

Trabajo: Albertini, Hugo Agustin

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

Actividades

1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas):

- ¿Qué es GitHub? **Es llevar a cabo un control muy eficiente del ciclo de vida de todo proyecto además de los colaboradores que trabajan sobre el**
- ¿Cómo crear un repositorio en GitHub? **1) Inicias sesión en GitHub.**
2) Clickeas en nuevo repositorio.
3) Completas los nombres y descripción que quieras, seleccionas privado o publico.
4) Creas repositorio
- ¿Cómo crear una rama en Git? **Para crear una rama en git tenes que ingresar el comando “git branch nombre-rama “**
- ¿Cómo cambiar a una rama en Git? **Para cambiar de rama ingresar comando “git checkout nombre-rama “**
- ¿Cómo fusionar ramas en Git? **Para fusionar, tenes que posicionarte en la rama destino con el comando “git checkout” después hacer el merge con el comando “git merge nombre-rama”**
- ¿Cómo crear un commit en Git? **Para crear un commit ocupamos el comando “git commit -m “mensaje”**
- ¿Cómo enviar un commit a GitHub? **Para enviar un commit a github tenemos que ingresar el comando “git push master nombre-rama”**
- ¿Qué es un repositorio remoto? **El repositorio remoto es una versión alojada en un servidor como en este caso es GitHub, para colaboración en conjunto**
- ¿Cómo agregar un repositorio remoto a Git? **Para agregar un repositorio remoto a git utilizamos el comando “git remote add master “link del repositorio”**
- ¿Cómo empujar cambios a un repositorio remoto? **Para empujar se utiliza el comando “git push”**

- ¿Cómo tirar de cambios de un repositorio remoto? Para traer los cambios tenemos que ocupar el comando **“git pull”**
- ¿Qué es un fork de repositorio? Un fork es la copia personal de un repositorio ajeno en la cuenta del usuario que lo solicito, se utiliza para proponer cambios sin afectar el original
- ¿Cómo crear un fork de un repositorio? En GitHub, vas al repositorio original y haces clic en Fork
- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio? Para solicitar un PR vamos al repositorio original > Pull request > Nuevo Pull request
- ¿Cómo aceptar una solicitud de extracción? En GitHub, revisas los cambios y haces clic en Merge pull request para fusionarlos
- ¿Qué es un etiqueta en Git? La etiqueta marca un commit específico, por ejemplo puedes aclarar algún punto importante para tener en cuenta de ese commit
- ¿Cómo crear una etiqueta en Git? Para crear una etiqueta se utiliza el comando **“git tag”**
- ¿Cómo enviar una etiqueta a GitHub? Para enviar etiquetas a GitHub, podremos utilizar **“git push origin nombre-etiqueta”** o para enviar todas **“git push - -tags”**
- ¿Qué es un historial de Git? El historial es útil para tener un registro cronológico de todos los commits realizados en el repositorio
- ¿Cómo ver el historial de Git? Para ver el historial tendremos que ocupar **“git log”** muestra todos los commits
- ¿Cómo buscar en el historial de Git? Para buscar utilizamos **“git log -s “texto”** para buscar commits que modificaron **“texto”**
- ¿Cómo borrar el historial de Git? Podremos utilizar **“git reset - - hard <commit>”**
- ¿Qué es un repositorio privado en GitHub? El repositorio privado es visible o editable solo por el dueño y colaboradores invitados
- ¿Cómo crear un repositorio privado en GitHub? Cuando creas un repositorio en GitHub tienes la opción de marcar como PRIVADO
- ¿Cómo invitar a alguien a un repositorio privado en GitHub? Para invitar, ingresamos a GitHub > vamos a Settings > Collaborators > Add people
- ¿Qué es un repositorio público en GitHub? A diferencia del privado, el público es visible para todos, pero solo los colaboradores pueden editarlo
- ¿Cómo crear un repositorio público en GitHub? En GitHub, al crear el repositorio lo seleccionamos como PUBLICO
- ¿Cómo compartir un repositorio público en GitHub? Compartiendo el URL, otras personas pueden clonarlo o hacer fork del repositorio

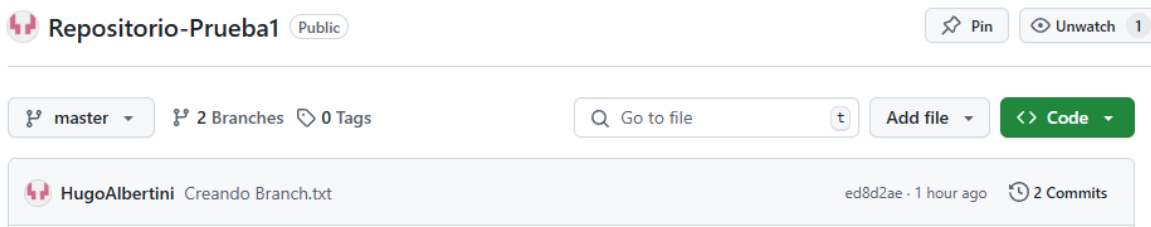
2) Realizar la siguiente actividad:

Crear un repositorio.

Dale un nombre al repositorio

Elije el repositorio sea público.

Inicializa el repositorio con un archivo.

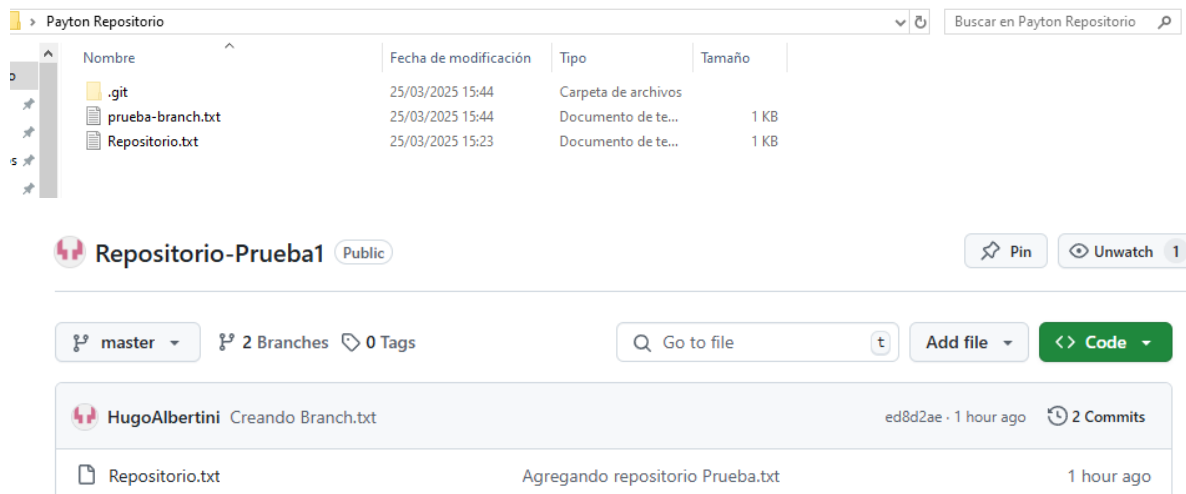


Agregando un Archivo.

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).



Creando Branchs

Crear una Branch

Realizar cambios o agregar un archivo

Subir la Branch

The screenshot shows a GitHub repository page for 'Repositorio-Prueba1' (Public). The repository has 2 branches and 0 tags. The current branch is 'master'. A commit by HugoAlbertini is shown, titled 'Creando Branch.txt', with commit hash 'ed8d2ae' and made '1 hour ago'. The commit includes two files: 'Repositorio.txt' (Agregando repositorio Prueba.txt) and 'prueba-branch.txt' (Creando Branch.txt), both committed '1 hour ago'.

3) Realizar la siguiente actividad:

Paso 1: Crear un repositorio en GitHub

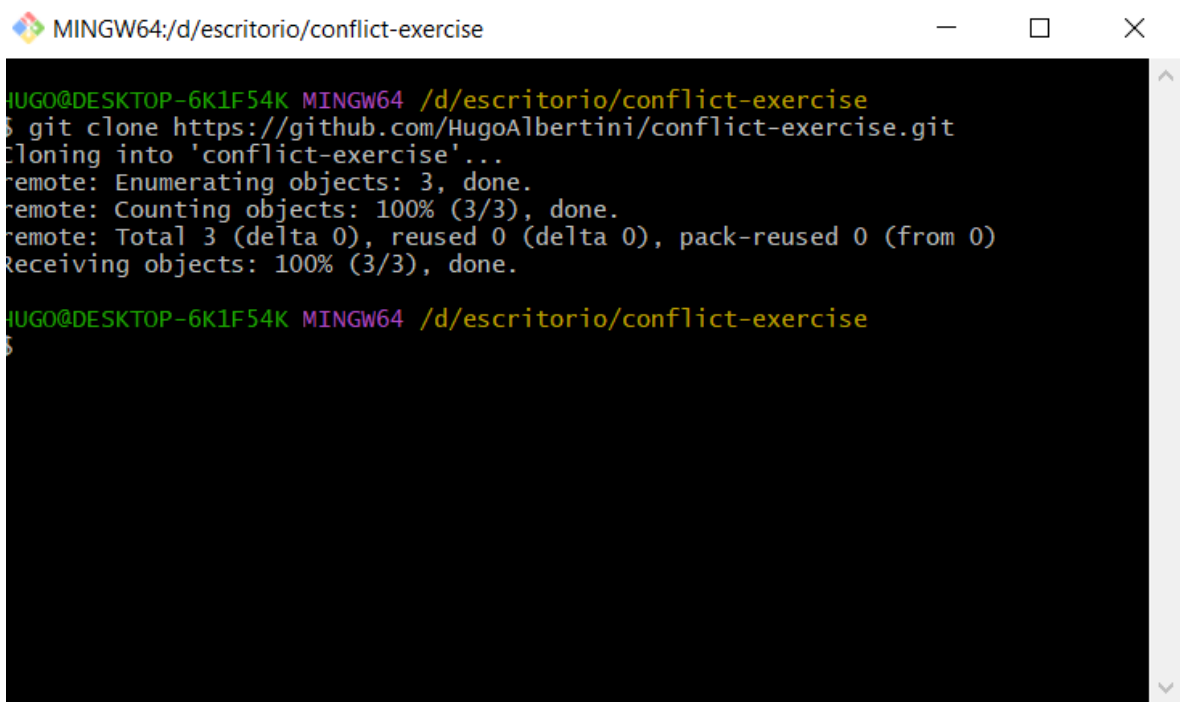
- Ve a GitHub e inicia sesión en tu cuenta.
- Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.
- Asigna un nombre al repositorio, por ejemplo, conflict-exercise.
- Opcionalmente, añade una descripción.
- Marca la opción "Initialize this repository with a README".
- Haz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

The screenshot shows a GitHub repository page for 'conflict-exercise' (Public). The repository has 1 branch and 0 tags. The current branch is 'main'. An initial commit by HugoAlbertini is shown, titled 'Initial commit', with commit hash '785f0f2' and made 'now'. The commit includes a file named 'README.md' (Initial commit) made 'now'. Below the commit, the README content is visible, showing the title 'conflict-exercise'.

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`

A screenshot of a terminal window titled "MINGW64:/d/escritorio/conflict-exercise". The terminal shows the command `git clone https://github.com/HugoAlbertini/conflict-exercise.git` being executed. The output indicates that the repository was successfully cloned, with details such as "Enumerating objects: 3, done.", "Counting objects: 100% (3/3), done.", and "Receiving objects: 100% (3/3), done.". The prompt `HUGO@DESKTOP-6K1F54K MINGW64 /d/escritorio/conflict-exercise` is visible at the bottom of the terminal.

```
MINGW64:/d/escritorio/conflict-exercise
HUGO@DESKTOP-6K1F54K MINGW64 /d/escritorio/conflict-exercise
$ git clone https://github.com/HugoAlbertini/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.
HUGO@DESKTOP-6K1F54K MINGW64 /d/escritorio/conflict-exercise
$
```

- Entra en el directorio del repositorio:

`cd conflict-exercise`

```
MINGW64:/d/escritorio/conflict-exercise/conflict-exercise
HUGO@DESKTOP-6K1F54K MINGW64 /d/escritorio/conflict-exercise
$ git clone https://github.com/HugoAlbertini/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.

HUGO@DESKTOP-6K1F54K MINGW64 /d/escritorio/conflict-exercise
$ cd conflict-exercise

HUGO@DESKTOP-6K1F54K MINGW64 /d/escritorio/conflict-exercise/conflict-exercise
(main)
$ |
```

Paso 3: Crear una nueva rama y editar un archivo

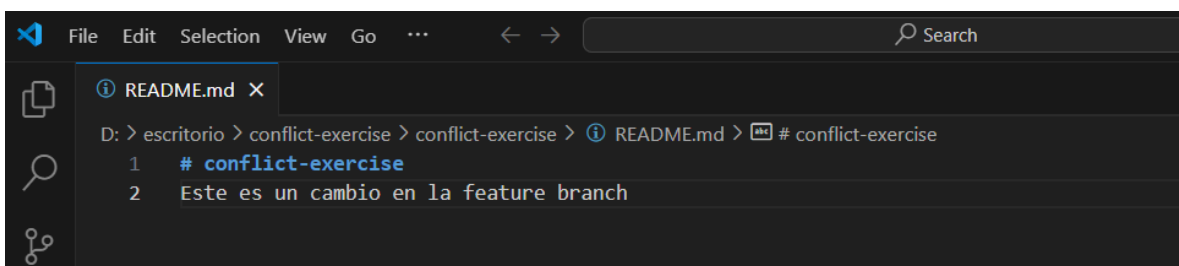
- Crea una nueva rama llamada feature-branch:

git checkout -b feature-branch

```
HUGO@DESKTOP-6K1F54K MINGW64 /d/escritorio/conflict-exercise/conflict-exercise
(main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

HUGO@DESKTOP-6K1F54K MINGW64 /d/escritorio/conflict-exercise/conflict-exercise
(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.



```
File Edit Selection View Go ... Search
D: > escritorio > conflict-exercise > conflict-exercise > README.md > # conflict-exercise
1 # conflict-exercise
2 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"

```
HUGO@DESKTOP-6K1F54K MINGW64 /d/escritorio/conflict-exercise/conflict-exercise
(feature-branch)
$ git add README.md

HUGO@DESKTOP-6K1F54K MINGW64 /d/escritorio/conflict-exercise/conflict-exercise
(feature-branch)
$ git commit -m "Added a line in feature-branch"
[feature-branch df1e1a9] Added a line in feature-branch
1 file changed, 2 insertions(+), 1 deletion(-)

HUGO@DESKTOP-6K1F54K MINGW64 /d/escritorio/conflict-exercise/conflict-exercise
(feature-branch)
$
```

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

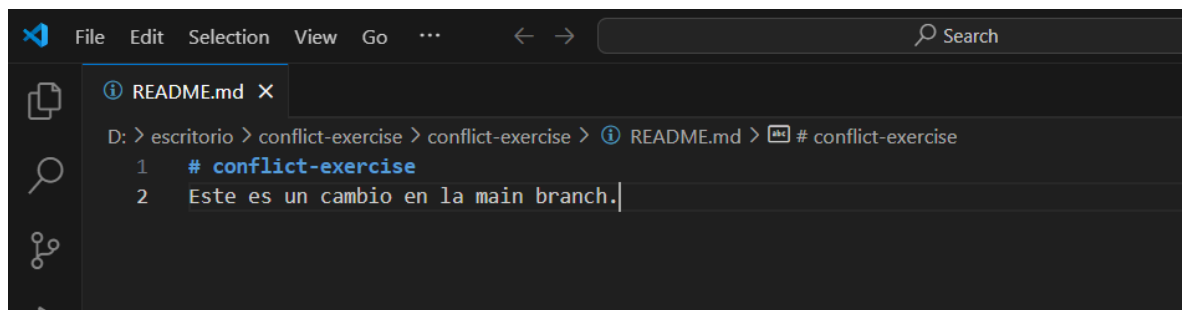
- Cambia de vuelta a la rama principal (main):

git checkout main

```
HUGO@DESKTOP-6K1F54K MINGW64 /d/escritorio/conflict-exercise/conflict-exercise
(feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

HUGO@DESKTOP-6K1F54K MINGW64 /d/escritorio/conflict-exercise/conflict-exercise
(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"

```
MINGW64:/d/escritorio/conflict-exercise/conflict-exercise
(feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

HUGO@DESKTOP-6K1F54K MINGW64 /d/escritorio/conflict-exercise/conflict-exercise
(main)
$ ^[[200~git add README.md
bash: $'\E[200~git': command not found

HUGO@DESKTOP-6K1F54K MINGW64 /d/escritorio/conflict-exercise/conflict-exercise
(main)
$ git add README.md

HUGO@DESKTOP-6K1F54K MINGW64 /d/escritorio/conflict-exercise/conflict-exercise
(main)
$ git commit -m "Added a line in main branch"
[main 7090be9] Added a line in main branch
1 file changed, 2 insertions(+), 1 deletion(-)

HUGO@DESKTOP-6K1F54K MINGW64 /d/escritorio/conflict-exercise/conflict-exercise
(main)
$
```

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**

```
HUGO@DESKTOP-6K1F54K MINGW64 /d/escritorio/conflict-exercise/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.

HUGO@DESKTOP-6K1F54K MINGW64 /d/escritorio/conflict-exercise/conflict-exercise
(main|MERGING)
$ |
```

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

Paso 6: Resolver el conflicto

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

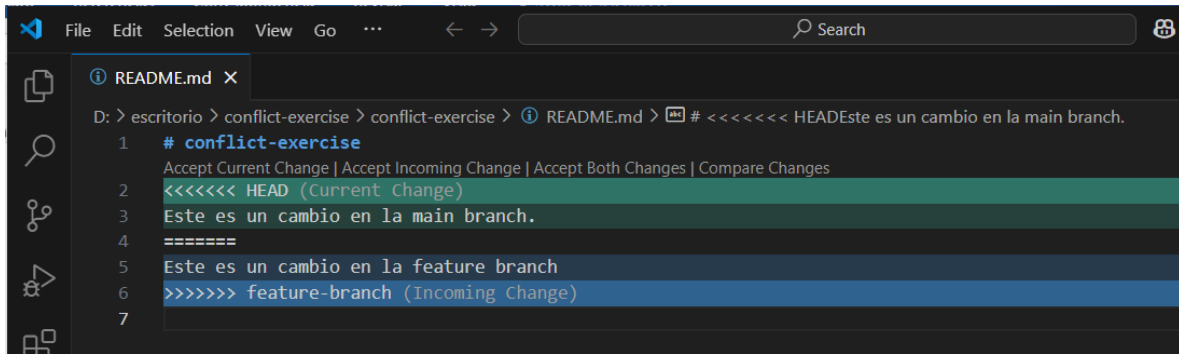
<<<<<<< HEAD

Este es un cambio en la main branch.

=====

Este es un cambio en la feature branch.

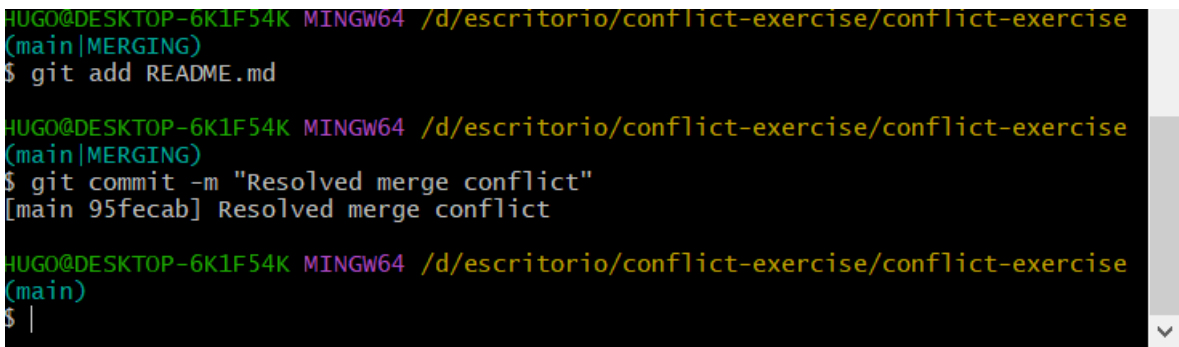
>>>>>>> feature-branch



- 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.

git add README.md

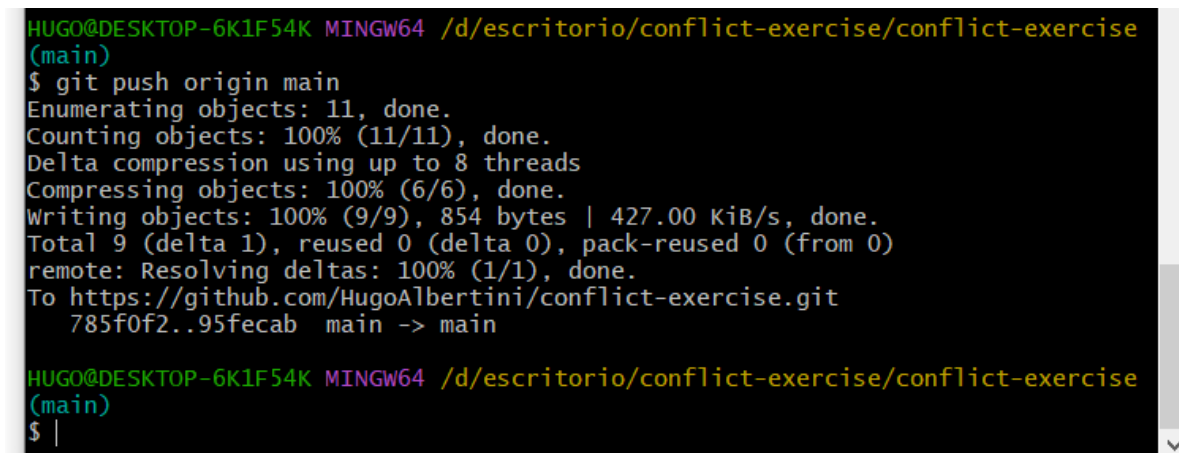
git commit -m "Resolved merge conflict"



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

```
HUGO@DESKTOP-6K1F54K MINGW64 /d/escritorio/conflict-exercise/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/HugoAlbertini/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/HugoAlbertini/conflict-exercise.git
 * [new branch]      feature-branch -> feature-branch

HUGO@DESKTOP-6K1F54K MINGW64 /d/escritorio/conflict-exercise/conflict-exercise
(main)
$ |
```

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.

The screenshot shows the GitHub interface for a repository named 'conflict-exercise' (Public). The repository has 2 branches and 0 tags. The main branch is selected. A commit by HugoAlbertini, titled 'Resolved merge conflict', is shown with a timestamp of 95fecab · 1 minute ago and 4 commits. Below the commit, a file named 'README.md' is listed with the status 'Resolved merge conflict' and a timestamp of 1 minute ago. The README content is displayed, showing a conflict resolution for the 'HEAD' branch. The text reads: '<<<<<< HEAD Este es un cambio en la main branch.' followed by 'Este es un cambio en la feature branch' and a visual representation of the merge conflict with vertical bars and the label 'feature-branch'.