

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 un repositorio remoto. Es un entorno o software que nos permite subir nuestro repositorio local para que podamos trabajar con otras personas o colaboradores sobre un mismo proyecto.

- ¿Cómo crear un repositorio en GitHub?

- Para crear un repositorio en GitHub tenemos que iniciar sesión en nuestra cuenta o crearnos una cuenta en GitHub, hacer clic en “crear nuevo repositorio”, agregarle un nombre y guardar. Luego ahí subimos archivos o los creamos.

- Si tenemos un repositorio local que queremos subir a GitHub lo que tenemos que hacer es iniciar sesión en GitHub, hacer clic en “crear nuevo repositorio”, agregarle un nombre, y al final de la página aparece una sección que dice “...or push an existing repository...”.

En esta sección tenemos 3 líneas de código, debemos copiar la primera línea y pegarla en nuestro git bash, y luego copiamos la tercera línea y la pegamos en nuestro git bash.

Cuando hagamos eso se nos va a abrir una ventana emergente para conectar con nuestra cuenta de GitHub. Una vez conectada la cuenta se nos va a clonar todo nuestro repositorio local en nuestro repositorio remoto.

- ¿Cómo crear una rama en Git?

Para crear una rama en Git debemos ejecutar en nuestro git bash el comando **git branch nombreDeRama**.

Es decir, git branch y el nombre de la nueva rama que queremos crear.

- ¿Cómo cambiar a una rama en Git?

Para cambiar de una rama a otra debemos usar el comando **git checkout nombreDeRama**.

Es decir, usamos git checkout y el nombre de la rama a la que queremos movernos.

- ¿Cómo fusionar ramas en Git?

Para fusionar ramas en Git debemos posicionarnos en la rama en la que queremos fusionar la otra rama, y usar el comando **git merge nombreDeRama**.

Es decir, usamos el comando git merge y el nombre de la rama que queremos fusionar a la rama en la que estamos posicionados actualmente.

- ¿Cómo crear un commit en Git?

Para crear un commit en git primero debemos usar el comando **git init**, para iniciar o crear el repositorio.

Luego usamos **git add** . para guardar todos los cambios y archivos en el repositorio, y ahora el comando **git commit -m ""** para crear el commit.

Dentro de las comillas debemos colocar un título o descriptor del commit o cambios que estamos commiteando.

- ¿Cómo enviar un commit a GitHub?

Para enviar un commit a GitHub se debe utilizar el comando **git push -u origin master**.

- ¿Qué es un repositorio remoto?

Es un entorno o software que contiene una copia clonada de nuestro repositorio local. Tener esto nos permite trabajar en nuestro proyecto desde cualquier lado y cualquier computadora, y nos permite trabajar de forma colaborativa con otras personas.

- ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto a git se agrega en Git Bash el comando **git remote add** y seguido de eso colocamos el nombre del repositorio, y separado por un espacio colocamos la URL del repositorio.

Para bajarnos de nuestro repositorio remoto a nuestro repositorio local usamos **git clone** (esto aplica cuando tenemos un repositorio remoto nuevo, que no tenemos creado en local).

- ¿Cómo empujar cambios a un repositorio remoto?

Para empujar cambios a un repositorio remoto, luego de los comandos **git add .**, **git commit -m ""**, se usa el comando **git push -u origin master**.

- ¿Cómo tirar de cambios de un repositorio remoto?

Para traer los cambios de un repositorio remoto se usa el comando **git pull**

- ¿Qué es un fork de repositorio?

Es una copia de un repositorio creada en una cuenta diferente permitiendo desarrollar cambios sin afectar el original.

A diferencia del clonado, que descarga el repositorio localmente, el fork se realiza generando una copia en la cuenta del usuario

- ¿Cómo crear un fork de un repositorio?

Para crear un fork de un repositorio ingreso a ese repositorio que quiero copiar y selecciono la opción **"fork"**.

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

Para enviar una solicitud de extracción primero debo realizar un Fork de un repositorio. Luego de realizar cambios en los archivos y guardarlos, voy al repositorio que copié y hago clic en “Compare & pull request”.

Vamos a poder agregar un mensaje o descripción de los cambios realizados antes de enviar la solicitud de pull request.

- ¿Cómo aceptar una solicitud de extracción?

El autor del repositorio verá en sus pull request el mensaje que le enviamos y los cambios que realizamos en el proyecto. Si el autor quiere agregar esas modificaciones a su repositorio hace clic en Merge pull request.

- ¿Qué es un etiqueta en Git?

Una etiqueta es una referencia que marca un punto específico en el historial de Git.

- ¿Cómo crear una etiqueta en Git?

Existen dos tipos de etiquetas: Ligeras y anotadas.

Para crear una etiqueta ligera en git debemos abrir git bash, ubicarnos en nuestro repositorio y usar el comando **git tag nombre-etiqueta**. Es útil para marcar commits rápidamente.

Para crear una etiqueta uso el comando **git tag -a nombre-etiqueta -m "Mensaje descriptivo"**. Este tipo de etiquetas incluye autor, fecha y mensaje (igual que un commit).

- ¿Cómo enviar una etiqueta a GitHub?

Para subir todas las etiquetas uso el comando **git push origin --tags**, y para subir una etiqueta específica uso el comando **git push origin nombre-etiqueta**.

- ¿Qué es un historial de Git?

El historial de Git es un registro cronológico de todos los cambios (commits) realizados en un repositorio.

- ¿Cómo ver el historial de Git?

Para ver el historial de git uso el comando **git log** (muestra autor del commit, fecha y hash o ID del commit) o **git log --online** (muestra solo el ID o Hash del commit y el mensaje).

- ¿Cómo buscar en el historial de Git?

Existen diversas maneras de buscar cambios en git:

→ Para buscar un commit por un mensaje se usa el comando **git log --grep="mensaje"**.

→ Para ver los commits de un archivo en específico se usa el comando **git log --nombre del archivo**.

→ Para buscar commits de un autor en específico se usa el comando **git log --author="nombre"**

→ Para buscar por fecha:

- Commits después de una fecha: **git log --since="año-mes-día"**

- Commits antes de una fecha: **git log --until="año-mes-día"**

- Commits entre un rango de fecha: **git log --since="año-mes-día" --until="año-mes-día"**

- ¿Cómo borrar el historial de Git?

Para eliminar un commit, por ejemplo, tengo 5 commits y quiero eliminar los últimos 2, lo que debo hacer técnicamente es agarrar el puntero y colocarlo dos commits antes, así se eliminarían los 2 últimos commits.

Tenemos 3 formas de utilizar git reset: soft - mixed- hard.

--soft: Los cambios quedan en staging.

--mixed (default): Los cambios quedan en working directory.

--hard: Borra todo (commits + cambios locales).

Comando: **git reset --soft HEAD~N # N = número de commits a borrar (ej: HEAD~3)**

- ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es un proyecto cuyo código y contenido solo son accesibles para usuarios específicos que hayas autorizado. Ofrecen control total sobre quién puede ver, clonar o modificar el código.

- ¿Cómo crear un repositorio privado en GitHub?

Para crear un repositorio privado en GitHub iniciamos sesión en nuestra cuenta de GitHub, hacemos clic en “nuevo repositorio” y nos va a aparecer la opción de “privado” o “público”.

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

Para invitar a alguien a un repositorio privado en GitHub debo ingresar al repositorio, hacer clic en “configuraciones” y en “colaboradores”. Ahí nos va a pedir

el nombre de usuario o email del colaborador que queremos agregar, y además vamos a poder elegir el tipo de permiso: Read (solo lectura), Write (puede editar y hacer push) y Admin (permisos completos).

- ¿Qué es un repositorio público en GitHub?

Un repositorio público o en GitHub es un proyecto cuyo código y contenido es visible para todo internet.

- ¿Cómo crear un repositorio público en GitHub?

Para crear un repositorio privado en GitHub iniciamos sesión en nuestra cuenta de GitHub, hacemos clic en “nuevo repositorio” y nos va a aparecer la opción de “privado” o “público”.

- ¿Cómo compartir un repositorio público en GitHub?

Puedo compartir un repositorio público copiando la URL del repositorio y pegándolo en donde quiero compartirlo.

2) Realizar la siguiente actividad:

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

```
MINGW64:/d/nuevo-repo

yane0@Yanela MINGW64 /d
$ mkdir nuevo-repo

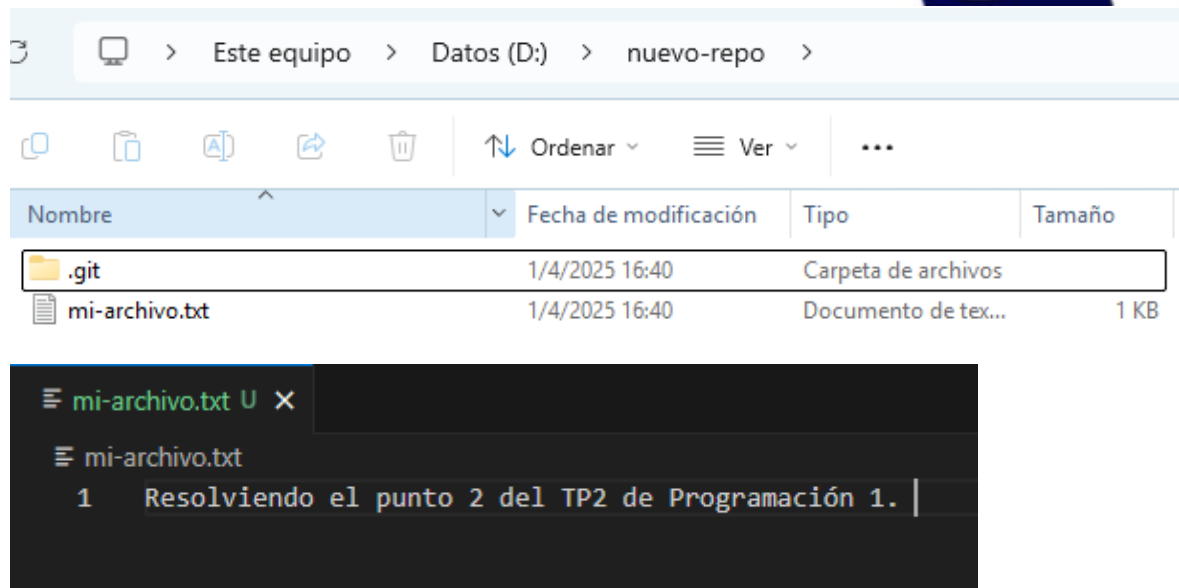
yane0@Yanela MINGW64 /d
$ cd nuevo-repo

yane0@Yanela MINGW64 /d/nuevo-repo
$ git init
Initialized empty Git repository in D:/nuevo-repo/.git/

yane0@Yanela MINGW64 /d/nuevo-repo (master)
$ touch mi-archivo.txt

yane0@Yanela MINGW64 /d/nuevo-repo (master)
$ code .

yane0@Yanela MINGW64 /d/nuevo-repo (master)
$ |
```



- Agregando un Archivo
 - o Crea un archivo simple, por ejemplo, "mi-archivo.txt".
 - o Realiza los comandos `git add .` y `git commit -m "Agregando mi-archivo.txt"` en la línea de comandos.
 - o Sube los cambios al repositorio en GitHub con `git push origin main` (o el nombre de la rama correspondiente).

```
yane0@Yanela MINGW64 /d/nuevo-repo (master)
$ git add .

yane0@Yanela MINGW64 /d/nuevo-repo (master)
$ git commit -m "Agregando mi-archivo.txt" -a
[master (root-commit) 91f2c] Agregando mi-archivo.txt
1 file changed, 1 insertion(+)
 create mode 100644 mi-archivo.txt

yane0@Yanela MINGW64 /d/nuevo-repo (master)
```


→ Creamos repo remoto y subimos el repo local

Crear un nuevo repositorio

Un repositorio contiene todos los archivos del proyecto, incluido el historial de revisiones. ¿Ya tienes un repositorio de proyectos en otro lugar? [Importa un repositorio.](#)

Los campos obligatorios están marcados con un asterisco (*).

Dueño *

 YaneMub

Nombre del repositorio *

nuevo-repo

✓ nuevo-repo está disponible.

Los buenos nombres de repositorios son cortos y fáciles de recordar. ¿Necesitas inspiración? ¿Qué te parece? **brócoli plateado** ?

Descripción (opcional)

Este repositorio es creado para resolver el punto 2 del TP 2.



Público

Cualquier persona en internet puede ver este repositorio. Tú decides quién puede contribuir.



Privado

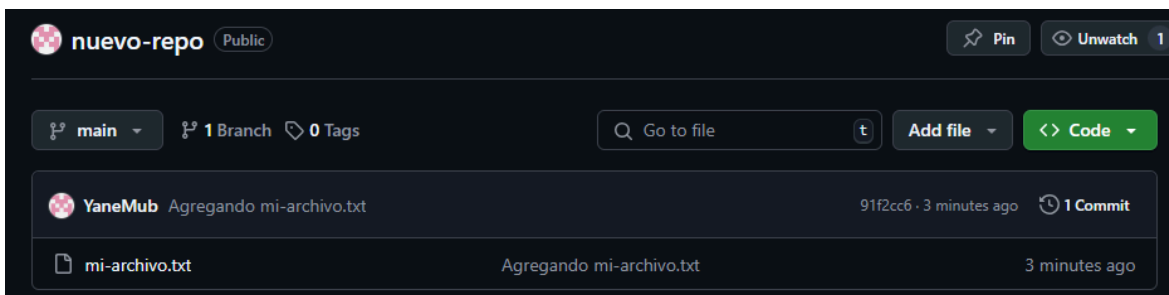
Tú eliges quién puede ver y comprometerse con este repositorio.

```
yane0@Yanela MINGW64 /d/nuevo-repo (master)
$ git remote add origin https://github.com/YaneMub/nuevo-repo.git

yane0@Yanela MINGW64 /d/nuevo-repo (master)
$ git branch -M main

yane0@Yanela MINGW64 /d/nuevo-repo (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 280 bytes | 280.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/YaneMub/nuevo-repo.git
* [new branch] main -> main
branch 'main' set up to track 'origin/main'.

yane0@Yanela MINGW64 /d/nuevo-repo (main)
$
```



The screenshot shows the GitHub interface for a repository named 'nuevo-repo' owned by 'YaneMub'. The repository is marked as 'Public'. At the top, there are buttons for 'Pin', 'Unwatch', and a notification badge showing '1'. Below this, the repository details show 'main' as the selected branch, '1 Branch', and '0 Tags'. There is a search bar labeled 'Go to file' and buttons for 'Add file' and 'Code'. The commit history shows a single commit by 'YaneMub' titled 'Agregando mi-archivo.txt' with the hash '91f2cc6' and a timestamp of '3 minutes ago'. Below the commit, a file named 'mi-archivo.txt' is listed with the description 'Agregando mi-archivo.txt' and a timestamp of '3 minutes ago'.

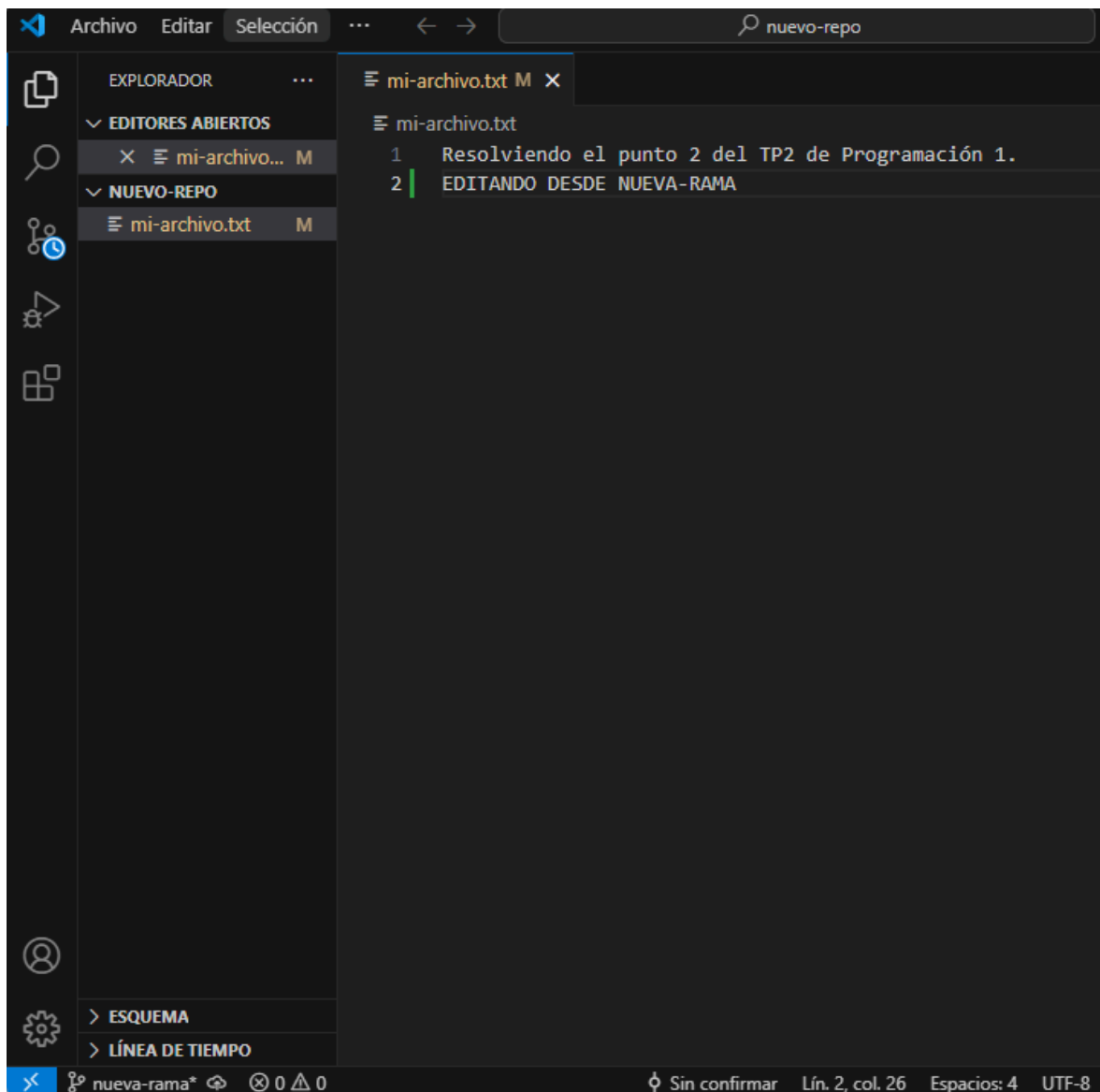
- Creando Branchs
 - o Crear una Branch

```
yane0@Yanela MINGW64 /d/nuevo-repo (main)
$ git branch nueva-rama

yane0@Yanela MINGW64 /d/nuevo-repo (main)
$ git checkout nueva-rama
Switched to branch 'nueva-rama'

yane0@Yanela MINGW64 /d/nuevo-repo (nueva-rama)
$ git branch
  main
* nueva-rama
```

- o Realizar cambios o agregar un archivo



- o Subir la Branch

```
yane0@Yanela MINGW64 /d/nuevo-repo (nueva-rama)
$ git add .
git
yane0@Yanela MINGW64 /d/nuevo-repo (nueva-rama)
$ git commit -m "Modificando desde nueva-rama" -a
[nueva-rama 81614] Modificando desde nueva-rama
1 file changed, 2 insertions(+), 1 deletion(-)

yane0@Yanela MINGW64 /d/nuevo-repo (nueva-rama)
$ git checkout master
error: pathspec 'master' did not match any file(s) known to git


yane0@Yanela MINGW64 /d/nuevo-repo (nueva-rama)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.


yane0@Yanela MINGW64 /d/nuevo-repo (main)
$ git branch
* main
  nueva-rama
```

```
yane0@Yanela MINGW64 /d/nuevo-repo (main)
$ git merge nueva-rama
Updating 91f2c..81614
Fast-forward
 mi-archivo.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

```
yane0@Yanela MINGW64 /d/nuevo-repo (main)
$ git push -u origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 339 bytes | 339.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/YaneMub/nuevo-repo.git
 91f2c..81614 main -> main
branch 'main' set up to track 'origin/main'.
```

nuevo-repo / mi-archivo.txt

 **YaneMub** Modificando desde nueva-rama

Code Blame 2 lines (2 loc) · 76 Bytes  Code 55% faster with GitHub Copilot

```
1 Resolviendo el punto 2 del TP2 de Programación 1.
2 EDITANDO DESDE NUEVA-RAMA
```

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".

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner * **Repository name ***


 YaneMub / conflict-exercise


✔ conflict-exercise is available.

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

Description (optional)

Este repositorio es creado para resolver el punto 3 del TP 2.

☒  **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.

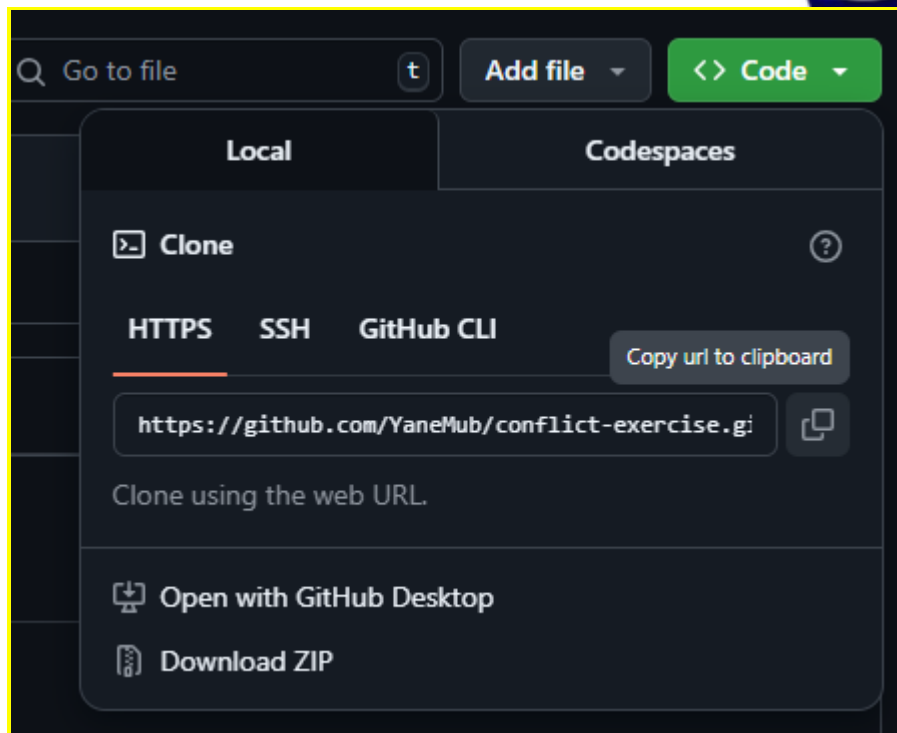
Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

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:/d/nuevo-repo/conflict-exercise

yane0@Yanela MINGW64 /d/nuevo-repo (main)
$ git clone https://github.com/YaneMub/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.

yane0@Yanela MINGW64 /d/nuevo-repo (main)
$ ls
conflict-exercise/  mi-archivo.txt

yane0@Yanela MINGW64 /d/nuevo-repo (main)
$ cd conflict-exercise

yane0@Yanela MINGW64 /d/nuevo-repo/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`

```
yane0@Yanela MINGW64 /d/nuevo-repo/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

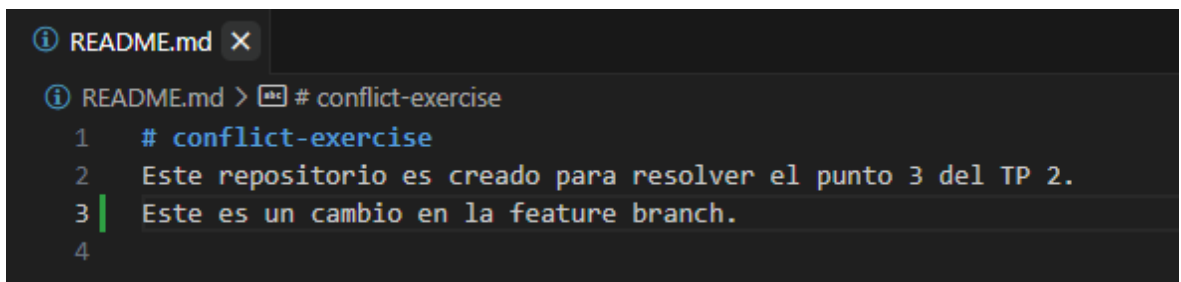
yane0@Yanela MINGW64 /d/nuevo-repo/conflict-exercise (feature-branch)
$ git branch
* feature-branch
  main

yane0@Yanela MINGW64 /d/nuevo-repo/conflict-exercise (feature-branch)
$ |
```

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

Este es un cambio en la feature branch.

```
yane0@Yanela MINGW64 /d/nuevo-repo/conflict-exercise (feature-branch)
$ code .
```



```
① README.md X
① README.md > # conflict-exercise
1 # conflict-exercise
2 Este repositorio es creado para resolver el punto 3 del TP 2.
3 Este es un cambio en la feature branch.
4
```

- Guarda los cambios y haz un commit:

```
git add README.md
```

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

```
yane0@Yanela MINGW64 /d/nuevo-repo/conflict-exercise (feature-branch)
$ git add README.md

yane0@Yanela MINGW64 /d/nuevo-repo/conflict-exercise (feature-branch)
$ git commit -m "Added a line in feature-branch"
[feature-branch 29dc5] Added a line in feature-branch
1 file changed, 1 insertion(+)
```

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

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

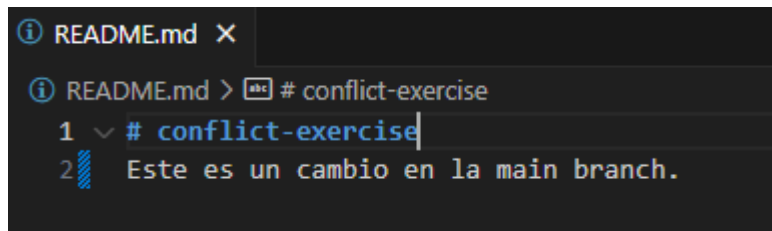
```
git checkout main
```

```
yane0@Yanela MINGW64 /d/nuevo-repo/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

yane0@Yanela MINGW64 /d/nuevo-repo/conflict-exercise (main)
$ git branch
  feature-branch
* main
```

- Edita el archivo README.md de nuevo, añadiendo una línea diferente:

Este es un cambio en la main branch.



```
① README.md X
① README.md > # conflict-exercise
1  # conflict-exercise
2  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"
```

```
yane0@Yanela MINGW64 /d/nuevo-repo/conflict-exercise (main)
$ git add README.md

yane0@Yanela MINGW64 /d/nuevo-repo/conflict-exercise (main)
$ git commit -m "Added a line in main branch" -a
[main 9eef3] Added a line in main branch
1 file changed, 1 insertion(+), 1 deletion(-)
```

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.

```
yane0@Yanela MINGW64 /d/nuevo-repo/conflict-exercise (main)
$ git merge feature-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

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:

```
README.md ! X
README.md > # <<<<<<< HEADEste es un cambio en la main branch.
1 # conflict-exercise
  Aceptar cambio actual | Aceptar cambio entrante | Aceptar ambos cambios | Comparar cambios
2 <<<<<<< HEAD (Cambio actual)
3 Este es un cambio en la main branch.
4 =====
5 Este repositorio es creado para resolver el punto 3 del TP 2.
6 Este es un cambio en la feature branch.
7 >>>>>>> feature-branch (Cambio entrante)
8
```

```
README.md ! X
README.md > # conflict-exercise
1 # conflict-exercise
2 Este repositorio es creado para resolver el punto 3 del TP 2.
3 Este es un cambio en la feature branch y main.
4
5
```

```
git add README.md
```

```
git commit -m "Resolved merge conflict"
```

```
yane0@Yanela MINGW64 /d/nuevo-repo/conflict-exercise (main|MERGING)
$ git add README.md

yane0@Yanela MINGW64 /d/nuevo-repo/conflict-exercise (main|MERGING)
$ git commit -m "Resolved merge conflict" -a
[main f5b79] 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`

```
yane0@Yanela MINGW64 /d/nuevo-repo/conflict-exercise (main)
$ 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), 853 bytes | 853.00 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/YaneMub/conflict-exercise.git
 84cc1..f5b79  main -> main
```

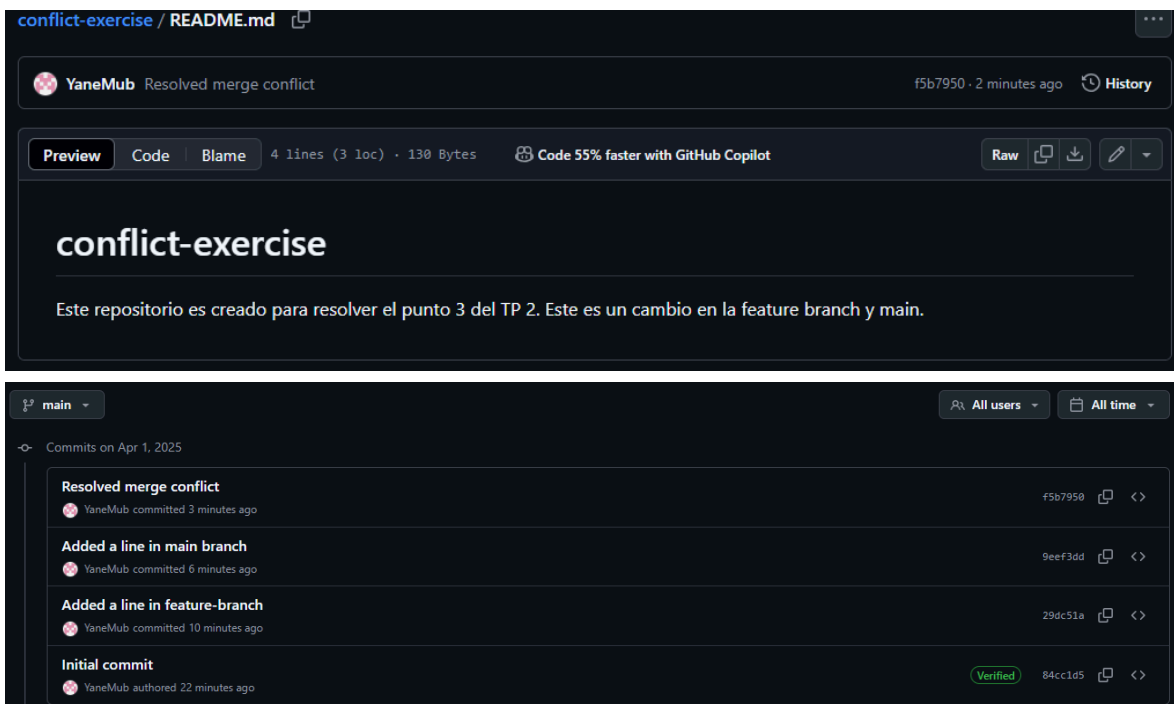

- También sube la feature-branch si deseas:

`git push origin feature-branch`

```
yane0@Yanela MINGW64 /d/nuevo-repo/conflict-exercise (main)
$ git push origin feature-branch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:   https://github.com/YaneMub/conflict-exercise/pull/new/feature-branch
remote:
remote:
To https://github.com/YaneMub/conflict-exercise.git
 * [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.



The screenshot shows the GitHub interface for the repository 'conflict-exercise'. The top section displays the README.md file, which contains the text: 'Este repositorio es creado para resolver el punto 3 del TP 2. Este es un cambio en la feature branch y main.' Below the README, the commit history is visible, showing four commits: 'Resolved merge conflict' (f5b7950), 'Added a line in main branch' (9eef3dd), 'Added a line in feature-branch' (29dc51a), and 'Initial commit' (84cc1d5). The 'Initial commit' is marked as 'Verified'.

