Introducción Git

| **Summary** | En este codelab aprenderás sobre los conceptos de git, Github, como utilizarlos y sus diferencias, a través de ejercicios prácticos. |
| --- | --- |
| **URL** | cp1 |
| **Category** | Web |
| **Environment** | web |
| **Status** | Published |
| **Feedback Link** |  |
| **Author** | Marlene Aguilar |

[Bienvenido](#_wsuihf2e4v1t)

[SISTEMA CONTROL DE VERSIONES](#_nkafiked5rrz)

[GIT](#_97yk41dnnnj6)

[GitHub](#_iq80rqw367rt)

[Iniciado en GitHub](#_27mgbjpadlxk)

[Instalando Git](#_5fjedrpecaah)

[Iniciando en Git](#_ntqw6wck1xgr)

[Estados en Git](#_xb5cp5n96zkk)

[El flujo de trabajo básico de Git](#_bw12sikg200i)

[Comandos](#_7gsc7it32aub)

[Commit](#_8ejabf358eok)

[Ejemplo práctico](#_75rnjunu8k7x)

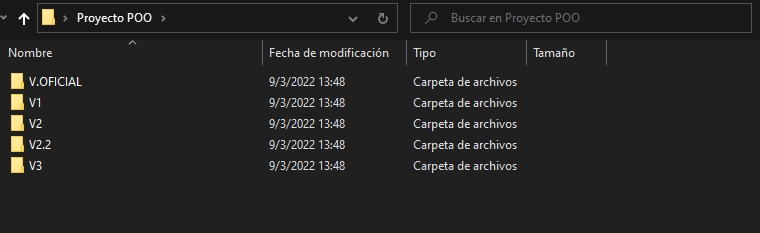
[Ejercicio evaluado](#_j83cbau04aqs)

# Bienvenido

En esta guía aprenderás sobre qué es Git y Github, sus diferencias y cómo utilizarlo.



# SISTEMA CONTROL DE VERSIONES



Las aplicaciones de control de versiones nos permite administrar las diferentes versiones de uno o varios archivos y en cualquier momento se puede acceder a cualquier versión del pasado y verificar cuales fueron los cambios, quien la realizó, a que horas y fecha se realiza esa modificación.

Existen muchos Sistemas de Control de Versiones siendo algunos de los más conocidos CVS, Git, Subversion, Mercurial y Bazaar.





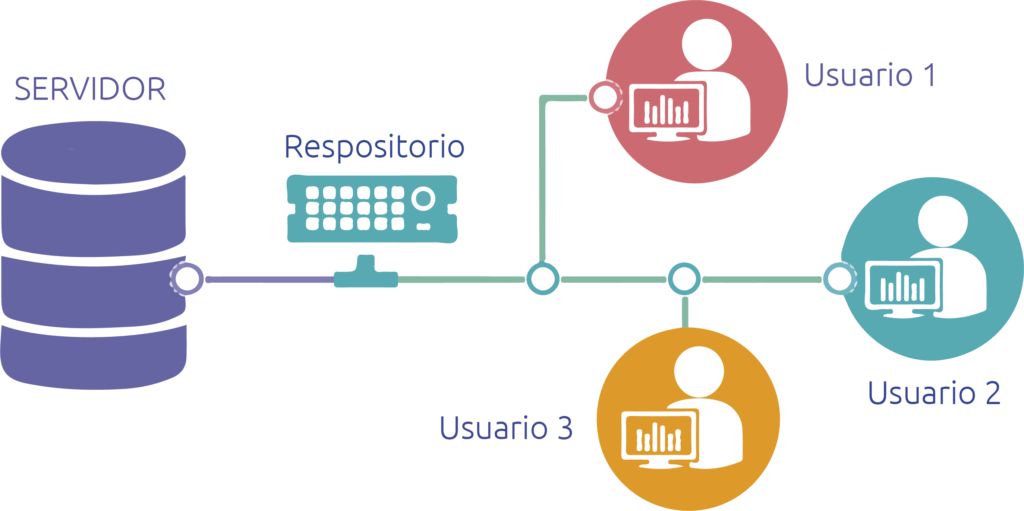
Leer más: [Sistemas de Control de Versiones, qué son y por qué amarlos. | by jointDeveloper | Medium](https://medium.com/@jointdeveloper/sistemas-de-control-de-versiones-qu%C3%A9-son-y-por-qu%C3%A9-amarlos-24b6957e716e)

# 

# GIT

Es un sistema de control de versiones distribuido (VCS), que permite almacenar diferentes versiones de uno o varios archivos para que cualquier versión sea recuperable cuando lo desee.

**Nota:** Git en realidad es un sistema de control de versiones *distribuido*, lo cual significa que se realiza una copia completa del repositorio que contiene la base de código en tu computadora (y en la de todos los demás). Realizas cambios en tu propia copia, y luego empujas esos cambios nuevamente al servidor, donde un administrador decidirá si fusiona tus cambios con la copia maestra. - MDN Mozilla



Leer más: <https://docs.github.com/es/get-started/using-git/about-git>

# GitHub

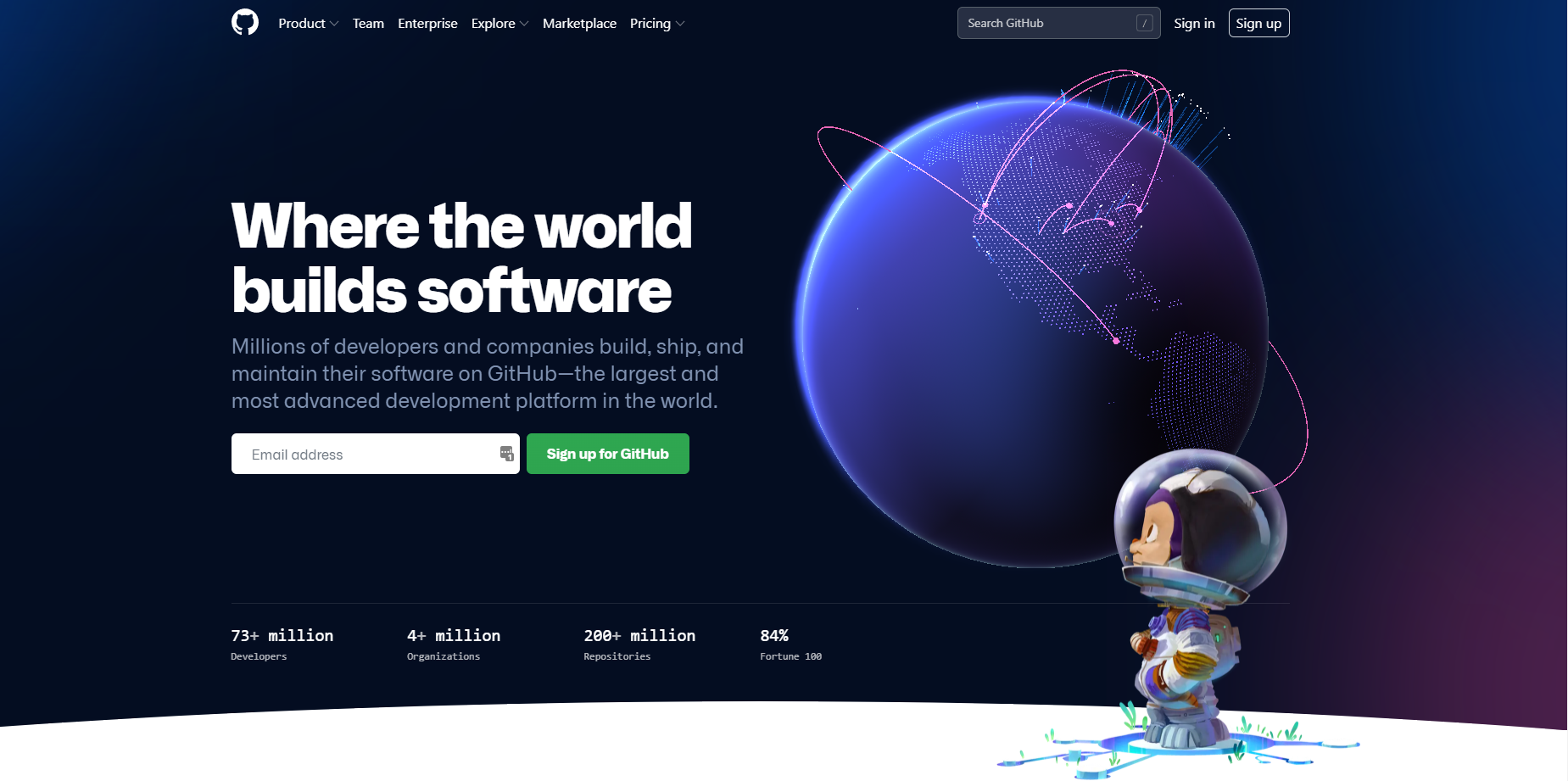
Es una plataforma web que permite alojar proyectos, utilizando el sistema de control de versiones distribuidos de GIT, su interfaz gráfica lo ha convertido en uno de los clientes Git más utilizados por los desarrolladores.



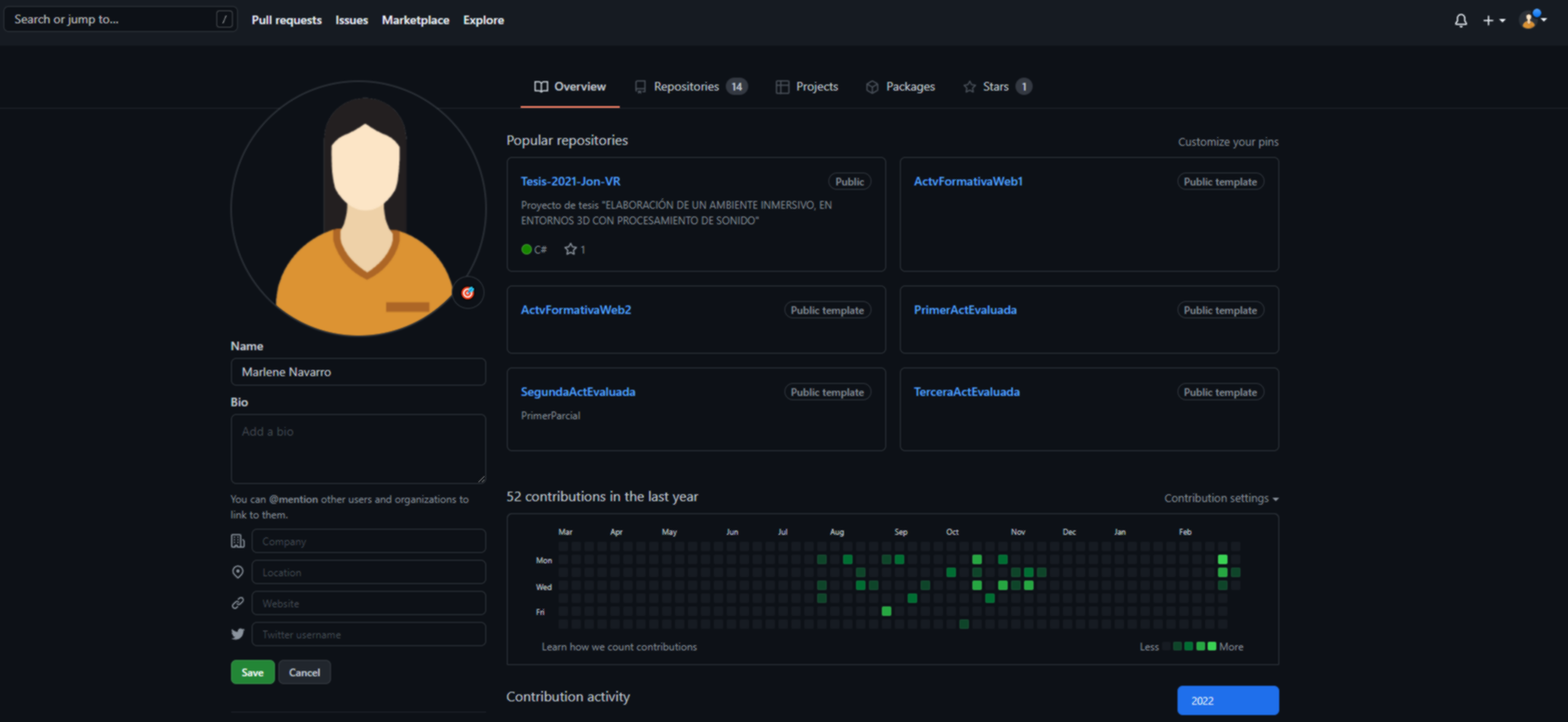
Enlace a sitio oficial: [www.github.com](https://github.com/)

# Iniciado en GitHub

Acceder al sitio oficial: [www.github.com](https://github.com/) , crear una cuenta de usuario, de preferencia utiliza tu correo electrónico institucional.



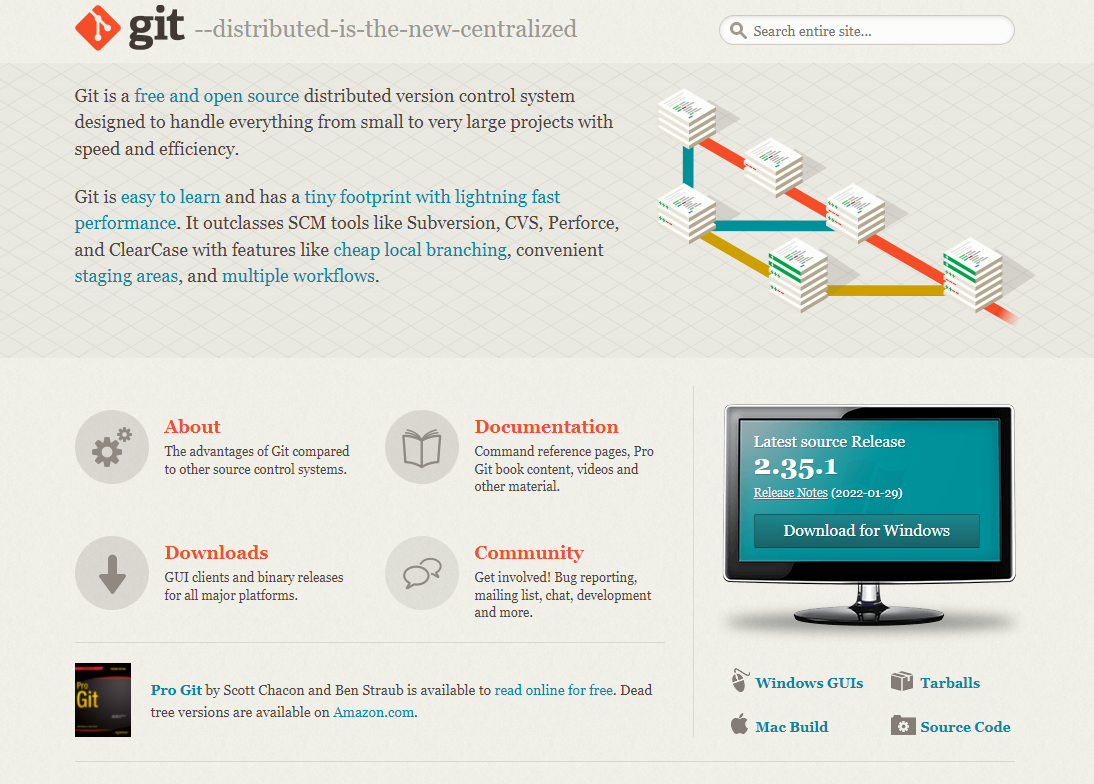
Una vez creada la cuenta, completa tu perfil coloca tu nombre y foto.



# Instalando Git

Antes de empezar a utilizar Git, debemos instalarlo en nuestra computadora. Si ya está instalado, puedes actualizarlo a su última versión.

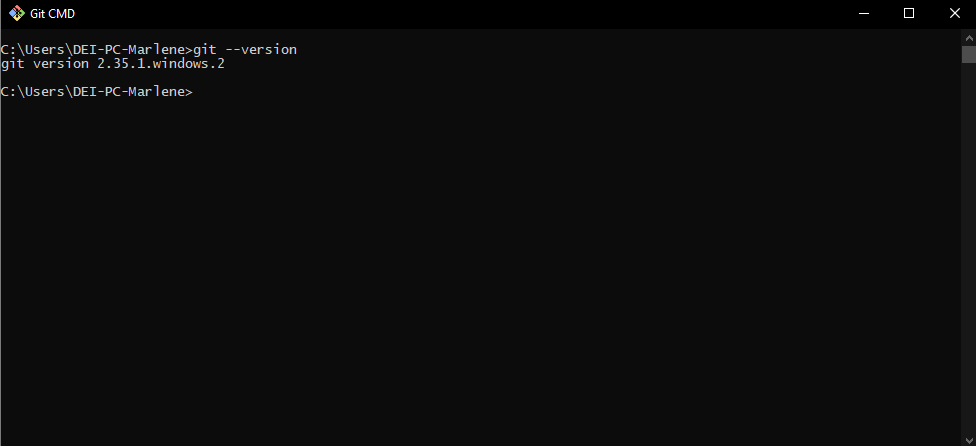
Enlace a sitio oficial Git: [Git](https://git-scm.com/)



**Verificar instalación**

Para verificar si la instalación se realizó de manera exitosa. Ejecutar el siguiente comando:

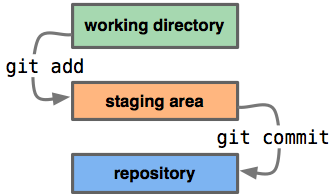
git --version



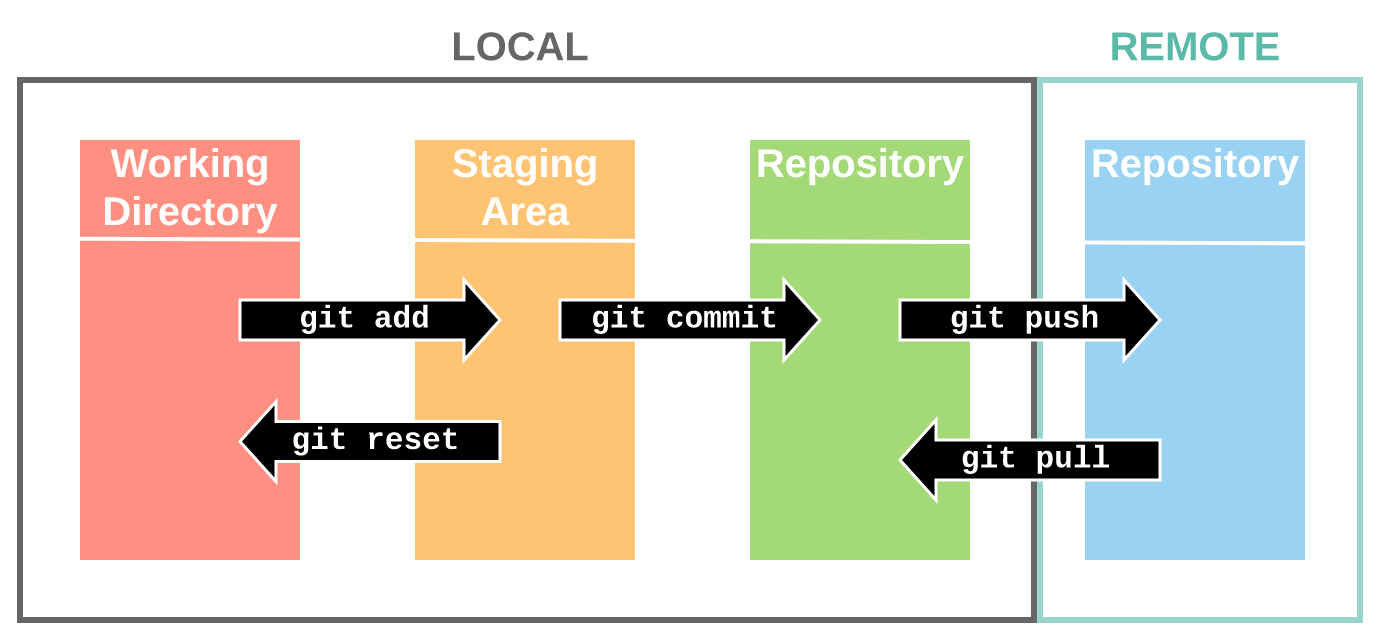
# Iniciando en Git

## Estados en Git

* Confirmado (committed): la información está almacenada en la base de datos local.
* Modificado (modified): un archivo ha sido modificado, pero se han confirmado a la base de datos.
* Preparado (staged): se ha marcado un archivo modificado en su versión actual para enviarlo a la próxima confirmación.

****

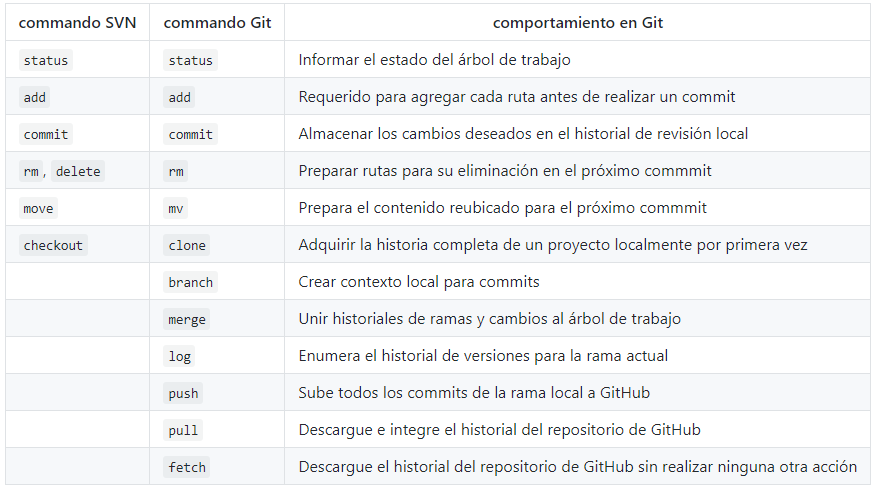
## El flujo de trabajo básico de Git

****

Leer más: [Estados Git](https://moisesdelacruz.medium.com/tutorial-b%C3%A1sico-de-git-y-github-42e46ff41194)

Ver video**:** [Git & GitHub: Git Status](https://www.youtube.com/watch?v=SxmveNrZb5k)

## Comandos



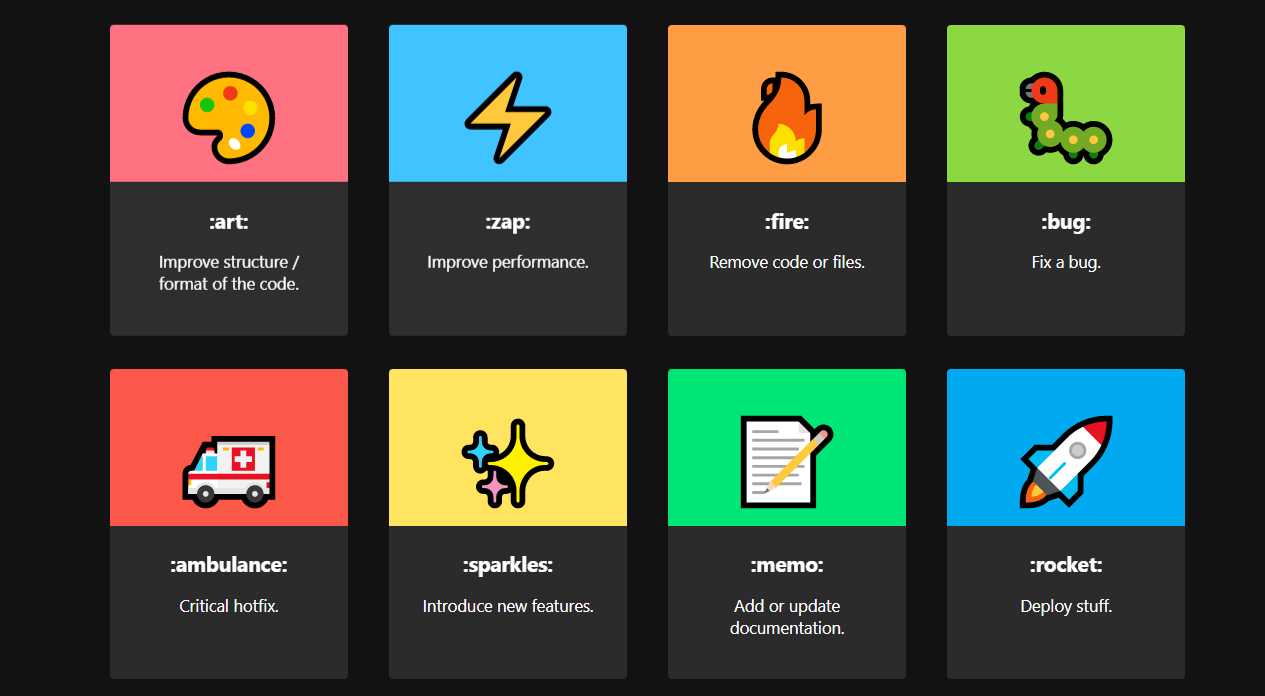
Leer más: [comandos-git.pdf](https://drive.google.com/file/d/1sAYpx-N9RcNDpeurbF8m7gggJsdgt_pX/view?usp=sharing)

## Commit

Se usa para confirmar los cambios en el área de preparación y guardarlos en el repositorio local en Git.

Algunas buenas prácticas:

1. Especifica el tipo de commit:  
   feat: La nueva característica que agregas a una aplicación en particular  
   fix: Un parche para un error  
   style: Características o actualizaciones relacionadas con estilos  
   refactor: Refactorizar una sección específica de la base de código  
   test: Todo lo relacionado con pruebas  
   docs: Todo lo relacionado con documentación  
   chore: Mantenimiento de código regular.
2. Quita signos de puntuación innecesarios.
3. Usa mayúsculas al inicio del título y por cada párrafo del cuerpo del mensaje.
4. Usa el cuerpo del mensaje para explicar los cambios.
5. Sigue la convención del mensaje de commit definida por tu equipo.
6. Agrega emojis a tus commit, debes buscar el emoji que describa lo realizado.

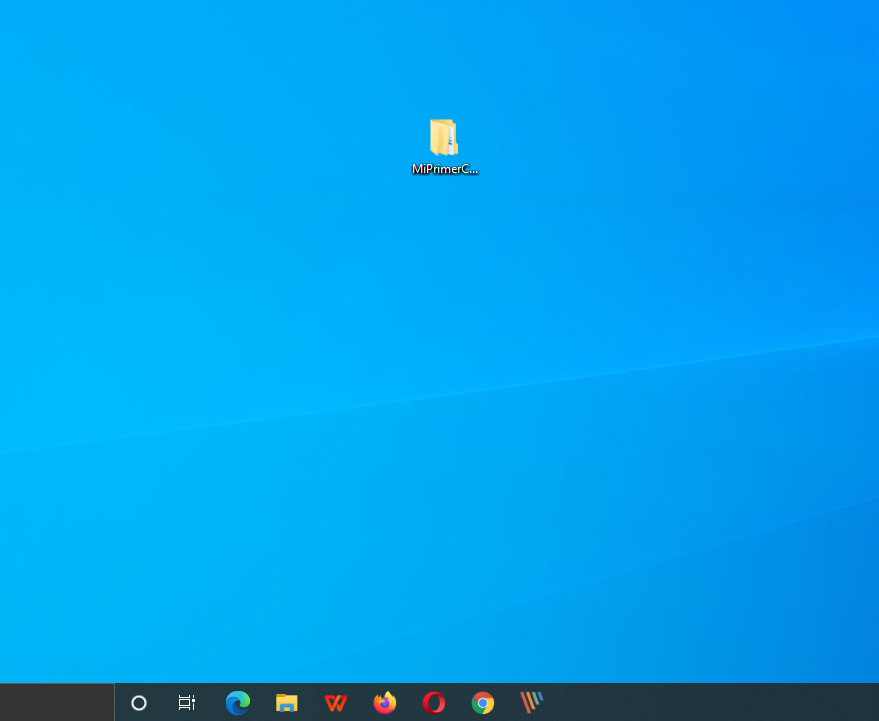


Enlace: [www.gitmoji.dev](https://gitmoji.dev/)

# Ejemplo práctico

**Clona un repositorio y sube los cambios**

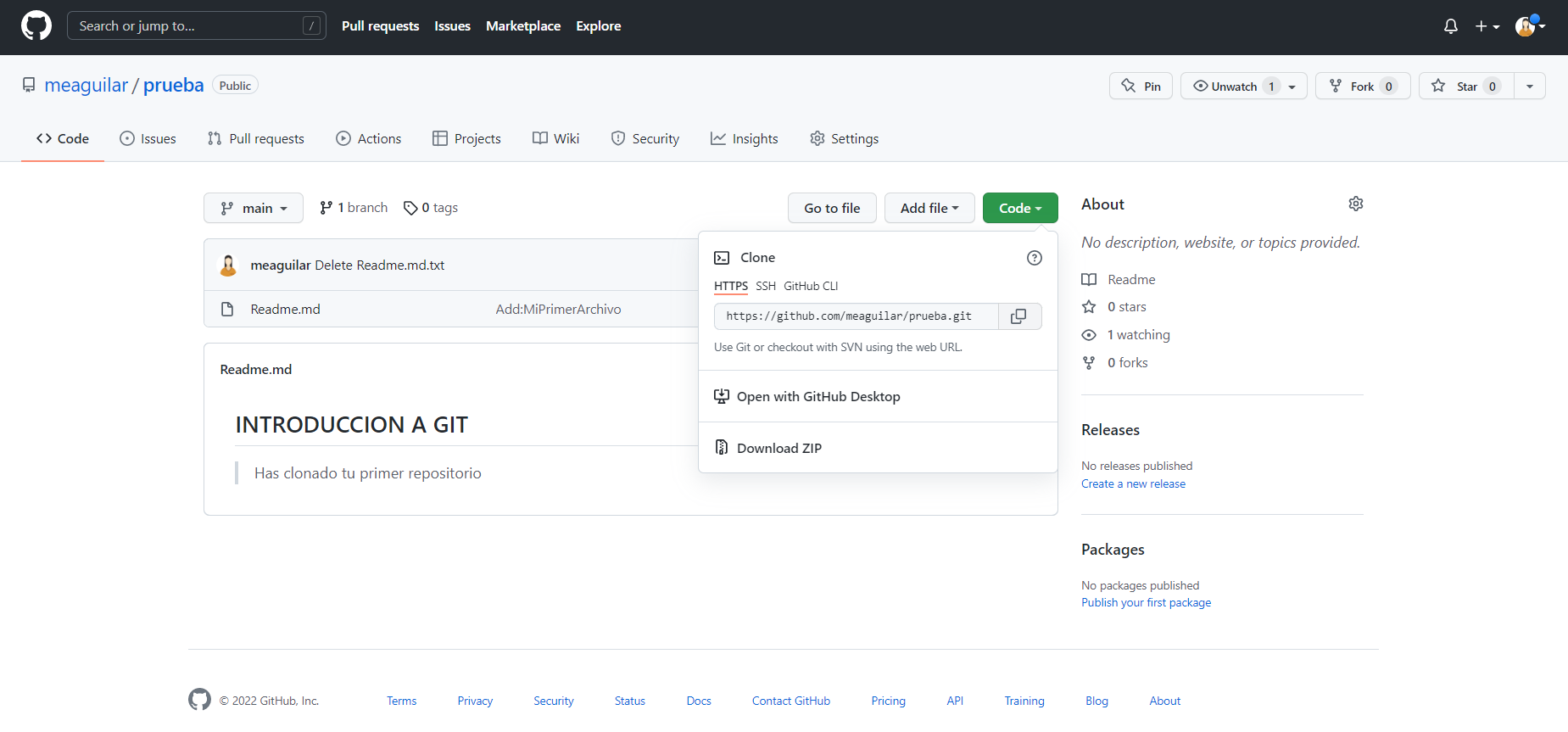
Crear un directorio en alguna ubicación de tu computadora, en este caso se creó una en el escritorio llamada MiPrimerClonacion



Abre una consola en la ubicación de la carpeta llamada MiPrimerClonacion

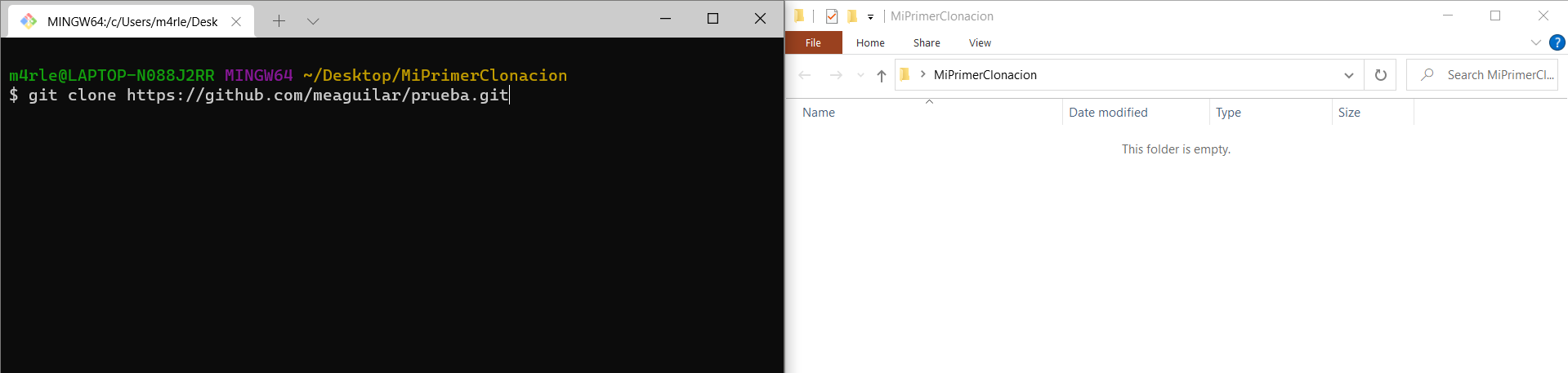
**Clonando**

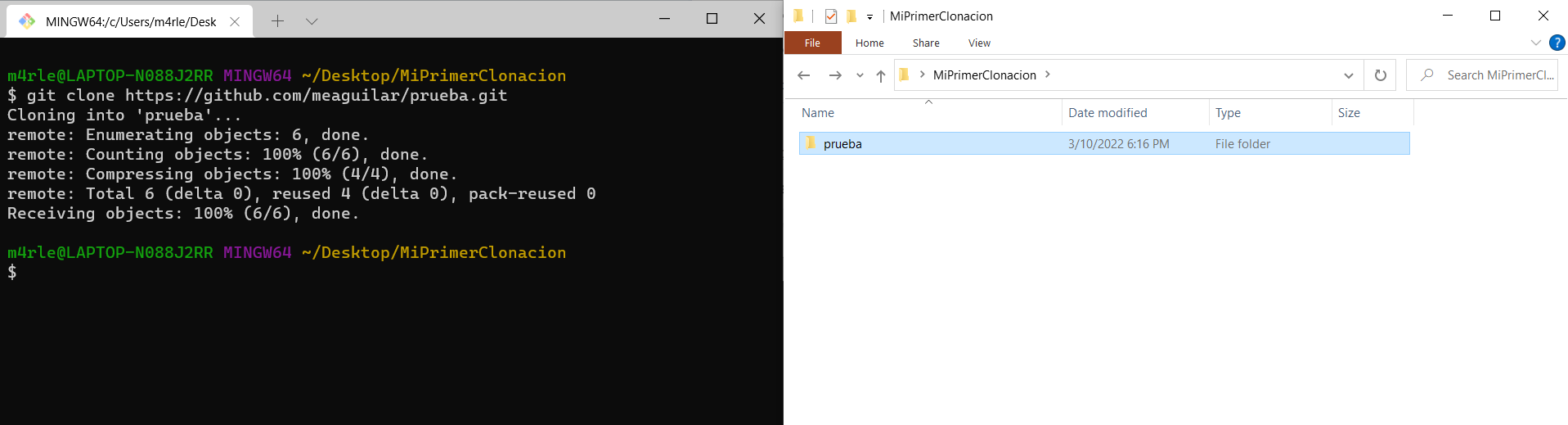
Copia la url del repositorio

****

* git clone [url]

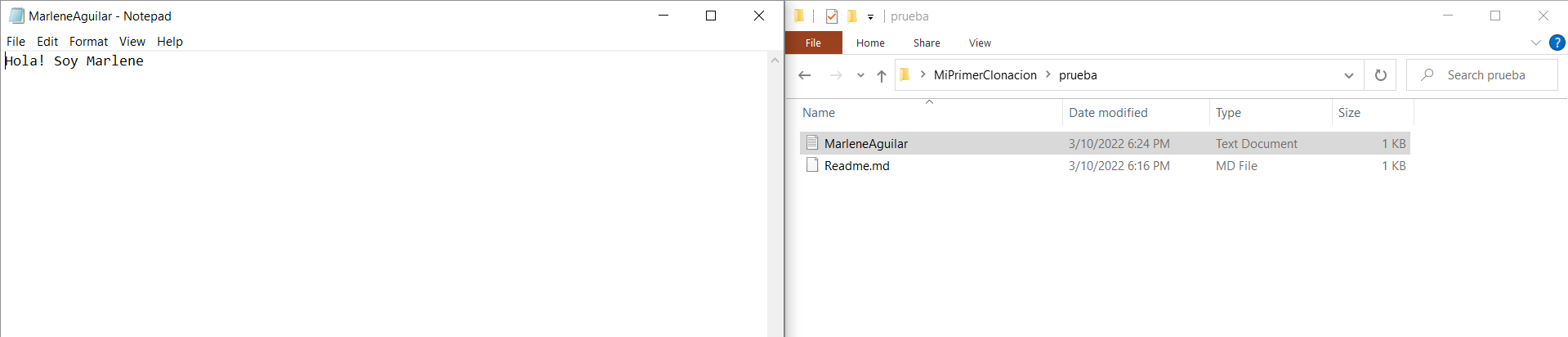
En este paso se copia la dirección utilizando el comando y clonara el proyecto del repositorio a tu carpeta.





**Crea un archivo txt con tu nombre.**

En el contenido del archivo también coloca tu nombre.

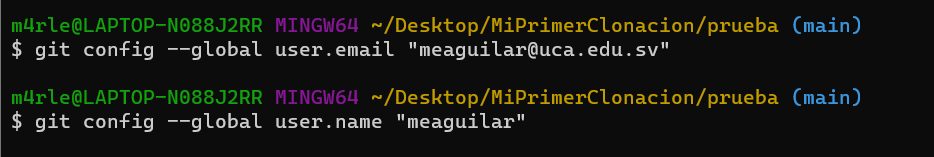


**Iniciando a subir los tus archivos**

**Configuración**

Al ser primera que se realizará un cambio en el repositorio, solicitara los siguientes datos:

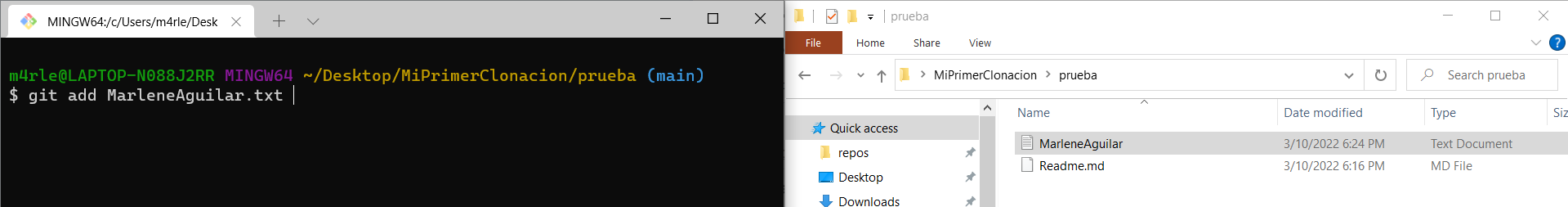
* git config --global user.name “[nombre apellido]”
* git config --global user.email “[email]”



**Agregando archivos**

Agregamos el nombre de la carpeta del proyecto es a la cual le realizamos las modificaciones

* git add

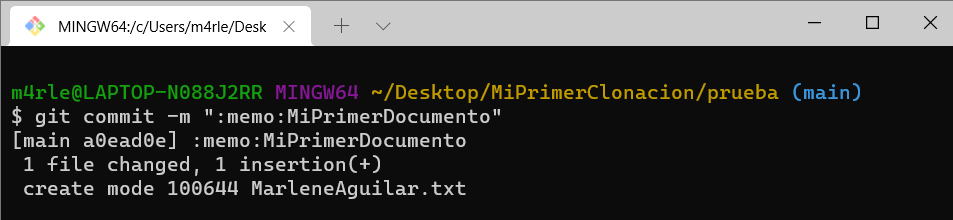


**Commit**

Agregamos el commit con un mensaje descriptivo de lo que se ha realizado, toma en cuenta las buenas prácticas, puedes usar <https://gitmoji.dev/> pero revisa cual es el más descriptivo para lo que has realizado a tu proyecto.

**IMPORTANTE: Puedes hacer varios commit, por cada modificación considerable puede realizar un commit, esto permitirá tener un registro de todo lo que vas realizando en el proyecto.**

* git commit

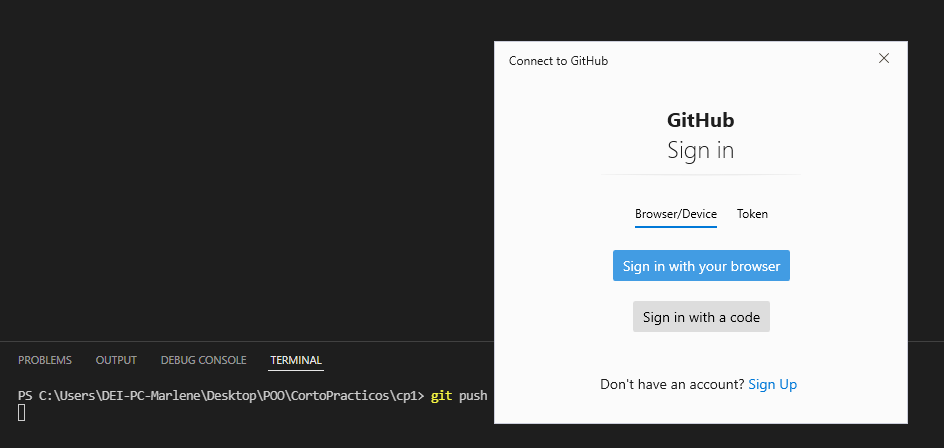


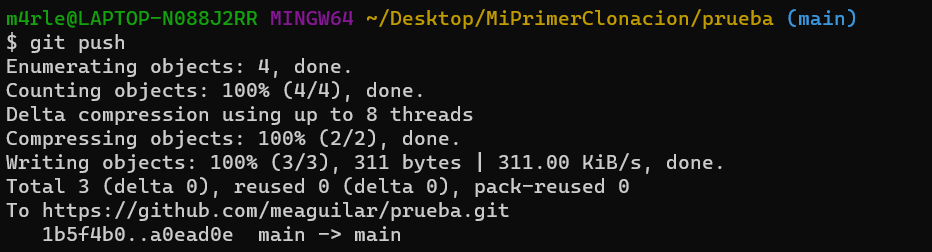
**Push**

Envía tus cambios al repositorio.

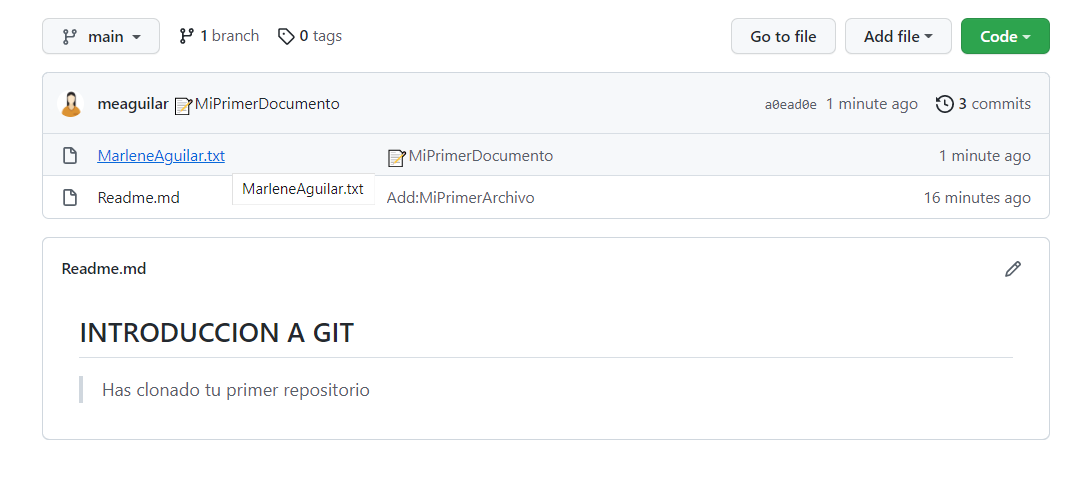
* git push

En este paso si solicitara autenticación, realizarla con el navegador.





Verificar su repositorio en GitHub, mostrará los nuevos archivos que se han agregado. **FELICIDADES!!!**



# Ejercicio evaluado

1. Crea una cuenta en github con tu cuenta institucional
2. Accede a la tarea que está en Moodle llamada cp1
3. Busca tu nombre y acepta la tarea
4. Clona el repositorio
5. Agrega un archivo de texto
6. Crea una rama en el repositorio clonado y sube los cambios a esa rama
7. Sube los cambios al repositorio utiliza los comandos en el orden correcto (git add, git commit, git push), mínimo de commit 3
8. Realiza la fusión de la rama con el main
9. Puedes verificar en tu repositorio de GitHub los cambios que realizaste al código.
10. Borrar las credenciales de windows
11. Borrar de la computadora el o los programas creados durante la práctica

