



## Inicio

<code>git --version</code>	Consultar versión de Git
<code>git help</code>	Ayuda
<code>git help comando</code>	Ayuda específica de un comando ( <b>q</b> para salir)

## Configuraciones típicas

<code>git config --global -e</code>	Ver configuración global
<code>git config --global user.name tuNombre</code>	Setear nombre
<code>git config --global user.email tuMail</code>	Setear mail
<code>git config core.autocrlf true</code>	Evitar <b>warning de CRLF</b>
<code>git config --global pull.ff only</code>	Estrategia "Fast-forward" por default para pulls remotos
<code>git config --global init.defaultBranch nombre-rama</code>	Setear nombre de rama por defecto

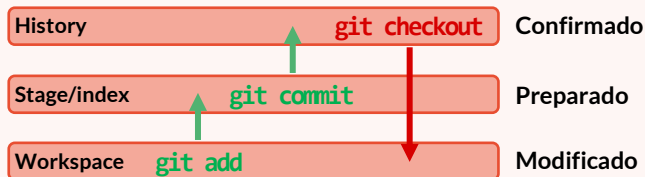
Crear/editar alias para comandos frecuentes

`git config --global alias.mi_alias "comando"`

Ejemplo: `git config --global alias.l "log --oneline"`

Ahora se puede usar `git l` para ver los commits de a uno por línea.

## Repositorio



<code>git init</code>	Inicializar repositorio
<code>git status</code>	Estado del repositorio
<code>git status -s</code>	Estado simplificado
<code>git status -s -b</code>	Estado simplificado con ramas
<code>git log</code>	Ver commits ( <b>q</b> para salir)
<code>git log --oneline</code>	Ver commits (una línea c/u)
<code>git log --decorate --all --graph --oneline</code>	Ver commits (graficado)
<code>git log ramaB..ramaA</code>	Commits en <b>ramaA</b> que no estén en <b>ramaB</b>
<code>git log --follow archivo</code>	Commits donde el <b>archivo</b> cambió

## Movimientos en el Stage

<code>git add .</code>	Agregar todos los archivos al <b>Stage</b> (Los tres comandos son equivalentes)
<code>git add -A</code>	
<code>git add --all</code>	
<code>git add arch1 arch2 ... archN</code>	Agregar archivos específicos
<code>git add *.extension</code>	Agregar archivos de extensión específica (solo en carpeta actual)
<code>git add carpeta/</code>	Agregar todos los archivos de una carpeta específica
<code>git add carpeta/*.extension</code>	Agregar archivos de una extensión específica en una carpeta específica
<code>git reset archivo</code>	Quitar archivo del <b>Stage</b>
<code>git restore --staged archivo</code>	(Los dos comandos son equivalentes)
<code>git mv nombre nuevo-nombre</code>	Cambiar nombre de archivo que ya esté en el <b>Stage</b>
<code>git rm archivo-o-carpeta</code>	Borrar archivo o carpeta
<code>git clean -f</code>	Borrar archivos sin seguimiento
<code>git clean -fd</code>	Borrar archivos y carpetas sin seguimiento

## Commit

<code>git commit -m "mensaje"</code>	Confirmar los cambios
<code>git commit -am "mensaje"</code>	Agregar al <b>Stage</b> y confirmar
<code>git commit --amend -m "nuevo-mensaje"</code>	Corregir mensaje de último commit
<code>git show hash-commit</code>	Ver detalle de commit
<code>git restore archivo</code>	Restablecer <b>archivo</b> al último commit
<code>git checkout -- archivo</code>	(Los dos comandos son equivalentes)
<code>git restore .</code>	Restablecer todos los archivos al último commit
<code>git checkout -- .</code>	(Los dos comandos son equivalentes)

## Etiquetas

<code>git tag</code>	Mostrar tags
<code>git tag nombre</code>	Crear tag
<code>git tag -a nombre -m mensaje</code>	Crear tag con mensaje
<code>git tag -a nombre hash -m mensaje</code>	Crear tag en commit específico
<code>git tag -d nombre-tag</code>	Borrar tag
<code>git show nombre-tag</code>	Mostrar tag en detalle

## Diff

<code>git diff</code>	Ver diferencias entre el <b>Workspace</b> y el <b>Stage</b>
<code>git diff --staged</code>	Ver diferencias entre el <b>Stage</b> y el <b>History</b>
<code>git diff rama1 rama2</code>	Ver diferencias entre ramas

## Ramas

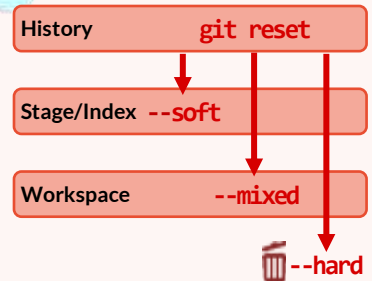
<code>git branch</code>	Ver ramas
<code>git branch -a</code>	Ver ramas (incluyendo remotas)
<code>git branch nombre-rama</code>	Crear rama
<code>git checkout rama</code>	Cambiar a otra rama
<code>git checkout -b nombre-rama</code>	Crear y cambiar a nueva rama
<code>git branch -m rama nuevo-nombre</code>	Modificar nombre de rama
<code>git branch -d rama</code>	Borrar rama
<code>git merge rama</code>	Fusionar con <b>rama</b> , en rama actual

## Volver en el tiempo

Modo **soft** (suave): Retrocede el puntero **HEAD** al commit especificado, pero no modifica el **Stage**

Modo **mixed** (mixto): Retrocede el puntero **HEAD** al commit especificado incluyendo también el **Stage**.

Modo **hard** (duro): Igual que **mixed**, pero también modifica de forma permanente los archivos y carpetas del proyecto. (Para deshacer, ver **reflog**)



<code>git reset --modo HEAD^</code>	Volver a commit anterior
<code>git reset --modo HEAD^N</code>	Volver hacia el <b>Nº</b> anterior commit
<code>git reset --modo hash-commit</code>	Volver hacia commit específico
<code>git revert hash-commit</code>	Volver hacia commit especificado y generar uno nuevo con su estado
<code>git reflog</code>	Mostrar historial del repositorio

## Remoto

**origin** es el alias que por defecto se le suele dar al repositorio remoto

<code>git clone url</code>	Clonar repositorio remoto
<code>git remote -v</code>	Ver repositorios remotos
<code>git remote add origin url</code>	Añadir nuevo repositorio remoto
<code>git remote set-url origin url</code>	Cambiar <b>url</b> del <b>origin</b>
<code>git push -u origin master</code>	Subir cambios al repositorio remoto
<code>git push</code> (Luego de la primera vez)	
<code>git push --tags</code>	Subir cambios incluyendo todos los tags
<code>git fetch</code>	Obtener cambios en el repositorio remoto, pero sin aplicar
<code>git pull origin master</code>	Aplicar cambios desde repositorio remoto
<code>git pull</code> (Si usamos <b>-u</b> en el <b>push</b> )	
<code>git merge origin/rama</code>	Fusionar con rama remota, en rama actual