

Integrante: Victor Manuel Barroeta Almedo

Matricula:100393

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 en línea que permite almacenar, compartir y colaborar en repositorios públicos o privados, que permiten a los desarrolladores gestionar proyectos de software utilizando un sistema de control de versiones por Git. Es el servicio más popular del mundo para desarrollar en equipo, ya que además de alojar código, permite gestionar tareas, hacer revisiones, fusionar cambios, intercambiar código y trabajar de forma remota.

- ¿Cómo crear un repositorio en GitHub?

1. Ingresar a <https://github.com>
2. Hacer clic en "+ New repository"
3. Escribir un nombre para el repositorio y una descripción opcional.
4. Elegir si será público o privado.

5. Hacer clic en "Create repository".

En el caso de tener ya un repositorio local en cualquier editor y deseas mandarlo o sincronizarlo a la plataforma, se debe ingresar en la consola de bash y tipear lo siguiente:

Ejemplo:

```
$ git remote add origin https://github.com/victormbar/UTN-TUPaD-P1
```

```
$ git push -u origin master
```

Después refrescar la página de github y se podrá ver subido en nuestro repositorio.

- ¿Cómo crear una rama en Git?

Usar el comando:

```
git branch nombre-de-la-rama
```

Crea una nueva rama sin cambiarte a ella.

- ¿Cómo cambiar a una rama en Git?

Usar el comando:

```
git checkout nueva-rama
```

Para crear y cambiar al mismo tiempo:

```
git checkout -b nueva-rama
```

- ¿Cómo fusionar ramas en Git?

Primero, debes estar en la rama a la que quieres fusionar los cambios. Por lo general, se cambia a la rama principal (master o main) o cualquier otra rama que quieras fusionar o integrar los cambios con **\$ git checkout main** o **\$ git checkout nombre-de-la-rama-primaria** y luego se ejecuta: **\$ git merge nombre-de-la-rama-secundaria**

- ¿Cómo crear un commit en Git?

Crear un commit básicamente es el proceso por el cual se guardan los cambios realizados en el repositorio, captura estos cambios y lo almacena en el historial del proyecto para en espera de la aprobación de los cambios si no es tu repositorio propio, si es el repositorio propio se guardan los cambios automáticamente para que no importe en qué lugar estes puedas ver esos cambios.

Para hacer un commit tienes que:

Después de realizar los cambios del archivo debe agregarlos en un área de preparación agregando un archivo en específico con el comando **\$git add archivo1** o **\$git add** . (para agregar todos los archivos)

Una vez asignada el área de preparación, puede hacer el commit con el comando **\$git commit**. A su vez debes proporcionar un mensaje en ese commit que describa los cambios que realizaste. Ejp: **\$git commit -m "Cambio de ciclo for"**

Esto guardara el estado del código, los cambios realizados en el historial con el mensaje que

realizaste para un mayor entendimiento de todo lo realizado.

- ¿Cómo enviar un commit a GitHub?

Primero antes de poder enviar el commit al Github, se debe clonar el repositorio en tu maquina de manera local con el comando **\$git clone**. Ejp:

\$git clone <https://github.com/victormanbar/UTN-TUPaD-P1>

cd Programación_1

Después se realizan los cambios del archivo deseado y se agregan con el siguiente comando:

\$git add nombre_del_archivo o **git add** .

Se crea un commit con los cambios

\$git commit -m "Se realizaron cambio en el código"

Y al finalizar con este procedimiento se envia o empuja a Github con el comando **git push origin**. Ejp:

\$git push origin nombre_de_la_rama

- ¿Qué es un repositorio remoto?

Un **repositorio remoto** es una versión de tu proyecto que está alojada en internet o en una red, en lugar de estar en tu computadora local. Piensa en él como una "copia de seguridad central" o un "punto de encuentro" para tu código.

- ¿Cómo agregar un repositorio remoto a Git?

Se agrega con el comando **git remote add <nombre> <url>**.

El nombre más común es origin. Ejemplo: **git remote add origin**

<https://github.com/victormanbar/UTN-TUPaD-P1>

- ¿Cómo empujar cambios a un repositorio remoto?

Se usa el comando **git push <remoto> <rama>**.

Ejemplo: **git push origin main** para enviar los cambios de la rama main al remoto origin.

- ¿Cómo tirar de cambios de un repositorio remoto?

Se usa el comando **git pull <remoto> <rama>**.

Ejemplo: **git pull origin main** para traer y fusionar los cambios de la rama main desde el remoto origin.

- ¿Qué es un fork de repositorio?

Un fork es una copia personal de un repositorio ajeno que reside en tu propia cuenta de Github. Te permite experimentar libremente con un proyecto sin afectar el original.

- ¿Cómo crear un fork de un repositorio?

En la página del repositorio en Github, haz clic en el botón "Fork" en la esquina superior derecha. En el cual haces una copia con el mismo nombre

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

1. Se hace push de los cambios a tu fork.
2. En la página de tu fork en GitHub, aparecerá un botón "Contribute" o "Pull request".
3. Al hacer clic, se abrirá un formulario para describir tus cambios y enviarlos al repositorio original para su revisión.

- ¿Cómo aceptar una solicitud de extracción?

El dueño o colaborador del repositorio original puede revisar los cambios en la pestaña "Pull requests" y, si todo está bien, hacer clic en el botón verde "Merge pull request".

- ¿Qué es una etiqueta en Git?

Git tiene la posibilidad de etiquetar puntos específicos del historial como importantes.

Una etiqueta (tag) es un marcador que se usa para señalar un punto específico en el historial, comúnmente utilizado para marcar versiones de lanzamiento (ej. v1.0, v2.1.5).

- ¿Cómo crear una etiqueta en Git?

Se crea con `git tag <nombre-etiqueta>`. Ejemplo: `git tag v1.0`.

Git utiliza dos tipos principales de etiquetas: ligeras y anotadas.

La ligera es no cambian, las anotadas se guardan y van cambiando en base a parámetros en git en el cual se guardan en el tag. Se recomienda usarla para más información en las etiquetas

- ¿Cómo enviar una etiqueta a GitHub?

Primero debes crear la etiqueta localmente. Puedes crear una etiqueta anotada (que incluye información adicional como el autor y la fecha) o una etiqueta ligera (simplemente un puntero a un commit). Y Para empujar todas las etiquetas creadas, usar el comando `git push origin --tags`.

- ¿Qué es un historial de Git?

Es el registro cronológico de todos los commits (guardados) que se han realizado en el proyecto. Contiene información sobre quién hizo el cambio, cuándo y qué se modificó.

- ¿Cómo ver el historial de Git?

Con el comando `git log`. Existen muchas variantes como `git log --oneline` para una vista más compacta. Muestra un resumen conciso de los commits recientes, con cada commit representado en una sola línea.

- ¿Cómo buscar en el historial de Git?

Usando opciones del comando `git log`:

- Por autor: `git log --author="Nombre"`
- Por contenido del cambio: `git log -S"texto_a_buscar"`
- Por mensaje de commit: `git log --grep="mensaje_a_buscar"`
- ¿Cómo borrar el historial de Git?

Borrar el historial es una operación destructiva y muy desaconsejada. No hay un comando simple para ello y usualmente implica técnicas avanzadas y peligrosas como el rebase interactivo o la creación de un nuevo repositorio desde cero.

- ¿Qué es un repositorio privado en GitHub?

Es un repositorio que solo es visible y accesible para su dueño y para los colaboradores que explícitamente invite.

- ¿Cómo crear un repositorio privado en GitHub?

Al crear un nuevo repositorio en la web de GitHub, selecciona la opción "Private".

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

1. Ve a la pestaña "Settings" de tu repositorio.
2. En el menú de la izquierda, entra en "Collaborators".
3. Agrega a la persona por su nombre de usuario de GitHub.

- ¿Qué es un repositorio público en GitHub?

Es un repositorio visible y accesible para cualquier persona en internet. Cualquiera puede verlo, clonarlo y hacer un fork.

- ¿Cómo crear un repositorio público en GitHub?

Al crear un nuevo repositorio en la web de GitHub, selecciona la opción "Public".


- ¿Cómo compartir un repositorio público en GitHub?


Simplemente copia y comparte la URL del repositorio. Al ser público, cualquier persona con el enlace podrá acceder.

2) Realizar la siguiente actividad:

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


Required fields are marked with an asterisk (*).


Owner *  victormanbar / Repository name *

 RepositorioPrueba1 is available.

Great repository names are short and memorable. Need inspiration? How about [miniature-octo-sniffle](#) ?

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.

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.](#)

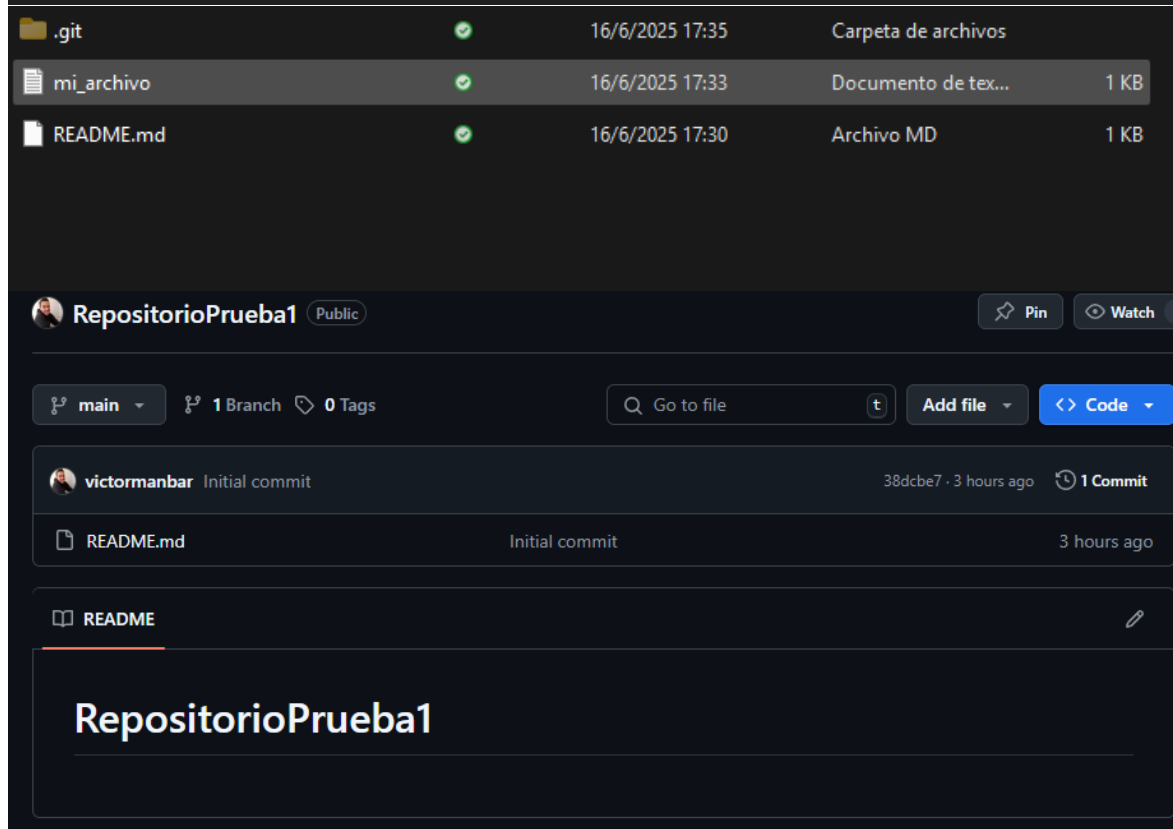
This will set  main as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

- 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).

```
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1> git clone https://github.com/victormanbar/RepositorioPrueba1
Cloning into 'RepositorioPrueba1'...
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\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1> |
```

```
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios trabajos\UTN-TUPaD-P1\02 Trabajo Colaborativo\RepositorioPrueb
a1> echo "Este es el contenido de mi archivo de texto." > mi_archivo.txt
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios trabajos\UTN-TUPaD-P1\02 Trabajo Colaborativo\RepositorioPrueb
a1> git add mi_archivo.txt
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios trabajos\UTN-TUPaD-P1\02 Trabajo Colaborativo\RepositorioPrueb
a1> git commit -m "Agrego mi archivo-txt al repositorio"
[main 86bd834] Agrego mi archivo-txt al repositorio
1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 mi_archivo.txt
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios trabajos\UTN-TUPaD-P1\02 Trabajo Colaborativo\RepositorioPrueb
a1>
```




The screenshot shows a GitHub repository interface. At the top, there's a file list with columns for file name, status (green checkmark), date/time, file type, and size. The files listed are '.git' (Carpeta de archivos), 'mi_archivo' (Documento de tex..., 1 KB), and 'README.md' (Archivo MD, 1 KB). Below the file list, the repository name 'RepositorioPrueba1' is shown with a 'Public' badge. There are buttons for 'Pin', 'Watch', and '0' notifications. The main branch is 'main', with '1 Branch' and '0 Tags' indicated. A search bar 'Go to file' and buttons 'Add file' and 'Code' are present. The commit history shows an 'Initial commit' by 'victormanbar' 3 hours ago, with a commit hash '38dcbe7'. The commit message is 'Initial commit'. Below the commit history, the 'README' file is shown with the title 'RepositorioPrueba1'.



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

```
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2> git branch
* main
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2> git push origin
main
Everything up-to-date
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2> git checkout -b
nuevaRama
Switched to a new branch 'nuevaRama'
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2> git branch
* nuevaRama
main
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2>
```




```
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2> echo "Modificac
ión hecha desde nuevaRama" > mi-archivo.txt
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2> git add .
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2> git status
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   mi-archivo.txt
```



```
PS C:\Users\vicman\OneDrive\Documentos\UTN Programación\1er Cuatrimestre\Programación I\Prácticas\Repositorios para trabajos prácticos GITHUB\UTN-TUPaD-P1.2> git push origin
nuevaRama
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 364 bytes | 364.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'nuevaRama' on GitHub by visiting:
remote:   https://github.com/victormbar/UTN-TUPaD-P1.2/pull/new/nuevaRama
remote:
To https://github.com/victormbar/UTN-TUPaD-P1.2.git
 * [new branch]      nuevaRama -> nuevaRama
```



**UTN-TUPaD-P1.2** Public



 Pin  Watch **0**










forked from [sbruselario/UTN-TUPaD-P1](#)

 **main**  **2 Branches**  **0 Tags**

 Add file  Code

This branch is **1 commit ahead of** [sbruselario/UTN-TUPaD-P1:main](#)  Contribute  Sync fork

**victormbar** Agregando mi-archivo.txt 46797dc · 11 minutes ago  **2 Commits**

 01 Estructuras Secuenciales	 ¡Bienvenid@ a Programación 1!	3 months ago
 02 Trabajo Colaborativo	Agregando mi-archivo.txt	11 minutes ago
 03 Estructuras Condicionales	 ¡Bienvenid@ a Programación 1!	3 months ago
 04 Estructuras Repetitivas	 ¡Bienvenid@ a Programación 1!	3 months ago
 05 Funciones	 ¡Bienvenid@ a Programación 1!	3 months ago

3) Realizar la siguiente actividad:

Paso 1: Crear un repositorio en GitHub

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

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
```

```
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2> git clone https://github.com/victormban/conflict-exercise
Cloning into 'conflict-exercise'...
warning: You appear to have cloned an empty repository.
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2>
```

- Entra en el directorio del repositorio:

```
cd conflict-exercise
```

```
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2\conflict-exercise>
```

Paso 3: Crear una nueva rama y editar un archivo

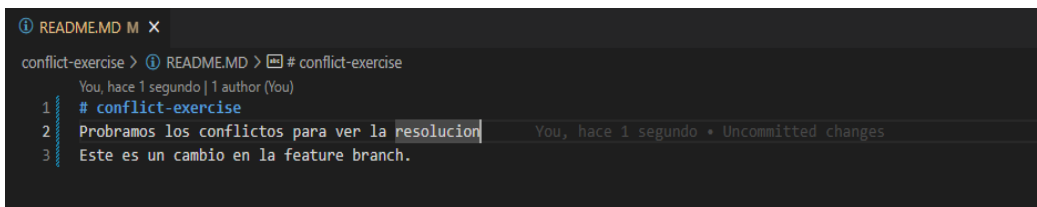
- Crea una nueva rama llamada feature-branch:

```
git checkout -b feature-branch
```

```
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2\conflict-exercise> git checkout -b feature-branch
Switched to a new branch 'feature-branch'
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2\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.



```
1 # conflict-exercise
2 Probramos los conflictos para ver la resolucion
3 Este es un cambio en la feature branch.
```

git

- Guarda los cambios y haz un commit:

```
git add README.md
```

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

```
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2\con  
flict-exercise> git commit -m "Added a line in feature-branch"  
[feature-branch 7b1211f] Added a line in feature-branch  
1 file changed, 3 insertions(+)  
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2\con  
flict-exercise> []
```

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

```
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2\con  
flict-exercise> git checkout main  
Switched to branch 'main'  
Your branch is up to date with 'origin/main'.
```

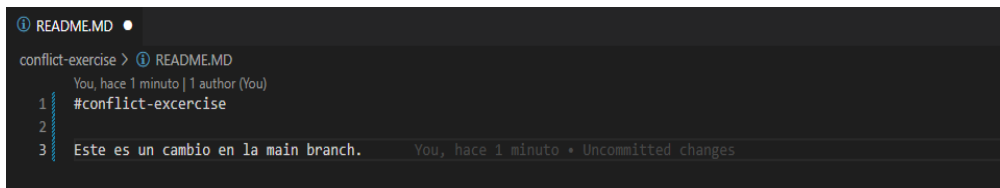
- Cambia de vuelta a la rama principal (main):

git checkout main

```
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2\conflict-exercise> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/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"

```
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2\conflict-exercise> git add README.md
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2\conflict-exercise> git commit -m "Added a line in main branch"
On branch main
Your branch is up to date with 'origin/main'.
```

Paso 5: Hacer un merge y generar un conflicto

- Intenta hacer un merge de la feature-branch en la rama main:

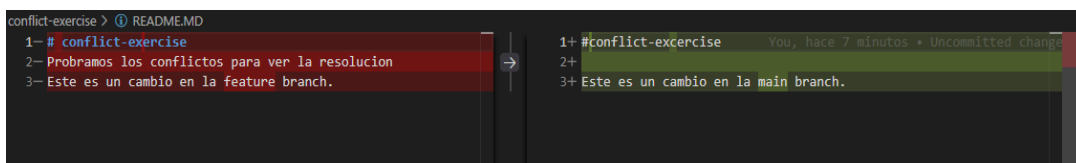
git merge feature-branch

```
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2\conflict-exercise> git merge feature-branch
Updating dif7041..7b1211f
Fast-forward
 README.md | 3 +++
 1 file changed, 3 insertions(+)
```

- 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

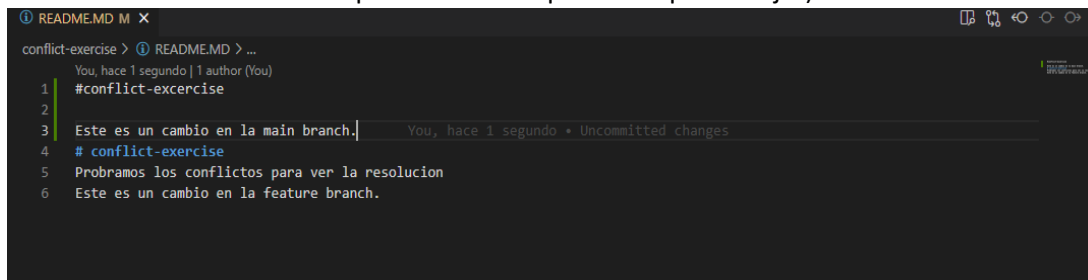
Este es un cambio en la main branch.

```
=====
```

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\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2\conflict-exercise> git add README.md
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2\conflict-exercise> git commit -m "Resolved merge conflict"
[main 5ae8443] Resolved merge conflict
1 file changed, 3 insertions(+)
```

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\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2\conflict-exercise> git push origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 591 bytes | 591.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/victorbar/conflict-exercise
d1f7041..5ae8443 main -> main
```

- También sube la feature-branch si deseas:

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

```
PS C:\Users\vcman\OneDrive\Documentos\UTN Programacion\1er Cuatrimestre\Programacion I\Practicas\Repositorios para trabajos practicos GITHUB\UTN-TUPaD-P1.2\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/victormbar/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/victormbar/conflict-exercise
 * [new branch]      feature-branch -> 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.

