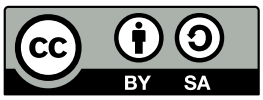
Apuntes de prácticas: Colaboración con Git

Antonio García Domínguez
nyoescape arroba gmail punto com

11 de agosto de 2008



Índice

1. Acciones básicas

1.1. Clonado de repositorios

Cuando varias personas han de colaborar en un mismo proyecto en el que se use Git, normalmente usarán un clon de algún repositorio destacado como punto inicial, creado mediante una orden del estilo de `git clone url`, donde la URL variará según qué protocolo usemos. Una vez se cree el clon, podemos indicar a Git la URL desde que clonamos utilizando el identificador «origin». Éste es un ejemplo de un «remote» de Git: una referencia por nombre a una URL con ciertos atributos adicionales configurables. Podemos añadir los nuestros propios mediante las órdenes `git remote add nombre url` y retirarlos con `git remote rm nombre`.

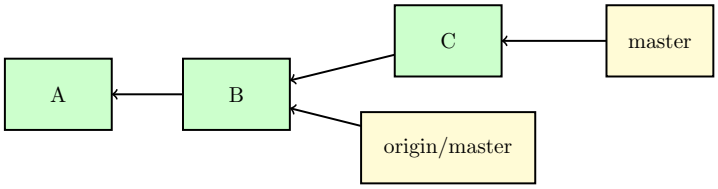
Nuestro clon del repositorio destacado incluirá, además de los objetos que en ese momento tenía el repositorio, referencias de sólo lectura a las revisiones en que se hallaban las ramas disponibles. Estas últimas son conocidas como *ramas remotas*, y para desarrollar a partir de ellas necesitaremos usar nuestras propias ramas locales: podemos comenzar nuevas ramas desde los puntos señalados por las ramas remotas, o reunir nuestras ramas locales con ellas.

1.2. Recepción de revisiones

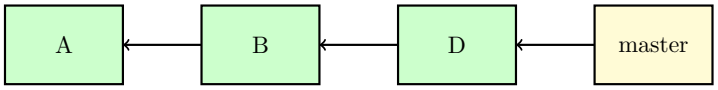
Periódicamente, podremos tomar los nuevos objetos que se hayan creado en cualquiera de los repositorios remotos que estemos monitorizando y añadirlos a nuestro repositorio, actualizando las ramas remotas. Para ello, usaremos la orden `git fetch [repositorio [refspec]]`. Según vayamos dando más o menos argumentos, su lógica cambiará:

- Sin argumentos, es equivalente a `git fetch origin`.
- Si se especifica sólo el repositorio, mediante su URL o el nombre de su *remote*, buscará su valor en varios sitios. Uno de ellos es la variable *remote.mirama.fetch* de `.git/config`, donde *mirama* es el nombre de la rama actual.

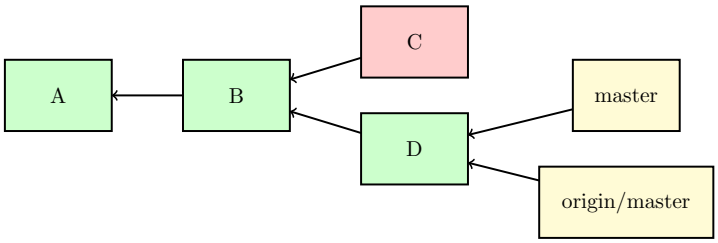
  Git aprovecha este comportamiento cuando clonamos un repositorio: al dar el valor correcto a la variable *remote.origin.fetch*, podemos hacer `git fetch` sin tener que pasar ningún argumento.
- Con los dos argumentos su comportamiento ya queda completamente definido. El segundo argumento es un tanto especial: la sintaxis es algo más compleja, teniendo la forma `[+]fuente[:destino]`. Su significado también se va construyendo poco a poco:
  - Con `fuente` a secas, decimos que queremos recibir la rama remota `fuente` y situar la punta en la referencia `FETCH_HEAD`. Esto es útil



(a) Local



(b) Remoto



(c) Local tras `git fetch origin +master:master`: se pierde `C`

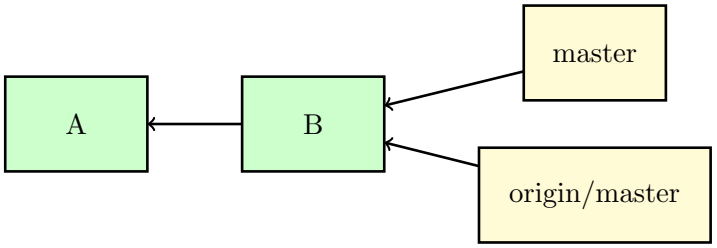
Figura 1: Problema introducido por actualizaciones no *fast forward*

sobre todo para ver qué cambios se han introducido en una rama remota sin que afecte a nuestro repositorio.

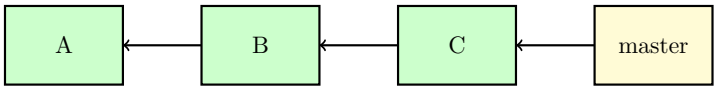
- `fuente:destino` toma la referencia fuente y actualiza o crea con ella la referencia destino. `refs/heads/*:refs/remotes/origin/*`, por ejemplo, es el valor configurado en *remote.origin.fetch* al clonar un repositorio. Este valor es el que indica a `git fetch` que actualice todas las ramas remotas con sus valores en el repositorio remoto, manteniendo los nombres.
- `+fuente:destino`, por otro lado, permite que aunque la punta de la fuente no sea descendiente de la punta actual del destino (es decir, no se pueda simplemente hacer un *fast forward*), se permita la actualización. Esta opción conlleva ciertos riesgos: puede verse en la figura 1 cómo forzar la actualización ha hecho que `C` quede sin referenciar por ninguna rama.

1.3. Reunión de las ramas locales con las remotas

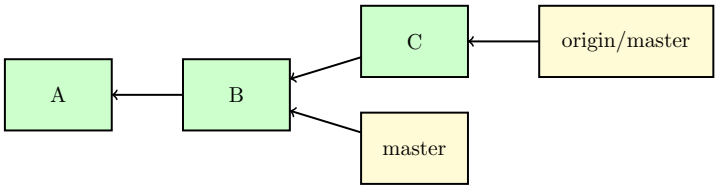
`git fetch` no reúne nuestras ramas locales con las ramas remotas (véase la figura 2 en la página siguiente): para ello tendremos que usar `git merge`, tal y como se vio en los apuntes de ramas. Podemos unir `git fetch` y `git merge` en una sola orden, `git pull [repositorio refspecs]`.



(a) Local



(b) Remoto



(c) Local tras `git fetch`: `master` no se mueve

Figura 2: Ejemplo de cómo `git fetch` no reúne ramas

> *Nota:* he estado intentando utilizar `git pull` especificando únicamente el repositorio y dejando las ramas en la configuración con la variable *branch.rama.merge*, pero parece depender del valor de la variable *branch.rama.remote*, y si ésta también se especifica ya no tiene sentido usar la versión de un solo argumento `git pull repositorio`. Si alguien tiene éxito en hacer esto que me envíe un correo, gracias.

En su forma completa con la referencia al repositorio y al menos el nombre de una rama de dicho repositorio, el proceso seguido por `git pull X Y...` es:

1. Primero se ejecuta `git fetch` con todos los argumentos recibidos para recibir los objetos del repositorio remoto.

2. Por último se reúne la rama actual con las puntas de todas las ramas referenciadas en los argumentos `Y...`, usando `git merge`. Si se pasa la opción `--rebase` se sustituye por `git rebase`. No olvidar que sólo puede usarse esta última orden si aún nadie tiene la parte del historial que va a ser afectada.

Cuando no decimos nada, `git pull` utiliza algunos ficheros de configuración, con ciertos cambios. Simplificando un poco, sería algo así:

1. La referencia al repositorio en cuestión se obtiene a partir de la variable local *branch.mirama.remote* de `.git/config`, suponiendo que nos hallamos en la rama `mirama`.

2. Los argumentos para `git fetch` se extraen de la variable local *remote.miremoto.fetch*, suponiendo que el remoto antes indicado era «miremoto».

3. Las ramas del repositorio remoto con que se debería reunir la rama actual se encuentran en la variable local *branch.mirama.merge*.

Estas tres variables son configuradas automáticamente para la rama principal y el *remote* «origin» creados tras un clonado. Si creamos una nueva rama local, tendremos que indicarle de qué *remote* y rama remota actualizarse. Suponiendo que queramos desarrollar a partir de la rama `mirama` del *remote* «miremoto» en una nueva rama `nuevarama`, tenemos varias opciones:

1. Ir indicándolo manualmente con más argumentos:

```
git pull miremoto mirama:nuevarama
```

2. Crear la rama con la opción `--track`. Hay varias formas de hacerlo. La más sencilla es:

```
git checkout -b nuevarama --track miremoto/mirama
```

   Se trata de un atajo para:

```
git branch --track nuevarama miremoto/mirama
git checkout nuevarama
```

3. Configurarlo manualmente. Hay varios posibles ficheros a modificar, pero si usáramos `.git/config`, por ejemplo, ejecutaríamos algo así:

```
git checkout -b nuevarama miremoto/mirama
git config branch.nuevarama.remote miremoto
git config branch.nuevarama.merge mirama
```

1.4. Envío de revisiones

Otras veces desearemos enviar nuestras propias revisiones a otro repositorio. Esta acción se realiza mediante la orden `git push [repositorio [refspecs]]`, y al igual que `git fetch`, su significado se va construyendo progresivamente a medida que se le pasan argumentos:

- `git push` sin argumentos equivale a `git push origin`.

- Si se ejecuta `git push nombre`, y existe una variable *remote.nombre.push* con una o más *refspecs* `Y...`, entonces la orden equivale a `git push nombre Y...`.
- Si se ejecuta `git push nombre` pero no existe la anterior variable, entonces equivale a `git push nombre :`.
- Una vez todos los argumentos quedan determinados, `git push X Y...` actualizaría cada una de las ramas destino con los objetos del repositorio local, recorriendo cada uno de los *refspec* proporcionados.

  Estos *refspec* son un tanto especiales, y difieren en bastantes cosas de los de `git fetch` (y por tanto de los de `git pull`). La sintaxis básica `[+]fuente[:destino]` sigue siendo admitida y funciona de la misma forma, pero hay algunas diferencias:

  - Si no damos el nombre del destino, se asume que es el mismo que el origen. Estas dos órdenes son equivalentes:

    ```
    git push origin foo
    git push origin foo:foo
    ```

  - La fuente puede ser cualquier *commitish*, y en particular podemos enviar etiquetas:

    ```
    git tag -a v1.0
    git push origin v1.0
    ```

    Si usamos un *commitish* a una revisión cualquiera, estaremos obligados a dar un destino:

    ```
    git push origin master~5:otrarama
    ```

  - Podemos omitir la fuente y dejar sólo el destino, borrando así la rama (`:mirama`)o etiqueta (`:v1.0`) indicada en el destino.
  - Podemos omitir tanto la fuente como el destino, quedando `:`, o `+:` para actualizar incluso cuando no sea un *fast forward*. El comportamiento de `git push` al recibir este argumento es un tanto peculiar: básicamente, cada rama en el repositorio indicado es actualizada desde la rama local con el mismo nombre, si existe.