

Control de Versiones

Electrónica IV

Mg.Ing. Esteban Volentini (evolentini@herrera.unt.edu.ar)

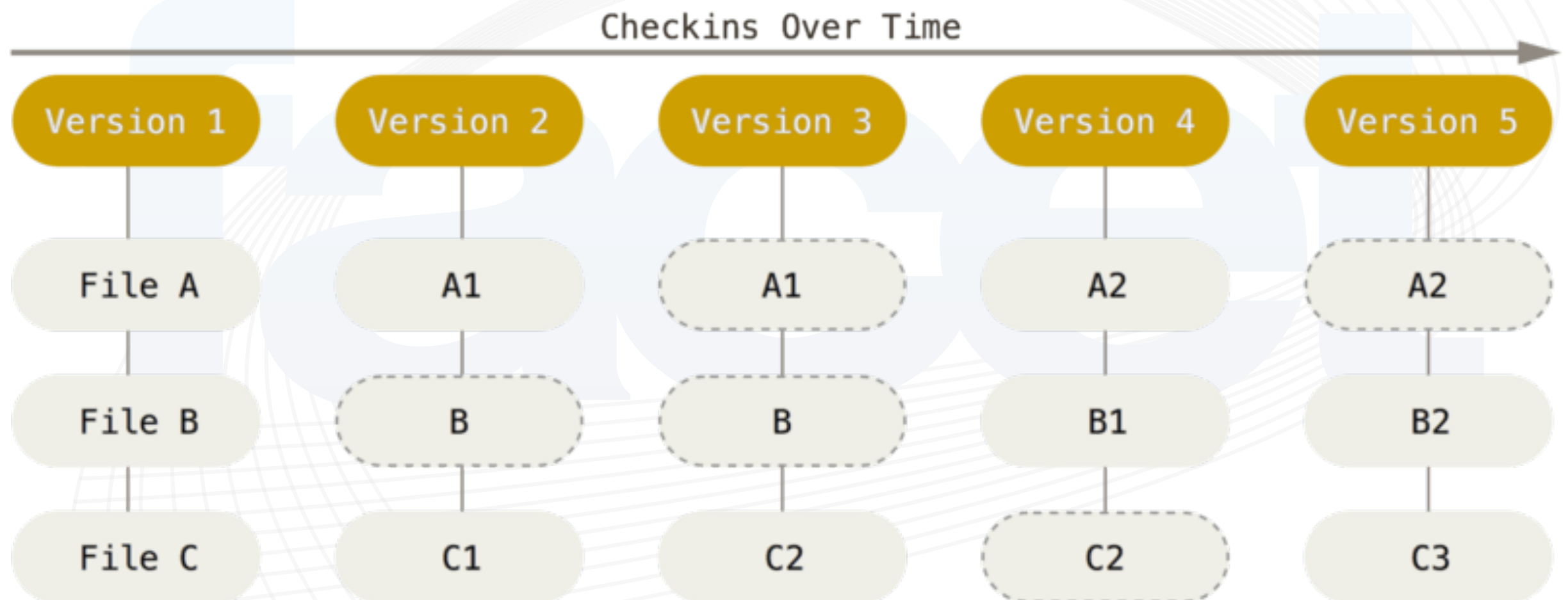
<https://facetvirtual.facet.unt.edu.ar/course/view.php?id=165>

Control de versiones con GIT

- ▶ Es una herramienta para control de versiones diseñada por Linus Torvald.
- ▶ Es un sistema distribuido donde cada programador tiene una copia completa del repositorio.
- ▶ Los cambios pueden intercambiarse directamente entre clientes usando http, ssh o rsync.
- ▶ El servidor no es más que un cliente que está siempre disponible.

GIT almacena instantáneas

- ▶ Almacena las versiones como una imagen instantánea del sistema de archivos

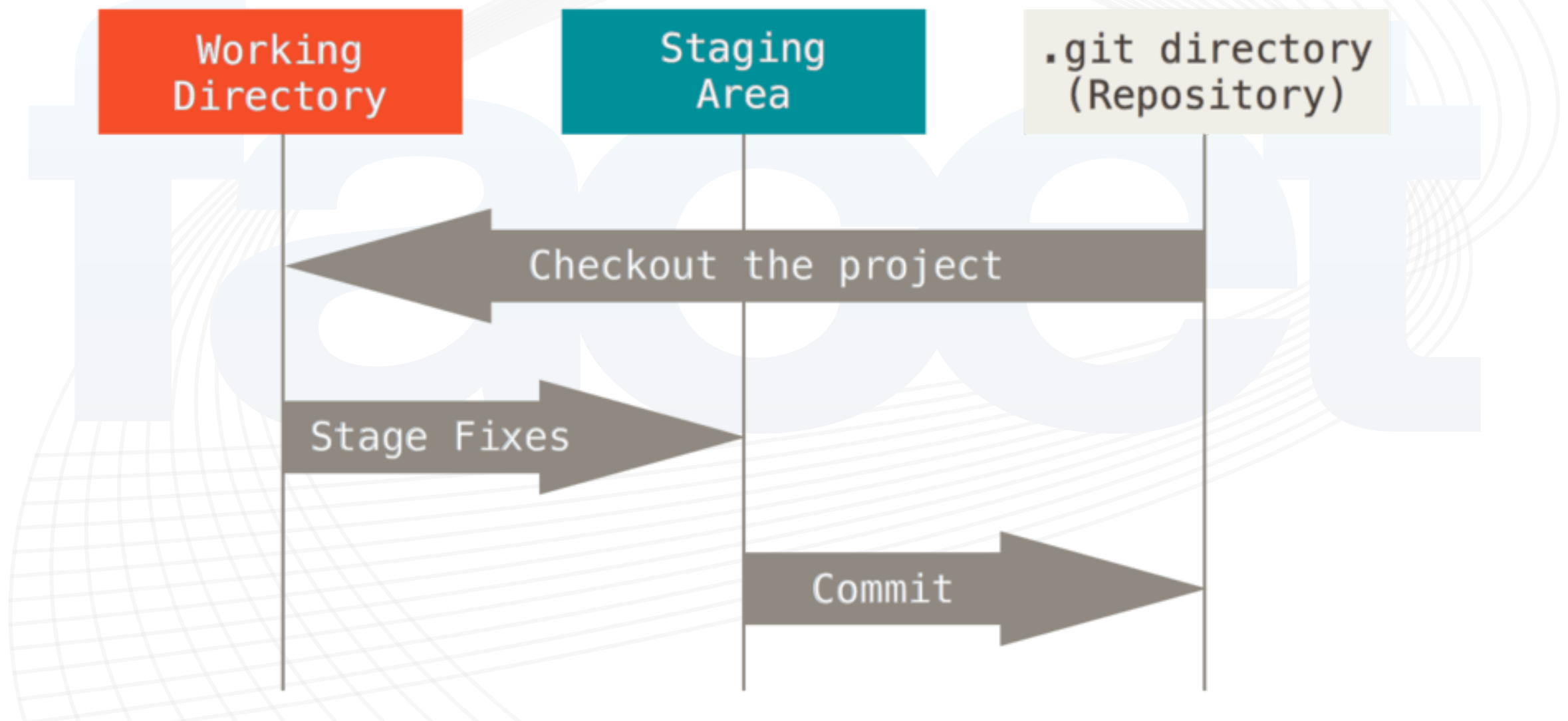


La función SHA1

- ▶ Una función de hash toma un conjunto de elementos, generalmente cadenas, y devuelve un valor en un rango definido, normalmente una cadena de longitud fija.
- ▶ Git utiliza la función SHA-1 que devuelve una cadena hexadecimal de 40 caracteres.
- ▶ Git calcula un hash para cada archivo que resume el contenido del mismo.
- ▶ Git almacena las versiones y los archivos por su hash.

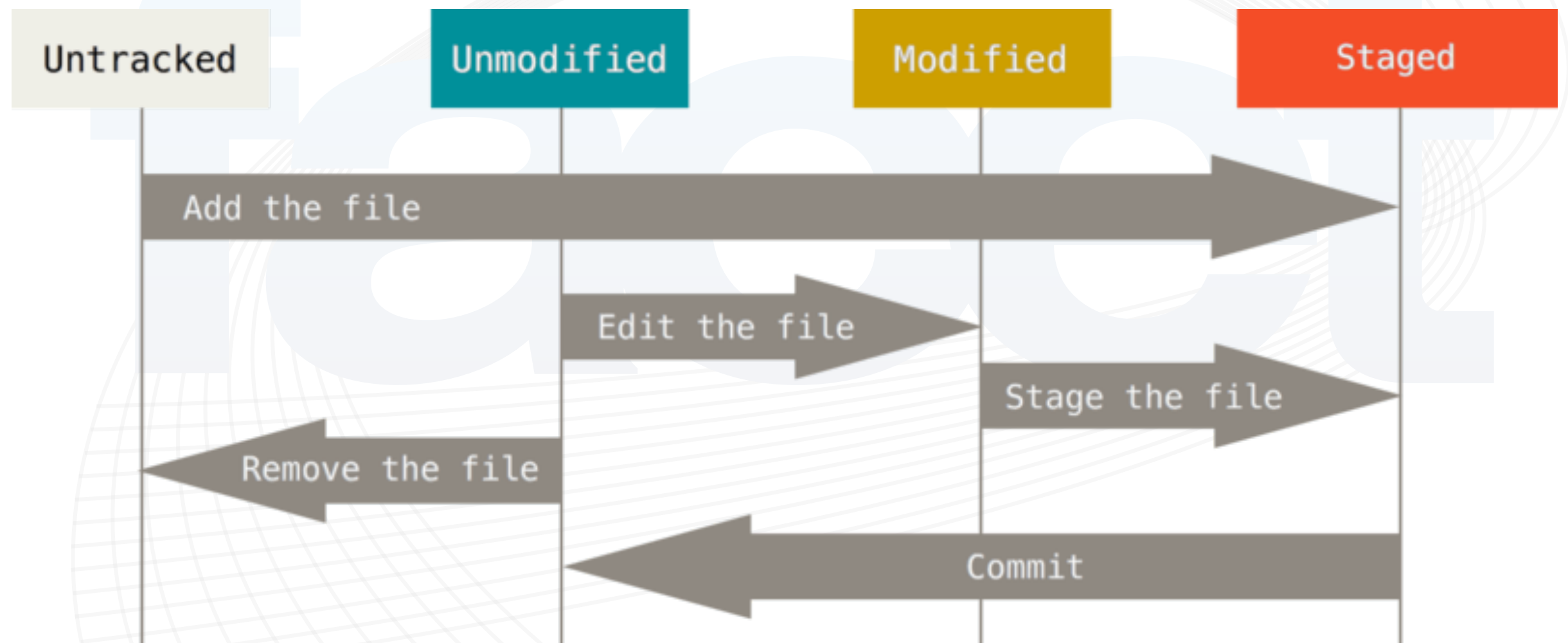
Las áreas de GIT

- ▶ Un repositorio GIT tiene diferentes áreas de trabajo



El estado de los archivos

- ▶ Cada archivo puede estar en uno de cuatro posibles estados

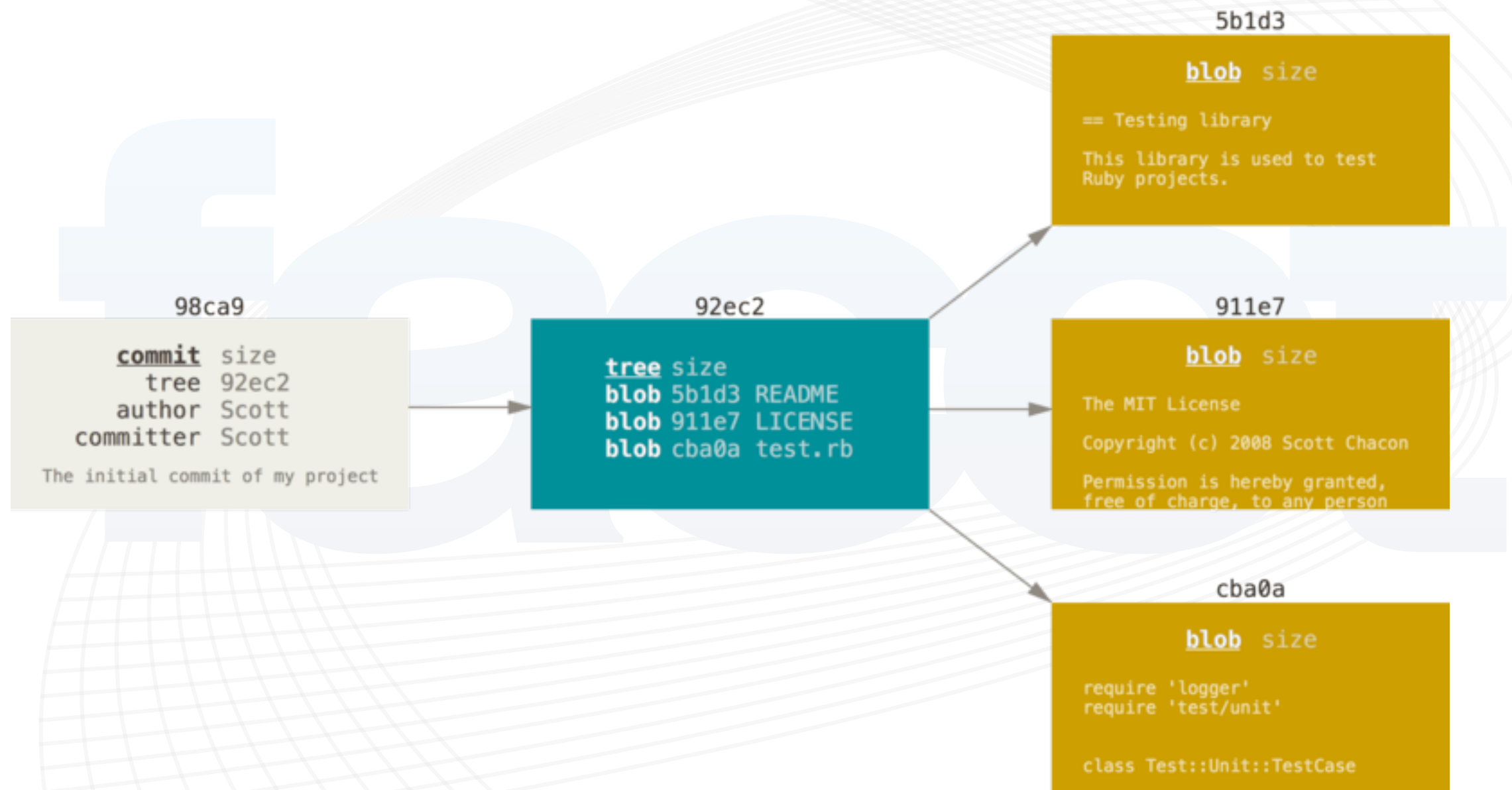


Un poco de práctica...

- ▶ Crearemos una carpeta con un archivo
- ▶ Lo convertiremos en un repositorio
- ▶ Veremos los estados en los archivos
- ▶ Veremos el seguimiento de las modificaciones

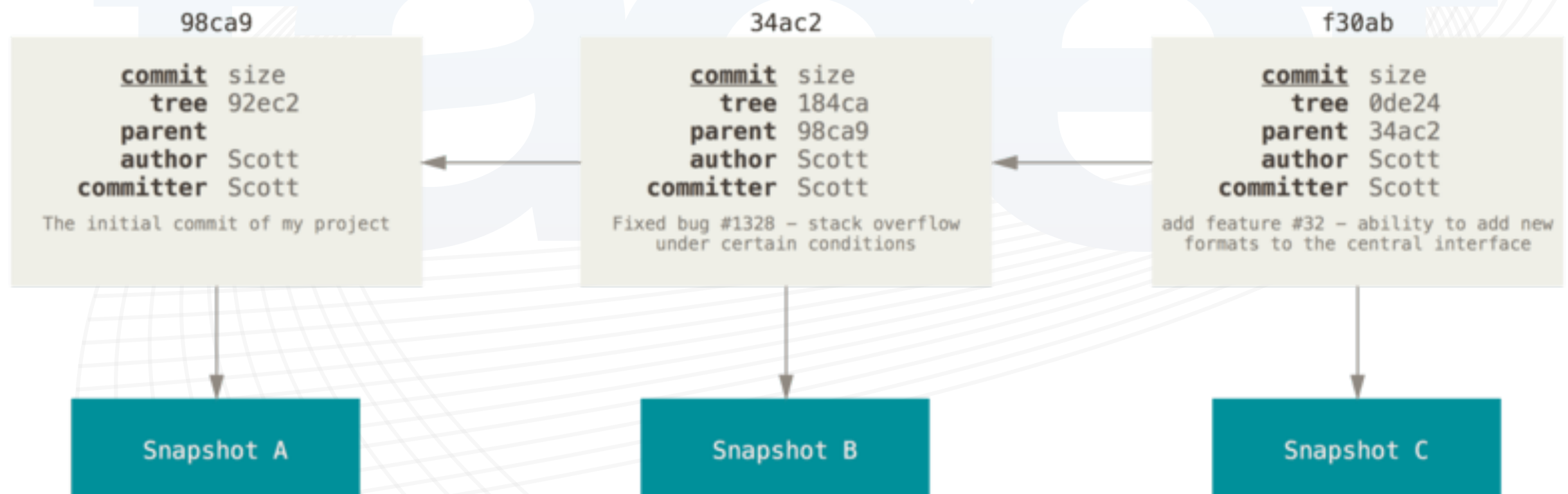
El Commit

- Un commit almacena una nueva versión en el repositorio



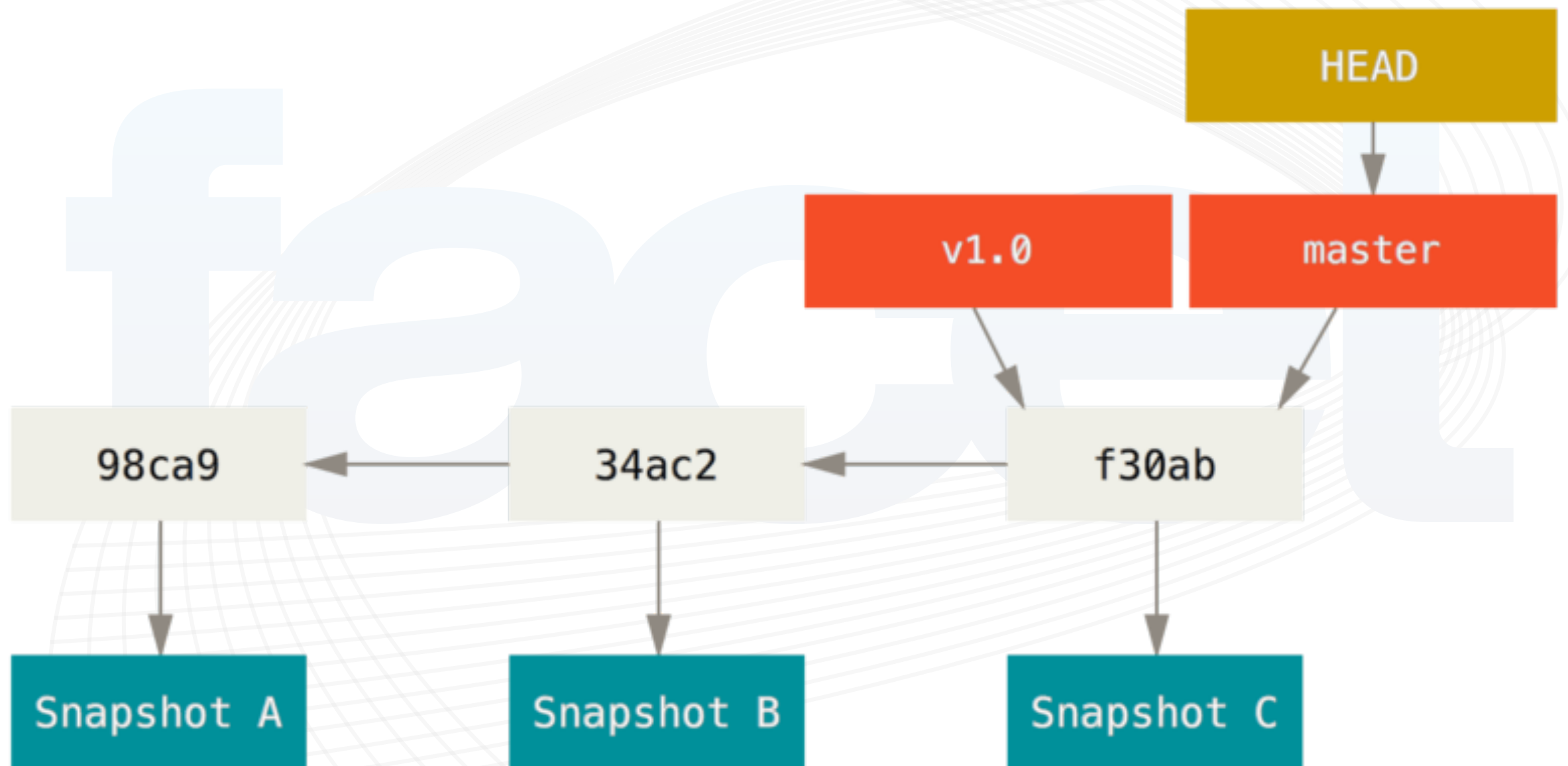
Los predecesores

- ▶ Un commit tiene un enlace a su predecesor
- ▶ El predecesor contiene la versión anterior del repositorio.



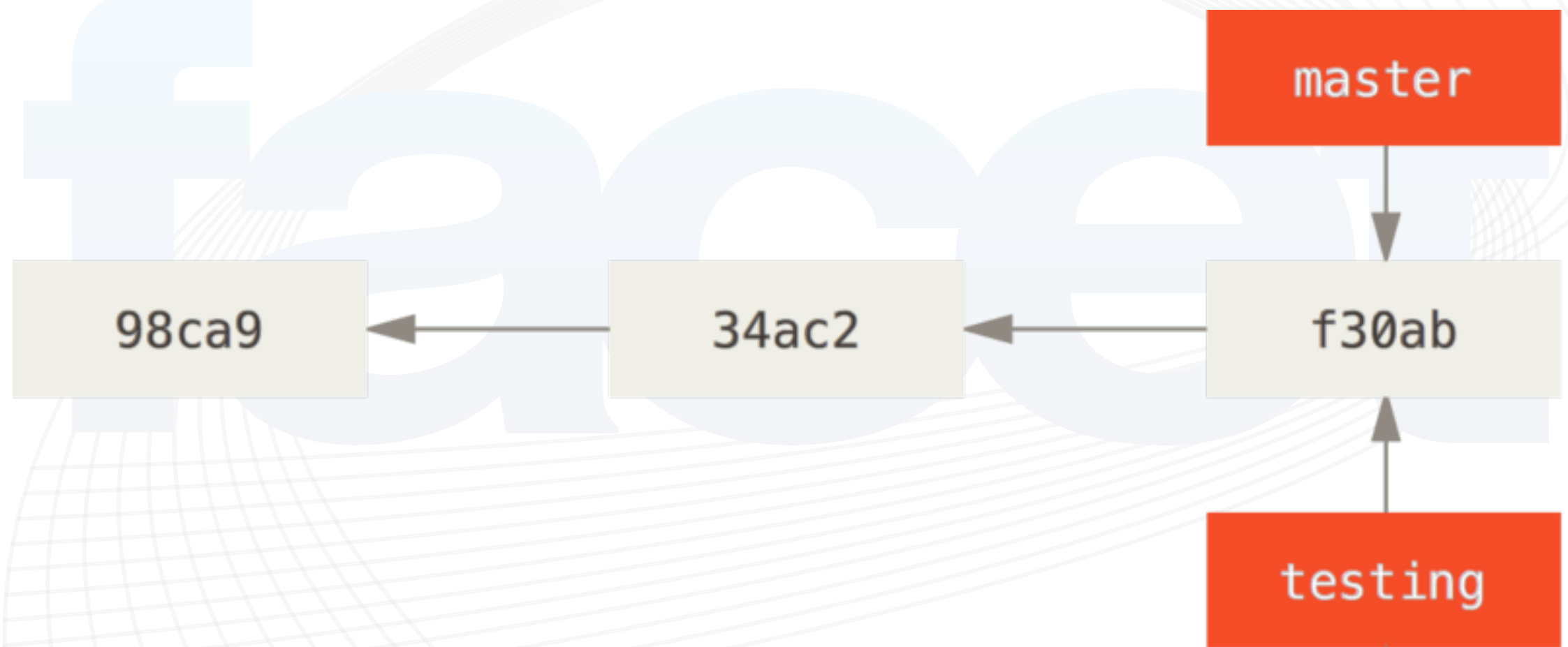
Las ramas

- Son referencias móviles con nombre a un commit

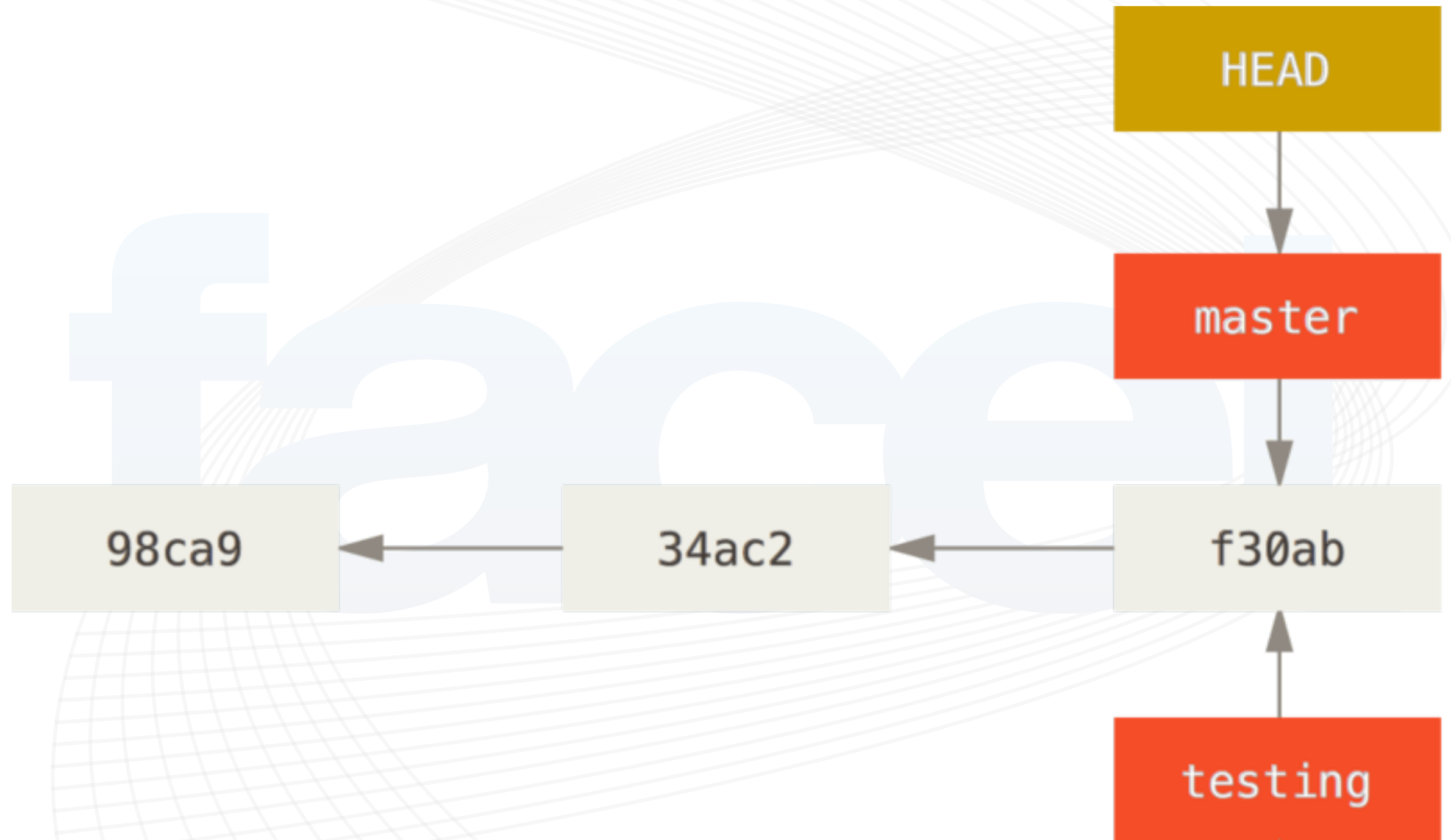


Creación de una nueva rama

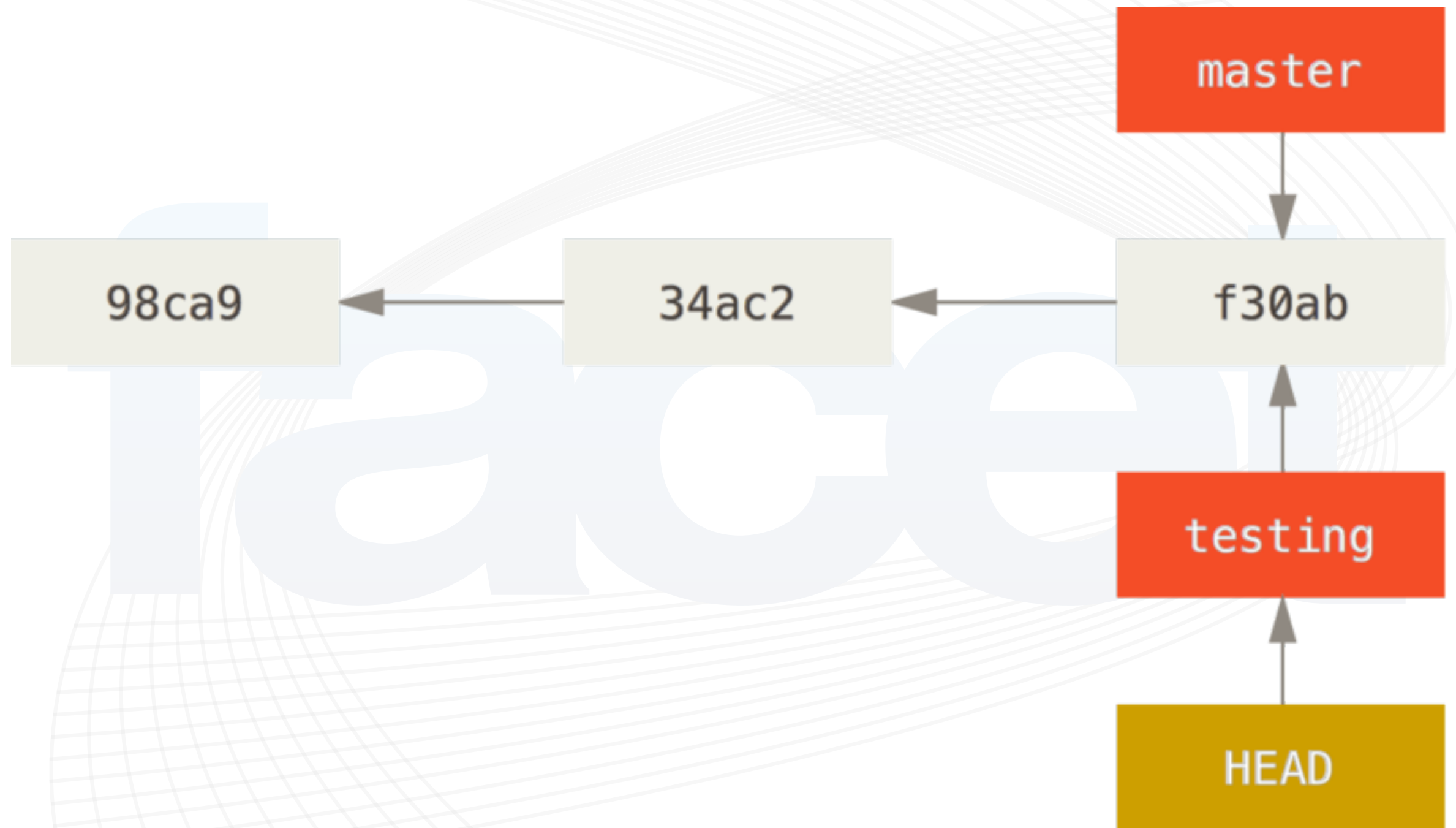
- Simplemente se agrega una nueva etiqueta



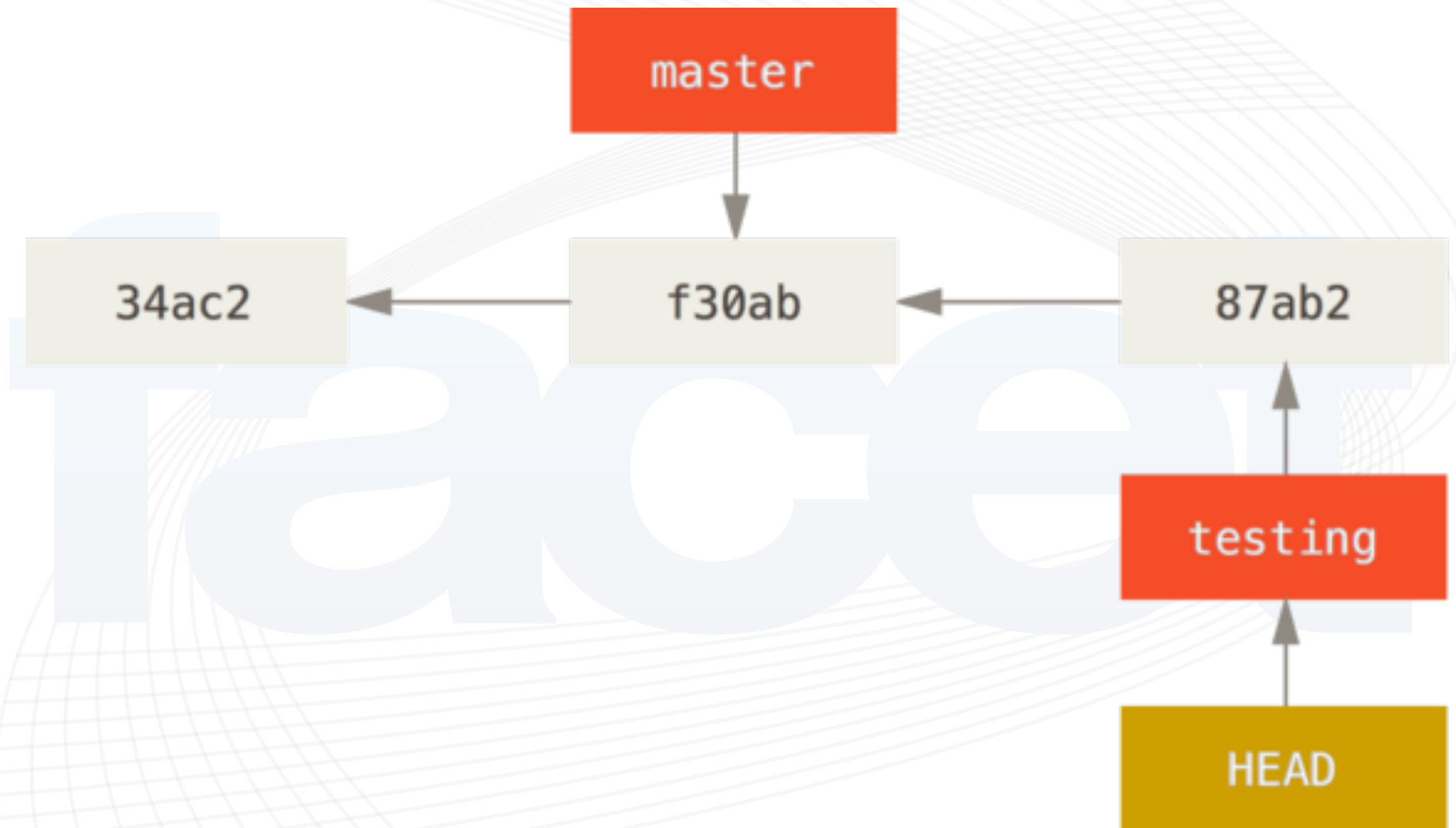
HEAD señala a la rama actual



Cambiando de rama



Agregando un commit a testing



El archivo `.gitignore`

- ▶ Permite ignorar los cambios en archivos que no deben almacenarse en el repositorio.
- ▶ Es un archivo de texto que define patrones que se aplican sobre los nombres de los archivos.
- ▶ Se pueden ubicar en la raíz del repositorio o en cualquier directorio.

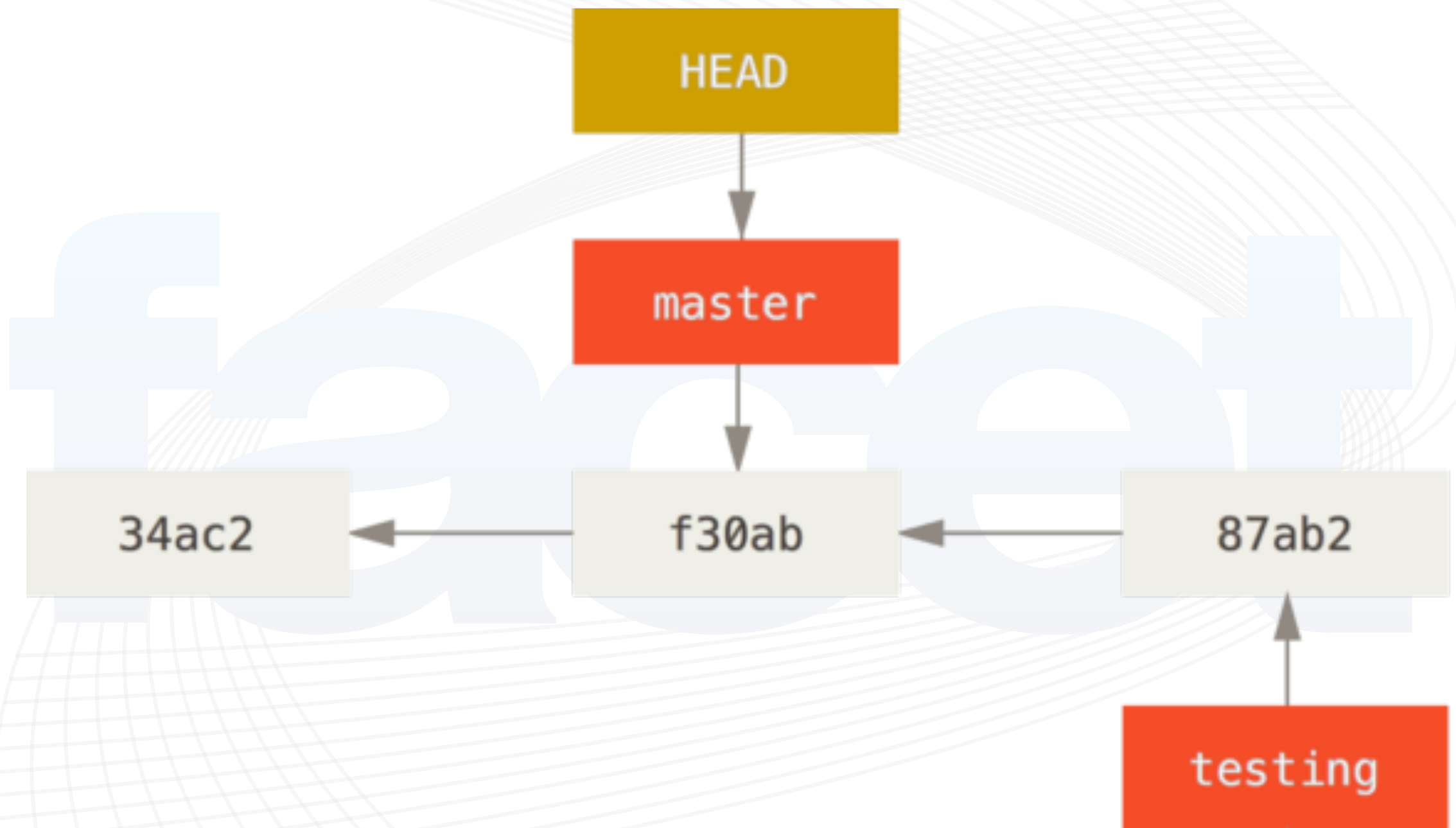
El archivo `.gitkeep`

- ▶ Como git no almacena carpetas sino archivos, se utiliza un archivo vacío con nombre `.gitkeep` para forzar la creación de una carpeta.
- ▶ Cuando se agregan archivos reales en la carpeta debe eliminarse el archivo `.gitkeep`.

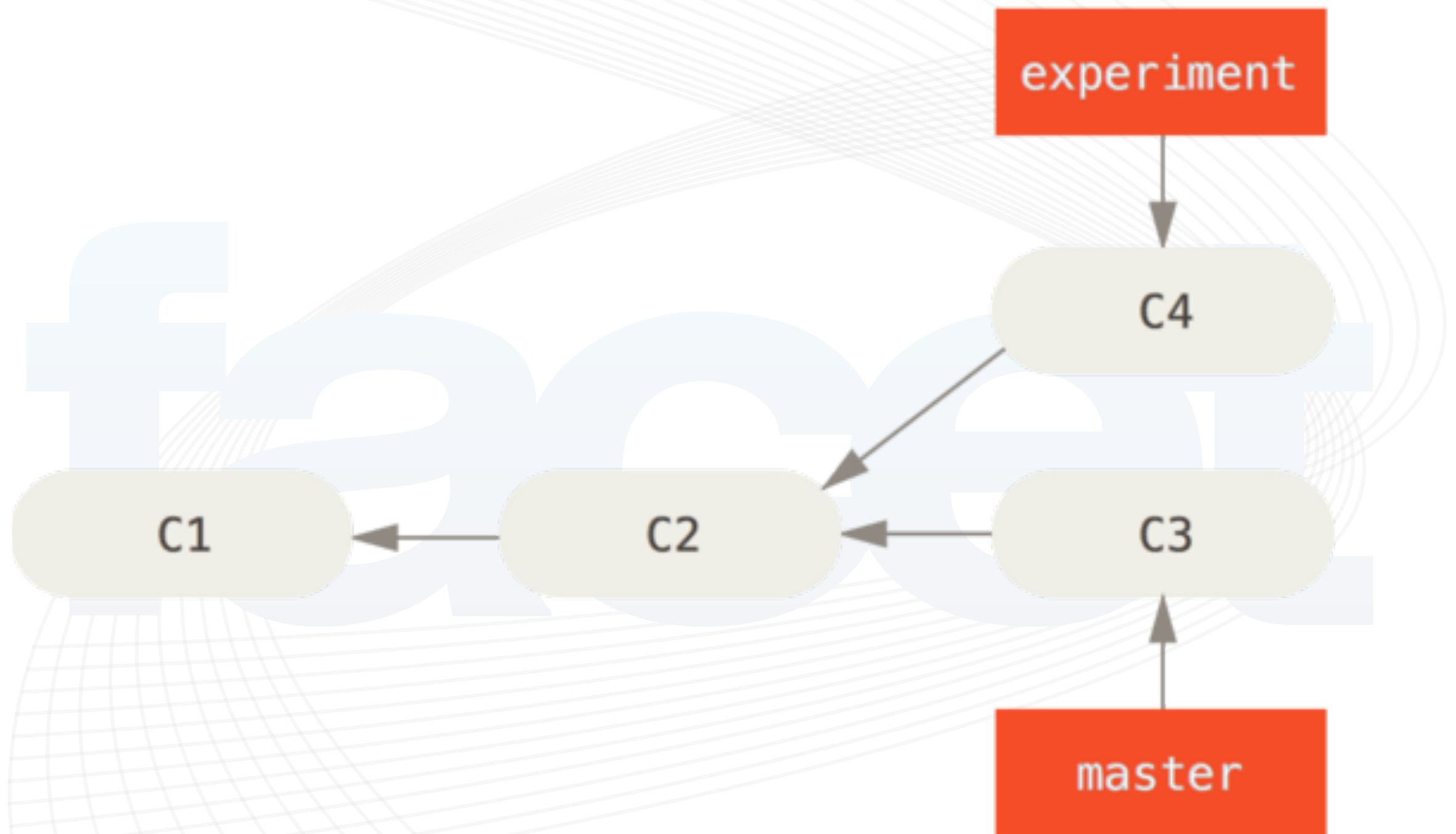
Un poco más de práctica...

- ▶ Crearemos dos commits
- ▶ Crearemos una nueva rama
- ▶ Agregaremos un commit en la nueva rama
- ▶ Volveremos a la rama (y a la versión) anterior

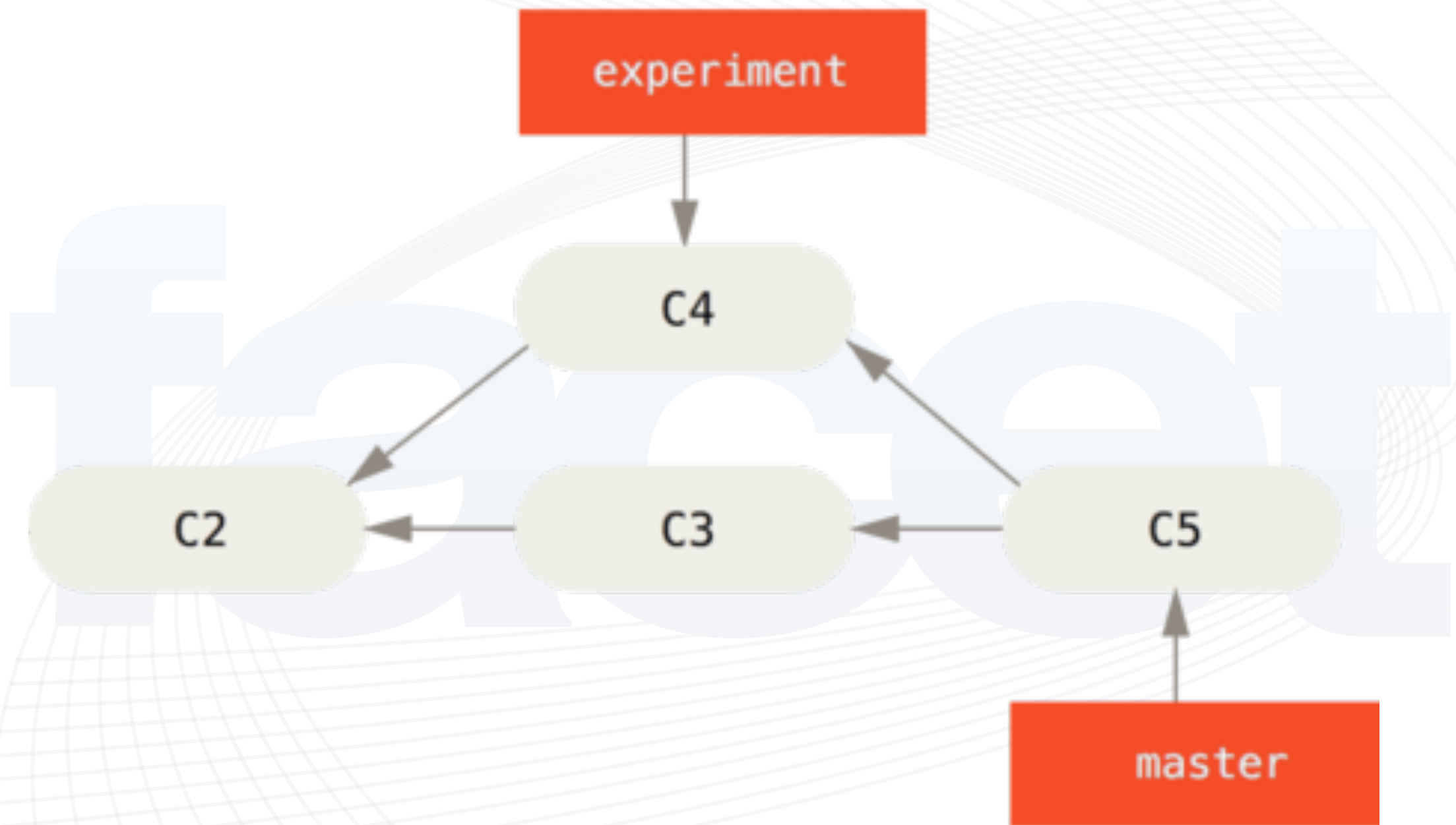
Recuperando la rama master



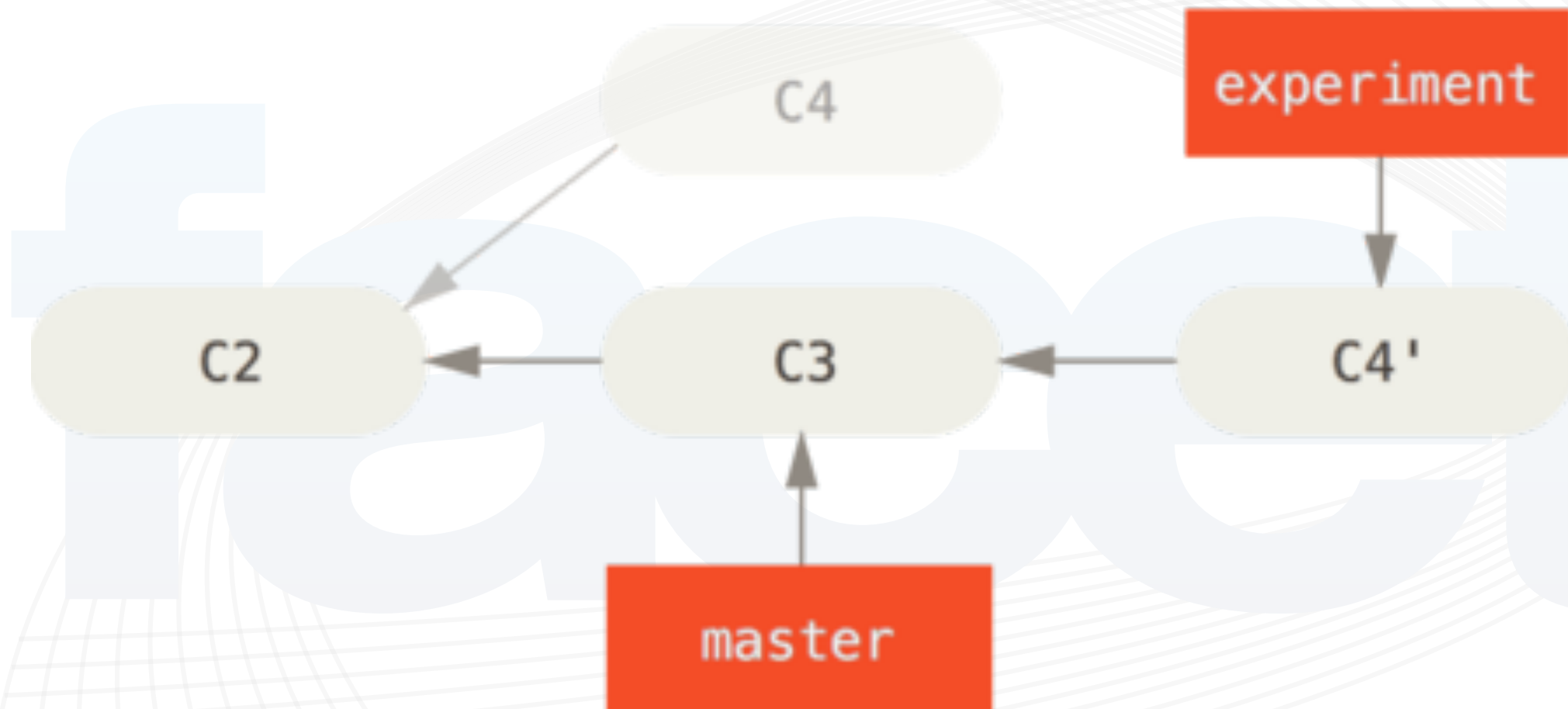
Cambios en dos ramas



Mezclando las ramas



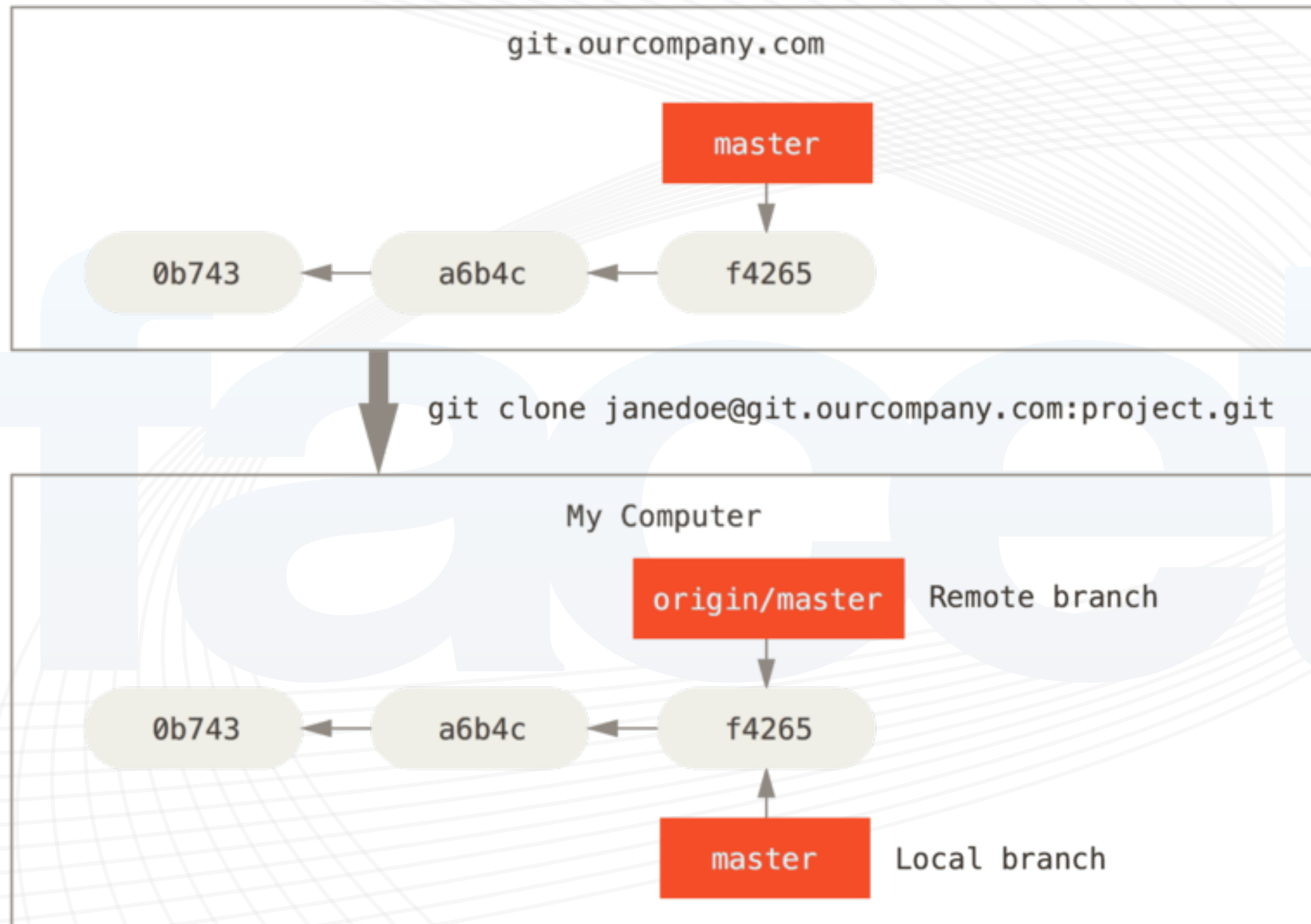
Reorganizando los cambios



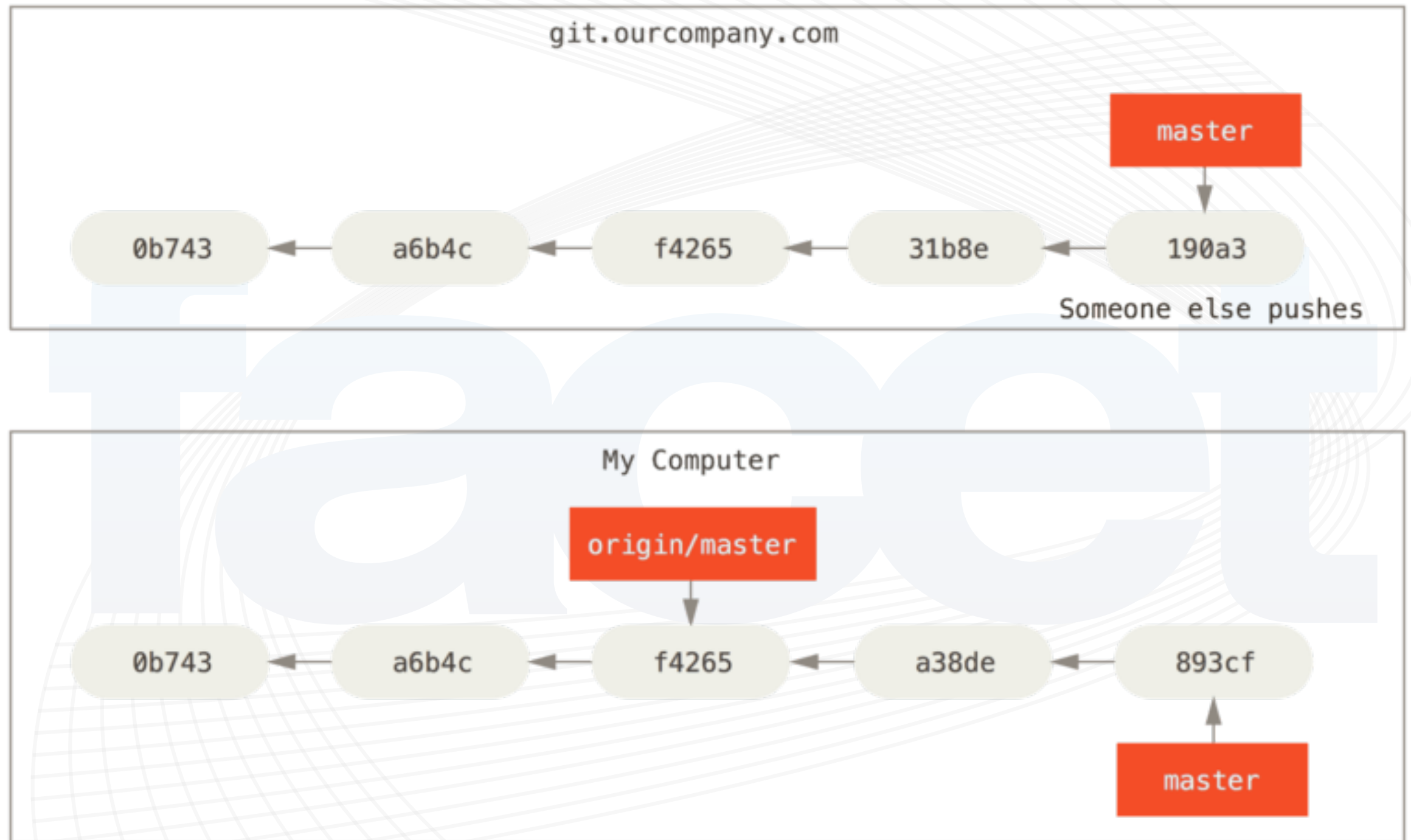
Todavía más práctica...

- ▶ Crearemos dos ramas
- ▶ Agregaremos un commit en cada rama
- ▶ Uniremos las ramas con una mezcla
- ▶ Cambiaremos una rama con una reorganización
- ▶ Repetiremos los pasos pero con conflictos

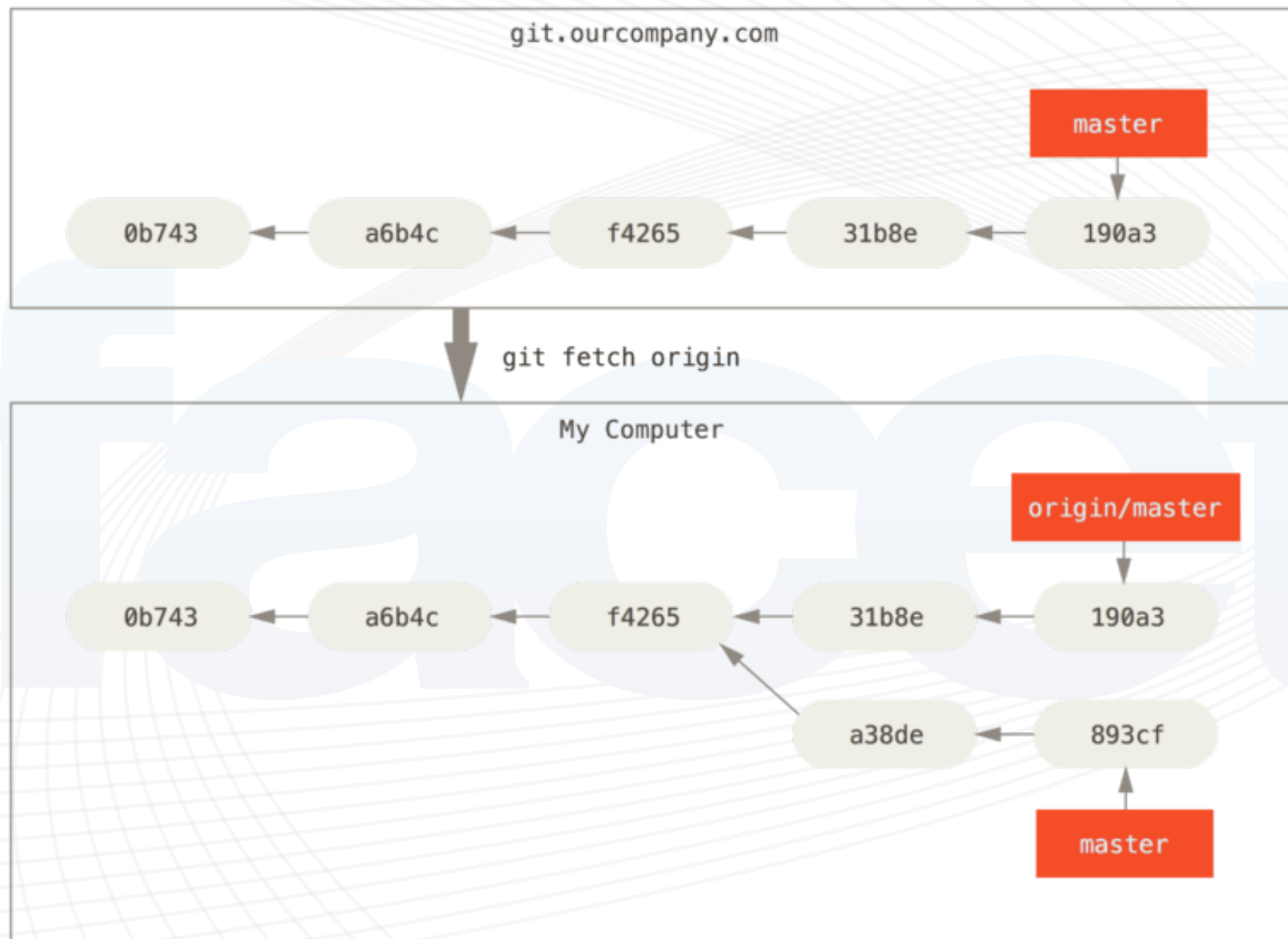
Clonando un repositorio



Nosotros hacemos cambios



Pero alguien más hizo cambios



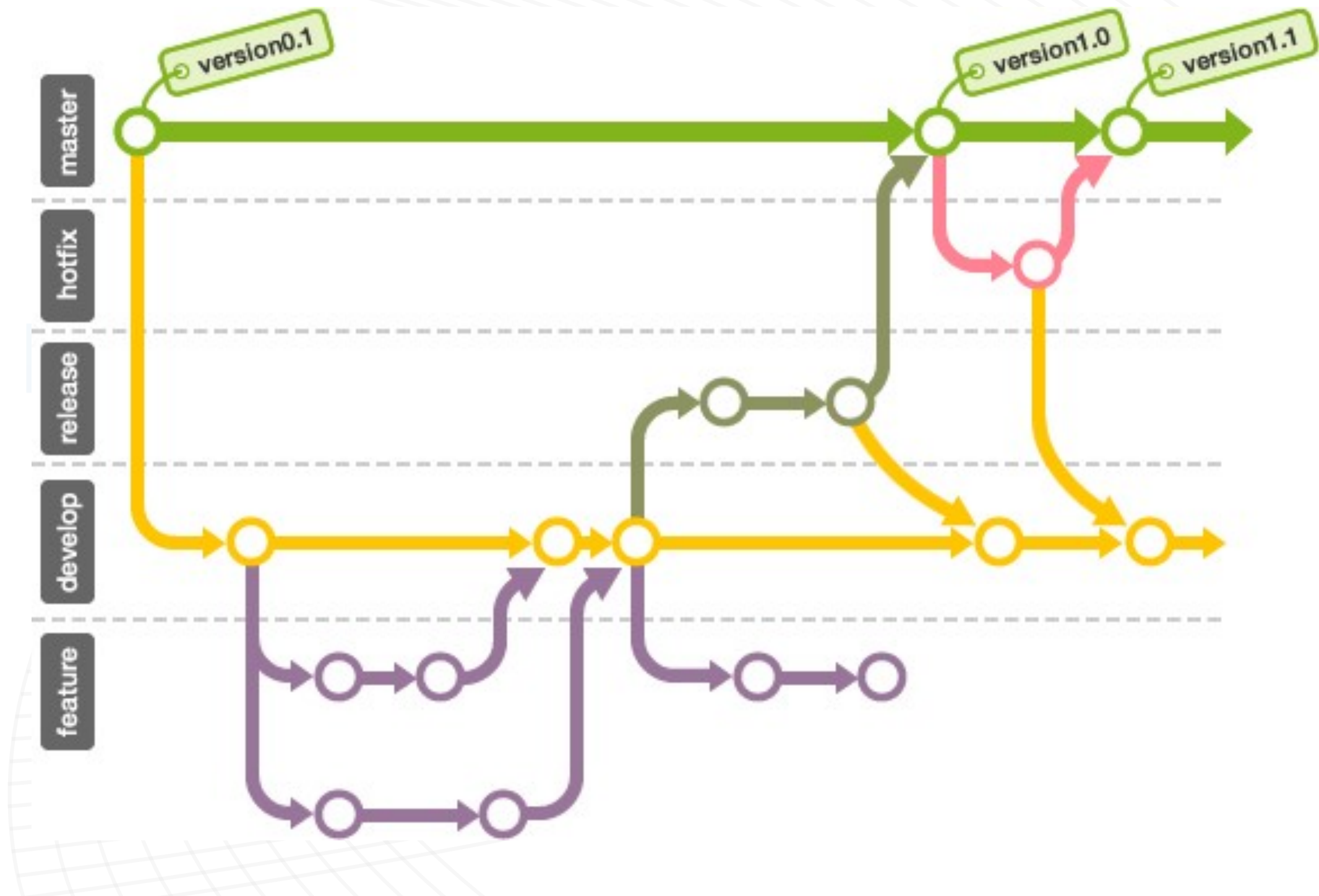
Y para terminar, mas práctica...

- ▶ Crearemos una cuenta y un repositorio en GitHub
- ▶ Definiremos un remoto para el repositorio
- ▶ Enviaremos los cambios al servidor
- ▶ Clonaremos el repositorio como un segundo usuario

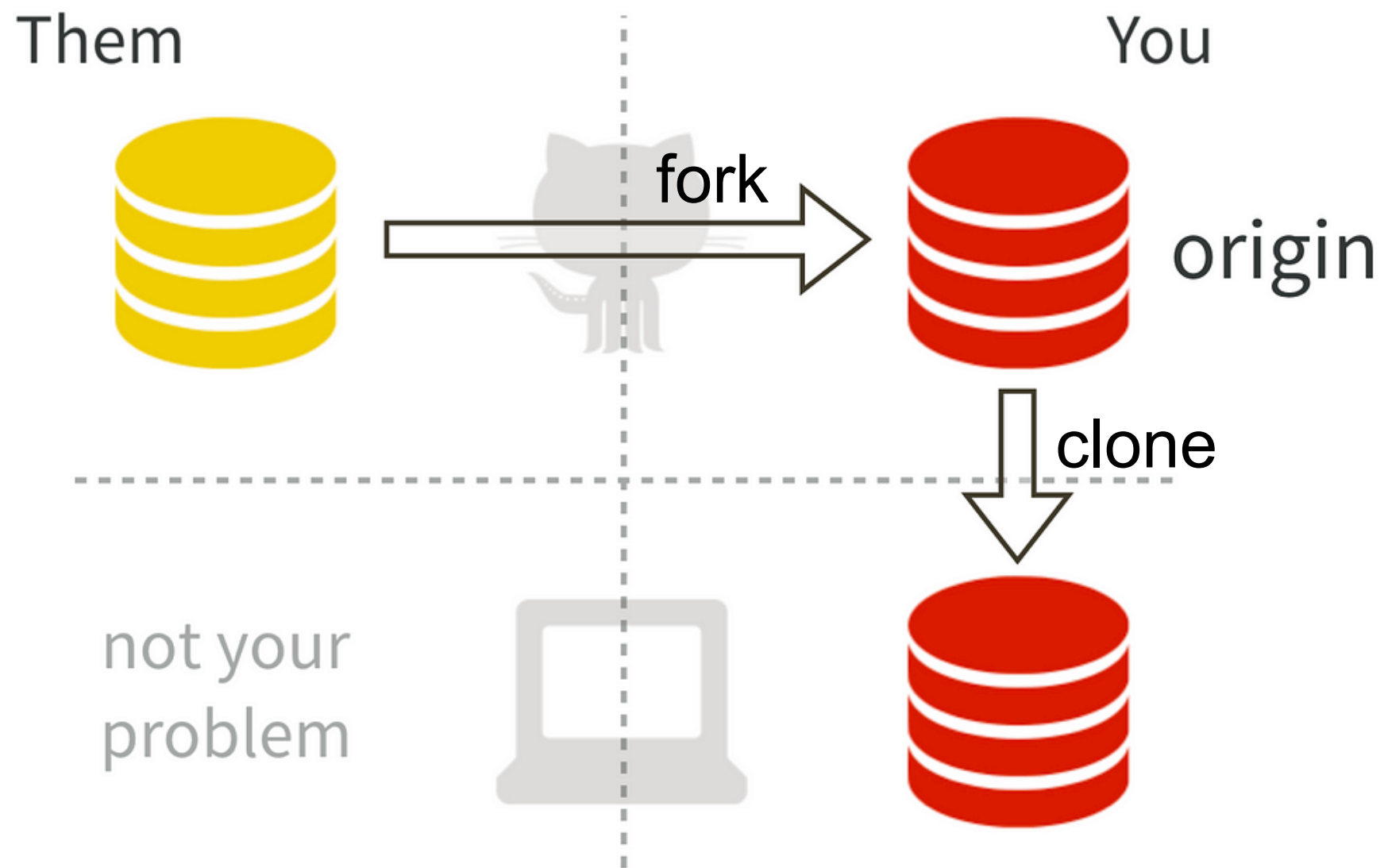
Buenas prácticas

- ▶ El objetivo de un sistema de control de versiones es poder analizar los cambios en el código
- ▶ Es muy importante definir una estructura clara de como se gestionan los cambios
- ▶ Tambien resulta clave que cambios almacenados deben sean significativos
- ▶ Para evitar cambios de formato es necesario definir un estándar de codificación y utilizar un formateado de código
- ▶ Resulta muy útil incluir un proceso formal de revisión

GitFlow: estructura para el proyecto

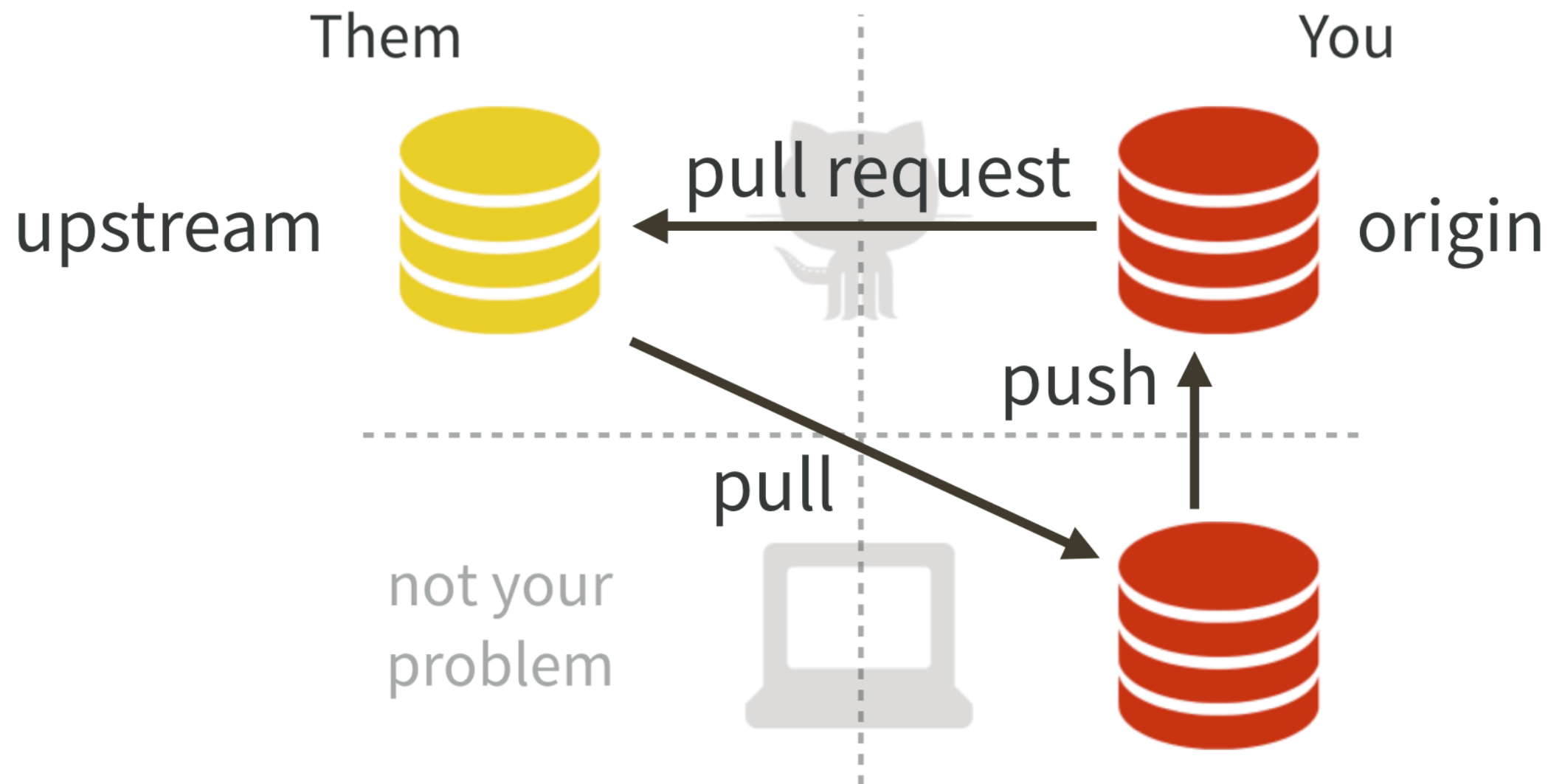


Fork Versus Clone



"fork and clone"

Pull, Push y Pull-Request



get changes from the main repo

Los hooks de Git

- ▶ La implementación de Git permite definir funciones de usuario que se ejecutan al realizar determinadas operaciones como un commit o un push.
- ▶ En estos hooks se pueden instalar scripts que revisen o cambien el formato el código antes de un commit.
- ▶ Un ejemplo es <https://pre-commit.com/>

Ejemplos

- ▶ <https://github.com/snorkman88/cese-ids-12Co-tp1b/commit/88bdd80a47fbcd352d2d0e90c65ec034e3509e89>
- ▶ <https://github.com/cese-ids/cese-ids-12Co-tp1a/commit/ef50f34d43aae7efc2c38d3ab838ec7e9bca5f3c>
- ▶ <https://github.com/cese-ids/cese-ids-12Co-tp1a/commit/a28a0d568017fc02ae6f9b04e3b0f096315e32e9>
- ▶ <https://github.com/cese-ids/cese-ids-12Co-tp1a/commit/e931b98e772b0c9eb0e135584ec370554fb74769>