Práctica 0

1. Ejecutá git o git help en la línea de comandos y mirá los subcomandos que tenés disponibles.

   Al ejecutar git o git help, se ven los subcomandos ofrecidos por git como: add, commit, push, pull, clone, status, log, branch, checkout, merge, fetch, etc.

2. Ejecutá el comando git help help. ¿Cuál fue el resultado?

   El comando muestra ayuda sobre cómo usar el sistema de ayuda de Git y cómo buscar documentación de subcomandos.

3. Utilizá el subcomando help para conocer qué opción se puede pasar al subcomando add para que ignore errores al agregar archivos

   Con git help add se ve que la opción para que se ignoren errores al agregar archivos es --ignore-errors.

4. ¿Cuáles son los estados posibles en Git para un archivo? ¿Qué significa cada uno?

   - Untracked: El archivo no está bajo control de versiones.
   - Tracked: El archivo está bajo control de versiones, puede estar en uno de estos subestados:
     - Unmodified: Sin cambios desde el último commit.
     - Modified: Editado desde el último commit.
     - Staged: Preparado para ser confirmado (commit).

5. Cloná algun repositorio que encuentres en GitHub que sea de tu interés. Una vez finalizado, ¿cuál es el hash del último commit que hay en el repositorio que clonaste?

   Al clonar (git clone <url>) y ejecutar git log, el primer commit que se ve es el último de la rama actual (HEAD). Si se realiza un git checkout el ultimo commit que se ve es donde esta el HEAD sin la necesidad de que este sea el ultimo de la rama.

6. ¿Para qué se utilizan los siguientes subcomandos?

   - init: inicializa un nuevo repositorio Git en el directorio actual.

   - status: muestra el estado actual del working directory y el staging área, indicando qué archivos han sido modificados, cuáles están listos para commitear y cuáles no están siendo seguidos por Git.

   - log: muestra el historial de commits. Se puede usar -n <#> para limitar la cantidad mostrada.

- fetch: descarga los cambios (nuevos commits, ramas, etc.) desde el repositorio remoto al local, pero no los fusiona con el trabajo actual. Permite revisar cambios antes de integrarlos.

- merge: fusiona la rama especificada con la rama actual, combinando los cambios de ambas. Agrega un commit extra de merge. Es ideal para escenarios simples y es el método por defecto de Git para unir ramas.

- pull: descarga los cambios de la rama especificada desde el repositorio remoto y los fusiona automáticamente con la rama actual del repositorio local.

- commit: confirma los cambios que están en el área de staging, creando un nuevo commit en el repositorio local. Se puede incluir un mensaje con -m describiendo el propósito de los cambios.

- stash: guarda temporalmente los cambios no confirmados (en el directorio de trabajo o en staging) y limpia el área de trabajo. Permite volver a un estado "limpio" para cambiar de rama o atender otra tarea, y luego restaurar esos cambios guardados.

- push: sube los commits de la rama actual al repositorio remoto. Se puede especificar el remoto y la rama (git push origin main). Existen variantes como --force (sobrescribe el remoto) y --force-with-lease (más segura, advierte si hubo cambios remotos intermedios).

- rm: elimina archivos tanto del directorio de trabajo como del control de versiones (repositorio). Al ejecutar git rm <archivo>, Git lo marcará para que se borre en el próximo commit.

- checkout: permite cambiar de rama o posicionarse en un commit determinado (modo "detached"). Para cambio de ramas, se recomienda el uso de git switch <nombre_de_rama>.

- tag: crea una marca o etiqueta inmutable en un commit específico, útil para señalar versiones (ejemplo: v1.0.0) o hitos importantes en el historial del proyecto.

- clean: elimina archivos no rastreados del directorio de trabajo, es decir, borra los archivos que aún no fueron añadidos al control de versiones y no figuran en el stash ni en el index.

7. Utilizá el subcomando log para ver los commits que se han hecho en el repositorio, tomá cualquiera de ellos y copiá su hash (por ejemplo, 800dcba6c8bb2881d90dd39c285a81eabee5effa), y luego utilizá el subcomando checkout para viajar en el tiempo (apuntar tu copia local) a ese commit. ¿Qué

commits muestra ahora git log? ¿Qué ocurrió con los commits que no aparecen? ¿Qué dice el subcomando status?

Después de hacer git checkout <hash>, git log solo muestra los commits hasta ese punto en el historial, los posteriores "desaparecen" temporalmente. git status indica que se está en un "detached HEAD", es decir, un commit que no es apuntado por ninguna rama.

8. Volvé al último commit de la rama principal (main) usando nuevamente el subcomando checkout. Corroborá que efectivamente haya ocurrido esto.

   Para volver al último commit de la rama principal se debe usar git checkout main o git switch -. Se puede confirmar que se volvió al último commit de la rama ejecutando git log y revisando que se esté en la rama (HEAD -> main).

9. Creá un repositorio en el servicio on line de hosting de Git de tu preferencia (los más conocidos son GitHub, GitLab y Bitbucket) y cloná el repositorio localmente, creá algunos archivos y directorios en el repositorio local, y subí tus cambios al remoto. Luego modificá desde la interfaz web que ofrecen estos servicios algún archivo de los que creaste, creá un commit desde allí mismo, y finalmente actualizá tu copia local de manera que puedas ver esos cambios. ¿Qué subcomando utilizaste para esto? ¿Qué hizo?

   El subcomando usado es git pull. Este comando descarga los cambios remotos y los fusiona con la copia local, permitiéndote ver los cambios hechos en el repositorio remoto.