Colaboración + Pull Request

Te recomiendo que también leas los tutoriales de GitHub por si llegas a tener alguna duda extra. Si no, puedes utilizar el sistema de discusiones.

La parte de colaboración es un tema que se discute demasiado más sólo cuando lo pones en práctica, es cuando realmente las dudas aparecen. Es por ello que nos enfocaremos en despejarlas y en impulsarte a que te integres a los millones de proyectos colaborativos que GitHub te ofrece.

Clone vs Fork

Aunque ambos procesos son similares, tienen 2 finalidades distintas.

Clone. Clonar un repositorio, como su nombre lo indica, es tener una copia exacta del proyecto en tu disco duro. Más la intención va vinculada para fines de uso personales u organizacionales, no de colaboración. Como la función de descarga de un .ZIP.

Ejemplos claros puede ser la descarga de "Rails" para instalarlo en tu ordenador. Ó jQuery Mobile para utilizarlo en tu próximo proyecto Web.

El comando que se utiliza dentro de la consola ó terminal es:

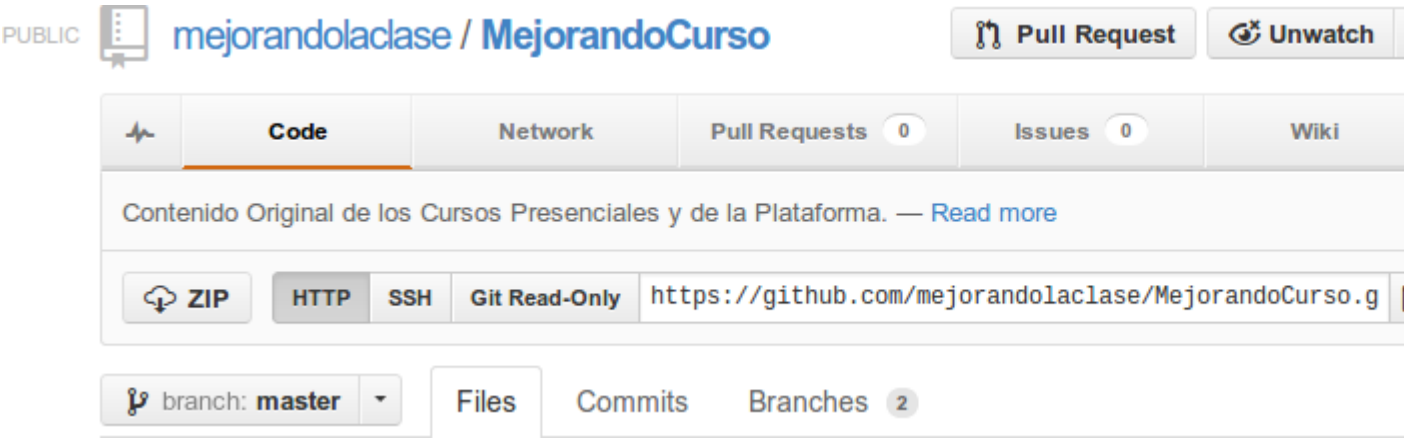
```
git clone https://github.com/rails/rails.git
```

Descargará el proyecto completo y ya podrás utilizarlo. ¿De dónde sale la dirección? En la página del proyecto, oprimiendo la opción "HTTP", te da el vínculo URL:

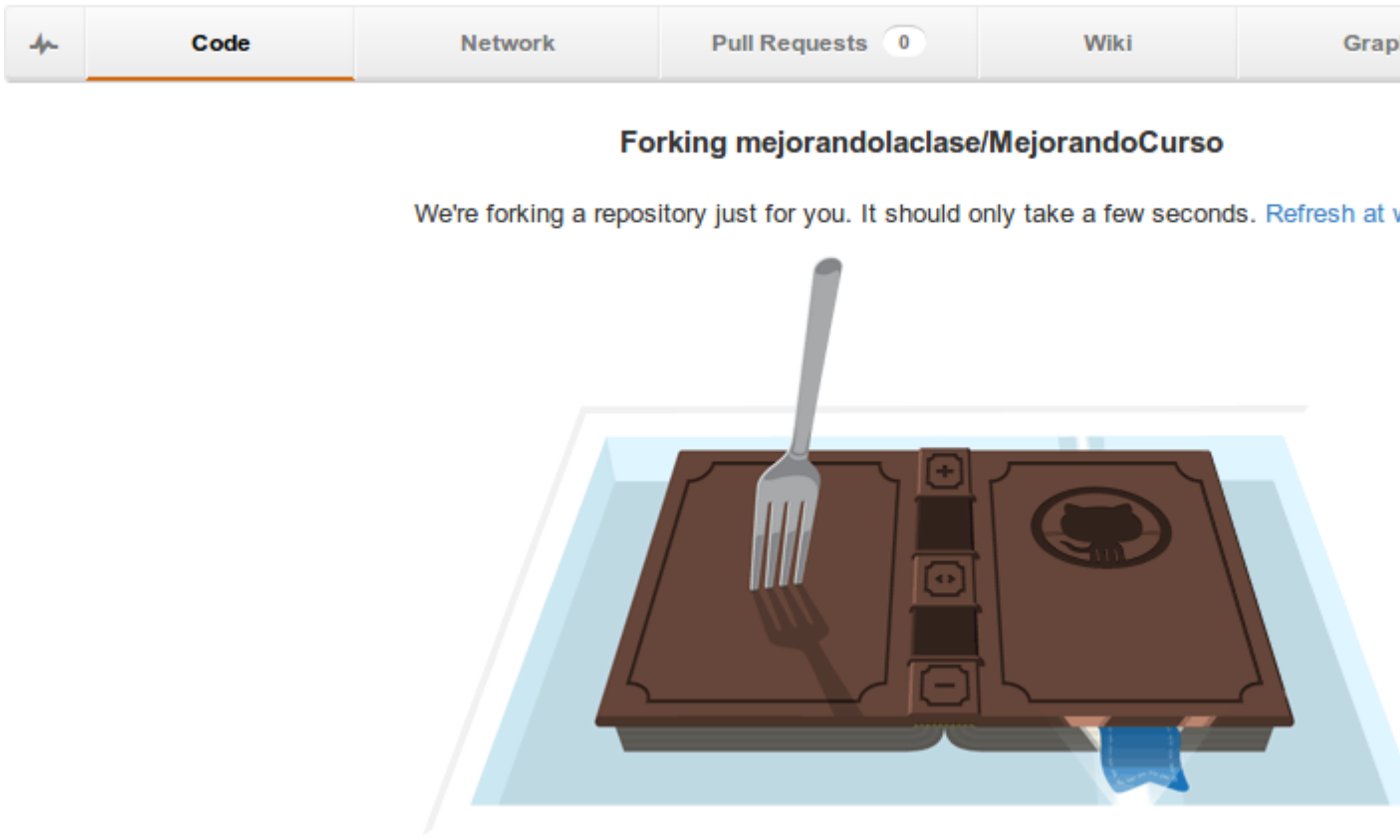
Fork (Colaboración). Este proceso habla de compartir y aportar. En ésta se crea un clón del repositorio original dentro de tu cuenta de GitHub y se colabora directamente desde ahí. Los pasos son los siguientes:

Descargando un repositorio

Lo primero que tienes que hacer es escoger un proyecto. En este caso será el de [mejorandocurso.]() Se puede ver el botón de Fork, el cual darás click para clonarlo a tu perfil:



En este momento, se estará generando una copia exacta del repositorio:



Conectar nuestro repositorio con el ordenador.

Debemos de generar una conexión entre tu ordenador y los repositorios para que se comuniquen y puedas enviar ó recibir actualizaciones. Para ello, clonaremos el repositorio de nuestro perfil (NO el original, el de nuestro perfil) con el comando "clone" en la terminal de nuestro ordenador:

```
git clone https://github.com/tuperfil/MejorandoCurso
# Cambia "tuperfil" por tu nombre de usuario en GitHub
```

Ahora bien, entramos a la carpeta donde clonaste el proyecto. Verás que tienes el proyecto completo. Con esto, ya puedes modificarlo. Realiza todos los cambios y avances que quieras aportar al repositorio.

Sincronizar un repositorio

Recurso Principal: Sync a Fork

Es posible que mientras avances en tu proyecto, se generen nuevas actualizaciones en el repositorio original. Sería muy desgastante estar haciendo "Fork" a cada momento para estar actualizado con el repositorio original. Por lo tanto, haremos que nuestro repositorio local (el que está en tu computadora) se conecte tanto al repositorio clon (el de tu perfil de GitHub) como el repositorio original (el perfil de mejorandocurso).

Para ello, quiero que introduzcas en tu consola, situado en la carpeta del proyecto:

```
$ git remote -v
origin    https://github.com/MiguelNieva/MejorandoCurso.git (fetch)
origin    https://github.com/MiguelNieva/MejorandoCurso.git (push)
```

Como podrás observar, el comando "remote" te permite saber a qué servidores estás conectado remotamente. Con esto, podemos ver que, debido a que realizamos git clone en el repositorio de nuestro perfil, la conexioń remota "origin" está vinculada con ella.

Lo que no está vinculado es el otro repositorio, el original. Por lo que desconocemos si están apareciendo nuevas actualizaciones, lo que tenemos que hacer es hacer otra conexión con este repositorio "original".

El remoto para conectarlo se llama "upstream". Por lo tanto, entramos a la consola y tecleamos:

```
$ git remote add upstream
https://github.com/mejorandolaclase/MejorandoCurso.git
```

Revisamos que estemos conectados:

```
$ git remote -v
origin    https://github.com/MiguelNieva/MejorandoCurso.git (fetch)
origin    https://github.com/MiguelNieva/MejorandoCurso.git (push)
upstream    https://github.com/mejorandolaclase/MejorandoCurso.git
(fetch)
upstream    https://github.com/mejorandolaclase/MejorandoCurso.git (push)
```

Genial, tenemos 2 conexiones remotas. Una con el repositorio original y otra con nuestro repositorio (donde nosotros mandaremos los cambios). Si te preguntas que es fetch y push, son 2 comandos que se utilizan para descargar la información ó enviarla.

Revisemos si hay información nueva en el repositorio original con:

```
$ git pull upstream
```

Nota importante: Existen otras formas de actualizar, como primero ejecutar fetch (descarga) y luego "merge" (fusión); ó utilizar "rebase" otro comando interesante alterno a "merge", pero eso lo veremos en el otro material de comandos con más detalle.

Si hay algún conflicto (ambas personas editaron el mismo archivo) es probable que lo tengan que reparar manualmente. La resolución de conflictos en GIT es muy sencillo, regularmente tendrás que abrir el archivo en el cual se generó el conflicto y encontrarás algo como esto:

```
<<<<<<< HEAD:index.html
<footer>contacto : m@mejorando.la</footer>
=======
<footer>
contacto: m@mejorando.la
</footer>
>>>>>>> iss53:index.html
```

Realmente, lo que significa es esto:

```
<<<<<<<
Cambios hechos en mi rama
=======
Cambios ajenos
>>>>>>>
```

Lo que debes hacer es sencillamente resolver el conflicto eligiendo el mejor para el proyecto, borrando todo lo demás y dejando el correcto:

```
<footer>contacto : m@mejorando.la</footer>
```

FInalmente, es cuestión de utilizar:

```
$ git add -A
$ git commit -m "Fusión"
```

Si no hay conflicto, no te preocupes, hará un "fast-forward" el cual anexará todos los cambios. Si requieres más información para la resolución de conflictos, entra al capítulo del manual de Git Oficial. Ó deja tu pregunta en el sistema de discusiones ;).

Enviar tus actualizaciones (Push)

Tenemos que enviar nuestras actualizaciones al repositorio clón (al que se encuentra en nuestro perfil). Utilizaremos:

```
$ git add -A
$ git commit -m "Propuesta de geolocalización"
$ git push origin master
```

Con esto, enviaremos todos los cambios a nuestro repositorio.

Pull Request

Te estarás preguntando: ¿Por qué mandamos todas nuestras actualizaciones a nuestro repositorio en GitHub en vez del original? Se debe al último movimiento de colaboración llamado "Pull Request". Como podrás observar, tendremos que ir al repositorio original (al del proyecto principal) y seleccionaremos "Pull Request":

Y lo que aparecerá es un selector que te pedirá qué propuesta te gustaría compartir con los propietarios del proyecto. Si ellos lo aceptan, se genera una fusión y habrás hecho la colaboración efectivamente.