

# Curso de Git interno parte 2

---

ENRIQUE CALDERÓN, HANSEL TEPAL Y LUIS UGARTECHEA  
Estudiante de Ingeniería en Computación

*Universidad Nacional Autónoma de México*  
*Facultad de Ingeniería*

---

Curso de Git interno 2025-2

# Información del Curso

## Tiempo estimado

Aproximadamente 60 minutos.

## Objetivos

Que los participantes entiendan:

- ▶ Creación de un repositorio en GitHub
- ▶ Manejo de ramas
- ▶ Flujo de trabajo básico de git

# Introducción al tema

## Qué es Github

Github es una plataforma de desarrollo colaborativo basada en el sistema de control de versiones de git que es ampliamente utilizada en la comunidad de desarrollo de software, pues nos permite almacenar nuestro código fuente y colaborar en diversos proyectos de software.

# Un poco de historia

- ▶ Github fue fundada en 2008 por Tom Preston-Werner, Chris Wanstrath, PJ Hyett y Scott Chacon. Su objetivo era facilitar la colaboración entre desarrolladores.
- ▶ Gracias a este enfoque despegó, convirtiéndose en el centro de código abierto más grande del mundo.
- ▶ En 2018 fue adquirida por Microsoft, lo que generó cierta controversia en la comunidad de código abierto. Sin embargo, GitHub ha mantenido su compromiso con el software de código abierto y ha seguido mejorando la plataforma.

# Crear una cuenta de Github

**Create your free account**

Explore GitHub's core features for individuals and organizations.

[See what's included](#) ▾

Already have an account? [Sign in](#) →

**Sign up to GitHub**

Email\*  
gerardoc6630@outlook.com ✓

Password\*  
\*\*\*\*\* ✓

Password should be at least 15 characters OR at least 8 characters including a number and a lowercase letter.

Username\*  
GerardoSoyYq

Username may only contain alphanumeric characters or single hyphens, and cannot begin or end with a hyphen.

Your country\*  
Mexico ▾

For compliance reasons, we're required to collect country information to send you occasional updates and announcements.

Email preferences  
☐ Receive occasional product updates and announcements

**Continue** >

By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails.

Figura 1: Crea una cuenta

# Crear una cuenta de Github

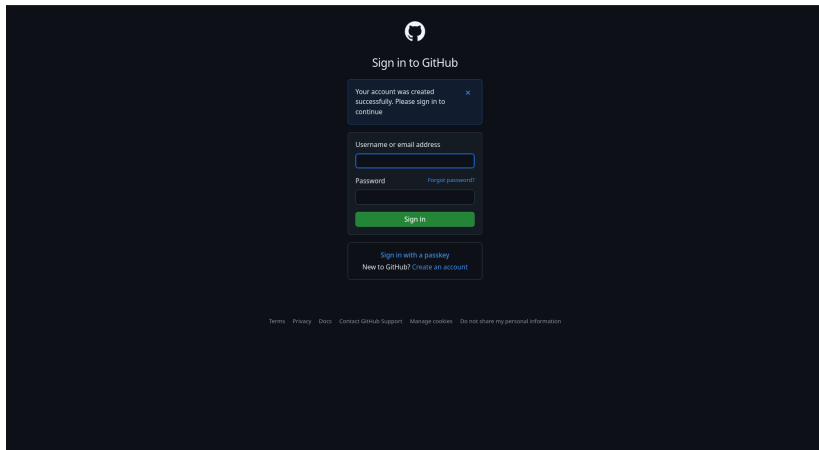


Figura 2: Inicia sesión

# Crear una cuenta de Github

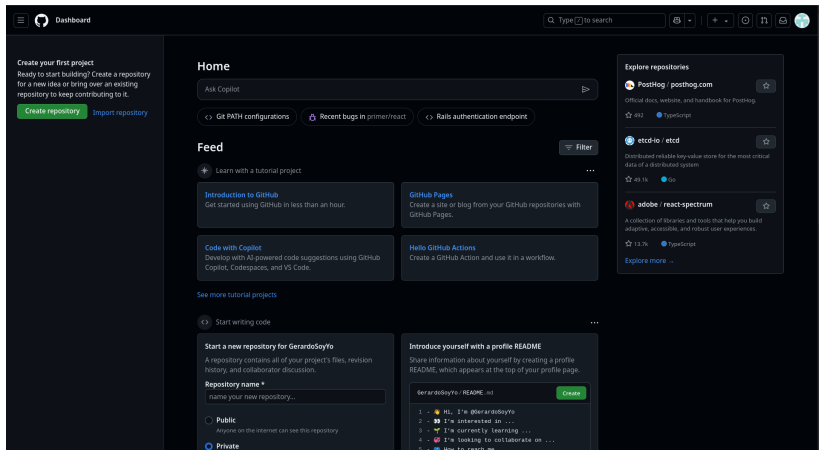


Figura 3: Interfaz de Github

# Configuración de Git

Crea una llave SSH para autenticarte en GitHub. Puedes hacerlo con el siguiente comando:

```
ssh-keygen -o -a 100 -t ed25519 -f /path/to/key -C  
"your@email.com"
```

Figura 4: Comando para generar SSH



# Configuración de Git

Añade la llave pública a tu cuenta de GitHub.

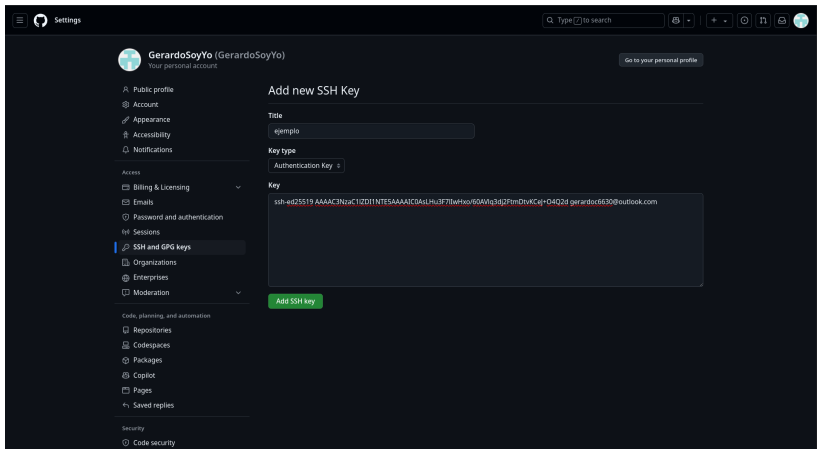


Figura 5: SSH and GPG keys

# Configuración de Git

Configura tu nombre de usuario y correo electrónico en Git. Esto es importante para que tus commits se registren correctamente.

```
# Configure name
git config --global user.name "Your Name"

# Configure email
git config --global user.email "your@email.com"

# Configure editor
git config --global core.editor "vim"

# Configure color
git config --global color.ui auto
```

Figura 6: Configuración de git

# Creación de un repositorio en GitHub

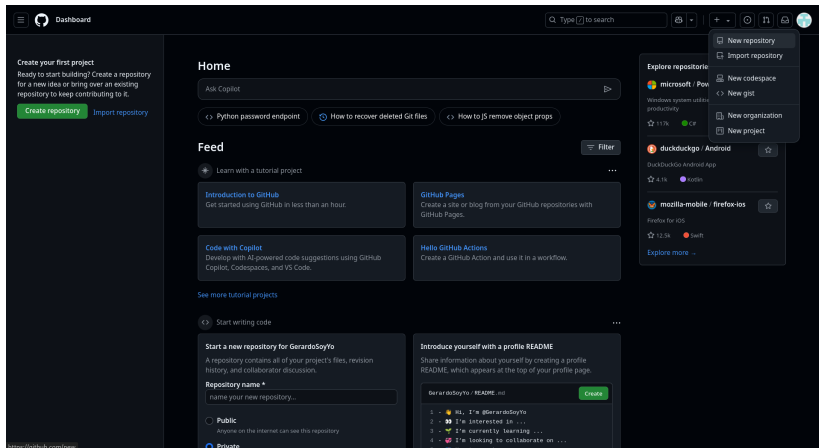
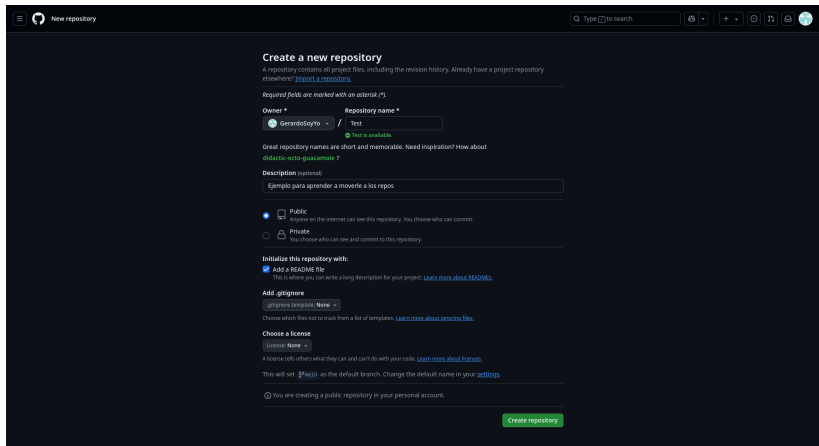


Figura 7: Crear un repositorio

# Creación de un repositorio en GitHub



The screenshot shows the GitHub 'Create a new repository' interface. At the top, there's a header with the GitHub logo and 'New repository' text. A search bar is on the right. The main heading is 'Create a new repository', followed by a subtext explaining that a repository contains project files and revision history. A link to 'Import a repository' is provided. Below this, a note states 'Required fields are marked with an asterisk (\*)'. The 'Owner' field is set to 'GerardoSoyto' and the 'Repository name' field contains 'Test', with a green checkmark indicating 'Test is available'. A tip suggests great repository names are short and memorable, with an example 'didactic-octo-guacamole'. The 'Description (optional)' field contains 'Ejemplo para aprender a moverle a los repos'. Under 'Visibility', the 'Public' radio button is selected, with a note that anyone on the internet can see the repository. The 'Private' option is also available. The 'Initialize this repository with:' section has the 'Add a README file' checkbox checked, with a link to 'Learn more about READMEs'. Below this is the 'Add .gitignore' section with a dropdown menu set to 'None'. The 'Choose a license' section also has a dropdown menu set to 'None', with a link to 'Learn more about licenses'. A note at the bottom states 'This will set main as the default branch. Change the default name in your settings.' and a checkbox for 'You are creating a public repository in your personal account.' is checked. A green 'Create repository' button is at the bottom right.

Create a new repository

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

Required fields are marked with an asterisk (\*).

Owner \* / Repository name \*

GerardoSoyto / Test

Test is available

Great repository names are short and memorable. Need inspiration? How about [didactic-octo-guacamole](#)?

Description (optional)

Ejemplo para aprender a moverle a los repos

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

☐ Private Only those you choose who can see and commit to this repository.

Initialize this repository with:

☒ Add a README file This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

gitignore template: None

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

Choose a license

License: None

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

This will set `main` as the default branch. Change the default name in your [settings](#).

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

Create repository

Figura 8: Configuración de un repositorio

# Creación de un repositorio en GitHub

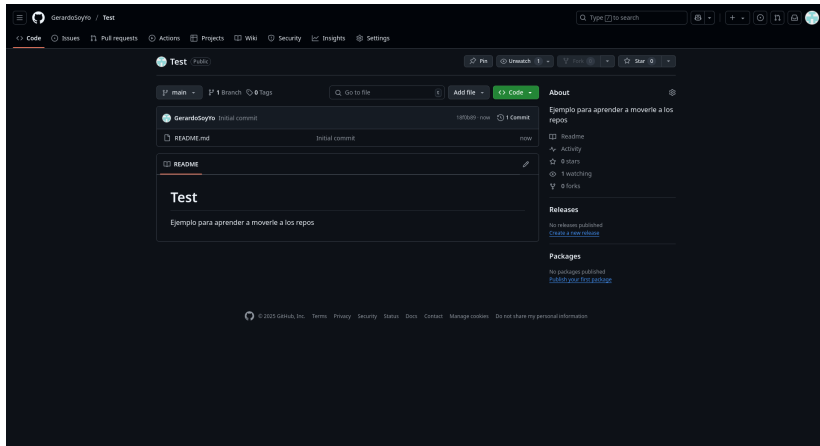


Figura 9: Nuevo repositorio

# Configuración de un Repositorio

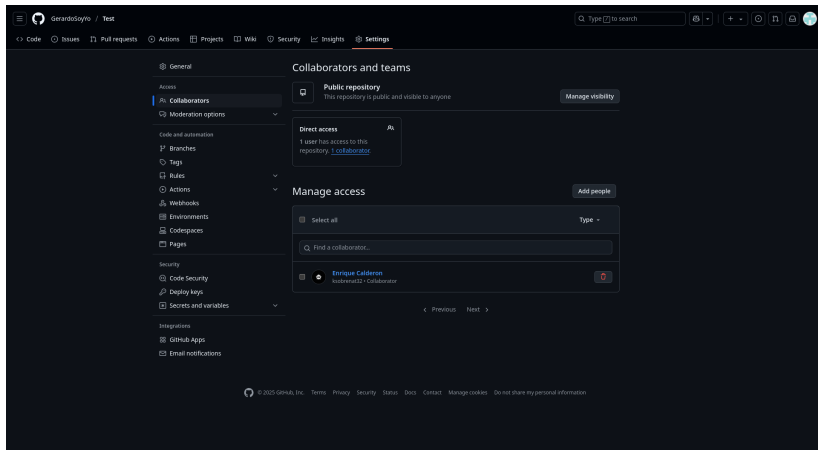


Figura 10: Añadir colaboradores

# Configuración de un Repositorio

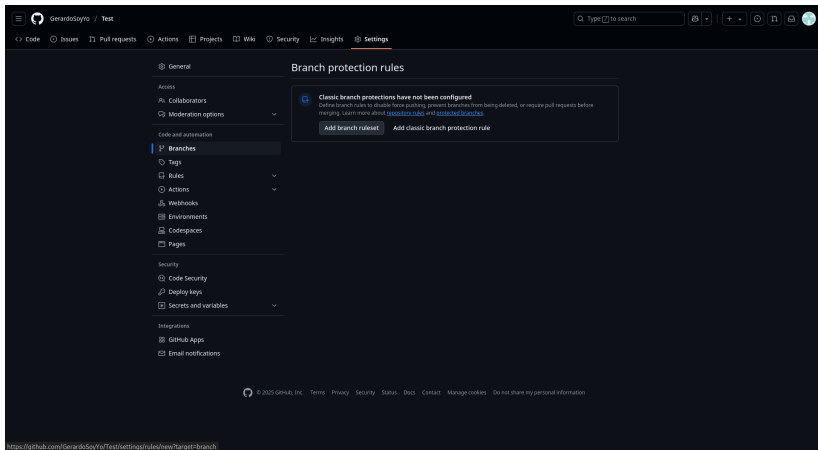


Figura 11: Añadir restricciones

# Configuración de un Repositorio

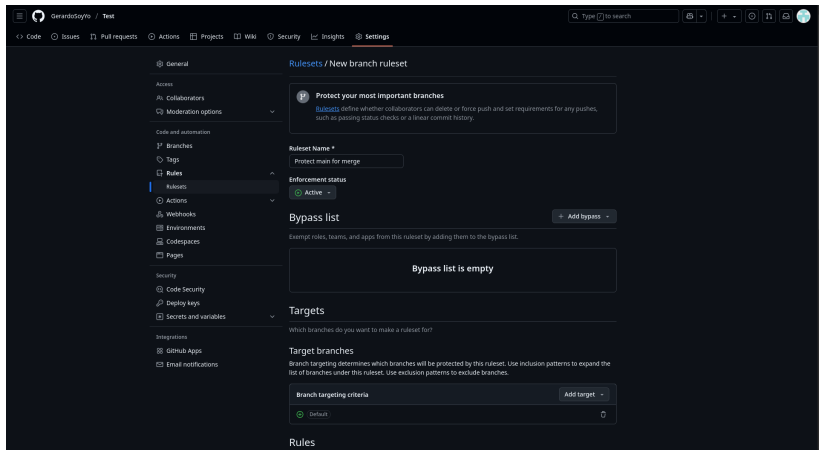
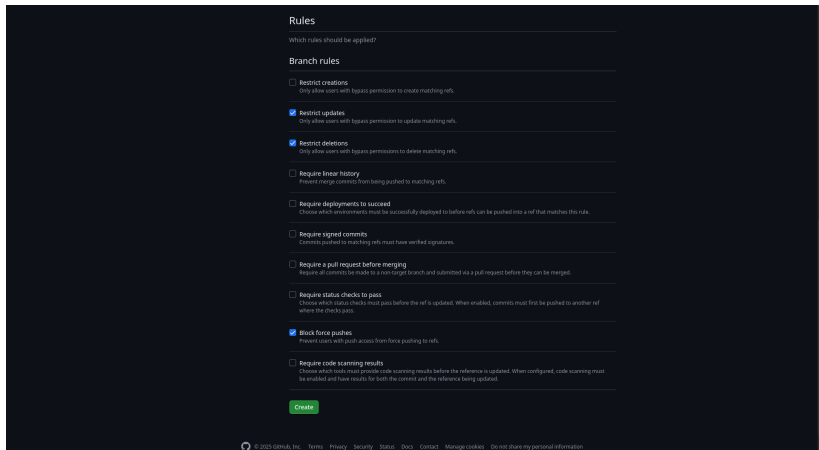


Figura 12: Configuración de restricciones



# Configuración de un Repositorio



**Rules**

Which rules should be applied?

**Branch rules**

- ☐ **Restrict creations**  
Only allow users with bypass permission to create matching refs.
- ☒ **Restrict updates**  
Only allow users with bypass permission to update matching refs.
- ☒ **Restrict deletions**  
Only allow users with bypass permissions to delete matching refs.
- ☐ **Require linear history**  
Prevent merge commits from being pushed to matching refs.
- ☐ **Require deployments to succeed**  
Choose which environments must be successfully deployed to before refs can be pushed into a ref that matches this rule.
- ☐ **Require signed commits**  
Commits pushed to matching refs must have verified signatures.
- ☐ **Require a pull request before merging**  
Require all commits be made to a non-target branch and submitted via a pull request before they can be merged.
- ☐ **Require status checks to pass**  
Choose which status checks must pass before the ref is updated. When enabled, commits must first be pushed to another ref where the checks pass.
- ☒ **Block force pushes**  
Prevent users with push access from force pushing to refs.
- ☐ **Require code scanning results**  
Choose which bots must provide code scanning results before the reference is updated. When configured, code scanning must be enabled and have results for both the commit and the reference being updated.

[Create](#)

© 2025 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact](#) [Manage cookies](#) [Do not share my personal information](#)

Figura 13: Configuración de restricciones

# Manejo de Ramas

Las ramas son una característica fundamental de Git y GitHub que permiten a los desarrolladores trabajar en diferentes versiones de un proyecto simultáneamente.

Las ramas son útiles para:

- ▶ **Desarrollo paralelo**
- ▶ **Pruebas y experimentación**
- ▶ **Revisión de código**
- ▶ **Mantenimiento**
- ▶ **Despliegue**

# Tipos de Ramas

En GitHub, existen varios tipos de ramas que se utilizan para diferentes propósitos en el desarrollo de software. Algunos de los tipos de ramas más comunes son:

## Ramas Locales

Es una rama que solo existe en tu entorno de trabajo. Estas son creadas, y son las que puedes modificar y gestionar dentro de tu equipo.

## Características

- ▶ **Visible solo localmente**
- ▶ **No se pueden compartir directamente**
- ▶ **Independientes**
- ▶ **Trabajo diario**

# Tipos de Ramas

## Ramas Locales

```
# Crear una rama local  
git checkout -b <nombre de la rama>  
#Listar las ramas locales  
git branch  
# Cambiar entre ramas locales  
git checkout <nombre de la rama>  
# Eliminar una rama local  
git branch -d <nombre de la rama>
```

Figura 14: Comandos básicos para manejar ramas

# Tipos de Ramas

## Ramas Remotas

Es una rama que existe desde un repositorio remoto. Estas se utilizan para compartir tu trabajo con otros desarrolladores o para mantener una versión centralizada del proyecto.

## Características

- ▶ **Visibles para todos los colaboradores**
- ▶ **Sincronización con el repositorio remoto**
- ▶ **Acceso Compartido**

# Tipos de Ramas

## Ramas Remotas

```
# Listar las ramas remotas
git branch -r
# Subir una rama local al repositorio remoto
git push origin <nombre de la rama>
# Eliminar una rama remota
git push origin --delete <nombre de la rama>
# Traer una rama remota a tu entorno local
git fetch origin <nombre de la rama>
# Sincronizar una rama remota con tu entorno local
git pull origin <nombre de la rama>
# Crear una rama local que rastree una rama remota
git checkout -b <nombre de la rama> origin/<nombre de la rama>
# Cambiar el nombre de una rama remota
git push origin :<nombre de la rama> <nuevo nombre>
```

Figura 15: Otros comandos para ramas remotas

# PULL Y PUSH

## Pull

- ▶ Es uno de los más utilizados en Github.
- ▶ Se utiliza para descargar cambios desde un repositorio remoto y aplicarlos a una rama local.
- ▶ Es un comando que combina dos acciones: `git fetch` y `git merge`.

Su sintaxis es la siguiente:

```
git pull origin <nombre de la rama>
```

# PULL Y PUSH

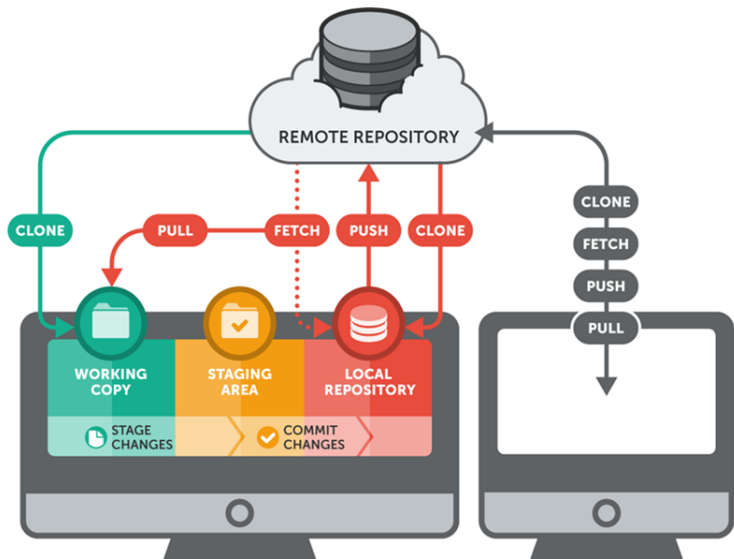
## Push

Lo que hace es “empujar” tus commits locales al repositorio remoto para que otros desarrolladores puedan ver tus cambios.

```
git push origin <nombre de la rama>
```



# PULL Y PUSH



# Clasificación de una rama

- ▶ Rama principal (main o master)
- ▶ Ramas de características (feature branches)
- ▶ Ramas de corrección de errores (fix branches)

# Flujo de trabajo básico de git

El flujo de trabajo básico con Git implica los siguientes pasos:

## 1. Clonar un repositorio

- ▶ Clonar un repositorio remoto a tu máquina local.
- ▶ Comando: `git clone <URL del repositorio>`.

## 2. Crear una rama

- ▶ Crear una nueva rama para trabajar en una característica o corrección de errores.
- ▶ Comando: `git checkout -b <nombre de la rama>`.

## 3. Asegurate de trabajar con los últimos cambios

- ▶ Asegúrate de que tu rama esté actualizada con la rama principal (main o master).
- ▶ Comando: `git pull origin <nombre de la rama>`.

## 4. Realizar cambios

- ▶ Realizar cambios en el código fuente en la nueva rama.

# Flujo de trabajo básico de git

## 5. Agregar cambios

- ▶ Agregar los cambios realizados al área de preparación (staging area).
- ▶ Comando: `git add <archivo>` o `git add .` para agregar todos los archivos.

## 6. Confirmar cambios

- ▶ Realiza una snapshot con los cambios más significativos de tu rama.
- ▶ Comando: `git commit -m "Descripción de los cambios"`.

## 7. Sincronizar cambios

- ▶ Sincroniza tu rama con el repositorio remoto.
- ▶ Comando: `git push origin <nombre de la rama>`.

# Flujo de trabajo básico de git

## 8. Crear un Pull Request

- ▶ Crea un Pull Request (PR) en GitHub para solicitar la revisión y fusión de tus cambios en la rama principal.

## 9. Revisar y fusionar

- ▶ Revisa los comentarios y sugerencias de otros desarrolladores en el PR.
- ▶ Si todo está bien, fusiona tu rama con la rama principal (main o master).

## 10. Eliminar la rama

- ▶ Una vez que la fusión se haya completado, puedes eliminar la rama que creaste.
- ▶ Comando: `git branch -d <nombre de la rama>` para eliminar la rama local y `git push origin --delete <nombre de la rama>` para eliminarla del repositorio remoto.

# Clonando un repositorio

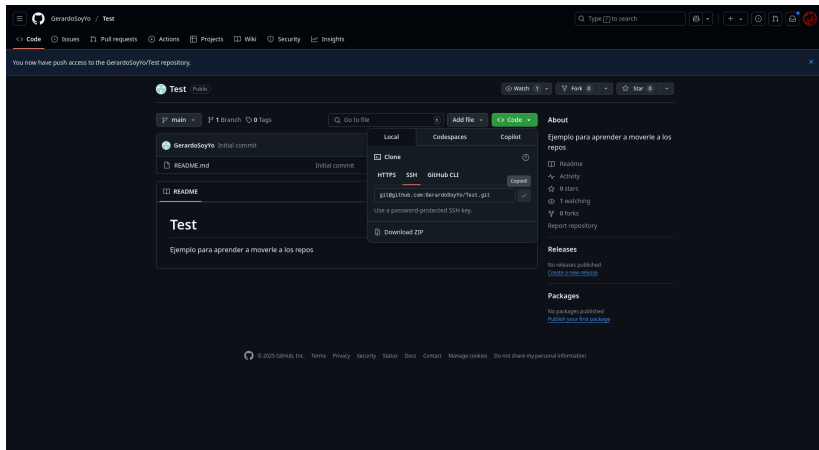
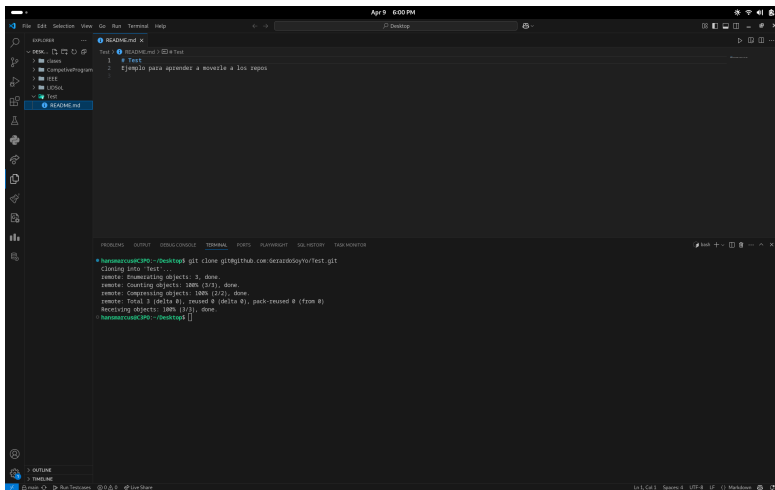


Figura 17: Copia la liga del repo

# Clonando un repositorio



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays a file tree with folders like 'DEMO', 'Clases', 'CompetitiveProgram', 'IEEE', 'LIDSOL', and 'Test'. The 'Test' folder is selected, showing a 'README.md' file. The main editor area displays the content of 'README.md', which includes a heading '# Test' and a paragraph 'Ejemplo para aprender a moverle a los repos'. The TERMINAL panel at the bottom shows the execution of the command 'git clone git@github.com:Serardosyo/Test.git'. The output indicates a successful clone, showing progress for enumerating, counting, and compressing objects, and finally receiving the repository data.

```
hansmarcusCPO@Desktop:~$ git clone git@github.com:Serardosyo/Test.git
Cloning into 'Test'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
hansmarcusCPO@Desktop:~$
```

Figura 18: Clonando Repositorio

# Como hacer un Fork

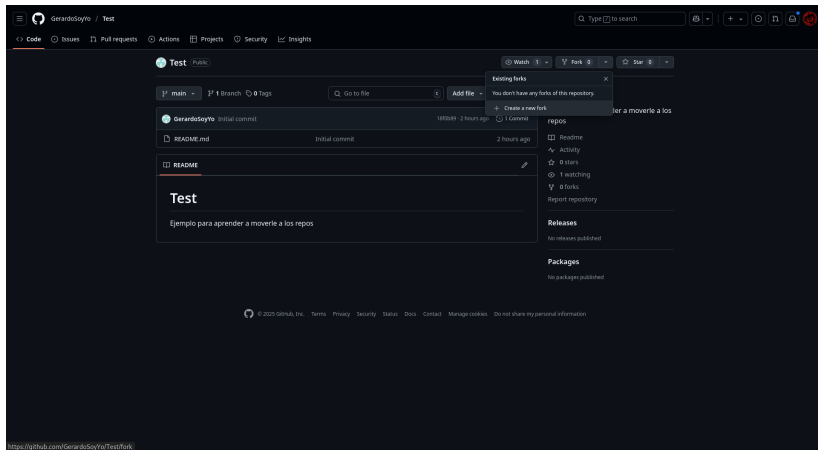


Figura 19: Creando un Fork



# Como hacer un Fork

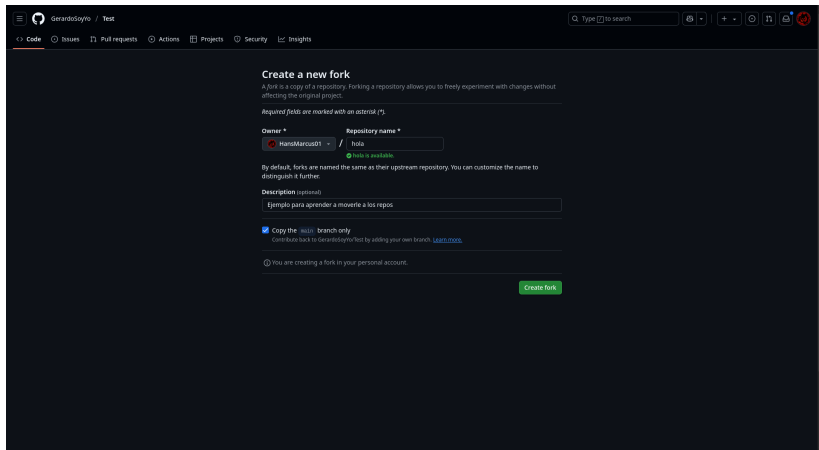
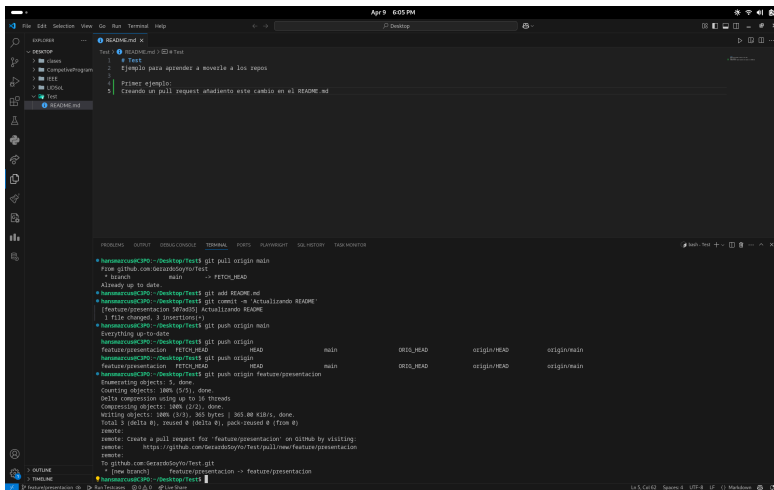


Figura 20: Configuración de un Fork

# Como hacer un PULL REQUEST



The screenshot shows a Visual Studio Code editor with a file explorer on the left showing a project structure with folders like 'classes', 'CompiladorPrograma', 'HERR', 'LOCAL', and 'Test'. The 'README.md' file is open in the editor, containing the following text:

```
1 # Test
2 Ejemplo para aprender a moverse a los repos
3
4 Primer ejemplo:
5 Creando un pull request añadiendo este cambio en el README.md
```

Below the editor, the 'TERMINAL' tab is active, displaying the following commands and their output:

```
harmaracuac@CPO:~/Desktop/Test$ git pull origin main
First github.com/GerardoSoyYo/Test
* branch main -> FETCH_HEAD
Already up-to-date.
harmaracuac@CPO:~/Desktop/Test$ git add README.md
harmaracuac@CPO:~/Desktop/Test$ git commit -m 'Actualizando README'
[feature/presentation 587ad30] Actualizando README
1 file changed, 3 insertions(+)
harmaracuac@CPO:~/Desktop/Test$ git push origin main
Everything up-to-date
harmaracuac@CPO:~/Desktop/Test$ git push origin
feature/presentation FETCH_HEAD HEAD main ORIS_HEAD origin/HEAD origin/main
harmaracuac@CPO:~/Desktop/Test$ git push origin
feature/presentation FETCH_HEAD HEAD main ORIS_HEAD origin/HEAD origin/main
harmaracuac@CPO:~/Desktop/Test$ git push origin feature/presentation
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 365 bytes | 365.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Create a pull request for 'feature/presentation' on github by visiting:
remote: https://github.com/GerardoSoyYo/Test/pull/new/feature/presentation
remote:
To github.com:GerardoSoyYo/Test git
+ (New branch) feature/presentation -> feature/presentation
harmaracuac@CPO:~/Desktop/Test$
```

Figura 21: Realizando un cambio

# Como hacer un PULL REQUEST

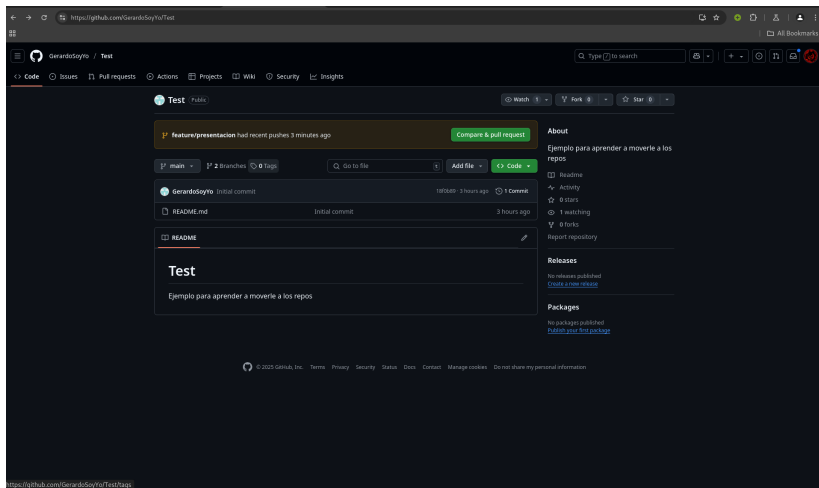


Figura 22: Interfaz del colaborador

# Como hacer un PULL REQUEST

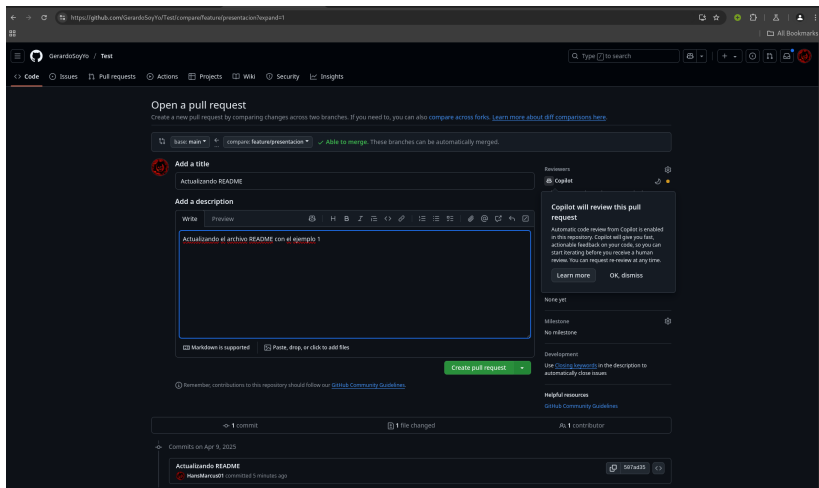


Figura 23: Creando el pull request

# Como hacer un PULL REQUEST

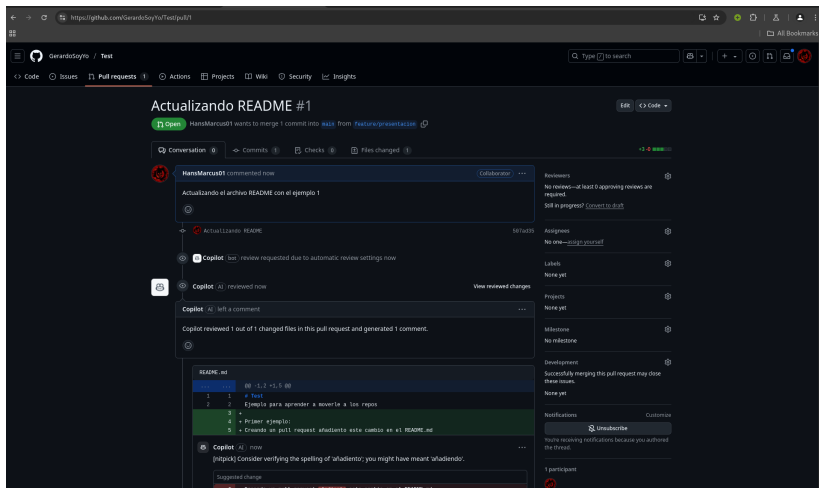


Figura 24: Visualización del pull request

# Como hacer un PULL REQUEST

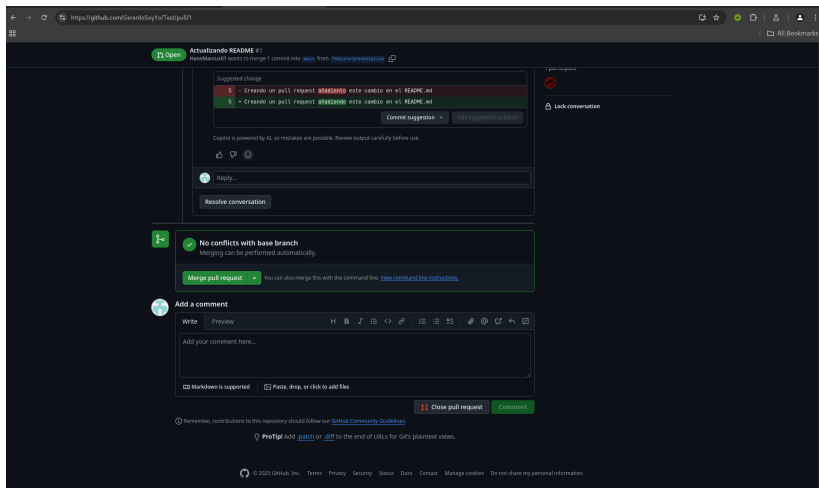


Figura 25: Perspectiva desde el dueño del repo

# Bibliografía

- ▶ <https://git-scm.com/book/en/v2>
- ▶ <https://marklodato.github.io/visual-git-guide/index-es.html>
- ▶ <https://blog.kinto-technologies.com/posts/2023-03-07-From-Git-flow-to-GitHub-flow/>