**GIT DESDE CERO CON EDTEAM**

**COMANDOS GIT:**

**Comprobamos que GIT este completamente instalado en el equipo:**

**git**

**Verificamos la version de GIT:**

git **--version**

**CONFIG**

**Definimos usuario global que escribira en la bitacora del equipo:**

git **config --global** user.name "Mi nombre"

**Definimos email de usuario global que escribira en la bitacora del equipo:**

git **config --global** user.email "Mi correo"

**Listamos toda la configuracion seteada de GIT:**

git config **--list** o git config **-l**

**Inicializamos un Proyecto en GIT:**

git **init**

**LOS ESTADOS DE GIT**

1. **ESPACIO DE TRABAJO**

Todo lo que podemos ver, osea sin la carpeta **.git**

1. **AREA DE PREPARACION**

Archivos cargados por el comando git **add**

1. **REPOSITORIO *(Carpeta .GIT)***

Archivos en nuestra bitacora por el comando git **commit**

**Ver el estado de nuestro proyecto:**

git **status**

**ADD**

**Para mover los archivos del estado 1.ESPACIO DEL TRABAJO al 2.AREA DE PREPARACION *(Tracker archivo):***

git **add nombre\_de\_archivo\_con\_ext**

**Para mover todos los archivos de la carpeta donde nos encontramos actualmente del estado 1.ESPACIO DEL TRABAJO al 2.AREA DE PREPARACION *(Tracker archivo):***

git add **.** *(Comodin de Unix, todos los archivos que estan en la carpeta en la que estoy)*

**COMMIT**

**Agregar archivos a mi repositorio:**

git **commit**

* Se abre el editor de texto en la terminal, debemos agregar nuestro msj del commit en nuestra bitacora. Se presiona **i** para **INSERT,** si el commit o msj esta vacio se aborta la operación, se presiona la tecla **ESC** para salir del editor y guardar el commit se presiona **:wq**  y presiono **ENTER.**

**Agregar archivos a mi repositorio con un msj directo sin abrir el editor de texto en la terminal:**

git commit **-m “aquí\_va\_nuestro\_msj”**

**Agregar los archivos modificados, no nuevos, directamente del estado 1.ESPACIO DEL TRABAJO al 2.AREA DE PREPARACION y al 3.REPOSITORIO *(commit directo a repositorio solo aplica para archivos modificados):***

git commit **-am** “aquí\_va\_nuestro\_msj”

**LOG**

**Ver todos los commit agregados a mi repositorio:**

git **log**

**Ver que fue lo que paso en un commit:**

git log **--raw**

* **M:** modifcado, **A:** Añadido.

**Ver todos los commit agregados a mi repositorio de forma condensada en una linea por commit, nos muestra el hash resumido y el comentario:**

git log **--online**

* --online **-n numero\_#**, nos muestra la cantidad del numero que le colocamos para los ultimos commit.
* --online **--all,** nos muestra todos los commit de todas las ramas que tengamos en nuestro proyecto.
* --online --all **--graph,** nos muestra todos los commit de todas las ramas que tengamos en nuestro proyecto de tipo arbol **\*.**
* --online --all--graph **--decorate,** nos muestra de una forma grafica todos los commit de todas las ramas que tengamos en nuestro proyecto de tipo arbol **\*|/.**

**Dar formato a git log, esto depende de los comodines que queramos usar de la documentacion oficial de git *(Aplicar filtros al log)*:**

git log **--pretty=format:“*El autor del commit* %h *fue* %an”**

* La frase: *El autor del commit* ***comodin1*** *fue* ***comodin2,* %h** nos va mostrar el **hash** resumido y **%an** el autor del commit. La documentacion oficial esta aquí: https://git-scm.com/docs

**CHECKOUT**

**Para ver el historial de cambios y viajar en el tiempo de nuestros commit, esto es abrir el proyecto en el estado de un commit hecho anteriormente, debemos posicionar la cabecera o HEAD en ese commit a travez de su hash:**

git **checkout hash\_id\_del\_commit**

* checkout **master**, si nos encontramos en una rama distinta a **master** este comando nos lleva al master de nuestro proyeto, al ultimo commit que hicimos.

**Deshacer los cambios hechos en los archivos de mi proyecto:**

git **checkout nombre\_de\_archivo\_con\_ext**

**Para deshacer todos los cambios hechos en los archivos de las carpetas que han sido modificadas donde nos encontramos actualmente:**

git checkout **.** *(Comodin de Unix, todos los archivos que estan en la carpeta en la que estoy)*

**Para forzar a deshacer todos los cambios hechos en los archivos de las carpetas que han sido modificadas donde nos encontramos actualmente:**

git checkout **-f**

**Para crear una nueva rama en nuestro proyecto nos situamos en la rama master en el ultimo commit de donde queremos que se desprenda esta nueva rama:**

git checkout **-b nombre\_nueva\_rama**

**Ver todas las ramas del proyecto:**

git checkout **branch**

**BRANCH**

**Manejador de ramas en git, D de delete:**

git **branch -D nombre\_rama\_eliminar**

**RESTORE**

**Quitar los archivos que hayamos cargado de nuestro 1.ESPACIO DE TRABAJO al 2. AREA DE PREPARACION:**

git **restore --staged nombre\_de\_archivo\_con\_ext**

**DIFF**

**Nos permite ver la diferencia entre lo que estamos haciendo con lo que ya tenemos almacenado en nuestro repositorio:**

git **diff nombre\_de\_archivo\_con\_ext**

**Ver estadistica de cuantas lineas de codigo hemos cambiado en un archivo**

git diff **--stat** nombre\_de\_archivo\_con\_ext

**Ver numeros de cuantas lineas de codigo hemos cambiado en un archivo**

git diff **--numstat** nombre\_de\_archivo\_con\_ext

**SWITCH**

**Para cambiar de ramas en nuestro proyecto:**

git **switch** **nombre\_rama**

**MERGE**

**Nos permite fusionar ramas, es importante y muy cuidadoso cuando se mezclen 2 ramas que muy probablemente tocaron archivos en comun. Por eso cuando las ramas a fusionar no tienen este conflicto, es decir que los archivos de las ramas se trabajaron de forma independiente podemos hacer el comando merge de forma muy simple, posicionando la cabecera o HEAD en al rama donde queremos fusionar las demas ramas:**

git **merge** **nombre\_rama\_fusionar**

* Cuando una las ramas que se van a fusionar fueron modificadas en el mismo codigo de los mismos archivos de ambas ramas, debemos solucionar el conflicto de forma manual en el editor de codigo despues guardamos los archivos,al hacer git status veremos que hay un msj que ambos modificamos el archivo, debemos hacer add y commit y las ramas quedaran fusionadas.
* git merge **–abort,** aborta el merge.

**CLONE**

**Nos permite clonar archivos de github en nuestro ordenador, lo podemos hacer mediante HTTPS o SSH *(se debe tener configurado la key)*:**

git **clone https\_ssh\_proyecto\_clonar**

**PUSH**

**Como hemos descargado de nuestro repositorio web el proyecto, una vez despues de haberle realizado las modificaciones en el pc el proyecto nos ha creado un ORIGIN *(rama remota),* podemos subir archivos a GITHUB desde nuestro ordenador, lo podemos hacer mediante HTTPS o SSH *(se debe tener configurado la key)* mediante el ORIGIN creado en el proyecto:**

git **push origin rama\_proyecto\_subir**

**Para subir un proyecto local a GITHUB debemos crear un repositorio nuevo en la pagina, y debemos crear un origen remoto desde nuestro ordenador *(Ver REMOTE)* y subir nuestra rama master *(Todo el proyecto):***

git push **-u master**

**Para eliminar una rama del proyecto en GITHUB:**

git push **--delete origin nombre\_rama\_eliminar**

**FETCH**

**Para traernos archivos del proyecto que estan en la nube pero no en el pc, esto se entiende con git en el ordenador pero aun no tenemos ese codigo actualizado en el proyecto de nuestro pc:**

git **fetch origin**

**PULL**

**Para traernos archivos a nuestro proyecto de la rama remota del proyecto que esta en la nube que ya hemos traido anteriormente con FETCH:**

git **pull origin nombre\_rama**

**REMOTE**

**Para saber a donde apuntamos en origin en el remoto:**

git **remote -v**

**Para subir un proyecto local a GITHUB debemos crear un repositorio nuevo en la pagina, debemos crear un origen remoto y despues hacer push al origen remoto de nuestra master local:**

git remote **add origin direccion\_del\_proyecto\_HTTPS\_SSH**

**AGREGAR LLAVE GITHUB EN MI PC**

**En la consola escribo el siguiente comando para poder crear mi key en mi pc, despues en mi usuario de GITHUB en configuraciones agrego esa key que se encuentra en el archivo id\_rsa.pub**

**ssh-keygen -t rsa**

Enter passphrase (empty for no passphrase): clave Mac Air

* **code id\_rsa.pub,** para poder ver la key en visual studio code. Para poder abrir code desde el terminal se debe configurar en la terminal y el gestor de arhivos el permiso de la terminal a code.