

Trabajo Práctico N°2

Git y GitHub

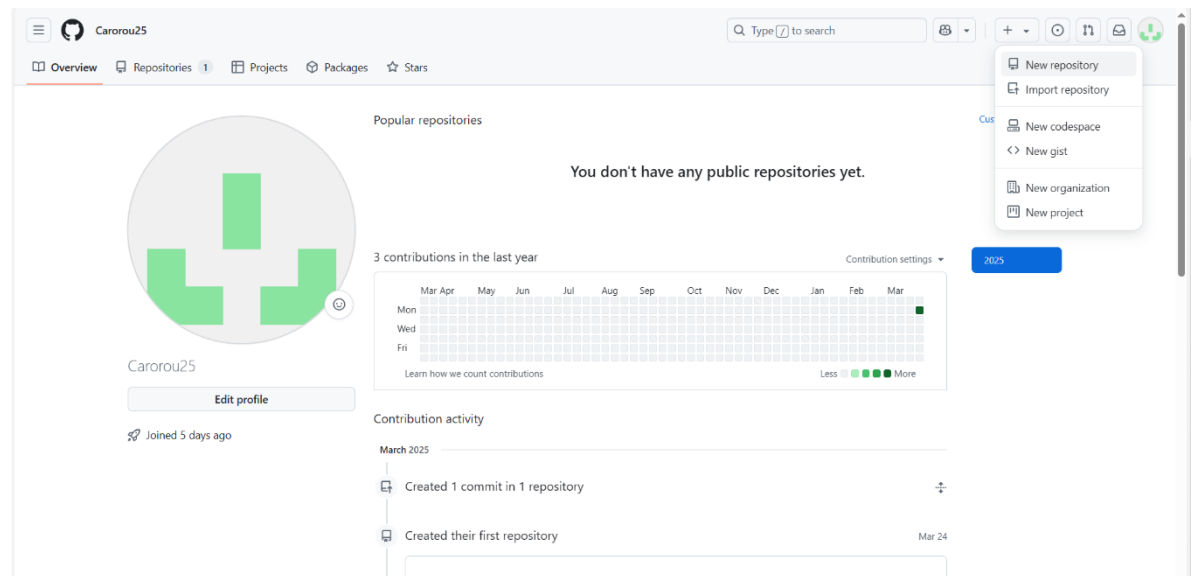
**Alumna: Rousseaux, Alexia
Carolina**

1. ¿Qué es GitHub?

Es una plataforma de hosting en donde alojamos nuestros repositorios de códigos, y donde podemos compartirlos con una comunidad. Permite almacenar, organizar y compartir nuestros proyectos de software. Es importante manejar Git y GitHub ya que es una herramienta que favorece el trabajo colaborativo entre un equipo de programadores.

¿Cómo crear un repositorio en GitHub?

Primero debemos crear una cuenta en github.com. Luego, hacemos clic en el botón (+) en la esquina superior derecha, y seleccionamos la opción “New repository”.



Una vez hecho esto asignamos un nombre al nuevo repositorio, y seleccionamos si será público o privado. Se le puede añadir una descripción. Para finalizar hacemos clic en “Create Repository”.

The screenshot shows the GitHub 'Create a new repository' page. At the top, it says 'Create a new repository' and provides a brief explanation of what a repository is. Below this, there's a note about required fields marked with an asterisk. The form includes fields for 'Owner' (set to 'Carorou25') and 'Repository name' (set to 'Ejemplo', with a green checkmark indicating it's available). There's a 'Description (optional)' text area. Under 'Visibility', the 'Private' option is selected. The 'Initialize this repository with:' section has 'Add a README file' checked. The 'Add .gitignore' section shows '.gitignore template: None'. The 'Choose a license' section shows 'License: None'. At the bottom, a note states 'You are creating a private repository in your personal account.'

¿Cómo crear una rama en Git?

Antes de trabajar con git debemos configurarlo si es que lo usamos por primera vez, con los siguientes comandos:

```
$ git config --global user.name tuNombre
```

```
$ git config --global user.email tuMail
```

```
$ git init → inicializo git
```

Las ramas pueden definirse como “punteros” para cada uno de los cambios o versiones que queremos armar de nuestro proyecto o desarrollo.

Cada rama representa una línea independiente de desarrollo. Git cuenta con una rama principal “master o main” de la cual parten las demás ramas que se pueden crear.

Si utilizo en la consola el comando `$ git branch` me dice en que rama estoy actualmente. Lo hacemos con el comando:

```
$ git branch
```

Para crear una rama nueva utilizo el siguiente comando:

```
$ git branch “nombre que le queremos asignar “
```

¿Cómo cambiar a una rama en Git?

Si quiero cambiar de rama trabajo, lo hago con el siguiente comando:

```
$ git checkout "Nombre de la rama a cambiar"
```

¿Cómo fusionar ramas en Git?

Si quiero unificar dos ramas, primero tengo que ubicarme sobre la rama en la que quiero recibir los cambios con el comando:

```
$ git checkout "Nombre de la rama a recibir los cambios"
```

Luego, para unificar:

```
$ git merge "rama origen" "rama de destino"
```

La rama origen es la rama que tiene los cambios que quiero que reciba la rama de destino.

¿Cómo crear un commit en Git?

El comando commit se utiliza para subir los archivos que tenemos en el área de preparación al repositorio.

Primero debemos agregar los cambios al área de preparación. Se puede agregar un archivo específico o varios archivos.

```
$ git add Archivo
```

Luego para pasar esos cambios de forma definitiva al repositorio se utiliza el siguiente comando:

```
$ git commit -m "mensaje" → se debe agregar un mensaje sobre los cambios
```

¿Cómo enviar un commit a GitHub?

Si queremos pasar este archivo a nuestro repositorio remoto, lo podemos hacer con el siguiente comando:

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

Origin: es el nombre por defecto del repositorio remoto.

Master: es el nombre de la rama creada por defecto.

¿Qué es un repositorio remoto?

Son los archivos que se encuentran alojados en algún servidor externo y que puede ser accedido desde cualquier lugar. Por ejemplo: Git Hub.

¿Cómo agregar un repositorio remoto a Git?

Para comunicar el repositorio local con el remoto, utilizamos el siguiente comando:

```
$ git remote add origin "url del repositorio remoto"
```

¿Cómo empujar cambios a un repositorio remoto?

Si queremos actualizar un repositorio local del repositorio remoto correspondiente, lo hacemos a través del comando:

```
$ git add Nombre del archivo
```

```
$ git commit -m "..."
```

```
$ git push -u origin "nombre de la rama"
```

¿Cómo tirar de cambios de un repositorio remoto?

Si quisiera descargar un repositorio completo, lo puedo clonar de la siguiente manera:

```
$ git clone "url del repositorio"
```

Si quiero solo un archivo del repositorio:

```
$ git pull origin "rama o archivo a descargar"
```

¿Qué es un fork de repositorio?

Un fork de repositorio es una copia de un repositorio base, para trabajar en un software de manera independiente y desarrollar una versión personalizada de un software. Es común en software de código abierto.

¿Cómo crear un fork de un repositorio?

Pasos para crear un fork de un repositorio:

- Buscar el repositorio que se desea forkear.

- Buscar y hacer clic en el botón Fork en la esquina superior derecha. Al hacer esto GitHub creará una copia exacta del repositorio en nuestra cuenta.
- Después de hacer el fork, serás redirigido a tu propia copia del repositorio. Aquí puedes realizar cambios libremente sin afectar el proyecto original.
- Si se desea contribuir al proyecto original con los cambios realizados, se puede hacer un pull request.

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

Una solicitud de extracción es un método de presentar contribuciones a un proyecto de desarrollo abierto. Ocurre cuando un colaborador pide al creador de un proyecto que considere usar su código.

Paso a seguir:

- Hacer un fork del repositorio donde quiero contribuir
- Clonar el repositorio a la máquina local y hacer los cambios.
- Confirmar cambios y enviarlos a nuestro repositorio de GitHub.
- Ir al repositorio original y hacer clic en el botón “Nueva solicitud de extracción”.
- Seleccionar la rama que contiene los cambios y crear una nueva solicitud de extracción.

¿Cómo aceptar una solicitud de extracción?

Para aceptar una solicitud de extracción hay que seguir los siguientes pasos:

- Ir al repositorio donde se encuentra la solicitud de extracción.
- Hacer clic en “pull requests”
- Seleccionar la solicitud de extracción que se desea aceptar
- Revisar los cambios en la pestaña “files changed”
- Si deseo incorporar los cambios a mi rama, hacemos clic en “merge pull request”
- Luego “confirm merge”.

¿Qué es una etiqueta en Git?

Una etiqueta en git es una referencia que se utiliza para marcar puntos importantes, como lanzamientos o versiones específicas.

Git utiliza dos tipos principales de etiquetas: ligeras y anotadas. Una etiqueta ligera es parecida a una rama que no cambia, es un puntero a un commit específico. Las

etiquetas anotadas se guardan en la base de datos de Git como objetos enteros. Contienen el nombre del etiquetador, correo electrónico y fecha.

Es recomendable crear etiquetas anotadas por la información que contienen. Pero si queremos una etiqueta temporal, es recomendable usar las ligeras.

¿Cómo crear una etiqueta en Git?

Para crear una etiqueta anotada en Git usamos el siguiente comando:

```
$ git tag -a "nombre etiqueta" -m "mensaje de la etiqueta"
```

Para ver la información que contiene usamos el comando:

```
$ git show "nombre etiqueta"
```

Para crear una etiqueta simple solo usamos el comando git tag.

¿Cómo enviar una etiqueta a GitHub?

Para enviar las etiquetas creadas a GitHub:

```
$ git push origin "nombre de la etiqueta"
```

Si tenemos varias etiquetas en enviar:

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

¿Qué es un historial de Git?

Es un registro de todos los commits que se han hecho a un proyecto junto con la fecha, el autor, y un mensaje.

¿Cómo ver el historial de Git?

El comando git log nos muestra el historial completo de commits. Pero si queremos ver solo un número reducido de commits, lo hacemos con el siguiente comando:

```
$ git log -n "limite"
```

¿Cómo buscar en el historial de Git?

Para buscar dentro del historial de git tenemos varias opciones:

- Para buscar commits que contengan una palabra clave
`$ git log - -grep = "palabra clave"`
- Para buscar commits que han modificado un archivo específico
`$ git log - -Nombre del Archivo`
- Para buscar commits con nombre de autor.
`$ git log - -author ="Nombre del autor"`
- Búsqueda de commits con fechas específicas
`$ git log - -since = "desde" - -until = "hasta"`

¿Cómo borrar el historial en Git?

Hay distintas formas de eliminar en git:

- Para eliminar del stage todos los archivos y carpetas del proyecto.
`$ git reset`
- Si quiero eliminar del stage un archivo específico
`$ git reset NombreArchivo`

¿Qué es un repositorio privado en GitHub?

Es un repositorio que no está disponible para cualquiera lo vea o lo clone.

¿Cómo crear un repositorio privado en GitHub?

Una forma es al momento de creación de un nuevo repositorio, cuando lo configuramos lo hacemos como privado.

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

Para invitar a alguien a nuestro repositorio privado debemos seguir los siguientes pasos:

- Solicitar nombre de usuario de la persona que se desea invitar.
- En la sección acceso, hacer clic en colaboradores.
- Hacer clic en agregar personas
- Hacer clic en agregar nombre al repositorio.

- El usuario recibirá un correo electrónico invitándolo al repositorio.

¿Qué es un repositorio público en GitHub?

Un repositorio público es un repositorio que es visible para cualquier persona.

¿Cómo crear un repositorio público en GitHub?

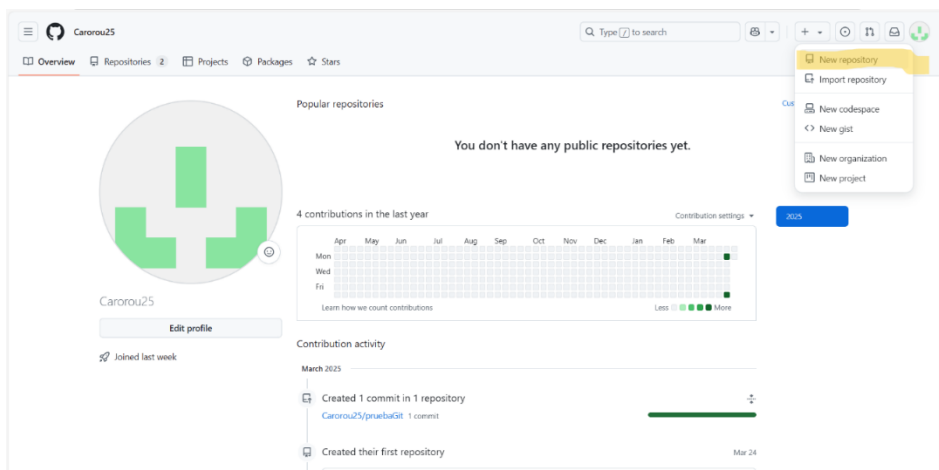
Al momento de creación de un nuevo repositorio en GitHub cuando lo configuramos, elegimos la opción de público.

¿Cómo compartir un repositorio público en GitHub?

Para compartir un repositorio público en GitHub copiamos y compartimos la url de nuestro perfil.

2) Realizar la siguiente actividad:

- **Crear un repositorio.**
 - o Dale un nombre al repositorio.
 - o Elije el repositorio sea público.
 - o Inicializa el repositorio con un archivo.



Create a new repository

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

Required fields are marked with an asterisk ().*

Owner * Carorou25 / Repository name * Repo1
Repo1 is available.

Great repository names are short and memorable. Need inspiration? How about [bug-free-funicular](#)?

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 .gitignore 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](#).

- **Agregando un Archivo**

- o Crea un archivo simple, por ejemplo, "mi-archivo.txt".
 - o Realiza los comandos `git add .` y `git commit -m "Agregando mi-archivo.txt"` en la línea de comandos.
 - o Sube los cambios al repositorio en GitHub con `git push origin main` (o el nombre de la rama correspondiente).

 MINGW64:/c/Users/PC/desktop/trabajo

```
PC@DESKTOP-J6RK3NT MINGW64 ~ (master)
$ cd desktop

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop (master)
$ mkdir trabajo

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop (master)
$ cd trabajo

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo (master)
$ git init
Initialized empty Git repository in C:/Users/PC/Desktop/trabajo/.git/

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo (master)
$ git remote add origin https://github.com/Carorou25/Repo1.git

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo (master)
$ git add archivo1.txt

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   archivo1.txt

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo (master)
$ git commits -m "Esto es un archivo de texto"
git: 'commits' is not a git command. See 'git --help'.

The most similar command is
    commit

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo (master)
$ git commit -m "Esto es un archivo de texto"
[master (root-commit) eb93a03] Esto es un archivo de texto
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 archivo1.txt

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo (master)
$ git log
commit eb93a03d91e22577da8784000c2cfcd3f0e93e8c (HEAD -> master)
Author: Carorou25 <alexiarousseaux20@gmail.com>
Date:   Mon Mar 31 20:52:52 2025 -0300

    Esto es un archivo de texto

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo (master)
$ |
```

MINGW64:/c/Users/PC/desktop/trabajo

```
PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo (master)
$ git add archivo1.txt
```

```
PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo (master)
$ git status
On branch master
```

No commits yet

```
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   archivo1.txt
```

```
PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo (master)
$ git commits -m "Esto es un archivo de texto"
git: 'commits' is not a git command. See 'git --help'.
```

The most similar command is
commit

```
PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo (master)
$ git commit -m "Esto es un archivo de texto"
[master (root-commit) eb93a03] Esto es un archivo de texto
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 archivo1.txt
```

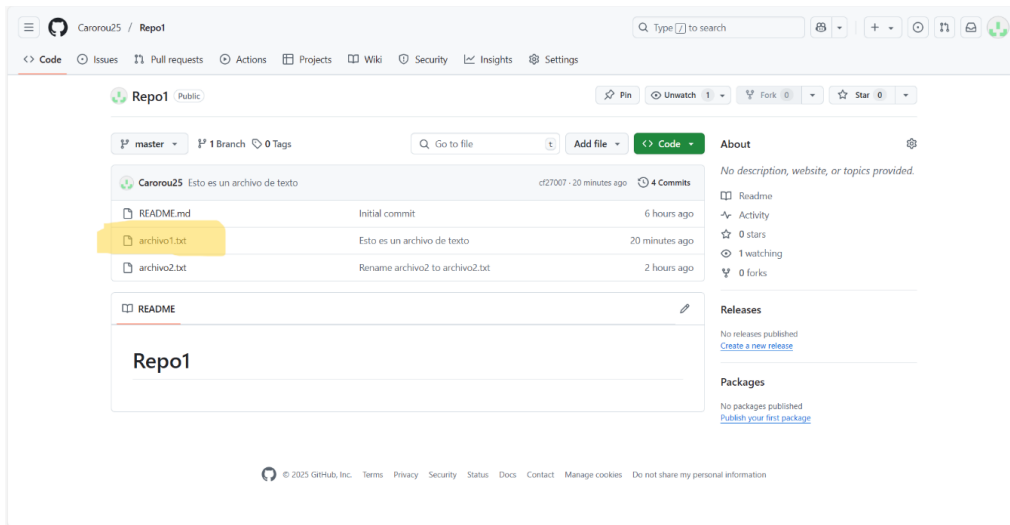
```
PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo (master)
$ git log
commit eb93a03d91e22577da8784000c2cfc3f0e93e8c (HEAD -> master)
Author: Carorou25 <alexiarousseaux20@gmail.com>
Date:   Mon Mar 31 20:52:52 2025 -0300
```

Esto es un archivo de texto

```
PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo (master)
$ git push -u origin master
To https://github.com/Carorou25/Repo1.git
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/Carorou25/Repo1.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

```
PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo (master)
$ git pull origin master --rebase
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (8/8), 2.64 KiB | 79.00 KiB/s, done.
From https://github.com/Carorou25/Repo1
* branch          master      -> FETCH_HEAD
* [new branch]     master      -> origin/master
Successfully rebased and updated refs/heads/master.
```

```
PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo (master)
$ |
```



• Creando Branchs

o Crear una Branch

o Realizar cambios o agregar un archivo

o Subir la Branch

MINGW64:/c:/Users/PC/Desktop/trabajo/clon/Repo1

```
PC@DESKTOP-J6RK3NT MINGW64 ~/Desktop/trabajo (master)
$ cd clon

PC@DESKTOP-J6RK3NT MINGW64 ~/Desktop/trabajo/clon (master)
$ git clone https://github.com/Carorou25/Repo1
Cloning into 'Repo1'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 11 (delta 1), reused 3 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (11/11), done.
Resolving deltas: 100% (1/1), done.

PC@DESKTOP-J6RK3NT MINGW64 ~/Desktop/trabajo/clon (master)
$ cd Repo1

PC@DESKTOP-J6RK3NT MINGW64 ~/Desktop/trabajo/clon/Repo1 (master)
$ git branch OtraRama

PC@DESKTOP-J6RK3NT MINGW64 ~/Desktop/trabajo/clon/Repo1 (master)
$ git branch
  OtraRama
* master

PC@DESKTOP-J6RK3NT MINGW64 ~/Desktop/trabajo/clon/Repo1 (master)
$ git checkout OtraRama
Switched to branch 'OtraRama'

PC@DESKTOP-J6RK3NT MINGW64 ~/Desktop/trabajo/clon/Repo1 (OtraRama)
$ git branch
* OtraRama
  master

PC@DESKTOP-J6RK3NT MINGW64 ~/Desktop/trabajo/clon/Repo1 (OtraRama)
$ |
```

```

PC@DESKTOP-J6RK3NT MINGW64 ~/Desktop/trabajo/clon/Repo1 (OtraRama)
$ touch "archivo3.txt"

PC@DESKTOP-J6RK3NT MINGW64 ~/Desktop/trabajo/clon/Repo1 (OtraRama)
$ git add .

PC@DESKTOP-J6RK3NT MINGW64 ~/Desktop/trabajo/clon/Repo1 (OtraRama)
$ git commit -m"Agrego cambios"
[OtraRama ef60aa5] Agrego cambios
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 archivo3.txt

PC@DESKTOP-J6RK3NT MINGW64 ~/Desktop/trabajo/clon/Repo1 (OtraRama)
$ git add clone
fatal: pathspec 'clone' did not match any files

PC@DESKTOP-J6RK3NT MINGW64 ~/Desktop/trabajo/clon/Repo1 (OtraRama)
$ git push origin OtraRama
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 268 bytes | 268.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'OtraRama' on GitHub by visiting:
remote:   https://github.com/Carorou25/Repo1/pull/new/OtraRama
remote:
To https://github.com/Carorou25/Repo1
 * [new branch]      OtraRama -> OtraRama

PC@DESKTOP-J6RK3NT MINGW64 ~/Desktop/trabajo/clon/Repo1 (OtraRama)
$ |

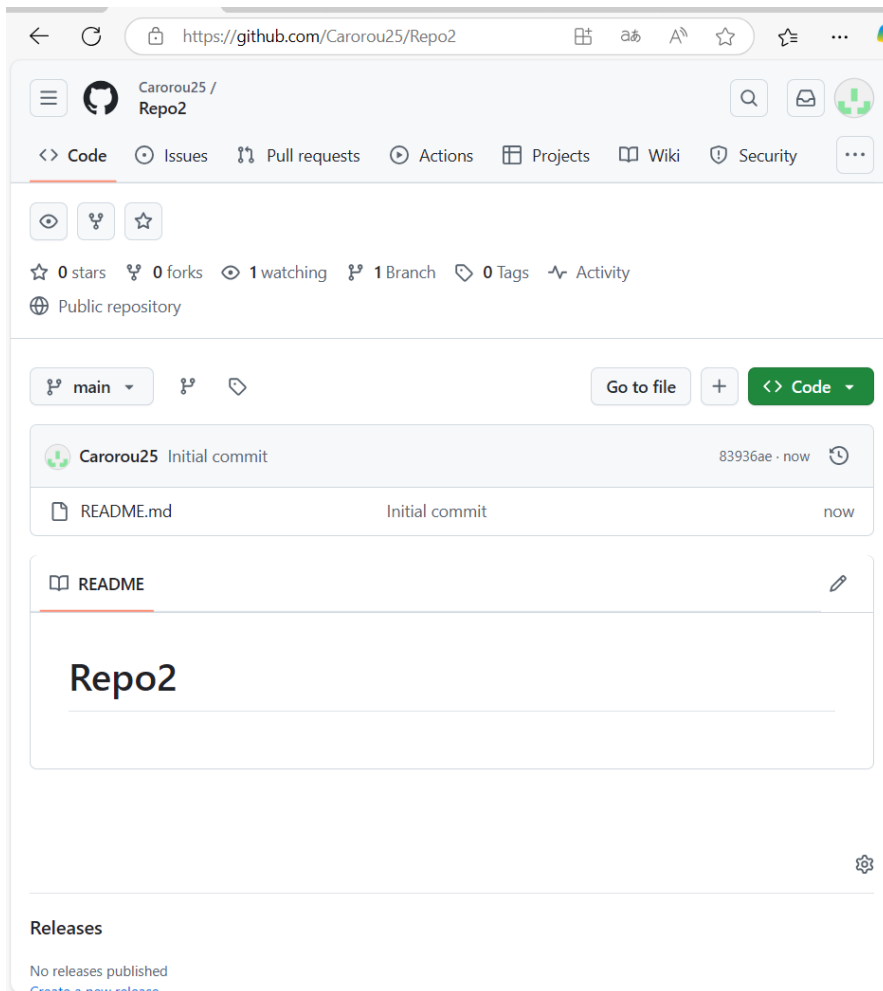
```

Carorou25 / Repo1

3) Realizar la siguiente actividad:

Paso 1: Crear un repositorio en GitHub

- Ve a GitHub e inicia sesión en tu cuenta.
- Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.
- Asigna un nombre al repositorio, por ejemplo, conflict-exercise.
- Opcionalmente, añade una descripción.
- Marca la opción "Initialize this repository with a README".
- Haz clic en "Create repository".




Paso 2: Clonar el repositorio a tu máquina local

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

`exercise.git`).

- Abre la terminal o línea de comandos en tu máquina.
- Clona el repositorio usando el comando:

`git clone`

 MINGW64:/c/Users/PC/desktop/trabajo/Repo2

```
PC@DESKTOP-J6RK3NT MINGW64 ~ (master)
$ cd trabajo
bash: cd: trabajo: No such file or directory

PC@DESKTOP-J6RK3NT MINGW64 ~ (master)
$ cd desktop

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop (master)
$ cd trabajo

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo (master)
$ git clone https://github.com/Carorou25/Repo2.git
Cloning into 'Repo2'...
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.

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo (master)
$ git init
Reinitialized existing Git repository in C:/Users/PC/Desktop/trabajo/.git/

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo (master)
$ cd Repo2

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (main)
$ git init
Reinitialized existing Git repository in C:/Users/PC/Desktop/trabajo/Repo2/.git/

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (main)
$
```

Entra en el directorio del repositorio:

`cd conflict-exercise`

Paso 3: Crear una nueva rama y editar un archivo

- Crea una nueva rama llamada feature-branch:

`git checkout -b feature-branch`

```
PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (main)
$ git branch feature-branch

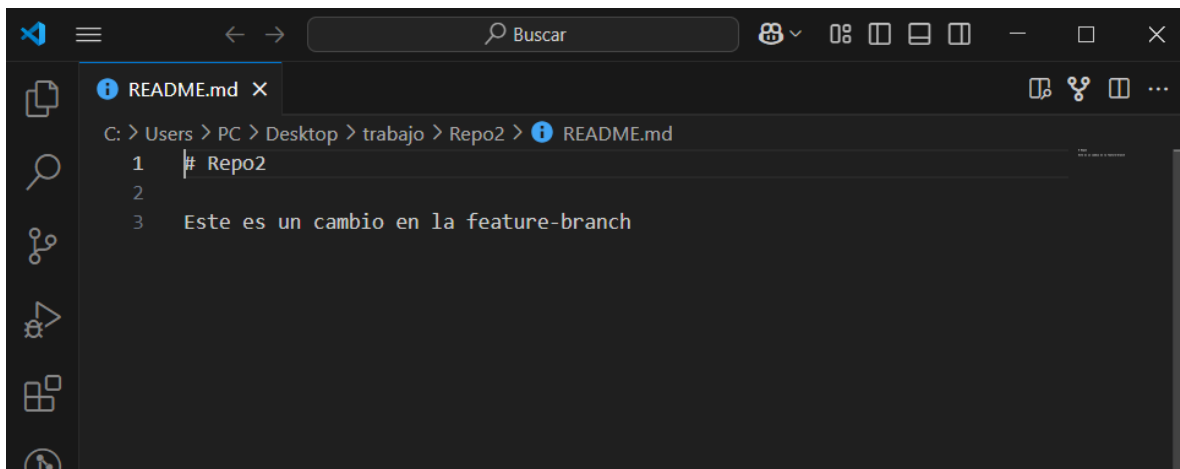
PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (main)
$ git checkout feature-branch
Switched to branch 'feature-branch'

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (feature-branch)
$ git branch
* feature-branch
  main

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (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:

```
PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (feature-branch)
$ git add README.md

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (feature-branch)
$ git commit -m "Added a line feature-branch"
[feature-branch bbc8b93] Added a line feature-branch
1 file changed, 3 insertions(+), 1 deletion(-)

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (feature-branch)
$ git status
On branch feature-branch
nothing to commit, working tree clean

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (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"

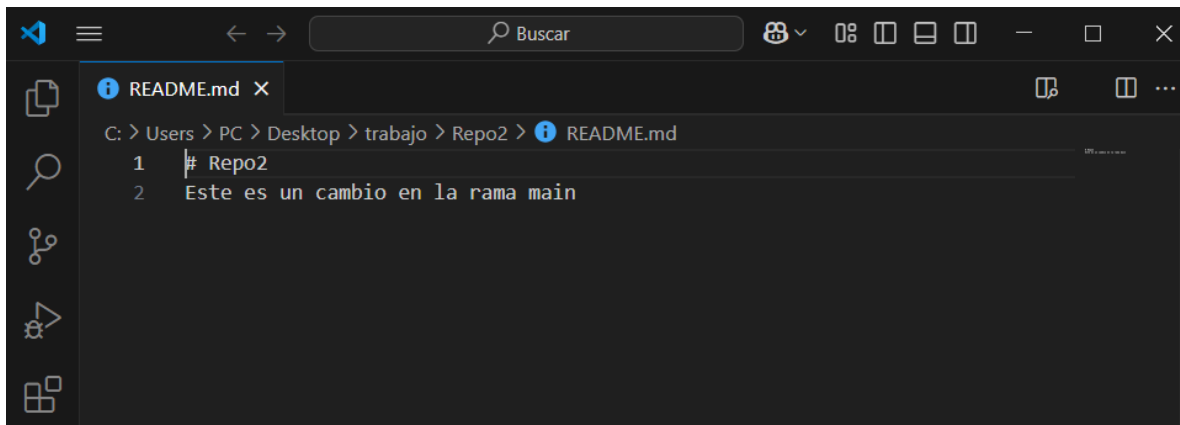
```
PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (feature-branch)
$ git branch
* feature-branch
  main

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (main)
$ git add README.md

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (main)
$ git commit -m"Added a line un main branch"
[main 2a14a2d] Added a line un main branch
1 file changed, 2 insertions(+), 1 deletion(-)

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (main)
$
```



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.

```
PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (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.

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (main|MERGING)
$
```

Paso 6: Resolver el conflicto

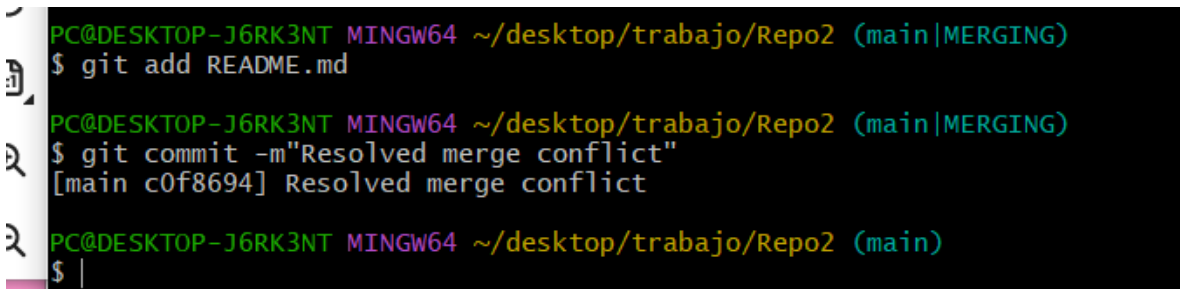
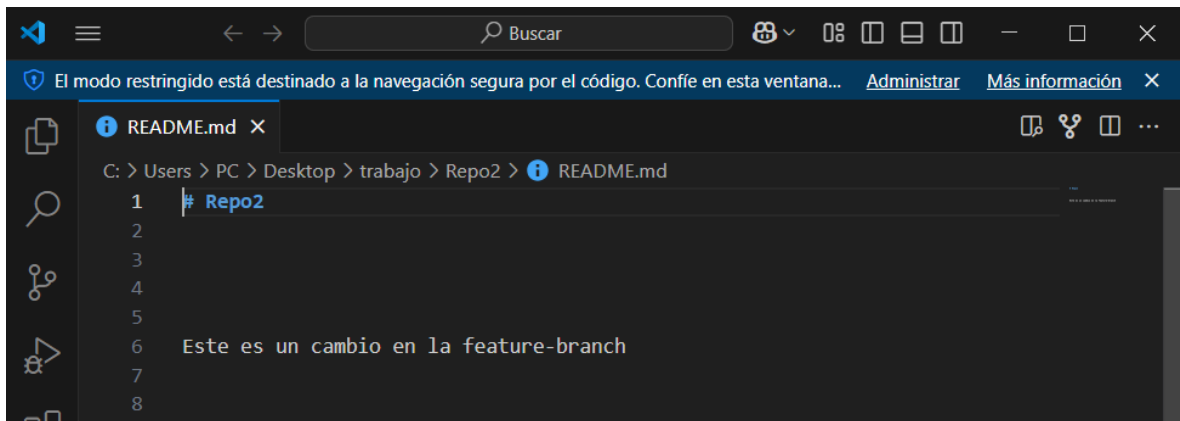
- Abre el archivo README.md en tu editor de texto. Verás algo similar a esto:
<<<<<<< HEAD

Este es un cambio en la main branch. =====

Este es un cambio en la feature branch. >>>>>>> feature-branch

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

- Edita el archivo para resolver el conflicto y guarda los cambios (Se debe borrar lo marcado en verde en el archivo donde estes solucionando el conflicto. Y se debe borrar la parte del texto que no se quiera dejar).
- Añade el archivo resuelto y completa el merge: `git add README.md` `git commit -m "Resolved merge conflict"`



Paso 7: Subir los cambios a GitHub

- Sube los cambios de la rama main al repositorio remoto en GitHub: `git push origin main`
- También sube la feature-branch si deseas: `git push origin feature-branch`

```
[main c0f8694] Resolved merge conflict
PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (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% (4/4), done.
Writing objects: 100% (9/9), 819 bytes | 136.00 KiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/Carorou25/Repo2.git
   83936ae..c0f8694  main -> main

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (main)
$ git push feature-branch
fatal: 'feature-branch' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (main)
$ git branch
   feature-branch
* main

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (main)
$ git push feature-branch
fatal: 'feature-branch' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (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/Carorou25/Repo2/pull/new/feature-branch
remote:
To https://github.com/Carorou25/Repo2.git
 * [new branch]      feature-branch -> feature-branch

PC@DESKTOP-J6RK3NT MINGW64 ~/desktop/trabajo/Repo2 (main)
$ |
```

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.

