github

CONFIGURAR GITHUB PAGES

Contenido

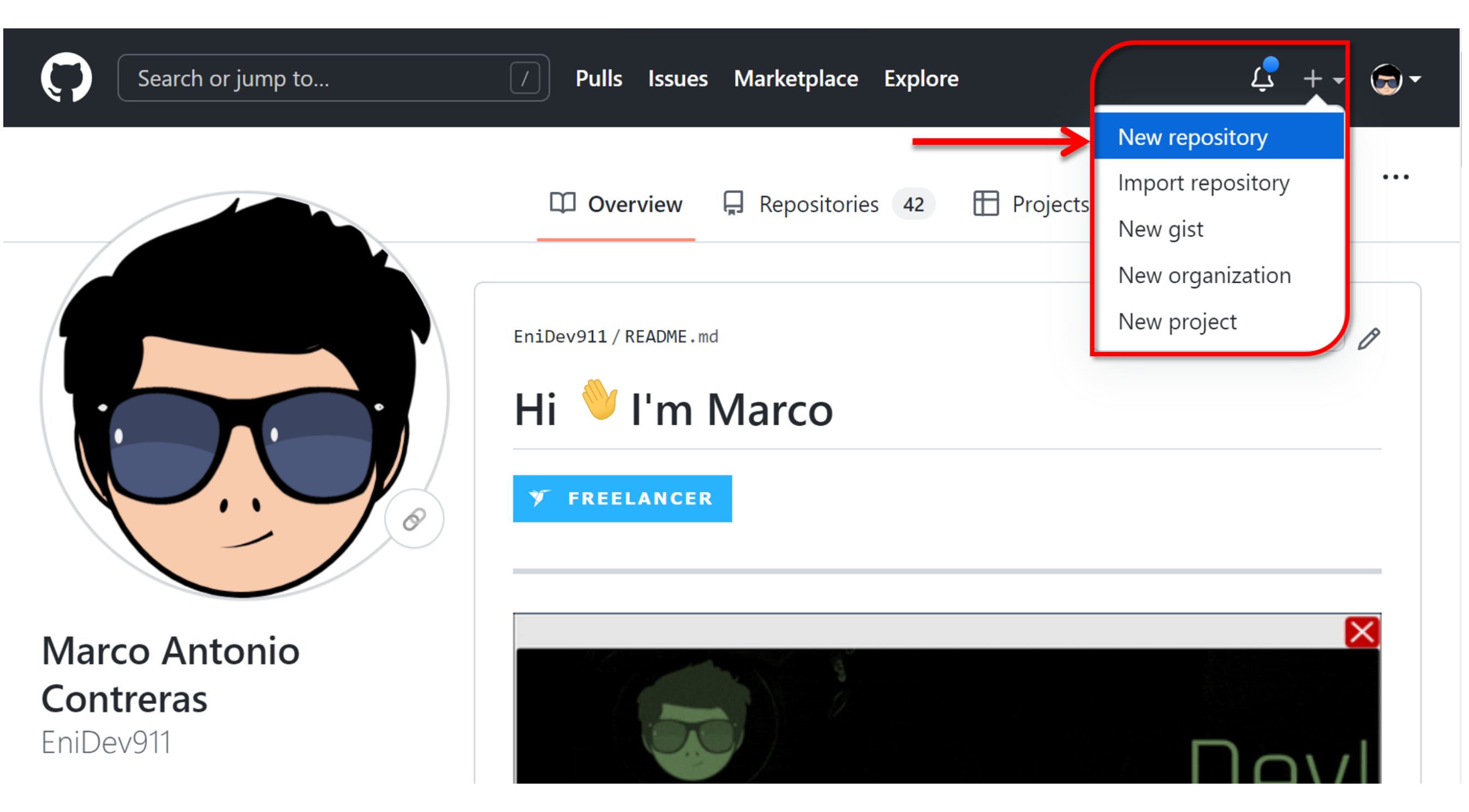
Configuración básica ⚓

1. Primero que todo, instala Git
2. Seguido, Registrate para una cuenta de GitHub
3. Una vez te hayas registrado, inicia sesión en https://github.com

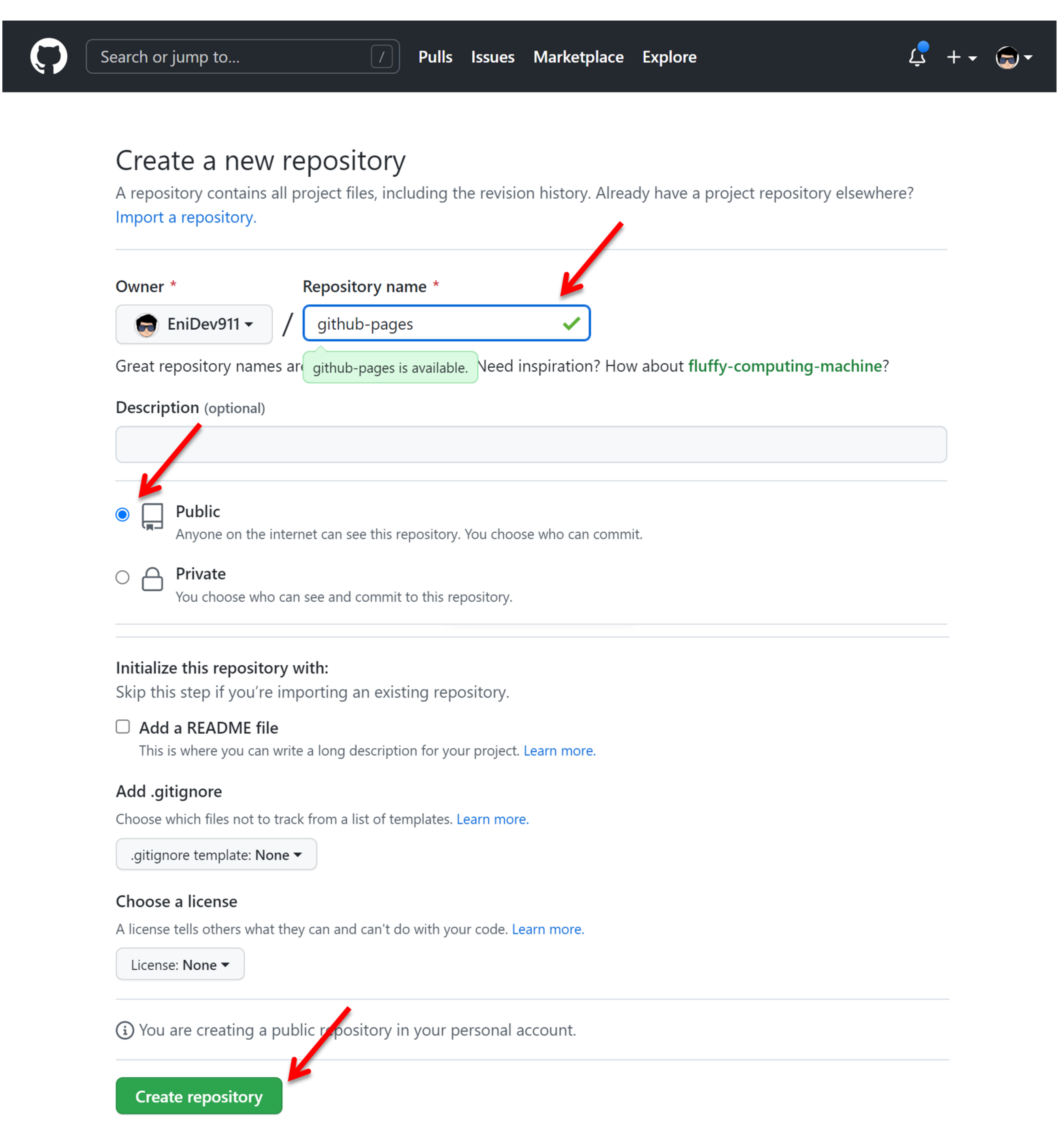
---

Crear un nuevo repositorio. ⚓

1. Iniciamos sesión en Github normalmente, luego hacemos clic en el ícono más "`(+)`" en la barra superior para abrir un submenú; ahora hacemos clic en la opción **New repository** para ir a la pantalla de **Crear nuevo repositorio**.



2. En esta pantalla, completamos los campos **Repository Name**, el campo **Description** es opcional y dejamos el repositorio como **Public** (las páginas de Github siempre son públicas, incluso cuando el repositorio que las aloja se hace privado). A continuación, hacemos clic en el botón verde **Create repository** en la parte inferior de la pantalla.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * EniDev911 / Repository name * github-pages ✓

Great repository names ar github-pages is available. Need inspiration? How about fluffy-computing-machine?

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.

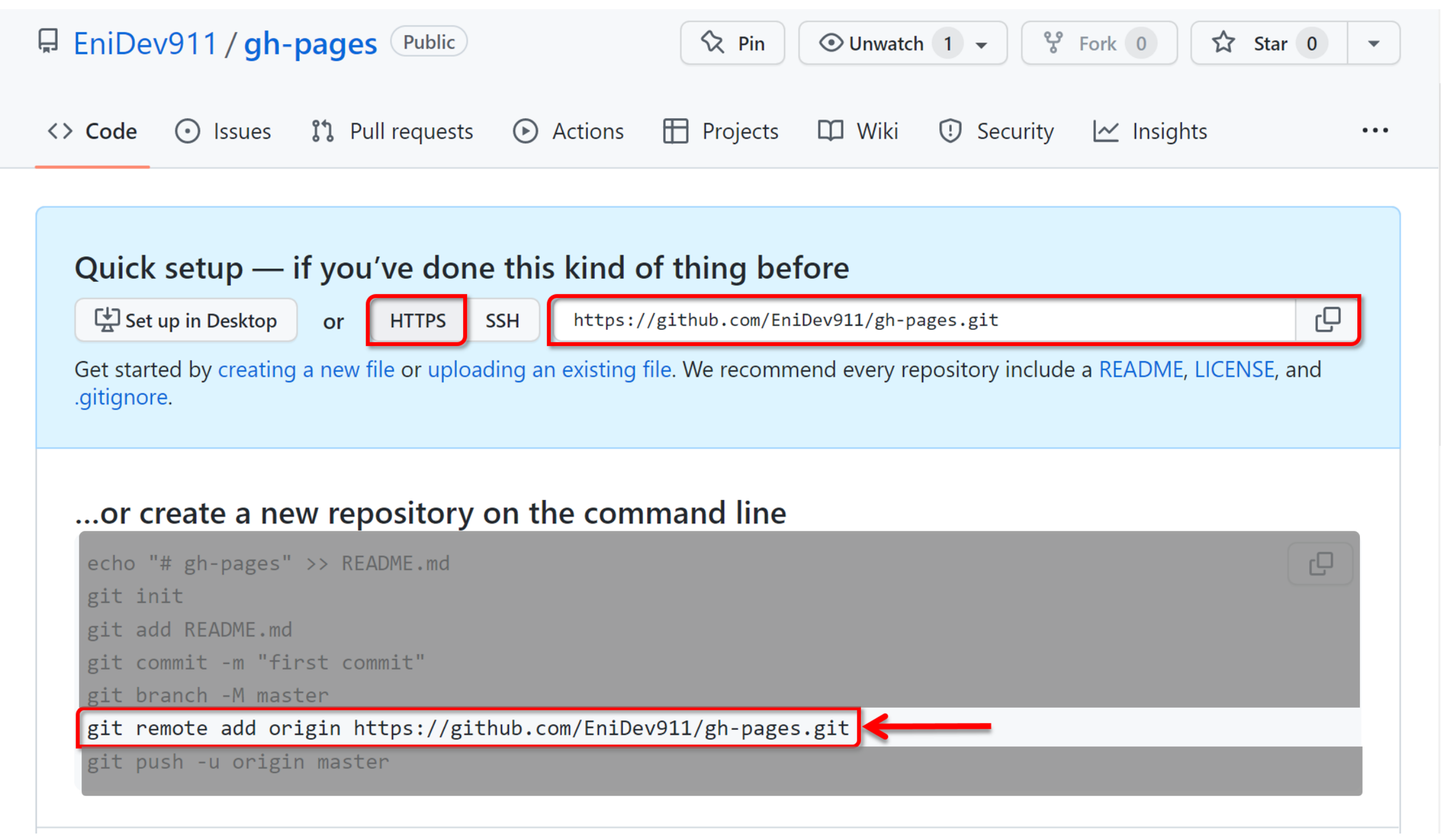
☐ Add a README file — This is where you can write a long description for your project. Learn more.

Add .gitignore — Choose which files not to track from a list of templates. Learn more.
.gitignore template: None

Choose a license — A license tells others what they can and can't do with your code. Learn more.
License: None

ⓘ You are creating a public repository in your personal account.

3. Ahora estamos en la pantalla de **configuración rápida** de los comandos que nos muestra solo utilizaremos el 6.



Quick setup — if you've done this kind of thing before

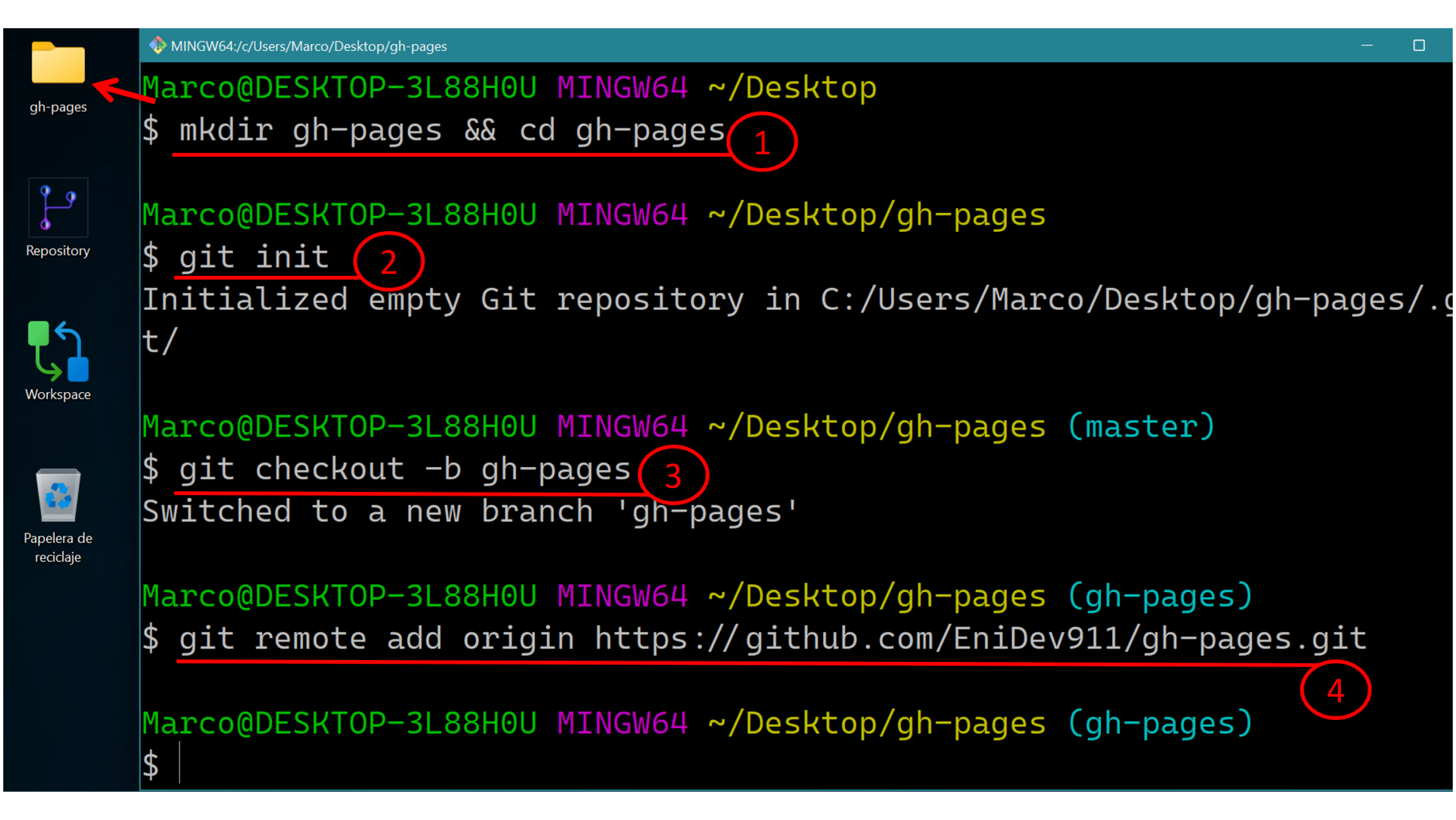
https://github.com/EniDev911/gh-pages.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# gh-pages" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M master
git remote add origin https://github.com/EniDev911/gh-pages.git
git push -u origin master
```

4. Creamos una nueva carpeta en nuestro sistema de archivos e inicializamos git y añadimos la línea de código 6 que nos muestra la página de configuración rápida `git remote add <protocol>/<host>/<username>/<repository-name.git>` para conectar nuestro repositorio local con el repositorio remoto que creamos.



```
Marco@DESKTOP-3L88H0U MINGW64 ~/Desktop
$ mkdir gh-pages && cd gh-pages

Marco@DESKTOP-3L88H0U MINGW64 ~/Desktop/gh-pages
$ git init
Initialized empty Git repository in C:/Users/Marco/Desktop/gh-pages/.g
t/

Marco@DESKTOP-3L88H0U MINGW64 ~/Desktop/gh-pages (master)
$ git checkout -b gh-pages
Switched to a new branch 'gh-pages'

Marco@DESKTOP-3L88H0U MINGW64 ~/Desktop/gh-pages (gh-pages)
$ git remote add origin https://github.com/EniDev911/gh-pages.git

Marco@DESKTOP-3L88H0U MINGW64 ~/Desktop/gh-pages (gh-pages)
$
```

Explicación:

(1): Creamos la carpeta y entramos en ella.

```
mkdir nombre_carpeta && cd nombre_carpeta
```

- **mkdir**: comando para crear carpeta (make directory).
- **cd**: comando para cambiar de carpeta (change directory).

> el '`&&`' es para garantizar que se ejecute solo si se cumple el primero.

(2): Creamos un repositorio de Git vacío.

```
git init
```

(3): Si queremos que nuestro sitio se despliegue automáticamente, debemos crear una rama con el nombre de `gh-pages`.

```
git checkout -b gh-pages
```

> Para cambiar lo hacemos con `checkout -b` así nos crea la rama y nos cambia a esa rama para trabajar en un solo comando.

(4): Agregamos el repositorio remoto que creamos.

```
git remote add origin https://github.com/EniDev911/gh-pages.git
```

`origin:` Es el nombre que le damos a la conexión remota.
`https://github.com/user/repositorio.git:` La url del repositorio remoto

> Podemos comprobar usar el comando `git remote -v` para mostrar los vínculos remotos.

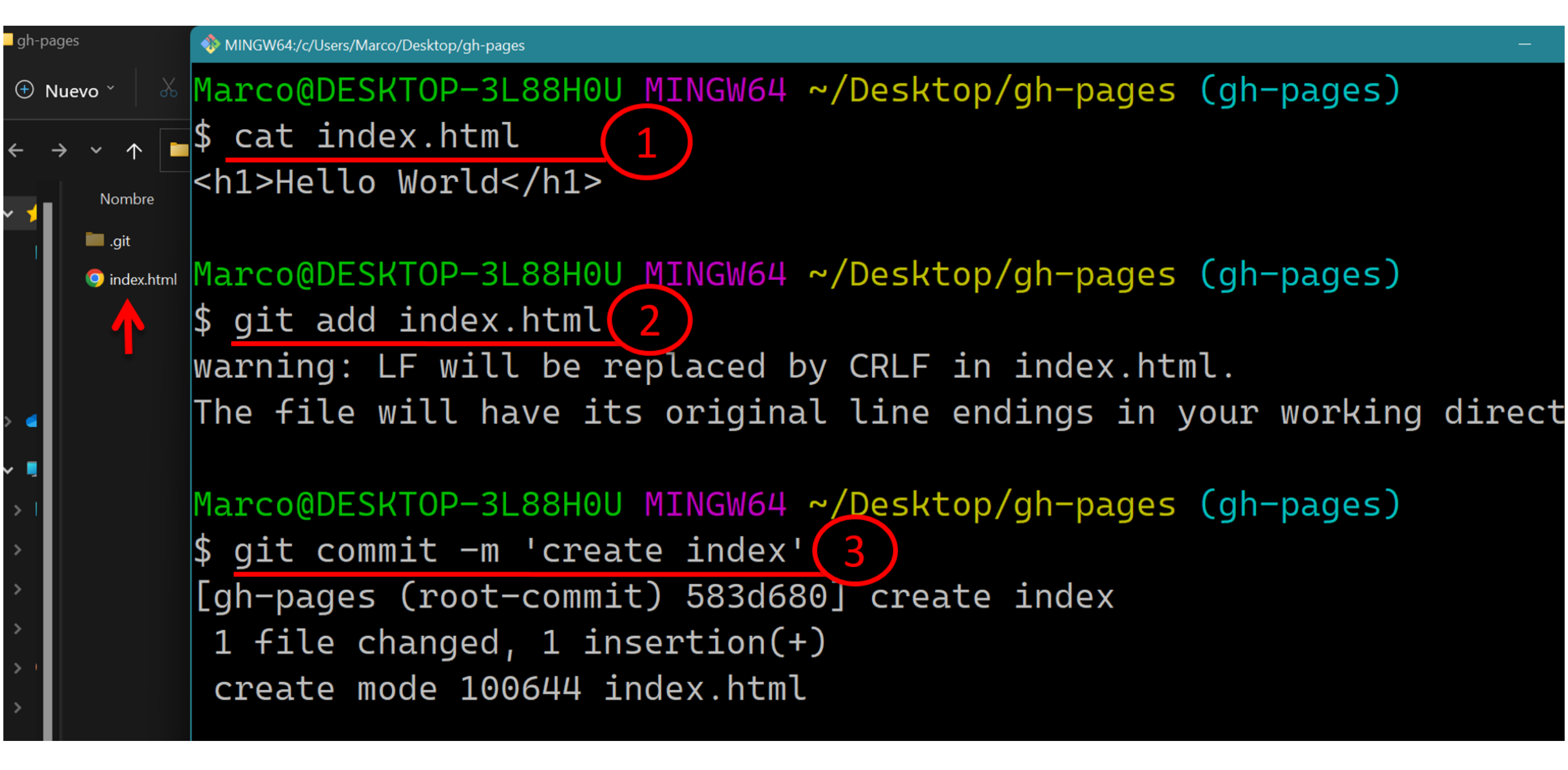
```
Marco@DESKTOP-3L88H0U MINGW64 ~/Desktop/gh-pages (gh-pages)
$ git remote -v
origin  https://github.com/EniDev911/gh-pages.git (fetch)
origin  https://github.com/EniDev911/gh-pages.git (push)
```

VOLVER AL ÍNDICE ↩

Preparando el código. ⚓

Tú puedes almacenar cualquier código que tu quieras en un repositorio de Github, pero para usar la característica de **github pages**, tu código debe estar estructurado como un sitio web típico, por ejemplo que el punto de entrada sea un archivo HTML llamado `index.html` (página de aterrizaje).

> La mejor manera de subir código a Github es mediante la línea de comandos.



```
Marco@DESKTOP-3L88H0U MINGW64 ~/Desktop/gh-pages (gh-pages)
$ cat index.html
<h1>Hello World</h1>

Marco@DESKTOP-3L88H0U MINGW64 ~/Desktop/gh-pages (gh-pages)
$ git add index.html
warning: LF will be replaced by CRLF in index.html.
The file will have its original line endings in your working direct

Marco@DESKTOP-3L88H0U MINGW64 ~/Desktop/gh-pages (gh-pages)
$ git commit -m 'create index'
[gh-pages (root-commit) 583d680] create index
 1 file changed, 1 insertion(+)
 create mode 100644 index.html
```

Explicación:

(1): Creamos un archivo `index.html` y dentro solo tenemos un `<h1>Hello World</h1>`. El cual lo podemos leer por medio de la línea de comandos con el comando `cat`.

```
cat index.html
```

> `cat`: es una herramienta que se utiliza en unix para concatenar archivos, de hecho viene por concatenar (con**cat**), pero también nos muestra los contenido de archivos.

(2): Preparamos nuestra página para la confirmación.

```
git add index.html
```

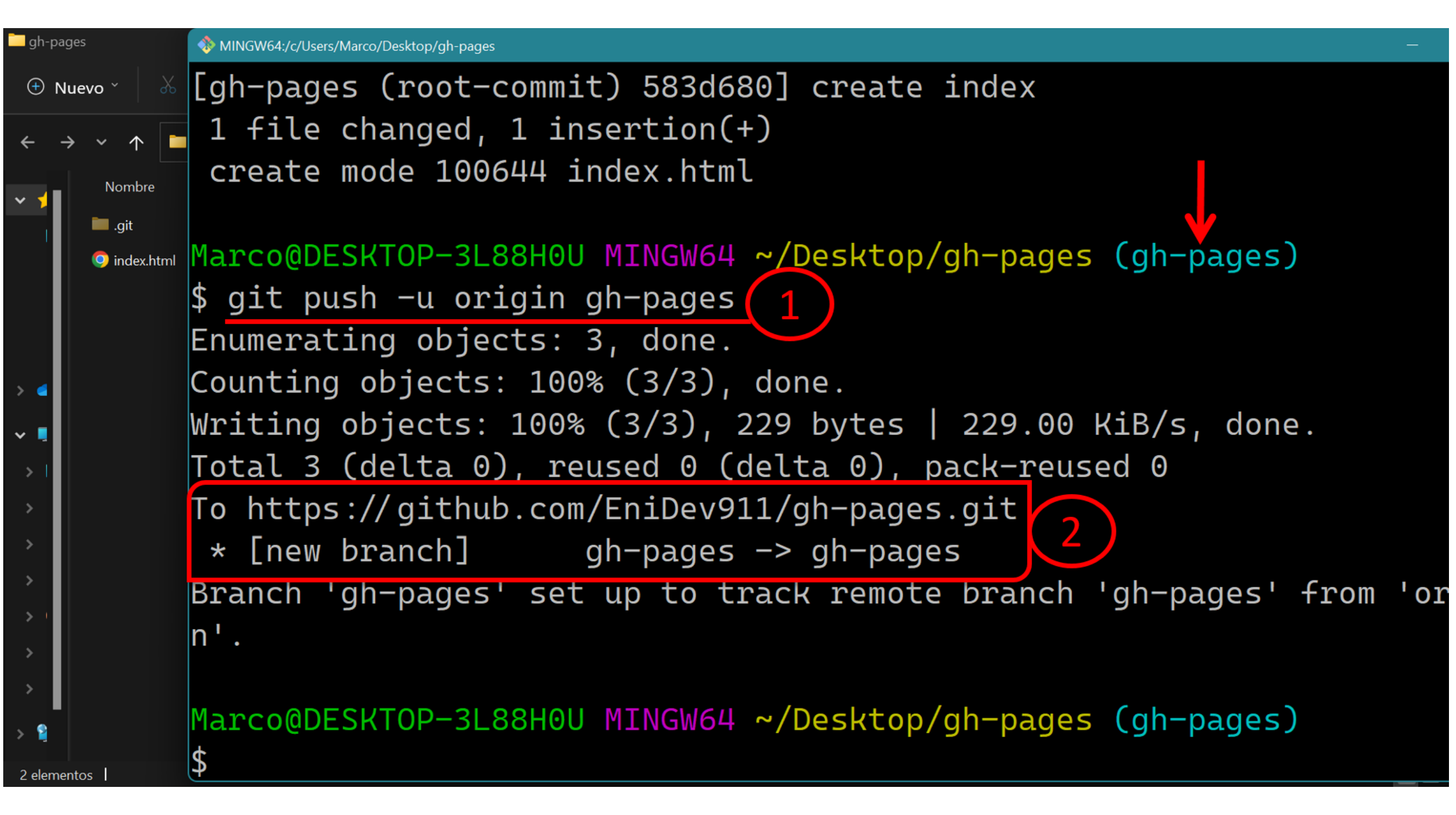
(3): Confirmamos los cambios para poder publicar.

```
git commit -m 'create index'
```

VOLVER AL ÍNDICE ↩

Publicando la página

Usemos `git push` para enviar las confirmaciones realizadas en nuestro repositorio local a nuestro repositorio remoto.



```
[gh-pages (root-commit) 583d680] create index
 1 file changed, 1 insertion(+)
 create mode 100644 index.html

Marco@DESKTOP-3L88H0U MINGW64 ~/Desktop/gh-pages (gh-pages)
$ git push -u origin gh-pages
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 229 bytes | 229.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/EniDev911/gh-pages.git
 * [new branch]      gh-pages -> gh-pages
Branch 'gh-pages' set up to track remote branch 'gh-pages' from 'or
n'.

Marco@DESKTOP-3L88H0U MINGW64 ~/Desktop/gh-pages (gh-pages)
$
```

Explicación:

(1): El comando `git push` recibe dos argumentos:

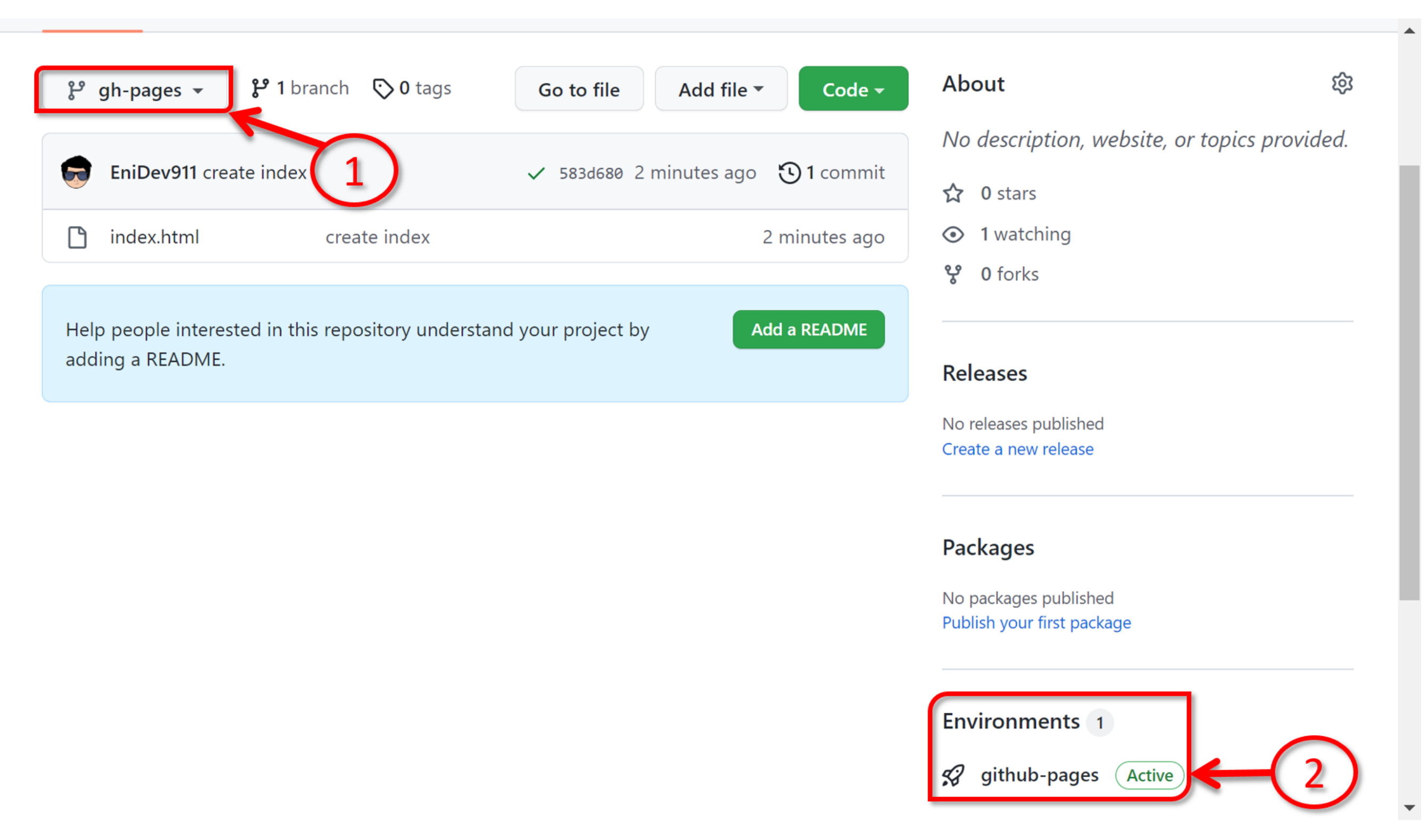
- El nombre de la conexión en la mayoría de los casos es **origin**.
- El nombre de la rama (*branch*) a la que queremos subir los cambios.

> La bandera `-u` significa *upstream* y se refiere al repositorio remoto principal al que haremos `pull` y `push`, esta opción se utiliza una sola vez, luego podemos seguir trabajando solo utilizando `git push` ya que Git recordará el nombre de la conexión y la rama en la que estamos trabajando.

```
git push -u origin gh-pages
```

> **Nota**: Es importante que nos encontremos en la rama **gh-pages** ya que de esta manera se configurará GitHub Pages automáticamente.

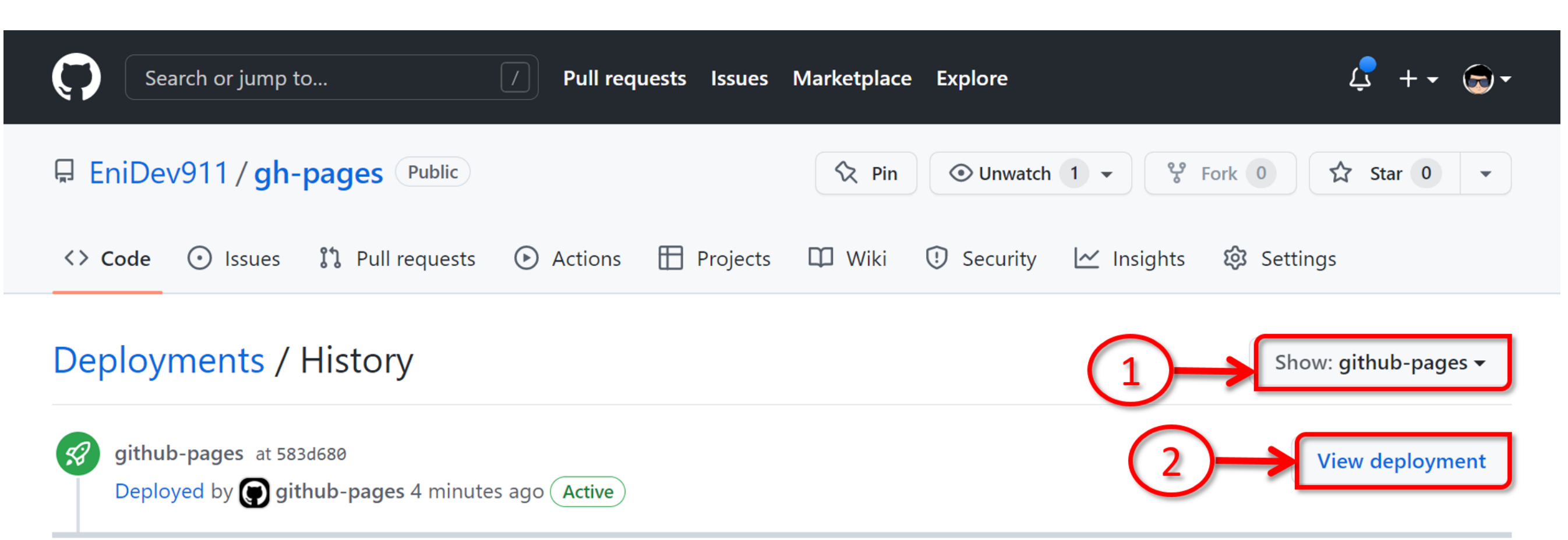
(2): El mensaje que nos muestra la URL donde se subieron los cambios y la nueva rama que se creó.



(1): Este es un control `dropdown` que tiene una lista desplegable de las ramas existentes:

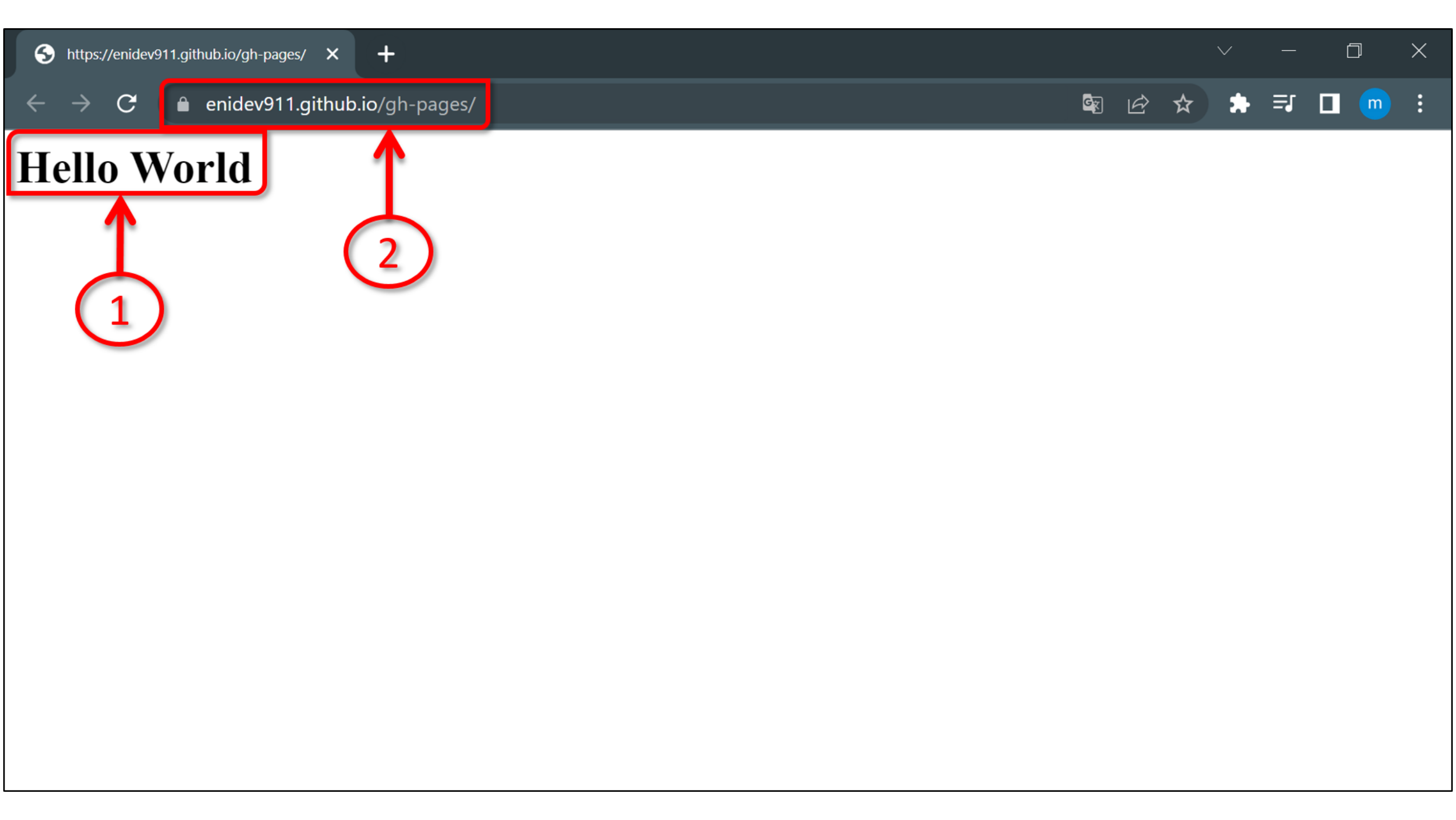
> **Nota**: Es nuestro caso como es el primer push solo tenemos la rama **gh-pages**

(2): Es el estado que describe el estado de despliegue general puede ser **pending** o **active**. Al dar clic sobre (🚀 **github-pages)** veremos el historial de todos los despliegues.



(1): Nos indica la rama a la cual se está haciendo el despliegue.

(2): Este botón nos permite visitar la página ya desplegada.



(1): Ya tenemos la página mostrando el encabezado que creamos en el archivo `index.html`.

(2): Este URL ya se puede compartir y el patrón se compone de:

```
https://<username>.github.io/<repository-name>
```

- **username**: el nombre de la cuenta de github.
- **github.io**: el servicio de hosting que nos da github.
- **repository-name**: es el nombre que tiene el repositorio.

> No es necesario que tengamos que indicar el nombre del archivo **index.html**, ya que por defecto se mostrará cualquier contenido que este dentro de un archivo llamado **index.html**.

VOLVER AL ÍNDICE ↩
