# GIT y GITHUB

Creando mi primer repositorio local

- $ git init  → crea el repositorio
- $ git config  - - global user.name "nombreusuario"→ configuro mi usuario de github
- $ git config  - - global user.email "email github"→ configuro mi email de github

---

Me paro en la carpeta en la que quiero crear un repositorio LOCAL

- $ git add . (agrega todos los archivos de la carpeta al repositorio local)
- $ git status → me indica el estado de los archivos
- $ git commit -m "mensaje" → hacer un commit

---

LLEVAR ARCHIVOS DEL REPO LOCAL A GITHUB
ANTES QUE NADA VER EN GITHUB SI LA RAMA PRINCIPAL SE LLAMA MASTER O MAIN. EN CASO DE QUE SEA DIFERENTE A LA DE MI TERMINAL, CREARLA Y HACER SWITCH A ESA RAMA.

- Crear un repositorio remoto en GitHub (directamente en la web)
- $ git remote add origin  URL_DE_MI_REPOSITORIO_GITHUB (pegar directamente dejando un espacio después del comando)
- $ git remote -v (me permite comprobar que se ejecutó correctamente
- $ git status
- $ git add . → o agrego los archivos que quiero puntualmente porque el punto agrega TODOS los cambios
- $ git commit -m

- $ git push origin master/main → llevar archivos de mi repo local al remoto (que se llama origin) y adicionalmente llevarlo a la rama principal que se llama master o main!

---

PARA ACTUALIZAR MI REPOSITORIO LOCAL TRAYENDO LOS CAMBIOS QUE SE HAYAN HECHO EN LA NUBE:

- $ git pull origin master/main

---

SI YO TRABAJO EN EQUIPO, voy a tener que enviar invitación al repositorio de GitHub o enviarla si lo he creado yo.

una vez que figura el repositorio externo compartido en mi cuenta de github

● Abrir una terminal en la ubicación en donde queramos clonar el proyecto.
● Copiar la URL del repositorio que queremos clonar.
● $ git clone URL →Pegar la URL después de la palabra clone (dejando un espacio de por medio)
y presionar enter

Cuando yo hago git push desde un repositorio clonado en mi pc, pararme en la carpeta "Padre" , por ejemplo cloné camada3_eri y adentro tiene otras carpetas... trabajando desde mi branch hago los push PARADA EN "camada3_eri"

si yo quiero actualizar los archivos que YA TENGO en mi pc → hago $ git pull origin main el mismo no descarga nuevamente todos los archivos sino que baja y actualiza archivos que sufrieron cambios o archivos nuevos!!!

Carpetas sincronizadas con github pero no actualizada : $git pull origin main (cambios que yo mismo hice en otros dispositivos o que hicieron mis compañeros) !

Conflictos: si trabajamos en el mismo archivo en el mismo lugar... git me indica que hay un conflicto y debo resolverlo de forma manual !

**RESUMEN**

| ACCION | COMANDO | Descripción |
| --- | --- | --- |

| ACCION | COMANDO | Descripción |
|---|---|---|
| configurar herramientas | $ git config --global user.name "[name]" | Establece el nombre que desea esté anexado a sus transacciones de commit |
| configurar herramientas | $ git config --global user.email "[email address]" | Establece el e-mail que desea esté anexado a sus transacciones de commit |
| crear repositorios | $ git init [project-name] | Crea un nuevo repositorio local con el nombre especificado |
| crear repositorios | $ git clone [url] | Descarga un proyecto y toda su historia de versión |
| efectuar cambios | $ git status | Enumera todos los archivos nuevos o modificados que se deben confirmar |
| efectuar cambios | $ git add [file] | Toma una instantánea del archivo para preparar la versión |
| efectuar cambios | $ git commit -m "[descriptive message]" | Registra las instantáneas del archivo permanentemente en el historial de versión |
| cambios grupales | $ git branch | Enumera todas las ramas en el repositorio actual |
| cambios grupales | $ git branch [branch-name] | Crea una nueva rama |
| cambios grupales | $ git checkout [branch-name] | Cambia a la rama especificada y actualiza el directorio activo |
| cambios grupales | $ git merge [branch] | Combina el historial de la rama especificada con la rama actual |
| cambios grupales | $ git branch -d [branch-name] | Borra la rama especificada |
| nombres del archivo de refactorizacion | $ git rm [file] | Borra el archivo del directorio activo y pone en el área de espera el archivo borrado |
| nombres del archivo de refactorizacion | $ git rm --cached [file] | Retira el archivo del control de versiones, pero preserva el archivo a nivel local |
| nombres del archivo de refactorizacion | $ git mv [file-original] [file-renamed] | Cambia el nombre del archivo y lo prepara para commit |
|  | $ git revert 3321844 | deshacer un cambio |

| ACCION | COMANDO | Descripción |
|---|---|---|
| repasar historial | $ git log | Enumera el historial de la versión para la rama actual |
| repasar historial | $ git log --oneline | ver nuestro historial de commits |
| sincronizar cambios | $ git fetch [bookmark] | Descarga todo el historial del marcador del repositorio |
| sincronizar cambios | $ git pull | Descarga el historial del marcador e incorpora cambios |
| sincronizar cambios | $ git push [alias] [branch] | Carga todos los commits de la rama local al GitHub |
| | Untitled | |
| | Untitled | |
| | Untitled | |
| | Untitled | |
| | Untitled | |