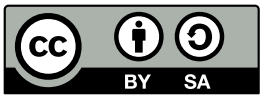
Guión de prácticas: Gestión de ramas*

Antonio García Domínguez

6 de agosto de 2008



Índice

*Sin preguntas por el momento.

1. Gestión básica de ramas

Las revisiones enviadas a un repositorio Git forman un grafo acíclico dirigido, en el que podemos tener líneas de revisiones que se dividen a partir de un cierto punto en múltiples ramas. Estas ramas posteriormente pueden reunirse opcionalmente, aunque no es estrictamente necesario.

Siempre estamos trabajando con una rama: por defecto, Git crea siempre la rama `master`, considerada normalmente como la rama principal de desarrollo (`trunk` para aquellos que conozcan Subversion). Podemos ver en qué rama estamos con:

```
git branch
```

Obtendremos una salida como:

```
* master
```

Esta salida indica que nos hallamos actualmente trabajando sobre la punta de la rama `master`, con lo que cualquier revisión que vayamos enviando no sólo creará el objeto correspondiente, sino que además hará avanzar el puntero `master` además del `HEAD`. Puede que no estemos sobre la punta de ninguna rama si hemos movido el `HEAD` manualmente (después veremos cómo). En dicho caso veríamos algo así:

```
* (no branch)
  master
```

Hay que tener cuidado: si creamos nuevas revisiones sin que sean alcanzables por una rama, estas revisiones no son alcanzables de forma normal y serán recolectadas como basura tras un período de gracia de 30 días por defecto. Podemos marcar la revisión actual como la punta de una rama (sin llegar a cambiarnos a ella) con:

```
git branch -a nombrerama
```

Por otro lado, podemos eliminar una rama (es decir, la referencia a su punta) con:

```
git branch -d nombrerama
```

Sin embargo, hay que tener cuidado en ciertos casos. Borrar la referencia a una rama que ya ha sido reunida con alguna otra no tiene problema, ya que sus revisiones son alcanzables desde la otra rama, como se ve en el caso de `develop` en 1 en la página siguiente. Sin embargo, si aún no se ha hecho esto, como en 2, podríamos acabar perdiendo las revisiones de dicha rama, al quedar inalcanzables por toda referencia.

Por ello, `git checkout -d develop` fallaría en el segundo caso, y si de verdad quisiéramos eliminar esa rama y descartar todas sus revisiones, sustituiríamos la opción `-d` por `-D`. Esto es útil, por ejemplo, para descartar una rama que hayamos visto improductiva por completo sin tener que reunirla.

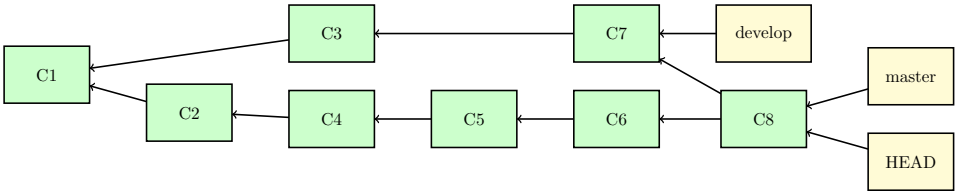


Figura 1: Situación no problemática al borrar la rama `develop`

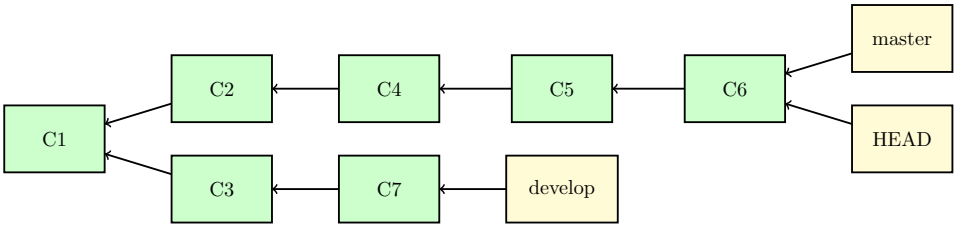


Figura 2: Situación problemática al borrar la rama `develop`

Se recuerda que para ver de forma cómoda el grafo de revisiones desde la revisión actual, podemos usar `gitk`, y para ver el grafo completo, se puede utilizar `gitk --all`.

2. Cambio entre ramas

3. Reunión de ramas

4. Reescribir varias revisiones en base a otra