

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.

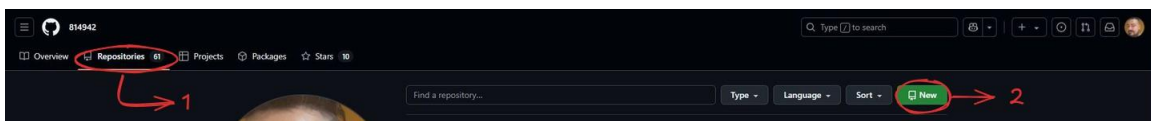
Actividades

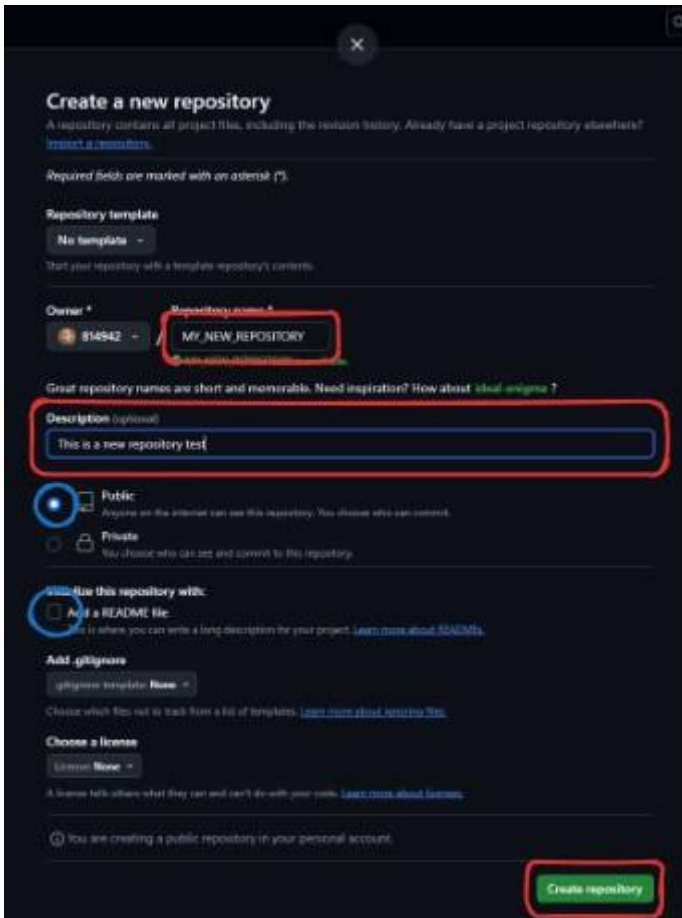
- 1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas) :

- ¿Qué es GitHub?
Es una plataforma para alojar código en repositorios. Sirve para ser compartido entre diferentes colaboradores, hacer backups y desplegar proyectos o, simplemente, para alojarlo de manera segura, tanto público como privado.
- ¿Cómo crear un repositorio en GitHub?

1- Entramos a nuestra cuenta de Github y vamos a la pestaña Repositories.

2- Luego vamos a New. Aquí ponemos el *Repository name*, el nombre que llevará nuestro repositorio, y, adicionalmente, una descripción. Así como también podemos elegir si es público o privado, y si queremos agregar un archivo README(un archivo tipo markdown que sirva para describir el repositorio), un archivo .gitignore(para incluir aquellas cosas que queremos que git ignore, que no queremos subir) y licencia.





- ¿Cómo crear una rama en Git?

Con el comando: `git branch <branchname>`

branchname es el nombre que le daremos a la rama

- ¿Cómo cambiar a una rama en Git?

Con el comando: `git checkout <branchname>`

Adicionalmente podemos realizar ambas acciones, creación y cambio a una rama, con: `git checkout -b <branchname>` esto nos cambiara a la rama en cuestión y, si no existe, la creara.

- ¿Cómo fusionar ramas en Git?

Con el comando: `git merge <branchname>`. Esto hará que fusionemos los cambios de la branchname a la que estemos llamando, con nuestro actual branch.

Imaginemos el siguiente escenario: supongamos que tenemos dos branch, main y user, cada branch con sus respectivos commits. Si hacemos `git merge user` desde main esto significa que traeremos todos los cambios de user hasta su último commit(C), y se registrará el resultado en un nuevo commit en main(E).

A—B—C user
 \
D—C—D—E main

- ¿Cómo crear un commit en Git?

Con el comando: `git commit -m <message>`

- ¿Cómo enviar un commit a GitHub?

Con el comando: `git push`, si tenemos configurado el upstream. La referencia a la rama remota asociada con una rama local. Si no la tenemos configurada o es la primera vez podemos utilizar el comando: `git push -u origin <branchname>`

`-u` establece `branchname` como upstream, para que en el futuro solo necesites `git push`

De otra forma lo podemos hacer con: `git push origin <branchname>`, esto enviará los cambios en nuestro branch al branch remoto, de main local a main en el repo, por ejemplo.

- ¿Qué es un repositorio remoto?

Es una versión de mi código alojado en remoto, Github, por ejemplo.

- ¿Cómo agregar un repositorio remoto a Git?

Primero deberíamos tener un repositorio inicializado a nivel local: `git init`

Luego, con el comando `git remote add origin <URL>`, agregamos el repositorio remoto. Donde URL es el link al repo en cuestión.

Adicionalmente podemos comprobar si fue agregado exitosamente con:

`git remote -v`

- ¿Cómo empujar cambios a un repositorio remoto?

Creo que la misma respuesta que para: *¿Cómo enviar un commit a GitHub?*, aplica aquí.

- ¿Cómo tirar de cambios de un repositorio remoto?

Para “tirar” o “bajarnos” los cambios desde un remoto a nuestro local utilizamos el comando: `git pull`

- ¿Qué es un fork de repositorio?

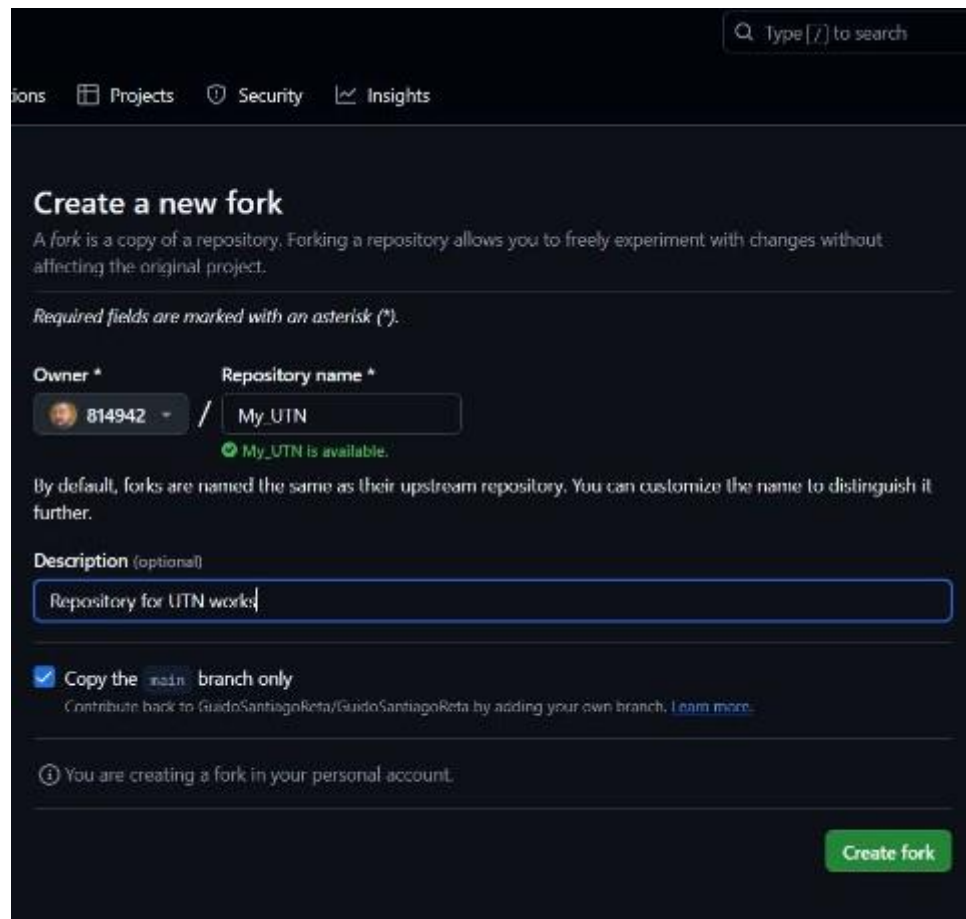
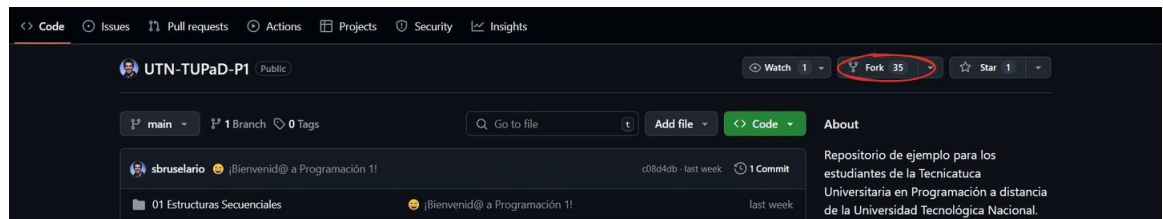
Es copiar un repositorio de alguien más a nuestra cuenta, a nuestro repositorio.

Crear un repositorio nuestro basado en la copia de un repositorio existente.

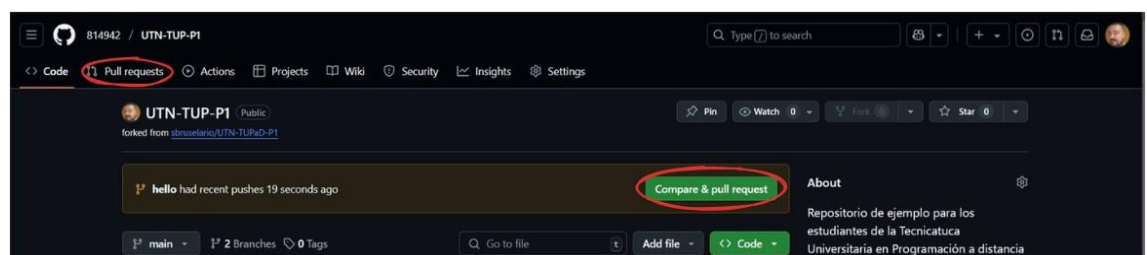
- ¿Cómo crear un fork de un repositorio?

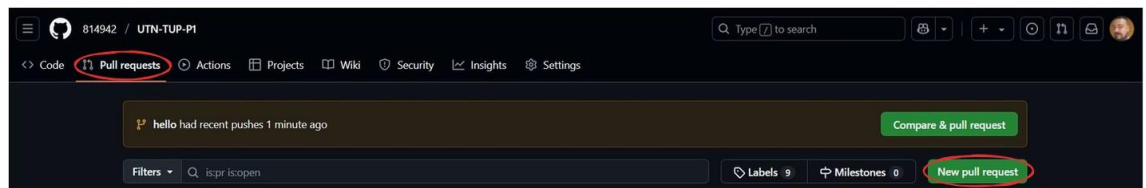
1- Buscamos el repositorio que queremos forkear y apretamos *Fork*.

2- Escribimos un nombre para el fork del repositorio y, adicionalmente, una descripción. También podemos elegir si copiar solo el branch principal, main, o no.



- ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
Una vez “pusheado” nuestros cambios al remoto. Podemos abrir un PR (pull request) para solicitar que nuestros cambios sean admitidos en otro branch, develop, por ejemplo.
1- Primero vamos a nuestro repositorio y aquí tenemos dos opciones. Si acabamos de pushear los cambios vamos a ver un pop-up para crear el PR, o clickeamos en la pestaña **Pull requests**.

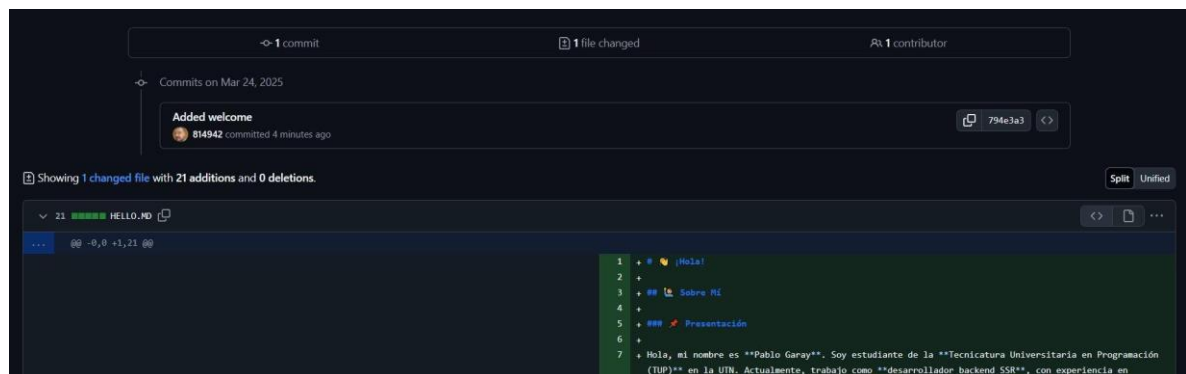
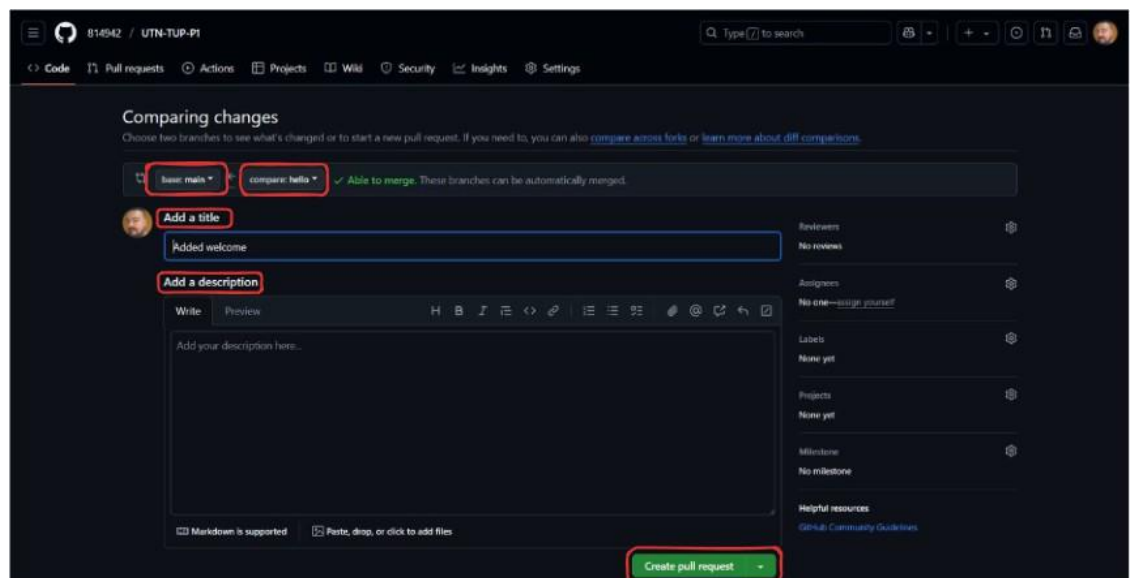




2- Una vez clickeado entramos en la pantalla de creación del PR.

Aquí, primero, debemos elegir el branch base y el branch a comparar. El branch base es el destino del PR, donde van a ir los cambios. El branch a comparar es nuestro branch donde tenemos los cambios y que queremos introducir en el branch base. Una vez elegido los branch le ponemos un título y una descripción para que quien sea que lo revise entienda de qué se trata los cambios que contiene.

Antes de crear el PR podemos también revisar lo que estamos intentando subir para asegurarnos de que esté bien. Podemos ver el historial de los commits hechos, los cambios hechos y cada archivo modificado.



Listo, hemos creado nuestro primer PR.

- ¿Cómo aceptar una solicitud de extracción?

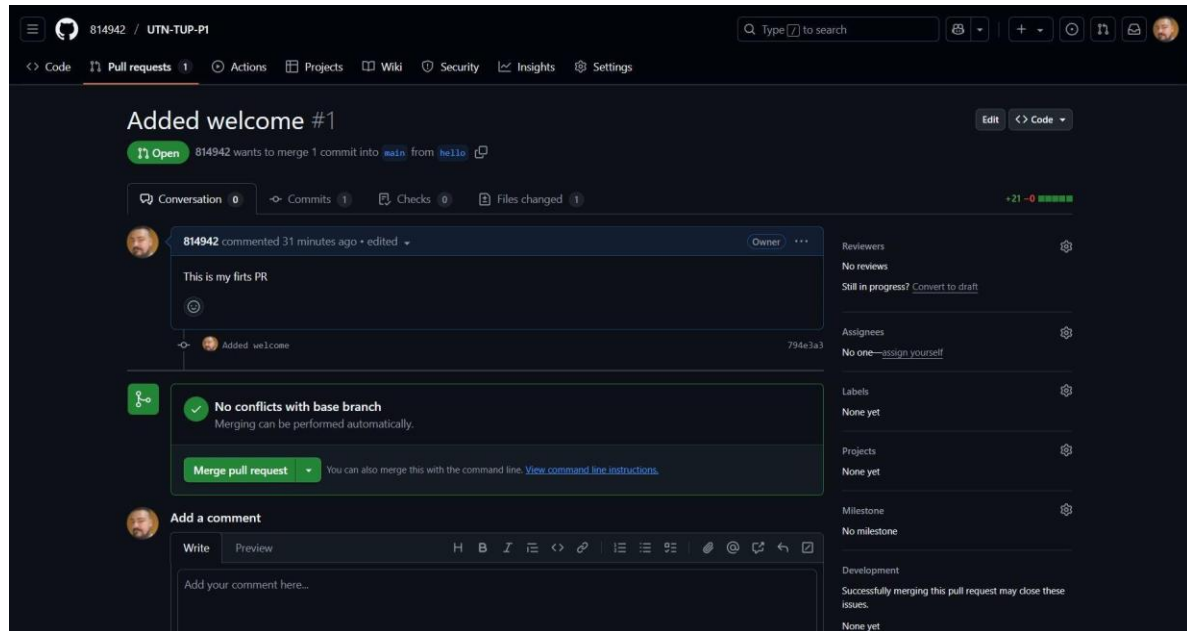
Una vez creado el PR podemos entrar en él, a través de la pestaña **Pull requests**. Si hay varios elegimos el que nos interesa, o el que hemos sido asignados.

Aquí revisamos los cambios que se realizaron y quieren ingresar en el branch base.

Si estamos de acuerdo con los cambios y los aceptamos le damos click en **Merge**

pull request.

Si, en cambio, vemos algo que no funciona deberia podemos solicitar que se realicen cambios antes de volver a revisarlo y aprobarlo eventualmente.



- ¿Qué es una etiqueta en Git?

Una etiqueta es una referencia a un punto específico en la historia del repositorio, generalmente utilizada para marcar versiones importantes, como lanzamientos de software

- ¿Cómo crear una etiqueta en Git?

Hay dos formas de crear una etiqueta, ligera(lightweight) o anotadas(annotated).

Las etiquetas ligeras son simplemente un puntero a un commit sin información adicional: `git tag v1.0.0`

Las etiquetas anotadas almacenan información adicional, como el autor, fecha y un mensaje: `git tag -a v1.0.0 -m "This is a comment"`

- ¿Cómo enviar una etiqueta a GitHub?

Las etiquetas no se suben por defecto con git push, deben subirse a mano con el comando: `git push origin <tagname>`

O también podemos subir todas las etiquetas de una vez, con el comando:

```
git push --tags
```

- ¿Qué es un historial de Git?

Es el registro completo de todos los cambios que se han realizado en un repositorio, incluyendo commits, ramas, etiquetas y fusión de ramas. Cada vez que realizas un commit, Git guarda un snapshot del estado de tu proyecto en ese momento, lo que permite ver qué cambios se hicieron, quién los hizo, cuándo se hicieron, y por qué.

- ¿Cómo ver el historial de Git?

Con el comando: `git log` podemos ver todos los commits.

Con el comando: `git branch` podemos ver todas las branches creadas.

Con el comando: `git tag` podemos ver todas las etiquetas creadas.

- ¿Cómo buscar en el historial de Git?

Cada commit tiene un hash asociado. Podemos usar este hash para buscar a través del historial con el comando: `git show <commit-hash>`

- ¿Cómo borrar el historial de Git?

Nunca es buena idea borrar el historial de Git. Sin embargo esto es posible para situaciones específicas.

1- Podemos eliminar todo el historial reiniciando el repositorio, con el comando:

`rm -rf .git`, esto nos obligará a inicializar el repositorio de nuevo.

2- Otra opción es borrar los commits locales, con el comando:

`git reset --hard HEAD`

3- O, si ya poseamos los cambios al remoto, utilizamos el comando:

`git push origin main --force`

4- La última opción es “rebasar” nuestra branch para borrar o modificar los commits, esto lo hacemos con el comando: `git rebase`, para cambiar replicar los cambios sobre nuestra branch actual. O `git rebase <branchname>`, para hacerlo sobre otra rama.

- ¿Qué es un repositorio privado en GitHub?

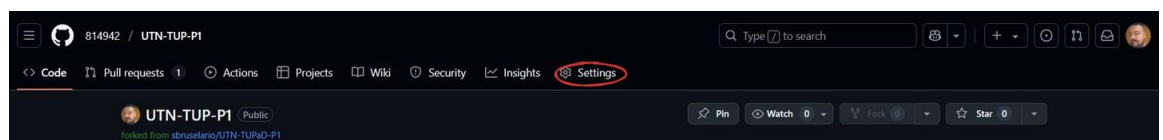
Un repositorio en donde solo yo, o a quien autorice, pueden entrar.

- ¿Cómo crear un repositorio privado en GitHub?

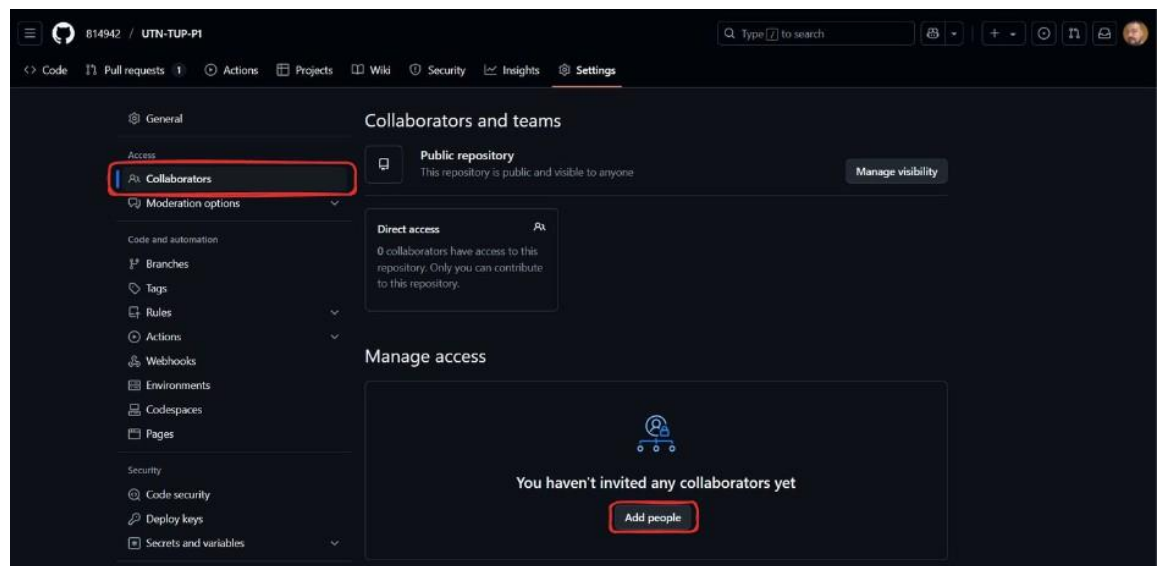
De la misma forma que creamos cualquier repositorio. Simplemente chequeamos *Private*, en lugar de *Public*.

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

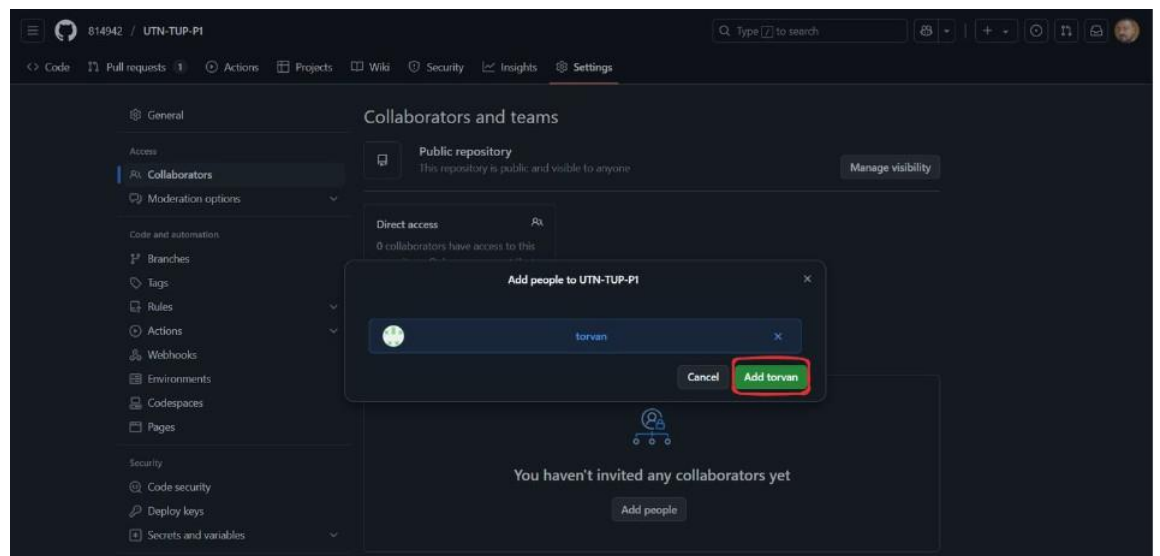
1- Vamos a nuestro repositorio y entramos en **Settings**.



2- Luego vamos al panel de la izquierda, dentro del apartado **Access** elegimos **Collaborators**. Y aquí podemos agregar a nuestros colaboradores que tendrán acceso al repositorio haciendo click en **Add people**.



3- Ahora simplemente elegimos al/los colaborador/es que deseamos incluir y le damos click en **Add ...**



- ¿Qué es un repositorio público en GitHub?

Por defecto todo repo que creamos se crea publico. Esto quiere decir que cualquier persona puede acceder a él y que nuestro código es “código abierto”.

- ¿Cómo crear un repositorio público en GitHub?

De la misma manera en que lo hicimos en la pregunta: *¿Cómo crear un repositorio en GitHub?*. Simplemente tenemos que dejar chequeado *Public*.

- ¿Cómo compartir un repositorio público en GitHub?

Simplemente copiando la URL de nuestro repositorio.


- 2) Realizar la siguiente actividad:
- Crear un repositorio.
 - Dale un nombre al repositorio.
 - Elige que el repositorio sea público.
 - Inicializa el repositorio con un archivo.


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 *



 Isragadiel / new_repo_test

 new_repo_test is available.

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

Description (optional)

Este es una prueba de repositorio

- ☒  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.](#)

- 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).

```
px005@Gladys MINGW64 ~/OneDrive/Desktop/new_repo_test
$ git clone https://github.com/814942/new_repo_test.git
Cloning into 'new_repo_test'...
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.

px005@Gladys MINGW64 ~/OneDrive/Desktop
$ cd new_repo_test

px005@Gladys MINGW64 ~/OneDrive/Desktop/new_repo_test (main)
$ ls
README.md

px005@Gladys MINGW64 ~/OneDrive/Desktop/new_repo_test (main)
$ touch mi-archivo.txt

px005@Gladys MINGW64 ~/OneDrive/Desktop/new_repo_test (main)
$ ls
README.md  mi-archivo.txt

px005@Gladys MINGW64 ~/OneDrive/Desktop/new_repo_test (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    mi-archivo.txt

nothing added to commit but untracked files present (use "git add" to track)

px005@Gladys MINGW64 ~/OneDrive/Desktop/new_repo_test (main)
$ git add .

px005@Gladys MINGW64 ~/OneDrive/Desktop/new_repo_test (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   mi-archivo.txt

px005@Gladys MINGW64 ~/OneDrive/Desktop/new_repo_test (main)
$ |
```

```
px005@Gladys MINGW64 ~/OneDrive/Desktop/new_repo_test (main)
$ git commit -m "Agregando mi-archivo.txt"
[main c00e8f5] Agregando mi-archivo.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 mi-archivo.txt

px005@Gladys MINGW64 ~/OneDrive/Desktop/new_repo_test (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean

px005@Gladys MINGW64 ~/OneDrive/Desktop/new_repo_test (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 288 bytes | 288.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/814942/new_repo_test.git
   0b61968..c00e8f5  main -> main

px005@Gladys MINGW64 ~/OneDrive/Desktop/new_repo_test (main)
$ |
```

Isragadiel / new_repo_test

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

new_repo_test Public Pin Unwatch 1 Fork 0 Star 0

main 1 Branch 0 Tags

Q Go to file t Add file <> Code About

Isragadiel adde new txt file 34a5746 · 1 minute ago 2 Commits

| | | |
|----------------|-------------------|----------------|
| README.md | Initial commit | 13 minutes ago |
| mi-archivo.txt | adde new txt file | 1 minute ago |

README

new_repo_test

Este es una prueba de repositorio

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

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

```

C:\Users\angel\OneDrive\Desktop\New_repo_test\new_repo_test
angel@DESKTOP-V9U2T9A MINGW64 ~/OneDrive/Desktop/New_repo_test/new_repo_test (main)
$ git checkout -b my_new_branch
Switched to a new branch 'my_new_branch'

angel@DESKTOP-V9U2T9A MINGW64 ~/OneDrive/Desktop/New_repo_test/new_repo_test (my_new_br
anch)
$ touch test.py

angel@DESKTOP-V9U2T9A MINGW64 ~/OneDrive/Desktop/New_repo_test/new_repo_test (my_new_br
anch)
$ ls
README.md  mi-archivo.txt  test.py

angel@DESKTOP-V9U2T9A MINGW64 ~/OneDrive/Desktop/New_repo_test/new_repo_test (my_new_br
anch)
$ git add .

angel@DESKTOP-V9U2T9A MINGW64 ~/OneDrive/Desktop/New_repo_test/new_repo_test (my_new_br
anch)
$ git status
On branch my_new_branch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   test.py

angel@DESKTOP-V9U2T9A MINGW64 ~/OneDrive/Desktop/New_repo_test/new_repo_test (my_new_br
anch)
$ git commit -m "adde new py file"
[my_new_branch 24be58f] adde new py file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 test.py

angel@DESKTOP-V9U2T9A MINGW64 ~/OneDrive/Desktop/New_repo_test/new_repo_test (my_new_br
anch)
$ git push origin my_new_branch
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 285 bytes | 142.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'my_new_branch' on GitHub by visiting:
remote:   https://github.com/Isragadiel/new_repo_test/pull/new/my_new_branch
remote:
To https://github.com/Isragadiel/new_repo_test.git
 * [new branch]    my_new_branch -> my_new_branch

angel@DESKTOP-V9U2T9A MINGW64 ~/OneDrive/Desktop/New_repo_test/new_repo_test (my_new_br

```

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"
```

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

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

```
git checkout main
```

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

Este es un cambio en la main branch.

- Guarda los cambios y haz un commit:

```
git add README.md
```

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

Paso 5: Hacer un merge y generar un conflicto

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

```
git merge feature-branch
```

- Se generará un conflicto porque ambos cambios afectan la misma línea del archivo README.md.

Paso 6: Resolver el conflicto

- Abre el archivo README.md en tu editor de texto. Verás algo similar a esto:

```
<<<<<< HEAD
```

Este es un cambio en la main branch.

```
=====
```

Este es un cambio en la feature branch.

```
>>>>>> feature-branch
```

- Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.
- Edita el archivo para resolver el conflicto y guarda los cambios (Se debe borrar lo marcado en verde en el archivo donde estes solucionando el conflicto. Y se debe borrar la parte del texto que no se quiera dejar).
- Añade el archivo resuelto y completa el merge:

```
git add README.md
```

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

Paso 7: Subir los cambios a GitHub

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

```
git push origin main
```

- También sube la feature-branch si deseas:

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

Paso 8: Verificar en GitHub

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

github.com/new

Required fields are marked with an asterisk (*).

Owner * / Repository name *

✓ conflict-exercise is available.

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

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

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

Choose a license

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

car



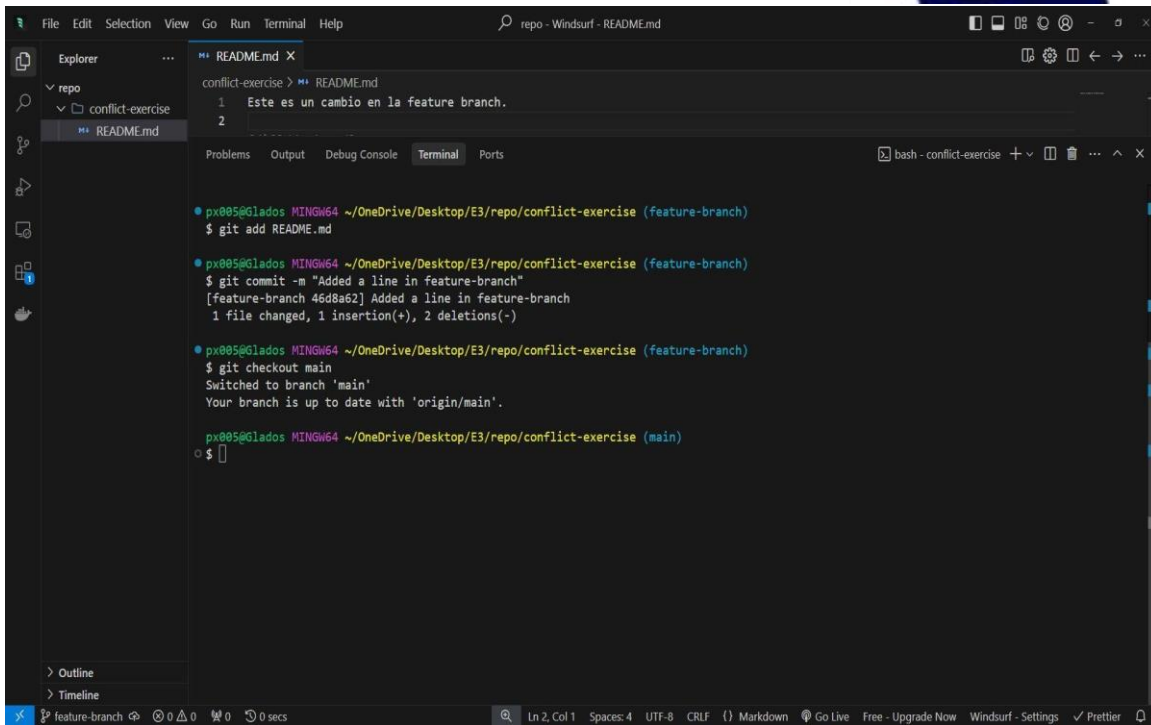
```
File Edit Selection View Go Run Terminal Help repo - Windsurf
Explorer Problems Output Debug Console Terminal Ports
repo
  conflict-exercise
    README.md

px005@Glados MINGW64 ~/OneDrive/Desktop/E3/repo
$ git clone https://github.com/814942/conflict-exercise.git
Cloning into 'conflict-exercise'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

px005@Glados MINGW64 ~/OneDrive/Desktop/E3/repo
$ cd conflict-exercise/

px005@Glados MINGW64 ~/OneDrive/Desktop/E3/repo/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

px005@Glados MINGW64 ~/OneDrive/Desktop/E3/repo/conflict-exercise (feature-branch)
$
```



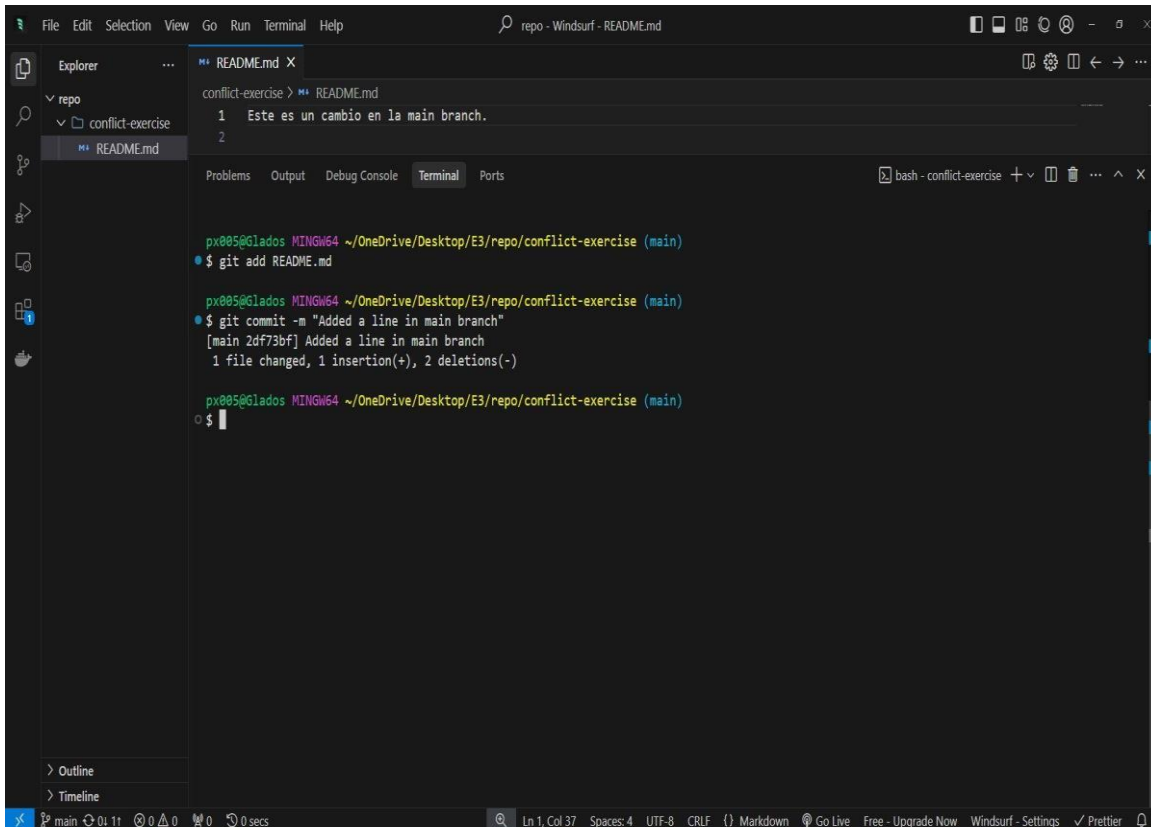
The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal is running a bash shell in the directory `~/OneDrive/Desktop/E3/repo/conflict-exercise` on the `feature-branch`. The user has just added a file `README.md` and committed it with the message "Added a line in feature-branch". The commit hash is `46d8a62`. The terminal output shows the following commands and their results:

```
px885@Gladys MINGW64 ~/OneDrive/Desktop/E3/repo/conflict-exercise (feature-branch)
$ git add README.md

px885@Gladys MINGW64 ~/OneDrive/Desktop/E3/repo/conflict-exercise (feature-branch)
$ git commit -m "Added a line in feature-branch"
[feature-branch 46d8a62] Added a line in feature-branch
1 file changed, 1 insertion(+), 2 deletions(-)

px885@Gladys MINGW64 ~/OneDrive/Desktop/E3/repo/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

px885@Gladys MINGW64 ~/OneDrive/Desktop/E3/repo/conflict-exercise (main)
$
```

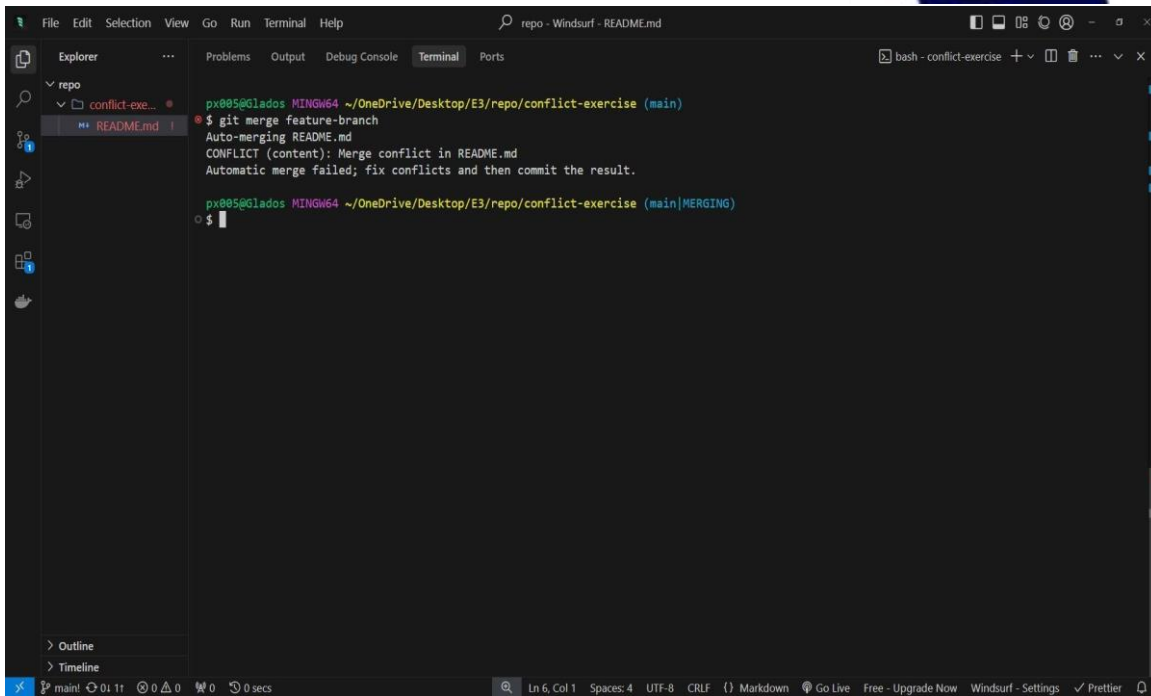


The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal is running a bash shell in the directory `~/OneDrive/Desktop/E3/repo/conflict-exercise` on the `main` branch. The user has just added a file `README.md` and committed it with the message "Added a line in main branch". The commit hash is `2df73bf`. The terminal output shows the following commands and their results:

```
px885@Gladys MINGW64 ~/OneDrive/Desktop/E3/repo/conflict-exercise (main)
$ git add README.md

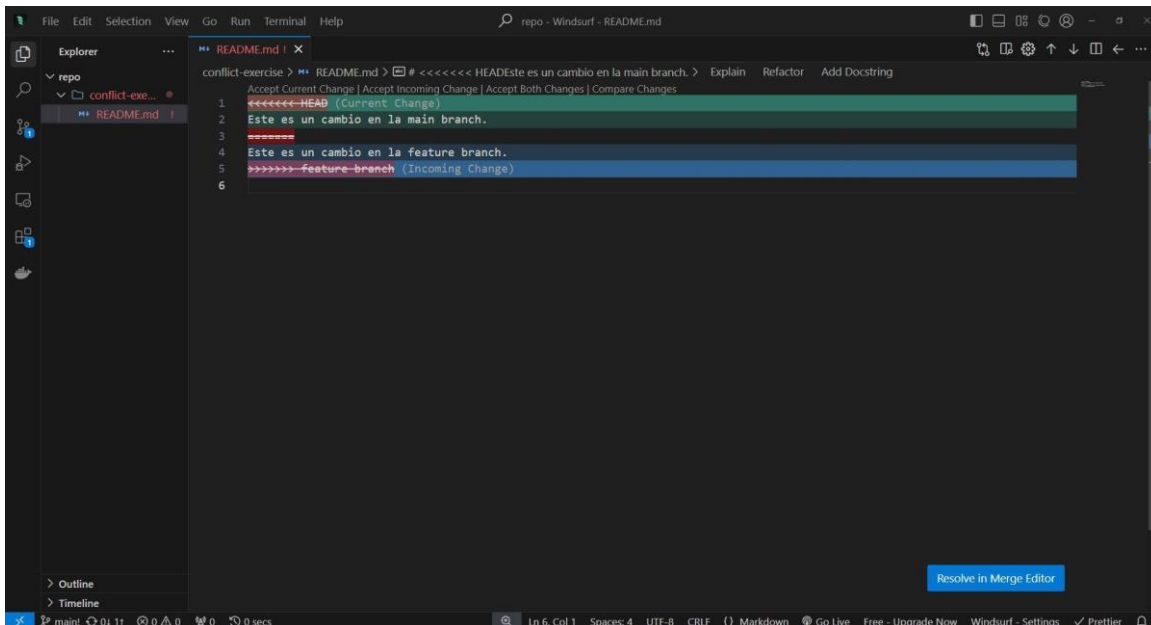
px885@Gladys MINGW64 ~/OneDrive/Desktop/E3/repo/conflict-exercise (main)
$ git commit -m "Added a line in main branch"
[main 2df73bf] Added a line in main branch
1 file changed, 1 insertion(+), 2 deletions(-)

px885@Gladys MINGW64 ~/OneDrive/Desktop/E3/repo/conflict-exercise (main)
$
```



```
File Edit Selection View Go Run Terminal Help repo - Windsurf - README.md
Explorer Problems Output Debug Console Terminal Ports
repo
  conflict-exe...
    README.md
px895@Glados MINGW64 ~/OneDrive/Desktop/E3/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.

px895@Glados MINGW64 ~/OneDrive/Desktop/E3/repo/conflict-exercise (main|MERGING)
$
```



```
File Edit Selection View Go Run Terminal Help repo - Windsurf - README.md
conflict-exercise > README.md > # <<<<<< HEAD Este es un cambio en la main branch. > Explain Refactor Add Docstring
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
1 <<<<<< HEAD (Current Change)
2 Este es un cambio en la main branch.
3 =====
4 Este es un cambio en la feature branch.
5 >>>>>> feature-branch (Incoming Change)
6
```

Resolve in Merge Editor

```
File Edit Selection View Go Run Terminal Help repo - Windsurf - README.md
Explorer Problems Output Debug Console Terminal Ports
repo
  conflict-exercise
    README.md
px005@Gladys MINGW64 ~/OneDrive/Desktop/E3/repo/conflict-exercise (main|MERGING)
$ git add README.md
px005@Gladys MINGW64 ~/OneDrive/Desktop/E3/repo/conflict-exercise (main|MERGING)
$ git commit -m "Resolved merge conflict"
[main f7f7665] Resolved merge conflict
px005@Gladys MINGW64 ~/OneDrive/Desktop/E3/repo/conflict-exercise (main)
$ git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (7/7), 671 bytes | 223.00 KiB/s, done.
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/814942/conflict-exercise.git
5b2eb62..f7f7665 main -> main
px005@Gladys MINGW64 ~/OneDrive/Desktop/E3/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/814942/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/814942/conflict-exercise.git
* [new branch]   feature-branch -> feature-branch
px005@Gladys MINGW64 ~/OneDrive/Desktop/E3/repo/conflict-exercise (main)
$
```

github.com/isragadiel/conflict

conflicto (Public) Pin Unwatch 1 Fork 0

Your main branch isn't protected
Protect this branch from force pushing or deletion, or require status checks before merging. [View documentation.](#) Protect this branch

main 2 Branches 0 Tags Go to file Add file Code

Isragadiel añadió una línea c7bdbfb · 7 minutes ago 3 Commits

README.md añadió una línea 7 minutes ago

README

este es un cambio en la main branch

About
tp conflicto
Readme
Activity
0 stars
1 watching
0 forks

Releases
No releases published
[Create a new release](#)

Packages
No packages published
[Publish your first package](#)

© 2025 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information

Buscar

17°C