

Aprenda herramientas digitales

Usa herramientas digitales gratuitas para alcanzar objetivos personales y profesionales.

Applied Digital Skills

[Abrir](#)

Pro Git, el libro oficial de Git

2.4. Deshaciendo cosas

En cualquier momento puedes querer deshacer algo. En esta sección veremos algunas herramientas básicas para deshacer cambios. Ten cuidado, porque no siempre puedes volver atrás después de algunas de estas operaciones. Ésta es una de las pocas áreas de Git que pueden provocar que pierdas datos si haces las cosas incorrectamente.

2.4.1. Modificando tu última confirmación

Uno de los casos más comunes en el que quieres deshacer cambios es cuando confirmas demasiado pronto y te olvidas de añadir algún archivo, o te confundes al introducir el mensaje de confirmación. Si quieres volver a hacer la confirmación, puedes ejecutar un `commit` con la opción `--amend`:

```
$ git commit --amend
```

Este comando utiliza lo que haya en tu área de preparación para la confirmación. Si no has hecho ningún cambio desde la última confirmación (por ejemplo, si ejecutas este comando justo después de tu confirmación anterior), esta instantánea será exactamente igual, y lo único que cambiarás será el mensaje de confirmación.

Se lanzará el editor de texto para que introduzcas tu mensaje, pero ya contendrá el mensaje de la confirmación anterior. Puedes editar el mensaje, igual que siempre, pero se sobrescribirá tu confirmación anterior.

Top 5 CRM Software

More Than 5+ Employees? Your Business Needs CRM Software.
Compare Quotes Now!

Expert Market

[Open](#)

Por ejemplo, si confirmas y luego te das cuenta de que se te olvidó preparar los cambios en uno de los archivos que querías añadir, puedes hacer algo así:

```
$ git commit -m 'initial commit'
$ git add forgotten_file
$ git commit --amend
```

Estos tres comandos acabarán convirtiéndose en una única confirmación — la segunda confirmación reemplazará los resultados de la primera —.

2.4.2. Deshaciendo la preparación de un archivo

Las dos secciones siguientes muestran como pelearse con las modificaciones del área de preparación y del directorio de trabajo. Lo bueno es que el comando que usas para determinar el estado de ambas áreas te recuerda como deshacer sus modificaciones. Por ejemplo, digamos que has modificado dos archivos, y quieres confirmarlos como cambios separados, pero tecleas accidentalmente `git add *` y preparas ambos. ¿Cómo

Indice de contenidos

1. Empezando

Capítulo 2. Fundamentos de Git

2.1. Obteniendo un repositorio Git

2.2. Guardando cambios en el repositorio

2.3. Viendo el histórico de confirmaciones

2.4. Deshaciendo cosas

2.5. Trabajando con repositorios remotos

2.6. Creando etiquetas

2.7. Consejos y trucos

2.8. Resumen

3. Trabajando con ramas en Git

4. Git en un servidor

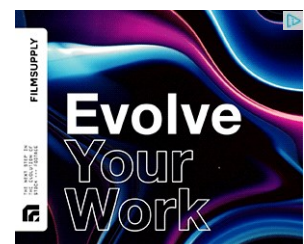
5. Git en entornos distribuidos

6. Las herramientas de Git

7. Personalizando Git

8. Git y otros sistemas

9. Funcionamiento interno de Git



puedes sacar uno de ellos del área de preparación? El comando `git status` te lo recuerda:

```
$ git add .
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   README.txt
#       modified:   benchmarks.rb
#
```



These Brilliant **Vehicle Trackers** Are Taking The US By Storm! **Expert Market**

Justo debajo de la cabecera “Cambios a confirmar” (“Changes to be committed”), dice que uses `git reset HEAD <archivo>...` para sacar un archivo del área de preparación. Vamos a aplicar ese consejo sobre `benchmarks.rb`:

```
$ git reset HEAD benchmarks.rb
benchmarks.rb: locally modified
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   README.txt
#
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   benchmarks.rb
#
```

El comando es un poco extraño, pero funciona. El archivo `benchmarks.rb` ahora está modificado, no preparado.

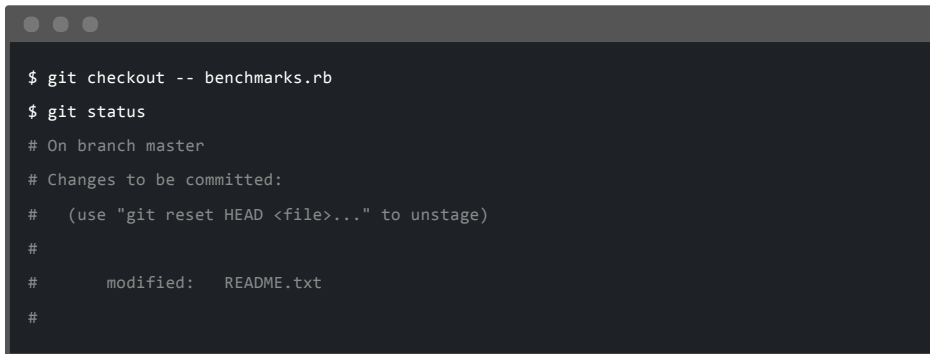
2.4.3. Deshaciendo la modificación de un archivo

¿Qué pasa si te das cuenta de que no quieres mantener las modificaciones que has hecho sobre el archivo `benchmarks.rb`? ¿Cómo puedes deshacerlas fácilmente — revertir el archivo al mismo estado en el que estaba cuando hiciste tu última confirmación — (o cuando clonaste el repositorio, o como quiera que metieses el archivo en tu directorio de trabajo)? Afortunadamente, `git status` también te dice como hacer esto. En la salida del último ejemplo, la cosa estaba así:

```
# Changed but not updated:
```

```
# (use "git add <file>..." to update what will be committed)
# (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   benchmarks.rb
#
```

Te dice de forma bastante explícita cómo descartar las modificaciones que hayas hecho (al menos las versiones de Git a partir de la 1.6.1 lo hacen — si tienes una versión más antigua, te recomendamos encarecidamente que la actualices para obtener algunas de estas mejoras de usabilidad —). Vamos a hacer lo que dice:

A terminal window with a dark background and light gray text. It shows the execution of two git commands. The first command is 'git checkout -- benchmarks.rb', which reverts changes to the benchmarks.rb file. The second command is 'git status', which shows the current state of the repository. The status indicates that the repository is on the master branch and that the README.txt file has been modified.

```
$ git checkout -- benchmarks.rb
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   README.txt
#
```

Puedes ver que se han revertido los cambios. También deberías ser consciente del peligro de este comando: cualquier modificación hecha sobre este archivo ha desaparecido — acabas de sobrescribirlo con otro archivo —. Nunca uses este comando a no ser que estés absolutamente seguro de que no quieres el archivo. Si lo único que necesitas es olvidarte de él momentáneamente, veremos los conceptos de apilamiento (*stashing*) y ramificación (*branching*) en el próximo capítulo; en general son formas más adecuadas de trabajar.

Recuerda, cualquier cosa que esté confirmada en Git casi siempre puede ser recuperada. Incluso confirmaciones sobre ramas que han sido eliminadas, o confirmaciones sobrescritas con la opción `--amend`, pueden recuperarse (véase el [Capítulo 9](#) para conocer más sobre recuperación de datos). Sin embargo, cualquier cosa que pierdas y que no estuviese confirmada, probablemente no vuelvas a verla nunca más.

Anterior

[2.3. Viendo el histórico de confirmaciones](#)

Siguiente

[2.5. Trabajando con repositorios remotos](#)