**GIT**

Sistema de control de versiones distribuido de código abierto, eficaz y fluido; tiene la capacidad de almacenar las distintas versiones de un mismo archivo para poder restaurarlas, mezclarlas y recuperarlas. Una de las características principales es que no depende de un repositorio central (lugar donde se almacenan las versiones y archivos). Que el sistema sea distribuido hace que gane fluides a la hora de trabajar.

**Controles principales GIT:**

* **Git init -------------------🡪** Creación de un repositorio (en la carpeta que deseemos)
* **Git status** ---------------🡪 Verificar el estado del repositorio
* **Git add**-------------------🡪 Pasa el archivo/carpeta al stage área (color verde)

\*Git add ------🡪Agrega todos los archivos modificados o trackeados al stage área.

\*Git add \*.jpj 🡪Agrega al stage área todos los elementos cuya terminación sea jpg

\*Git add ruta al /directorio/ o del archivo 🡪 agrega todos los archivos o elementos de ese directorio

* **Git commit -m “nombre-comic”** -----🡪Confirma los cambios del stage área y los guarda en el repositorio (“-m” se utiliza para asociar el comic con un mensaje descriptivo”
* **Git log** -------------🡪 Sirve para ver el histórico o la línea de tiempo de los cambios realizados en el repositorio.
* **Git diff** -------------🡪 Compara la situación actual, con el ultimo comic que hemos hecho (generalmente aparece en colores <rojo=como estaban antes>: <verde=actual>)
* **Git reflog** ---------🡪 Se ve el historial mas detallado de los cambios en el repositorio, aun aquellos comic editados y borrados. Describe todo.
* **Git checkout** --🡪 Una de sus tantas funciones es la de cambiar de ramas.
* **Git clone** ------🡪 Comando para descargarte el código fuente que se encuentra en un repositorio remoto (Git hub); realiza una copia idéntica de la ultima versión de un proyecto de un repositorio y lo guarda en tu ordenador.
* **Git commit –amen -m “nombre-comic” -----🡪** Sirve renombrar el comic.
* **Git mv nombre-archivo nuevo-nombre-archivo** --🡪 Renombra el archivo.
* **Git rm archivo-eliminar -------------🡪** Borrar/deleted archivo o carpeta.
* **Git Branch nombre-rama---------------🡪** Crea una rama y le coloca un nombre.
* **Git Branch -d nombre-rama-eliminar -------------------🡪** Elimina la rama
* **Git log –oneline –decorate –all –graph ---🡪** Forma resumida y visual de donde nos encontramos.
* **Git checkout -b nombre-rama -----🡪** Sirve para crear una rama y directamente dirigirnos a ella.
* **Git tag nombre-etiqueta -----------🡪** Sirve para crear una etiqueta del commit o rama.
* **Git tag -d nombre-etiqueta -----🡪** Elimina la etiqueta del commit o rama
* **Git tag -a v1.0.0 -m “nombre-versión” ---🡪** Agrega una versión a la etiqueta ( -a 🡪adicional).