Práctica Git y Github

Índice:

Instalaciones y creación de cuenta:

- Instalación de Git
- Creación de cuenta de Github

Primera práctica:

- **Explicación general de la práctica**
- **Ejercicios de la práctica**
    - **Primer Ejercicio:** Crear nuestro repositorio de GIT y agregarle un archivo
    - **Segundo Ejercicio:** Subir un repositorio de GIT a un repositorio vacío de Github
        - Creación del repositorio vacío con Github
        - Subir un repositorio de Git a un repositorio de Github + Enlazamiento de las cuentas
    - **Tercer Ejercicio:** Uso de comandos varios de GIT

Teoría sobre la práctica y resolución de incidentes

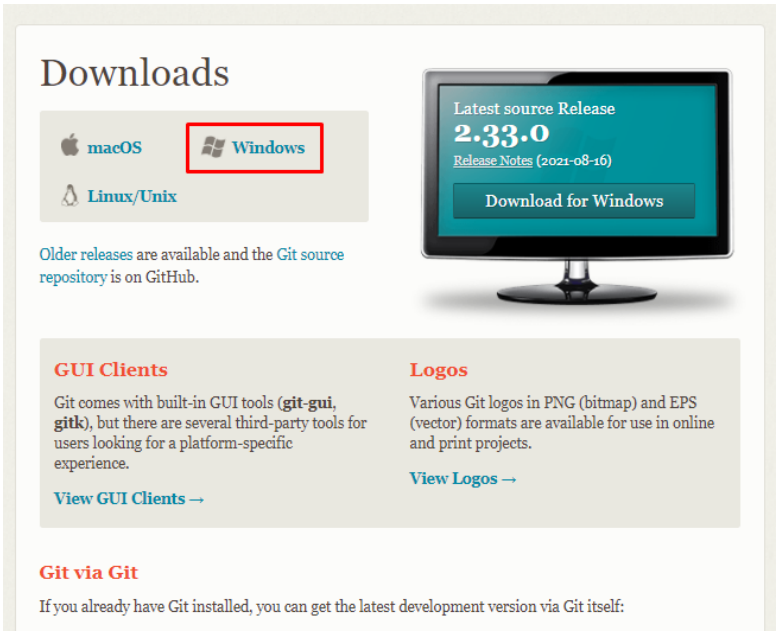
- Guía de comandos
- Manejo de credenciales.
- Resolución de incidentes.

Instalaciones y creación de cuenta:

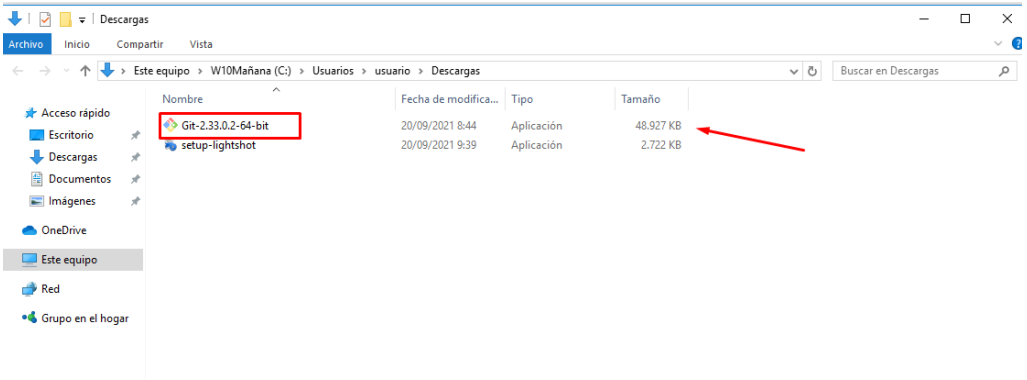
1. **Descarga de Git**

En primer lugar deberemos dirigirnos a la página web https://git-scm.com y una vez dentro de la página clickearemos sobre “Downloads”

Dentro del menú de las distintas descargas para cada uno de los sistemas operativos clickearemos sobre el nuestro, en este caso “Windows”.
Automáticamente comenzará la descarga de la última versión de Git.

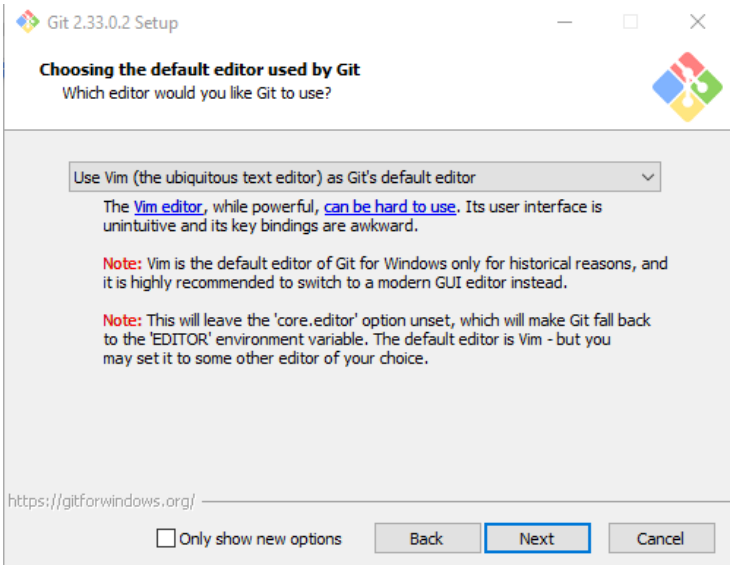


Iremos a la carpeta donde hemos descargado el instalador y clickearemos sobre él para comenzar la instalación

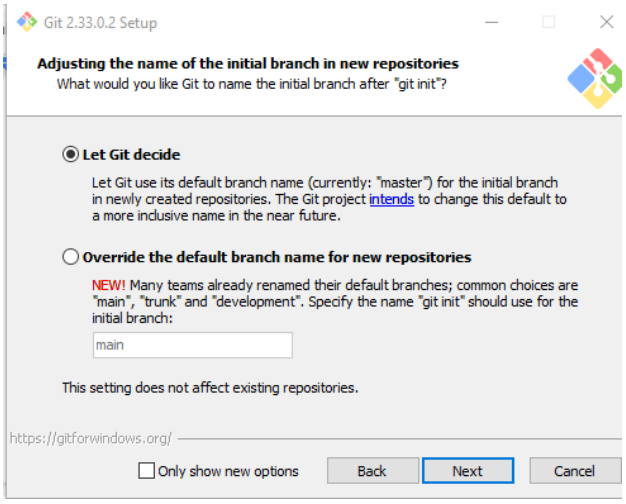


Una vez iniciada la instalación pulsaremos “next” en todas las opciones que se nos vayan abriendo

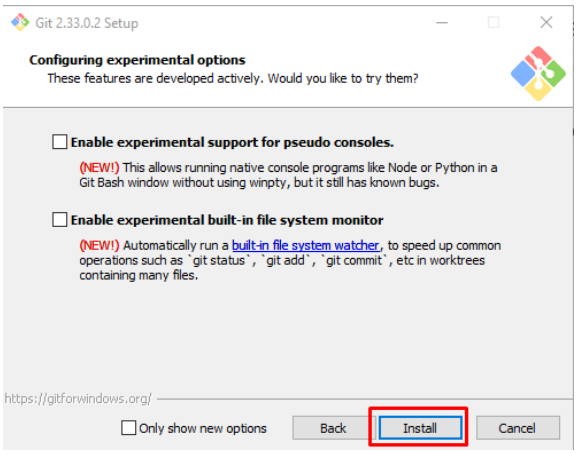
Podremos decidir que editor queremos utilizar, en mi caso utilizaré “Vim” el cual es la opción por defecto.



Y ya para el resto de ventanas de la instalación simplemente pulsaremos “Next” dejando las opciones por defecto.

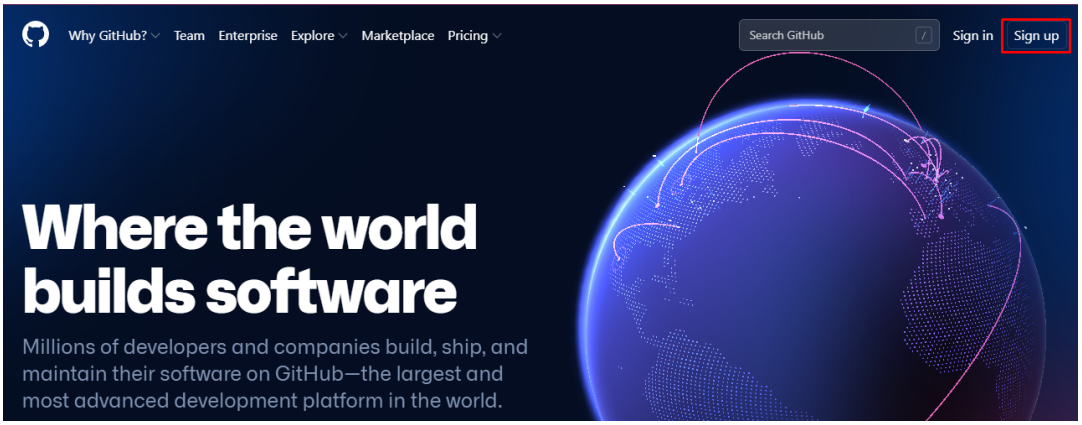


Finalmente pulsaremos “Install” para concluir así con la instalación de Git.



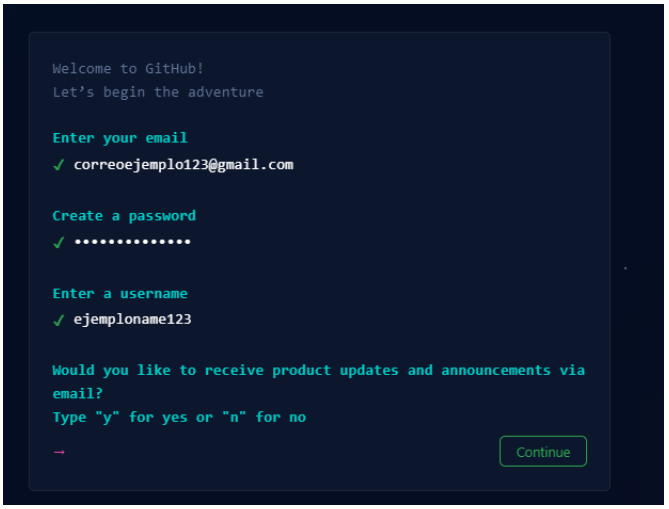
2. **Creación de cuenta en Github**

Procederemos con la creación de cuenta de Github dirigiéndonos a la página de Github https://github.com, dentro de la cual haremos click sobre “Sign up” para proceder a crear la cuenta

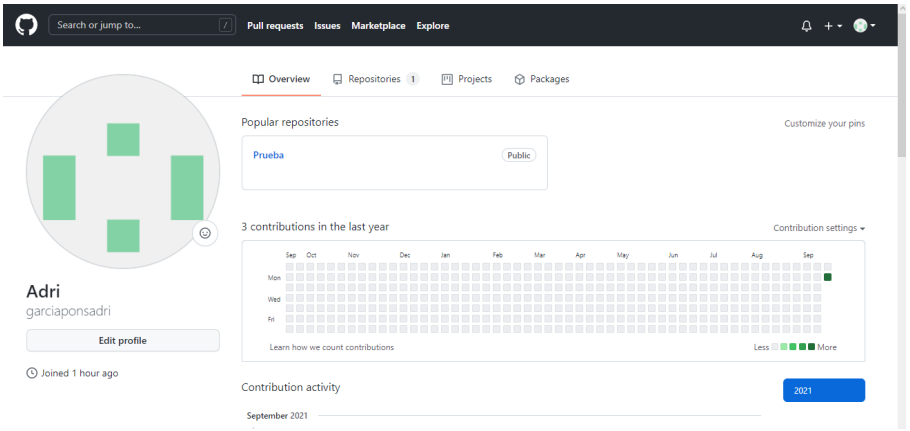


Dentro de la página introduciremos el correo que queramos, contraseña, username (el cual es único) y por último si queremos recibir spam de github (sugiero decir No).

Una vez realizados estos pasos nos enviará un correo de verificación a la dirección email que hayamos introducido.



Y así tendremos lista ya nuestra cuenta de Github.



Primera práctica:

1. Explicación general de la práctica

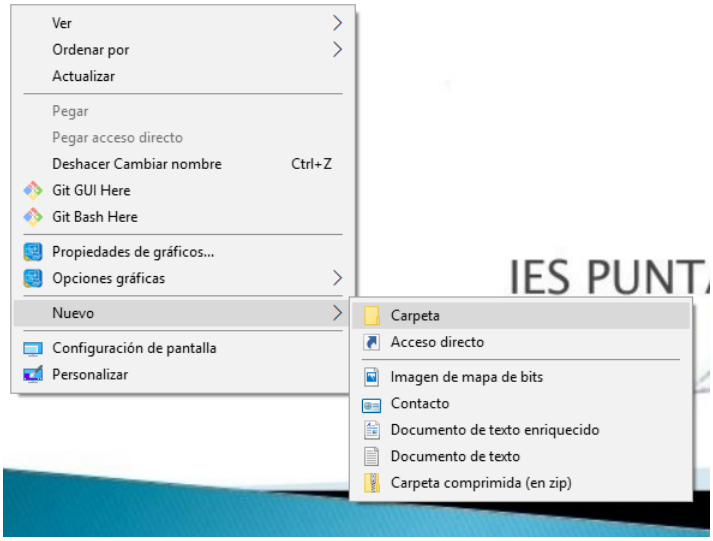
En esta primera práctica precisamos de:

- Cuenta en Github
- Aplicación de Git de escritorio.
- Microsoft Powershell (también se pueden usar otras consolas).

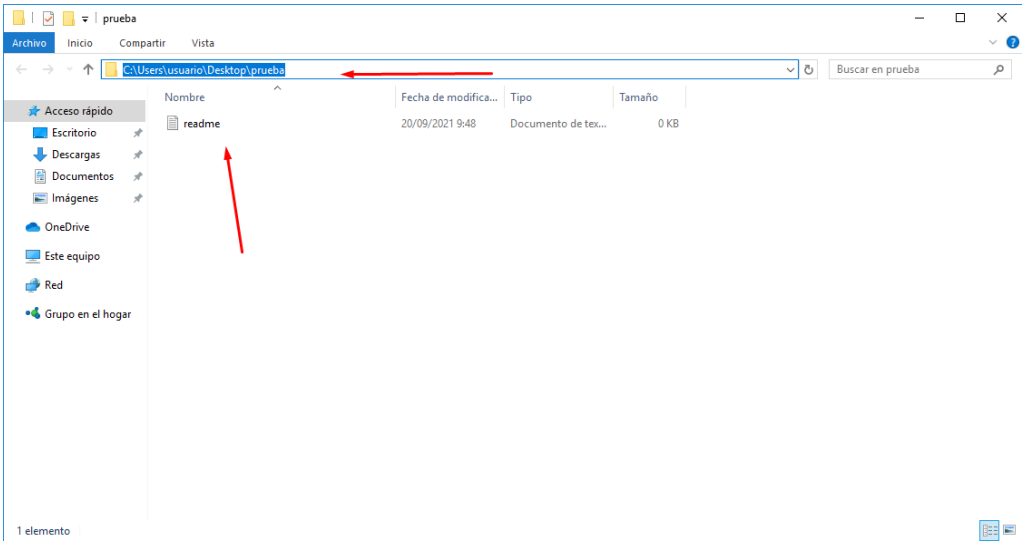
La práctica consistirá en distintos ejercicios básicos en los que veremos los distintos comandos de Git y algunas funciones básicas de este.

2. **Primer ejercicio:** Creación de nuestro repositorio GIT y agregarle un archivo

En primer lugar deberemos crear una carpeta contenedora de los archivos que queramos agregar a nuestro repositorio.



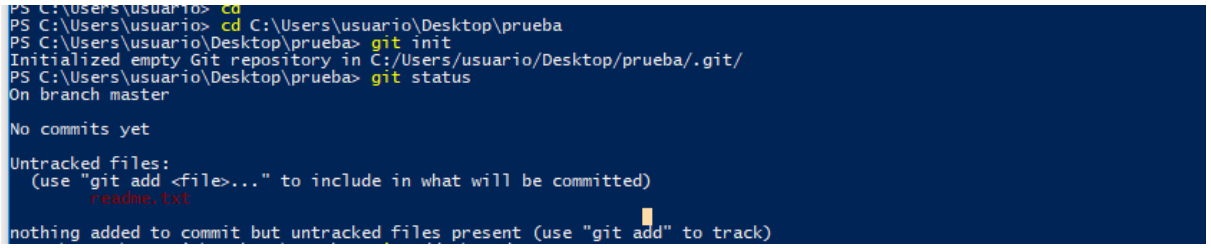
Una vez creada la carpeta crearemos un archivo de texto (por ejemplo) y cogeremos la URL de la carpeta para situarnos dentro de esa carpeta con nuestro Powershell.



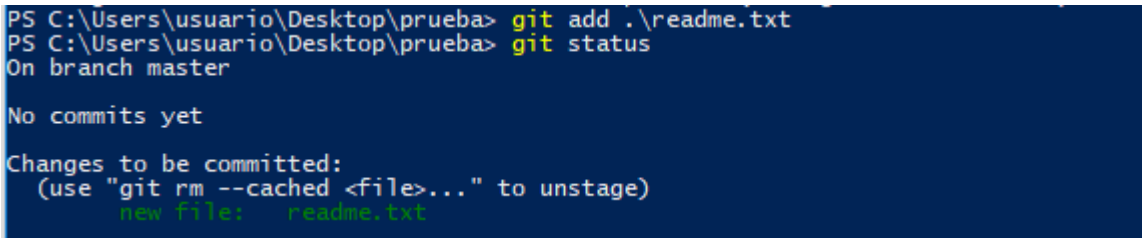
Cogeremos la URL y dentro de la consola de powershell haremos una orden “**cd <url>**” para situarnos dentro de la carpeta donde tendremos los archivos que queramos agregar al repositorio (En este caso el documento de texto).

Acto seguido haremos un”**git init**” para inicializar el repositorio de Git.

Y con “**Git status**” veremos como se ha creado el repositorio pero está vacío.



Mediante la orden “**git add <nombre archivo.tipo archivo>**” meteremos el archivo de texto creado en el repositorio que hemos creado y además con la orden “git status” comprobaremos que la operación se haya realizado con éxito.

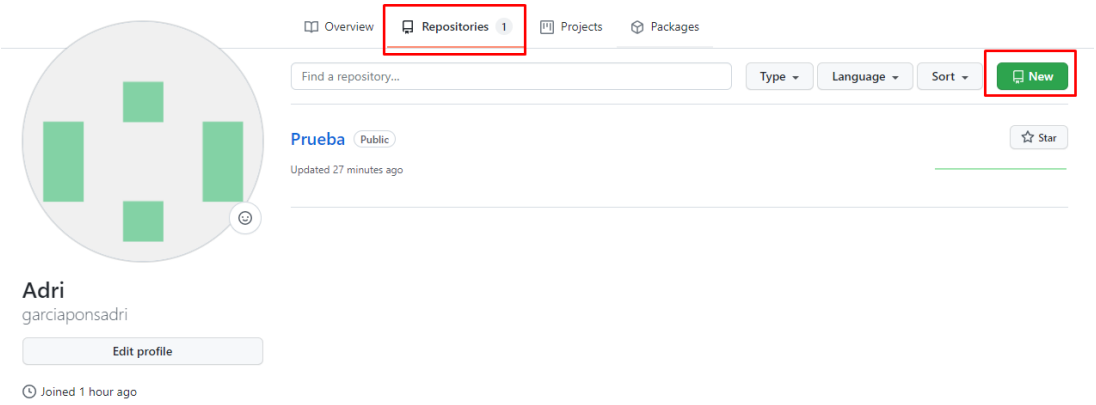


3. **Segundo Ejercicio:** Subir un repositorio de GIT a un repositorio vacío de Github

a. Creación del repositorio vacío en Github

Partiremos de la base del primer ejercicio en el que creamos un repositorio GIT con al menos un archivo.

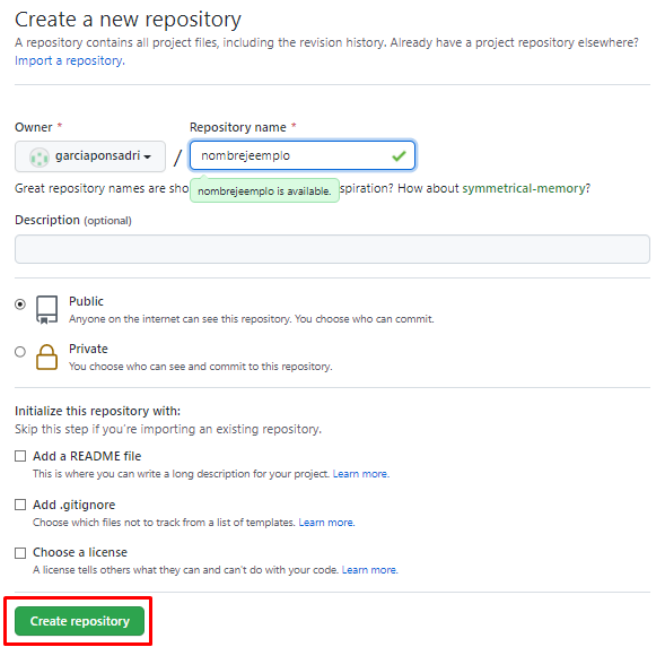
Dentro de nuestro perfil daremos **click sobre "Repositorios"** y **luego sobre "New"** para crear un nuevo repositorio dentro de Github



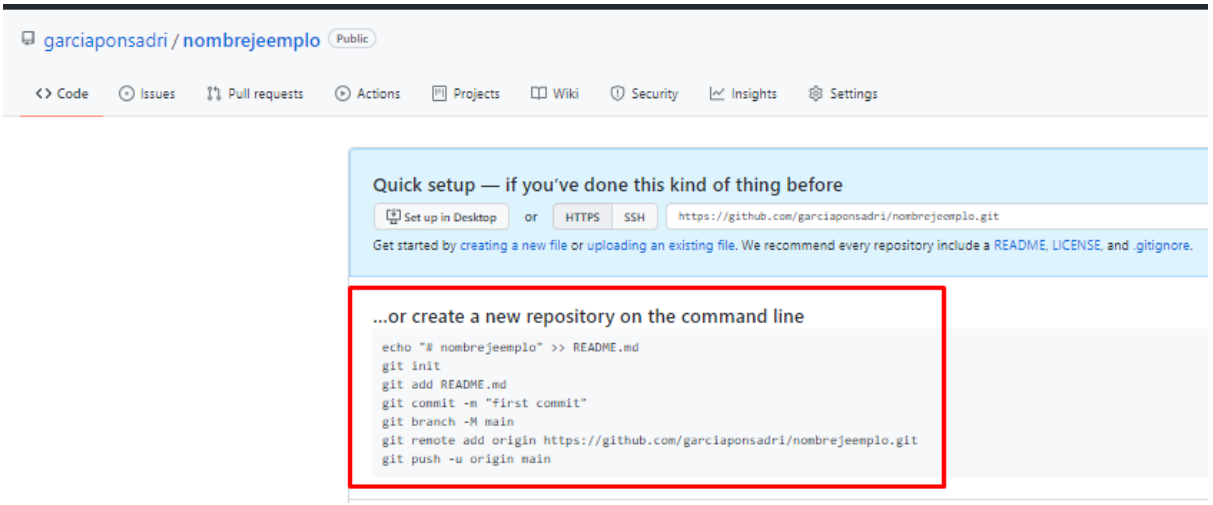
Comenzaremos a rellenar los campos siendo "repository name" el nombre que queramos para el repositorio pero teniendo en cuenta que no se puede repetir el nombre en otro.

También tendremos opción a crear ficheros adicionales como el "readme", el cual es un fichero de documentación, y si queremos que nuestro repositorio sea privado o público

Una vez completados los campos simplemente daremos click a crear repositorio.



Una vez creado el repositorio nos abrirá la siguiente ventana la cual en la pestaña de “code” nos dará el código que necesitaremos para importar un repositorio desde la aplicación de Git, lo cual es justamente lo que realizaremos en la práctica



Dejaremos esta pestaña abierta ya que mediante esos mismos comandos realizaremos el repositorio con Git y lo exportaremos a Github.

b. Subir un repositorio de Git a un repositorio de Github + Enlazamiento de las cuentas

Una vez introducido el archivo, haremos un”**git commit**” para comittear nuestro repositorio

Al ser la primera vez que utilizamos Git en nuestro dispositivo se nos pedirá las credenciales (email y username), así que las rellenaremos utilizando los comandos que el mismo git nos da. “**git config --global user.email “correo”**” y “**git config --global user.name “nombre”**”. Deberemos introducir nuestro email y username exactos de github respectivamente.

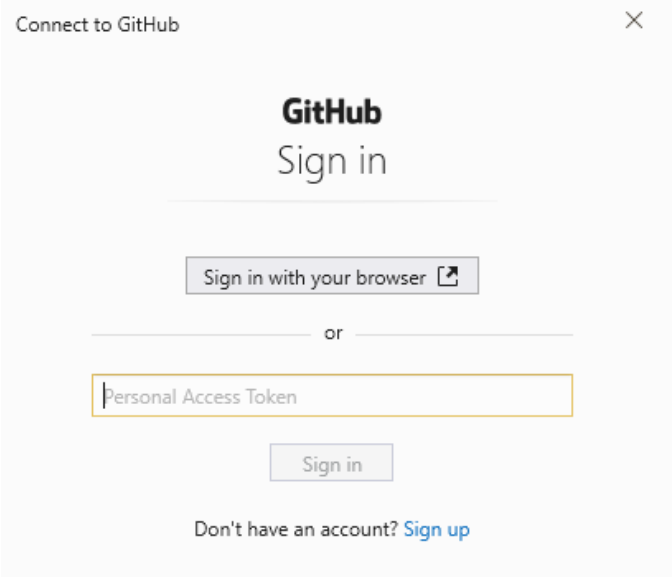
Una vez dados los credenciales, al hacer el “**git commit**” veremos como comittea ya sin pedirnos las credenciales.

En caso de utilizar distintas cuentas desde un mismo ordenador deberemos eliminar los credenciales “antigüos” para un funcionamiento correcto del git (Podremos ver en el apartado de manejo de credenciales).

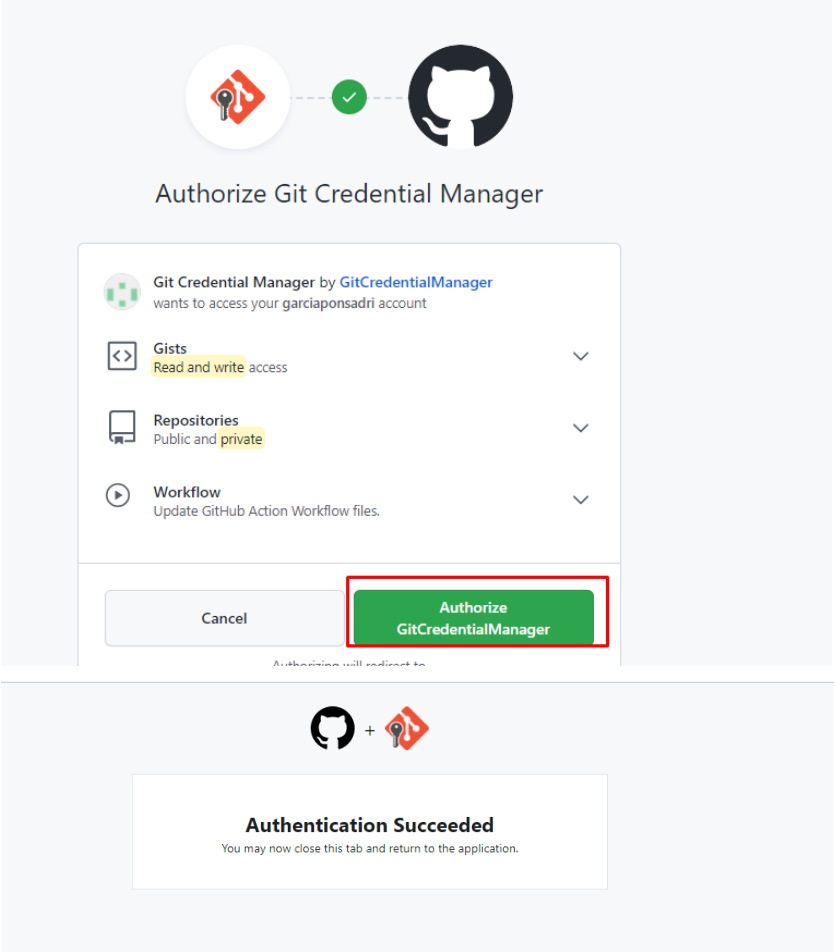
Una vez hecho el commit, haremos una instrucción “**git branch -M <main>**”, un “**git remote add origin <url del repositorio de github>**”, y finalmente una orden push; “**git push -u <repositorio github> <main>**”, que nos abrirá una ventana. La cual una vez abierta seleccionaremos la opción de entrar mediante web.

```
PS C:\Users\usuario\Desktop\prueba> git branch -M main
PS C:\Users\usuario\Desktop\prueba> git remote add origin https://github.com/garciaponsadri/Prueba.git
PS C:\Users\usuario\Desktop\prueba> git push -u origin main
info: please complete authentication in your browser...
```

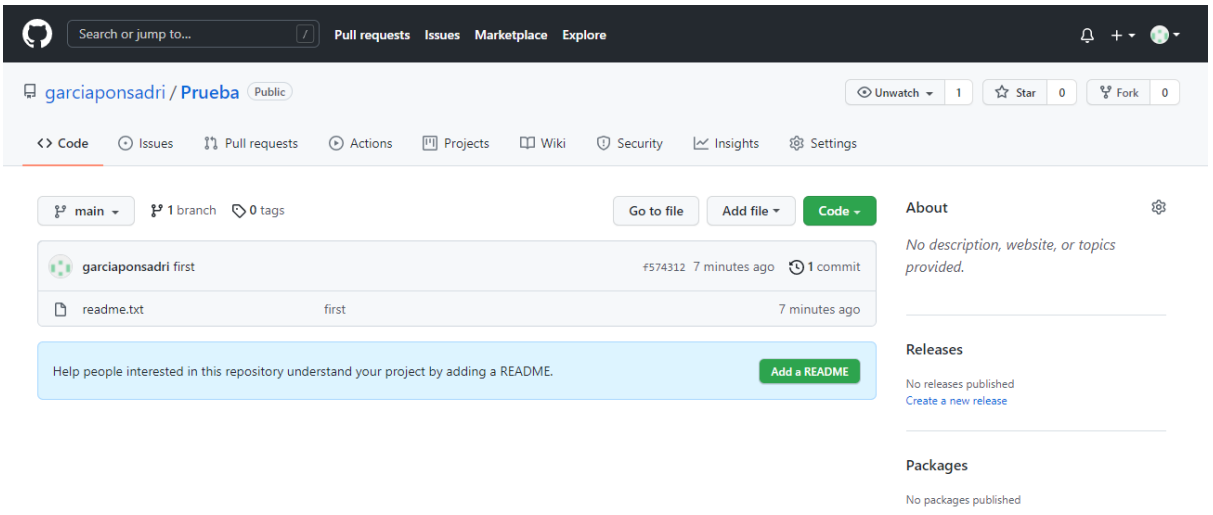
Justo después de realizar el push se nos abrirá la siguiente pestaña en la que seleccionaremos la opción de “**Sign in with your browser**”



Nos saldrá la siguiente ventana en la cual autorizamos al “**GitCredentialManager**” mediante el cual finalmente linkearemos la cuenta de github web y nuestra aplicación de escritorio Git.



Una vez realizada la autenticación veremos en el repositorio web lo que hemos hecho en el repositorio de Git desde nuestro ordenador.

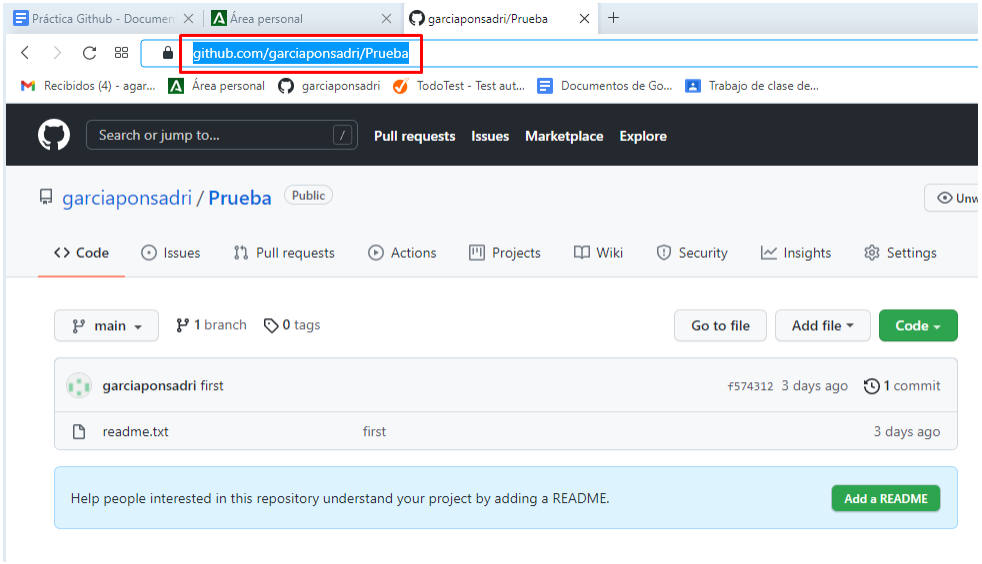


4. **Tercer Ejercicio:** Uso de comandos varios de GIT

a. Git clone

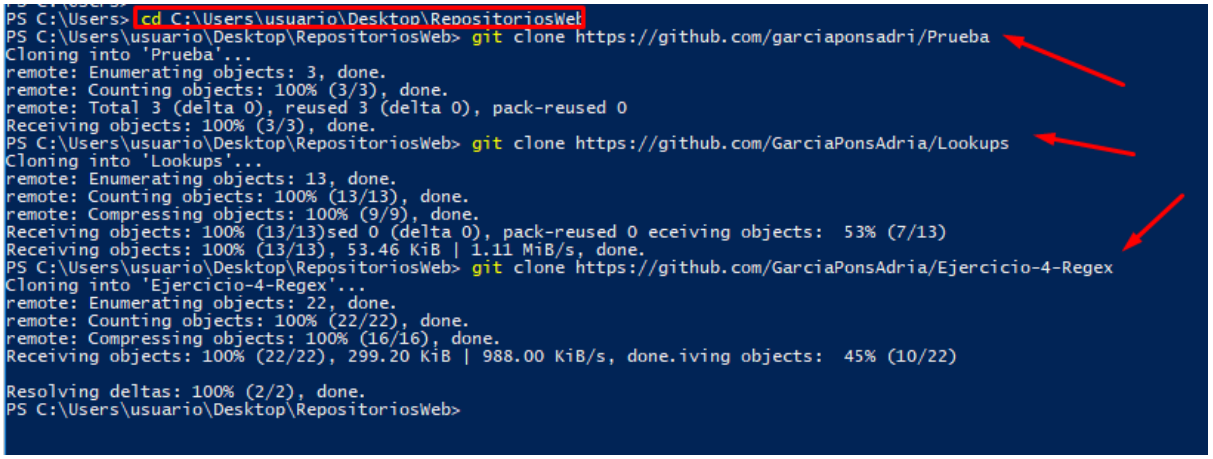
El comando git clone nos permite clonar un repositorio web a una carpeta contenedora mediante GIT. Aquí tenemos por ejemplo una carpeta contenedora “RepositoriosWeb” donde clonaremos varios repositorios de Github ahí

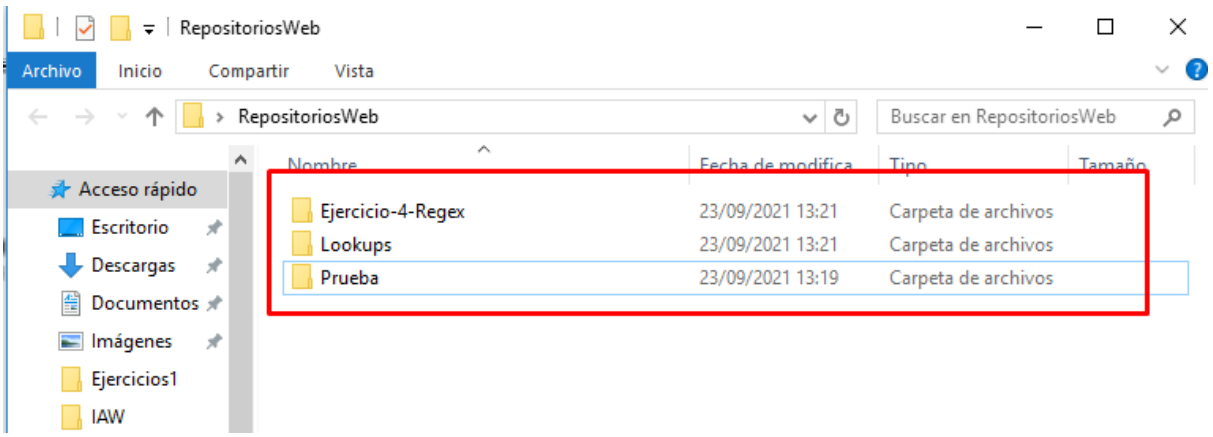
Nos dirigiremos a Github y copiaremos la URL del repositorio que queramos clonar



Ya dentro de Shell nos iremos mediante la orden “**cd <url>**” al directorio que queramos clonar los repositorios web, en este caso a la carpeta “RepositoriosWeb”, y una vez dentro pondremos el comando “**git clone <url repositorio web>”**

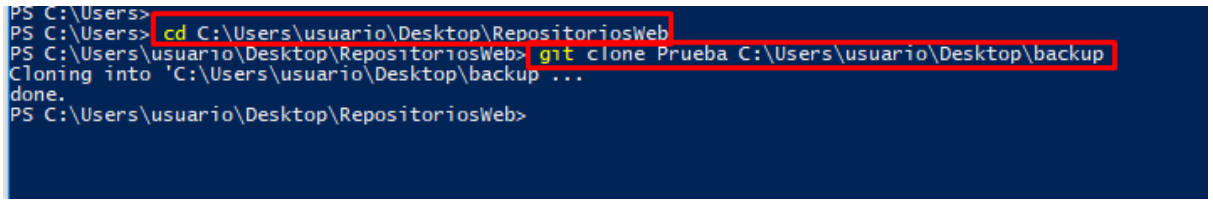
En Este caso hemos clonado 3 repositorios web en la carpeta (De hecho, 2 de estos repositorios no son de la cuenta que tenemos vinculada, de forma que podremos clonar cualquier repositorio sea nuestro o no).

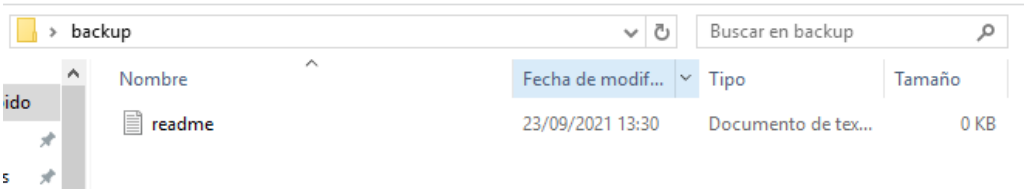




También podremos clonar repositorios existentes colocandonos en su directorio mediante “**cd**” y utilizando el comando “**git clone <repositorio> <url donde queramos clonarlo>**.

Pasando la carpeta destino de estar vacía al tener el contenido del repositorio clonado.





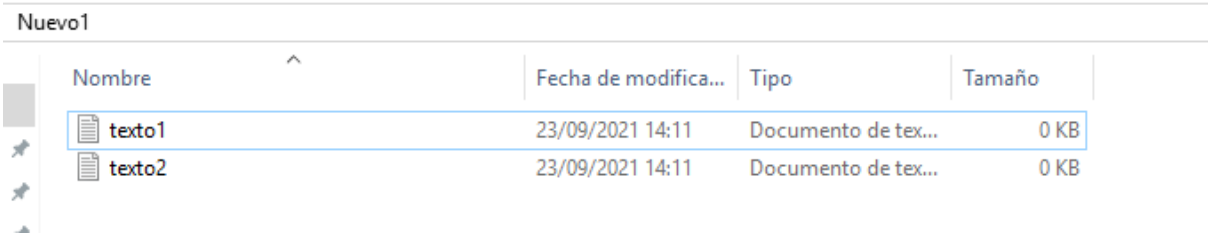
b. Git rm

En este caso hemos creado un repositorio con 3 archivos de texto, sin embargo, tras añadirlos al repositorio eliminaremos uno de ellos.

Lo haremos mediante el comando **git rm <archivo>**, y luego comprobaremos que haya realizado correctamente la operación realizando un **git commit**

```
PS C:\Users>
PS C:\Users>
PS C:\Users> cd C:\Users\usuario\Desktop\Nuevo1
PS C:\Users\usuario\Desktop\Nuevo1> git init
Initialized empty Git repository in C:/Users/usuario/Desktop/Nuevo1/.git/
PS C:\Users\usuario\Desktop\Nuevo1> git add .
PS C:\Users\usuario\Desktop\Nuevo1> git commit -m "Agregamos los archivos del directorio"
[master (root-commit) 99d56ae] Agregamos los archivos del directorio
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 ignorame.txt
 create mode 100644 texto1.txt
 create mode 100644 texto2.txt
PS C:\Users\usuario\Desktop\Nuevo1> git rm .\ignorame.txt
rm 'ignorame.txt'
PS C:\Users\usuario\Desktop\Nuevo1> git commit -m "Hemos eliminado el archivo"
[master 445a908] Hemos eliminado el archivo
 1 file changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 ignorame.txt
PS C:\Users\usuario\Desktop\Nuevo1>
```

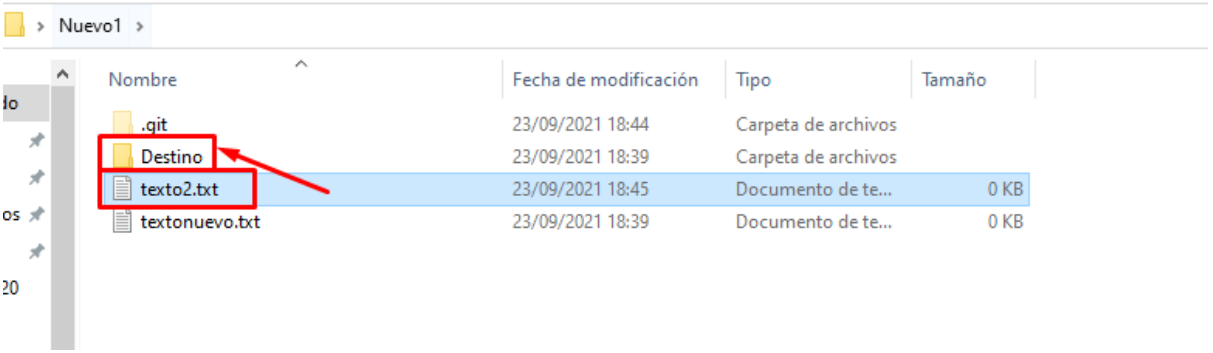
Aqui tenemos la carpeta del repositorio con el archivo ignorame.txt eliminado



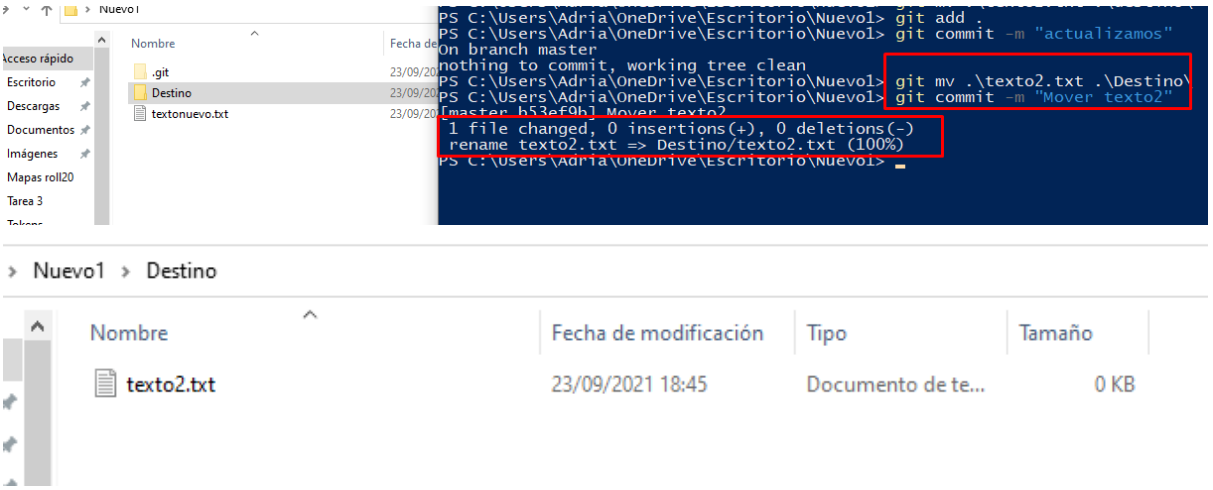
c. Git mv

Ahora con el comando **git mv** veremos como podemos mover o cambiar el nombre a archivos, como en el ejemplo siguiente que aprovechando el repositorio anterior cambiaremos el nombre del archivo "**texto1**" (Parte de los siguientes comandos los muestro desde mi equipo personal).

Ahora en lugar de renombrar moveremos un archivo a una carpeta existente dentro del repositorio, en este caso a la carpeta destino



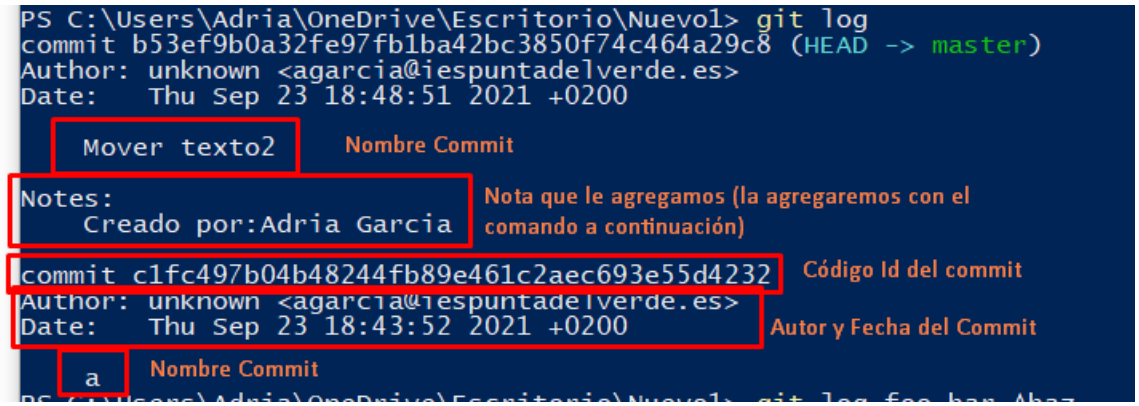
Mediante la línea "**git mv .\texto2.txt .\Destino**", y habiendo anteriormente añadido al repositorio la carpeta destino y comitteado veremos como el documento texto2 desaparece de nuestra carpeta y el "**git commit**" nos avisa que algo se ha actualizado.



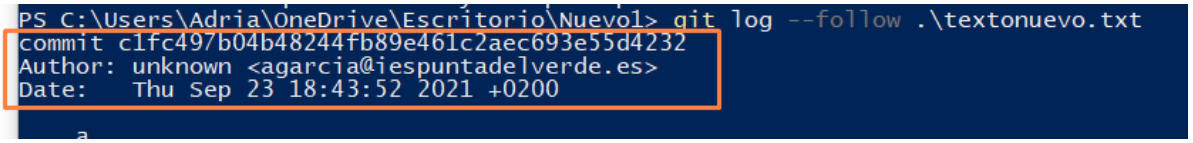
d. Git log

Este comando nos dará un registro de Commits a modo de historial, lo cual nos permitirá tener todo mucho mejor controlado.

Tenemos en primer lugar el comando a secas "**git log**" el cual nos mostrará un registro con algunos de los commits realizados como vemos a continuación. (También, podremos poner **"git log <id commit>"** para que nos salga directamente el commit que queramos ver.



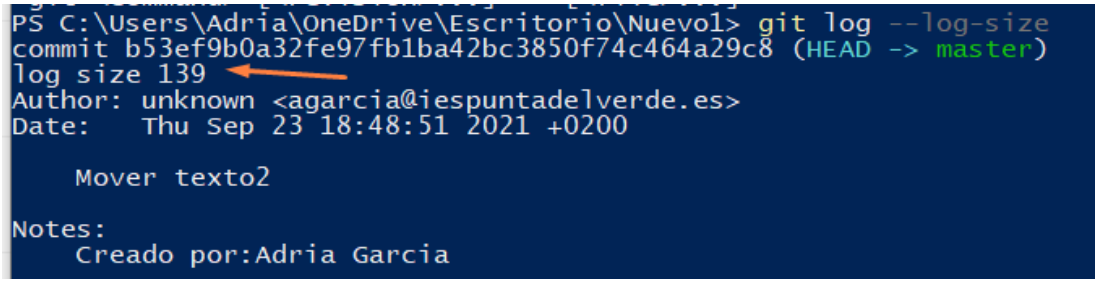
También podremos examinar el historial de un archivo viendo los commits que lo han llegado a modificar mediante el comando "**git log --folow <archivo>"**



Estas dos formas de utilizar el comando "**git log**" son las que hallo más importantes pero aquí dejo otros parámetros que pueden ser útiles:

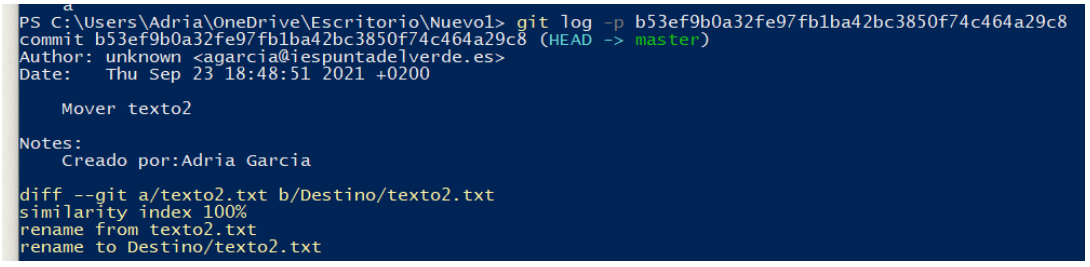
- **git log --log-size <id>:**

  Simplemente agrega una variable extra que nos indicará el número de bytes del commit



- **git log -p <id>**:

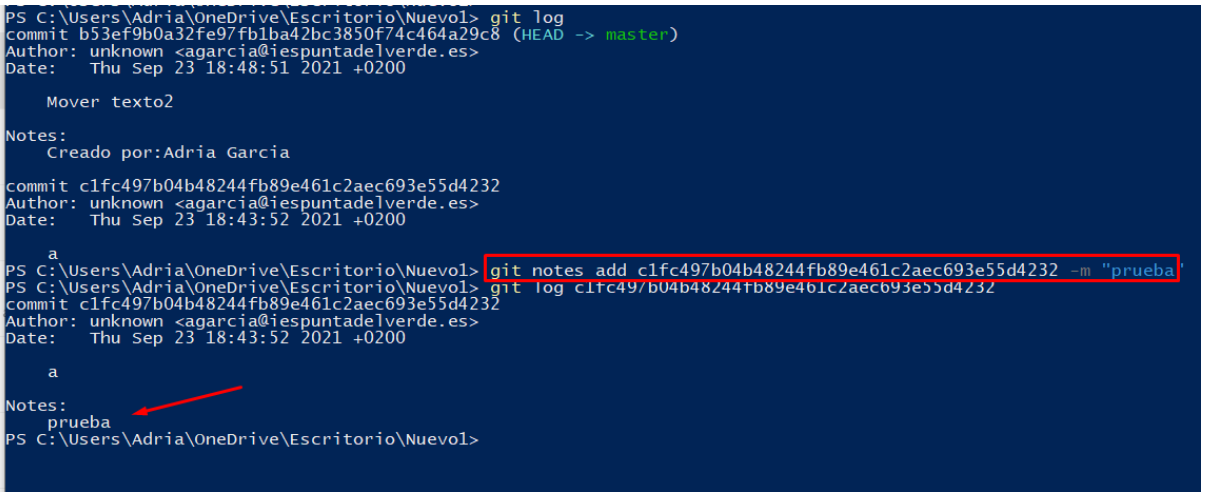
  Este comando examinará más a fondo lo que se realizó en cada commit actuando como un "diff", en este ejemplo vemos como nos indica que en el commit dado modificamos el nombre del archivo



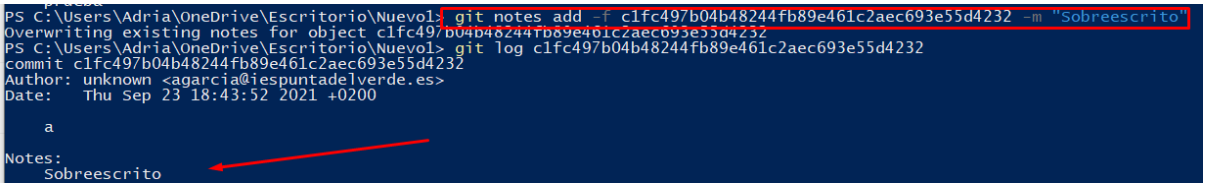
e. Git notes

En el apartado anterior hemos visto que mediante el comando "**git log**" podemos ver los commits que hemos realizado, sin embargo, en un proyecto en el que se hagan muchos commits puede resultar difícil reconocer que se hace únicamente con el commit, con las notas podremos etiquetar o añadir información adicional a nuestros commits.

Como vemos en este ejemplo, primero ponemos un "**git log**" para ver a que commit ponerle una nota, una vez hayamos elegido utilizaremos el comando "**git notes add <id commit> -m "nota"**", una vez efectuado el comando, haremos "**git log <id commit>**" para asegurarnos de que hemos realizado la nota con éxito.

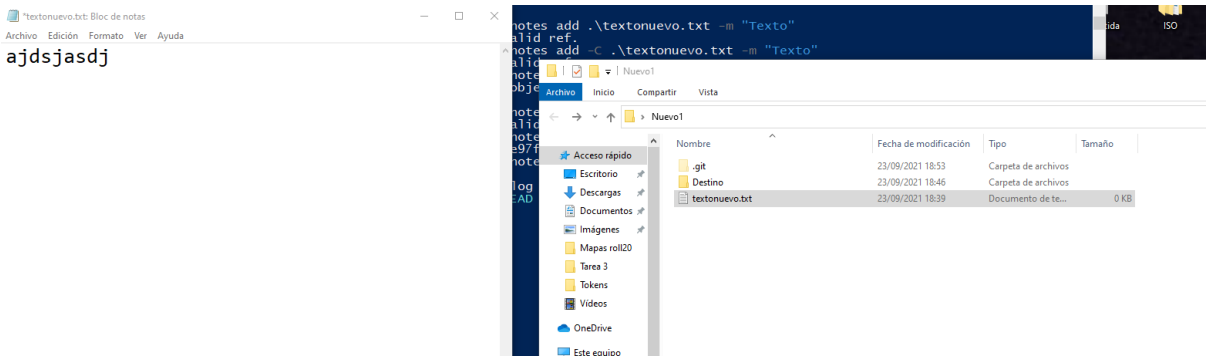


En caso de querer poner una nota a un commit que ya tiene, tendremos 2 formas de hacerlo, una primera mediante el comando **"git notes remove <id commit>"**, o una segunda mediante **"git notes add -f <id commit> -m "mensaje",** mediante el parámetro **"-f"** es con el que sobreescribiremos la nota anterior.

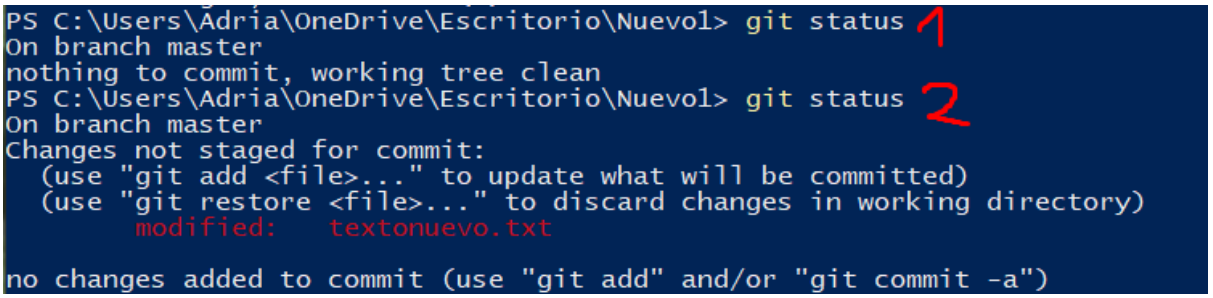


f. Git restore

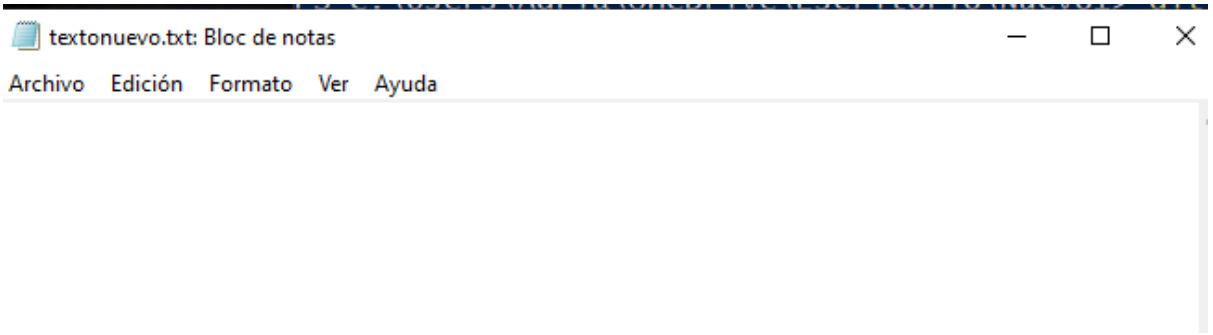
Con el comando “**git restore**” podremos volver para atrás a un cambio que hayamos realizado, por ejemplo, vamos a modificar uno de los archivos de nuestro repositorio



Al realizar un **Git status** veremos como nos aparece un aviso de que o bien añadimos de nuevo el archivo al repositorio o bien lo devolvemos a su estado anterior. Entre el “**git status**” 1 y el segundo hemos realizado la modificación del archivo.



Al realizar el comando **“git restore .\textonuevo.txt”** lo escrito anteriormente en el bloc de notas desaparecerá.

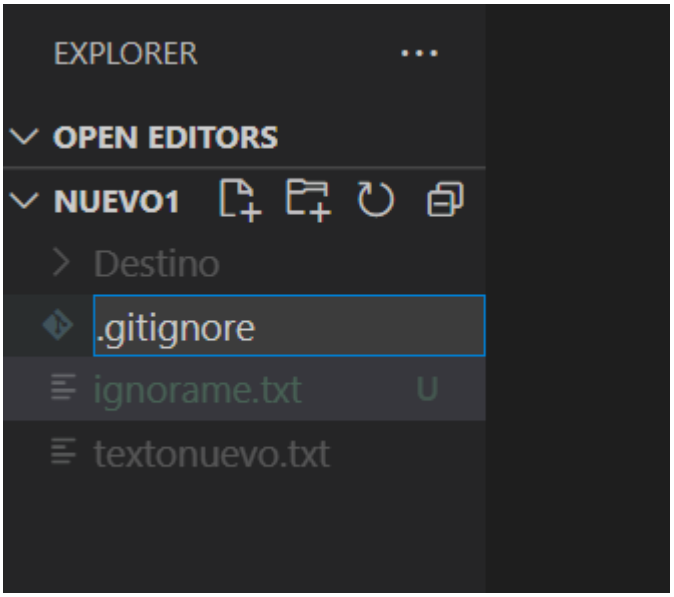


g. .gitignore

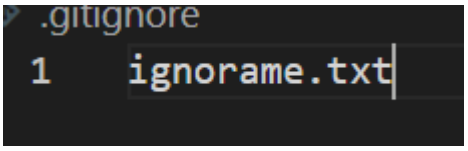
Este en concreto no es un comando, sino un archivo que incluiremos en la carpeta contenedora del repositorio. La finalidad de este archivo es simplemente agregar el nombre de los directorios o archivos que queremos que el repositorio ignore.

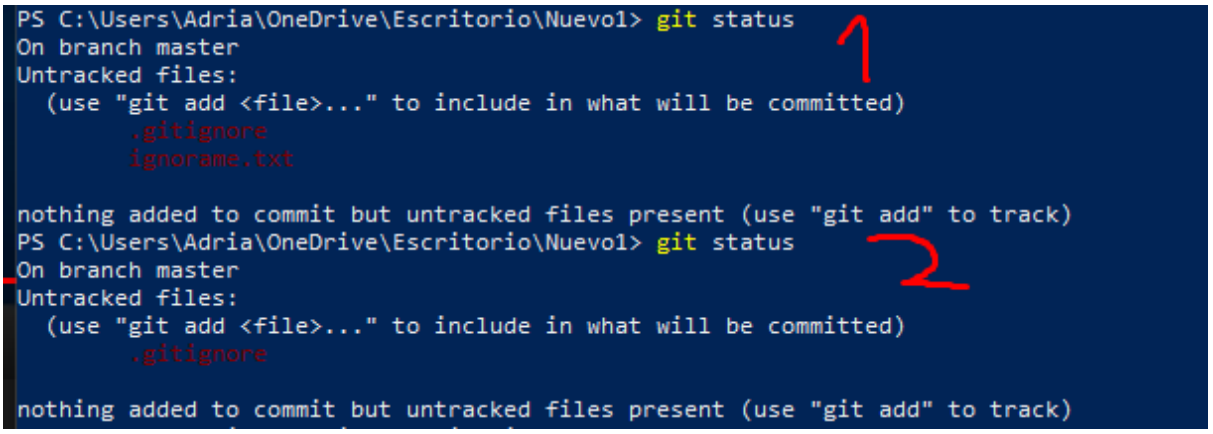
Este archivo es muy útil por ejemplo, cuando tenemos archivos que hemos empleado para hacer tests por ejemplo, y por tanto preferimos que no esté en el repositorio en sí.

En el siguiente ejemplo el cual considero el más visual, nuestro repositorio detectará que hay una carpeta “ignorame” la cual está detectando y nos pide que la agreguemos.
Sin embargo, crearemos un archivo .gitignore en el cual introduciremos el nombre de la carpeta “ignorame” y veremos que desaparecerá la advertencia.

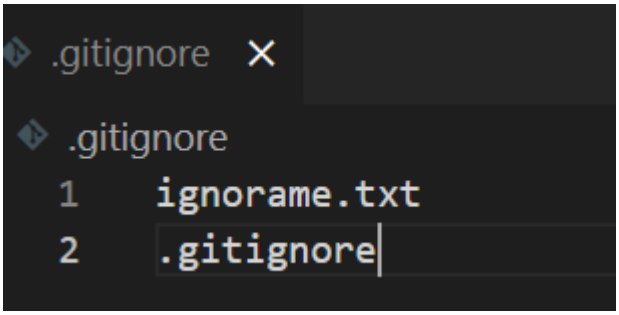


Ahora, haremos git status, un primer git status con el archivo .gitignore vacío y un segundo habiendo introducido en él el nombre del archivo a ignorar.





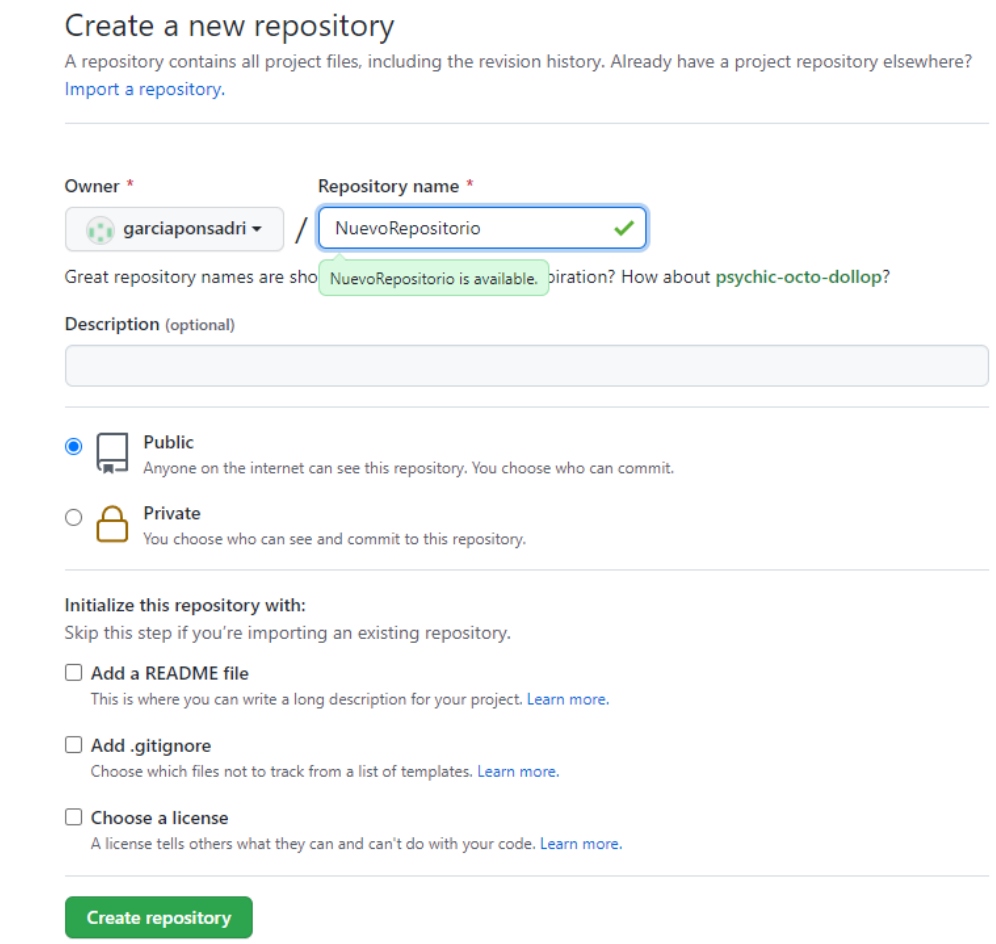
Para evitar que salga el gitignore al hacer el status podremos ponerle a él mismo dentro del archivo de .gitignore. Veremos en el segundo “**git status**”, como ya no sale el .gitignore



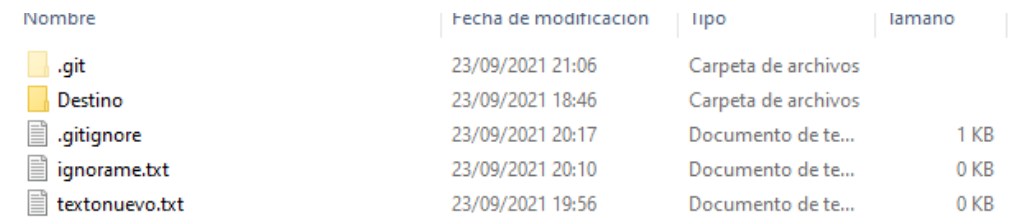

```
PS C:\Users\Adria\OneDrive\Escritorio\Nuevo1> git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\Adria\OneDrive\Escritorio\Nuevo1> git status
On branch master
nothing to commit, working tree clean
```

h. Git pull

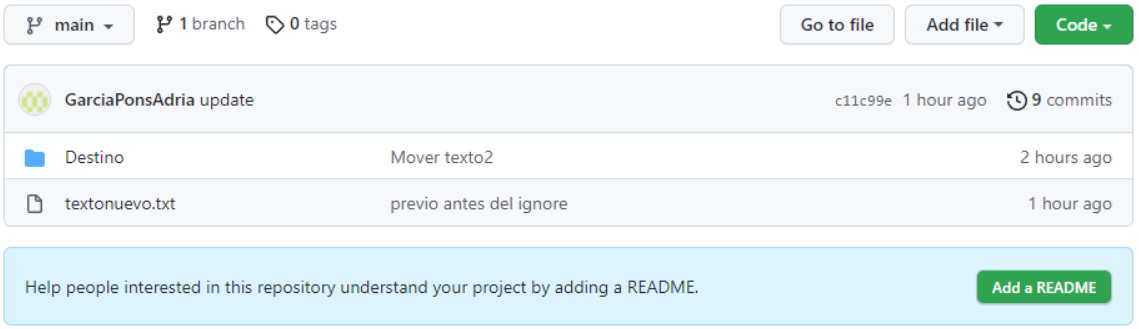
Primero de todo crearemos un repositorio vacío de Github y mediante un push exportaremos un repositorio de git a la web.



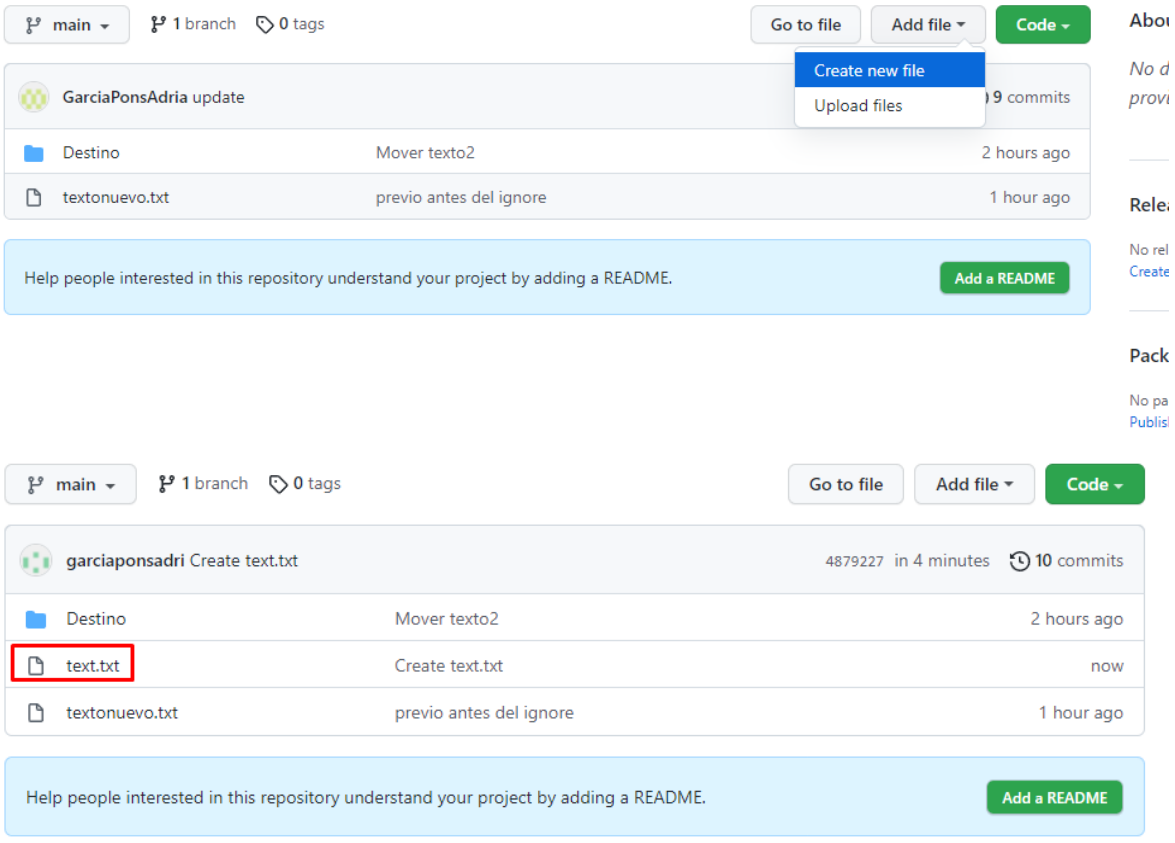

```
PS C:\Users\Adria\OneDrive\Escritorio\Nuevo1> git remote remove origin
PS C:\Users\Adria\OneDrive\Escritorio\Nuevo1> git remote add origin https://github.com/garciaponsadri/NuevoRepositorio
PS C:\Users\Adria\OneDrive\Escritorio\Nuevo1> git branch main
fatal: A branch named 'main' already exists.
PS C:\Users\Adria\OneDrive\Escritorio\Nuevo1> git push origin main
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 4 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (20/20), 1.78 KiB | 455.00 KiB/s, done.
Total 20 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), done.
To https://github.com/garciaponsadri/NuevoRepositorio
 * [new branch]      main -> main
PS C:\Users\Adria\OneDrive\Escritorio\Nuevo1>
```



Nombre	Fecha de modificación	Tipo	Tamaño
.git	23/09/2021 21:06	Carpeta de archivos	
Destino	23/09/2021 18:46	Carpeta de archivos	
.gitignore	23/09/2021 20:17	Documento de te...	1 KB
ignorame.txt	23/09/2021 20:10	Documento de te...	0 KB
textonuevo.txt	23/09/2021 19:56	Documento de te...	0 KB

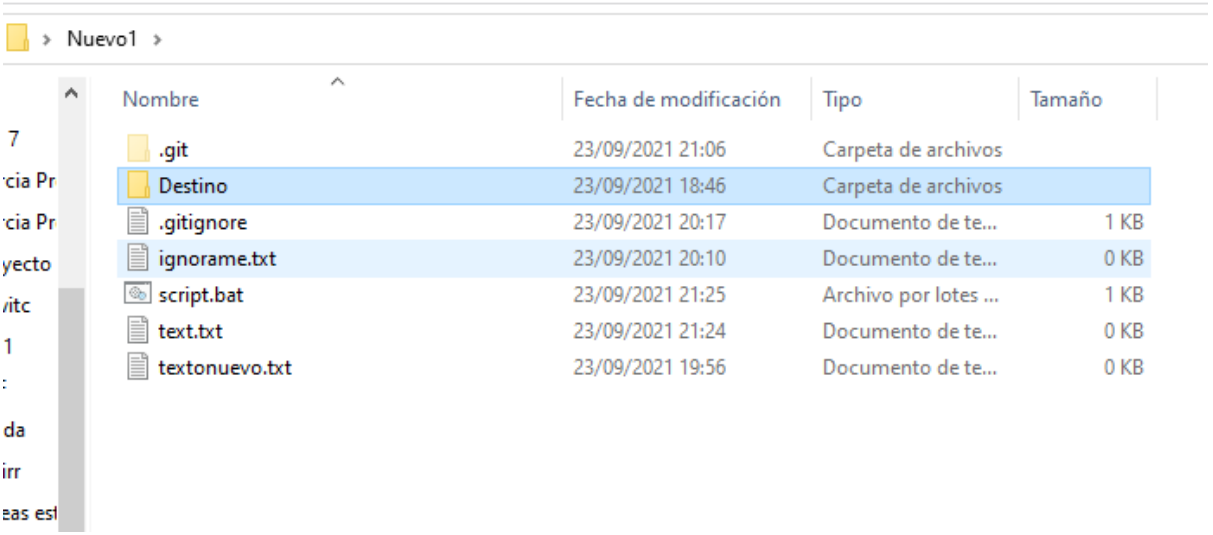


Tras esto, agregaremos un nuevo archivo lo cual provocará cambios en nuestro repositorio



Vamos al shell y tras haber realizado el cambio haremos “**git pull**” con el cual veremos como se agrega el nuevo archivo que hemos agregado.

```
remote: Enumerating objects: 4, done.
remote: Counting objects: 100 (4/4), done.
remote: Compressing objects: 100 (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100 (3/3), 693 bytes | 69.00 KiB/s, done.
From https://github.com/garciaponsadri/NuevoRepositorio
        c11c99e..4879227  main       -> origin/main
Fast-forward
text.tx | 1 +
1 file changed, 1 insertion(+)
created mode 100644 text.txt
C:\Users\Adria\OneDrive\Escritorio\Nuevo1>
```



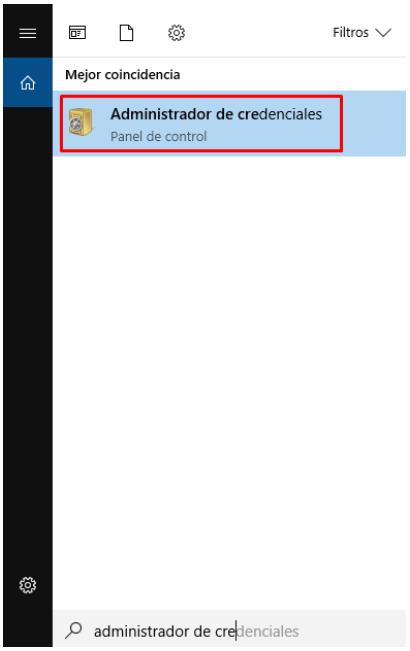
Teoría sobre la práctica y resolución de incidentes

1. Guía de comandos

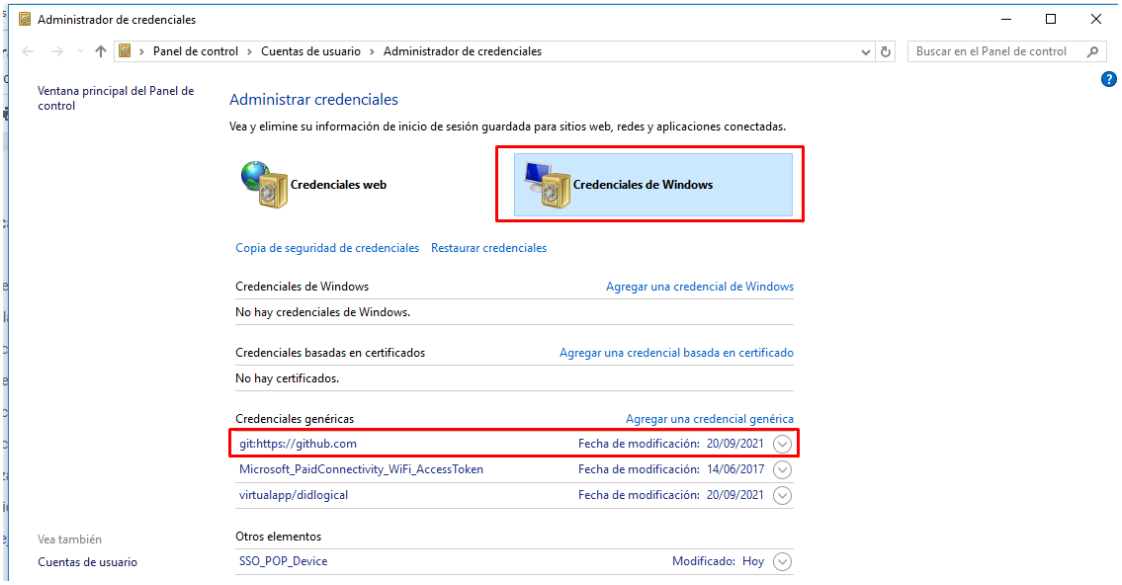
- **git init**: Inicializas un repositorio de GIT en la ubicación que le hayas puesto mediante cd en powershell
- **git status:** Te dice el estado actual del repositorio, es muy útil ya que nos dirá si tenemos algo por comittear, nos avisará si no tenemos archivos en el repositorio, etc.
- **git add <archivo>**: Agrega al repositorio un archivo (en caso de estar en la misma carpeta bastará poner el nombre, si está en otra poner la url + nombre archivo).
  Si utilizamos **git add .**, agregaremos al repositorio todos los elementos del directorio en que nos encontremos.
- **git commit** - Actualizamos el repositorio con los cambios que hayas realizado
- **git config --global <variable>** - Agregas las credenciales al git (las cuales deberán ser igual que en github), user.name para nombre de usuario y user.email para el correo eléctronico
- **git branch <tipo branch>** - Nos permitirá cambiar de rama, según el tipo de Branch que utilicemos tendremos unas funcionalidades u otras.
- **git remote add <repositorio2> <link>** - agregas el link del repositorio de github y le asignas una variable <nombre>
- **git push -u <repositorio1> <repositorio2>** - Realizas un push de <repositorio1> (el repositorio que hemos hecho GIT) al repositorio remoto (El repositorio de github)
- **git rm** - Nos permitirá eliminar un fichero o directorio de nuestro repositorio
- **git mv** - Podremos cambiar de directorio y de nombre a archivos y directorios de nuestro repositorio.
- **git clone** - Podremos clonar repositorios web (de github) o locales permitiéndonos por ejemplo crear copias de seguridad.
- **git pull** - Nos permitirá mediante una conexión entre un repositorio de Github y uno de Git realizar funciones pull con el fin de importar/exportar los archivos que se agreguen o modifiquen
- **git push** - Con un push podremos subir un repositorio por ejemplo local a la web por medio de un repositorio vacío de Github.
- **git log** - Gracias a este comando podremos ver el historial de commits que hemos realizado.
- **git notes** - Nos permitirá crear notas para cada uno de los commits con el fin de tener una mayor organización sobre nuestro repositorio.
- **.gitignore** - No es un comando sino un documento en el que pondremos los nombres de los archivos y directorios que queremos que git ignore.
- **git restore** - Nos permitirá revertir un cambio que hayamos realizado antes de realizar el correspondiente “git add /rm <archivo>” comando tras el cual perderíamos la posibilidad de restaurar una versión anterior de nuestro archivo/directorio.

2. Manejo de credenciales

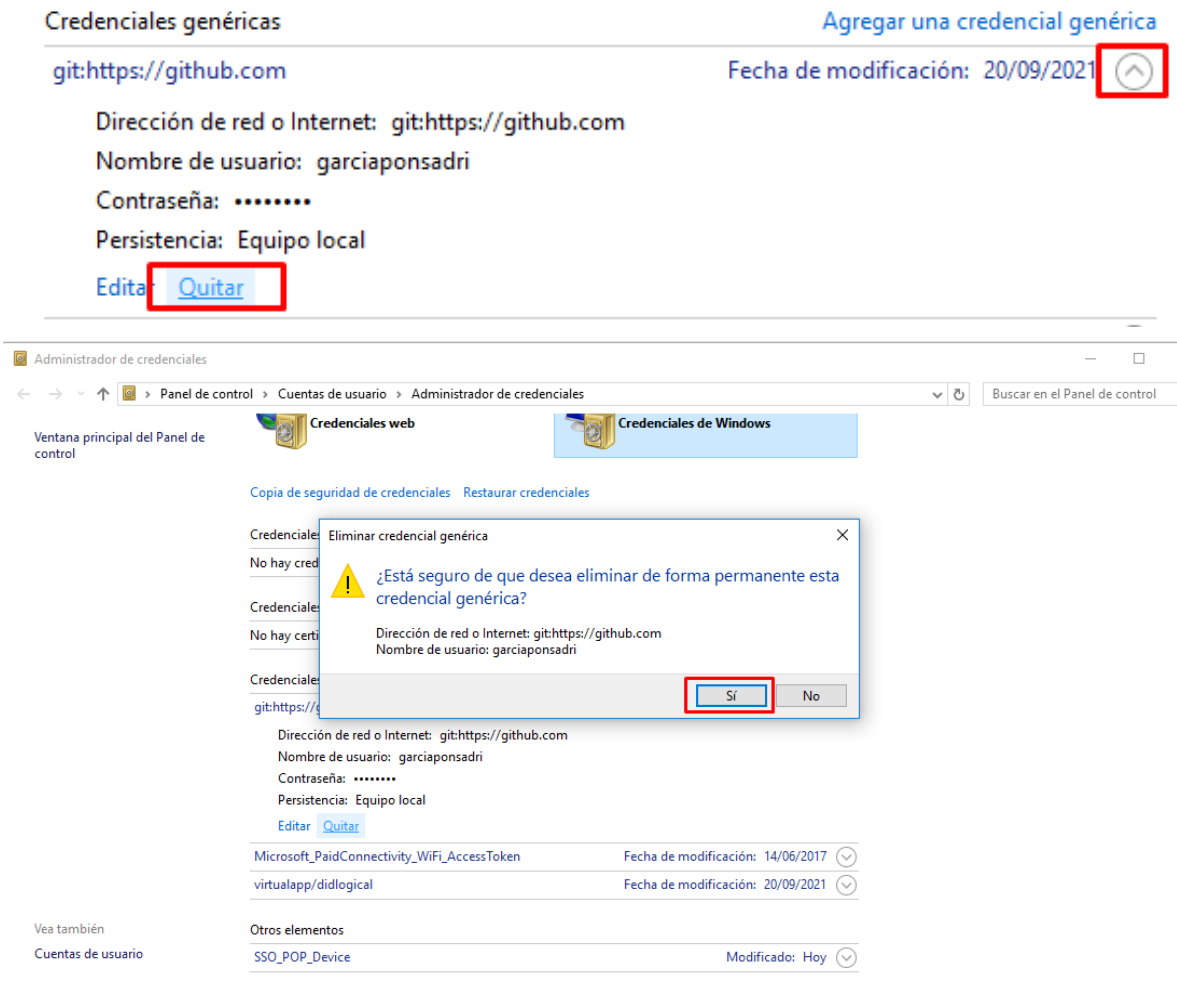
En git, una vez realicemos el primer commit nos pedirá las credenciales y vinculará la cuenta de github con nuestro git, sin embargo, en caso de querer vincular una cuenta distinta deberemos ir al administrador de credenciales de windows



Dentro del administrador de credenciales daremos click sobre “Credenciales de Windows” y nos saldrá una lista, dentro de la cual, si tenemos vinculada una cuenta de Github con la de Git (como es mi caso tras realizar la práctica anterior), podremos desloggear la cuenta.



Una vez clickeemos sobre las credenciales de Git-Github se nos desplegará esta ventana en la cual tendremos la opción de quitar tal vinculación



Una vez eliminadas las credenciales repetiremos el paso de la práctica de “enlazamiento de cuenta de Github con Git” para vincular las cuentas.

3. Resolución de Incidentes

Para la creación de la **cuenta de github** deberemos tener en cuenta para evitar incidentes:

- El username es único en toda la plataforma, un username = un usuario.
- Dentro de nuestra cuenta no podremos repetir los nombres de los repositorios.
- Para realizar un push deberemos haber creado un repositorio en github (aunque vacío).

Para el manejo de **credenciales** y **vinculación de GIT-Github** deberemos tener en cuenta:

- Cuando nos pida las credenciales tendremos que poner nuestro username y emails exactos para una correcta vinculación.
- En caso de equivocarnos al introducir las credenciales antes de llevar a cabo la vinculación de cuentas (antes de realizar el primer push), podremos simplemente cambiar las variables sin necesidad de ir a la administrador de credenciales (Como se explica en el apartado de manejo de credenciales).

Motivos varios:

- Cuando hagamos un "**Git notes add**" por ejemplo a un commit puede ser que nos diga que ha habido un error ya que existe una nota, podremos solucionarlo con el parámetro "**-f**" antes de poner el código del commit.
- Cuando realicemos un **"git remote add origin <url>"** nos puede dar error en ocasiones porque exista ya un valor para esa variable origin, simplemente haremos un **"git remote remove origin"** y ya podremos agregar la nueva URL sin problemas.
- Al realizar un **"git push"** en caso de que el repositorio de Github tenga contenido podremos forzar que se realice este push mediante el parámetro **"-f"**