

En este enlace podéis ver un resumen pero en el documento veréis capturas que os facilitarán el proceso. Buen día.

<https://usodegit.netlify.app/>

1. Crear una cuenta en git

A screenshot of the GitHub registration interface. The background is dark blue. At the top right, the text "del mundo." is visible. In the center, there is a light blue rounded rectangular input field containing the text "Introduce tu correo electrónico" and the email address "macontreras2709@gmail.com". To the right of this field is a green rounded rectangular button with the text "Regístrate en GitHub".

del mundo.


Introduce tu correo electrónico
macontreras2709@gmail.com

Regístrate en GitHub

2. Iniciar nuestra sesión en git



Iniciar sesión en GitHub

¡Tu cuenta se creó correctamente! Inicia sesión  para continuar.

Nombre de usuario o dirección de correo electrónico

Contraseña

[¿Has olvidado tu contraseña?](#)

Iniciar sesión

o



Continuar con Google

¿Eres nuevo en GitHub? [Crea una cuenta](#)

3. Vamos a comenzar desde cero vamos a verificar que tengamos la versión correcta y que no haya otro usuario acreditado en nuestro ordenador para git
Verificar que no haya otro usuario con credenciales de git en nuestro ordenador

Verificar que no haya otro usuario con credenciales de git en nuestro ordenador. Verificar y eliminar credenciales guardadas

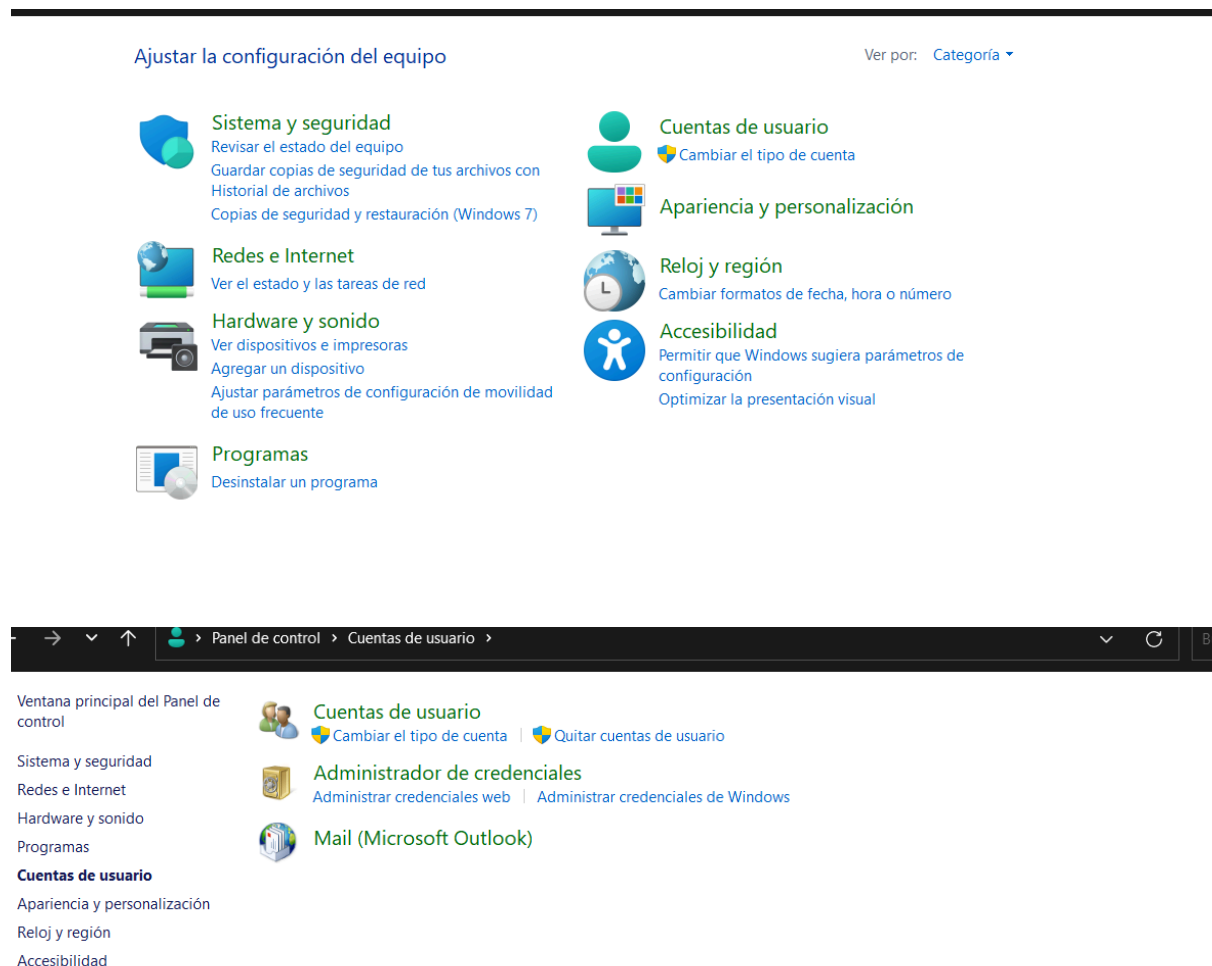
Abri el Panel de control. La forma más fácil es escribir "Panel de control" en el menú de inicio y seleccionarlo.

Una vez allí, ve a Cuentas de usuario y luego a Administrador de credenciales.

En la ventana del Administrador de credenciales, selecciona Credenciales de Windows.

Busca cualquier entrada que esté relacionada con Git o con servicios de repositorios como "git:https://github.com" o similares.

Si encuentras alguna, haz clic en la entrada para expandirla y luego selecciona "Quitar" para eliminarla. Esto asegurará que no queden credenciales guardadas en tu sistema que Git pudiera usar en el futuro.



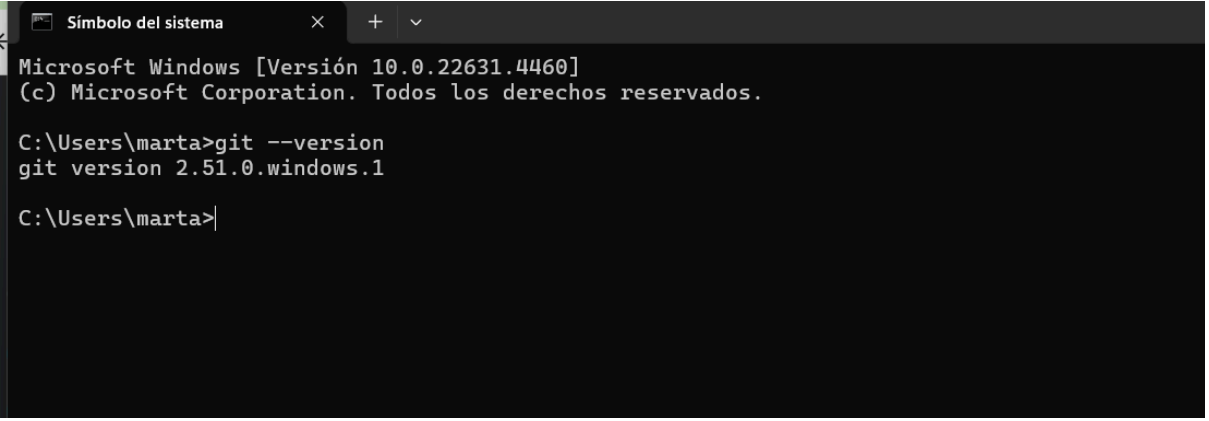
Credenciales genéricas	Agregar una credencial genérica
DriveFS_105597518035132389449	Fecha de modificación: Hoy ▾
UnityHub/combinedTokens	Fecha de modificación: 30/04/2025 ▾
git:https://github.com	Fecha de modificación: 22/04/2025 ▾
LenovoSsoSdkDidToken	Fecha de modificación: 01/03/2023 ▾
Microsoft OneDrive Generic Data - Personal Vault VHD...	Fecha de modificación: 05/03/2023 ▾
MicrosoftAccount:user=martabatan21@gmail.com	Fecha de modificación: 24/04/2025 ▾
MicrosoftOffice16_Data:OAUTH2:10559751803513238...	Fecha de modificación: 03/01/2021 ▾
MicrosoftOffice16_Data:OAUTH2:10559751803513238...	Fecha de modificación: 03/01/2021 ▾
Git/BuildModification/BuildKey...	Fecha de modificación: 03/03/2024 ▾

Había un git ya vinculado entonces vamos a eliminar estas credenciales pues si no acudirá a esas credenciales y al no coincidir no se vincularía nuestro repo con nuestro archivo.

las volveremos a vincular con nuestro propio repositorio al hacer nuestro primer commit.

4. Comprueba la versión de Git que tenemos. Si no hay un git ya instalado pasamos al paso 5. Y si lo está o lo instalaste tú verifica que sea la versión correcta.

Vamos a verificar que tenemos git en nuestro ordenador abrimos la terminal de window en la lupa escribe cmd y luego escribe este comando git --version te dirá que versión de git tenemos



```

Microsoft Windows [Versión 10.0.22631.4460]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\martas>git --version
git version 2.51.0.windows.1

C:\Users\martas>

```

Esta es la última versión es correcta podemos dejarla. Asegurate del tema de los credenciales.

5. Instalar Git

1. Instala Git 📦

Antes de nada, necesitas tener Git instalado en tu computadora. Puedes descargarlo desde el sitio oficial de Git.

- Ve a git-scm.com/downloads.
- Descarga la versión adecuada para tu sistema operativo (Windows, macOS o Linux).
- Sigue las instrucciones del instalador. Generalmente, puedes dejar las opciones por defecto, a menos que sepas que necesitas una configuración específica.



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

Instalamos con la configuración por defecto

Elegimos como editor visual studio code la normal no la que pone insider
dale a next y seguimos con la instalación

La razón por la que eliges Visual Studio Code es que ya lo estás usando para programar. Al seleccionarlo, Git se abrirá automáticamente en VS Code cada vez que necesites escribir un mensaje, lo cual hace el proceso más fácil y rápido para ti.

Después te va a preguntar en qué rama quieres trabajar, vamos a dejar que git elija así Git creará la rama con el nombre por defecto que la comunidad de desarrollo usa hoy en día, que es **main**. Esta es la mejor opción para la mayoría de los usuarios. No obstante en posteriores pasos verificaremos que realmente esté creando la rama main y no master que es el nombre que está ya en desuso.

6. Vamos a **verificar que tenemos git en nuestro ordenador**
abrimos la terminal de window en la lupa escribe cmd y luego escribe este comando `git --version` te dirá que versión de git tenemos



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.22631.4460]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\martas>git --version
git version 2.51.0.windows.1

C:\Users\martas>
```

Coincide con la que hemos bajado no tenemos ninguna anterior y hemos descargado correctamente el programa. Vamos por buen camino

7. Ahora vamos a configurar git para que funcione en nuestro ordenador

Al ejecutar el comando `git config` vas a configurar tu identidad en Git.

Este es un paso fundamental y la primera cosa que debes hacer después de instalar Git. Piensa en ello como registrarte en una plataforma: le estás diciendo a Git quién eres.

Sin esta configuración, no podrás hacer "commits", y Git te pedirá que la establezcas antes de permitirte guardar cambios. Así que, en resumen, `git config` es el paso inicial para que Git sepa quién eres y pueda atribuirte correctamente tus contribuciones a los proyectos.

Paso 1: Configurar Git en tu computadora

Abre la terminal integrada de VS Code. Ve a **Terminal > Nueva Terminal** o presiona **Ctrl + Shift + Ñ** (o **Ctrl +** en algunos teclados).

Una vez que la terminal esté abierta, configura tu nombre de usuario y correo electrónico. Esto es lo que aparecerá en tus "commits".

Bash

```
git config --global user.name "Tu Nombre"
git config --global user.email "tu.correo@ejemplo.com"
```

Nota: El nombre que uses no tiene que ser exactamente el mismo que tu nombre de usuario de GitHub, pero es buena práctica que sea tu nombre real.

Introducimos en el terminal esta configuración.

git config --global user.name "Tu Nombre"
git config --global user.email "tu.correo@ejemplo.com"

Con los comandos que te he dado (git config --global user.name y git config --global user.email), estás estableciendo tu nombre de usuario y tu dirección de correo electrónico a nivel global en tu computadora.

Para verificar que tu nombre y correo electrónico se han guardado correctamente, lo que acabas de configurar, puedes usar los siguientes comandos, uno por uno:

Para ver tu nombre de usuario:

git config --global user.name

Para ver tu correo electrónico:

git config --global user.email

Si después de ejecutar estos comandos ves tu nombre y tu correo, significa que la configuración se realizó correctamente y ya estás listo para usar Git.

```
PS C:\Users\marta\OneDrive\Escritorio\git> git config --global user.name
inma2709
PS C:\Users\marta\OneDrive\Escritorio\git> git config --global user.email
inmacontreras2709@gmail.com
PS C:\Users\marta\OneDrive\Escritorio\git> █
```

Tu identidad está establecida y que Git está listo para asociar tus futuras contribuciones a tu nombre.

8. Vamos a crear un repositorio. ¿Qué es un repositorio?

Un repositorio, a veces llamado "repo" para abreviar, es básicamente el lugar central donde se almacena tu proyecto. Le estamos diciendo donde va a guardar nuestro proyecto. Cada proyecto que hagamos tendrá su propio repositorio.

Piensa en él como una carpeta especial que Git usa para guardar todo lo que necesitas para tu proyecto, incluyendo: Tu código y archivos: Todas las carpetas y archivos de tu proyecto (código, imágenes, documentos, etc.).

- **El historial de cambios:** Lo más importante es que el repositorio almacena todas las versiones y cambios que has hecho a lo largo del tiempo. Cada vez que haces un "commit", Git crea una "instantánea" de tu proyecto y la guarda en el repositorio.



Abre la terminal de VS Code y navega hasta esa carpeta de tu proyecto que quieres subir a git.

Ejecuta git init. Esto inicializa un repositorio local. Si has podido inicializarlo sin problema vamos a hacer nuestro primer commit (busca el punto 10 donde hacemos nuestro primer commit)

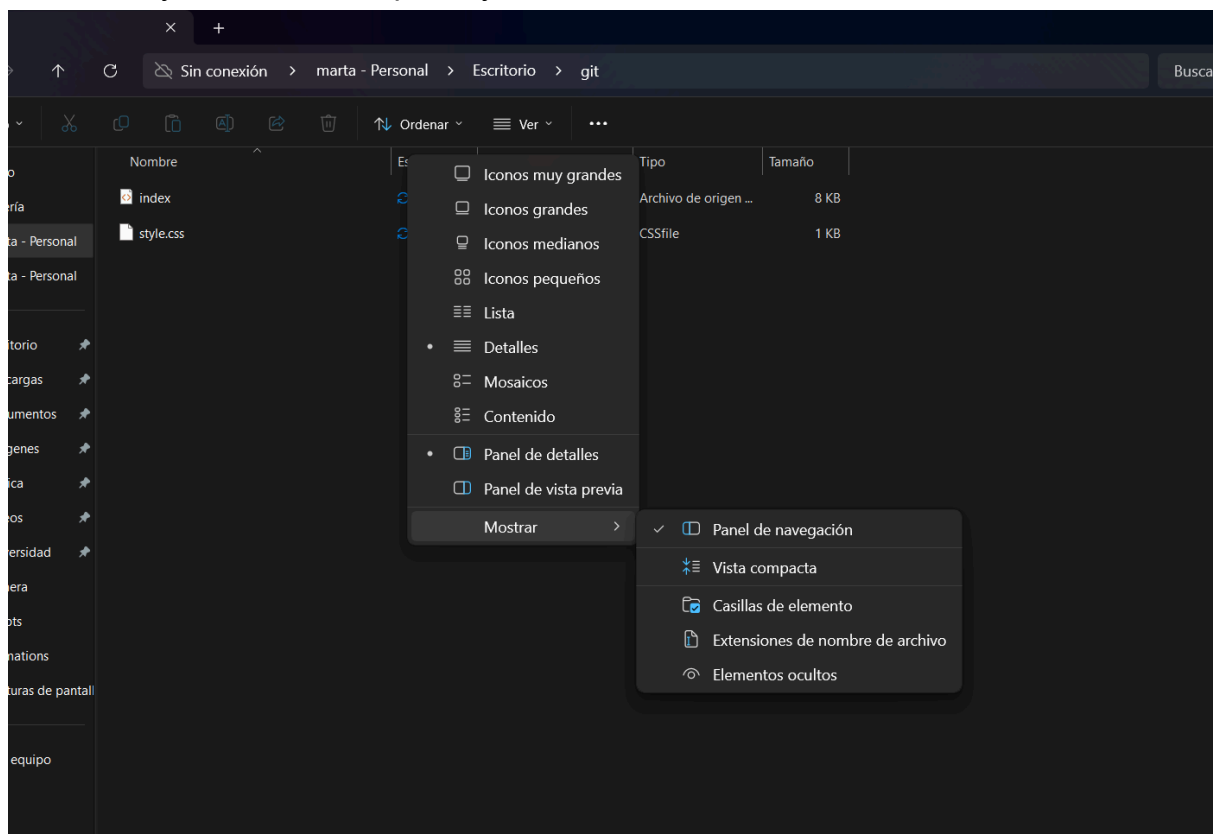
¿Te ha dado algún error? Si el error que te ha dado es de que ya había un repositorio en esa carpeta y no estamos seguros si está bien asociado a nuestro repo lo mejor es eliminarlo. Realmente no sería un error si sabemos que lo hemos configurado bien pero si no estamos seguros lo mejor es comenzar desde cero

```
PS C:\Users\marta\OneDrive\Escritorio\git> git config --global user.email  
inmacontreras2709@gmail.com  
PS C:\Users\marta\OneDrive\Escritorio\git> git init  
Reinitialized existing Git repository in C:/Users/marta/OneDrive/Escritorio/git/.git/  
PS C:\Users\marta\OneDrive\Escritorio\git> git init  
Initialized empty Git repository in C:/Users/marta/OneDrive/Escritorio/git/.git/  
PS C:\Users\marta\OneDrive\Escritorio\git> █
```

En este caso me lanzó el error de que ese archivo ya tenía un repositorio enlazado como no estábamos seguros que repositorio era lo mejor es eliminarlo.

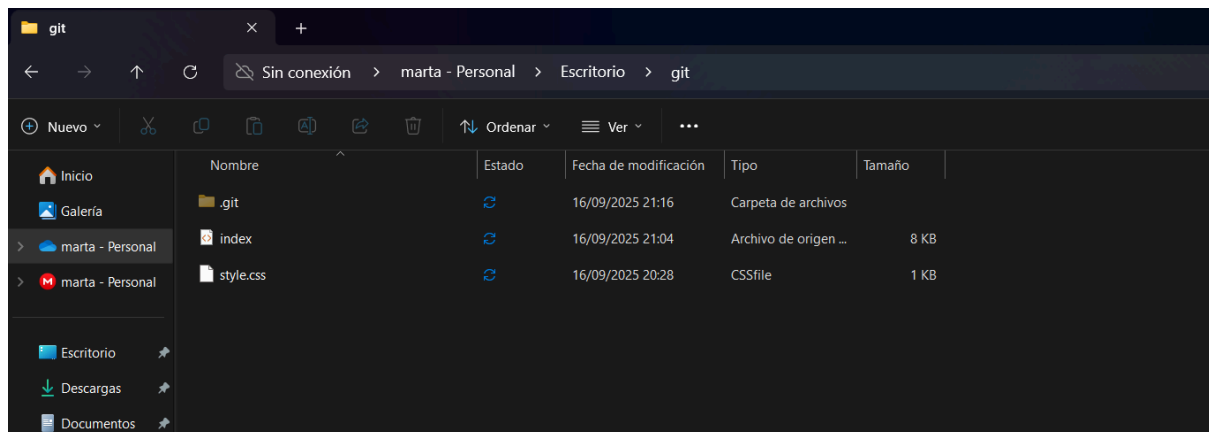
¿Cómo eliminar esa carpeta? es un elemento oculto se muestra así

Vamos a buscar la carpeta de nuestro proyecto en nuestro caso se llama git la abrimos y vemos dos carpetas y vamos a mostrar los elementos ocultos



y ahí ya vemos la carpeta, vamos a eliminarla vamos a volver a iniciar git y volveremos a crear nuestra carpeta ya esta vez con nuestro usuario correcto

```
PS C:\Users\marta\OneDrive\Escritorio\git> git init
Reinitialized existing Git repository in C:/Users/marta/OneDrive/Escritorio/git/.git/
PS C:\Users\marta\OneDrive\Escritorio\git> git init
Initialized empty Git repository in C:/Users/marta/OneDrive/Escritorio/git/.git/
PS C:\Users\marta\OneDrive\Escritorio\git> █
```



Una vez eliminada la carpeta volvemos a inicializar git init y debe inicializar empty es decir la última línea.

```
PS C:\Users\marta\OneDrive\Escritorio\git> git config --global user.email
inmacontreras2709@gmail.com
PS C:\Users\marta\OneDrive\Escritorio\git> git init
Reinitialized existing Git repository in C:/Users/marta/OneDrive/Escritorio/git/.git/
PS C:\Users\marta\OneDrive\Escritorio\git> git init
Initialized empty Git repository in C:/Users/marta/OneDrive/Escritorio/git/.git/
PS C:\Users\marta\OneDrive\Escritorio\git> █
```

10. Haz tu primer "commit":

Ahora que el repositorio está inicializado, el siguiente paso es decirle a Git qué archivos quieres que incluya en tu primer "punto de guardado" o "commit".

Agrega los archivos para rastrearlos. Escribe este comando git add .

en la terminal y presiona Enter:

El punto (.) le dice a Git que agregue todos los archivos en tu carpeta. Si quieres

agregar un sólo archivo, reemplazamos el punto por el nombre del archivo (por ejemplo, `git add index.html`).

- Ejecuta `git add .` para preparar todos tus archivos. (atención al punto `git add .`)
- Ejecuta `git commit -m "Primer commit: subo mis archivos"`.

```
C:\Users\inmac\Desktop\mis cursos creados\curso web>git add .
warning: in the working copy of 'cssgeneral.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'itinerarioformativo.html', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'ud01Lenguajes-de-marcas/tema01-caracteristicas.html', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'ud01Lenguajes-de-marcas/tema02-estructura.html', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'ud01Lenguajes-de-marcas/tema03-navegadores.html', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'ud01Lenguajes-de-marcas/tema04-formato.html', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'ud01Lenguajes-de-marcas/tema05-planes.html', LF will be replaced by CRLF the next time Git touches it
```

Este aviso que ves no es un error, es una advertencia de Git sobre los **finales de línea** (line endings). Puedes ignorarla, ya que no impedirá que subas tus archivos ni afectará la funcionalidad de Git.

```
C:\Users\inmac\Desktop\mis cursos creados\curso web>git commit -m "Primer commit: subo mis archivos"
```

`git commit -m "Primer commit: subo mis archivos"`

```
C:\Users\inmac\Desktop\mis cursos creados\curso web>git commit -m "Primer commit: subo mis archivos"
[main de7fed8] Primer commit: subo mis archivos
19 files changed, 115 insertions(+), 55 deletions(-)

C:\Users\inmac\Desktop\mis cursos creados\curso web>
```

¿Qué es lo que hace exactamente esta orden?

Cuando ejecutas los comandos `git add .` y `git commit -m "..."`, estás realizando el proceso de guardar tus cambios en el repositorio local. Es decir si entras en tu repositorio seguirá vacío pero ya estas en el camino.

Aquí te explico lo que hace cada uno:

git add .

Este comando le dice a Git que quieres **preparar** o **poner en escena** (del inglés "stage") todos los archivos que has modificado, creado o eliminado en tu

proyecto. Es la acción de añadir archivos a un área temporal llamada el **"área de preparación"** o **"staging area"**.

. (el punto): Este es un atajo que significa "todos los archivos". También puedes especificar archivos individuales, por ejemplo, `git add index.html` si solo quieres añadir ese archivo.

El área de preparación es como un borrador o una "cesta de la compra" donde seleccionas los cambios que quieres incluir en el siguiente commit.

git commit -m "Mensaje"

Este comando es el que guarda de forma permanente la "instantánea" de tus archivos que están en el área de preparación.

- `git commit`: Es la acción de guardar los cambios.
- `-m`: Es un "flag" o bandera que significa "message" (mensaje).
- **"Mensaje"**: Es una descripción corta y clara de lo que hiciste en este commit. Es crucial para que tú y otros colaboradores puedan entender fácilmente el historial de cambios. No es obligatorio ponerle un mensaje pero si recomendable.

Una vez que ejecutas `git commit`, los cambios que estaban en el área de preparación se guardan en el historial del repositorio. Ahora, tus archivos están seguros y Git tiene un registro de esta versión específica de tu proyecto. El siguiente paso sería `git push` para subir estos cambios al repositorio en línea.

```
Reinitialized existing Git repository in C:/Users/marta/OneDrive/Escritorio/git/.git/
PS C:\Users\marta\OneDrive\Escritorio\git> git init
Initialized empty Git repository in C:/Users/marta/OneDrive/Escritorio/git/.git/
PS C:\Users\marta\OneDrive\Escritorio\git> git add .
PS C:\Users\marta\OneDrive\Escritorio\git> git commit -m "Primer commit: se añadieron los archivos iniciales"
[master (root-commit) f2b4dcf] Primer commit: se añadieron los archivos iniciales
 2 files changed, 168 insertions(+)
 create mode 100644 index.html
 create mode 100644 style.css
PS C:\Users\marta\OneDrive\Escritorio\git> █
```

Es decir todavía no hay nada subido a tu git

11. Enlazar el repositorio local con el remoto (git remote)

El comando `git remote add origin <URL-del-repositorio>` le dice a Git que quieres establecer un enlace. Es como si le dieras a tu proyecto local la dirección de tu proyecto en la nube.

- `git remote`: Un comando para gestionar las conexiones a repositorios remotos.

- add origin: Le das a esta conexión el nombre origin por convención. Es el nombre estándar para el repositorio remoto principal.
- <URL-del-repositorio>: Es la dirección de tu repositorio vacío en GitHub.

Este paso es fundamental porque, hasta que no lo hagas, tu commit solo existe en tu máquina. Git no sabe adónde debe enviar tus archivos. Es la orden que le indica el camino le dice a nuestra máquina donde va a mandar nuestros archivos.

Esta orden la podemos ver en nuestro git en nuestro repositorio vacío de hecho verás varias líneas de instrucciones puedes copiar en este caso las que nos interesa. Deben ejecutarse por independiente y ahora vamos a enlazar con la línea remote add.

git remote add origin https://github.com/inma2709/cursoweb.git: Este comando establece el enlace entre tu repositorio local y el repositorio remoto que creaste en GitHub. Le dice a Git la dirección de tu proyecto en la nube.

12. Preparar y subir los cambios (git branch y git push)

Después de enlazar los repositorios, la mejor práctica es asegurarte de que tu rama principal se llame main y luego subir tus archivos.

1. **Asegurar la rama principal:** El comando git branch -M main cambia el nombre de tu rama local (si se llamaba master por defecto) a main. Esto es importante porque main es el nombre estándar actual y te evitará problemas de compatibilidad en el futuro.

Vamos a averiguar como se llama la rama
Este comando le dice a Git:

- push: Sube los cambios.
- -u: Configura el origen remoto para que no tengas que escribirlo de nuevo en el futuro.
- origin: Es el nombre predeterminado de la conexión a tu repositorio remoto.
- main: Es el nombre de la rama que estás subiendo.

¿Donde los vamos a subir? hay que verificar como se llama nuestra rama se puede llamar main o master

```
PS C:\Users\marta\OneDrive\Escritorio\git> git branch
* master
PS C:\Users\marta\OneDrive\Escritorio\git> 
```

En este caso se llama master le vamos a cambiar el nombre a main si no nos va a dar problemas en las siguientes subidas pues siempre va a buscar la rama main por defecto

```
PS C:\Users\marta\OneDrive\Escritorio\git> git branch -M main
PS C:\Users\marta\OneDrive\Escritorio\git> git branch
* main
PS C:\Users\marta\OneDrive\Escritorio\git> █
```

git branch -M main

El comando git branch -M main se usa para renombrar tu rama actual a main. Esto es útil y necesario si la rama por defecto que se creó en tu máquina se llama master (el nombre antiguo) y quieres que coincida con el estándar actual de la industria, que es main

git push -u origin main: Este es el comando final que sube tus archivos desde la rama local (main) al repositorio en línea (origin). El -u le dice a Git que esta es la rama predeterminada para el futuro.

13. Subir los archivos: Una vez que la conexión está establecida y la rama tiene el nombre correcto, usas el comando git push para "empujar" (enviar) los commits locales a GitHub.

git push -u origin main

The screenshot shows a code editor with HTML code. The code includes a main section with a paragraph and a section titled 'Paso 3: Crear un repositorio'. The terminal window at the bottom shows the following commands and output:

```
Users\marta\OneDrive\Escritorio\git> git branch
er
Users\marta\OneDrive\Escritorio\git> git branch -M main
Users\marta\OneDrive\Escritorio\git> git branch
* main
Users\marta\OneDrive\Escritorio\git> git remote add origin https://github.com/inma2709/cursoweb.git
Users\marta\OneDrive\Escritorio\git> git push -u origin main
```

Overlaid on the terminal is a 'Connect to GitHub' dialog box. The dialog has a title bar 'Connect to GitHub' and a close button. It says 'GitHub Sign in' and has two tabs: 'Browser/Device' (selected) and 'Token'. Under 'Browser/Device', there are two buttons: 'Sign in with your browser' (highlighted in green) and 'Sign in with a code'. At the bottom, it says 'Don't have an account? Sign up'.

Esta es una ventana de seguridad que te pide **iniciar sesión en tu cuenta de GitHub** para permitir que Git suba tus archivos.

¿Qué hacer ahora?

Debes hacer clic en **Sign in with your browser**. Esto abrirá tu navegador web, donde serás redirigido a una página de GitHub para que inicies sesión.

Paso a paso:

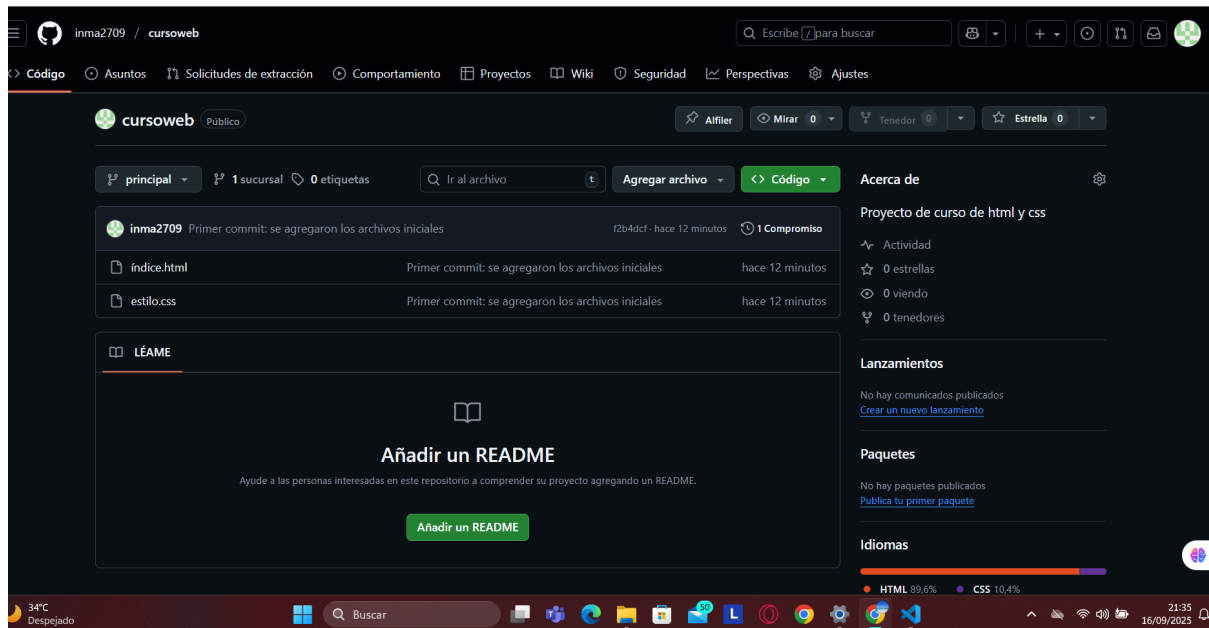
1. Haz clic en el botón verde **Sign in with your browser**.
2. Tu navegador se abrirá y te pedirá que inicies sesión en GitHub.
3. Una vez que inicies sesión, te pedirá que autorices a Git a acceder a tu cuenta. Acepta la autorización.
4. Luego de la autorización, el navegador te mostrará un mensaje para que regreses a Visual Studio Code.

Una vez que regreses a VS Code, la terminal continuará el proceso de git push, y tus archivos se subirán a tu repositorio de GitHub. ¡Ya casi lo tienes!



Authentication Succeeded

You may now close this tab and return to the application.



- git push: El comando para subir los cambios.

Nuestros archivos están en el repositorio

—————En resumen, el proceso es: commit (guardar localmente) → remote add (conectar) → push (subir a GitHub).

¿Cómo funciona el proceso de la subida a git?

El proceso de subir archivos a GitHub se divide en dos fases principales: una fase local y una fase remota. El git commit es el último paso de la fase local, mientras que git remote add origin marca el inicio de la fase remota.

1. Fase local (commit):

- git init: Inicializas el repositorio local en tu computadora.
- git add .: Le dices a Git qué archivos quieres incluir en tu próximo guardado.
- git commit -m "...": Creas una instantánea local de tu proyecto. En este punto, los cambios están guardados de forma segura en tu disco duro (dentro de la carpeta oculta .git), pero aún no han salido de tu ordenador.

2. Fase remota (remote y push):

- git remote add origin ...: Este comando enlaza tu repositorio local con el repositorio vacío que creaste en GitHub. Le estás dando a Git la dirección (URL) de tu proyecto en la nube. Con esta orden, Git ya sabe adónde enviar tus cambios.

- `git push -u origin main`: Finalmente, "empujas" (push) los commits que guardaste localmente para que se suban a la rama main del repositorio remoto (origin).

En resumen, no puedes subir un "punto de guardado" a un lugar que Git aún no conoce. Primero, debes crear el punto de guardado (commit) en tu máquina y, luego, establecer el enlace (`remote add origin`) para que Git sepa adónde enviar ese guardado (push).

Y a partir de ahora ¿Qué?

Ya sabes cómo hacer tu primer *commit* y subir tu proyecto a GitHub. A partir de ahora, el proceso para subir los cambios será mucho más sencillo. No necesitas inicializar el repositorio, ni enlazarlo de nuevo; solo tienes que seguir estos tres pasos:

Cómo subir los cambios a partir de ahora

Agrega los cambios: Cuando modifiques, añadas o elimines archivos, dile a Git que los tenga en cuenta para el próximo *commit*.

`git add .`

1. El comando `git add .` le dice a Git que agregue todos los archivos que has cambiado. Si solo has modificado un archivo, puedes ser más específico, por ejemplo: `git add mi-archivo.html`.

Crea un *commit*: Haz un "punto de guardado" de los cambios con un mensaje claro y conciso.

`git commit -m "Descripción de los cambios"`

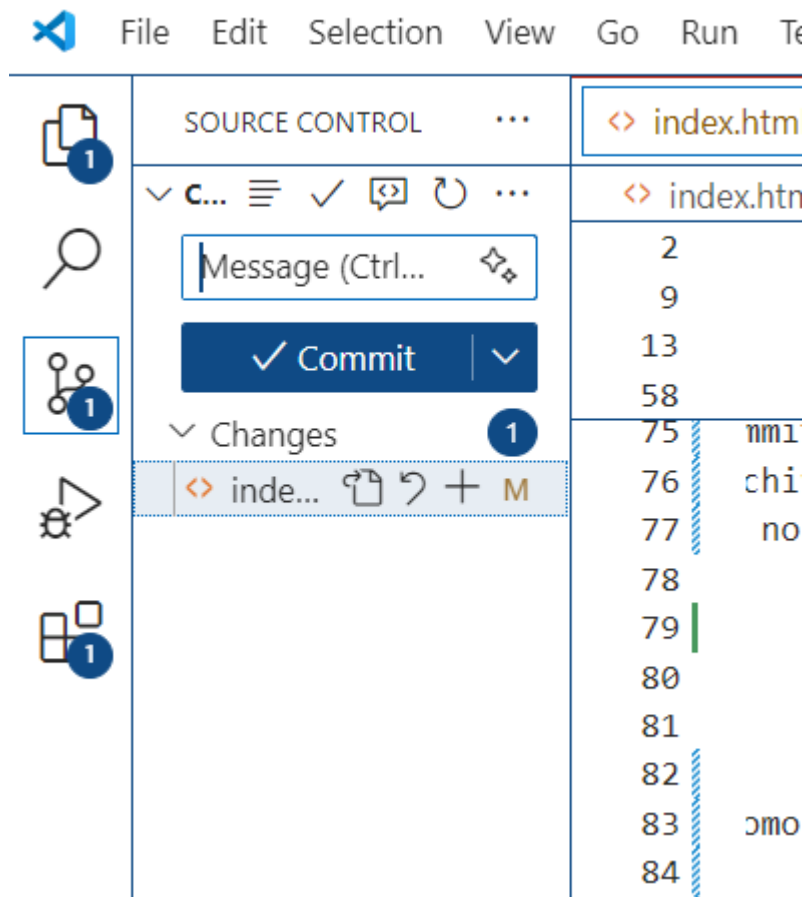
2. Asegúrate de que la descripción sea útil y breve (por ejemplo: "Se corrigió el color del encabezado" o "Se añadió la sección de contacto").

Sube los cambios: Envía los *commits* locales a tu repositorio en GitHub.

`git push`

3. Como ya configuraste el enlace la primera vez con `git push -u origin main` (o `master`), ahora solo necesitas usar `git push`. Git ya sabe a dónde subir tus cambios.

¡Y listo! Con estos tres comandos (`add`, `commit` y `push`) podrás mantener tu repositorio de GitHub siempre actualizado con los cambios que hagas en tu proyecto local. Es el ciclo de trabajo básico de Git que usarás a diario.



Cómo hacer un commit desde la consola

Guarda los cambios: Antes de cualquier cosa, asegúrate de que el archivo `index.html` no tenga la "bolita blanca" en la pestaña que indica que hay cambios sin guardar. Puedes hacerlo presionando `Ctrl + S`.

Abre la terminal: En Visual Studio Code, ve a Terminal > Nueva terminal.

Agrega los cambios: Escribe `git add .` para que Git prepare todos los archivos modificados.

```
git add .
```

Haz el commit: Ahora, haz el *commit* con un mensaje que describa tus cambios.

```
git commit -m "Se agregaron los pasos para subir cambios"
```

Sube los cambios a GitHub: Finalmente, usa `git push` para subir los cambios a tu repositorio en línea.

```
git push
```

Este es el proceso completo para hacer un *commit* y subir los cambios desde la terminal de VS Code, sin usar la interfaz gráfica. Si en el futuro te sale el aviso de la imagen, es una buena práctica hacer clic en **"Save All & Commit Changes"** para que VS Code guarde los archivos automáticamente antes de hacer el *commit*.

Pasos para usar la interfaz de Visual Studio Code.

1. **Abre el panel de Control de Código Fuente:** Haz clic en el tercer icono de la barra lateral izquierda, que parece una **rama de árbol** o una "Y" invertida. Si has hecho cambios en tus archivos, verás un número en el icono que indica la cantidad de archivos modificados.
2. **Agrega los cambios:** En la sección "Cambios" (Changes), verás los archivos que has modificado.
 - Para agregar todos los cambios, haz clic en el signo **+** que aparece junto a la sección "Cambios".
 - Si solo quieres agregar un archivo específico, pasa el cursor sobre él y haz clic en el signo **+** que aparece a la derecha.
3. **Crea el *commit*:** Una vez que los archivos estén en el área de "Cambios preparados" (Staged Changes), escribe tu mensaje de *commit* en el cuadro de texto que dice "Mensaje". Este mensaje debe describir los cambios que has hecho.
4. **Haz el *commit*:** Haz clic en el botón **"Commit"** que está justo encima del cuadro de texto del mensaje. Si no has agregado los archivos, este botón te dará la opción de "Commit & Stage All".
5. **Sube los cambios:** Después de hacer el *commit*, los cambios se guardan localmente. Para subirlos a GitHub, haz clic en el botón **"Sincronizar cambios"** (Sync Changes) que aparece en la parte inferior de la ventana.

Usar este panel es una excelente manera de visualizar tus cambios y de realizar los comandos `git add`, `git commit` y `git push` con un solo clic, sin tener que escribir los comandos manualmente.