

Actividades

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

¿Qué es GitHub?

GitHub es una plataforma basada en la nube donde se puede guardar en un repositorio, compartir y trabajar en forma colaborativa, proyectos de código. También tiene algunas características de red social, enfocada a desarrolladores.

¿Cómo crear un repositorio en GitHub?

Para utilizar Github es necesario crear una cuenta. A partir de ahí es posible crear un repositorio desde la web directamente desde cero o hacer un “fork” para copiar un repositorio ya existente.

¿Cómo crear una rama en Git?

Desde la terminal hay que posicionarse en la ubicación donde se desea crear la rama y utilizar el comando: `git branch *nombre de la rama*`.

¿Cómo cambiar a una rama en Git?

Para elegir en que rama se va a trabajar se utiliza el comando: `git checkout *nombre de la rama*`.

¿Cómo fusionar ramas en Git?

Para fusionar dos ramas primero hay que cambiar a la rama de destino y luego se usa el comando: `git merge *nombre de la rama*` (siendo esta última la rama que se fusiona a la primera).

¿Cómo crear un commit en Git?

Un commit se realiza con el comando: `git commit`.

¿Cómo enviar un commit a GitHub?

El commit con los cambios se envía a Github con el comando: `git push`

¿Qué es un repositorio remoto?

Un repositorio remoto es una ubicación fuera de nuestra computadora local donde se puede almacenar nuestros proyectos, para poder tener una copia de seguridad, compartirlos y poder trabajar en conjunto con otros. En el caso de plataformas como Github además tenemos herramientas para publicación, control de versiones, etc.

¿Cómo agregar un repositorio remoto a Git?

Para esto es necesario configurar la dirección del repositorio remoto donde Git va a sincronizar los archivos que indiquemos, para esto es necesario obtener la dirección del mismo. Se usa el comando: `git remote add origin *direccion del repositorio*` .

¿Cómo empujar cambios a un repositorio remoto?

Una vez realizado el cambio y luego de hacer el commit se procede a usar el comando: `git push -u origin *nombre del repositorio*` (para la primera vez) y luego solo el comando: `git push`

¿Cómo tirar de cambios de un repositorio remoto?

Para traer datos del repositorio remoto se utiliza el comando: `git pull`

¿Qué es un fork de repositorio?

Hacer un fork de un repositorio es realizar una copia de dicho repositorio en nuestro propio repositorio de Github.

¿Cómo crear un fork de un repositorio?

Para realizar un fork se utiliza el botón Fork desde la pantalla principal del repositorio que se necesite copiar.

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

Las solicitudes de extracción solo se pueden abrir entre dos ramas que sean diferentes. Si uno desea que quien administra un repositorio fusione modificaciones o agregados creadas por nosotros se debe enviar una pull request. Se realiza desde la pagina principal del repositorio eligiendo la rama que contiene la modificaciones y elegir la opcion "Crear solicitud de incorporación de cambios".

¿Cómo aceptar una solicitud de extracción?

La persona con acceso al repositorio que recibe la notificación de que tiene un pull request a revisar es quien realiza la acción. Para ello lo que tiene que hacer es entrar a la sección de pull requests del repositorio, revisar los cambios, aprobarlos y por ultimo Enviar revisión.

¿Qué es un etiqueta en Git?

Git tiene permite etiquetar puntos específicos del historial como importantes (ej: v1.0 ; v1.1, v2.0 , etc) . Es una forma de marcar los cambios que tuvo nuestro proyecto en el ciclo de vida del mismo.

¿Cómo crear una etiqueta en Git?

Para crear una etiqueta se usa el comando: `git tag -a [etiqueta]`.

¿Cómo enviar una etiqueta a GitHub?

Para enviar una etiqueta se usa el comando: `git push origin [etiqueta]`.

¿Qué es un historial de Git?

El historial de Git es una forma ordenada de ver los diferentes commits que fueron realizados sobre un repositorio de forma cronológica.

¿Cómo ver el historial de Git?

Para ver el historial del repositorio se usa el comando: `git log`

¿Cómo buscar en el historial de Git?

Para buscar en el historial es útil la utilización de filtros. Se puede filtrar por fecha, autor, o también un texto específico utilizando la opción `-S`

¿Cómo borrar el historial de Git?

La única forma de borrar el historial de los commits es borrando el repositorio y creando uno nuevo, para eso se elimina la carpeta `.git` de nuestro repositorio. En el caso de querer volver a querer reiniciar el mismo se debe agregar todo el repositorio como si se empezara nuevamente, esta operación se utiliza generalmente si se quiere pasar un repositorio de privado a público.

¿Qué es un repositorio privado en GitHub?

Un repositorio privado es aquel al cual nosotros elegimos quien tiene acceso al mismo, tanto para ver como para colaborar con el mismo.

¿Cómo crear un repositorio privado en GitHub?

La elección acerca de como manejamos el acceso a nuestro repositorio, tanto para ver como para trabajar en el mismo. La elección se realiza desde la pantalla de creación del mismo repositorio, bajo la opción "Private".

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

Desde la configuración del repositorio bajo la sección Collaborators están todas las opciones de acceso y visibilidad de dicho repositorio. Los colaboradores se agregan utilizando su cuenta de Github.

¿Qué es un repositorio público en GitHub?

Contrario a un repositorio privado, al elegir que nuestro repositorio sea publico cualquiera puede tener acceso a ver y hacer un fork de nuestro proyecto.

¿Cómo crear un repositorio público en GitHub?

La elección acerca de como manejamos el acceso a nuestro repositorio, tanto para ver como para trabajar en el mismo. La elección se realiza desde la pantalla de creación del mismo repositorio, bajo la opción "Public".

¿Cómo compartir un repositorio público en GitHub?

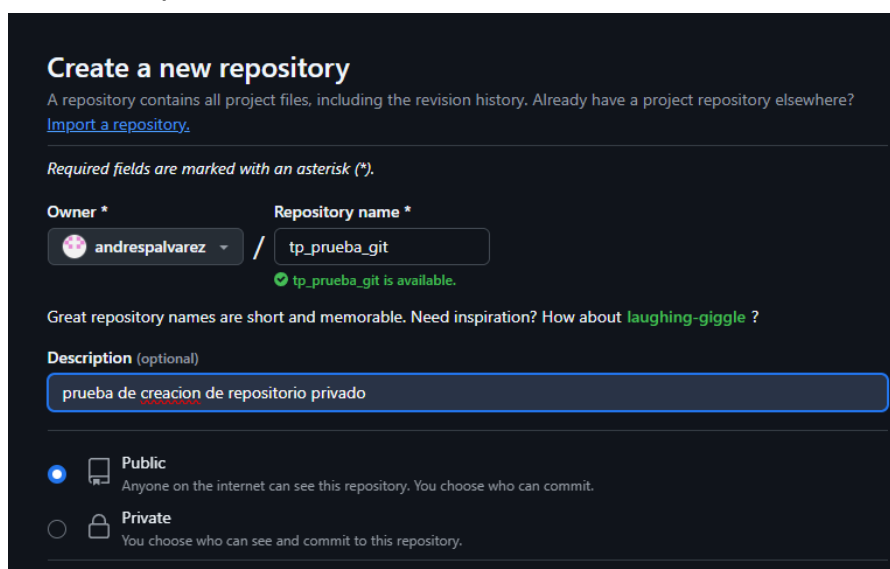
Un repositorio público, puede ser encontrado en buscadores externos o el de Github o directamente compartido a partir del link que genera la plataforma para ello.

2) Realizar la siguiente actividad:

- **Crear un repositorio.**

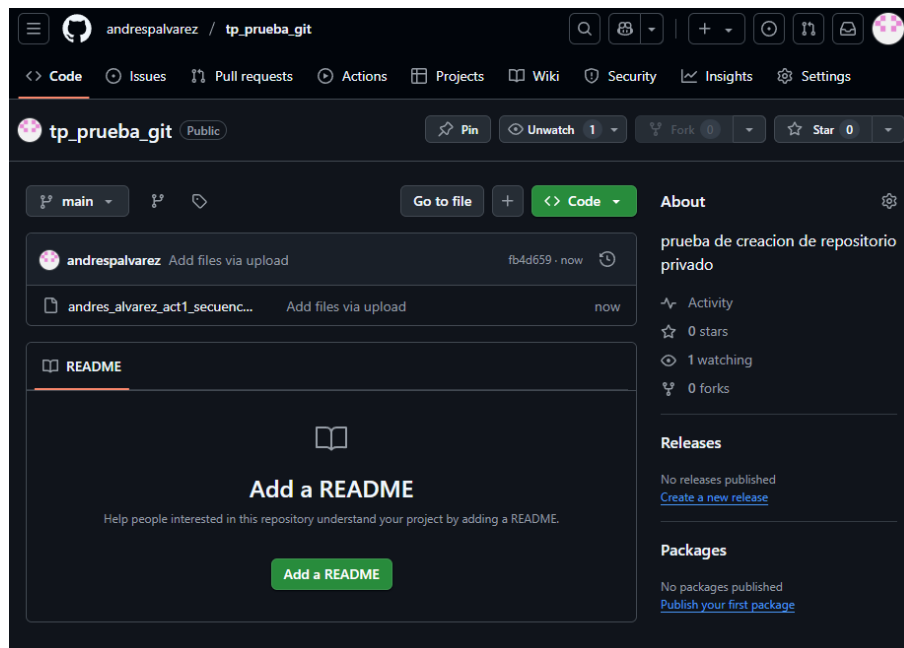
- o Dale un nombre al repositorio.

- o Elije el repositorio sea público.



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 section for 'Required fields are marked with an asterisk (*)'. The 'Owner' field is set to 'andrespalvarez' and the 'Repository name' field is 'tp_prueba_git'. A green checkmark indicates that 'tp_prueba_git' is available. Below the name fields, there's a suggestion for repository names: 'Great repository names are short and memorable. Need inspiration? How about laughing-giggle?'. The 'Description' field is optional and contains the text 'prueba de creacion de repositorio privado'. At the bottom, there are two radio buttons for visibility: 'Public' (selected) and 'Private'. The 'Public' option is described as 'Anyone on the internet can see this repository. You choose who can commit.' and the 'Private' option is described as 'You choose who can see and commit to this repository.'

- o Inicializa el repositorio con un archivo.



• 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.

```
andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/p1_tp2 (master)
$ git add .

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/p1_tp2 (master)
$ git commit -m "Agregando mi-archivo.txt"
[master (root-commit) 0d0ea0f] Agregando mi-archivo.txt
1 file changed, 1 insertion(+)
create mode 100644 mi-archivo.txt

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/p1_tp2 (master)
$
```

o Sube los cambios al repositorio en GitHub con `git push origin main` (o el nombre de la rama correspondiente).

```
andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/p1_tp2 (master)
$ git remote add origin https://github.com/andrespalvarez/tp_prueba_git.git

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/p1_tp2 (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 253 bytes | 253.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/andrespalvarez/tp_prueba_git/pull/new/master
remote:
To https://github.com/andrespalvarez/tp_prueba_git.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/p1_tp2 (master)
$
```

- **Creando Branchs**

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

```
andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/p1_tp2 (prueba_branch)
$ git commit -m "agregado de rama y archivo texto 2"
[prueba_branch 05ccc29] agregado de rama y archivo texto 2
1 file changed, 4 insertions(+)
 create mode 100644 mi-archivo2.txt

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/p1_tp2 (prueba_branch)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/p1_tp2 (master)
$ git checkout prueba_branch
Switched to branch 'prueba_branch'

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/p1_tp2 (prueba_branch)
$ git push -u origin master
Everything up-to-date
branch 'master' set up to track 'origin/master'.

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/p1_tp2 (prueba_branch)
$
```

COMO NO VI LOS CAMBIOS EN GITHUB TUVE QUE BUSCAR LA SOLUCION Y
ENCONTRE ESTO

```
MINGW64:/c/TUP/Q1/programacion1/p1_tp2

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/p1_tp2 (prueba_branch)
$ git add .

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/p1_tp2 (prueba_branch)
$ git commit -m "archivo 2 modificado"
[prueba_branch 98a90cc] archivo 2 modificado
1 file changed, 4 insertions(+)

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/p1_tp2 (prueba_branch)
$ git push origin head
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 572 bytes | 572.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
remote:
remote: Create a pull request for 'prueba_branch' on GitHub by visiting:
remote:   https://github.com/andrespalvarez/tp_prueba_git/pull/new/prueba_bra
remote:   nch
remote:
To https://github.com/andrespalvarez/tp_prueba_git.git
 * [new branch]      head -> prueba_branch

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/p1_tp2 (prueba_branch)
$ |
```

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 *

 andrespalvarez ▾

Repository name *

conflict-exercise

✔ conflict-exercise is available.

Great repository names are short and memorable. Need inspiration? How about **refactored-sniffle** ?

Description (optional)

ejercicio de conflicto de ramas



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.](#)

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

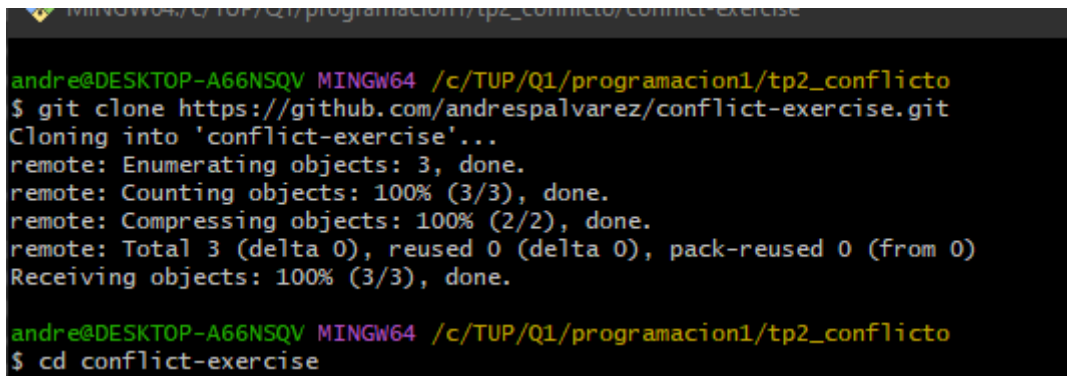
Paso 2: Clonar el repositorio a tu máquina local

- 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`



```
andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/tp2_conflicto/conflict-exercise
$ git clone https://github.com/andrespalvarez/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.

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/tp2_conflicto
$ 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`

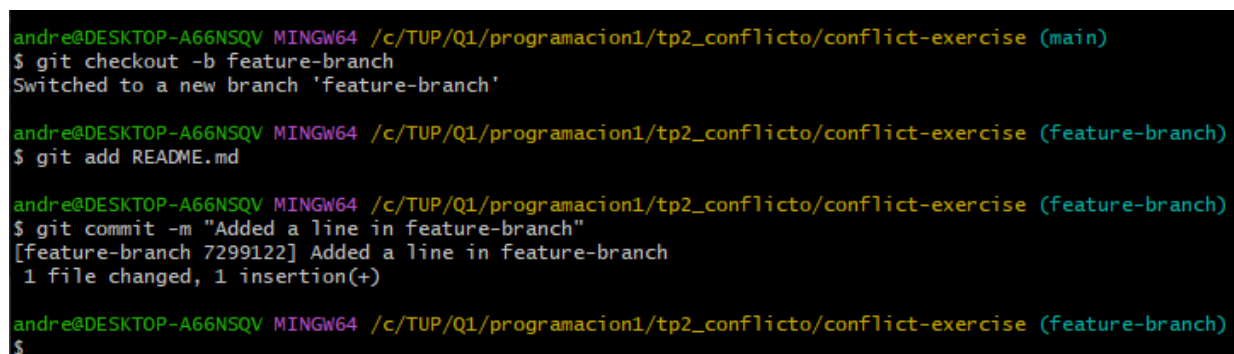
- Abre el archivo README.md en un editor de texto y añade una línea nueva, por ejemplo:

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"`



```
andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/tp2_conflicto/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/tp2_conflicto/conflict-exercise (feature-branch)
$ git add README.md

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/tp2_conflicto/conflict-exercise (feature-branch)
$ git commit -m "Added a line in feature-branch"
[feature-branch 7299122] Added a line in feature-branch
1 file changed, 1 insertion(+)

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/tp2_conflicto/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"`

```
andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/tp2_conflicto/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/tp2_conflicto/conflict-exercise (main)
$ git add README.md

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/tp2_conflicto/conflict-exercise (main)
$ git commit -m "Added a line in main branch"
[main 68657f6] Added a line in main branch
1 file changed, 1 insertion(+)

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/tp2_conflicto/conflict-exercise (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.

```
andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/tp2_conflicto/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.

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/tp2_conflicto/conflict-exercise (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"`

```
# conflict-exercise
ejercicio de conflicto de ramas
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
<<<<<< HEAD (Current Change)
Este es un cambio en la main branch
=====
Este es un cambio en la feature branch
>>>>>> feature-branch (Incoming Change)
```

```
andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/tp2_conflicto/conflict-exercise (main|MERGING)
$ git add README.md

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/tp2_conflicto/conflict-exercise (main|MERGING)
$ git commit -m "Resolved merge conflict"
[main 5d0b13e] Resolved merge conflict

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/tp2_conflicto/conflict-exercise (main)
$ |
```

Paso 7: Subir los cambios a GitHub

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

`git push origin main`

```
andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/tp2_conflicto/conflict-exercise (main)
$ git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 762 bytes | 762.00 KiB/s, done.
Total 9 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/andrespalvarez/conflict-exercise.git
  6a65e9d..5d0b13e  main -> main

andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/tp2_conflicto/conflict-exercise (main)
$
```

- También sube la feature-branch si deseas:

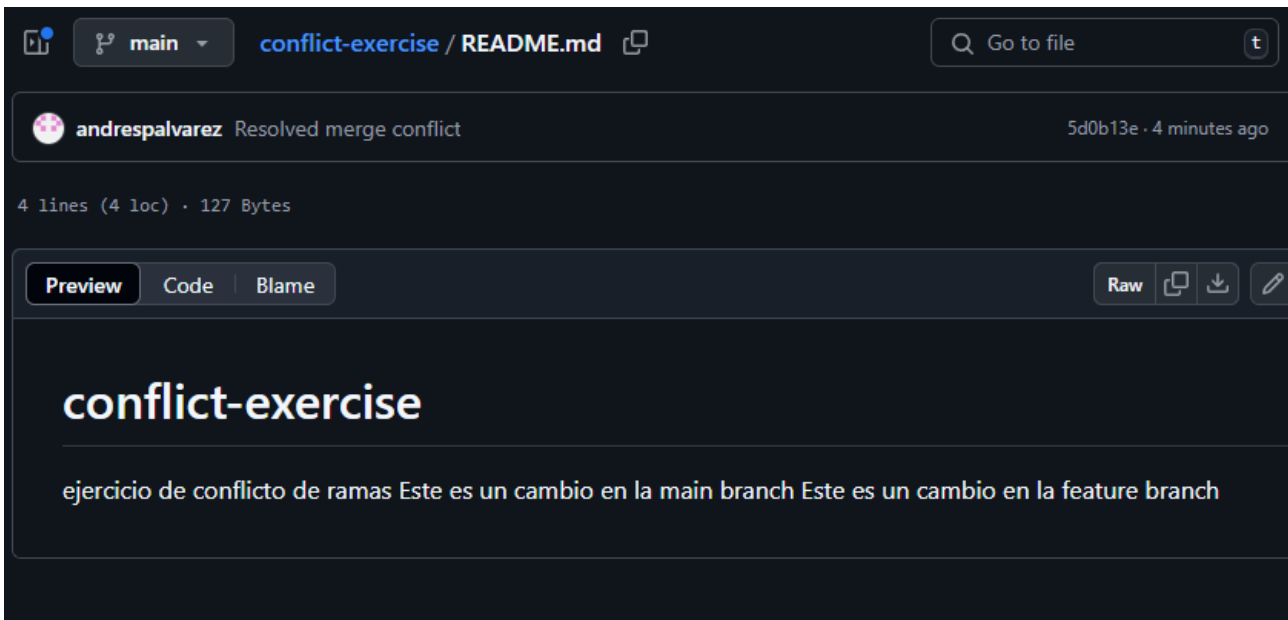
`git push origin feature-branch`

```
andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/tp2_conflicto/conflict-exercise (main)
$ git push origin feature-branch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:   https://github.com/andrespalvarez/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/andrespalvarez/conflict-exercise.git
 * [new branch]      feature-branch -> feature-branch

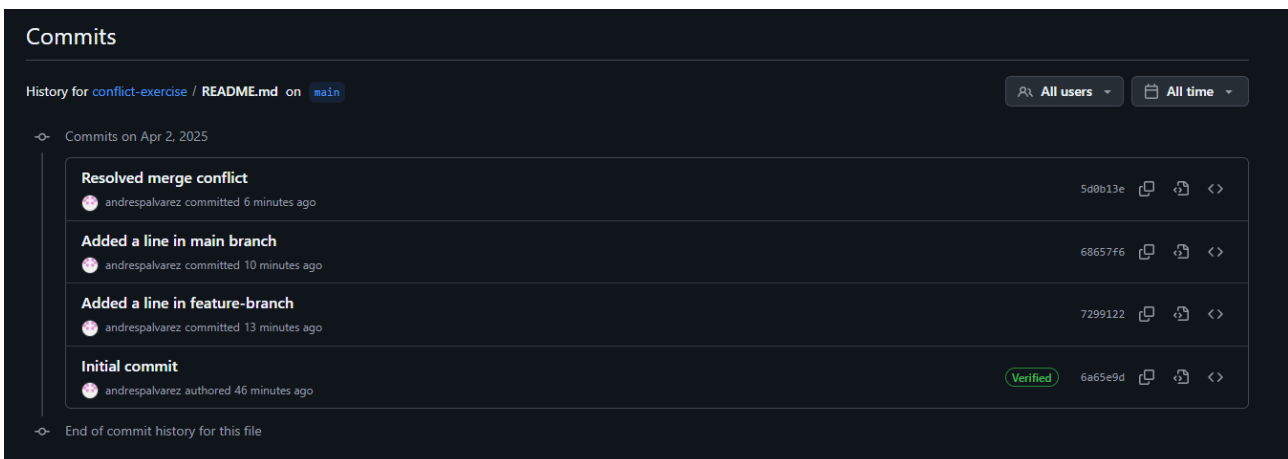
andre@DESKTOP-A66NSQV MINGW64 /c/TUP/Q1/programacion1/tp2_conflicto/conflict-exercise (main)
$
```

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



1 file changed +2 -1 lines changed

▼ README.md

...

@@ -1,3 +1,4 @@

1 1 # conflict-exercise

2 2 ejercicio de conflicto de ramas

3 - Este es un cambio en la main branch

3 + Este es un cambio en la main branch

4 + Este es un cambio en la feature branch

Comments 0

Lock conversation

Comment

Unsubscribe

You're receiving notifications because you're subscribed to this thread.