

ALUMNO: MARTIN ARJONA

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?
Rta: GitHub es una plataforma en línea que permite almacenar proyectos que usan el sistema de control de versiones Git.
 - ¿Cómo crear un repositorio en GitHub?
Rta: Paso 1: Ingresar en GitHub.com
Paso 2: Click en el botón new .
Paso 3: Configurar el nuevo repositorio (nombre- descripción- readme- publico/privado)
Paso 4: Click en crear
 - ¿Cómo crear una rama en Git?
Rta: 4. Crear y cambiar a una nueva rama
1. Crear rama: *(Creamos una nueva rama para desarrollar una funcionalidad)*
git branch nueva-rama
2. Verificá las ramas disponibles:
git branch
 - ¿Cómo cambiar a una rama en Git?

Rta: Cambiar de rama: *(Nos movemos a esa nueva rama para trabajar de forma aislada)*
git checkout nueva-rama
Crear y cambiar a nueva rama:

git checkout -b nombre-rama

- ¿Cómo fusionar ramas en Git?
Rta: Fusionar con rama, en rama actual: git merge rama
- ¿Cómo crear un commit en Git?
 1. Git add . (agrega todos los archivos al stage)
 2. Git commit -m "mensaje" (crea el commit con un mensajero descriptivo)
- ¿Cómo enviar un commit a GitHub?
 1. Abrir la terminal
 2. Git add (nombre del archivo)
 3. Git commit -m "mensaje"
 4. Git push origin main o master (para enviar el comit a github)
- ¿Qué es un repositorio remoto?
Repositorio remoto: generalmente alojado en plataformas como GitHub o GitLab.
Permite sincronizar el trabajo con otros desarrolladores.
- ¿Cómo agregar un repositorio remoto a Git?
Añadir nuevo repositorio remoto : git remote add origin url
- ¿Cómo empujar cambios a un repositorio remoto?
Subir cambios al repositorio remoto: git push -u origin master o
git push (Luego de la primera vez)
- ¿Cómo tirar de cambios de un repositorio remoto?
Aplicar cambios desde repositorio remoto: git pull origin master
O git pull (Si usamos -u en el push)
- ¿Qué es un fork de repositorio?
Un fork es una copia completa de un repositorio de GitHub que se crea dentro de tu cuenta. Te permite:
 - Probar cambios sin afectar el proyecto original.
 - Proponer mejoras enviando los cambios mediante un pull request.
 - Trabajar con repositorios a los que no tenés permisos directos.
- ¿Cómo crear un fork de un repositorio?
 1. Ingresá a un repositorio público de GitHub (por ejemplo, <https://github.com/otro-usuario/proyecto>).
 2. Hacé clic en el botón Fork (esquina superior derecha).
 3. Elegí tu cuenta para crear una copia del proyecto en tu propio GitHub.

- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
 1. Haz un fork del repositorio.
 2. Clona tu fork:
`git clone https://github.com/tu_usuario/repo.git`
 3. Crea una rama nueva:
`git checkout -b mi-cambio`
 4. Haz tus cambios, guarda y commitea:
`git add .`
`git commit -m "Descripción de los cambios"`
 5. Sube tu rama:
`git push origin mi-cambio`
 6. Ve a GitHub y haz clic en "Compare & pull request".
 7. Escribe una descripción y haz clic en "Create pull request".
- ¿Cómo aceptar una solicitud de extracción?
 1. Ve al repositorio original en GitHub.
 2. Haz clic en la pestaña "Pull requests".
 3. Selecciona la pull request que quieres revisar.
 4. Revisa los cambios (archivos modificados, comentarios, etc.).
 5. Si todo está bien, haz clic en "Merge pull request".
 6. Luego, haz clic en "Confirm merge".
 7. (Opcional) Borra la rama con "Delete branch".
- ¿Qué es una etiqueta en Git?

Una etiqueta (tag) en Git es un marcador que se usa para señalar un punto específico en la historia del repositorio, normalmente para marcar versiones importantes como lanzamientos (v1.0, v2.1, etc.).
- ¿Cómo crear una etiqueta en Git?

Crear tag: `git tag nombre`

Crear tag con mensaje: `git tag -a nombre -m mensaje`

Crear tag en commit específico: `git tag -a nombre hash -m mensaje`
- ¿Cómo enviar una etiqueta a GitHub?

Subir cambios al repositorio remoto:

`git push -u origin master`

`git push` (Luego de la primera vez)
- ¿Qué es un historial de Git?

Rta: El historial de Git es el registro de todos los cambios realizados en un repositorio: commits, ramas, fusiones, etc.
- ¿Cómo ver el historial de Git?

Ver commits (q para salir): `git log`

- ¿Cómo buscar en el historial de Git?

`git log --oneline` Ver commits (una línea c/u)

`git log --decorate --all --graph --oneline` Ver commits (graficado)

- ¿Cómo borrar el historial de Git?

`git reset --modo HEAD^` Volver a commit anterior

`git reset --modo HEAD^N` Volver hacia el N° anterior commit

`git reset --modo hash-commit` Volver hacia commit específico

- ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es un repositorio cuyo acceso está restringido. Solo las personas que tienen permiso explícito pueden ver, clonar o contribuir al proyecto.

- ¿Cómo crear un repositorio privado en GitHub?

- a. Ve a GitHub y haz login en tu cuenta.
- b. Haz clic en el botón "New" (Nuevo) en la página principal de repositorios o en tu perfil.
- c. Llena el formulario:
- d. Nombre del repositorio
- e. (Opcional) Descripción
- f. En "Visibility" selecciona "Private".
- g. Haz clic en "Create repository".

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

- a. Ve a tu repositorio en GitHub.
- b. Haz clic en la pestaña "Settings" (Configuración).
- c. En el menú de la izquierda, selecciona "Collaborators" (Colaboradores).
- d. En el campo "Search by username, full name or email", ingresa el nombre de usuario o correo de la persona que quieres invitar.
- e. Haz clic en "Add collaborator".
- f. La persona recibirá una invitación y podrá acceder al repositorio privado.

- ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un repositorio cuyo código y archivos son accesibles para cualquier persona en internet. Cualquier usuario puede ver, clonar, bifurcar (fork) y contribuir al proyecto.

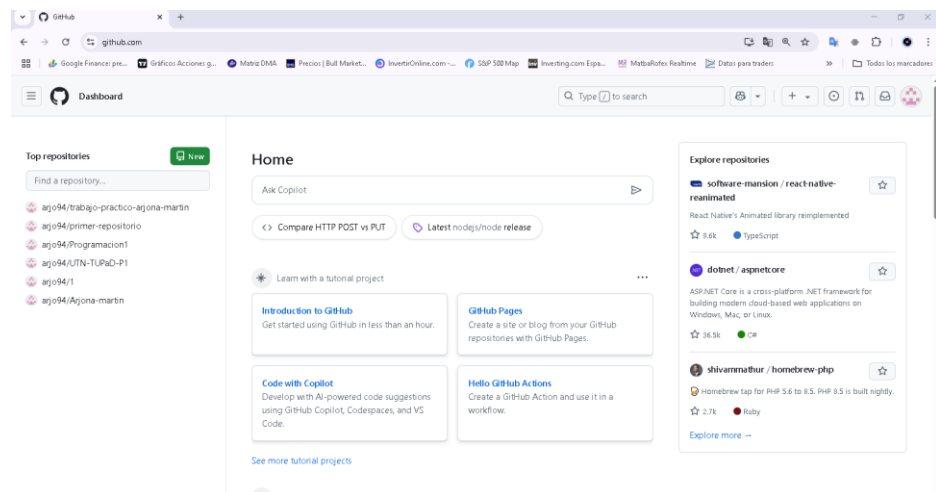
- ¿Cómo crear un repositorio público en GitHub?

1. Inicia sesión en tu cuenta de GitHub.
2. En la página principal de tu perfil, haz clic en "New" (Nuevo) para crear un repositorio.
3. Llena el formulario:
4. Nombre del repositorio.
5. (Opcional) Descripción.
6. En "Visibility", selecciona "Public".
7. Haz clic en "Create repository".

- ¿Cómo compartir un repositorio público en GitHub?
 1. Ve a tu repositorio en GitHub.
 2. Copia la URL en la barra de direcciones de tu navegador (por ejemplo: https://github.com/tu_usuario/nombre_del_repositorio).
 3. Comparte ese enlace con quien quieras, ya sea por correo electrónico, redes sociales o mensajes.
- 2) Realizar la siguiente actividad:

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

PASO 1: INGRESAR EN NEW (BOTON VERDE EN GITHUB)



PASO 2: AGREGAR NOMBRE AL REPOSITORIO. ELEGIR SI SERÁ PUBLICO O PRIVADO Y SELECCIONAR EL ADD A README file

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 *

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

☐ **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

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

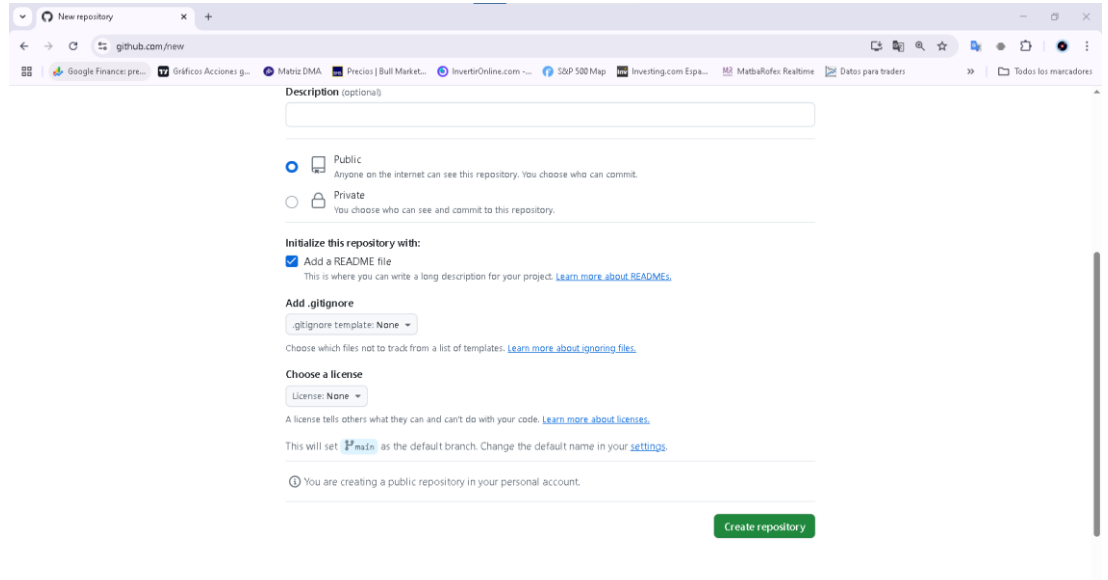
Choose a license

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

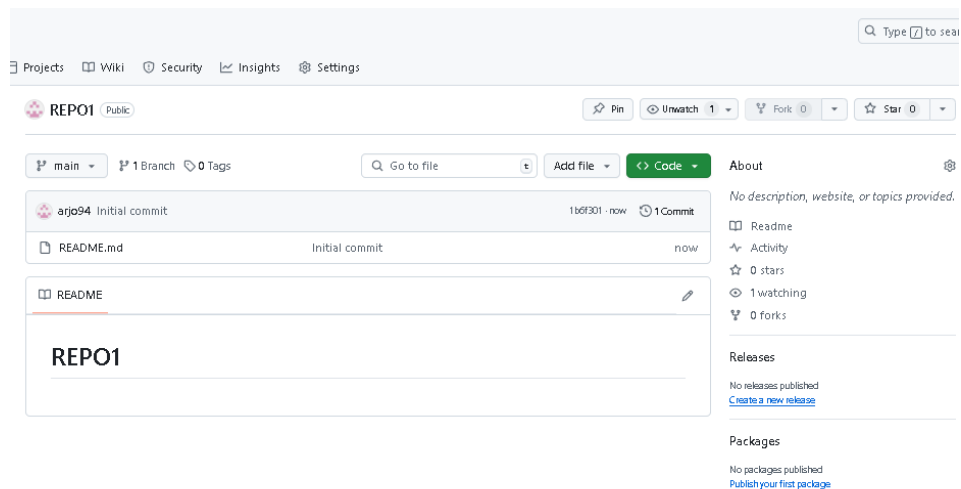
This will set `main` as the default branch. Change the default name in your [settings](#).

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

PASO 3: PRESIONAR EL BOTON VERDE CREATE REPOSITORY PARA FINALIZAR LA CREACION DEL REPOSITORIO



PASO 4: UNA VEZ CREADO EL REPOSITORIO IR A CODE BOTON VERDE Y COPIAR EL link PARA EL SIGUIENTE PASO



PASO 5: ABRIR LA TERMINAL Y CLONAR EN ELD ISCO LOCAL EL REPOSITORIO

```
MINGW64~/c/Users/PC/Desktop/PROYECTOS/REP01
PC@PC-PC MINGW64 ~/Desktop/PROYECTOS
$ git clone https://github.com/arjo94/REP01.git
Cloning into 'REP01'...
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.

PC@PC-PC MINGW64 ~/Desktop/PROYECTOS
$ cd REP01

PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/REP01 (main)
$
```

- 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).

EN LA CARPETA CREAR UN ARCHIVO DE BLOC DE NOTAS

ste equipo > Escritorio > PROYECTOS > REP01

Nombre	Fecha de modificación	Tipo	Tamaño
.git	25/04/2025 20:05	Carpeta de archivos	
mi-archivo.txt	25/04/2025 20:11	Documento de te...	0 KB
README	25/04/2025 20:05	Archivo de origen ...	1 KB

EN LA TERMINAL: comandos git add . y git commit -m "Agregando mi-archivo.txt"

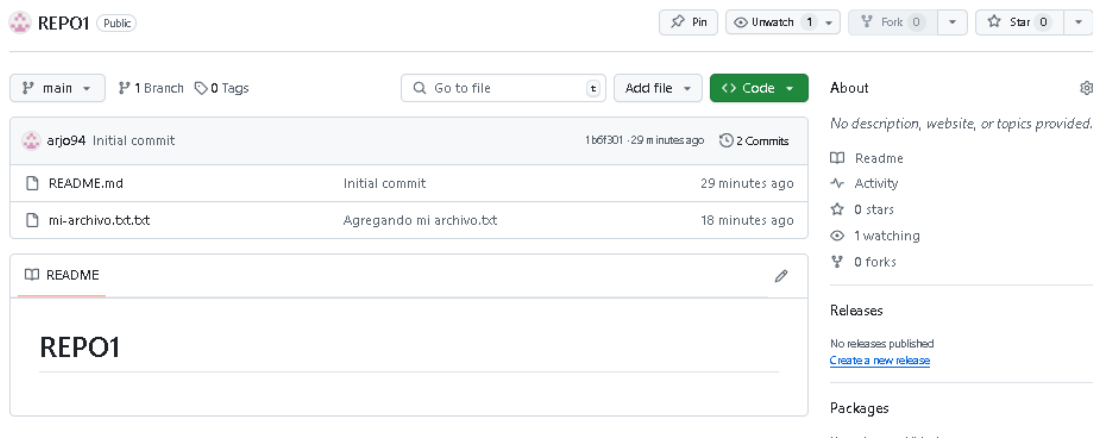
```
PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/REP01 (main)
$ git add .

PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/REP01 (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)
    new file:   mi-archivo.txt.txt

PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/REP01 (main)
$ git commit -m "Agregando mi archivo.txt"
[main ff9c4f1] Agregando mi archivo.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 mi-archivo.txt.txt
```

Con git push origin main se sube los cambios al repositorio en github.



- Creando Branchs

- Crear una Branch

```
PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/REPO1 (main)
$ git checkout pepe1
Switched to branch 'pepe1'

PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/REPO1 (pepe1)
$ git branch
  main
* pepe1

PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/REPO1 (pepe1)
$ ...

PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/REPO1 (main)
$ git branch
* main
  pepe1

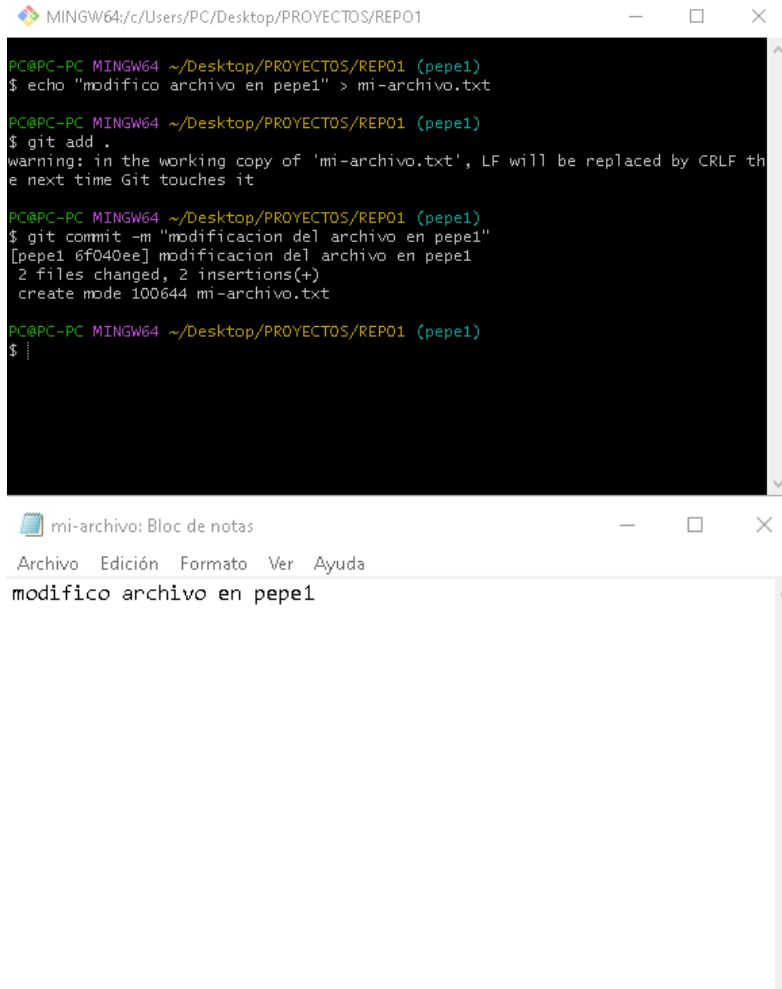
PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/REPO1 (main)
$ ...

MINGW64:/c:/Users/PC/Desktop/PROYECTOS/REPO1

PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/REPO1 (main)
$ git branch
* main

PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/REPO1 (main)
$ git branch pepe1
```


- Realizar cambios o agregar un archivo



The screenshot shows a terminal window titled 'MINGW64/c/Users/PC/Desktop/PROYECTOS/REP01' and a text editor window titled 'mi-archivo: Bloc de notas'. The terminal shows the following commands and output:

```
PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/REP01 (pepe1)
$ echo "modifico archivo en pepe1" > mi-archivo.txt

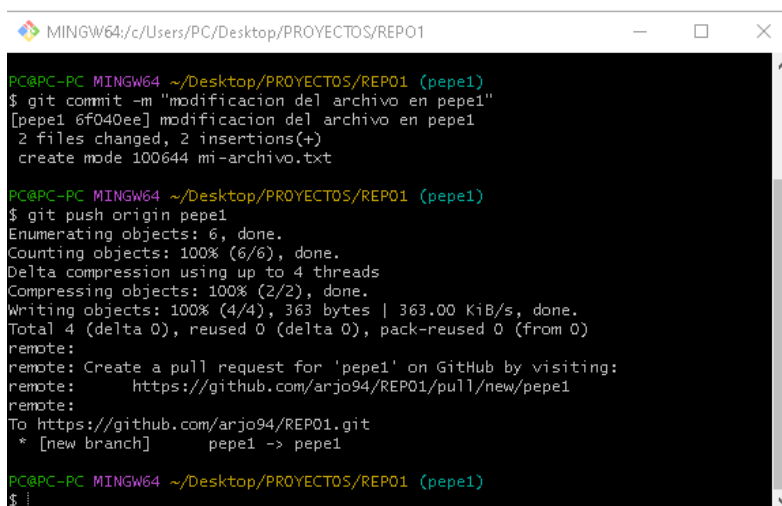
PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/REP01 (pepe1)
$ git add .
warning: in the working copy of 'mi-archivo.txt', LF will be replaced by CRLF th
e next time Git touches it

PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/REP01 (pepe1)
$ git commit -m "modificacion del archivo en pepe1"
[pepe1 6f040ee] modificacion del archivo en pepe1
2 files changed, 2 insertions(+)
create mode 100644 mi-archivo.txt

PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/REP01 (pepe1)
$
```

The text editor shows the content of 'mi-archivo.txt' as 'modifico archivo en pepe1'.

- Subir la Branch

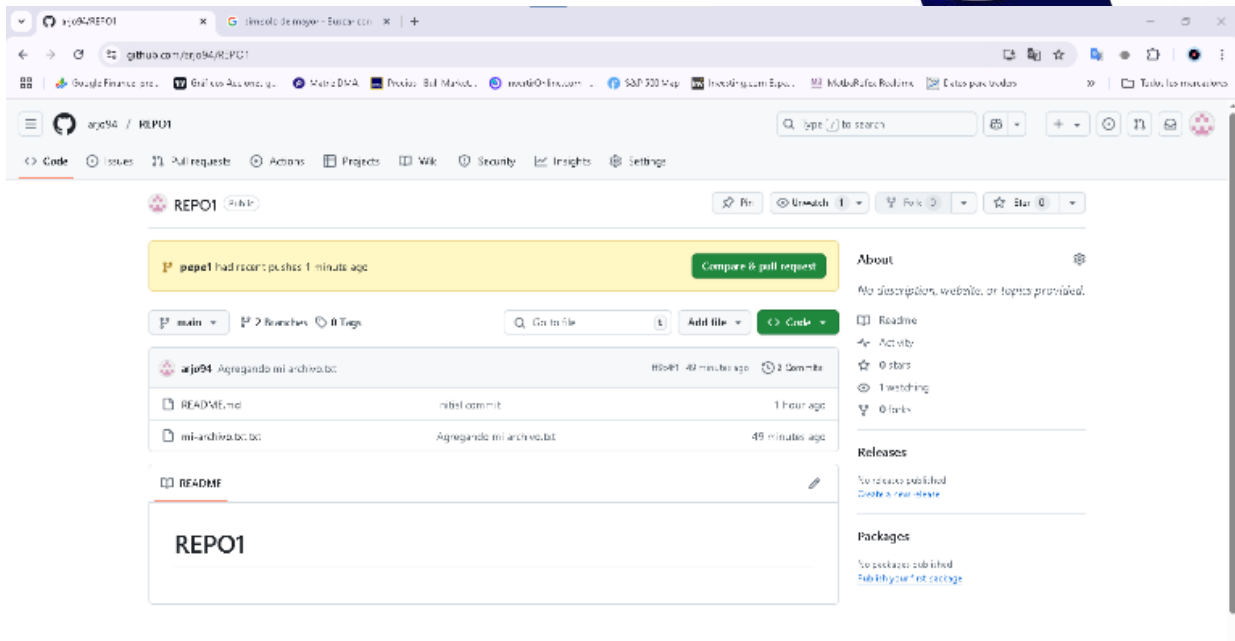


The screenshot shows a terminal window titled 'MINGW64/c/Users/PC/Desktop/PROYECTOS/REP01' with the following commands and output:

```
PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/REP01 (pepe1)
$ git commit -m "modificacion del archivo en pepe1"
[pepe1 6f040ee] modificacion del archivo en pepe1
2 files changed, 2 insertions(+)
create mode 100644 mi-archivo.txt

PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/REP01 (pepe1)
$ git push origin pepe1
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 363 bytes | 363.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'pepe1' on GitHub by visiting:
remote:   https://github.com/arjo94/REP01/pull/new/pepe1
remote:
To https://github.com/arjo94/REP01.git
 * [new branch]      pepe1 -> pepe1

PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/REP01 (pepe1)
$
```

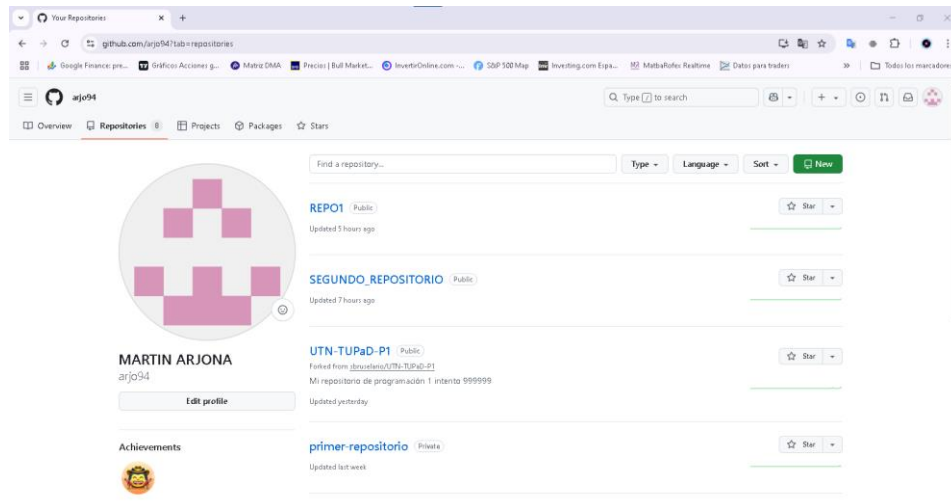


The screenshot shows a web browser displaying a GitHub repository page for 'arj094 / REPO1'. The repository is public and has 0 stars and 0 forks. A yellow banner at the top indicates that 'pepet' had recent pushes 1 minute ago, with a 'Compare & pull request' button. Below the banner, the repository's main branch is 'main', and there are 2 branches and 0 tags. A list of recent commits is shown, including 'Agregando mi archivo.txt' by 'arj094' 49 minutes ago, 'README.txt' by 'nibel commit' 1 hour ago, and 'mi-archivo.txt' by 'Agregando mi archivo.txt' 49 minutes ago. The 'README' file is selected, showing the text 'REPO1'. The right sidebar contains sections for 'About', 'Releases', and 'Packages', all of which are currently empty.

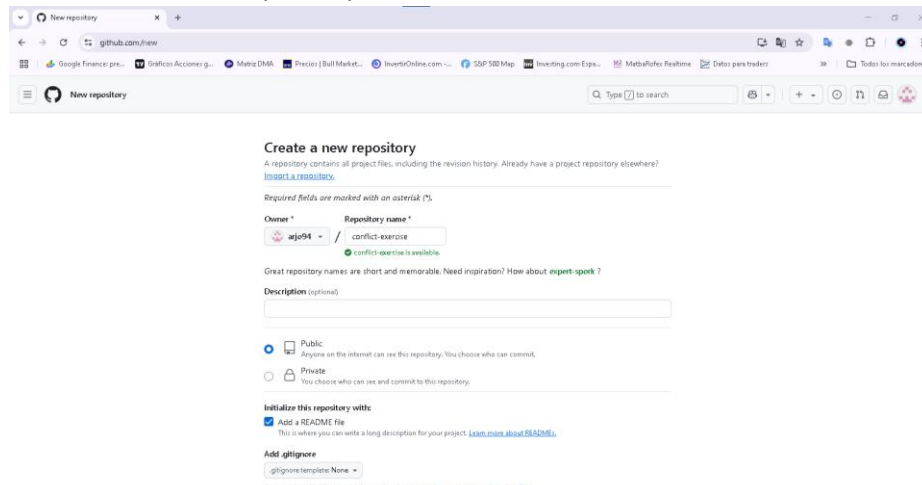
3) Realizar la siguiente actividad:

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.

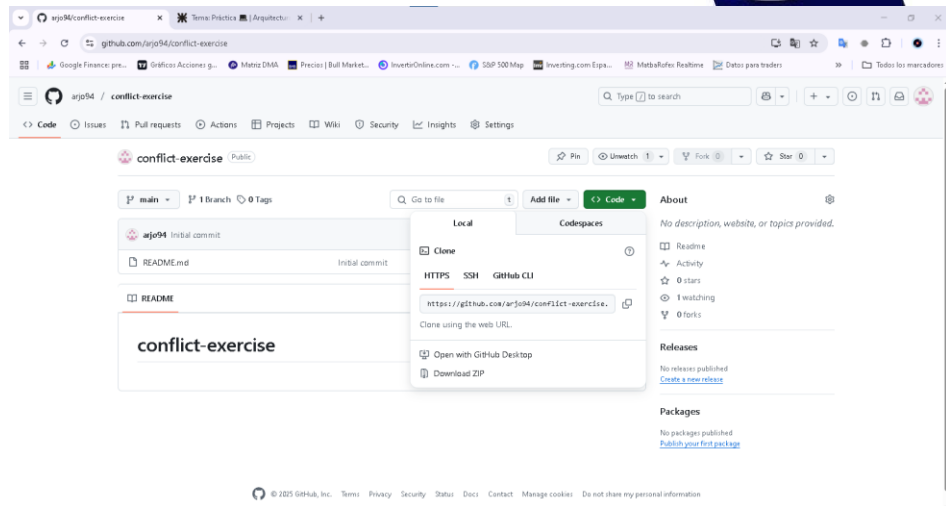


- Asigna un nombre al repositorio, por ejemplo, conflict-exercise.
- Opcionalmente, añade una descripción.
- Marca la opción "Initialize this repository with a README".
- Haz clic en "Create repository".



Paso 2: Clonar el repositorio a tu máquina local

- Copia la URL del repositorio (usualmente algo como <https://github.com/tuusuario/conflict-exercise.git>).

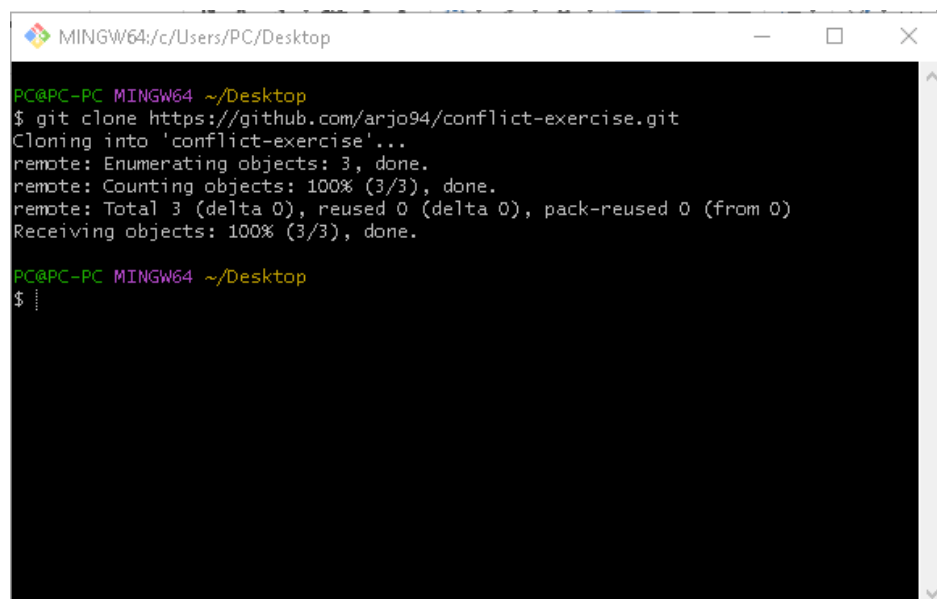


- Abre la terminal o línea de comandos en tu máquina.



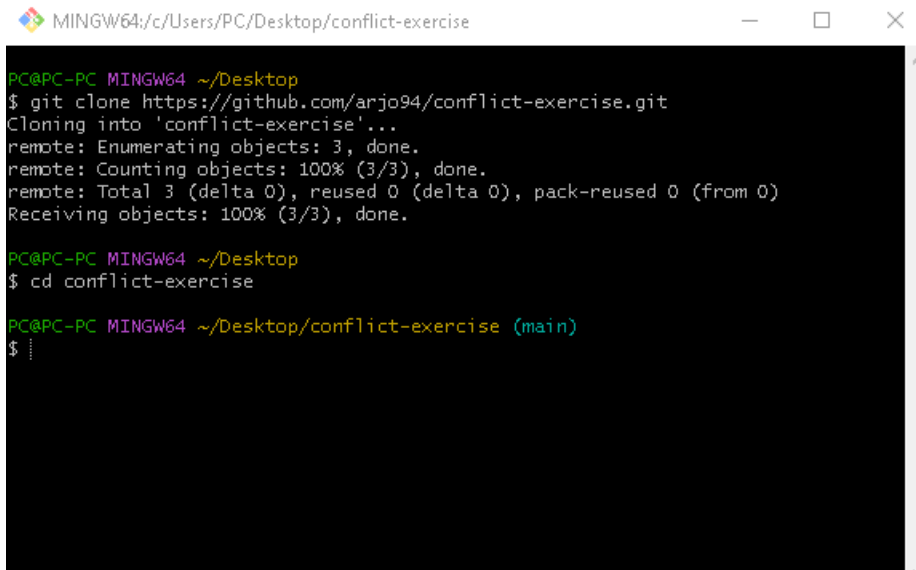
- Clona el repositorio usando el comando:

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



- Entra en el directorio del repositorio:

`cd conflict-exercise`



```
MINGW64/c:/Users/PC/Desktop/conflict-exercise

PC@PC-PC MINGW64 ~/Desktop
$ git clone https://github.com/arjo94/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.

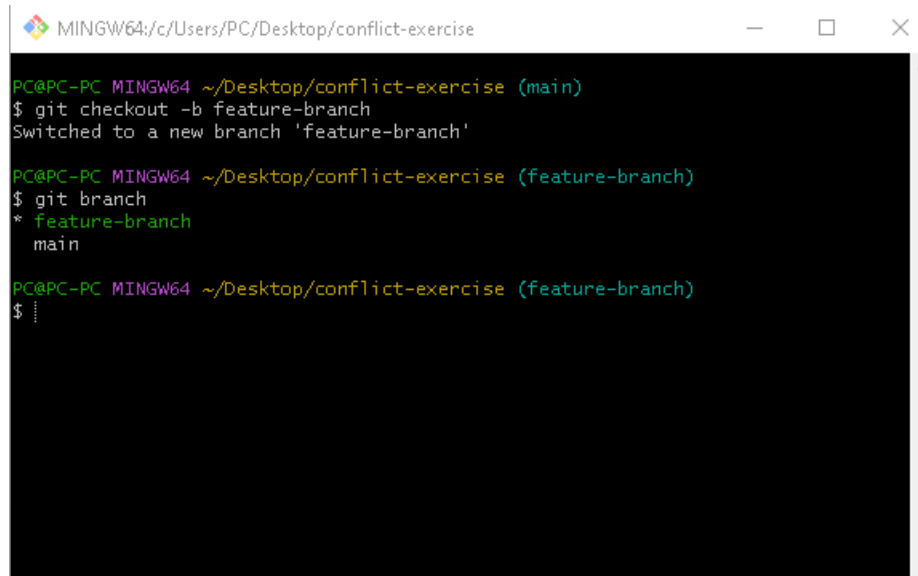
PC@PC-PC MINGW64 ~/Desktop
$ cd conflict-exercise

PC@PC-PC MINGW64 ~/Desktop/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`



```
MINGW64/c:/Users/PC/Desktop/conflict-exercise

PC@PC-PC MINGW64 ~/Desktop/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

PC@PC-PC MINGW64 ~/Desktop/conflict-exercise (feature-branch)
$ git branch
* feature-branch
  main

PC@PC-PC MINGW64 ~/Desktop/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.

- Guarda los cambios y haz un commit:

`git add README.md`

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

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\PC\Desktop\PROYECTOS\conflict-exercise> git add README.md
PS C:\Users\PC\Desktop\PROYECTOS\conflict-exercise> git commit -m "Added a line in feature-branch"
[feature-branch 81616f4] Added a line in feature-branch
1 file changed, 2 insertions(+), 1 deletion(-)
PS C:\Users\PC\Desktop\PROYECTOS\conflict-exercise> █
```

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

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

git checkout main

```
MINGW64/c:/Users/PC/Desktop/conflict-exercise
PC@PC-PC MINGW64 ~/Desktop/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PC@PC-PC MINGW64 ~/Desktop/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"

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\PC\Desktop\PROYECTOS\conflict-exercise> git add README.md
PS C:\Users\PC\Desktop\PROYECTOS\conflict-exercise> git commit -m "Added a line in main branch"
[main 2b13728] Added a line in main branch
1 file changed, 2 insertions(+), 1 deletion(-)
PS C:\Users\PC\Desktop\PROYECTOS\conflict-exercise> █
```

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.

```
MINGW64:/c:/Users/PC/Desktop/PROYECTOS/conflict-exercise

PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/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.

PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/conflict-exercise (main|MERGING)
$
```

Paso 6: Resolver el conflicto

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

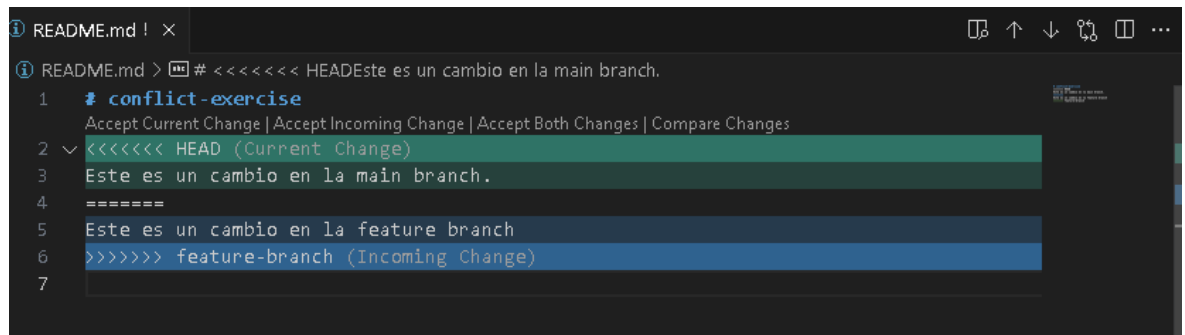
```
<<<<<<< HEAD
```

Este es un cambio en la main branch.

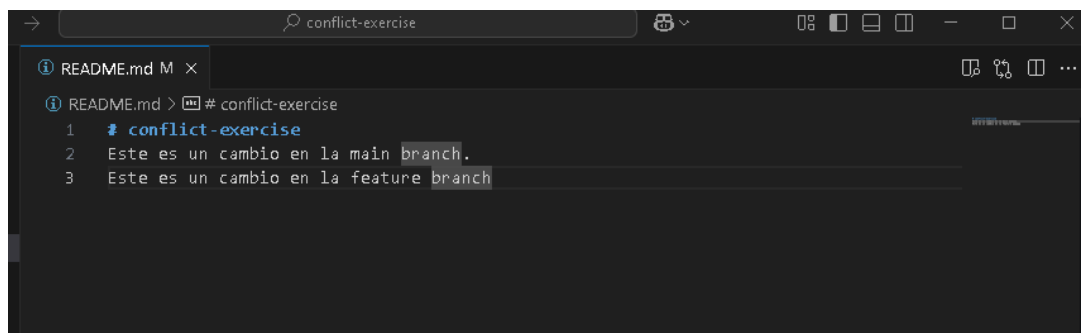
```
=====
```

Este es un cambio en la feature branch.

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



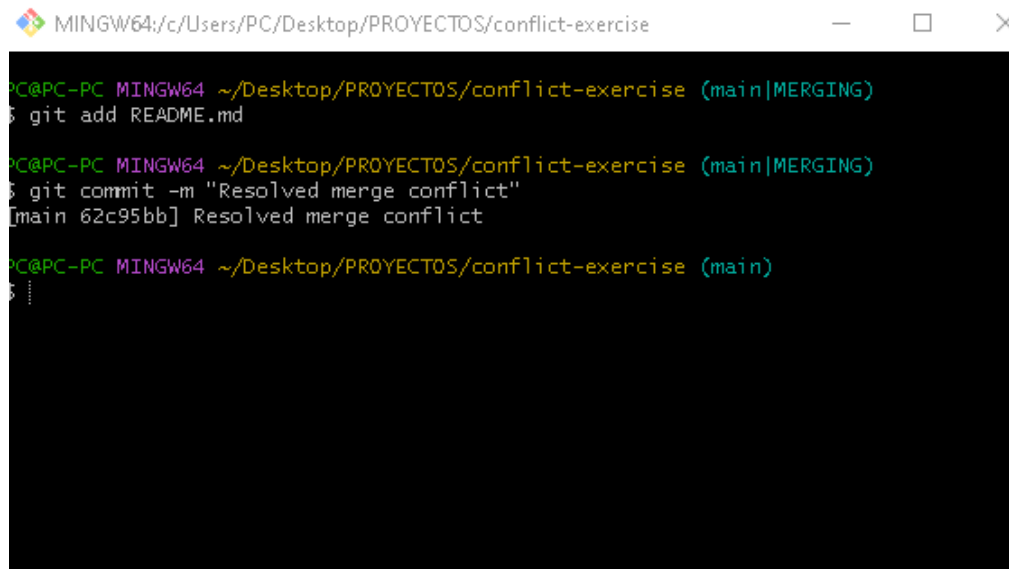
- Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.



- 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"
```

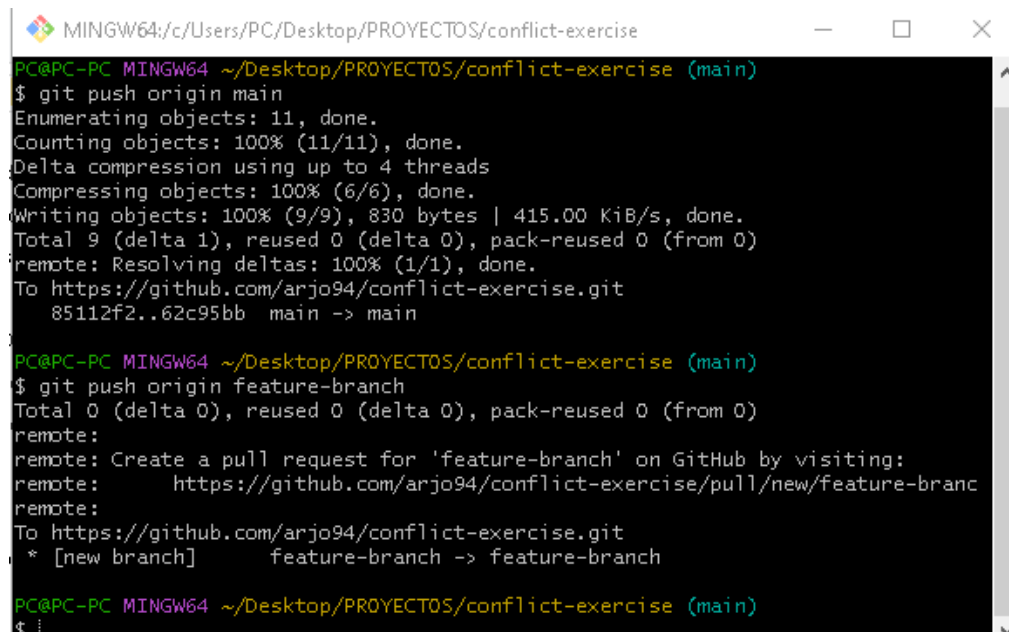


```
MINGW64:/c:/Users/PC/Desktop/PROYECTOS/conflict-exercise  
PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/conflict-exercise (main|MERGING)  
$ git add README.md  
  
PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/conflict-exercise (main|MERGING)  
$ git commit -m "Resolved merge conflict"  
[main 62c95bb] Resolved merge conflict  
  
PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/conflict-exercise (main)  
$
```

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:

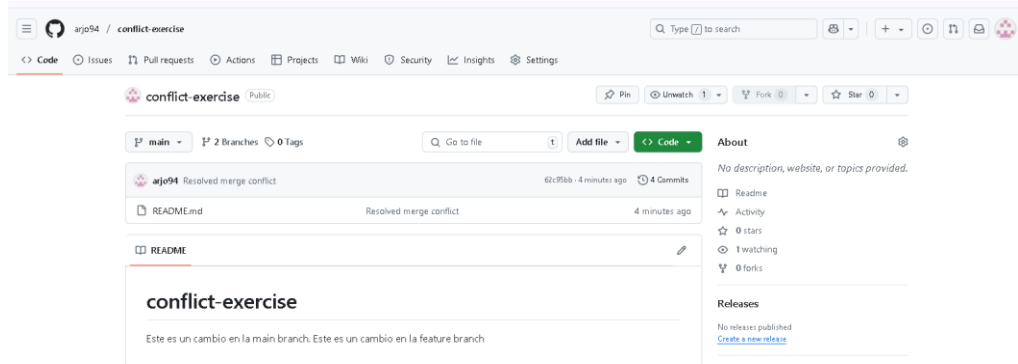
```
git push origin feature-branch
```



```
MINGW64:/c:/Users/PC/Desktop/PROYECTOS/conflict-exercise  
PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/conflict-exercise (main)  
$ git push origin main  
Enumerating objects: 11, done.  
Counting objects: 100% (11/11), done.  
Delta compression using up to 4 threads  
Compressing objects: 100% (6/6), done.  
Writing objects: 100% (9/9), 830 bytes | 415.00 KiB/s, done.  
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)  
remote: Resolving deltas: 100% (1/1), done.  
To https://github.com/arjo94/conflict-exercise.git  
85112f2..62c95bb main -> main  
  
PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/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/arjo94/conflict-exercise/pull/new/feature-branc  
remote:  
To https://github.com/arjo94/conflict-exercise.git  
* [new branch] feature-branch -> feature-branch  
  
PC@PC-PC MINGW64 ~/Desktop/PROYECTOS/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.



- Puedes revisar el historial de commits para ver el conflicto y su resolución.

