

1 ¿Qué es GitHub?

Es una plataforma que los desarrolladores utilizan para almacenar y además subir o controlar sus proyectos de software. Es de código abierto y gratuito, donde podemos trabajar en grupo de una manera más eficiente, intercambiar códigos, clonarlos o modificarlos.

2 ¿Cómo crear un repositorio en GitHub?

Primero creamos una cuenta desde la página web de GitHub para poder configurar el perfil y poder trabajar de una manera tranquila y explorar todas las funciones de la misma.

Buscamos en la parte superior izquierda el botón “New”, una vez dentro vamos a tener que darle un nombre a nuestro proyecto, agregar una descripción en caso de querer hacerlo, configurarlo de manera publica o privada, si queremos agregar el archivo README. Luego le damos a “Create Repository”. Luego nos va a mostrar unos comandos para poder inicializarlo y que cada cambio o avance que realicemos durante el trabajo se sincronice y nos permita tenerlo actualizado.

3 ¿Cómo crear una rama en Git?

La rama “master” en Git luego de crear el repositorio y configurarlo ya viene creada por defecto. Pero en cualquier momento podemos crear una nueva dentro de nuestro proyecto, pero deberemos usar el comando `$git branch “nombre de la rama”`

4 ¿Cómo cambiar a una rama en Git?

Para cambiar de una rama a otra debemos utilizar el comando `$git checkout` como por ejemplo: `$git checkout “nombre de la nueva rama”`.

5 ¿Cómo fusionar ramas en Git?

Para fusionar ramas en Git vamos a tener que realizar los siguientes pasos:

Primero tenemos que estar en la rama que queremos fusionar los cambios, donde generalmente estamos sobre la rama principal (main/master).

Segundo ya teniendo la segunda rama creada “nuevaRama” por ejemplo, vamos a necesitar realizar estos comandos

`$git checkout master` (para estar posicionados sobre la rama principal)

`$git merge “nuevaRama”` (utilizamos el comando `git merge` para fusionar ambas

ramas)

Y luego de eso, ya los cambios realizamos de manera separada van a unificarse.

6 ¿Cómo crear un commit en Git?

Para crear un commit en Git vamos a necesitar realizar un comando para que los cambios que realizamos en el o los archivos del proyecto puedan guardarse en el repositorio para mantenerlo actualizado, pero además lo guarda en el historial.

`$git add .` (para agregar todos los archivos) o `git add archivoA` (para agregar el archivo "archivoA")

`$git commit -m "Comentario de cambios realizados"`

7 ¿Cómo enviar un commit a GitHub?

Para crear un commit debemos tener cambios a realizar dentro del proyecto y que podamos subirlo al repositorio ya clonado en nuestra computadora de forma local.

Debemos utilizar los siguientes comandos:

`$git add .` (para agregar cambios)

`$git commit -m "Mensaje sobre los cambios realizados"`

`$git push origin NombreDeLaRama`

8. ¿Qué es un repositorio remoto?

Los repositorios remotos son versiones de nuestros proyectos que están ubicadas en Internet o en cualquier otra red, Podremos colaborar con otras personas enviando o trayendo cada vez que necesitemos compartir nuestro trabajo.

9. ¿Cómo agregar un repositorio remoto a Git?

Utiliza el comando `git remote add` para vincular tu repositorio local con un repositorio remoto y asignar un nombre a ese remoto:

```
$ git remote add (nombre / url)
```

```
$ git remote -v
```

 (Esto mostrará una lista de todos los remotos configurados con sus URLs)

Para traer toda la información del repositorio remoto que aún no tenemos en el repositorio local, usaremos el comando `$git fetch` seguido del nombre del remoto.

El cual descarga todos los cambios del repositorio remoto asociado con el nombre, pero no fusiona esos cambios con la rama actual.

10. ¿Cómo empujar cambios a un repositorio remoto?

Para empujar los cambios a un repositorio remoto vamos a utilizar el siguiente comando:

```
git push origin "NombreRama"
```

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

De la manera que mencionamos anteriormente usamos el comando `git pull` para descargar y fusionar los cambios del repositorio remoto con la rama local
Ejemplo: `$git pull origin "nombre de la rama"`

12. ¿Qué es un fork de repositorio?

Es una herramienta que nos ofrece GitHub para que nosotros podamos tener una copia y realizar modificaciones, utilizándolo también como base para nuestro trabajo a realizar.

Se puede buscar el botón "fork" que esta en la parte superior derecha de la página. Una vez utilizado, crea una copia del repositorio en el cual nos interesamos en nuestra cuenta de GitHub. Es una copia independiente a la del autor del proyecto.

13. ¿Cómo crear un fork de un repositorio?

Se puede buscar cualquier repositorio un proyecto, buscamos el botón fork lo apretamos y clonamos el proyecto que nos gustó y modificamos algún archivo que podamos mejorar para nosotros, luego lo agregamos y commiteamos, por último, mandamos con un git push origin main. Además, podemos hacer que esos cambios los vea el creador del repositorio o no.

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

Para hacer el pull request nos dirigiremos a la solapa de Pull requests allí daremos click en new pull request, vamos a ver ventana de resumen en donde se muestran los cambios que hemos realizado en comparación a el código original.

Donde veremos el asunto colocamos algún mensaje y más abajo tenemos para comunicar porque ese cambio que hemos realizado nosotros, que el autor lo vea y si le parece una buena modificación la realice.

15. ¿Cómo aceptar una solicitud de extracción?

El autor del repositorio puede ver en sus pull requests el mensaje que le enviamos, para que lo pueda ver y si le gusta realizar el cambio que le planteamos. Si el usuario original considera que esta modificación es buena, se clickea en Merge pull request y sumará a su repositorio los cambios que hicimos.

16. ¿Qué es una etiqueta en Git?

Una etiqueta puede mostrarnos ciertos puntos específicos del historial como lo más importante, como por ejemplo la versión que esta actualmente el programa ejemplo: v2.0

17. ¿Cómo crear una etiqueta en Git?

Git utiliza dos tipos principales de etiquetas: ligeras donde es temporal y anotadas se guardan en la base de datos de git como objetos enteros.

18. ¿Cómo enviar una etiqueta a GitHub?

Primero tenemos que crear una etiqueta en tu repositorio local. Puedes crear una etiqueta anotada (que incluye información adicional como autor y fecha) o una etiqueta ligera (simplemente un puntero a un commit):

Ejemplo de etiqueta ligera: `$git tag v1.0`

Una vez que has creado la etiqueta en tu repositorio local, necesitamos empujarla al repositorio remoto en GitHub con el siguiente comando:

`$git push origin v1.0`

19. ¿Qué es un historial de Git?

El historial de Git es una secuencia de todos los cambios que realizamos en un repositorio de Git. Cada cambio lo guardamos como un commit y cada commit contiene un mensaje que nos permita identificar el estado del proyecto hasta ese momento, incluyendo: Identificador del commit, Autor, Fecha de realización y Mensaje enviado.

20. ¿Cómo ver el historial de Git?

Esto lo conseguimos con el comando de Git:

`$git log`

Estando en la carpeta de nuestro proyecto, podemos colocar el comando anteriormente mostrado en el bash de Git y que nos muestre el historial de commits, lo vamos a encontrar invertido.

21. ¿Cómo buscar en el historial de Git?

Para buscar commits que tengan una palabra o frase en el mensaje de commit, usaremos: `$git log` con la opción `--grep`: `git log --grep="palabra clave"`

Si buscamos commits que han modificado un archivo específico, usaremos `git log` seguido del nombre del archivo: `$git log -- nombre_del_archivo`

Para buscar commits en un rango de fechas específico, usaremos las opciones `--since` y `--until`: `$git log --since="2024-01-01" --until="2024-01-31"`

Y por último para encontrar commits hechos por un autor específico, usaremos `--author`: `$git log --author="Nombre del Autor"`

22. ¿Cómo borrar el historial de Git?

El comando `git reset` se utiliza para deshacer las cosas.

A continuación, mostraremos distintas formas de utilizarlo:

\$git reset: Quita del stage todos los archivos y carpetas del proyecto.

\$git reset NombreArchivo: Elimina del stage el archivo indicado.

\$git reset NombreCarpeta/: Elimina del stage todos los archivos de esa carpeta.

\$git reset NombreCarpeta/NombreArchivo: Elimina ese archivo del stage (está dentro de una carpeta).

\$git reset nombreCarpeta/*.extensión: Quita todos los archivos que cumplan con la condición indicada previamente dentro de esa carpeta del stage.

23. ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es un tipo de repositorio en donde el contenido solo es accesible para usuarios específicos que han sido autorizados. Es de gran ayuda para proyectos que contienen información sensible o que aún están en desarrollo y no deseamos que estén disponibles públicamente.

24. ¿Cómo crear un repositorio privado en GitHub?

Para crear un repositorio privado tendremos que seguir ciertos pasos:

1. Inicia sesión en GitHub
2. Ingresa a la página de creación de repositorios:
3. En la esquina superior derecha de la página principal, debes hacer clic en el botón “+” y seleccionar “New
4. Repository” o hacer clic en “New”:
5. Completamos la información del repositorio (nombre del repositorio, descripción)
6. Seleccionamos la configuración de privacidad en “Private”. Esto asegura que el repositorio será privado y solo accesible para los colaboradores que tu elijas.

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

Para que podamos invitar a alguien a un repositorio privado en GitHub debemos concebir ciertos permisos:

1. Accedemos al repositorio, hacemos clic en la pestaña "Settings" del repositorio. Está en la parte superior del repositorio, junto a las pestañas como "Code" y "Security".

2. Selecciona "Collaborators" en el menú de la izquierda. Aquí podemos administrarlos.

3. En la sección "Collaborators", hacemos clic en el botón "Add people" e ingresamos el nombre de usuario de GitHub de la persona que deseas invitar.

4. Seleccionamos el nivel de acceso que deseas otorgar: Read, Triage, Write, Maintain, o Admin. Por último, hacemos clic en el botón "Add" para enviar la invitación.

26. ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un repositorio el cual el contenido es accesible a cualquier persona en Internet. Un repositorio público permite que cualquier persona pueda ver, clonar y además si tienen los permisos adecuados, contribuir al proyecto modificándolo de una mejor manera.

27. ¿Cómo crear un repositorio público en GitHub?

Para crear un repositorio público en GitHub seguimos los siguientes pasos:

1. Iniciamos sesión en GitHub

2. Ingresamos a la página de creación de repositorios:

3. En la esquina superior derecha de la página principal, hacemos clic en el botón "+" y seleccionamos "New

4. Repository" o hacer clic en "New":

5. Completamos la información del repositorio (nombre del repositorio, descripción)

6. Seleccionamos la configuración de privacidad en "Public": Esto asegura que el repositorio sea público.

28. ¿Cómo compartir un repositorio público en GitHub?

La manera más fácil de compartir tu repositorio es accediendo a tu repositorio, copia la URL de tu repositorio que la encontramos en un cuadro de texto que dice "<> Code":

Podemos copiar la URL directamente haciendo clic en el botón de copiar a la derecha de la URL.

Actividad2.

1. Crear un repositorio:

Dale un nombre al repositorio.

Elije el repositorio sea público.

Inicializa el repositorio con un archivo.

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.

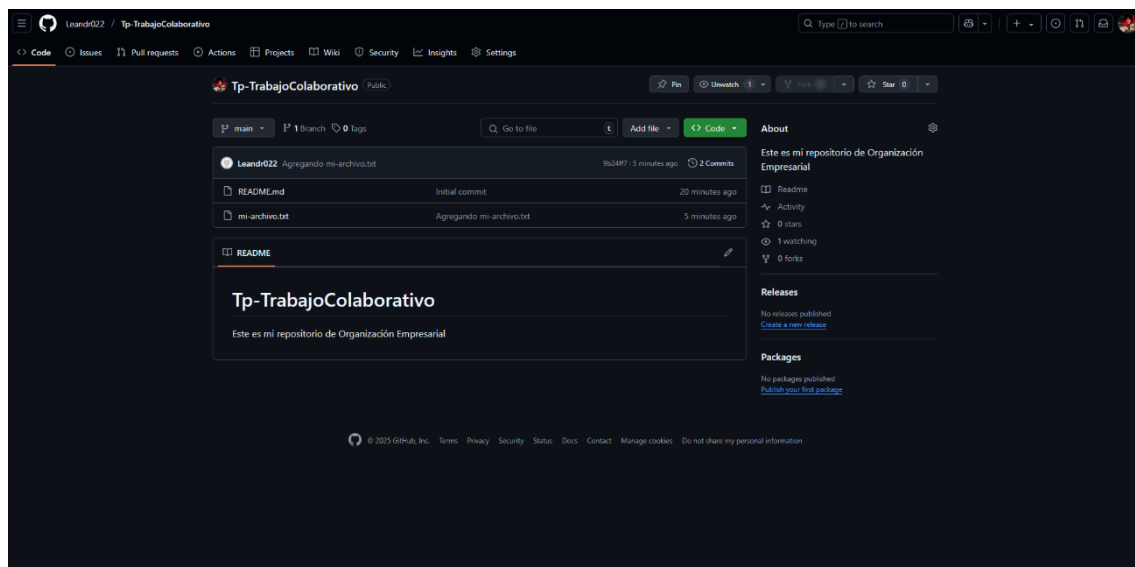
Sube los cambios al repositorio en GitHub con `git push origin main` (o el nombre de la rama correspondiente).

3. Creando Branchs:

Crear una Branch

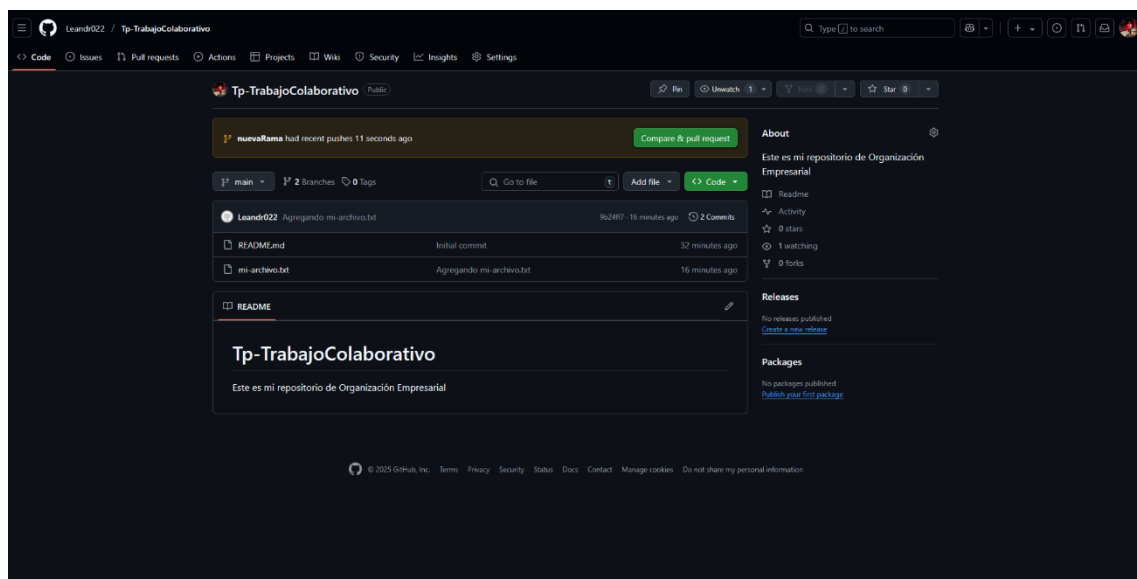
Realizar cambios o agregar un archivo

Subir la Branch

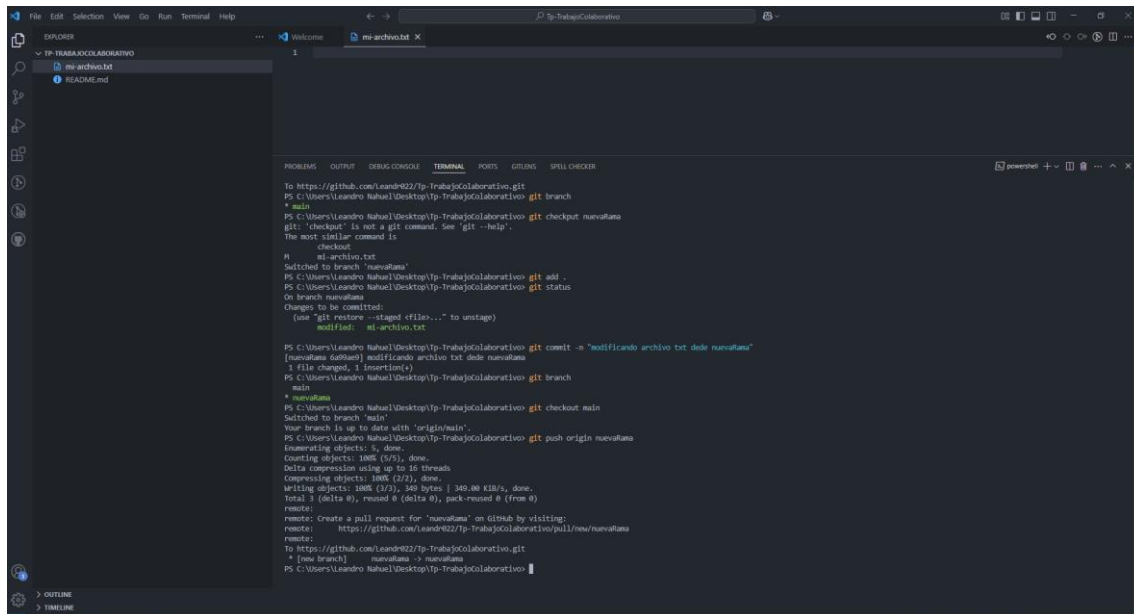



```
PS C:\Users\Leandro.Nahuel\Desktop\Tp-TrabajoColaborativo> git add .
PS C:\Users\Leandro.Nahuel\Desktop\Tp-TrabajoColaborativo> git commit -m "Agregando mi-archivo.txt"
[main 9b24ff7] Agregando mi-archivo.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 mi-archivo.txt
PS C:\Users\Leandro.Nahuel\Desktop\Tp-TrabajoColaborativo> git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (1/1), 204 bytes | 204.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Leandro022/Tp-TrabajoColaborativo.git
9b24ff7..9b24ff7 main -> main
PS C:\Users\Leandro.Nahuel\Desktop\Tp-TrabajoColaborativo>
```

Nueva rama en GitHub

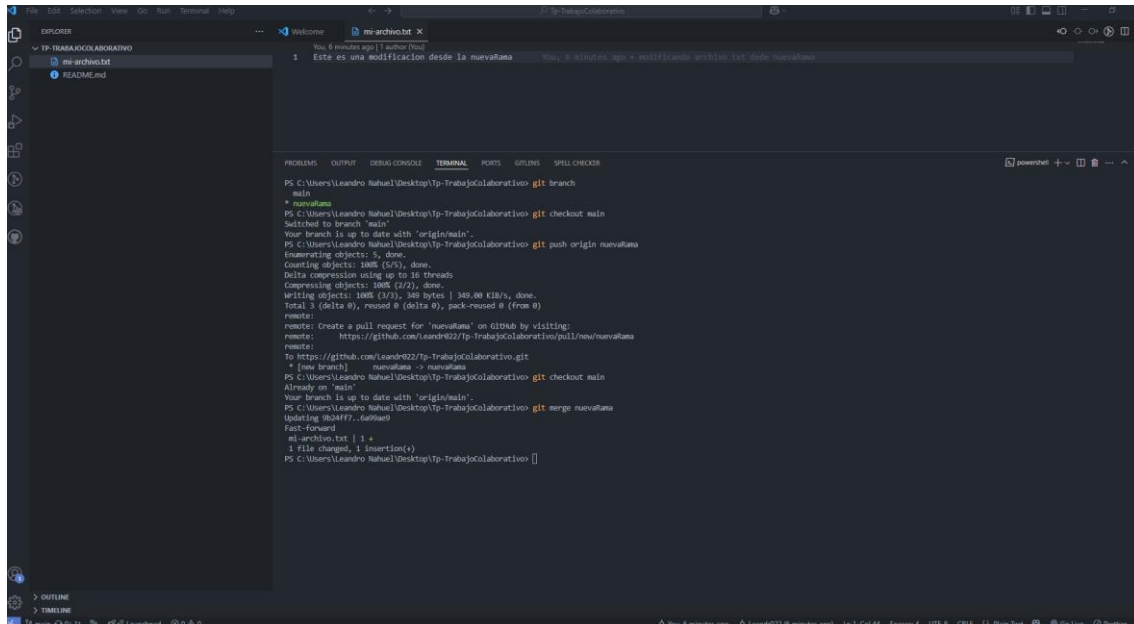


Comandos de nueva rama y modificaciones en mi-archivo.txt



```
PS C:\Users\Leandro.Nahuel\Desktop\Tp-TrabajoColaborativo> git branch
* main
PS C:\Users\Leandro.Nahuel\Desktop\Tp-TrabajoColaborativo> git checkout nuevaRama
git: 'checkout' is not a git command. See 'git --help'.
The most similar command is
    checkout
M       nl-archivo.txt
Switched to branch 'nuevaRama'
PS C:\Users\Leandro.Nahuel\Desktop\Tp-TrabajoColaborativo> git add .
PS C:\Users\Leandro.Nahuel\Desktop\Tp-TrabajoColaborativo> git status
On branch nuevaRama
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   nl-archivo.txt
PS C:\Users\Leandro.Nahuel\Desktop\Tp-TrabajoColaborativo> git commit -m "modificando archivo txt desde nuevaRama"
[nuevaRama 66bba0] modificando archivo txt desde nuevaRama
1 file changed, 1 insertion(+)
PS C:\Users\Leandro.Nahuel\Desktop\Tp-TrabajoColaborativo> git branch
* nuevaRama
main
PS C:\Users\Leandro.Nahuel\Desktop\Tp-TrabajoColaborativo> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\Leandro.Nahuel\Desktop\Tp-TrabajoColaborativo> git push origin nuevaRama
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 349 bytes | 349.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Create a pull request for 'nuevaRama' on GitHub by visiting:
remote:   https://github.com/Leandro022/Tp-TrabajoColaborativo/pull/new/nuevaRama
remote:
To https://github.com/Leandro022/Tp-TrabajoColaborativo.git
 * [new branch]   nuevaRama -> nuevaRama
PS C:\Users\Leandro.Nahuel\Desktop\Tp-TrabajoColaborativo>
```

Unificamos los cambios con el comando \$git merge nuevaRama para que aparezcan los cambios en la rama main



```
PS C:\Users\Leandro.Nahuel\Desktop\Tp-TrabajoColaborativo> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\Leandro.Nahuel\Desktop\Tp-TrabajoColaborativo> git push origin nuevaRama
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 349 bytes | 349.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Create a pull request for 'nuevaRama' on GitHub by visiting:
remote:   https://github.com/Leandro022/Tp-TrabajoColaborativo/pull/new/nuevaRama
remote:
To https://github.com/Leandro022/Tp-TrabajoColaborativo.git
 * [new branch]   nuevaRama -> nuevaRama
PS C:\Users\Leandro.Nahuel\Desktop\Tp-TrabajoColaborativo> git checkout main
Already on 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\Leandro.Nahuel\Desktop\Tp-TrabajoColaborativo> git merge nuevaRama
Updating 604ff7...66bba0
Fast-forward
 nl-archivo.txt | 1 +
 1 file changed, 1 insertion(+)
PS C:\Users\Leandro.Nahuel\Desktop\Tp-TrabajoColaborativo>
```

Actividad 3.

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 * Leandri022 / Repository name * conflict-exercise

[conflict-exercise](#) is available.

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

Description (optional)
Probaremos los conflictos en Organización Empresarial

☒ 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](#).

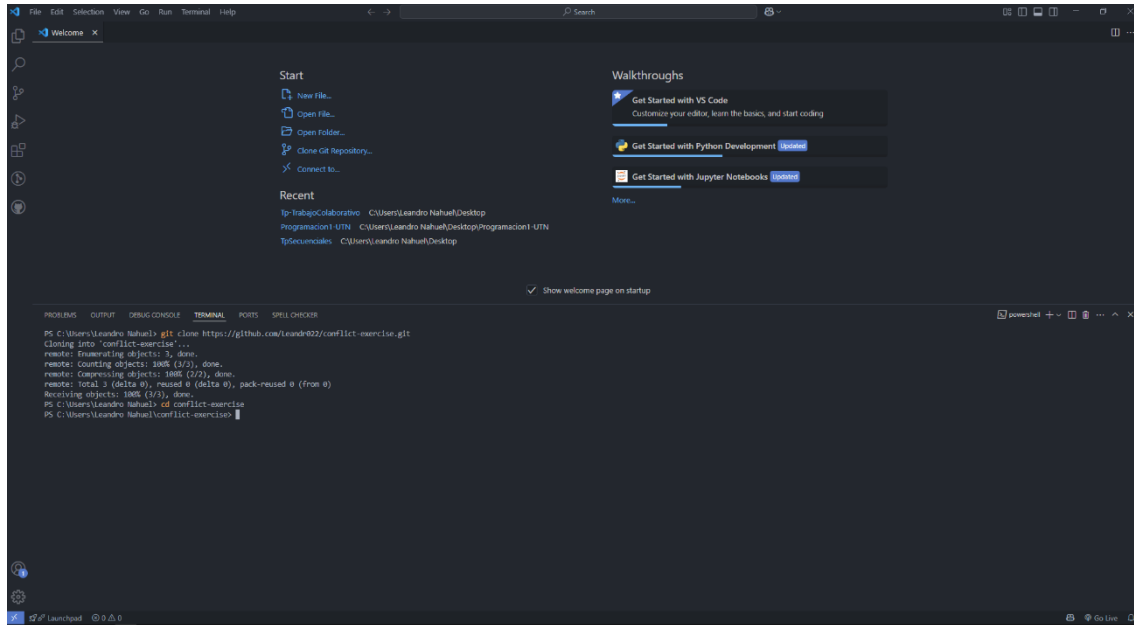
ⓘ You are creating a public repository in your personal account.

[Create repository](#)

Paso 2: Clonar el repositorio a tu máquina local

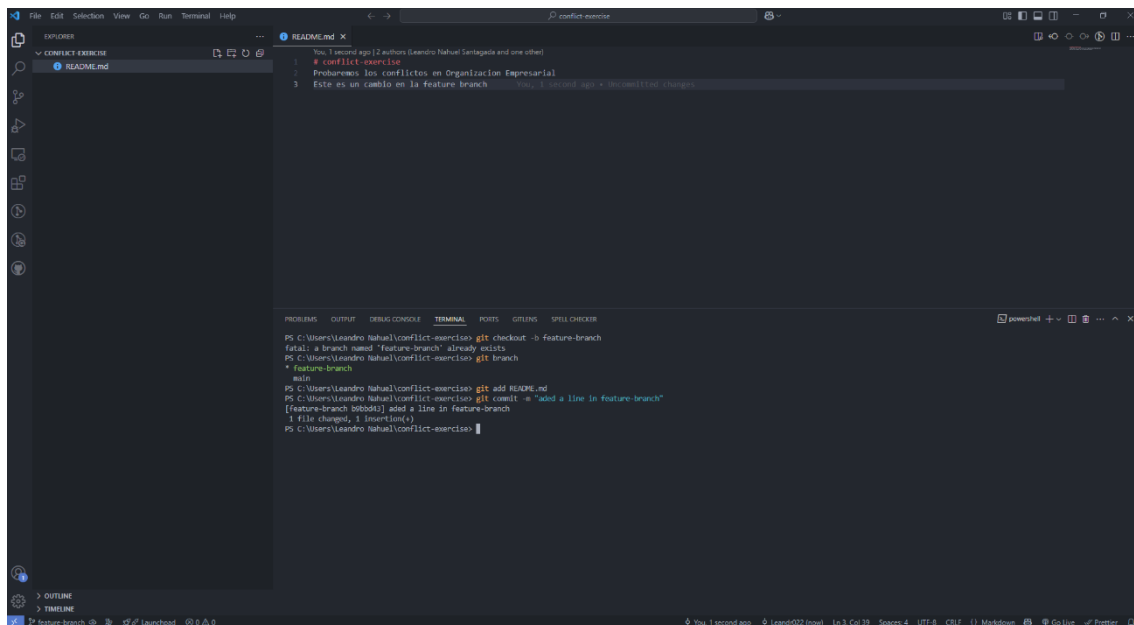
- Copia la URL del repositorio (usualmente algo como <https://github.com/tuusuario/conflict-exercise.git>).
- Abre la terminal o línea de comandos en tu máquina.
- Clona el repositorio usando el comando:

`git clone https://github.com/tuusuario/conflict-exercise.git`



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.



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.

The screenshot shows the VS Code interface. The Explorer pane on the left shows the file structure with 'CONFLICT-EXERCISE' and 'README.md'. The editor shows the content of README.md, which includes a header, a list of authors, and a list of tasks. The terminal window at the bottom shows the following commands and output:

```
PS C:\Users\Leandro Nahuel\Desktop\conflict-exercise> git branch
* main
PS C:\Users\Leandro Nahuel\Desktop\conflict-exercise> git checkout -b feature-branch
Switched to a new branch 'feature-branch'
PS C:\Users\Leandro Nahuel\Desktop\conflict-exercise> git branch
* feature-branch
main
PS C:\Users\Leandro Nahuel\Desktop\conflict-exercise> git add README.md
PS C:\Users\Leandro Nahuel\Desktop\conflict-exercise> git commit -m "added a line in feature-branch"
[feature-branch 1f75a5] added a line in feature-branch
1 file changed, 1 insertion(+)
PS C:\Users\Leandro Nahuel\Desktop\conflict-exercise> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\Leandro Nahuel\Desktop\conflict-exercise> git add README.md
PS C:\Users\Leandro Nahuel\Desktop\conflict-exercise> git commit -m "added a line in main branch"
[main 7d8b78] added a line in main branch
1 file changed, 1 insertion(+)
PS C:\Users\Leandro Nahuel\Desktop\conflict-exercise>
```

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.

The screenshot shows the VS Code interface with a merge conflict in the README.md file. The conflict is highlighted with green and blue bars. The terminal window shows the following commands and output:

```
PS C:\Users\Leandro Nahuel\Desktop\conflict-exercise> git branch
* main
PS C:\Users\Leandro Nahuel\Desktop\conflict-exercise> git checkout -b feature-branch
Switched to a new branch 'feature-branch'
PS C:\Users\Leandro Nahuel\Desktop\conflict-exercise> git branch
* feature-branch
main
PS C:\Users\Leandro Nahuel\Desktop\conflict-exercise> git add README.md
PS C:\Users\Leandro Nahuel\Desktop\conflict-exercise> git commit -m "added a line in feature-branch"
[feature-branch 1f75a5] added a line in feature-branch
1 file changed, 1 insertion(+)
PS C:\Users\Leandro Nahuel\Desktop\conflict-exercise> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\Leandro Nahuel\Desktop\conflict-exercise> git add README.md
PS C:\Users\Leandro Nahuel\Desktop\conflict-exercise> git commit -m "added a line in main branch"
[main 7d8b78] added a line in main branch
1 file changed, 1 insertion(+)
PS C:\Users\Leandro Nahuel\Desktop\conflict-exercise> git merge feature-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
PS C:\Users\Leandro Nahuel\Desktop\conflict-exercise>
```

Paso 6: Resolver el conflicto

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

<<<<<<< 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.
- Añade el archivo resuelto y completa el merge:

git add README.md

git commit -m "Resolved merge conflict"

The screenshot shows a VS Code editor with a file named README.md open. The editor displays a merge conflict with the following content:

```

You, 1 minutes ago | authors (you and one other)
# conflict-exercise
1. Probaremos los conflictos en Organización Empresarial
2. Este es un cambio en la main branch
3. Este es un cambio en la feature branch
5 |
You, 3 minutes ago | uncommitted changes

```

The terminal at the bottom shows the following commands and output:

```

PS C:\Users\Leandro\Hual\Desktop\conflict-exercise> git branch
* main
PS C:\Users\Leandro\Hual\Desktop\conflict-exercise> git checkout -b feature-branch
Switched to a new branch 'feature-branch'
PS C:\Users\Leandro\Hual\Desktop\conflict-exercise> git branch
* feature-branch
main
PS C:\Users\Leandro\Hual\Desktop\conflict-exercise> git add README.md
PS C:\Users\Leandro\Hual\Desktop\conflict-exercise> git commit -m "added a line in feature-branch"
[feature-branch 3975a6] added a line in feature-branch
1 file changed, 1 insertion(+)
PS C:\Users\Leandro\Hual\Desktop\conflict-exercise> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\Leandro\Hual\Desktop\conflict-exercise> git add README.md
PS C:\Users\Leandro\Hual\Desktop\conflict-exercise> git commit -m "added a line in main branch"
[main 7d802a] added a line in main branch
1 file changed, 1 insertion(+)
PS C:\Users\Leandro\Hual\Desktop\conflict-exercise> git merge feature-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
PS C:\Users\Leandro\Hual\Desktop\conflict-exercise> git add README.md
PS C:\Users\Leandro\Hual\Desktop\conflict-exercise> git commit -m "resolved merge conflict"
[main 754ed6] resolved merge conflict
PS C:\Users\Leandro\Hual\Desktop\conflict-exercise>

```

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`

```

You 3 minutes ago | 2 authors (you and one other)
# conflict-exercise
Probaremos los conflictos en Organización Empresarial
Este es un cambio en la main branch
Este es un cambio en la feature branch
-----
You 3 minutes ago | 2 authors (you and one other)

[main 7c857f] added a line in main branch
1 file changed, 1 insertion(+)
PS C:\Users\leandro\huah\Desktop\conflict-exercise> git merge feature-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
PS C:\Users\leandro\huah\Desktop\conflict-exercise> git add README.md
PS C:\Users\leandro\huah\Desktop\conflict-exercise> git commit -m "resolved merge conflict"
[main 7554eb6] resolved merge conflict
PS C:\Users\leandro\huah\Desktop\conflict-exercise> git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 306 bytes | 306.00 KiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/leandro02/conflict-exercise.git
  efefc81..7554eb6 main -> main
PS C:\Users\leandro\huah\Desktop\conflict-exercise> git push origin feature-branch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:   https://github.com/leandro02/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/leandro02/conflict-exercise.git
  [new branch] feature-branch -> feature-branch
PS C:\Users\leandro\huah\Desktop\conflict-exercise>
  
```

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.

```

conflict-exercise

Probaremos los conflictos en Organización Empresarial <<<<<< HEAD Este es un cambio en la main branch. Este es un cambio en la feature branch

===== Este es un cambio en la feature branch.
| | | | | feature-branch
  
```