

## Trabajo Práctico 2: Git y GitHub

Álvarez, Juan Ignacio. Comisión 24.

### Actividades

#### a) **Contestar las siguientes preguntas utilizando las guías y documentación proporcionada.**

##### 1) ¿Qué es GitHub?

GitHub es el mayor proveedor de alojamiento de repositorios Git, y es el punto de encuentro para que millones de desarrolladores colaboren en el desarrollo de sus proyectos. Un gran porcentaje de los repositorios Git se almacenan en GitHub, y muchos proyectos de código abierto lo utilizan para hospedar su Git, realizar su seguimiento de fallos, hacer revisiones de código y otras cosas.

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

Los repositorios de GitHub almacenan una gran variedad de proyectos. Para crear un repositorio debe:

- ✓ En la esquina superior derecha de cualquier página, selecciona **+** y luego haz clic en **Nuevo repositorio**.
- ✓ Escriba un nombre corto y fácil de recordar para el repositorio. Por ejemplo: *"hola-mundo"*.
- ✓ Opcionalmente, puede agregar una **descripción** del repositorio. Por ejemplo, *"Mi primer repositorio en GitHub"*.
- ✓ Elige la **visibilidad** del repositorio: puede ser **Público** o **Privado**.
- ✓ Seleccione **Initialize this repository with a README** (Inicializar este repositorio con un archivo Léame).
- ✓ Haga clic en **Create repository** (Crear repositorio).

##### 3) ¿Cómo crear una rama en Git?

Para crear una rama en Git, puedes usar el comando **git branch**. Por ejemplo, para crear una rama llamada *"new\_branch"* a partir de la rama *main*, puedes usar **git branch new\_branch**.

Para crear una rama en GitHub:

- ❖ Ir a la página principal del repositorio.
- ❖ En la vista de **árbol de archivos**, seleccionar el **menú desplegable** de ramas.
- ❖ Hacer clic en **Ver todas las ramas**.
- ❖ Hacer clic en **Nueva rama**.
- ❖ Escribir un nombre para la rama en el campo **"Nombre de rama"**

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

Para cambiar a la rama *new\_branch*, puedes usar **git checkout new\_branch**. También puedes usar **git checkout -b new\_branch** para crear y cambiar a la rama *new\_branch* al mismo tiempo.

### 5) ¿Cómo fusionar ramas en Git?

La herramienta **git merge** se utiliza para fusionar uno o más ramas dentro de la rama que tienes activa. Por tanto, se fusionará cualquier cambio que se haya hecho en la base de código en una rama separada de tu rama actual como un nuevo *commit*.

Si hay algún cambio al que no se le ha hecho *commit* en la rama actual, Git no te permitirá fusionar hasta que se hayan realizado *commit* todos los cambios en tu rama actual.

### 6) ¿Cómo crear un commit en Git?

El comando **git commit** guardará todos los cambios hechos en la zona de montaje o área de preparación (staging area), junto con una breve descripción del usuario, en un "commit" al repositorio local.

La opción más común utilizada es **git commit -m "mensaje"**. El mensaje debe ser una breve descripción de los cambios a los que se les está realizando commit.

### 7) ¿Cómo enviar un commit a GitHub?

Para hacer un **commit** en GitHub, puedes usar el comando **git commit** en la línea de comandos:

- Usa **git add** para seleccionar los cambios que quieres preparar para la confirmación.
- Usa **git commit** para crear una instantánea de los cambios preparados.
- Incluye el indicador -m y tu mensaje entre comillas.
- Escribe un mensaje de confirmación que describa el cambio realizado.
- Haz clic en "Commit & Push".

### 8) ¿Qué es un repositorio remoto?

Los repositorios remotos son versiones de un proyecto que están hospedadas en Internet o en cualquier otra red. Puedes tener varios de ellos, y en cada uno tendrás generalmente permisos de solo lectura o de lectura y escritura. Colaborar con otras personas implica gestionar estos repositorios remotos enviando y trayendo datos de ellos cada vez que necesites compartir tu trabajo.

### 9) ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto a Git, puedes usar el comando **git remote add** en la terminal. Este comando se debe ejecutar dentro del directorio donde está el repositorio.

Paso a paso:

- ❖ Entrar al directorio donde está el repositorio
- ❖ Usar el comando **git remote add**
- ❖ Especificar un nombre remoto, por ejemplo, *origin*

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

Para hacer push a un repositorio remoto en Git, se puede usar el comando **git push**.

Paso 1: Agregar el servidor remoto a Git

- Ejecutar **git remote add <url>**
- Confirmar que se ha agregado el servidor remoto con **git remote -v**

Paso 2 Subir la rama local al repositorio remoto

- Ejecutar **git push <remote> <branch>**
- Por ejemplo, si la rama actual es main, ejecutar **git push origin main**

Paso 3 Confirmar que la rama ha sido empujada correctamente

- Ir a GitHub y hacer clic en el menú colapsable que muestra la rama actual y las disponibles
- Verificar que la nueva rama está ahí

Consideraciones:

- Si se está colaborando con otros en la rama, *se debe evitar usar --force o usar --force-with-lease* para evitar perder los cambios de otros colaboradores
- El comando *git push sobrescribe los cambios*, por lo que solo debe ejecutarse con una rama vacía como objetivo.
- Se puede usar GitHub Desktop para hacer push de contenido local a GitHub

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

Para obtener los cambios de un repositorio remoto en Git, puedes usar el comando **git fetch**. Esto recupera el trabajo nuevo de otras personas sin combinar los cambios en las ramas propias.

Procedimiento:

- ✓ Usa **git fetch** para recuperar los cambios.
- ✓ Obtienes todas las ramas de seguimiento remoto nuevas y etiquetas sin combinar estos cambios en las ramas propias.

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

Un fork de repositorio es una copia exacta de un repositorio original que se crea para colaborar en un proyecto. La palabra fork se traduce como bifurcación. Se usa para:

- Para participar en proyectos en los que no se tienen permisos de escritura
- Para contribuir a proyectos de código abierto
- Para experimentar con nuevas características
- Para crear una propia versión del proyecto

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

Creación de una rama mediante la lista desplegable de ramas

- ✓ En GitHub, navegue hasta la página principal del repositorio.
- ✓ Selecciona el **menú desplegable** de ramas en la vista de árbol de archivos o en la parte superior del editor de archivos integrado.
- ✓ Opcionalmente, si deseas crear la nueva rama desde una rama distinta de la rama predeterminada del repositorio, haz clic en otra rama y, a continuación, vuelve a seleccionar el menú desplegable de ramas.
- ✓ En la sección "**Buscar o crear una rama...**" campo de texto, escriba un nombre único para la nueva rama y, a continuación, haga clic en **Crear rama**.

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

Para enviar una solicitud de extracción (pull request) a un repositorio de GitHub, puedes seguir estos pasos:

- Ir a la página principal del repositorio.
- Seleccionar la rama que contiene los cambios.
- Hacer clic en "**Comparar y solicitud de incorporación de cambios**".
- Elegir la rama base y la rama de comparación.
- Escribir un título y una descripción.
- Hacer clic en "**Crear solicitud de incorporación de cambios**".

Una vez enviada la solicitud, los interesados pueden revisar los cambios, debatir modificaciones y enviar confirmaciones de seguimiento.

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

Para aceptar una solicitud de **pull request** en GitHub, puedes aprobarla y fusionarla con la rama principal.

Para aprobar una solicitud de **pull request**:

- Ir a la pestaña **Solicitudes de extracción**.
- Seleccionar la solicitud de extracción que se quiere aprobar.
- Revisar los cambios.
- Hacer clic en **Revisar cambios**.
- Escribir un comentario sobre los cambios propuestos.
- Hacer clic en **Aprobar**.
- Hacer clic en **Enviar revisión**.

Para fusionar una solicitud de **pull request**:

- Hacer clic en **Solicitudes de incorporación de cambios**.
- Seleccionar la solicitud de extracción que se quiere fusionar.
- Desplazarse hasta la parte inferior de la solicitud.
- Seleccionar la opción de fusión que se quiera usar.
- Escribir un mensaje de confirmación o aceptar el mensaje predeterminado
- Hacer clic en **Confirm merge**, **Confirm squash and merge** o **Confirm rebase and merge**

Opcionalmente, se puede eliminar la rama para mantener ordenado el listado de ramas en el repositorio.

### **16) ¿Qué es una etiqueta en Git?**

Git tiene la posibilidad de **etiquetar** puntos específicos del historial como importantes. Esta funcionalidad se usa típicamente para marcar versiones de lanzamiento (v1.0, por ejemplo). Una **etiqueta** es como una rama que no cambia. *A diferencia de las ramas, las etiquetas, una vez creadas, no tienen historial de confirmaciones.*

Git utiliza dos tipos principales de etiquetas: ligeras y anotadas. Una **etiqueta ligera** es muy parecido a una rama que no cambia - simplemente es un puntero a un commit específico.

Sin embargo, las **etiquetas anotadas** se guardan en la base de datos de Git como objetos enteros. Tienen un *checksum*; contienen el nombre del etiquetador, correo electrónico y fecha; tienen un mensaje asociado; y pueden ser firmadas y verificadas con GNU Privacy Guard (GPG). Normalmente se recomienda que crees **etiquetas anotadas**.

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

Para crear una etiqueta anotada en Git se ejecuta el comando **git tag -a**. La opción **git tag -a -m** especifica el mensaje de la etiqueta, el cual es guardado junto con ella. Si no especificas el mensaje de una etiqueta anotada, Git abrirá el editor de texto para que lo escribas.

Puedes ver la información de la etiqueta junto con el commit que está etiquetado al usar el comando **git show**.

Para crear una etiqueta ligera se ejecuta el comando **git tag** (no pases las opciones -a, -s ni -m). Una etiqueta ligera no es más que el checksum de un commit guardado en un archivo - no incluye más información. Esta vez, si ejecutas **git show** sobre la etiqueta no verás la información adicional. El comando solo mostrará el commit.

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

Compartir etiquetas es similar a enviar ramas. De manera predeterminada, **git push** no enviará etiquetas. Las etiquetas se tienen que usar explícitamente en git push. Para enviar una etiqueta a GitHub, puedes usar el comando **git push --tags**. Cuando otro usuario clone un repositorio o incorpore cambios en él, recibirá las nuevas etiquetas. También puedes crear etiquetas desde la interfaz web de GitHub.

### 19) ¿Qué es un historial de Git?

El historial de Git es un registro de los cambios realizados en un repositorio de código. Se almacena como un gráfico de instantáneas, llamadas confirmaciones, que apuntan a confirmaciones anteriores.

El historial permite ver quiénes son los autores de los cambios, averiguar dónde se introdujeron los errores, revertir los cambios problemáticos.

La opción de **git log -L** muestra el historial de una función o línea de código (para búsqueda de registros de línea).

### 20) ¿Cómo ver el historial de Git?

Para ver el historial de Git se utiliza el comando **git log**. Características del historial de Git:

- cada confirmación muestra la fecha, el usuario, el SHA de confirmación, y un enlace para navegar al archivo.
- las confirmaciones pueden tener varios padres, lo que crea un historial que parece un gráfico.
- es diferente al de los sistemas de control de versiones centralizados (CVCS)
- se almacena en la máquina local del usuario

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

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

**git grep**

- Busca un patrón en cualquier árbol o directorio de trabajo
- Es rápido y puede buscar en cualquier árbol de Git
- Tiene opciones como -n, --count, -p, --and, --break y --heading

#### **git log**

- Muestra el historial de una función o línea de código.
- Puedes usar la opción **git log -L** para mostrar el historial de una función o línea de código.
- Puedes usar la opción **git log -S** (pickaxe) para buscar líneas perdidas por palabra clave.

#### **git rev-list**

- Puedes usar **git grep <pattern> \$(git rev-list --all)** para buscar un patrón en todo el historial de confirmaciones.

#### **git show**

- Puedes usar **git show** para ver el historial de confirmaciones

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

Un repositorio privado en GitHub es un espacio donde se puede guardar código, archivos y el historial de revisiones de forma segura y sin que pueda verlo nadie más que el usuario que lo creó. Características:

- Los repositorios privados pueden tener varios colaboradores.
- Los repositorios de GitHub están cifrados en reposo y en vuelo.
- El personal de GitHub no accede a la información de los repositorios privados sin el consentimiento del usuario.

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

Para crear un repositorio privado:

- Ir a <https://github.com/new>.
- Configurar la visibilidad del repositorio.
- En la página principal del repositorio, hacer clic en Configuración.
- En la sección "Zona de peligro", hacer clic en Cambiar visibilidad.
- Seleccionar la visibilidad privada.

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

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

- ✓ Ir a la página del repositorio.
- ✓ Hacer clic en **Configuración**.
- ✓ En la barra lateral, hacer clic en **Colaboradores y equipos**.
- ✓ Hacer clic en **Agregar personas**.
- ✓ Escribir el *nombre de usuario o correo electrónico* de la persona a invitar.
- ✓ Elegir el **rol** de repositorio que se le quiere conceder.
- ✓ Hacer clic en **Agregar NOMBRE al REPOSITORIO**

También se puede invitar a alguien a una organización de GitHub. Para ello, se puede:

- ❖ Seleccionar **la foto del perfil** en la esquina superior derecha de GitHub.
- ❖ Hacer clic en **Sus organizaciones**.
- ❖ Hacer clic en el **nombre** de la organización.
- ❖ Hacer clic en **Personas**.
- ❖ Hacer clic en **Invitar a miembros**.
- ❖ Escribir el *nombre de usuario*, el nombre completo o la dirección de correo electrónico de la persona a invitar.

- ❖ Hacer clic en **Invitar**.

## 25) ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un espacio virtual donde se almacena código, archivos y revisiones de forma que sea accesible para cualquier persona en internet. Se utiliza para::

- Compartir software de código abierto.
- Almacenar y administrar cambios en el código fuente de una aplicación.
- Guardar versiones del código a las que se pueda acceder cuando sea necesario

Características:

- ❖ Los repositorios públicos son accesibles para todos en internet.
- ❖ Los repositorios pueden contar con múltiples colaboradores.
- ❖ Contienen todo el código, los archivos y el historial de revisiones de cada uno de ellos.
- ❖ Permiten debatir y administrar el trabajo dentro del repositorio

## 26) ¿Cómo crear un repositorio público en GitHub?

Para crear un repositorio público en GitHub, puedes seguir estos pasos:

- Ir a **GitHub**.
- En la esquina superior derecha, seleccionar **Nuevo repositorio**.
- Escribir un *nombre* para el repositorio.
- Agregar una *descripción opcional*.
- Elegir la **visibilidad pública**.
- Seleccionar **Inicializar este repositorio con un archivo README**.
- Hacer clic en **Crear repositorio**.

## 27) ¿Cómo compartir un repositorio público en GitHub?

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

### Invitar colaboradores

- Ir a la *página principal* del repositorio.
- Hacer clic en **Configuración**.
- En la sección "**Acceso**", hacer clic en **Colaboradores y equipos**.
- Hacer clic en **Agregar personas o Agregar equipos**.
- Escribir el *nombre* de la persona o equipo a invitar.
- Seleccionar *el rol de repositorio* que se desea otorgar.
- Hacer clic en **Agregar NOMBRE AL REPOSITORIO**.

### Compartir la URL del repositorio

- Ir al *repositorio* que se desea compartir.
- Hacer clic en el botón verde **Código**.
- Hacer clic en el ícono de **copia de la URL web**.
- Pegar el *enlace* en el sitio web deseado.

GitHub limita la cantidad de personas a las que se puede invitar a un repositorio en un período de 24 horas.



b) **Realizar la siguiente actividad:**

- Crear un repositorio.
  - Dale un nombre al repositorio.
  - Elije 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 (\*).

Repository owner and name:  Repository name \*:

Your new repository will be created as TP-2-Programaci-n-1-Ej-2. The repository name can only contain ASCII letters, digits, and the characters -, ., and \_.

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

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.](#)

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**

---

juanalvarez87 / TP-2-Programaci-n-1-Ej-2

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#)

**main** [TP-2-Programaci-n-1-Ej-2 /](#)

juanalvarez87 Create mi-archivo.txt

Name	Last commit message
README.md	Initial commit
mi-archivo.txt	Create mi-archivo.txt

---

README.md

## TP-2-Programaci-n-1-Ej-2

Repositorio del Ejercicio 2

- 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).

```
Usuario@Usuario-PC MINGW32 ~/Desktop/TP-2-Programacion-1-Ej-2-B (master)
$ git init
Initialized empty Git repository in C:/Users/Usuario/Desktop/TP-2-Programacion-1-Ej-2-B/.git/

Usuario@Usuario-PC MINGW32 ~/Desktop/TP-2-Programacion-1-Ej-2-B (master)
$ git add .

Usuario@Usuario-PC MINGW32 ~/Desktop/TP-2-Programacion-1-Ej-2-B (master)
$ git commit -m "Agregando mi-archivo.txt"
[master (root-commit) 440d7c5] Agregando mi-archivo.txt
 2 files changed, 5 insertions(+)
 create mode 100644 desktop.ini
 create mode 100644 mi-archivo.txt

Usuario@Usuario-PC MINGW32 ~/Desktop/TP-2-Programacion-1-Ej-2-B (master)
$ git remote add origin https://github.com/juanialvarez87/TP-2-Programacion-1-Ej-2-B.git

Usuario@Usuario-PC MINGW32 ~/Desktop/TP-2-Programacion-1-Ej-2-B (master)
$ git push origin master
```

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

### c) 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".

## 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 (\*).

Repository owner and name:  / Repository name \*:   
☒ conflict-exercise is available.

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

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](#).

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 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
```

```
Usuario@Usuario-PC MINGW32 ~/Desktop (master)
$ git clone https://github.com/juanialvarez87/conflict-exercise.git
Cloning into 'conflict-exercise'...
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.

Usuario@Usuario-PC MINGW32 ~/Desktop (master)
$ |

Usuario@Usuario-PC MINGW32 ~/Desktop (master)
$ 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"
```

```
Usuario@Usuario-PC MINGW32 ~/Desktop (master)
$ cd conflict-exercise

Usuario@Usuario-PC MINGW32 ~/Desktop/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

Usuario@Usuario-PC MINGW32 ~/Desktop/conflict-exercise (feature-branch)
$ git branch
* feature-branch
  main

Usuario@Usuario-PC MINGW32 ~/Desktop/conflict-exercise (feature-branch)
$ git add README.md

Usuario@Usuario-PC MINGW32 ~/Desktop/conflict-exercise (feature-branch)
$ git commit -m "Added a line in feature-branch"
[feature-branch 033fccc] Added a line in feature-branch
 1 file changed, 2 insertions(+)

Usuario@Usuario-PC MINGW32 ~/Desktop/conflict-exercise (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"
```

```
Usuario@Usuario-PC MINGW32 ~/Desktop/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Usuario@Usuario-PC MINGW32 ~/Desktop/conflict-exercise (main)
$ git add README.md

Usuario@Usuario-PC MINGW32 ~/Desktop/conflict-exercise (main)
$ git commit -m "Added a line in main branch"
[main 9d8b554] Added a line in main branch
1 file changed, 1 insertion(+)
```

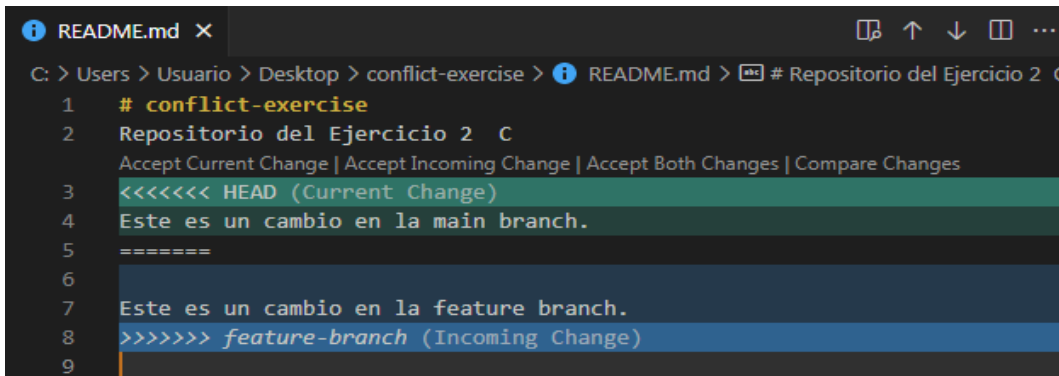
### Paso 5: Hacer un merge y generar un conflicto

- 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.

```
Usuario@Usuario-PC MINGW32 ~/Desktop/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.
```



```
1 # conflict-exercise
2 Repositorio del Ejercicio 2 C
3 Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
4 <<<<<< HEAD (Current Change)
5 Este es un cambio en la main branch.
6 =====
7 Este es un cambio en la feature branch.
8 >>>>>> feature-branch (Incoming Change)
```

### Paso 6: Resolver el conflicto

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

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

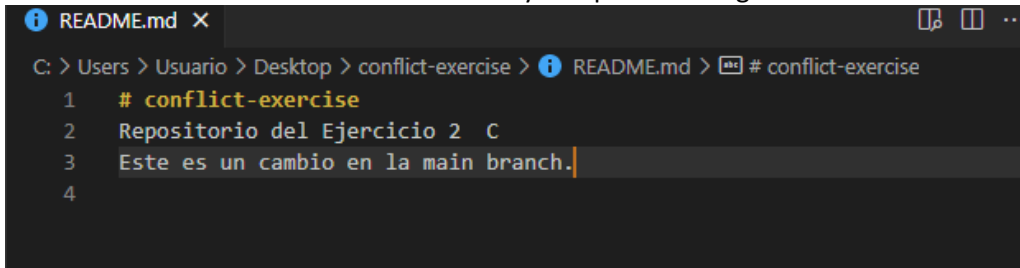
Este es un cambio en la main branch.

```
=====
```

Este es un cambio en la feature branch.

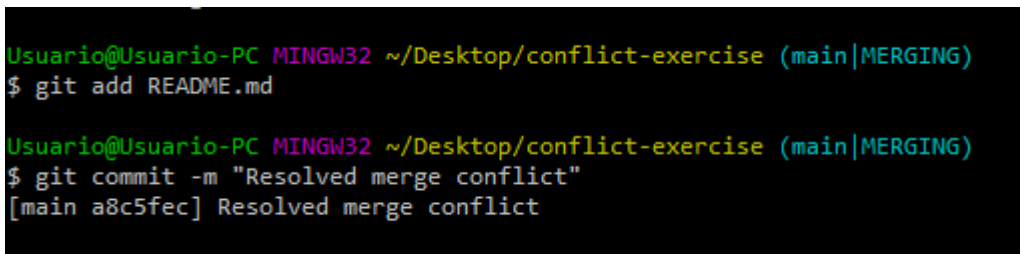
```
>>>>>> feature-branch
```

- 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:



```
README.md X
C: > Users > Usuario > Desktop > conflict-exercise > README.md > # conflict-exercise
1 # conflict-exercise
2 Repositorio del Ejercicio 2 C
3 Este es un cambio en la main branch.
4
```

```
git add README.md
git commit -m "Resolved merge conflict"
```



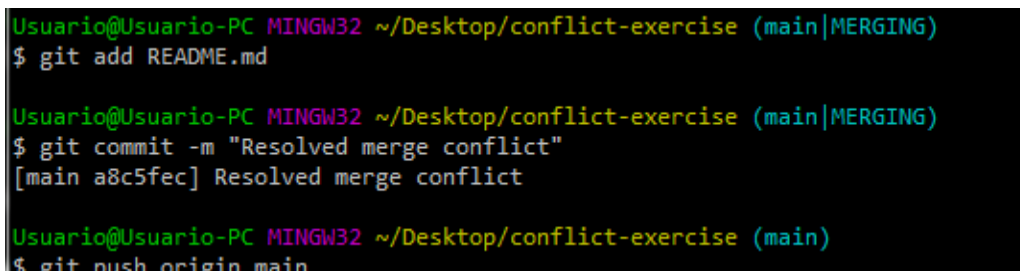
```
Usuario@Usuario-PC MINGW32 ~/Desktop/conflict-exercise (main|MERGING)
$ git add README.md

Usuario@Usuario-PC MINGW32 ~/Desktop/conflict-exercise (main|MERGING)
$ git commit -m "Resolved merge conflict"
[main a8c5fec] 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
```



```
Usuario@Usuario-PC MINGW32 ~/Desktop/conflict-exercise (main|MERGING)
$ git add README.md

Usuario@Usuario-PC MINGW32 ~/Desktop/conflict-exercise (main|MERGING)
$ git commit -m "Resolved merge conflict"
[main a8c5fec] Resolved merge conflict

Usuario@Usuario-PC MINGW32 ~/Desktop/conflict-exercise (main)
$ 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.