Comandos de Git

Todos los comandos que a continuación se citan son modificadores del comando principal `git`.

Ej. `git status`

Ejecutando cualquier orden seguido del modificador -h se muestra una breve ayuda sobre la misma.

Ej. `git init -h`

Ejecutando cualquier orden seguido del modificador --help se muestra en el navegador por defecto una ayuda detallada sobre la misma.

Ej. `git clone --help`

A

add

Hace que un archivo pase a tener seguimiento y esté listo para hacer un `commit` sobre él.

Orden	Resultado
`git add README`	El archivo README pasa a tener seguimiento
`git add .`	Todos los archivos sin seguimiento pasan a tenerlo

B

branch

Permite trabajar con ramas.

Orden	Resultado
`git branch`	Lista las ramas del repo local (con * la que se está usando)
`git branch -r`	Lista las ramas del repositorio remoto
`git branch -a`	Lista las ramas en el repositorio local y remoto
`git branch <nombre_rama>`	Crea una rama en el repo local
`git branch <nombre_rama> -d`	Elimina una rama (ya fusionada) en un repo local
`git branch <nombre_rama> -D`	Elimina una rama (esté fusionada o no) en un repo local
`git branch <nombre_rama> -m <nuevo_nombre>`	Renombra una rama
`git branch <nombre_rama> <id_commit>`	Crea una nueva rama desde un commit existente
`git branch --no-merged`	Lista sólo ramas no fusionadas

C

checkout

Permite cambiar de rama y deshacer cambios

Orden	Resultado
`git checkout <nombre_rama_destino>`	Cambia de rama
`git checkout -b <nombre_rama>`	Crea una rama en el repo local y cambia a ella en un paso
`git checkout -- <archivo>`	Deshace los cambios realizados sobre un archivo con seguimiento que ha sido modificado (y aún no añadido).
`git checkout <id_commit>^ -- <nombre_archivo>`	Recupera el estado de un archive desde un *commit* pasado

cherry-pick

Copia un *commit* existente, creando un nuevo *commit* en la rama actual

Orden	Resultado
`git cherry-pick b578c`	Copia el commit de hash `b578c`, creando un nuevo *commit* en la rama actual, y tendrá el mismo mensaje y un hash diferente

clean

Elimina archivos no supervisados del directorio de trabajo

Orden	Resultado
`git clean -n`	No se elimina nada, sólo se muestra lo que se eliminaría
`git clean -f`	Fuerza la eliminación

clone

Se clona un repositorio. El repositorio de origen puede ser remoto o local; el destino será local

Orden	Resultado
`git clone <URL_repo_remoto>`	Se clona un repositorio remoto en uno local (éste tendrá el mismo nombre que el remoto)
`git clone <ruta_repo_local> <nombre_nuevo_repo_local>`	Se clona un repositorio local en otro local

commit

Confirma un cambio (lo hace definitivo) en archivos de un repositorio local.

Orden	Resultado
`git commit –m 'mensaje'`	Confirma el cambio de los recursos que se encuentren preparados para ello (en el *staging area*)
`git commit –a`	Fuerza un commit de todos los archivos modificados

config

Permite establecer y consultar parámetros de configuración (*global*, *system*, *local*)

Orden	Resultado
`git config –l`	Lista de todos los parámetros
`git config --global -l`	Lista de parámetros de configuración global (usuario)
`git config --system –l`	Lista de parámetros de configuración del sistema (todos los usuarios)
`git config --local –l`	Lista de parámetros de configuración local (repositorio)
`git config --add <parámetro> <valor>`	Establece un nuevo parámetro y su valor
`git config --get <parámetro>`	Obtiene el valor de un parámetro de configuración

D

diff

Muestra las diferencias entre dos estados del repositorio local

Orden	Resultado
`git diff`	Diferencias entre archivos que hayan sido modificados, o creados o eliminados, pero aun no añadidos al *staging área*
`git diff <id_commit_1> <id_commit_1>`	Muestra las diferencias existentes
`git diff --color`	

difftool

Trata de arrancar una herramienta externa de comparación de diferencias. Se configura en el parámetro `diff.tool`.

Orden	Resultado
`git difftool`	Trata de arrancar una herramienta externa de comparación de diferencias

F

fetch

Descarga las referencias actualizadas a un repo remoto

Orden	Resultado
`git fetch`	

G

gc

Elimina archivos innecesarios y optimiza el repositorio local. Se recomienda su uso de forma regular para optimizar el uso de disco y el rendimiento de las operaciones.

Orden	Resultado
`git gc`	Elimina archivos innecesarios y optimiza el repositorio local

gui

Trata de arrancar una herramienta gráfica para trabajar con Git.

Orden	Resultado
`git gui`	Trata de arrancar una herramienta gráfica para trabajar con Git

H

help

Muestra la ayuda.

Orden	Resultado
`git help`	Muestra la lista de comandos más habitual
`git help -a`	Lista de comandos extendida

hash-object

Muestra el hash de un objeto.

Orden	Resultado
`git hash-object <archivo>`	Muestra el hash del archivo

I

init

Crea un repositorio local.

Orden	Resultado
`git init <nombre_repositorio>`	Crea un nuevo repositorio en local sin ninguna rama (`NO-HEAD`)

L

log

Listado de los *commits* realizados en un repositorio.

Orden	Resultado
`git log`	Listado de los *commits* realizados en un repositorio
`git log –decorate`	Listado de los *commits* realizados en un repositorio más completo y bonito
`git log –graph`	Listado de los *commits* realizados en un repositorio en forma de grafo (commits como nodos, y aristas que los conectan)

ls-files

Listado de los archivos ubicados en el índice (*staging area*). Realmente, los archivos que son con seguimiento.

Orden	Resultado
`git ls-files`	Lista los archivos que con seguimiento.
`git ls-files -s`	Lista los archivos que con seguimiento, con su hash.

ls-remote

Listado de referencias en un repo remoto

Orden	Resultado
`git ls-remote`	Listado de referencias en un repo remoto

ls-tree

Listado de los archivos asociados a un *commit* determinado.

Orden	Resultado
`git ls-tree <id_commit>`	Listado de los archivos asociados a un *commit* (por su id)

M

merge

Fusiona ramas

Orden	Resultado
`git merge <nombre_rama>`	Fusiona la rama actual con la rama llamada <nombre_rama>

mergetool

Trata de arrancar una herramienta gráfica externa de resolución de conflictos. Se debe fijar esta herramienta dándole un valor al parámetro `merge.tool`.

Orden	Resultado
`git mergetool`	Trata de arrancar una herramienta gráfica externa de resolución de conflictos

mv

Mueve o renombra un archivo, un directorio o un enlace simbólico.

Orden	Resultado
`git mv`	

P

pull

Permite descargar el contenido del repositorio remoto a uno remoto.

Orden	Resultado
`git pull`	Descarga un repositorio remoto
`git pull --all`	

push

Permite subir el contenido de un repositorio local a uno remoto.

Orden	Resultado
`git push`	Sube los cambios del repo local al remoto
`git push --set-upstream origin master`	Establece que la rama master local se asociará con su equivalente remota en master y hace el `push`
`git push –u origin master`	Recomendable en un primer push, para vincular la rama local master con origin/master.

R

rebase

Permite coger todas las modificaciones aplicadas en una rama y re-aplicarlas en otra.

Orden	Resultado
`git rebase <nombre_rama>`	Aplica todos los *commits* de la rama actual en otra rama existente.

remote

Permite realizar operaciones sobre repositorios remotos.

Orden	Resultado
`git remote`	Si hay un repositorio remoto vinculado, se mostrará `origin` por defecto. Sino, no se mostrará nada.
`git remote –v`	Si hay un repositorio remoto vinculado, se mostrará la URL del mismo. Sino, no se mostrará nada.
`git remote show origin`	Muestra información detallada sobre el repositorio trmoto de nombre `origin`
`git remote add origin <url_repo_remoto>`	Se vincula un repo local con uno remoto, referenciándolo como `origin` (por convención).
`git remote rename origin destiny`	Renombra el repositorio remoto `origin`, llamándole `destiny`
`git remote remove origin`	Desvincula el repo local con el remoto llamado `origin`

reset

Fija el `HEAD` actual a otro estado

Orden	Resultado
`git reset HEAD <archivo>`	Permite sacar del *staging area* un archivo modificado que ha sido añadido después. Pasa a estar *unstaged*.

rev-list

Lista objetos *commit* en orden cronológico inverso.

Orden	Resultado
`git rev-list -n 1 HEAD -- <nombre_archivo>`	Determina el último *commit* en el que se alteró (añadió, modificó o eliminó) un archivo

rm

Elimina archivos sólo del índice o del índice y árbol de trabajo.

Orden	Resultado
`git rm <nombre_archivo>`	Elimina un archivo supervisado del árbol de trabajo y añade al índice su eliminación, para reflejarlo en el siguiente *commit*.
`git rm –f <nombre_archivo>`	Elimina un archivo no supervisado o modificado (sucio) del árbol de trabajo y del índice.
`git rm --cached <nombre_archivo>`	Elimina un archivo no supervisado o modificado (sucio) sólo del índice.
`git rm --cached –r <nombre_dir>`	Elimina un directorio de un repositorio remoto, pero no su copia en el local

S

show

Muestra información sobre el último *commit* y los archivos afectados.

Orden	Resultado
`git show`	Información sobre el último *commit*, y los cambios que produjo.
`git show 1a2b3c`	Información sobre el *commit* cuyo hash es `1a2b3c`, y los cambios que produjo.

`git show HEAD`	Información sobre el último *commit* y los cambios que produjo.
`git show HEAD~` `git show HEAD~1`	Información sobre el *commit* inmediatamente anterior (padre) al último, y los cambios que produjo.
`git show HEAD~~` `git show HEAD~2`	Información sobre el *commit* inmediatamente anterior al anterior (abuelo) al último, y los cambios que produjo.
`git show <id_tag>`	Información sobre un *tag*.

stash

Permite guardar (reservar) el trabajo a medias sobre una rama sin necesidad de aplicar *commit*. Tras un *stash*, el directorio de trabajo aparece "limpio".

Orden	**Resultado**
`git stash`	Reserva el trabajo sobre una rama y el directorio de trabajo aparece "limpio"
`git stash list`	Muestra las reservas que hay en la pila de reservas
`git stash apply`	Vuelve al estado existente en la última reserva apilada
`git stash apply stash@{1}`	Vuelve a la reserva etiquetada como `stash@{1}`
`git stash branch <nombre_rama>`	Recupera el estado reservado volcándolo en una nueva rama

status

Muestra el estado del repositorio local.

Orden	**Resultado**
`git status`	Muestra el estado del repositorio local

T

tag

Permite etiquetar *commits* relevantes para identificarlos más fácilmente después.

Orden	**Resultado**
`git tag`	Lista de etiquetas disponibles
`git tag –a <nombre_tag> -m '<mensaje_tag>'`	Crea una etiqueta anotada, asociándole un nombre y un mensaje
`git tag <nombre_tag>`	Crea una etiqueta ligera, asociándole un nombre
`git tag –d <etiqueta>`	Elimina una etiqueta

V

var

Muestra una variable lógica de Git.

Orden	**Resultado**
`git var –l`	Muestra la lista de todas las variables y sus valores
`git var <nombre_variable>`	Muestra el valore de una variable concreta

ANEXO I: Combinación de sentencias en Git

En Git se pueden combinar sentencias, de manera que se obtenga en una sola sentencia el efecto que en dos o más.

Ej. Para obtener el id del último *commit* en el que se alteró (creó, modificó, eliminó) el estado de un archivo de nombre `arch_01`, se ejecutará:

```
git rev-list -n 1 HEAD -- arch_01
```

El resultado es un id de *commit*.

Si a continuación se quisiera recuperar el estado del archivo en dicho *commit*, se ejecutará:

```
git checkout <id_commit_obtenido_antes>^ -- arch_01
```

Esto producirá que se recupere el estado del archivo llamado `arch_01` en el último *commit* donde se alteró.

También se podría haber conseguido el mismo efecto combinando ambas sentencias:

```
git checkout $(git rev-list -n 1 HEAD -- arch_01)^
      -- arch_01
```

La sentencia interior, rodeada de $(…) se ejecuta primero, y su resultado se le pasa a la sentencia exterior.