Git y GitHub

Introducción

- **Git**: es un **sistema de control de versiones** que te permite guardar el historial de tu código, volver a versiones anteriores y trabajar en paralelo con otros desarrolladores.
- **GitHub**: es una **plataforma en la nube** que aloja repositorios Git y facilita la colaboración, revisión de código, reportes de issues, y publicación de proyectos.

Con Git trabajas **localmente** en tu computadora, y con GitHub **remotamente** para compartir y colaborar.

Ejemplo:

Fork + Pull Request en GitHub

1.- Fork

Un **fork** es una copia completa de un repositorio en tu cuenta de GitHub.

1. Ve al repositorio en GitHub que quieres contribuir.
2. Haz clic en **"Fork"** (arriba a la derecha).
3. Esto creará una copia en tu cuenta para que puedas modificarla sin afectar el original.

2.- Clonar tu fork

```
git clone https://github.com/TU-USUARIO/REPO-FORKEADO.git
cd REPO-FORKEADO
```

- Esto copia tu fork a tu máquina.

3.- Crear una rama para tus cambios

```
git checkout -b mi-rama-feature
```

- Trabajar en una rama evita modificar directamente la rama principal (`main` o `master`).

4.- Hacer cambios y confirmarlos

```
# Editas los archivos en tu editor...
git add .
git commit -m "Agregué nueva funcionalidad X"
```

5.-Subir tus cambios a tu fork

```
git push origin mi-rama-feature
```

6.- Hacer un Pull Request(PR)

1. Ve a tu fork en GitHub.
2. Verás un botón para **"Compare & pull request"**.
3. Escribe una descripción clara de tus cambios.
4. Envía el PR al repositorio original para que lo revisen y puedan integrarlo.

Comandos clave para trabajar con Git/GitHub

Inicializar repositorio

`git init` → Crea un repositorio Git en la carpeta actual

Clonar repositorio

`git clone URL` → Copia un repositorio remoto en local

Ver estado de cambios

`git status` → Muestra archivos modificados y sin confirmar

Agregar cambios al área de staging

```
git add archivo.txt
git add .  → Prepara cambios para confirmar (commit)
```

Confirmar cambios

`git commit -m "Mensaje descriptivo"` → Guarda cambios en el historial local

Cambiar de rama

`git checkout -b nueva-rama` → Crea y cambia a una nueva rama

Subir cambios al remoto

`git push origin rama` → Envía commits a GitHub en la rama especificada

Traer cambios del remoto

`git pull origin rama` → Actualiza la rama local con cambios de GitHub

Ver historial de commits

`git log --oneline --graph --decorate` → Muestra historial de manera resumida y visual

Fusionar ramas

`git merge otra-rama` → Une cambios de otra rama en la actual

Agregar un remoto adicional

`git remote add upstream URL` → Conecta tu repo local con otro remoto (ej. original antes del fork)

Traer cambios del repo original

```
git fetch upstream
git merge upstream/main
```
→ Actualiza tu fork con los cambios del original

Resolver conflictos (Editar archivos, luego)

```
git add archivo
git commit
```
→ Guarda la resolución del conflicto

Explicacion general de que hace cada comando

- **`git init`** → empieza a seguir los cambios de tu proyecto.
- **`git clone`** → copia un repositorio de internet a tu PC.
- **`git status`** → te dice qué archivos cambiaste y cuáles están listos para guardar.
- **`git add`** → selecciona los cambios que quieres confirmar.
- **`git commit`** → guarda un "punto en el tiempo" del proyecto.
- **`git push`** → envía tus cambios a GitHub.
- **`git pull`** → trae cambios nuevos desde GitHub.
- **`git checkout`** → cambia de rama o crea una nueva.
- **`git merge`** → combina cambios de diferentes ramas.
- **`git remote add`** → conecta tu repo local con otro en línea.
- **`git fetch`** → descarga cambios sin aplicarlos todavía.
- **`git log`** → ve el historial de cambios.
- **`git branch`** → lista o crea ramas.