

## TRABAJO PRÁCTICO N°2: PROGRAMACIÓN (GIT/GITHUB)

Alumno: Romero Juan Agustin

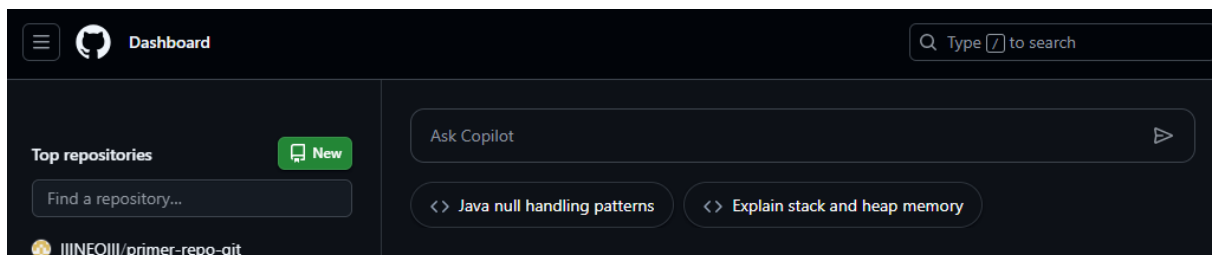
Actividades 1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas) :

### 1. ¿Qué es GitHub?

GitHub es una plataforma de código abierto, que permite alojar repositorios de control de versiones para que las personas almacenen su código, intercambien los mismos con otros desarrolladores o realicen trabajos en conjunto en el desarrollo de software de una forma más organizada.

### 2. ¿Cómo crear un repositorio en GitHub?

Para crear un repositorio en GitHub, primero debemos contar con una cuenta o crear una en caso de no contar con la misma. Esta cuenta te permitirá crear tu repositorio y configurar la misma, como por ejemplo: Si queremos que nuestro repositorio sea Privado o Público.



Habiendo ingresado a nuestra cuenta GitHub, debemos ir a “HOME” que es donde nos encontramos en el ejemplo gráfico del recorte de pantalla, en el cual ingresamos en la caja rectangular color verde, que dice “New” haciendo click en la misma, en la

cual nos dirá si queremos crear un nuevo repositorio. Nos saldrá un formulario para completar de configuraciones previas a crear el mismo.

### 3. ¿Cómo crear una rama en Git?

Para crear una rama en Git, utilizaremos el comando:

```
$ git branch nombreNuevaRama
```

Donde el fragmento “git branch” nos creará la rama y “nombreNuevaRama” es un ejemplo referencial en donde nosotros elegiremos como se llamará esta nueva rama que se creará a continuación.

### 4. ¿Cómo cambiar a una rama en Git?

Para cambiar de rama en Git, utilizaremos el comando:

```
$ git checkout nombreRama
```

En este caso al utilizar el comando “git checkout”, lo que hace este fragmento de código, es mover el apuntador HEAD a la rama “nombreRama”, que es la que utilizamos como ejemplo referencial. Si se quiere crear una nueva rama y saltar a ella en un solo movimiento, utilizaremos el comando:

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

En el cual al agregar el fragmento “-b” hace que esta fuerce el salto directo a esa rama que llamamos “nombre-rama” en este ejemplo.

### 5. ¿Cómo fusionar ramas en Git?

Para fusionar dos ramas en Git, es necesario posicionarse en la rama deseada y utilizar el comando:

```
$ git merge nuevaRama
```

Ejemplo:

- 1) primero debemos estar en la rama a la que se le quiere fusionar otra rama.  
Ejemplo: rama “Master o Main”.

En el cual debemos movernos a ella con el comando:

```
$ git checkout master
```

2) Estando en la rama “Master o Main”, utilizaremos el comando:

```
$ git merge nuevaRama
```

En el cual este comando “merge” lo que hará es fusionar la rama “Master o Main” que es la rama en la cual nos encontramos posicionado, con la rama “nuevaRama” que es la rama que le estaremos indicando que nos funcione el comando. Esta acción lo que hará es que se incorporen los cambios de “nuevaRama” en “Master o Main”.

6. ¿Cómo crear un commit en Git?

Para crear un “commit” primero debemos entender que significa este. Un “commit” es un registro en el cual guardamos los cambios realizados en nuestro repositorio en un momento dado, quedando como una marca en nuestro historial del proyecto que estamos realizando.

comando a utilizar:

```
$git commit -m “Mensaje”
```

Donde “mensaje”, es un título o mensaje que dejamos como texto informativo que describe qué cambios o acciones hemos hecho.

7. ¿Cómo enviar un commit a GitHub?

Para enviar un commit a GitHub debemos utilizar el comando:

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

Con este comando lo que hacemos, es empujar nuestro commit a nuestro repositorio.

8. ¿Qué es un repositorio remoto?

Un repositorio remoto es una copia de tu proyecto que se encuentra alojado en una base de datos en la red, esto permite poder colaborar con otras personas. Pudiendo enviar o descargar datos de ellos cada vez que necesites compartir tu trabajo.

9. ¿Cómo agregar un repositorio remoto a Git?

Utiliza el comando “\$ git remote add” para vincular tu repositorio local con un repositorio remoto y asignar un nombre a ese repositorio remoto, ejemplo:

```
$ git remote add nombre url
```

Donde en “nombre” agregaremos el nombre que queramos y en “url”, agregaremos la dirección “URL”. Este nombre nos sirve para buscarlo o llamarlo en la terminal de una forma más rápida y fácil que colocando la dirección “URL”, que esta suele ser más larga y compleja de tipear.

#### 10. ¿Cómo empujar cambios a un repositorio remoto?

Para eso utilizamos el comando:

```
$ git push -u origin main
```

Donde “origin” es el nombre por defecto del repositorio remoto.

Dónde “main” es el nombre de la rama. Puedes cambiarlo si estás usando otra rama

```
$ git push origin nombreRama
```

#### 11. ¿Cómo tirar de cambios de un repositorio remoto?

Para poder tirar los cambios del repositorio remoto, usa el comando:

```
$ git pull
```

Si estás trabajando en la rama “ Master o Main” usamos el comando:

```
$ git pull origin master
```

#### 12. ¿Qué es un fork de repositorio?

En github existen repositorios de otros usuarios en el cual nosotros podemos hacer una copia y modificarlas, seleccionando un repositorio de nuestro interés, este puede ser cualquier proyecto de código abierto que quieras modificar o utilizar como base para tu propio proyecto.

#### 13. ¿Cómo crear un fork de un repositorio?

Para crear un fork, primero debemos ir al repositorio que queremos forkar. En la esquina superior derecha de la página, se encontrará un botón que dice “Fork” en el cual hacemos click y seleccionas donde guardar el fork y listo!.

#### 14. ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?

Iremos a “Pull requests” allí daremos click en new pull request, luego nos saldrá una ventana donde se reflejarán los cambios que hemos hecho del repositorio original, hacemos click en “Create pull request”, y colocamos un mensaje para saber en detalle qué cambios hemos realizado.

#### 15. ¿Cómo aceptar una solicitud de extracción?

Dentro de la pestaña “Pull Requests” veremos una lista de mensajes de solicitudes de cambios, en el cual podemos observarlos y considerar si deseamos hacer los cambios solicitados, hasta responderle al usuario. Podemos clicar en Merge pull request y de esta manera sumar su repositorio.

#### 16. ¿Qué es un etiqueta en Git?

Una etiqueta es una herramienta que nos permite categorizar cambios específicos y de ésta manera tener mejor seguimiento de ellos.

#### 17. ¿Cómo crear una etiqueta en Git?

Podremos crear una etiqueta con el comando tag:

```
git tag nombre-tag. .
```

#### 18. ¿Cómo enviar una etiqueta a GitHub?

Una vez creada la etiqueta debemos utilizar el comando:

```
$ git push origin nombreEtiqueta
```

(origin es el nombre del repositorio remoto y “nombreEtiqueta” es el nombre de la etiqueta.)

Para empujar todas las etiquetas creadas, usar el comando:

```
$ git push origin --tags
```

#### 19. ¿Qué es un historial de Git?

El historial de Git es un registro de todos los cambios realizados en un repositorio. Cada cambio en el repositorio se guarda como un commit, y cada commit contiene información sobre el estado del proyecto en un momento específico, incluyendo: Identificador del commit como: Autor, Fecha de realización Mensaje enviado

## 20. ¿Cómo ver el historial de Git?

Esto lo conseguimos con el comando:

```
$ git log
```

Al escribir en la terminal de Git podremos ver el historial de commits, estando situados en la carpeta de nuestro proyecto.

El listado de commits estará invertido, nos aparecerán los últimos commit primero en la lista.

El comando:

```
$ git log --oneline
```

Muestra un resumen del historial de commits en un repositorio Git.

Para ver un número de logs determinado introducimos ese número como opción, con el signo "-" delante. Por ejemplo, esto muestra los últimos tres commits con el comando:

```
$ git log -3
```

Si queremos que el log también nos muestre los cambios en el código de cada commit podemos usar la opción -p. Esta opción genera una salida mucho más larga, por lo que seguramente nos tocará movernos en la salida con los cursores y usaremos CTRL + Z para salir.

```
$ git log -2 -p
```

## 21. ¿Cómo buscar en el historial de Git?

Para buscar en el historial de commits de Git, podemos utilizar varios comandos y opciones que te permiten filtrar y localizar commits específicos.

Para encontrar commits hechos por un autor específico, usamos --author:

```
$ git log --author="NombreAutor"
```

Para buscar commits que han modificado un archivo específico, usa git log seguido del nombre del archivo:

```
$git log -- nombreArchivo
```

Para buscar commits que contengan una palabra o frase específica en el mensaje de commit, usa git log con la opción `--grep`:

```
$ git log --grep="palabra clave"
```

Para buscar commits en un rango de fechas específico, usa las opciones `--since` y `--until`:

```
$ git log --since="2025-01-01" --until="2025-01-31"
```

## 22. ¿Cómo borrar el historial de Git?

Con el comando `git reset` podemos borrar, pero para eso debemos saber que deseamos quitar y no borrar cosas que no deseamos, para eso tenemos una lista de comandos `reset` con sus variables:

Quita del stage todos los archivos de esa carpeta:

```
$ git reset nombreCarpeta/
```

Quita del stage todos los archivos y carpetas del proyecto:

```
$ git reset
```

Quita del stage el archivo indicado:

```
$ git reset nombreArchivo
```

Quita todos los archivos que cumplan con la condición indicada previamente dentro de esa carpeta del stage.

```
$ git reset nombreCarpeta/*.extensión
```

Quita ese archivo del stage (que a la vez está dentro de una carpeta).

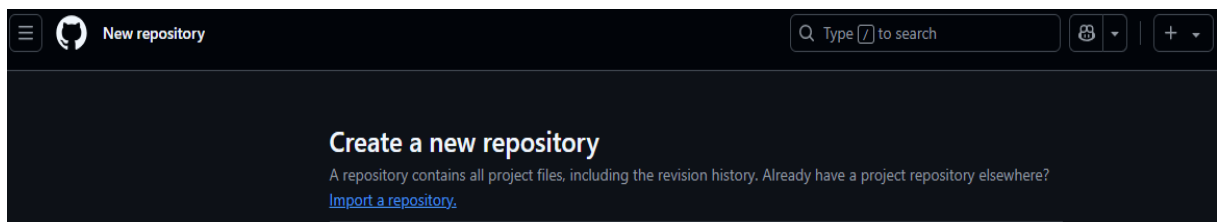
```
$ git reset nombreCarpeta/nombreArchivo
```

23. ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es un repositorio al que pueden tener solo acceso usuarios que tu autorices. Esto hace que mantenga la privacidad de los colaboradores y sus proyectos antes de que estos se hagan público o no.

24. ¿Cómo crear un repositorio privado en GitHub?

- 1) Ingresa a la página de GitHub y ve a la parte de creación de repositorio
- 2) En la esquina superior de la derecha de la página, has click sobre el simbolo “+” y selecciona “New Repository” o has click en “New”:



Completamos la información del repositorio:

-nombre del repositorio

-descripción

Seleccionar la configuración de privacidad:




## 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 \***

 IIINEOIII /

Great repository names are short and memorable. Need inspiration? How about **silver-palm-tree** ?

**Description** (optional)

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

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

**Initialize this repository with:**

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

**Add .gitignore**

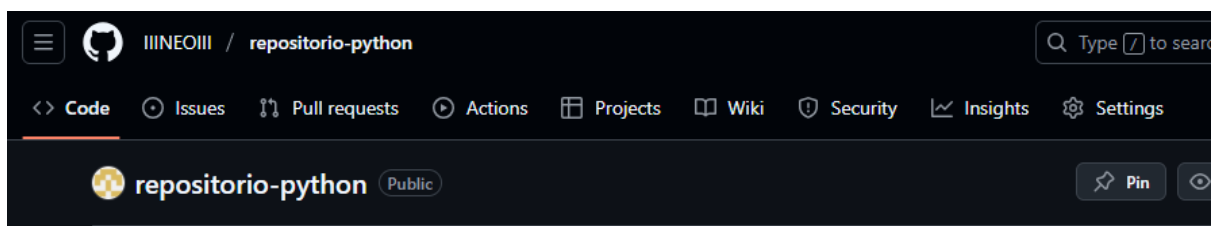
.gitignore template: **None** ▼

Como podemos apreciar en la imagen, en la parte de “Private” en la cual está tildada con color azul. Esto generará que sea accesible para los colaboradores que tu elijas.

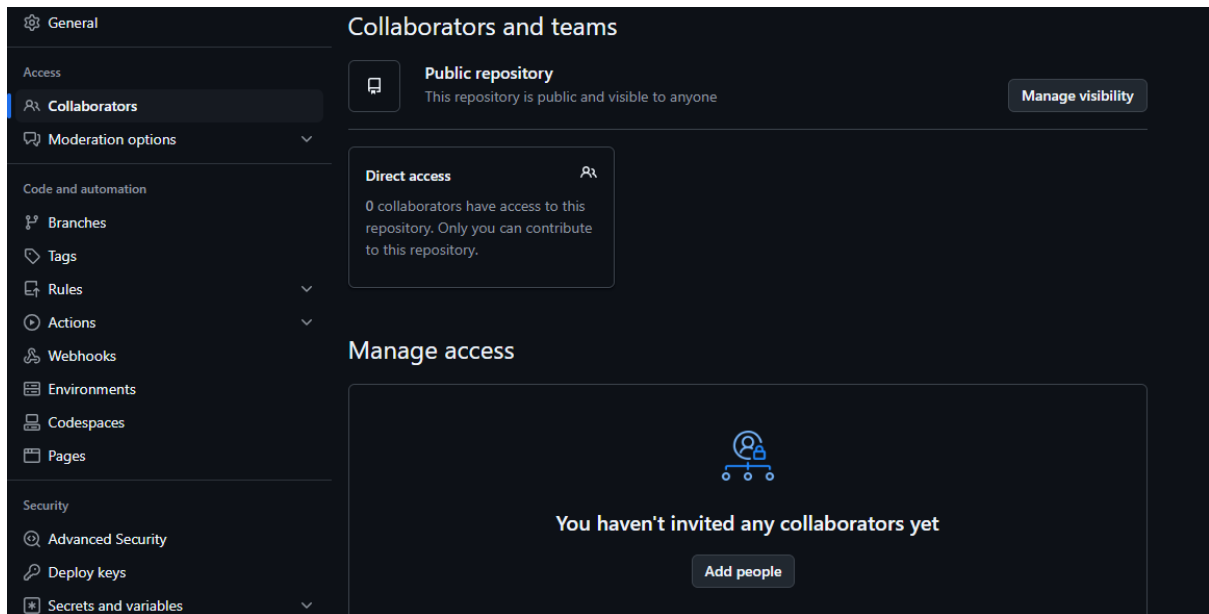
25. ¿Cómo invitar a alguien a un repositorio privado en GitHub?

Invitar a alguien a un repositorio privado en GitHub es un proceso sencillo, pero requiere permisos adecuados.

Accede al repositorio, haz clic en la pestaña "Settings" que se encuentra, en la esquina superior derecha



Selecciona "Collaborators" en el menú de la izquierda. Esto te llevará a la página donde puedes administrar colaboradores.



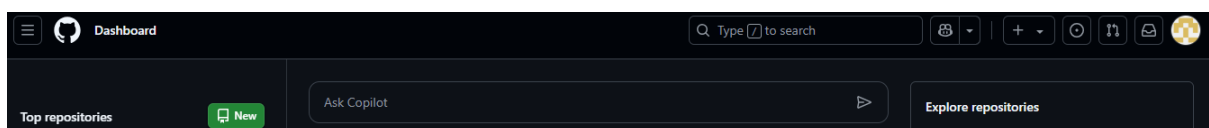
En la sección "Collaborators", haz clic en el botón "Add people" e ingresa el nombre de usuario de GitHub de la persona que deseas invitar. Selecciona el nivel de acceso que deseas otorgar: Read, Triage, Write, Maintain, o Admin. Haz clic en el botón "Add" para enviar la invitación.

26. ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un repositorio cuyo contenido es accesible a cualquier persona en Internet. A diferencia de un repositorio privado, que está restringido a un grupo específico de colaboradores, un repositorio público permite que cualquier persona pueda ver, clonar y si tienen los permisos adecuados, contribuir al proyecto.

27. ¿Cómo crear un repositorio público en GitHub?

- 1) Inicia sesión en GitHub
- 2) Ingresa a la página de creación de repositorios
- 3) En la esquina superior derecha de la página principal, debes hacer clic en el botón "+" y seleccionar "New Repository" o hacer clic en "New" que es la casilla color verde:



Completar la información del repositorio

- nombre del repositorio.
- descripción.


Seleccionar la configuración de privacidad:

**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 \***

 IIIINEOIII /

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

**Description** (optional)

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

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

**Initialize this repository with:**

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

**Add .gitignore**


.gitignore template: **None** ▾

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

**Choose a license**

License: **None** ▾

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

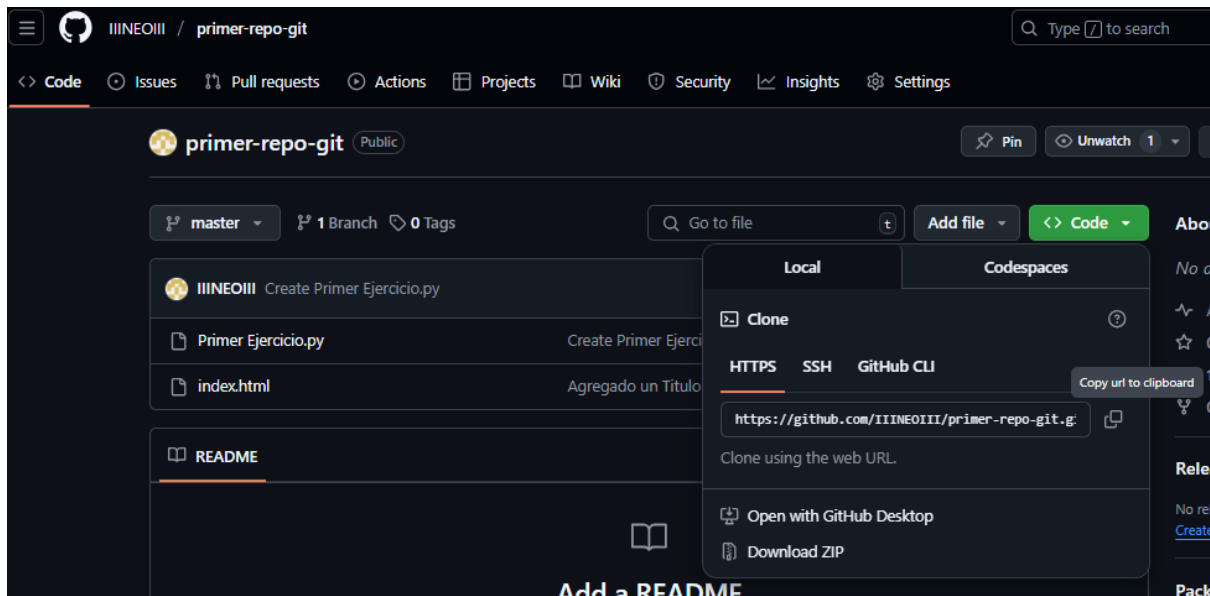
 You are creating a public repository in your personal account.

[Create repository](#)

tildando en “Public” con color azul, haremos que nuestro repositorio sea publico y visible para todos.

28. ¿Cómo compartir un repositorio público en GitHub?

Proporciona el enlace directo , para esto accede a tu repositorio, copia la URL de tu repositorio que se encuentra en un cuadro de texto que dice "<> Code" en color verde:



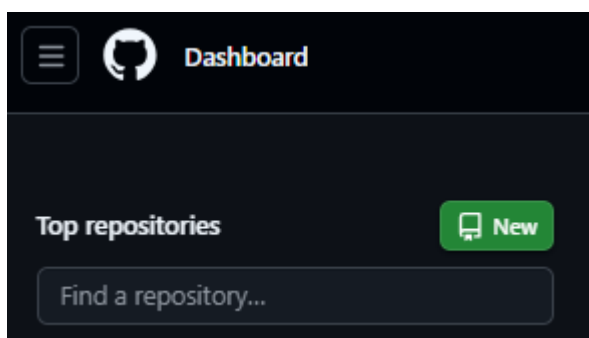
Puedes copiar la URL directamente haciendo clic en el botón de copiar a la derecha de la URL, ejemplo:

<https://github.com/IIINEOIII/primer-repo-git.git>

2) Realizar la siguiente actividad:

Crear un repositorio.

Selecciona la casilla color verde que dice “New”.



Dale un nombre al repositorio.

Elige que el repositorio sea público.

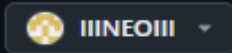
Inicializa el repositorio con un archivo.

# Create a new repository

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

Required fields are marked with an asterisk (\*).

Owner \*



Repository name \*

/ systemRepositorio

systemRepositorio is available.

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

Description (optional)

Este repositorio es de Arquitectura y OS



Public

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



Private

You choose who can see and commit to this repository.

Initialize this repository with:

☒ Add a README file

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

Add .gitignore

.gitignore template: None

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

Choose a license

License: None

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

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



systemRepositorio

Public

Pin

Unwatch 1

main

1 Branch 0 Tags

Go to file



Add file

<> Code



Initial commit

9299bed · now

1 Commit



README.md

Initial commit

now



README



## systemRepositorio

Este repositorio es de Arquitectura y OS

- Agregando un Archivo o 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).

### **Clonar el Repositorio a tu Computadora**

Utilizando el comando:

\$ git clone <https://github.com/IIINEOIII/UTNPRO1U2ACT2.git>

```
kayn@DESKTOP-7M270V8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO (master)
$ git clone https://github.com/IIINEOIII/UTNPRO1U2ACT2.git
Cloning into 'UTNPRO1U2ACT2'...
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.
```

### **Agregar un Archivo al Repositorio**

- 1) Ingresa a la carpeta del repositorio con el comando:

\$ cd UTNPRO1U2ACT2

```
kayn@DESKTOP-7M270V8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO (master)
$ cd UTNPRO1U2ACT2
```

- 2) Crear un archivo simple llamado "mi-archivo.txt"

\$ echo "Este es mi archivo de prueba" > mi-archivo.txt

```
kayn@DESKTOP-7M270V8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT2 (main)
$ echo "Este es mi archivo de prueba" > mi-archivo.txt
```

- 3) Agregar el archivo al área de preparación:

\$ git add .

```
kayn@DESKTOP-7M270V8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT2 (main)
$ git add .
```

4) Realiza el commit con un mensaje descriptivo:

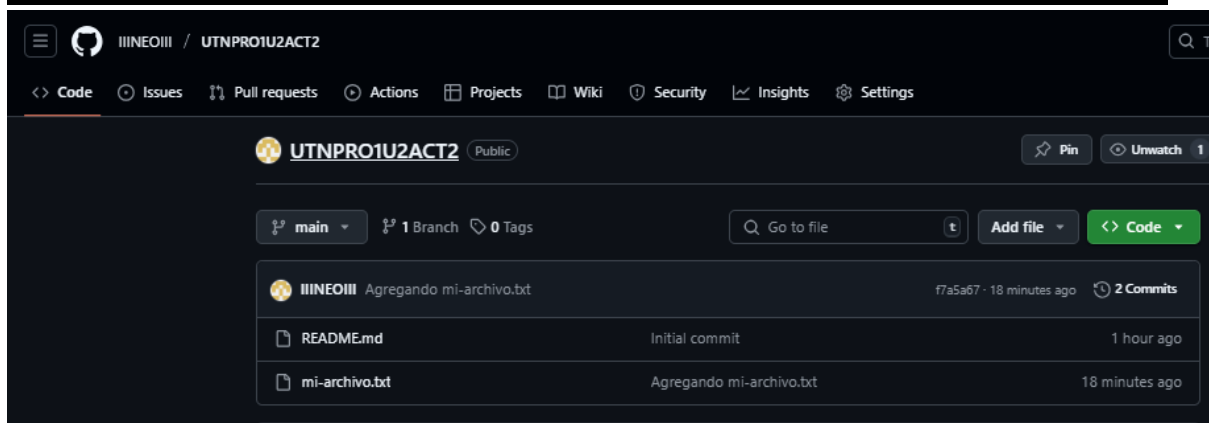
```
$ git commit -m "Agregando mi-archivo.txt"
```

```
kayn@DESKTOP-7M270V8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT2 (main)
$ git commit --amend -m "Agregando mi-archivo.txt"
[main f7a5a67] Agregando mi-archivo.txt
Date: Sun Apr 6 15:26:26 2025 -0300
1 file changed, 1 insertion(+)
create mode 100644 mi-archivo.txt
```

5) Sube los cambios al repositorio en GitHub:

```
$ git push origin main
```

```
kayn@DESKTOP-7M270V8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT2 (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 328 bytes | 328.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/IIINEOIII/UTNPRO1U2ACT2.git
cc83c1c..29dbf04 main -> main
```



## Crear una Nueva Rama (Branch)

1) Crea una nueva rama llamada Nueva-Rama:

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

```
kayn@DESKTOP-7M27OV8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT2 (main)
$ git checkout -b nueva-rama
Switched to a new branch 'nueva-rama'
```

```
kayn@DESKTOP-7M27OV8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT2 (main)
$ git checkout -b nueva-rama
Switched to a new branch 'nueva-rama'
```

```
kayn@DESKTOP-7M27OV8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT2 (nueva-rama)
$ git branch
  main
* nueva-rama
```

2) Agrega un nuevo archivo o edita uno existente:

\$ echo "Este es un archivo en nueva rama" > mi-archivo.txt

```
kayn@DESKTOP-7M27OV8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT2 (nueva-rama)
$ echo "Modificaicon hecha desde nueva-rama" > mi-archivo.txt
```

3) Agrega y confirma los cambios:

\$ git add .

```
kayn@DESKTOP-7M27OV8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT2 (nueva-rama)
$ git add .
```

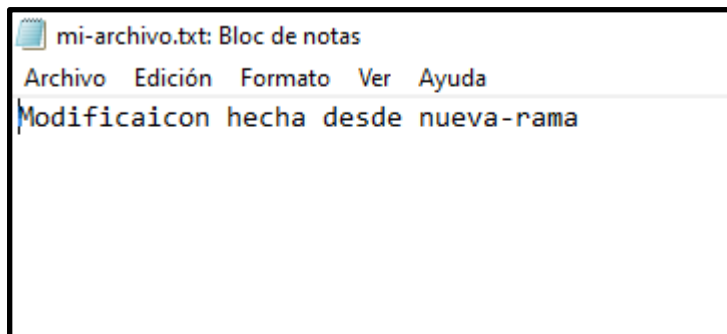
```
kayn@DESKTOP-7M27OV8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT2 (nueva-rama)
$ git status
On branch nueva-rama
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   mi-archivo.txt
```

\$ git commit -m "Agregando modificacion desde nueva-rama"

```
kayn@DESKTOP-7M27OV8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT2 (nueva-rama)
$ git commit -m "Agregando modificacion desde nueva-rama"
[nueva-rama c635499] Agregando modificacion desde nueva-rama
1 file changed, 1 insertion(+), 1 deletion(-)
```

```
kayn@DESKTOP-7M27OV8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT2 (nueva-rama)
$ git status
On branch nueva-rama
nothing to commit, working tree clean
```





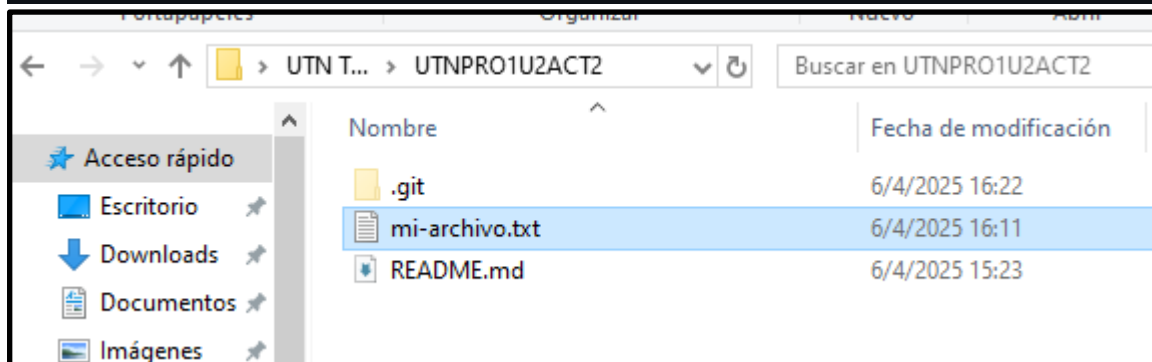
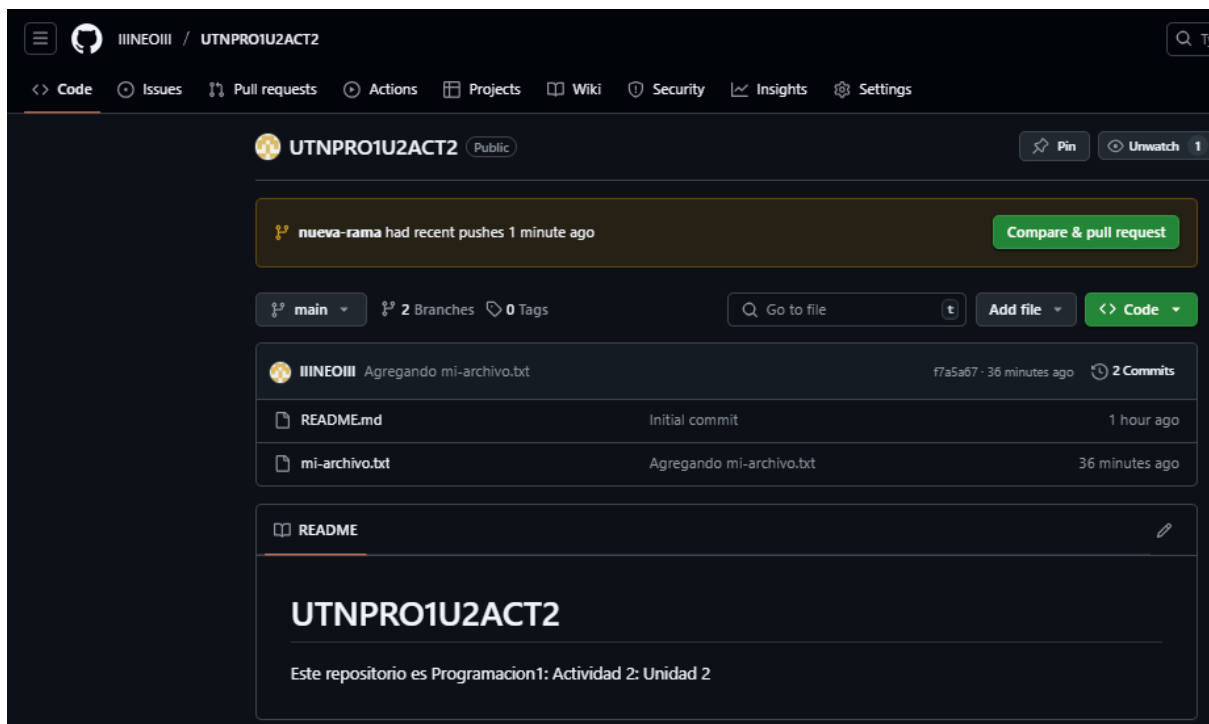
4) Sube la rama al repositorio en GitHub:

\$ git push origin nueva-rama

```
kayn@DESKTOP-7M27OV8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT2 (nueva-rama)
$ git push origin nueva-rama
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 340 bytes | 170.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'nueva-rama' on GitHub by visiting:
remote:   https://github.com/IIINEOIII/UTNPRO1U2ACT2/pull/new/nueva-rama
remote:
To https://github.com/IIINEOIII/UTNPRO1U2ACT2.git
 * [new branch]      nueva-rama -> nueva-rama
```

\$ git status

```
kayn@DESKTOP-7M27OV8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT2 (nueva-rama)
$ git status
On branch nueva-rama
nothing to commit, working tree clean
```

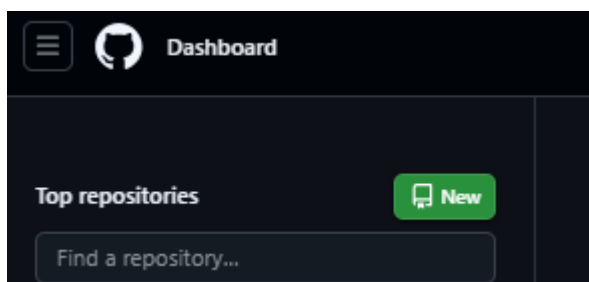


3) Realizar la siguiente actividad:

1) Crear un repositorio en GitHub

1) Ve a GitHub e inicia sesión en tu cuenta.

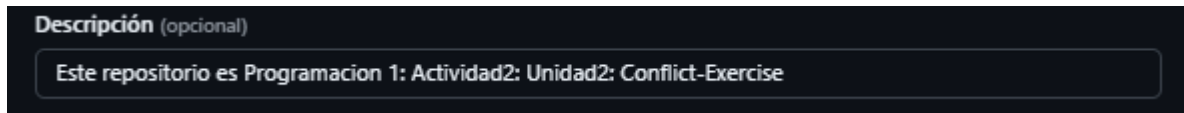
2) Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.



3) Asigna un nombre al repositorio, por ejemplo, conflict-exercise.



4) Opcionalmente, añade una descripción.



5) Marca la opción "Initialize this repository with a README".

6) Haz clic en "Create repository".

Paso 2: Clonar el repositorio a tu máquina local

1) Copia la URL del repositorio (usualmente algo como <https://github.com/tuusuario/conflict-exercise.git>).

<https://github.com/IIINEOIII/UTNPRO1U2ACT3.git>

2) Abre la terminal o línea de comandos en tu máquina.

3) Clona el repositorio usando el comando: `git clone`

<https://github.com/IIINEOIII/UTNPRO1U2ACT3.git>

```
kayn@DESKTOP-7M270V8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO (master)
$ git clone https://github.com/IIINEOIII/UTNPRO1U2ACT3.git
Cloning into 'UTNPRO1U2ACT3'...
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.
```

\$ `git clone` <https://github.com/IIINEOIII/UTNPRO1U2ACT3.git>

4) Entra en el directorio del repositorio: `cd UTNPRO1U2ACT3`

\$ `cd UTNPRO1U2ACT3`

```
kayn@DESKTOP-7M270V8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO (master)
$ cd UTNPRO1U2ACT3

kayn@DESKTOP-7M270V8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT3 (main)
$
```

Paso 3: Crear una nueva rama y editar un archivo

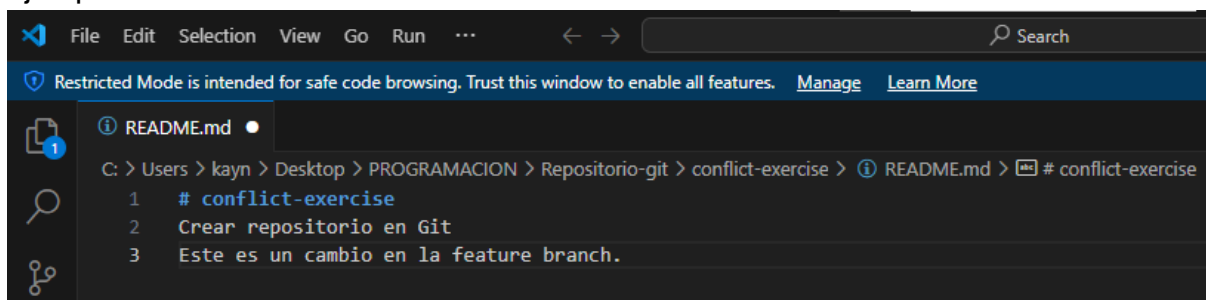
1) Crea una nueva rama llamada feature-branch:

\$ git checkout -b feature-branch

```
kayn@DESKTOP-7M270V8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT3 (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

kayn@DESKTOP-7M270V8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT3 (feature-branch)
$
```

2) 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.



3) Guarda los cambios y haz un commit:

\$ git add README.md

```
kayn@DESKTOP-7M270V8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT3 (feature-branch)
$ git add README.md
```

\$ git commit -m "Added a line in feature-branch"

```
kayn@DESKTOP-7M270V8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT3 (feature-branch)
$ git commit -m "Added a line in feature-bracnch"
[feature-branch d60f5bc] Added a line in feature-bracnch
1 file changed, 2 insertions(+), 1 deletion(-)
```

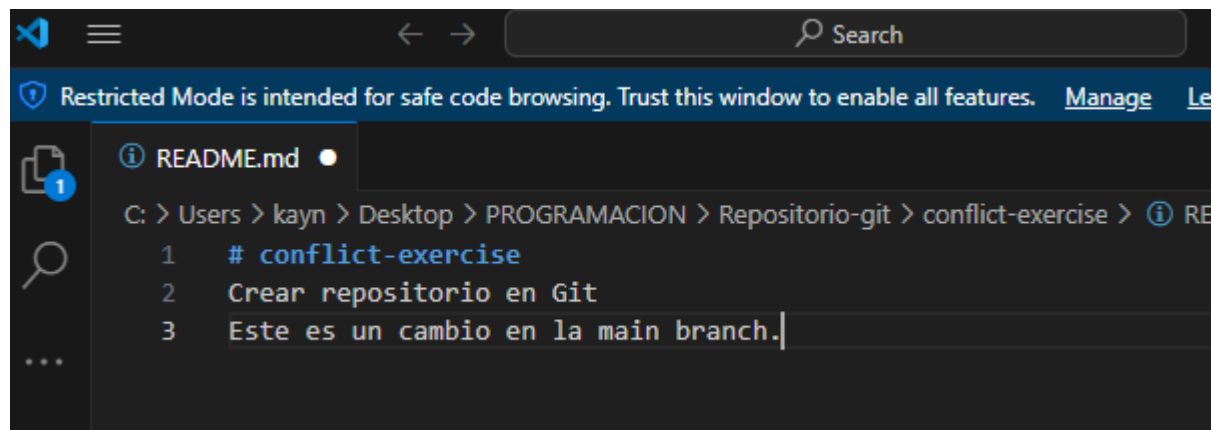
Paso 4: Volver a la rama principal y editar el mismo archivo

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

```
$ git checkout main
```

```
kayn@DESKTOP-7M270V8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT3 (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

2) Edita el archivo README.md de nuevo, añadiendo una línea diferente: Este es un cambio en la main branch.



3) Guarda los cambios y haz un commit:

```
$ git add README.md
```

```
kayn@DESKTOP-7M270V8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT3 (main)
$ git add README.md

kayn@DESKTOP-7M270V8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT3 (main)
```

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

```
kayn@DESKTOP-7M270V8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT3 (main)
$ git commit -m "Added a line in main branch"
[main 27c7f9a] Added a line in main branch
1 file changed, 2 insertions(+), 1 deletion(-)
```

Paso 5: Hacer un merge y generar un conflicto

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

```
$ git merge feature-branch
```

```
kayn@DESKTOP-7M270V8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT3 (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.

kayn@DESKTOP-7M270V8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT3 (main|MERGING)
$
```

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

#### Paso 6: Resolver el conflicto

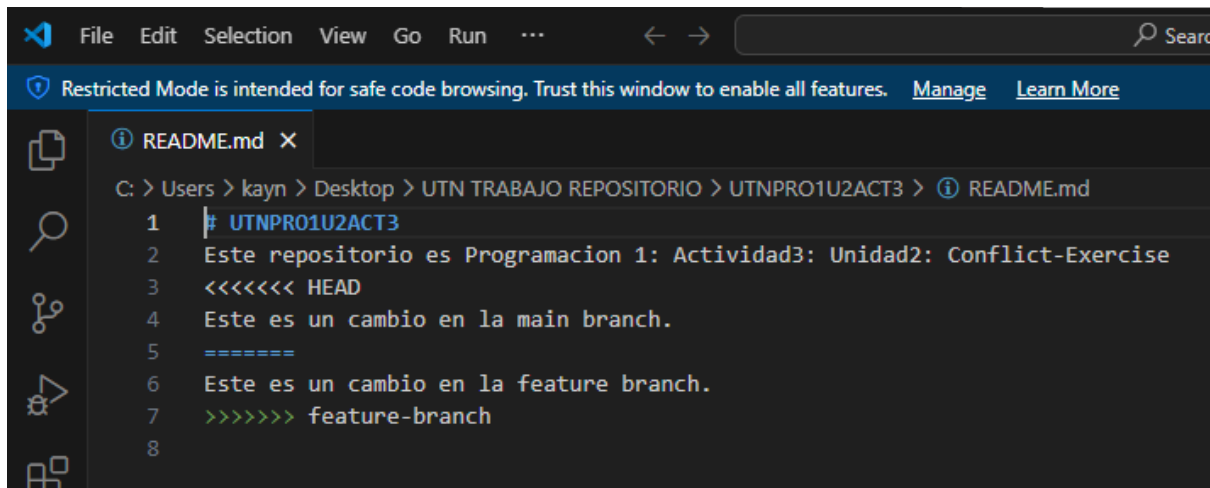
1) Abre el archivo README.md en tu editor de texto. Verás algo similar a esto:

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

Este es un cambio en la main branch.

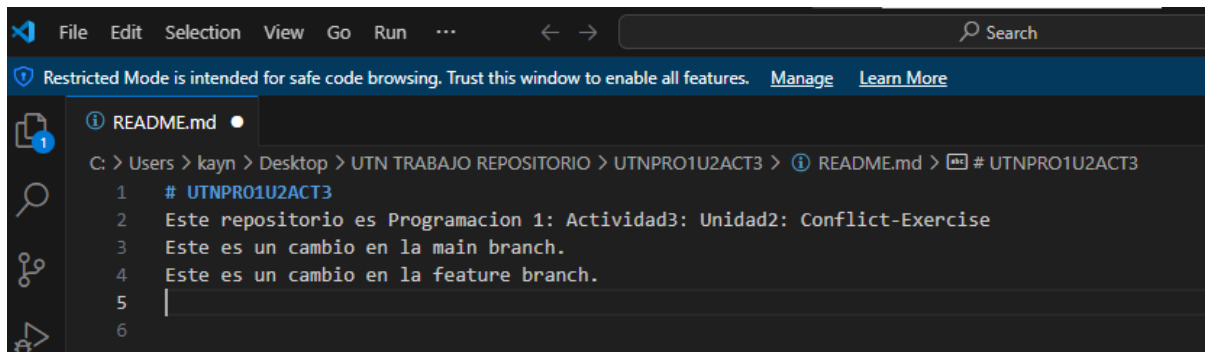
```
===== Este es un cambio en la feature branch.
```

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



2) Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.

3) 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).



4) Añade el archivo resuelto y completa el merge:

```
$ git add README.md
```

```
kayn@DESKTOP-7M27OV8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT3 (main|MERGING)
$ git add README.md
```

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

Paso 7: Subir los cambios a GitHub

```
kayn@DESKTOP-7M27OV8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT3 (main|MERGING)
$ git commit -m "Resolved merge conflict"
[main 146cb10] Resolved merge conflict
```

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

```
$ git push origin main
```

```
kayn@DESKTOP-7M27OV8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT3 (main)
$ git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 801 bytes | 400.00 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/IIINEOIII/UTNPRO1U2ACT3.git
   d372e3a..146cb10  main -> main

kayn@DESKTOP-7M27OV8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT3 (main)
$
```

2) También sube la feature-branch si deseas:

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

```

kayn@DESKTOP-7M27OV8 MINGW64 ~/Desktop/UTN TRABAJO REPOSITORIO/UTNPRO1U2ACT3 (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/IIINEOIII/UTNPRO1U2ACT3/pull/new/feature-branch
remote:
To https://github.com/IIINEOIII/UTNPRO1U2ACT3.git
 * [new branch]      feature-branch -> feature-branch

```

## Paso 8: Verificar en GitHub

1) Ve a tu repositorio en GitHub y revisa el archivo README.md para confirmar que los cambios se han subido correctamente.

2) Puedes revisar el historial de commits para ver el conflicto y su resolución.

