



Git y GitHub



Software para control de versiones



Bibliografía

Cursos Udacity:

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

<https://classroom.udacity.com/courses/ud805> (Lesson 4)

Cheat-sheet:

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

Pre-pasos: clonar un repositorio

```
$ mkdir git_journal
```

```
$ cd git_journal
```

```
$ git clone https://github.com/cnegrelli/charla\_git\_journal\_fcaglp.git git_clonado
```

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

Git: Control de versiones

‘Un control de versiones es un sistema que registra los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante.’

Unidad fundamental: commit (check-point)



Version control tool



Service that hosts
Git projects



0:54 / 1:20



YouTube



<https://git-scm.com/>

<https://github.com/>

Git



Iniciar un repositorio (carpeta git)

```
$ cd git_journal
```

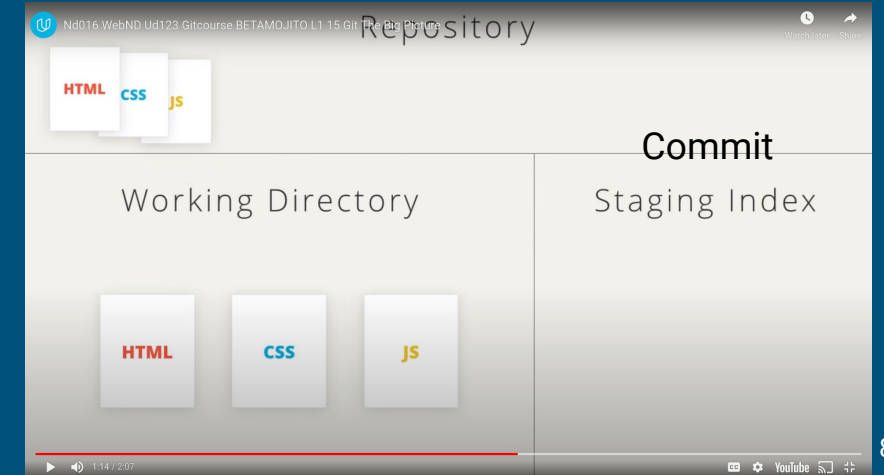
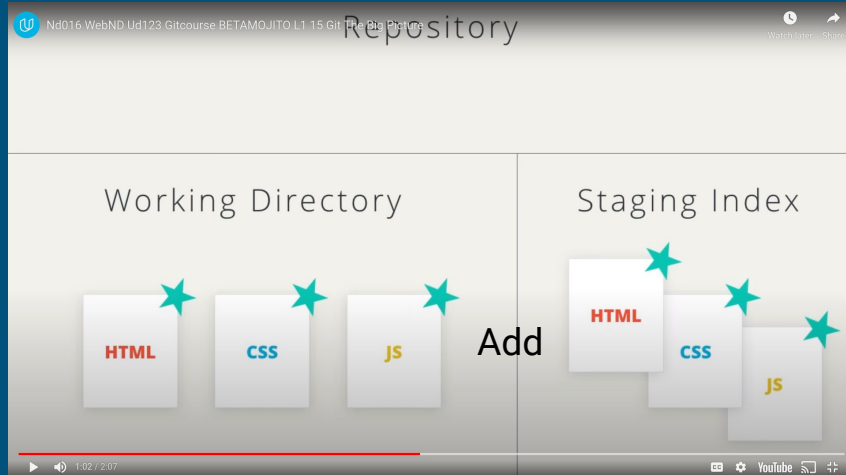
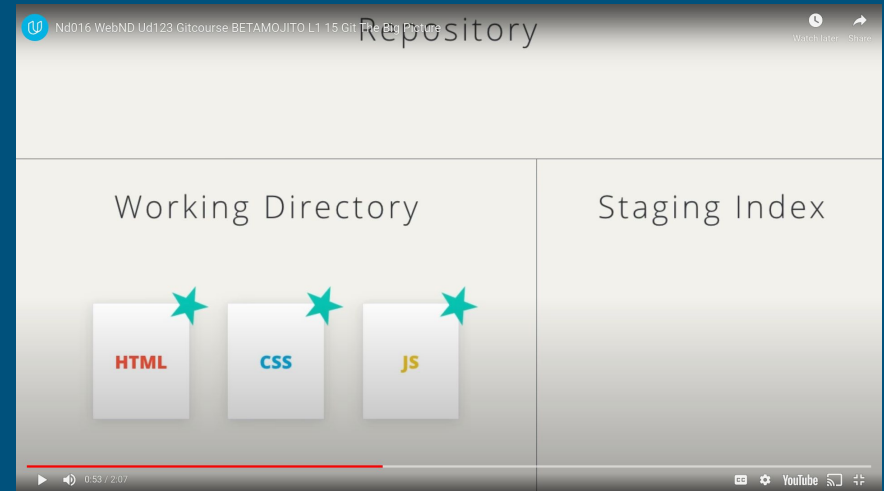
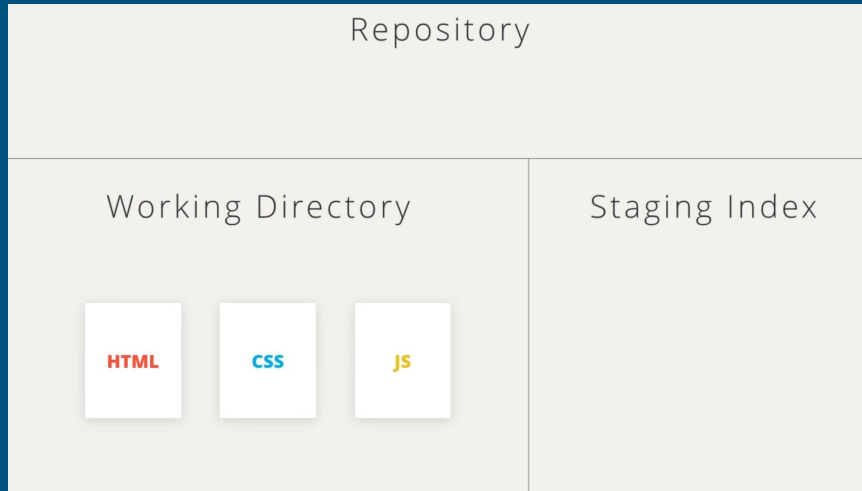
```
$ mkdir git_nuevo
```

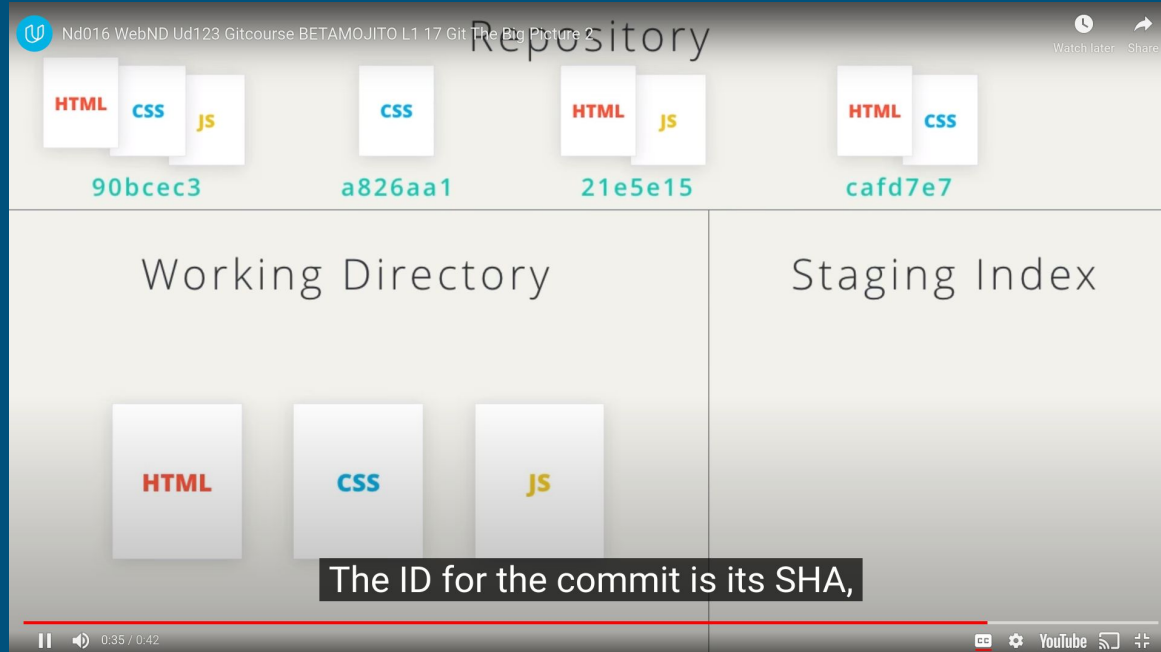
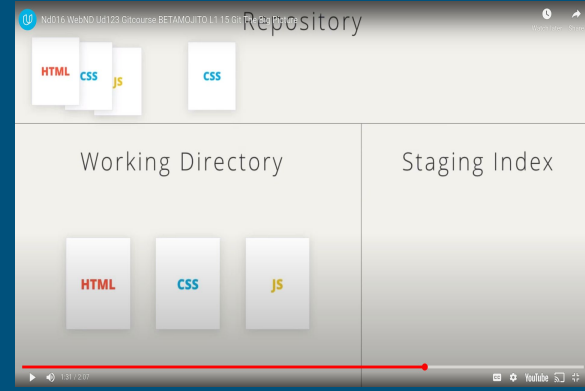
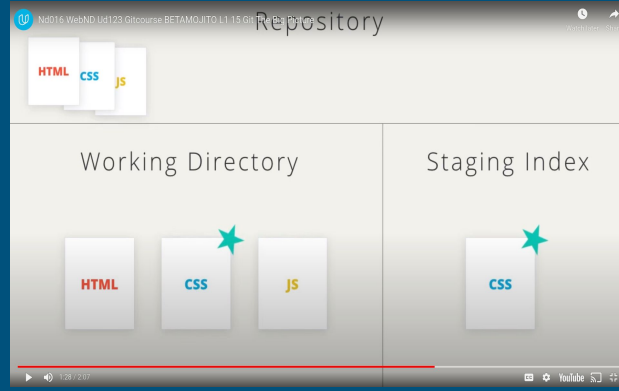
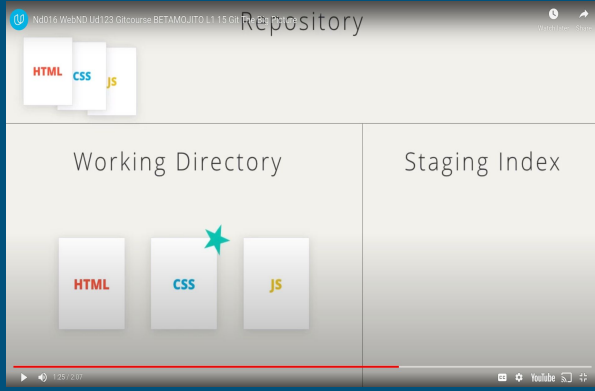
```
$ cd git_nuevo
```

```
$ 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





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

```
caro@earth: ~/Dropbox/computacion/python/udacity/data analysis/proyecto 7/project
File Edit View Search Terminal Help

commit d4147de3db41eff2897de2f7a195bf352301a110
Author: Carolina <caro.negrelli@gmail.com>
Date: Sun Feb 23 12:03:52 2020 -0500

New files: exploration and explanatory analysis files.

- exploration.html: exploration analysis done in the previous added jupyter
notebook in html format.
- slide_deck.ipynb: explanatory analysis with the main results of the
exploration analysis
- slide_deck.html: explanatory analysis presentation slides.
- output_toggle.tpl: style file for nbconvert.

commit 147ffb3a332695cc427306e936761c19b8c0dab0
Author: Carolina <caro.negrelli@gmail.com>
Date: Sun Feb 23 10:20:56 2020 -0500

style: delete a Udacity comment.

commit de90cc0d9845a270db7c65b20c4769e20efela93
Author: Carolina <caro.negrelli@gmail.com>
Date: Sat Feb 22 11:16:53 2020 -0500

New file: presentation notebook

This python file contains plots to present the results.

commit 622503b210e5897be4d76d4b5287352cd65a4f51
Author: Carolina <caro.negrelli@gmail.com>
Date: Fri Feb 21 13:06:30 2020 -0500

feat: new exploratory visualizations

I added bivariete and multivariete exploratory visualizations.
```

Flags

--online -> 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

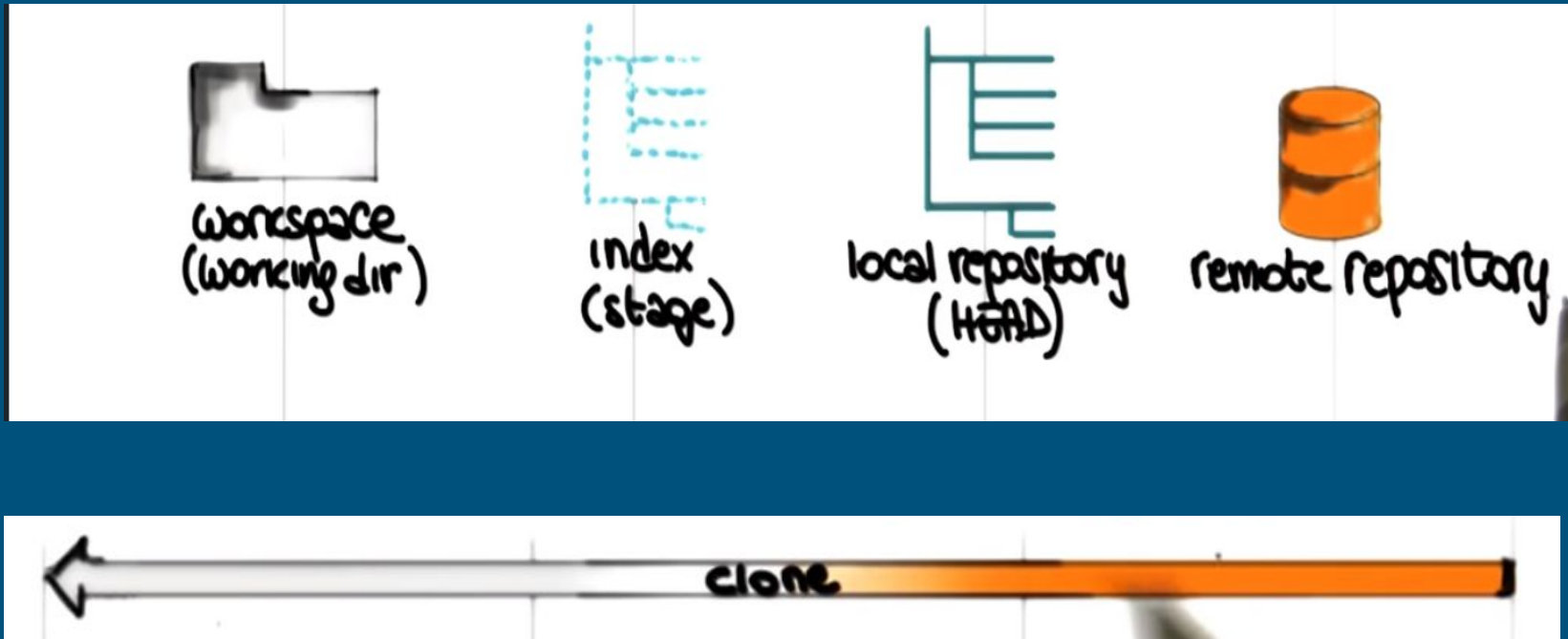
Cuando hacer un commit?

- 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 con código.
- Antes de hacer una reestructuración.

GitHub

The GitHub logo, which is a small blue horizontal line, is positioned directly below the text.

Repositorio remoto: GitHub

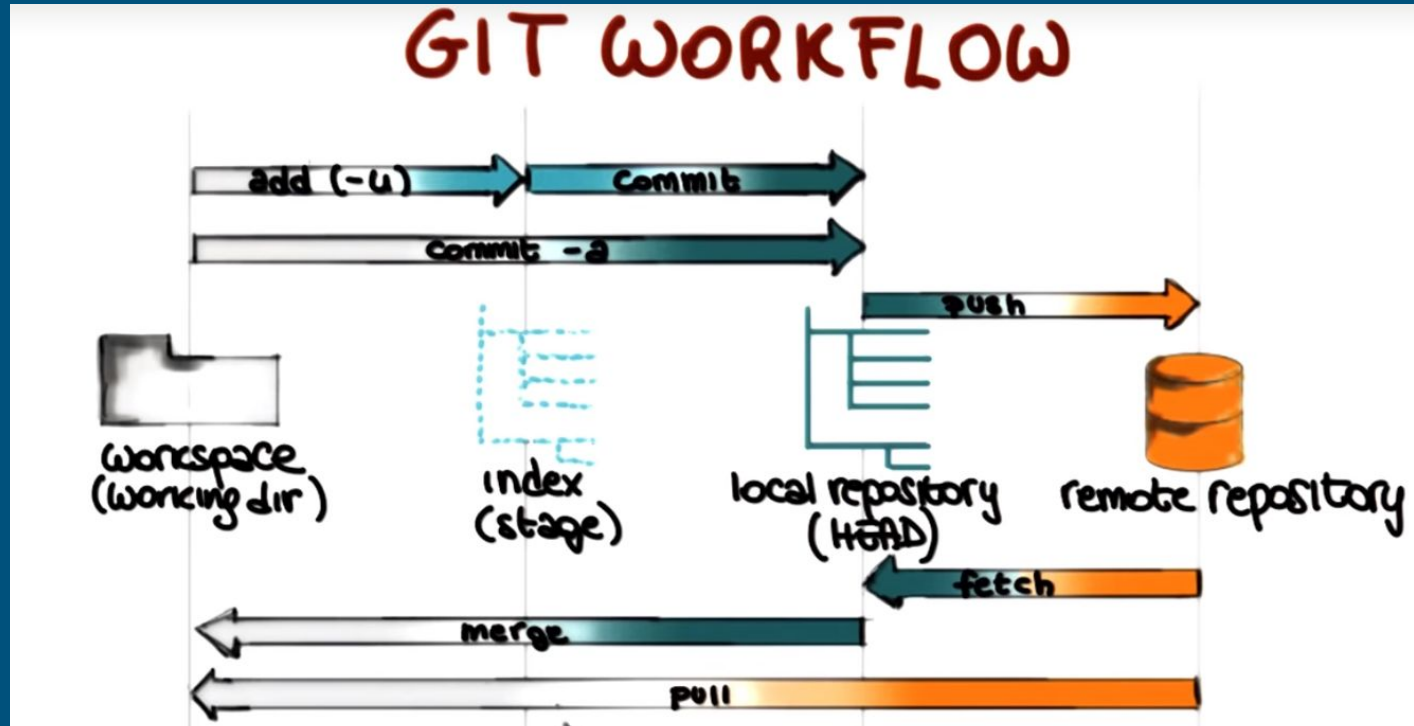


Usos de GitHub

- Colaboraciones
- Portfolio
- Red Social
- Backup

Nota: hay un límite de tamaño por repositorio

Subir un repositorio a GitHub: push



Pushear

```
$ cd git_nuevo
```

\$ git remote -v ----> nos indica el repositorio remoto, si no sale nada no hay ninguno asociado todavía.

Ir a GitHub y crear un nuevo repositorio. Seguir las instrucciones dadas en:

push an existing repository from the command line

```
$ git remote add origin https://github.com/.....
```

\$ git push -u origin master ---> esto es solo la primera vez, desp solo \$ git push

Actualizar la pag de GitHub.

Parte dos

Repaso

git_journal
(no git)

```
graph TD; A["git_journal (no git)"] --> B["git_nuevo"]; A --> C["git_clonado"];
```

git_nuevo

- Inicializamos como git
- Dos commits
- Asociamos un remoto propio en github
- Pusheamos a github

git_clonado

- Clonamos un repositorio de Github
- Chequeamos que el remoto es el repositorio que 'robamos' y no nos pertenece

Actualizar el repositorio clonado: pull

El repositorio que clonamos 'avanzó' respecto al nuestro.

Para actualizarlo:

```
$ cd git_clonado
```

```
$ git status ---> chequear que no modificamos nada
```

```
$ git remote -v ---> para chequear que siga asociado al original
```

```
$ git pull
```

Nota: el pull en la practica no se suele usar con este fin pero sirve como ejemplo básico del pull.

Pull en un repositorio propio: Agregar un readme desde GitHub

Se puede agregar/modificar el readme directamente desde GitHub.

Ojo! Cada vez que hacen eso están haciendo un commit en GitHub que NO está en el repositorio local. Entonces si hacemos commits en el local e intentamos pushear va a haber un conflicto!

Solución: luego de hacer el commit en GitHub y antes de hacer modificaciones localmente, hacer un `$ git pull` ---> actualizamos lo local

Si se olvidaron e hicieron un commit local: `$ git push -f` ---> fuerza el commit pero van a perder las modificaciones del readme

Otra opción: `$ git stash -> git pull -> git stash apply`. Es como si te dejara en 'stand by' los cambios locales (no commiteados).

Trabajar desde dos compus en el mismo rep

- Usamos github para 'sincronizar' la compu de la ofi y de casa
- 1) Armar el repositorio git en alguna de las compus, cuando este todo lo que queremos commiteado lo asociamos a un rep en github y pusheamos.
 - 2) Desde la otra compu clonamos el repositorio (git clone htt..)
 - 3) Ambos repositorios locales van a tener asociado el mismo repositorio remoto.
 - 4) Trabajamos en la compu 1, commiteamos y pusheamos a github.
 - 5) Antes de trabajar en la compu 2 hacemos `$git pull` para actualizar en la compu 2 lo que habíamos hecho en la 1 y ahí seguimos trabajando.

Nota: funciona igual para dos personas trabajando en el mismo repositorio, pero el creador del repositorio en github le tiene que dar permiso de colaborador al otro..

Qué pasa si quiero pushear un repositorio clonado?

`$ cd git_clonado`

`$ git remote -v` ---> tiene como remoto el repositorio desde donde se clonó.

Si hacen un commit e intentan un `$ git push` les va a fallar el permiso. Así que si quieren subirlo a github tienen que tener su propio repositorio y cambiar el remoto.

1) crean el nuevo repositorio en GitHub, 2) `$ git remote set-url origin NEW_URL` ---> la NEW_URL es la del nuevo repositorio, 3) `$ git push`

Nota: hay MUCHAS formas de hacer estas cosas. Otra opción: hacer un fork cliqueando el botón en github (crea una copia del repositorio en NUESTRO github) y después clonar nuestro repositorio a la compu local (el remoto va a ser nuestro github).

Deshacer cosas

Volver un archivo al último commit

Ejemplo: se pusieron a modificar el archivo hola.py y ya no les funciona, no encuentran el error y cambiar a mano lo que hicieron es imposible/difícil o no tienen ganas.

Para volver ese archivo a su último check-point (commit), ponen:

```
$ git checkout -- file
```

Deshacer cambios commiteados

\$ git commit --amend ---> cuando te olvidaste algo, lo agregas siguiendo los pasos normales pero al git commit le agregas la flag --amend. Es para no crear un commit extra con algo que debería formar parte del anterior.

\$ git revert id ---> hace un nuevo commit que revierte todos los cambios hechos por el commit con ese id.

\$ git reset ---> este es el comando para BORRAR commits. Se recomienda NO utilizarlo. Tiene muchas opciones de flags dependiendo de qué commit queremos borrar. <https://git-scm.com/docs/git-reset>

Parte tres

Git: Ramas y más

Tags

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

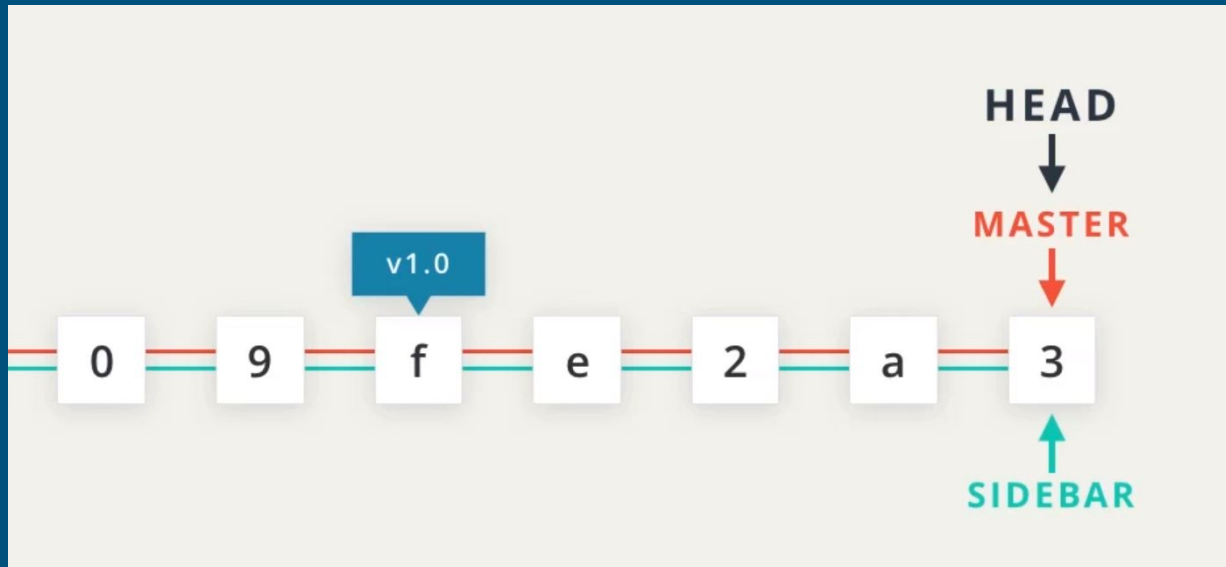
\$ git tag -a V1.0 (SHA) ---> despliega el editor para anotar un mensaje

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

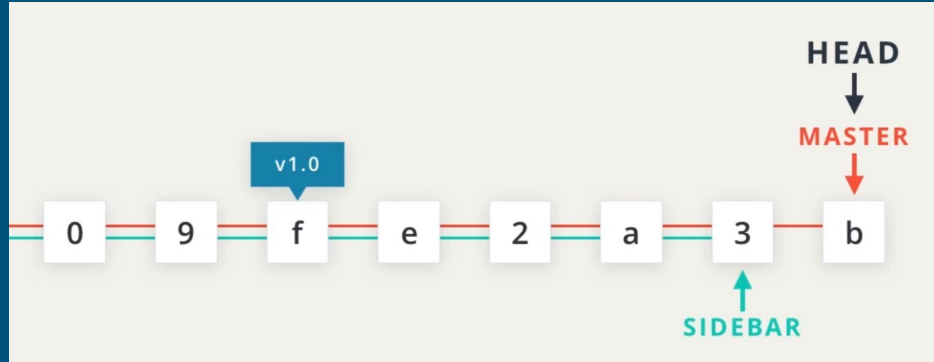
Ramas (branches)

Permite diferentes líneas de desarrollo.

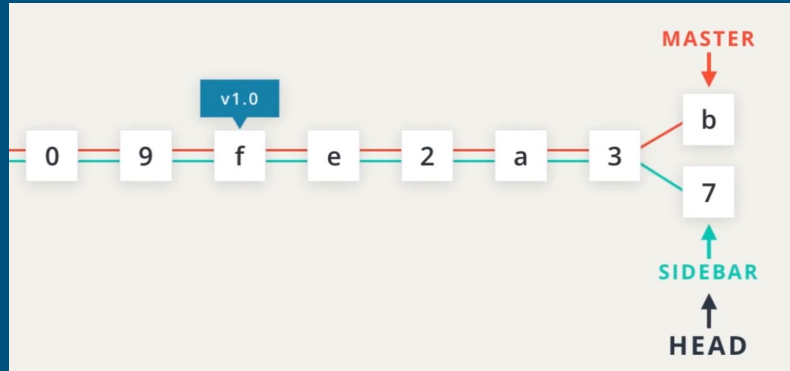
\$ git branch sidebar ---> crea la branch sidebar



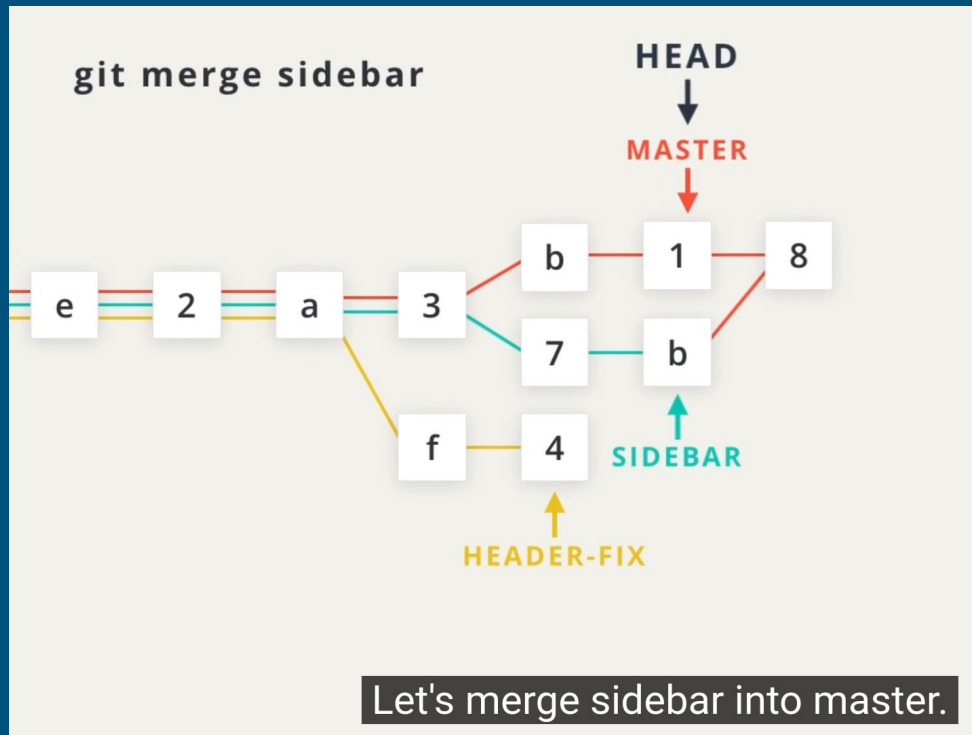
\$ git branch ---> muestra la lista de branches y marca con un * la actual



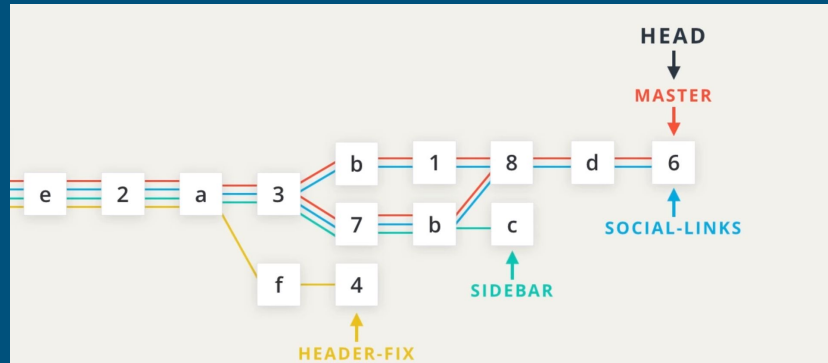
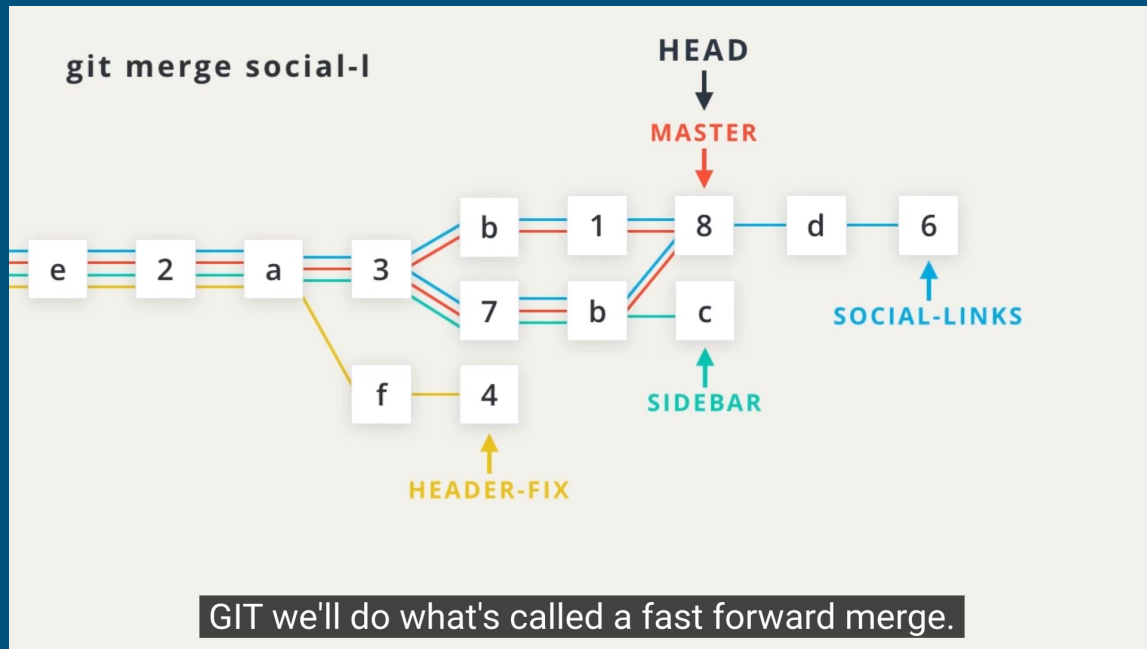
\$ git checkout sidebar ---> activa la branch sidebar



Merge -> queremos unir los cambios



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



- 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 branches se cambian las mismas líneas del mismo archivo.

GitHub: un poco más

Issues and pull request

Supongamos que le queremos resolver algún problema a AstroML (ver issues) o agregar algo nuevo. 1) Fork, 2) Clone, 3) New branch, 4) Cambios necesarios, 5) push la new branch a nuestro repositorio remoto.

Si resolvimos el problema y queremos que AstroML actualice el repositorio, hay que pedir un pull request desde la new branch en github, van a evaluar los cambios y eventualmente hacer un merge.



Ultima cosita

Practiquen mucho :)
