cd Documents/Pagos (El "cd" es para ingresar a la ruta especificada)

cd .. (Se regresa una carpeta desde la carpeta actual)

cd ../.. (Se regresa dos carpetas desde la carpeta actual)

mkdir (Se crea una carpeta en la ruta actual)

clear (Limpia el cmd)

dir (Nos dice los archivos que se encuentran en la ruta actual)

ls (Nos dice los archivos que se encuentran en la ruta actual)

git --version (Nos dice la versión de git)

git init (Crea un repositorio en la carpeta actual)

git status (Nos indica el estado de la carpeta y de los archivos en la dirección actual)

git add <file> (Se agrega un archivo para que git lo monitoree)

git add . (Se agregan todos los archivos de la carpeta)

git rm --cached <file> (Git deja de monitorear el archivo indicado)

git commit -m "<message>" (Se crea el punto de guardado y se da una breve explicación de los cambios realizados)

hash es una identificación del commit

git config --global user.name "<name>" (Se agrega el nombre a la configuración global de git)

git config --global user.email "<email>" (Se agrega el email a la configuración global de git)

git config --local user.name "<name>" (Se agrega el nombre a la configuración local de la rama de git)

git config --local user.email "<email>" (Se agrega el email a la configuración local de la rama de git)

git log (Nos muestra información)

/\*

commit 94ncoin902308jdn0923dmnvr809n (HEAD -> master)

------------hash------------- ----branch----

HEAD = Estado actual de nuestro código, es decir, donde nos colocó Git

\*/

git log --oneline (Nos muestra información de los commits en una sola linea)

git log -p (Nos muestra TODA la información con detalles de los commits)

git log --graph (Nos muestra el historial de una manera gráfica)

:q , :x (Nos sirve para salir de un editor)

.gitignore (Se crea un archivo llamado ".gitignore" y se escriben los archivos que no se quieren monitorear. Posteriormente se agreda

el .gitignore al repositorio)

git init --bare (Crea un servidor en la ruta actual donde se guardarán las modificaciones de los archivos. Los archivos aqui guardados no son los archivos

que se editan, solo se guardan las copias)

git remote (Nos muestra los servidores disponibles)

git remote add <nombre> <direccion> (Este comando se ejecuta en la dirección de la carpeta la cual se quiere tener un servidor,

y la dirección es de la carpeta donde estará el servidor que se creó con el --bare)

git remote raname <nombreactual> <nombrenuevo> (Para renombrar un servidor existente)

git clone <direccion> (Crea una copia de la carpeta especificada en la dirección, en la ruta actual)

/\*

direccion = /c/User/Alura

= (Si se quiere crear una carpeta llamada proyecto) /c/User/Alura/ proyecto

\*/

git push <direccionreceptor> <direccionemisor> (Se utliza para enviar los archivos al servidor previamente creado)

git pull <nombredelservidor> <ramadelservidor> (Se utiliza para recibir los archivos del servidor)

git branch (Nos muestra la rama en la que estamos)

git branch <nombre> (Crea una nueva rama)

git checkout <nombre> (Cambia a la otra rama)

git checkout -b <nombre> (Crea una nueva rama y se localiza en esa nueva rama)

git merge <rama> (Junta la rama <rama> con la rama master)

git rebase <rama> (Mueve los commits de la rama actual a la rama indicada colocando el commit despues de todos los demas commits)

git restore (Deshacemos una modificación que aún no fue agregada al index o stage, o sea, antes de hacer git add)

git restore --staged (Después de agregar con git add, para deshacer una modificación, necesitamos sacarlo de este estado, con git restore --staged)

git revert (Si ya hicimos el commit, el comando git revert puede salvarnos)

git stash (Guarda las modificaciones para despues)

git remote add origin https://github.com/LRPG2410/Certificados-de-Licencia.git

$ git push origin master