Uso avanzado de Git

Deshacer cambios

Deshaciendo cambios antes de la fase de staging.

Volvemos a la rama máster y vamos a modificar el comentario que pusimos:

```
$ git checkout master
Previous HEAD position was 3283e0d... Se añade un parámetro por defecto
Switched to branch 'master'
```

Recordamos, la situación es la siguiente:

```
+-------------+  +-------------+  +-------------+
|  Working    |  |   Staging   |  |    Local    |
|  Directory  |  |     Area    |  |  Repository |
+------+------+  +------+------+  +------+------+
       |                |                |
       |                |             hola.php (fd4da94) tag: v1
       |                |             hola.php (3283e0d) tag: v1-beta
       |                |             hola.php (efc252e)
       |                |             hola.php (e19f2c1)
       |                |                |
       +                +                +
```

Modificamos *hola.php* de la siguiente manera:

```
<?php
// Este comentario está mal y hay que borrarlo
$nombre = isset($argv[1]) ? $argv[1] : "Mundo";
@print "Hola, {$nombre}\n";
?>
```

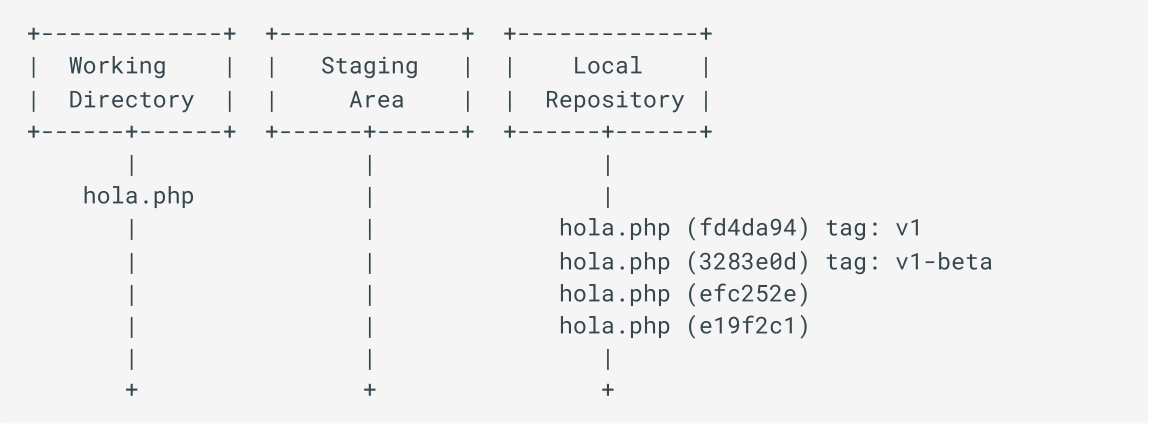
Y comprobamos:

```
    $ git status
    On branch master
    Changes not staged for commit:
      (use "git add <file>..." to update what will be committed)
      (use "git restore <file>..." to discard changes in working directory)

       modified:   hola.php

    no changes added to commit (use "git add" and/or "git commit -a")
```

Tenemos hola.php en Working Directory y nada en Staging Area.



El mismo Git nos indica que debemos hacer para añadir los cambios o para deshacerlos. En este caso los desharemos:

```
$ git restore hola.php

$ git status
On branch master
nothing to commit, working tree clean

$ cat hola.php
<?php
// El nombre por defecto es Mundo
$nombre = isset($argv[1]) ? $argv[1] : "Mundo";
@print "Hola, {$nombre}\n";
?>
```

Deshaciendo cambios antes del commit

Vamos a hacer lo mismo que la vez anterior, pero esta vez sí añadiremos el cambio al *staging* (sin hacer *commit*). Así que volvemos a modificar *hola.php* igual que la anterior ocasión:

```
<?php
// Este comentario está mal y hay que borrarlo
$nombre = isset($argv[1]) ? $argv[1] : "Mundo";
@print "Hola, {$nombre}\n";
?>
```

Y lo añadimos al *staging*

```
$ git add hola.php

$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
```

```
       modified:   hola.php
```

Ahora tenemos una nueva versión de hola.php en Staging Area.

```
+-------------+  +-------------+  +-------------+
|  Working    |  |   Staging   |  |    Local    |
|  Directory  |  |     Area    |  |  Repository |
+------+------+  +------+------+  +------+------+
       |                |                |
       |             hola.php            |
       |                |             hola.php (fd4da94) tag: v1
       |                |             hola.php (3283e0d) tag: v1-beta
       |                |             hola.php (efc252e)
       |                |             hola.php (e19f2c1)
       |                |                |
       +                +                +
```

De nuevo, Git nos indica qué debemos hacer para deshacer el cambio. Primero lo sacamos del Staging Area.

```
    $ git restore --staged hola.php

    $ git status
    On branch master
    Changes not staged for commit:
      (use "git add <file>..." to update what will be committed)
      (use "git restore <file>..." to discard changes in working directory)

       modified:   hola.php

    no changes added to commit (use "git add" and/or "git commit -a")
```

Vuelve a estar en Working Directory.

```
+-------------+  +-------------+  +-------------+
|  Working    |  |   Staging   |  |    Local    |
|  Directory  |  |     Area    |  |  Repository |
+------+------+  +------+------+  +------+------+
       |                |                |
    hola.php            |                |
       |                |             hola.php (fd4da94) tag: v1
       |                |             hola.php (3283e0d) tag: v1-beta
       |                |             hola.php (efc252e)
       |                |             hola.php (e19f2c1)
       |                |                |
       +                +                +
```

Y ahora restaruramos la última versión en Local Repository, eliminando la versión en Working Directory.

```
$ git restore hola.php
```

Y ya tenemos nuestro repositorio limpio otra vez. Como vemos hay que hacerlo en dos pasos: uno para pasar el fichero de Staging Area a Working Directory y limpiar así la Staging Area; y otro para descartar los cambios en Working Directory.

Deshaciendo commits no deseados.

Si a pesar de todo hemos hecho un commit y nos hemos equivocado, podemos deshacerlo con la orden `git revert`. Modificamos otra vez el archivo como antes:

```
<?php
// Este comentario está mal y hay que borrarlo
$nombre = isset($argv[1]) ? $argv[1] : "Mundo";
@print "Hola, {$nombre}\n";
?>
```

Pero ahora sí hacemos commit:

```
    $ git add hola.php

    $ git commit -m "Ups... este commit está mal."
    master 5a5d067] Ups... este commit está mal
     1 file changed, 1 insertion(+), 1 deletion(-)
```

Bien, una vez confirmado el cambio, vamos a deshacer el cambio con la orden `git revert`:

```
    $ git revert HEAD --no-edit
    [master 817407b] Revert "Ups... este commit está mal"
    1 file changed, 1 insertion(+), 1 deletion(-)
```

Explicación del comando:

- git revert HEAD: Reviertes el último commit (el que apunta HEAD). Esto crea un nuevo commit que deshace los cambios realizados en ese commit.
- --no-edit: Este parámetro le indica a Git que use el mensaje de commit por defecto que genera automáticamente (algo como "Revert 'mensaje original del commit'") y no abra el editor para modificarlo.

```
    $ git hist
    * 817407b 2013-06-16 | Revert "Ups... este commit está mal" (HEAD ->
master) [Sergio Gómez]
    * 5a5d067 2013-06-16 | Ups... este commit está mal [Sergio Gómez]
    * fd4da94 2013-06-16 | Se añade un comentario al cambio del valor por
defecto (tag: v1) [Sergio Gómez]
    * 3283e0d 2013-06-16 | Se añade un parámetro por defecto (tag: v1-beta)
[Sergio Gómez]
    * efc252e 2013-06-16 | Parametrización del programa [Sergio Gómez]
    * e19f2c1 2013-06-16 | Creación del proyecto [Sergio Gómez]
```

Borrar commits de una rama

El anterior apartado revierte un commit, pero deja huella en el historial de cambios. Para hacer que no aparezca hay que usar la orden `git reset`.

```
    $ git reset --hard v1
    HEAD is now at fd4da94 Se añade un comentario al cambio del valor por
defecto

    $ git hist
    * fd4da94 2013-06-16 | Se añade un comentario al cambio del valor por
defecto (HEAD -> master, tag: v1) [Sergio Góme
    * 3283e0d 2013-06-16 | Se añade un parámetro por defecto (tag: v1-beta)
[Sergio Gómez]
    * efc252e 2013-06-16 | Parametrización del programa [Sergio Gómez]
    * e19f2c1 2013-06-16 | Creación del proyecto [Sergio Gómez]
```

El resto de cambios no se han borrado (aún), simplemente no están accesibles porque git no sabe como referenciarlos. Si sabemos su hash podemos acceder aún a ellos. Pasado un tiempo, eventualmente Git tiene un recolector de basura que los borrará. Se puede evitar etiquetando el estado final.

Danger

La orden *reset* es una operación delicada. Debe evitarse si no se sabe bien lo que se está haciendo, sobre todo cuando se trabaja en repositorios compartidos, porque podríamos alterar la historia de cambios lo cual puede provocar problemas de sincronización.

Modificar un commit

Esto se usa cuando hemos olvidado añadir un cambio a un commit que acabamos de realizar. Tenemos nuestro archivo *hola.php* de la siguiente manera:

```php
<?php
// Autor: Sergio Gómez
// El nombre por defecto es Mundo
$nombre = isset($argv[1]) ? $argv[1] : "Mundo";
@print "Hola, {$nombre}\n";
?>
```

Y lo confirmamos:

```
$ git commit -a -m "Añadido el autor del programa"
[master cf405c1] Añadido el autor del programa
 1 file changed, 1 insertion(+)
```

> **Tip**
>
> El parámetro `-a` hace un `git add` antes de hacer *commit* de todos los archivos modificados o borrados (de los nuevos no), con lo que nos ahorramos un paso.

Ahora nos percatamos que se nos ha olvidado poner el correo electrónico. Así que volvemos a modificar nuestro archivo:

```
<?php
// Autor: Sergio Gómez <sergio@uco.es>
// El nombre por defecto es Mundo
$nombre = isset($argv[1]) ? $argv[1] : "Mundo";
@print "Hola, {$nombre}\n";
?>
```

Y en esta ocasión usamos `commit --amend` que nos permite modificar el último estado confirmado, sustituyéndolo por el estado actual:

```
    $ git add hola.php

    $ git commit --amend -m "Añadido el autor del programa y su email"
    [master 96a39df] Añadido el autor del programa y su email
     1 file changed, 1 insertion(+)

    $ git hist
    * 96a39df 2013-06-16 | Añadido el autor del programa y su email (HEAD ->
master) [Sergio Gómez]
    * fd4da94 2013-06-16 | Se añade un comentario al cambio del valor por
defecto (tag: v1) [Sergio Gómez]
    * 3283e0d 2013-06-16 | Se añade un parámetro por defecto (tag: v1-beta)
[Sergio Gómez]
    * efc252e 2013-06-16 | Parametrización del programa [Sergio Gómez]
    * e19f2c1 2013-06-16 | Creación del proyecto [Sergio Gómez]
```

> **Danger**
>
> Nunca modifiques un *commit* que ya hayas sincronizado con otro repositorio o que hayas recibido de él. Estarías alterando la historia de cambios y provocarías problemas de sincronización.

Moviendo y borrando archivos

Mover un archivo a otro directorio con git

Para mover archivos usaremos la orden `git mv`:

```
$ mkdir lib

$ git mv hola.php lib

$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

  renamed:    hola.php -> lib/hola.php
```

Mover y borrar archivos.

Podíamos haber hecho el paso anterior con la órden del sistema *mv* y el resultado hubiera sido el mismo. Lo siguiente es a modo de ejemplo y no es necesario que lo ejecutes:

```
$ mkdir lib
$ mv hola.php lib
$ git add lib/hola.php
$ git rm hola.php
```

Y, ahora sí, ya podemos guardar los cambios:

```
$ git commit -m "Movido hola.php a lib."
[master 8c2a509] Movido hola.php a lib.
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename hola.php => lib/hola.php (100%)
```

Info

Hasta aquí hemos aprendido los aspectos básicos de git trabajando en entorno local. Hemos instalado git, configurado sus parámetros globales, creado un proyecto y aprendido los 3 estados en los que puede estar un archivo. También hemos aprendido los comandos para incorporar cambios a la zona "staged" y a "working directory". En las próximas secciones aprenderemos a trabajar con ramas y a utilizar un repositorio compartido en GitHub