Práctica 1: GIT y GITHUB

1º ¿Qué es git y github?

2º Instalación de git

3º Crear usuario y repositorio en GITHUB

4º Caso práctico

5º Conclusiones

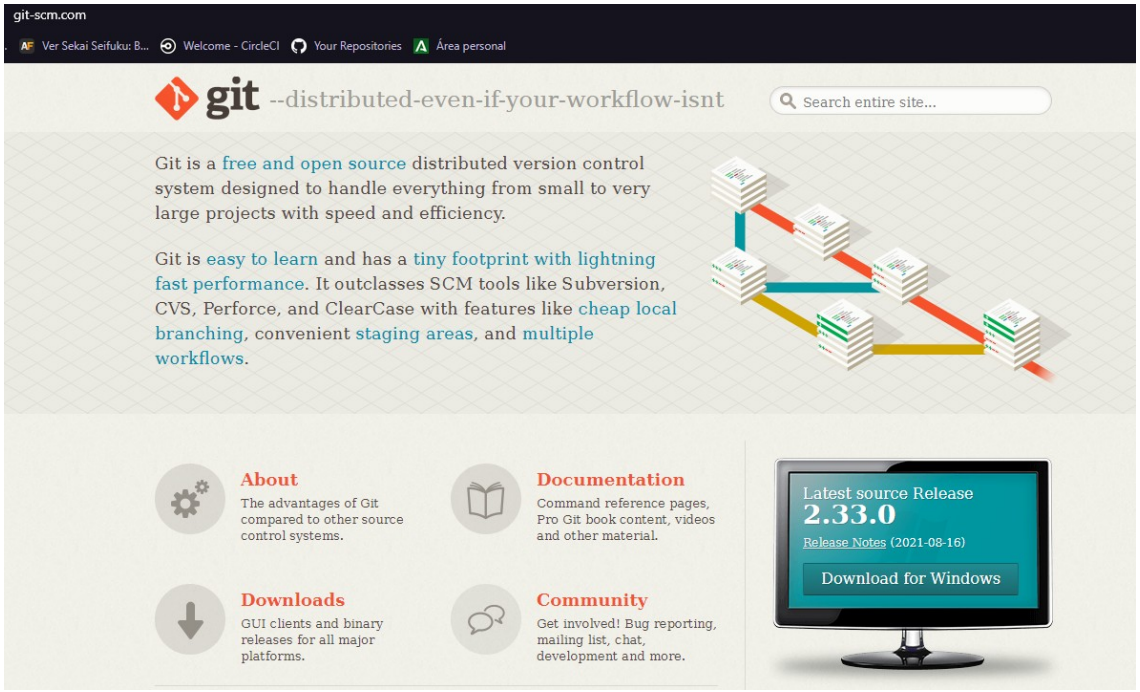
1º ¿Qué es git y github?

Git es un sistema de control de versiones, que en vez de tener un único espacio para el historial de versiones del software, con Git, la copia de trabajo del código de cada desarrollador es también un repositorio que puede albergar el historial completo de todos los cambios.

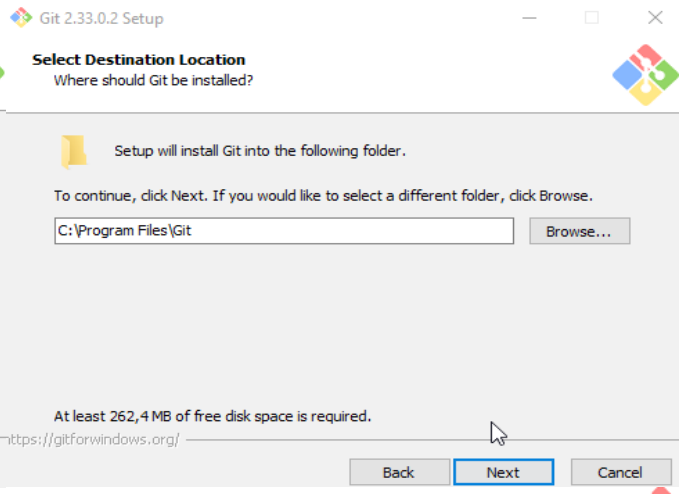
Github por otro lado es un entorno en la nuve donde se pueden almacenar dichos repositorios creados a través de Git, para poder acceder a ellos desde el equipo que deseemos e incluso que compañeros que trabajan en un proyecto común puedan aportar sus líneas al código con el que se trabaja, sin alterar lo que ya existe, pudiendo crear muchas versiones del programa hasta llegar a la versión óptima.

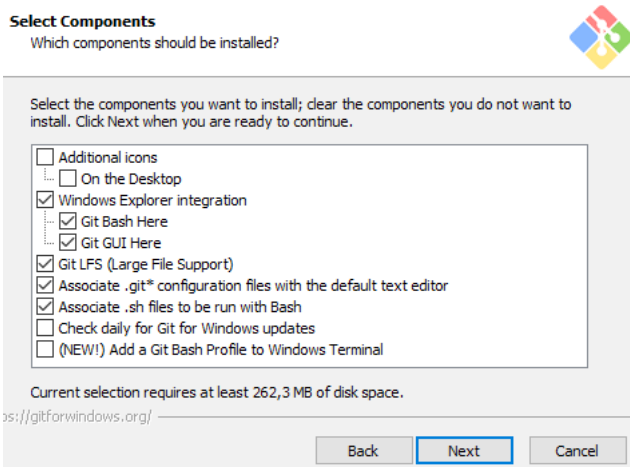
2º Instalación de git

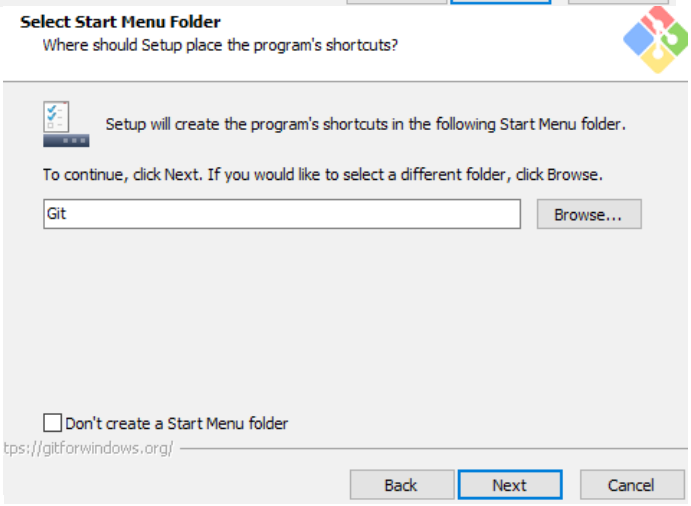
Primero debemos instalar git, para poder hacer uso de sus funciones y comandos, para ello accedemos a la siguiente url: https://git-scm.com y hacemos clic en “Download for Windows”.

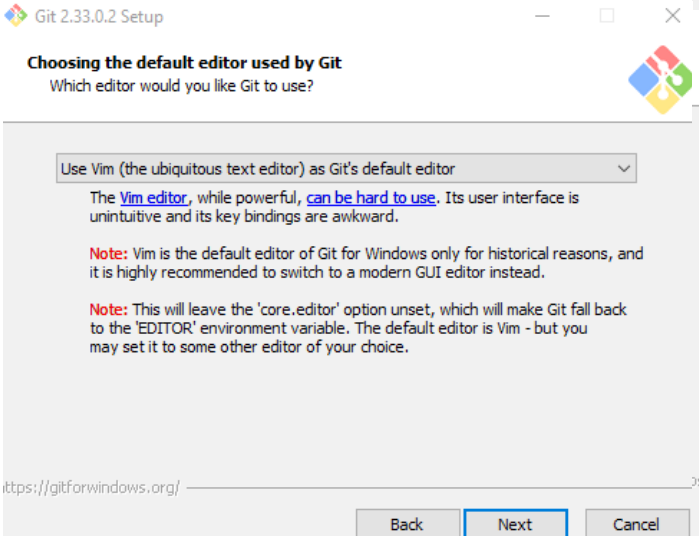


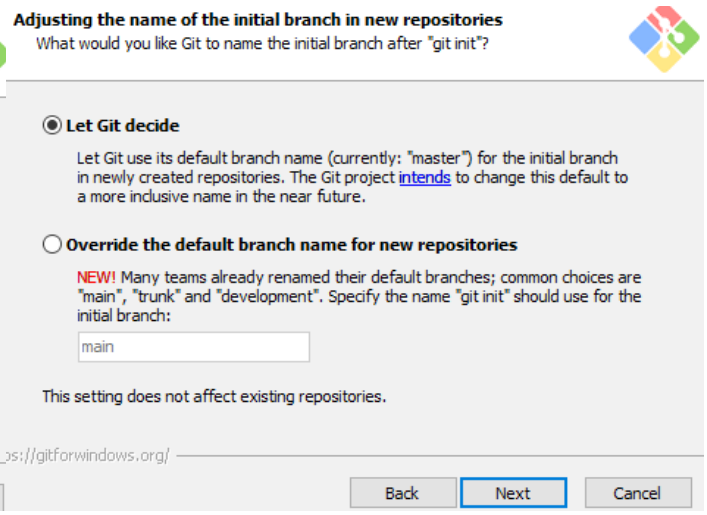
Ahora ejecutamos el instalador, y empezamos a ver las configuraciones de git a la hora de instalar. Vamos a darle siguiente a todo “Next”, ya que queremos las configuraciones predeterminadas.

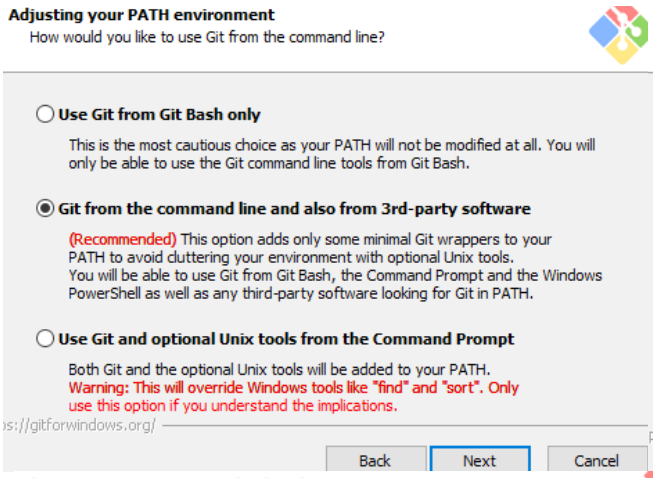


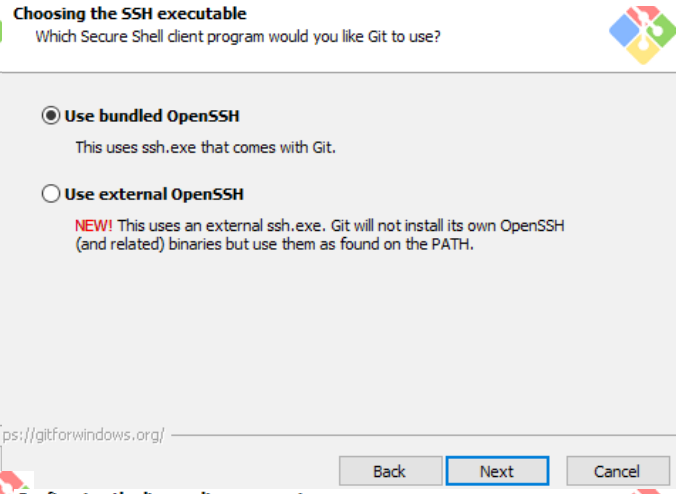


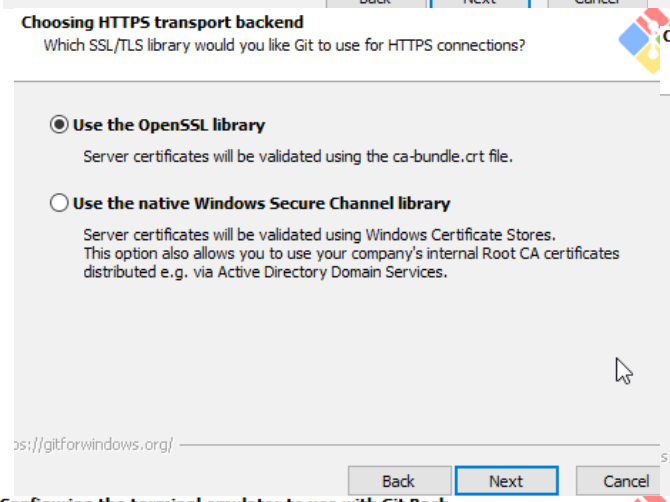


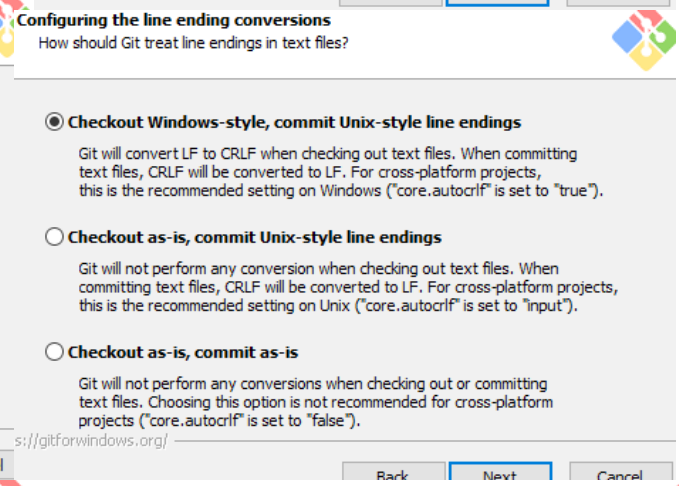


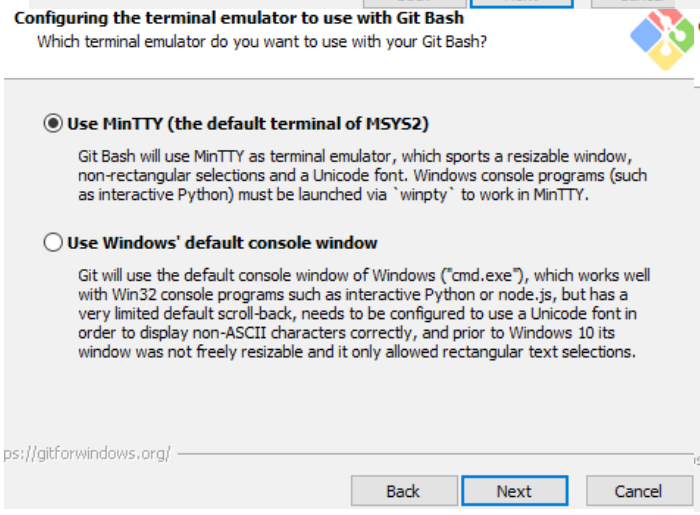


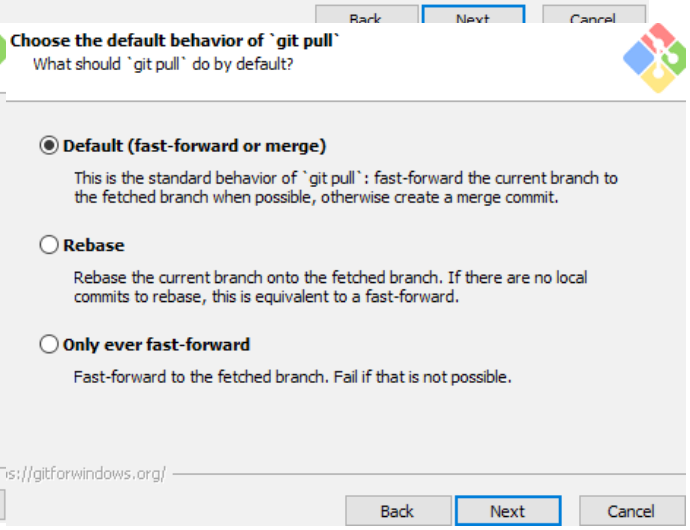

Adjusting your PATH environment
How would you like to use Git from the command line?
Use Git from Git Bash only
This is the most cautious choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.
Git from the command line and also from 3rd-party software
(Recommended) This option adds only some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools.
You will be able to use Git from Git Bash, the Command Prompt and the Windows PowerShell as well as any third-party software looking for Git in PATH.
Use Git and optional Unix tools from the Command Prompt
Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.
Back
Next
Cancel

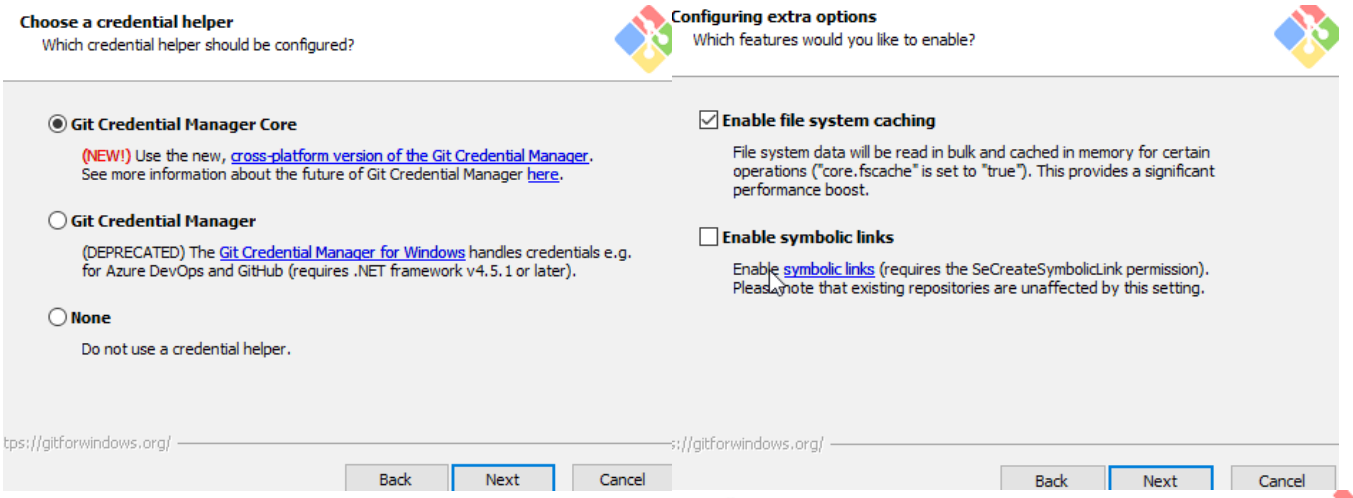

Choosing the SSH executable
Which Secure Shell client program would you like Git to use?
Use bundled OpenSSH
This uses ssh.exe that comes with Git.
Use external OpenSSH
NEW! This uses an external ssh.exe. Git will not install its own OpenSSH (and related) binaries but use them as found on the PATH.
Back
Next
Cancel

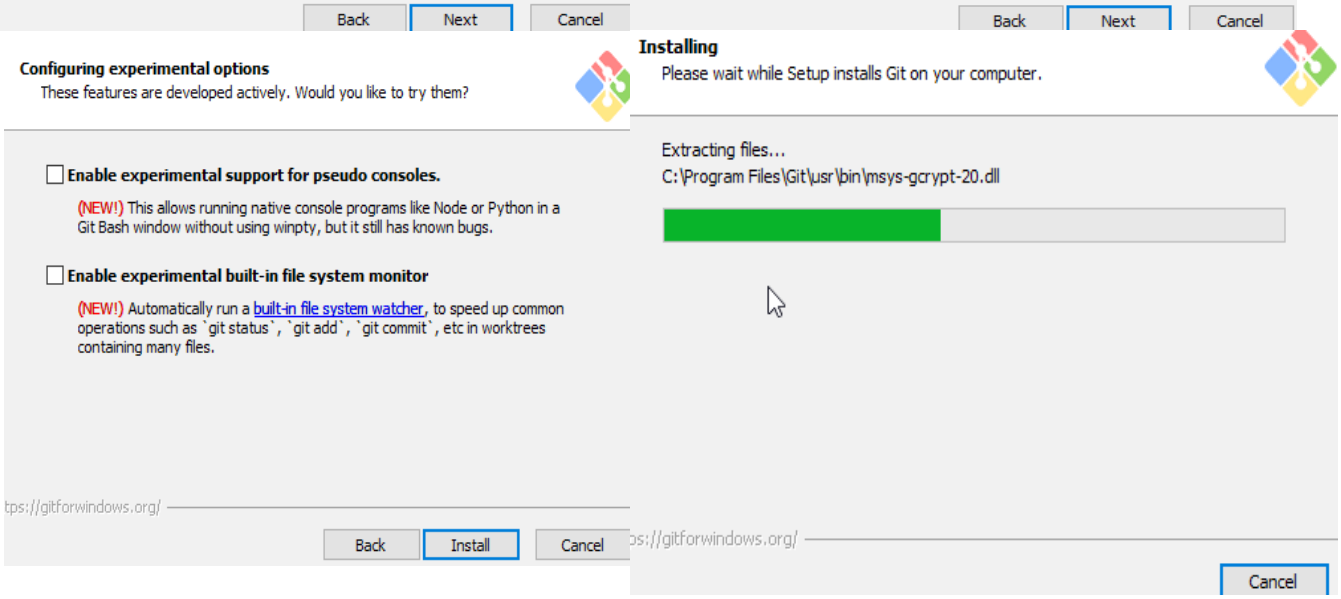

Choosing HTTPS transport backend
Which SSL/TLS library would you like Git to use for HTTPS connections?
Use the OpenSSL library
Server certificates will be validated using the ca-bundle.crt file.
Use the native Windows Secure Channel library
Server certificates will be validated using Windows Certificate Stores. This option also allows you to use your company's internal Root CA certificates distributed e.g. via Active Directory Domain Services.
Back
Next
Cancel


Configuring the line ending conversions
How should Git treat line endings in text files?
Checkout Windows-style, commit Unix-style line endings
Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").
Checkout as-is, commit Unix-style line endings
Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").
Checkout as-is, commit as-is
Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").
Back
Next
Cancel

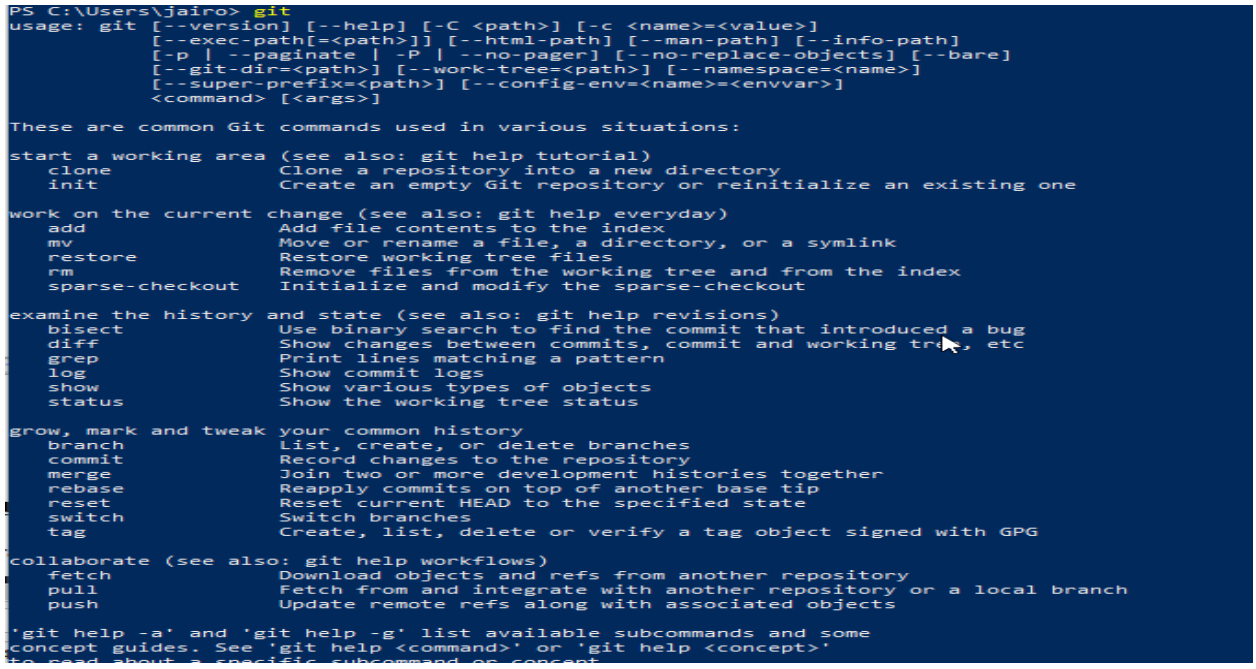

Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?
Use MinTTY (the default terminal of MSYS2)
Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.
Use Windows' default console window
Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.
Back
Next
Cancel


Choose the default behavior of `git pull`
What should `git pull` do by default?
Default (fast-forward or merge)
This is the standard behavior of `git pull`: fast-forward the current branch to the fetched branch when possible, otherwise create a merge commit.
Rebase
Rebase the current branch onto the fetched branch. If there are no local commits to rebase, this is equivalent to a fast-forward.
Only ever fast-forward
Fast-forward to the fetched branch. Fail if that is not possible.
Back
Next
Cancel



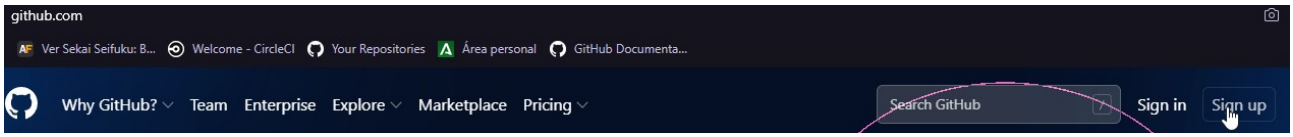


Ya hemos instalado git, para comprobar que se ha instalado nos vamos a un terminal (en mi caso utilizaré powershell) y usamos el comando git, lo cual nos mostrará las opciones de ayuda del comando, así confirmamos que se ha instalado.



3º Crear usuario y repositorio en GITHUB

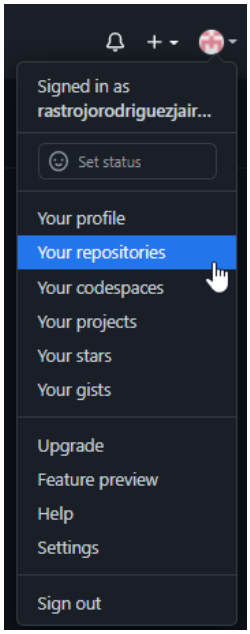
Primero accedemos a la página https://github.com y clicamos en “Sign up” para crearnos una cuenta

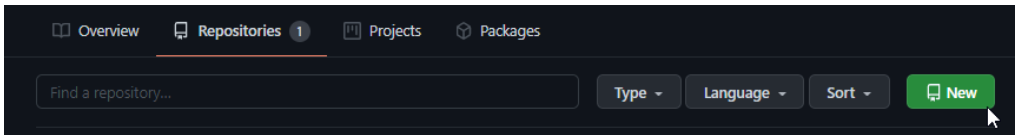


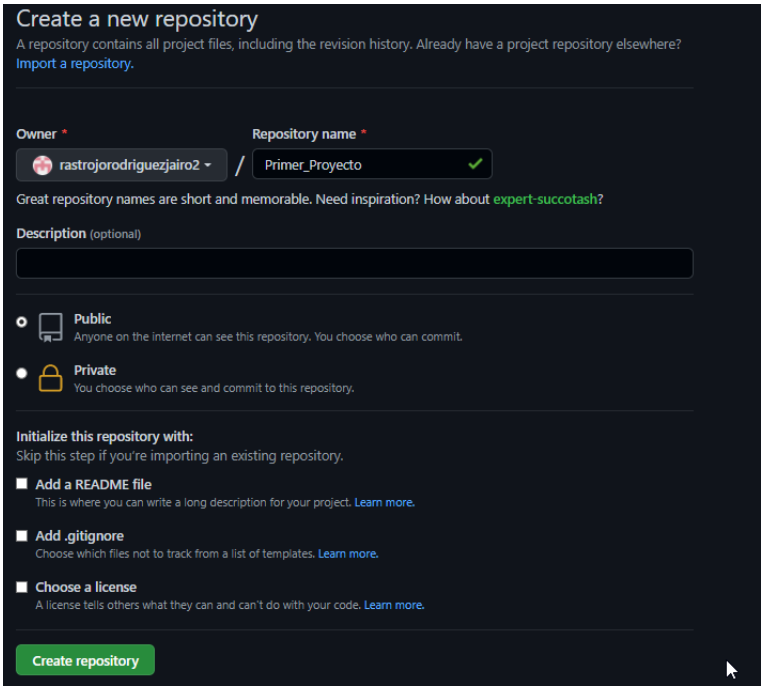
Tras lo cual pondremos un correo que queramos que sea el enlazado con nuestra cuenta, también designamos una contraseña para la cuenta, elegimos un nombre de cuenta y tambien podemos permitir o denegar publicidad sobre los cambios en github, por ultimo deberemos verificar la cuenta resolviendo un rompecabezas sencillo para demostrar que no somos robots.

ATENCIÓN. La cuenta que usaremos para esta práctica, no es la que se ve en la imagen, usaremos la cuenta enlazada con el correo corporativo de la junta de Andalucía.

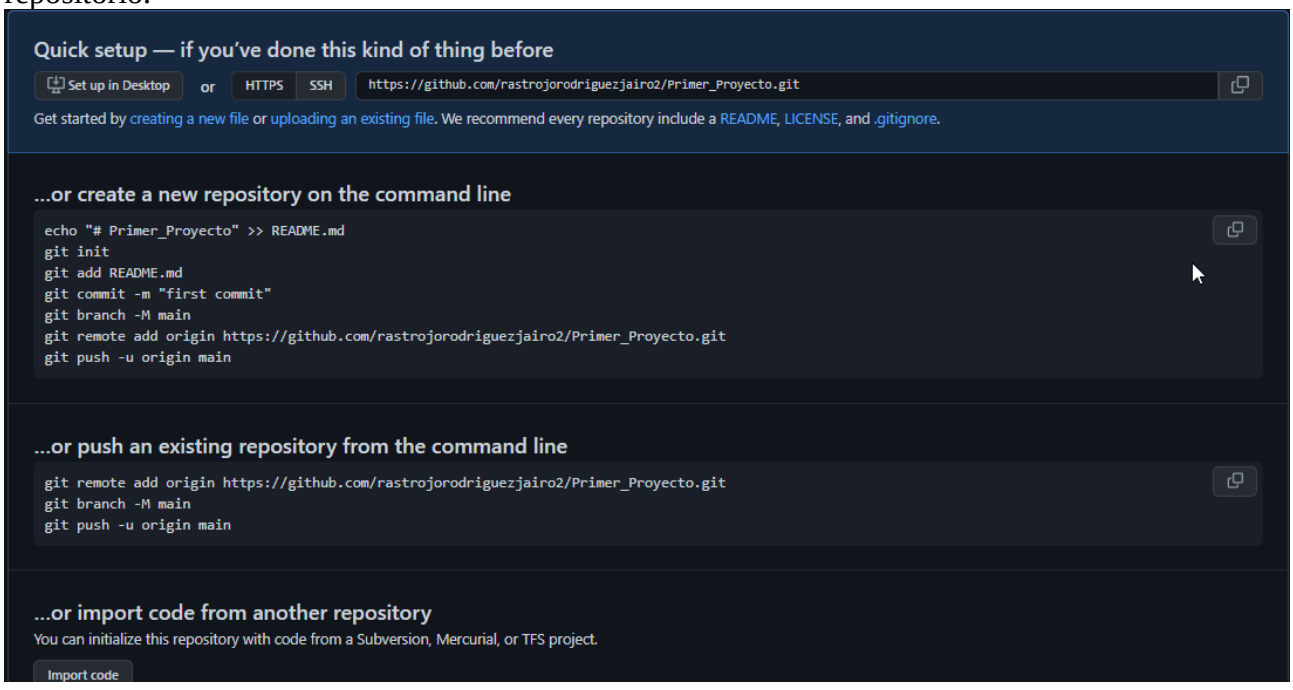
Una vez logeados en Github, vamos a crear un nuevo repositorio, haciendo clic en la esquina superior derecha, donde el menú desplegable, mostrara “Your repositories” en el cual clicamos y a continuación le damos a “New”, tras lo cual pasaremos a un apartado donde podremos especificar el nombre de nuestro repositorio y también el nivel de seguridad.







Tras lo cual le damos a “Create repository”, lo que nos mostrará el link del repositorio, que necesitaremos a continuación y los pasos a seguir para subir el contenido que queremos en el repositorio.



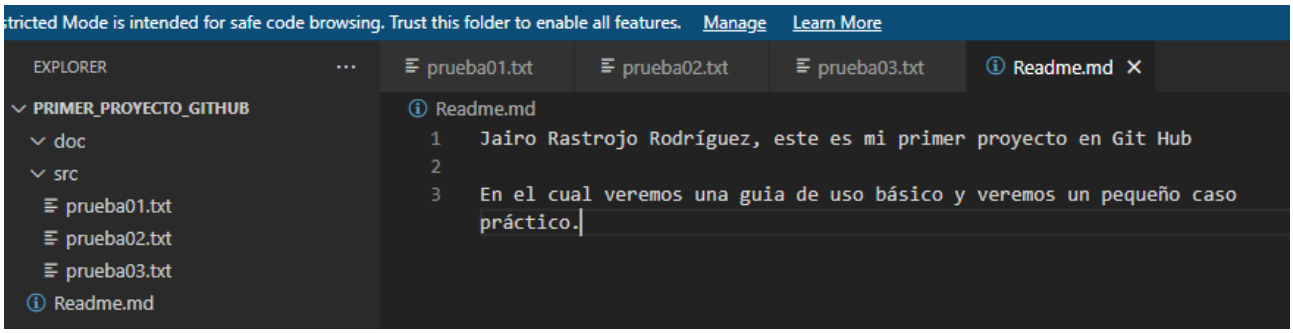
4º Caso práctico

En este caso práctico vamos a empezar por crear una carpeta que será la elegida para ser convertida en repositorio local y subida junto a su contenido.

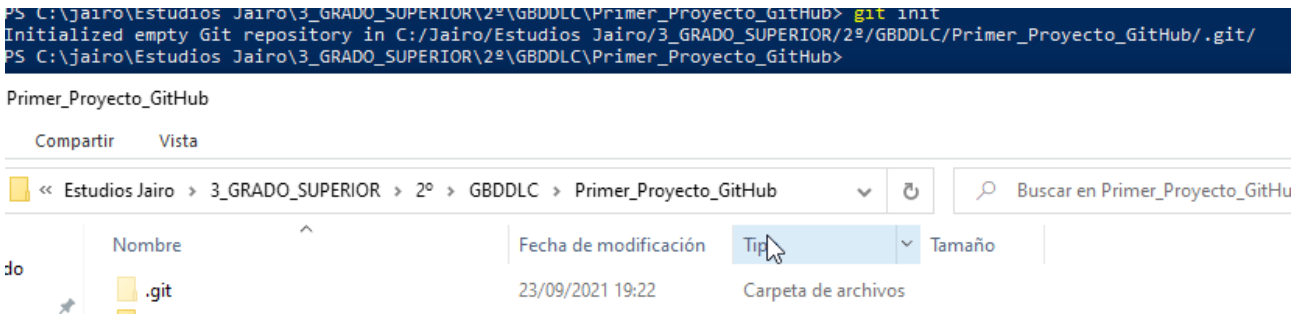
Usaremos en esta práctica los comandos necesariois para crear la carpeta “Primer_Proyecto_GitHub”.

```
PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC> mkdir Primer_Proyecto_GitHub


    Directorio: C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        23/09/2021     19:10                Primer_Proyecto_GitHub


PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC> cd Primer*
PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\Primer_Proyecto_GitHub> nano Prueba01
```

Creamos dentro de dicha carpeta la estructura que queremos logar, para ello usaremos “Visual Estudio Code”. Primero abrimos la carpeta recién creada y le creamos las carpetas src (dentro de ella 3 archivos de texto), otra carpeta con nombre doc (tendrá este PDF cuando este acabado) y un archivo Readme.md, para introducir el proyecto.



Usando de nuevo el powershell, vamos a ejecutar el comando “git . init”, lo cual genera en el directorio en el que estamos situados un archivo oculto llamado “.git” que acaba de convertir nuestra carpeta en un repositorio local.



A continuación ejecutamos el comando “git status” para comprobar si los archivos que se encuentran en el repositorio están añadidos, en este caso no lo están.

```
PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\Primer_Proyecto_GitHub> git init
Initialized empty Git repository in C:/Jairo/Estudios Jairo/3_GRADO_SUPERIOR/2º/GBDDLC/Pr
PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\Primer_Proyecto_GitHub> git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Readme.md
        src/
```

Para que se añadan usaremos el comando “git add .” y volvemos a lanzar “git status”, comprobando que ahora si vamos bien.

```
PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\Primer_Proyecto_GitHub> git add .
PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\Primer_Proyecto_GitHub> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Readme.md
        new file:   src/prueba01.txt
        new file:   src/prueba02.txt
        new file:   src/prueba03.txt
```

Ahora podremos pasar a usar el comando “git commit -m” seguido de entre comillas un comentario, este comando se usa para poner un título o comentario a todo lo que se subirá de este repositorio en esta subida, así podemos diferenciar las versiones que vamos subiendo.

```
PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\Primer_Proyecto_GitHub> git commit -m "Versión 0.1"
[master (root-commit) 94bd13c] Versión 0.1
 4 files changed, 3 insertions(+)
 create mode 100644 Readme.md
 create mode 100644 src/prueba01.txt
 create mode 100644 src/prueba02.txt
 create mode 100644 src/prueba03.txt
PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\Primer_Proyecto_GitHub>
```

Vamos a configurar tanto el correo que emplearemos en github como el nombre de usuario que lo maneja.

```
PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\Primer_Proyecto_GitHub> git config --global user.email "jrasrod934@g.educaand.es"
PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\Primer_Proyecto_GitHub> git config --global user.name "rastrojorodriguezjairo2"
PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\Primer_Proyecto_GitHub>
```

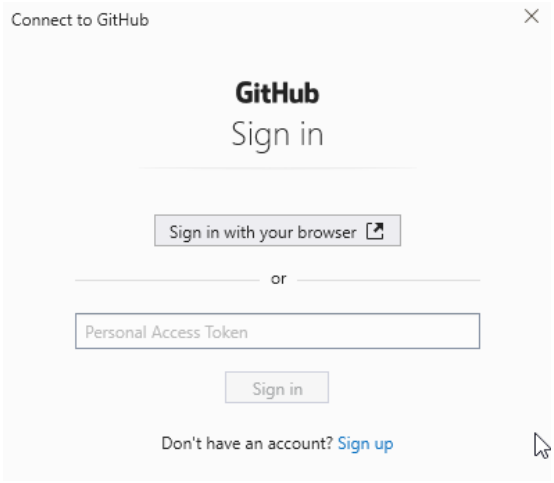
Ahora usaremos el comando “git branch -M main” que determina cual sera la ruta maestra de unión con el repositorio y el comando “git remote add origin” seguido de la url que vimos en la imagen a la hora de crear el repositorio en git hub, para que estén enlazados

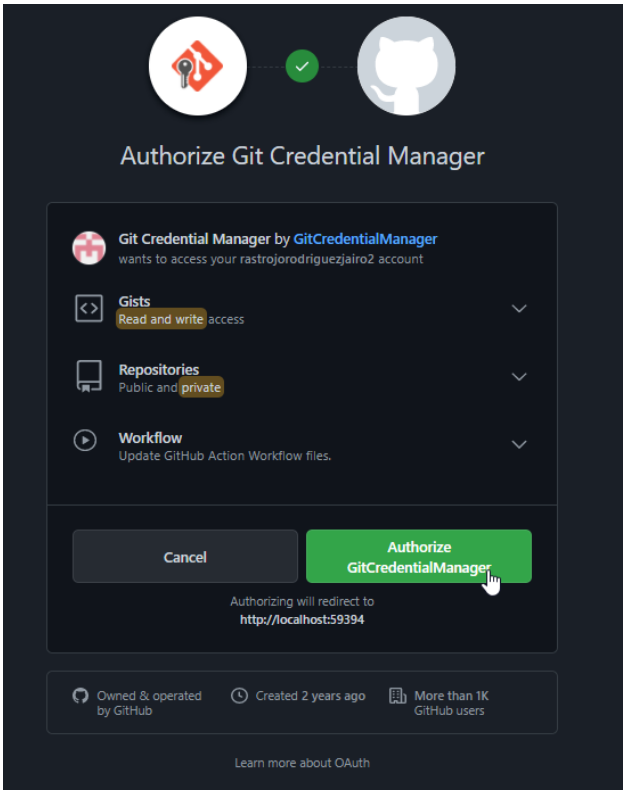
```
PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\Primer_Proyecto_GitHub> git branch -M main
PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\Primer_Proyecto_GitHub> git remote add origin https://git
hub.com/rastrojorodriguezjairo2/Primer_Proyecto.git
```

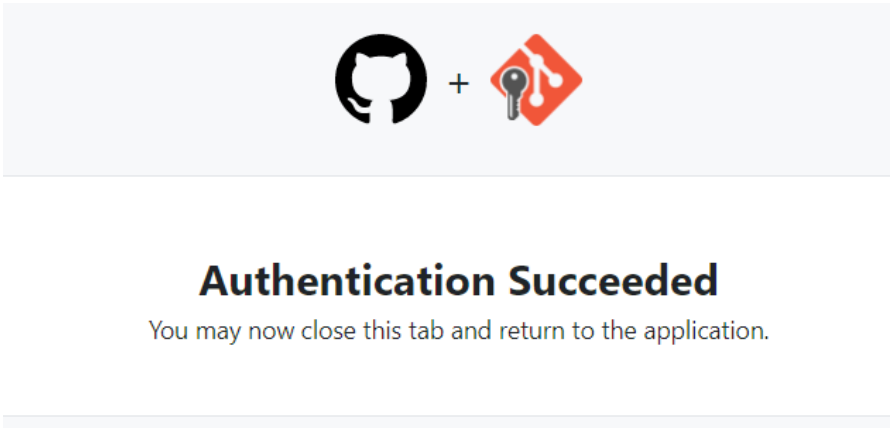
Ahora usaremos el comando “git push -u origin main” para que todo el contenido que haya en el repositorio en el que estamos se suba al repositorio remoto en GitHub.

```
PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\Primer_Proyecto_GitHub> git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 583 bytes | 291.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/rastrojorodriguezjairo2/Primer_Proyecto.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\Primer_Proyecto_GitHub>
```

Sin embargo la primera vez que hagamos esto nos saltará la siguiente ventana, en la que clicaremos en “Sign in with your browser”, lo cual nos abrirá el navegador y nos permitira autoricar que nos identifique Github con nuestras credenciales de windows, autorizamos y veremos la última imagen.

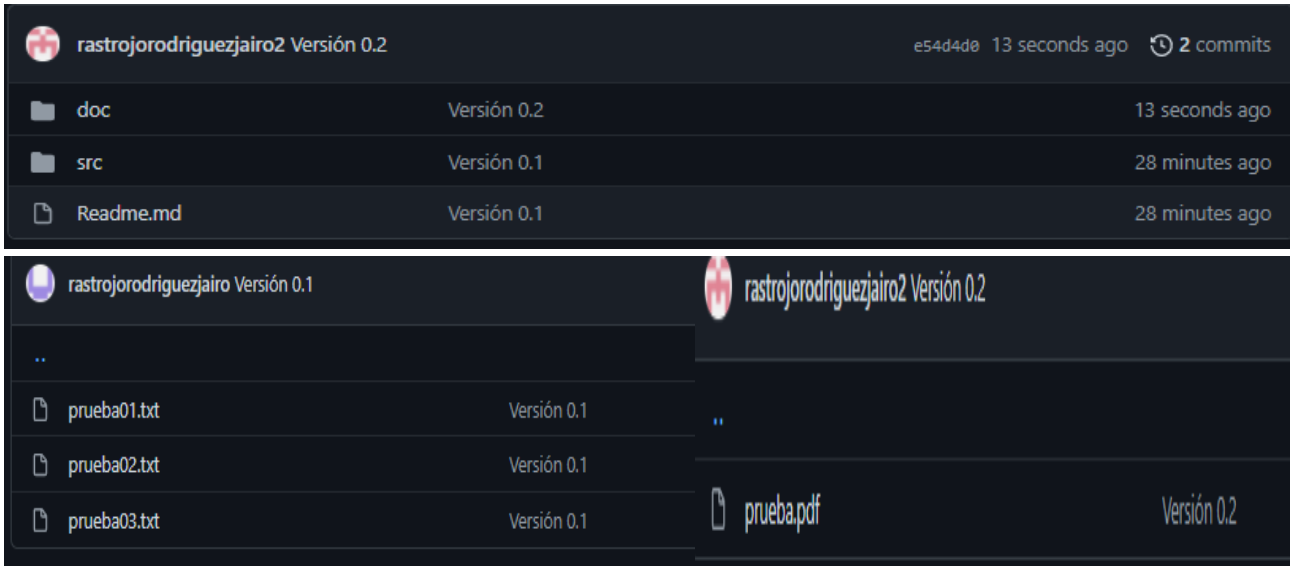




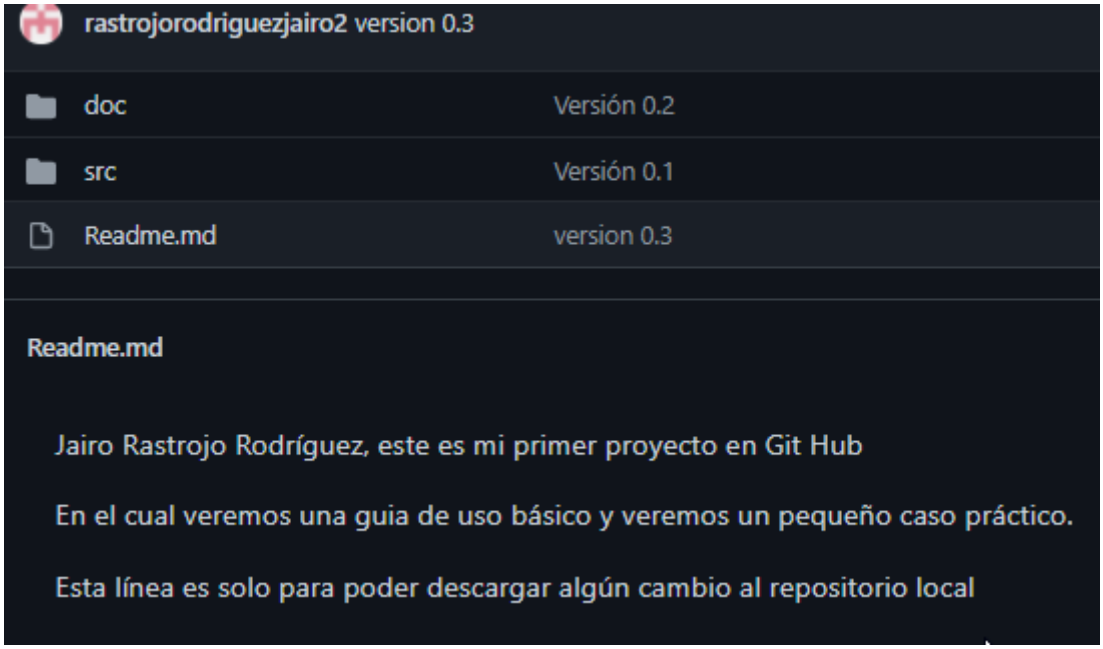


Podemos consultar las credenciales, poniendo en el buscador de windows “administrador de credenciales”, en la siguiente imagen podemos ver la credencial que se ha creado con github.

Como podemos ver se ha subido, pero pasa algo y es el commit, el motivo de que src y doc tengan un comit diferente es que como el doc estaba vació en la versión 0.1, no se subió, por lo que al darle contenido y resubirlo solo podía subirse si era con el commit versión 0.2, para ello he creado también un archivo de texto simple con nombre prueba pdf con commit versión 0.2, pero es para diferenciarla de la que se subirá al final.



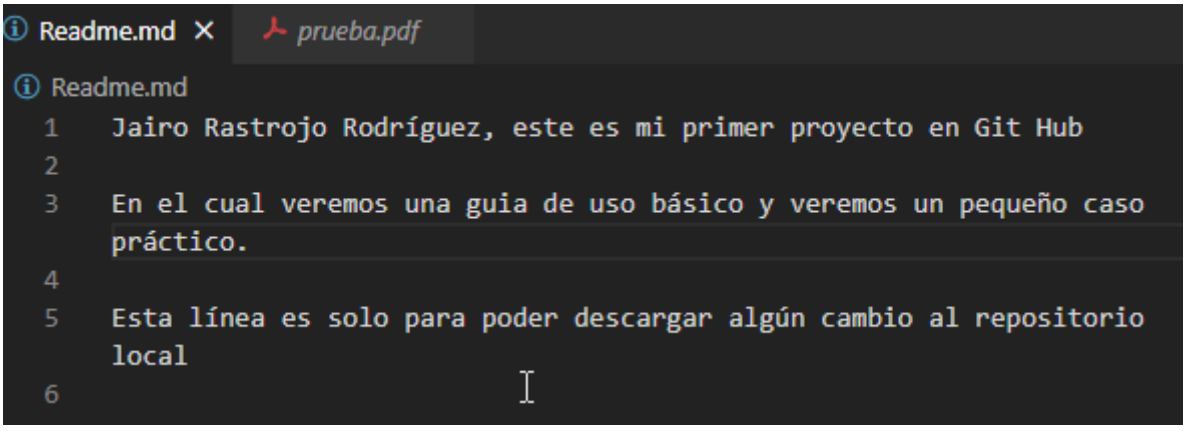
Ahora vamos a cambiar el Readme.md, añadiendo en la nube una línea más y cambiando el commit a version 0.3.



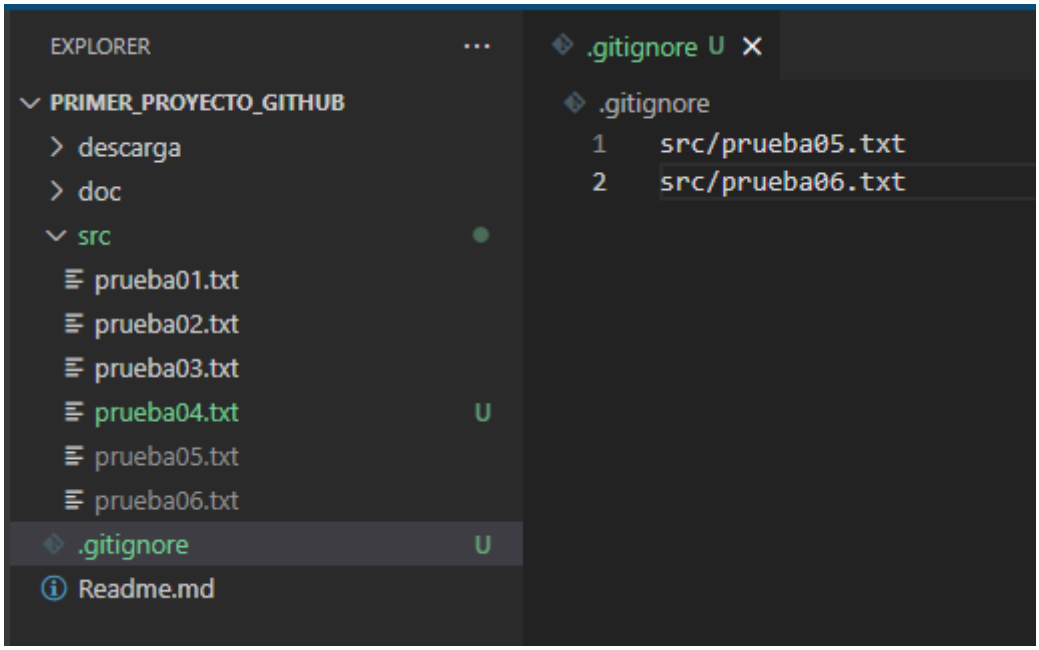
Como ha habido una variación ahora vamos a descargarlo al repositorio local, pero solo se descargarán los archivos que han sido alterados.

```
PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\Primer_Proyecto_GitHub\descarga> git pull htt
ithub.com/rastrojorodriguezjairo2/Primer_Proyecto
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), 782 bytes | 4.00 KiB/s, done.
From https://github.com/rastrojorodriguezjairo2/Primer_Proyecto
 * branch            HEAD       -> FETCH_HEAD
Updating 6b54939..3848cf5
Fast-forward
 Readme.md | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\Primer_Proyecto_GitHub\descarga>
```

Y vemos que el readme.md efectivamente se ha actualizado.



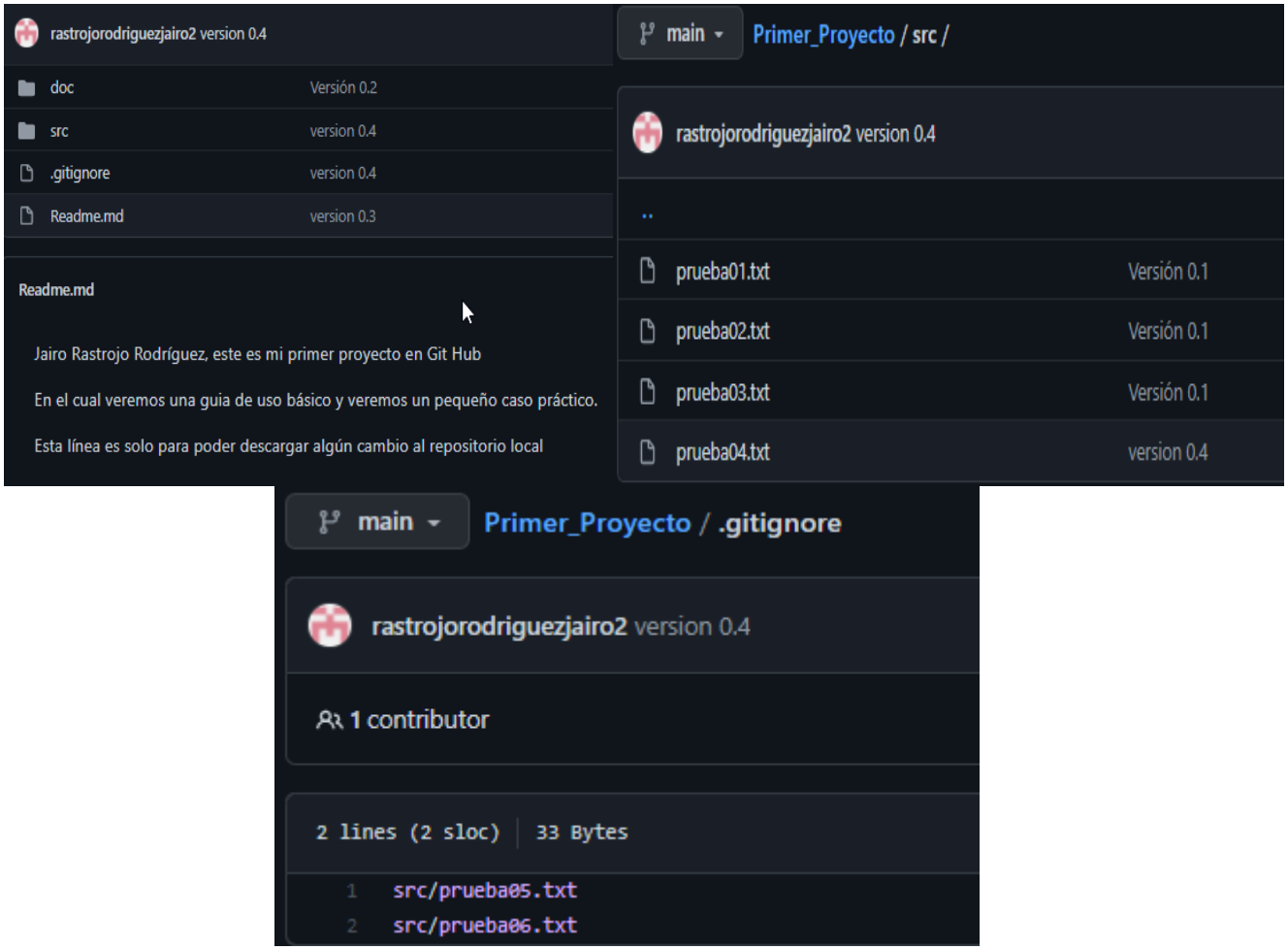
Ahora hemos añadido tres archivos de texto más al directorio src y vamos ahora a crear el archivo ".gitignore", el cual hara que todo archivo cuyo nombre está escrito en el, será ignorado por git y no se subirá al repositorio de github.



Ahora volvemos a usar git add . Y git commit como la versión 0.4, y lo subimos a la github

```
PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\Primer_Proyecto_GitHub> git add .
PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\Primer_Proyecto_GitHub> git status
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .gitignore
        new file:   src/prueba04.txt

PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\Primer_Proyecto_GitHub> git commit -m "versio

[main 15d471c] version 0.4
 2 files changed, 3 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 src/prueba04.txt
PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\Primer_Proyecto_GitHub> git push -u origin ma
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 465 bytes | 465.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/rastrojorodriguezjairo2/Primer_Proyecto.git
   3848cf5..15d471c  main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
PS C:\jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\Primer_Proyecto_GitHub>
```

Ahora veremos en github que por una parte se ha actualizado el directorio src a la versión 0.4, dentro de la cual se ha añadido el documento prueba04.txt y se mantienen las versiones 0.1 que no han sido alteradas, ademas no se han añadido los ignorados que eran el prueba 05 y prueba06, los cuales se encuentran registrados en el archivo .gitignore.



5º Conclusiones

Por último recalcar que habrá una ultima actualización de github con este mismo archivo en formato pdf, lo cual no se verá aquí pero al corregirlo se verá el commit diferente.