

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.

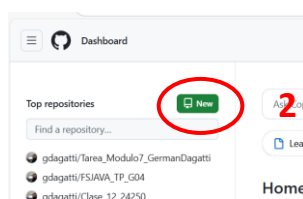
Resultados de aprendizaje:

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?
 - En GitHub, es una plataforma de alojamiento de proyectos. Los proyectos se organizan en **repositorios**, que pueden ser públicos o privados. Los desarrolladores pueden crear **ramas** (branches) para trabajar en nuevas características o correcciones de errores sin afectar el código principal, y luego fusionar esos cambios
- ¿Cómo crear un repositorio en GitHub?
 1. Ve a GitHub e inicia sesión en tu cuenta.
 2. Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.



3. Asigna un nombre al repositorio.
4. Puedes agregar una descripción del repositorio.
5. Marca la opción "Initialize this repository with a README".
6. 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 (*).

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner * / Repository name * **3**

gdagatti /

Great repository names are short and memorable. Need inspiration? How about [refactored-rotary-phone](#) ?

Description (optional) **4**

☒ 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.](#)

5 

 **6**

Create repository

- ¿Cómo crear una rama en Git?
git branch nombre_rama
- ¿Cómo cambiar a una rama en Git?
git checkout rama_nueva
- ¿Cómo fusionar ramas en Git?
git merge rama_a_fusionar
- ¿Cómo crear un commit en Git?

git commit -m "mensaje"

- ¿Cómo enviar un commit a GitHub?

git add .

git commit -m "mensaje_cambios"

git push -u origin main/master (La primera vez)

git push

- ¿Qué es un repositorio remoto?

Un proyecto en la nube, en donde pueden participar de manera colaborativa mas de una persona a la vez.

- ¿Cómo agregar un repositorio remoto a Git?

1. Abre la terminal o línea de comandos en tu máquina.
2. Navega al directorio de tu proyecto local usando el comando cd:
3. **cd ruta/al/proyecto**
4. Añade el repositorio remoto usando el comando git remote add seguido del nombre del repositorio (por ejemplo, origin) y la URL del repositorio remoto:
5. **git remote add origin**
<https://github.com/usuario/nombre-del-repositorio.git>
6. Verifica que el repositorio remoto se haya añadido correctamente usando el comando:
git remote -v

- ¿Cómo empujar cambios a un repositorio remoto?

git push origin main

- ¿Cómo tirar de cambios de un repositorio remoto?

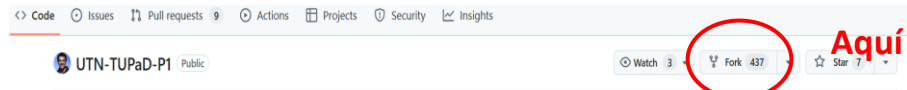
git pull origin main

- ¿Qué es un fork de repositorio?

Es una bifurcación en la línea temporal de un repositorio. Esto es una copia de un repositorio existente que se crea en tu propia cuenta de GitHub. La cual puede ser modificada íntegramente sin alterar el repositorio original.

- ¿Cómo crear un fork de un repositorio?

En el repositorio que se quiera duplicar, diríjase a la parte superior izquierda de la pantalla. Y haga click en Fork

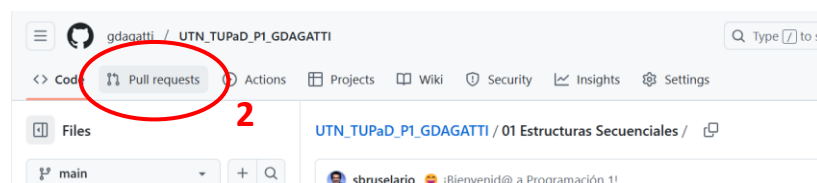


Puedes modificar el nombre del repositorio y la descripción, a fin de personalizarlo.

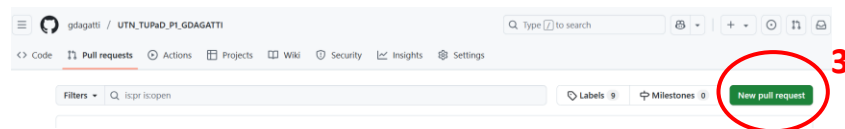
Por último, hacer click en "Crear Fork".

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

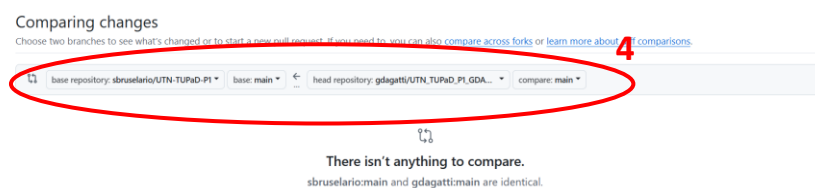
1. Ve al repositorio en GitHub donde deseas enviar la solicitud de extracción.
2. Haz clic en la pestaña "Pull requests".



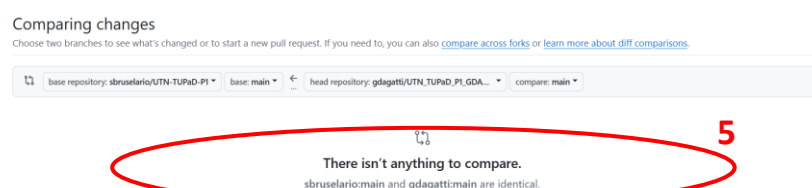
3. Haz clic en el botón "New pull request".



4. Selecciona la rama base y la rama de comparación. La rama base es donde deseas fusionar los cambios, y la rama de comparación es la que contiene los cambios que deseas fusionar.



5. Revisa los cambios propuestos y asegúrate de que todo esté correcto.



6. Añade un título y una descripción para tu solicitud de extracción.
7. Haz clic en "Create pull request" para enviar la solicitud.

- ¿Cómo aceptar una solicitud de extracción?

1. Ve al repositorio en GitHub donde se ha enviado la solicitud de extracción.
2. Haz clic en la pestaña "Pull requests".
3. Busca la solicitud de extracción que desees aceptar y haz clic en ella para abrirla.
4. Revisa los cambios propuestos y asegúrate de que todo esté correcto.
5. Si estás satisfecho con los cambios, haz clic en el botón "Merge pull request".
6. Confirma la fusión haciendo clic en "Confirm merge".
7. Opcionalmente, puedes eliminar la rama de la solicitud de extracción haciendo clic en "Delete branch"

- ¿Qué es una etiqueta en Git?

Una etiqueta en Git es una referencia que apunta a un punto específico en la historia del repositorio.

- ¿Cómo crear una etiqueta en Git?

git tag "Nombre_etiqueta"

git tag -a "Nombre_etiqueta" -m "mensaje_etiqueta"

- ¿Cómo enviar una etiqueta a GitHub?

git push origin "Nombre_etiqueta"

- ¿Qué es un historial de Git?

Es un registro de todos los cambios realizados en un repositorio a lo largo del tiempo.

- ¿Cómo ver el historial de Git?

git tag

- ¿Cómo buscar en el historial de Git?

git log

git log --grep="término_búsqueda" (para buscar un commit específico)

- ¿Cómo borrar el historial de Git?

rm -rf .git

- ¿Qué es un repositorio privado en GitHub?

Es un tipo de repositorio que solo es accesible para ti y las personas a las que invite el creador de este.

- ¿Cómo crear un repositorio privado en GitHub?

Siguiendo los pasos para crear un repositorio, explicado anteriormente, entre los pasos 4 y 5. Deténgase en la sección de visibilidad, selecciona "Private" para que el repositorio sea privado

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 (*).

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner *

gdagatti ▾

Repository name *

/

Great repository names are short and memorable. Need inspiration? How about [refactored-rotary-phone](#) ?

Description (optional)

☒ 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.

Aquí

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.](#)

 You are creating a public repository in your personal account.

Create repository

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

1. Ve a GitHub e inicia sesión en tu cuenta.

2. Navega al repositorio privado al que deseas invitar a alguien.
3. Haz clic en la pestaña "Settings" (Configuración) en la parte superior del repositorio.
4. En el menú lateral, selecciona "Manage access" (Gestionar acceso).
5. Haz clic en el botón "Invite a collaborator" (Invitar a un colaborador).
6. Escribe el nombre de usuario o la dirección de correo electrónico de la persona que deseas invitar.
7. Haz clic en "Add" (Agregar) para enviar la invitación.

- ¿Qué es un repositorio público en GitHub?

Es un tipo de repositorio que es accesible para todas las personas que tengan acceso al link de este.

- ¿Cómo crear un repositorio público en GitHub?

Siguiendo los pasos para crear un repositorio, explicado anteriormente, entre los pasos 4 y 5. Deténgase en la sección de visibilidad, selecciona "Public" para que el repositorio sea privado

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 (*).

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner *

Repository name *

 gdagatti ▾ /

Great repository names are short and memorable. Need inspiration? How about [refactored-rotary-phone](#) ?

Description (optional)

☒

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.

 **Aquí**

- ¿Cómo compartir un repositorio público en GitHub?

Compartiendo el link del repositorio.

2) Realizar la siguiente actividad:

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

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner * gdagatti ▾ / Repository name * Tp2_Git_GitHub

✓ Tp2_Git_GitHub is available.

Great repository names are short and memorable. Need inspiration? How about [animated-giggle](#) ?

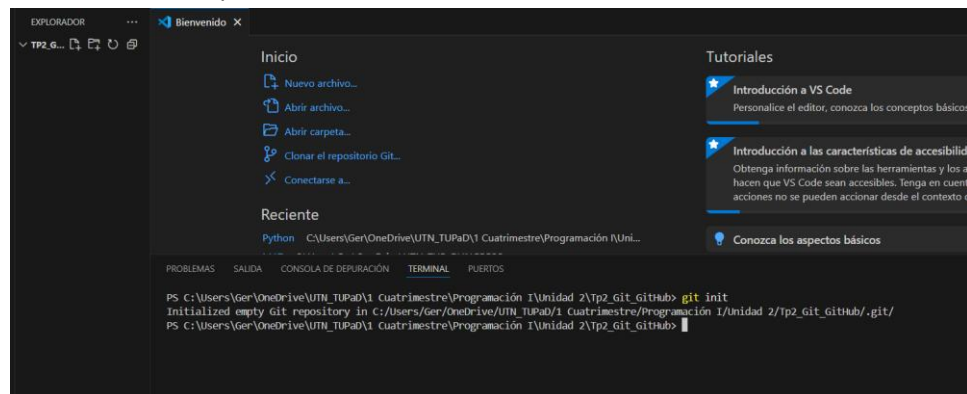
Description (optional)

Ejercicio2_TP2

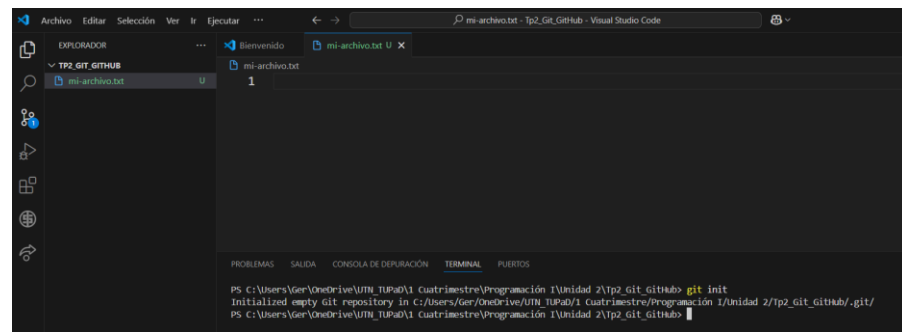
☒ Public
Anyone on the internet can see this repository. You choose who can commit.

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

- Inicializa el repositorio con un archivo.

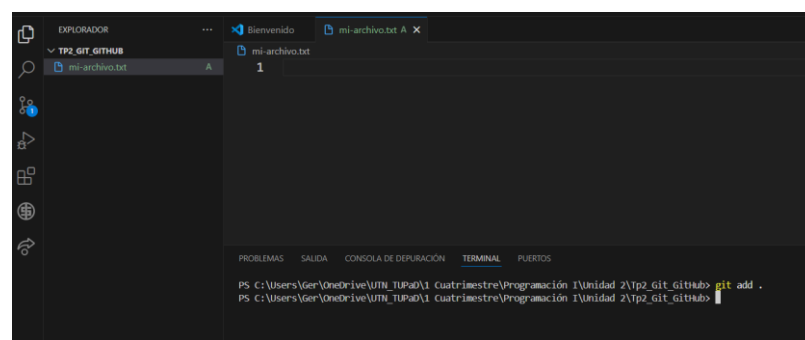


- Agregando un Archivo
 - Crea un archivo simple, por ejemplo, "mi-archivo.txt".

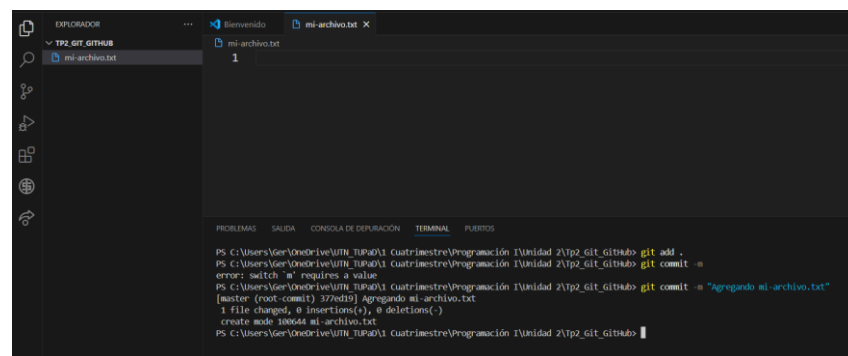


```
PS C:\Users\Ger\OneDrive\UTN_TUPad\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_Github> git init
Initialized empty Git repository in C:\Users\Ger\OneDrive\UTN_TUPad\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_Github\.git\
PS C:\Users\Ger\OneDrive\UTN_TUPad\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_Github>
```

- Realiza los comandos `git add .` y `git commit -m "Agregando mi-archivo.txt"` en la línea de comandos.

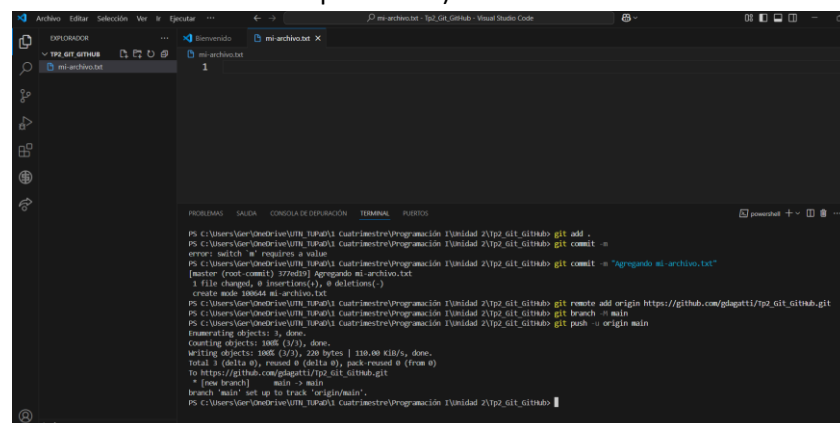


```
PS C:\Users\Ger\OneDrive\UTN_TUPad\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_Github> git add .
PS C:\Users\Ger\OneDrive\UTN_TUPad\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_Github>
```

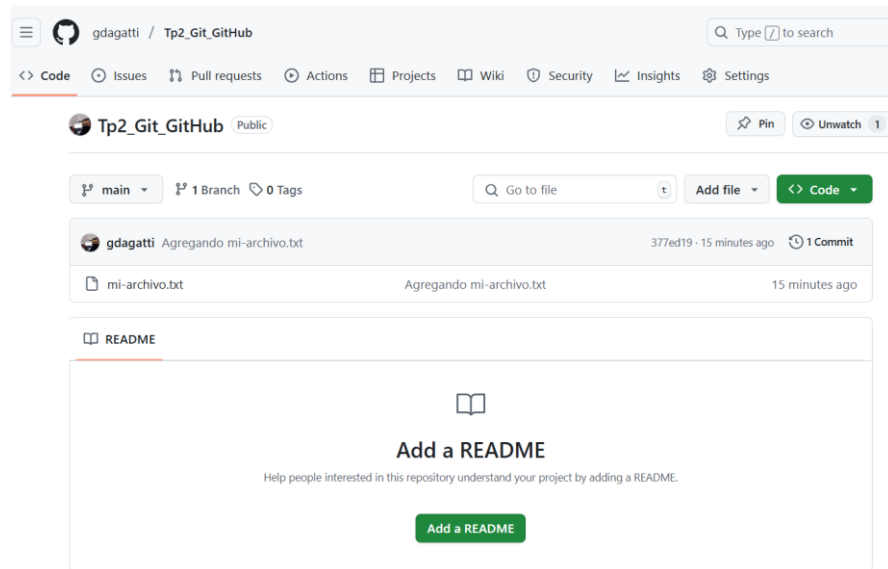


```
PS C:\Users\Ger\OneDrive\UTN_TUPad\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_Github> git commit -m "Agregando mi-archivo.txt"
[master (root-commit) 377ed19] Agregando mi-archivo.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 mi-archivo.txt
PS C:\Users\Ger\OneDrive\UTN_TUPad\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_Github>
```

- Sube los cambios al repositorio en GitHub con `git push origin main` (o el nombre de la rama correspondiente).

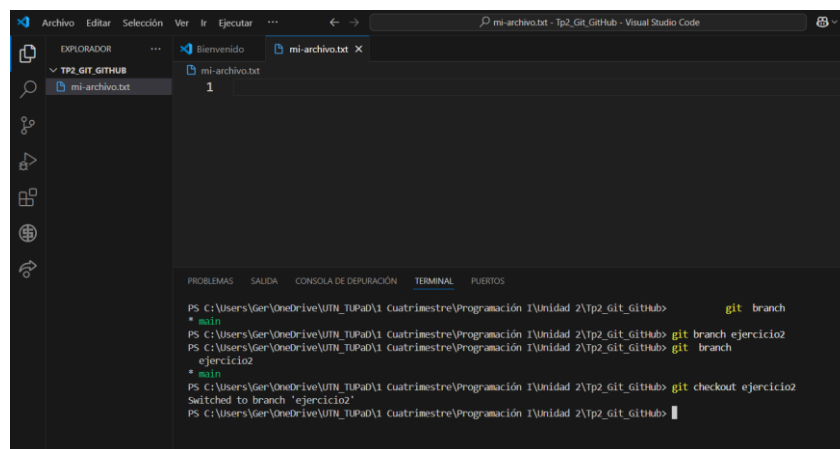


```
PS C:\Users\Ger\OneDrive\UTN_TUPad\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_Github> git push origin main
Enumerating objects: 3, done.
Writing objects: 100% (3/3), 220 bytes | 110.00 KiB/s, done.
Total: 3 (delta 0), reused 0 (delta 0), pack reused 0 (from 0)
To https://github.com/plagatti/tp2_git_github.git
 * [new branch]   main -> main
PS C:\Users\Ger\OneDrive\UTN_TUPad\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_Github>
```

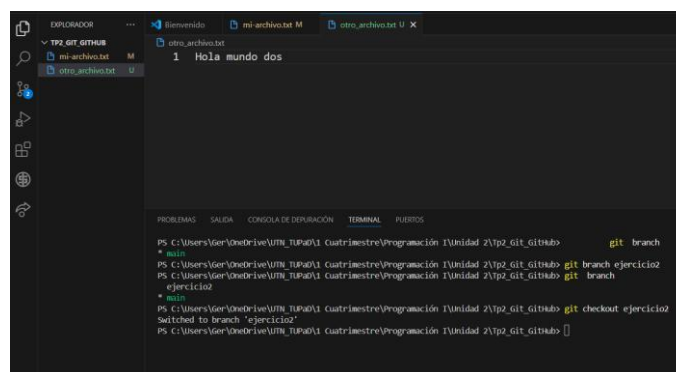


- Creando Branchs

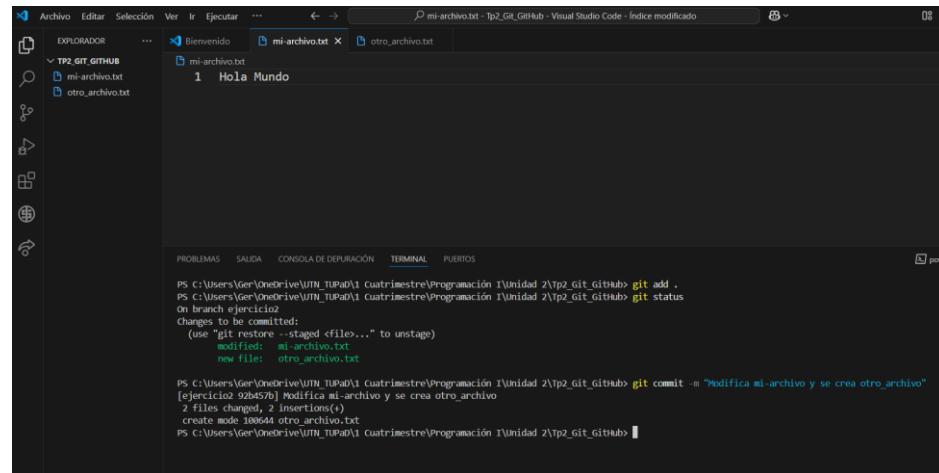
- Crear una Branch



- Realizar cambios o agregar un archivo



○ Subir la Branch

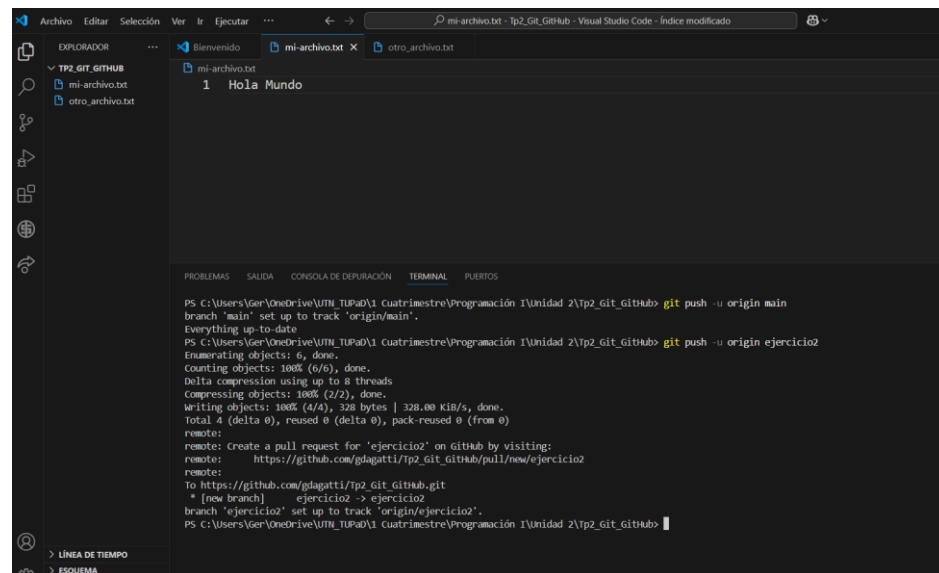


```

PS C:\Users\Ger\OneDrive\UTN_TUPAD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub> git add .
PS C:\Users\Ger\OneDrive\UTN_TUPAD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub> git status
On branch ejercicio2
Changes to be committed:
  (use "git restore --staged <files>..." to unstage)
        modified:   mi_archivo.txt
        new file:   otro_archivo.txt

PS C:\Users\Ger\OneDrive\UTN_TUPAD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub> git commit -m "Modifica mi-archivo y se crea otro_archivo"
[ejercicio2 92b457b] Modifica mi-archivo y se crea otro_archivo
 2 files changed, 2 insertions(+)
 create mode 100644 otro_archivo.txt
PS C:\Users\Ger\OneDrive\UTN_TUPAD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub>

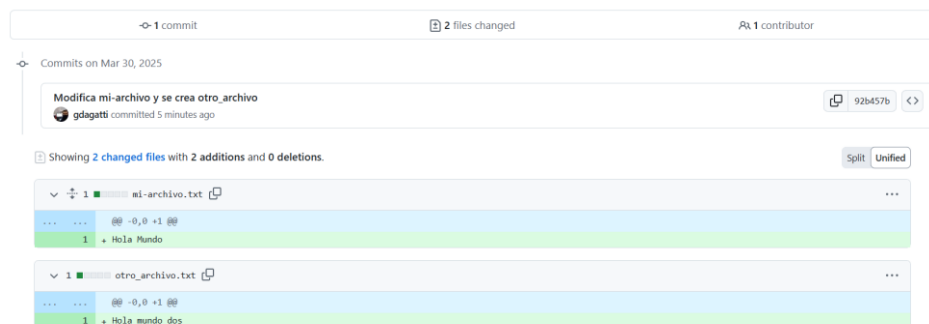
```



```

PS C:\Users\Ger\OneDrive\UTN_TUPAD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub> git push -u origin main
branch 'main' set up to track 'origin/main'.
Everything up-to-date
PS C:\Users\Ger\OneDrive\UTN_TUPAD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub> git push -u origin ejercicio2
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 328 bytes | 328.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'ejercicio2' on GitHub by visiting:
remote:   https://github.com/gdagatti/tp2_git_github/pull/new/ejercicio2
remote:
To https://github.com/gdagatti/tp2_git_github.git
 * [new branch] ejercicio2 -> ejercicio2
branch 'ejercicio2' set up to track 'origin/ejercicio2'.
PS C:\Users\Ger\OneDrive\UTN_TUPAD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub>

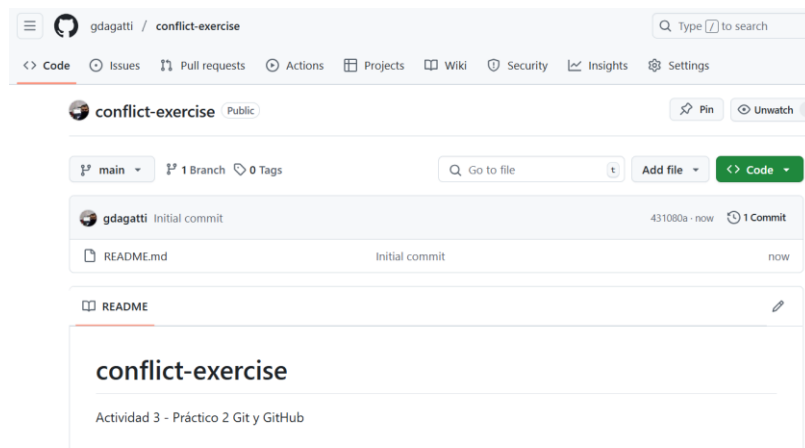
```



3) Realizar la siguiente actividad:

Paso 1: Crear un repositorio en GitHub

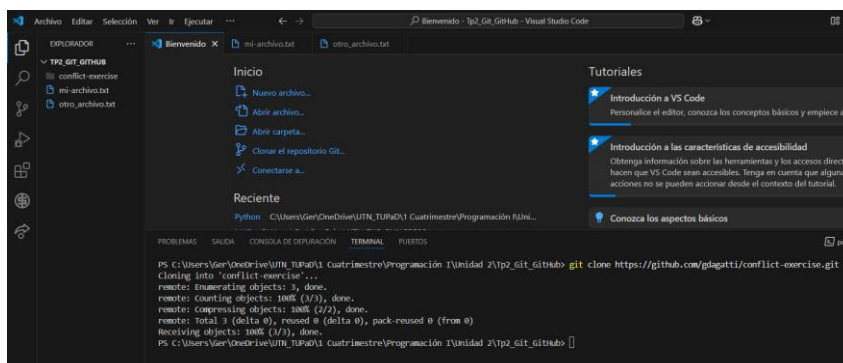
- Ve a GitHub e inicia sesión en tu cuenta.
- Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.
- Asigna un nombre al repositorio, por ejemplo, conflict-exercise.
- Opcionalmente, añade una descripción.
- Marca la opción "Initialize this repository with a README".
- Haz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



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`

```
Python C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub Conozca los aspectos b
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub> git clone https://github.com/
Cloning into 'conflict-exercise'...
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.
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub> cd conflict-exercise
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> |
```

Paso 3: Crear una nueva rama y editar un archivo

- Crea una nueva rama llamada feature-branch:

`git checkout -b feature-branch`

```
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub> git clone https://github.com/gdagatti/conflict-exercise.git
Cloning into 'conflict-exercise'...
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.
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub> cd conflict-exercise
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> git checkout -b feature-branch
Switched to a new branch 'feature-branch'
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> |
```

- 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.

```
EXPLORADOR ... Bienvenido mi-archivo.txt README.md X otro_archivo.txt
TP2_G... conflict-exercise > README.md > # conflict-exercise
1 # conflict-exercise
2 Actividad 3 - Práctico 2 Git y GitHub
3 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"`

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  powershell

PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub> git clone https://github.com/gdagatti/conflict-exercise.git
Cloning into 'conflict-exercise'...
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.
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub> cd conflict-exercise
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> git checkout -b feature-branch
Switched to a new branch 'feature-branch'
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> git add README.md
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> git commit -m "Added a line in feature-branch"
[feature-branch e0133c1] Added a line in feature-branch
1 file changed, 1 insertion(+)
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> |
```

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

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

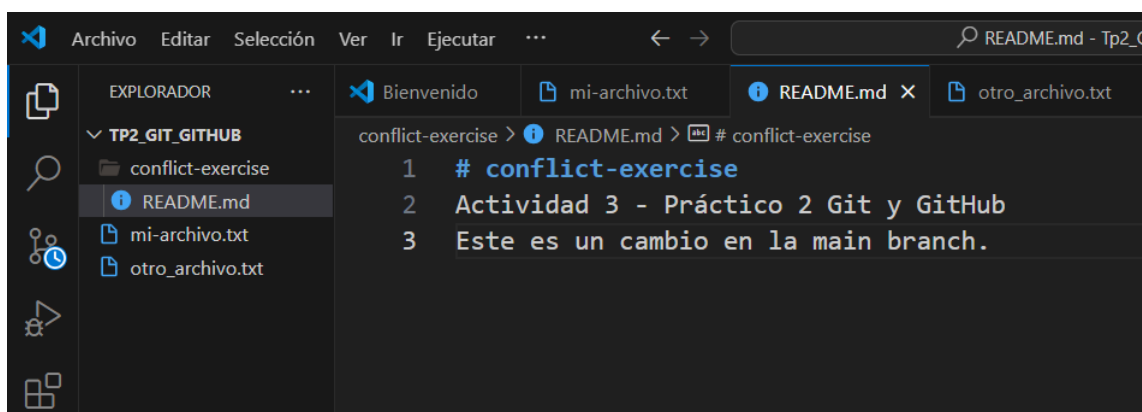
git checkout main

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  powershell

PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub> git clone https://github.com/gdagatti/conflict-exercise.git
Cloning into 'conflict-exercise'...
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.
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub> cd conflict-exercise
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> git checkout -b feature-branch
Switched to a new branch 'feature-branch'
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> git add README.md
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> git commit -m "Added a line in feature-branch"
[feature-branch e0133c1] Added a line in feature-branch
1 file changed, 1 insertion(+)
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\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"**

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub> git clone https://github.com/gdagatti/conflict-exercise.git
Cloning into 'conflict-exercise'...
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.
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub> cd conflict-exercise
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> git checkout -b feature-branch
Switched to a new branch 'feature-branch'
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> git add README.md
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> git commit -m "Added a line in feature-branch"
[feature-branch e0133c1] Added a line in feature-branch
1 file changed, 1 insertion(+)
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> git add README.md
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> git commit -m "Added a line in main branch"
[main 3375bcf] Added a line in main branch
1 file changed, 1 insertion(+)
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\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

```
Bienvenido  mi-archivo.txt  README.md  otro_archivo.txt
conflict-exercise > README.md > # Actividad 3 - Práctico 2 Git y GitHub<<<<<< HEADEste es un cambio en la main branch.
1 # conflict-exercise
2 Actividad 3 - Práctico 2 Git y GitHub
  Aceptar cambio actual | Aceptar cambio entrante | Aceptar ambos cambios | Comparar cambios
3 <<<<<< HEAD (Cambio actual)
4 Este es un cambio en la main branch.
5 =====
6 Este es un cambio en la feature branch.
7 >>>>>> feature-branch (Cambio entrante)
8

PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\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\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> |
```

- 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

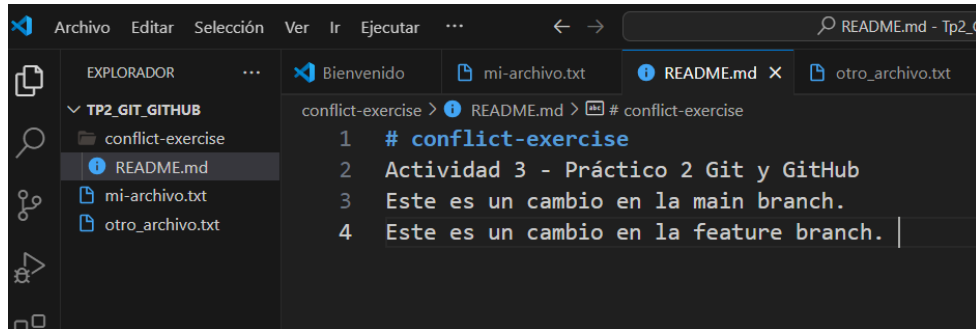
Este es un cambio en la main branch.

=====

Este es un cambio en la feature branch.

>>>>>> feature-branch

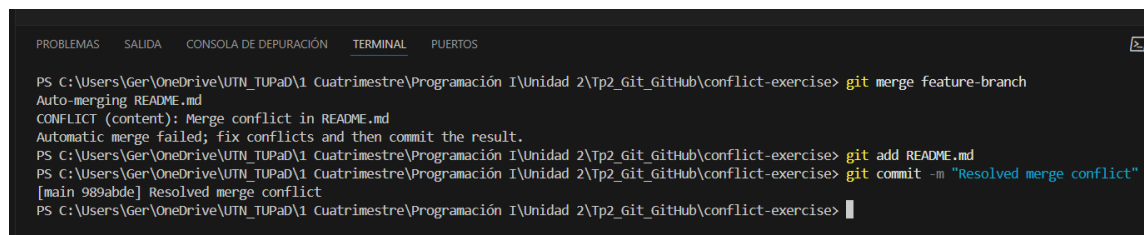
- Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.
- Edita el archivo para resolver el conflicto y guarda los cambios (Se debe borrar lo marcado en verde en el archivo donde estes solucionando el conflicto. Y se debe borrar 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"

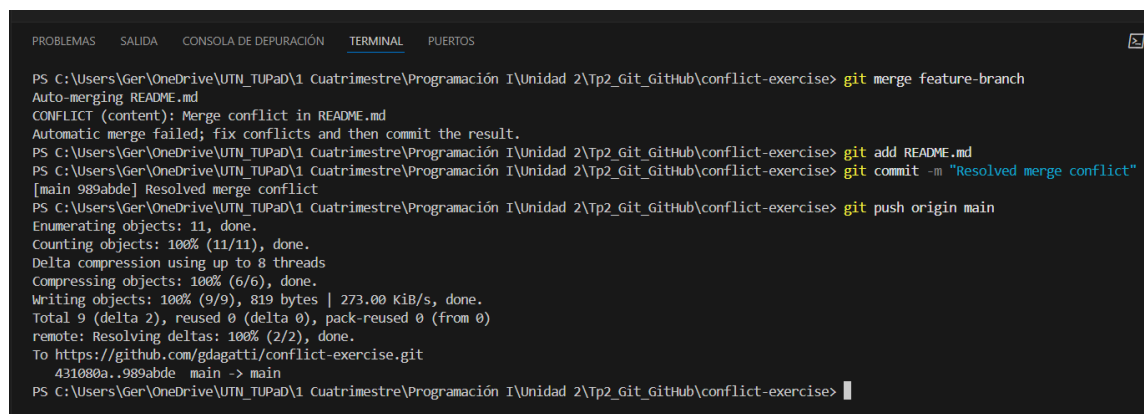


```
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> git merge feature-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> git add README.md
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> git commit -m "Resolved merge conflict"
[main 989abde] Resolved merge conflict
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\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`



```
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\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\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> git add README.md
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise> git commit -m "Resolved merge conflict"
[main 989abde] Resolved merge conflict
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\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), 819 bytes | 273.00 KiB/s, done.
Total 9 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/gdagatti/conflict-exercise.git
431080a..989abde main -> main
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\conflict-exercise>
```

- También sube la feature-branch si deseas:

git push origin feature-branch

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS

PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\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/gdagatti/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/gdagatti/conflict-exercise.git
 * [new branch]      feature-branch -> feature-branch
PS C:\Users\Ger\OneDrive\UTN_TUPaD\1 Cuatrimestre\Programación I\Unidad 2\Tp2_Git_GitHub\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.

History for [conflict-exercise](#) / [README.md](#) on [main](#)

Commits on Apr 1, 2025

Resolved merge conflict

gdagatti committed 6 minutes ago

Added a line in main branch

gdagatti committed 13 minutes ago

Added a line in feature-branch

gdagatti committed 28 minutes ago

Initial commit

gdagatti authored 39 minutes ago

End of commit history for this file

German Luis Dagatti
DNI: 31.230.293