Git y GitHub

Software para control de versiones

Difereciar git de github
Instalacion?
Para que usar GitHub? Por que?

Paso 1: clonar un repositorio

$ git clone https://github.com/cnegrelli/charla_git_journal_fcaglp.git

- Chequear que no les de error
- Vamos a usar estos archivos más adelante

https://github.github.com/training-kit/downloads/es_ES/github-git-cheat-sheet.pdf

Curso udacity: https://www.udacity.com/course/version-control-with-git--ud123

Git: Control de versiones

Check point -> commit

https://git-scm.com/

https://github.com/

Iniciar un repositorio (carpeta git)

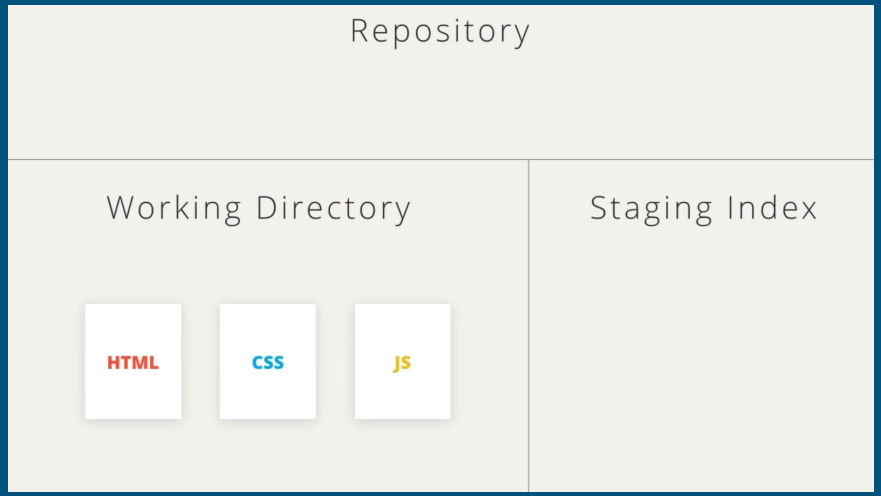
$ mkdir git_prueba

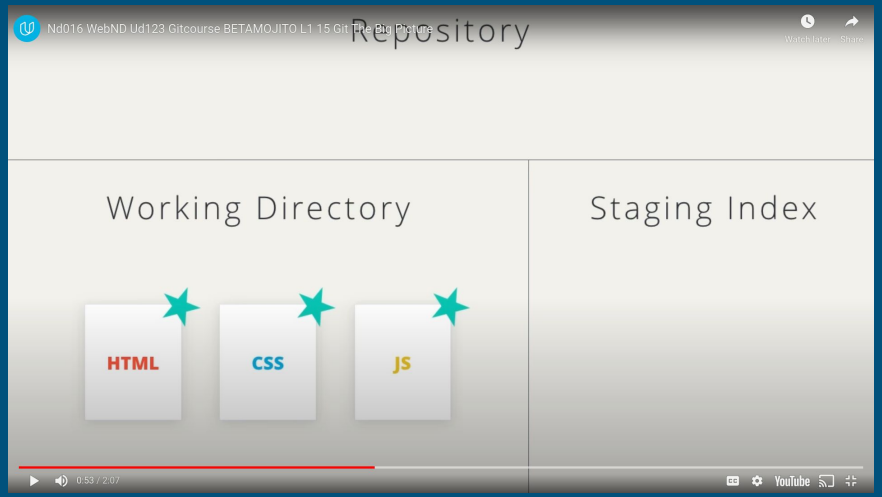
$ cd git_prueba

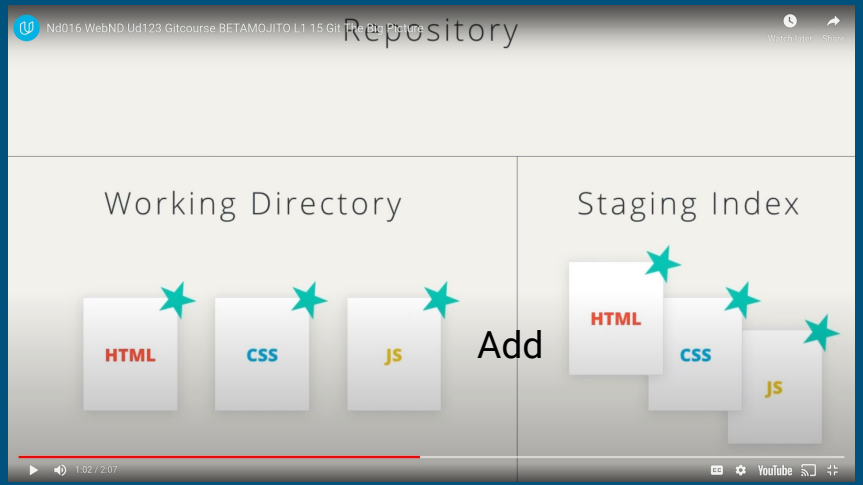
$ git init ----> crea una carpeta ‘.git’ que no hay que manipular

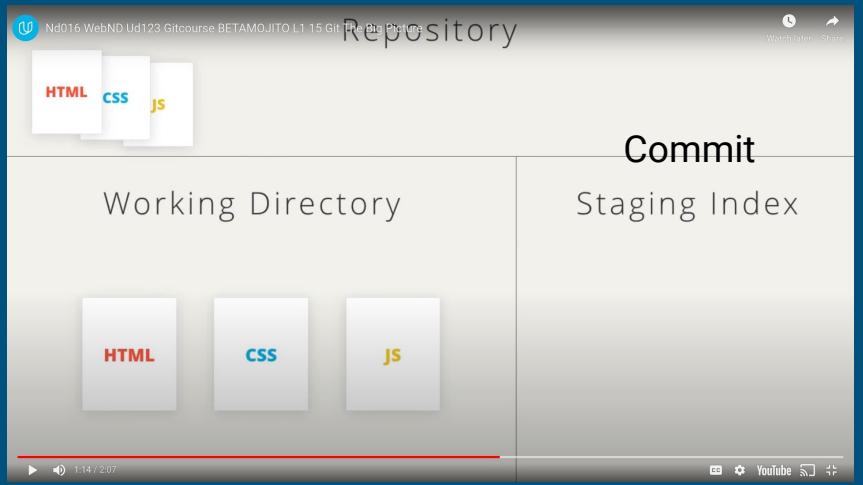
$ git status ----> para tener información del estado del repositorio

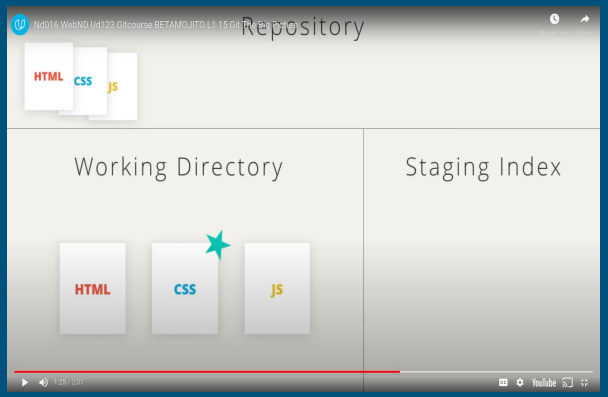
Nota: NO se pueden tener repositorios anidados

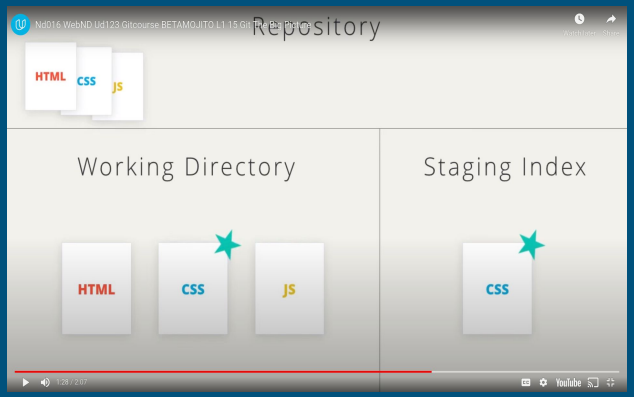

Repository
Working Directory
Staging Index
HTML
CSS
JS

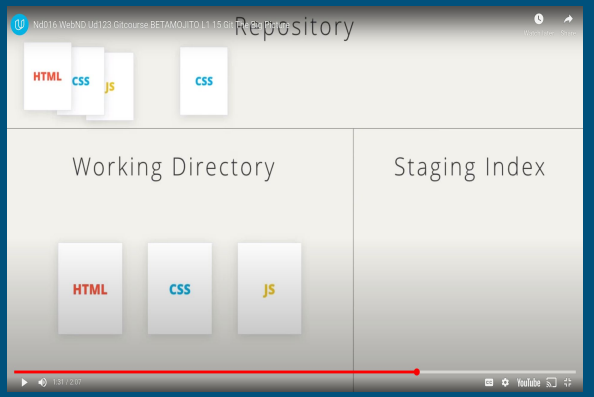

Nd016 WebND Ud123 Gitcourse BETAMOJITO L1 15 Git The Big Picture
Repository
Working Directory
Staging Index
HTML
CSS
JS
0:53 / 2:07
YouTube

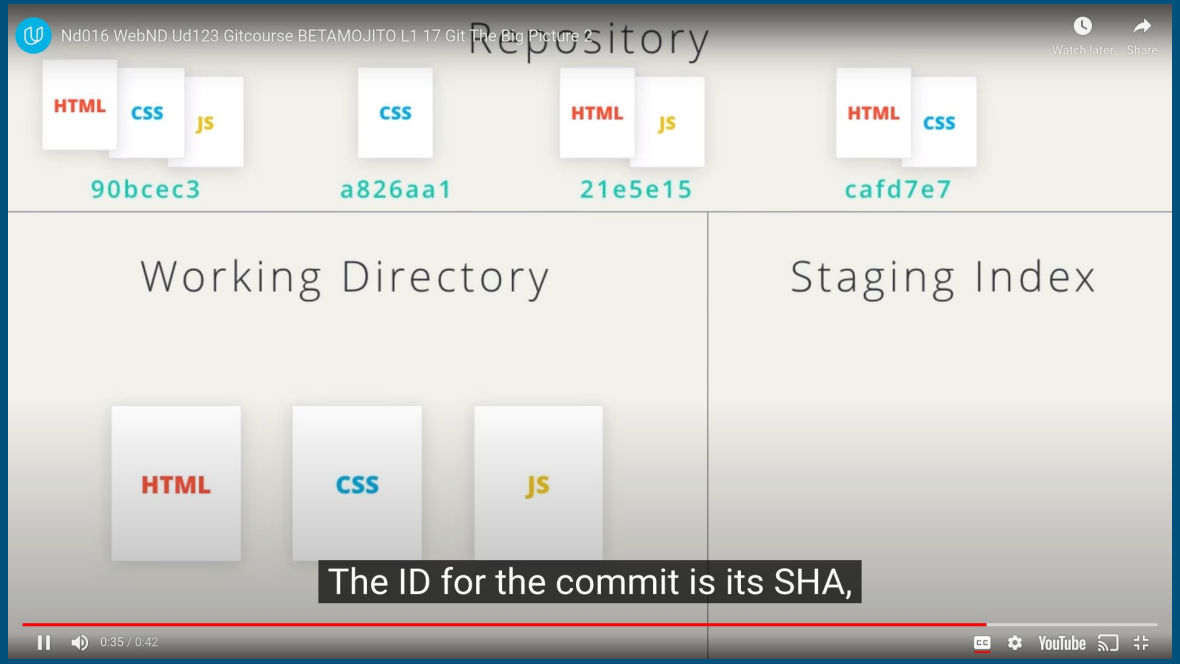

Nd016 WebND Ud123 Gitcourse BETAMOJITO L1 15 Git The Big Picture
Repository
Working Directory
Staging Index
HTML
CSS
JS
Add
HTML
CSS
JS
1:02 / 2:07
YouTube


Nd016 WebND Ud123 Gitcourse BETAMOJITO L1 15 Git The Big Picture
Repository
HTML
CSS
JS
Commit
Working Directory
Staging Index
HTML
CSS
JS
1:14 / 2:07
YouTube


Repository
HTML
CSS
JS
Working Directory
Staging Index
HTML
CSS
JS


Repository
HTML
CSS
JS
Working Directory
Staging Index
HTML
CSS
JS
CSS


Repository
HTML
CSS
JS
CSS
Working Directory
Staging Index
HTML
CSS
JS


Nd016 WebND Ud123 Gitcourse BETAMOJITO L1 17 Git The Big Picture 2
Watch later
Share
Repository
HTML
CSS
JS
90bcec3
CSS
a826aa1
HTML
JS
21e5e15
HTML
CSS
cafd7e7
Working Directory
Staging Index
HTML
CSS
JS
The ID for the commit is its SHA,
0:35 / 0:42
YouTube

Hacer un commit

$ git status ---> muestra los archivos modificados y los nuevos

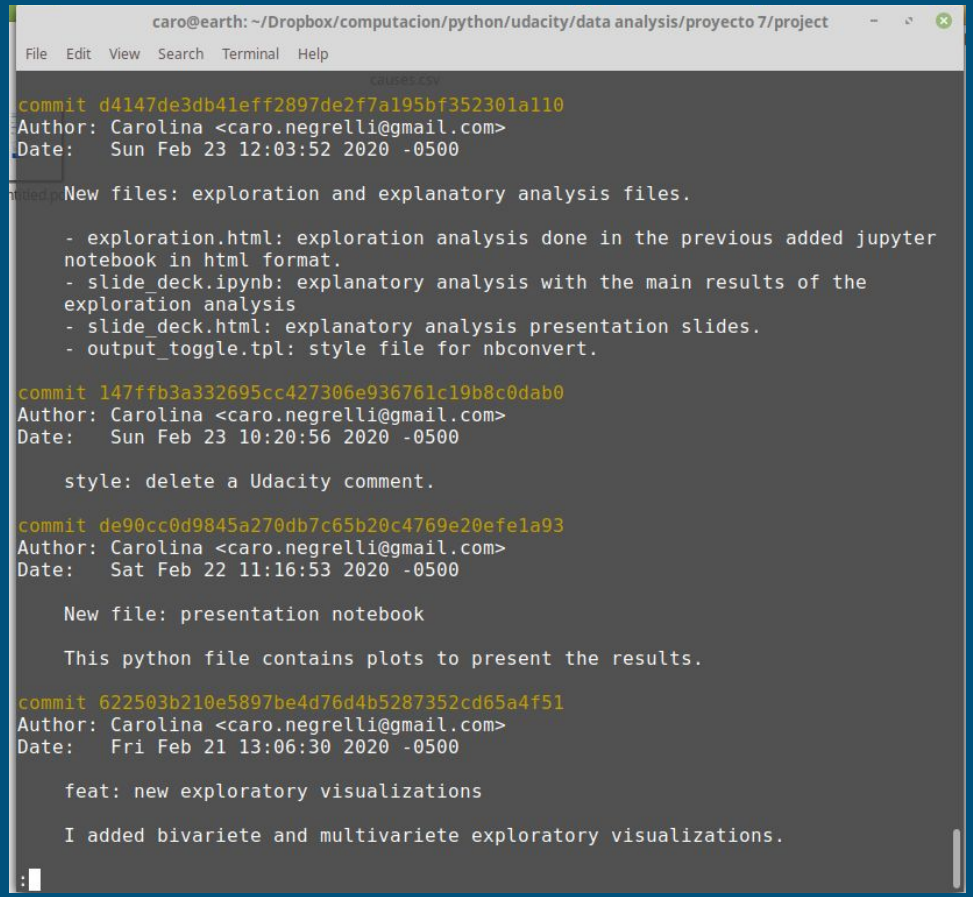
$ git add file_name1 file_name2 ---> agrega los archivos al staging index

$ git commit ---> crea un commit con todo lo que está en el staging index

(opcion: $ git commit -m ‘mensaje corto’)

Cómo escribir un buen commit? https://udacity.github.io/git-styleguide/

$ git log ----> información de todos los commit hechos hasta el momento



Flags

--oneline -> una linea por commit

--stat -> muestra los archivos

-p -> muestra los cambios

Nota: para ver un commit particular agregar el SHA al final
$ git log -p fdf5493
Otra opción:
$ git show fdf5493

Cuando hacer un commit?

- Idealmente: cada vez que se agrega/cambia algo significativo que está funcionando correctamente. Ej: una nueva función
- Cuando tenés un programa funcionando y estas por realizar cambios de los cuales no estas seguro y pueden romper todo.
- Al agregar nuevos archivos
- Antes de hacer una reestructuración

Usos de GitHub

- Colaboraciones
- Portfolio
- Red Social

Ramas y más

Tags

Las tags se usan para apuntar a un commit ‘especial’ que queremos que sobresalga.

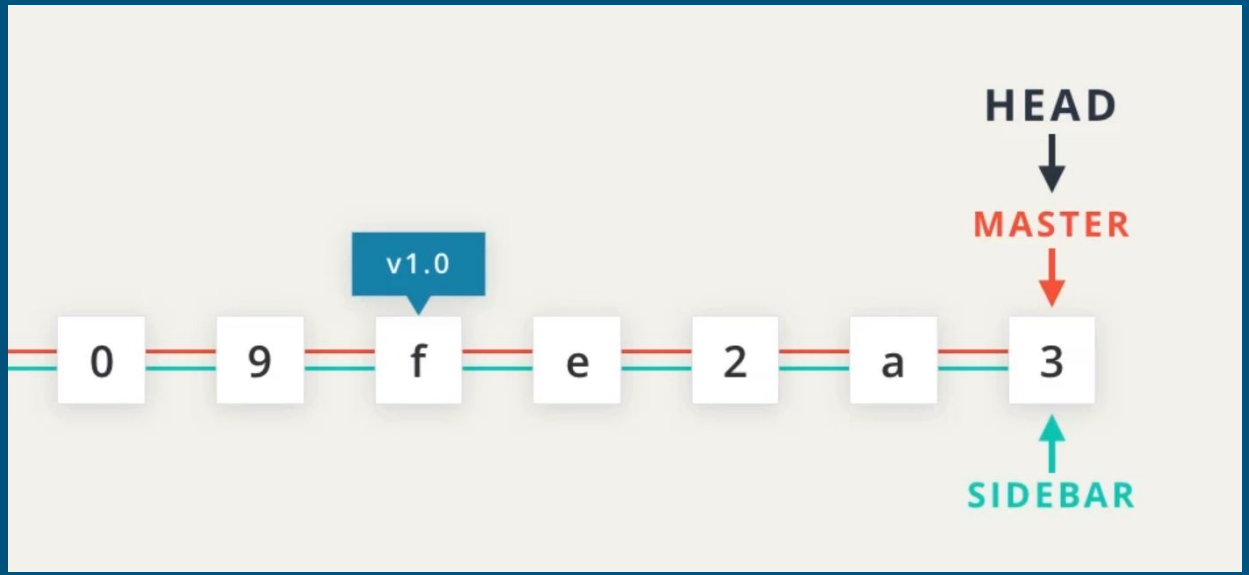
$ git tag -a V1.0 (SHA) ---> despliega el editor para anotar un mensaje

$ git tag ---> muestra todas las que hay

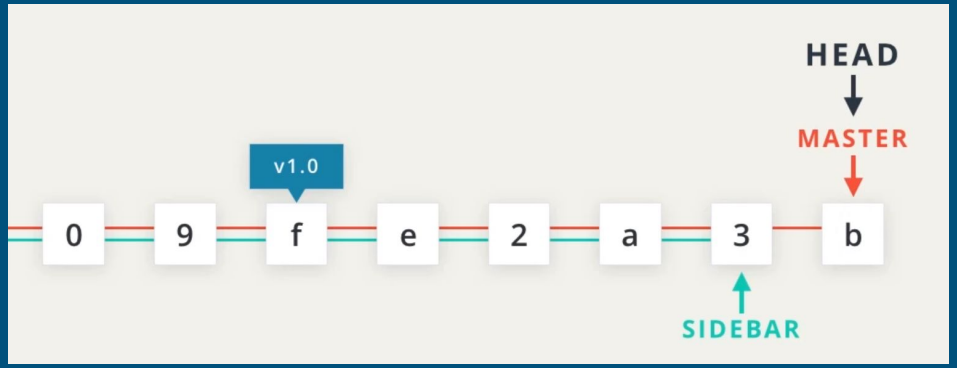
Ramas (branches)

Permite diferentes líneas de desarrollo.

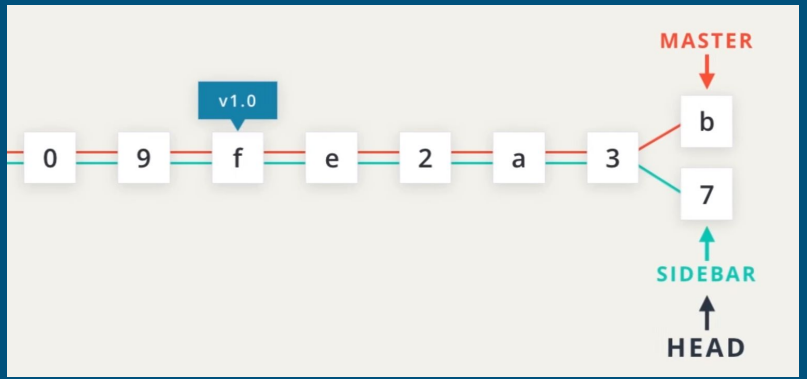
$ git branch sidebar ---> crea la branch sidebar



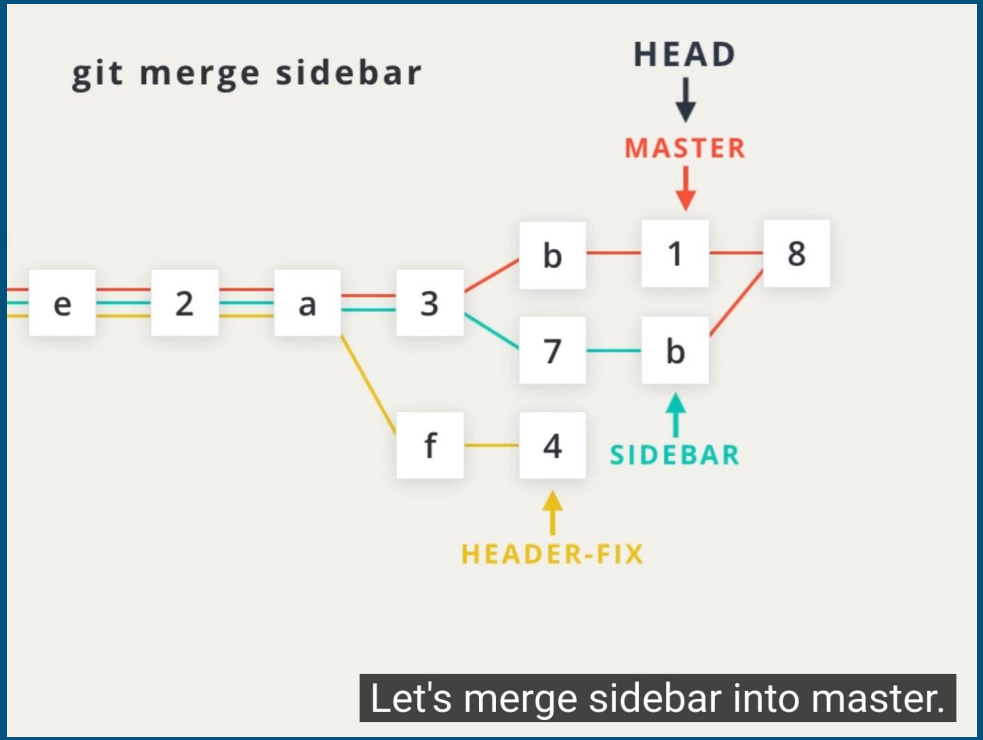
$ git branch ---> muestra la lista de branchs y marca con un * la actual



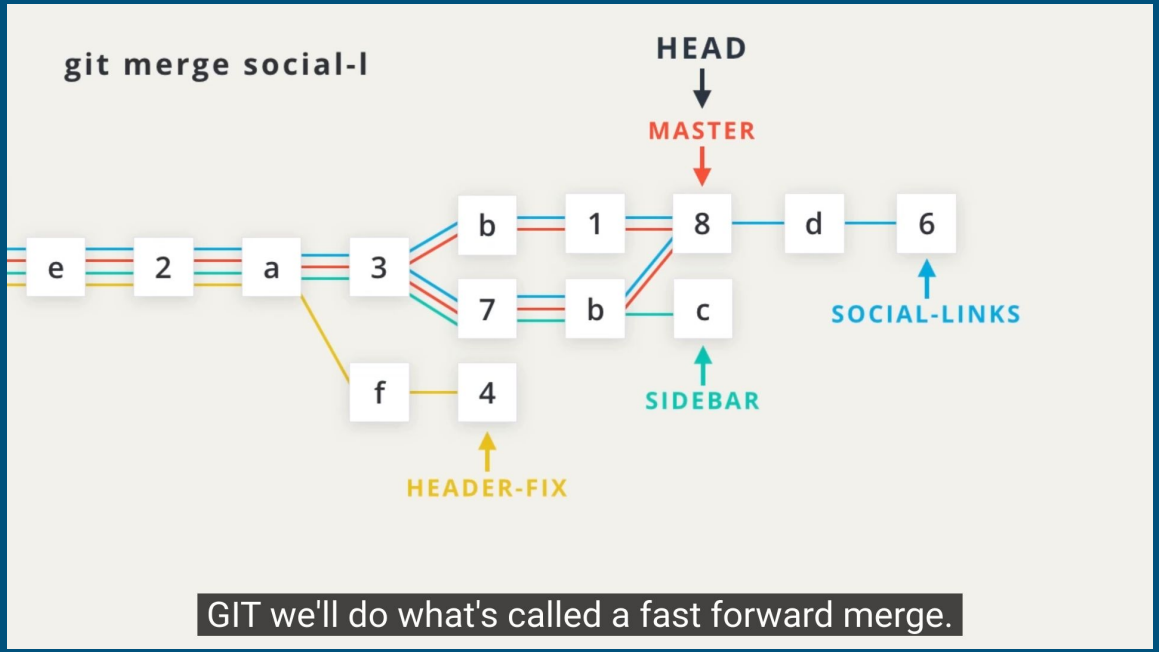
$ git checkout sidebar ---> activa la branch sidebar



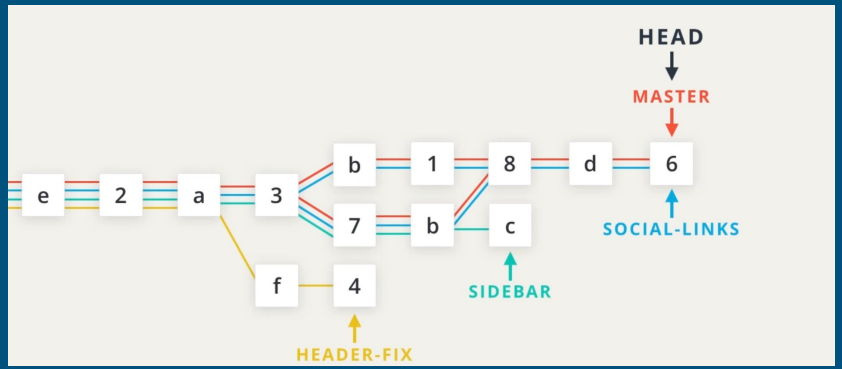
Merge -> queremos unir los cambios



Importante: los cambios se van a ver en la branch en la que estamos parados.



GIT we'll do what's called a fast forward merge.



- Fast-forward merge: son los más simples, una rama está adelantada a la otra.

- Regular merge: las dos ramas tienen distintos commits. Posibles conflictos: en las dos branchs se cambian las mismas líneas del mismo archivo.