

## UTN - Zerpa Margarita – Comisión 23

### Práctico 2: Git y GitHub

#### Objetivo:

El estudiante desarrollará competencias para trabajar con Git y GitHub, aplicando conceptos fundamentales de control de versiones, colaboración en proyectos y resolución de conflictos, en un entorno simulado y guiado.

#### Resultados de aprendizaje:

1. Comprender los conceptos básicos de Git y GitHub: Identificar y explicar los principales términos y procesos asociados con Git y GitHub, como repositorios, ramas, commits, forks, etiquetas y repositorios remotos.
2. Manejar comandos esenciales de Git: Ejecutar comandos básicos para crear, modificar, fusionar y gestionar ramas, commits y repositorios, tanto en local como en remoto.
3. Aplicar técnicas de colaboración en GitHub: Configurar y utilizar repositorios remotos, realizar forks, y gestionar pull requests para facilitar el trabajo colaborativo.
4. Resolver conflictos en un entorno de control de versiones: Identificar, analizar y solucionar conflictos de merge generados en un flujo de trabajo con múltiples ramas.

#### Actividades

- 1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas):
  - **¿Qué es GitHub?**

Es una plataforma de hosting de código para control de versiones y colaboración. Nos permite trabajar junto con otras personas en proyectos desde cualquier lugar, usando GIT.

Te permite:

    - Presentar o compartir** el trabajo
    - Seguir y Administrar** los cambios en el código a lo largo del tiempo.
    - Dejar que otros usuarios revisen el código y realicen sugerencias para mejorarlo.
    - Colaborar** en un proyecto compartido
  - **¿Cómo crear un repositorio en GitHub?**

Ingresar a la página <https://github.com/> y crear cuenta

New repositorio, dar nombre al repositorio

Opcionalmente inicializar con README

Puede ser privado o público

Terminar con Create repository



Sign in to GitHub

Username or email address

Password

[Forgot password?](#)

Sign in

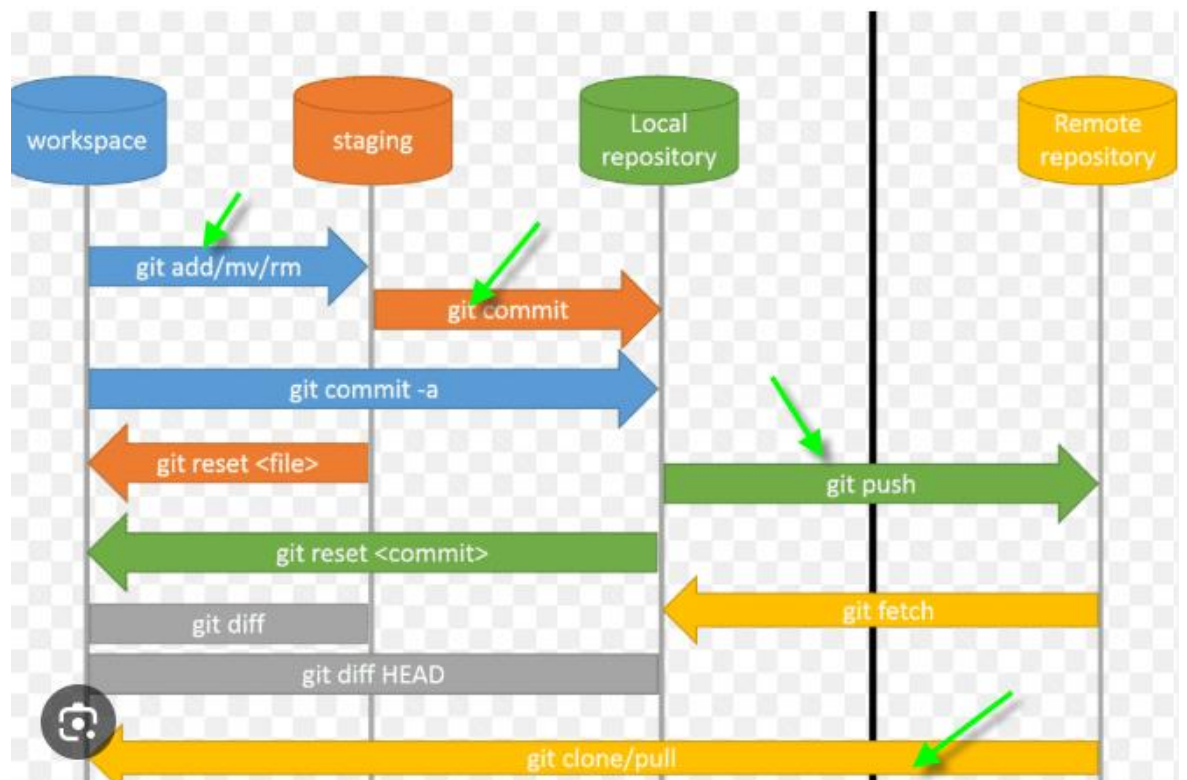
[Sign in with a passkey](#)

New to GitHub? [Create an account](#)

- ¿Cómo crear una rama en Git?  
Comando **git branch** <RamaNueva>, este crea una nueva rama.  
Comando **git checkout -b** <RamaNueva >, este crea y cambia a la nueva rama.
- ¿Cómo cambiar a una rama en Git?  
Comando **git checkout** <Rama>, este cambia a otra rama.  
Comando **git checkout -b** <RamaNueva >, este crea y cambia a la nueva rama.
- ¿Cómo fusionar ramas en Git?
  - Requisitos, estar parados en la rama principal, main o master.
  - Comando **git merge** <Rama>
- ¿Cómo crear un commit en Git?

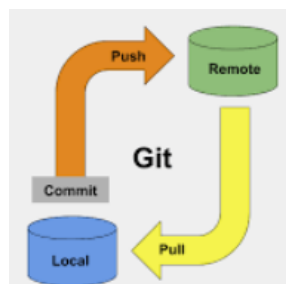


- Previamente agregar todos los archivos al Stage con alguno de estos comandos (git add . / git add -A / git add -all / git add archivo1 archivo2 ... archive N)
  - Comando **git commit -m** "mensaje Opcion Ej Primer Commit"
- ¿Cómo enviar un commit a GitHub?



- Previamente haber realizado un **git clone**, **git add**, **git commit -m**
- Comando **git push -u origin master**, este sube los cambios al remoto

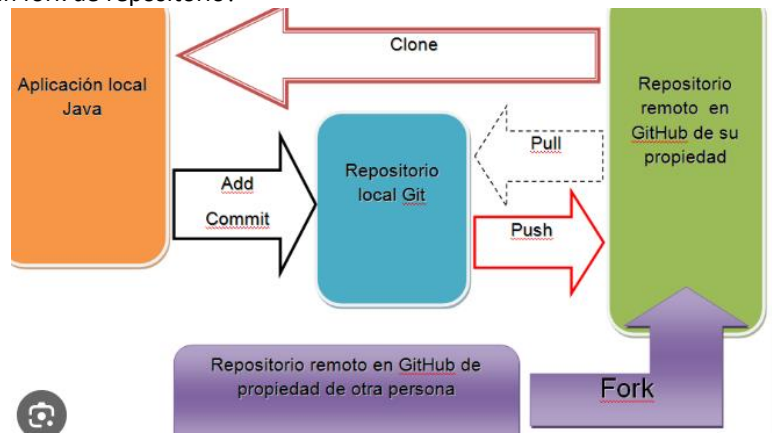
- ¿Qué es un repositorio remoto?  
Un repositorio remoto, es un lugar en internet donde almacenamos nuestro repositorio, estos repositorios remotos facilitan el trabajo colaborativo
- ¿Cómo agregar un repositorio remoto a Git?  
- Comando **git remote add** origin <url de nuestro repositorio>, añadimos un nuevo repositorio al remoto.  
- Comando **git remote -v**, con este visualizamos los distintos repositorios remotos que están conectados a nuestro repositorio local.
- ¿Cómo empujar cambios a un repositorio remoto?  
Este comando **git push** se usa para enviar cambios de un repositorio local a un repositorio remoto,  
-Comando **git push <nombre\_repo\_remoto> <nombre\_rama\_enviar>**, este comando sube cambios al repositorio remoto.
- ¿Cómo tirar de cambios de un repositorio remoto?  
El comando **git pull** permite descargar contenido de un repositorio remoto y actualizar el repositorio local.



- Debo estar parada en mi rama < nombreRama>.
- Comando **git pull <origin> < nombreRama>**, este comando aplica cambios desde el

repositorio remoto.

- ¿Qué es un fork de repositorio?



Un fork es una copia exacta de un repositorio de propiedad de otra persona. Es una copia en mi cuenta de GitHub.

- ¿Cómo crear un fork de un repositorio?

**Seleccionar el repositorio:** Inicia sesión en GitHub y navega hasta el repositorio que deseas bifurcar.

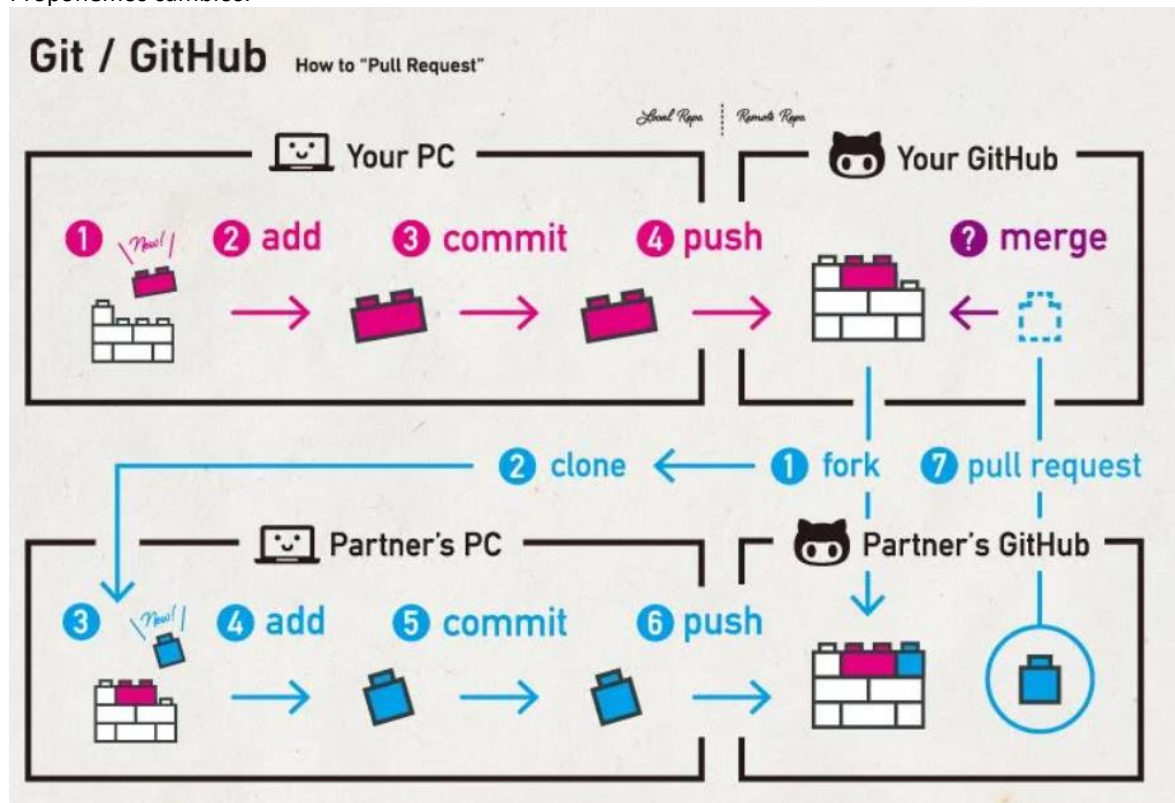
**Hacer clic en "Fork":** En la parte superior derecha de la página del repositorio, verás un botón que dice "Fork". Al hacer clic en este botón, GitHub creará una copia del repositorio en tu cuenta.

**Clonar el fork:** Una vez que has bifurcado el repositorio, puedes clonarlo en tu máquina local usando el comando `git clone` y la URL de tu fork. Esto te permitirá trabajar en el código de manera local.

**Hacer cambios y comitear:** Ahora puedes realizar los cambios que desees en el código. Una vez que hayas terminado, puedes hacer commits de tus modificaciones.

**Push y pull request:** Finalmente, sube tus cambios a tu repositorio en GitHub usando el comando `git push`. Si deseas contribuir al proyecto original, puedes crear un pull request, lo que permitirá a los administradores del repositorio original revisar y, potencialmente, aceptar tus cambios en su código.

- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?  
Proponemos cambios.



Pasos:

- 1) Hacer un fork del repositorio
  - 2) Clonar el repositorio a mi maquina local: `git clone`
  - 3) Realizar los cambios y hacer los commits.
  - 5) Subir la rama a mi repositorio: `git push origin mi-nueva-rama`
  - 6) Crear la pull-request en GitHub: Describir cambios y propósitos del cambio.
- ¿Cómo aceptar una solicitud de extracción?  
Para aceptar la solicitud de unas extracciones (PR) en GitHub, revisar los cambios, dejar comentarios, y aprobar la solicitud.
  - ¿Qué es una etiqueta en Git?  
Comando `git tag` comprende los diferentes tipos de etiquetas, como crearlas, eliminarlas, compartirlas para un flujo eficiente.
  - ¿Cómo crear una etiqueta en Git?  
Usar un identificador que refleje estado del repositorio al momento de su creación.  
Git admite etiquetas anotada y ligeras.  
**Las etiquetas anotadas** almacenan información adicional, como la fecha, el etiquetador y el correo electrónico, y son ideales para lanzamientos públicos.  
**Las etiquetas ligeras** son más sencillas y se utilizan como marcadores para una confirmación específica.  
Comando `git tag <nombre-Etiqueta>`
  - ¿Cómo enviar una etiqueta a GitHub?  
Para enviar etiquetas específicas usar:  
Comando `git push origin`  
Para enviar varias etiquetas a la vez  
Comando `git push origin --tags`

- ¿Qué es un historial de Git?  
El historial de GIT es un registro de todas las confirmaciones de un repositorio. Cada confirmación (commit) es una foto instantánea, se puede ver quien hizo los cambios, cuando y que archivos modifíco.  
Secure Hash Algorithm (SHA).  
Autor.  
Fecha.  
Mensaje del commit.
- ¿Cómo ver el historial de Git?  
Comando **git log**

- ¿Cómo buscar en el historial de Git?

<b>git log</b>	<b>La variante estándar enumera todos los commit en orden inverso.</b>
git log -p	Con este comando, puedes enumerar en detalle todos los cambios.
git log --stat	Muestra las estadísticas de los cambios y los enumera claramente.
git log --graph --decorate --oneline	Muestra los cambios en forma de gráfica.
git log -n <cantidad>	Usa este comando para limitar el número de commit que muestra.
git log --oneline	Resume cada commit en una sola línea para una mejor visión de conjunto.
git log --author="nombre del autor"	Muestra solo los commit de un autor específico.
git log --grep="mensaje"	Filtra los commit en función de su contenido.
git log <desde>.. <b>&lt;hasta&gt;</b>	Muestra solo los commit realizados entre un rango concreto de revisiones.
git log <archivo>	Muestra solo los commit que contienen un archivo específico.
git log --pretty	Cambia el formato de salida según tus especificaciones.

- ¿Cómo borrar el historial de Git?  
**git reset**  
Restablece hasta el último commit válido que quieres mantener  
Haz un nuevo push con la opción -f para forzar la sobrescritura del historial  
Usa git reset --hard <commit> para eliminar todos los cambios desde el commit <commit>

#### **git rebase**

Usa git rebase para mover tu rama a cualquier punto del historial de commits  
Reemplaza <commit-hash> con el hash de la confirmación justo antes de la que deseas eliminar.

- ¿Qué es un repositorio privado en GitHub?  
Un repositorio privado en GitHub es un repositorio que solo puede ser accedido por el propietario y las personas a las que se les haya dado permiso explícito. A diferencia de un repositorio público, donde cualquier persona puede ver el código, un repositorio privado está protegido y solo visible para los colaboradores autorizados.
- ¿Cómo crear un repositorio privado en GitHub?  
Iniciar sesión GitHub, acceder a Repositorios.  
Boton "Create a new repository"

Colocar nombre/ descripción al repositorioEscribe  
Click opción Private  
Haz clic en Create repository.

- ¿Cómo invitar a alguien a un repositorio privado en GitHub?

Settings → Collaborators → Add people e ingresa el usuario de GitHub  
Asigna los permisos adecuados (lectura, escritura, administración).

- ¿Qué es un repositorio público en GitHub?  
Un repositorio público en GitHub es un espacio donde se puede almacenar código, archivos y revisiones de forma que esté disponible para cualquier persona en internet. Cualquiera puede ver el código, hacer un fork, clonarlo, etc
- ¿Cómo crear un repositorio público en GitHub?  
Iniciar sesión GitHub, acceder a Repositorios.  
Boton “Create a new repository”  
Colocar nombre/ descripción al repositorioEscribe  
Click opción Publico  
Haz clic en Create repository.

## 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 \*** MBZ7745 / **Repository name \*** Repositorio1MBZ

✓ Repositorio1MBZ is available.

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

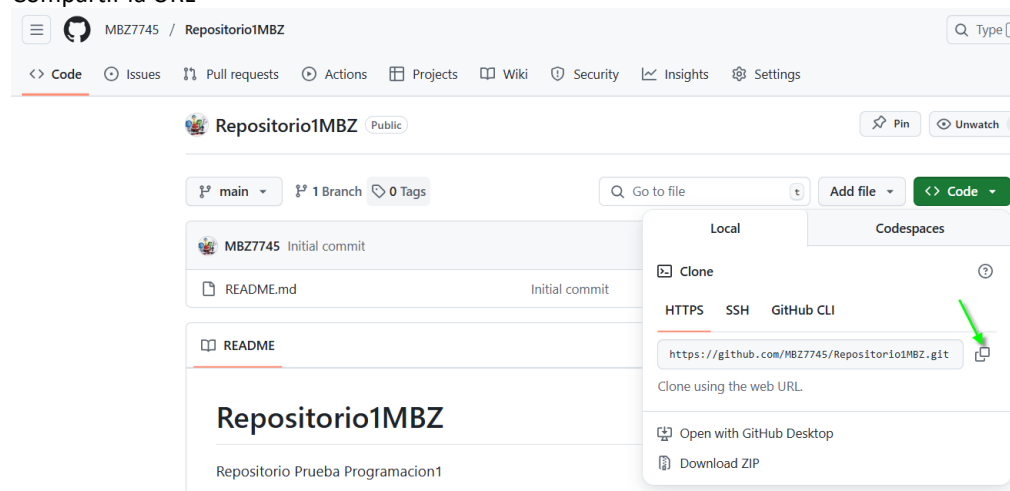
**Description** (optional)

Repositorio Prueba Programacion1

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

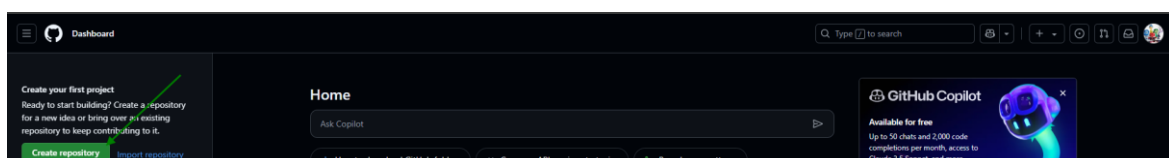
☐ **Private** You choose who can see and commit to this repository.

- ¿Cómo compartir un repositorio público en GitHub?  
Compartir la URL



2) Realizar la siguiente actividad:

- Crear un repositorio.
  - Dale un nombre al repositorio.
  - Elige el repositorio sea público.
  - Inicializa el repositorio con un archivo.






## 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 \*

 MBZ7745

Repository name \*

Repositorio1MBZ

✓ Repositorio1MBZ is available.

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

Description (optional)

Repositorio Prueba Programacion1

☒ 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:

☒ 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


 Repositorio1MBZ Public


 Pin  Unwatch **1**  Fork **0**  Star **0**

 main


 1 Branch

 0 Tags


 Add file


 Code



About


 MBZ7745 Initial commit

b2ed62a · now

 1 Commit

 README.md Initial commit now


 README 


 Repositorio1MBZ


Repositorio Prueba Programacion1


About


Repositorio Prueba Programacion1

 Readme

 Activity

 0 stars

 1 watching

 0 forks

Releases

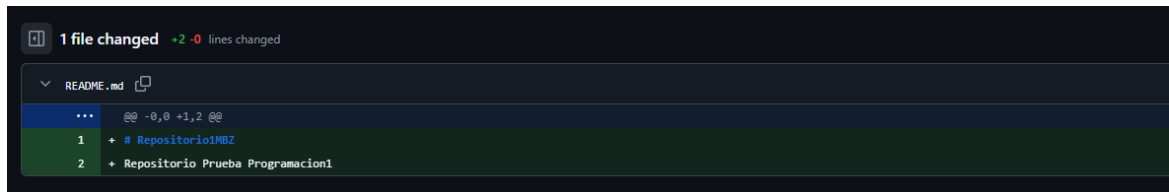
No releases published

[Create a new release](#)

Packages

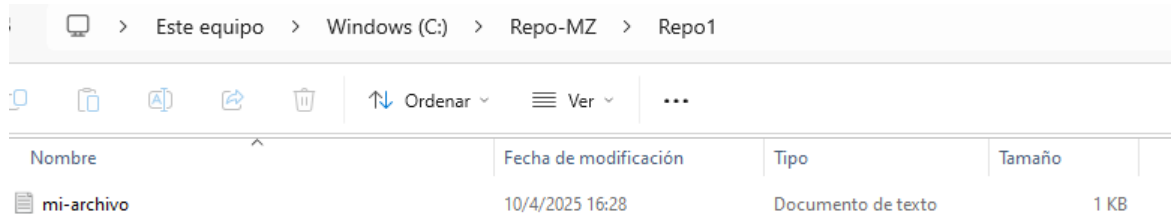
No packages published

[Publish your first package](#)



- Agregando un Archivo
  - Crea un archivo simple, por ejemplo, "mi-archivo.txt".
  - Realiza los comandos git add . y git commit -m "Agregando mi-archivo.txt" en la línea de comandos.
  - Sube los cambios al repositorio en GitHub con git push origin main (o el nombre de la rama correspondiente).

Creo mi archivo



Visualización archivo desde GitBash

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo1
$ ls
mi-archivo.txt
```

Inicialización del Repo1 Local

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo1
$ git init
Initialized empty Git repository in C:/Repo-MZ/Repo1/.git/
```

Configuración Repo1 Local

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo1
$ git config --global user.email "zerpamargarita2712@gmail.com"

zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo1
$ git config --global user.name "MargaritaBZ"
```

Comandos git add . y git commit -m "Agregando mi-archivo.txt"

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo1 (master)
$ git add .

zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo1 (master)
$ git commit -m "Agregando mi-archivo.txt"
[master (root-commit) 250d7bb] Agregando mi-archivo.txt
1 file changed, 1 insertion(+)
create mode 100644 mi-archivo.txt
```

Visualización desde carpeta

Este equipo > Windows (C:) > Repo-MZ > Repo1 >			
<div> <div> <div></div> <div></div> <div></div> <div></div> <div></div> </div> <div> <div>Ordenar</div> <div>Ver</div> <div></div> </div> </div>			
Nombre	Fecha de modificación	Tipo	
.git	10/4/2025 16:43	Carpeta de archivos	
mi-archivo	10/4/2025 16:31	Archivo de origen Text	

GitHub - Creo un nuevo repositorio

## 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 \***  
 MBZ7745

**Repository name \***

Repo1 is available.

Sigo estos pasos, ya que tengo un repo en mi PC local

Repo1 Public

Pin

Unwatch 1

Fork

Star 0

**Set up GitHub Copilot**  
 Use GitHub's AI pair programmer to autocomplete suggestions as you code.  
[Get started with GitHub Copilot](#)

**Add collaborators to this repository**  
 Search for people using their GitHub username or email address.  
[Invite collaborators](#)

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

Set up in Desktop

or

HTTPS

SSH

https://github.com/MBZ7745/Repo1.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 "# Repo1" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/MBZ7745/Repo1.git
git push -u origin main

```

**...or push an existing repository from the command line**  

```

git remote add origin https://github.com/MBZ7745/Repo1.git
git branch -M main
git push -u origin main

```

Copiamos estos comando en GitBash

```

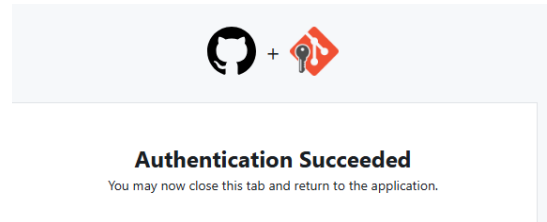
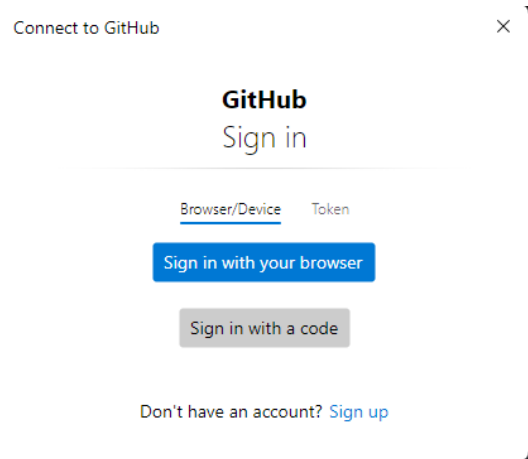
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo1 (master)
$ git remote add origin https://github.com/MBZ7745/Repo1.git

zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo1 (master)
$ git branch -M main

zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo1 (main)
$ git push -u origin main

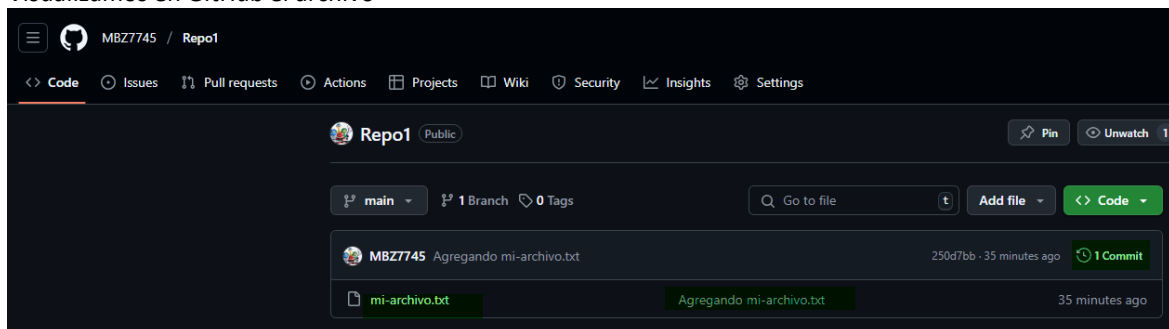
```

Autenticación GitHub



```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo1 (main)
$ git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 265 bytes | 88.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/MBZ7745/Repo1.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Visualizamos en GitHub el archivo



- Creando Branchs
  - Crear una Branch
  - Realizar cambios o agregar un archivo
  - Subir la Branch

Visualizamos la rama git branch, posicionada en \*main

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo1 (main)
$ git branch
* main
```

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo1 (main)
$ git branch Repo1-rama1
```

Creó una rama comando git branch

Cambiar de rama git checkout

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo1 (main)
$ git checkout Repo1-rama1
Switched to branch 'Repo1-rama1'

zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo1 (Repo1-rama1)
```

Visualizamos la rama git branch, posicionada en la \*Repo1-rama1

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo1 (Repo1-rama1)
$ git branch
* Repo1-rama1
  main
```

Comandos git add . y git commit -m "Agregando mi-archivo-2.txt" a la rama Repo1-rama1

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo1 (Repo1-rama1)
$ git add .

zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo1 (Repo1-rama1)
$ git commit -m "Agregando mi-archivo-2.txt"
[Repo1-rama1 1094e51] Agregando mi-archivo-2.txt
1 file changed, 1 insertion(+)
create mode 100644 mi-archivo-2.txt
```

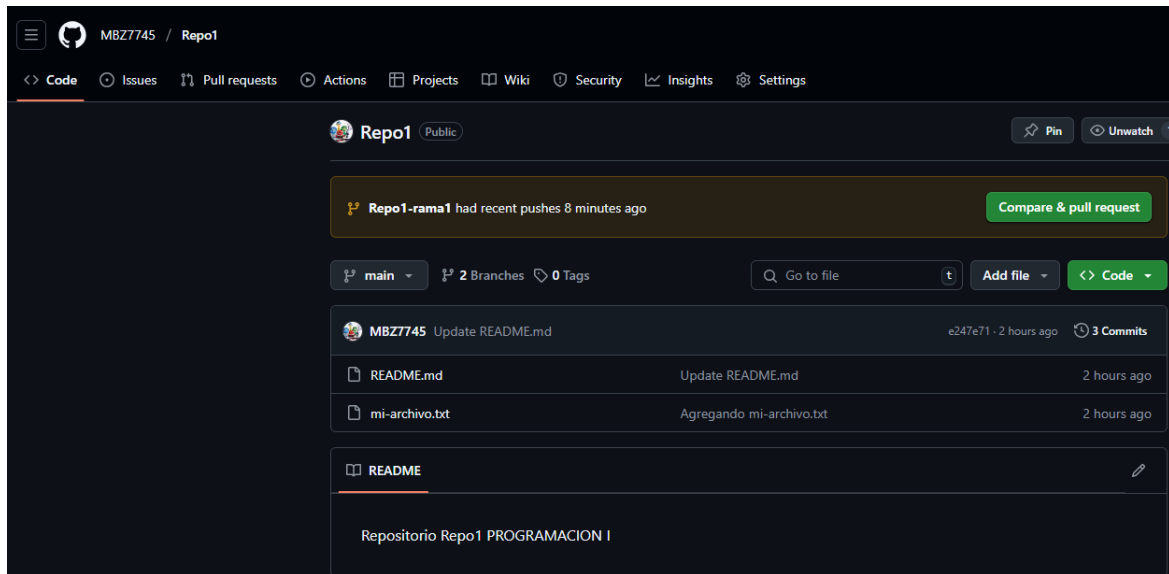
Comando git remote -v

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo1 (main)
$ git remote -v
origin https://github.com/MBZ7745/Repo1.git (fetch)
origin https://github.com/MBZ7745/Repo1.git (push)
```

Subir la Branch(Rama)

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo1 (main)
$ git push origin Repo1-rama1
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 505 bytes | 168.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
remote:
remote: Create a pull request for 'Repo1-rama1' on GitHub by visiting:
remote:   https://github.com/MBZ7745/Repo1/pull/new/Repo1-rama1
remote:
To https://github.com/MBZ7745/Repo1.git
 * [new branch]      Repo1-rama1 -> Repo1-rama1
```

Branch en GitHub



3) Realizar la siguiente actividad:

Paso 1: Crear un repositorio en GitHub


- Ve a GitHub e inicia sesión en tu cuenta.
- Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.
- Asigna un nombre al repositorio, por ejemplo, conflict-exercise.
- Opcionalmente, añade una descripción.
- Marca la opción "Initialize this repository with a README".
- Haz clic en "Create repository".

# 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 \*

 MBZ7745

Repository name \*



conflict-exercise

 conflict-exercise is available.

Great repository names are short and memorable. Need inspiration? How about **super-duper-octo-fiesta** ?

## Description (optional)

Prueba de conflictos

- ☒  **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:

- ☒ **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

 **conflict-exercise** **Public**

 Pin

 Unwatch **1**

 **main**  **1 Branch**  **0 Tags**

 Go to file





**Add file**

**<> Code**

 **MBZ7745** Initial commit


3add3b3 · now

 **1 Commit**


 README.md

Initial commit

now

 **README**



 **conflict-exercise**

Prueba de conflictos

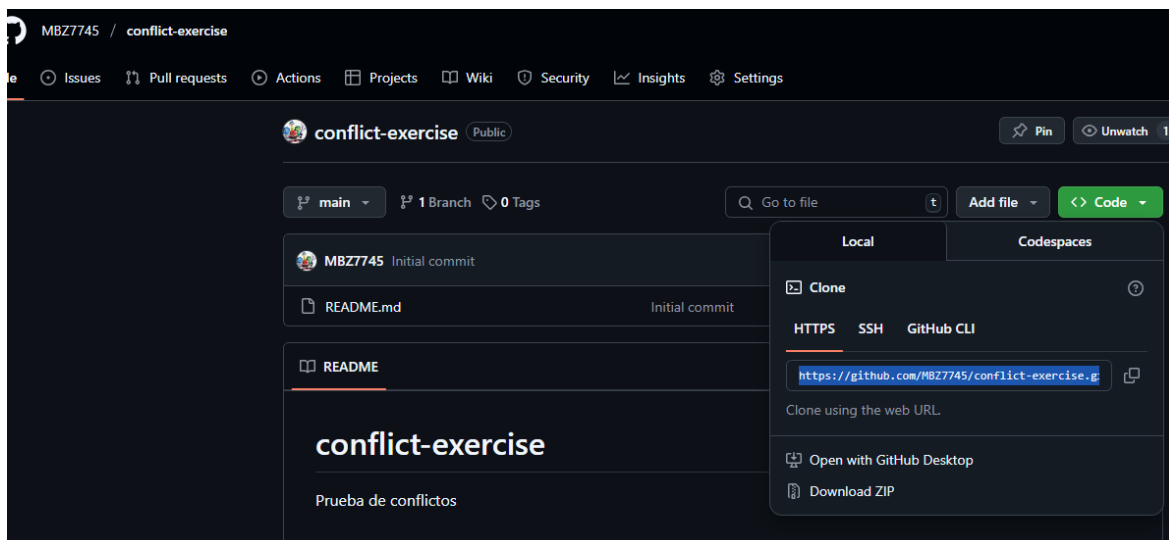
Paso 2: Clonar el repositorio a tu máquina local

- Copia la URL del repositorio (usualmente algo como `https://github.com/tuusuario/conflict-exercise.git`).
- Abre la terminal o línea de comandos en tu máquina.
- Clona el repositorio usando el comando:

```
git clone https://github.com/tuusuario/conflict-exercise.git
```

- Entra en el directorio del repositorio:

```
cd conflict-exercise
```



Comando **git clone**

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2
$ git clone https://github.com/MBZ7745/conflict-exercise.git
```

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2
$ git clone https://github.com/MBZ7745/conflict-exercise.git
Cloning into 'conflict-exercise'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
```

comando **cd**

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2
$ ls
conflict-exercise/
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2
$ cd conflict-exercise
```



Paso 3: Crear una nueva rama y editar un archivo

- Crea una nueva rama llamada feature-branch:

```
git checkout -b feature-branch
```

- Abre el archivo README.md en un editor de texto y añade una línea nueva, por ejemplo:

Este es un cambio en la feature branch.

- Guarda los cambios y haz un commit:

```
git add README.md
```

```
git commit -m "Added a line in feature-branch"
```

Comando git branch

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (main)
$ git branch
* main
```

Comando git checkout

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'
```

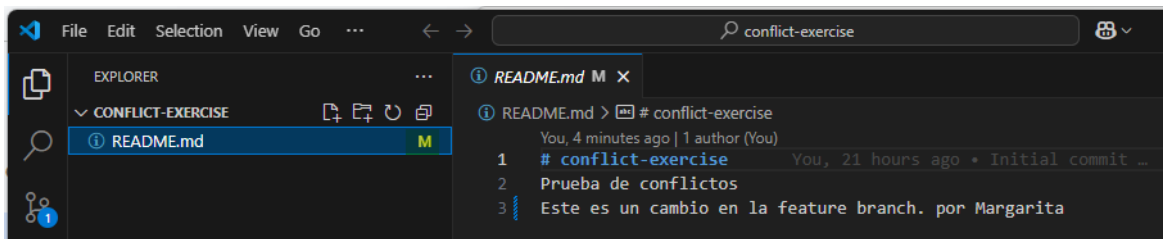
comando git branch

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (feature-branch)
$ git branch
* feature-branch
  main
```

Edición archivo README

Nombre	Fecha de modificación	Tipo
.git	11/4/2025 16:25	Carpeta de archivos
README	11/4/2025 16:31	Archivo de origen Mark...

```
C: > Repo-MZ > Repo2 > conflict-exercise > README.md > # conflict-exercise
1 # conflict-exercise
2 Prueba de conflictos
3 Este es un cambio en la feature branch. por Margarita
```



Comando status, git add

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (feature-branch)
$ git status
On branch feature-branch
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")

zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (feature-branch)
$ git add README.md
```

Comando git status, git commit

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (feature-branch)
$ git status
On branch feature-branch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   README.md
```

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (feature-branch)
$ git commit -m "Added a line in feature-branch-Margarita"
[feature-branch ddf0c02] Added a line in feature-branch-Margarita
1 file changed, 1 insertion(+)
```

Comando git status

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (feature-branch)
$ git status
On branch feature-branch
nothing to commit, working tree clean
```

Paso 4: Volver a la rama principal y editar el mismo archivo

- Cambia de vuelta a la rama principal (main):

`git checkout main`

- Edita el archivo README.md de nuevo, añadiendo una línea diferente: Este es un cambio en la main branch.

- Guarda los cambios y haz un commit:

`git add README.md`

`git commit -m "Added a line in main branch"`

Comando git branch

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (feature-branch)
$ git branch
* feature-branch
main
```

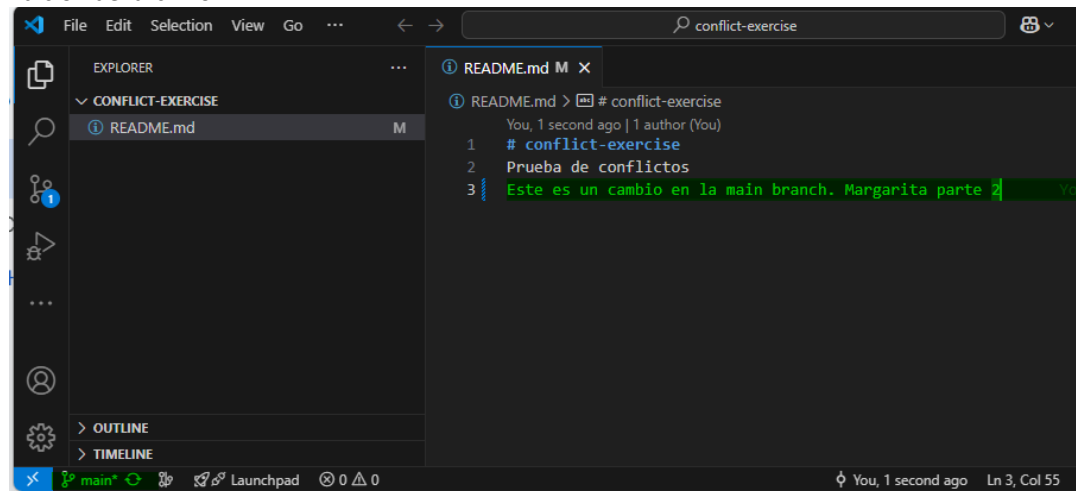
comando git checkout main

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

comando git branch, estoy posicionada en main

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (main)
$ git branch
  feature-branch
* main
```

Edición del archivo README



comando git status

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md
```

comando git add README , git status

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (main)
$ git add README.md

zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
```

comando git commit -m "Added a line main branch"

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (main)
$ git commit -m "Added a line in main-branch-Margarita"
[main d868f93] Added a line in main-branch-Margarita
1 file changed, 1 insertion(+)
```

Paso 5: Hacer un merge y generar un conflicto

- Intenta hacer un merge de la feature-branch en la rama main:

```
git merge feature-branch
```

- Se generará un conflicto porque ambos cambios afectan la misma línea del archivo README.md.

comando git branch, visualizo estoy parada en main

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (main)
$ git branch
  feature-branch
* main
```

comando git merge

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (main)
$ git merge feature-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

Paso 6: Resolver el conflicto

- Abre el archivo README.md en tu editor de texto. Verás algo similar a esto:

```
<<<<<<< HEAD
```

Este es un cambio en la main branch.

```
=====
```

Este es un cambio en la feature branch.

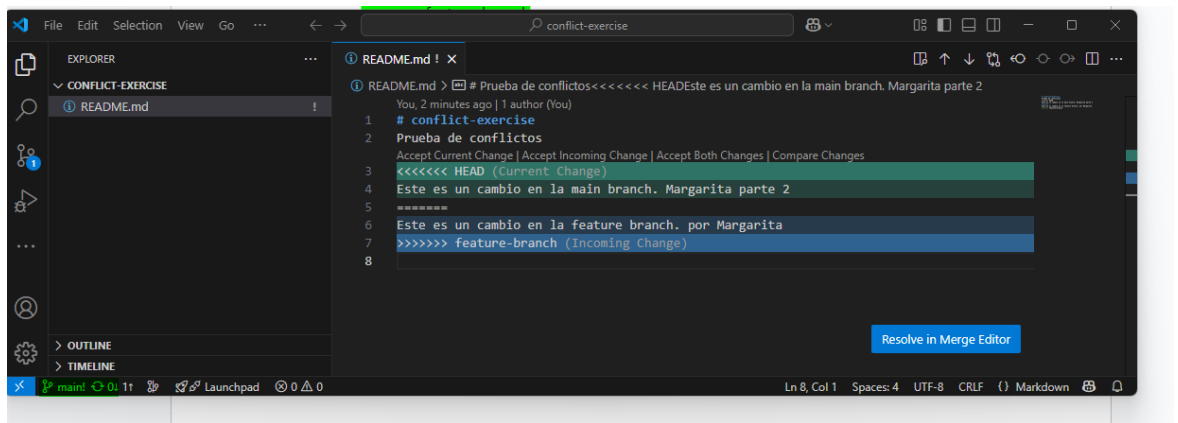
```
>>>>>>> feature-branch
```

- Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.
- Edita el archivo para resolver el conflicto y guarda los cambios (Se debe borrar lo marcado en verde en el archivo donde estes solucionando el conflicto. Y se debe borrar la parte del texto que no se quiera dejar).
- Añade el archivo resuelto y completa el merge:

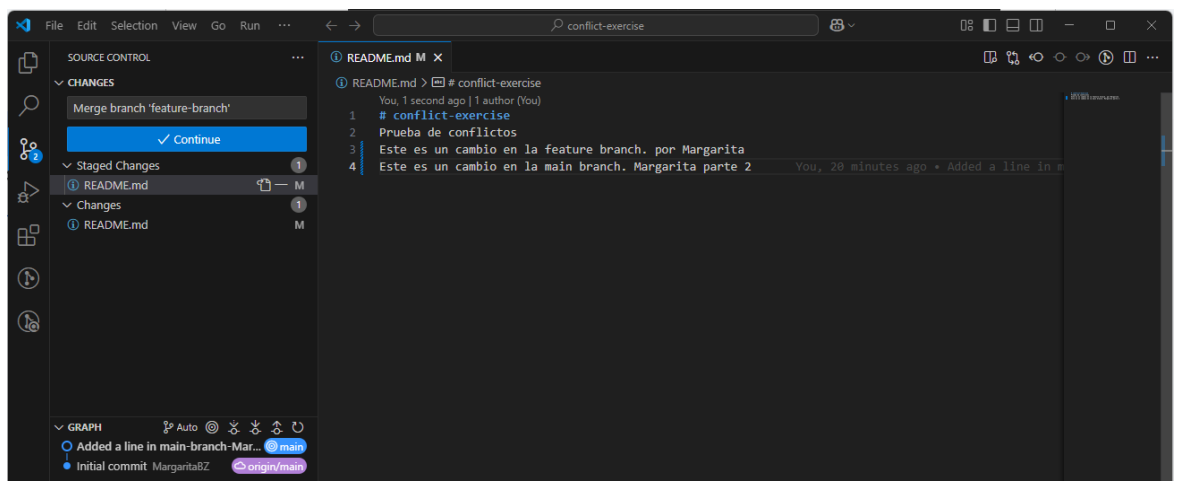
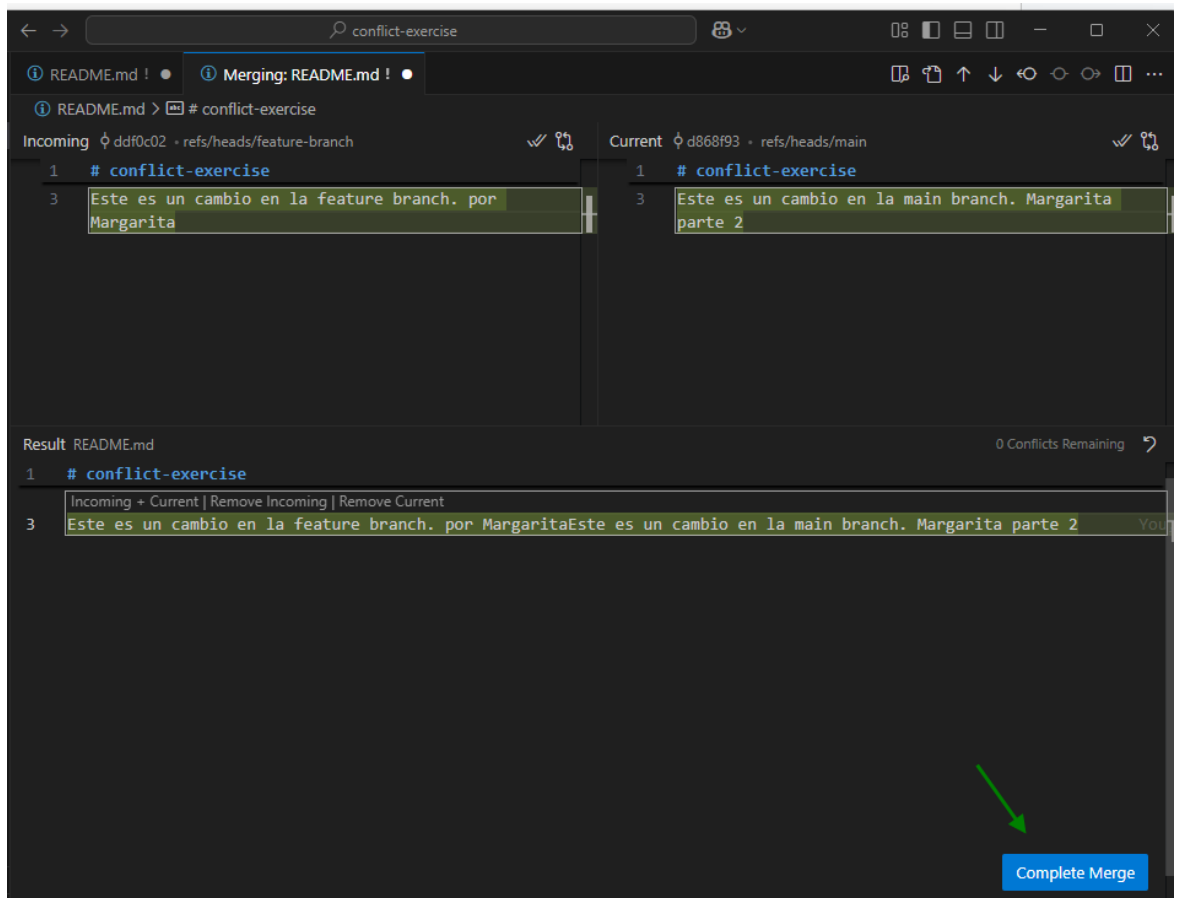
```
git add README.md
```

```
git commit -m "Resolved merge conflict"
```

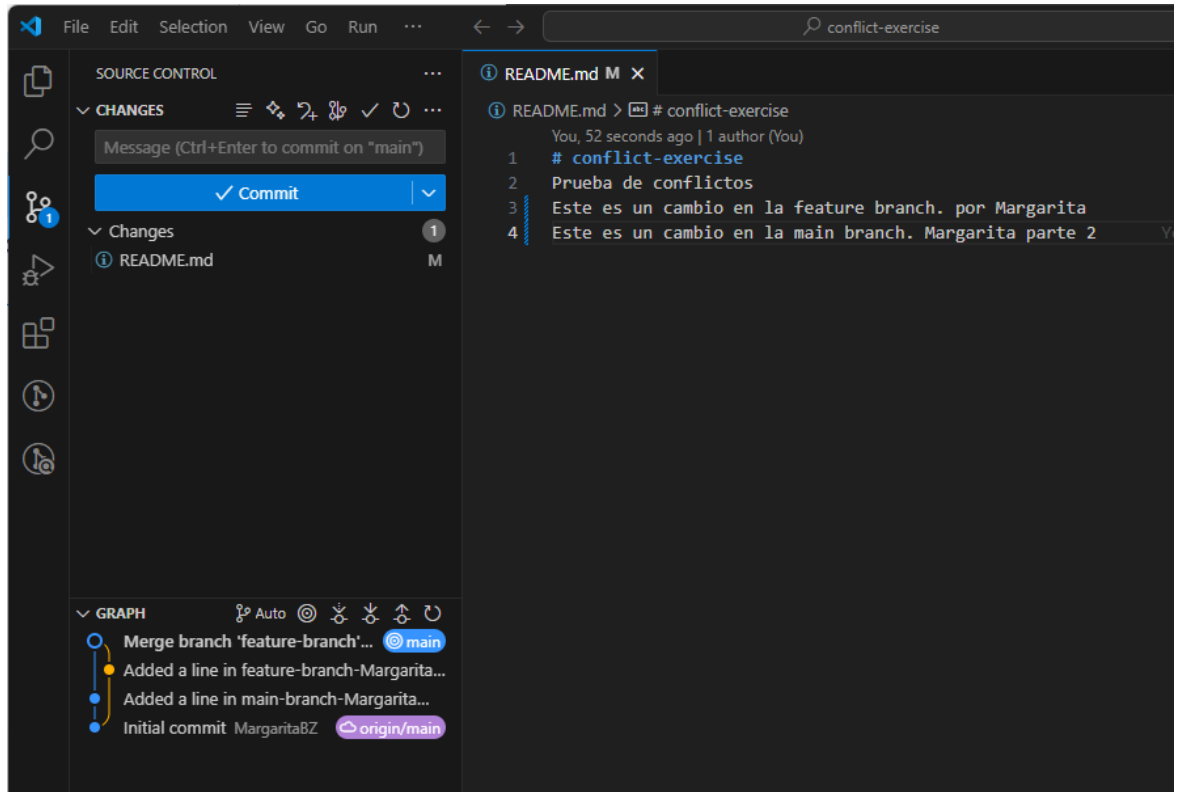
se visualiza en VSC



Soluciono mis cambios haciendo fusión de los mismos



continuar



comando git status

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md
```

comando git add

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (main)
$ git add README.md
```

comando git status

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
```

comando git commit -m "Resolviendo merge conflict"

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (main)
$ git commit -m "Resolviendo merge conflict"
[main 162396c] Resolviendo merge conflict
1 file changed, 2 insertions(+), 1 deletion(-)
```

comando git status

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 4 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

Paso 7: Subir los cambios a GitHub

- Sube los cambios de la rama main al repositorio remoto en GitHub:

**git push origin main**

- También sube la feature-branch si deseas:

**git push origin feature-branch**

Comando git push origin main

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (main)
$ git push origin main
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 8 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (12/12), 1.00 KiB | 256.00 KiB/s, done.
Total 12 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), done.
To https://github.com/MBZ7745/conflict-exercise.git
3add3b3..162396c main -> main
```

Comando git push origin feature-branch

```
zerpa@DESKTOP-237H7UN MINGW64 /c/Repo-MZ/Repo2/conflict-exercise (main)
$ git push origin feature-branch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:   https://github.com/MBZ7745/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/MBZ7745/conflict-exercise.git
* [new branch]   feature-branch -> feature-branch
```

Paso 8: Verificar en GitHub

- Ve a tu repositorio en GitHub y revisa el archivo README.md para confirmar que los cambios se han subido correctamente.
- Puedes revisar el historial de commits para ver el conflicto y su resolución.

GitHub

code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

MBZ7745 / conflict-exercise

Public

Pin

Unwatch 1

main

2 Branches

0 Tags

Go to file

Add file

Code

MBZ7745

Resolviendo merge conflict

162396c · 21 minutes ago

5 Commits

README.md

Resolviendo merge conflict

21 minutes ago

README

# conflict-exercise

Prueba de conflictos Este es un cambio en la feature branch. por Margarita Este es un cambio en la main branch. Margarita parte 2

Commits

main

All users

All time

Commits on Apr 11, 2025

Resolviendo merge conflict

MBZ7745 committed 22 minutes ago

162396c

Merge branch 'feature-branch'

MBZ7745 committed 28 minutes ago

88d4526

Added a line in main-branch-Margarita

MBZ7745 committed 48 minutes ago

d868f93

Added a line in feature-branch-Margarita

MBZ7745 committed 1 hour ago

ddf0c02

Commits on Apr 10, 2025

Initial commit

MBZ7745 authored yesterday

Verified

3add3b3

MBZ7745 / conflict-exercise

Search

code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

## Branches

New branch

Overview

Yours

Active

Stale

All

Search branches...

Default

Branch	Updated	Check status	Behind	Ahead	Pull request
main	19 minutes ago			(Default)	

Your branches

Branch	Updated	Check status	Behind	Ahead	Pull request
feature-branch	4 minutes ago		3	0	

Active branches

Branch	Updated	Check status	Behind	Ahead	Pull request
feature-branch	4 minutes ago		3	0	



