Comandos Principales

* pwd nos muestra el path o ruta de la carpeta en donde nos encontramos ubicados
* cd me permite acceder (entrar) a una carpeta en un nivel o varios niveles
* cd .. me permite salir de una carpeta en un nivel o varios niveles **OJO los dos puntos deben ser separados por un espacio del comando cd**
* ls me muestra los archivos que contiene una carpeta, puede ser la ubicación actual o una ruta especifica, no muestra los archivos ocultos
* ls -a me muestra los archivos que contiene una carpeta, puede ser la ubicación actual o una ruta especifica, incluyendo los archivos ocultos
* ls -l me lista los archivos que contiene una carpeta con sus atributos, puede ser la ubicación actual o una ruta especifica, no muestra los archivos ocultos
* ls -la me lista los archivos que contiene una carpeta con sus atributos, puede ser la ubicación actual o una ruta especifica, incluyendo los archivos ocultos
* clear limpiar la consola o terminal, o un shorcut crtl + L
* mkdir <nombre carpeta> nos permite crear una carpeta
* touch <nombre del archivo> nos permite crear un archivo
* cat <nombre del archivo> me permite visualizar el contenido del un archivo y lo muestra en el terminal
* history nos muestra un historial de los comandos que hemos utilizado
* rm <nombre del archivo> me permite borrar un archivo

**OJO** en Windows el terminal no es case sensitive (Sensible las mayusculas), con Linux,y UNIX si son case sensitive

comandos de git aprendidos en esta clase:

* **git init:** lo usamos para determinar la carpeta en la que vamos a trabajar.
* **git status:** lo usamos para saber si tenemos un archivo añadido o borrado en nuestro proyecto, para saber en la rama en la que estamos y si tenemos commits.
* **git add:** es para añadir un archivo a nuestra rama seguidamente ponemos entre comillas el nombre de nuestro archivo o poner un punto para añadir todos los archios de nuestra carpeta.
* **git rm:** lo usamos para borrar un archivo que hayamos añadido, para eliminarlo por completo de nuestra rama usamos**git rm --cached**.
* **git commit:** se usa para añadir un commit a nuestra rama, también podemos ponerle un **-m** seguidamente ponemos entre comillas nuestro ensaje.
* **git config:** muestra configuraciones de git también podemos usar **–list** para mostrar la configuración por defecto de nuestro git y si añadimos --show-origin inhales nos muestra las configuraciones guardadas y su ubicación.
* **git config --global**[**user.name**](http://user.name/)**:** cambia de manera global el nombre del usuario, seguidamente ponemos entre comillas nuestro nombre.
* **git config --global user.email:** cambia de manera global el email del usuario, seguidamente ponemos entre comillas nuestro nombre.
* **git log:** se usa para ver la historia de nuestros archivos, los commits, el usuario que lo cambió, cuando se realizaron los cambios etc. seguidamente ponemos el nombre de nuestro archivo.

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.

* git init: Se crea staging en memoria ram y en el repositorio en HDD
* git add: trakea cambios en el staging
* commit: cambios en el repositorio
* checkout: atrae cambios de alguna rama
* git config --global user.email ["tu@email.com](mailto:%22tu@email.com)"
* git config --global [user.name](http://user.name/) “Tu Nombre”

Hola hice un paso a paso de clase

1. Creamos una carpeta llamada proyecto1
2. Le damos el comando git init
3. Le damos el comando ls -al y podemos ver las carpetas vacías y comprobar que efectivamente tenemos la de git
4. Abrimos visual studio code y guardamos un archivo de texto y lo guardamos en la carpeta,
5. Le damos git status y ahi vemos cuales son los archivos que aún no se han trackeado
6. Luego la damos git add para empezar a trackearlo
7. Lo sacamos dandole git rm —cached

\*\* Nota  
Cached - Significa que esta en memoria RAM - no en GIT

1. Le damos Commit y enviamos al repositorio
2. Configuramos quien somos nosotros para poder determinar quien hizo que
3. Le damos el siguiente comando Git config - para mostrar la configuración

\*\* Nota

Git Config - nos da los comandos principales de GIT

1. Le damos el comando GIT config - - list para poder ver la configuración por defecto de nuestro Git - esto para ver el nombre y correo del usuario
2. Le damos el comando GIT config - - list - - origin y ahi sabemos donde están guardadas las configuraciones - casos más avanzados
3. Le damos GIT consigo - - global eso lo usamos para cambiar todos los usuarios global
4. Para cambiar el nombre en la configuración y el email le damos el siguiente comando:
5. Cambiamos el archivo historia.txt - eso lo podemos hacer con el comando CODE
6. Usamos el comando CAT historia.txt - para ver el contenido del archivo - pero no puedo modificarlo
7. Hacemos los cambios y guardamos - ahora cuando hacemos git status el reconoce que se modifico el archivo
8. Hacemos un nuevo commit - PERO NO PASA NADA! SIEMPRE LOS CAMBIOS DEBEMOS AGREGARLOS CON EL BOTON DE ADD
9. Hacemos el comando ADD . -> El punto al final lo que hace es que agregamos todos los archivos de una vez
10. Ahora si podemos entrar a hacer el commit
11. Para ver toda la historia le damos git log historia.txt