para listar archivos ls

para listar archivos con detalles

para listar archivos + ocultos ls -al

para crear archivos touch .....

como ver el contenido de un archivo cat ....

para crear carpetas mkdir .....

como ver la historia completa de los history

comandos utilizados

si queremos repetir el comando !numerodecomandoarepetir

como eliminar archivos rm .....

para cambiar el email de git

git config --global user.email "<email>"

para ver las configuraciones que tenemos en nuestro git

git config

para verlo de mejor manera

git confit --list

para ver los commmits realizados

git log nombrearchivo

para ver los cambios que existen sobre un archivo

git show nombrearchivo

para comparar los commits que hayamos realizados

hacemos git lot y luego los ids lo colocamos en

git diff <version1> <version2>

como podemos volver a una version anterior con los codigos cuando usemos

git log

es con git reset <codigo> --<hard|soft>

hay 2 tipos

duro(--hard) todo vuelve al estado anterior y borra todo lo que nosotros hubieramos tenido antes de esa version

y

suave(--soft) volvemos a la version anterior pero lo que tengamos en stagin se queda ahi solo cambia en el directorio

como podriamos traer los cambios

git checkout <codigo> <nombrearchivo>

git checkout master <nombrearchivo>

PARA CREAR UNA NUEVA RAMA

git branch <nombre>

PARA VER SI SE CREO ESA RAMA PODEMOS VERLA COLOCANDO

git status

o

git show

COMO NOS MOVEMOS A ESA NUEVA RAMA

git checkout <nombre>

PARA LISTAR LAS RAMAS

git branch

COMO HACEMOS UN MERGE

posicionamos el HEAD en master (o donde queramos hacer la fusion)

y con la rama que seleccionemos se fusionara en master

creara un nuevo commit de master y de cabecera

git merge cabecera

COMO CREAMOS UNA LLAVE SSH EN GIT (llave publica y privada)

Generar una nueva llave SSH:

ssh-keygen -t rsa -b 4096 -C "youremail@example.com"

Comprobar proceso y agregarlo (Windows)

eval $(ssh-agent -s) (para saber si el servidor de ssh esta corriendo)

ssh-add ~/.ssh/id\_rsa (para agregar la llave a nuestro sistema a ese servidor que evaluamos) se agrega la llave sin pub porque pub es la publica la que se va a compartir

~(hace referencia al home de nuestro equipo -> /c/Users/Rogua)

de donde se trave poniendo git show o git log salimos apretando qq

------CUANDO EMPEZAMOS A SUBIR EL REPO A GITHUB-----

COMO AGREGAMOS UN ORIGEN REMOTO DE NUESTROS ARCHIVOS (debemos estar dentro del repositorio local que queramos añadir el repo de github online)

git remote set-url origin url-ssh-del-repositorio-en-github

COMO LO PODEMOS VER

git remote

para mas detalles

git remote -v

COMO ENVIAMOS LOS ARCHIVOS DE NUESTRA CARPETA A GITHUB (deberiamos estar dentro de nuestro proyecto)

git push origin master (o puede ser main ya que github se actualizo)

COMO MIGRAMOS DE master a MAIN

git branch -m main

git push origin main

CUANDO QUERAMOS SUBIR LOS ARCHIVOS

(como creamos un readme ese es un commit remoto y todo el proyecto que nosotros

hallamos trabajado antes en la maquina local sera otros commits

asi que tendremos que hacer pull al repositorio remoto)

pero git pull origin main (no servira porque son diferentes historias de commits)

asi que se aumenta

git pull origin main --allow-unrelated-histories

esto deberia traer el README a nuestro repo local

y ahora si lo que hacemos

git push origin main

UNA MANERA DE VER EL HISTORIAL QUE HALLAMOS TENIDO EN GIT ES CON

git log

PARA VER ABSOLUTAMENTE TODO

git log --all

PARA VER DE UNA MANERA DECORADA Y TRATAR DE ENTENDER LOS BRANCHS QUE HAYAMOS TENIDO

git log --all --graph --decorate --oneline

COMO CREAMOS UN ALIAS DE UN CODIGO SUPER LARGO

alias arbolito="git log --all --graph --decorate --oneline"

COMO TRATAMOS DE CREAR VERSIONES DE NUESTRO PROYECTO

OSEA TRATANDO DE SEPARAR VERSIONES DE NUESTRO PROYECTO CONFORME A LOS COMMITS

QUE HAYAMOS REALIZADO

git tag -a v0.1 -m "<descripciondeltag>" <codigo ese larguisimo que vemos al hacer gitlog>

COMO SABEMOS QUE TAGS TENEMOS

git tag

o

git show-ref --tags (el codigo que nos mostrara sera diferente al que pusimos para crearlo ya que solo es una referencia)

LOS TAGS SON UTILES EN GITHUB PARA QUE OTROS VEAN QUE VERSION TENEMOS

como lo enviamos?

git push origin --tags

COMO ELIMINARIAMOS UN TAG

git tag -d <nombredeltag>

(volvemos a subir pero) git push origin :refs/tags/<nombredeltag>

COMO PODEMOS VER LAS RAMAS QUE TENEMOS A DETALLE

git show-branch --all

COMO ENVIAMOS DIFERENTES RAMAS AL SERVIDOR (de nuestro directorio a github servidor)

como tenemos una rama llamada cabecera

git checkout cabecera

git push origin cabecera