GIT COMMANDS

-VERIFICAR VERSION ---> $ git -- version

-Ir a Folder del proyecto ---> $ cd path/

-Verificar dónde

estoy en el path ---> $ pwd

-Mirar qué Folders hay en

el Directorio ---> $ ls

-Crear repositorio local

del proyecto ---> $ git init

// Se crea la carpeta oculta .git

// Se crea la rama MASTER

-Visualizar cambios y

verificar estado de archivos ---> $ git status

// Indica la rama

// Visualiza el último commit

//Visualiza los archivos no-trackeados

-Iniciar inclusión

de archivos ---> $ git add <file's\_name>

---> $ gir add .

//El punto asume la inclusión de TODOS los archivos con modificaciones del Folder

-Para dejar de monitorear

un archivo ---> $ git rm --cached <file's\_name>

-Crear checkpoint

historico del archivo ---> $ git commmit -m "..........."

//Entre comillas va el comentario descriptivo del checkpoint

-Configurar usuario de

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

$ git config --global user.name "....."

-Visualizar commits ---> $ git log

...>> parametros:

Visualiza commits ...>> --oneline

Visualiza diferencias ...>> -p

Salir de editor log ...>> :q

Obtener ayuda ...>> --help

-Ignorar ciertos archivos

dentro del Folder ---> .gitignore

// Este sera un Folder dentro del proyecto; allí incluimos manualmente

los Folders y/o archivos que deseamos ignorar

TRABAJANDO EN EQUIPO - REPOSITORIO REMOTO

-Se crea Folder repositorio remoto

-Se inicializar git ---> $ git init --bare

// Parámetro "bare" significa que allí solo se guardaran los cambios. No tendrá el "working tree"

-Se crea servidor ---> $ git remote add <server\_name> conexión

//La conexión puede ser path o URL

-Para verificar le path ---> $ git remote -v

// Aquí se encontraran el path para FETCH y para PUSH

-Anclando al proyecto

por parte del otro colaborador ---> $ git clone <path> <name>

// El colaborador trae a su área local el proyecto y lo puede nombrar a su gusto

// clone solo crea la conexión, más todavía no retrotrae el contenido

-Enviando proyecto a

repo remoto ---> $ git push <repo> <branch>

-Trayendo proyecto o

actualizaciones ---> $ git pull <repo> <branch>

-REPOSITORIO EN LA NUBE-GITHUB

1. Crear el repo en Github ---> Crear repo

2. Conectar repo al local ---> $ git remote add origin <url>

---> $ git branch -M main

---> $ git push -u origin main

// Este set de comandos es para cuando YA tenemos el repo en el local

// Si no lo tenemos aun y lo estamos creando DESDE la nube, el set de comandos cambia

3. Cada colaborador

toma una rama de desarrollo

3.1. Creando rama ---> $ git branch <branch\_name>

3.2. Yendo a rama ---> $ git checkout <branch\_name>

3.3. Creando y yendo

al mismo timpo ---> $ git checkout -b <branch\_name>

// Todos los commit que se hagan en cada rama pertenecen a cada rama

4. Fusionando ramas entre sí

(Posicionado en main) ---> $ git merge <branch\_name>

5. Fusionando ramas, conservando

el commit de la rama

(Posicionado en main) ---> $ git rebase <branch\_name>

6. Revertiendo cambios:

6.1. Revertir sin haber

hañadido al stage area ---> $ git revert <file\_name>

6.2. Revertir después de

haber añadido ---> $ git revert --staged <file\_name>

6.3. Revertir después de

haber commiteado ---> $ git revert (#hashHex del commit)

7. Guardar cambios temporales

sin commitear ---> $ git stash

// Esto coloca al arhcivo en WIP "Working In Progress"

7.2. Para sacar al archivo

de stach

a. Trae al archivo, pero no

lo borra ---> $ git stash apply (#index)

b. Para borrarlo de stash ---> $ git stash drop

c. Para traerlo (el último)

y borrarlo ---> $ git stash pop

8. Para navegar a versiones

anteriores del código ---> $ git checkout #hash

9. Observando diferencias entre

commits ---> $ git diff #hashA..#hashB

10. Generando releases ---> $ git tag -a tag\_name -m "comment"