# Cómo hacer un GitHub repo

Sofia Salazar

2023-10-18

## 1. Crear un repositorio vacío en GitHub

1. Ir a la tu perfil de GitHub, dar click en el botón de `+` y luego `New Repository`, aparecerá una pantalla que dice "Create a new repository". En donde dice "Repository name" escribe el nombre de tu nuevo repositorio, por ejemplo, "Proyecto"
2. Puedes elegir poner tu repositorio en modo "Público" o "Privado", dependiendo de quien quieres que vea y pueda subir cambios a tu repositorio
3. En donde dice "Add a README file", puedes hacer *check*, los README son archivos que sirven para informar a otros desarrolladores de lo que se trata tu proyecto, es recomendado que lo añadas a todos tus repositorios.
4. Da click en el botón de crear y se habrá creado un repositorio de GitHub en tu perfil

## 2. Conecta tu repositorio de GitHub a una carpeta en tu computadora local

En este caso, vamos a conectar nuestro nuevo repositorio de GitHub que acabamos de crear (también llamado un repositorio **remoto**) a un repositorio **local** en nuestra computadora que esté vacío.

En GitHub, un repositorio remoto se refiere a un repositorio alojado en los servidores de GitHub, a diferencia de un repositorio local que existe en su computadora local. Los repositorios remotos son repositorios a los que puede acceder y colaborar a través de Internet. Son un lugar central donde múltiples usuarios o contribuyentes almacenan, comparten y administran el código de forma colaborativa.

1. Creamos una carpeta en nuestra computadora para ahí alojar el repositorio

Puedes hacerlo desde la terminal (linux) de la siguiente manera

```
cd dirreccion/en/tu/computador # lugar en mi computadora donde guardar el nuevo folder
mkdir proyecto # nombre de tu nuevo folder
cd proyecto # nos movemos a ese folder
```

2. Inicializar tu nuevo folder como un repositorio de git

Para poder designar esta nueva carpeta que creamos como un repositorio de git, escribe el siguiente comando

```
git init
```

3. Verificar la `branch` en la que esta el repositorio local

Cuando inicializamos un repositorio de git con `git init`, por default se inicia dentro de una rama o `branch`, este es como una "copia" de tu repositorio desde donde tu puedes hacer cambios. Cuando creamos un repositorio en la página de github (nuestro repositorio remoto), también se inicia en una `branch` por default.

Ve a tu repositorio en github.com y en la parte superior estará el nombre de la `branch` en la que está tu repositorio remoto, en mi caso es `main`, pero en ocasiones se puede llamar también `master`

Ahora, verifiquemos en que `branch` está tu repositorio local

```
git status
```

En mi caso, aparece `on branch master`, yo quiero que esté en el mismo `branch` que mi repositorio remoto, en `main`, de esta forma puedo hacer cambios directamente desde esa rama.

Escribe el siguiente comando para crear y cambiarse a la branch `main`

```
git branch -M main
```

Escribe el siguiente comando para vincular tu carpeta al repositorio remoto, sustituye el link por el que obtienes desde el botón verde `Code` en el reposorio en github.com

```
git remote add origin https://github.com/tu/link
```

Escribe el siguiente comando para hacer `pull` y jalar los cambios del repositorio remoto, esto es importante si creamos un archivo README.

```
git pull origin main
```

*Nota: las siguientes veces que hagas* ***`pull`*** *solo será necesario escribir* ***`git pull`***

**Ahora puedes escribir archivos en este folder desde tu computadora**

## 3. Subir tus cambios a GitHub

Si ya trabajaste en algunos archivos y quieres subirlos a GitHub, sigue los siguientes pasos:

1. Añade los archivos que vas a subir utilizando `add`

Para añadir todos los archivos que modificaste o creaste, escribe:

```
git add .
```

Para añadir solo un archivo por su nombre:

```
git add nombre_del_archivo
```

Para añadir archivos que tengan una "frase" en su nombre:

```
git add *.png # archivos que terminen en .png
git add script* # archivos que empiecen con la palabra script
git add *2023* # archivos que en alguna parte de su nombre tengan el numero 2023
```

2. Haz `commit`

Hacer `commit` es como asignamos una "marca de tiempo" a nuestro trabajo, cada commit en GitHub se guarda con un identificador único que son una serie de números, podemos usar este identificador para "volver en el tiempo" y ver nuestros archivos en ese momento.

Para hacer `commit` de los archivos que ya hicimos `add`, usamos el siguiente comando, escribe un mensaje informativo acerca de lo que identifique a tu `commit`:

```
git commit -m "mensaje"
```

3. Haz `push`

Finalmente, para subir los cambios a GitHub, hacemos `push`

La opción `--set-upstream origin main` es solo necesaria la primera vez que haces `push`

```
git push --set-upstream origin main
```

En siguientes ocasiones, solo escribe

```
git push
```

¡Listo! tus cambios ahora están en GitHub, lo puedes verificar desde el repositorio en github.com