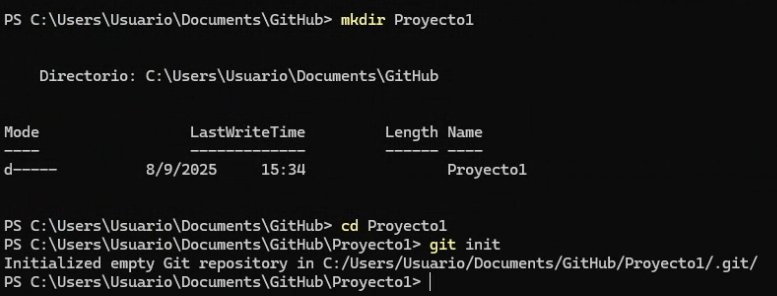
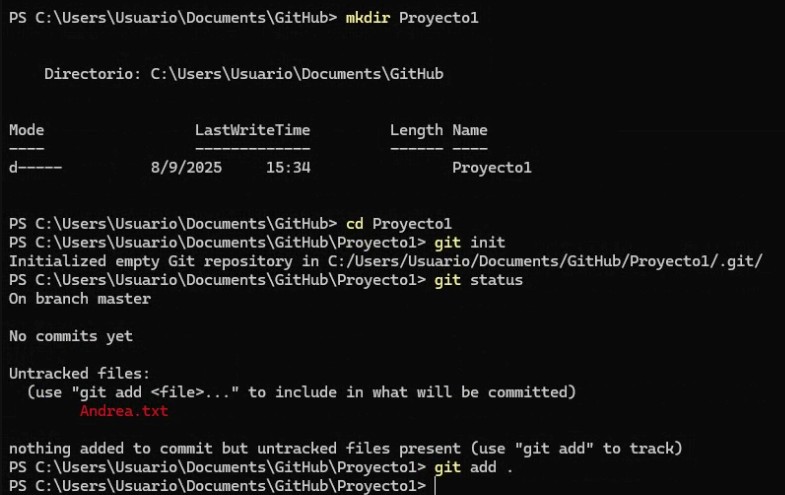
**CAPTURAS**

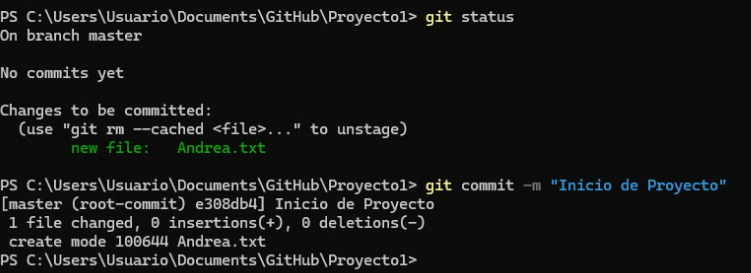


* **mkdir Proyecto1**
* Crea una carpeta nueva llamada **Proyecto1** dentro de C:\Users\Usuario\Documents\GitHub.
* **Confirmación del directorio creado**
* La terminal muestra que efectivamente se creó el directorio con fecha y hora de creación.
* **cd Proyecto1**
* El usuario entra a la carpeta recién creada.
* **git init**
* Inicializa un repositorio vacío de **Git** dentro de esa carpeta.
* Esto crea un subdirectorio oculto llamado **.git**, que es donde Git guarda toda la información de control de versiones del proyecto.
* La captura muestra el inicio de un nuevo proyecto en GitHub localmente. Se creó una carpeta llamada **Proyecto1**, se accedió a ella e inicializó un repositorio Git vacío en esa carpeta.



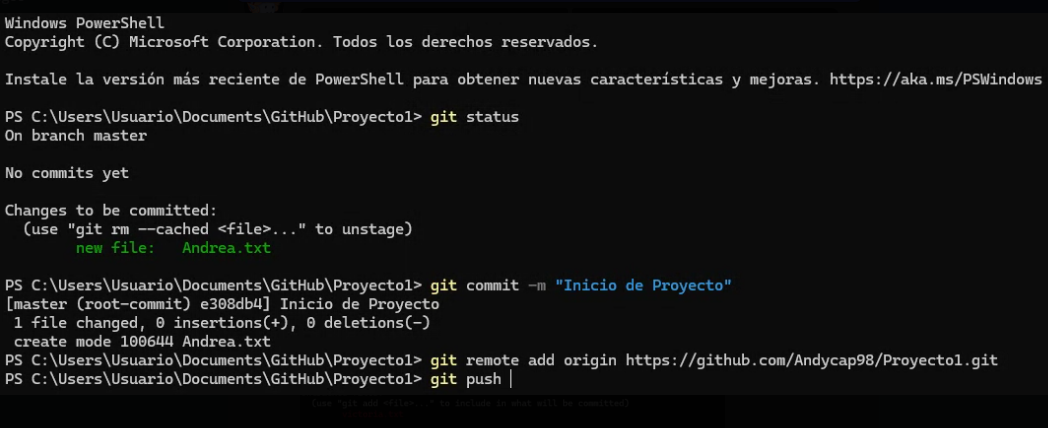
1. Ya se había creado la carpeta **Proyecto1** y se había inicializado un repositorio con git init.
2. **git status**
   * Muestra el estado del repositorio.
   * Git informa que aún no hay commits (No commits yet) y que existe un archivo **Andrea.txt** como **untracked file** (archivo no rastreado por Git).
3. **git add .**
   * Con este comando se añaden todos los archivos de la carpeta (en este caso, **Andrea.txt**) al **área de preparación (staging area)**.
   * Eso significa que el archivo ya está listo para ser guardado en el próximo commit.

👉 En resumen:  
El usuario creó un archivo **Andrea.txt** dentro de la carpeta del proyecto, verificó con git status que Git aún no lo estaba siguiendo, y luego usó git add . para que Git lo rastree y quede listo para un commit.



* En esta tercera imagen ya se hizo el **primer commit** en el repositorio.
* **git status**
  + Muestra que en el área de *staging* ya está el archivo **Andrea.txt** listo para ser confirmado en un commit.
  + Git lo indica en verde como new file: Andrea.txt.
* **git commit -m "Inicio de Proyecto"**
  + Se crea el primer commit en la rama **master**.
  + El mensaje "Inicio de Proyecto" describe el commit.
  + Git informa:
    - El identificador del commit: e308db4 (abreviado).
    - Que se agregó **1 archivo** (Andrea.txt).
    - Que ese archivo fue creado (con create mode 100644).
* En resumen:  
  El usuario ya **guardó la primera versión del proyecto** en Git. El archivo **Andrea.txt** ahora forma parte del historial del repositorio.
* Lo siguiente que se suele hacer es:
* Crear un repositorio remoto en GitHub.
* Conectarlo con el local usando:
* git remote add origin https://github.com/usuario/Proyecto1.git
* Subir el commit con:

git push -u origin master



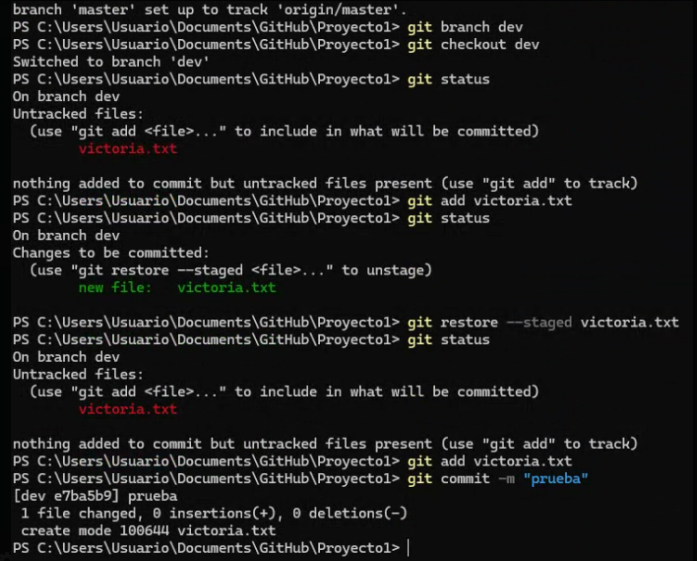
 Se confirma el archivo **Andrea.txt** con el commit "Inicio de Proyecto".

 Se conecta el repositorio local con GitHub usando:

git remote add origin <https://github.com/Andycap98/Proyecto1.git>

1. Finalmente, se usa git push para subir el proyecto al repositorio remoto en GitHub.

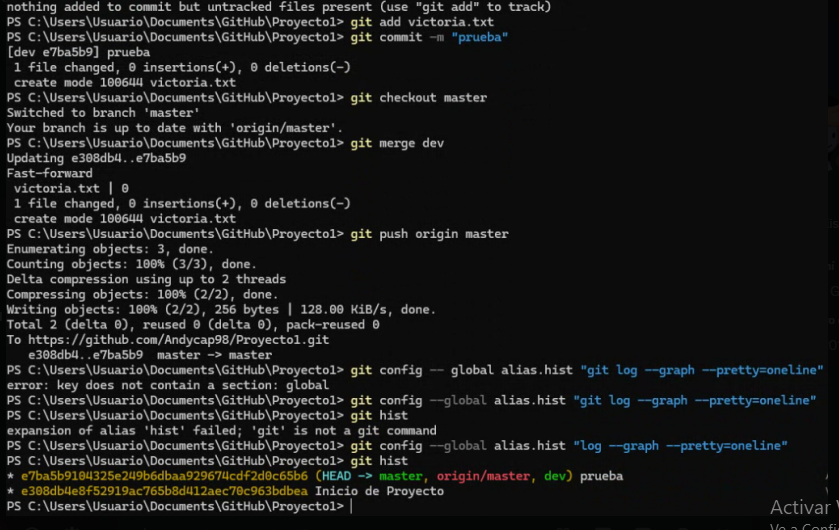
👉 En resumen: ya quedó publicado el **primer commit** de tu proyecto en GitHub ✅.



En esta captura se trabaja con **ramas en Git**. Lo más importante:

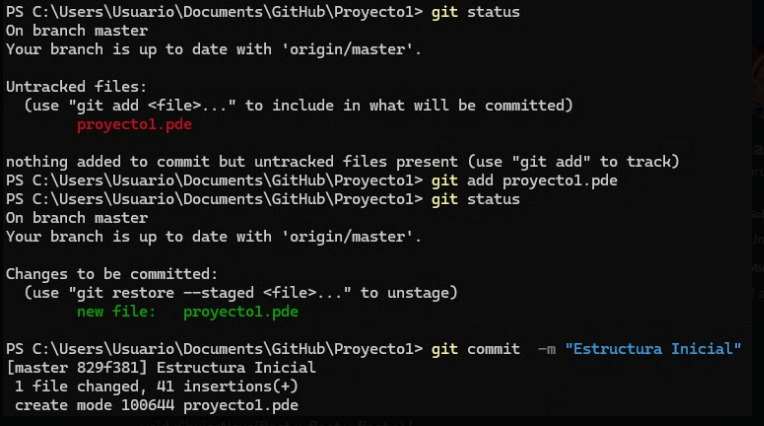
1. **git branch dev** → Creaste una rama nueva llamada **dev**.
2. **git checkout dev** → Cambiaste a la rama **dev**.
3. Creaste un archivo **victoria.txt** y lo agregaste con git add victoria.txt.
4. Probaste git restore --staged victoria.txt para sacarlo del *staging*, y luego lo volviste a agregar.
5. Finalmente hiciste un commit con:

git commit -m "prueba"



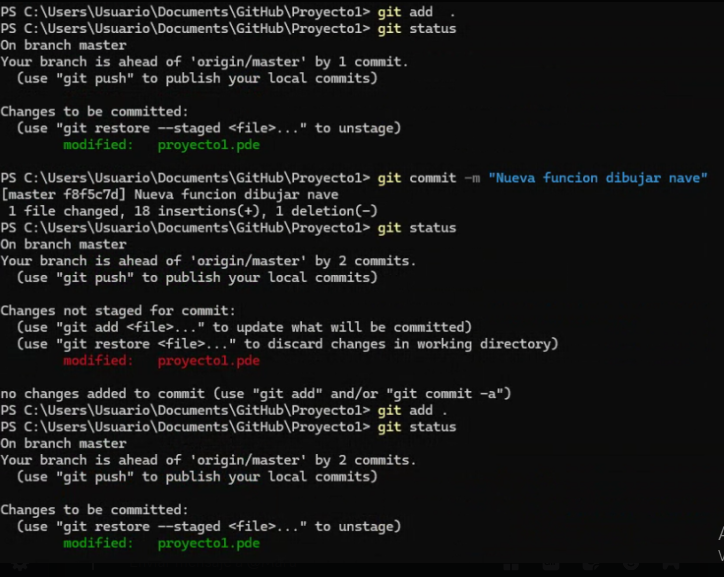
1. **Preparación del commit**
   * git add victoria.txt → Se agrega el archivo victoria.txt al área de preparación.
   * git commit -m "prueba" → Se crea un commit con el mensaje "prueba" incluyendo ese archivo.
2. **Cambio de rama**
   * git checkout master → Cambias a la rama master.
   * git merge dev → Se fusiona la rama dev en master. Se hizo un **fast-forward** (no hubo conflictos, solo se avanzó la referencia).
3. **Envío al repositorio remoto**
   * git push origin master → Subes los cambios de master al repositorio remoto en GitHub.
4. **Intentos de configurar alias de Git**
   * Se creó un alias hist para mostrar el historial con formato bonito, pero hubo errores en la sintaxis de los comandos (git config --global alias.hist "log …").
   * Algunos errores se deben a comillas incorrectas o a que git no reconoce el comando completo.
5. **Estado final**
   * git log --oneline --graph --all --pretty=oneline (o comando similar que querías) aún no funciona, pero Git te muestra que la rama master está alineada con dev y que el HEAD apunta correctamente:
   * e308bd4 (HEAD -> master, origin/master, dev) Inicio de Proyecto
   * Esto significa que la rama master local, la rama master remota y la rama dev tienen el mismo commit más reciente llamado "Inicio de Proyecto".

En resumen: **Se hizo un commit, se combinó ramas y se subió todo al repositorio remoto. Los alias de Git no quedaron configurados correctamente.**

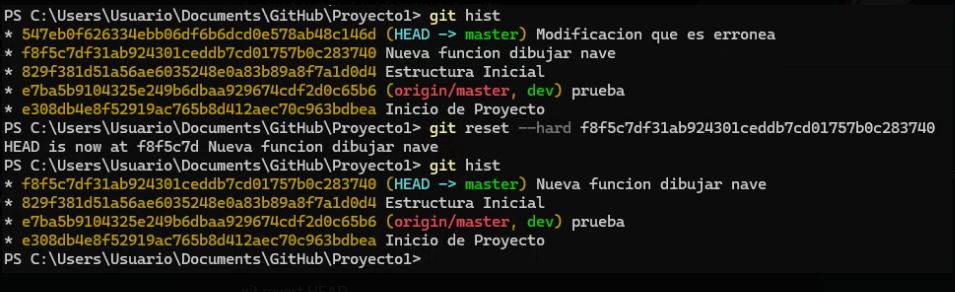


1. **git status**
   * Muestra el estado actual del repositorio.
   * Git indica que hay un archivo no rastreado (proyecto1.pde). Esto significa que todavía no forma parte del control de versiones.
2. **git add proyecto1.pde**
   * Se añade el archivo al área de preparación (*staging area*), listo para ser confirmado (*commit*).
3. **git status** (de nuevo)
   * Ahora Git muestra que proyecto1.pde está listo para ser confirmado. Aparece en verde como *new file*, indicando que se agregará al próximo commit.
4. **git commit -m "Estructura Inicial"**
   * Se crea un commit con el mensaje "Estructura Inicial".
   * Esto guarda el estado actual del archivo en el historial del repositorio.

**En resumen:**  
Se agregó un nuevo archivo (proyecto1.pde) al repositorio de Git y se creó un commit inicial con ese archivo. Ahora el repositorio tiene su primera versión de ese archivo bajo control de versiones.



1. **Estado inicial**
   * El usuario ejecuta git add . → esto agrega todos los cambios al *staging area*.
   * git status muestra que hay cambios listos para ser confirmados (el archivo proyecto1.pde está modificado).
   * El branch local está **1 commit adelantado** respecto a origin/master (todavía no hizo git push).
2. **Primer commit**
   * Ejecuta git commit -m "Nueva funcion dibujar nave" → crea un nuevo commit con ese mensaje.
   * El commit registra 18 líneas agregadas y 1 eliminada.
   * git status ahora indica que el branch está **2 commits adelantado** respecto a origin/master.
3. **Nuevos cambios sin commitear**
   * Después del commit, git status muestra que el archivo proyecto1.pde volvió a modificarse, pero esos cambios aún no están en el staging area.
4. **Segundo git add**
   * El usuario ejecuta git add . nuevamente → agrega los nuevos cambios al staging area.
   * git status muestra que ahora **hay cambios listos para ser confirmados otra vez** (otra modificación en proyecto1.pde).
   * El branch sigue estando **2 commits adelantado** porque aún no se ha hecho el git commit ni el git push.



### 1. Comando git hist

El usuario visualiza el historial simplificado de commits.  
Se ven los siguientes commits (de más reciente a más antiguo):

* **HEAD → master**: *“Modificación que es errónea”*
* *“Nueva función dibujar nave”*
* *“Estructura inicial”*
* *(origin/master, dev)* *“prueba”*
* *“Inicio de proyecto”*

Esto significa que el último commit hecho fue uno erróneo (y el usuario lo quiere descartar).

**2. git reset --hard <commit>**

El usuario ejecuta:

git reset --hard f8f5c7d31ab92431ceddb7cd12757bc2837740

Ese f8f5c7d... corresponde al commit **“Nueva función dibujar nave”**.

Lo que hace este comando:

* **Mueve HEAD y master** hacia el commit *“Nueva función dibujar nave”*.
* Elimina el commit posterior (*“Modificación que es errónea”*) como si nunca hubiera existido.
* También descarta los cambios de trabajo (por el --hard).

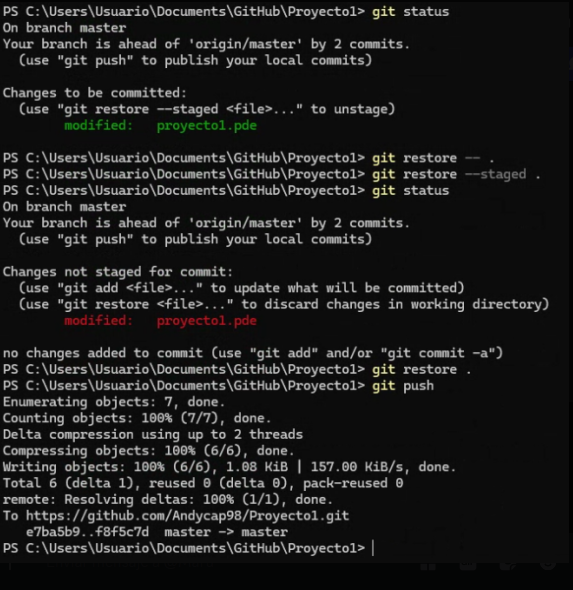
**3. git hist (segunda vez)**

Después del reset, al volver a listar el historial:

* El commit más reciente ahora es *“Nueva función dibujar nave”* (y HEAD → master apunta allí).
* El commit erróneo desapareció.

**En resumen**:

* Antes había un commit equivocado.
* El usuario hizo git reset --hard para regresar al commit correcto.
* Ahora la rama master está limpia, sin el commit erróneo.



### 1. ****Estado inicial (****git status****)****

* El branch está en **master**.
* Está adelantado 2 commits con respecto a origin/master (o sea, hay commits locales que todavía no se subieron al remoto).
* proyecto1.pde aparece en verde como **staged** (listo para commit).

### 2. ****Restaurando cambios****

El usuario prueba varios comandos de git restore:

* git restore . → descarta cambios en el working directory.
* git restore --staged . → saca los archivos del staging area.
* Luego hace git status y ahora proyecto1.pde aparece en rojo: significa que **está modificado en el directorio de trabajo, pero no en staging**.

Después, ejecuta:

* git restore . nuevamente → descarta por completo las modificaciones locales en proyecto1.pde.
* git status confirma: **no hay cambios pendientes**.

En este punto, el proyecto quedó limpio (sin modificaciones locales).

### 3. git push

* Como todavía había 2 commits locales que no estaban en el remoto, se ejecuta git push.
* Se suben los commits al repositorio en GitHub (origin master).
* El proceso de envío muestra:
  + Enumerating objects, Counting objects, Writing objects...
  + Al final confirma:
  + master -> master
  + Esto significa que ahora la rama local y la remota están sincronizadas.

**En resumen**:

1. Había cambios en el archivo proyecto1.pde que estaban staged.
2. El usuario los descartó con git restore (primero del staging, luego del working directory).
3. Finalmente hizo git push y subió los commits pendientes al repositorio remoto.
4. Ahora el repositorio local y remoto están iguales, sin cambios pendientes.