

Eduardo Adrián Lopez comisión 16

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 para alojar repositorios.

- ¿Cómo crear un repositorio en GitHub?

1-Tenes que loguearte o crear una cuenta. 2-ir “new repository”. 3-Repository name :escribe un nombre para tu repositorio

4-Description(opcional) Agregar una breve descripción del proyecto

5- visibility(para poner el repositorio en publico o privado): public : cualquiera puede ver el código. private: solo tú y las personas que invites pueden acceder.

6-opcionalmente, puedes: incluir
un archivo README

Agregar un .gitignore(para excluir archivos innecesarios según lenguaje).

Elegir una licencia para definir los permisos del código.

7-crear el repositorio: hacinedo click en el
botón “Create repository”.

- ¿Cómo crear una rama en Git?

Con
“git branch ” (nombre de la rama)”

- ¿Cómo cambiar a una rama en Git?

Con “ git checkout (nombre de la rama) “ o si usas git 2.23
“git switch (nombre de la rama)”

- ¿Cómo fusionar ramas en Git?

Con “git merge(nombre de la rama)”

- ¿Cómo crear un commit en Git?

1- Git add (nombre del archivo) o git add. (“enviar todo el contenido del
proyecto”)

2-Con “ git commit –m “(descripción de lo que vas a subir)”

- ¿Cómo enviar un commit a GitHub? haces los pasos de la pregunta anterior mas
este: 1- git push origin (“nombre de la rama”)

2- Verificar en GitHub

- ¿Qué es un repositorio remoto?

Son copias de un repositorio de Git que está alojado en un servidor en línea o una
computadora. Permite que múltiples desarrolladores colaboren en un mismo
proyecto, sincronizando sus cambios através de comandos de Git.

- ¿Cómo agregar un repositorio remoto a Git?

con git remote add origin URL-del-repositorio ejemplo: git
remote add origin https://github.com/usuario/proyecto.git

- ¿Cómo empujar cambios a un repositorio remoto?

Haciendo un commit primero,
con git push origin ("nombre de la rama") . empujas los cambios ejemplo
:

```
git push origin main.
```

- ¿Cómo tirar de cambios de un repositorio remoto?

Primero verificas si estas parado en la rama que quieres traer los cambios. con
git pull origin ("nombre de la rama") te traes los cambios de dicha rama.

- ¿Qué es un fork de repositorio?

Es una copia de un repositorio de Git que se crea en tu cuenta.

- ¿Cómo crear un fork de un repositorio?

- 1- Ir a la página del repositorio original en GitHub.
- 2- Hacer click en "Fork": que se encuentra en la parte superior derecha del repositorio encontraras el boton "Fork", despliega un sub menú con
- 3- "+ Create a new for"
- 4- Donde te llevara a otra pantalla para personalizar tu copia
- 5- Hazclick en "create for"y te creara una copia en tu GitHub.

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

1-Primero tienes que subir tus cambios en el repositorio donde hiciste el "fork". 2-ir a la pagina del repositorio original hacer click en la solapa "Pull requests"

3-hacer click en new pull request.

4- configurar el pull request: base: selecciona la rama principal del repositorio oriinal (generalmente main o master).

Compare: slecciona la rama de tu repositorio forked que contiene los cambios(la rama que acabas de subir).

6- Añadir detalles

7- Hacer clicl en "Create pull request" para enviar tu solicitud de extracción.

8- Si todo esta bien se hace merge con tu rama o si hay nuevos comentarios o correcciones podras hacer nuevos commits en tu rama y el pull request se actualizará automáticamente.

- ¿Cómo aceptar una solicitud de extracción?

es lo mismo que el paso anterior, te logueas en tu cuenta, entras al repositorio donde quieres aceptar la pull request y entras a la solapa del Pull request, visualizas todas las pull request, entras en una revisas y hace comentarios (si es necesario).hacers pruebas en local y si todo esta ok haces merge(aceptando la solicitud de extracción).

- ¿Qué es un etiqueta en Git?

Son puntero a un commit específico. Contiene nombre y la referencia al commit

Se usan para marcar puntos de interés en el historial del repositorio de maera rápida y sencilla.

- ¿Cómo crear una etiqueta en Git?

`git tag nombre-etiqueta`

`git tag -a nombre-etiqueta -m "Mensaje de la etiqueta"`

ejemplo : `git tag -a v1.0 -m`

"Primera versión estable"

- ¿Cómo enviar una etiqueta a GitHub?

Espesifica : `git push origin nombre-etiqueta todas`
: `git push --tags`

- ¿Qué es un historial de Git?

Es un registro completo de todos los cambios realizados en un repositorio.

Permite raestrar el progreso y evolución de un proyecto ver que cambios se han realizado, quién los realizó y cuándo

- ¿Cómo ver el historial de Git?

Se puede ver con `git log`

- ¿Cómo buscar en el historial de Git?

Algunos parámetros útiles son: `git log --online` muestra un historial compacto

`git log --graph` : muestra un historial de forma de un grafico ramas y funciones

`git log --author="nombre"` : filtrar los commits por autor específico

git reflog : historial referencias y cambios que se han hecho en el repositorios incluyendo actualizaciones en las ramas

- ¿Cómo borrar el historial de Git?

Reescribir el historial: Usa git rebase o git reset para eliminar commits o restablecer el historial.

Borrar completamente el historial: Crea una nueva rama y realiza un commit vacío para empezar de cero.

Eliminar un archivo específico del historial: Usa git filter-branch o herramientas como BFG Repo-Cleaner.

- ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es un repositorio cuyo acceso está restringido. Solo las personas que tienen permisos específicos pueden ver, clonar o modificar los archivos dentro de ese repositorio.

- ¿Cómo crear un repositorio privado en GitHub?

1- Inicia sesión en GitHub: Ve a GitHub y accede con tu cuenta

2- Crear un nuevo repositorio: En tu página de inicio, haz clic en el botón "New" o en "Crear un repositorio" en el menú superior 3- Configura el repositorio: Escribe el nombre del repositorio.

Marca la opción "Private" para hacerlo privado.

Si lo deseas, puedes agregar una descripción, un archivo README inicial, un archivo .gitignore, o una licencia.

4- Crear el repositorio: Haz clic en "Create repository"

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

1-Ve a la configuración del repositorio: Abre tu repositorio y ve a la pestaña "Settings".

2-Selecciona "Manage access": En la sección "Access", encontrarás la opción para gestionar el acceso a tu repositorio. Invitar colaboradores: Haz clic en "Invite a collaborator".

3-Busca a la persona que deseas invitar por su nombre de usuario o correo electrónico y selecciona su cuenta. Decide el nivel de acceso (lectura, escritura o administrador) y envíales la invitación.

4-Confirmación: La persona recibirá una invitación por correo electrónico para unirse al repositorio y podrá aceptar para acceder a él.

- ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un repositorio cuyo contenido está abierto y accesible para cualquiera. Cualquier persona, incluso sin una cuenta en GitHub, puede ver, clonar, bifurcar (hacer un fork) y contribuir a un repositorio público. Este tipo de repositorio es útil cuando deseas que tu código o proyecto sea accesible para la comunidad, como en proyectos de código abierto, colaboraciones o cuando quieres compartir tu trabajo con el mundo.

- ¿Cómo crear un repositorio público en GitHub?

1-Inicia sesión en GitHub: Accede a GitHub con tu cuenta.

2-Crear un nuevo repositorio: Haz clic en el botón "New" o en "Crear un repositorio" en la página de inicio de GitHub.

3: Configura el repositorio: Nombre del repositorio: Asigna un nombre a tu repositorio. Descripción (opcional): Agrega una descripción de lo que trata el proyecto. Visibilidad: Marca la opción "Public" para hacerlo público. (Opcional) Puedes agregar un archivo README, un archivo .gitignore, o elegir una licencia para tu proyecto.

4-Crear el repositorio: Haz clic en el botón "Create repository" y tu repositorio será creado como público.

- ¿Cómo compartir un repositorio público en GitHub?

Cualquiera puede clonar un repositorio público (usando la url del proyecto) en su máquina local usando el comando: ejemplo : git clone <https://github.com/usuario/nombre-repositorio.git> 2) Realizar la siguiente actividad:

- Crear un repositorio.
 - Dale un nombre al repositorio. ○ Elije el repositorio sea público.
https://github.com/eduardoadrian1994/practicaGitUTN_1
 - Inicializa el repositorio con un archivo.

```
MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestr
e/PROGRAMACION I/practica_git
$ git clone https://github.com/eduardoadrian1994/practicaGitUTN_1.git
Cloning into 'practicaGitUTN_1'...
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.

MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestr
e/PROGRAMACION I/practica_git
$ git init
Initialized empty Git repository in C:/Users/DY38684017/Desktop/utn-a distancia-
tecnatura/primer semestre/PROGRAMACION I/practica_git/.git/

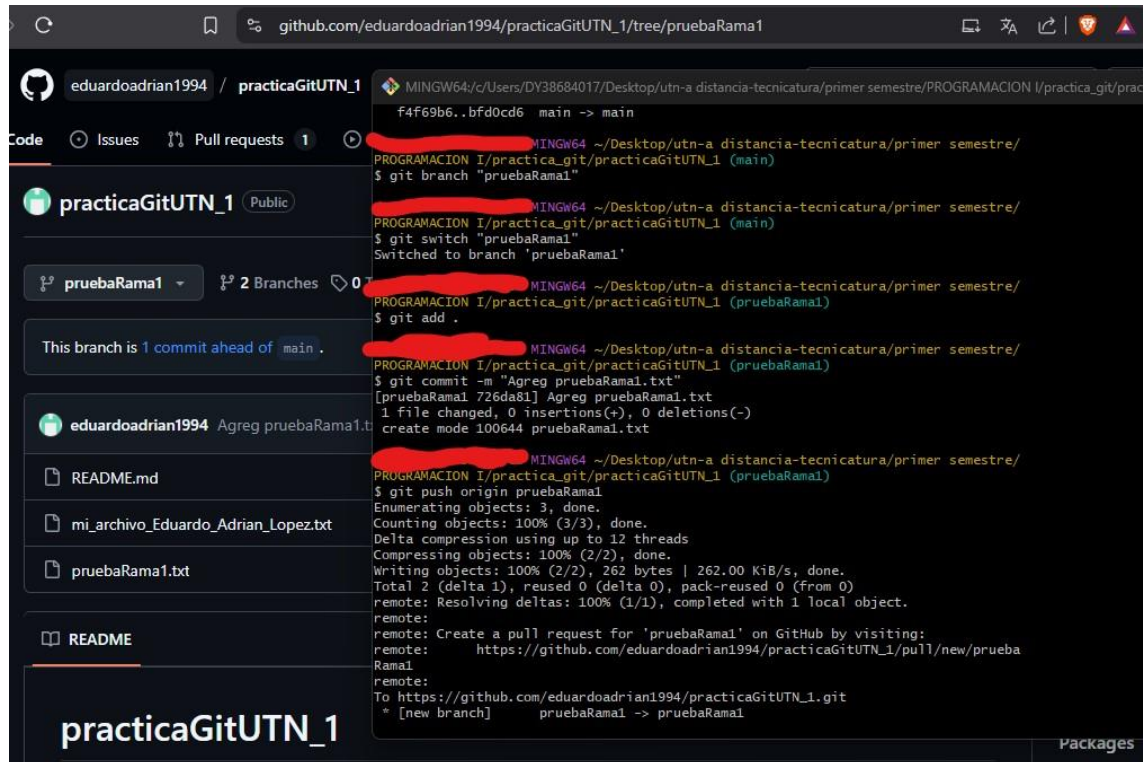
MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestr
e/PROGRAMACION I/practica_git (master)
$ |
```

git

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

```
MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestre/  
PROGRAMACION I/practica_git/practicaGitUTN_1 (main)  
$ git add .  
  
MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestre/  
PROGRAMACION I/practica_git/practicaGitUTN_1 (main)  
$ git commit -m "Agregando el mi_archivo_Eduardo_Adrian_Lopez.txt"  
[main bfd0cd6] Agregando el mi_archivo_Eduardo_Adrian_Lopez.txt  
1 file changed, 0 insertions(+), 0 deletions(-)  
create mode 100644 mi_archivo_Eduardo_Adrian_Lopez.txt  
  
MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestre/  
PROGRAMACION I/practica_git/practicaGitUTN_1 (main)  
$ git push origin main  
info: please complete authentication in your browser...  
Enumerating objects: 4, done.  
Counting objects: 100% (4/4), done.  
Delta compression using up to 12 threads  
Compressing objects: 100% (2/2), done.  
Writing objects: 100% (3/3), 338 bytes | 338.00 KiB/s, done.  
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)  
To https://github.com/eduardoadrian1994/practicaGitUTN_1.git  
f4f69b6..bfd0cd6 main -> main
```

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



```
MINGW64/c:/Users/DY38684017/Desktop/utn-a distancia-tecnatura/primer semestre/PROGRAMACION I/practica_git/prac
f4f69b6..bfd0cd6 main -> main
PROGRAMACION I/practica_git/practicaGitUTN_1 (main)
$ git branch "pruebaRama1"
PROGRAMACION I/practica_git/practicaGitUTN_1 (main)
$ git switch "pruebaRama1"
Switched to branch 'pruebaRama1'
PROGRAMACION I/practica_git/practicaGitUTN_1 (pruebaRama1)
$ git add .
PROGRAMACION I/practica_git/practicaGitUTN_1 (pruebaRama1)
$ git commit -m "Agreg pruebaRama1.txt"
[pruebaRama1 726da81] Agreg pruebaRama1.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 pruebaRama1.txt
PROGRAMACION I/practica_git/practicaGitUTN_1 (pruebaRama1)
$ git push origin pruebaRama1
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 262 bytes | 262.00 KiB/s, done.
Total 2 (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 'pruebaRama1' on GitHub by visiting:
remote:   https://github.com/eduardoadrian1994/practicaGitUTN_1/pull/new/prueba
remote:
remote: To https://github.com/eduardoadrian1994/practicaGitUTN_1.git
* [new branch] pruebaRama1 -> pruebaRama1
```

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` el mio :

`https://github.com/eduardoadrian1994/conflicto-ejercicio-UTN`

- Entra en el directorio del repositorio: `cd conflict-exercise`

```
MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestr
e/PROGRAMACION I/practica_git_Conflicto
$ git init
Initialized empty Git repository in C:/Users/DY38684017/Desktop/utn-a distancia-
tecnatura/primer semestre/PROGRAMACION I/practica_git_Conflicto/.git/

MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestr
e/PROGRAMACION I/practica_git_Conflicto (master)
$ git clone https://github.com/eduardoadrian1994/conflicto-ejercicio-UTN.git
Cloning into 'conflicto-ejercicio-UTN'...
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.

MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestr
e/PROGRAMACION I/practica_git_Conflicto (master)
$ cd conflicto-ejercicio-UTN

MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestre/
PROGRAMACION I/practica_git_Conflicto/conflicto-ejercicio-UTN (main)
$
```

Paso 3: Crear una nueva rama y editar un archivo

- Crea una nueva rama llamada feature-branch:

`git checkout -b feature-branch`

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

Este es un cambio en la feature branch.

- Guarda los cambios y haz un commit: `git add README.md` `git commit -m`

`"Added a line in feature-branch"`

```
MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestre/  
PROGRAMACION I/practica_git_Conflicto/conflicto-ejercicio-UTN (main)  
$ git branch "feature-branch"  
  
MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestre/  
PROGRAMACION I/practica_git_Conflicto/conflicto-ejercicio-UTN (main)  
$ git switch "feature-branch"  
Switched to branch 'feature-branch'  
  
MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestre/  
PROGRAMACION I/practica_git_Conflicto/conflicto-ejercicio-UTN (feature-branch)  
$ git add README.md  
  
MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestre/  
PROGRAMACION I/practica_git_Conflicto/conflicto-ejercicio-UTN (feature-branch)  
$ git commit -m "Added a line in feature-branch"  
[feature-branch c63da13] Added a line in feature-branch  
1 file changed, 1 insertion(+)  
  
MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestre/  
PROGRAMACION I/practica_git_Conflicto/conflicto-ejercicio-UTN (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"`

```
MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestre/  
PROGRAMACION I/practica_git_Conflicto/conflicto-ejercicio-UTN (feature-branch)  
$ git checkout main  
Switched to branch 'main'  
Your branch is up to date with 'origin/main'.  
  
MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestre/  
PROGRAMACION I/practica_git_Conflicto/conflicto-ejercicio-UTN (main)  
$ git add README.md  
  
MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestre/  
PROGRAMACION I/practica_git_Conflicto/conflicto-ejercicio-UTN (main)  
$ git commit -m "Added a line in main branch"  
[main 32155af] Added a line in main branch  
1 file changed, 1 insertion(+)  
  
MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestre/  
PROGRAMACION I/practica_git_Conflicto/conflicto-ejercicio-UTN (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`
- 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"
```

```
MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestre/  
PROGRAMACION I/practica_git_Conflicto/conflicto-ejercicio-UTN (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.  
  
MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestre/  
PROGRAMACION I/practica_git_Conflicto/conflicto-ejercicio-UTN (main|MERGING)  
$ git add README.md  
  
MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestre/  
PROGRAMACION I/practica_git_Conflicto/conflicto-ejercicio-UTN (main|MERGING)  
$ git commit -m "Resolved merge conflict"  
[main 62e399c] Resolved merge conflict  
  
MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestre/  
PROGRAMACION I/practica_git_Conflicto/conflicto-ejercicio-UTN (main)  
$
```

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
```

```
MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestre/  
PROGRAMACION I/practica_git_Conflicto/conflicto-ejercicio-UTN (main)  
$ git push origin main  
Enumerating objects: 14, done.  
Counting objects: 100% (14/14), done.  
Delta compression using up to 12 threads  
Compressing objects: 100% (8/8), done.  
Writing objects: 100% (12/12), 1.02 KiB | 1.02 MiB/s, done.  
Total 12 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)  
remote: Resolving deltas: 100% (4/4), done.  
To https://github.com/eduardoadrian1994/conflicto-ejercicio-UTN.git  
    adelc15..62e399c main -> main  
  
MINGW64 ~/Desktop/utn-a distancia-tecnatura/primer semestre/  
PROGRAMACION I/practica_git_Conflicto/conflicto-ejercicio-UTN (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/eduardoadrian1994/conflicto-ejercicio-UTN/pull/new  
remote:   /feature-branch  
remote:  
To https://github.com/eduardoadrian1994/conflicto-ejercicio-UTN.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.

