



# Revirtiendo Cambios

## 1. Introducción.

En el mundo del desarrollo no es raro encontrarse con la necesidad de revertir o deshacer cambios. Ya sea para corregir errores, retroceder a un estado anterior o simplemente explorar diferentes ramas de desarrollo, conocer las formas adecuadas de revertir cambios en Git es esencial.

Este documento tiene como objetivo ser una guía práctica en el arte de deshacer alteraciones en Git. Exploraremos varias técnicas que abarcan desde revertir cambios locales no confirmados hasta deshacer confirmaciones previas y restaurar archivos específicos. Descubriremos cómo utilizar comandos como `git checkout` y `git restore` de manera efectiva, así como otras estrategias que te permitirán mantener el control sobre tu repositorio en cualquier situación.

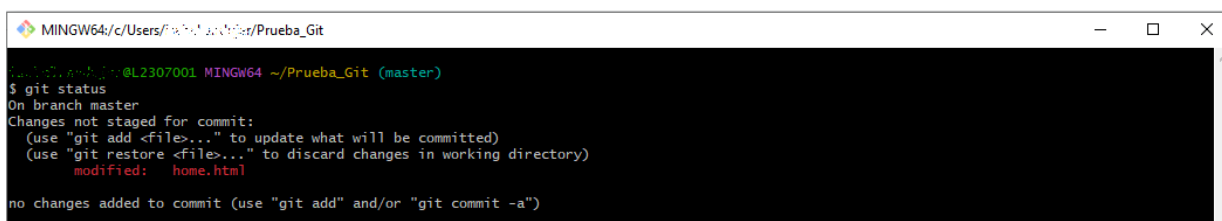
A lo largo de este documento, desentrañaremos los conceptos clave y proporcionaremos ejemplos prácticos para asegurarnos de que estés equipado con las habilidades necesarias para navegar por los desafíos que puedan surgir durante el desarrollo.

## 2. 'git restore'.

Para empezar, vamos a volver a modificar nuestro archivo 'home.html' y vamos a eliminar todo el cuerpo del documento, como si lo hubiéramos hecho sin querer:



```
home.html M
C: > Users > [usuario] > Prueba_Git > home.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE-edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Nter</title>
8  </head>
9  <body>
10 </body>
11 </html>
```



```
MINGW64/c:/Users/[usuario]/AppData/Local/Prueba_Git
$ 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:   home.html

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

Como podemos ver los cambios están hechos y GIT lo ha detectado, de hecho, si leemos con atención veremos que nos dice “use ‘git restore <file>’ to discard changes in working directory”, es decir, que si queremos deshacer los cambios que hemos hecho sin querer, solamente tenemos que ejecutar dicho comando, añadiendo el fichero del que queremos deshacer los cambios.



```

MINGW64/c/Users/fernando/Prueba_Git
fernando@L2307001 MINGW64 ~/Prueba_Git (master)
$ git restore home.html

fernando@L2307001 MINGW64 ~/Prueba_Git (master)
$ git status
On branch master
nothing to commit, working tree clean

```

Como vemos ahora ya no tenemos cambios en el working area y, si miramos nuestro archivo, vemos que ha vuelto al estado anterior:

```

home.html
C: > Users > fernando > Prueba_Git > home.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE-edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Nter</title>
8  </head>
9  <body>
10     <h1>Bienvenidos a Nter 3.0</h1>
11 </body>
12 </html>

```

### 3. 'git reset'.

El comando anterior está genial, pero ¿Qué ocurriría si los cambios realizados los hubiéramos enviado al staging área?

```

MINGW64/c/Users/fernando/Prueba_Git
fernando@L2307001 MINGW64 ~/Prueba_Git (master)
$ 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:   home.html

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

fernando@L2307001 MINGW64 ~/Prueba_Git (master)
$ git add .

fernando@L2307001 MINGW64 ~/Prueba_Git (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   home.html

fernando@L2307001 MINGW64 ~/Prueba_Git (master)
$ git restore home.html

fernando@L2307001 MINGW64 ~/Prueba_Git (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   home.html

```



Vemos que el comando **'git restore'** no tiene ningún efecto sobre los cambios. Por lo tanto, para revertir los cambios debemos usar otro comando. En este caso, usaremos el comando **'git reset'** que utilizará la información de algún **commit** anterior para revertir los cambios.

```

MINGW64/c/Users/fernando/Prueba_Git
fernando@L2307001 MINGW64 ~/Prueba_Git (master)
$ git log --oneline
9954a6e (HEAD -> master) Estamos trabajando en el proyecto
fe731e6 Comenzamos el proyecto

fernando@L2307001 MINGW64 ~/Prueba_Git (master)
$ git reset HEAD home.html
Unstaged changes after reset:
M   home.html

fernando@L2307001 MINGW64 ~/Prueba_Git (master)
$ 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:   home.html

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

```

Como vemos, GIT ha eliminado el archivo del staging area, devolviendo los cambios de este archivo al working directory. Esto quiere decir que ahora sí podemos utilizar nuestro comando de confianza **'git restore'** para eliminar los cambios.

```

MINGW64/c/Users/fernando/Prueba_Git
fernando@L2307001 MINGW64 ~/Prueba_Git (master)
$ git restore home.html

fernando@L2307001 MINGW64 ~/Prueba_Git (master)
$ git status
On branch master
nothing to commit, working tree clean

```



Cuando usamos **git reset** debemos tener en cuenta que:

- **'git reset nombre\_archivo'**: Este comando deshace los cambios que están en el área de preparación, pero deja los cambios en el directorio de trabajo sin afectar.
- **'git reset <hash\_del\_comit> --hard.'**: Este comando deshace los cambios en el área de preparación y también en el directorio de trabajo.
- **'git reset <hash\_del\_comit> --mixed.'**: Este comando deshace los cambios en el área de preparación pero los mantiene en el directorio de trabajo. Este es el comportamiento por defecto del comando si no se especifica ninguna etiqueta.
- **'git reset <hash\_del\_comit> --soft.'**: Este comando mantiene los cambios en el área de preparación

**HEAD** hace referencia al hash del último commit.

Para poder hacer algunas pruebas más con este comando vamos a realizar varios cambios en nuestro archivo y más commits, para así poder tener más opciones para hacer pruebas y entender mejor el concepto.



La intención ahora es utilizar el comando para volver al commit que le indiquemos. Debemos tener en cuenta que esto eliminará todos los commits que se hicieron después del indicado.

```

MINGW64~/c:/Users/fernandoj/Prueba_Git
fernandoj@L2307001 MINGW64 ~/Prueba_Git (master)
$ git log --oneline
1ff31bb (HEAD -> master) Añadimos párrafo sobre casos de éxito
30d262b Añadimos párrafo sobre tecnologías
b8ab302 Añadimos párrafo e introducción
9954a6e Estamos trabajando en el proyecto
fe731e6 Comenzamos el proyecto

fernandoj@L2307001 MINGW64 ~/Prueba_Git (master)
$ git reset b8ab302
Unstaged changes after reset:
M   home.html

fernandoj@L2307001 MINGW64 ~/Prueba_Git (master)
$ git log --oneline
b8ab302 (HEAD -> master) Añadimos párrafo e introducción
9954a6e Estamos trabajando en el proyecto
fe731e6 Comenzamos el proyecto

```

Como podemos ver, tras revertir los cambios, los commits anteriores ha desaparecido y podemos ver las diferencias con el código anterior.

```

MINGW64~/c:/Users/fernandoj/Prueba_Git
fernandoj@L2307001 MINGW64 ~/Prueba_Git (master)
$ git diff
diff --git a/home.html b/home.html
index 0c2e0a9..463e297 100644
--- a/home.html
+++ b/home.html
@@ -9,5 +9,7 @@
<body>
  <h1>Bienvenidos a Nter 3.0</h1>
  <p>Esta es nuestra página web.</p>
+  <p>Esta son nuestras tecnologías.</p>
+  <p>Esta son nuestros casos de éxito.</p>
</body>
</html>
\ No newline at end of file

```



Hay que tener en cuenta que el comando **git reset** es peligroso, ya que como hemos visto, elimina el resto de commits hechos posteriormente al indicado para revertir los cambios.

## 4. 'git revert'

Por último, veremos el comando '**git revert**' es otra herramienta en Git que se utiliza para deshacer cambios en el historial de commits, pero a diferencia de '**git reset**', no elimina los commits anteriores ni cambia el historial existente. '**git revert**' crea nuevos commits que deshacen los cambios introducidos por los commits que desees revertir.

En otras palabras, cuando ejecutas '**git revert**', se crea un nuevo commit que deshace los cambios introducidos por el commit especificado. Esto es útil cuando quieres mantener un historial de commits completo y no desees reescribir la historia del repositorio, especialmente si ya has compartido el historial con otros.



Cuando ejecutamos el comando **'git revert'** debemos tener en cuenta que a veces se pueden producir conflictos a la hora de intentar fusionar cambios en un archivo.

```

MINGW64/c/Users/fernando/Prueba_Git
$ git log --oneline
c5c53ee (HEAD -> master) Añadimos párrafo e introducción
50f7d35 Estamos trabajando en el proyecto
fe731e6 Comenzamos el proyecto

$ git revert 50f7d35
Auto-merging home.html
CONFLICT (content): Merge conflict in home.html
error: could not revert 50f7d35... Estamos trabajando en el proyecto
hint: After resolving the conflicts, mark them with
hint: "git add/rm <paths>", then run
hint: "git revert --continue".
hint: You can instead skip this commit with "git revert --skip".
hint: To abort and get back to the state before "git revert",
hint: run "git revert --abort".

$ git revert 50f7d35

```

Git nos está informando sobre el conflicto y a la vez nos proporciona sugerencias sobre cómo proceder.

Para resolver el conflicto, en primer lugar, debemos abrir nuestro archivo en el editor de código, Visual Studio Code:

```

C:\Users\fernando\Prueba_Git> home.html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Nter</title>
8  </head>
9  <body>
10     <h1>Bienvenidos a Nter</h1>
11 </body>
12 </html>
13
14

```

Hacemos click en **'Resolve in Merge Editor'** y se nos abrirá una nueva pestaña en la que nos aparecerá a la izquierda la versión futura del archivo, a la derecha la versión actual y, en la parte inferior, vemos el resultado final del merge. Una vez estemos de acuerdo con los cambios a realizar hacemos click en **'Complete merge'**.

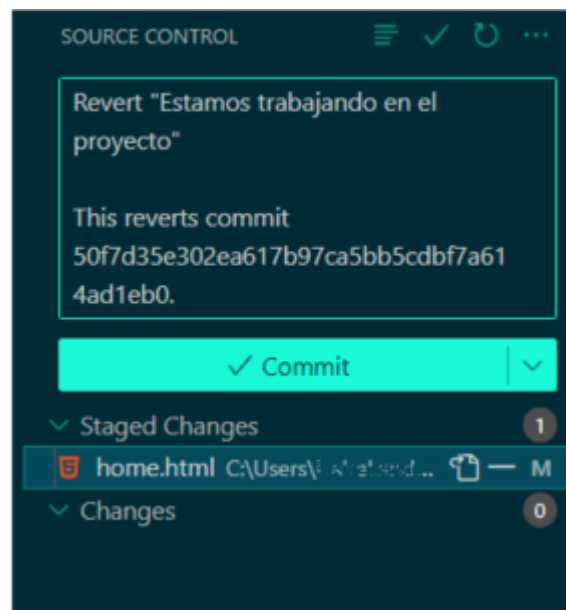


```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Nterc</title>
8 </head>
9 <body>
10  <h1>Bienvenidos a Nterc</h1>
11 </body>
12 </html>
13
```

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Nter</title>
8 </head>
9 <body>
10  <h1>Bienvenidos a Nter 3.0</h1>
11  <p>Esta es nuestra página web.</p>
12 </body>
13 </html>
```

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Nterc</title>
8 </head>
9 <body>
10  <h1>Bienvenidos a Nter</h1>
11  <p>Esta es nuestra página web.</p>
12 </body>
13 </html>
```

Por último, realizamos el commit con los cambios y con esto, habríamos terminado.



```
MINGW64/c:/Users/.../Prueba_Git
$ git log --oneline
6b2ac28 (HEAD -> master) Revert "Estamos trabajando en el proyecto"
c5c53ee Añadimos párrafo e introducción
50f7d35 Estamos trabajando en el proyecto
fe731e6 Comenzamos el proyecto
```

¡Listo! Ya estás preparado para revertir cambios en caso de ser necesario.