Git, inicio rápido

David, Alpe Formación

22 de noviembre de 2018

Índice

1. ¿Por dónde empiezo?

1.1. clone

La forma más sencilla de empezar a trabajar con *git* es clonando un repositorio existente con el comando *clone*[1]

```
$ git clone https://github.com/dblancoformacion/latex_pruebas
```

Verás que se crea una carpeta nueva en el directorio en el que estás trabajando. Si accedes al directorio latex_pruebas con

```
$ cd latex_pruebas
```

tendrás control de tu copia local del repositorio.

1.2. log

El historial de cambios que se han realizado en el proyecto se pueden visualizar con:

```
$ git log --oneline --all
```

Obtendrás algo parecido a esto:

[1]Si aún no has instalado el *git* en tu equipo, visita https://git-scm.com/, instálate el software y, con el botón derecho en el escritorio o en cualquier otra carpeta en la que quieras trabajar con el repositorio a descargar, selecciona la opción Git Bash here y se abrirá el intérprete de comandos en el que podrás introducir los comandos que comentamos en este documento.

```
89a7615 (HEAD -> master, origin/master, origin/HEAD) Git push
7f84d5a Segunda modificación
881a3bd Inicio
```

1.3. pull

Para actualizar nuestro repositorio local, utilizamos el comando *pull*

```
$ git pull
```

2. Viajado en el tiempo

Hasta el momento, continuamos trabajando en el repositorio de otra persona, aún no estamos registrando nuestros propios cambios.

Que no los registremos, no impide que podamos realizarlos. Un repositorio descargado en nuestro equipo es perfectamente posible modificarlo.

2.1. diff

Con el comando *diff* podremos visualizar los cambios realizados en los ficheros del repositorio[2] desde el último *pull*

```
$ git diff
```

2.2. reset

Para descartar todos los cambios realizados y volver a la versión almacenada en el repositorio, tenemos el comando *reset*

[2]Si añadimos ficheros nuevos en el directorio de trabajo, *diff* no los visualizará hasta que se lo indiquemos con un *add*.

```
$ git reset --hard
```

ADVERTENCIA: Se perderán los cambios descartados.

2.3. checkout

A partir del historial de versiones obtenido en el apartado 1.2, pág. 2, podremos llevar nuestro directorio de trabajo al estado en el que se encontraba en cualquiera de las versiones almacenadas.

```
$ git checkout 7f84d5a
```

ADVERTENCIA: Modificar el pasado de la rama principal tendrá consecuencias sobre los commits siguientes

3. Creando nuestro propio repositorio

Hasta ahora, únicamente hemos visualizado repositorios de otros autores. Aunque es posible trabajar en un repositorio local para llevar un control de versiones de nuestros propios desarrollos, haremos uso de un repositorio en la nube para poder compartir nuestro trabajo.

Para poder subir nuestros contenidos a internet, debemos crear una cuenta de usuario en un alojamiento de repositorios en la nube como github.com o gitlab.com

Una vez creada la cuenta y generado un nuevo repositorio, podremos clonarlo en nuestro equipo y trabajar con una copia de repositorio en local con el comando *clone* como se ha visto en el apartado 1.1, pág. 2.

3.1. add

De momento, el repositorio estará vacío. Sólo contendrá la carpeta oculta .git, pero ya tendrá configurada la dirección del repositorio remoto y otros parámetros que, siguiendo estas instrucciones, no tendremos que configurar manualmente.

Podemos crear o copiar cualquier contenido en la carpeta de trabajo.

Para que *git* pueda visualizar los cambios realizados en un determinado fichero, debemos pedirle que lo tenga en cuenta[3] con el comando *add* y su nombre o, con un punto, para que realice el seguimiento de todos lo ficheros del directorio

```
$ git add .
```

3.2. commit

3.3. push

4. Cuando las cosas se complican

Ya somos capaces de mantener nuestro propio repositorio y compartirlo con nuestros seguidores. Si queremos localizar algún cambio en una versión anterior, visualizar los cambios entre dos versiones o ver cómo estaba el proyecto hace dos meses, ya sabemos hacerlo.

Lo que aún no sabemos es cómo gestionar una situación de este tipo:

- Tenemos una versión de nuestro libro publicada en la web y, mientras tanto, estamos escribiendo un nuevo capítulo que aún no queremos que se vea.
- Detectamos una errata en la versión publicada y nos gustaría corregirla, pero tampoco queremos perder todo lo que hemos trabajado en el nuevo capítulo. Es decir, no queremos modificar la versión publicada y que la nueva versión mantenga los antiguos errores, y no queremos repetir el mismo trabajo en ambas versiones.
- Cuando publiquemos la nueva versión con el nuevo capítulo, nos gustaría que todas las erratas de capítulos anteriores estuvieran corregidas.

[3] *Stage*

4.1. branch

4.2. merge

5. Trabajando en equipo