Repositorios:

Crear repositorio:

- git init *// crea el repositorio*

Para identificarse en la terminal Git con el repositorio debemos agregar la cuenta con la cual nos registramos en Git, a través de la terminal, con los siguientes comandos:

Identifica temporalmente:

- git config user.name "mi-usuario" *// no tendremos respuesta de la terminal*
- git config user.email "micorreo@mail.com" *// no tendremos respuesta de la terminal*

Identifica permanentemente:

- git config --global user.name "mi-usuario" *// no tendremos respuesta de la terminal*
- git config --global user.email "micorreo@mail.com" *// no tendremos respuesta de la terminal*

Para chequear la configuración:

- git config user.name *// respuesta user.name*
- git config user.email "micorreo@mail.com" *// respuesta user.email*

Agregar archivos al repositorio local:

- Ingresar a la carpeta en la cual está inicializado el repositorio
- git add nombre.archivo *// indicar el o los archivos que queremos agregar*
- git add .(punto) *// indica que se desea agregar todos los archivos que se encuentran en el repositorio*
- git status *// te dice el status de los archivos y el estado de los mismos con respecto al repositorio*
- git add .(punto) o git add nombre.archivo *// al modificar un archivo debemos agregarlo al commit nuevamente*

Crear commit:

- git add .(punto) o git add nombre.archivo *// Previo debemos agregar los archivos modificados al repositorio*
- git commit -m "mensaje" *// En el cuerpo del mensaje deberemos escribir, de manera resumida, el trabajo hecho hasta ese momento. El mensaje va en " "*
- git add "nombre.archivo"
- Cada vez que se agrega / modifica un archivo se debe realizar este proceso

Historial de proyectos:

- git log *// Registra un historial de los cambios de nuestros proyectos (commits)*

Explorador de Visual Studio Code:

- U (Untracked) *// No están agregados al repositorio*
- A (Added) *// Agregados en el repositorio*
- M (Modified) *// Archivo del repositorio modificado*
- Sin estado (Ni U ni A) *// Ya están comiteados*

Conectando el repositorio local a GitHub

Para conectar el repositorio debemos tener:

1. Una cuenta en GitHub.
2. Un repositorio local que utilizaremos para poder conectarlo.

Configurar repositorio local:

1. Debemos inicializar un repositorio
   → git init *// en la carpeta que queramos conectar el repositorio.*
2. Luego tenemos que indicar al repositorio nuestro usuario ejecutando dos comandos:
   → git config user.name "mi usuario" *// escribimos nuestro nombre de usuario*
   → git config user.email "miCorreo@email.com" *// escribimos nuestra dirección de correo*

Conectar al repositorio remoto:

- Crear repositorio en GitHub
- git remote add origin URL.exacta.del.repositorio.remoto *// La terminal no envía mensaje*
- git remote -v *// Sirve para verificar este correcta la sincronización*
  → origin https://github.com/EmiiFernandez/PrimerRepositorio.git (fetch)
  → origin https://github.com/EmiiFernandez/PrimerRepositorio.git (push)

Subir archivos a GitHub

- git add .(punto) *// agrega todos los archivos*
- git commit -m "mensaje" *// comitea los cambios hechos*
- git push origin main *// envía los cambios al repositorio remoto (GitHub)*
- git status *// seguimiento del estado de los archivos*

Descargar los proyectos de GitHub a la PC

- git clone URL.exacta.del.repositorio.remoto *// Permite crear una copia exacta en la computadora de todos los archivos existentes en un repositorio remoto*
- Se ejecuta por única vez en la computadora
- Si se selecciona Download ZIP nos descargará los archivos, pero no estarán sincronizados

Actualizar los proyectos de GitHub a la PC

- En caso de ya tener descargado el repositorio y debemos actualizar los archivos que se encuentran en la computadora.
- git pull origin main *// solo descarga los archivos modificados o nuevos que no están en la computadora*

Resolviendo Conflictos

Conflicto de fusión en archivo

- Fallo en proceso de guardado por que dos o más personas intentaron guardar al mismo tiempo el mismo archivo.
  1. git pull origin main *// traer los archivos modificados a nuestra computadora*
     → En el editor que usemos se verá un mensaje donde "HEAD" es nuestro archivo modificado y después de "======" es el archivo modificado de otra persona.
  2. Definir si unificamos los archivos o si dejamos alguno en particular
  3. Resolver el conflicto:
     → Automáticamente con el editor, como con Visual Studio Code que puede comprender el formato y nos brindará una opción para solucionar el conflicto.

→ Manualmente sería dejar la parte del código que determinamos que es la correcta y eliminar el resto.
→ En algunos casos, Git podrá determinar cómo unificar los cambios de manera automática y, en ese caso, todo el trabajo se hace instantáneamente. Esto sucede en casos donde se trabajó en partes separadas del mismo archivo.

4. Guardamos el archivo actualizado. Repetimos el proceso como si se tratará de un nuevo cambio
   → git add "archivo"
   → git commit -m "mensaje"
   → git push origin main

- Recomendaciones:
  → Intentar tener los commit relativamente pequeños y subir al servidor frecuentemente
  → Usar ramas para trabajar en paralelo a la versión original del código

Ramas

Comandos:

● git branch *// nos permite crear, enumerar, cambiar el nombre y eliminar ramas. No permite cambiar entre ramas o volver a unir un historial bifurcado. Enumera todas las ramas de tu repositorio, es similar a git branch --list.*

● git branch <branch> *// Crea una nueva rama llamada <branch>.*

● git branch -d <branch> *// Elimina la rama 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 nombre_rama *// Para moverse de una rama a otra, se ejecuta el comando. 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).

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.

● git push origin <branch> *// Cuando queramos subir los cambios, debemos utilizar git push con el nombre de la rama en que estamos posicionados*

● git pull origin <branch> *// para traer los cambios de esa rama utilizamos el git pull agregando desde donde queremos traer los cambios*