`Un proyecto en Git se llama REPOSITORIO

COMANDOS DE GIT

Git –versión = significa que quieres saber la versión de Git el doble guion medio junto”- - “indica que Git de toda la palabra completa

Git config –global -e = sirve para saber el nombre del usuario y su correo

Esc :q! = sirve para salir de modo de visualización de quien es el usuario y su correo

q > para salir

¡ > salir inmediatamente

Git init > inicializa el repositorio

Git status > da información sobre los commit y la rama sobre la cual estamos trabajando

Git add .> sirve para dar en uso algún archivo

Git reset > para remover un archivo

Ctrl + l = limpia la pantalla

Git checkout -- .= reconstruye el proyecto como estaba antes de las modificaciones

Git commit = realiza una foto del repositorio

Git Branch = indica la rama sobre la cual estamos trabajando

Git branch -m master main = se cambia el nombre de master a main

Git config --global init.defaultBranch main = de manera global la rama se llama main

Git log = sirve para ver los commit realizados

Git add css/ = agrega todas las capetas y subcarpetas

Git config –global alias.s status short = forma de crear un alias para Git status

Git log –oneline = da una descripción corta de los commit hechos

Git diff –staged = para ver los cambios en el satges

Git diff= sirve para ver los cambios sin que estén en el staged primero guardar los cambios

Git commit -am = añade al staged y al mismo tiempo se hace el commit

forma de actualizar el mensaje de un commit

1.- Git commit --amend -m "mensaje correcto" == para el ultimo commit realizado

El comando Git reset --soft HEAD^ se usa para agregar cambios al último commit

Git reset –soft “numero de anterior commit” = sirve para modificar el commit HEAD”

Git reset -mixed “número del commit al que deseamos regresar =” saca todo del stage y los cambios quedan listos para añadir

Git reflog = muestra todo lo que ha sucedido en orden cronológico

Git mv destruir-mundo.md salvar-mundo.md = cambia el nombre destruir-mundo.md por salvar-mundo.md debe ser dentro del mismo repositorio

Git reset –hard = de esta forma el comando funciona parecido al checkout

Git rm “nombre del archivo o carpeta” = Comando para borrar un archivo

Git Branch “nombre de la rama” = forma de crear una rama

Git checkout “nombre de la rama” = de esta forma nos podemos mover a la X

Git merge “nombre de la rama” = los cambios de la rama X serán agregados a la rama máster

Git branch -d “nombre de la rama” = comando para borrar una rama cuando ya se necesite

Git branch -d rama-villanos -f = comando para forzar el borrado de los cambios que Git comenta

Git checkout -b “nombre de rama” = comando para crear una rama y movernos a ella automáticamente

Git tag -a versión -m “nombre del tag” = sirve para crear un tag

Git tag = sirve para observar los tags

Git tag -d = sirve para borrar un tag

Git push –tags = comando para subir los tags a GitHub en la nube

Git tag -a “versión” “número del commit en donde se realizará tag para commits ya hechos” -m "mensaje"

STASH = es un lugar donde se puede almacenar la información de manera temporal y posteriormente recuperarla

Git stash pop = comando para retomar los cambios que se estaban realizando dentro del repositorio y borra el stash

Git stash clear = comando para borrar la información almacenada dentro del stash

Git stash show 'stash@{1}' = comando para mostrar más información acerca del stash

Git stash save "mensaje" = comando para guardar en le stash con un mensaje para identificar que hace ese stash

Git rebase -i HEAD~3 = comando para modificar el nombre del commit o editar el commit

- Git reset HEAD^ para irnos al último commit y bajar del stage las modificaciones hechas y poder cambiarlas y separa commit

Git push = comando para subir los cambios al repositorio en GitHub

Git pull = comando para traer los datos que están en el origen por defecto

Git rebase = comando para corregir conflictos, aunque los cambios se realizaron desde GitHub

Git fech = comando para actualizar los cambios que se realizaron en el servidor (nube) en el repositorio local, pero el HEAD apunta al último commit hecho de manera local

hacer un Git pull para que el HEAD local apunte al último commit que se realizó en el servidor(nube)

Git remote -v = comando para ver las direcciones del push

Git checkout 4e7113d miembros.md = restaura el personaje borrado, el hash el del commit anterior además se agrega el nombre del archivo para que todos los demás archivos se queden igual (sin modificar)

Git push –set-upstream origin “Nombre de la rama” = comando para subir una rama al repositorio en GitHub

Git branch -a = comando para obtener ramas remotas que otra persona creo para poder ver su trabajo

Git remote prune origin = comando para limpiar las ramas eliminadas en el remote

Git push origin : “nombre de la rama” = comando para actualizar que se borro la rama y se subieron los cambios a GitHub

git commit -am "Fixes #3: borre a la capitán Marvel" = comando para cerrar un issue desde consola

cuando no hace match

```
git checkout -b master
```
luego

```
git push origin master
```
luego cambiar nombre de rama

para subir los commit de la rama secundaria

git push --set-upstream origin “Nombre de la rama”