**Comandos para GitBash:**

* pwd: te dice en donde estas parado
* cd “nombre de la carpeta”: te lleva a donde queres ir
* cd .. : para volver una carpeta hacia atras
* cd ../”nombre de carpeta” : para volver uno hacia atras y meterse en otra
* cd ../../nombre de carpeta : para volver dos hacia atras y meterse en otra (y asi sucesivamente)
* ls: te muestra los archivos o carpetas que hay dentro de la carpeta que vos estas
* ls -lh: te muestra los archivos de forma organizada.
* ls -a: te muestra los archivos ocultos.
* mkdir “nombre de la carpeta”: para crear una carpeta nueva en donde estoy situada. En ingles viene de make directory
* mv “nombre del archivo” ruta/de/donde/quiero/moverlo/segun/donde/estoy/parada: para mover un archivo o carpeta a otra carpeta
* mv “nombre del archivo” “nuevo nombre del archivo”: para cambiar el nombre de un archivo
* touch “nombre del archivo”: para crear un archivo
* rm “nombre del archivo”: para borrar un archivo. Se pueden borrar varios al mismo tiempo separando los nombres de cada archivo. Ej: rm “achivo1” “archivo2” “archivo3”
* rm -r “nombre de la carpeta”: para eliminar una carpeta entera, ya sea si tiene archivos dentro o no.
* rm -dr .git: para eliminar carpetas ocultas (creo ???)
* nano “nombre del archivo”: para editar texto de un archivo

**Para Git:**

* git --version: es para certificar que tengo descargado git; me muestra la version que tengo descargada
* **git init**: para iniciar un repositorio local en la carpeta en la que estoy
* git config user.name “mi nombre de usuario”: para que git sepa que el repositorio que acabo de crear es mio
* git config --global user.name “mi nombre de usario”: para que git sepa que todos los repositorios que voy a hacer en esta pc son míos, y no tener que aclararlo cada vez que creo un repositorio nuevo
* git config user.email “mi mail”: ademas de mi usuario, cuando hago un repositorio nuevo tengo que aclararle el mail con el que hice mi usuario
* git config --global user.email “mi mail”: para que ya sepa que son todos los repositorios que cree, con el mismo mail, y no tener que aclararlo mil veces.
* git add “nombre del archivo”: para agregar al repositorio, uno de los archivos que estan dentro de la carpeta en la que estoy parada y donde cree el repositorio.
* **git add .** : para agregar al repositorio todos los archivos y cambios de la carpeta en la que estoy parada y donde cree mi repositorio.
* **git commit -m “mensaje que quiero guardar en este commit”**: sirve para hacer como una cajita que va a contener y guardar todos los archivos que estaban en ese momento en el repositorio. Si luego de hacer este commit agrego uno mas archivos nuevos al repositorio, o modifico un archivo (con git add) tengo que ejecutar este comando nuevamente, y se hara un commit nuevo, mas actual en el tiempo, con los cambios que realicé.
* git log: me muestra todos los commits que tengo en mi repositorio, junto con la fecha en que se creó, el autor del commit y el mensaje con el que lo guarde.
* git status. Me va a mostrar si hay algun archivo que esté sin seguimiento.

Previo a hacer un commit nuevo, debería utilizar git status para ver si algún archivo o cambio aún no tiene seguimiento. En ese caso utilizo primero git add, y luego git commit.

* git checkout -- “nombreArchivoQueBorre”: me sirve para recuperar un archivo que eliminé de mi carpeta master/main (la carpeta física que veo en mi ordenador), porque el archivo sigue estando en la carpeta git entonces realiza una copia y la devuelve a la carpeta main. (esto es para recuperar archivos del commit anterior. Si quiero recuperar uno de un commit mas viejo, es otro comando el que utilizo)
* git rm -- cached “nombreDeArchivo”: para decirle a git que quiero dejar de hacerle el seguimiento a determinado archivo. El archivo seguirá en la carpeta main pero si le hago modificaciones, esto ya no se guardará en la carpeta git, entonces podré acceder a este archivo en el commit anterior a dejar de hacerle seguimiento, pero no tendrá los cambios que le realicé luego de ejecutar la accion rm -- cached
* **git branch -m main:** para que el repositorio sea main en vez de master
* git branch dev: para crear una branch paralela a la main (CREO)
* git branch -a: me muestra lista de las branch que ya estan sincronizadas con el repositorio de la nube (CREO)
* **git remote add origin “el url del repositorio remoto”**: para sincronizar nuestro repositorio local al repositorio remoto
* **rm -rf .git** : para eliminar el repositorio local. Debo estar ubicada en la carpeta a la cual deseo eliminarle el repo local. (la f es de forzar)
* git remote -v: para verificar que logramos la sincronizacion de ambos repositorios
* git push origin main: Para enviar los archivos al repositorio remoto correr el comando
* git pull origin main: Para poder actualizar los archivos de nuestro repositorio local con respecto a los que están en el repositorio remoto, debemos correr este comando.

**IMPORTANTE**: Siempre hacer primero un pull, y luego un push.

Para descargar por primera vez un repositorio remoto a nuestra máquina, tendremos que clonarlo.

● Abrir una terminal en la ubicación en donde queramos clonar el proyecto.

● Copiar la URL del repositorio que queremos clonar.

● Escribir el comando git clone .

● Pegar la URL después de la palabra clone (dejando un espacio de por medio) y presionar enter.