

Git: Instalación, comandos, pruebas

Antonio Ávila Delgado

2º ASIR A

ÍNDICE

- Instalación de Git.....pág 3.
- Creación de cuenta en GitHub.....pág 12.
- Comandos de Git.....pág 17.
- Pruebas PUSH, .GITIGNORE, PULL, README.md, DOWNLOAD, CLONE.....pág 22.



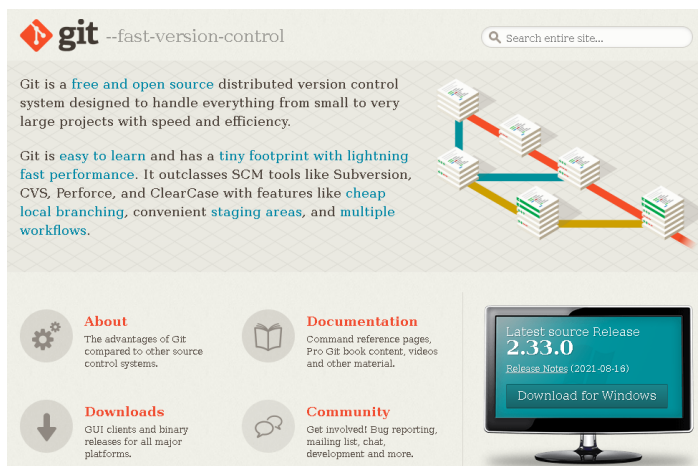
INTRODUCCIÓN

En este documento vamos a explicar, paso a paso, lo siguiente:

- Instalación de Git.
- Creación de cuenta en GitHub.
- Comandos de Git.
- pruebas con PUSH, .GITIGNORE, PULL, README.md, DOWNLOAD, CLONE

Instalación de Git


En primer lugar nos dirigimos a: <https://git-scm.com/> y nos aparece la siguiente pantalla:



Elegimos la pantalla para bajar la versión más actual que es la 2.33.0



Bajamos el archivo de instalación para Windows

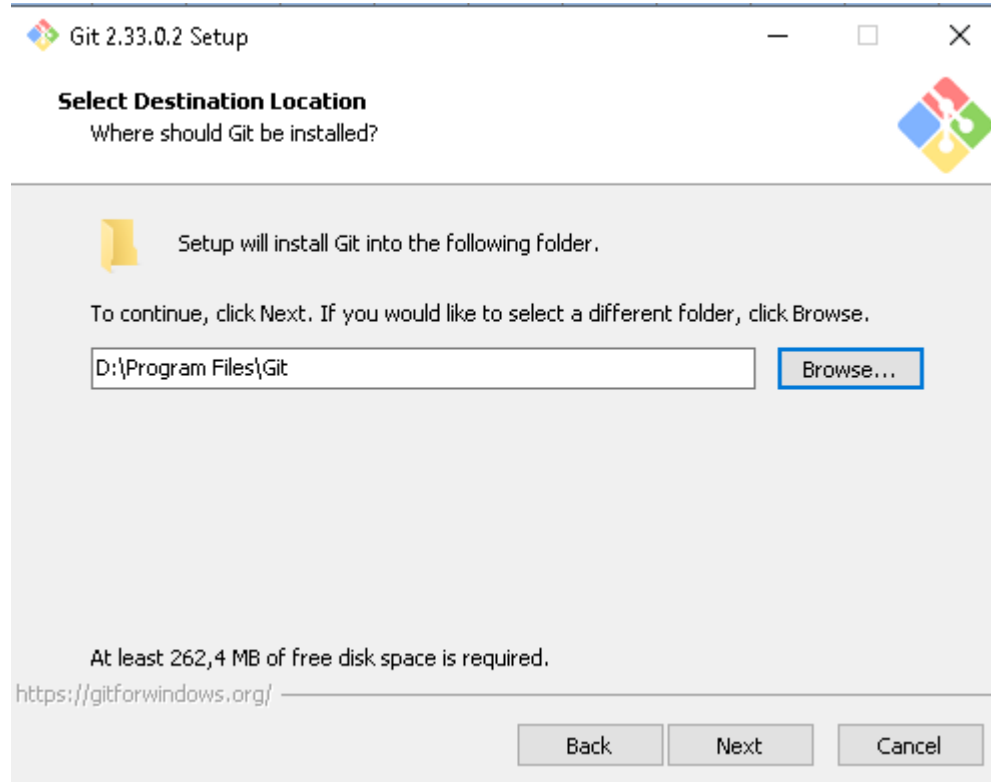
Nombre	Fecha de m
 Git-2.33.0.2-64-bit.exe	20/09/2021

Lo ejecutamos y aceptamos todas las opciones por defecto:

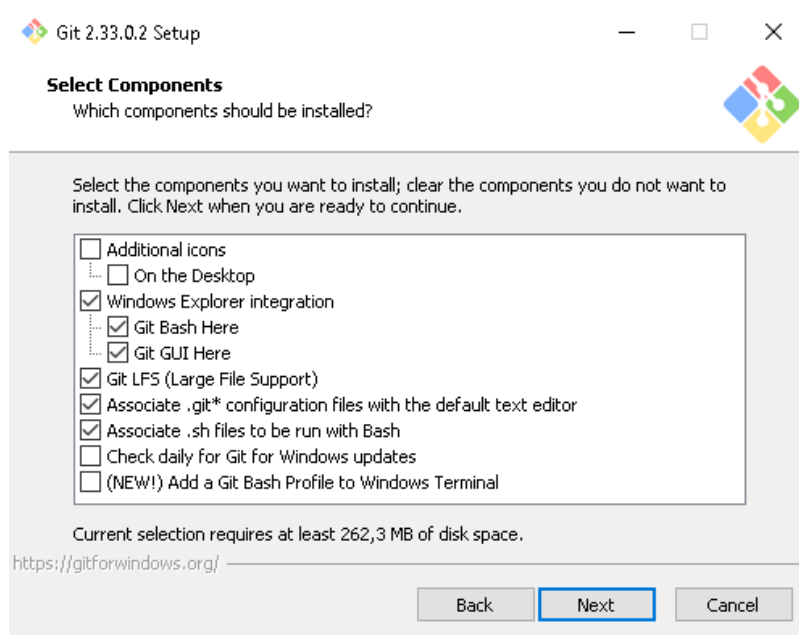
En esta pantalla, le damos a NEXT



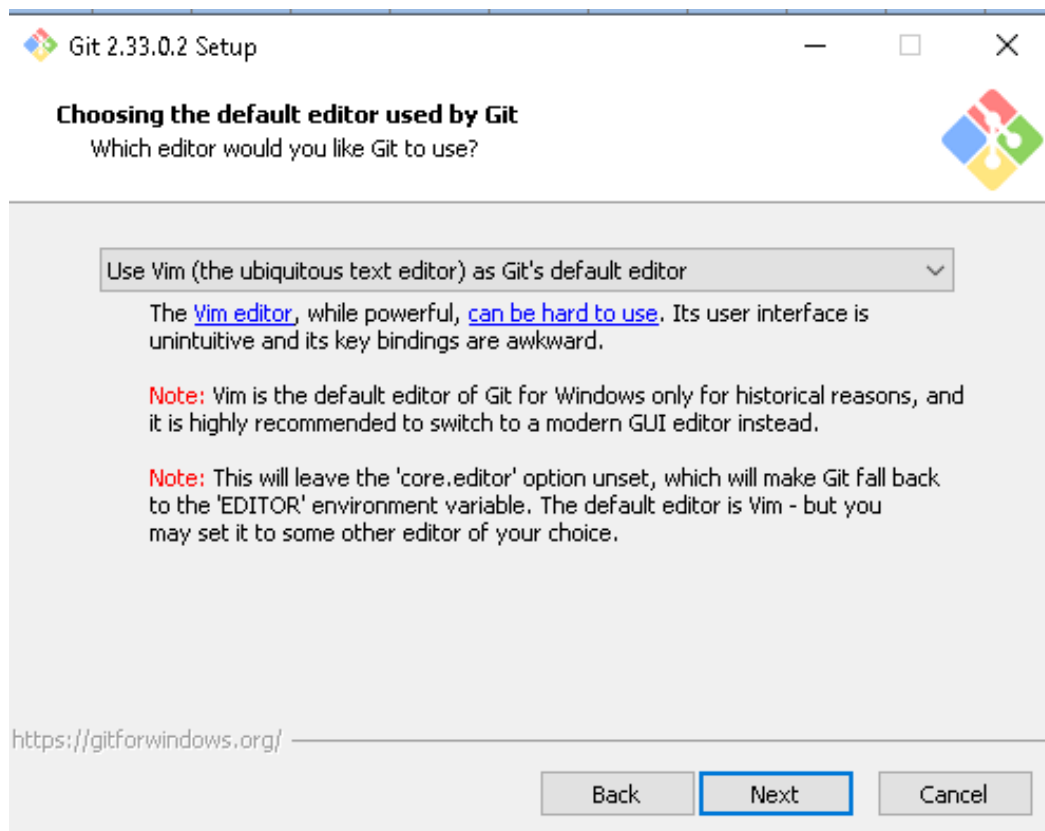
Elegimos la ruta de destino de la instalación y pulsamos NEXT



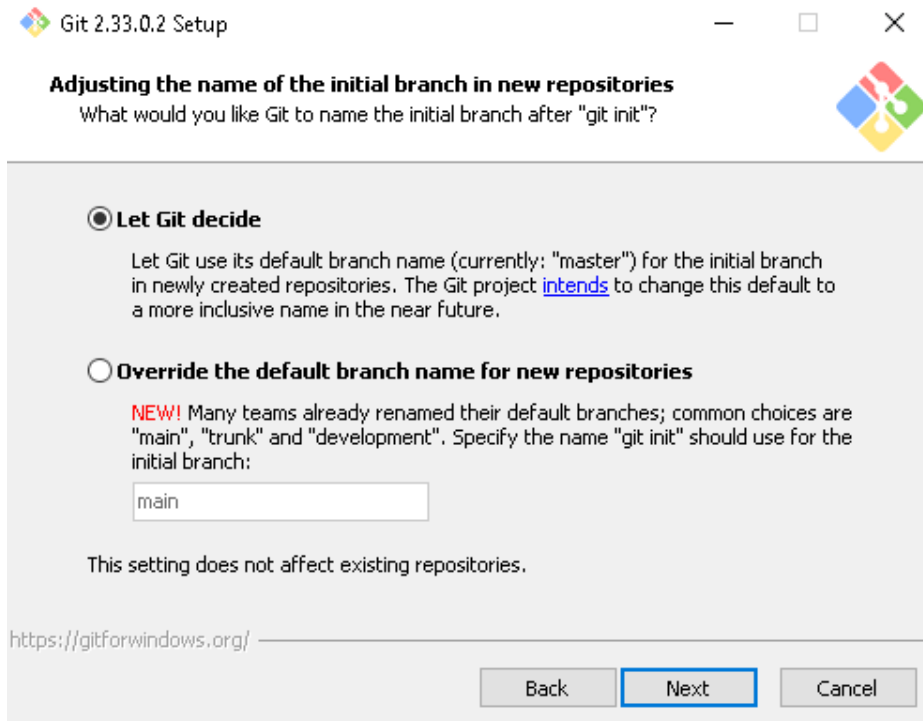
Elegimos los componentes que salen por defecto y le damos a NEXT



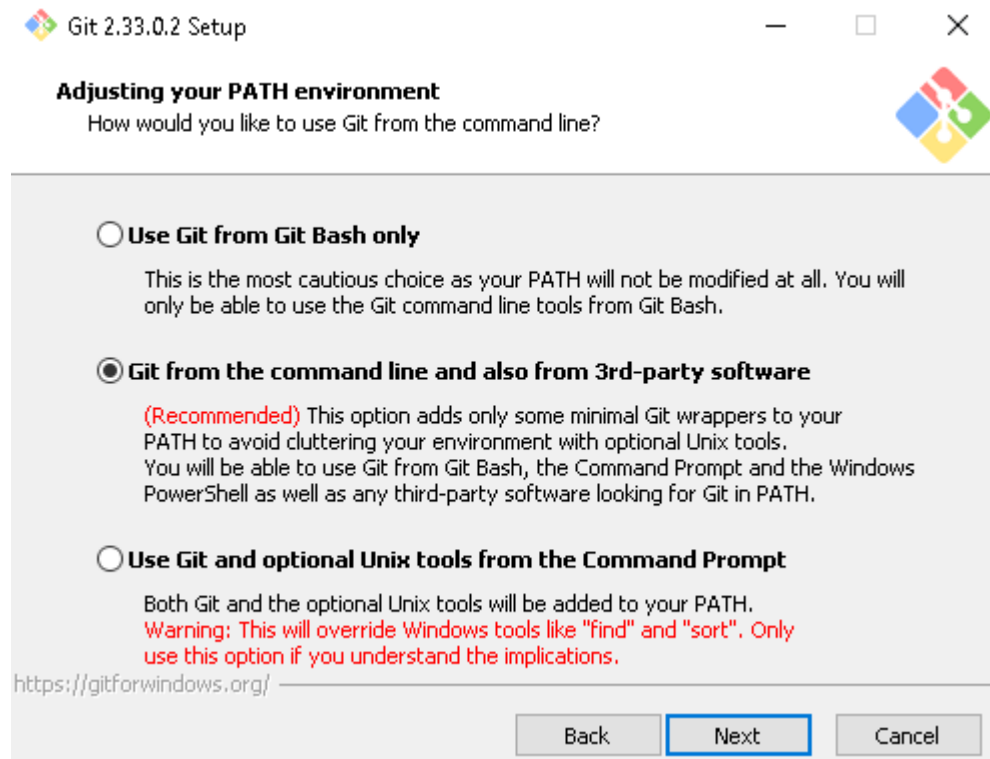
En la siguiente pantalla, damos a NEXT:



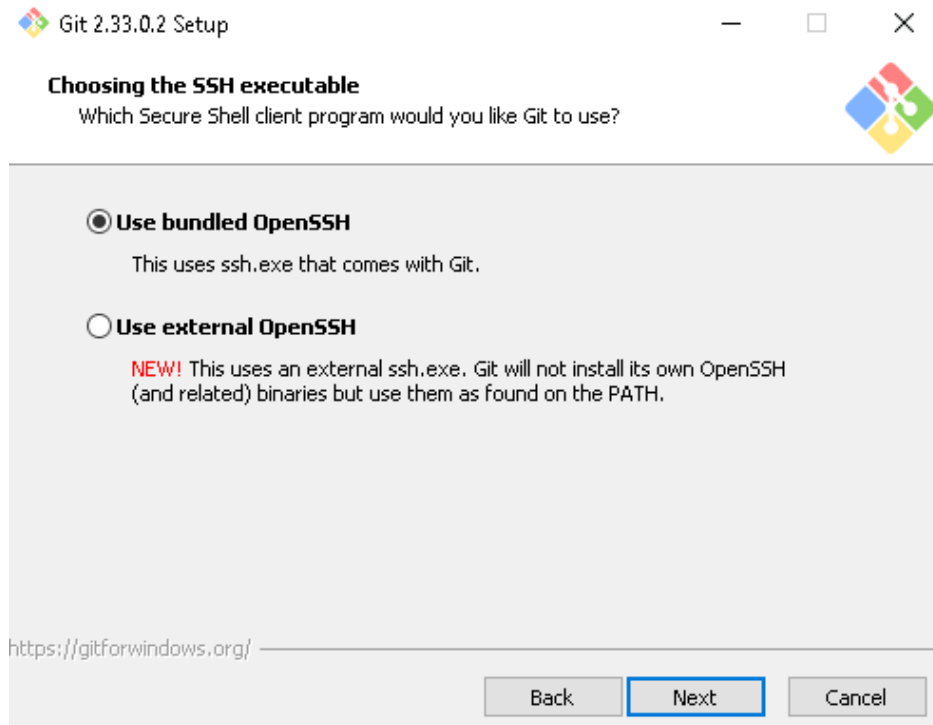
En la siguiente pantalla, damos a NEXT:



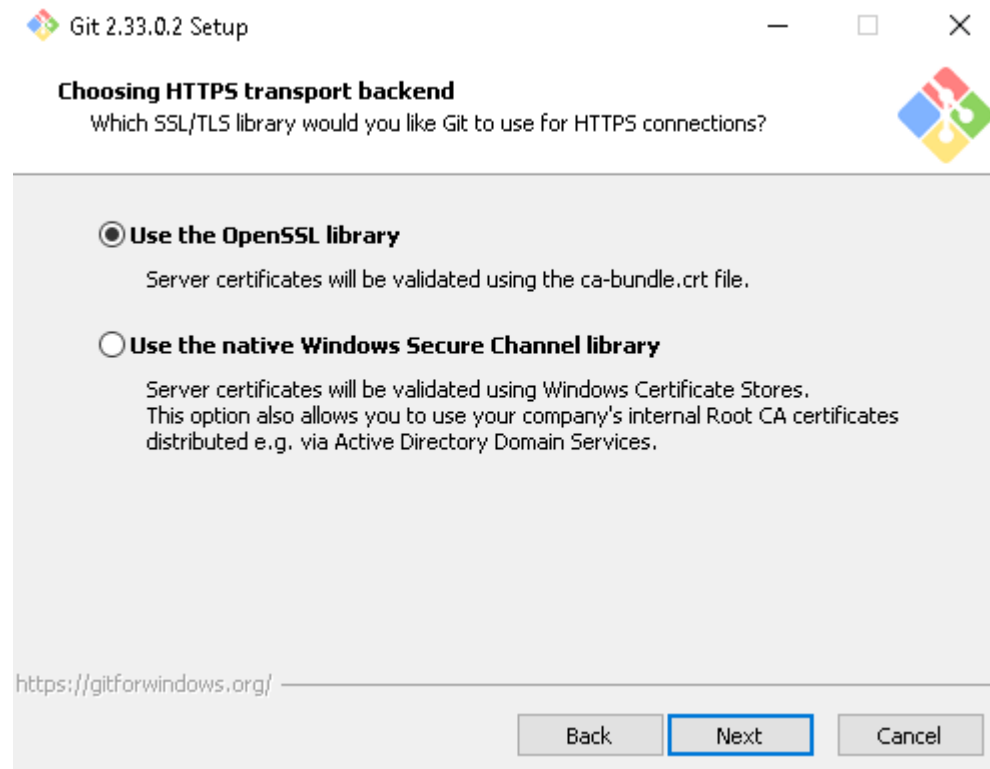
En la siguiente pantalla, damos a NEXT:



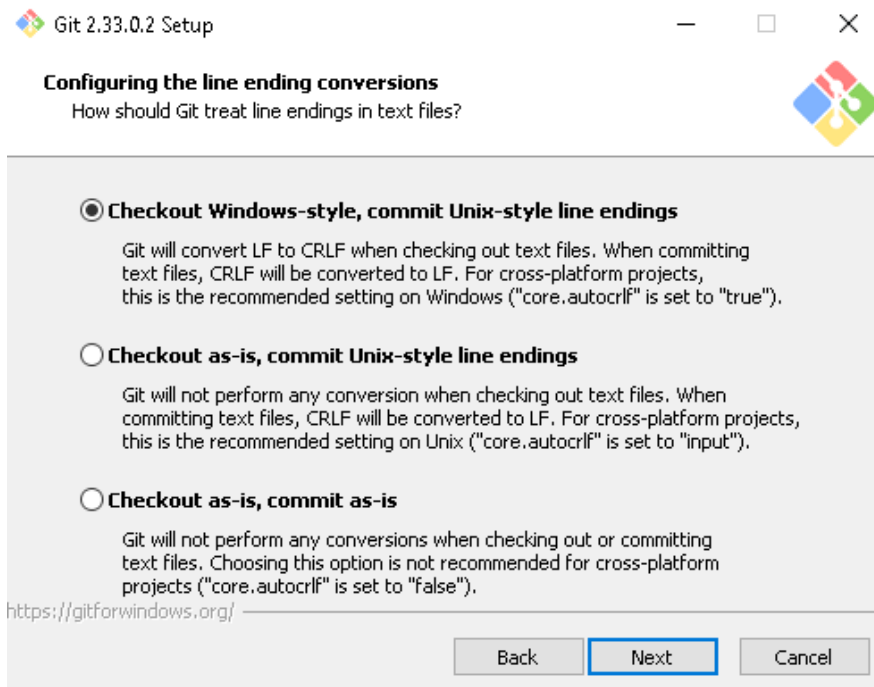
En la siguiente pantalla, damos a NEXT:



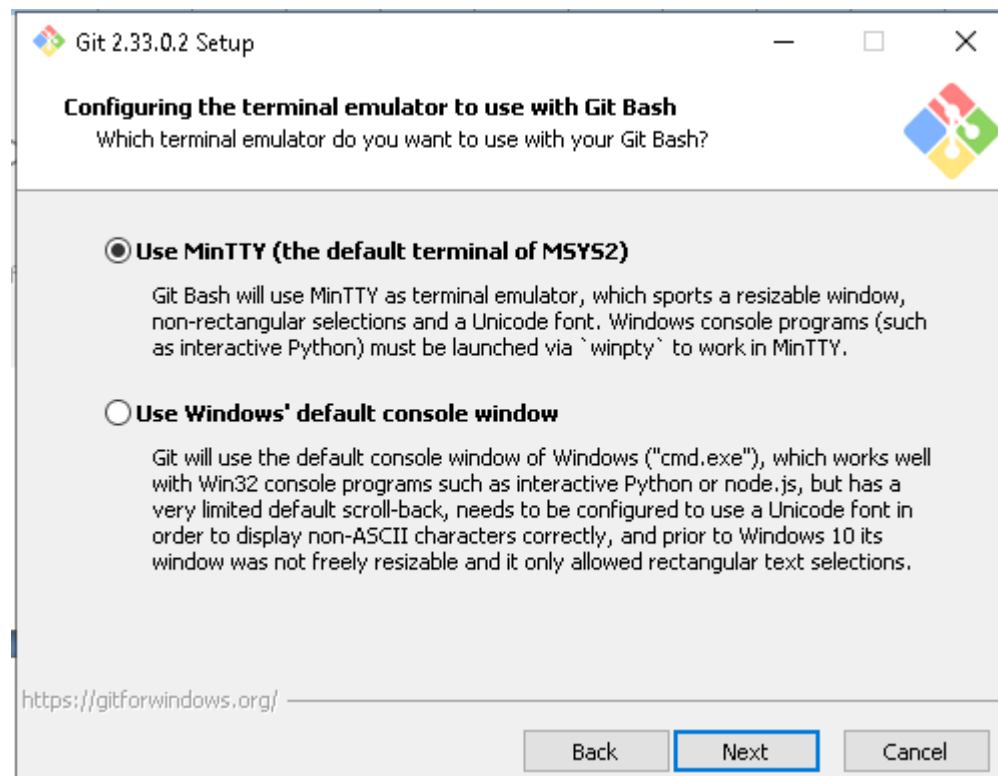
En la siguiente pantalla, damos a NEXT:



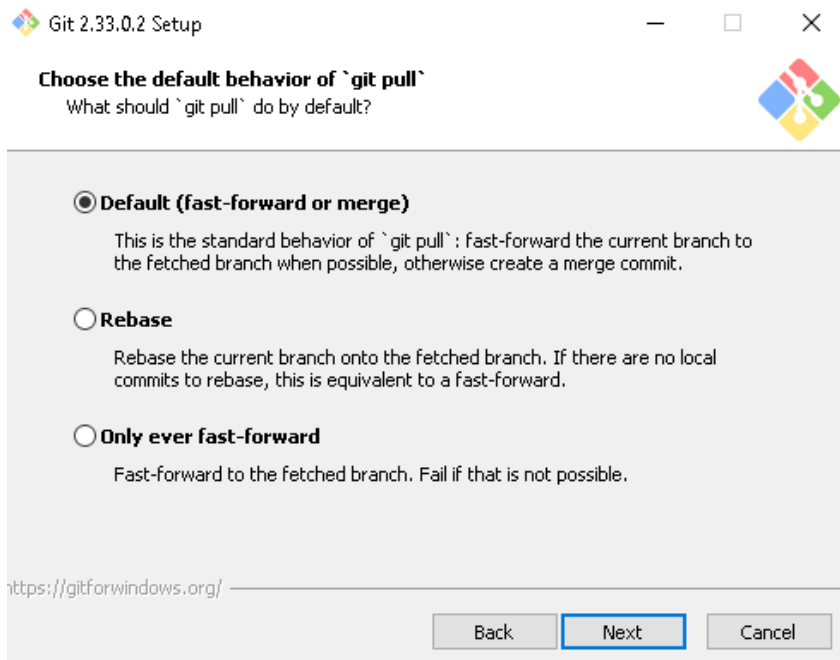
En la siguiente pantalla, damos a NEXT:



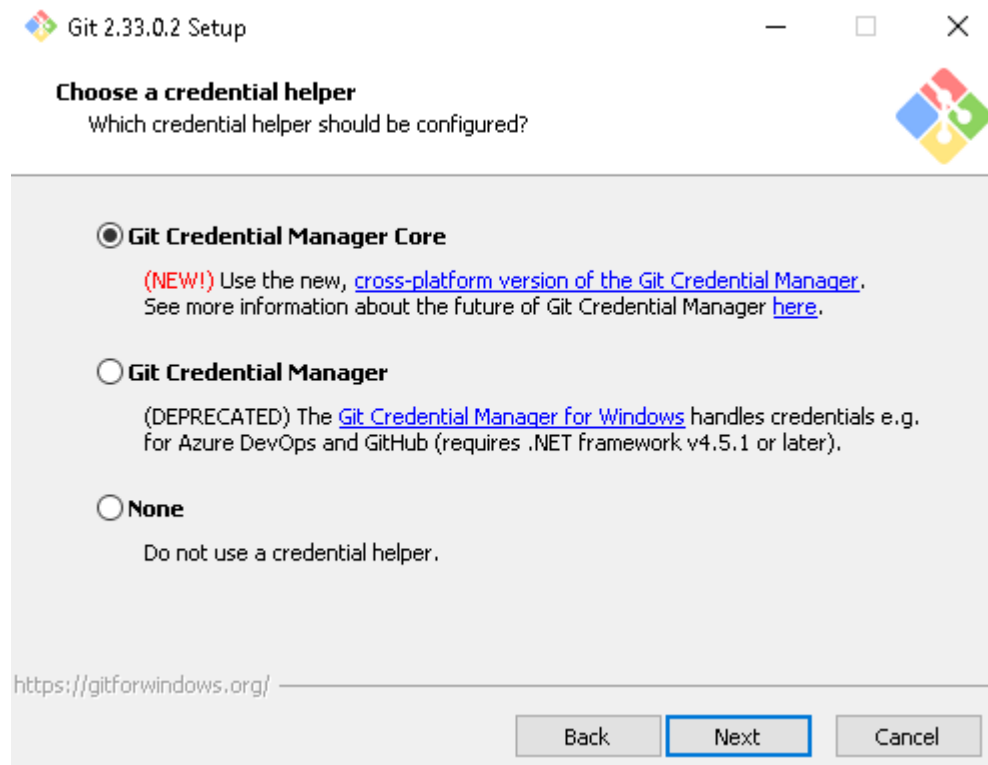
En la siguiente pantalla, damos a NEXT:



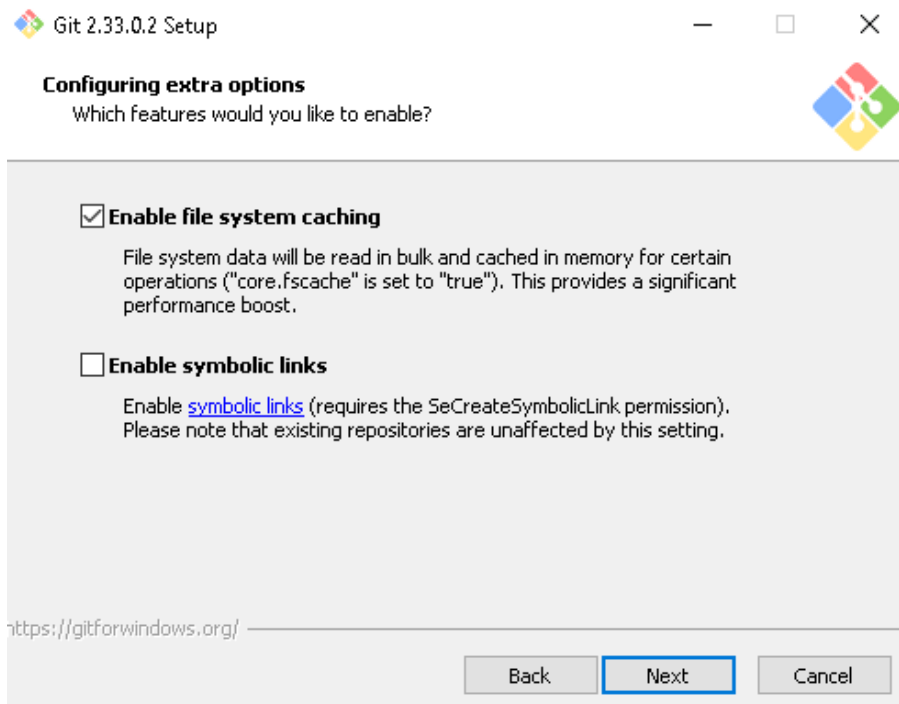
En la siguiente pantalla, damos a NEXT:



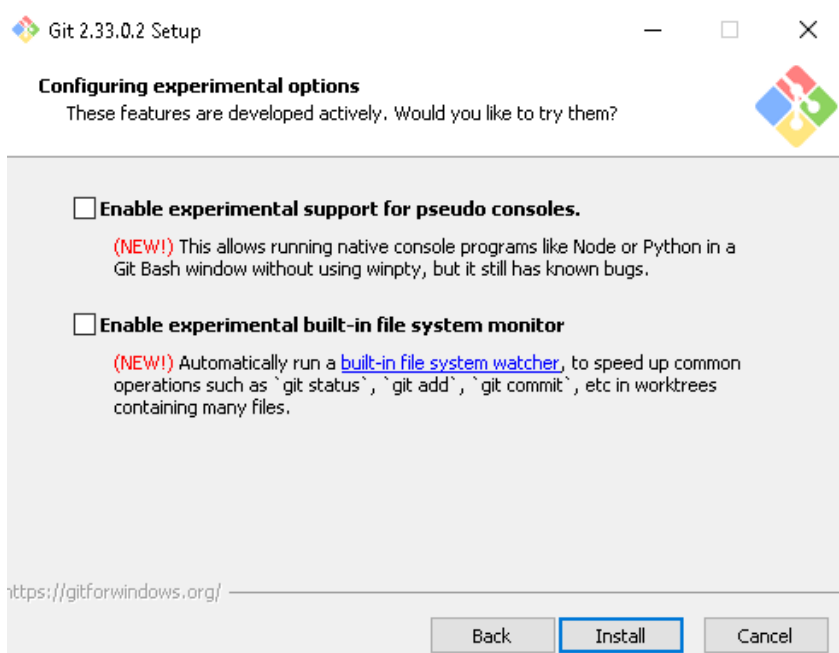
En la siguiente pantalla, damos a NEXT:



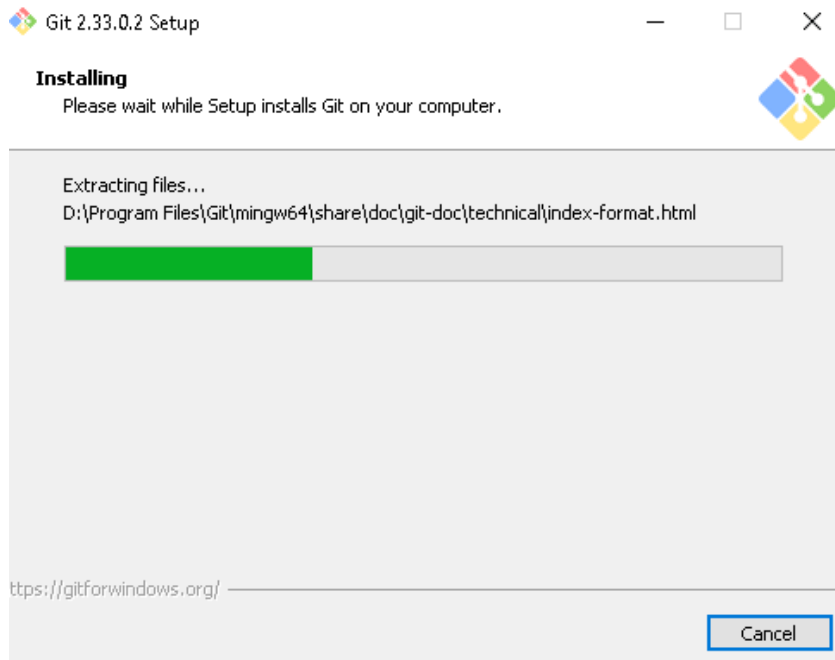
En la siguiente pantalla, damos a NEXT:



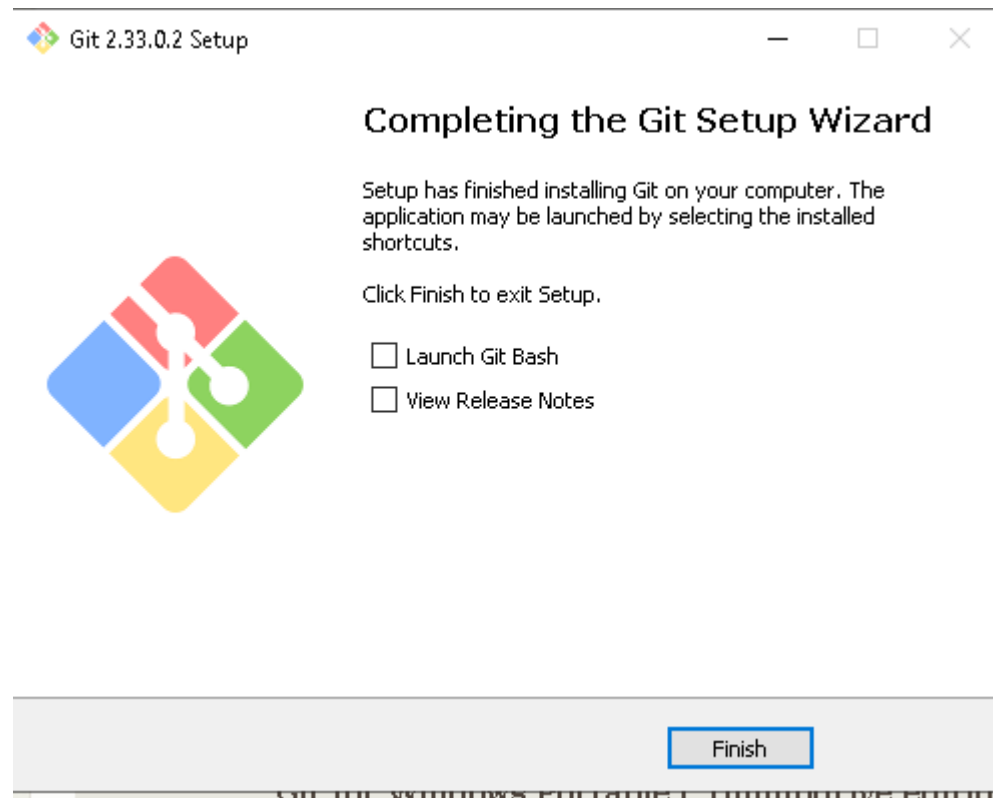
En la siguiente pantalla, damos a NEXT:



En la siguiente pantalla, damos a NEXT:

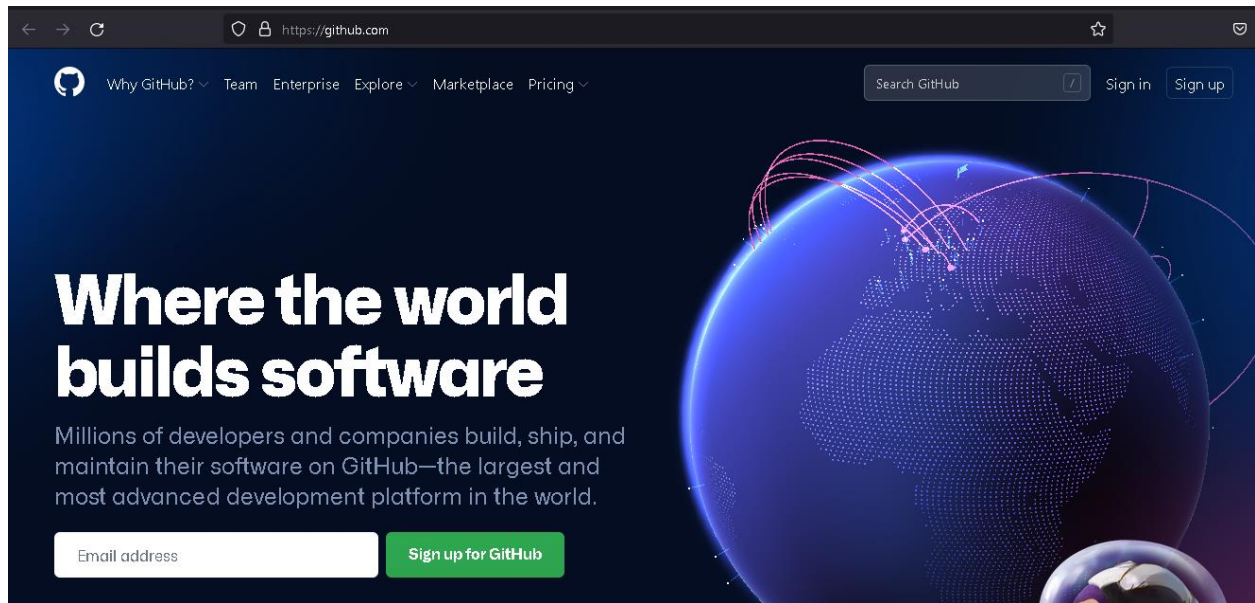


La instalación termina y le damos a FINISH



Creación de cuenta en GitHub

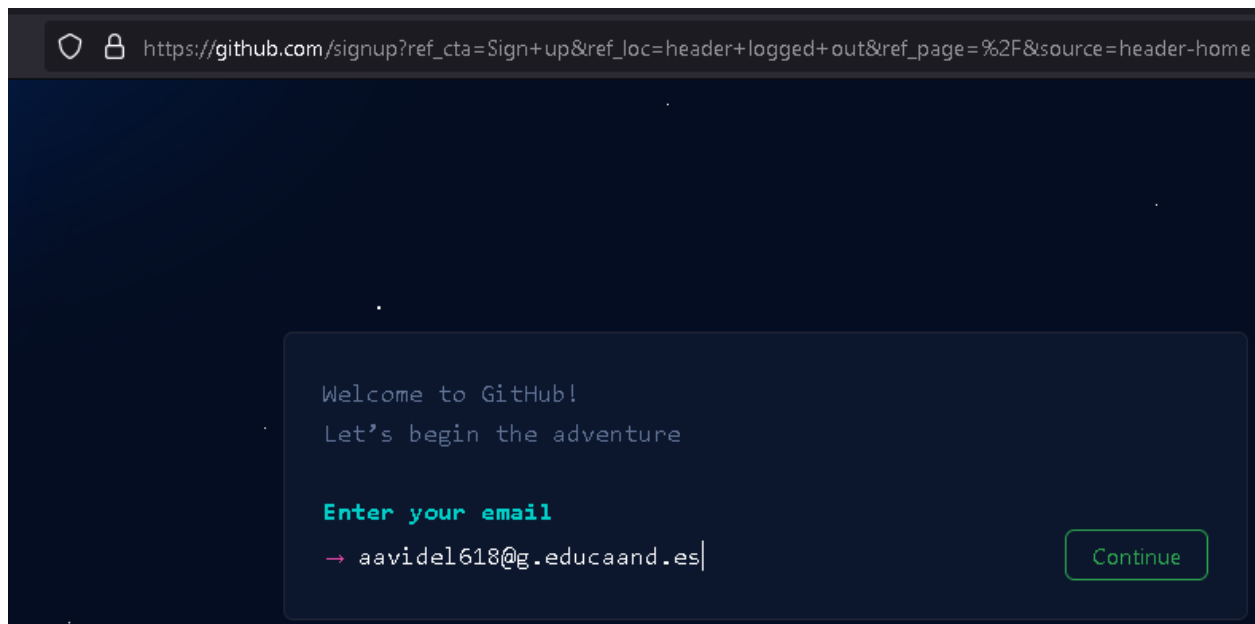
En primer lugar nos dirigimos a: <https://github.com/> y nos aparece la siguiente pantalla:



En la pantalla, como es la primera vez (no tenemos cuenta), le damos a **“sign up”**

Nota: una vez creada la cuenta, y en posteriores sesiones, se accede via **“sign in”**

Tras pulsar sobre **“sign up”** nos aparece lo siguiente:



Ponemos nuestra dirección de correo electrónico y le damos a “continue” y a aparece esto:



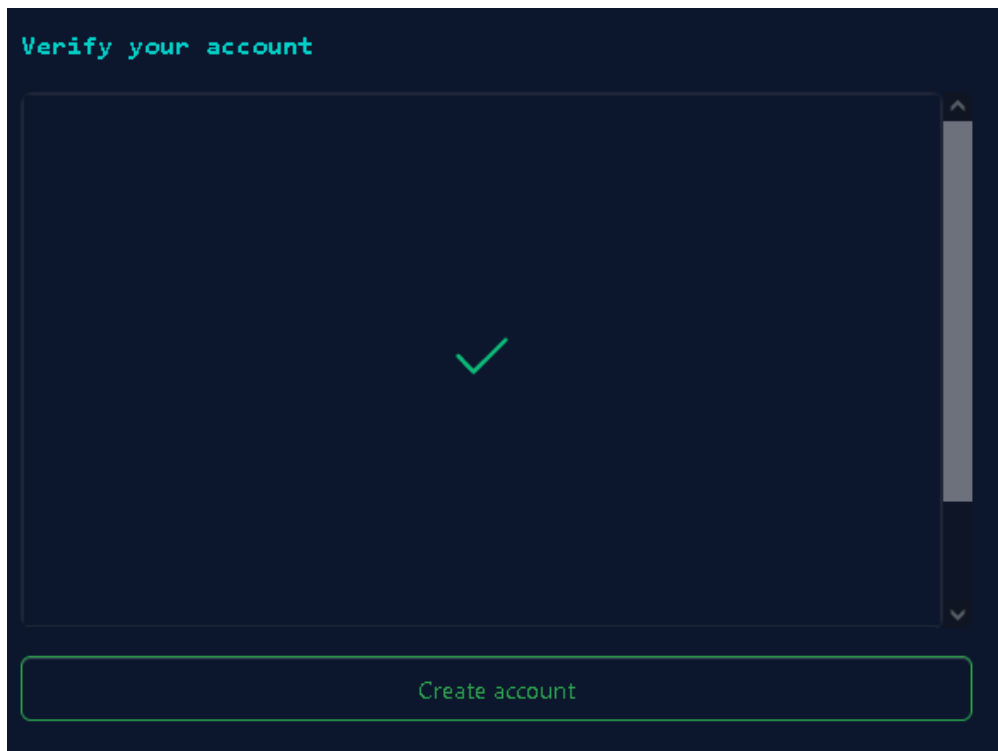
Creamos una contraseña robusta y le damos a “continue” y a aparece esto:



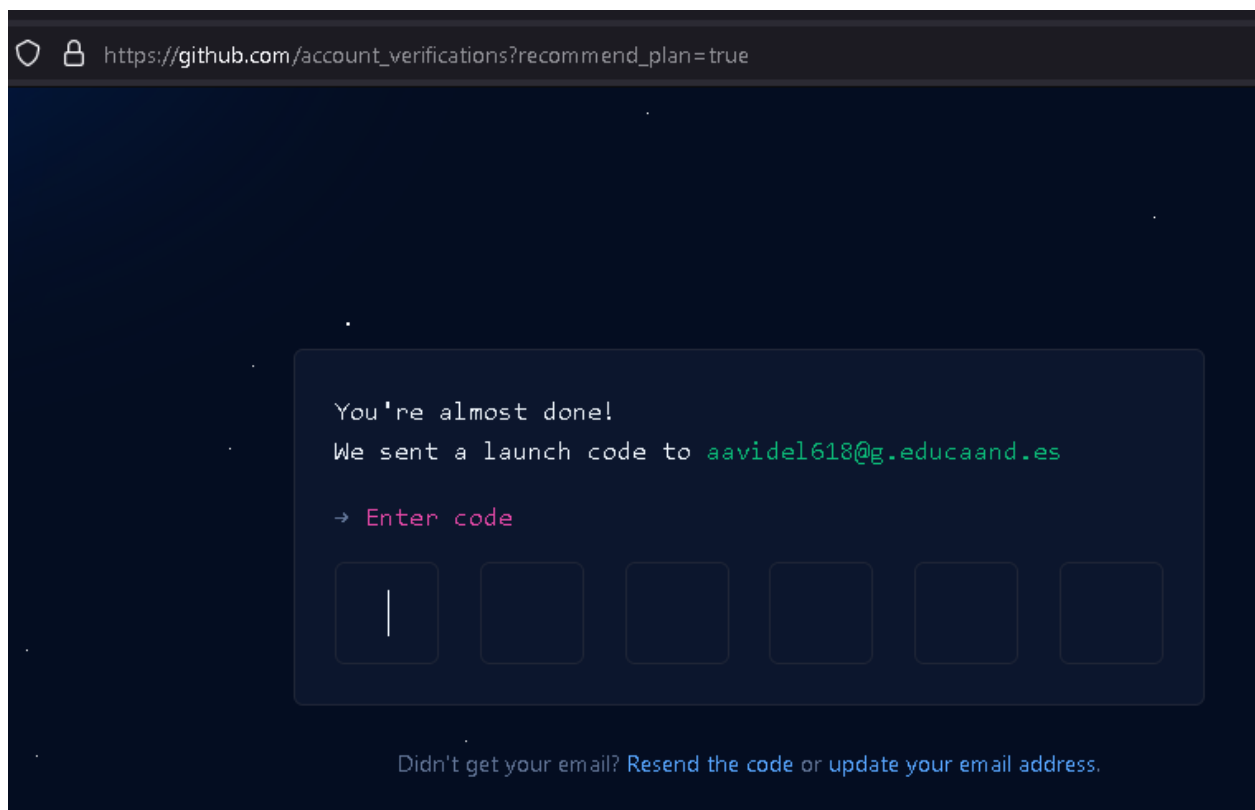
Creamos un usuario y le damos a “continue” y a aparece esto:



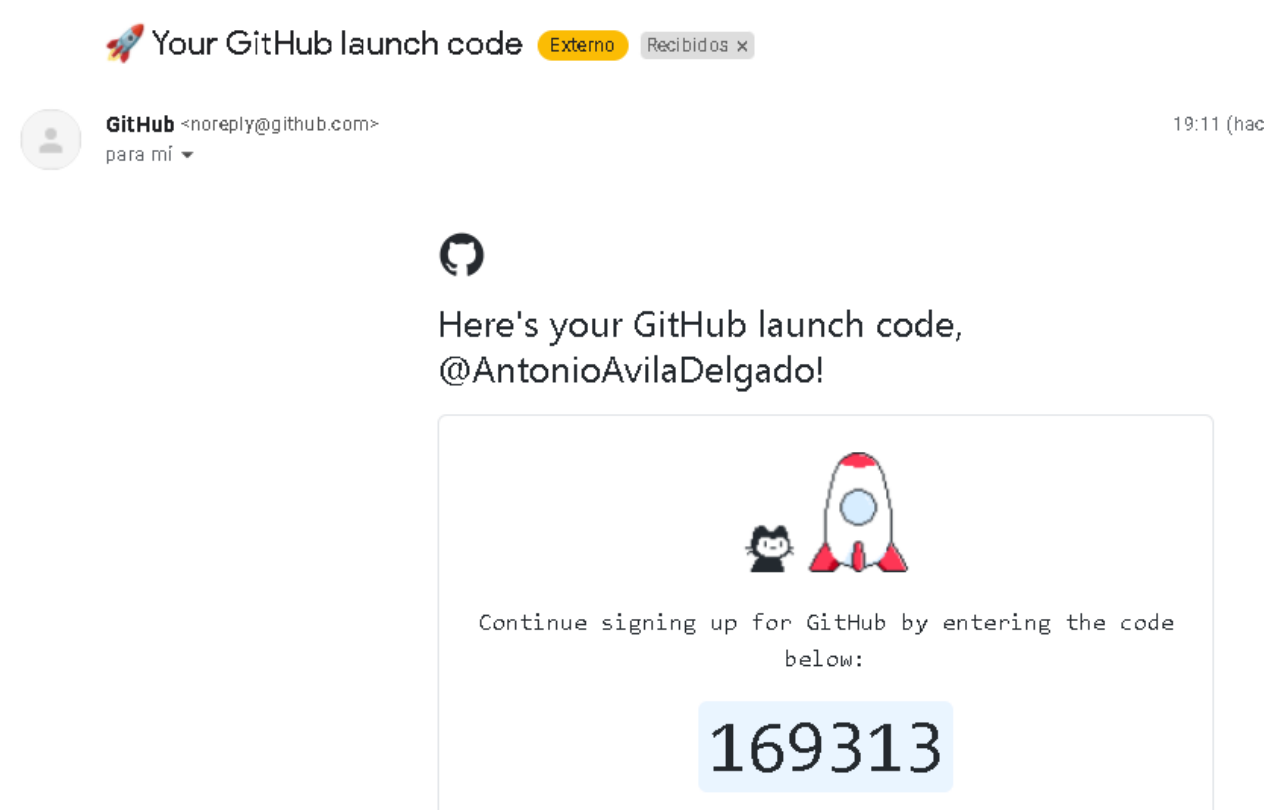
Demostramos que no somos un “bot”



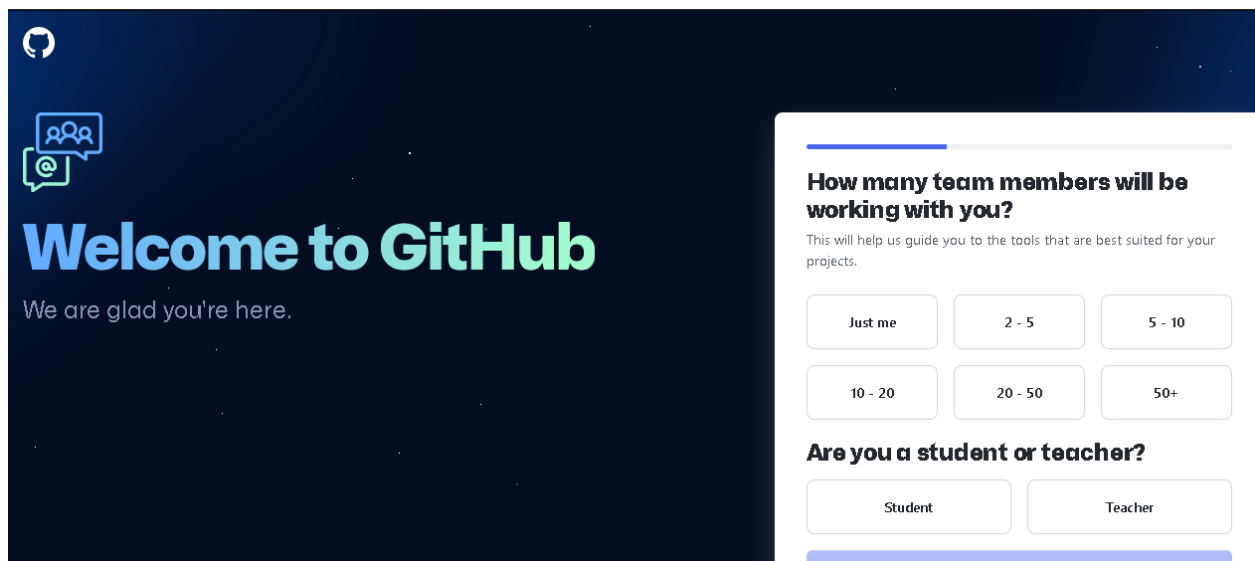
Y le damos a “create account” y sale esto:



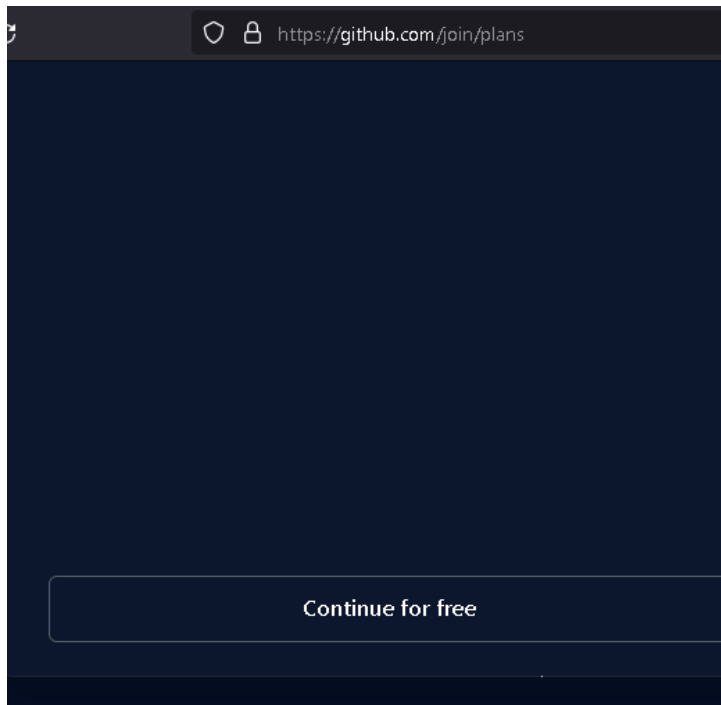
Este es el código recibido:



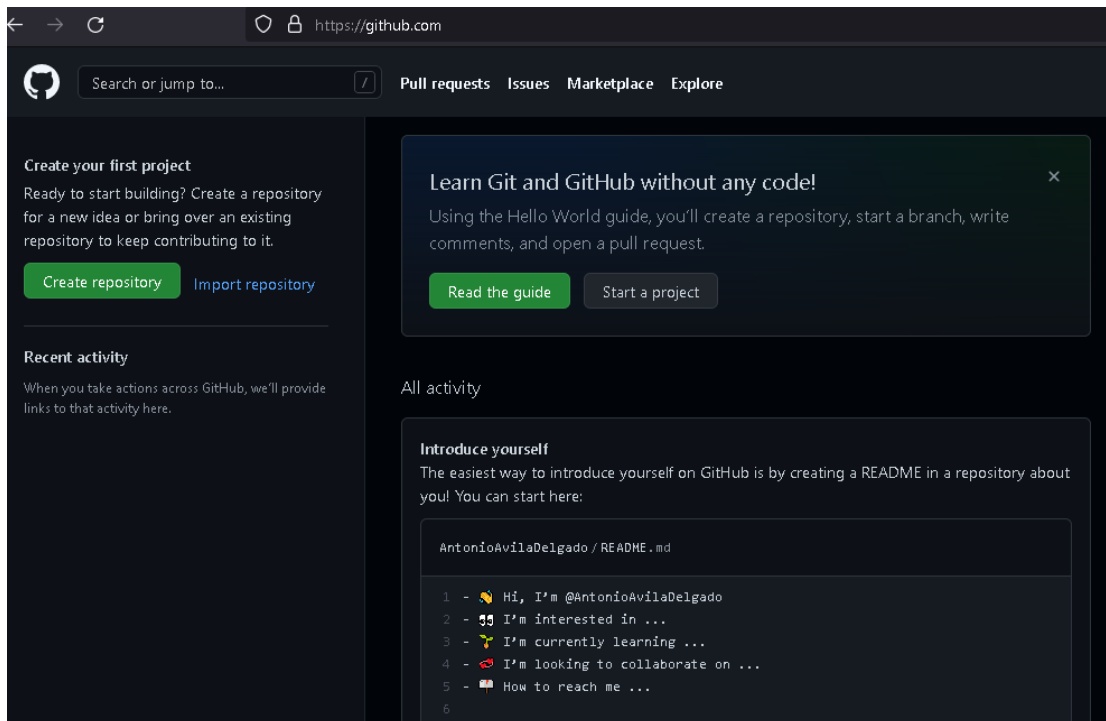
Lo ponemos y sale esto:



Respondemos a las preguntas finales y sale esto:



Nos aseguramos que le damos a “continue for free” y nos sale esto:



Lo cual demuestra que hemos creado la cuenta con éxito.

Comandos de Git

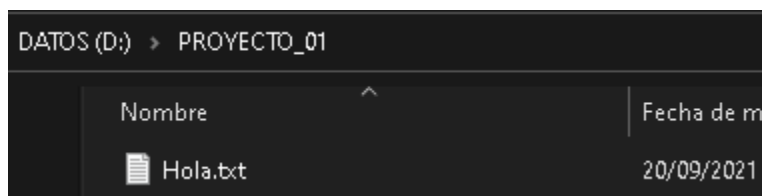
En primer lugar abrimos una sesión de powershell.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

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

PS C:\Users\Usuario>
```

Creamos una carpeta de proyecto, y dentro de ella un archivo

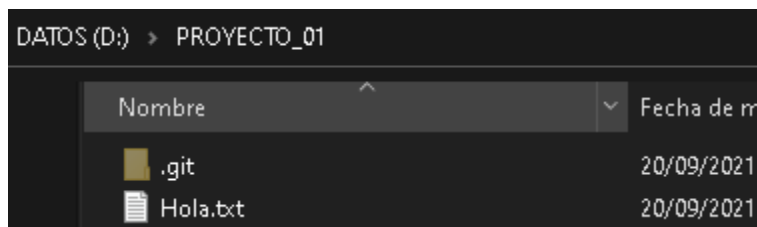


Me posiciono en powershell en el directorio,

```
PS D:\proyecto_01>
```

Y lanzamos el comando git init, que inicializa y crea la carpeta .git

```
PS D:\proyecto_01> git init
Initialized empty Git repository in D:/PROYECTO_01/.git/
```



Usamos el comando git status para comprobar que efectivamente nuestro archivo hola.txt existe y que está listo para usar el comando git add

```
PS D:\proyecto_01> git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  Hola.txt
```

Usamos pues el comando git add .

```
PS D:\proyecto_01> git add .
PS D:\proyecto_01>
```

Y hacemos otro git status para comprobar que ha salido bien, visualmente se comprueba que nuestro archivo hola.txt ya no está en rojo, sino en verde, y está preparado para el “commit”

```
PS D:\proyecto_01> git status
On branch master

No commits yet

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

Lanzamos el comando git commit, pero como es la primera vez quiere saber detalles como correo electrónico y usuario.

```
PS D:\proyecto_01> git commit -m "first commit"
Author identity unknown

*** Please tell me who you are.

Run

    git config --global user.email "you@example.com"
    git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.
```

Así que le proporcionamos dichos detalles

```
PS D:\proyecto_01> git config --global user.email antavidel@gmail.com
PS D:\proyecto_01> git config --global user.name AvilaDelgadoAntonio
PS D:\proyecto_01>
```

Y proseguimos con el commit

```
PS D:\proyecto_01> git commit -m "first commit"
[master (root-commit) c320916] first commit
1 file changed, 1 insertion(+)
create mode 100644 Hola.txt
```

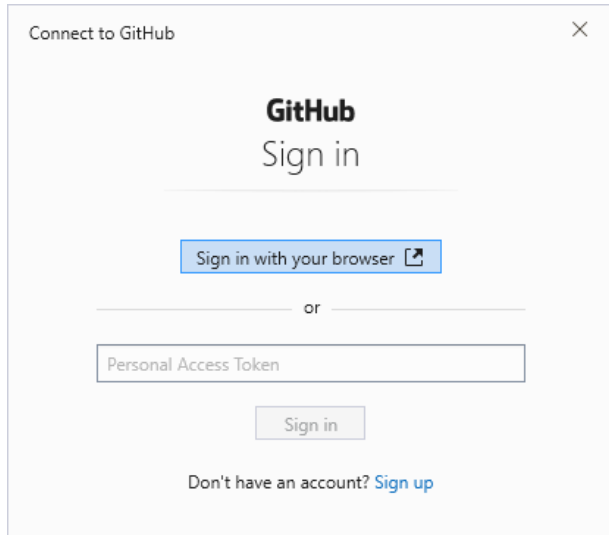
El siguiente paso es asociar la carpeta local con el repositorio de GitHub

```
PS D:\proyecto_01> git remote add origin https://github.com/AvilaDelgadoAntonio/PROYECTO_01.git
PS D:\proyecto_01>
```

Y ahora lo lanzamos para subirlo gracias al comando git push

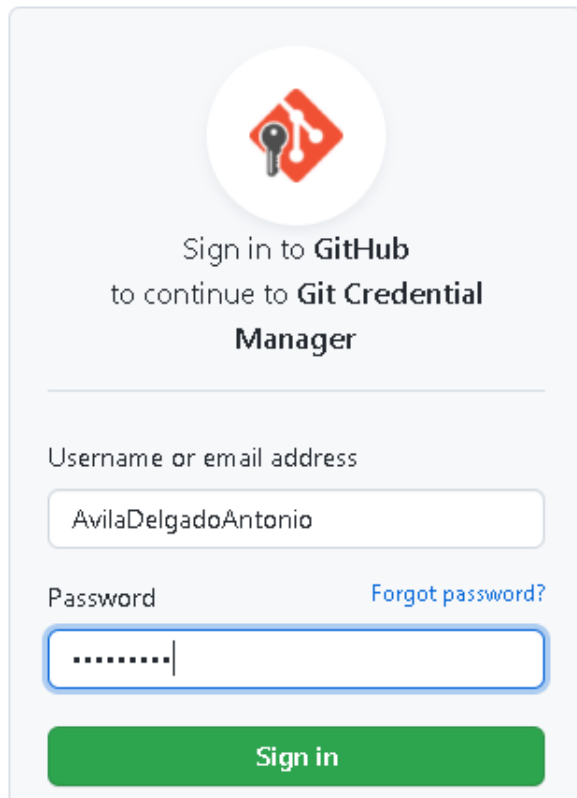
```
PS D:\proyecto_01> git push -u origin master
```

Pero como es la primera vez que lo hacemos, pide las credenciales



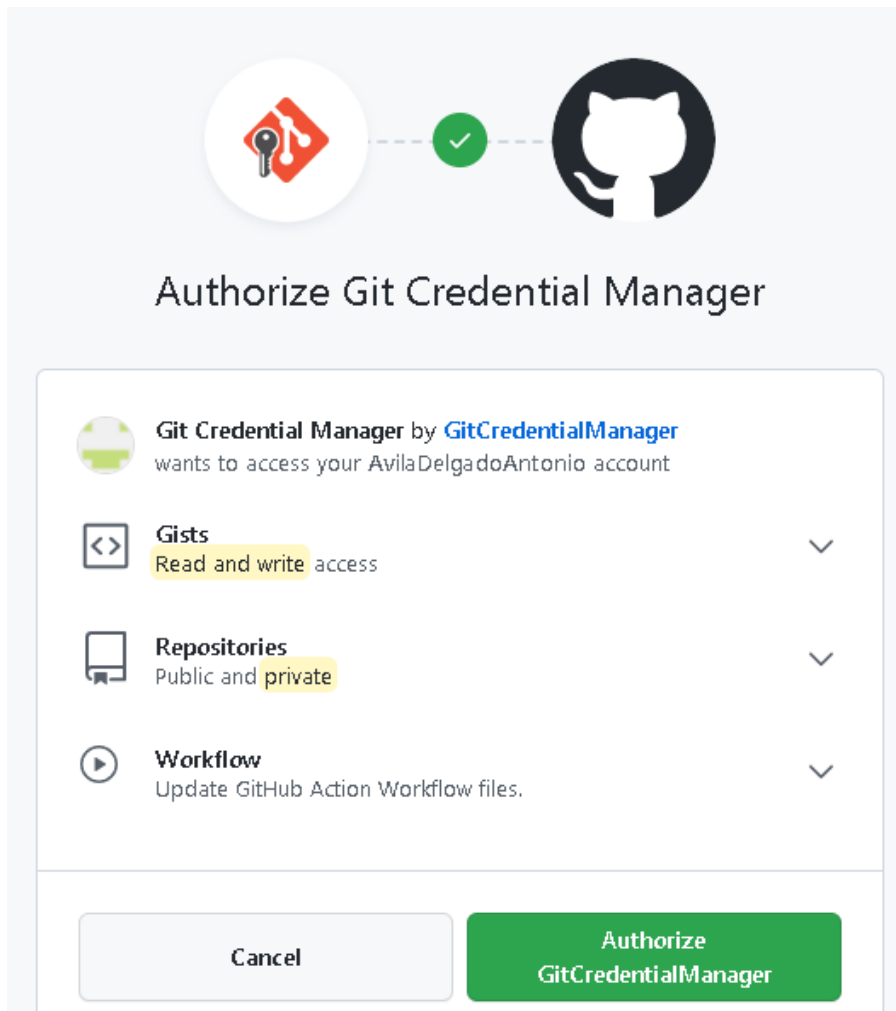
A screenshot of a 'Connect to GitHub' dialog box. It features the GitHub logo and the text 'Sign in'. There are two main options: 'Sign in with your browser' with an external link icon, and a section separated by 'or' for 'Personal Access Token' with a corresponding input field and a 'Sign in' button. At the bottom, it says 'Don't have an account? Sign up' with a link.

Nos conectamos de esta manera para dar las credenciales de GitHub

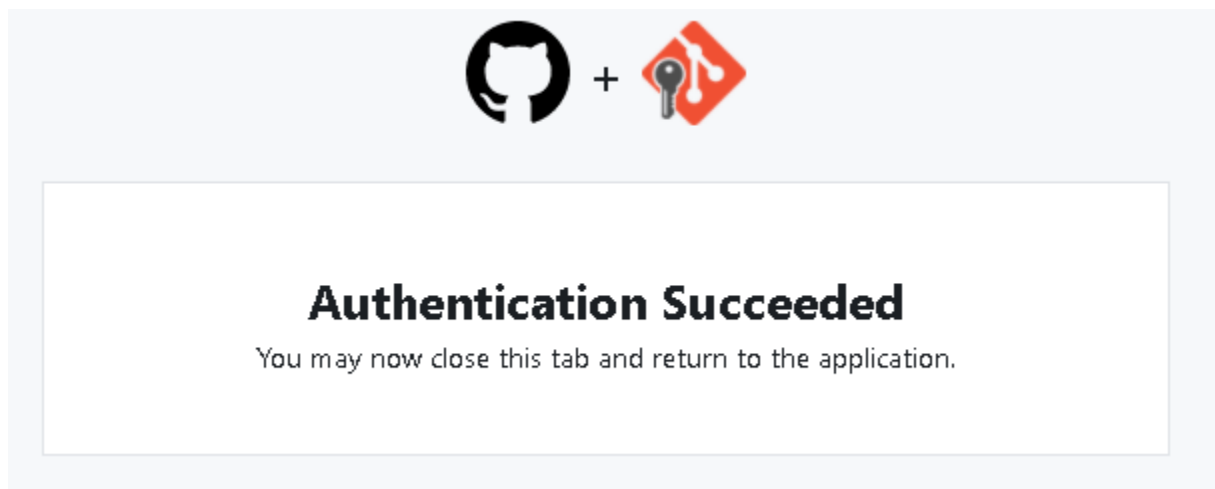


A screenshot of the Git Credential Manager sign-in screen. It has a GitHub logo at the top. Below it, the text reads 'Sign in to GitHub to continue to Git Credential Manager'. There are two input fields: 'Username or email address' containing 'AvilaDelgadoAntonio' and 'Password' with masked characters. A 'Forgot password?' link is next to the password field. At the bottom is a large green 'Sign in' button.

Y en esta pantalla autorizamos



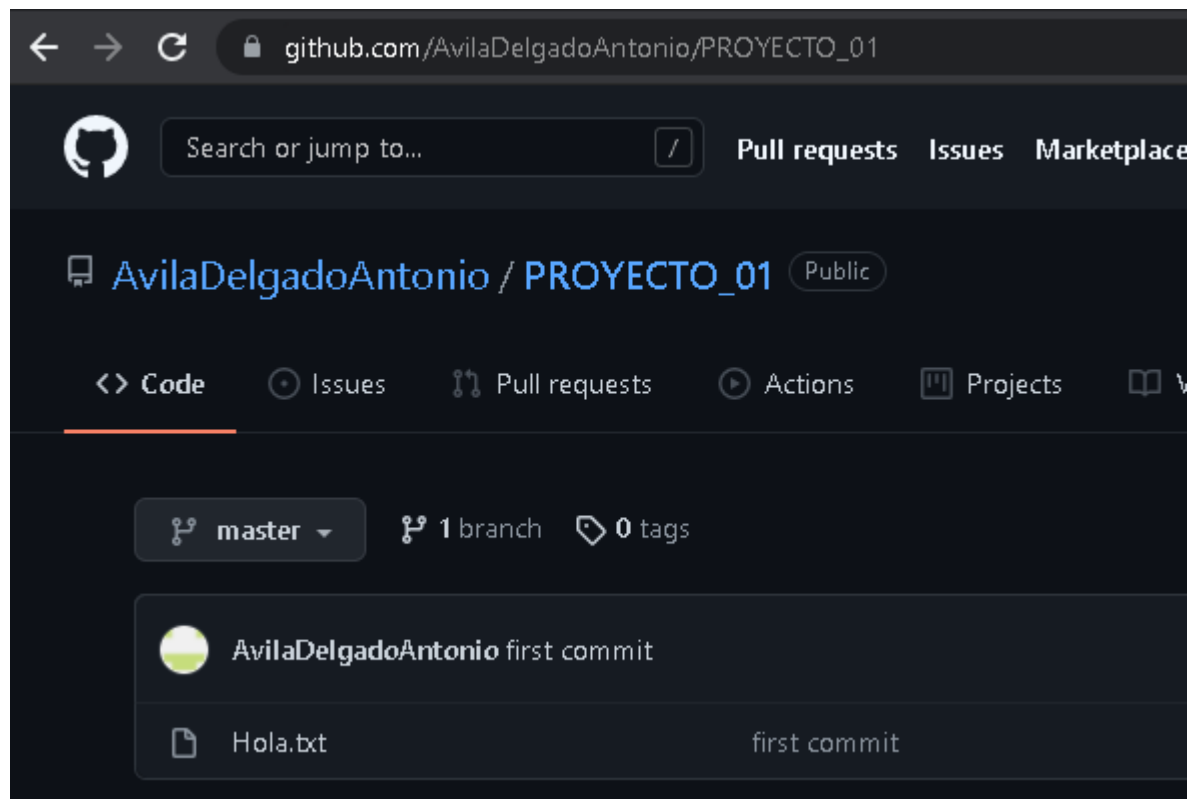
Y conseguimos un mensaje de éxito



Y todo funciona a la perfección al hacer el “push”

```
PS D:\proyecto_01> git push -u origin master
info: please complete authentication in your browser...
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 249 bytes | 249.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/AvilaDelgadoAntonio/PROYECTO_01.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

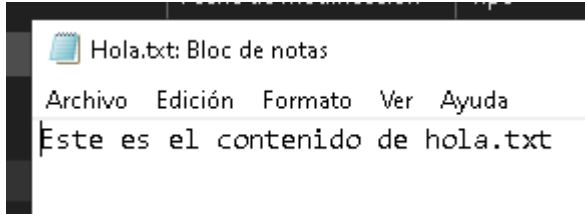
Comprobamos que ha sido así, que nuestro archivo hola.txt ha subido con éxito a GitHub



PRUEBAS

a) Pruebas PUSH

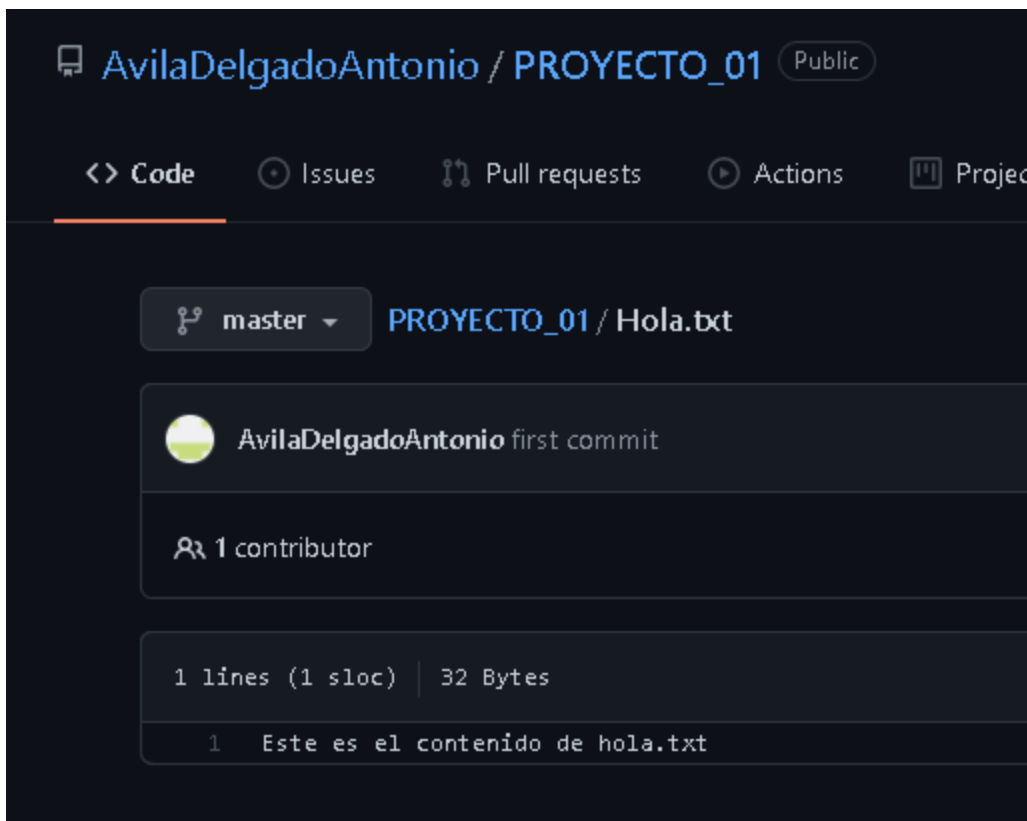
Usamos PUSH para subir este archivo con este contenido



Y el PULL funciona correctamente

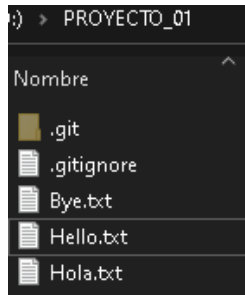
```
PS D:\proyecto_01> git push -u origin master
info: please complete authentication in your browser...
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 249 bytes | 249.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/AvilaDelgadoAntonio/PROYECTO_01.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Y vemos que efectivamente el archivo, con el contenido íntegro, ha subido bien a GitHub

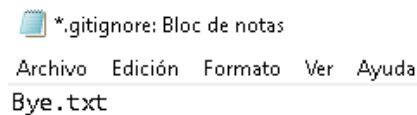


b) Pruebas .gitignore

Tenemos estos archivos:



Hemos especificado en el .gitignore que NO haga un PUSH con el archivo Bye.txt, pero sí con los demás



Me posiciono en powershell en el directorio,

```
PS D:\proyecto_01>
```

Y lanzamos el comando git init

```
PS D:\proyecto_01> git init
Reinitialized existing Git repository in D:/PROYECTO_01/.git/
```

Es interesante anotar que ha detectado que git ya se usó con anterioridad, por tanto ahora aparece la palabra “reinitialized” en el mensaje que nos lanza el sistema.

Usamos el comando git add . (y aquí es donde nos debería funcionar el .gitignore con respecto al archivo Bye.txt). Hacemos un git status y nos aparece lo siguiente

```
PS D:\proyecto_01> git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .gitignore
        new file:   Hello.txt
```

Lo que recibimos se interpreta de la siguiente manera:

- El archivo Bye.txt NO aparece, porque fue incluido en .gitignore
- El archivo Hola.txt NO aparece, porque ya hicimos un PUSH con anterioridad, por lo cual no lo vuelve a considerar porque no se ha hecho ningún cambio desde entonces.
- El archivo Hello.txt Sí aparece, ya que es nuevo, y aún no se ha sometido a un PUSH.

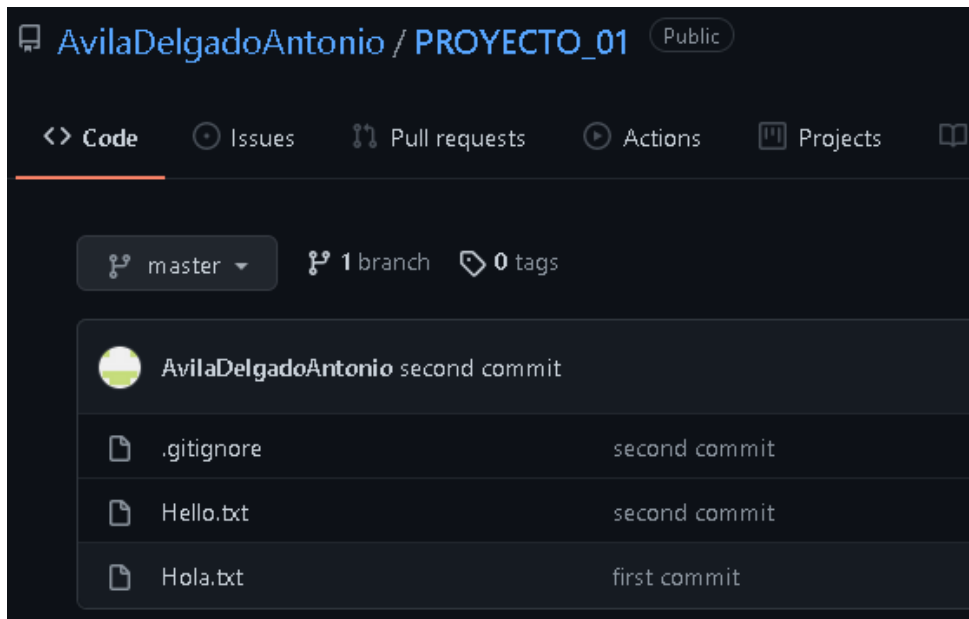
Y proseguimos con el commit

```
PS D:\proyecto_01> git commit -m "second commit"
[master 6527e44] second commit
 2 files changed, 2 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 Hello.txt
PS D:\proyecto_01>
```

Y ahora lo lanzamos para subirlo gracias al comando git push

```
PS D:\proyecto_01> git push -u origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 346 bytes | 346.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/AvilaDelgadoAntonio/PROYECTO_01.git
 c320916..6527e44 master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
PS D:\proyecto_01>
```

Vamos a GitHub y vemos los siguiente:



Lo cual concuerda con la interpretación que hicimos antes, la cual era:

- El archivo Bye.txt NO aparece, porque fue incluido en .gitignore
- El archivo Hola.txt ahora SÍ aparece, porque ya hicimos un PUSH con anterioridad con el “first commit”. Simplemente no ha subido, porque ya estaba en el repositorio de GitHub.
- El archivo Hello.txt SÍ aparece, ya que es nuevo, y aún no se había sometido a un PUSH.

Es interesante notar que el archivo .gitignore, cuya función es simplemente excluir archivos del proceso PUSH, también sube al repositorio.

c) **Pruebas PULL**

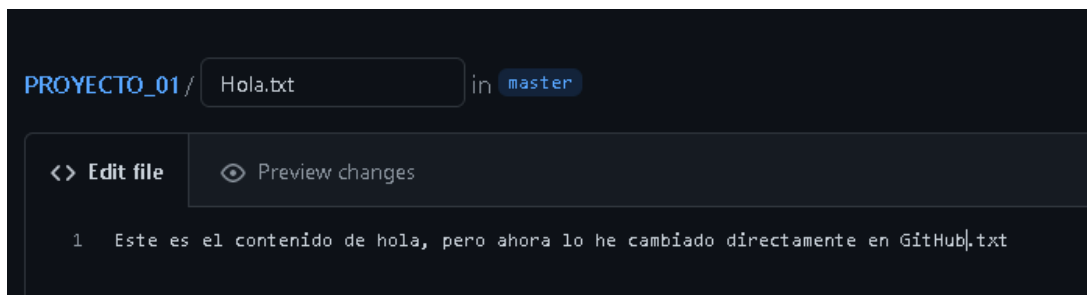
Para hacer la prueba con PULL, primero vamos a hacer un cambio directamente en el repositorio, y después hacemos un PULL para sincronizarlo con la carpeta local.

Hacemos este cambio:

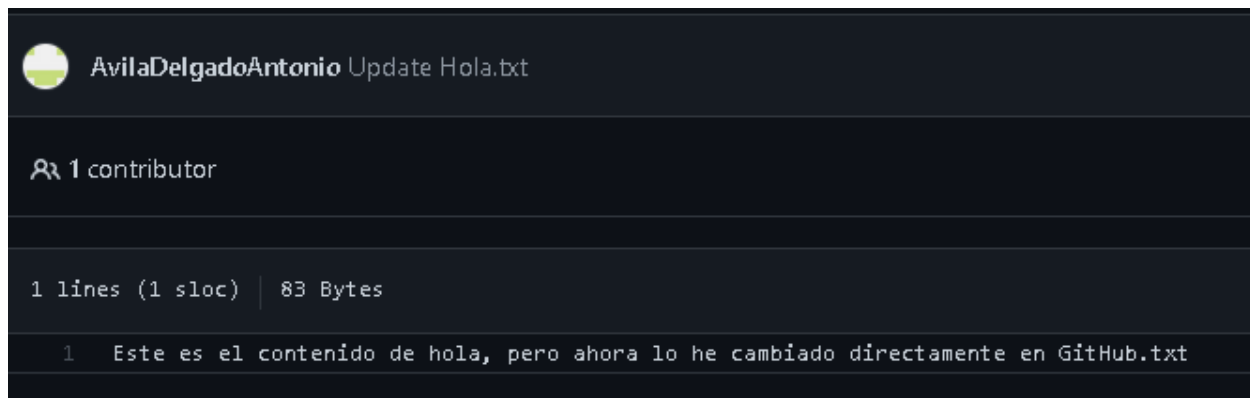
ANTES del cambio



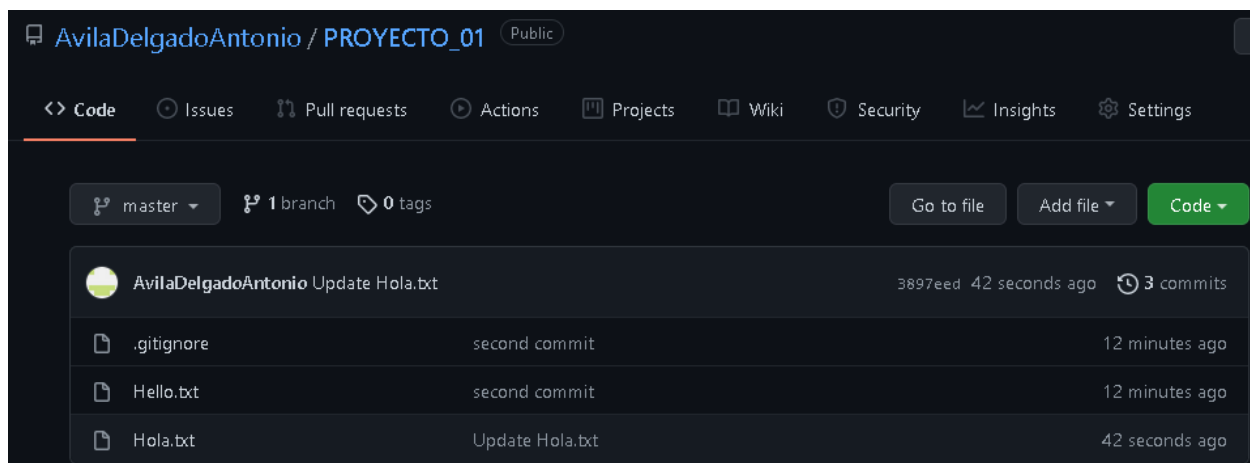
DESPUÉS del cambio



Y se ve así



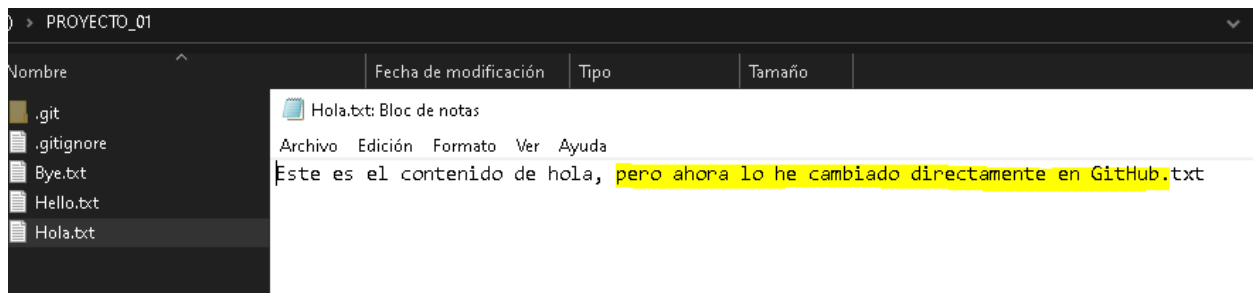
Y se refleja de esta manera en la lista de archivos del repositorio



Ahora hacemos un PULL

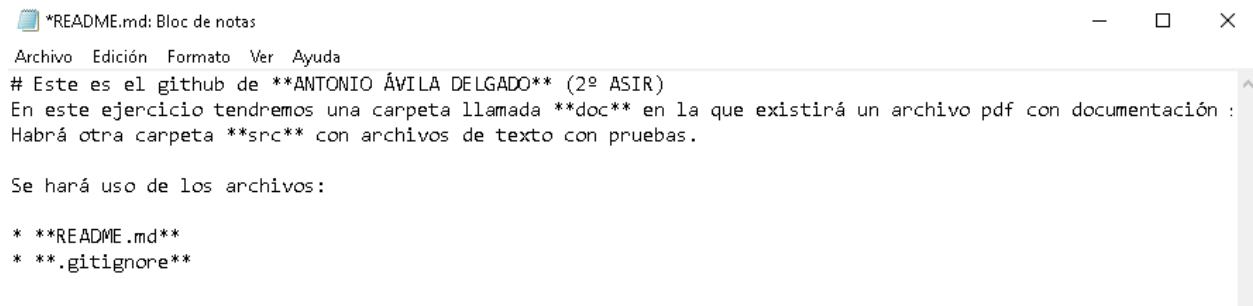
```
PS D:\proyecto_01> git pull origin master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 706 bytes | 2.00 KiB/s, done.
From https://github.com/AvilaDelgadoAntonio/PROYECTO_01
 * branch                master      -> FETCH_HEAD
    6527e44..3897eed      master      -> origin/master
Updating 6527e44..3897eed
Fast-forward
  Hola.txt | 2 +-
  1 file changed, 1 insertion(+), 1 deletion(-)
PS D:\proyecto_01>
```

Y comprobamos que se ha sincronizado perfectamente el cambio en la carpeta local

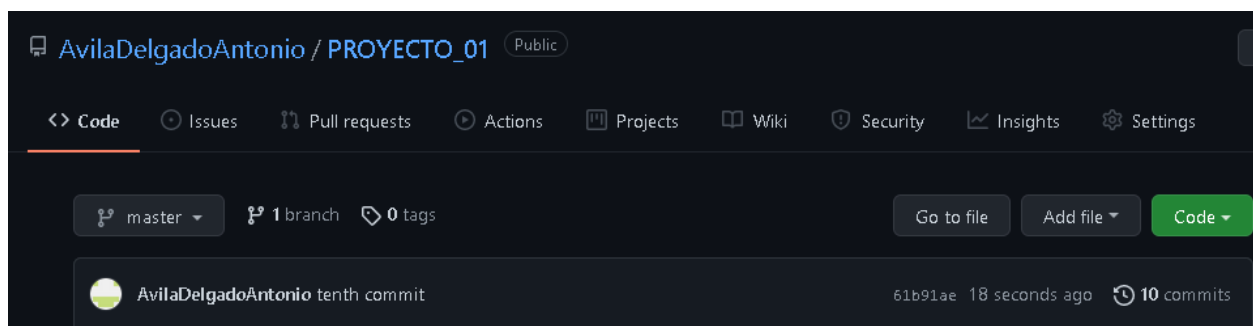


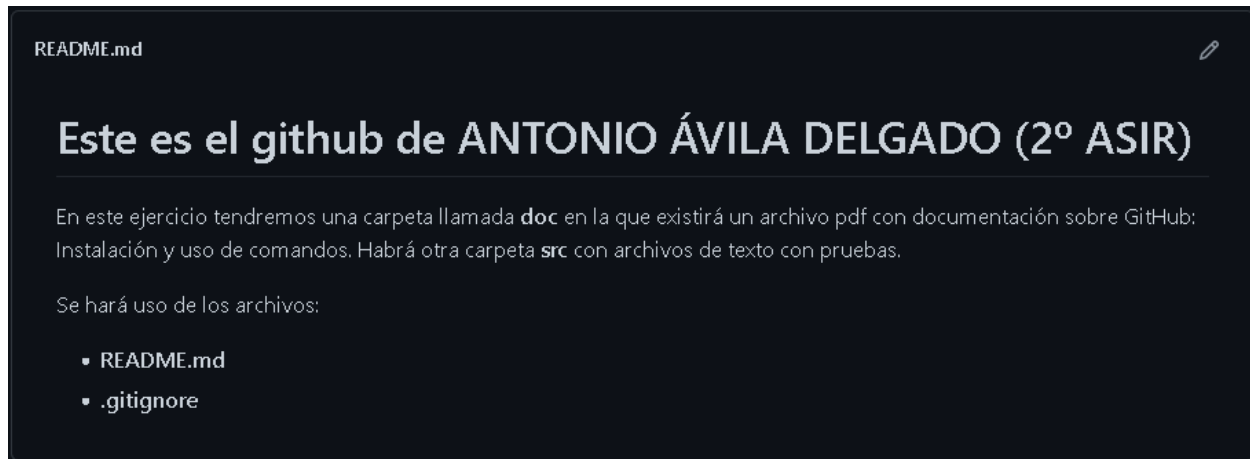
d) Pruebas README.md

Creamos un archivo README.md con su tipología correspondiente



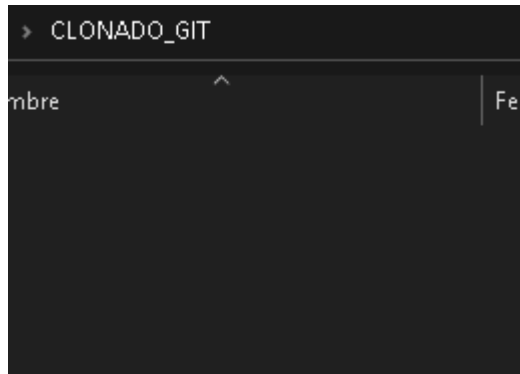
Y hacemos un PUSH para sincronizar con el repositorio GitHub, y esto es lo que obtenemos:





e) Pruebas DOWNLOAD

Creamos una carpeta distinta (ej: CLONADO_GIT), donde vamos a clonar todo el contenido que tenemos en GitHub para PROYECTO_01



Nos situamos en CLONADO_GIT y usamos entonces la siguiente línea de comandos para clonar

```
PS D:\clonado_git> git clone --branch master https://github.com/AvilaDelgadoAntonio/PROYECTO_01.git
```

Se inicia el proceso

```
Cloning into 'PROYECTO_01'...
```

Se concluye con éxito

```
PS D:\clonado_git> git clone --branch master https://github.com/AvilaDelgadoAntonio/PROYECTO_01.git
Cloning into 'PROYECTO_01'...
remote: Enumerating objects: 28, done.
remote: Counting objects: 100% (28/28), done.
remote: Compressing objects: 100% (19/19), done.
Receiving objects: 71% (19/28), 228.00 KiB | 389.00 KiB/sd 0 eceiving objects: 50% (14/28), 228.00 KiB | 389.00 KiB/s
Receiving objects: 100% (28/28), 457.21 KiB | 461.00 KiB/s, done.
Resolving deltas: 100% (2/2), done.
PS D:\clonado_git>
```

Y efectivamente tenemos todo clonado. Comprobamos por ejemplo la carpeta src donde están todos los archivos



e) Pruebas DOWNLOAD

```
PS D:\proyecto_01> git download origin master
git: 'download' is not a git command. See 'git --help'.
```

No se puede realizar porque el comando download no existe en Git. Se recomienda usar en su lugar PULL.