

## Práctico 2: Git y GitHub

Nombre: Jennifer Franco

### 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

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

#### 1. ¿Qué es GitHub?:

GitHub es una plataforma para alojar repositorios de control de versiones, donde se pueden almacenar y gestionar los proyectos.

#### 2. ¿Cómo crear un repositorio en GitHub?:

Para crear un repositorio en GitHub, hay dos opciones: crearlo mediante la página (previamente se debe abrir una cuenta) o desde el repositorio local.

Mediante la página se debe hacer click en New, luego la página brinda las indicaciones a seguir para asociarlo con el repositorio local.

#### 3. ¿Cómo crear una rama en Git?

Mediante el comando git branch: git branch nombreNuevaRama

#### 4. ¿Cómo cambiar a una rama en Git?:

Mediante el comando git checkout: git checkout nombreNuevaRama

- git checkout -b nombreNuevaRama (si se quiere crear y mover a la rama en un solo paso)

#### 5. ¿Cómo fusionar ramas en Git?

Mediante el comando git merge: primero debo posicionarme en la rama donde me quiero fusionar y una vez en ella ejecuto el comando git merge nombreNuevaRama.

#### 6.¿Cómo crear un commit en Git?

Para realizar un commit primero se ejecuta el comando `git add` para agregar los cambios al área de preparación, se puede optar por agregar un sólo archivo (`git add "nombreArchivo"`) o todos los archivos `git add .` Luego se ejecuta el comando `git commit` y se agrega un mensaje: `git commit -m "ejemplo de commit"`.

#### 7.¿Cómo enviar un commit a GitHub?

Se puede enviar un commit mediante `git push origin nombreDeLaRama`

#### 8.¿Qué es un repositorio remoto?

Un repositorio remoto son versiones del proyecto alojadas en la web

#### 9.¿Cómo agregar un repositorio remoto a Git? `git remote add origin URL_DEL_REPO`

Mediante el comando `git remote add` se puede vincular un repositorio local con un repositorio remoto y asignar un nombre a ese remoto: `git remote add nombre URL_DEL_REPO`. Para asegurar que el paso se concrete correctamente se puede ejecutar `git remote -v`, se mostrará una lista de todos los remotos configurados.

Luego, para traer toda la información del repositorio remoto al repositorio local se usa el comando `git fetch` seguido del nombre del remoto: `git fetch nombre`

#### 10.¿Cómo empujar cambios a un repositorio remoto?

Mediante el comando `git push origin nombreDeLaRama`

#### 11.¿Cómo tirar de cambios de un repositorio remoto?

Mediante el comando `git pull nombreDeLaRama`

#### 12.¿Qué es un fork de repositorio?

Un fork de repositorio es una manera para tener una copia de un repositorio que nos llamó la atención y poder hacerle modificaciones sin alterar el repo original.

#### 13.¿Cómo crear un fork de un repositorio?

En la página de GitHub nos da la opción de crear un fork.

#### 14.¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?

Para realizar un pull request, primero debemos acceder a la pestaña de "Pull requests" y hacer clic en "New Pull Request", se nos mostrará un resumen de los cambios que hemos realizado en comparación con el repositorio original. Luego, presionamos el botón "Create Pull Request", donde podremos añadir un título (generalmente un mensaje breve que resuma el cambio) y, en el campo de descripción, proporcionar detalles más específicos sobre las razones que justifican nuestro cambio, explicando por qué creemos que esta modificación sería beneficiosa para el repositorio original.

#### 15.¿Cómo aceptar una solicitud de extracción?

Haciendo click en Merge pull request se agregan al repositorio original los cambios que hizo un usuario y propuso mediante pull request.

#### 16.¿Qué es un etiqueta en Git?

Es una referencia que señala un punto específico en la historia de un repositorio, generalmente utilizado para marcar versiones importantes.

#### 17.¿Cómo crear una etiqueta en Git?

Hay dos tipos de etiquetas, las ligeras y las anotadas. Las primeras son similar a la creación de una rama pero las anotadas tienen más datos (nombre, mail y fecha), se le puede agregar un mensaje y además pueden ser verificadas.

Etiqueta ligera: se usa el comando `git tag nombre_etiqueta`

Etiqueta anotada: se usa el comando `git tag -a nombre_etiqueta <identificador_del_comit> -m "mensaje asociado"`

#### 18.¿Cómo enviar una etiqueta a GitHub?

Con el comando `git push origin nombre_etiqueta`.

Para enviar todas las etiquetas a la vez: `git push origin —tags`

#### 19.¿Qué es un historial de Git?

El historial es una secuencia de todos los cambios realizados en un repositorio de Git.

#### 20.¿Cómo ver el historial de Git?

Podemos ver el historial con el comando `git log`

#### 21.¿Cómo buscar en el historial de Git?

Comandos claves:

`git log -grep="palabra clave"` (para buscar por palabra que aparezca en un mensaje de commit)

`git log —nombre_del_archivo` (para buscar los commits de un archivo puntual)

`git log —since= "fecha de inicio" —until="fecha final"` (buscar en un rango de fechas)

`git log —author="nombre"` (para buscar los commits de un autor puntual)

#### 22.¿Cómo borrar el historial de Git?

Mediante el comando `git reset`

#### 23.¿Qué es un repositorio privado en GitHub?

El repositorio privado tiene acceso exclusivo, solo para personas autorizadas.

24.¿Cómo crear un repositorio privado en GitHub

A la hora de crear el repositorio hay que seleccionar la opción “ Private”

25.¿Cómo invitar a alguien a un repositorio privado en GitHub?

En la pestaña “Settings” se encuentra la opción “Collaborators” donde se puede ingresar el nombre de usuario de GitHub y seleccionar el nivel de acceso que le queremos otorgar.

26.¿Qué es un repositorio público en GitHub?

Al repositorio público puede acceder cualquier persona.

27.¿Cómo crear un repositorio público en GitHub?

Es el mismo proceso que para crear uno privado pero esta vez seleccionamos la opción “Public”

28.¿Cómo compartir un repositorio público en GitHub?

La forma más común es enviando la URL.

1) Realizar la siguiente actividad:

. Crear un repositorio.

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

i. 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).

The screenshot displays a Windows 10 desktop environment. The primary focus is the Visual Studio Code (VS Code) editor, which is running in administrator mode. The editor's interface includes a sidebar on the left with icons for Explorer, Search, Source Control, and Run and Debug. The main editor area shows a file named 'prueba.html' with the content `<html lang="en">`. Below the editor, the TERMINAL panel is active, showing the following commands and output:

```
PS C:\Users\Jennifer\Desktop\TP N2 programacion> git clone https://github.com/JenniferFranco/prueba_repositorio.git
Cloning into 'prueba_repositorio'...
warning: You appear to have cloned an empty repository.
PS C:\Users\Jennifer\Desktop\TP N2 programacion> ls

Directorio: C:\Users\Jennifer\Desktop\TP N2 programacion

Directorio: C:\Users\Jennifer\Desktop\TP N2 programacion

Mode                LastWriteTime         Length Name
----                -
d-----          3/4/2025   20:45                prueba_repositorio

PS C:\Users\Jennifer\Desktop\TP N2 programacion> cd .\prueba_repositorio\
PS C:\Users\Jennifer\Desktop\TP N2 programacion\prueba_repositorio> git add .
PS C:\Users\Jennifer\Desktop\TP N2 programacion\prueba_repositorio> git status
On branch main

No commits yet
```

The taskbar at the bottom of the screen shows several application icons, including the Start button, Search, File Explorer, Task View, and various web browsers. The system clock in the bottom right corner indicates the date as 3/4/2025 and the time as 20:51.

The screenshot shows a Windows 10 desktop with a VS Code editor window. The editor is open to a file named 'prueba.html' in a directory structure that includes 'TP N2 programaci...' and 'prueba\_reposito...'. The file content is `<html lang="en">`. The terminal window at the bottom shows the following commands and output:

```

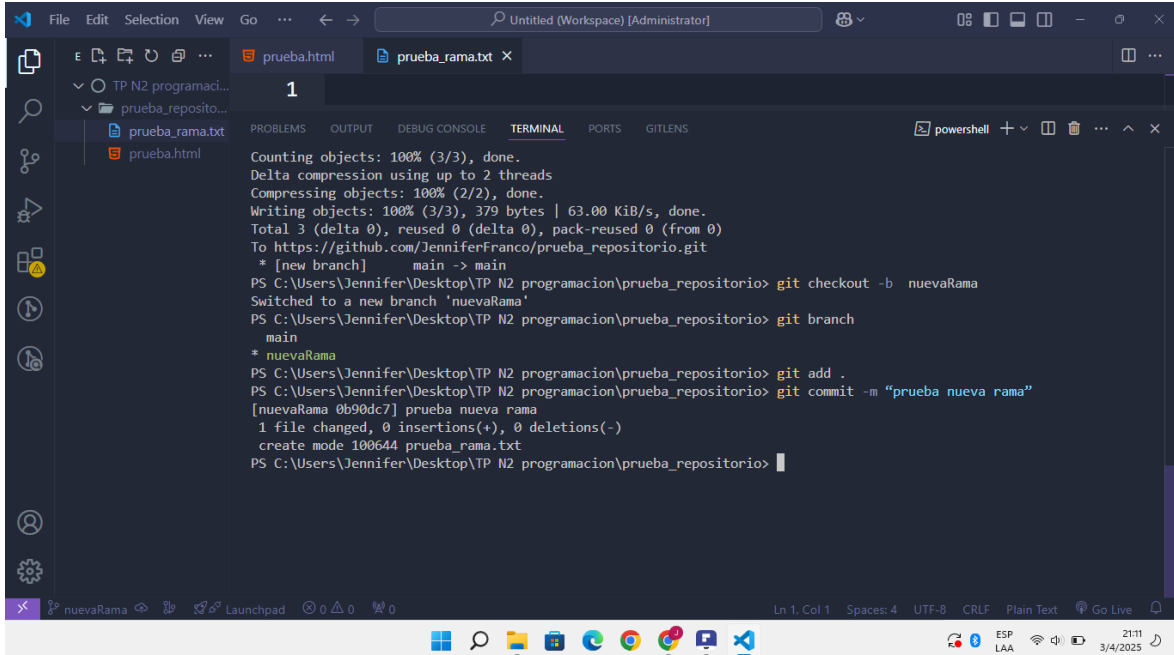
PS C:\Users\Jennifer\Desktop\TP N2 programacion\prueba_repositorio> git commit -m "primer commit"
[main (root-commit) c21ff2f] primer commit
1 file changed, 11 insertions(+)
create mode 100644 prueba.html
PS C:\Users\Jennifer\Desktop\TP N2 programacion\prueba_repositorio> git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 379 bytes | 63.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/JenniferFranco/prueba_repositorio.git
 * [new branch]      main -> main
PS C:\Users\Jennifer\Desktop\TP N2 programacion\prueba_repositorio>

```

The status bar at the bottom of VS Code indicates the current branch is 'main', the file is encoded in UTF-8, and the language is HTML. The system taskbar at the very bottom shows the time as 20:51 on 3/4/2025.

## ii. Creando Branchs

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



```
Counting objects: 100% (3/3), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 379 bytes | 63.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/JenniferFranco/prueba_repositorio.git
* [new branch]    main -> main
PS C:\Users\Jennifer\Desktop\TP N2 programacion\prueba_repositorio> git checkout -b nuevaRama
Switched to a new branch 'nuevaRama'
PS C:\Users\Jennifer\Desktop\TP N2 programacion\prueba_repositorio> git branch
main
* nuevaRama
PS C:\Users\Jennifer\Desktop\TP N2 programacion\prueba_repositorio> git add .
PS C:\Users\Jennifer\Desktop\TP N2 programacion\prueba_repositorio> git commit -m "prueba nueva rama"
[nuevaRama 0b90dc7] prueba nueva rama
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 prueba_rama.txt
PS C:\Users\Jennifer\Desktop\TP N2 programacion\prueba_repositorio>
```

## 2) 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`

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"`

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"
```

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
```

- 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"
```

Paso 7: Subir los cambios a GitHub

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

```
git push origin main
```

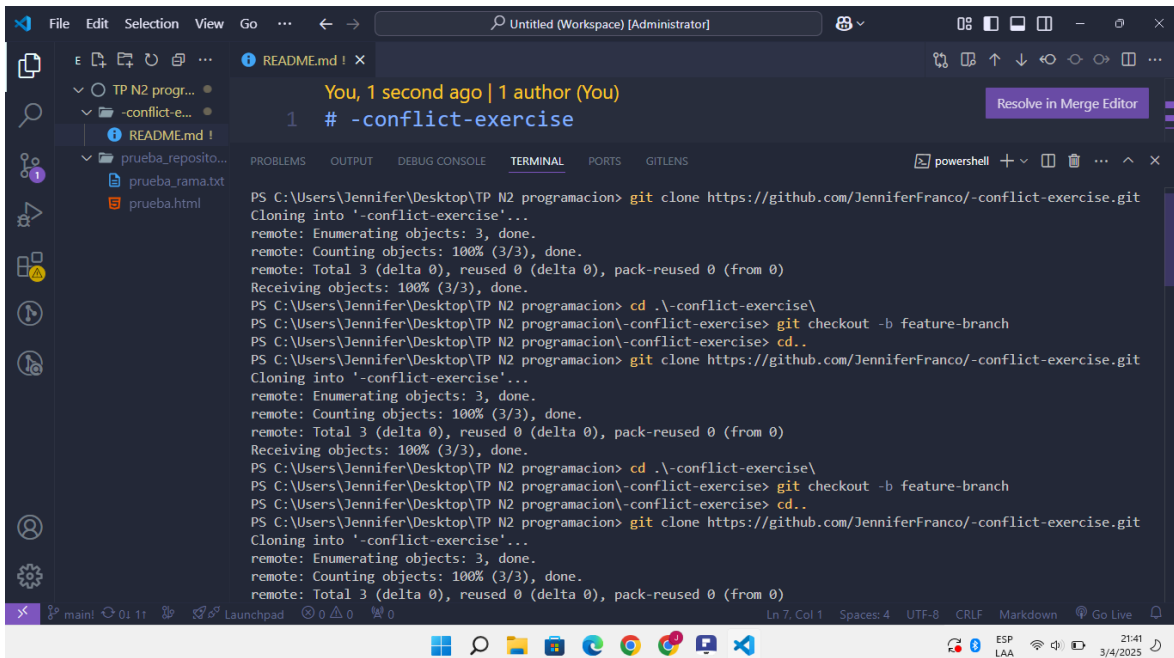


- También sube la feature-branch si deseas:

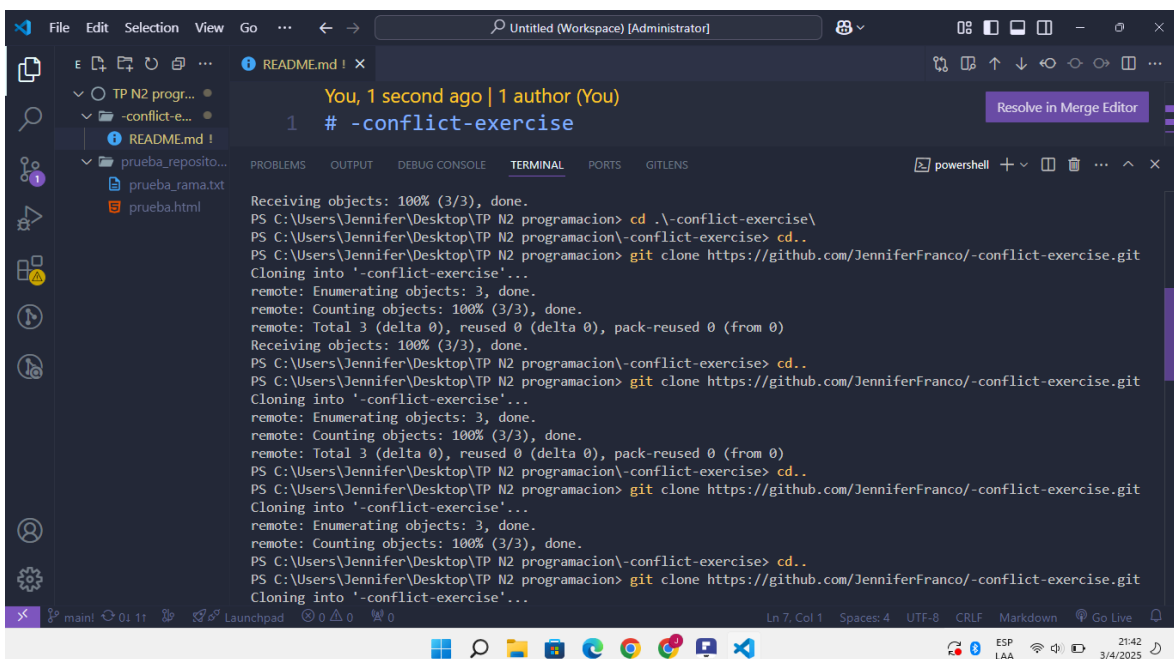
**git push origin feature-branch**

#### Paso 8: Verificar en GitHub

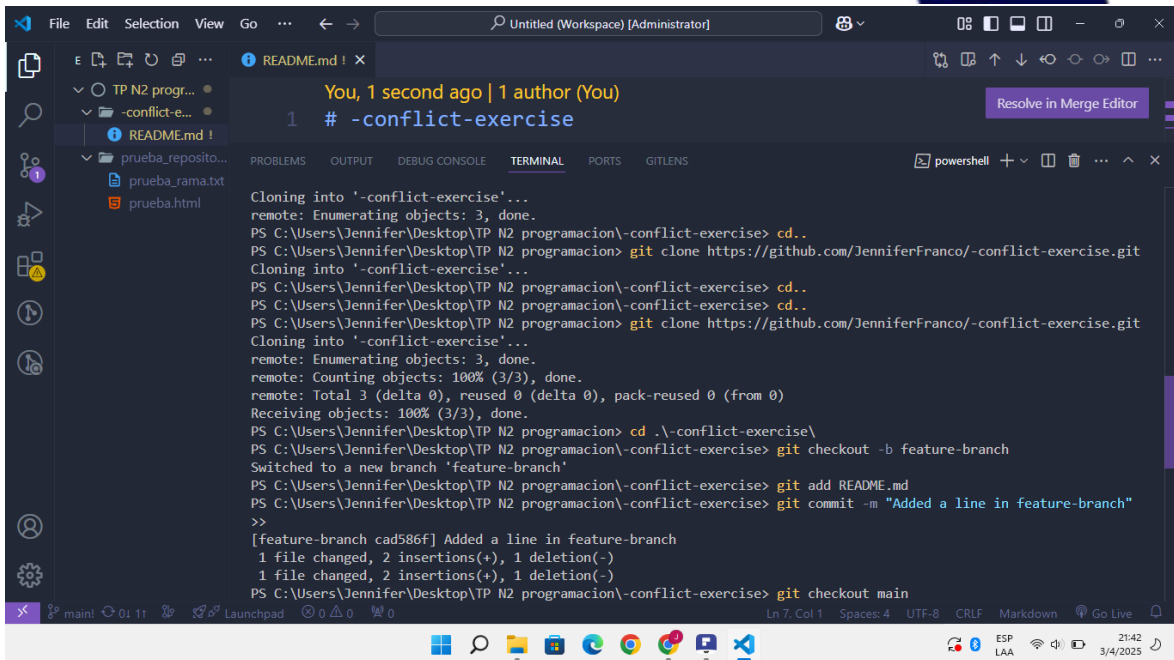
- Ve a tu repositorio en GitHub y revisa el archivo README.md para confirmar que los cambios se han subido correctamente.
- Puedes revisar el historial de commits para ver el conflicto y su resolución.



```
PS C:\Users\Jennifer\Desktop\TP N2 programacion> git clone https://github.com/JenniferFranco/-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.
PS C:\Users\Jennifer\Desktop\TP N2 programacion> cd .\-conflict-exercise\
PS C:\Users\Jennifer\Desktop\TP N2 programacion\conflict-exercise> git checkout -b feature-branch
PS C:\Users\Jennifer\Desktop\TP N2 programacion\conflict-exercise> cd..
PS C:\Users\Jennifer\Desktop\TP N2 programacion> git clone https://github.com/JenniferFranco/-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.
PS C:\Users\Jennifer\Desktop\TP N2 programacion> cd .\-conflict-exercise\
PS C:\Users\Jennifer\Desktop\TP N2 programacion\conflict-exercise> git checkout -b feature-branch
PS C:\Users\Jennifer\Desktop\TP N2 programacion\conflict-exercise> cd..
PS C:\Users\Jennifer\Desktop\TP N2 programacion> git clone https://github.com/JenniferFranco/-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.
PS C:\Users\Jennifer\Desktop\TP N2 programacion> cd .\-conflict-exercise\
PS C:\Users\Jennifer\Desktop\TP N2 programacion\conflict-exercise> cd..
PS C:\Users\Jennifer\Desktop\TP N2 programacion> git clone https://github.com/JenniferFranco/-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.
PS C:\Users\Jennifer\Desktop\TP N2 programacion\conflict-exercise> cd..
PS C:\Users\Jennifer\Desktop\TP N2 programacion> git clone https://github.com/JenniferFranco/-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)
PS C:\Users\Jennifer\Desktop\TP N2 programacion\conflict-exercise> cd..
PS C:\Users\Jennifer\Desktop\TP N2 programacion> git clone https://github.com/JenniferFranco/-conflict-exercise.git
Cloning into '-conflict-exercise'...
```

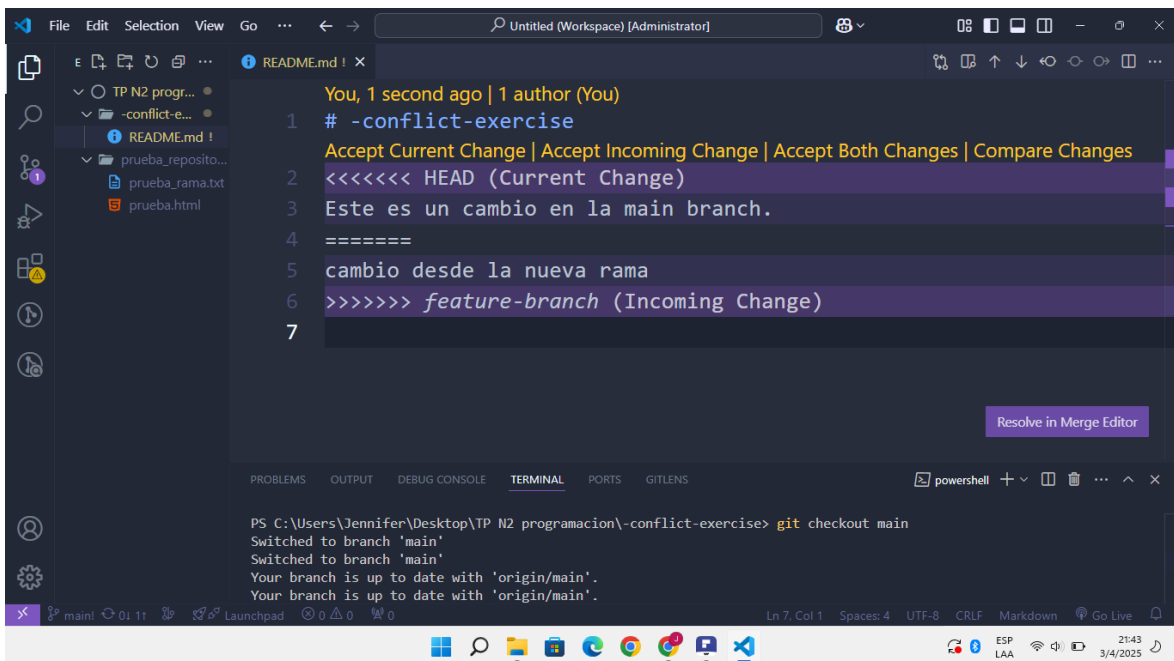


The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows a project structure with a folder named 'prueba\_reposito...' containing files 'prueba\_rama.txt' and 'prueba.html'. The main editor area displays a file named 'README.md' with the following content:

```
1 You, 1 second ago | 1 author (You)
# -conflict-exercise
```

The TERMINAL pane at the bottom shows the following commands and output:

```
Cloning into '-conflict-exercise'...
remote: Enumerating objects: 3, done.
PS C:\Users\Jennifer\Desktop\TP N2 programacion> cd..
PS C:\Users\Jennifer\Desktop\TP N2 programacion> git clone https://github.com/JenniferFranco/-conflict-exercise.git
Cloning into '-conflict-exercise'...
PS C:\Users\Jennifer\Desktop\TP N2 programacion> cd..
PS C:\Users\Jennifer\Desktop\TP N2 programacion> cd..
PS C:\Users\Jennifer\Desktop\TP N2 programacion> git clone https://github.com/JenniferFranco/-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.
PS C:\Users\Jennifer\Desktop\TP N2 programacion> cd .\-conflict-exercise\
PS C:\Users\Jennifer\Desktop\TP N2 programacion> git checkout -b feature-branch
Switched to a new branch 'feature-branch'
PS C:\Users\Jennifer\Desktop\TP N2 programacion> git add README.md
PS C:\Users\Jennifer\Desktop\TP N2 programacion> git commit -m "Added a line in feature-branch"
>>
[feature-branch cad586f] Added a line in feature-branch
1 file changed, 2 insertions(+), 1 deletion(-)
1 file changed, 2 insertions(+), 1 deletion(-)
PS C:\Users\Jennifer\Desktop\TP N2 programacion> git checkout main
```



The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows a project structure with a folder named 'prueba\_reposito...' containing files 'prueba\_rama.txt' and 'prueba.html'. The main editor area displays a file named 'README.md' with the following content:

```
1 You, 1 second ago | 1 author (You)
# -conflict-exercise
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 cambio desde la nueva rama
7 >>>>>> feature-branch (Incoming Change)
```

The TERMINAL pane at the bottom shows the following commands and output:

```
PS C:\Users\Jennifer\Desktop\TP N2 programacion> git checkout main
Switched to branch 'main'
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
Your branch is up to date with 'origin/main'.
```

