



# Certified Tech Developer

The Ultimate Degree

## Comandos Git

### → Ayuda

- ☐ `git help`

### → Comando específico

- ☐ `git help add`
- ☐ `git help commit`
- ☐ `git help <cualquier_comando_git>`

### → Establecer el usuario y el e-mail

- ☐ `git config --global user.name "nombre de usuario"`
- ☐ `git config --global user.email email@email.com`

### → Eliminar todos los registros que se refieren al usuario y al e-mail

- ☐ `git config --global --unset user.name "nombre de usuario"`
- ☐ `git config --global --unset user.email email@email.com`

### → Ver la configuración de Git

- ☐ `git config --list`

### → Crear un nuevo repositorio

- ☐ `git init`

→ **Verificar el estado de los archivos/directorios**

- ❑ `git status` (muestra el estado de los archivos en su repositorio)

→ **Añadir un archivo**

- ❑ `git add nombre_archivo_directorio` (archivo específico)
- ❑ `git add . / git add --all` (todos los archivos)

→ **Commitear un archivo/directorio**

- ❑ `git commit nombre_archivo -m "mensaje del commit"`

→ **Remover un archivo o directorio**

- ❑ `git rm archivo`
- ❑ `git rm -r directorio` (remueve el directorio y los archivos que contiene)

→ **Ver el historial de actividad**

- ❑ `git log` (muestra el historial)
- ❑ `git log -- <ruta del archivo>` (muestra el historial de un archivo específico)
- ❑ `git log --author=usuario` (muestra el historial de un usuario en particular)

Deshacer operaciones

→ **Deshaciendo el cambio local en su directorio de trabajo local**

- ❑ `git checkout -- archivo` (solo debe usarse mientras el archivo no se haya añadido todavía al área de trabajo temporal)

→ **Deshaciendo el cambio local en el área de trabajo temporal (*staged area*)**

- ❑ `git reset HEAD archivo` (debe usarse cuando el archivo ya ha sido añadido en el área temporal)

*"Unstaged changes after reset:M archivo"* (si se muestra el siguiente resultado, el comando reset no ha cambiado el directorio de trabajo)

- ❑ `git checkout nombre_archivo` (permite realizar el cambio de directorio)

## Repositorio Remoto

### → Ver los repositorios remotos (para saber a dónde se envían los cambios o de dónde los descargamos)

- ❑ `git remote`
- ❑ `git remote -v`
- ❑ `git remote add origin git@github.com:minombre/archivo-git.git` (enlaza el repositorio local con un repositorio remoto)
- ❑ `git remote show origin` (permite ver la información de los repositorios remotos)
- ❑ `git remote rename origin nombre_nuevo` (renombrar un repositorio remoto)
- ❑ `git remote rm nombre_git` (desvincula un repositorio remoto)
- ❑ `git push -u origin master` (el primer push en el repositorio debe contener su nombre y branch)
- ❑ `git push` (los otros pushes no necesitan otras informaciones)

### → Actualizar el repositorio local según el repositorio remoto

- ❑ `git pull` (actualizar los archivos contra la branch actual)
- ❑ `git fetch` (obtener los cambios, pero no aplicarlos a la branch actual)

### → Clonar un repositorio remoto existente

- ❑ `git clone git@github.com:minombre/archivo-git.git`

## Branches

El master es la branch principal de Git.

El HEAD es un puntero especial que indica cuál es la branch actual. Por defecto, HEAD

apunta a la branch principal, la master.

- ❑ `git branch nuevaBranch_nombre` (crea una nueva branch)
- ❑ `git checkout nuevaBranch_nombre` (cambia a una branch existente) - En este caso, el principal puntero HEAD está apuntando a la branch llamada `nuevaBranch_nombre`.
- ❑ `git checkout -b nuevaBranch_nombre` (crea una nueva branch y apunta a ella)
- ❑ `git checkout master` (vuelve a la branch principal-master-)
- ❑ `git merge nuevaBranch_nombre` (resuelve la unión (*merge*) entre las branches) - Para realizar la unión (*merge*), debe estar en la branch que debe recibir los cambios.
- ❑ `git branch -d nuevaBranch_nombre` (apagando una branch)
- ❑ `git branch` (lista branches)
- ❑ `git branch -v` (lista branches con información de los últimos commits)
- ❑ `git branch --merged` (lista branches que ya se han unido (*merged*) con la master)
- ❑ `git branch --no-merged` (listar branches que no se han unido (*merged*) con la master)
- ❑ `git pull origin nombreBranch` (saca los archivos de una branch existente)
- ❑ `git push origin nuevaBranch_nombre` (crea una branch remota con el mismo nombre)
- ❑ `git merge --abort` o `git reset --merge` (cuando tenemos problemas con la unión (*merge*) y queremos deshacerla)
- ❑ `git reset HEAD` (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: `HEAD~2`)

→ Reescribiendo la historia

- ❑ `git commit --amend -m "Mi nuevo mensaje"` (cambia los mensajes del commit)

## Comandos de la terminal

→ **ctrl+l o clear**

- ☐ Limpiar la consola

→ **mkdir nombre\_de\_carpeta**

- ☐ Crear una carpeta

→ **cd**

- ☐ Entrar en la carpeta

→ **cd ..**

- ☐ Salir de la carpeta

→ **ls**

- ☐ Ver lo que hay dentro de la carpeta

→ **rm nombre**

- ☐ Borrar archivo

→ **rm -r nombre**

- ☐ Borrar directorio y todos los archivos que contiene

→ **rm -rf nombre**

- ☐ Borrar directorio y todos los archivos que contiene en forma forzada