**Comandos básicos en la terminal:**

* **pwd**: Nos muestra la ruta de carpetas en la que te encuentras ahora mismo.
* **mkdir**: Nos permite crear carpetas (por ejemplo, mkdir Carpeta-Importante).
* **touch**: Nos permite crear archivos (por ejemplo, touch archivo.txt).
* **rm**: Nos permite borrar un archivo o carpeta (por ejemplo, rm archivo.txt). Mucho cuidado con este comando, puedes borrar todo tu disco duro.
* **cat**: Ver el contenido de un archivo (por ejemplo, cat nombre-archivo.txt).
* **ls**: Nos permite cambiar ver los archivos de la carpeta donde estamos ahora mismo. Podemos usar uno o más argumentos para ver más información sobre estos archivos (los argumentos pueden ser -- + el nombre del argumento o - + una sola letra o shortcut por cada argumento).  
  - ls -a: Mostrar todos los archivos, incluso los ocultos.  
  - ls -l: Ver todos los archivos como una lista.
* **cd**: Nos permite navegar entre carpetas.  
  - cd /: Ir a la ruta principal:  
  - cd o cd ~: Ir a la ruta de tu usuario  
  - cd carpeta/subcarpeta: Navegar a una ruta dentro de la carpeta donde estamos ahora mismo.  
  - cd .. (cd + dos puntos): Regresar una carpeta hacia atrás.  
  - Si quieres referirte al directorio en el que te encuentras ahora mismo puedes usar cd . (cd + un punto).
* **history**: Ver los últimos comandos que ejecutamos y un número especial con el que podemos repetir su ejecución.
* **! + número**: Ejecutar algún comando con el número que nos muestra el comando history (por ejemplo, !72).
* **clear**: Para limpiar la terminal. También podemos usar los atajos de teclado Ctrl + L o Command + L.

**Comandos básicos Git:**

**git config --global user.email "**[**tu@email.com**](mailto:tu@email.com)**"**

**git config --global user.name "Tu Nombre"**

**git config --list:** ver configuraciones

**git init**: iniciar Git en un directorio.

**git status**: revisar el estado de Git.

**git add**: empezar a trackear archivos ***tracked.***

**git commit -m “*mensaje*”**: guardar los archivos en el repositorio local ***staged.***

**git commit -am “*mensaje*”**: trackear y guardar al mismo tiempo.

**rm =** Borrar, sin eliminar la historia.

**reset =** Volver en el tiempo, perdiendo la historia.

**git rm --cached:** mueve los archivos que le indiquemos al estado Untracked.  
 **git rm --force:** elimina los archivos de Git y del disco duro.

**git show**: nos muestra los cambios que han existido sobre un archivo.

**git log:** muestra los últimos commits con sus IDs.

**git log --graph --oneline --decorate:** arbolito

**git diff <id-commit-a><id-commit-b>:** muestra la diferencia entre dos commits.

**git checkout** **<ID>:** Vuelve en el tiempo a cierto commit.

**git reset** **<ID>:** restaura el proyecto a cierto commit.

**git reset --soft:** borramos todo el historial pero guardamos los cambios que tengamos en Staging, así podemos aplicar las últimas actualizaciones a un nuevo commit.

**git reset --hard:** borra TODO, absolutamente todo. Toda la información de los commits y del área de staging se borra del historial.

**git reset HEAD**: sacar archivos del área de Staging.

**Servidores remotos y GitHub:**

**git push**: Luego de hacer git add y git commit debemos ejecutar este comando para mandar los cambios al servidor remoto.

**git fetch**: Lo usamos para traer actualizaciones del servidor remoto y guardarlas en nuestro repositorio local.

**git merge**: Lo necesitamos para combinar los últimos cambios del servidor remoto y nuestro directorio de trabajo y viceversa.

**git pull**: Básicamente, git fetch y git merge al mismo tiempo.

**git remote add origin <URL>:** guardar el HTTPs o SSH del repositorio de GitHub con nombre origin, para poder interactuar con él.

**git remote -v:** revisar las urls.

**git pull origin master --allow-unrelated-histories:** Traer la versión del repositorio remoto y hacer merge para crear un commit con los archivos de ambas partes.

**git push origin master:** guardar los cambios de nuestro repositorio local en GitHub:

**git clone <url\_del\_servidor\_remoto>**: Nos permite descargar los archivos de un repositorio de GitHub + el historial de commits y merges.

**Branches:**

**git branch:** crear una rama.

**git checkout:** cambiar entre ramas.

**git branch -d:** borrar rama

Combinar dos ramas:

**git checkout <branch>:** para dirigirnos a una de las ramas (que recibirá a la otra).

**git merge <other\_branch>:** para combinar la rama en la que estamos con otra.

Actualizar una rama de GitHub (como gh-pages):

**git pull origin <branch>:** traemos master a una rama.

**git push origin <branch>:<remote-branch>:** subimos esa rama a la rama de GitHub.

**SSH**

**Primer paso: Generar tus llaves SSH**.

Recuerda que es muy buena idea proteger tu llave privada con una contraseña.

**ssh-keygen -t rsa -b 4096 -C "tu@email.com"**

**Segundo paso**:

Terminar de configurar nuestro sistema.

*# Encender el "servidor" de llaves SSH de tu computadora:*

**eval $(ssh-agent -s)**

*# Añadir tu llave SSH a este "servidor":*

**ssh-add <ruta-donde-guardaste-tu-llave-privada>**

**TAGS**

Crear un nuevo tag y asignarlo a un commit:

**git tag -a <nombre-del-tag> -m “notas del tag” <id-del-commit>**

Borrar un tag en el repositorio local:

**git tag -d nombre-del-tag**

Listar los tags de nuestro repositorio local:

**git tag o git show-ref --tags.**

Publicar un tag en el repositorio remoto:

**git push origin --tags.**

Borrar un tag del repositorio remoto:

**git tag -d <nombre-del-tag>**

**git push origin :refs/tags/<nombre-del-tag>**

**Trabajando con más de 1 repositorio remoto**

Cuando trabajas en un proyecto que existe en diferentes repositorios remotos (normalmente a causa de un fork) es muy probable que desees poder trabajar con ambos repositorios, para ésto puedes crear un remoto adicional desde consola.

**git remote add <nombre\_del\_remoto> <url\_del\_remoto>**

**git remote upstream <url>**

Al crear un remoto adicional podremos, hacer pull desde el nuevo origen (en caso de tener permisos podremos hacer fetch y push)

**git pull <remoto> <rama>**

**git pull upstream master**