

Alumna: Andrea Julia Ayala – Comisión 24

Práctico 2: Git y GitHub

Objetivo:

El estudiante desarrollará competencias para trabajar con Git y GitHub, aplicando conceptos fundamentales de control de versiones, colaboración en proyectos y resolución de conflictos, en un entorno simulado y guiado.

Resultados de aprendizaje:

1. Comprender los conceptos básicos de Git y GitHub: Identificar y explicar los principales términos y procesos asociados con Git y GitHub, como repositorios, ramas, commits, forks, etiquetas y repositorios remotos.
2. Manejar comandos esenciales de Git: Ejecutar comandos básicos para crear, modificar, fusionar y gestionar ramas, commits y repositorios, tanto en local como en remoto.
3. Aplicar técnicas de colaboración en GitHub: Configurar y utilizar repositorios remotos, realizar forks, y gestionar pull requests para facilitar el trabajo colaborativo.
4. Resolver conflictos en un entorno de control de versiones: Identificar, analizar y solucionar conflictos de merge generados en un flujo de trabajo con múltiples ramas.

Actividades

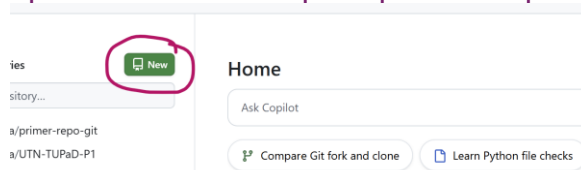
- 1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas) :

- ¿Qué es GitHub?

Es una plataforma en línea para gestionar proyectos de software usando Git. Allí los desarrolladores pueden subir repositorios de código, colaborar entre sí en proyectos de software, revisar cambios y gestionar versiones.

- ¿Cómo crear un repositorio en GitHub?

Para crear un repositorio en GitHub, primero debemos iniciar sesión en la plataforma (suponiendo que ya tenemos una cuenta creada; de lo contrario debemos hacer eso primero). Luego, debemos seleccionar la opción “new” que aparece en el menú principal a la izquierda:



Dentro de esta ventana debemos ponerle un nombre. Opcionalmente podemos incluir una descripción y un ReadMe. Además podemos elegir si queremos que sea público (gratuito) o privado (para esto pueden haber recargos).

Por último, apretamos en “create repository”, que se encuentra al final de la pantalla:


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 *

 andyjayala ▾

Repository name *

/

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

Description (optional)

☒



Public

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

☐



Private

You choose who can see and commit to this repository.

Initialize this repository with:

☐

Add a README file

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

Add .gitignore

.gitignore template: None ▾

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

Choose a license

License: None ▾

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

 You are creating a public repository in your personal account.

Create repository

- ¿Cómo crear una rama en Git?

Podemos crear una rama en Git en la terminal, mediante la opción “branch” seguido del nombre que queremos darle a esa rama, de la siguiente manera:

git branch nombre

- ¿Cómo cambiar a una rama en Git?

Colocamos en la terminal la siguiente frase, siendo “nombre” la rama a la que queremos acceder:

`git checkout nombre`

- ¿Cómo fusionar ramas en Git?

Colocamos en la terminal la siguiente frase, siendo “nombre” la rama que queremos fusionar:

`git merge nombre`

- ¿Cómo crear un commit en Git?

Primero añadimos las modificaciones usando add: `git add archivo_o_carpeta`

Luego hacemos el commit, con una breve descripción: `git commit -m “Descripción”`

- ¿Cómo enviar un commit a GitHub?

`git push origin nombre`

- ¿Qué es un repositorio remoto?

Es la versión del repositorio que se aloja en un servidor (en nuestro caso, GitHub). Permite colaboración e intercambio de código entre diferentes desarrolladores

- ¿Cómo agregar un repositorio remoto a Git?

`git remote add origin https://github.com/tu-usuario/nombre-del-repositorio.git`

- ¿Cómo empujar cambios a un repositorio remoto?

`git push origin nombre`

- ¿Cómo tirar de cambios de un repositorio remoto?

`git pull origin nombre`

- ¿Qué es un fork de repositorio?

Es un duplicado de un repositorio alojado en GitHub, que permite editar el código sin alterar el repositorio original.

- ¿Cómo crear un fork de un repositorio?

Ingresamos al repositorio original en Github, hacemos click en el boton “Fork” y se creará automáticamente una copia del repositorio en la cuenta

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

Después de subir los cambios a GitHub, en el repositorio original hay que hacer click en 'New Pull Request', elegir la rama del fork con las modificaciones y crear la solicitud. Esto permitirá que revisen los cambios y los aprueben si son correctos

- ¿Cómo aceptar una solicitud de extracción?

Se integran los cambios con con la opcion “Merge pull request”

- ¿Qué es un etiqueta en Git?

Como su nombre lo indica, sirven para señalar puntos específicos en la historia de un repositorio

- ¿Cómo crear una etiqueta en Git?

`git push tag nombre_etiqueta`

- ¿Cómo enviar una etiqueta a GitHub?

`git push origin nombre_etiqueta`

- ¿Qué es un historial de Git?

Es el registro completo del repositorio en el que se incluyen todos los commits realizados

- ¿Cómo ver el historial de Git?

Se puede ver ingresando “git log”

- ¿Cómo buscar en el historial de Git?

Depende de lo que se quiera buscar, podemos usar distintas opciones:

`git log --grep="texto a buscar"` (para buscar por mensajes de commit)

`git log -S "texto"` (para buscar por cambios de archivo)

- ¿Cómo borrar el historial de Git?

No se recomienda borrar el historial de Git. Una opción es reescribir el historial utilizando comandos como `git rebase` o `git filter-branch`

- ¿Qué es un repositorio privado en GitHub?

Es un repositorio al cual solo tiene acceso el dueño de la cuenta y a quienes quiera compartirles el acceso.

- ¿Cómo crear un repositorio privado en GitHub?

Como se mostró en las primeras imágenes, se selecciona la opción “private” en vez de “public” al momento de crear el repositorio.

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

Dentro del repositorio, en “settings”, “Manage access” se invita al usuario ingresando su nombre o correo electrónico y asignándole un rol

- ¿Qué es un repositorio público en GitHub?

Es un repositorio al cual puede acceder cualquier persona (a diferencia del privado para el cual se necesita un permiso/acceso).

- ¿Cómo crear un repositorio público en GitHub?

Como se mencionó anteriormente, seleccionando la opción “public” en la parte de visibilidad.

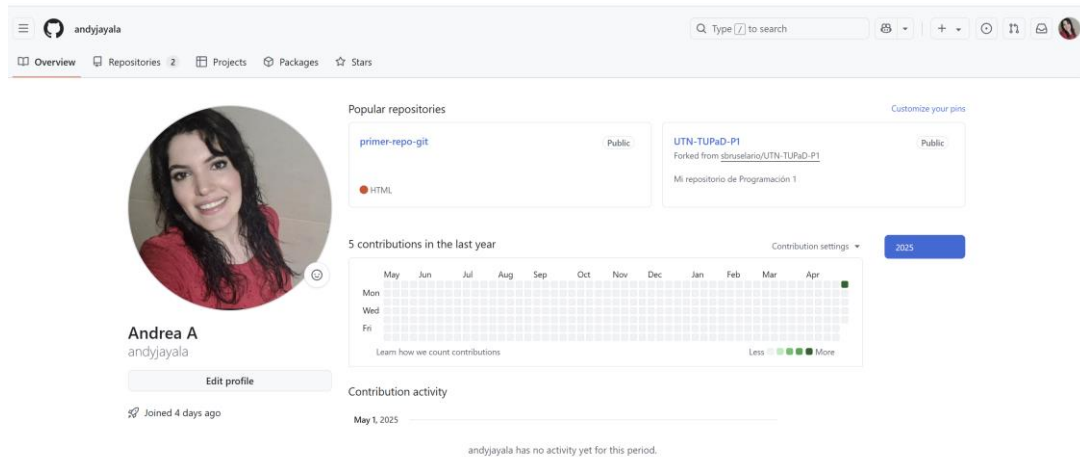
- ¿Cómo compartir un repositorio público en GitHub?

Se puede compartir mediante el envío de la URL del repositorio. Al ser público no se necesitan accesos o permisos específicos.

2) Realizar la siguiente actividad:

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

Cree mi cuenta con el nombre de usuario andyjayala, un repositorio público y subí un archivo basándome en uno de los ejemplos de los videos propuestos en la cursada. Además hice el Fork indicado en clase como se ve en la imagen:



- 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).

```
C0804d0..a14e8f6 main -> main
PS C:\Users\andre\OneDrive\Desktop\CARPETAS GIT - VSC\UTN TP\UTN-TUPaD-P1> git add .
PS C:\Users\andre\OneDrive\Desktop\CARPETAS GIT - VSC\UTN TP\UTN-TUPaD-P1> git commit -m "Agregué un archivo 'mi-archivo.txt'"
[main 6047cdb] Agregué un archivo 'mi-archivo.txt'
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 02 Trabajo Colaborativo/mi-archivo.txt
PS C:\Users\andre\OneDrive\Desktop\CARPETAS GIT - VSC\UTN TP\UTN-TUPaD-P1> git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 351 bytes | 175.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/andyjayala/UTN-TUPaD-P1.git
a14e8f6..6047cdb main -> main
PS C:\Users\andre\OneDrive\Desktop\CARPETAS GIT - VSC\UTN TP\UTN-TUPaD-P1> |
```

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

```
andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/UTN TP/UTN-TUPaD-P1/02 Trabajo Colaborativo (ramatp)
• $ git branch ramanueva

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/UTN TP/UTN-TUPaD-P1/02 Trabajo Colaborativo (ramatp)
• $ git checkout ramanueva
Switched to branch 'ramanueva'

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/UTN TP/UTN-TUPaD-P1/02 Trabajo Colaborativo (ramanueva)
○ $

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/UTN TP/UTN-TUPaD-P1/02 Trabajo Colaborativo (ramatp)
• $ git checkout ramanueva
Switched to branch 'ramanueva'

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/UTN TP/UTN-TUPaD-P1/02 Trabajo Colaborativo (ramanueva)
• $ touch mi-archivo-dos.txt

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/UTN TP/UTN-TUPaD-P1/02 Trabajo Colaborativo (ramanueva)
• $ git add .

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/UTN TP/UTN-TUPaD-P1/02 Trabajo Colaborativo (ramanueva)
• $ git commit -m "Cree una rama nueva y un archivo al que llamé mi-archivo-dos.txt"
[ramanueva 3d8f74d] Cree una rama nueva y un archivo al que llamé mi-archivo-dos.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 02 Trabajo Colaborativo/mi-archivo-dos.txt

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/UTN TP/UTN-TUPaD-P1/02 Trabajo Colaborativo (ramanueva)
• $ git commit -m "Cree una rama nueva y un archivo al que llamé mi-archivo-dos.txt"
[ramanueva 3d8f74d] Cree una rama nueva y un archivo al que llamé mi-archivo-dos.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 02 Trabajo Colaborativo/mi-archivo-dos.txt

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/UTN TP/UTN-TUPaD-P1/02 Trabajo Colaborativo (ramanueva)
• $ git push origin ramanueva
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 372 bytes | 31.00 KiB/s, done.
Total 3 (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 'ramanueva' on GitHub by visiting:
remote:   https://github.com/andyjayala/UTN-TUPaD-P1/pull/new/ramanueva
remote:
To https://github.com/andyjayala/UTN-TUPaD-P1.git
 * [new branch]      ramanueva -> ramanueva

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/UTN TP/UTN-TUPaD-P1/02 Trabajo Colaborativo (ramanueva)
○ $
```

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".

Create a new repository

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

Required fields are marked with an asterisk (*).

Owner * / Repository name *



 andyjayala / conflict-exercise

✓ conflict-exercise is available.

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

Description (optional)

Ejercicio TP semana 2

- ☒  Public
Anyone on the internet can see this repository. You choose who can commit.
- ☐  Private
You choose who can see and commit to this repository.

Initialize this repository with:

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

Add .gitignore

.gitignore template: None

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

Choose a license

License: None

```
andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto (master)
$ git clone https://github.com/andyjayala/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.

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto (master)
$
```

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`

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"`

(((En esta parte tuve problemas así que hice primero los cambios en el archivo en el main y luego en la rama 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"`

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.

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
```

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.

```
andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto (master)
● $ git clone https://github.com/andyjayala/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.

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto (master)
● $ cd conflict-exercise

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto/conflict-exercise (main)
○ $
```

```
andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto (master)
● $ cd conflict-exercise

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto/conflict-exercise (main)
● $ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto/conflict-exercise (feature-branch)
○ $
```

Tuve un inconveniente y realicé los cambios requeridos comenzando en la rama main y luego pasando a la rama feature-branch

```
andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto/conflict-exercise (main)
● $ git commit -m "Added a line in main branch"
[main c73ee02] Added a line in main branch
1 file changed, 2 insertions(+)

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto/conflict-exercise (main)
○ $
```

Ln 4, Col 36 Spaces: 4 UTF-8 CRLF {} Markdown

```
andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto/conflict-exercise (main)
• $ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto/conflict-exercise (main)
• $ git add .

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto/conflict-exercise (main)
• $ git commit -m "Modifico la main branch"
[main 0319770] Modifico la main branch
 1 file changed, 1 insertion(+), 2 deletions(-)

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto/conflict-exercise (main)
○ $ git checkout feature-branch
```

```
andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto/conflict-exercise (main)
• $ git checkout feature-branch
Switched to branch 'feature-branch'

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto/conflict-exercise (feature-branch)
• $ git add .

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto/conflict-exercise (feature-branch)
• $ git commit -m "agregué texto en rama feature-branch"
[feature-branch 6aacef5] agregué texto en rama feature-branch
 1 file changed, 1 insertion(+)

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto/conflict-exercise (feature-branch)
○ $
```

```
andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto/conflict-exercise (feature-branch)
• $ git merge main
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

README.md ! ✕

README.md > # Ejercicio TP semana 2<<<<<< HEADAdded a line in feature-branch

```
1 # conflict-exercise
2 Ejercicio TP semana 2
   Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
3 <<<<<< HEAD (Current Change)
4 Added a line in feature-branch
5 =====
6 Este es un cambio en la main branch
7 >>>>>> main (Incoming Change)
8
```

```
andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto/conflict-exercise (feature-branch|MERGING)
• $ git add README.md

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto/conflict-exercise (feature-branch|MERGING)
• $ git commit -m "se resuelve el conflicto generado con el merge de las ramas"
[feature-branch 8153be9] se resuelve el conflicto generado con el merge de las ramas

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto/conflict-exercise (feature-branch)
○ $
```

```
andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto/conflict-exercise (feature-branch)
• $ git status
On branch feature-branch
nothing to commit, working tree clean

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto/conflict-exercise (feature-branch)
• $ git push origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 575 bytes | 191.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/andyjayala/conflict-exercise.git
b198f9c..0319770 main -> main

andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto/conflict-exercise (feature-branch)
• $ git push origin feature-branch
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 718 bytes | 179.00 KiB/s, done.
Total 6 (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/andyjayala/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/andyjayala/conflict-exercise.git
* [new branch]      feature-branch -> feature-branch
⚡
andre@BOOK-PTH14F7CGI MINGW64 ~/OneDrive/Desktop/CARPETAS GIT - VSC/Conflicto/conflict-exercise (feature-branch)
```

conflict-exercise / README.md

andyjayala se resuelve el conflicto generado con el merge de las ramas

Preview

Code

Blame

7 lines (7 loc) · 143 Bytes

Code 55% faster with GitHub Copilot