

# **Instalación recursos Git, administración de credenciales y Uso de los principales comandos de GIT:**

## **Instalación recursos GIT:**

En primer lugar procedemos a acceder a la siguiente dirección:

<https://git-scm.com/>

Una vez aquí descargamos la última versión:



Una vez abrimos el instalado pulsamos en next hasta que se instale:



Para comprobar si se ha instalado correctamente vamos a Windows Power Shell y escribimos Git:

```
Windows PowerShell
Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\rafag> git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
      [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
      [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
      [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
      <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone                Clone a repository into a new directory
  init                 Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add                  Add file contents to the index
  mv                   Move or rename a file, a directory, or a symlink
  restore              Restore working tree files
  rm                   Remove files from the working tree and from the index
  sparse-checkout      Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
  bisect               Use binary search to find the commit that introduced a bug
  diff                 Show changes between commits, commit and working tree, etc
  grep                 Print lines matching a pattern
  log                  Show commit logs
  show                 Show various types of objects
  status               Show the working tree status

grow, mark and tweak your common history
  branch               List, create, or delete branches
  commit               Record changes to the repository
  merge                Join two or more development histories together
  rebase               Reapply commits on top of another base tip
  reset                Reset current HEAD to the specified state
  switch               Switch branches
  tag                  Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
  fetch                Download objects and refs from another repository
  pull                 Fetch from and integrate with another repository or a local branch
  push                 Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
```

Ya tendríamos instalado el servicio de git por lo que ya podríamos subir nuestros archivos a github.

## Administración de credenciales:

Ahora procederemos a ver como se administran las credenciales de GitHub:

Vamos a este apartado y buscamos Github:


Administrador de credenciales


Panel de control > Cuentas de usuario > Administrador de credenciales

[Ventana principal del Panel de control](#)

### Administrar credenciales

Vea y elimine su información de inicio de sesión guardada para sitios web, redes y aplicaciones conectadas.

 **Credenciales web**

 **Credenciales de Windows**

[Copia de seguridad de credenciales](#) [Restaurar credenciales](#)

#### Credenciales de Windows

[Agregar una credencial de Windows](#)

192.168.6.100	Modificado: 28/09/2020	⌵
DANIEL-PC	Modificado: 25/10/2019	⌵

#### Credenciales basadas en certificados

[Agregar una credencial basada en certificado](#)

No hay certificados.

#### Credenciales genéricas

[Agregar una credencial genérica](#)

GÉANTLink/urn:UUID:19e76b30-8a55-4b52-8473-c74f2...	Fecha de modificación: 25/09/2019	⌵
MongoDB Compass/Connections/040e2d08-52fe-405...	Fecha de modificación: 20/02/2021	⌵
MongoDB Compass/Connections/0c6da3e9-e89b-4c7...	Fecha de modificación: 20/11/2020	⌵
MongoDB Compass/Connections/197a511a-8740-465f...	Fecha de modificación: 11/01/2021	⌵
MongoDB Compass/Connections/228d936d-3fa8-4dd...	Fecha de modificación: 18/04/2021	⌵
MongoDB Compass/Connections/2e7612f4-4ca6-49f7...	Fecha de modificación: 21/02/2021	⌵
MongoDB Compass/Connections/37ddbdbbe-1c87-41...	Fecha de modificación: 09/02/2021	⌵
MongoDB Compass/Connections/3ace4559-3207-4c1...	Fecha de modificación: 10/02/2021	⌵
MongoDB Compass/Connections/4066f5ce-4d7e-4a3...	Fecha de modificación: 20/02/2021	⌵
MongoDB Compass/Connections/4d3d05ce-da44-474...	Fecha de modificación: 11/01/2021	⌵

Vea también  
Cuentas de usuario

Pulsamos en editar e introducimos nuestras credenciales de GitHub:

git:https://github.com Fecha de modificación: 20/09/2021 ⌵

Dirección de red o Internet: git:https://github.com

Nombre de usuario: 91051062

Contraseña: .....

Persistencia: Equipo local

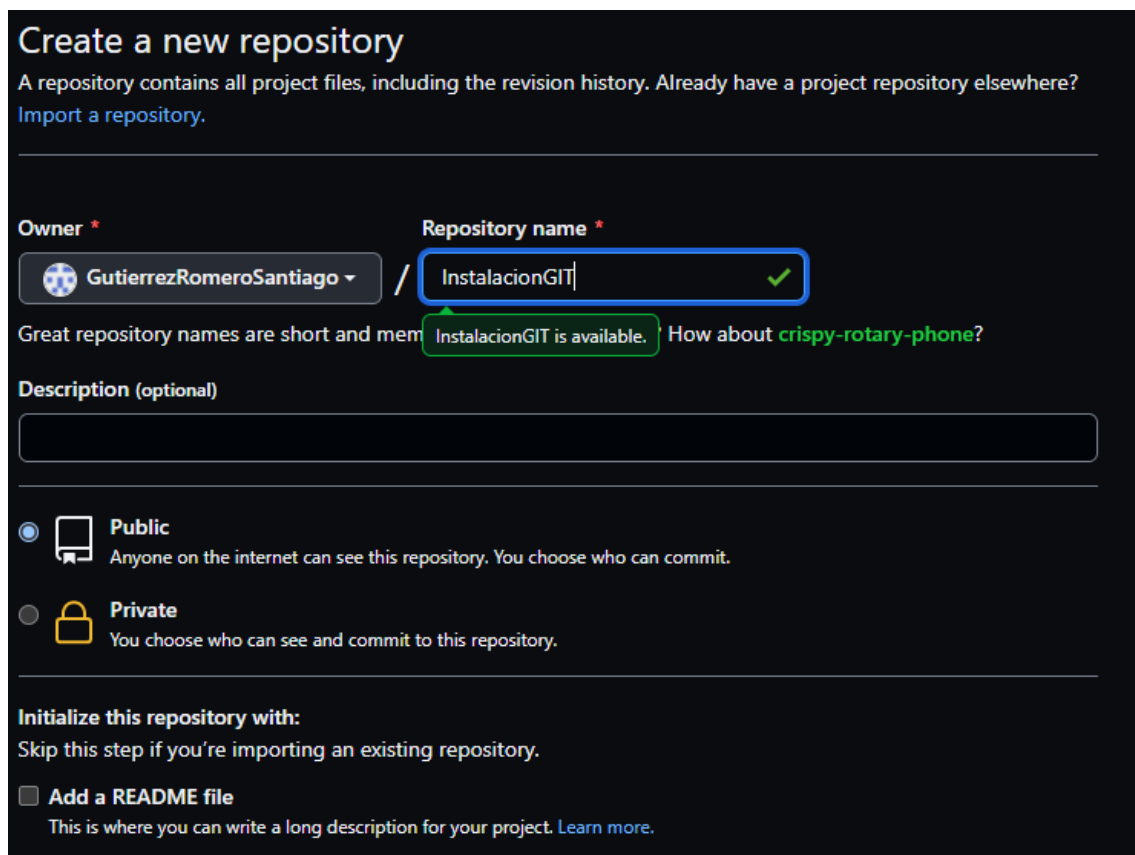
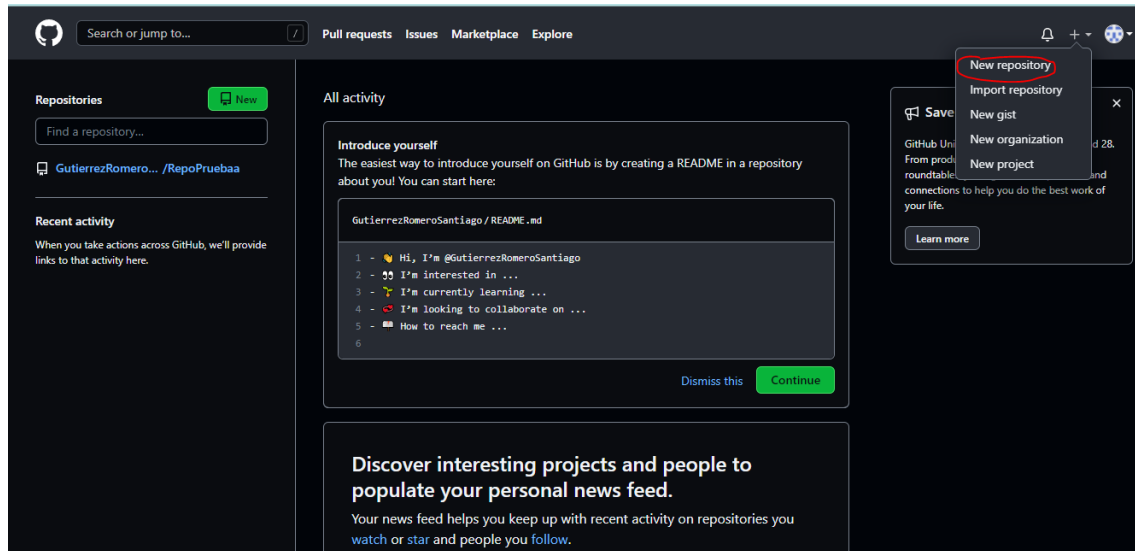
[Editar](#) [Quitar](#)

## Creación y utilización de los repositorios GitHub:

Ahora veremos como crear un repositorio en github y subirlo utilizando git:

<https://github.com/>

En primer lugar creamos un repositorio:



E introducimos los siguientes comandos desde Visual Studio code:

```
git init
```

```
git commit -m "first commit"
```


```
git branch -M main
```

```
git remote add origin https://github.com/GutierrezRomeroSantiago/InstalacionGIT.git
```

```
git push -u origin main
```

## Uso de los principales comandos GIT:

En este apartado realizaremos una revisión de los comandos que se suelen utilizar para subir contenido a un repositorio de GitHub, para ello vamos a crear un repositorio de prueba:

 Prueba\_02

22/09/2021 9:04

Carpeta de archivos

### GIT INIT:

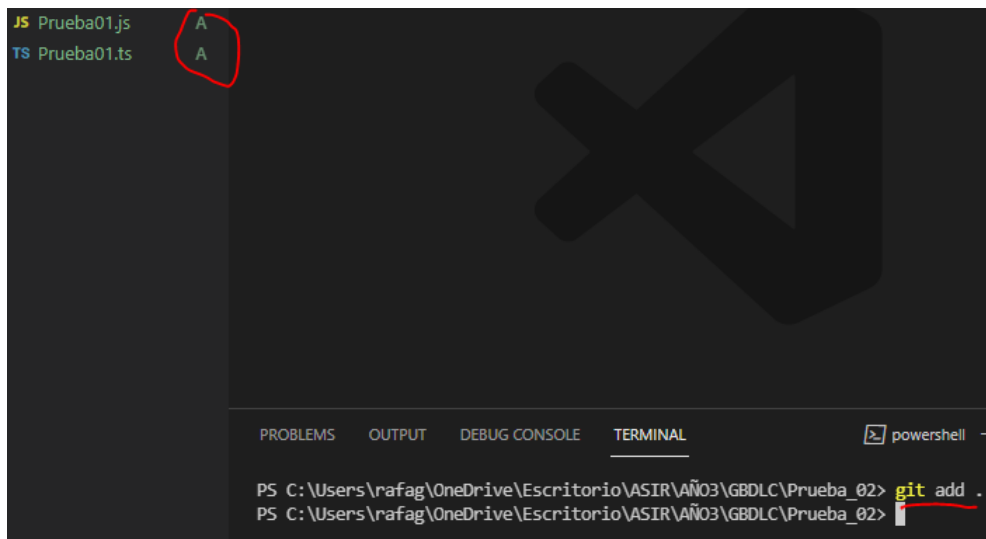
Una vez hemos creado un repositorio y queremos añadir contenido empezaremos por ejecutar el comando git init el cual supone el inicio del proceso:

```
PS C:\Users\rafag\OneDrive\Escritorio\ASIR\AÑO3\GBDLC\Prueba_02> git init
Initialized empty Git repository in C:/Users/rafag/OneDrive/Escritorio/ASIR/AÑO3/GBDLC/Prue
ba_02/.git/
PS C:\Users\rafag\OneDrive\Escritorio\ASIR\AÑO3\GBDLC\Prueba_02> |
```

Como podemos ver en la sentencia Git init crea un directorio oculto llamado .git

### GIT ADD:

Con el comando git add prepararemos los archivos que serán subidos al repositorio, en nuestro caso utilizaremos la sentencia "git add ." con el fin de añadir todos los archivos contenidos en la carpeta de prueba:



Si todo ha ido bien podremos observar que junto a el nombre de los archivos aparece una A la cual hace referencia a la palabra added. Aun así para realizar una comprobación más fiable utilizaremos el comando git status.

### GIT STATUS:

Con git status podremos comprobar si los archivos que queremos se han añadido correctamente:

```
PS C:\Users\rafag\OneDrive\Escritorio\ASIR\AÑO3\GBDLC\Prueba_02> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Prueba01.js
        new file:   Prueba01.ts
```

Como podemos observar los cambios que hemos realizado están listos para ser entregados.

### GIT COMMIT:

El siguiente paso será especificar que queremos hacer la entrega, para ello utilizaremos el siguiente comando git commit -m "MI PRIMERA ENTREGA DE PRUEBA", como podemos observar podemos añadir un comentario con el fin de contralar la versión del proyecto que se ha subido al repositorio:

```
PS C:\Users\rafag\OneDrive\Escritorio\ASIR\AÑO3\GBDLC\Prueba_02> git commit -m "MI PRIMERA ENTREGA DE PRUEBA"
[master (root-commit) 87f21e6] MI PRIMERA ENTREGA DE PRUEBA
 2 files changed, 2 insertions(+)
 create mode 100644 Prueba01.js
 create mode 100644 Prueba01.ts
PS C:\Users\rafag\OneDrive\Escritorio\ASIR\AÑO3\GBDLC\Prueba_02> █
```

Como podemos observar los archivos están listos para ser entregados.

### GIT BRANCH:

Con el uso de Git Branch pretendemos indicar cual es la rama principal:

```
PS C:\Users\rafag\OneDrive\Escritorio\ASIR\AÑO3\GBDLC\Prueba_02> git branch -M main
PS C:\Users\rafag\OneDrive\Escritorio\ASIR\AÑO3\GBDLC\Prueba_02> █
```

El siguiente paso es de gran importancia ya que debemos indicar la dirección del repositorio:

### GIT REMOTE ADD ORIGIN:

En este paso indicaremos la dirección URL en la que se encuentra nuestro repositorio, en nuestro caso ejecutaremos la siguiente sentencia, `git remote add origin`

<https://github.com/GutierrezRomeroSantiago/PruebasGIT.git> :

```
PS C:\Users\rafag\OneDrive\Escritorio\ASIR\AÑO3\GBDLC\Prueba_02> git remote add origin http
s://github.com/GutierrezRomeroSantiago/PruebasGIT.git
PS C:\Users\rafag\OneDrive\Escritorio\ASIR\AÑO3\GBDLC\Prueba_02> |
```

### GIT PUSH:

El último paso será lanzar el contenido hacia github para ello utilizaremos la siguiente sentencia, `git push -u origin main`:

```
PS C:\Users\rafag\OneDrive\Escritorio\ASIR\AÑO3\GBDLC\Prueba_02> git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 330 bytes | 82.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/GutierrezRomeroSantiago/PruebasGIT.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
PS C:\Users\rafag\OneDrive\Escritorio\ASIR\AÑO3\GBDLC\Prueba_02> |
```

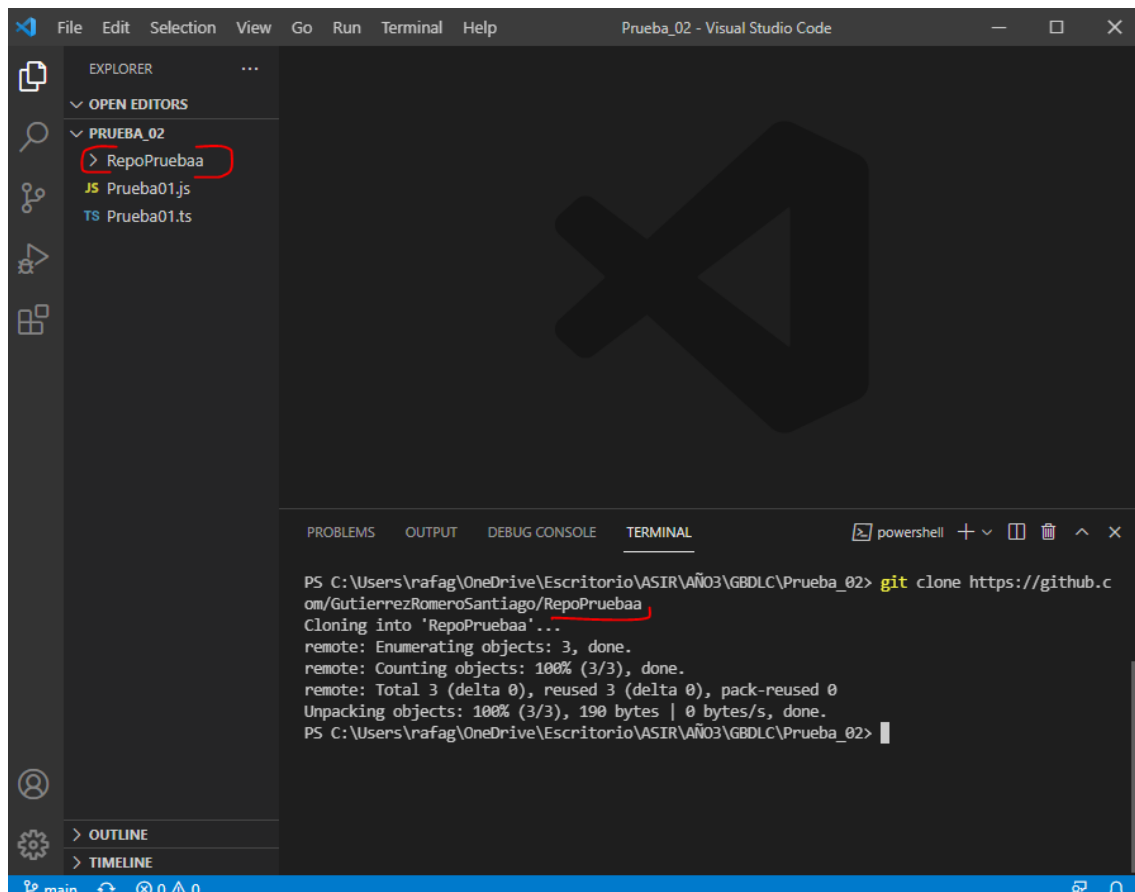
Como podemos observar el contenido se ha subido al repositorio de forma exitosa:



### GIT CLONE:

Para clonar repositorios tendríamos que ejecutar la siguiente sentencia, `git clone`  
REPOSITORIO A COPIAR :





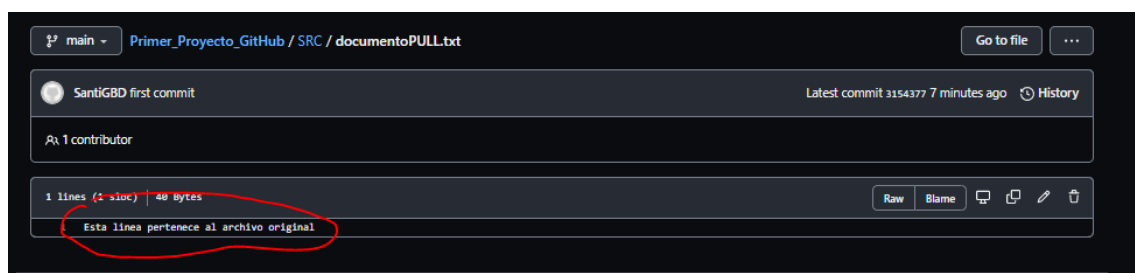
The screenshot shows the Visual Studio Code interface. In the Explorer panel on the left, the 'PRUEBA\_02' folder is expanded, showing a subfolder 'RepoPruebaa' which contains 'Prueba01.js' and 'Prueba01.ts'. The 'RepoPruebaa' folder is highlighted with a red circle. The Terminal panel at the bottom shows the execution of the command: `git clone https://github.com/GutierrezRomeroSantiago/RepoPruebaa`. The output of the command is visible, showing the cloning process and the creation of the local repository. The 'RepoPruebaa' folder name in the terminal output is also highlighted with a red circle.

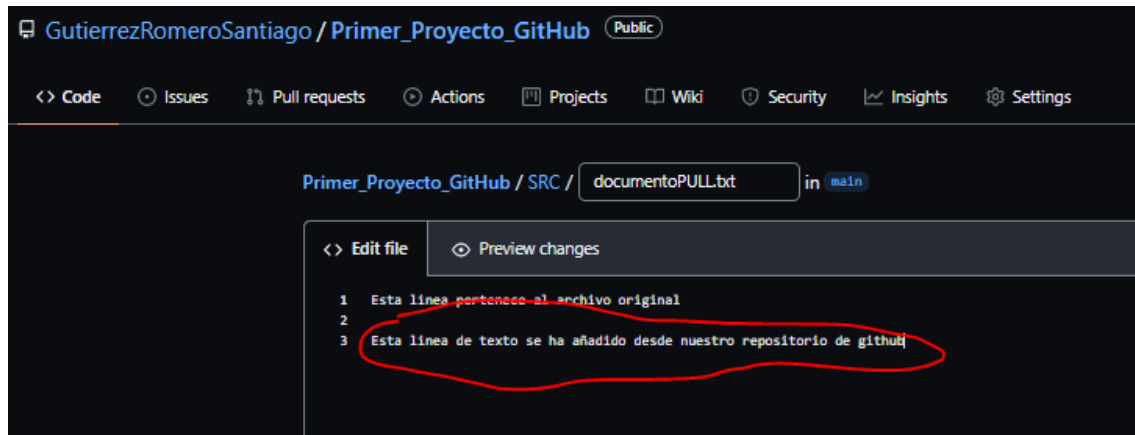
Como se puede observar en la imagen hemos clonado a nuestro repositorio local un repositorio de GitHub llamando RepoPruebaa.

### GIT PULL:

En último lugar veremos como hacer uso de git pull el cual nos permitirá modificar un archivo en nuestro github y traer dichos cambios a nuestro repositorio local.

Para poder demostrarlo hemos creado previamente un archivo txt cuyo contenido modificaremos desde github:

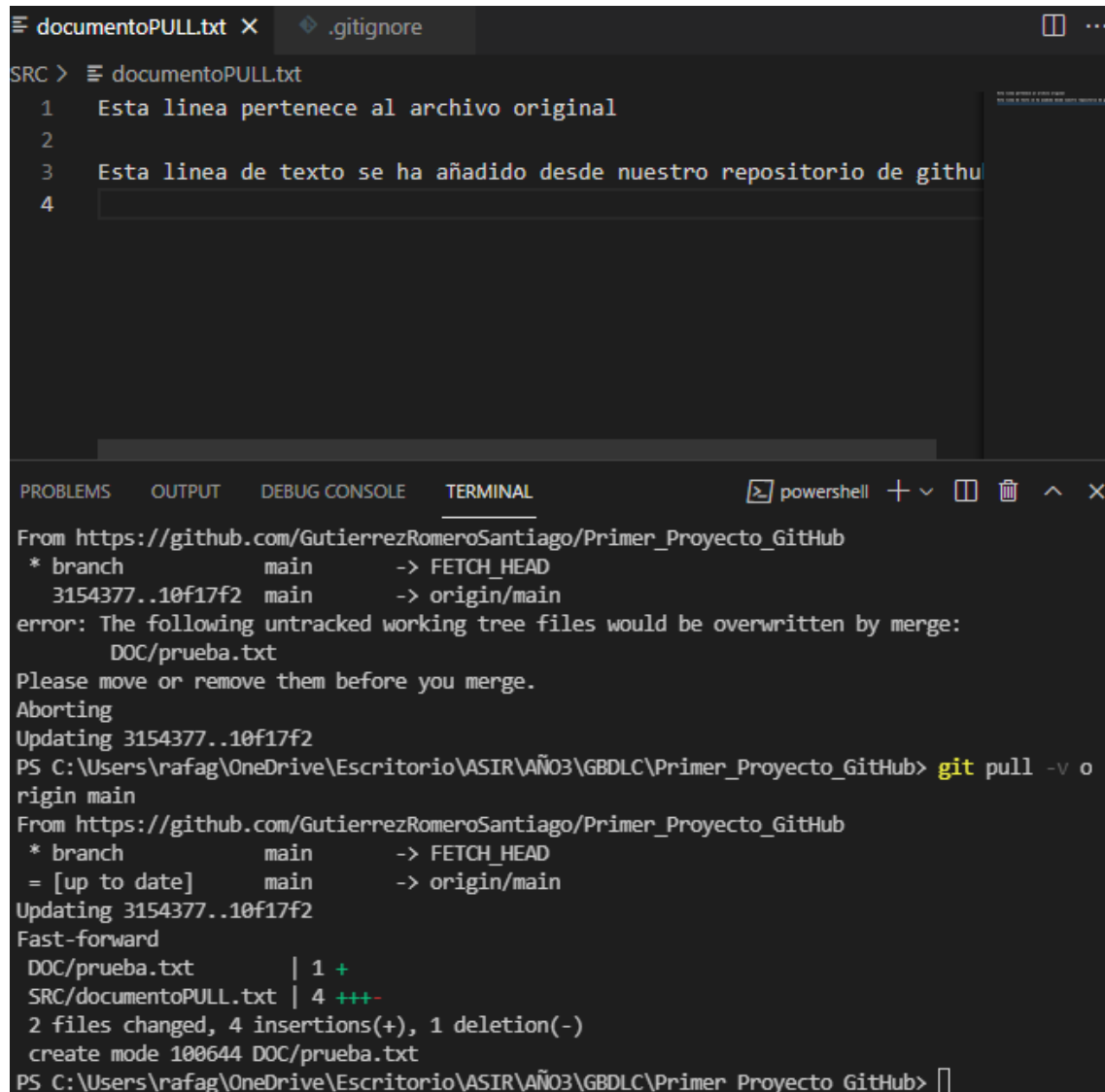




Como podemos ver hemos añadido una línea de texto la cual trataremos de traer a nuestro repositorio local mediante el comando pull, para ello ejecutaremos en el terminal la siguiente sentencia, git pull -v origin main:

```
PS C:\Users\rafag\OneDrive\Escritorio\ASIR\AÑO3\GBDLC\Primer_Proyecto_GitHub> git pull -v origin main
From https://github.com/GutierrezRomeroSantiago/Primer_Proyecto_GitHub
* branch          main          -> FETCH_HEAD
= [up to date]    main          -> origin/main
Updating 3154377..10f17f2
Fast-forward
 DOC/prueba.txt    | 1 +
 SRC/documentoPULL.txt | 4 +++-
 2 files changed, 4 insertions(+), 1 deletion(-)
 create mode 100644 DOC/prueba.txt
PS C:\Users\rafag\OneDrive\Escritorio\ASIR\AÑO3\GBDLC\Primer_Proyecto_GitHub>
```

Como vemos la terminal indica que ha habido cambios en el documento los cuales se han hecho efectivos en el repositorio local:



The image shows a Visual Studio Code editor window. The top part displays a text file named 'documentoPULL.txt' with the following content:

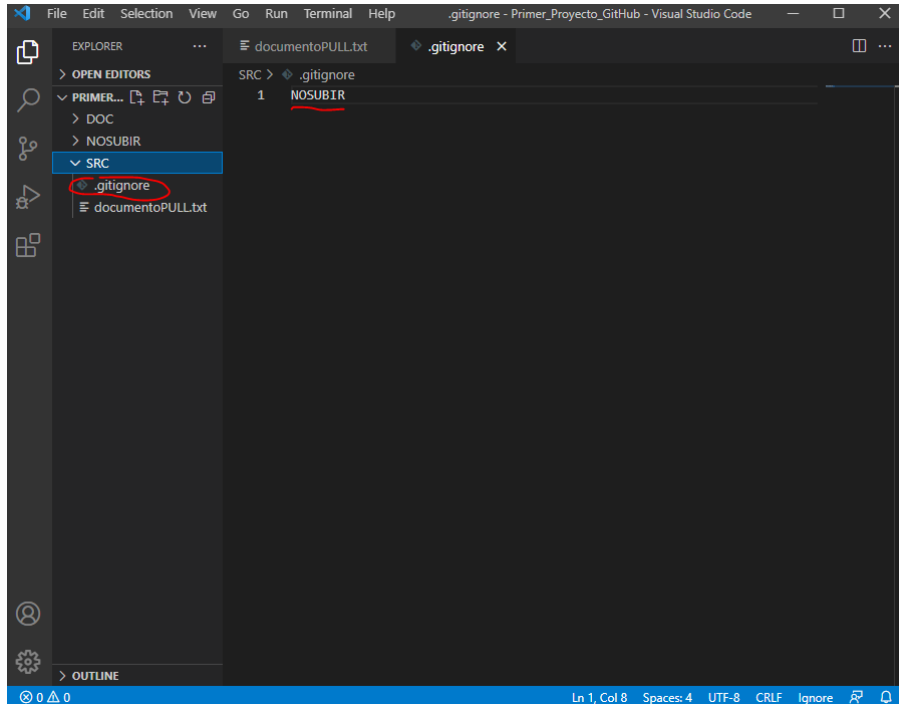
```
1 Esta linea pertenece al archivo original
2
3 Esta linea de texto se ha añadido desde nuestro repositorio de github
4
```

The bottom part of the window shows the 'TERMINAL' tab with the following output:

```
From https://github.com/GutierrezRomeroSantiago/Primer_Proyecto_GitHub
* branch      main      -> FETCH_HEAD
3154377..10f17f2 main      -> origin/main
error: The following untracked working tree files would be overwritten by merge:
      DOC/prueba.txt
Please move or remove them before you merge.
Aborting
Updating 3154377..10f17f2
PS C:\Users\rafag\OneDrive\Escritorio\ASIR\AÑ03\GBDLC\Primer_Proyecto_GitHub> git pull -v o
rigin main
From https://github.com/GutierrezRomeroSantiago/Primer_Proyecto_GitHub
* branch      main      -> FETCH_HEAD
= [up to date]   main      -> origin/main
Updating 3154377..10f17f2
Fast-forward
 DOC/prueba.txt      | 1 +
 SRC/documentoPULL.txt | 4 +++-
2 files changed, 4 insertions(+), 1 deletion(-)
create mode 100644 DOC/prueba.txt
PS C:\Users\rafag\OneDrive\Escritorio\ASIR\AÑ03\GBDLC\Primer_Proyecto_GitHub>
```

## UTILIZACIÓN DE .GITIGNORE:

El uso del archivo .gitignore es simple, en este archivo simplemente tendremos que especificar el nombre exacto del contenido que no queremos subir al repositorio, en nuestro caso hemos especificado el nombre de una carpeta:



A continuación procederemos a comprobar que la carpeta llamada NOSUBIR no se ha subido a github:



Como podemos comprobar todo el contenido se ha subido correctamente excepto la carpeta NOSUBIR ya que esta venía especificada en .gitignore