

Trabajo practico 2: Git y Github

Nombre: Walter

Apellido: Verdun

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

1- ¿Qué es GitHub?

Es un programa donde se puede compartir, guardar y también colaborar proyectos de programación.

2- ¿Cómo crear un repositorio en GitHub?

Para crear un proyecto debemos tener iniciado una cuenta en Github, luego debemos crear un nuevo repositorio, colocar un nombre al mismo y elegir si debe ser publico o privado, por ultimo debemos seleccionar crear repositorio

3- ¿Cómo crear una rama en Git?

Para crear una rama debemos utilizar el comando (Branch)
ejemplo: `git branch nombre-que-elijas`.

4- ¿Cómo cambiar a una rama en Git?

Para cambiarse a una rama especifica utilizamos el comando (Checkout)
ejemplo: `git checkout nombre-de-la-rama-que-quieras-posicionarte`

5- ¿Cómo fusionar ramas en Git?

Para esto debes estar posicionado en la rama principal (master/main)
primero debemos realizar un (Pull)

ejemplo: `git pull origin main`

Esto hace que se actualice los últimos cambios, de la versión local de un repositorio desde otro remoto.

Luego de hacer esto debemos hacer el fusionar de las ramas.

Ejemplo: `git merge nombre-de-la-rama`

Pero si hay conflictos, tendremos que resolverlo manualmente en los archivos afectados.

6- ¿Cómo crear un commit en Git?

Para crear uno debemos primero tener inicializado nuestro repositorio, luego de esto agregamos los archivos al area de preparacion, para esto debemos hacer lo siguiente:

ejemplo: `git add .`

Ahora debemos realizar nuestro (Commit)

ejemplo: `git commit -m "mensaje o descripcion del cambio"`.

7-¿Cómo enviar un commit a GitHub?

Para esto debemos estar en la rama principal (main/master), debemos hacer lo siguiente.

Ejemplo: `git push origin main`

Si estamos en otra rama debemos colocar lo siguiente.

Ejemplo: `git push origin nombre-rama`

8- ¿Qué es un repositorio remoto?

Es una version de proyecto o codigo que esta alojado en un servidor en linea.

-GitHub

-GitLab

9- ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto debemos tener inicializado nuestro repositorio.

Ejemplo: `git init`

Luego debemos agregar nuestro repositorio remoto, debemos de utilizar el siguiente comando.

Ejemplo: `git remote add nombre-remoto Url-del-repositorio`

10- ¿Cómo empujar cambios a un repositorio remoto?

Para esto debemos ejecutar los siguientes comandos.

Ejemplo: `git add .` (guarda los cambios)

`git commit -m "primer commit"` (deja un mensaje del cambio)

`git push -u origin main` (esto hace que suba los cambios)

11- ¿Cómo tirar de cambios de un repositorio remoto?

Tenemos que ejecutar el siguiente comando.

Ejemplo: `git pull origin <branch>` (branch deberia ser la rama principal por defecto)

12- ¿Qué es un fork de repositorio?

Un fork es una copia de un repositorio que ya esta creado, generalmente en GitHub, GitLab. Se utiliza para poder contribuir en proyectos de codigo abiertos sin afectar el repositorio original.

13- ¿Cómo crear un fork de un repositorio?

Tenemos dos maneras de hacer un fork.

-La primera es entra a la pagina que quieras hacer un fork y pulsa el boton (fork), una vez completado, nos aparecera en nuestra cuenta el repositorio "forkeado"

-La segunda es clonando, podemos hacer una copia del repositorio en nuestra pc para editarlo. Devemos abrir la terminal e introducir los siguientes comandos:

`$ git clone <url_del_repo>`

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

Primero, necesitamos hacer un fork del repositorio al que queremos contribuir. Luego, clonar en repositorio en la maquina local y realizar los cambios. Despues de confirmar los cambios, enviar a tu repositorio de GitHub bifucado.

A continuación, navega al repositorio original donde quieres contribuir con tus cambios. Haz clic en el botón (Nueva solicitud de extraccion). Selecciona la rama que contiene tus cambios y crea una nueva solicitud de extraccion.

Para crear la solicitud de extraccion (PR):

- Ir a la pagina del repositorio en GitHub
- Tendremos una notificacion para crear una nueva rama solicitud de extraccion (Pull Request)
- Hacemos clic en “Compare & pull request.
- Tendremos que asegurarnos que la base del PR es la rama principal
- Escribe un titulo y una descripcion detallada de PR
- Haz clic en “Create pull request”.

15- ¿Cómo aceptar una solicitud de extracción?

Primeros debemos derigirnos a la Pull Request, entrar en el repositorio en GiHub, hacer clic en la pestaña “Pull request” y luego seleccionar la solicitud de extraccion que quieres revisar. Revisar los cambios, mirar que los archivps fueron modificados y que cambios se hicieron, si todo esta bien. Aceptar la Pull Request, debemos hacer clic en el boton verde “Merge pull request”, luego confirmar con “Confirm merge”. Con eso obtenemos la solicitud aceptada y los cambios se agregaron al repositorio.

16- ¿Qué es un etiqueta en Git?

Es una referencia permanente a un punto especifico en el historial del repositorio.

Se utiliza para marcar puntos importantes, como lanzamientos o versiones especifica.

17- ¿Cómo crear una etiqueta en Git?

Para crear una etiqueta devemos usar el siguiente comando.

Ejemplo: `git tag nombre-etiqueta`

18- Cómo enviar una etiqueta a GitHub?

Para enviar una etiqueta debemos crear una local, luego subirla al repositorio remoto.

Ejemplo: `git push origin nombre-etiqueta`

19- ¿Qué es un historial de Git?

Es el registro de todos los cambios que realizamos en un repositorio, organizados en commits. Cada commit contiene informacion sobre que se modifiko.

20- ¿Cómo ver el historial de Git?

Para ver el historial de git debemos hacer el siguiente comando.

Ejemplo: `git log`

Esto nos muestra un historial básico, muestra todos los commits en orden cronológico inversos.

21- ¿Cómo buscar en el historial de Git? ¿Cómo borrar el historial de Git?

Para buscar en el historial debemos buscar por palabra clave en los mensajes de commit.

Ejemplo: `git log --grep="palabra-clave"`

El historial de git se borra con el siguiente comando.

Ejemplo: `rm -rf .git`

22- ¿Qué es un repositorio privado en GitHub?

Es un repositorio que solo puede ser visto por la persona que lo creo o personas con permisos especificos. Es totalmente restringido y solo los autorizados pueden ver y trabajar en el.

23- ¿Cómo crear un repositorio privado en GitHub?

Para crear un repositorio privado tenemos que dirigirnos a GitHub y hacer clic en “New repositorio”, escribir un nombre para el repositorio, elegir la opcion privada y por ultimo hacer clic en “Create repository”.

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

Para invitar a personas a nuestro repositorio privado, debemos dirigirnos a nuestro repositorio luego ir a “Configuracion/Settings”, seleccionar “Colaboradores/Collaborators”, luego hacer clic en “Add people/Agregar personas”, por ultimo asignar permisos y enviar la invitacion

25- ¿Qué es un repositorio público en GitHub?

Es un repositorio que cualquier persona puede ver y clonar, sin necesidad de permisos especiales.

26- ¿Cómo crear un repositorio público en GitHub?

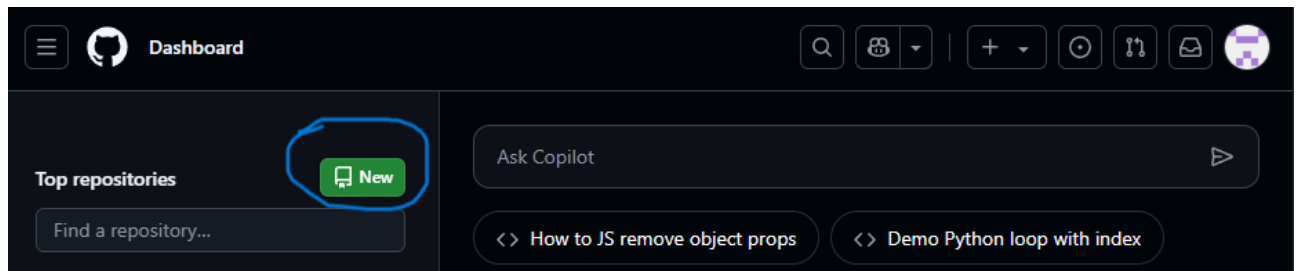
Para crear un repositorio publico tenemos que dirigirnos a GitHub y hacer clic en “New repositorio”, escribir un nombre para el repositorio, elegir la (opcion publica) y por ultimo hacer clic en “Create repository”.

27- ¿Cómo compartir un repositorio público en GitHub?

Para compartir un repositorio, debemos copiar y enviar el enlace del repositorio. Abrimos nuestro repositorio en GitHub, copiamos la URL en la barra de direcciones y luego de esto compartimos nuestro repositorio.

2) Realizar la siguiente actividad

Crear un repositorio



- Dale un nombre al repositorio.
- Elije el repositorio que sea publico.
- Inicializar el repositorio con un archivo.

A screenshot of the 'Create a new repository' form on GitHub. The title is 'Create a new repository'. Below it, a subtitle says 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. A note states 'Required fields are marked with an asterisk (*)'. The form has two main sections: 'Owner' and 'Repository name'. The 'Owner' dropdown is set to 'walter404'. The 'Repository name' input field contains 'repositorio-actividad2-3', and a green checkmark indicates 'repositorio-actividad2-3 is available'. Below this, a tip says 'Great repository names are short and memorable. Need inspiration? How about [effective-dollop](#) ?'. The 'Description (optional)' field is empty. The 'Public' radio button is selected, with the text 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' radio button is unselected, with the text 'You choose who can see and commit to this repository.' The 'Initialize this repository with:' section has 'Add a README file' checked, with a note 'This is where you can write a long description for your project. [Learn more about READMEs.](#)'. The 'Add .gitignore' section has a dropdown set to 'None', with a note 'Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)'. The 'Choose a license' section has a dropdown set to 'None', with a note 'A license tells others what they can and can't do with your code. [Learn more about licenses.](#)'. At the bottom, a note says 'This will set `main` as the default branch. Change the default name in your [settings](#).' A footer note says 'You are creating a public repository in your personal account.' A green 'Create repository' button is at the bottom right.

Agregar un Archivo

- Crea un archivo simple, por ejemplo, “mi-archivo.txt”.
- Realizar los comandos git add . Y git commit -m “Agregar 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).

```

Walter@DESKTOP-EBKGK0Q MINGW64 ~/Desktop
$ git clone https://github.com/walter404/repositorio-actividad2-3.git
Cloning into 'repositorio-actividad2-3'...
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.

Walter@DESKTOP-EBKGK0Q MINGW64 ~/Desktop
$

```

◦ Nos ubicamos en la carpeta que clonamos y creamos archivo.txt

```

Walter@DESKTOP-EBKGK0Q MINGW64 ~/Desktop
$ cd repositorio-actividad2-3/

Walter@DESKTOP-EBKGK0Q MINGW64 ~/Desktop/repositorio-actividad2-3 (main)
$ echo "mi-archivo.txt" > mi-archivo.txt

Walter@DESKTOP-EBKGK0Q MINGW64 ~/Desktop/repositorio-actividad2-3 (main)
$ |

```

◦ Guardamos los cambios, verificamos y commitiamos

```

Walter@DESKTOP-EBKGK0Q MINGW64 ~/Desktop/repositorio-actividad2-3 (main)
$ git add .
warning: in the working copy of 'mi-archivo.txt', LF will be replaced by CRLF the
next time Git touches it

Walter@DESKTOP-EBKGK0Q MINGW64 ~/Desktop/repositorio-actividad2-3 (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

Walter@DESKTOP-EBKGK0Q MINGW64 ~/Desktop/repositorio-actividad2-3 (main)
$ git commit -m "se agrego archivo.txt"
[main ee7bce2] se agrego archivo.txt
1 file changed, 1 insertion(+)
create mode 100644 mi-archivo.txt

Walter@DESKTOP-EBKGK0Q MINGW64 ~/Desktop/repositorio-actividad2-3 (main)
$

```

◦ Mostramos nuestra carpeta local, con los archivos agregamos

 .git	27/03/2025 13:17	Carpeta de archivos	
 mi-archivo.txt	27/03/2025 13:13	Documento de tex...	1 KB
 README.md	27/03/2025 13:05	Archivo de origen ...	1 KB

° Nuestro repositorio en GitHub



The screenshot shows the GitHub interface for a repository named 'repositorio-actividad2-3' owned by 'walter404'. The repository is public and has 1 branch (main) and 0 tags. The commit history shows three commits: the latest commit by 'walter404' 2 minutes ago added a comment to the README, followed by a commit 7 minutes ago that added 'mi-archivo.txt'. The README file is currently selected, showing the title 'repositorio-actividad2-3' and the content 'En este repositori se utilizo para actividad 2 y 3'.

° Hacemos que nuestros cambios se suba a GitHub

```
PS C:\Users\Walter\Desktop\repositorio-actividad2-3> git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 620 bytes | 310.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/walter404/repositorio-actividad2-3.git
5066639..7f7152f  main -> main
PS C:\Users\Walter\Desktop\repositorio-actividad2-3>
```

◦ Repositorio de GitHub



Creando Branch

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

Se crea
rama

```
PS C:\Users\Walter\Desktop\repositorio-actividad2-3> git checkout -b Ramanueva
Switched to a new branch 'Ramanueva'
PS C:\Users\Walter\Desktop\repositorio-actividad2-3> git branch
* Ramanueva
  main
PS C:\Users\Walter\Desktop\repositorio-actividad2-3> 
```

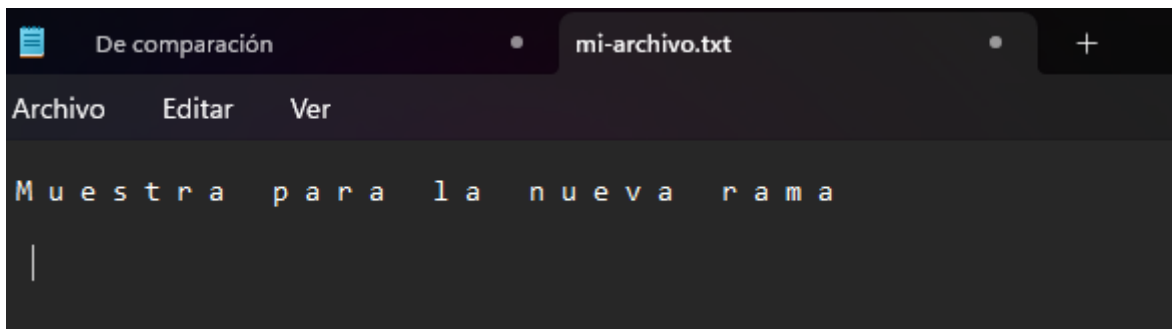
◦
nueva

```
PS C:\Users\Walter\Desktop\repositorio-actividad2-3> echo "Muestra para la nueva rama" > mi-archivo.txt
PS C:\Users\Walter\Desktop\repositorio-actividad2-3> git add .
PS C:\Users\Walter\Desktop\repositorio-actividad2-3> git status
On branch Ramanueva
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   mi-archivo.txt
PS C:\Users\Walter\Desktop\repositorio-actividad2-3> 
```

- Se modifica mi-archivo


```
PS C:\Users\Walter\Desktop\repositorio-actividad2-3> git commit -m "Se agrega modificacion a la ramaNueva"
[Ramanueva 47332e6] Se agrega modificacion a la ramaNueva
 1 file changed, 0 insertions(+), 0 deletions(-)
PS C:\Users\Walter\Desktop\repositorio-actividad2-3> git status
On branch Ramanueva
nothing to commit, working tree clean
```

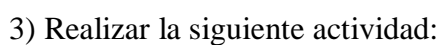
Ahora nos vamos a mi-archivo y vemos que estará modificado.



Subimos la nueva rama a github:

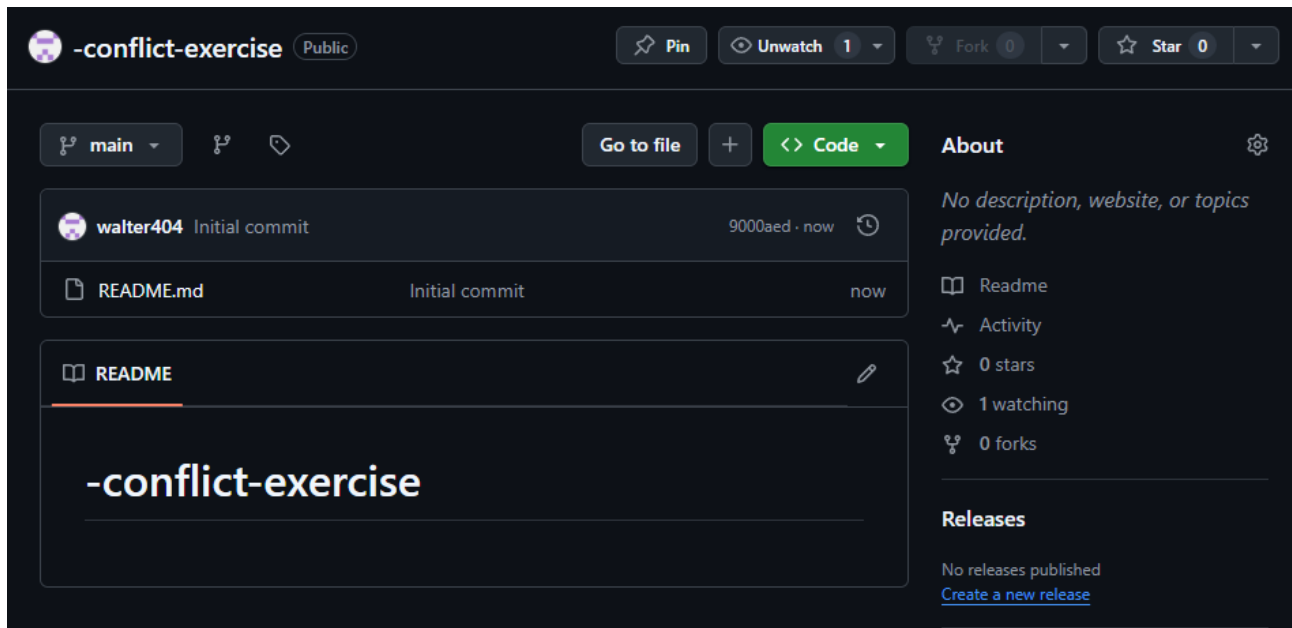
```
PS C:\Users\Walter\Desktop\repositorio-actividad2-3> git push origin Ramanueva
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 340 bytes | 340.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'Ramanueva' on GitHub by visiting:
remote:   https://github.com/walter404/repositorio-actividad2-3/pull/new/Ramanueva
remote:
To https://github.com/walter404/repositorio-actividad2-3.git
 * [new branch]      Ramanueva -> Ramanueva
```

° Mostramos la ramas main y Ramanueva



- Ve a GitHub e inicia sesión en una cuenta
- Debemos hacer clic en el botón “New” o “Create repository”
- Asigna un nombre al repositorio
- Marca la opción “Initialize this repository with a README”
- Haz clic en “Create repository”

Paso
2:



Clonar el repositorio a tu maquina local

- Copia la URL del repositorio
- Abre la terminal
- Clona el repositorio usando el comando:
git clone

```
Walter@DESKTOP-EBKGKQ MINGW64 ~/Desktop
$ git clone https://github.com/walter404/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.
```

◦ Entra en el directorio del repositorio con el siguiente comando:
cd conflict-exercise

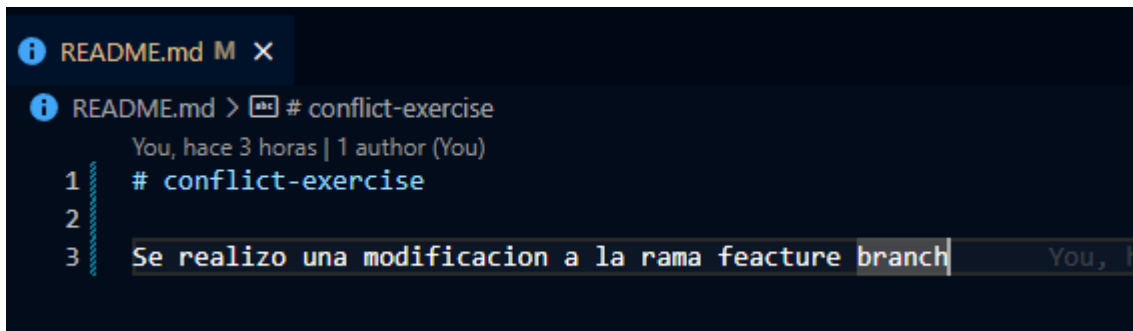
Paso 3: Crear una nueva rama y editar un archivo

◦ Crea una nueva rama llamada feature-branch:
git checkout -b feature-branch

```
Walter@DESKTOP-EBKGKQ MINGW64 ~/Desktop
$ cd conflict-exercise

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

- ° Abre el archivo README.md en un editor de texto y añadir una línea nueva.



```
README.md M X
README.md > # conflict-exercise
You, hace 3 horas | 1 author (You)
1 # conflict-exercise
2
3 Se realizo una modificacion a la rama feacture branch
```

Guardamos los cambios y hacemos nuestro respectivo commit:

git add README.md

El siguiente comando:

git commit -m “mensaje”

```
PS C:\Users\Walter\Desktop\conflict-exercise> git add README.md
PS C:\Users\Walter\Desktop\conflict-exercise> git commit -m "Se ha añadido una línea en feature-branch"
[feature-branch f18a1c4] Se ha añadido una línea en feature-branch
1 file changed, 3 insertions(+), 1 deletion(-)
```

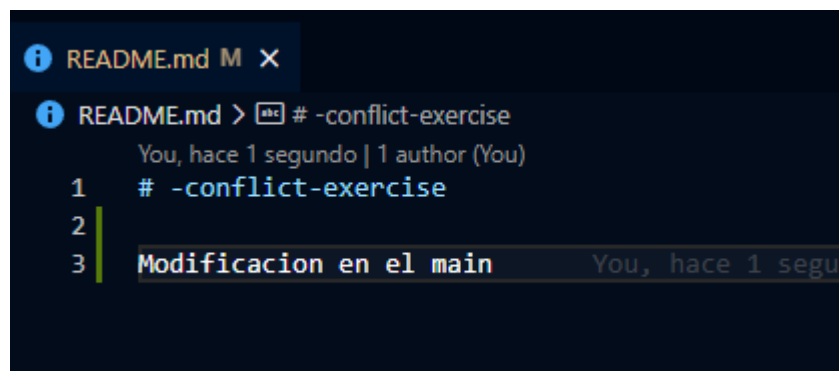
Paso 4: Tenemos que volver a la rama principal y editar el mismo archivo

- ° Posicionarnos a la rama principal (main/master):

git checkout main

```
PS C:\Users\Walter\Desktop\conflict-exercise> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

- ° Debemos editar el archivo README.md de nuevo, añadiendo una línea diferente: este cambio pertenece al main branch.



```
README.md M X
README.md > # -conflict-exercise
You, hace 1 segundo | 1 author (You)
1 # -conflict-exercise
2
3 Modificacion en el main
```

- ° Guardamos nuestros cambios y hacemos un commit:

git add README.md

git commit -m "mensaje del cambio"

```
PS C:\Users\Walter\Desktop\conflict-exercise> git add README.md
PS C:\Users\Walter\Desktop\conflict-exercise> git commit -m "Se agregó una línea en la rama principal"
[main 12d7e4b] Se agregó una línea en la rama principal
1 file changed, 3 insertions(+), 1 deletion(-)
```

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

```
PS C:\Users\Walter\Desktop\conflict-exercise> 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.
```

Se

genera un conflicto porque ambos cambios afectan la misma línea de archivo
README.md

Paso 6: Resolver el conflicto

- ° Debemos abrir el archivo README.md en nuestro editor de texto. Veras algo similar a esto:

<<<<<<< HEAD

Este es un cambio en la main branch.

=====

Este es un cambio en la feature branch.

>>>>>>> feature-branch

```
README.md ! X
README.md > # conflict-exercise
You, hace 1 segundo | 1 author (You) | Aceptar cambio actual | Aceptar cambio entrante | Acepta
1 <<<<<< HEAD (Cambio actual)
2 # -conflict-exercise
3
4 Modificacion en el main
5 =====
6 # conflict-exercise
7
8 Se realizo una modificacion a la rama feacture branch
9 >>>>>> feature-branch (Cambio entrante)
10
```

Decide cómo resolver conflicto. Puedes mantener ambos cambios, elegir de ellos,

el

uno

o fusionar los contenidos de alguna manera.

° Edita el archivo para resolver el conflicto y guarda los cambios.

```
README.md ! ●
README.md > # conflict-exercise
You, hace 10 minutos | 1 author (You)
1 # conflict-exercise
2 Modificacion en el main
3 Se realizo una modificacion a la rama feacture branch
4
5
```

Añade el archivo resuelto y completa el merge:

```
git add README.md
```

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

```
PS C:\Users\Walter\Desktop\conflict-exercise> git add README.md
PS C:\Users\Walter\Desktop\conflict-exercise> git commit -m "Resolviendo conflicto con un merge"
[main bfefa0b] Resolviendo conflicto con un merge
```

Paso 7: Subir los cambios a GitHub

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

```
git push origin main
```

```
PS C:\Users\Walter\Desktop\conflict-exercise> git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (9/9), 976 bytes | 488.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/walter404/conflict-exercise.git
 9000aed..bfefa0b  main -> main
```

° También sube la feature-branch si deseas

git push origin feature-branch

Paso

```
PS C:\Users\Walter\Desktop\conflict-exercise> 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/walter404/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/walter404/conflict-exercise.git
 * [new branch]      feature-branch -> feature-branch
```

8:

Verificar en GitHub

° Ve a tu repositorio en GitHub y revisar 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

Public



Pin



Unwatch

1



main



Go to file



<> Code



walter404 Resolución de conflicto

5687d23 · 1 minute ago



README.md

Resolución de conflicto

1 minute ago



README



conflict-exercise

Modificación en el main Se realizo una modificacion a la rama feacture branch