# clase n°8
# Git & GitHub


<animate a/> programar
revøluciøn* digital_

# Qué es GitHub y cómo usarlo para aprovechar sus beneficios.

- GitHub es una plataforma de alojamiento, propiedad de Microsoft, que ofrece a los desarrolladores la posibilidad de crear repositorios de código y guardarlos en la nube de forma segura, usando un sistema de control de versiones, llamado Git.


- Facilita la organización de proyectos y permite la colaboración de varios desarrolladores en tiempo real. Es decir, nos permite centralizar el contenido del repositorio para poder colaborar con los otros miembros de nuestra organización.

- GitHub esta basada en el sistema de control de versiones distribuida de Git, por lo que se puede contar con sus funciones y herramientas, aunque GitHub ofrece varias opciones adicionales y su interfaz es mucho más fácil de manejar, por lo que no es absolutamente necesario que las personas que lo usan tengan un gran conocimiento técnico.

  Una de sus funciones adicionales es le alojamiento de sitios web.

# Ventajas de GitHub

- Existe un gran número de razones que convierten a GitHub en una gran opción para el control y gestión de tus proyectos de código. Aquí algunas de ellas:

- GitHub permite que alojemos proyectos en repositorios de forma gratuita
- Te brinda la posibilidad de personalizar tu perfil en la plataforma
- Los repositorios son públicos por defecto. Sin embargo, GitHub te permite también alojar tus proyectos de forma privada
- Puedes crear y compartir páginas web estáticas con GitHub Pages
- Facilita compartir tus proyectos de una forma mucho más fácil y crear un portafolio
- Te permite colaborar para mejorar los proyectos de otros y a otros mejorar o aportar a los tuyos
- Ayuda reducir significativamente los errores humanos y escribir tu código más rápido con GitHub Copilot

**Debes tener en cuenta que Git es un sistema que permite establecer un control de versiones, mientras que GitHub es una plataforma que ofrece un grupo de funciones que facilitan el uso de Git y la colaboración en tiempo real, así como el almacenamiento en la nube.**

***¿Cómo empezar a usar GitHub?***

Una vez que ya conocimos todo sobre GitHub y sus bondades, es hora de ponernos manos a la obra para adentrarnos en este interesante mundo:

Lo primero que debes tener es una cuenta creada en ***GitHub***. Registrarse es gratuito. Una vez que tengas la cuenta, inicia sesión con tu usuario y clave.

**https://github.com/**

# Git GUI


- Cómo instalar Git

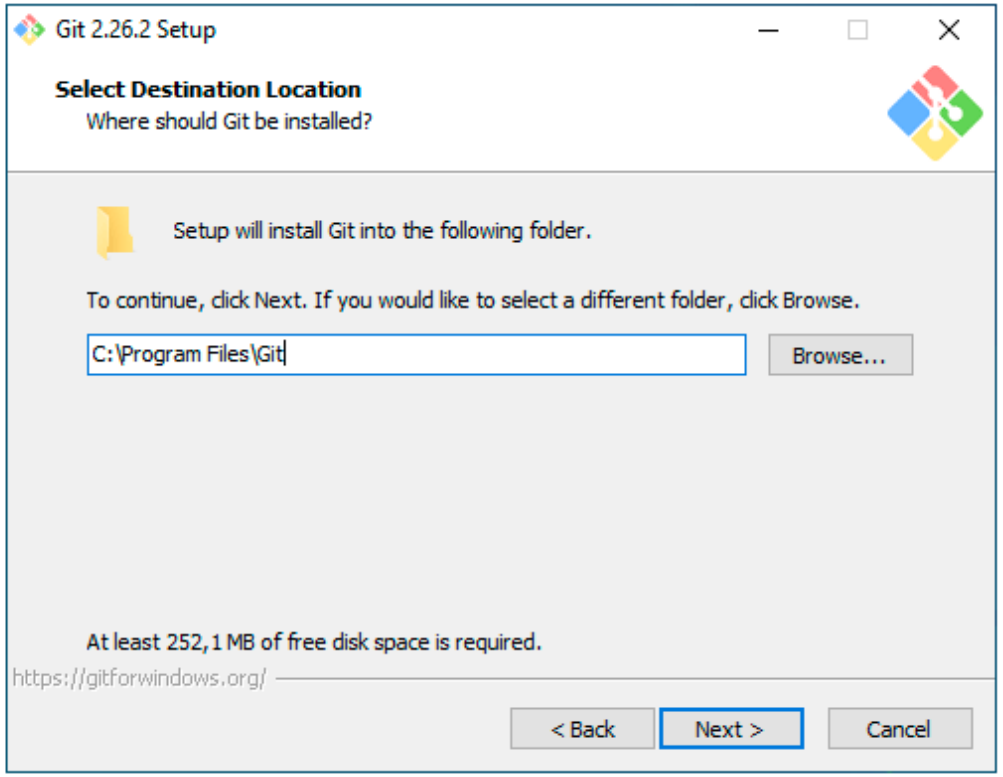
Para instalar Git en Windows debes descargar Git for Windows desde aquí. Puedes descargar la versión que incluye el instalador u otra portable que podrás ejecutar en cualquier ordenador. En este tutorial usaremos el instalador, que dependiendo de tu sistema operativo, podrá ser de 32 o de 64 bits, aunque lo más probable es que la versión correcta sea la de 64 bits salvo que tu PC tenga más de 10 años.

1 Primero acepta las **condiciones** que se muestran en pantalla y haz clic en **Next** para continuar.

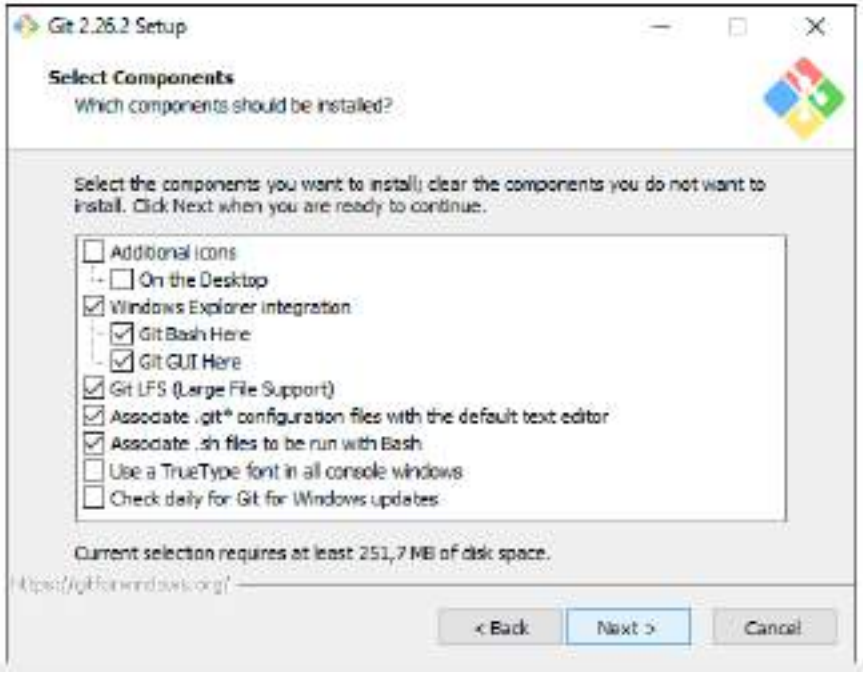

***Git - Downloading Package (git-scm.com)***

https://git-scm.com/download/win

2 Luego selecciona el directorio de instalación y haz clic en **Next** para continuar.

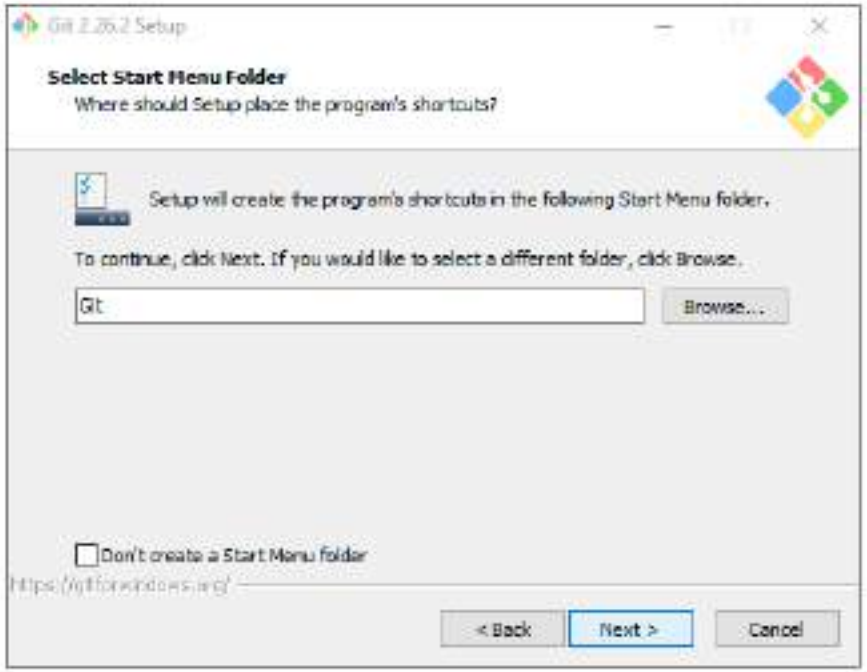



3 Seguidamente, selecciona los **componentes** que se instalarán. Es recomendable que selecciones los componentes que integrarán **Git** con el **explorador de Windows** para así evitar navegar excesivamente por los directorios usando la terminal.

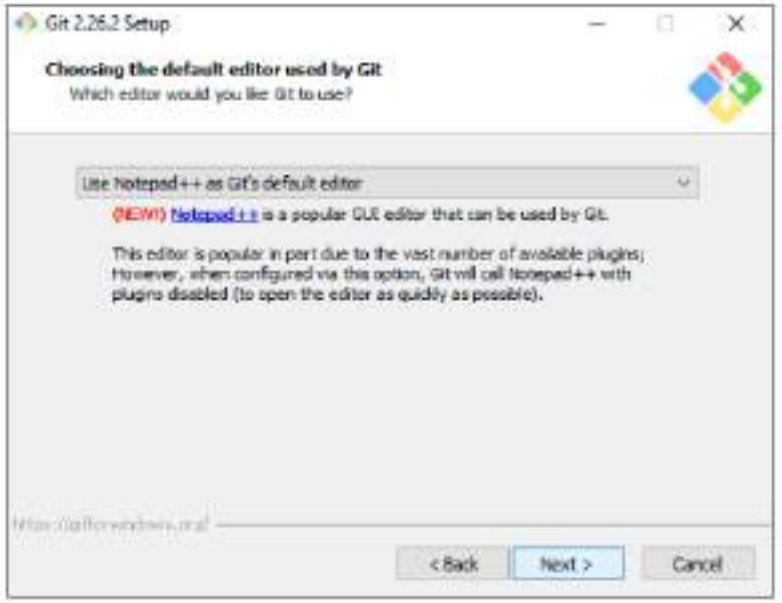


- Mediante ***«Git Bash Here»*** podrás iniciar una ventana de terminal desde cualquier carpeta de tu sistema. Del mismo modo, la opción ***«Git GUI Here»*** te permitirá iniciar la interfaz de Git desde cualquier lugar. Haz clic en ***Next*** para continuar.

**4** Selecciona el directorio en el que se crearán los accesos directos del menú de inicio (Solo Windows 7, Windows 8 y Windows XP). Haz clic en **Next** para continuar.

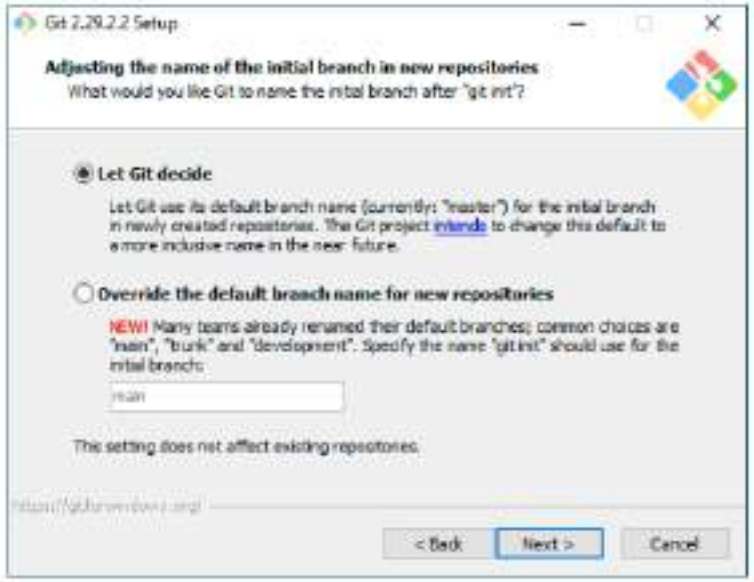



**5** Ahora tendrás que seleccionar el editor que usará Git de forma predeterminada. Puedes seleccionar **Vim**, que es el que se usa por defecto, aunque si no lo has utilizado nunca, es mejor que selecciones cualquier otro, como Nano, Notepad++ o VS Code.



- Haz clic en ***Next*** para continuar.

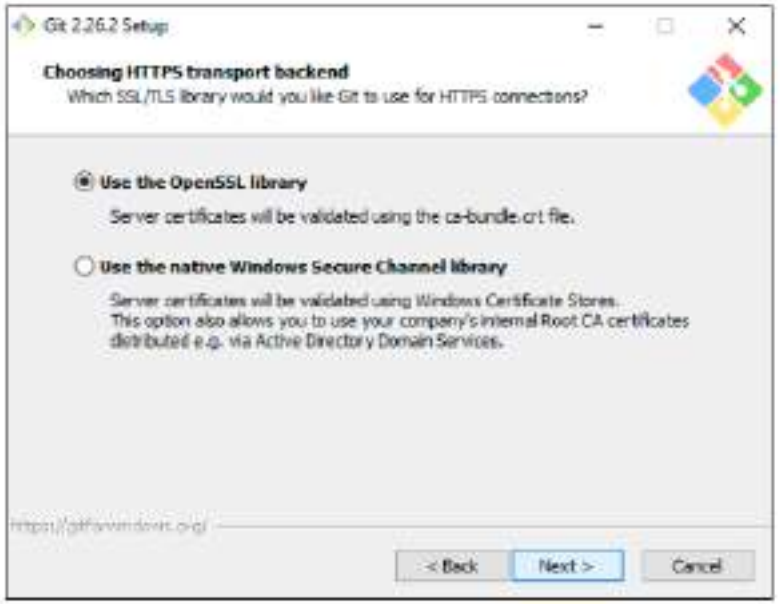
**6** Git ha decidido que su rama principal por defecto deje de ser `master` y pase a ser `main` por temas de inclusividad racial. Por ello y dependiendo de la versión de Git que instales, es posible que se te pregunte si prefieres usar la rama de Git por defecto, o si por el contrario quieres especificar otra. Escoge la opción «**Let Git decide**» y haz clic en **Next** para continuar:



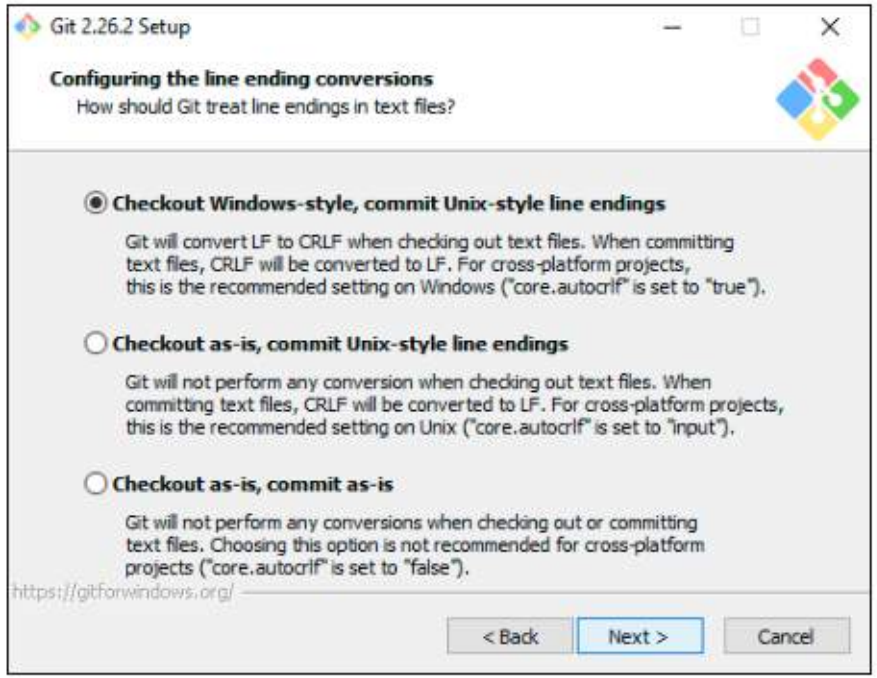

**7** Luego tendrás que seleccionar las opciones de configuración del **PATH** de Windows. Para evitar problemas es mejor que selecciones la opción por defecto, que es la recomendada. Es decir, «***Git form the command line and also from 3rd-pary software***», ya que la última opción podría sobrescribir ciertos comandos de Windows.

- Haz clic en ***Next*** para continuar.

**8** Ahora tendrás que seleccionar el tipo de librería que usará Git para gestionar las conexiones **SSL/TLS**. Lo recomendable es que uses la opción por defecto, que es la librería **OpenSSL**, salvo que necesites algún tipo de configuración especial. Haz clic en **Next** para continuar.

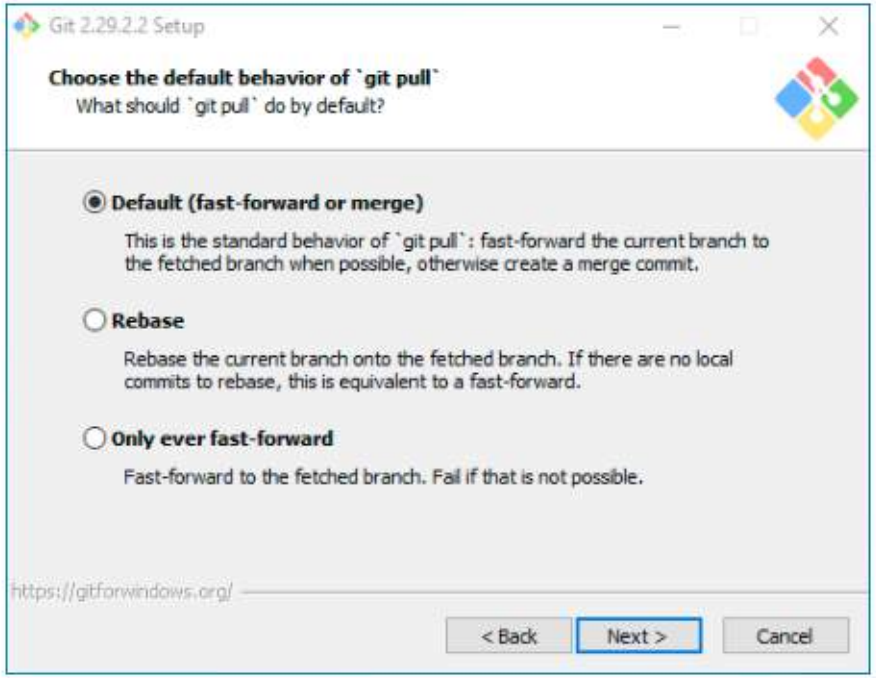



**9** A continuación tendrás que seleccionar el trato que Git dará a los marcadores de final de línea, puesto que Windows y los sistemas Unix difieren en este aspecto. No te compliques y selecciona la opción ***«Checkout Windows-style, commit Windows-style line endings»***, que es la opción por defecto, maximizando la compatibilidad entre sistemas Unix y Windows. Haz clic en **Next** para continuar.

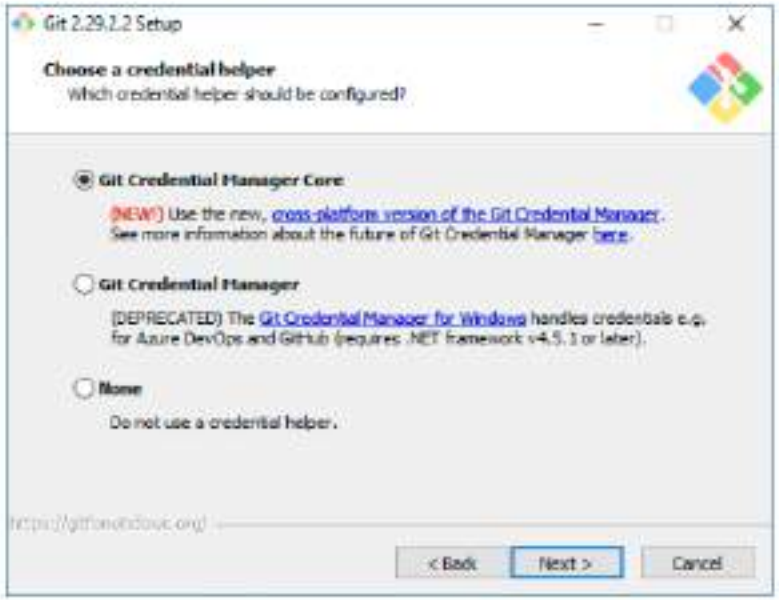


10 A continuación tendrás que escoger el tipo de terminal que se utilizará como interfaz. Podrás escoger **MinTTY**, que es la opción por defecto y la recomendada, o la típica **terminal de Windows**. Escoge **MinTTY**, ya que apenas dará problemas y es mucho más fácil de utilizar. Haz clic en **Next** para continuar.

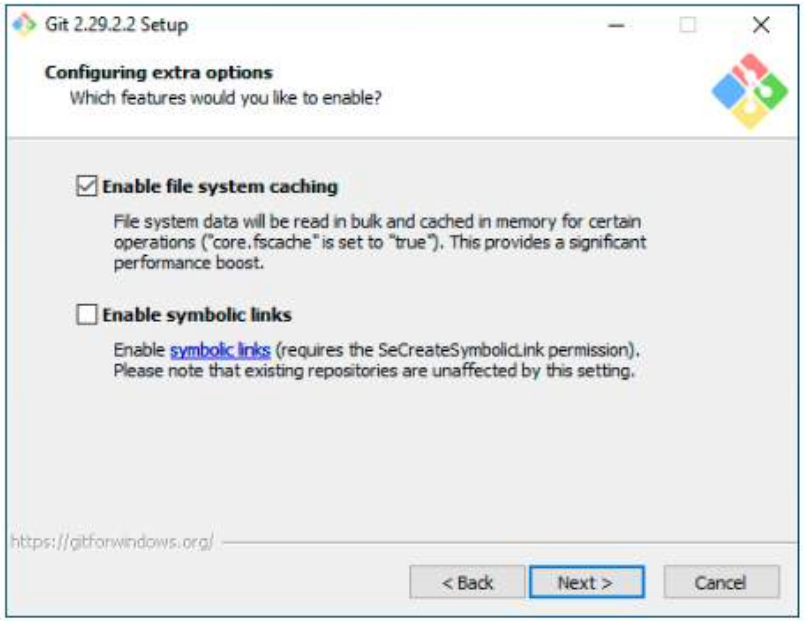

11 Luego se te preguntará por el funcionamiento que tendrá el comando `git pull`. Escoge la opción por defecto.



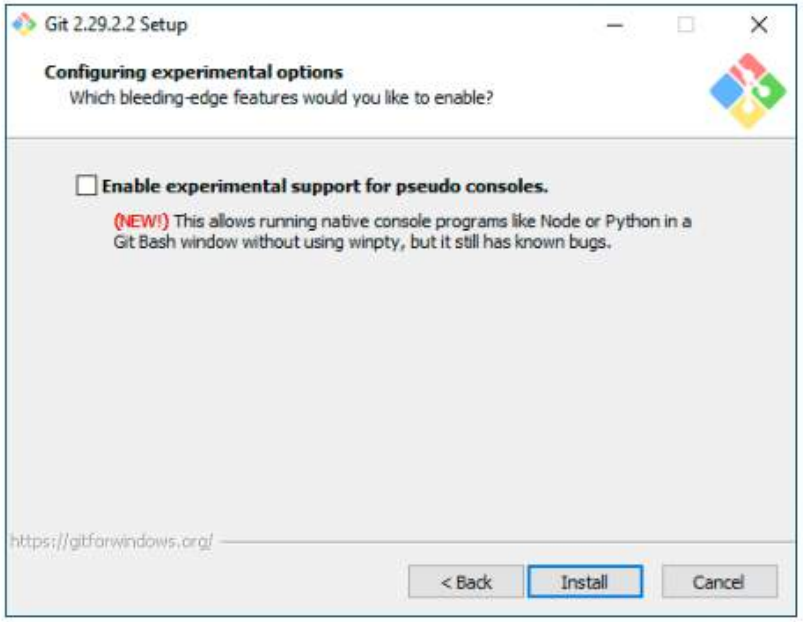
12 Seguidamente tendrás que escoger la aplicación que se usará para gestionar las credenciales de Git. Puedes escoger entre **Git Credential Manager Core** o **Git Credential Manager** para Windows. Es recomendable que escojas la primera opción, que es la que está seleccionada por defecto, ya que hace uso de varias funcionalidades específicas de Windows.




13 A continuación deja activada la opción que activa la **caché de Git** y activa también la compatibilidad con **enlaces simbólicos**. Luego haz clic en **Next** para continuar.



14 Luego podrás escoger una opción experimental, que es el soporte de pseudo consolas, que permite la ejecución de aplicaciones nativas de consola en Git Bash sin necesidad de usar **Wimpty**. Desactiva esta opción salvo que sepas lo que estás haciendo y finalmente haz clic en **Install**.




15 Una vez finalizada la instalación, haz clic en **Next** para continuar e iniciar Git Bash si así lo deseas.


**Y con esto, Git y Git Bash se habrán instalado correctamente. No solamente podrás utilizar Git Bash con Git, ya que también podrás utilizar esta aplicación para ejecutar otras aplicaciones de Linux como por ejemplo Node.js.**

# Publicar un sitio web en GitHub Pages.

- GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git.

- El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago. GitHub aloja tu repositorio de código y te brinda herramientas muy útiles para el trabajo en equipo.

- Una de esas herramientas es Github Pages, la cual permite alojar sitios web estáticos sin necesidad de tener conocimientos en servidores. GitHub pages permite dos modalidades de publicación:

- La primera es ***"User site"*** (solo se podrá tener un sitio de este tipo por cuenta); en este caso el sitio web será publicado en ***username.github.io*** (siendo username el nombre de usuario de la cuenta).

- La segunda opción es ***"Project site"*** (proyectos ilimitados) el cual será publicado en username***.github.io***/repository (siendo repository el nombre del repositorio).

En esta ocasión se utilizará la primera opción ***"User site"***, antes de comenzar debes de tomar en consideración lo siguiente:

- Tener una cuenta de GitHub creada. En caso de no tenerla la pueden crear aquí: https://github.com/join
- Debes tener instalado GitHub Destock en tu equipo.
- Github para que puedas sincronizar con los archivos de tu PC.
- Vas a tener que autorizar a la aplicación.
- Tu proyecto debe contar con un archivo index.html

***Instalación git:***

**https://desktop.github.com/**

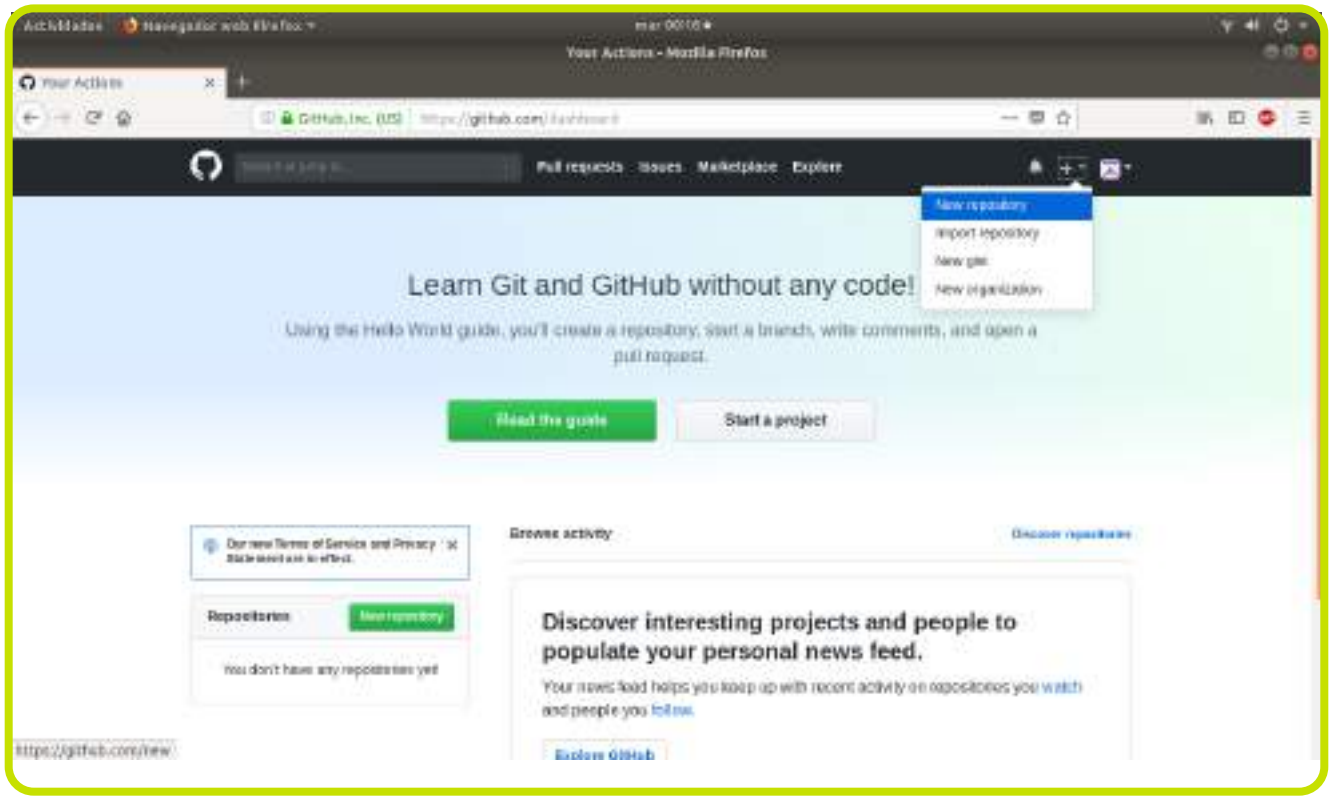
***Crear repositorio***

Lo primero que se debe realizar es ingresar a la página principal de GitHub e iniciar sesión.
Una vez hayan ingresado se encontrarán en el dashboard.

# Dashboard de GitHub

- Lo siguiente será crear un nuevo repositorio, para ello daremos click sobre el ícono de + que se encuentra en la esquina superior derecha, se mostrará un menú con diversas opciones, se seleccionará ***New repository.***

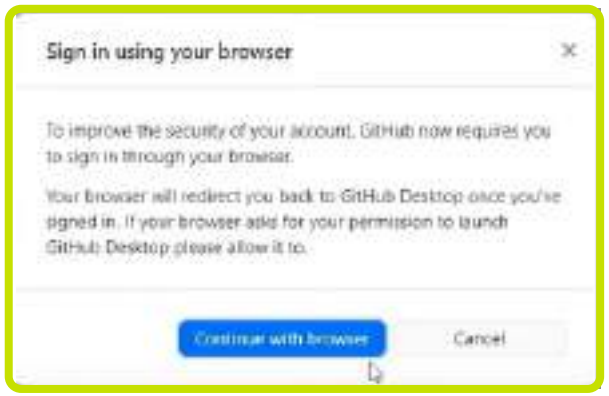
- Se debe crear un nuevo repositorio en tu disco duro y el nombre del repositorio debe de ser el nombre de usuario, se ingresará el nombre del repositorio y opcionalmente una descripción del proyecto.

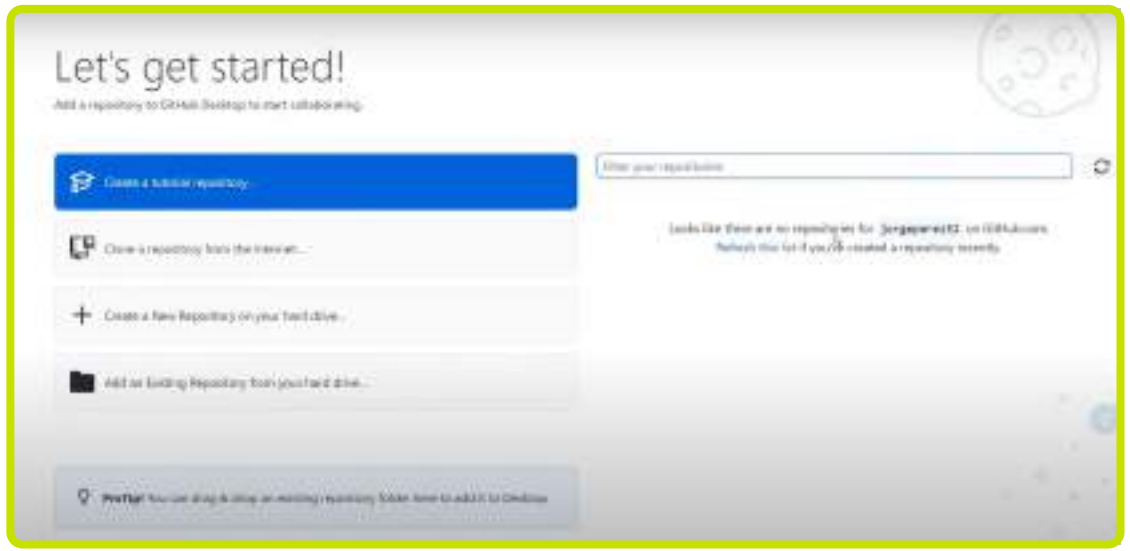


- Opción New repository

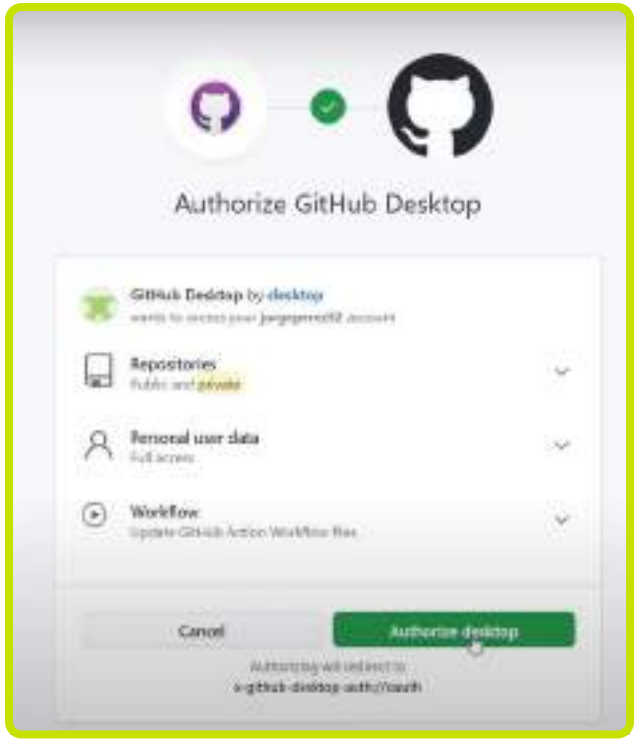

- Opción New repository

- Selecciono Accounts – Sign in

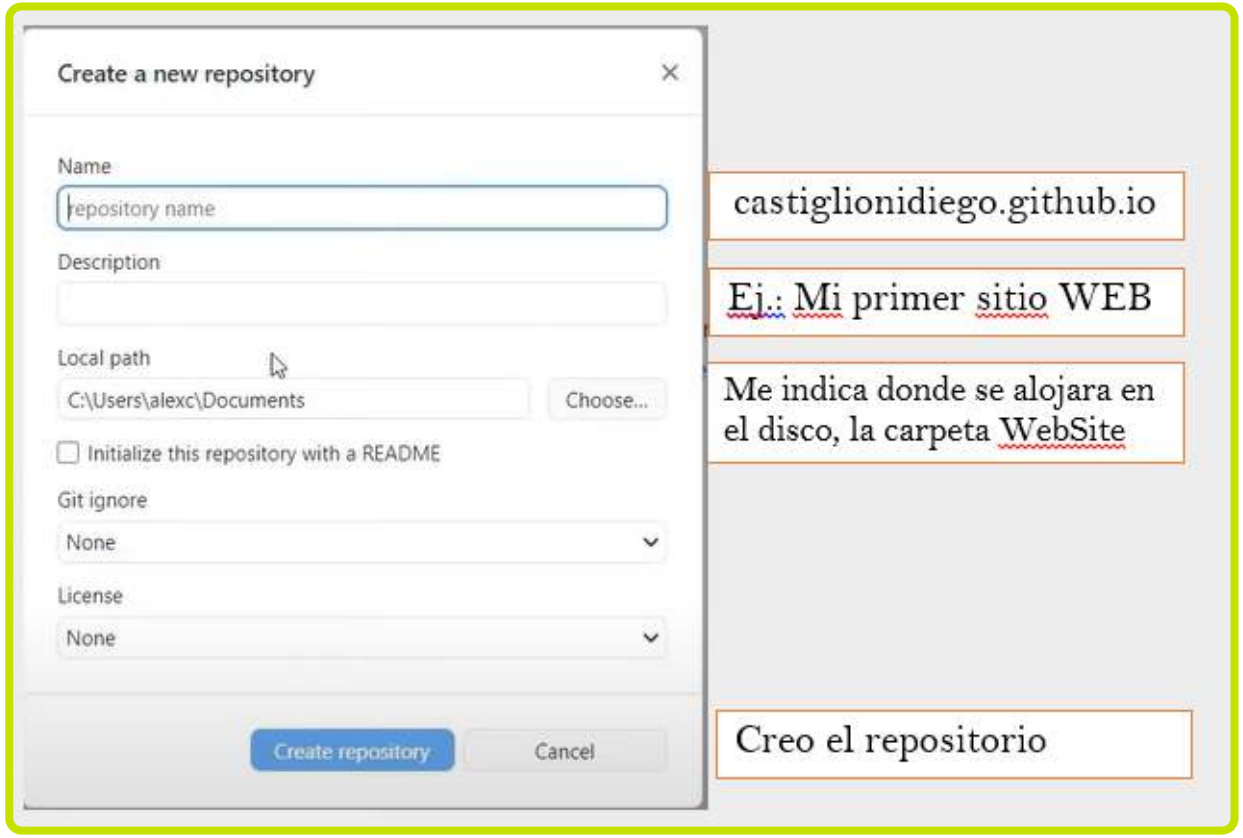



- Vas a tener que autorizar a la aplicación.




Es importante mencionar que el nombre de este repositorio debe ser de la siguiente manera: ***username.github.io***; donde username es el nombre de usuario de la cuenta. En este ejemplo el nombre de usuario es ***castiglionidiego***, por lo tanto ingresaré como nombre de repositorio ***castiglionidiego.github.io***

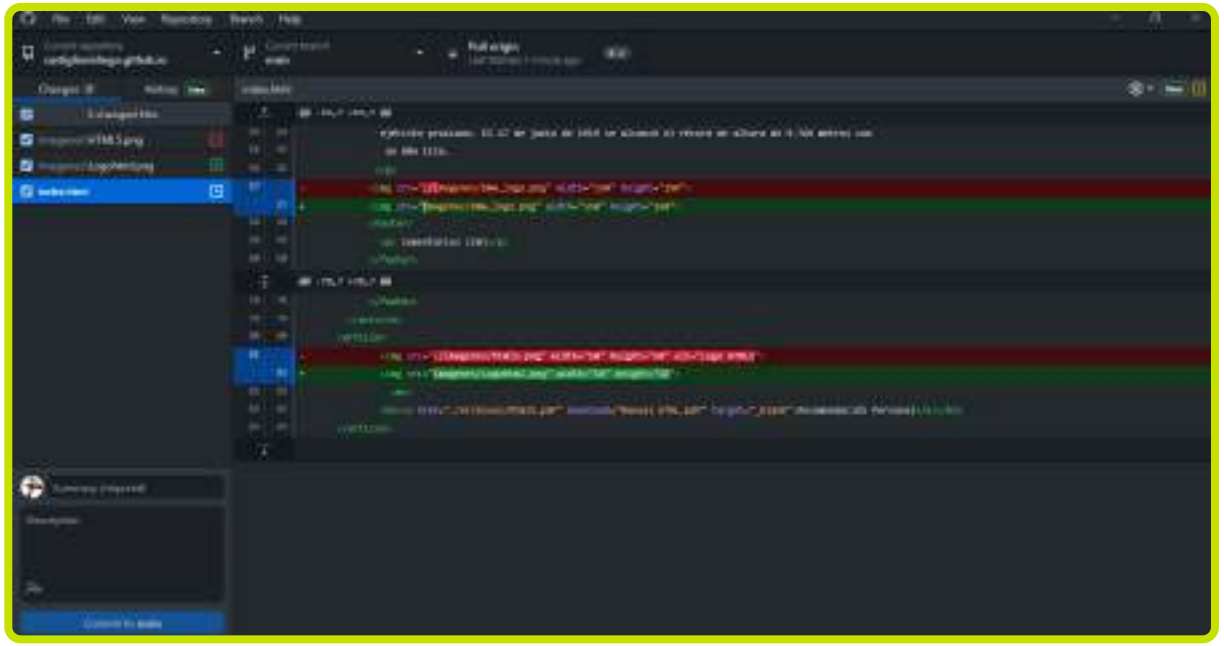



- Ahora vamos a pasar (copiar) a la carpeta que se ha creado, todos los archivos que integran el sitio creado.

Se pasa a la carpeta que se creó, la cual se llama:

- Detalle de los archivos copiados.

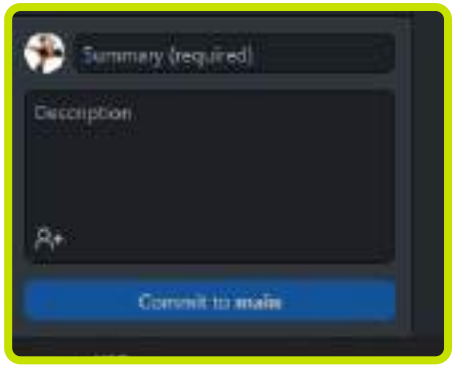
***Regresamos a GitHub Destock***



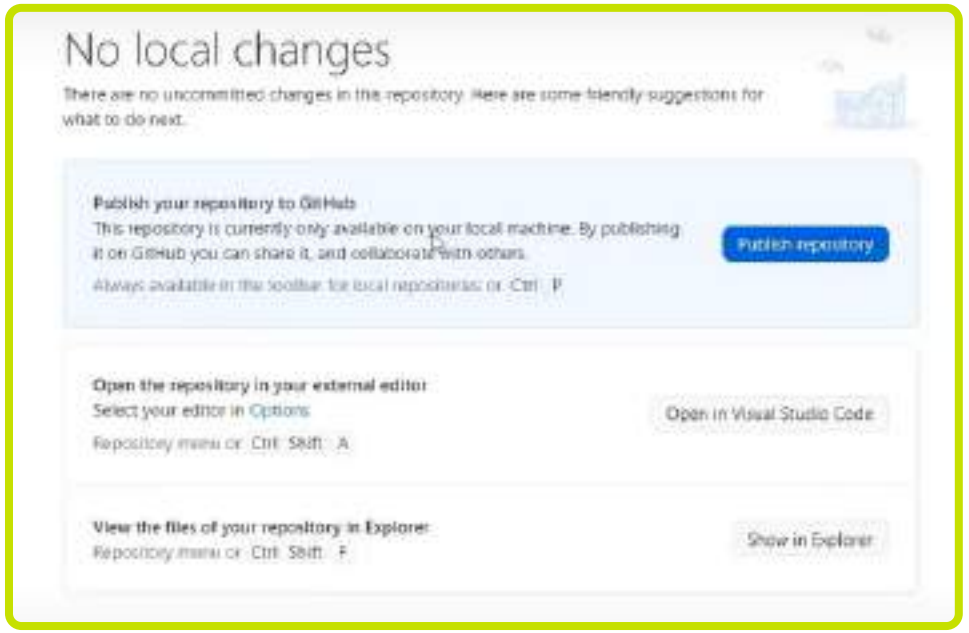
- Veremos todos los archivos que hemos pasado a la carpeta.

Crearemos un Commit – Ej. Creando mi primer Sitio Web

Luego hacemos click en el botón Commit to main.



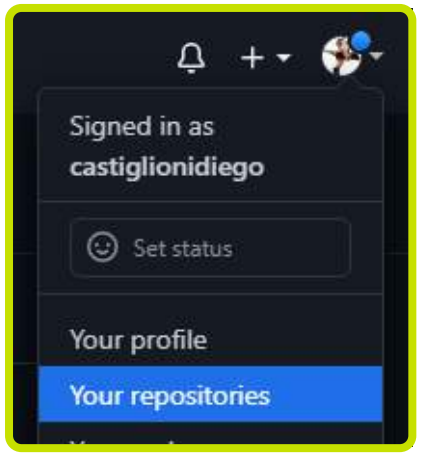
- Elegimos pública tu repositorio en GitHub



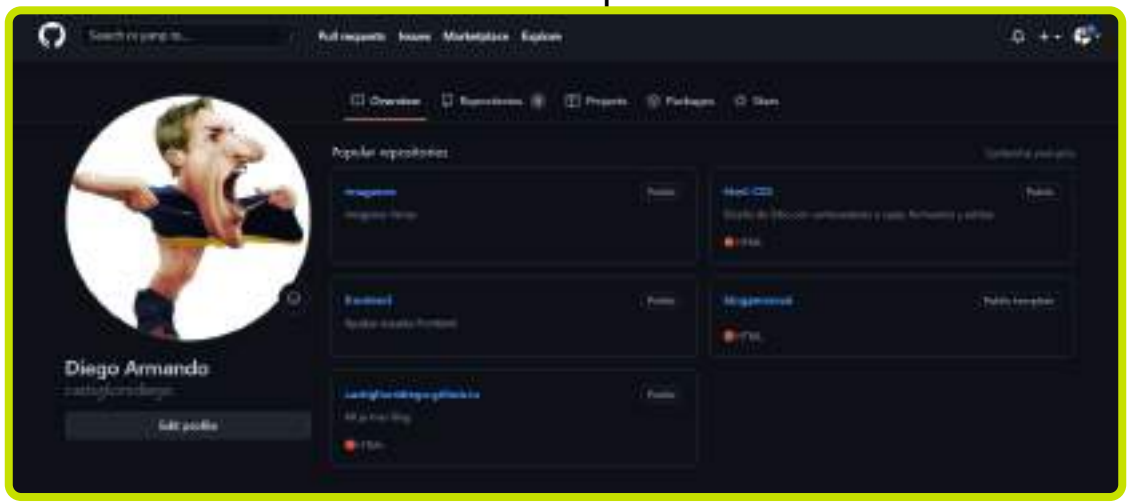

Desmarcamos la casilla Mantener el código en privado.

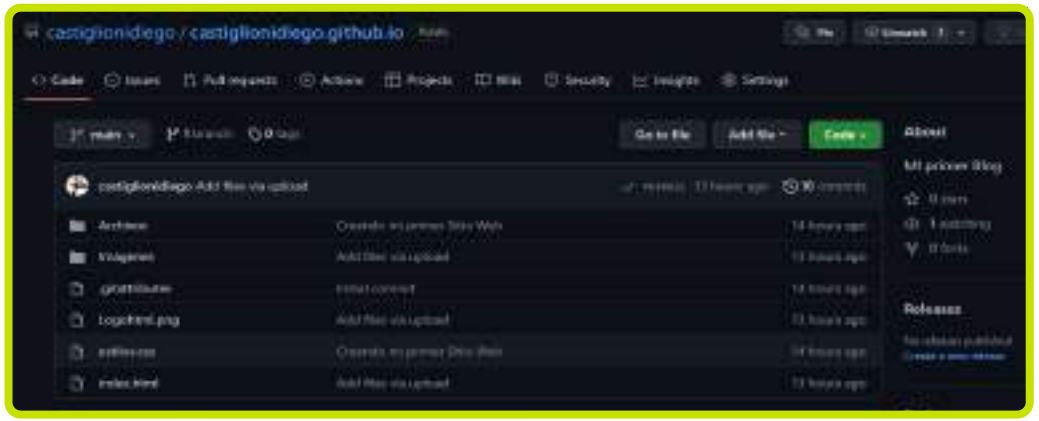

Asegurarse de que tenga nuestro nombre de repositorio.

- Y publicamos. Esperamos unos segundos y volvemos a GitHub, en nuestro navegador.
- Actualizamos la página, ahí debemos ver nuestro repositorio.
- Vamos al logo de nuestra cuenta y en la lisa elegimos Your repositorios.

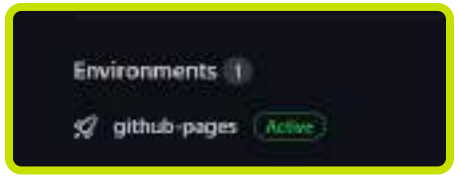



Lo selecciono entre todos mis repositorios.



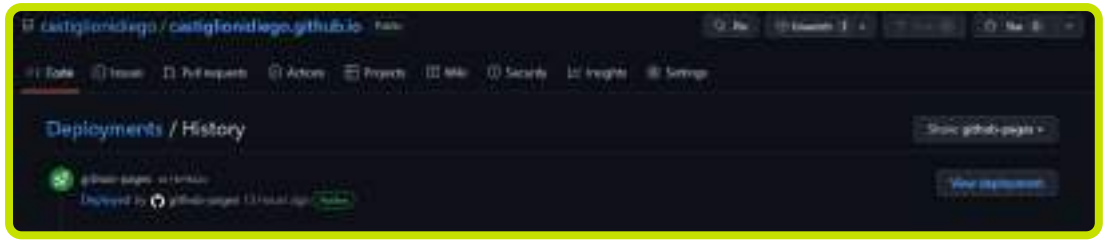



- Debería ver dentro de dicho repositorio todos los archivos que cargue en la carpeta.
  Si te fijas en la parte inferior derecha, encontraras un apartado llamado Environments.




- Cliquemos donde dice github-pages.
  Nos mostrara el historial de nuestros cambios en el Sitio, en este caso solo veremos un solo movimiento porque recién hemos subido el sitio.

  Hacemos clic en el botón View deployment.




  Ahí veremos nuestro sitio publicado.
  Ya lo podemos compartir con nuestros contactos.

revølución*
digital_