

# Como eliminar de nuestro repositorio git un fichero o directorio

## Introducción

Un error muy común es que algunas veces, con las prisas olvidamos añadir a nuestro `.gitignore` algo que no está en el template con el que trabajamos.

No es correcto dejarnos ciertos ficheros o directorios en el repositorio.

## Como eliminar un fichero o directorio en el git (repositorio)

- Primero un backup de nuestro repositorio local, por favor. Siempre un backup.
- Añade el fichero o directorio al fichero `.gitignore`
- En tu `branch` comprueba que el fichero está fuera del master

```
git checkout master -- .gitignore
git add .
git commit -m 'Update .gitignore'
git pull
```

- Remueve el fichero o directorio del árbol de git

```
git rm --cached -r nombre_de_fichero_o_directorio
git add .
git commit -m 'Remove nombre_de_fichero_o_directorio'
git push
```

Esta operación sólo elimina del git (repositorio) el fichero, pero no de la historia, ni de las ramas.

# Como eliminar un fichero o directorio en el git (repositorio) en todas las ramas y en la historia

Alguna vez he visto un repositorio que por error contenía datos sensibles, en alguna parte de la historia.

Como norma general, se está usando un método complejo y que es propenso a errores utilizando `git filter-branch` pese a que recibimos un aviso a navegantes.

```
git filter-branch --tree-filter "rm -f myssh.sh" --prune-empty HEAD
WARNING: git-filter-branch has a glut of gotchas generating mangled history
rewrites. Hit Ctrl-C before proceeding to abort, then use an
alternative filtering tool such as 'git filter-repo'
(https://github.com/newren/git-filter-repo/) instead. See the
filter-branch manual page for more details; to squelch this warning,
set FILTER_BRANCH_SQUELCH_WARNING=1.
```

Ni se os ocurra. Es el mejor camino para tener que restaurar la copia y luego forzar un push. Todo ello, peligros para la estabilidad de tu repositorio.

Para hacerlo usaremos `git-filter-repo`, un paquete de **python3** que nos hará la vida más fácil con estas cosas, y que es recomendado por el propio git.

## Instalación de git-filter-repo

[Repositorio de git-filter-repo](#)

```
python3 -m pip install --user git-filter-repo
Collecting git-filter-repo
  Downloading git_filter_repo-2.29.0-py2.py3-none-any.whl (97 kB)
    |████████████████████████████████████████| 97 kB 967 kB/s
Installing collected packages: git-filter-repo
Successfully installed git-filter-repo-2.29.0
```

Añadimos el comando a nuestro `.bashrc` o `.zshrc` su path. En mi caso la instalación se efectuó en `:$HOME/.local/bin`

```
export PATH=$HOME/bin:$HOME/.local/bin:/usr/local/bin:$PATH
```

Hacemos un reload de nuestro RC o salimos de la sesión para aplicar los cambios.

```
reload
```

## Limpieza con git-filter-repo

Ahora, es hora de trabajar con el comando para limpiar nuestro repositorio.

```
> git-filter-repo --path myssh.sh --invert-paths --force
Parsed 108 commits
New history written in 0.06 seconds; now repacking/cleaning...
Repacking your repo and cleaning out old unneeded objects
HEAD está ahora en cce4227 Sin especificar
Enumerando objetos: 405, listo.
Contando objetos: 100% (405/405), listo.
Compresión delta usando hasta 12 hilos
Comprimiendo objetos: 100% (232/232), listo.
Escribiendo objetos: 100% (405/405), listo.
Total 405 (delta 232), reusado 299 (delta 157)
Completely finished after 0.18 seconds.
```

Como veis en la salida, el programa se ha encargado de viajar por la historia de los commits, y realizar la limpieza y cambiar la historia.

Ahora actualizaremos el remoto. Como veremos, los datos de nuestro `.git/config` han sufrido una actualización (que bueno tener backups si no tenemos o conocemos bien ciertas cosas)

```
git add .
git commit -m 'Update gitignore and clean'
[master 7d77a36] Update gitignore and clean
 1 file changed, 1 insertion(+), 1 deletion(-)
git push
fatal: No se ha configurado un destino para el push.
Puedes o especificar una URL desde la línea de comandos o configurar un repositorio remoto usando

    git remote add <nombre> <url>

y luego haciendo push al nombre del remoto

git push <nombre>
```

oops... no pasa nada. Se trata de reconstruir la parte que se ha eliminado de la configuración relativa al repositorio remoto.

```
[remote "origin"]
  url = git@gitlab.castris.com:root/nombre_del_repo.git
  fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
  remote = origin
```

Podemos hacerlo editando `.git/config` o vía comando

```
git remote add master git@gitlab.castris.com:root/utilidades.git
git remote add origin git@gitlab.castris.com:root/utilidades.git
git push -f --set-upstream origin master
Enumerando objetos: 405, listo.
Contando objetos: 100% (405/405), listo.
Compresión delta usando hasta 12 hilos
Comprimiendo objetos: 100% (157/157), listo.
Escribiendo objetos: 100% (405/405), 401.98 KiB | 100.50 MiB/s, listo.
Total 405 (delta 232), reusado 405 (delta 232)
remote: Resolving deltas: 100% (232/232), done.
To gitlab.castris.com:root/utilidades.git
+ 1ba0d77...cd97f4c master -> master (forced update)
```

Ahora ya podemos estar tranquilos. Hemos borrado los datos sensitivos de nuestro repositorio.

## Agradecimientos y enlaces interesantes

- [How to remove the .idea folder from git](#)
- [Remove folder and its contents from git/GitHub's history](#)
- [How do you install git-filter-repo?](#)
- [Remove file from git repository history](#)

## Aviso

Esta documentación y su contenido, no implica que funcione en tu caso o determinados casos. También implica que tienes conocimientos sobre lo que trata, y que en cualquier caso tienes copias de seguridad. El contenido el contenido se entrega, tal y como está, sin que ello implique ningún obligación ni responsabilidad por parte de [Castris](#)

Si necesitas soporte profesional puedes contratar con Castris [soporte profesional](#).

---

Revision #5

Created 2021-05-19 20:29:15 UTC by Abkrim

Updated 2023-02-06 19:04:35 UTC by Abkrim