/exercises/relative-reference



Move into commits

Advanced concepts

- git checkout master^
  - First parent
- git checkout master^2
  - Second parent



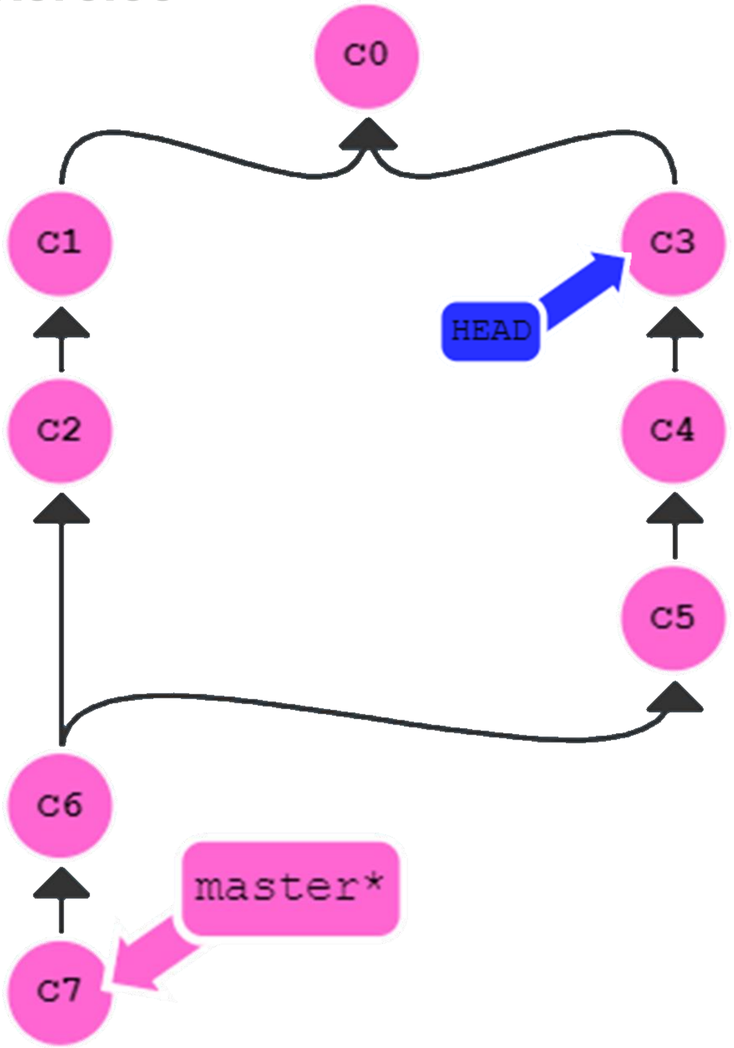
Move into commits

Advanced concepts - Exercice

I use the command line bellow

git checkout HEAD~^2~2

Where HEAD will be in the commits tree after this command ?



5. Revert vs Reset

Reset vs Revert

- **Reset**: cancel a commit
- **Revert**: create a new commit to undo the changes

Let’s Practice

Revert commit & Reset commit

https://github.com/fabremx/git-formation/tree/master/exercises/basic-revert

https://github.com/fabremx/git-formation/tree/master/exercises/basic-reset

Revert merge & reset rebase

6. Rewrite history

Rewrite history

Git amend

- **Replaces** that last commit with your new, improved commit

Usefull to override a WIP commit or change/update message in commit

Let’s Practice

Amend commit

https://github.com/fabremx/git-formation/tree/master/exercises/amend

Rewrite history

Cherry-pick

- git cherry-pick <Commit1> <Commit2> <...>
- Used to **copy** a series of commits below our current location (HEAD)

Usefull to copy paste commit of branch to another

Let’s Practice

Cherry-pick

https://github.com/fabremx/git-formation/tree/master/exercises/cherry-pick

Rewrite history

Interactive Rebase

git rebase –i HEAD~4

- **Move** a the sequence of commits between HEAD and HEAD~4 (Rewrite history) to HEAD ~4
- Allow before rebasing to :

  - Choose commits to move
  - Change commits messages
  - Merge commits ...

Usefull to move commits sequence of branch to another

Rewrite history

Interactive Rebase - Commands

`pick`

`pick` simply means that the commit is included. Rearranging the order of the `pick` commands changes the order of the commits when the rebase is underway. If you choose not to include a commit, you should delete the entire line.

`reword`

The `reword` command is similar to `pick`, but after you use it, the rebase process will pause and give you a chance to alter the commit message. Any changes made by the commit are not affected.

`edit`

If you choose to `edit` a commit, you'll be given the chance to amend the commit, meaning that you can add or change the commit entirely. You can also make more commits before you continue the rebase. This allows you to split a large commit into smaller ones, or, remove erroneous changes made in a commit.

`squash`

This command lets you combine two or more commits into a single commit. A commit is squashed into the commit above it. Git gives you the chance to write a new commit message describing both changes.

`fixup`

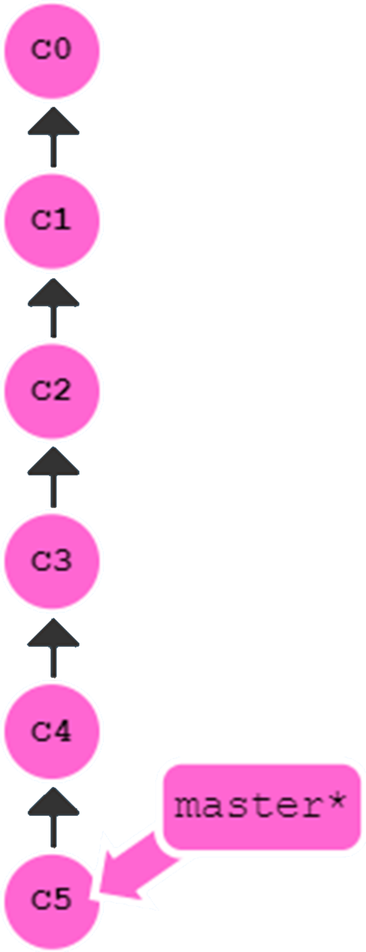
This is similar to `squash`, but the commit to be merged has its message discarded. The commit is simply merged into the commit above it, and the earlier commit's message is used to describe both changes.

`exec`

This lets you run arbitrary shell commands against a commit.

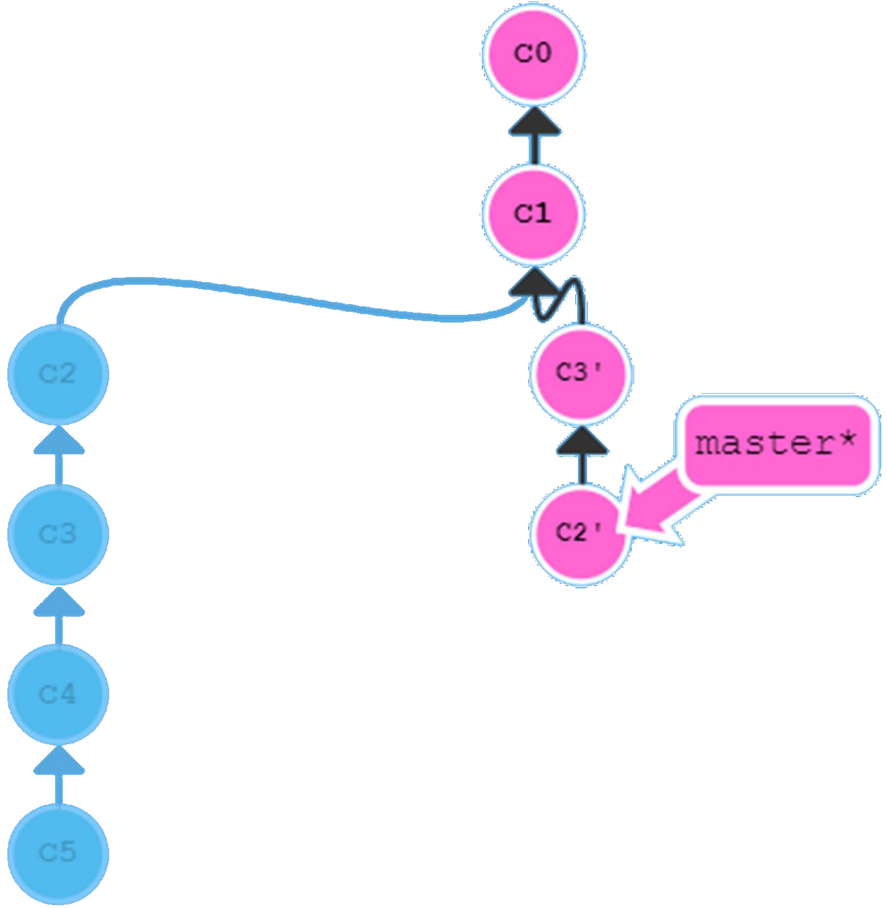
Rewrite history

Interactive Rebase – Case n°1



git rebase –i HEAD ~4

- Pick c3
- Pick c2



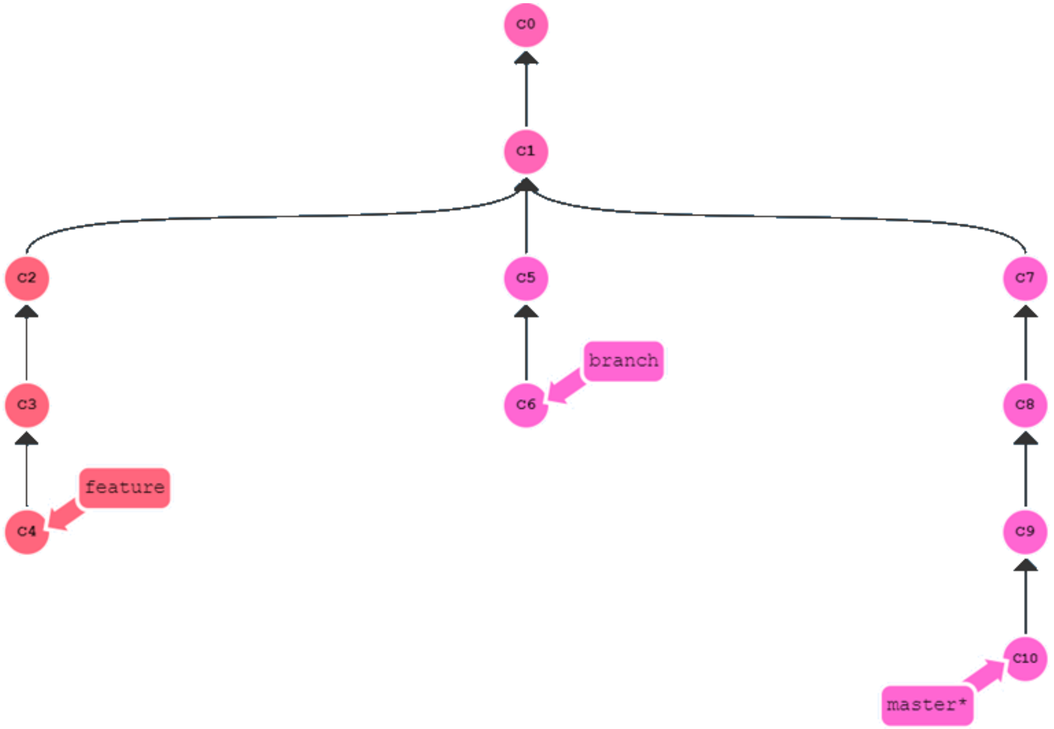
Let’s Practice

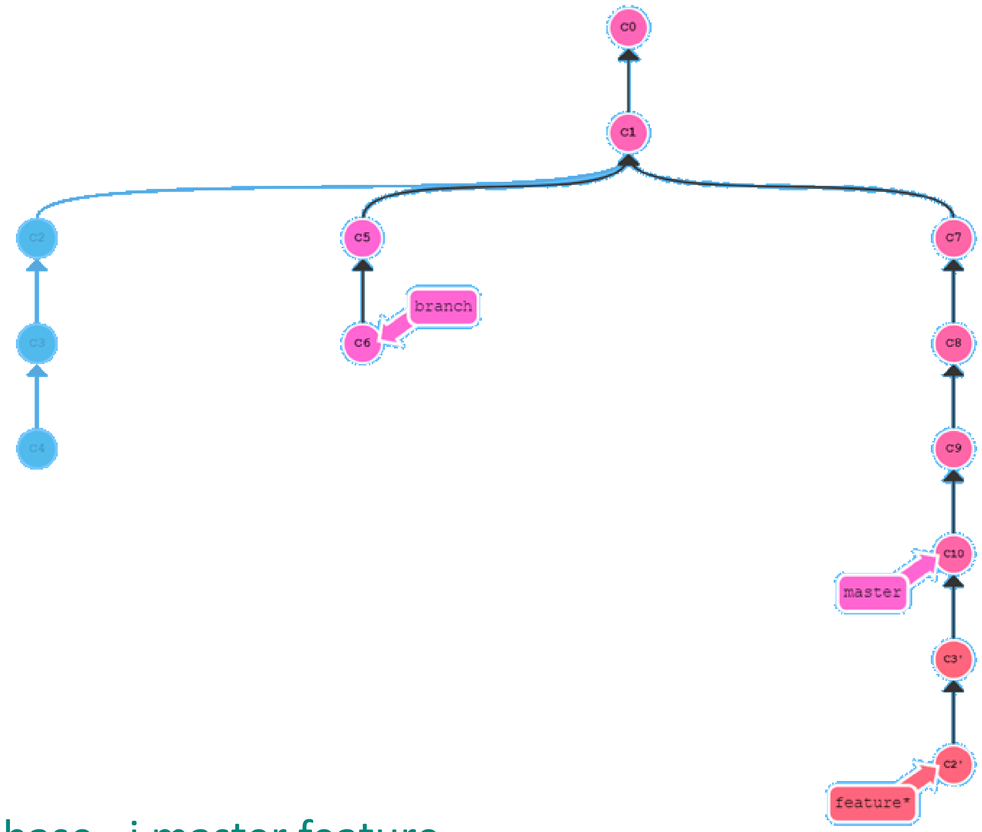
Interactive Rebase

https://github.com/fabremx/git-formation/tree/master/exercises/interactive-rebase

Rewrite history

Interactive Rebase – Case n°2





git checkout feature
git rebase –i master

- Pick c3
- Pick c2

git rebase –i master feature

- Pick c3
- Pick c2

7. Origin and remotes branches

Origin and remote branches

What means origin ?

```
fabremx@FR-L2870047 MINGW64 ~/Projets/Chatbot/Generique/Dev/ChatBot_Client (master)
$ git branch
  BSR_remove-client-scss
  TEST_branch
  develop
* master
```

```
fabremx@FR-L2870047 MINGW64 ~/Projets/Chatbot/Generique/Dev/ChatBot_Client (master)
$ git branch -a
  BSR_remove-client-scss
  TEST_branch
  develop
* master
  remotes/origin/307-logique-starting
  remotes/origin/BSR_Accessibility-audit
  remotes/origin/BSR_remove-client-scss
  remotes/origin/FEAT_Add--zoom-in/zoom-out--functions
  remotes/origin/FEAT_ent-front
  remotes/origin/FEAT_update-the-project-for-prod-build
  remotes/origin/FIX_correct-notation-module
  remotes/origin/FIX_notation-par-company
  remotes/origin/HEAD -> origin/master
  remotes/origin/TEST_branch
  remotes/origin/ameli
  remotes/origin/develop
  remotes/origin/master
  remotes/origin/tmp
```

Origin and remote branches

Remote branches

- The origin/... branches are called: remote branches
- Remote branches reflect the status of remote repositories (since the last time you spoke to them).
- They help you understand the differences between your work and public work
- Remote branches have a special property: when you go in (checkout), HEAD is detached.

8. Fetch vs Pull

Fetch VS Pull

Git Fetch

1. Download commits on remote repository
2. Update remotes branches

Fetch doesn’t change your local repo

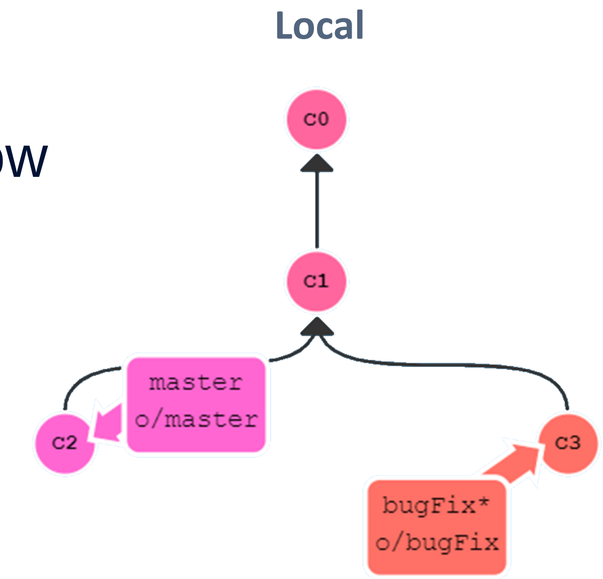
Fetch VS Pull

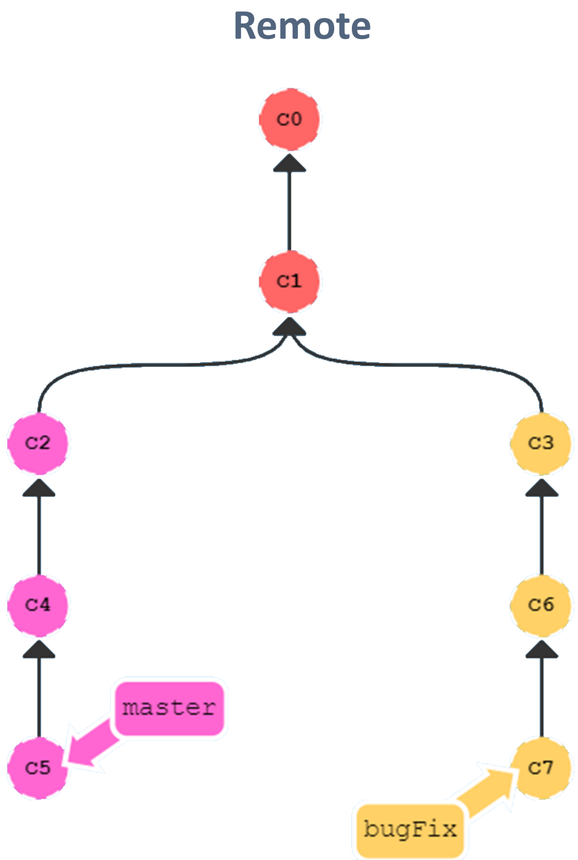
Git Fetch - Exercise

I use the command line bellow

git fetch

What will be happening?

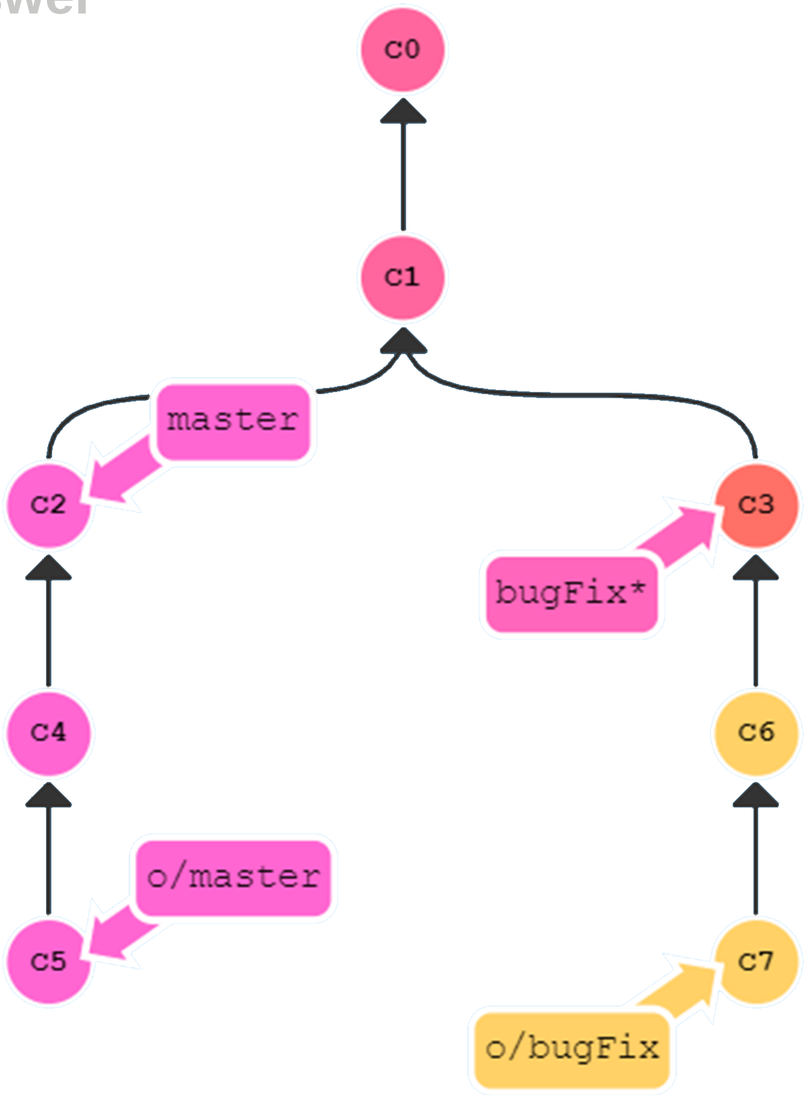




Fetch VS Pull

Git Fetch - Answer

How merge my work with remote work downloaded previously ?



Fetch VS Pull

Git pull

- Fetch download remote commits
- Now I want to merge remote work with my local work

Pull = fetch + merge

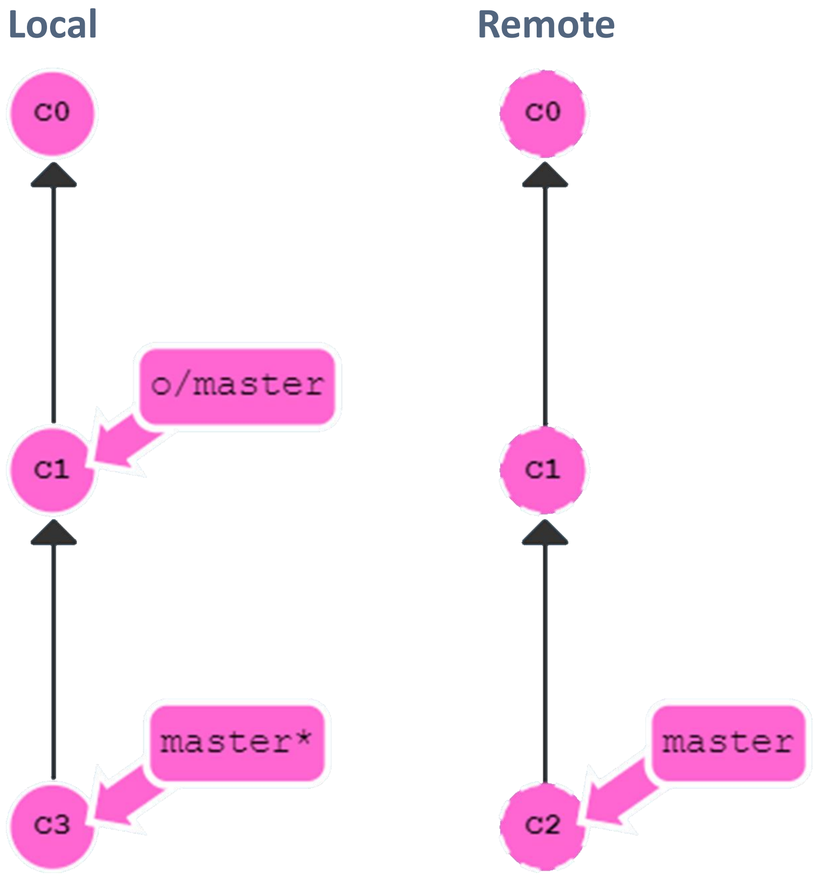
Fetch VS Pull

Working with updated remote repository

Exemple

Try to pull with merge streategy and rebase strategy

Try to guess what will be hapening ?



Let’s Practice

Learngibranching - Excercie 2:1

8. Tracking branches

Tracking branches

Tracking remote branches

- By default master follow origin/master
- You can configure this by yourself

git checkout -b foo o/master

git branch -u o/master foo

➢ In this case each commit on foo will update master branch on remote repo after push

Let’s Practice

Learngibranching - Excercie 2:3

To be continued …

8. Good practices

Good practices

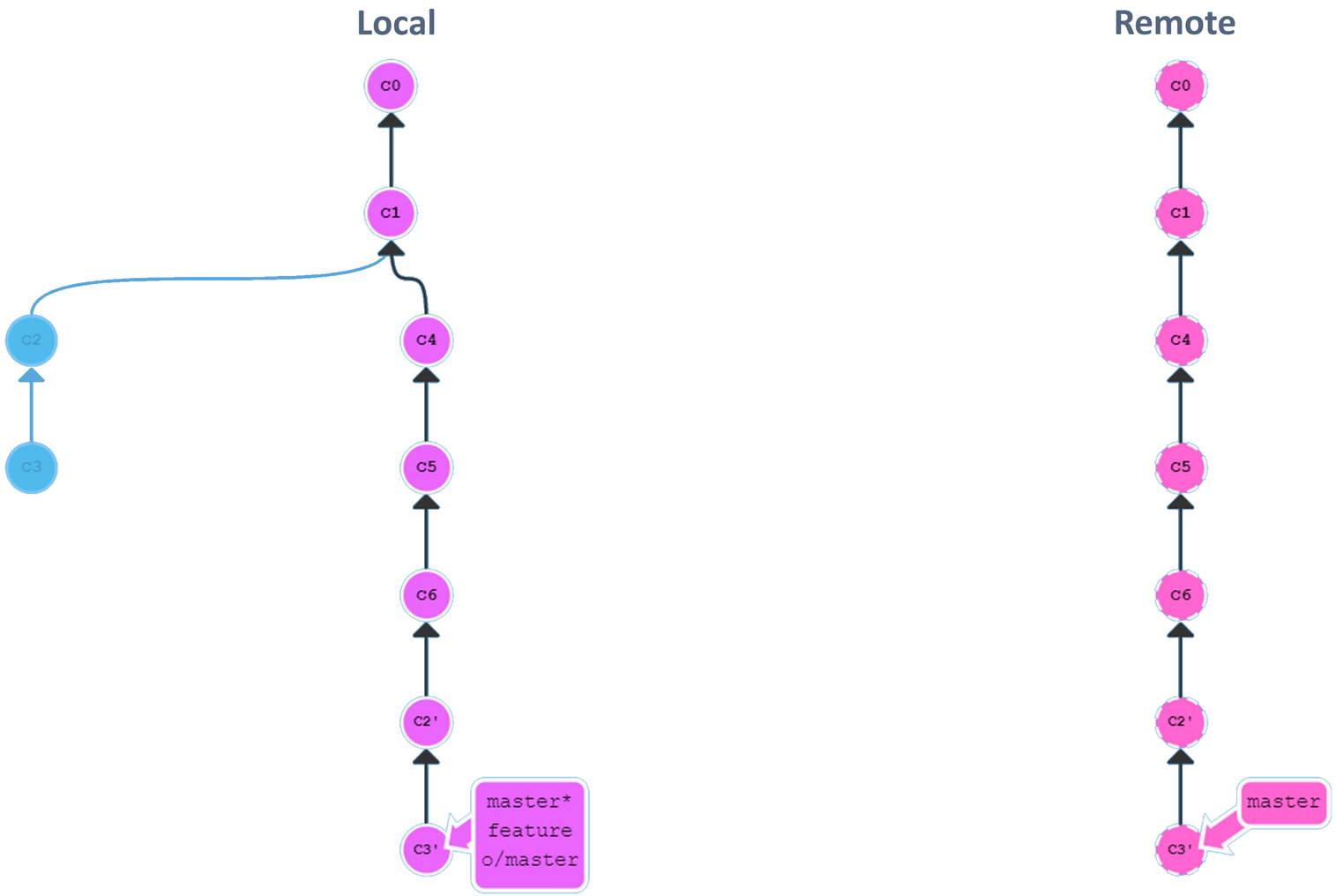
1. **Make clean, single-purpose commits**
2. **Write meaningful commit messages**

```
feat: add beta sequence
^--^  ^---------------^
|     |
|     +-> Summary in present tense.
|
+-------> Type: chore, docs, feat, fix, refactor, style, or test.
```

3. **Commit early, commit often**
4. **Don’t alter published history**
   - Use git-rebase only on branches that only you are working with.
5. **Don’t commit generated files**
   - Use gitignore

Good practices

Use Rebase Workflow



11. Let’s practice with scenarised exercises

Let’s Practice

Commit on wrong branch

https://github.com/fabremx/git-formation/tree/master/exercises/commit-on-wrong-branch

Let’s Practice

Reverted merge

https://github.com/fabremx/git-formation/tree/master/exercises/reverted-merge

Let’s Practice

Wrong commit on branch #2

https://github.com/fabremx/git-formation/tree/master/exercises/commit-on-wrong-branch-2