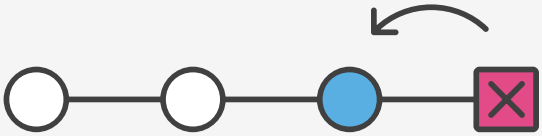
Aprende Git
Principiante
Empieza ya
Configuración de un repositorio
Guardar cambios
Examen de un repositorio
Deshacer cambios
git checkout
git clean
git revert
git reset
git rm
Reescribir el historial
Colaboración
Migración a Git
Consejos avanzados

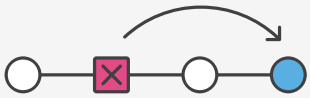


git revert

git checkout / git clean / git revert / git reset / git rm

Al comando `git revert` se le puede considerar un comando para "deshacer", pero lo cierto es que no lo es en el sentido tradicional. En lugar de eliminar la confirmación del historial del proyecto, resuelve cómo invertir los cambios introducidos por la confirmación y añade una nueva con el contenido inverso resultante. Así, se evita que Git pierda el historial, lo cual resulta importante para la integridad del historial de revisiones y para que la colaboración sea fiable.

Deberías usar la reversión cuando desees aplicar lo contrario a una confirmación del historial del proyecto. Esto puede ser útil, por ejemplo, si estás haciendo el seguimiento de un error y descubres que se introdujo mediante una sola confirmación. En vez de entrar, corregirlo y confirmar una nueva instantánea manualmente, puedes usar `git revert` para que todo este proceso se lleve a cabo de forma automática.



Funcionamiento

El comando `git revert` sirve para deshacer cambios efectuados en el historial de confirmaciones de un repositorio. Otros comandos para "deshacer" como, por ejemplo, git checkout y git reset, mueven los punteros de referencia `HEAD` y de la rama a una confirmación especificada. El comando `git revert` también toma una confirmación especificada, pero `git revert` no mueve los punteros de referencia a esta confirmación. Una operación de reversión tomará la confirmación especificada, invertirá los cambios de dicha confirmación y creará una "confirmación de reversión" nueva. Entonces, los punteros de referencia se actualizarán para apuntar a la nueva confirmación de reversión, lo cual la convertirá en la punta de la rama.

Para demostrarlo, vamos a crear un repositorio de muestra usando los siguientes ejemplos de líneas de comandos:

```
$ mkdir git_revert_test
$ cd git_revert_test/
$ git init .
Initialized empty Git repository in /git_revert_test/.
$ touch demo_file
$ git add demo_file
$ git commit -am"initial commit"
```

```
[main (root-commit) 299b15f] initial commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 demo_file
$ echo "initial content" >> demo_file
$ git commit -am"add new content to demo file"
[main 3602d88] add new content to demo file
n 1 file changed, 1 insertion(+)
$ echo "prepended line content" >> demo_file
$ git commit -am"prepend content to demo file"
[main 86bb32e] prepend content to demo file
 1 file changed, 1 insertion(+)
$ git log --oneline
86bb32e prepend content to demo file
3602d88 add new content to demo file
299b15f initial commit
```

Aquí hemos inicializado un repositorio en un directorio recién creado al que hemos llamado `git_revert_test`. Hemos efectuado 3 confirmaciones en el repositorio, mediante las cuales hemos añadido el archivo `demo_file` y hemos modificado su contenido dos veces. Al final del procedimiento de confirmación del repositorio, invocamos `git log` para mostrar el historial de confirmaciones, mediante lo cual mostramos un total de 3 confirmaciones. Con el repositorio en este estado, lo tenemos todo listo para iniciar un `git revert`.

```
$ git revert HEAD [main b9cd081] Revert "prepend conte
```

El comando `Git revert` espera que se haya efectuado una referencia a una confirmación, por lo que no se ejecutará si no la hay. En este ejemplo, hemos utilizado la referencia `HEAD`. Esta acción revertirá la última confirmación. Se trata del mismo comportamiento que si revirtiéramos a la confirmación `3602d8815dbfa78cd37cd4d189552764b5e96c58`. De forma semejante a lo que sucede con una fusión, una reversión creará una nueva confirmación, que abrirá el editor configurado del sistema para solicitar un nuevo mensaje de confirmación. En cuanto lo hayamos introducido y guardado, Git reanudará la operación. Llegados a este punto, podemos examinar el estado del repositorio mediante el comando `git log` y observar que se ha añadido una nueva confirmación al registro anterior:

```
$ git log --oneline 1061e79 Revert "prepend content to
```

Ten en cuenta que la tercera confirmación seguirá en el historial del proyecto tras la reversión. El vez de eliminarla, `git revert` ha añadido una nueva confirmación para deshacer los cambios. Por consiguiente, la segunda y la cuarta confirmaciones representan exactamente la misma base de código, y la tercera confirmación seguirá en el historial solo por si acaso deseamos volver a ella más adelante.

Opciones comunes

```
-e
--edit
```

Esta es una opción predeterminada que no hace falta especificar. Abrirá el editor configurado del sistema y te pedirá que edites el mensaje de confirmación antes de confirmar la reversión.

```
--no-edit
```

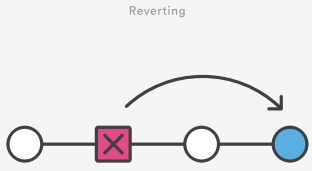
Esto es lo opuesto a la opción `-e`. La reversión no abrirá el editor.

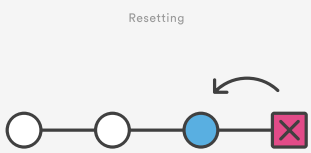
```
-n
--no-commit
```

Si utilizas esta opción, impedirás que `git revert` cree una nueva confirmación que invierta la confirmación objetivo. En vez de crear la nueva confirmación, esta opción añadirá los cambios inversos al índice del entorno de ensayo y al directorio de trabajo. Estos son los otros árboles que Git utiliza para gestionar el estado del repositorio. Para más información, visita la página sobre git reset.

Diferencia entre restablecer y revertir

Resulta primordial entender que `git revert` solo deshace una confirmación; no "revierte" el proyecto a un estadio anterior eliminando todas las confirmaciones posteriores. En Git, a esto último se le llama en realidad "restablecer", no "revertir".





Revertir presenta dos ventajas importantes con respecto a restablecer. En primer lugar, no cambia el historial del proyecto, lo que la convierte en una operación "segura" para las confirmaciones que ya se han publicado en un repositorio compartido. Para saber en detalle por qué es peligroso alterar un historial compartido, consulta la página sobre git reset.

En segundo lugar, el comando `git revert` puede dirigirse a una sola confirmación en un punto arbitrario del historial, mientras que `git reset` solo puede volver hacia atrás desde la confirmación actual. Por ejemplo, si quisieras deshacer una confirmación anterior mediante `git reset`, tendrías que eliminar todas las confirmaciones que se hubieran producido después de la confirmación a la que va destinada la acción, eliminarla y, acto seguido, volver a confirmar todas las confirmaciones posteriores. No hace falta decir que esta no es una solución nada elegante para deshacer acciones. Si quieres ver un análisis en mayor detalle sobre las diferencias entre `git revert` y otros comandos para "deshacer", consulta Restablecimiento, extracción y reversión.

Resumen

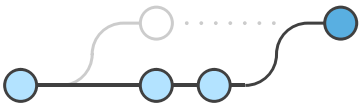
El comando `git revert` es una operación para deshacer de forma progresiva que ofrece una forma segura de deshacer los cambios. En vez de eliminar confirmaciones o dejarlas huérfanas en el

historial de confirmaciones, la reversión creará una nueva confirmación que invierte los cambios especificados. El comando `git revert` es una alternativa más segura que `git reset` en lo referente a la pérdida de trabajo. Para demostrar los efectos de `git revert`, hemos aprovechado otros comandos con una documentación más pormenorizada en sus respectivas páginas: git log, git commit y git reset .

¿Quieres aprender a usar git revert?

Prueba este tutorial interactivo.

Comienza ahora



Siguiente paso:

git reset

EMPEZAR EL SIGUIENTE TUTORIAL