

## 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?  
GitHub es una plataforma de alojamiento de repositorios Git en la nube, usada para control de versiones, colaboración en proyectos y despliegue de software.  
Es como una “nube para programadores”, permite guardar todas las versiones del proyecto y trabajar en equipo de forma mas ordenada
- ¿Cómo crear un repositorio en GitHub?

Ve a [github.com](https://github.com) y haz clic en "New".

Asigna un nombre, descripción y elige visibilidad (público/privado).

Opcional: Añade un .gitignore o licencia.

Haz clic en "Create repository".

- ¿Cómo crear una rama en Git?  
`git branch nombre-rama` # Crea la rama  
`git checkout nombre-rama` # Cambia a la rama  
# O en un solo paso:  
`git checkout -b nombre-rama` # Crea y cambia a la rama

- ¿Cómo cambiar a una rama en Git?

`git checkout nombre-rama`

- ¿Cómo fusionar ramas en Git?

Cambia a la rama destino (ej: main)

`git checkout main`

Fusiona la rama:

`git merge nombre-rama`

- ¿Cómo crear un commit en Git?

Añade los cambios al staging área

`git add .`

Crea el commit: `git commit -m "Mensaje descriptivo"`

- ¿Cómo enviar un commit a GitHub?

`git push origin nombre-rama`

- ¿Qué es un repositorio remoto?

Es una copia del repositorio alojada en un servidor (como GitHub, GitLab o Bitbucket), para colaboración y respaldo.

- ¿Cómo agregar un repositorio remoto a Git?

`git remote add origin https://github.com/usuario/repo.git`

se usa el commando `git remote add`

- ¿Cómo empujar cambios a un repositorio remoto?

`git push -u origin nombre-rama` # Primera vez

`git push` # Sigüientes veces

- ¿Cómo tirar de cambios de un repositorio remoto?

`git pull origin nombre-rama`

- ¿Qué es un fork de repositorio?

Es una copia personal de un repositorio público en tu cuenta de GitHub, para modificarlo sin afectar el original.

- ¿Cómo crear un fork de un repositorio?

Ve al repositorio en GitHub.

Haz clic en "Fork" (esquina superior derecha).

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

Haz cambios en tu fork y haz push.

En GitHub, ve a "Pull requests" > "New pull request".

Compara tu rama con la original y envía la PR.

- ¿Cómo aceptar una solicitud de extracción?

Ve a la PR en GitHub.

Revisa los cambios y haz clic en "Merge pull request".

- ¿Qué es un etiqueta en Git?

Es un marcador para versiones específicas (ej: v1.0.0).

- ¿Cómo crear una etiqueta en Git?

```
git tag -a v1.0.0 -m "Versión 1.0.0"
```

- ¿Cómo enviar una etiqueta a GitHub?

```
git push --tags
```

- ¿Qué es un historial de Git?

Registro de todos los commits realizados en el repositorio.

- ¿Cómo ver el historial de Git?

```
git log    # Historial completo
```

```
git log --oneline # Versión resumida
```

- ¿Cómo buscar en el historial de Git?

```
git log --grep="texto" # Busca en mensajes de commit
```

- ¿Cómo borrar el historial de Git?

```
git checkout --orphan nueva-rama
```

```
git add -A
```

```
git commit -m "Nuevo inicio"
```

```
git branch -D main    # Borra la rama main
```

```
git branch -m main    # Renombra la rama actual a main
```

```
git push -f origin main # Fuerza el push (sobrescribe)
```

- ¿Qué es un repositorio privado en GitHub?

Solo visible/editable por usuarios con permisos. Requiere suscripción en planes gratuitos (excepto GitHub Free para equipos).

- ¿Cómo crear un repositorio privado en GitHub?

Al crear el repositorio, selecciona "Private" en lugar de "Public".

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

Ve a Settings > Collaborators.

Añade el nombre de usuario o email del colaborador.

- ¿Qué es un repositorio público en GitHub?

Visible para todos, pero editable solo por colaboradores.

- ¿Cómo crear un repositorio público en GitHub?

Igual que el privado, pero seleccionando "Public".

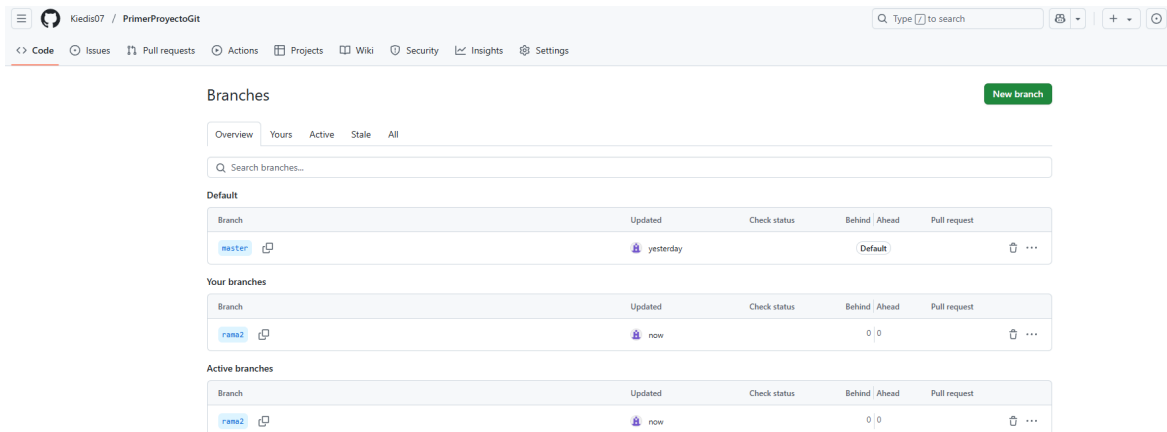
- ¿Cómo compartir un repositorio público en GitHub?

Comparte el enlace de GitHub: <https://github.com/usuario/repo>

## 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



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

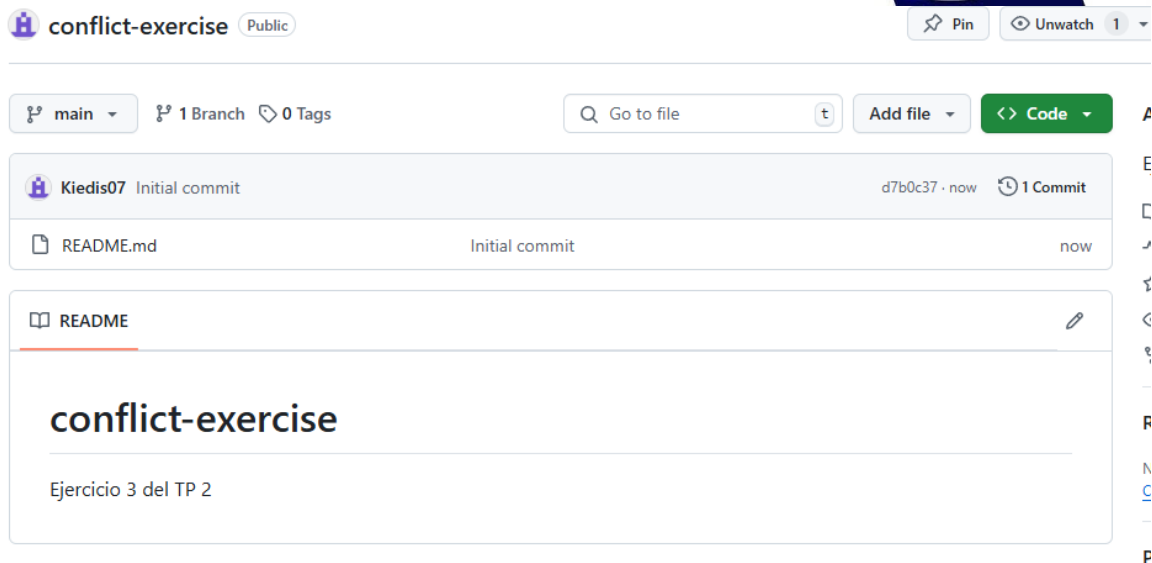
PS C:\Users\Fede\Desktop\git init> git branch rama2
PS C:\Users\Fede\Desktop\git init> git add .
PS C:\Users\Fede\Desktop\git init> git commit -m " se agrego una rama2"
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
PS C:\Users\Fede\Desktop\git init> git push origin rama2
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'rama2' on GitHub by visiting:
remote:   https://github.com/Kiedis07/PrimerProyectoGit/pull/new/rama2
remote:
To https://github.com/Kiedis07/PrimerProyectoGit.git
 * [new branch]      rama2 -> rama2
PS C:\Users\Fede\Desktop\git init> 
```

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".



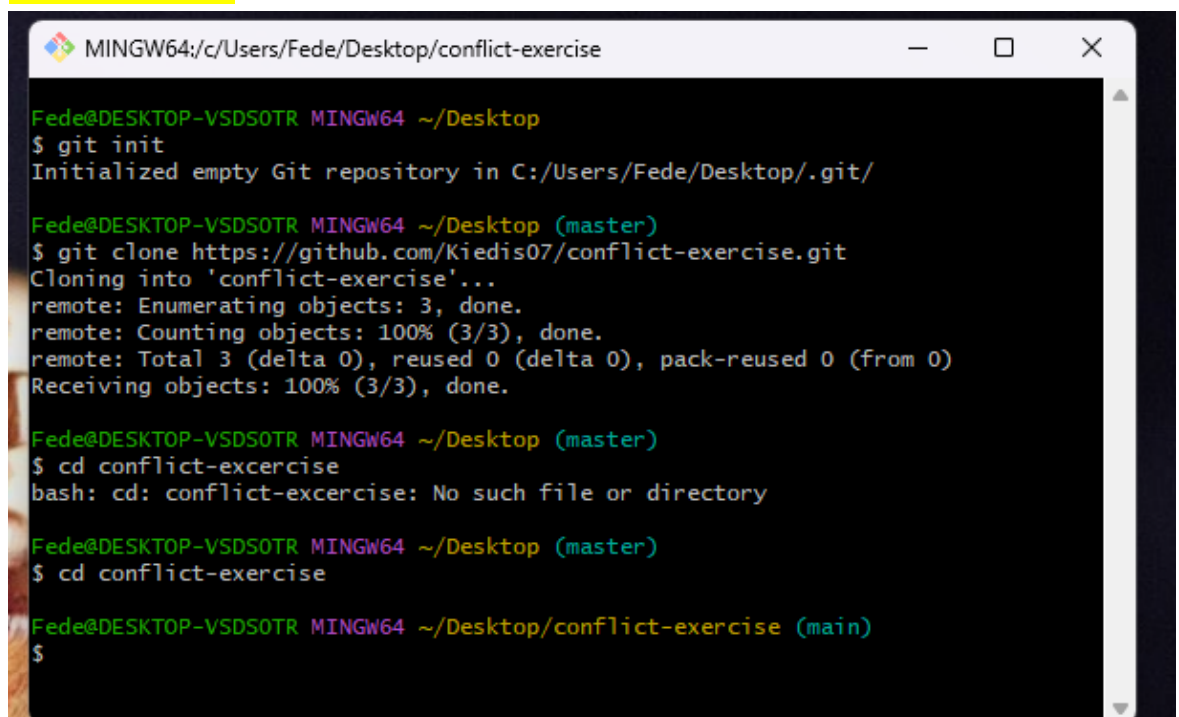
Paso 2: Clonar el repositorio a tu máquina local

- 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/Fede/Desktop/conflict-exercise

Fede@DESKTOP-VSDS0TR MINGW64 ~/Desktop
$ git init
Initialized empty Git repository in C:/Users/Fede/Desktop/.git/

Fede@DESKTOP-VSDS0TR MINGW64 ~/Desktop (master)
$ git clone https://github.com/Kiedis07/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.

Fede@DESKTOP-VSDS0TR MINGW64 ~/Desktop (master)
$ cd conflict-exercise
bash: cd: conflict-exercise: No such file or directory

Fede@DESKTOP-VSDS0TR MINGW64 ~/Desktop (master)
$ cd conflict-exercise

Fede@DESKTOP-VSDS0TR 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
```

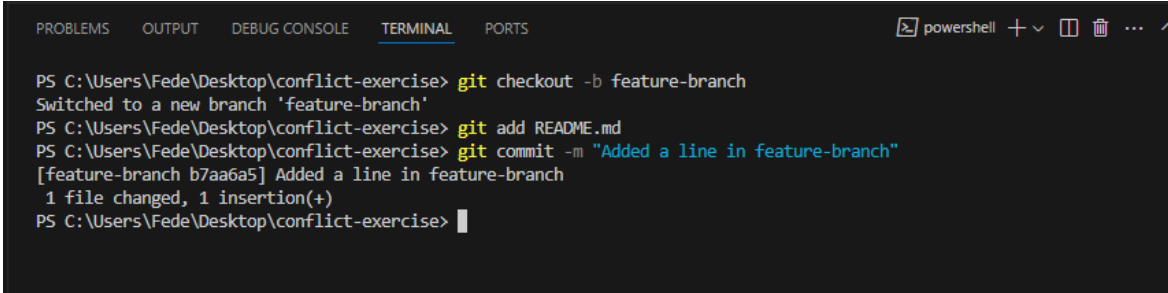
- Abre el archivo README.md en un editor de texto y añade una línea nueva, por ejemplo:

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"
```



```
PS C:\Users\Fede\Desktop\conflict-exercise> git checkout -b feature-branch
Switched to a new branch 'feature-branch'
PS C:\Users\Fede\Desktop\conflict-exercise> git add README.md
PS C:\Users\Fede\Desktop\conflict-exercise> git commit -m "Added a line in feature-branch"
[feature-branch b7aa6a5] Added a line in feature-branch
1 file changed, 1 insertion(+)
PS C:\Users\Fede\Desktop\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`

```
PS C:\Users\Fede\Desktop\conflict-exercise> git checkout -b feature-branch
Switched to a new branch 'feature-branch'
PS C:\Users\Fede\Desktop\conflict-exercise> git add README.md
PS C:\Users\Fede\Desktop\conflict-exercise> git commit -m "Added a line in feature-branch"
[feature-branch b7aa6a5] Added a line in feature-branch
 1 file changed, 1 insertion(+)
PS C:\Users\Fede\Desktop\conflict-exercise> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\Fede\Desktop\conflict-exercise> |
```

- 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"`

```
PS C:\Users\Fede\Desktop\conflict-exercise> git checkout -b feature-branch
Switched to a new branch 'feature-branch'
PS C:\Users\Fede\Desktop\conflict-exercise> git add README.md
PS C:\Users\Fede\Desktop\conflict-exercise> git commit -m "Added a line in feature-branch"
[feature-branch b7aa6a5] Added a line in feature-branch
 1 file changed, 1 insertion(+)
PS C:\Users\Fede\Desktop\conflict-exercise> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\Fede\Desktop\conflict-exercise> git add README.md
PS C:\Users\Fede\Desktop\conflict-exercise> git commit -m "Added a line in main branch"
[main 423bcab] Added a line in main branch
 1 file changed, 1 insertion(+)
PS C:\Users\Fede\Desktop\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.

Paso 6: Resolver el conflicto

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

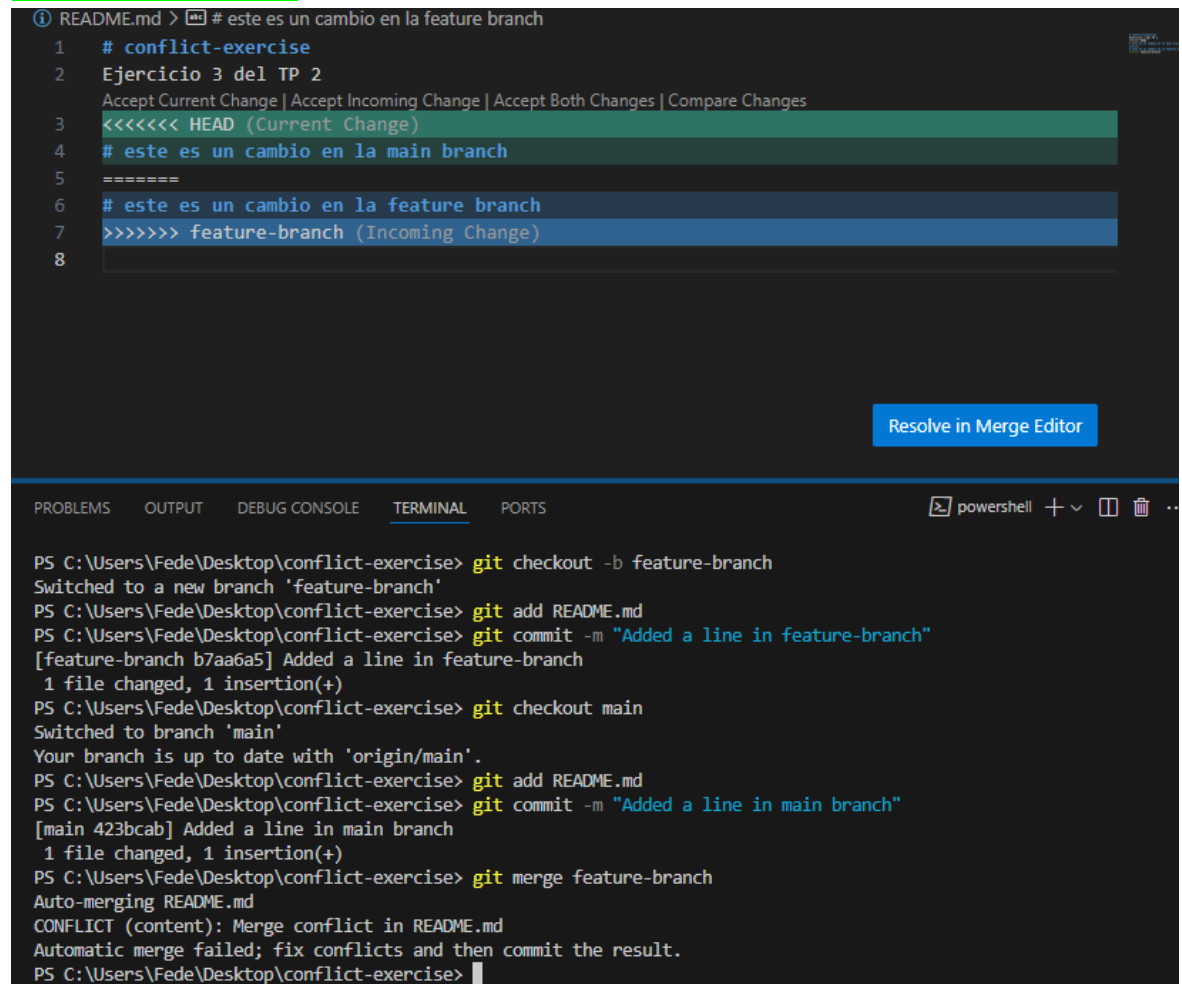
`<<<<<< HEAD`

Este es un cambio en la main branch.

```
=====
```

Este es un cambio en la feature branch.

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



```
① README.md > # este es un cambio en la feature branch
1 # conflict-exercise
2 Ejercicio 3 del TP 2
  Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
3 <<<<<< HEAD (Current Change)
4 # este es un cambio en la main branch
5 =====
6 # este es un cambio en la feature branch
7 >>>>>> feature-branch (Incoming Change)
8
```

Resolve in Merge Editor

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell + -

```
PS C:\Users\Fede\Desktop\conflict-exercise> git checkout -b feature-branch
Switched to a new branch 'feature-branch'
PS C:\Users\Fede\Desktop\conflict-exercise> git add README.md
PS C:\Users\Fede\Desktop\conflict-exercise> git commit -m "Added a line in feature-branch"
[feature-branch b7aa6a5] Added a line in feature-branch
 1 file changed, 1 insertion(+)
PS C:\Users\Fede\Desktop\conflict-exercise> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\Fede\Desktop\conflict-exercise> git add README.md
PS C:\Users\Fede\Desktop\conflict-exercise> git commit -m "Added a line in main branch"
[main 423bcab] Added a line in main branch
 1 file changed, 1 insertion(+)
PS C:\Users\Fede\Desktop\conflict-exercise> 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.
PS C:\Users\Fede\Desktop\conflict-exercise>
```

- 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"
```

```
PS C:\Users\Fede\Desktop\conflict-exercise> git add README.md
PS C:\Users\Fede\Desktop\conflict-exercise> git commit -m "Added a line in main branch"
[main 423bcab] Added a line in main branch
1 file changed, 1 insertion(+)
PS C:\Users\Fede\Desktop\conflict-exercise> 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.
PS C:\Users\Fede\Desktop\conflict-exercise> git add README.md
PS C:\Users\Fede\Desktop\conflict-exercise>
>> git commit -m "Resolved merge conflict"
>> git commit -m "Resolved merge conflict"
[main a411b1f] Resolved merge conflict
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
PS C:\Users\Fede\Desktop\conflict-exercise>
```

Paso 7: Subir los cambios a GitHub

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

git push origin main

```
>> git commit -m "Resolved merge conflict"
>> git commit -m "Resolved merge conflict"
[main a411b1f] Resolved merge conflict
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
PS C:\Users\Fede\Desktop\conflict-exercise> git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 772 bytes | 154.00 KiB/s, done.
Total 9 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/Kiedis07/conflict-exercise.git
d7b0c37..a411b1f main -> main
PS C:\Users\Fede\Desktop\conflict-exercise>
```

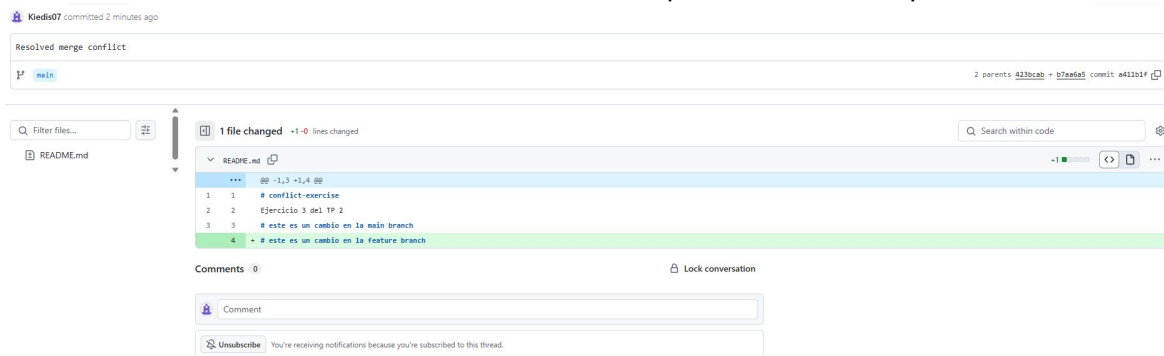
- También sube la feature-branch si deseas:

`git push origin feature-branch`

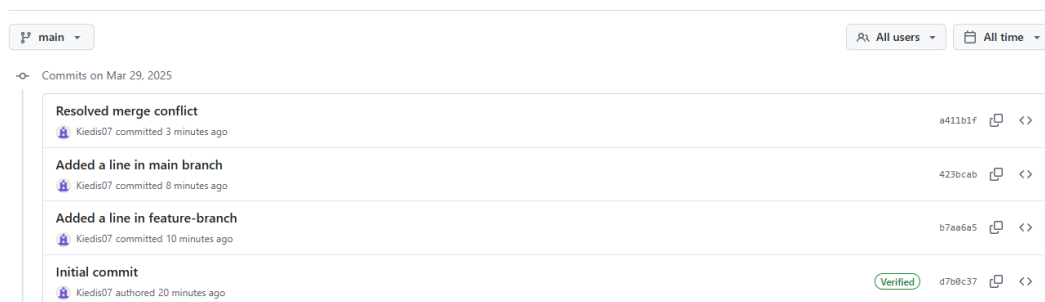
```
PS C:\Users\Fede\Desktop\conflict-exercise> git push origin feature-branch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:   https://github.com/Kiedis07/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/Kiedis07/conflict-exercise.git
 * [new branch]      feature-branch -> feature-branch
PS C:\Users\Fede\Desktop\conflict-exercise>
```

Paso 8: Verificar en GitHub

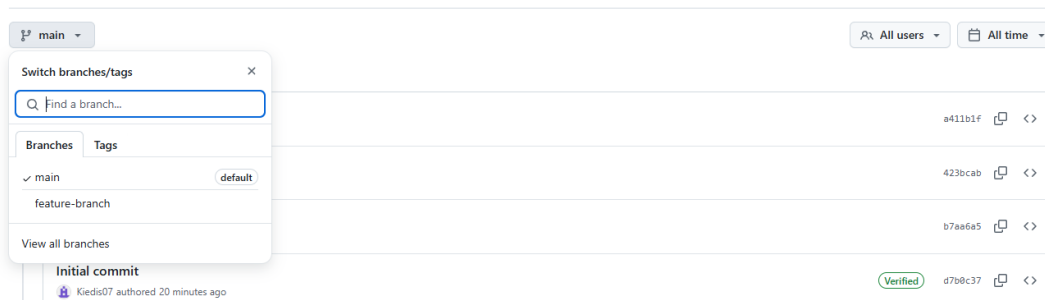
- Ve a tu repositorio en GitHub y revisa el archivo README.md para confirmar que los cambios se han subido correctamente.
- Puedes revisar el historial de commits para ver el conflicto y su resolución.



## Commits



## Commits



LINK

<https://github.com/Kiedis07/conflict-exercise>