

~$ clear

Permite limpiar pantalla

~$ git config –global/local user.name “Jhon Doe”

Configura el nombre del usuario de tus proyectos git y github

~$ git config –global/local user.mail “test@correo.com”

Configura el correo electrónico de tus proyectos git y github

~$ git config user.name/user.mail

Muestra el nombre de usuario o correo electrónico del tus proyectos git y github

~$ git init

Permite crear o inicializar un repositorio local de git. Creará una carpeta .git que está oculta

~$ git config – global init.defaultBranch main

Cambia el nombre de la rama principal por “main”

Directorio de trabajo/Working directory

Es la carpeta del proyecto donde están los archivos y el directorio .git. Todo lo que se haga en esta área se entiende que está en el estado de **Modificada/Modified**

Area de preparación/Staging area

Son los archivos y cambios que serán incluidos en el próximo commit. Todos los archivos que estén en esta área pasan al estado de **Preparada/Stage**



Repositorio / Repository

Es la parte más importante y en ella están todos los archivos con su histórico de cambios. Esta área es la que se clona. Todos los archivos que estén en esta área pasan al estado de **Confirmada/Commited**



~$ git status

Verifica el estado de nuestro repositorio. En pocas palabra muestra los archivos que están incluidos en la staging área (texto verde), o aquellos con cambios que están en el Working Directory (texto rojo) no se está llevando registro de los cambios (tracking)

~$ git add nombre\_archivo.txt

Ingresa el archivo a el área de preparación o staging area para ser incluido en el próximo commit. Puede usar espacio para incluir varios archivos, ejemplo; archivo1.txt archivo2.txt

~$ git add .

Ingresa todos los archivos sin registro de cambios a el área de preparación o staging area para ser incluidos en el próximo commit.

~$ git rm –cached nombre\_archivo.txt

Retira el archivo del área de preparación y no será tenido en cuenta para el próximo commit.

~$ git log

Muestra todos los commit del respositorio.



~$ git log --oneline

Muestra resultado de los commit en una sola línea.



~$ git log --p

Muestra los commit con las diferencias en los cambios de los archivos.



~$ git commit -m “aquí pones el mensaje o commit”

Envía los cambios al repositorio para que registre los cambios y los archivos del proyecto.



~$ git commit

Realiza un commmit usando el editor de texto configurado con git. Debes escribir el comentario en la ventana que se abre, guardar y cerrar.



~$ git config –global core.editor “code --wait”

Cambia el editor de código que está por defecto para la gestión del repositorio. En este caso code es abreviatura de VS Code, se pueden asociar otros editores (consultar la documentación de git)



~$ git commit --amend

Permite cambiar el mensaje del último commit. Hay que tratar de realizarlo solo en repositorios locales porque en los compartidos puede afectar a nuestro equipo de desarrollo.

~$ git reset --soft HEAD~1

Permite eliminar el commit que se especifica en ~1 (en este caso el último commit). Este cambio logra retroceder el estado del archivo a el working directory o primer área donde podemos hacer cambios.



branch

Las ramas nos permiten desarrollar de forma independiente sin afectar el desarrollo en vivo.



~$ git branch nombre-rama

Permite crear una rama a partir de la rama en la que está ubicado.



~$ git branch

Permite ver las ramas creadas en el repositorio.

~$ git checkout nombre-rama

Permite cambiar de rama



~$ git checkout -b nombre-rama

Permite crear y cambiar de rama inmediatamente.

~$ git branch -m nuevo-nombre-rama

Permite modificar el nombre de la rama en la que estás situado.

~$ git branch -m nombre-rama nuevo-nombre-rama

Permite modificar el nombre de una rama sin estar situado en ella.



~$ git branch -d nombre-rama

Permite eliminar ramas locales. No debes estar ubicado en ella.



~$ git merge nombre-rama

Permite combinar cambios realizados en otra rama hacia la rama en la que te encuentras situado (Normalmente de donde se sacó la nueva rama).



~$ git clone https://github.com/williamcortes10/mi-primer-repositorio-github.git

Permite clonar un repositorio remoto en mi maquina local.



~$ git remote

Muestra el nombre que se le asignó al repositorio remoto.



~$ git remote add origin url-repositorio-github

Permite crear un repositorio remoto en algún servidor destino (github, gitlab, bicbucket)



~$ git remote -v

Muestra a detalle las operaciones que se pueden hacer con el repositorio remoto. Normalmente son **fecth** para ver los cambios que otros desarrolladores han realizado sobre el repositorio remoto. Y **push** para enviar cambios del repositorio local al remoto.



~$ git push origin main

Permite enviar nuestros cambios locales a él repositorio remoto.



~$ git push hhtps://token@github.com/usergihub/reposito-remoto.git

Permite enviar nuestros cambios locales a él repositorio remoto con autenticación por token.

~$ git pull origin main

Permite descargar el contenido de un repositorio remoto (origin) y actualizar un repositorio local (main) para que ambos tengan la misma información.



~$ git fetch

Permite saber si hay cambios en el repositorio remoto sin tener que actualizar el local.



~$ git checkout origin/main

Permite tener los cambios del repositorio remoto en el repositorio local sin mezclar los cambios.