**Practica Git**

**1- Investigar 2 comandos no vistos en clase (Ej: Git rebase, Git Fetch)**

|  |  |
| --- | --- |
| $ git add \_myfile.txt \_ // | Agrega el archivo en Staging |
| git diff | Muestra los cambios realizados a un archivo |

**2- Investigar Git Fork y cómo funciona**

Un fork significa bifurcación, y en programación supone copiar un proyecto y partir de este, hacerle modificaciones. Cuando trabajamos con repositorios Git, supone hacer una copia exacta del proyecto, generando dos URL distintas.

Para qué necesito un Fork

Un fork es una copia de un repositorio, pero ¿por qué no clonamos el repositorio que queremos copiar y listo?

Si haces un clon normal de un repositorio, el espacio en GitHub de ese clon seguirá asociado al repositorio que has clonado. De este modo, si realizas cambios sobre el clon y los quieres publicar en GitHub, probablemente no los podrás subir.

Obviamente, si clonas un repositorio que era tuyo, podrás realizar cambios en local y subirlos a GitHub siempre que quieras. Pero si el repositorio era de otro desarrollador y tú no tenías permisos de escritura sobre él, entonces no podrás subir cambios, porque GitHub no te lo permitirá. Para este caso es donde necesitas un fork.

Como hemos dicho, un fork es una copia de un repositorio, pero creado en tu propia cuenta de GitHub, donde sí que tienes permisos de escritura. Por tanto, si tienes intención de bajarte un repositorio de GitHub para hacer cambios en él y ese repositorio no te pertenece, lo más normal es que crees un fork primero y luego clones en local tu propio fork.