**COMANDOS BASICOS GIT**

git init ( sirve para indicar la carpeta “previamente selecccionada” en la que se creara un repositorio en el disco duro o staggng)

git add “e.g. Archivo.txt” (sirve para agregar un archivo especifico al area de stagging)

git add . (sirve para agregar todos los archivos al area de stagging)

git commit -m “mensaje” (sirve para enviar todos los archivos del area de stagging al area del repositorio o “master”)

git show ( me muestra los cambios que han sido realizados a traves de las versiones)

git log ( me muestra las versiones que han sido guardadas en “Master”)

git checkout ( me permite devolver en el tiempo un archivo a una version anterior sin borrar la actual o simplemente toda la carpeta

para que me muestre la version anterior en un archivo en especifico debo indicar al final el nombre del archivo)

git diff (indicandole las versions que queremos, nos muestra las diferencias entre ellas)

git reset xxxxxxxxxxxxxxxxxxxxxx --hard (este devuelve en el tiempo el projecto y elimina todo lo realizado posteriormente a la version a la que estamos volviendo)

git reset xxxxxxxxxxxxxxxxxxxxxx – soft (este elimina los archivos del commit pero mantiene los archivos del area de stagging)

git reset xxxxxxxxxxxxxxxxxxxxxx head ( sirve para pasar los archivos de staging a unstaged en caso de que queramos editar algun archivo o lanzar en otra actualizacion

algun cambio que no era para este commit sino para un commit futuro)

git rm --cached: (Elimina los archivos del área de Staging y del próximo commit pero los mantiene en nuestro disco duro.)

git rm --force: (Elimina los archivos de Git y del disco duro. Git siempre guarda todo, por lo que podemos acceder al registro de la existencia de los archivos,

de modo que podremos recuperarlos si es necesario

(pero debemos usar comandos más avanzados).)

git clone url (sirve para clonar el proyecto de un repositorio ubicado en un servidor a mi repositorio y directorio de trabajo)

git push (sirve para enviar todos los datos actualizados en mi repositorio al repositorio del servidor.)

git fetch (importa un archivo que alguien mas cambio recientemente y no tienes en tu repositorio local, puedes usar git merge para combinar tus archivos,

en el directorio de trabajo con tu repositorio local para que quede unidos los cambios )

git merge (combina dos versiones)

git pull (a diferencia de fetch y merge al mismo tiempo, esta lo que hace es tomar los archivos,

actualizados del repositorio remoto y copiarlos tanto en mi repositorio local como en

mi directorio de trabajo, para asi tener una version actualizada del repositorio remoto.)

git commit -a (sirve para agregar directamente a master sin pasar por el staging)

git branch (sirve para ver las ramas creadas o sirve para crear una rama dependiendo de su uso, si usas el comando tal cual lo leiste,

solo te mostrara las ramas existentes, pero si lo usas de este modo e.g. (git branch "inserte aca nombre que le quiera proporcionar a la rama"

obviamente sin las comillas, crearas una rama nueva con el nombre que quieras.)

git commit -am (sirve para agregar al repositorio local los archivos que estas trackeando con un mensaje)

git remote add “origin” ( sirve para enviar el projecto en el que estamos trabajando a la base de datos o repositorio remoto)

git remote ( nos mostrara el repositorio remoto dispobible o existente)

git remote -v (nos indicara la direccion web del repositorio remoto)

git push “origin” “master”

git pull “origin” “master”

git pull “origin” “master” --allow-unrelated-histories ( sirve para forzar a combinar los archivos del repositoerio remoto con el repositorio local)

git config -l ( nos muestra las configuraciones de git)

gir config global --user.email "jdbshfjbhfdsbjfh@gmail.com"

git log --all

git log --all --graph --decorate --oneline" ( me muestra todos los commits realizados en el proyecto de manera que se pueden observar los mensajes y los merges)

alias arbolito=“git log --all --graph --decorate --oneline” (crear un alias, ese es el ejemplo, un alias ya sabes para que funciona)

git tag (me muestra las versionas realizadas es decir los tags

git tag -a v0.1 -m “Resultado de las primeras clases del curso” 5d10f9c (ejempplo de como se hacen los tag)

git show-ref --tags ( me muestra los tags con las versiones y a que commit estan relacionados.)

git push origin --tags ( enviar los tags)

git tag -d “NombreDeTag” (me permite eliminar un tag que le indique)

git push origin :refs/tags/dormido ( codigo para eliminar el tag de github)

git show-branch (nos muestra todas las ramas, y si le agregamos --all nos muestra mas datos)

gitk (abre en un software la historia del proyecto, de una manera ultravisual)

git remote add “upstream” (nos crea otrwa fuente nueva de donde podemos traer datos a nuestras rama amster)

git remote -v

git clone “link” (nos permite clonar un projecto publico)

cuando queremos que git ignore algun archivo, debemos crear un new file en el editor y guardarlo en el directorio

como “.gitignore” luego en el editos utilizaremos “\*” para indicarle la extension de los archivos que queramos que

ignore

git rebase ( sirve para el momento de realizar un hotfix y que no se vea en el log de ramas el error)

git branch -D “rama Name” (elimina una rama)

git stash (me devuelve el proyecto al estado anterior antes de ralizar algun cambio, manteniendo el cambio guardado temporalmente, es como presionar cntrl+z en Vs code)

git stash pop ( me devuelve al proyecto luego de hacer los cambios antes de aplicar el stash)

git stash list ( me muestra una lista de los stash)

git stash branch “branch name” (lo que hace es que si existe un stash, esto lo mueve a esa nueva rama, realizas un commit y queda gurdado el trabajo alli)

git stash drop ( sirve para borrar los stash realizados)

git clean (sirve para borrar archivos no deseados, o que no utilizas realmente en tu repositorio, agregando “–dry-run” muestra todo lo que va a borrar el comando

pero si agregamos -f borra todo el listado que te muestra el comando anterior sin preguntar

git cherry-pick (sirve para traer un commit de otro rama a la que te encuentres(ES MALA PRACTICA))

git reflog (aun habiendo eliminado los errores con un git reset --hard, este muestra el log de los errores cometidos, aquellos que no

se ven en un “git log”)

git grep (busca en el proyecto las palabras usadas, el nombre de un archivo o lo que sea que le mandes a buscar, y si agregas “-n” te dice inclusive

en que lineas se utiliza… si utilizamos “-c” nos indica las veces que utilizamos esas palabras)

git log -S “palabra a buscar” (busca la palabra que estas buscando en el historial o commits)

git blame “nombre del archivo” (sirve para ver los commits realizados por las diferentes personas en orden con todos los datos si agregamos “-c” se ve un poco mejor)

si utilizamos

git blame “nombre del archivo” “-L35,53” (nos muestra los cambios realizados por las personas en entre las lineas especificadas.)

comandos y recursos colaborativos en git y github

git shortlog ( me muestra el nombre de las personas colaborando y los commits realizados por cada una de ellas.)

git shortlog -sn (me muestra los nombres y la cantidad en numeros de los commits que hicieron)

git shortlog -sn --all (muestra inclusive los commits borrados)

git shortlog -sn -all --no-merges (lo mismo que el anterior pero omite contar los merges)

git congfig --global alias.(nombre del comando) “git shortlog -sn -all --no-merges”

git branch -r (te permite ver las ramas remotas en tu servidor)

git branch - a (te muestra las ramas locales y las remotas)

agrego el amend aca porque es un proceso por pasos:

para que sirve: cuaqndo realizas un commit pero en el commit olvidaste realizar una parte del trabajo o arreglar un detalle.

1ro. debes hacer tu cambio y verificar que colocaste el codigo bien y todo lo demas en tu editor de codigo como normalmente lo haces.

2do. vas a realizar un git add “aqui vas a indicar el archivo que tiene los cambios que no colocaste”.

3ro. luego de agregar los archivos a staging vas a realizar un git commit --amend (esto agrega el archivo al que le hiciste los cambios al commi anterior.)