

# Ejercicios - Programación

---

## TP 2 Git

Santiago Nicolas Nievas

1)

### ¿Qué es GitHub?

GitHub es una plataforma online que sirve para guardar y gestionar proyectos de desarrollo, usando un sistema llamado Git. Lo usamos mucho para trabajar en grupo, porque nos permite ver los cambios que hacemos en el código, colaborar con otros compañeros y mantener todo ordenado. Además, se pueden subir versiones del proyecto, compartirlo o mantenerlo privado.

### ¿Cómo crear un repositorio en GitHub?

Primero entramos a nuestra cuenta de GitHub. Después, hacemos clic en el símbolo de “+” arriba a la derecha y elegimos la opción “New repository”. Ahí le ponemos un nombre al repositorio, si queremos una descripción, y elegimos si va a ser público o privado. Opcionalmente, podemos marcar para que se cree con un archivo README. Por último, le damos a “Create repository”.

### ¿Cómo crear una rama en Git?

Para crear una rama nueva, usamos el comando:

```
git branch nombre_de_la_rama
```

Esto lo que hace es crear una copia del proyecto desde donde estamos, para que podamos hacer cambios sin tocar la rama principal.

### ¿Cómo cambiar a una rama en Git?

Podemos usar el siguiente comando:

```
git checkout nombre_de_la_rama
```

O también, si estamos usando una versión más nueva de Git:

```
git switch nombre_de_la_rama
```

Así nos movemos a la rama que queremos modificar o revisar.

### ¿Cómo fusionar ramas en Git?

Primero nos movemos a la rama donde queremos traer los cambios (por ejemplo, main) y después usamos:

```
git merge nombre_de_la_rama
```

Esto va a unir los cambios de esa rama en la que estamos trabajando.

### ¿Cómo crear un commit en Git?

Primero agregamos los archivos al área de preparación:

```
git add .
```

Y después hacemos el commit con:

```
git commit -m "Descripción de lo que hicimos"
```

Esto deja registrado el cambio, con un mensaje que explique qué hicimos.

### **¿Cómo enviar un commit a GitHub?**

Una vez hecho el commit, usamos:

```
git push origin nombre_de_la_rama
```

Esto sube nuestros cambios al repositorio que tenemos en GitHub.

### **¿Qué es un repositorio remoto?**

Es básicamente una copia del proyecto que está guardada en internet (por ejemplo, en GitHub), y que nos permite trabajar desde distintos lugares o en equipo, sincronizando los cambios.

### **¿Cómo agregar un repositorio remoto a Git?**

Si tenemos un repositorio en GitHub, lo podemos vincular así:

```
git remote add origin https://github.com/usuario/repositorio.git
```

Ese “origin” es un nombre que se le da por defecto al repositorio remoto.

### **¿Cómo empujar cambios a un repositorio remoto?**

El comando es:

```
git push origin nombre_de_la_rama
```

Esto sube los cambios que hicimos en nuestra máquina al repositorio de GitHub.

### **¿Cómo tirar de cambios de un repositorio remoto?**

Si queremos traer los cambios que se hicieron en GitHub al proyecto local, usamos:

```
git pull origin nombre_de_la_rama
```

Así se sincronizan los cambios que hayan subido otros.

### **¿Qué es un fork de repositorio?**

Un fork es una copia de un proyecto que está en GitHub, pero que se guarda en nuestra cuenta. Sirve para hacer pruebas o proponer cambios sin modificar el original.

### **¿Cómo crear un fork de un repositorio?**

Entramos al repositorio que queremos forkar y hacemos clic en el botón que dice “Fork”. Automáticamente se crea una copia del proyecto en nuestra cuenta.

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

Una vez que hicimos cambios en nuestra copia (el fork), vamos al repositorio original, entramos a la sección de “Pull requests” y hacemos clic en “New pull request”. Ahí comparamos los cambios y mandamos la solicitud para que el dueño del repositorio la revise.

### **¿Cómo aceptar una solicitud de extracción?**

Entramos al repositorio, vamos a la pestaña de “Pull requests”, abrimos la que queremos aceptar, revisamos los cambios y si está todo bien, hacemos clic en “Merge pull request” para aprobarla.

### **¿Qué es una etiqueta en Git?**

Una etiqueta o “tag” es una forma de marcar un punto específico en la historia del proyecto, por ejemplo cuando lanzamos una versión estable. Es útil para tener un control de versiones.

### **¿Cómo crear una etiqueta en Git?**

```
git tag nombre_etiqueta
```

O también:

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

### **¿Cómo enviar una etiqueta a GitHub?**

```
git push origin nombre_etiqueta
```

O si queremos subir todas las etiquetas juntas:

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

### **¿Qué es un historial de Git?**

Es el registro de todos los commits que se hicieron en el proyecto. Permite ver qué se cambió, cuándo y quién lo hizo.

### **¿Cómo ver el historial de Git?**

```
git log
```

También se puede usar `git log --oneline` para verlo más resumido.

### **¿Cómo buscar en el historial de Git?**

```
git log --grep="palabra_clave"
```

Eso nos permite encontrar commits que tengan esa palabra en el mensaje.

### **¿Cómo borrar el historial de Git?**

No es algo recomendable, pero si realmente se necesita reiniciar el historial, se puede hacer una rama nueva y borrar el resto:

```
git checkout --orphan nueva_rama
```

```
git add .
```

```
git commit -m "Nuevo comienzo"
```

```
git branch -D main
```

```
git branch -m main
```

```
git push -f origin main
```

### **¿Qué es un repositorio privado en GitHub?**

Es un repositorio que solo pueden ver y acceder las personas que nosotros autorizamos. Es ideal para proyectos personales o trabajos que todavía no queremos mostrar públicamente.

### **¿Cómo crear un repositorio privado en GitHub?**

Al momento de crear un nuevo repositorio, simplemente seleccionamos la opción “Private” y después hacemos clic en “Create repository”.

### **¿Cómo invitar a alguien a un repositorio privado en GitHub?**

Vamos al repositorio, entramos a “Settings”, después a “Collaborators” o “Colaboradores” y agregamos el usuario de GitHub de la persona que queremos invitar.

### **¿Qué es un repositorio público en GitHub?**

Es un repositorio que puede ser visto por cualquier persona, aunque no tengan cuenta en GitHub. Es útil para compartir proyectos con la comunidad o hacer código abierto.

### **¿Cómo crear un repositorio público en GitHub?**

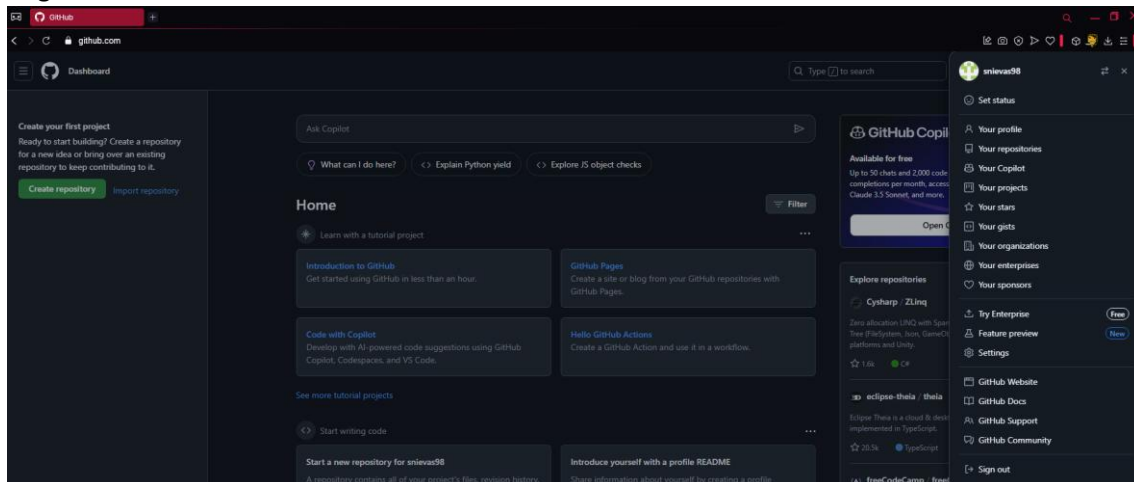
Cuando creamos un repositorio nuevo, elegimos la opción “Public” y luego le damos a “Create repository”.

### **¿Cómo compartir un repositorio público en GitHub?**

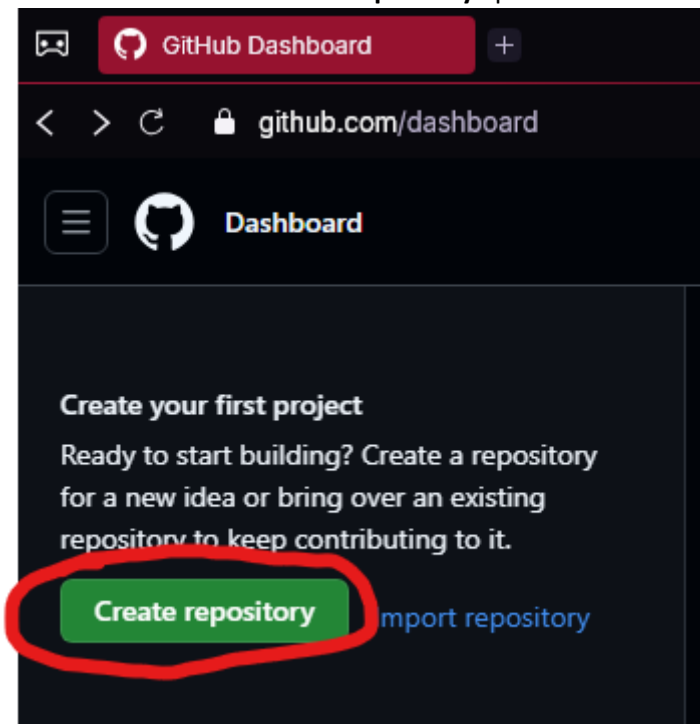
Solo tenés que copiar la URL del repositorio (por ejemplo:  
<https://github.com/miusuario/miproyecto>) y compartirla con quien quieras.

## 2) Crear un repositorio

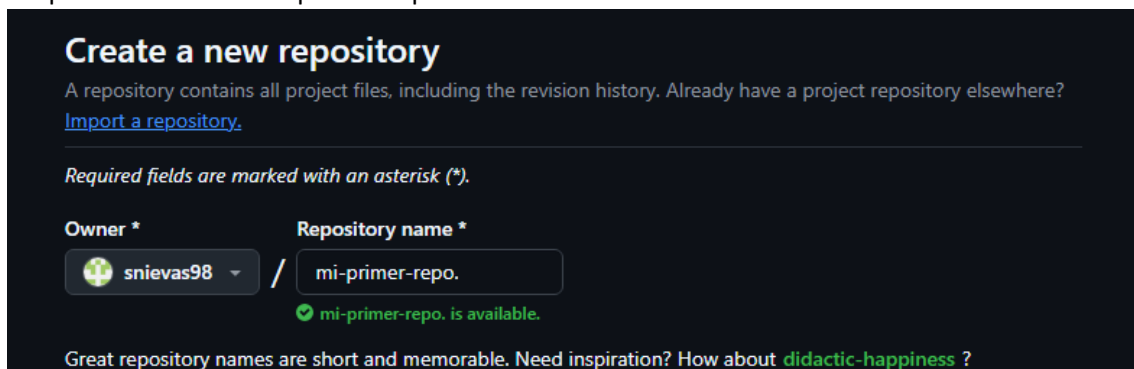
Ingresé a GitHub con mi cuenta.



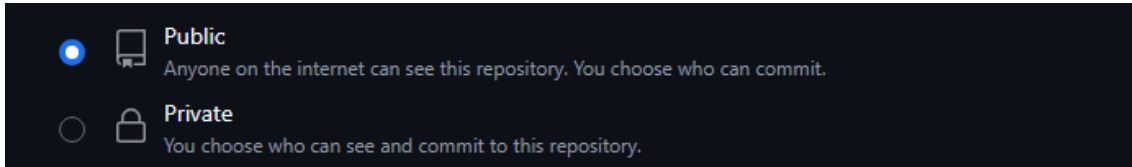
Hice clic en el botón “Create repository” para crear un nuevo repositorio.



Le puse de nombre: mi-primer-repo.

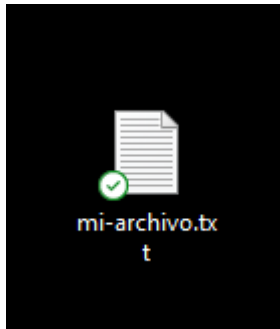


Lo dejé como **público**, ya que no necesito que sea privado por ahora.



### Agregando un archivo

En mi computadora, creé un archivo llamado `mi-archivo.txt` con algo de contenido simple "Este es mi primer archivo en GitHub".



Realice los comandos `git add .` y `git commit -m "Agregando mi-archivo.txt"` y subí los cambios al repositorio de GitHub con `git push -u origin main`.

```
Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/mi-primer-repo
$ git init
Initialized empty Git repository in C:/Users/Santi/OneDrive/Desktop/mi-primer-repo/.git/

Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/mi-primer-repo (master)
$ git add .

Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/mi-primer-repo (master)
$ git commit -m "Agregando mi-archivo.txt"
[master (root-commit) 0ce8d25] Agregando mi-archivo.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 mi-archivo.txt

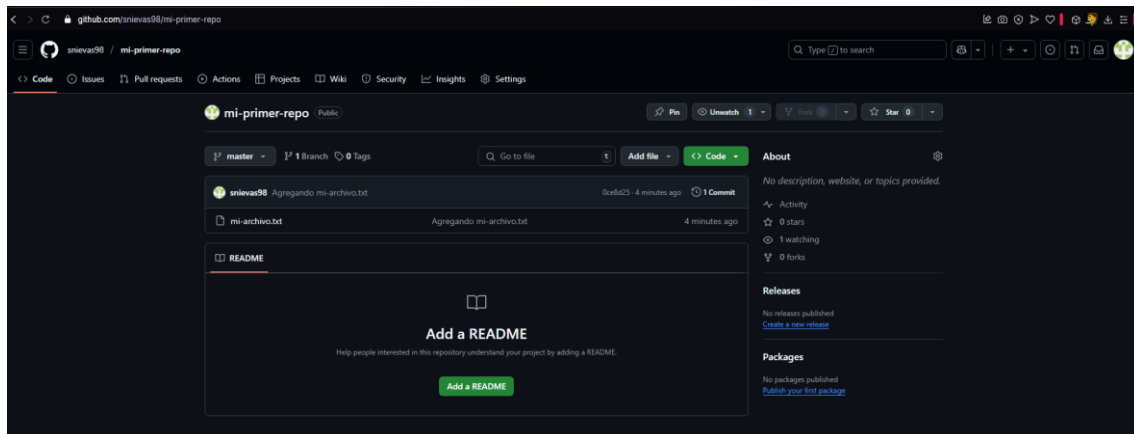
Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/mi-primer-repo (master)
$ git remote add origin https://github.com/sniefas98/mi-primer-repo.git

Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/mi-primer-repo (master)
$ git push -u origin main
error: src refspec main does not match any
error: failed to push some refs to 'https://github.com/sniefas98/mi-primer-repo.git'

Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/mi-primer-repo (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 226 bytes | 226.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/sniefas98/mi-primer-repo.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/mi-primer-repo (master)
$
```

Ya subidos los cambios se ven así



## Creando Branchs

Luego creé la Branch “nueva-rama” con **git checkout -b nueva-rama**

```
Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/mi-primer-repo (master)
$ git checkout -b nueva-rama
Switched to a new branch 'nueva-rama'

Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/mi-primer-repo (nueva-rama)
$
```

Realice un cambio agregando un archivo usando el comando echo que se usa para mostrar texto o escribir contenido en un archivo **echo "Archivo en la nueva rama" > archivo-branch.txt**, luego utilice **git add .** y al final **git commit -m "Agregado archivo en nueva rama"**

```
Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/mi-primer-repo (nueva-rama)
$ echo "Archivo en la nueva rama" > archivo-branch.txt

Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/mi-primer-repo (nueva-rama)
$ git add .
warning: in the working copy of 'archivo-branch.txt', LF will be replaced by CRLF
the next time Git touches it

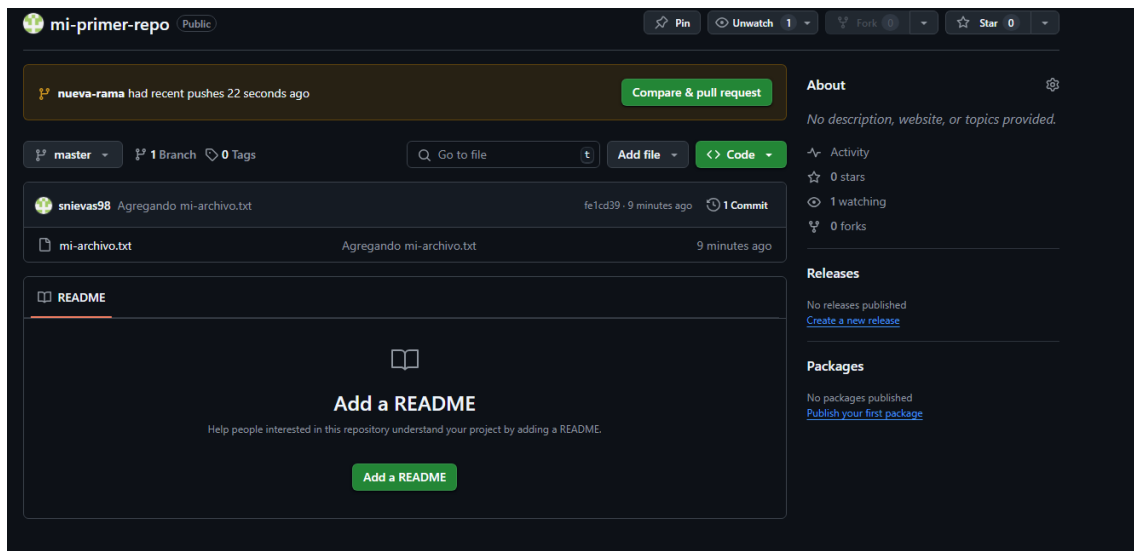
Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/mi-primer-repo (nueva-rama)
$ git commit -m "Agregado archivo en nueva rama"
[nueva-rama a53e23a] Agregado archivo en nueva rama
1 file changed, 1 insertion(+)
create mode 100644 archivo-branch.txt
```

Nombre	Estado	Fecha de modificación	Tipo	Tamaño
.git	✓	02/04/2025 2:13	Carpeta de archivos	
archivo-branch.txt	✓	02/04/2025 2:14	Documento de tex...	1 KB
mi-archivo.txt	✓	01/04/2025 18:56	Documento de tex...	0 KB

Luego para subir la rama a GitHub utilice **git push origin nueva-rama**

```
Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/mi-primer-repo (nueva-rama)
$ git push origin nueva-rama
Everything up-to-date

Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/mi-primer-repo (nueva-rama)
$
```



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

## 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 \* snievas98 / Repository name \* conflict-exercise  
conflict-exercise is available.

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

Description (optional)  
Test

☒ 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 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](#).

ⓘ You are creating a public repository in your personal account.

Create repository



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

Copie la URL de mi repositorio <https://github.com/sniewas98/conflict-exercise.git> y en la terminal ejecute

**git clone** <https://github.com/sniewas98/conflict-exercise.git>

**cd** conflict-exercise

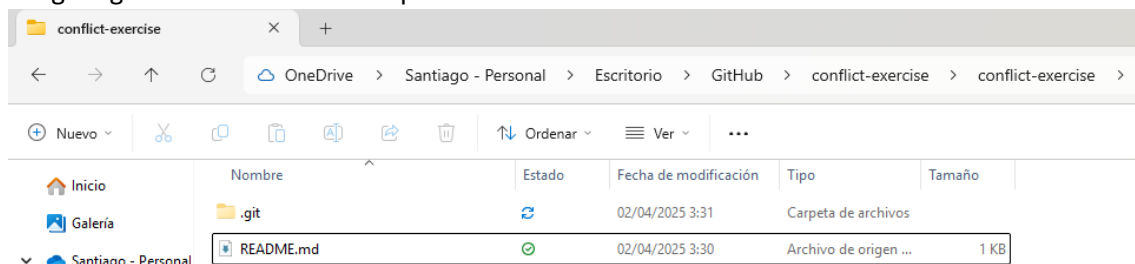
```
Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/GitHub/conflict-exercise
$ git clone https://github.com/sniewas98/conflict-exercise.git
Cloning into 'conflict-exercise'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/GitHub/conflict-exercise
$ cd conflict-exercise

Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/GitHub/conflict-exercise/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/GitHub/conflict-exercise/conflict-exercise (feature-branch)
$ |
```

Luego ingrese al directorio del repositorio



## Paso 3: Crear una nueva rama y editar un archivo

```
Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/GitHub/conflict-exercise/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/GitHub/conflict-exercise/conflict-exercise (feature-branch)
$ git add README.md

Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/GitHub/conflict-exercise/conflict-exercise (feature-branch)
$ git commit -m "Added a line in feature-branch"
[feature-branch 54f7302] Added a line in feature-branch
1 file changed, 1 insertion(+)
```

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

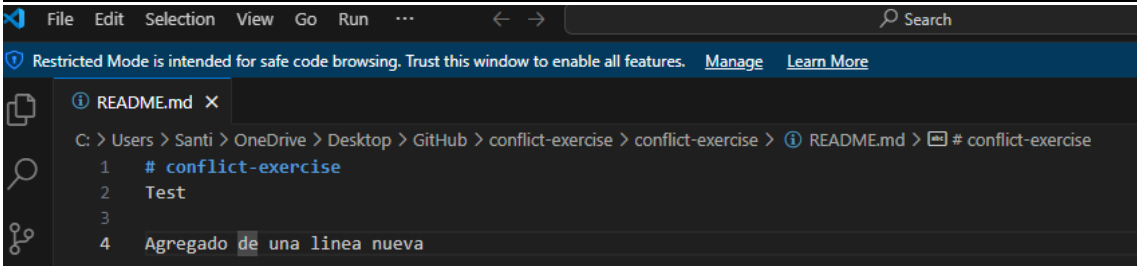
Aquí ejecute en la terminal los comandos resaltados en un recuadro rojo para volver a la rama principal y editar el mismo archivo README.md para luego realizar un commit.

```
Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/GitHub/conflict-exercise/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/GitHub/conflict-exercise/conflict-exercise (main)
$ git add README.md

Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/GitHub/conflict-exercise/conflict-exercise (main)
$ git commit -m "Added a line in main branch"
[main 3224dc1] Added a line in main branch
1 file changed, 2 insertions(+)

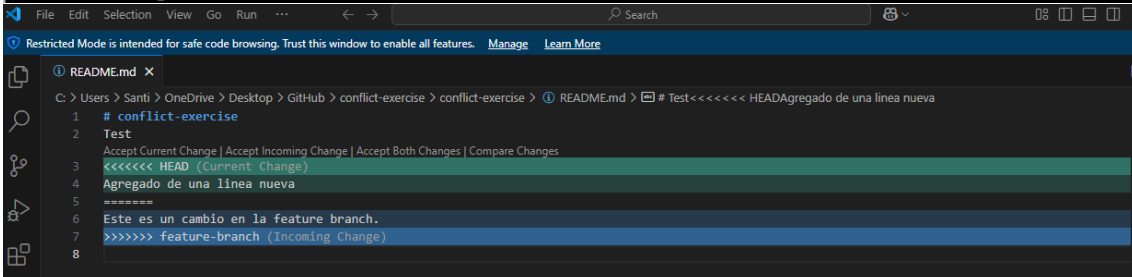
Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/GitHub/conflict-exercise/conflict-exercise (main)
$
```



#### Paso 5: Hacer un merge y generar un conflicto

Ejecuto el comando `git merge feature-branch` y se visualiza el conflicto

```
Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/GitHub/conflict-exercise/conflict-exercise (main)
$ git merge feature-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```



#### Paso 6: Resolver el conflicto

Ejecuto el comando `git add README.md` y `git commit -m "Resolved merge conflict"` para dejar asentado los cambios

```
Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/GitHub/conflict-exercise/conflict-exercise (main|MERGING)
$ git add README.md

Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/GitHub/conflict-exercise/conflict-exercise (main|MERGING)
$ git commit -m "Resolved merge conflict"
[main ce857d6] Resolved merge conflict
```

#### Paso 7: Subir los cambios a GitHub

```
Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/GitHub/conflict-exercise/conflict-exercise (main)
$ git push origin main
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 20 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (12/12), 1.07 KiB | 1.07 MiB/s, done.
Total 12 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/sniefas98/conflict-exercise.git
  543bde3..ce857d6  main -> main

Santi@DESKTOP-DMJOSMS MINGW64 ~/OneDrive/Desktop/GitHub/conflict-exercise/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/sniefas98/conflict-exercise/pull/new/feature-branch
remote:
```

Paso 8: Verificar en GitHub

