CREANDO UN REPOSITORIO LOCAL

Para **crear un repositorio local** en una carpeta: Debo posicionarme sobre la carpeta que quiero y

`$git init` → inicio repositorio

`$git config user.name "mi-usuario"` (para verificar es el mismo comando, sin el nombre)

`$git config user.email mail@email.com` (para verificar es el mismo comando, sin el mail)

Si quiero que esto quede seteado para siempre porque es mi compu, se hace así:

`$git init` → inicio repositorio

`$git config --global user.name "mi-usuario"` (para verificar es el mismo comando, sin el nombre)

`$git config --global user.email mail@email.com` (para verificar es el mismo comando, sin el mail)

AGREGANDO O MODIFICANDO ARCHIVOS

`$git add archivo` → Agrego archivos individuales nuevos o modificados

`$git add .` → Agrego TODOS los archivos del repositorio (ojo donde estoy parada, esto hace en subcarpetas pero no hacia arriba)

`$git status` → Me dice el status de los archivos y el estado respecto al repositorio

Cada vez que se modifica un archivo, hay que volverlo a sincronizar con `$git add` y esto permite el control de cambios.

CONFIRMAR ARCHIVOS PARA VUELTA ATRÁS = CREAR COMMITS

Creo un commit en cada instancia importante. Se hace con:

`$git commit -m "mensaje descriptivo"`

`$git log` nos muestra un historial de los cambios realizados al repositorio con commits

Si modifico un archivo luego de hacer un commit, no se tendrá en cuenta para ese commit. Debo primero agregarlo al repositorio con add, y luego hacer un nuevo commit.

DESHACER

Para deshacer un commit: `$git checkout carpeta`

Para deshacer una acción recién hecha hago: `$git restore .`

GitHub

Para UNIR MI REPOSITORIO LOCAL CON LA NUBE copio este link y en la consola hago:

```
$git remote add origin http://github.com/mi-usuario/nombre-del-repositorio.git
```

Para verificar que está bien hago:

`$git remote -v` → Esto nos muestra el origen

Subir archivos del repositorio local a GitHub

Hay que tener en cuenta que sólo se van a subir los commits que hayamos hecho con:

```
$git add .
$git commit -m "mensaje"
```

Para mandar todo a la nube hacemos:

`$git push origin main` → donde: "Origin" es como git denomina al repositorio local
"Main" es la rama principal del repositorio en la nube

Al hacer esto se nos va a pedir usuario y contraseña (en la terminal o como ventana emergente)

Bajar archivos de GitHub al repositorio local

POR PRIMERA Y UNICA VEZ: CLONAR

Si ya creé un repositorio en GitHub y lo quiero descargar en mi compu para trabajarlo debo hacer.

`$git clone http://github.com/mi-usuario/nombre-del-repositorio.git`

Este comando, además de descargar todos los archivos, me sincroniza la carpeta con la nube.

PARA SINCRONIZAR LOS CAMBIOS: ACTUALIZAR

Si se subieron cambios a GitHub y yo quiero actualizarlos en mi compu, debo hacer:

`$git pull origin main` → donde: "Origin" es como git denomina al repositorio local
"Main" es la rama principal del repositorio en la nube

Este comando solo descarga y actualiza los archivos que hayan sufrido modificaciones o hayan sido agregados

RESOLVER CONFLICTOS:

PASO 1 → Traer esos cambios:

```
$git pull origin main
```

Sale un mensaje del cual importan las 3 ultimas líneas, que dicen que Git quiso unificar los cambios del archivo index.html pero no pudo, que lo tenemos que hacer manual

PASO 2 → Abrir el archivo en conflicto: Veremos esto por cada conflicto que haya en el archivo:

PASO 3 → Decidir qué hacer con el código, qué versión dejar.
VSC nos permite hacer esto con un solo click

Pero si lo quiero hacer manual, debo dejar solamente la parte del código que quiero como correcta, y borrar las 3 líneas que agrega Git.

PASO 4 → Guardar y Volver a subir el archivo.

```
$git add index.html

$git commit -m "Resolvi el conflicto de index.html"

$git push origin main
```

RAMAS

GIT BRANCH: Git branch nos permite crear, enumerar, cambiar el nombre y eliminar ramas. No permite cambiar entre ramas o volver a unir un historial bifurcado.

`$git Branch` → **Enumera** todas las ramas de tu repositorio, es similar a `$git branch --list`

`$git branch <branch>` → **Crea una nueva rama** llamada <branch>

`$git branch -m nombreViejoBranch nombreNuevoBranch` → **Cambio nombre de la rama**

`$git branch -d <branch>` → **Elimina la rama** sin cambios llamada <branch>. (Git evita que eliminemos la rama si tiene cambios que aún no se han fusionado con la rama Main)

`$git branch -D <branch>` → **Fuerza la eliminación** de la rama especificada, incluso si tiene cambios sin fusionar

GIT CHECKOUT: Para moverse de una rama a otra, se ejecuta el comando

```
$git checkout nombre_rama
```

Generalmente, Git solo permitirá que nos movamos a otra rama si no tenemos cambios. Si tenemos cambios, para cambiarnos de rama, debemos:

1. Eliminarlos (deshaciendo los cambios).
2. Confirmarlos (haciendo un git commit).

`$git checkout -b nombre_nueva_rama` → creo una rama y me muevo a ella automáticamente. (Para no crearla con Branch y luego pasarme con checkout)

Guardar cambios y subirlos al repositorio remoto:

Una vez que terminamos de realizar los cambios que queremos en nuestra branch, ejecutamos los mismos comandos que vimos hasta ahora: `git add, git commit, git status y git log`.

Pero cuando queramos subir esos cambios, debemos utilizar `git push` con el nombre de la rama en que estamos posicionados:

`$git push origin <branch>` (en vez de main)

Así también, para traer los cambios de esa rama utilizamos el `git pull` agregando desde donde

queremos traer los cambios:

`$git pull origin <branch>` (en vez de main)

MOVERME ENTRE COMMITS:

HEAD me indica en qué commit estoy posicionada y con qué rama:

`$git checkout numero_de_hash_del_commit` → si hago `$git log` me da los commits "resumidos"

Me devuelve en el tiempo a cómo estaban los archivos al haber hecho ese commit. Si en este momento hago git log, me va a mostrar de ese punto hacia atrás. No puedo volver hacia adelante:

Si quiero adelantarme a un commit posterior debo volver a la rama donde estaba trabajando y después hacer git log, ahí me va a mostrar todos los commits que se hicieron (anteriores y posteriores)