**GIT**

Es un **software de control de versiones** pensado para mantener eficientemente las actualizaciones sobre el código fuente (las modificaciones que hagamos. Permite el acceso a versiones anteriores). Permite hacer un completo back up de tus archivos y al mismo tiempo compartirlos con tus colaboradores. Lleva registro de los cambios de archivos y coordina el trabajo que varias personas realizan sobre los mismos. (permite ver quién realizó las últimas modificaciones)

Si tenes git, en la terminal pones git –version y aparecerá a versión.

**Repositorios:** son “almacenes” que contienen a los archivos en “paquetes” (commits 🡪 historial de cambios. Tienen fecha y hora y creador). el **local** vive en la pc, y el **remoto** en GitHub

Git init 🡪 crea repositorio

Para decirle a git quien soy:

Git config user.email “mail”

Git config user.name “nombre de usuario con el que te registraste en git”

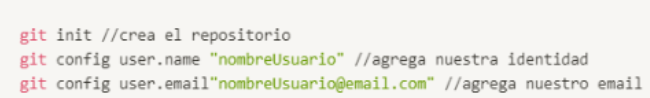
En ambos casos, si escribo el **comando solo** luego de hacerlo me trae la información de las comillas que ingresé **(para corroborar)**

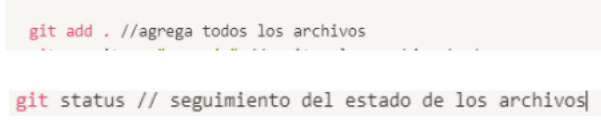
Si luego de config agrego **- -global** le indico que **CUALQUIER repositorio de mi pc me pertenece**

Los repositorios creados en github tienen una URL única. Con esa url escribo el comando

Git remote add origin \*pego url\* y asi sincronizo los repositorios

Git remote -v para chequear





Los que aparezcan sin seguimiento son porque o no fueron agregados al repositorio o bien porque fueron modificados luego de ser agregados por lo que el programa detecta que hay una nueva versión.



Git log carga lo que fue commiteado y muestra por quien, fecha y hora.

Git rm saca de seguimiento (contrario a git add)

Para especificar que git no realice más seguimiento a un archivo, usamos el comando git rm --cached donde si bien el archivo como tal no se ha eliminado, Git procede a ignorarlo y pasa a un estado de untracked.

