

GIT

[devops, info](#)

related

- [.git*](#) mate
 - [Crear repositorio de solo lectura \(para algunos\)](#) mate
 - [git "rejected"](#) mate
 - [git \(altran\)](#) mate
 - [git \(git-book\)](#) mate
 - [git \(libro Amazon\)](#) mate
 - [git \(openwebinars\)](#) mate
 - [git autopush shell script](#) mate
 - [git CHERRY-PICK](#) mate
 - [git config](#) mate
 - [git DIFF](#) mate
 - [git diff\[tool\]](#) mate
 - [git FETCH](#) mate
 - [git LFS](#) mate
 - [git MERGE](#) mate
 - [git REBASE](#) mate
 - [Git rename branch \(master to main\)](#) mate
 - [git RESET](#) mate
 - [git REVERT](#) mate
 - [git STASH](#) mate
 - [git submodulos](#) mate
 - [git TAG](#) mate
 - [git tips](#) mate
 - [git: fusion varios repos en 1 conservando historia](#) mate
 - [git: migrar repositorio](#) mate
 - [git: replicación o duplicación de repositorios](#) mate
 - [GIT: unable to create thread: Resource temporarily unavailable](#) mate
- [git repo server](#)
 - [Anotaciones curso GIT](#)
 - [Crear repositorio de solo lectura \(para algunos\)](#)

cheatsheet

- <https://github.com/arslanbilal/git-cheat-sheet/blob/master/other-sheets/git-cheat-sheet-es.md>
 - [workflow-of-version-control.pdf](#)
 - [git-cheatsheet-en-white.pdf](#)
 - [git-cheatsheet-en-grey.pdf](#)
 - [git-cheatsheet-en-dark.pdf](#)
- [git-for-subversion-cheat-sheet.pdf](#)
 - [atlassian-git-cheatsheet.pdf](#)

- [gitcheatsheet.pdf](#)
- [github-git-cheat-sheet.pdf](#)
- [zt_git_cheat_sheet.pdf](#)
- <https://nvie.com/posts/a-successful-git-branching-model/>

doc

- <https://git-scm.com/book/es/v2>
- <https://esparta.github.io/gitimmersion-spanish/index.html>
- `.git*`

todo

git extras

<https://github.com/tj/git-extras/>

git submodules

<https://stackoverflow.com/questions/3796927/how-to-git-clone-including-submodules>

git resumen

<http://rogerdudler.github.io/git-guide/index.es.html>

git command tricks

<https://medium.freecodecamp.org/bash-shortcuts-to-enhance-your-git-workflow-5107d64ea0ff>

- git-config condicional :
<https://www.kevinkuszyk.com/2018/12/10/git-tips-6-using-git-with-multiple-email-addresses/>

config

- ubicación:
 - system : `/etc/gitconfig` [-system]
 - global : `~/.gitconfig` [-global]
 - repo : `.git/config`
 - **`git config --list --show-origin`**
- `git config [-global] -list` : si omitimos el ámbito, muestra todas las configuraciones disponibles
- `git config -global user.name «Mi Nombre»`
- `git config -global user.email «mi@email»`

- `git config --global http.sslVerify false` : no verifica el certificado en peticiones por *https*
- `git config --global core.editor «'$(which vim)'»` : asignar VIM como editor por defecto
- `git config credential.helper store` : store contraseñas (texto plano) autenticación
 - `git config --global credential.helper 'cache --timeout 7200'`
 - `git config credential.helper libsecret?`

otros

- [git alias](#)
- [git autopush shell script](#)
- [git DIFF](#)
- [fls](#)

errores

- [GIT: unable to create thread: Resource temporarily unavailable](#)

tips

- `HEAD == @`
- `HEAD^` : commit anterior al HEAD
- `HEAD@{1.month}`
- `-` (guión) : te devuelve a la rama que acabas de dejar

comandos interesantes

- [git: replicación o duplicación de repositorios](#)
- [git submodulos](#)
- [git: fusion varios repos en 1 conservando historia](#)
- [git diff\[tool\]](#)
- [git bitbucket tutorial](#)
- <https://ohshitgit.com/es>

git fetch VS git pull

- `git pull` = `git fetch` (+ `git checkout`) + `git merge`
- modo seguro:

```
git fetch origin
git log --oneline main..origin/main
git checkout main
git log origin/main
git merge origin/main
```

- <https://www.atlassian.com/es/git/tutorials/syncing/git-fetch>
- <https://www.atlassian.com/es/git/tutorials/using-branches/git-checkout>
- <https://www.atlassian.com/es/git/tutorials/using-branches/git-merge>
- <https://www.atlassian.com/es/git/tutorials/syncing/git-pull>

ramas

renombrado rama en local y remoto

```
git checkout <old_name> && git branch -m <new_name> # o git branch -m
<old_name> <new_name>
git push -u origin <new_name> # y comprobar en remoto que todo ha ido bien
git push origin :<old_name> # elimina la rama vieja del remoto!
```

mover rama, trayendo solo los commits que se hayan realizado

```
git ck -b rama_destino
git cherry-pick <commit1>..<commit2> # commit1^..commit2 para incluir commit1
en el movimiento
git push origin rama_destino # envíamos los cambios
git push origin :rama_origen # elimina la rama mal colocada
```

traer los cambios de rama develop a la actual

```
git pull --rebase origin develop
```

comandos con TAGs

```
git tag [-l]
git show <tag>
git tag -a <TAG> -m "<MENSAJE>" # tag local annotated
git push origin <TAG> # subir tag a remoto
git -d <tag> # borrar tag local
git push --delete origin <TAG> # borrado tag en remoto
```

actualizar listado ramas

```
git remote update origin --prune
git branch -a
```

/via: <https://stackoverflow.com/questions/36358265/when-does-git-refresh-the-list-of-remote-branches>

(ohshitgit) Sacar commit de una rama para llevarlo a una nueva

```
git branch nueva-rama
git reset HEAD~ --hard
git checkout nueva-rama
```

(ohshitgit) Commit en rama equivocada

```
# deshaz el último commit, pero deja los cambio disponibles
git reset HEAD~ --soft
git stash
# muevete a la rama correcta
```

```
git checkout nombre-de-la-rama-correcta
git stash pop
git add . # or add individual files
git commit -m "your message here";
# ahora tus cambios estan en la rama correcta
```

(ohshitgit) Commit en rama equivocada - cherry-pick

```
git checkout nombre-de-la-rama-correcta
# coge el último commit de master
git cherry-pick master
# borralo de master
git checkout master
git reset HEAD~ --hard
```

pequeños cambios en commits (no push)

(ohshitgit) agregar cambios al último commit

```
git add .
git commit --amend --no-edit
```

(ohshitgit) cambiar mensaje último commit

```
git commit --amend
```

retroceder o cambiar el pasado(rebase/reset)

(ohsgitgit) todo lo hecho, en todas las ramas

```
git reflog # cada entrada tiene un HEAD@{index}
```

(ohsgitgit) vuelta atrás

```
git reset HEAD@{index}
```

cambiar el mensaje de un commit (el último -1)

```
git rebase -i HEAD^ # y marcar con reword
```

fusionar varios commits

```
git rebase -i <commit>^ # con ^ se incluye el que indiquemos
git rebase -i HEAD~n # los últimos n
# marcar con SQUASH aquellos que queramos fusionar, dejando el mayor ancestro
como destinatario de todos ellos
```

```
# git push origin <rama> -f # fuerza la subida de los cambios
```

anular último commit (se pierde)

```
git rebase -i HEAD^ # y marcar con drop
```

anular último commit dejando en working area

```
git reset HEAD~1
```

ficheros

recuperar un fichero

```
git checkout <COMMIT|HEAD|HEAD^> <path_file>
```

(ohshitgit) recuperar un fichero

```
# busca el hash del commit anterior de cuando se cambio el archivo
git log
# usa las flechas para moverte para arriba y abajo en la historia
# una vez que encuentraste el commit, guarda su hash
git checkout [hash guardado] -- path/to/file
# la version anterior del archivo estará en tu index
git commit -m "Waw, no tienes que hacer copiar-pegar para deshacer"
```

ficheros modificados en commit concreto

```
git log --oneline --max-count=10
git diff-tree --no-commit-id --name-only -r <COMMIT_ID>
git show --pretty="" --name-only <COMMIT_ID>
```

seguir traza cambios a fichero

```
git log --follow -p -- <FILE>
git log --stat <FILE>
```

- <https://git-scm.com/docs/git-log>

```
git whatchanged [<FILE>]
```

- <https://git-scm.com/docs/git-whatchanged>
- mantenido por razones históricas

diferencias entre commits

- (ohshitgit) diferencias de ficheros en **staged**:

```
git diff --staged
```

- diferencia de ficheros entre dos commits:

```
git diff --name-only SHA1 SHA2
```

- generar ZIP con ficheros cambiados entre 2 commits:

```
git archive --output=file.zip HEAD $(git diff --name-only SHA1 SHA2)
```

- /via:<https://www.solucionex.com/blog/ficheros-modificados-entre-dos-commits-con-git>

recuperar

- localizar fichero:

```
git log -- <fichero>
```

- recuperar fichero:

```
git checkout <COMMIT_ID> -- <fichero>
```

- <https://recoverit.wondershare.es/file-recovery/recover-files-from-local-repository-git.html>

búsquedas

localizar fichero

```
git log -- <fichero>
```

seguir traza cambios a fichero

```
git log --follow -p -- <FILE>
```

buscar ficheros y estatus

```
git log --name-status -- "*<FILE>*"
```

- – indica a git que lo que viene a continuación son rutas de ficheros y no ramas

buscar líneas de código

```
git log -S"Hello, World!"
```

/via: <https://www.atlassian.com/git/tutorials/git-log>

From:

<https://miguelangel.torresegea.es/wiki/> - **miguel angel torres egea**

Permanent link:

<https://miguelangel.torresegea.es/wiki/tech:git:start?rev=1750327049>

Last update: **19/06/2025 02:57**

