**GUIA RAPIDA GITHUB – PARTE 2**

# ¿Qué es un fork?

La palabra fork se traduce al castellano, dentro del contexto que nos ocupa, como bifurcación. Cuando hacemos un fork de un repositorio, **se hace una copia exacta en crudo (en inglés “bare”) del repositorio original que podemos utilizar como un repositorio git cualquiera.**Después de hacerfork tendremos dos repositorios git idénticos pero con distinta URL. Justo después de hacer el fork, estos dos repositorios tienen exactamente la misma historia, son una copia idéntica. Finalizado el proceso, tendremos dos repositorios independientes que pueden cada uno evolucionar de forma totalmente autónoma. De hecho, los cambios que se hacen el repositorio original NO se transmiten automáticamente a la copia (fork). Esto tampoco ocurre a la inversa: las modificaciones que se hagan en la copia (fork) NO se transmiten automáticamente al repositorio original.

# ¿Y en qué se diferencia un fork de un clon?

Cuando hacemos un clon de un repositorio, te bajas una copia del mismo a tu máquina. Empezás a trabajar, hacés modificaciones y hacés un push. Cuando haces el push estás modificando el repositorio que has clonado.

Cuando hacés un fork de un repositorio, se crea un nuevo repositorio en tu cuenta de Github, con una URL diferente (fork). Acto seguido tenés que hacer un clon de esa copia sobre la que empezás a trabajar de forma que cuando haces push, estás modificando TU COPIA (fork). El repositorio original sigue intacto. Lo vamos a ver en breve con un ejemplo.

# ¿Para qué sirve?

Tiene varios usos. El más común es el de**permitir a los desarrolladores contribuir a un proyecto de forma segura.**

¿Por qué decimos de forma segura? Imaginaos un super proyecto como puede ser el código fuente de Apache. ¿Cómo se trabajaba antes de existir git? con Sub versión o CVS existía un servidor centralizado que tenía dos tipos de usuarios: lo que podían “escribir” en el repositorio (subir cambios al código fuente) y los que sólo podían “leer” el repositorio. Estos últimos sólo podían bajarse el código a su máquina y podían modificarlo sólo en su copia local. No podían subir ninguna modificación al servidor central.

¿Qué tenías que hacer para contribuir? Tenías que solicitar permiso de escritura y que alguien te lo diese. Una vez te lo concedían, ya podías subir tus modificaciones y, por supuesto, liarla si no sabías lo que estabas haciendo. Otra opción era enviar parches, trabajar con ramas… había varias formas pero todas bastante engorrosas.

Además, este procedimiento de dar acceso de escritura a un repositorio centralizado es un poco arriesgado. Siguiendo con el ejemplo que he puesto de Apache ¿cómo sé yo, responsable del repositorio, que esta persona que está a 10.000Km de mí en la otra punta del planeta sabe lo que está haciendo? ¿puedo confiar en él?. Al final, contribuir a un proyecto se convertía en una tarea tediosa tanto para el que pretendía contribuir como para el que lo tenía que gestionar. Y no hablemos de lo divertido que era hacer un merge…

Git, al tratarse de un sistema distribuido, resuelve este tipo de problemas de una forma muy elegante a través de los forks. Digamos que Pepito es una persona que quiere contribuir al proyecto. Ha encontrado un bug y sabe cómo corregirlo. Como propietario del repositorio me interesa que Pepito pueda enviarme el parche de forma rápida, que no pierda mucho tiempo. Si es así ¡Pepito estará encantado de colaborar con nosotros! ;-). Además, necesito que el proceso sea ágil, no quiero tener que invertir 5 horas de mi tiempo cada vez que tenga que hacer un merge del trabajo que Pepito me envíe. ¿Cómo resuelve git el problema?

1. Pepito hace un fork de mi repositorio, para lo que sólo necesito darle permiso de lectura.
2. Pepito trabaja en SU COPIA (fork) del repositorio. Como es suya, puede hacer lo que quiera, la puede borrar, corromper, dinamitar, reescribir la historia del proyecto… nos da lo mismo, es SU COPIA (fork).
3. Cuando Pepito termina de programar y testear el parche, me avisa de que ya lo tiene y me dice “En la rama parche\_de\_pepito de MI COPIA (fork), tienes el parche que corrige el Bug XXXX”.
4. Yo voy a su repositorio, miro lo que ha hecho y si está bien lo incorporo (merge) a mi repositorio, que es el original.

Las ventajas de este proceso son las siguientes:

1. Pepito trabaja con SU COPIA. En ningún momento le tengo que dar acceso al repositorio central.
2. El proceso de incorporación de los cambios de Pepito es muy sencillo. Si no hay conflictos en los ficheros puede que sea tan simple como ejecutar un par de comandos git.
3. Pepito tiene muy fácil contribuir, no le cuesta esfuerzo.
4. Yo puedo gestionar muy fácilmente las contribuciones de muchas personas ¡me cuesta muy poco trabajo!

Ahora cerrá los ojos e imaginá que esto mismo lo podés hacer con tus compañeros de trabajo, en tu equipo…

# Haciendo Fork de un repositorio en Github

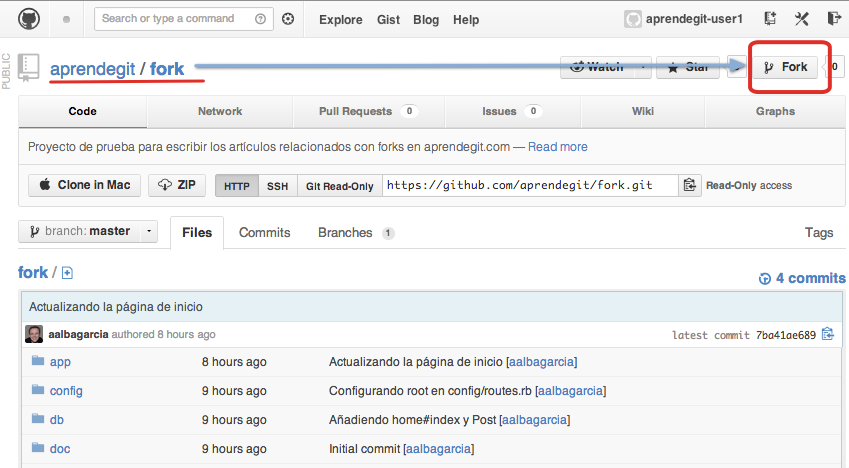
Bueno, vamos a ver esto con un ejemplo práctico. La situación es la siguiente: en nuestra organización, www.aprendegit.com, tenemos un repositorio de un proyecto RubyOnRails y que usaremos a modo de ejemplo. Este repositorio se encuentra en la siguiente URL: <https://github.com/aprendegit/fork>. Podéis reproducir los pasos que vamos a dar a continuación con vuestra cuenta de github.

Para ayudarnos con el desarrollo de este proyecto, contamos con la ayuda de varias personas:

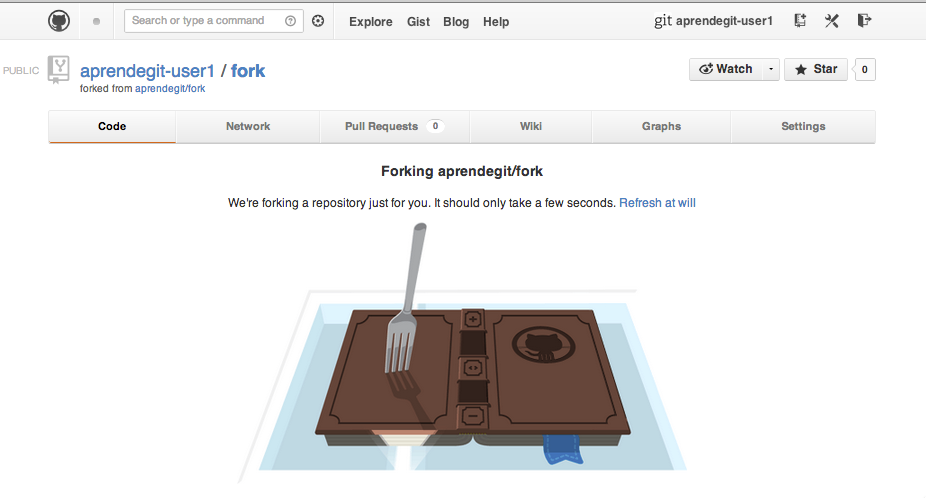
1. El usuario aalbagarcia (que soy yo) y que es uno de los administradores del repositorio y de la organización.
2. El usuario aprendegit-user1, que es un colaborador altruista que quiere ayudarnos con la web.
3. Ustedes: si hacen el tutorial y siguen los pasos (tengan en cuenta que habrán dos entregas más) terminarán enviando sus “mejoras ficticias” a través de un pull request.

No dudamos de las buenas intenciones de aprendegit-user1 ni de sus capacidades y aptitudes (ni de las de ustedes tampoco). Aún así, siguiendo las

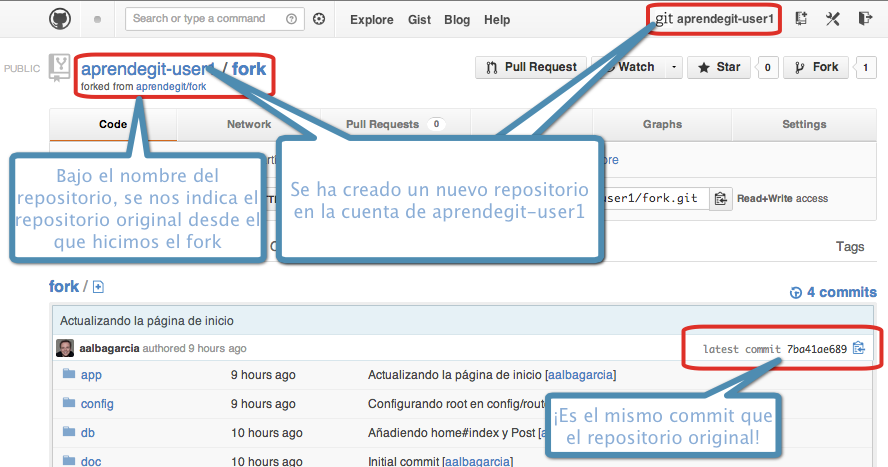
buenas prácticas en el uso de git, **aprendegit-user1 sólo tendrá permiso de lectura en el repositorio**[**https://github.com/aprendegit/fork**](https://github.com/aprendegit/fork)**.** Si no tiene permiso de escritura, el usuario aprendegit-user1 podrá clonar el repositorio (porque es público) pero no podrá hacer un push. De poco le va a servir clonarlo y trabajar sobre él si luego no va a poder hacer un push ¿qué debe hacer? Hacer un fork.El usuario aprendegit-user1 va a github, accede con su usuario y contraseña y abre la URL del repositorio original [**https://github.com/aprendegit/fork**](https://github.com/aprendegit/fork):

[](http://aprendegit.com/wp-content/uploads/2013/01/Voila_Capture319.png)

En la página del repositorio, aprendegit-user1 hace clic sobre el botón fork. Al hacer clic, Github va a crear un nuevo repositorio en su cuenta que es una copia del repositorio aprendegit/fork:

[](http://aprendegit.com/wp-content/uploads/2013/01/Voila_Capture322.png)

Cuando el proceso de forking termina (puede durar varios minutos si el repositorio que estamos bifurcando es muy grande) aprendegit-user1 acaba en la siguiente pantalla:

[](http://aprendegit.com/wp-content/uploads/2013/01/Voila_Capture3232.png)

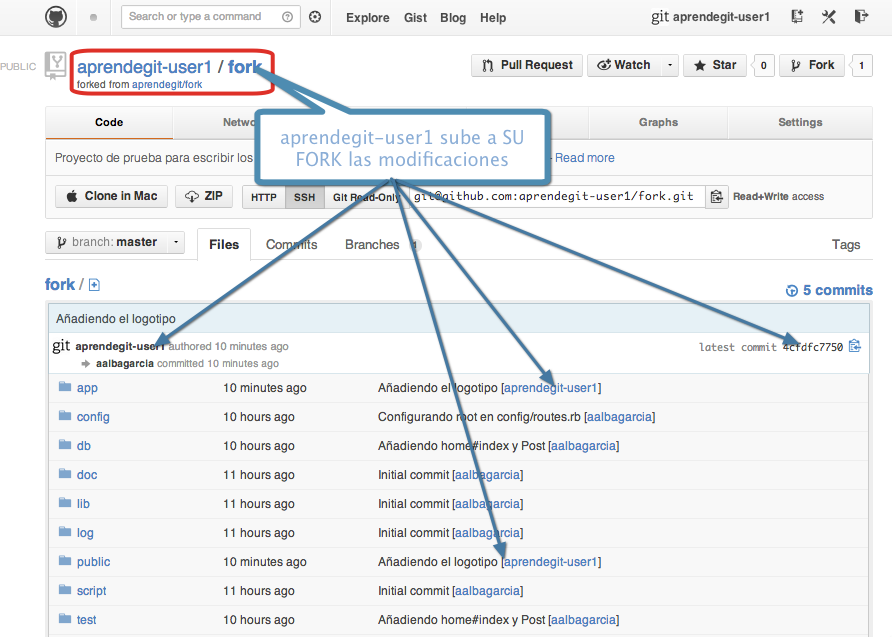
Este repositorio es una copia del original ([**https://github.com/aprendegit/fork**](https://github.com/aprendegit/fork)). Si comparáis la historia de los dos repositorios veréis que es idéntica.

Este nuevo repositorio (<https://github.com/aprendegit-user1/fork>) es el que nuestro usuario aprendegit-user1 va a utilizar para trabajar y, cuando termine, enviarnos sus modificaciones.

# aprendegit-user1 empieza a colaborar con nosotros

La primera tarea que aprendegit-user1 va a hacer como colaborador nuestro es modificar la página de inicio añadiendo nuestro logo. Para ello, **clonará su fork y trabajará sobre él como si fuese un repositorio normal**. Dado que el fork está en su cuenta, él es el propietario y como tal  podrá hacer push sin ningún problema. Recordad que al repositorio original ([**https://github.com/aprendegit/fork**](https://github.com/aprendegit/fork)) este usuario no puede hacer push.

Al cabo de unos minutos, aprendegit-user1 ha clonado su repositorio (<https://github.com/aprendegit-user1/fork>) en su máquina, ha añadido el logo y ha hecho un push a su fork:

[](http://aprendegit.com/wp-content/uploads/2013/01/Voila_Capture324.png)

Si vuelven al repositorio original ([**https://github.com/aprendegit/fork**](https://github.com/aprendegit/fork)**)** verán que este sigue igual, las modificaciones que aprendegit-user1 ha subido están sólo en el fork de aprendegit-user1.

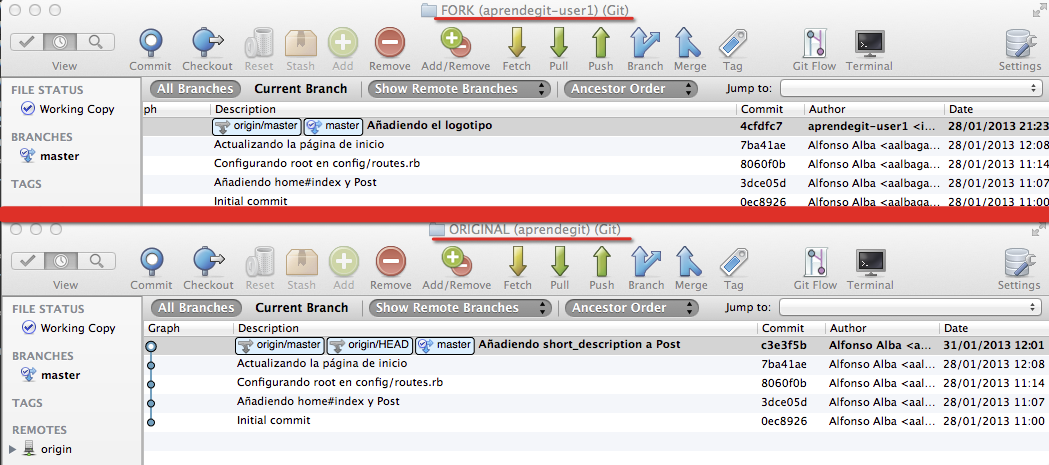
Misión cumplida: aprendegit-user1 ya está trabajando en su repositorio sin preocuparnos ninguno de los dos de que algo salga mal.

# Mantener tu fork al día

1. El usuario aprendegit-user1 hizo un fork del repositorio <https://github.com/aprendegit/fork>
2. El usuario aprendegit-user1 modificó la página de inicio del proyecto (ficticio) que estamos usando e hizo un push de sus modificaciones a su fork (<https://github.com/aprendegit-user1/fork>)
3. Vimos cómo las modificaciones que aprendegit-user1 hizo a su fork no se “sincronizan” automáticamente con el repositorio original

A la vez que prendegit-user1 ha avanzado con su tarea, nuestro primer usuario (aalbagarcia) y administrador del repositorio original (<https://github.com/aprendegit/fork>), ha continuado también trabajando. A la vez que el usuario aprendegit-user1 ha cambiado la página de inicio, aalbagarcia ha añadido un nuevo campo a la clase Post de nuestro blog. Cuando aalbagarcia ha acabado su trabajo, lo sube al repositorio original (<https://github.com/aprendegit/fork>) haciendo un push.

El estado de ambos repositorios es el siguiente:

[](http://aprendegit.com/wp-content/uploads/2013/01/Voila_Capture325.png)

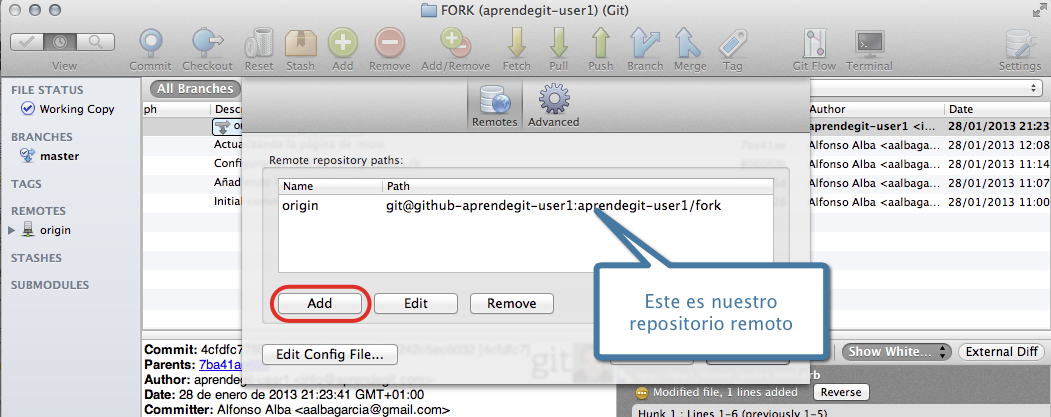
En la parte superior de la imagen tenemos una captura del estado del repositorio de aprendegit-user1 (fork) y en la parte inferior una captura del repositorio de aalbagarcia. Vemos que cada uno tiene en su repositorio su trabajo. Ha llegado la hora de sincronizarlos, cosa que haremos en dos pasos:

* Paso 1: aprendegit-usuario1 traerá a su fork el trabajo de aalbagarcia (en este caso el commit c3e3f5b que podéis ver arriba)
* Paso 2: aalbagarcia incluirá en el repositorio original el trabajo de aprendegit a través de un pull request.

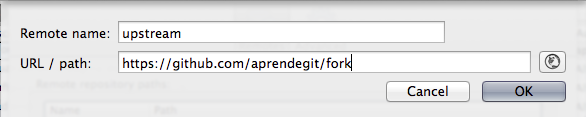
¡Empezamos por el Paso 1!

# Trabajando con múltiples repositorios remotos

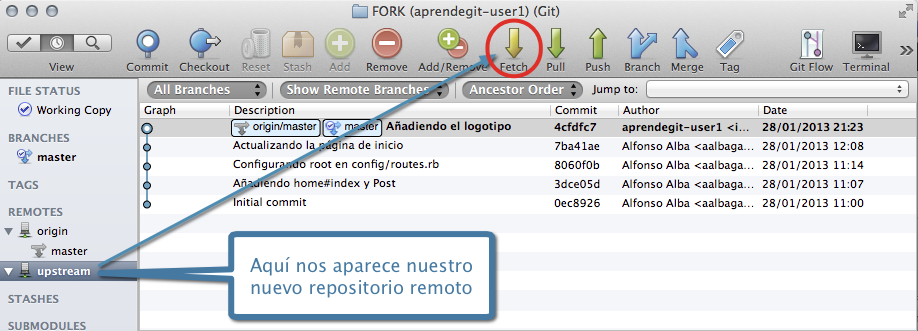
**Lo primero que aprendgit-user1 tiene que hacer es añadir el repositorio original a su lista de remotes**. El usuario aprendegit-user1 abre SourceTree y va a Repository->Repository settings:

[](http://aprendegit.com/wp-content/uploads/2013/01/Voila_Capture326.png)

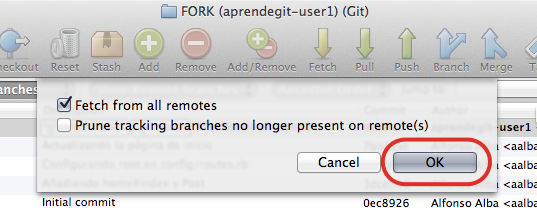
Dentro de la pantalla de “Repository settings”, selecciona la pestaña “Remotes”. En esta pantalla está su fork (origin). Se hace clic sobre el botón “Add” para añadir el repositorio original:

[](http://aprendegit.com/wp-content/uploads/2013/01/Voila_Capture327.png)

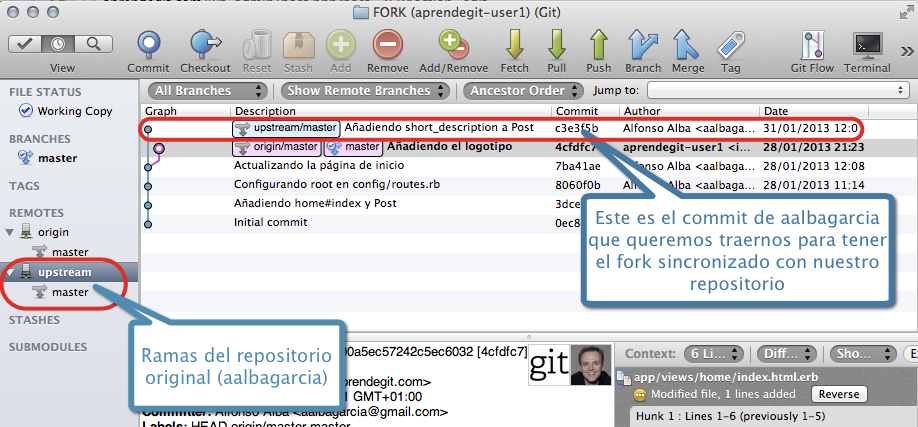
En la ventana emergente, se introduce “upstream” como nombre del remote (upstream es una sugerencia, podéis poner otro si lo preferís) y se pone la URL del repositorio original (<https://github.com/aprendegit/fork>). Rellenado el formulario, se hace clic en OK y de nuevo clic en OK en la ventana de preferencias. Una vez cerradas todas las ventanas de diálogo, el repositorio de aprendegit-user1 tendrá este aspecto:

[](http://aprendegit.com/wp-content/uploads/2013/01/Voila_Capture328.png)

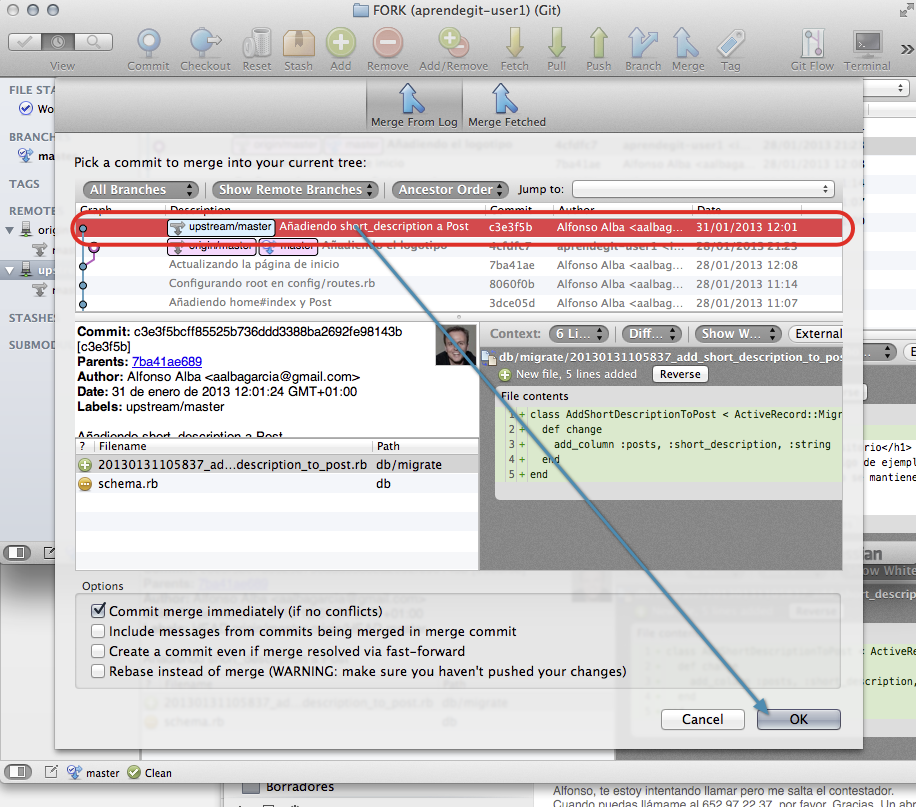
En la sección “REMOTES” aparece junto a origin el nuevo origen upstream, pero si se despliega el contenido veremos que está vacío. Pulsando sobre “Fetch” descargaremos el contenido del repositorio remoto “upstream”:

[](http://aprendegit.com/wp-content/uploads/2013/01/Voila_Capture329.png)

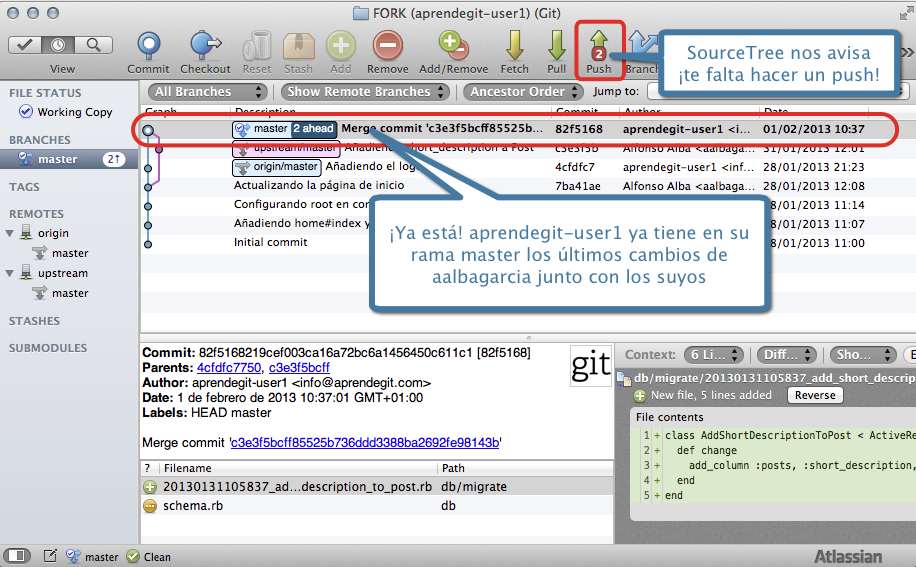
…y cuando el fetch finaliza, el repositorio queda de la siguiente manera:

[](http://aprendegit.com/wp-content/uploads/2013/01/Voila_Capture3301.png)

El fork de aprendegit-user1 ya muestra las ramas del repositorio original ¿véis el commit c3e3f5b? ¡ya lo tenemos ahí!. ¿Qué nos falta? Pues tan sólo hacer un merge de la rama upstream/master a la rama master para “sincronizar” el fork con el original. Aprendegit-user1 selecciona el commit correspondiente a nuestra rama master (en la captura anterior ya está seleccionado) y hace clic sobre el icono merge en la barra de herramientas:

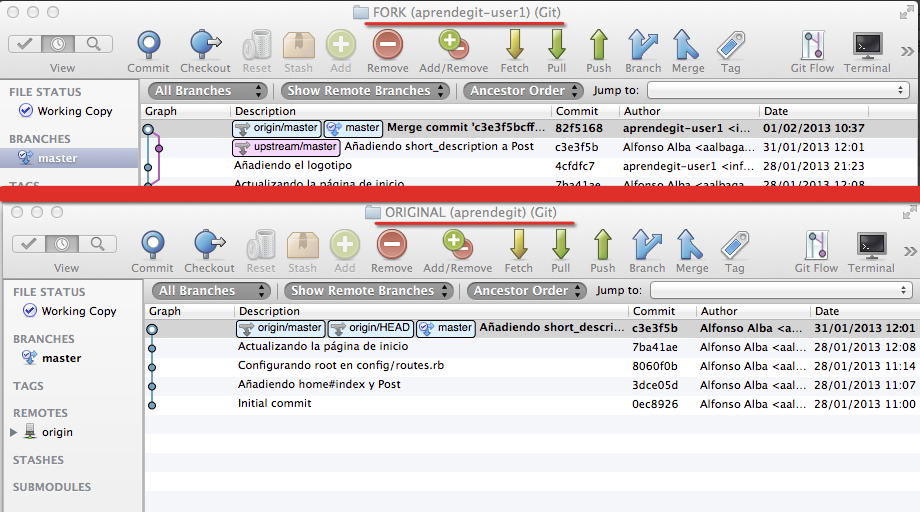
[](http://aprendegit.com/wp-content/uploads/2013/01/Voila_Capture331.png)

En la pantalla de merge, se selecciona el commit correspondiente a la rama upstream/master y se hace clic sobre OK. Cuando el merge termina, así queda el repositorio de aprendegit-user1:

[](http://aprendegit.com/wp-content/uploads/2013/01/Voila_Capture332.png)

¡Ya está! aprendegit-user1 ya tiene su rama sincronizada con el repositorio original. Ahora sólo queda hacer un push; como se puede ver el propio SourceTree nos avisa de que la rama master de aprendegit-user1 está dos commits por delante de la rama remota origin/master. Basta con hacer clic sobre el icono push para que aprendegit-user1 actualice su repositorio remoto.

Cuando aprendegit-user1 termina, así quedan su repositorio (fork) y el de aalbagarcia (original):

[](http://aprendegit.com/wp-content/uploads/2013/01/Voila_Capture333.png)

**¡aalbagarcia todavía no tiene en su repositorio el trabajo de aprendegit-user1!**Para eso, aprendegit-user1 hará un pull request… en la [siguiente entrada](http://aprendegit.com/que-es-un-pull-request/).

¡Happy gitting!