Git CheatSheet

Versión
2023-03-26

Charly Cimino

Inicio

`git --version`	Consultar versión de Git
`git help`	Ayuda
`git help comando`	Ayuda específica de un comando (`q` *para salir*)

Configuraciones típicas

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

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

Movimientos en el Stage

`git add .`
`git add -A`
`git add --all`
	Agregar todos los archivos al **Stage**
	(Los tres comandos son equivalentes)

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

Commit

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

Etiquetas

`git tag`	Mostrar tags
`git tag nombre`	Crear tag
`git tag -a nombre -m mensaje`	Crear tag con mensaje
`git tag -a nombre hash -m mensaje`	Crear tag en commit específico
`git tag -d nombre-tag`	Borrar tag
`git show nombre-tag`	Mostrar tag en detalle

Diff

`git diff`	Ver diferencias entre el **Workspace** y el **Stage**
`git diff --staged`	Ver diferencias entre el **Stage** y el **History**
`git diff rama1 rama2`	Ver diferencias entre ramas

Ramas

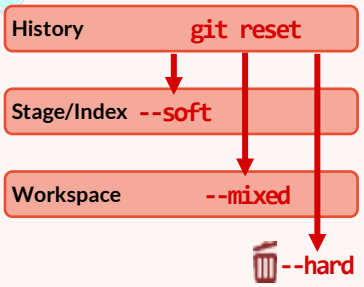
`git branch`	Ver ramas
`git branch -a`	Ver ramas (incluyendo remotas)
`git branch nombre-rama`	Crear rama
`git checkout rama`	Cambiar a otra rama
`git checkout -b nombre-rama`	Crear y cambiar a nueva rama
`git branch -m rama nuevo-nombre`	Modificar nombre de rama
`git branch -d rama`	Borrar rama
`git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



`git reset --modo HEAD^`	Volver a commit anterior
`git reset --modo HEAD^N`	Volver hacia el N° anterior commit
`git reset --modo hash-commit`	Volver hacia commit especifico
`git revert hash-commit`	Volver hacia commit especificado y generar uno nuevo con su estado
`git reflog`	Mostrar historial del repositorio

Remoto

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

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