

# Práctico:

# Git y Github

## Programación 1

1. Responde las siguientes preguntas y desarrolla las respuestas:

- ¿Qué es GitHub?

**GitHub** es una plataforma basada en la nube que permite gestionar repositorios de Git. Facilita la colaboración entre desarrolladores mediante herramientas como control de versiones, pull requests (solicitudes de revisión y aprobación de cambios en el código de un repositorio) y revisiones de código.



- ¿Cómo crear un repositorio en GitHub?

Para crear un repositorio en GitHub puedes seguir los siguientes pasos:

- i. Inicia sesión en **GitHub**.
- ii. Haz clic en el botón "+" en la esquina superior derecha y selecciona **"Nuevo repositorio"**.
- iii. Introduce un **nombre** para el repositorio.
- iv. Selecciona si este será un repositorio **público** o un repositorio **privado**.
- v. Opcionalmente, puedes agregar un archivo **README** con una descripción del repositorio.
- vi. Haz clic en **"Crear repositorio"**.

Create a new repository

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

Owner \*  Repository name \*

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

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:  
Skip this step if you're importing an existing repository.

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

Add .gitignore  
Choose which files not to track from a list of templates. [Learn more.](#)

Choose a license  
A license tells others what they can and can't do with your code. [Learn more.](#)

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.

- ¿Cómo crear una rama en Git?

Las ramas permiten trabajar en nuevas características sin afectar la versión principal del código.

Para crear una nueva rama:

```
git branch nombre-rama
```

Esto crea una nueva rama sin cambiar a ella.

Para crear y cambiar a la nueva rama en un solo comando:

```
git checkout -b nombre-rama
```

- ¿Cómo cambiar a una rama en Git?

Para cambiar de rama:

```
git checkout nombre-rama
```

O, en versiones recientes de Git:

```
git switch nombre-rama
```

Esto moverá el estado de trabajo a la nueva rama seleccionada.

- ¿Cómo fusionar (mergear) ramas en Git?

Para fusionar una rama con la rama principal (por ejemplo, **main** o **master**):

```
git checkout main
```

```
git merge nombre-rama
```

Si hay conflictos, Git pedirá resolverlos manualmente editando los archivos afectados.

- ¿Cómo crear un commit en Git?

Después de hacer cambios en los archivos del repositorio, se deben guardar con un commit:

```
git add .
```

```
git commit -m "Descripción breve del cambio"
```

El primer comando **add .** agrega todos los cambios al área de preparación, y **commit** los guarda en el historial del repositorio.

- ¿Cómo enviar un commit a GitHub?

```
git push origin nombre-rama
```

Esto subirá los cambios confirmados a la rama especificada en el repositorio remoto.

- ¿Qué es un repositorio remoto?

Un repositorio remoto es una versión del código alojada en un servidor como **GitHub**, **GitLab** o **Bitbucket**. Permite la colaboración entre desarrolladores y el almacenamiento seguro del código.



- ¿Cómo agregar un repositorio remoto a Git?

```
git remote add origin URL-del-repositorio
```

Lautaro Ariel Cejas – Comisión M2025-11

Esto vincula el repositorio local con el remoto en GitHub.

- ¿Cómo empujar (pushear) cambios a un repositorio remoto?

```
git push origin nombre-rama
```

Esto sube los cambios de la rama local a la misma rama en el repositorio remoto.

- ¿Cómo tirar (pullear) de cambios de un repositorio remoto?

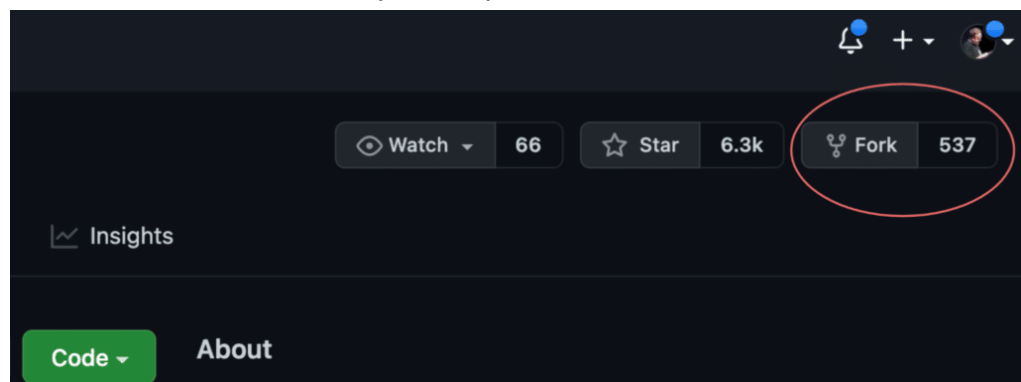
```
git pull origin nombre-rama
```

Esto descarga y fusiona los cambios más recientes del repositorio remoto en la rama actual.

- ¿Qué es un fork de repositorio?

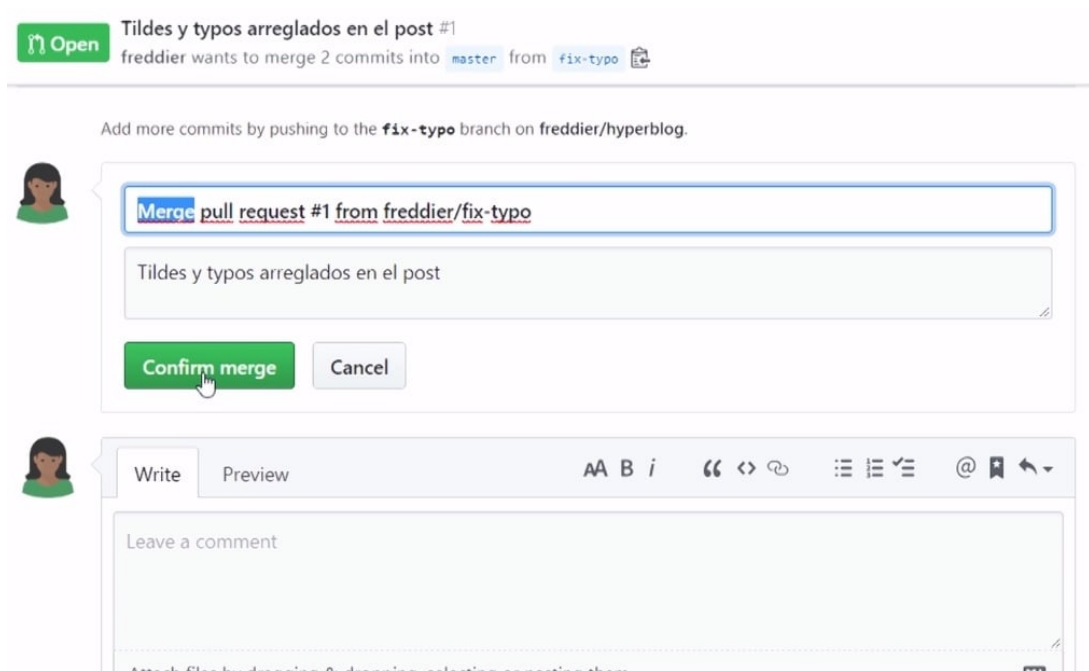
Un **fork** es una copia de un repositorio en otra cuenta. Se usa para contribuir a proyectos sin afectar el original, permitiendo experimentar libremente antes de proponer cambios.

- ¿Cómo crear un fork de un repositorio?
  - i. Ve al repositorio en GitHub.
  - ii. Haz clic en "**Fork**" en la esquina superior derecha.

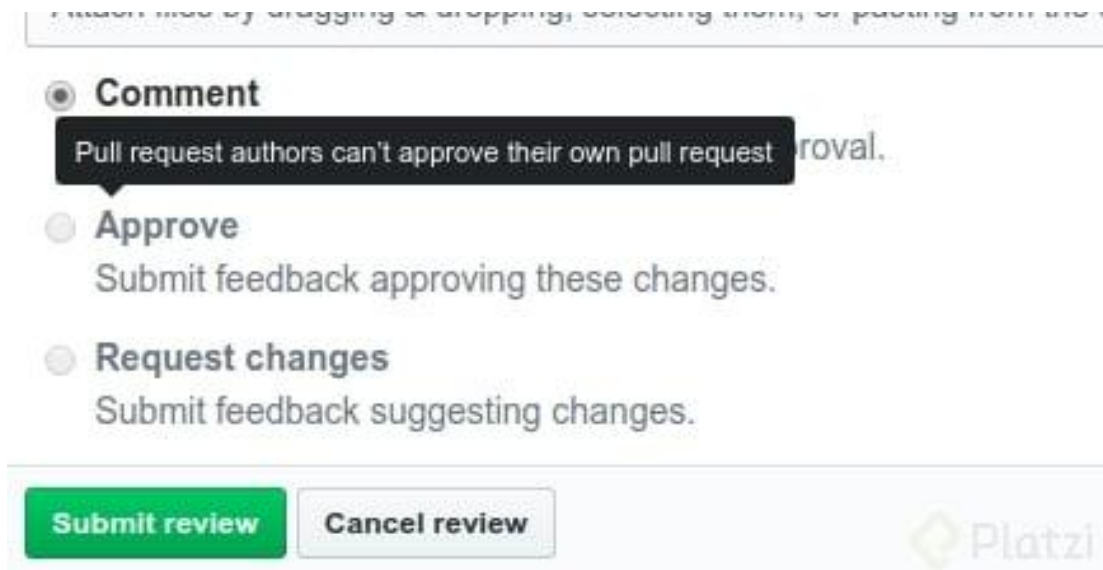


Esto crea una copia del repositorio en tu cuenta.

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



- i. **Crea una rama paralela:** Antes de hacer cambios en el código, utiliza el comando `git checkout -b nombre-rama` para crear una nueva rama. Así, podrás hacer tus modificaciones sin afectar la rama principal (por ejemplo, `main`).
  - ii. **Realiza commits:** Después de hacer cambios en los archivos, usa `git commit -m 'Comentario'` para hacer un commit con un mensaje descriptivo.
  - iii. **Sube los cambios:** Usa `git push origin nombre-rama` para subir tus cambios de la rama local al repositorio remoto. Reemplaza `nombre-rama` con el nombre de tu rama.
  - iv. **Crea un pull request:** En el repositorio remoto (como GitHub), crea un nuevo pull request. Selecciona la rama principal como destino y tu rama con los cambios como comparación.
  - v. **Feedback:** Los revisores examinarán los cambios. Usa la sección de comentarios del pull request para discutir los cambios y proporcionar feedback adicional.
  - vi. **Realiza los cambios solicitados:** Si se solicitan cambios, regresa a tu rama local y haz las modificaciones necesarias. Luego, sube los cambios al repositorio remoto usando `git push origin nombre-rama`.
- ¿Cómo aceptar una solicitud de extracción?



- i. **Acepta los cambios en GitHub:** Si estás satisfecho con los cambios propuestos en el pull request y consideras que están listos para ser fusionados con la rama principal, acepta el pull request en GitHub. De esta forma, los cambios se fusionarán en la rama principal del repositorio.
- ii. **Realiza el merge en la rama principal:** Después de aceptar el pull request, selecciona la opción para realizar el merge en GitHub. Esto combinará los cambios de la rama con los cambios existentes en la rama principal (**main**).

- ¿Qué es una etiqueta en Git?

Una etiqueta (**tag**) es una referencia a un punto específico en la historia de Git, utilizada para marcar versiones importantes del código.

- ¿Cómo crear una etiqueta en Git?

Para crear una **etiqueta** en Git sigue los siguientes pasos:

- i. Escribe **git tag** seguido del nombre de la etiqueta, por ejemplo, **git tag v1.4**.
  - ii. Reemplaza **<tagname>** (nombre de la etiqueta, se escribe después de git tag) por un nombre que represente el estado del repositorio.
- ¿Cómo enviar una etiqueta a GitHub?

**git push origin nombre-etiqueta**

Esto sube la etiqueta al repositorio.



- ¿Qué es un historial de Git?

El **historial de Git** muestra los cambios realizados en un repositorio, permitiendo rastrear modificaciones y colaboraciones.

- ¿Cómo ver el historial de Git?

**git log**

Esto muestra una lista de commits con sus detalles.

- ¿Cómo buscar en el historial de Git?

Para buscar en el historial de Git, puedes usar los comandos **git log**, **git grep**, y **git rev-list**.

- git grep patron \$(git rev-list --all)** Busca un patrón en todo el historial de confirmaciones
  - git log -p -G patron** Busca confirmaciones que introduzcan o eliminen un patrón
  - git log -p -S cadena** Busca confirmaciones que agreguen o eliminen una cadena específica
  - git log --grep=patron** Busca un patrón en los mensajes de confirmación
  - git grep** Busca a través de cualquier branch o directorio de trabajo con commit por una cadena o expresión regular
- ¿Cómo borrar el historial de Git?

No se puede borrar completamente el historial de Git, pero se puede reescribir:

**git rebase -i HEAD~n**

**Precaución:** Esto altera la historia, debe usarse con cuidado.

- ¿Qué es un repositorio privado en GitHub?

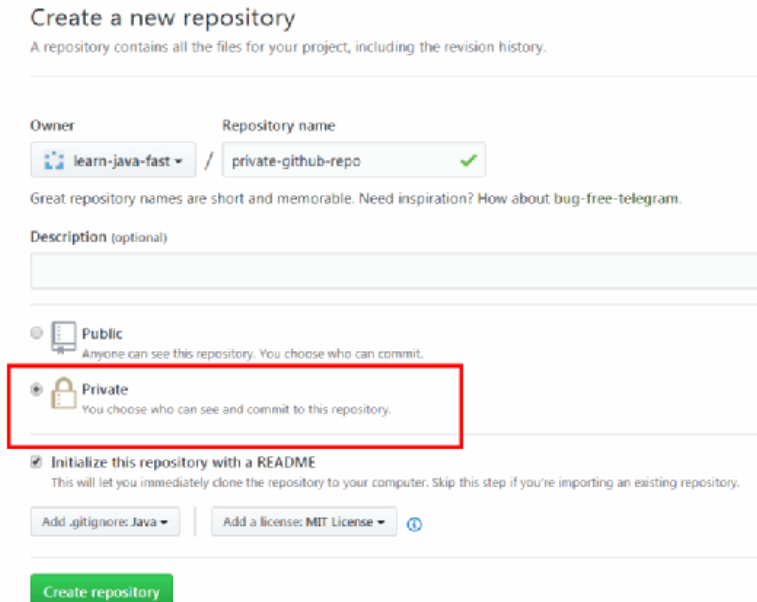
Un **repositorio privado en Github** es un repositorio visible solo para personas autorizadas, útil para proyectos internos o confidenciales. Por ejemplo; la mayoría de las empresas IT tiene el desarrollo de su software bajo un repositorio de este tipo.

- ¿Cómo crear un repositorio privado en GitHub?



Para crear un repositorio privado en GitHub, puedes:

- i. Ir a **GitHub**
- ii. Seleccionar **Nuevo repositorio** en la esquina superior derecha
- iii. Nombrar el repositorio
- iv. Elegir la visibilidad del repositorio. (desde aquí ya puedes ponerle **Privado**)
- v. Hacer clic en **Crear repositorio**
- vi. Navegar al repositorio
- vii. Hacer clic en **Configuración** en el menú
- viii. Junto a **Visibilidad**, seleccionar **Privado**
- ix. Verificar que estás cambiando la visibilidad del repositorio correcto
- x. Hacer clic en **Lo entiendo**, cambia la visibilidad del repositorio

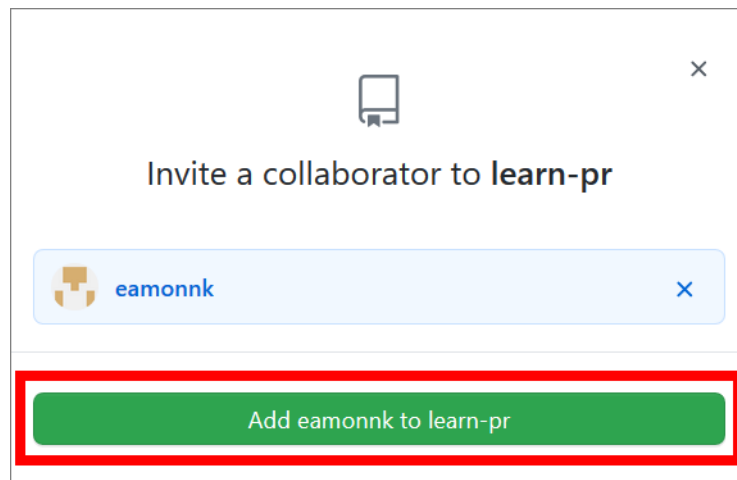


- ¿Cómo invitar a alguien a un repositorio privado en GitHub?

Para invitar a alguien a un repositorio privado en GitHub, puedes seguir estos pasos:

- i. Ir a la página principal del repositorio
- ii. Hacer clic en **Configuración**
- iii. En la barra lateral, hacer clic en **Colaboradores y equipos**
- iv. Debajo de **Administrar acceso**, hacer clic en **Agregar personas**
- v. Escribir el nombre de usuario o la dirección de correo electrónico de la persona a invitar

- vi. Elegir el rol de repositorio que se le quiere conceder
- vii. Hacer clic en **Agregar <NOMBRE> al <REPOSITORIO>**





- ¿Qué es un repositorio público en GitHub?

Un **repositorio público en GitHub** es un espacio virtual donde se puede almacenar, compartir y administrar código, archivos y revisiones de forma que sea accesible para cualquier persona en internet.

- ¿Cómo crear un repositorio público en GitHub?

Para crear un repositorio público en GitHub, puedes:

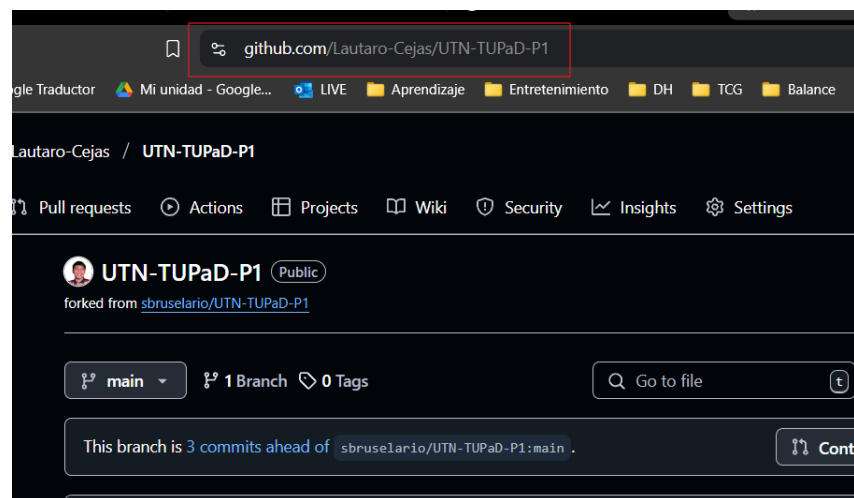
- i. Iniciar sesión en **GitHub**
- ii. En la esquina superior derecha, seleccionar **Nuevo repositorio**
- iii. Escribir un nombre para el repositorio
- iv. Añadir una descripción opcional
- v. Elegir la visibilidad del repositorio, en este caso **Pública**
- vi. Seleccionar Inicializar este repositorio con un **README**
- vii. Hacer clic en **Crear repositorio**

- ☒  **Public**  
Anyone can see this repository. You choose who can commit.
- ☐  **Private**  
You choose who can see and commit to this repository.

- ¿Cómo compartir un repositorio público en GitHub?

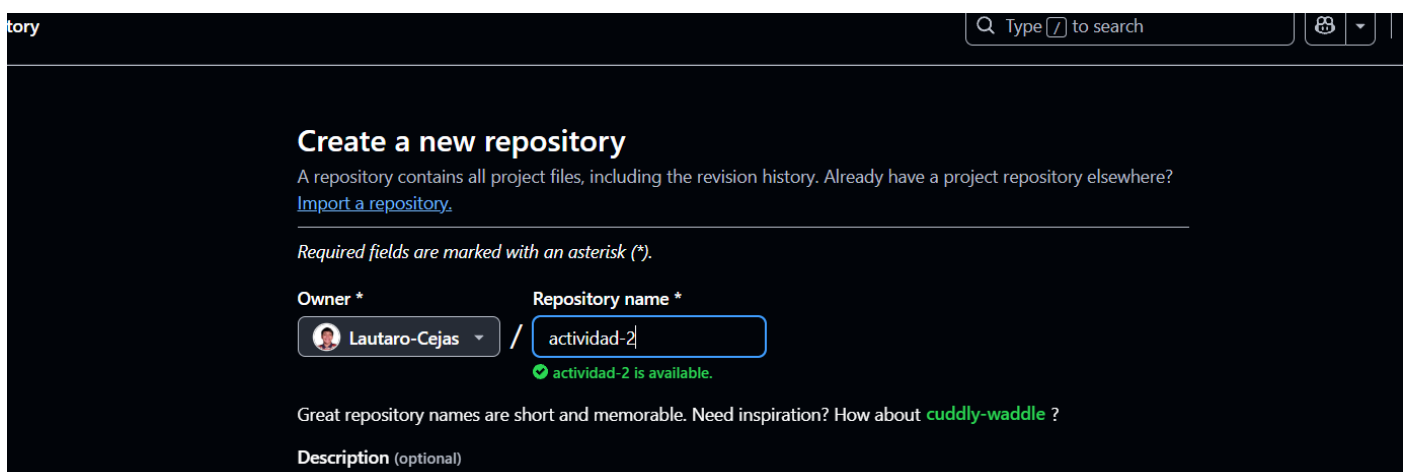
Para compartir un repositorio público en GitHub, puedes invitar a colaboradores o cambiar la visibilidad del repositorio.

Si solo deseas compartir la dirección del mismo, puedes copiar el enlace que está en la pestaña del repositorio que deseas compartir.

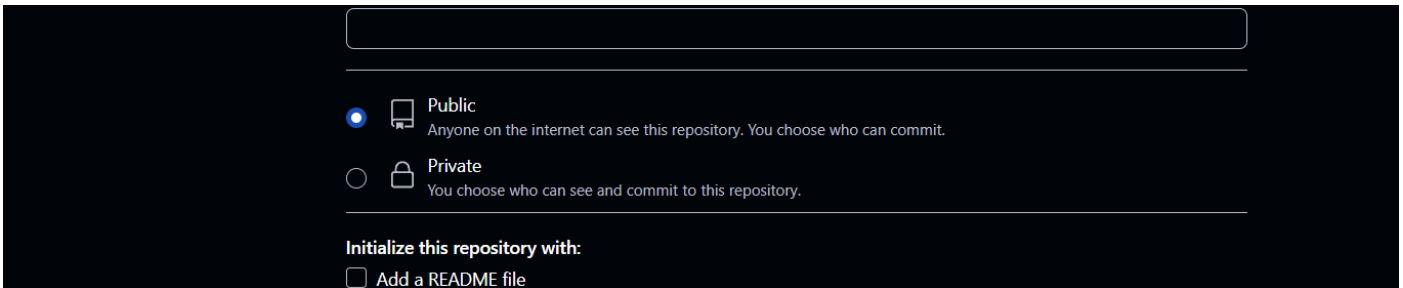


## 2. Realizar la siguiente actividad:

- Crear un repositorio.
  - i. Dale un nombre al repositorio.



- ii. Elije el repositorio sea público.



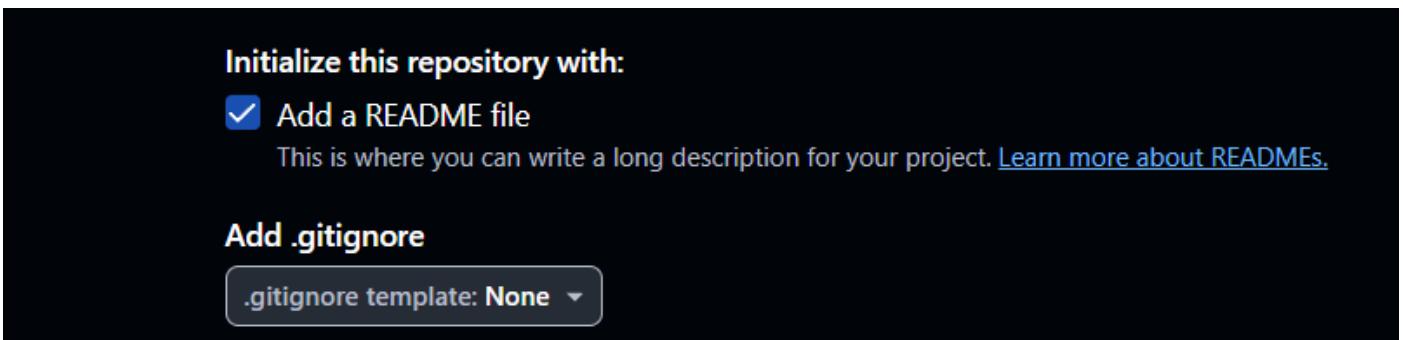
Public  
Anyone on the internet can see this repository. You choose who can commit.

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

Initialize this repository with:

☒ Add a README file

iii. Inicializa el repositorio con un archivo.



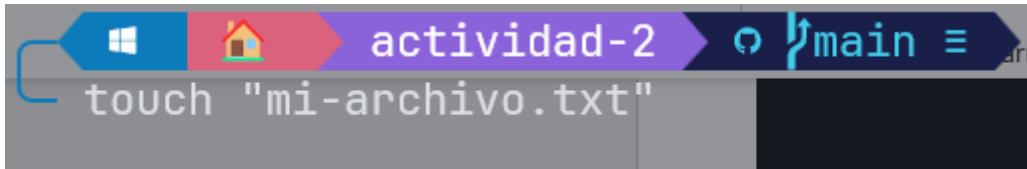
Initialize this repository with:

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

Add .gitignore

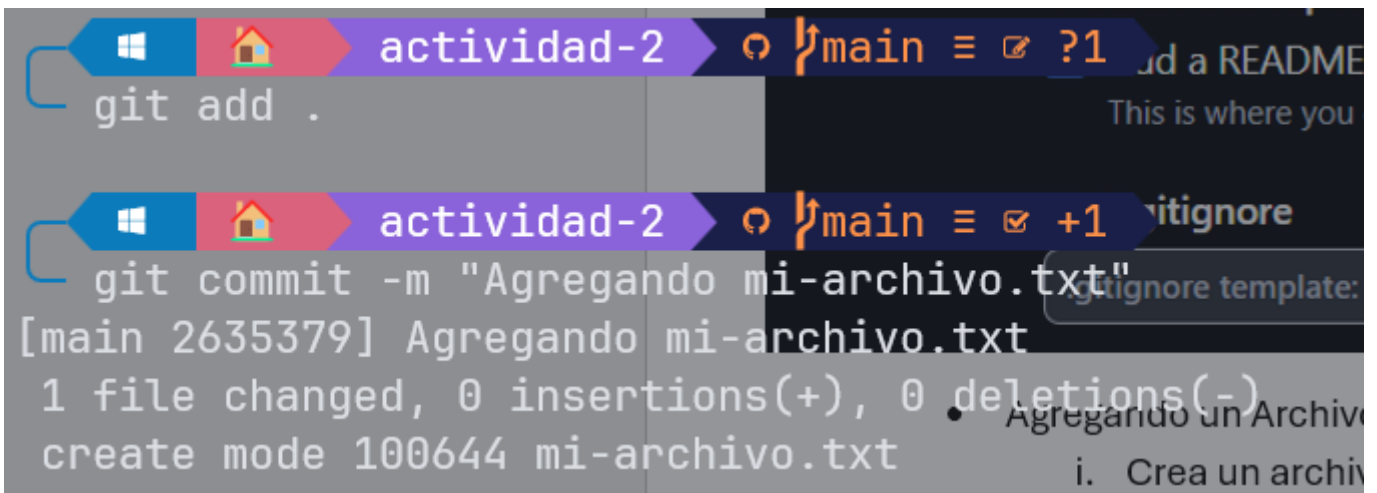
.gitignore template: None

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



```
touch "mi-archivo.txt"
```

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



```
git add .

git commit -m "Agregando mi-archivo.txt"
[main 2635379] Agregando mi-archivo.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 mi-archivo.txt
```

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

```
git push origin main
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), 291 bytes | 291.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Lautaro-Cejas/actividad-2.git
c88dd63..2635379  main -> main
```

- Creando Branchs
  - i. Crear una Branch

```
git branch rama
```

- ii. Realizar cambios o agregar un archivo

```
git checkout rama
Switched to branch 'rama'
```

```
touch "otro-archivo.txt"
```

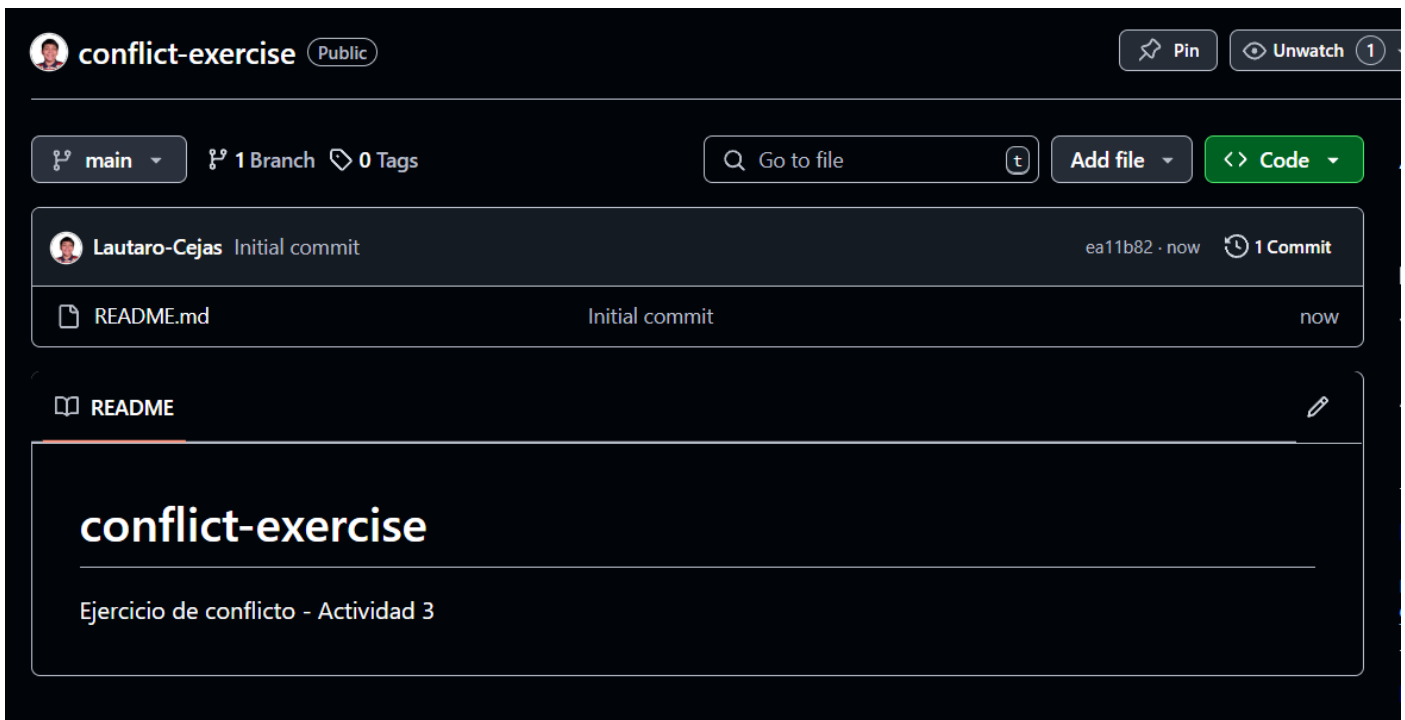
- iii. Subir la Branch

```
git add .
git checkout rama
Switched to branch 'rama'
git commit -m "Subiendo otro-archivo.txt"
[rama af665a] Subiendo otro-archivo.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 otro-archivo.txt
git push origin rama
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), 293 bytes | 293.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'rama' on GitHub by visiting:
remote: https://github.com/Lautaro-Cejas/actividad-2/pull/new/rama
remote:
To https://github.com/Lautaro-Cejas/actividad-2.git
* [new branch] rama -> rama
```

### 3. Realizar la siguiente actividad:

#### Paso 1: Crear un repositorio en GitHub

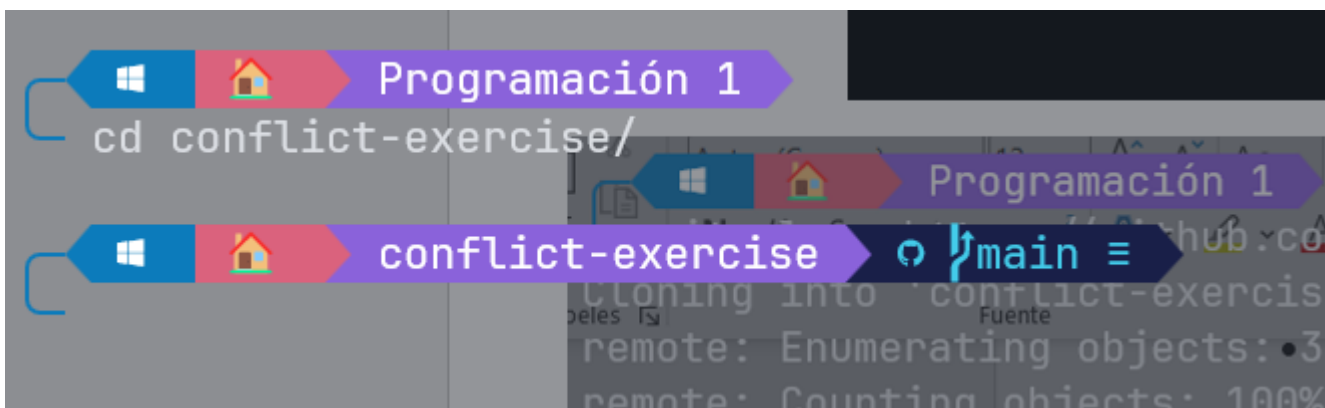
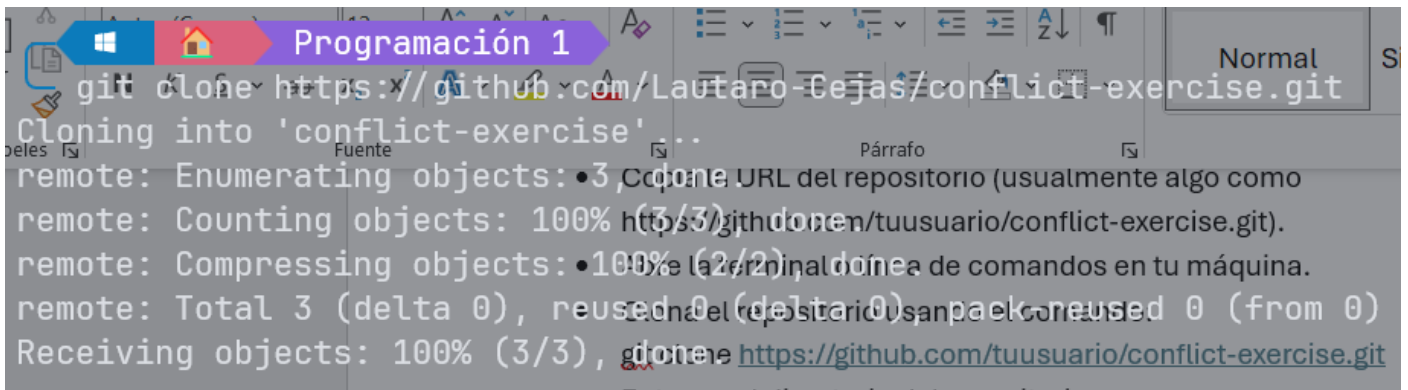
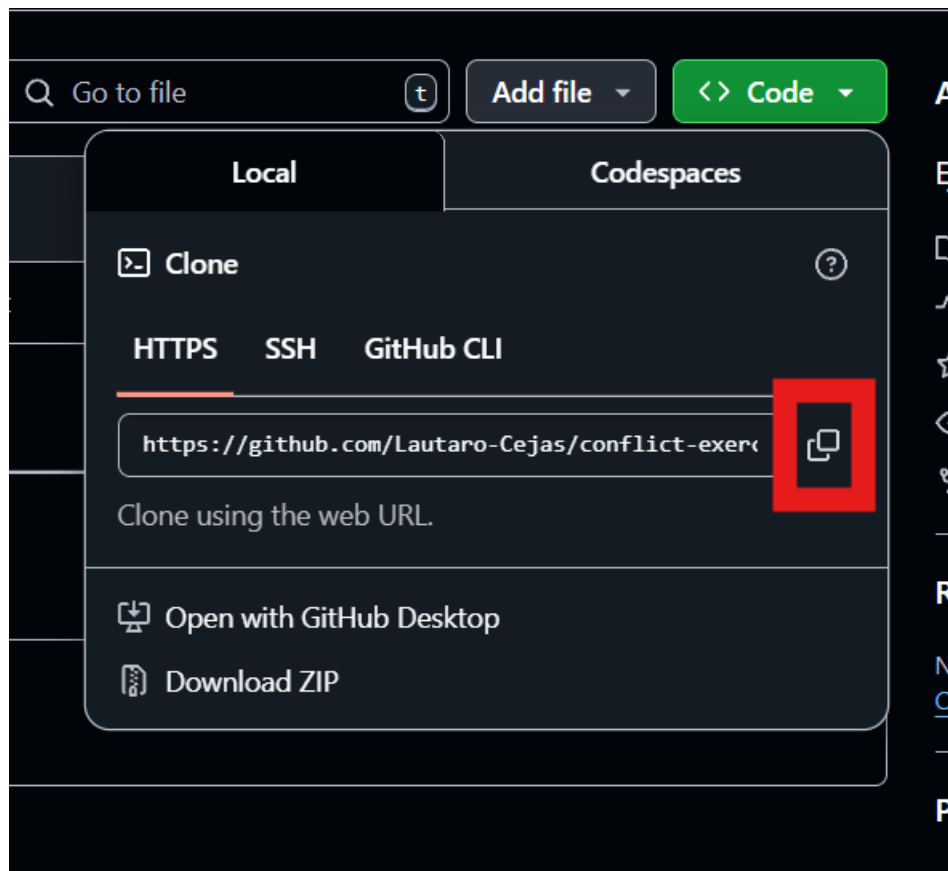
- Ve a GitHub e inicia sesión en tu cuenta.
- Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.
- Asigna un nombre al repositorio, por ejemplo, conflict-exercise.
- Opcionalmente, añade una descripción.
- Marca la opción "Initialize this repository with a README".
- Haz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



## Paso 2: Clonar el repositorio a tu máquina local

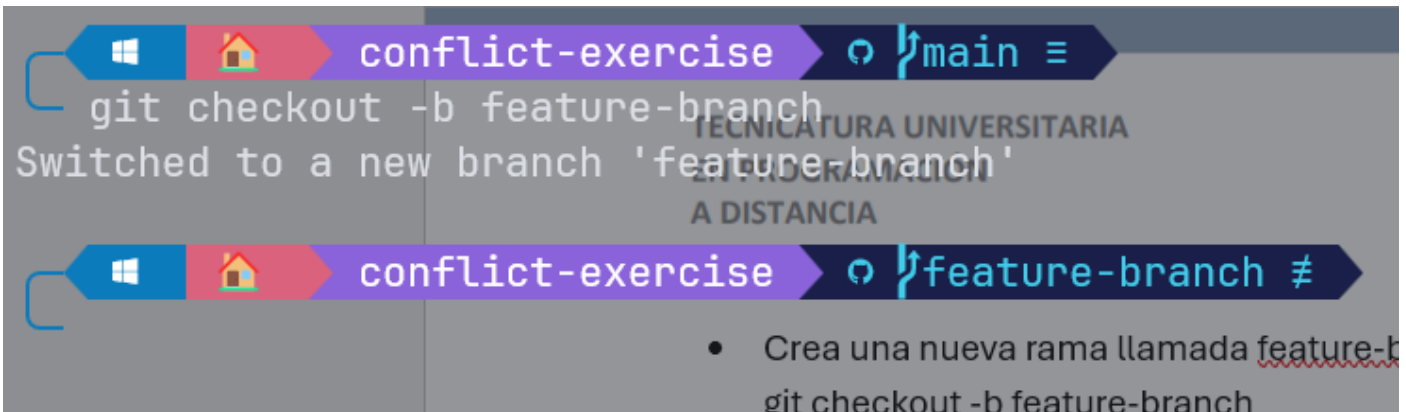
- Copia la URL del repositorio (usualmente algo como <https://github.com/tuusuario/conflict-exercise.git>).
- Abre la terminal o línea de comandos en tu máquina.
- Clona el repositorio usando el comando:  
`git clone https://github.com/tuusuario/conflict-exercise.git`
- Entra en el directorio del repositorio:  
`cd conflict-exercise`



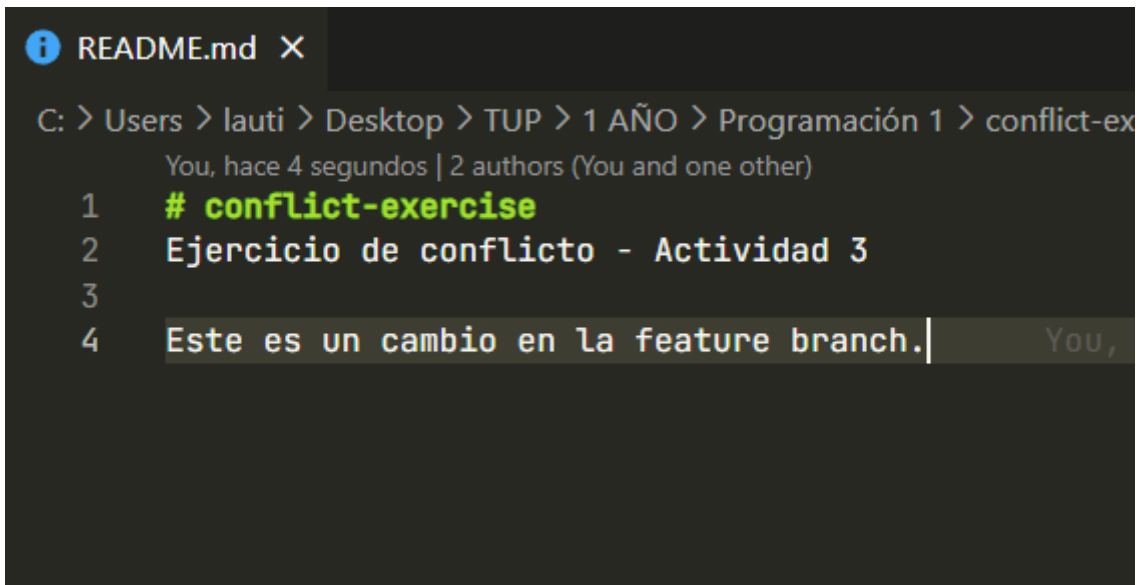


Paso 3: Crear una nueva rama y editar un archivo

- 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"`

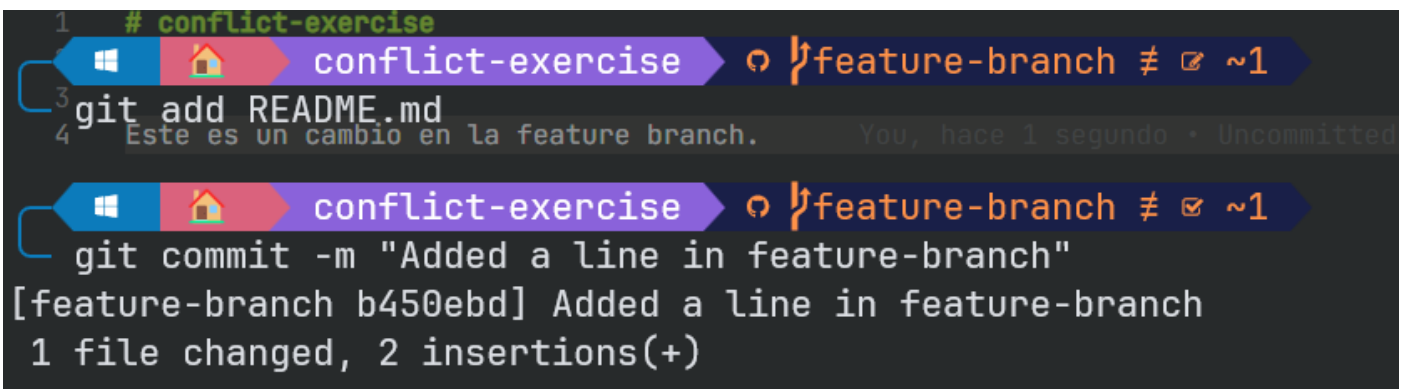


The diagram shows a Git branch structure. At the top, a branch named 'main' is selected. Below it, the command `git checkout -b feature-branch` is shown, followed by the message 'Switched to a new branch 'feature-branch''. Below this, a new branch named 'feature-branch' is shown, indicating the user has switched to it.



The screenshot shows a text editor window titled 'README.md'. The file path is 'C: > Users > lauti > Desktop > TUP > 1 AÑO > Programación 1 > conflict-ex'. The content of the file is as follows:

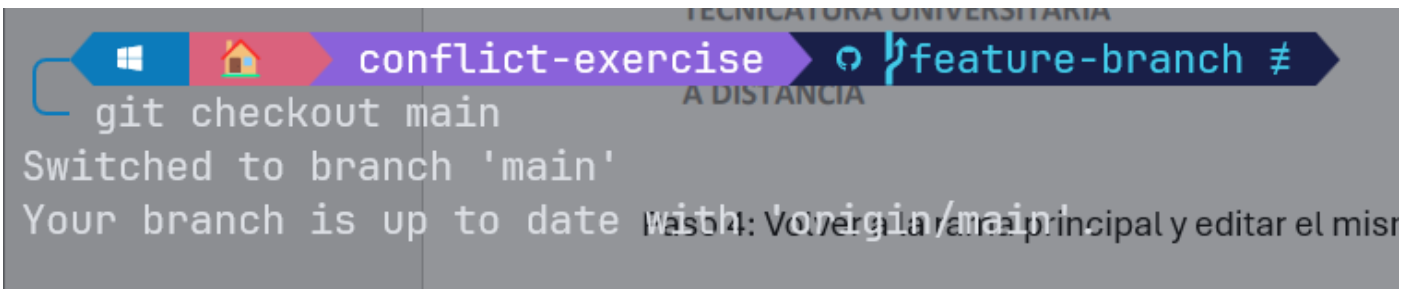
```
1 # conflict-exercise
2 Ejercicio de conflicto - Actividad 3
3
4 Este es un cambio en la feature branch.
```



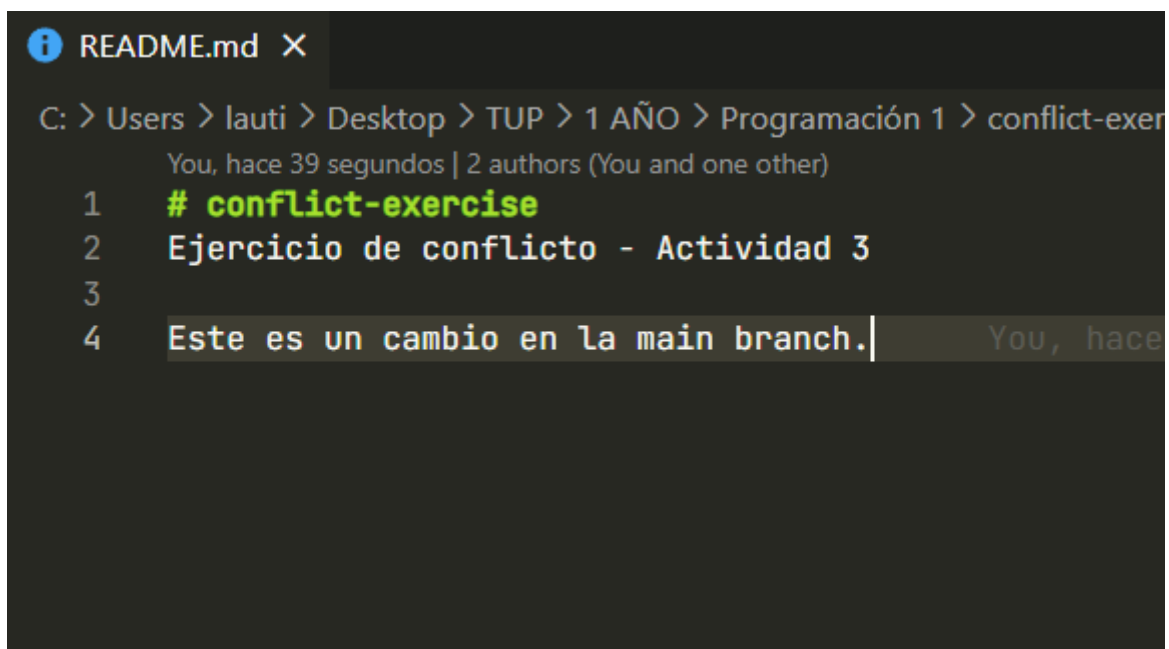
The screenshot shows the output of the `git add` and `git commit` commands. The first command is `git add README.md`, which adds the file to the staging area. The second command is `git commit -m "Added a line in feature-branch"`, which creates a new commit. The output shows the commit hash 'b450ebd' and the message 'Added a line in feature-branch'.

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

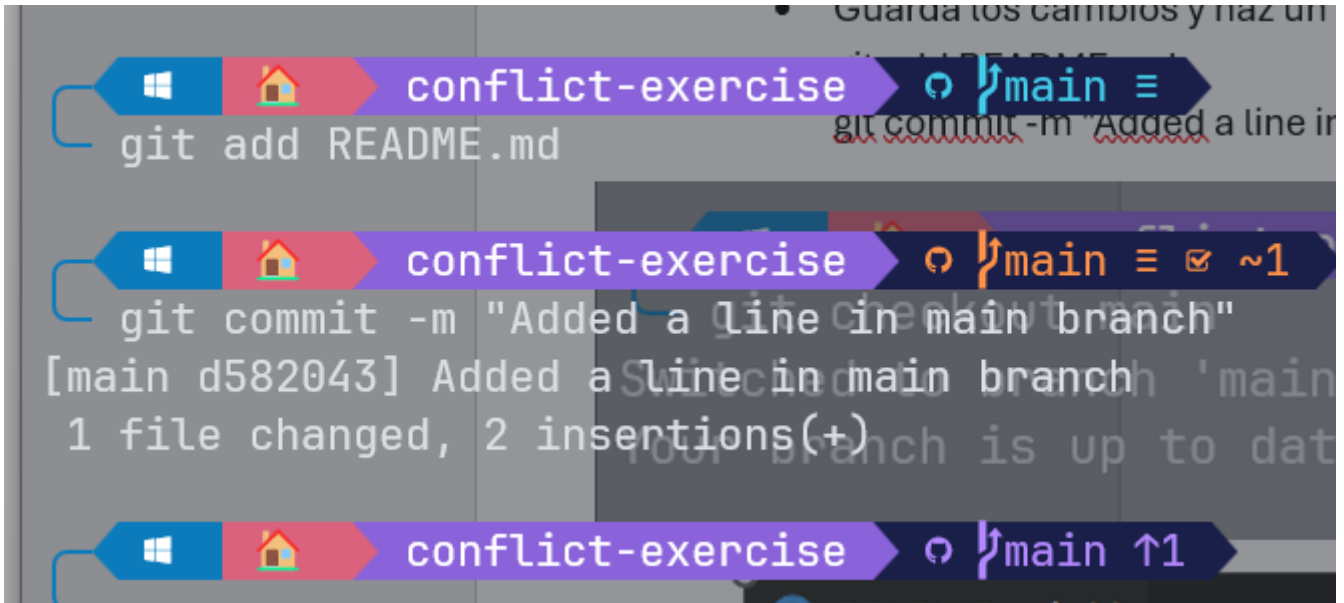
- Cambia de vuelta a la rama principal (main):  
`git checkout main`
- Edita el archivo README.md de nuevo, añadiendo una línea diferente:  
Este es un cambio en la main branch.
- Guarda los cambios y haz un commit:  
`git add README.md`  
`git commit -m "Added a line in main branch"`



```
git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'
```



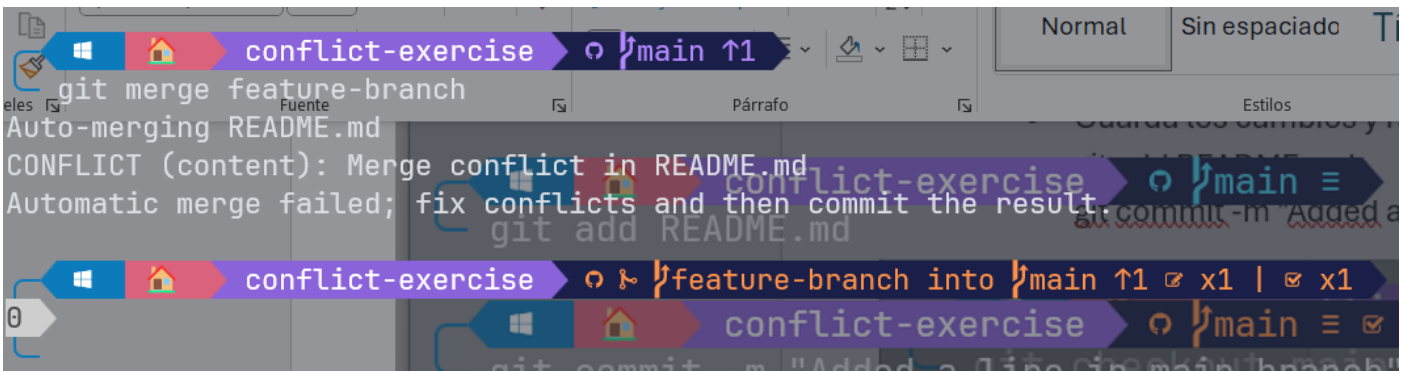
```
1 # conflict-exercise
2 Ejercicio de conflicto - Actividad 3
3
4 Este es un cambio en la main branch.
```



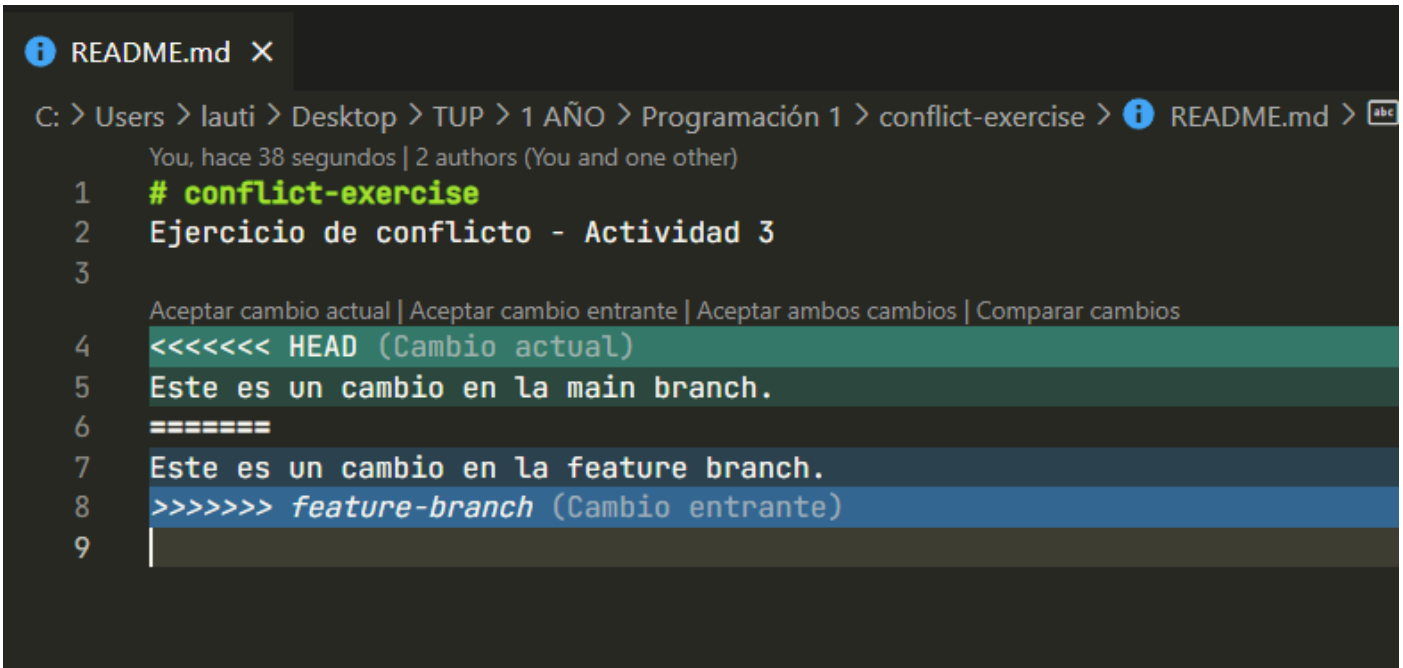
```
conflict-exercise ➤ main ≡  
git add README.md  
git commit -m "Added a line in main branch"  
[main d582043] Added a line in main branch  
1 file changed, 2 insertions(+)  
conflict-exercise ➤ main ↑1
```

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.



```
conflict-exercise ➤ main ↑1  
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.  
conflict-exercise ➤ feature-branch into main ↑1 x1 | x1  
conflict-exercise ➤ main ≡
```



The screenshot shows a code editor window titled "README.md" with a file explorer on the left. The file path is "C: > Users > lauti > Desktop > TUP > 1 AÑO > Programación 1 > conflict-exercise > README.md". The editor content shows a Git conflict resolution for the file "README.md". The conflict is between the "HEAD (Cambio actual)" and the "feature-branch (Cambio entrante)". The conflict is resolved by accepting the changes from the feature branch. The text in the editor is as follows:

```
1 # conflict-exercise
2 Ejercicio de conflicto - Actividad 3
3
4 <<<<<< HEAD (Cambio actual)
5 Este es un cambio en la main branch.
6 =====
7 Este es un cambio en la feature branch.
8 >>>>>> feature-branch (Cambio entrante)
9
```

#### 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"
```

```
README.md X
C: > Users > lauti > Desktop > TUP > 1 AÑO > Programación 1 > conflict-exercise > README.md >
You, hace 58 segundos | 2 authors (You and one other)
1 # conflict-exercise
2 Ejercicio de conflicto - Actividad 3
3
4 Aceptar cambio actual | Aceptar cambio entrante | Aceptar ambos cambios | Comparar cambios
5 <<<<<<< HEAD (Cambio actual)
6 Este es un cambio en la main branch.
7 =====
8 Este es un cambio en la feature branch.
9 >>>>>>> feature-branch (Cambio entrante)
```

```
conflict-exercise • Editar el archivo para resolver el conflicto y guarda los cambios (Se debe
git add README.md la parte del texto que no se quiera dejar).
conflict-exercise • Añadir el archivo resuelto y completa el merge:
git commit -m "Resolved merge conflict" git add README.md
[main b860436] Resolved merge conflict
conflict-exercise • main ↑3
```

#### 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



```
conflict-exercise main ↑3
git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 758 bytes | 252.00 KiB/s, done.
Total 9 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/Lautaro-Cejas/conflict-exercise.git
ea11b82..b860436  main → main
```

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.

```
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
remote: https://github.com/Lautaro-Cejas/conflict-exercise
remote:
To https://github.com/Lautaro-Cejas/conflict-exercise.git
* [new branch] feature-branch → feature-branch
```

Paso 7: Subir los cambios a GitHub

- Sube los cambios de la rama `main` al repositorio `git push origin main`

#### 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.



conflict-exercise / README.md



Lautaro-Cejas Resolved merge conflict

Preview

Code

Blame

5 lines (4 loc) · 135 Bytes

Code 55% faster with GitHub Copilot

## conflict-exercise

Ejercicio de conflicto - Actividad 3

Este es un cambio en la main branch. Este es un cambio en la feature branch.

```
commit b86043685ee1d20860c21920b7d2b8c60d723352 (HEAD → main, origin/main, origin/HEAD)
Merge: d582043 b450ebd
Author: Lautaro Cejas <lauti217@live.com>
Date: Fri Mar 28 01:01:28 2025 -0300

    Resolved merge conflict

commit d582043fd6f6900fb3995d4b74647d16f279639b
Author: Lautaro Cejas <lauti217@live.com>
Date: Fri Mar 28 00:58:21 2025 -0300

    Added a line in main branch

commit b450ebd42641ca497090479eee4bc6ac20760432 (origin/feature-branch, feature-branch)
Author: Lautaro Cejas <lauti217@live.com>
Date: Fri Mar 28 00:56:18 2025 -0300

    Added a line in feature-branch

commit ea11b82f02a8bdafadb05baf818c61e60408dec6
Author: taro <79883620+Lautaro-Cejas@users.noreply.github.com>
Date: Fri Mar 28 00:51:16 2025 -0300

    Initial commit
```