



Eidos

ALWAYS LEARNING

Tutorial Git + GitHub

1. [¿Qué son Git y GitHub? ¿Para qué sirven?](#)
2. [Instalar Git](#)
3. [Crear cuenta en GitHub](#)
4. [Configurar Git](#)
5. [Crear repositorio local \(Git\)](#)
6. [Crear repositorio remoto y vincular con repo local \(GitHub\)](#)
7. [Actualizar cambios](#)
8. Subir mi web a GitHub Pages

1. ¿Qué son Git y GitHub? ¿Para qué sirven?

Git es un sistema de control de versiones, permite que cualquier versión del archivo esté disponible cuando se desee.

Es de gran ayuda también para el trabajo en equipo, pues permite registrar qué cambios se realizan versión a versión y quién los realiza.

GitHub es una plataforma web que permite alojar y compartir repositorios Git. En este curso lo utilizaremos para:

- a. **Compartir nuestro código cuando necesitemos ayuda del resto del curso o de quien lo facilite.** Es importante entender que para resolver un problema de nuestra web probablemente haya que revisar diversas partes del código y no alcance con una captura de pantalla o copiar una línea.
- b. **Tener copia de nuestro código en la nube**, así en caso de que tengamos algún problema con nuestra máquina el código se mantiene a salvo.
- c. **Subir nuestra web a Github Pages** que simula un servidor y provee una url para visitar nuestra página y entender cómo la verá el resto desde sus navegadores.

¿Quieren saber más? Pueden [ingresar aquí](#) para una explicación más detallada.

2. Instalar Git

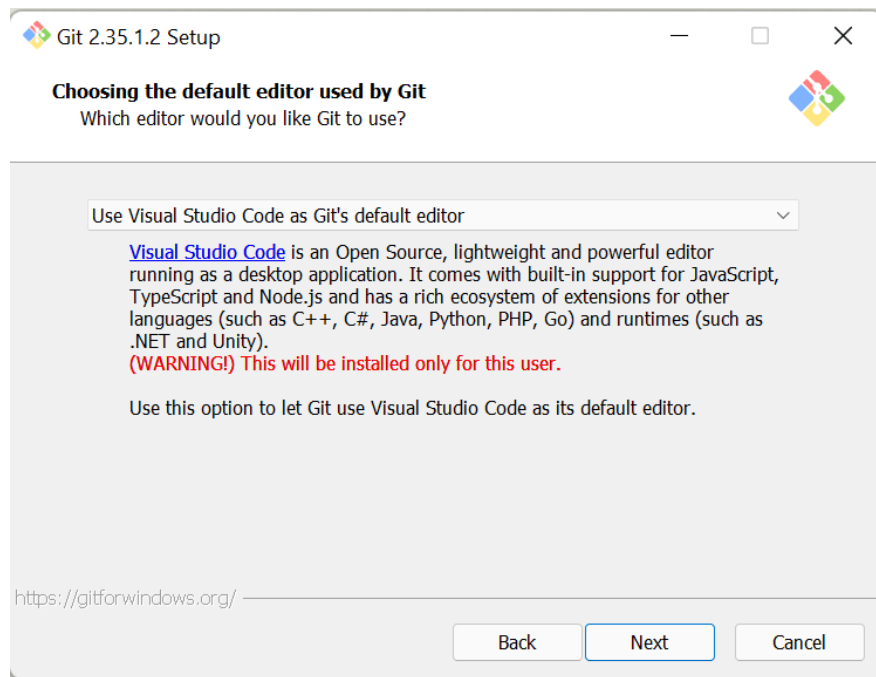
En el siguiente enlace encontrarás los instaladores e indicaciones para distintos sistemas operativos: <https://git-scm.com/downloads>

Este punto lo haremos **solo una vez**. Si ya instalaste Git en tu pc, puedes continuar al punto siguiente.

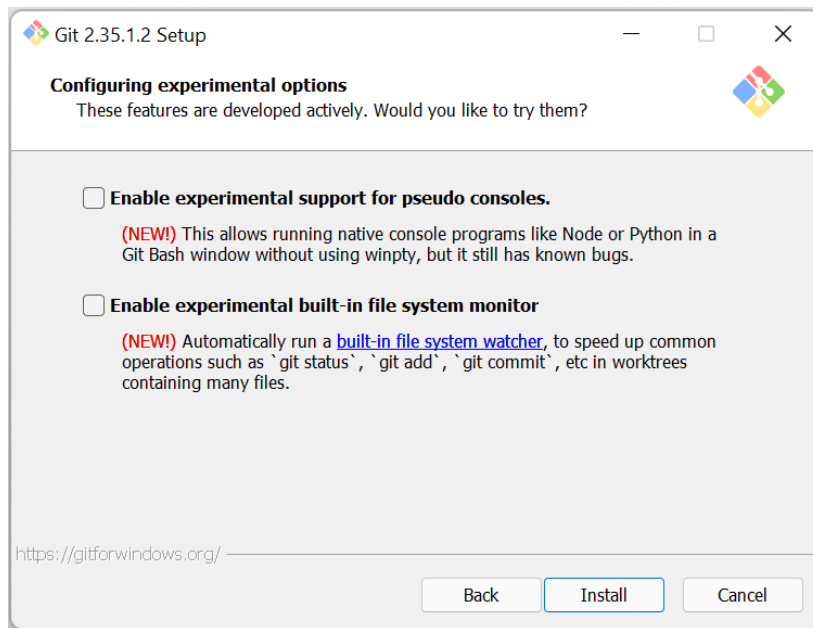
¿Aún no instalaste Git? No te preocupes, es muy sencillo.

Debes seguir los pasos que marcamos aquí abajo. Verás que el programa te va guiando en la instalación, **en las capturas te mostramos las opciones que debes marcar distinto que las que vienen por defecto**. En el resto alcanza con poner “next”.

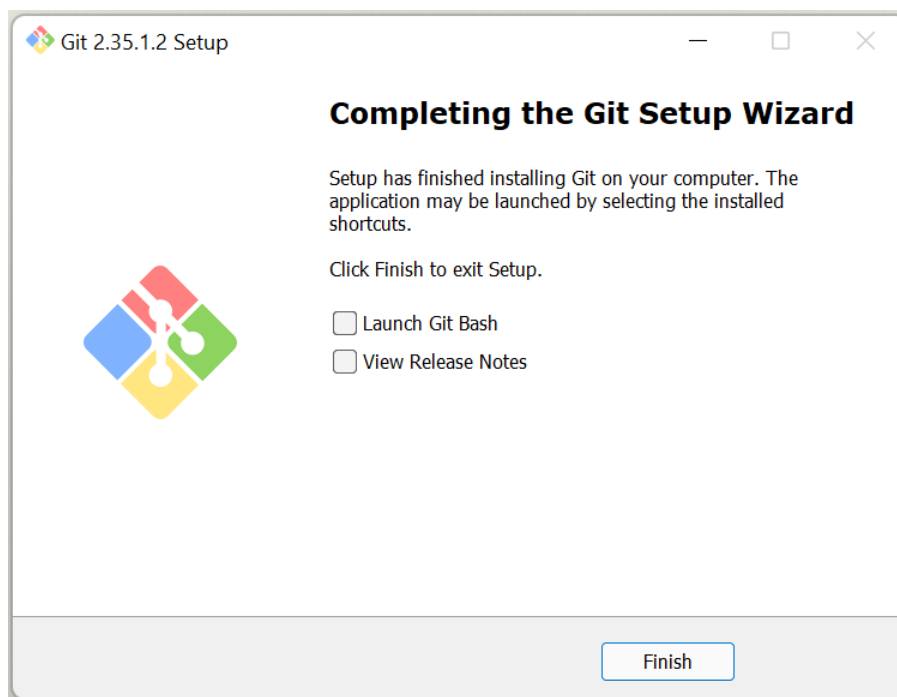
1. Ingresar al enlace <https://git-scm.com/downloads> y seleccionar tu sistema operativo
2. Seguir las sugerencias de instalación
3. Seleccionar Visual Studio Code como editor por default.



4. Antes de tocar el botón Instalar, aparecerá esta pantalla. Avanzar con los dos casilleros en blanco.



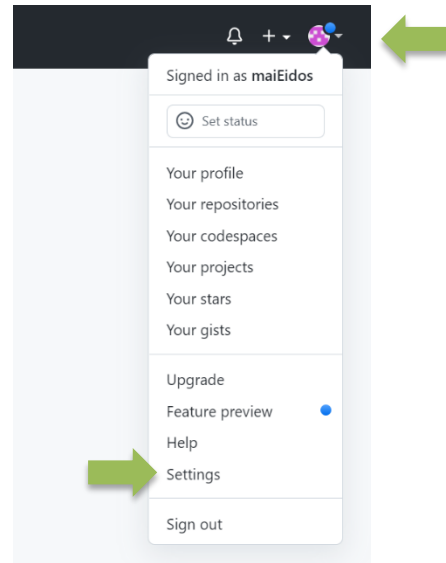
5. En caso de querer abrir la consola al finalizar la instalación seleccionar "Launch Git Bash". De lo contrario finalizar la instalación con ambos casilleros en blanco.



6. Crear una cuenta en Github

- Ingresar en el sitio web de GitHub y crear una cuenta: <https://github.com/>
- Generar un "token de acceso" como lo indicamos en las siguientes capturas

Selecciona el ícono de tu perfil y luego la opción "Settings". Se abrirá un nuevo recuadro donde deberás seleccionar "Generate new token"



Settings / Developer settings

GitHub Apps
OAuth Apps
Personal access tokens

Personal access tokens

Generate new token

Need an API token for scripts or testing? [Generate a personal access token](#) for quick access to the [GitHub API](#).

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

repomai

What's this token for?

Expiration *

No expiration

The token will never expire!

En la página de título "New personal access token" elegir un nombre para el repositorio, seleccionar "No expiration" y luego tildar la opción "repo". Ir al final de la página y clicar "Generate token".

GitHub strongly recommends that you set an expiration date for your token to help keep your information secure. [Learn more](#)

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

- ☒ **repo** Full control of private repositories
 - ☒ repo:status Access commit status
 - ☒ repo_deployment Access deployment status
 - ☒ public_repo Access public repositories
- ☐ **admin:org_hook** Full control of organization hooks
- ☐ **gist** Create gists
- ☐ **notifications** Access notifications
- ☐ **user** Update ALL user data
 - ☐ read:user Read ALL user profile data
 - ☐ user:email Access user email addresses (read-only)
 - ☐ user:follow Follow and unfollow users
- ☐ **delete_repo** Delete repositories
- ☐ **write:discussion** Read and write team discussions
 - ☐ read:discussion Read team discussions
- ☐ **admin:enterprise** Full control of enterprises
 - ☐ manage_runners:enterprise Manage enterprise runners and runner-groups
 - ☐ manage_billing:enterprise Read and write enterprise billing data
 - ☐ read:enterprise Read enterprise profile data
- ☐ **admin:pgp_key** Full control of public user GPG keys ([Developer Preview](#))
 - ☐ write:pgp_key Write public user GPG keys
 - ☐ read:pgp_key Read public user GPG keys

Generate token


Cancel

Personal access tokens

[Generate new token](#)[Revoke all](#)

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

✓ 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 

[Delete](#)

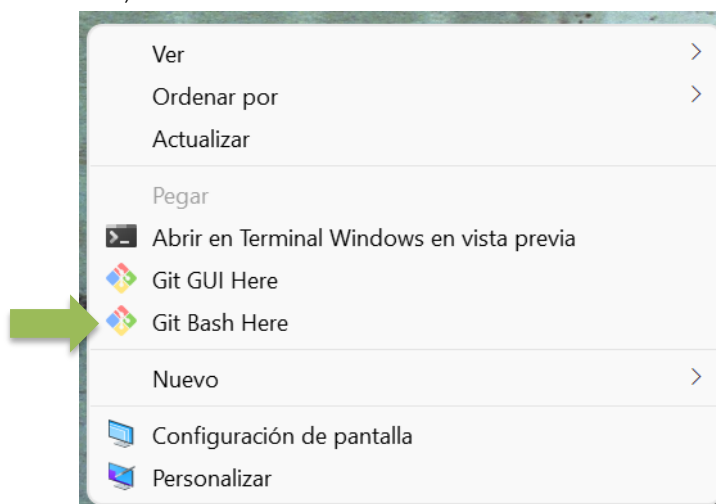
Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Copiar el token clickeando la imagen señalada (los cuadrados) pegarlo en un block de notas o algún lugar a mano, lo necesitaremos pronto.

7. Configurar Git

Para poder comenzar a crear repositorios locales y conectarlos con GitHub, debemos configurar Git localmente utilizando las credenciales de la cuenta creada en GitHub.

Abrir consola de Git con click derecho y seleccionando “Git Bash here” (desde cualquier carpeta, puede ser desde Escritorio).



Luego vamos a ingresar algunos comandos en la consola, si es tu primera vez trabajando en una terminal puede ser un poco intimidante, pero ¡no te preocupes! Te guiaremos a cada paso.

Importante: la consola tiene sus propias reglas, aunque puede parecer un block de notas, no lo es.

--> Luego de cada línea de código deberás presionar "Enter"

--> Si presionas la tecla "arriba" ↑ copiarás la última línea de código ingresada

--> Para pegar código debes seleccionar click derecho + "Pegar", el atajo ctrl + V no funciona

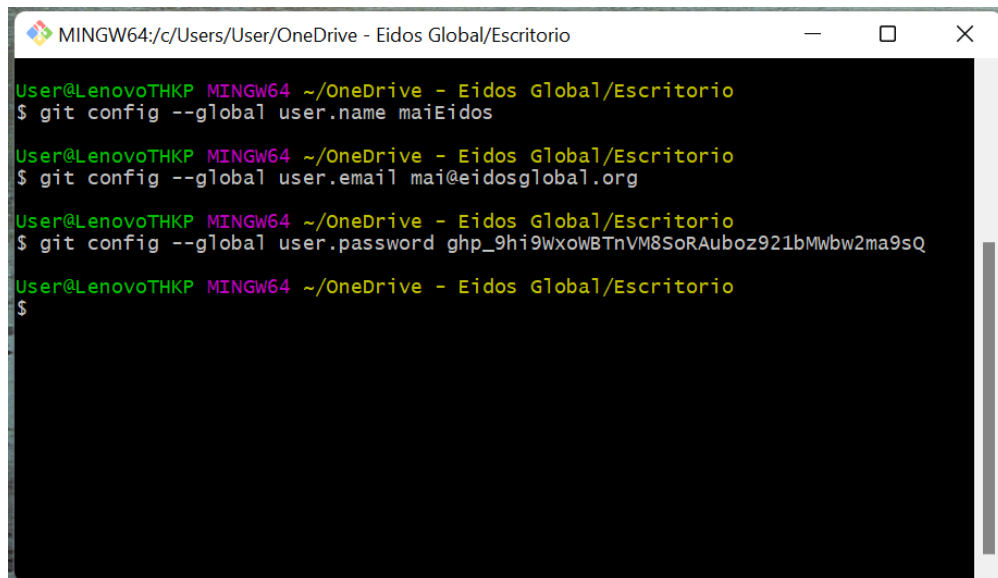
Ahora sí, en la consola escribirán las siguientes líneas con **sus datos de cuenta** de GitHub:

```
git config --global user.name maiEidos
```

```
git config --global user.email mai@eidosglobal.org
```

```
git config --global user.password tokengeneradoengithub
```

Recuerden que esta captura es un ejemplo, deben completar cada línea con los datos de su cuenta.

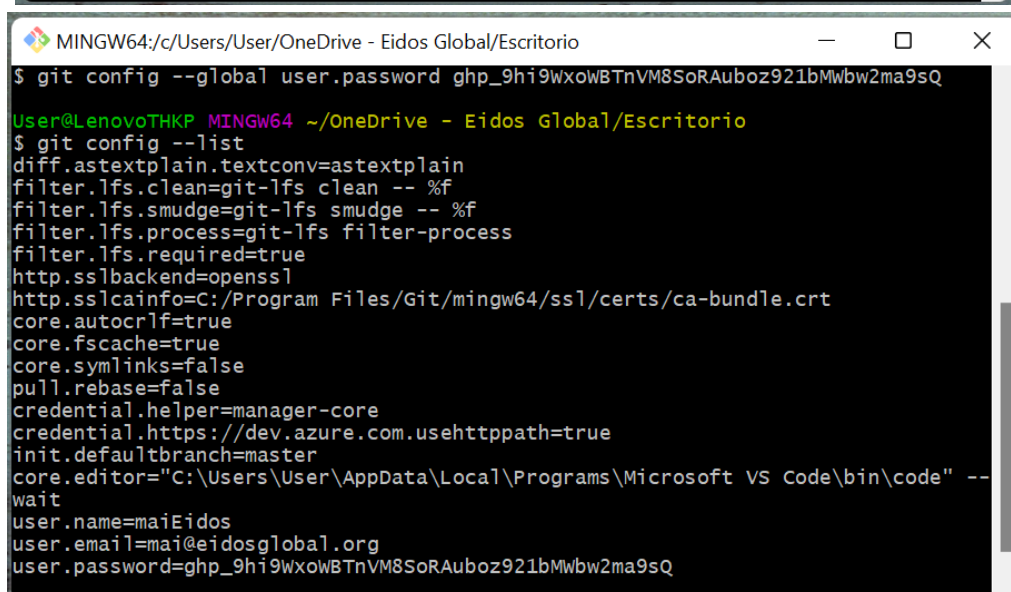


```
MINGW64:/c/Users/User/OneDrive - Eidos Global/Escritorio
User@LenovoTHKP MINGW64 ~/OneDrive - Eidos Global/Escritorio
$ git config --global user.name maiEidos

User@LenovoTHKP MINGW64 ~/OneDrive - Eidos Global/Escritorio
$ git config --global user.email mai@eidosglobal.org

User@LenovoTHKP MINGW64 ~/OneDrive - Eidos Global/Escritorio
$ git config --global user.password ghp_9hi9WxowBTnVM8SoRAuboz921bMwbw2ma9sQ

User@LenovoTHKP MINGW64 ~/OneDrive - Eidos Global/Escritorio
$
```



```
MINGW64:/c/Users/User/OneDrive - Eidos Global/Escritorio
$ git config --global user.password ghp_9hi9WxowBTnVM8SoRAuboz921bMwbw2ma9sQ

User@LenovoTHKP MINGW64 ~/OneDrive - Eidos Global/Escritorio
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fsckcache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
core.editor="C:\Users\User\AppData\Local\Programs\Microsoft VS Code\bin\code" --wait
user.name=maiEidos
user.email=mai@eidosglobal.org
user.password=ghp_9hi9WxowBTnVM8SoRAuboz921bMwbw2ma9sQ
```


Para corroborar que haya funcionado la configuración, pueden escribir este comando:

```
git config --list
```

Al presionar “Enter” enlistará diversas propiedades, vamos a prestarle atención a las últimas tres líneas.

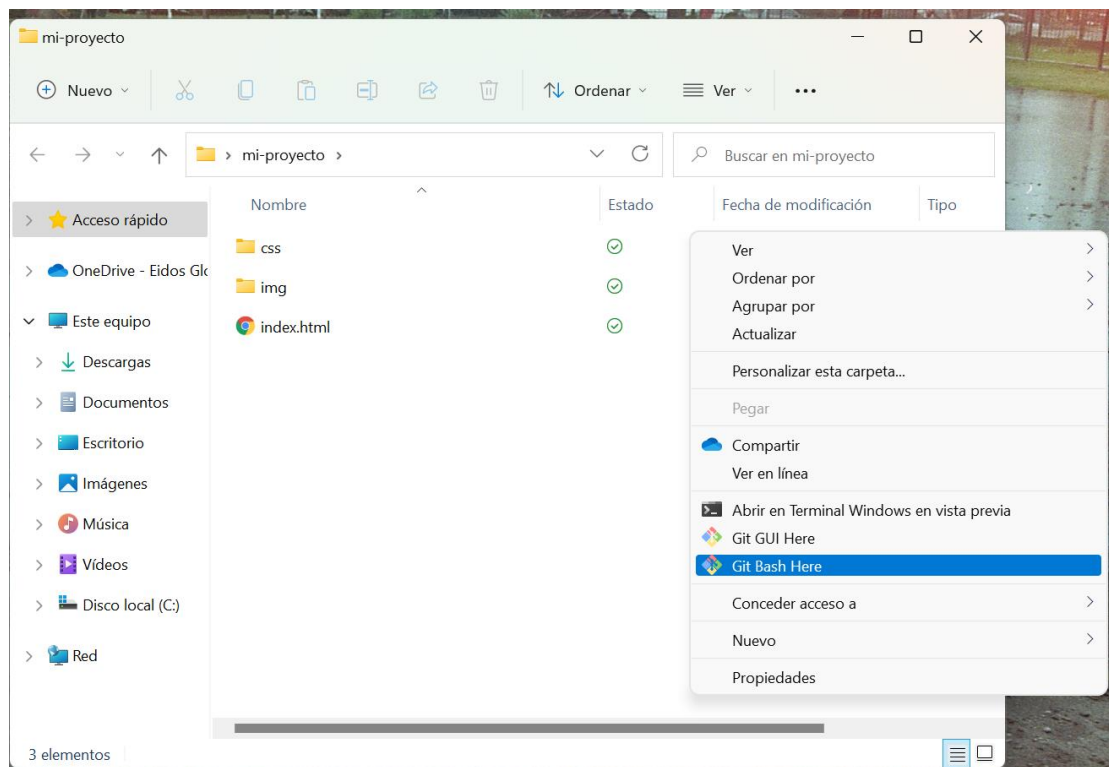
¿Coincide con lo ingresado? En ese caso, hecho. Pasar al siguiente paso.

¿No coincide? No desesperes, puedes volver al punto anterior y volver a escribir las líneas con los datos correctos. No es necesario reiniciar ni borrar nada, en la consola al escribir una nueva línea con el mismo comando **sobreescribimos el anterior**. Por eso, si hubiera un error de tipeo (por ejemplo en el nombre de usuario) no es necesario modificar las otras dos propiedades, se puede volver a escribir el nombre de usuario (`git config --global user.name tunombredeusuario`) y se guardará la última línea ingresada.

5. Crear repositorio local (Git)

Para esto es necesario abrir la consola “git bash” **desde la carpeta donde trabajaremos** (click derecho dentro de la carpeta, “Git Bash here”).

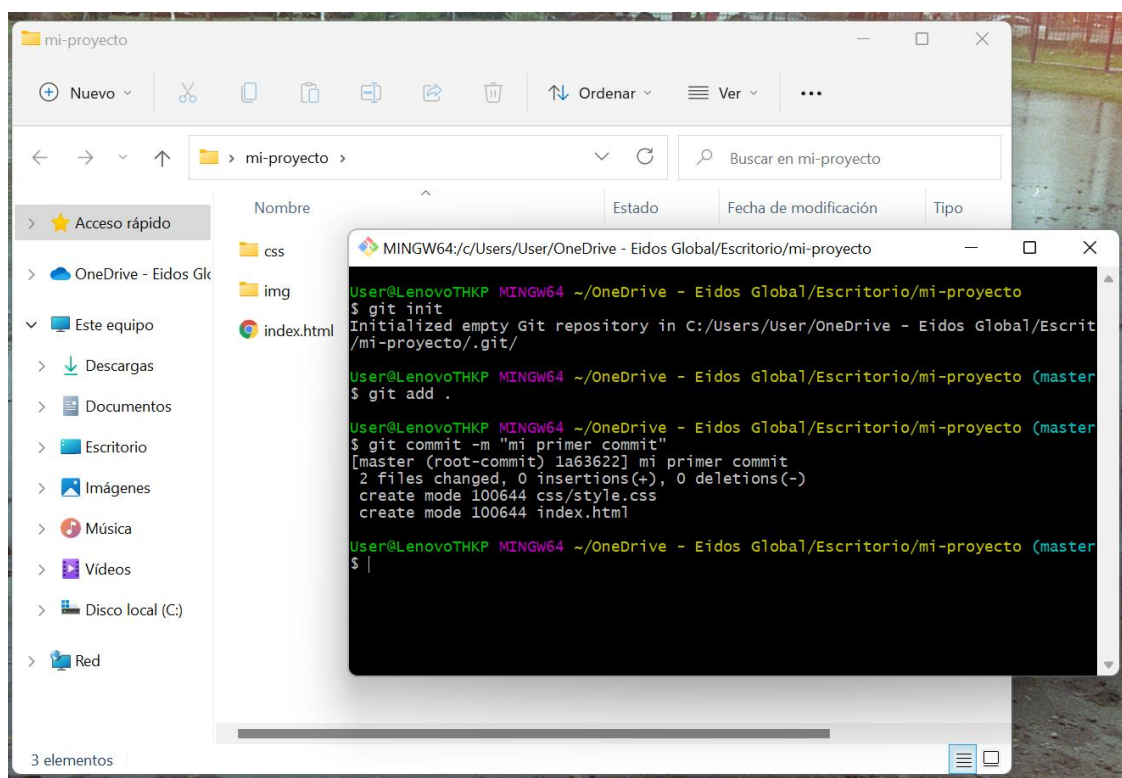
Usaremos la misma consola para el paso 6, ¡no la cierres!



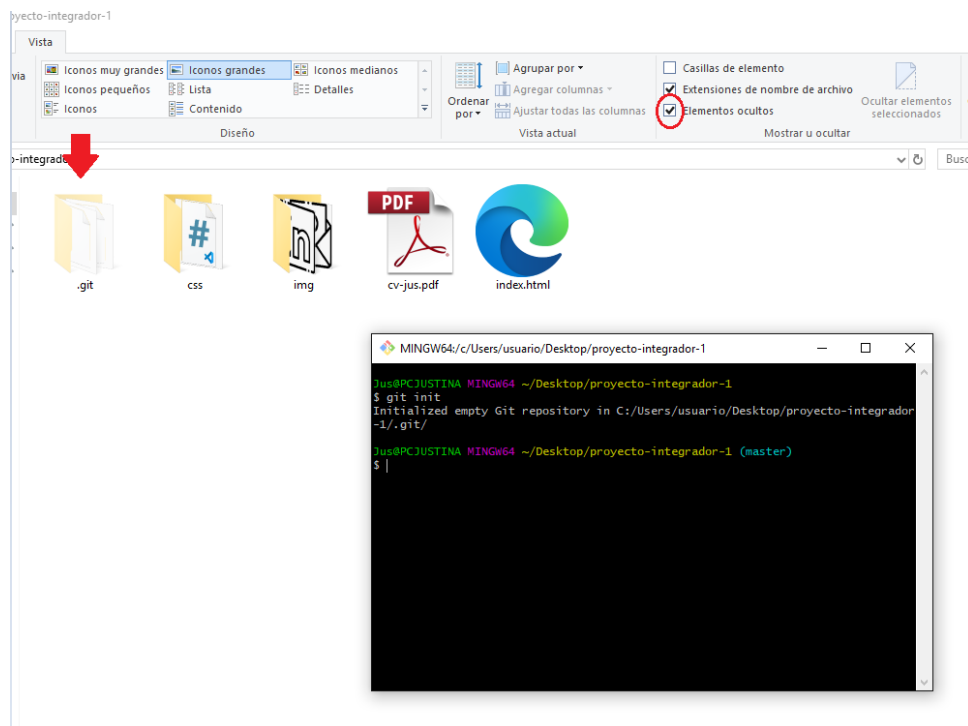
Con los siguientes comandos vamos a crear un repositorio en la carpeta de nuestro proyecto y luego vamos a guardar los cambios que hayamos hecho.

- a. **git init** para inicializar el repositorio local (este comando se escribe una única vez, al iniciar el repositorio)
- b. **git add .** para agregar todos los cambios hechos
- c. **git commit -m "mensaje del commit"** para agregar los cambios en el repositorio, en el mensaje se describe de forma muy resumida los cambios hechos en el código.

Revisá la captura de abajo para ver qué debería volver Git luego de cada "Enter".
¿Todo en orden? ¡Seguimos!



TIP para asegurarse que hayan creado correctamente su repositorio local: una vez hayan escrito el comando **git init** para inicializar el repositorio local, si habilitan "ver archivos ocultos" en el explorador de archivos, verán esta carpeta llamada ".git"



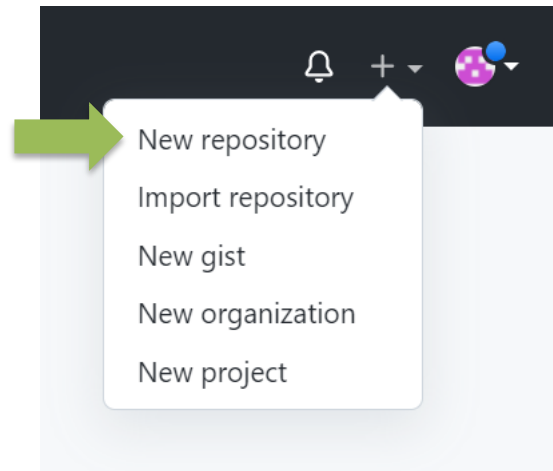
Si algo llega a salir mal en los siguientes pasos y tienen muchos errores, pueden eliminar esa carpeta, lo cual eliminará el repositorio local.

Luego, vuelven a iniciar otro repositorio local con **git init** y realizan los pasos nuevamente. Por supuesto, esto no afecta al repositorio remoto.

6. Crear y vincular repositorio remoto (GitHub)

Para poder guardar nuestros cambios en la nube necesitamos vincular nuestro repositorio local con un repositorio remoto.


Lo primero que haremos será crearlo, para eso es necesario entrar a nuestra cuenta de GitHub y seguir estos pasos:



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner *

 maiEidos ▾

Repository name *

/ mi-proyecto ✓

Great repository names are short, lowercase, and contain only alphanumeric characters and hyphens. mi-proyecto is available. Need inspiration? How about **sturdy-octo-waffle**?

Description (optional)

☒



Public

Anyone on the internet can see this repository. You choose who can commit.

☐



Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore

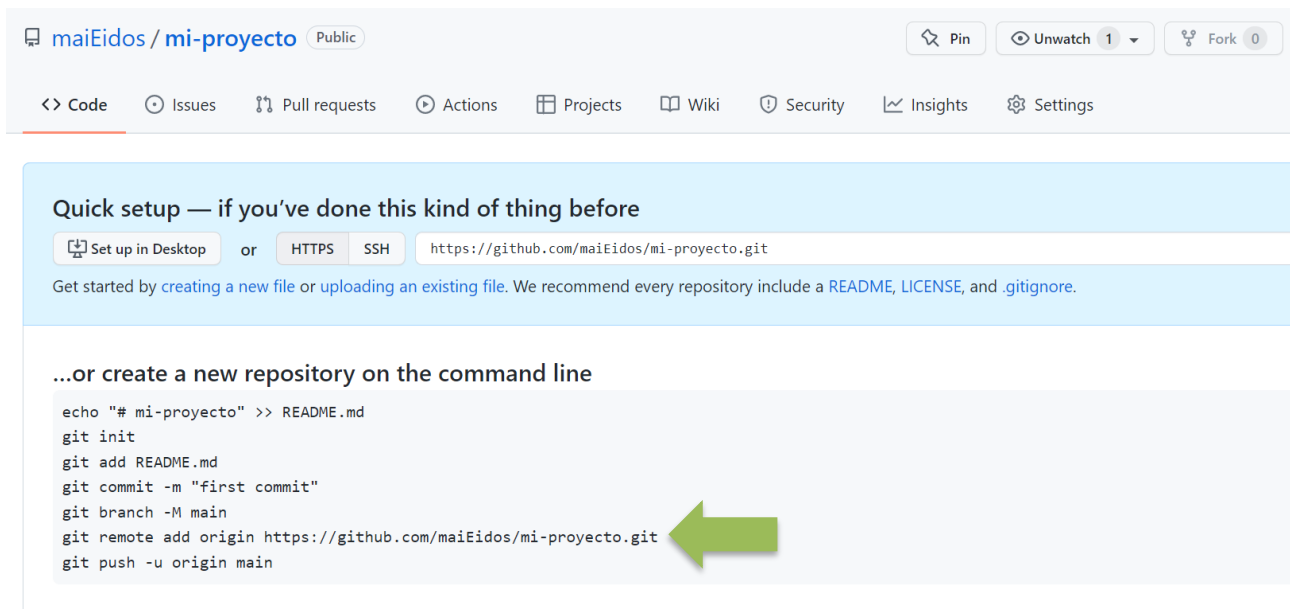
Choose which files not to track from a list of templates. [Learn more.](#)

☐ Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

 You are creating a public repository in your personal account.

Create repository



The screenshot shows the GitHub interface for creating a new repository. At the top, it says 'maiEidos / mi-proyecto' with a 'Public' badge. Below this is a navigation bar with links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The main content area has a light blue header 'Quick setup — if you've done this kind of thing before' with buttons for 'Set up in Desktop', 'HTTPS', and 'SSH'. Below this is a text box with the URL 'https://github.com/maiEidos/mi-proyecto.git'. A second light blue box says 'Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.' Below that is a section '...or create a new repository on the command line' with a code block containing the following commands:

```
echo "# mi-proyecto" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/maiEidos/mi-proyecto.git
git push -u origin main
```

A green arrow points to the last line of the code block.

¡Listo!

Ahora es momento de copiar la línea señalada que dice "git remote add origin" y un link.

Ya tenemos un repositorio local (Git) y un repositorio remoto (GitHub).

¡Necesitamos vincularlos!

Para eso, volveremos a la misma consola que ya habíamos usado (si cerraste la consola, no hay problema, puedes volver a abrirla desde la misma carpeta).

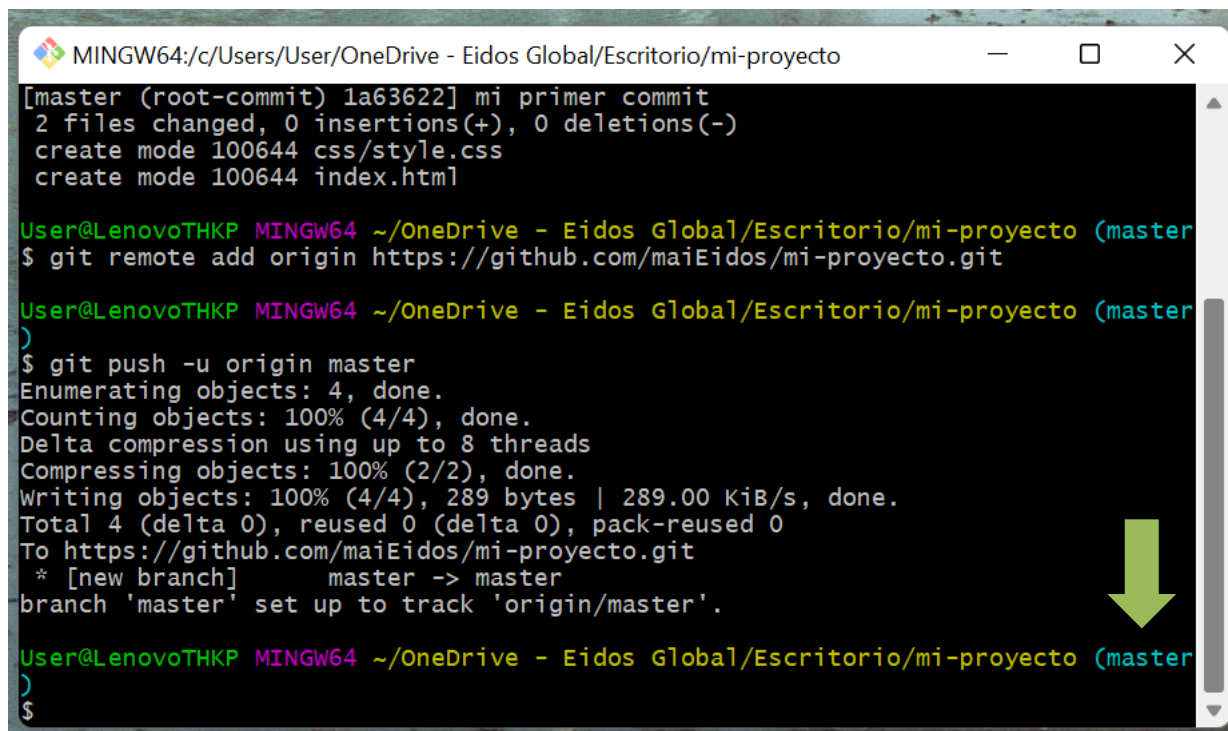
Vamos a **pegar** la línea que habíamos copiado para vincular repo local con repo remoto:

git remote add origin url-del-proyecto para vincular repo local con repo remoto

Luego escribiremos esta línea para enviar los cambios realizados al repositorio remoto y actualizarlo:

```
git push -u origin master
```

¿Por qué "master"? En la captura señalamos dónde dice "master", es la ubicación dentro de nuestro repositorio, la rama central (porque no hemos creado ninguna otra y por el momento no vamos a hacerlo).



```
MINGW64:/c/Users/User/OneDrive - Eidos Global/Escritorio/mi-proyecto
[master (root-commit) 1a63622] mi primer commit
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 css/style.css
create mode 100644 index.html

User@LenovoTHKP MINGW64 ~/OneDrive - Eidos Global/Escritorio/mi-proyecto (master)
$ git remote add origin https://github.com/maiEidos/mi-proyecto.git

User@LenovoTHKP MINGW64 ~/OneDrive - Eidos Global/Escritorio/mi-proyecto (master)
$ git push -u origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 289 bytes | 289.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/maiEidos/mi-proyecto.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

User@LenovoTHKP MINGW64 ~/OneDrive - Eidos Global/Escritorio/mi-proyecto (master)
$
```

Puede ser que en este momento aparezca una ventana de GitHub pidiendo que ingresemos nuestras credenciales nuevamente, en ese caso seleccionamos la opción que dice "Token" e ingresamos el token que habíamos generado en el paso 3.b.

¿Hecho? ¡Ahora a codear!

Pero... ¿qué pasa si luego de vincular mi repositorio sigo agregando archivos o escribiendo código en los que ya creé? Para eso, ir al paso siguiente.

7. Actualizar cambios

Debemos mantener nuestros repositorios local y remoto actualizados. Es una buena práctica, cada vez que terminamos de trabajar en alguna sección de nuestro código, o al finalizar el día de trabajo, subir al repositorio los cambios realizados.

Para esto utilizamos los siguientes comandos:

- a. `git add .`
- b. `git commit -m "mensaje del commit"`
- c. `git push`

Si comienzan a investigar sobre los comandos de GitHub verán que hay muchísimos comandos y funcionalidades. Por ahora nos centraremos en estos tres: add, commit y push. Son los que nos permiten compartir nuestro código para que pueda ser leído desde otra computadora y tener guardada una copia de seguridad en la nube para tenerlo disponible en caso de tener algún problema con nuestra máquina.

Si querés aprender más sobre GitHub...

Recomendamos este tutorial gratuito de Microsoft Learn, profundiza en contenidos que exceden el programa de este curso pero es útil para comprender cómo trabajar con ramas (branches):

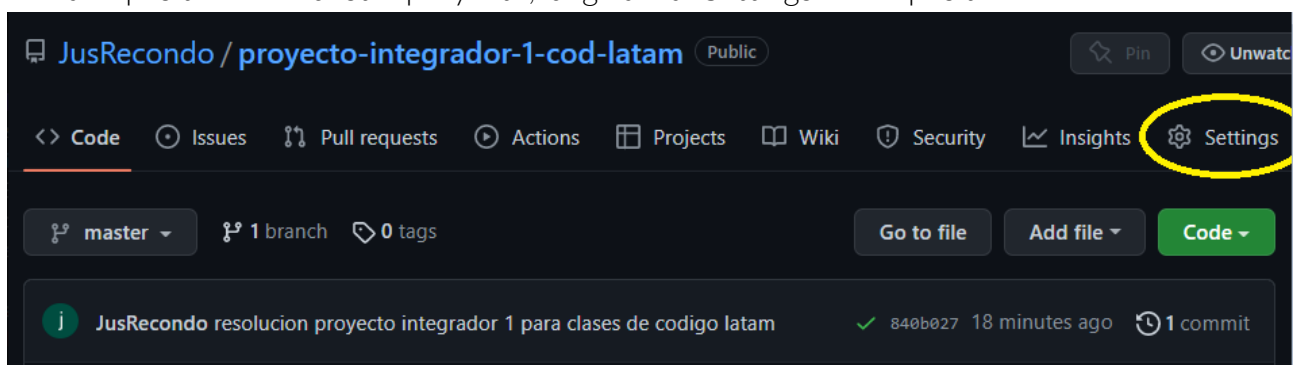
<https://docs.microsoft.com/es-es/learn/modules/introduction-to-github/>

8. Subir mi web a GitHub Pages

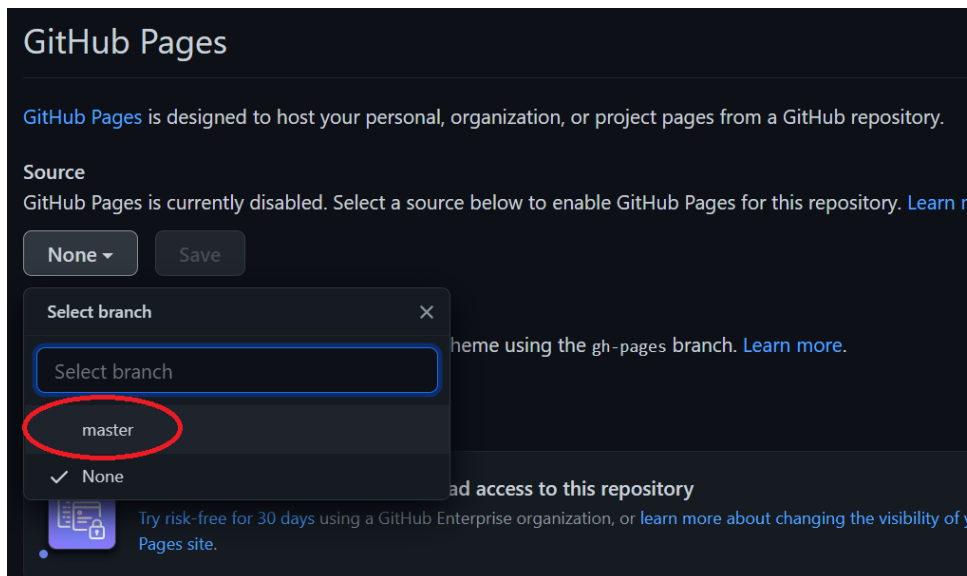
Para entregar los trabajos del curso les pedimos que suban las Webs que crearon a Git Hub Pages. En su [web](#) hay una explicación de cómo subir un proyecto propio y de cómo crear una página nueva usando las herramientas que Git Hub Pages propone.

De todas maneras, les dejamos los pasos a seguir para que no queden dudas, en nuestro caso subiremos un proyecto propio, por lo tanto los pasos a seguir son:

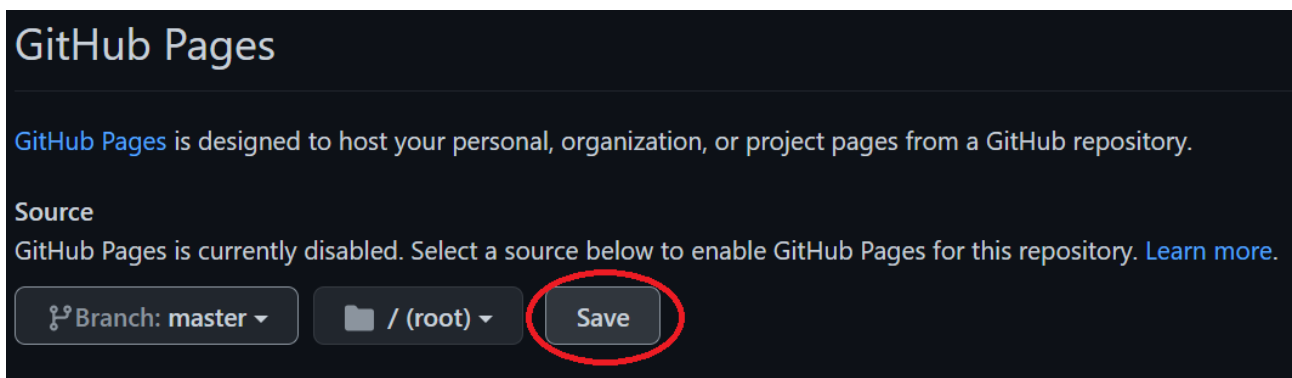
1. Ingresar a GitHub con su usuario.
2. Ir al repositorio de nuestro proyecto, luego a ir a "Settings" del repositorio.



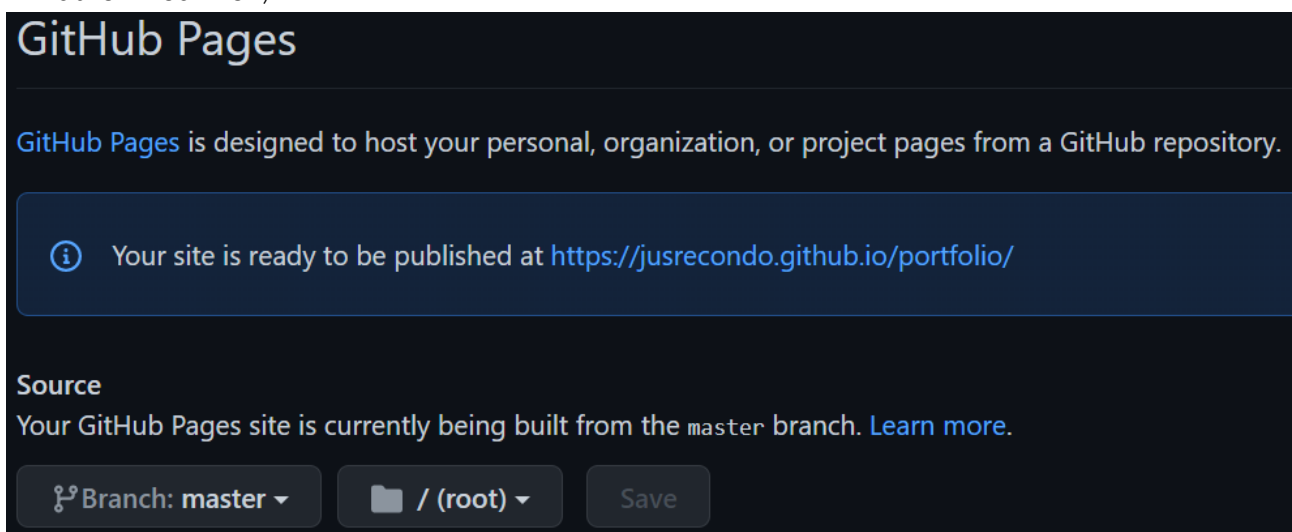
3. Ir a "Pages" en menú lateral.
4. En "Source" clicar la lista desplegable y elegir "Master"



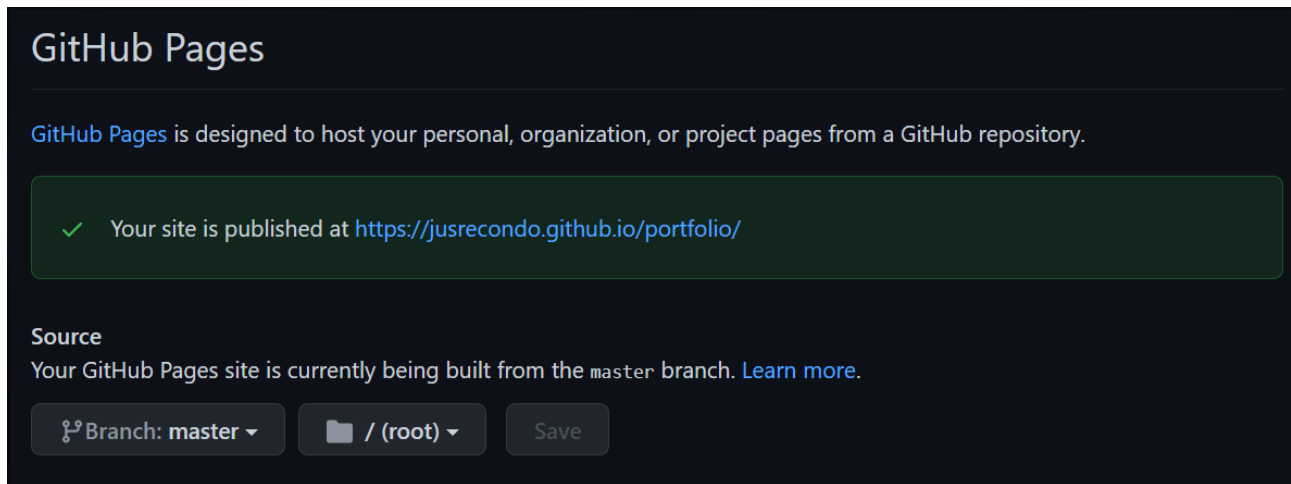
4. Clickear en "Save"



5. ¡Listo! Aparece un mensaje con el link al sitio (no va a ser inmediato, puede tardar algunos minutos en subirse).

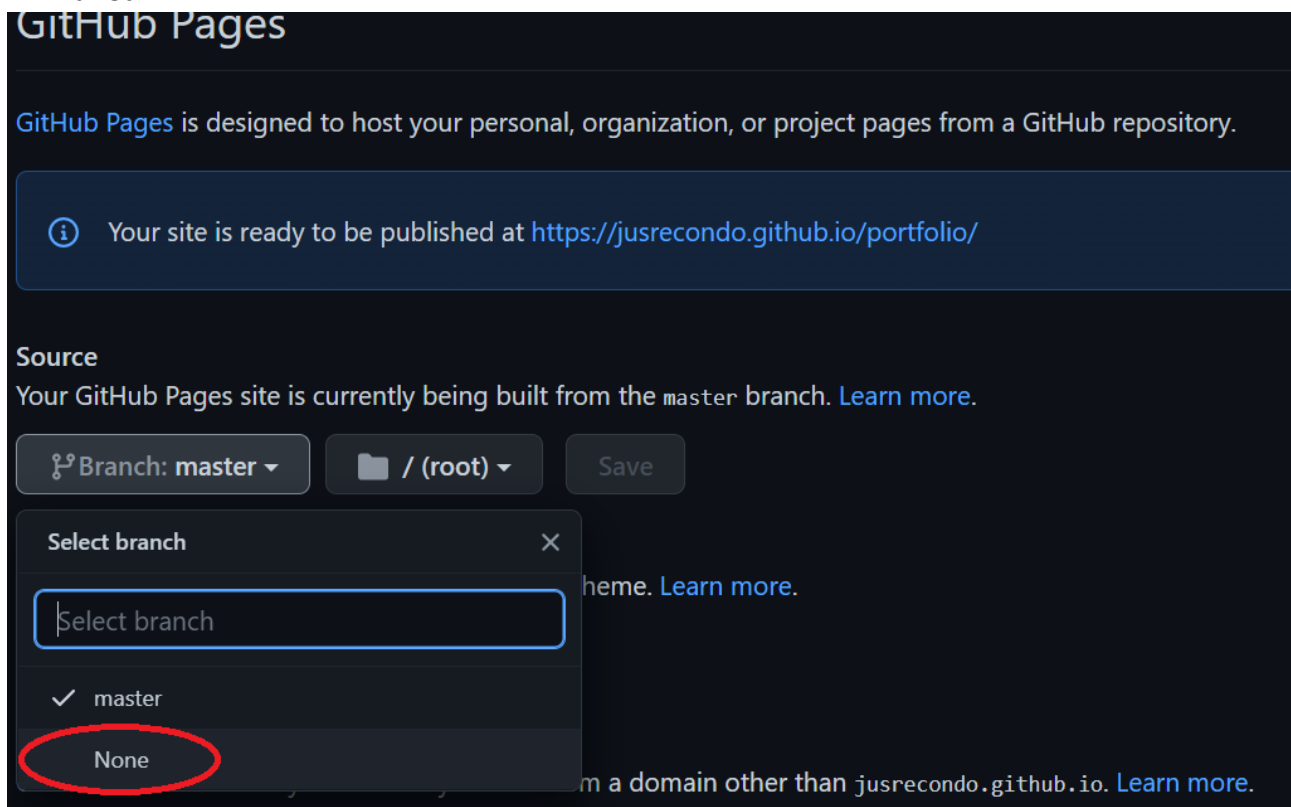


Cuando esté listo, se verá así



Bonus Track

Si luego quieren eliminar la página publicada, solo deben seleccionar "None" en "Source" y darle click a "Save".



¡Esto fue todo!

Pueden volver a leer este documento todas las veces que sea necesario hasta interiorizar el orden en que sucede cada paso y los comandos que debemos usar.

Recuerden que:

- Instalar, crear usuario y vincular sucede **una sola vez**.
- Actualizar repos remoto y local sucede **cada vez que avanzamos con nuestro código (puede ser al final del día o cuando terminamos una parte importante)**.
- Subir a **GitHub Pages es parte de la entrega de los Trabajos Prácticos** pero pueden hacerlo cuando aún no está terminado y se irá actualizando automáticamente.