

# Práctico 2: Git y GitHub

## 1) Respuestas

### a) ¿Qué es GitHub?}

Es una plataforma similar a una red social que te permite alojar repositorios y poder tenerlos de manera pública o privada para que otros usuarios lo vean o no .

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

Crearse una cuenta

Simplemente en el lado superior y derecho de la pantalla hay un botón con el signo más (+) y entrar a nuevo repositorio. Se le pone un nombre al repositorio (seria bueno que sea el mismo nombre del local ) .

### c) ¿Cómo crear una rama en Git?

Se ejecuta Git como consola (Git Bash), y se ingresa el comando “ git branch <nombre>”, de este modo le ponemos el nombre que queramos a la nueva ram y la creamos. Por sí solo el comando “git branch” nos muestra todas las ramas que tengamos.

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

- i) Para cambiar de rama se abre la terminal en el editor de texto, y se utiliza el comando git checkout <nombre>, con el correspondiente nombre de la rama en la que te quieres posicionar.

### e) ¿Cómo fusionar ramas en Git?

- i) Para fusionar ramas primero hay que saber dónde estamos parados, con el comando “git branch” me muestra las ramas y con un asterisco donde estamos situados. Si quieres cambiar de rama comando “checkout <nombre\_rama>”. Si ya estamos en la rama que queremos que reciba los cambios colocamos el comando “git merge <nombre>”, con el nombre de la otra rama.

### f) ¿Cómo crear un commit en Git?

Tener definido el usuario y mail para que git sepa quien modifica de manera local.

Entramos a Git Bash (modo consola) y escribimos el comando ‘ git commit ‘ y si queremos agregar un comentario a ese commit ‘git commit -m ”mensaje”’.

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

- i) Primero hay que tener algo de código en nuestro repositorio local, luego abro la terminal en git bash o en el editor de código. Colocamos “git add .” (significa todos los archivos incluidos los que estén fuera de la carpeta se va a registrar su cambio), de no querer eso se coloca “git add file 1 file2 folder1/folder2” dependiendo que queramos o no registrar.

Luego se hace el commit, con su respectivo comentario  
Y con el comando “git push” va a subir el código a la nube en nuestro caso a GitHub (repositorio remoto).

h) ¿Qué es un repositorio remoto?

Un repositorio remoto es lo mismo que uno local, sería la copia del proyecto que tenemos en el repositorio local. Solo que con este se puede interactuar desde cualquier otro dispositivo sin la necesidad de tener el repositorio local en el mismo. Y además también permite el trabajo colaborativo o interacciones remotas, teniendo en cuenta que para estas últimas fusiones debe ser público el repositorio.

i) ¿Cómo agregar un repositorio remoto a Git?

- i) Primero debemos tener ya un repositorio remoto, en el propio GitHub en nuestro caso, hay una sección con comandos.

**...or push an existing repository from the command line**

```
git remote add origin https://github.com/LucasZzarate/repo-tp.git
git branch -M main
git push -u origin main
```

Ahora debemos crear un repositorio local, con el mismo nombre en lo posible. Y abrimos git bash (git modo consola), y copiamos la primera línea de instrucción, que lo que hace es agregar nuestra dirección de repositorio remoto.

Ejecutando la siguiente línea que nos va a cambiar de rama a la principal.

Ahora deberíamos ejecutar la 3er línea de instrucción y listo.

j) ¿Cómo empujar cambios a un repositorio remoto?

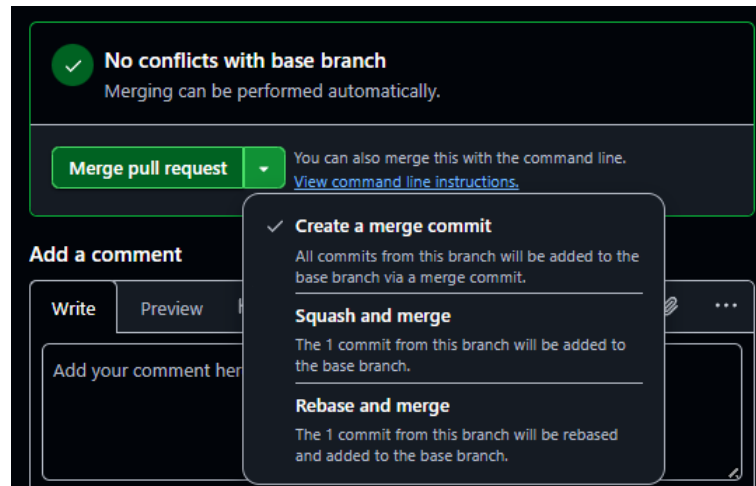
- i) Primero abrimos la terminal colocamos “git add .” o la carpeta o archivo que deseamos que se carguen los cambios, agregamos un commit con “git commit -m “comentario de lo que se hizo”” y luego hacemos un push, con el comando “git push -u origin master”.

k) ¿Cómo tirar de cambios de un repositorio remoto?

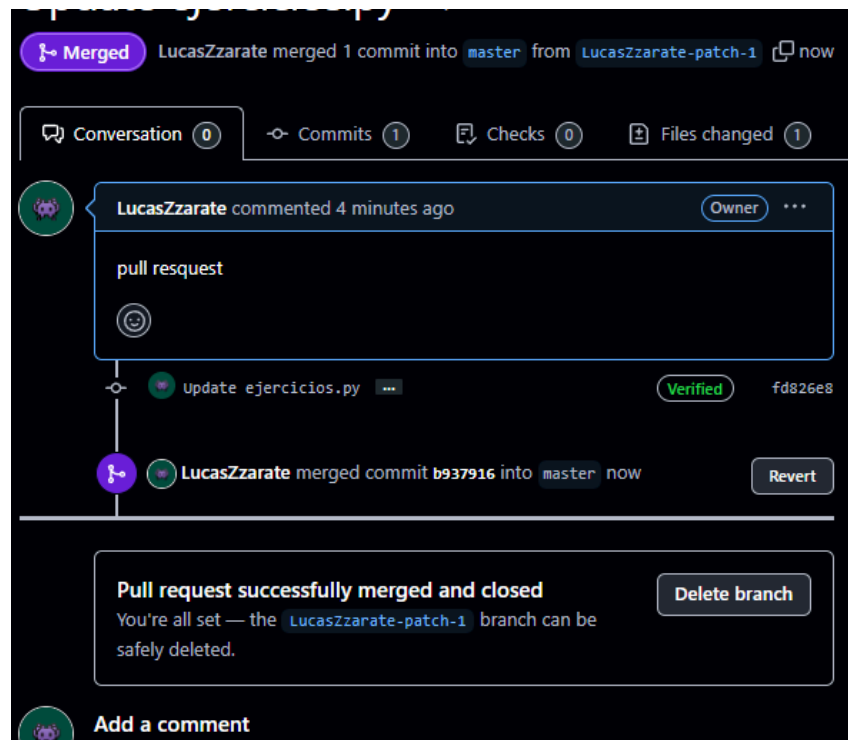
Primero nos hallamos con cambios en nuestro repositorio o proyecto remoto, y queremos pasarlos o copiarlos a nuestro repositorio local.

Entonces después de agregar nuestros cambios en GitHub nos sugiere con un botón verde hacer un pull request entonces, hacemos click y se abre lo siguiente.





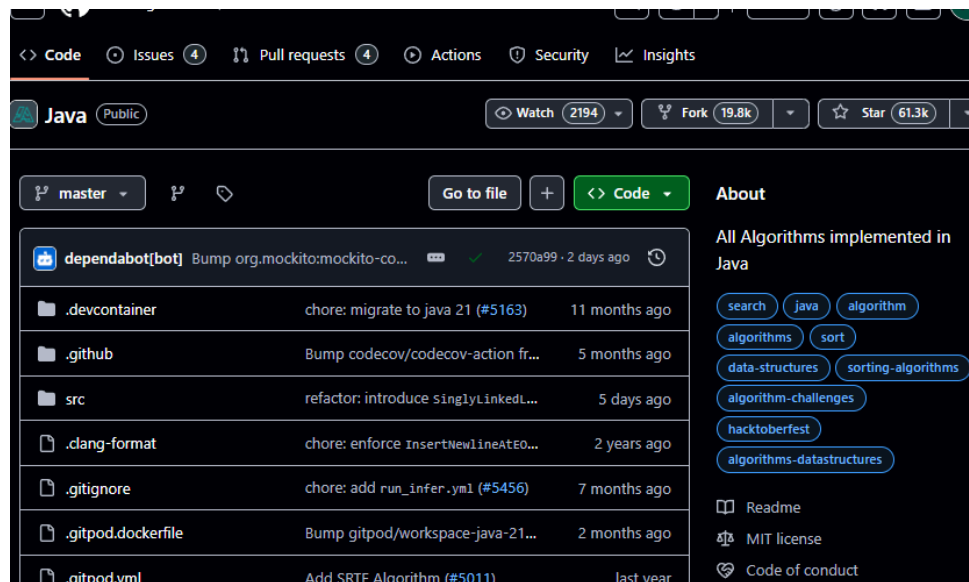
Una vez confirmado se pone de color violeta.  
Y ahora para que se sincróni



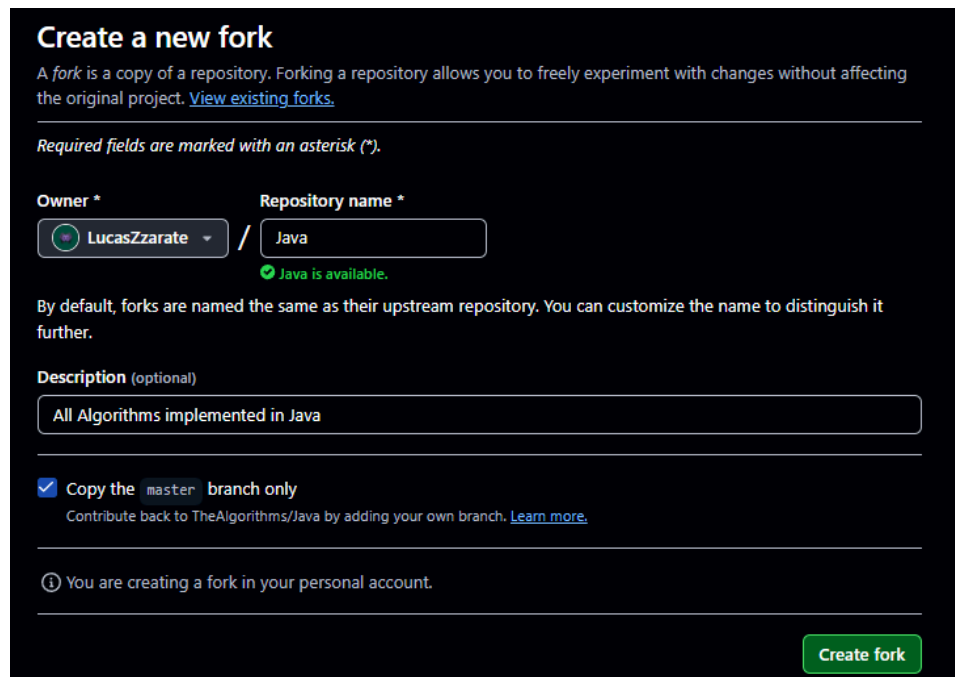
## TERMINAR DE DESARROLLAR

- l) ¿Qué es un fork de repositorio?
  - i) Un fork es una copia de un repositorio ajeno al que hace el fork, se sepe hacer para poder trabajar en el y despues hacer una solicitud de cambios a ese proyecto copiado, se hace para evitar que puedan cambiar el proyecto original al que solo tienen acceso los desarrolladores de dicho proyecto y el administrador.
- m) ¿Cómo crear un fork de un repositorio?

- i) Voy al proyecto de interés en GitHub, por ejemplo un repositorio en donde hay ejercicios de mi interés.



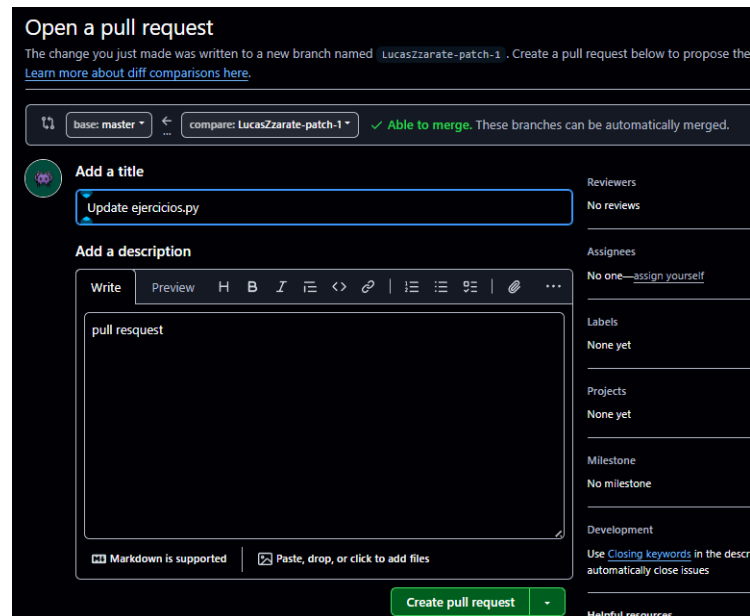
Hago click en Fork (al lado nos muestra el número de personas que han hecho un fork a dicho repositorio).



Elijo el perfil en el que quiero hacer el fork con el nombre que quiero ponerle. Y listo finalizó con Create fork..

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

Pull request es una solicitud de cambio, luego de hacer el fork y haber hecho cambios en el proyecto, para hacer la petición hay que entrar desde GitHub a create un pull request



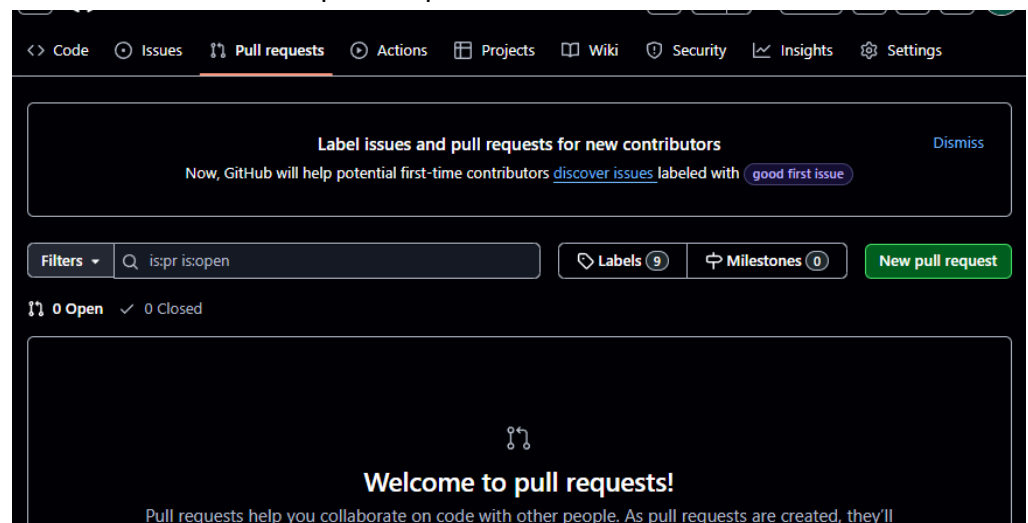
The screenshot shows the 'Open a pull request' interface on GitHub. At the top, it says 'Open a pull request' and 'The change you just made was written to a new branch named LucasZarate-patch-1. Create a pull request below to propose the change.' Below this, there are two dropdown menus: 'base: master' and 'compare: LucasZarate-patch-1'. A green checkmark indicates 'Able to merge. These branches can be automatically merged.' The main section is 'Add a title' with a text input field containing 'Update ejercicios.py'. Below that is 'Add a description' with a rich text editor containing 'pull request'. On the right side, there are sections for 'Reviewers' (No reviews), 'Assignees' (No one—assign yourself), 'Labels' (None yet), 'Projects' (None yet), 'Milestone' (No milestone), and 'Development' (Use Closing keywords in the description to automatically close issues). At the bottom right, there is a 'Helpful resources' link. A green 'Create pull request' button is at the bottom center.

Dentro de este va a haber que completar lo siguiente, con todos los cambios que sugieres y un contexto.

Se puede agregar a compañeros de trabajo en la barra lateral, etiquetas o documentos.

Y al final se crea esta solicitud.

- o) ¿Cómo aceptar una solicitud de extracción?
  - i) Para aceptar o rechazar una extracción o pull request, solo hay que entrar en la sección de pull request



En mi caso no hay sugerencia pero si de haberlas aquí se encontraron.

- p) ¿Qué es una etiqueta en Git?

Una etiqueta o tag en git es una referencia que apunta a un punto específico en el historial de commits, un ejemplo puede ser que hayamos creado una etiqueta para una serie de commits de una versión de 1.0 por ejemplo, con esto me facilita la búsqueda y la identificación de los commits que pertenecen a este tag. Se utilizan para marcar versiones importantes o puntos específicos en la historia de un proyecto.

q) ¿Cómo crear una etiqueta en Git?

i) Existen 2 tipos de tags: uno 'liviano' y otro anotado.

Tag "liviano":

Para este tipo de tag debemos utilizar el comando "git tag <nombre del tag>", y si lo queremos utilizar para ir agrupando commits se utiliza el comando "git tag <nombre\_commit>".

```
zarat@Lucasnotebook MINGW64 ~/OneDrive/Escritorio/primer-repo-git (master)
$ git tag v1.0

zarat@Lucasnotebook MINGW64 ~/OneDrive/Escritorio/primer-repo-git (master)
$ git tag
v1.0
```

En la primera línea creamos un tag con el nombre de v1.0.

Y en la segunda línea corroboramos que se haya cargado nuestra etiqueta.

Tag anotados, a diferencia del primero contienen más información (quien creó el tag, la fecha, la hora y un mensaje) y se realiza con el siguiente comando "git tag -a <nombre\_tag>" después de ejecutarlo va a pedir un mensaje.

Para ver que hay en cada tag se debe ingresar el comando "git show <nombre\_tag>" y si deseamos ver todos los tags "git tag --list"

r) ¿Cómo enviar una etiqueta a GitHub?

Luego de haber creado la etiqueta se debe hacer un push para enviar dichos cambios al repositorio remoto.

En la terminal ingresamos el siguiente comando "git push origin <nombre\_etiqueta>".

s) ¿Qué es un historial de Git?

El historial de git es un registro preciso de todas las confirmaciones, que contienen los cambios en los archivos. Puede rastrear que cambios y cuando o comparar la versión actual con la anterior.

t) ¿Cómo ver el historial de Git?

Se puede ver a través de ejecutar Git como consola (Git Bash) dentro del repositorio de interés y ejecutar el comando "git log".

u) ¿Cómo buscar en el historial de Git?

Se puede navegar de diferentes formas:

buscando fechas puntuales de confirmación del commit con “git log %cd <fecha\_de\_confirmacion>”, también con franjas de tiempo “git log --since=2.weeks” o “git log --until=2008-01-15” este comando acepta fechas puntuales y franjas en el tiempo.

También se puede buscar por el autor con el comando “tag log --author”.

Navegar entre los mensajes de confirmación con el comando “git log --grep” permite buscar palabras claves.

Y si queremos aplicar varias de estos filtros de búsqueda y que coincida con todos debemos poner a lo último --all-match

v) ¿Cómo borrar el historial de Git?

w) ¿Qué es un repositorio privado en GitHub?

La diferencia con el público es que el acceso que se tiene a él es limitado al igual que su visibilidad. Al ganar privacidad y minimizar la vulnerabilidad de datos o información que pueda ser utilizada en contra se pierde ciertas funciones que tendríamos en un repositorio público a las cuales se podría acceder por medio de planes pagos.

x) ¿Cómo crear un repositorio privado en GitHub?

A la hora de crear un repositorio como lo hicimos antes para el publico, la opción que cambiaría es que sea privado.



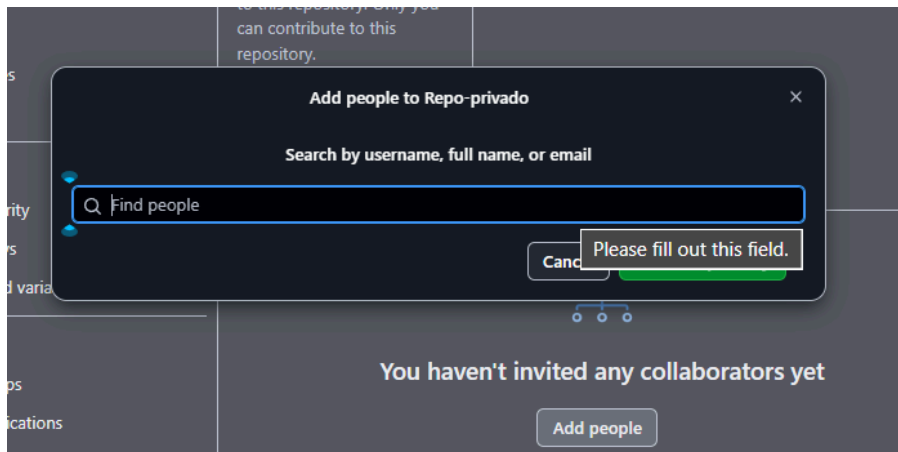
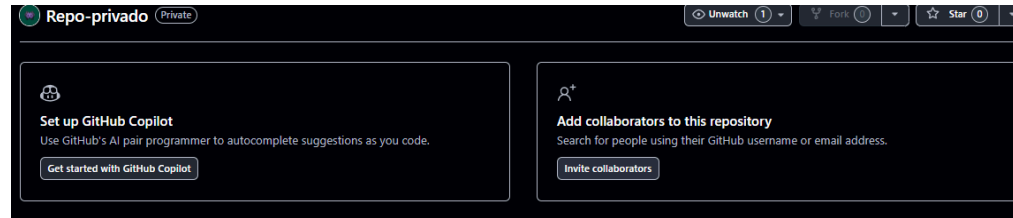
The screenshot shows the GitHub repository creation interface. At the top, there are fields for 'Owner \*' (LucasZarate) and 'Repository name \*'. Below these is a hint: 'Great repository names are short and memorable. Need inspiration? How about ideal-potato?'. Then there is a 'Description (optional)' text area. At the bottom, there are two radio button options: 'Public' (with an open lock icon) and 'Private' (with a closed lock icon). The 'Private' option is selected, indicated by a blue dot next to its radio button. The text for 'Public' says 'Anyone on the internet can see this repository. You choose who can commit.' The text for 'Private' says 'You choose who can see and commit to this repository.'



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

Primero que nada deberíamos tenerlo creado y se podría invitar a alguien a colaborar de dos formas vía usuario de GitHub o por un correo electrónico.

En al siguiente imagen deberíamos hacer click en “Add collaborators to this repository”

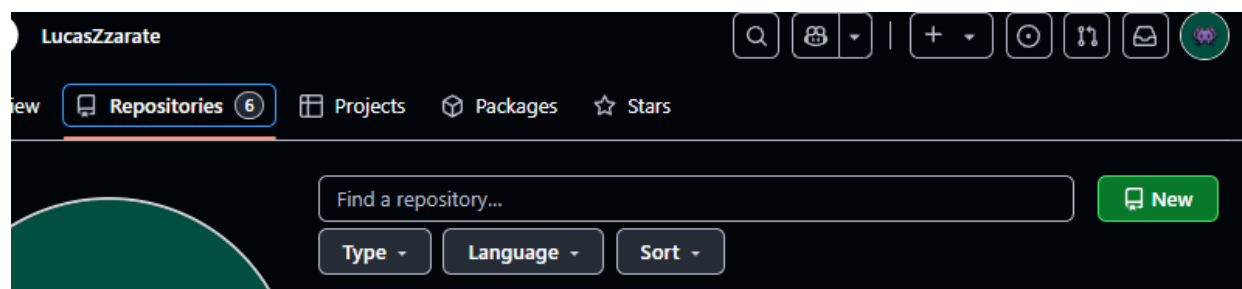


z) ¿Qué es un repositorio público en GitHub?

Exactamente lo mismo que en el caso anterior, un repositorio remoto es decir un lugar donde se almacena código , los archivos y el historial de revisiones de cada archivo, donde el repositorio puede controlar con múltiples colaboradores, en este caso sería público. Es decir que todo el mundo puede ver lo que haya en ese repositorio.

aa) ¿Cómo crear un repositorio público en GitHub?

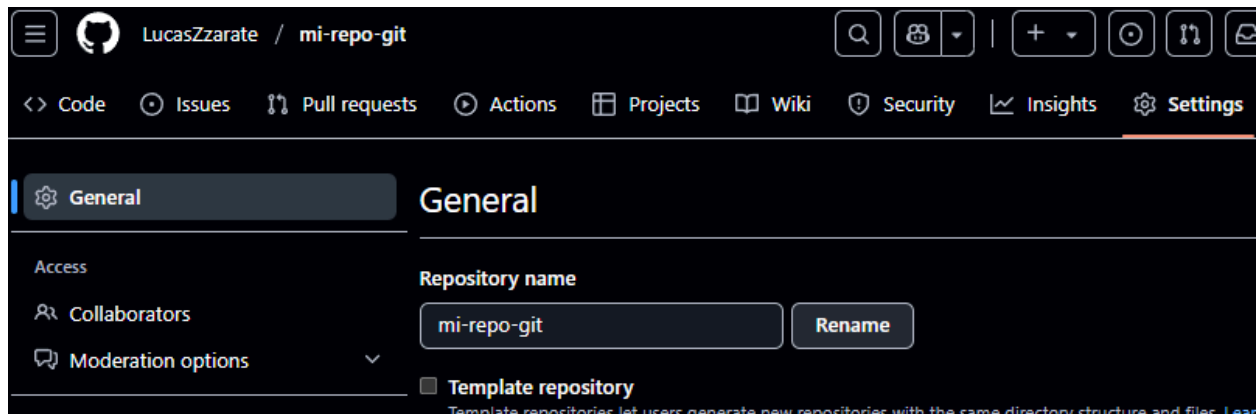
Lo mismo que en caso anterior para un repositorio privado, se debe acceder a GitHub entramos a nuestro perfil. Y clickeamos la sección de repositorios y entramos en New.



En este caso deberíamos ponerlo público, con su respectivo nombre y alguna descripción que sea de interés y que ayude a las personas que quieran colaborar con el proyecto.

bb) ¿Cómo compartir un repositorio público en GitHub?

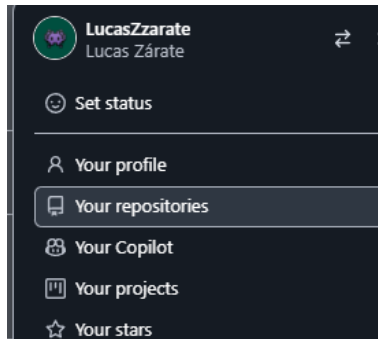
Primero entramos en el repositorio de interés (en este caso se llama mi-repo-git). Luego entramos a configuraciones (Settings), y hacemos click en la sección de la izquierda que dice Collaborators y se realiza lo mismo que en el caso anterior.



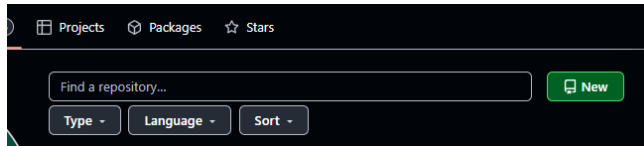
## 2) Realizar la siguiente actividad:

### Crear un repositorio

- Dale un nombre al repositorio
  - Elige el repositorio se publicó
  - Inicializa el repositorio con un archivo
1. Primero entramos a GitHub, con nuestro usuario de tenerlo, sino deberíamos crearlo.
  2. Una vez dentro vamos a your repositories



3. Cliqueamos en New.



4. Nos va a aparecer la siguiente imagen.

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

*Required fields are marked with an asterisk (\*).*

**Repository template**  
 Start your repository with a template repository's contents.

**Owner \*** **Repository name \***  
 /

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

**Description (optional)**

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

☐ **Private**  
You choose who can see and commit to this repository.

**Initialize this repository with:**  
☐ Add a README file  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

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

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

① You are creating a public repository in your personal account.

5. Dentro de dicha “planilla”, debemos darle un nombre a nuestro repositorio.

**Owner \*** **Repository name \***  
 /

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

**Description (optional)**

---

**Owner \*** **Repository name \***  
 /

✔ mi-nuevo-repo is available.

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

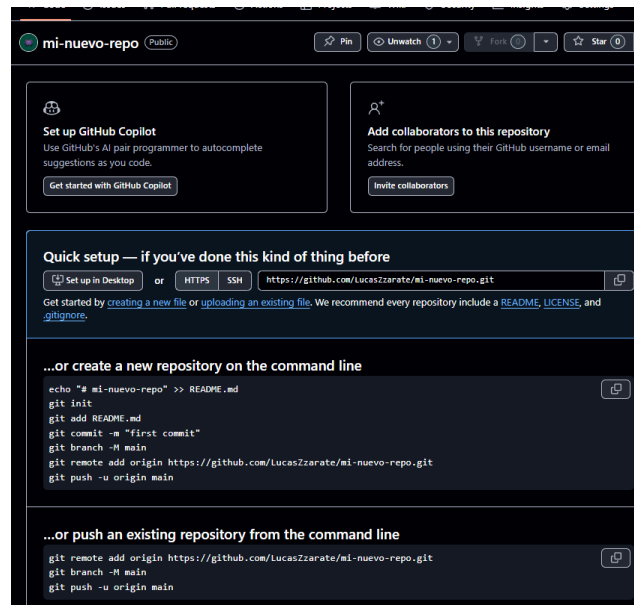
**Description (optional)**

6. Debo poner en este caso que sea ‘publico’.

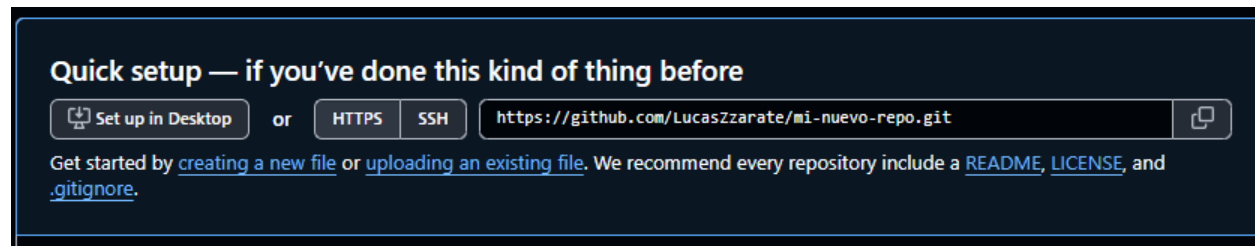
☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

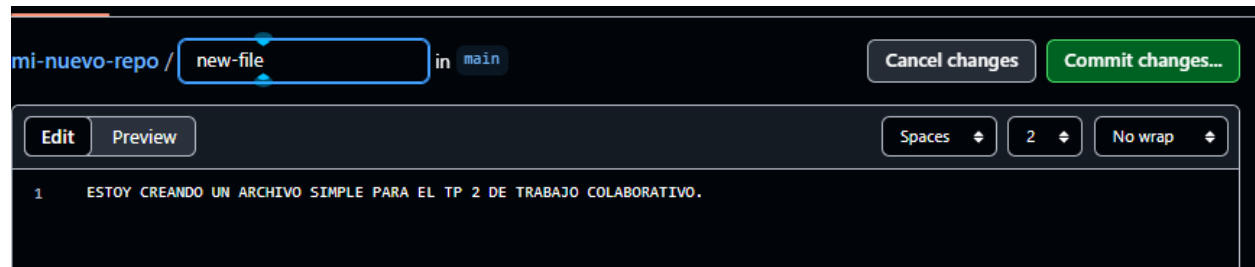
7. Al terminar debemos hacer clic en create repository.



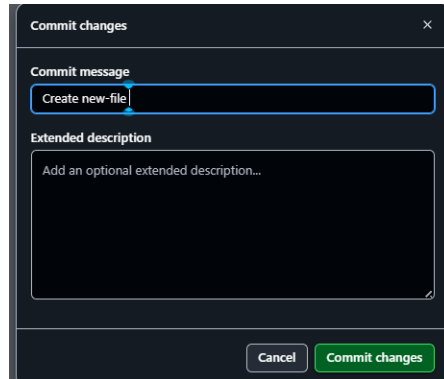
8. Por último hay que inicializar el repositorio con un archivo. Hago click en [creating a new file](#)



9. Una vez que entramos. Le damos un nombre a nuestro archivo “new-file” y agregamos algo de texto plano para poder guardarlo.



10. Hacemos un commit para guardar nuestros avances y listo. Es decir entramos a [Commit changes](#)., nos muestra la siguiente imagen una vez que cliqueamos.



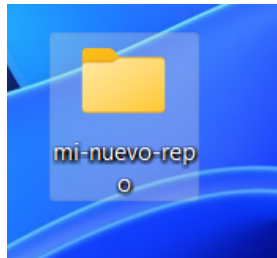
11. Por último “commit changes”, y se nos guarda un checkpoint de nuestro nuevo archivo creado.

### **Agregando un archivo**

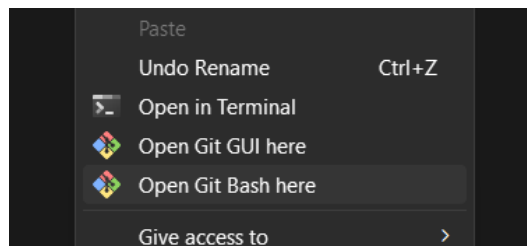
- Crea un archivo simple, por ejemplo, “mi-archivo.txt”.
- Realiza los comandos `git add .` y `git commit -m “Agregando mi-archivo.txt”` en línea de comando
- Sube los cambios al repositorio en GitHub con `git push origin main` (o el nombre de la rama correspondiente).

Para este punto deberíamos inicializar un repositorio local y sincronizarlo con nuestro repositorio remoto creado en el punto anterior.

1. Debo crear una carpeta en el escritorio, con el nombre de nuestro repositorio (una buena práctica es ponerle el mismo nombre al remoto).



2. Entrar a la carpeta y dentro hacer clic derecho, para abrir como consola a Git (Open Git Bash here).

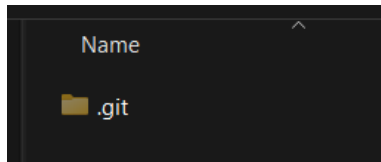


3. Inicializar el repositorio, con el comando `$ git init`. La segunda línea aparece luego de pulsar enter.

```
zarat@Lucasnotebook MINGW64 ~/OneDrive/Escritorio/mi-nuevo-repo (master)
$ |

zarat@Lucasnotebook MINGW64 ~/OneDrive/Escritorio/mi-nuevo-repo (master)
$ git init
Initialized empty Git repository in C:/Users/zarat/OneDrive/Escritorio/mi-nuevo-repo/.git/
```

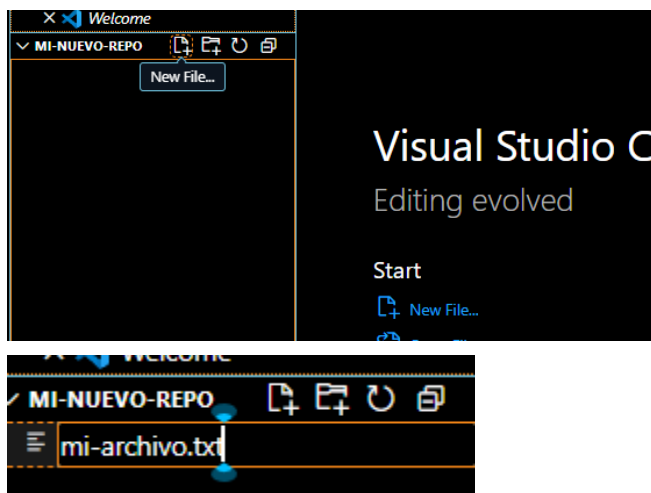
Si se ejecuta correctamente debería aparecer una nueva carpeta invisible dentro de la carpeta mi-nuevo-repo llamada “.git.”.

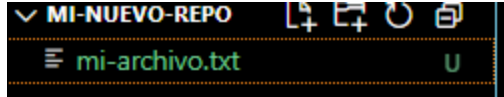


4. Una vez creado nuestro repo local, vamos a crear dentro de el un archivo .txt. Primero vamos a abrir nuestro editor de código en mi caso Visual Studio Code. Voy a mover la carpeta del escritorio y abrirla en el editor de código. Debería aparecer de la siguiente manera si lo hicimos bien.



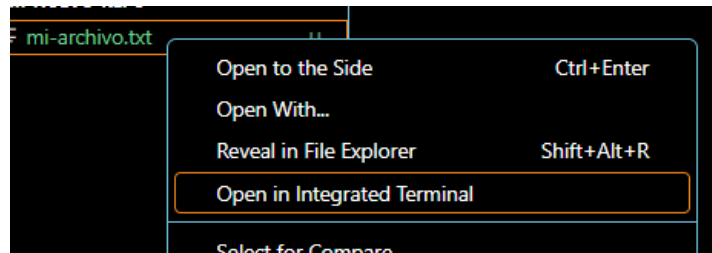
5. Abrimos la pestaña mi-nuevo-repo, y creamos un .txt con el nombre correspondiente. Podemos hacer clic en cualquiera de las dos opciones de new file.



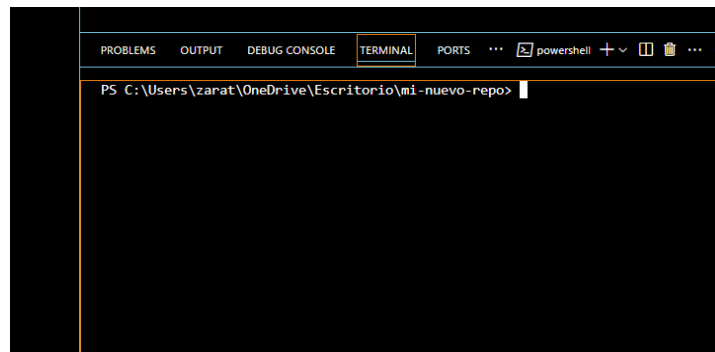


La U en verde nos indica que hay cambio para guardar, hasta ahora seria nuestro nuevo archivo.

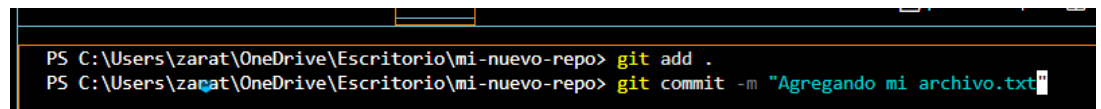
6. Para guardar nuestros cambios y hacer un checkpoint debemos:
  - a. Abrimos terminal con clic derecho en el archivo. Y luego clic en open in integrated Terminal



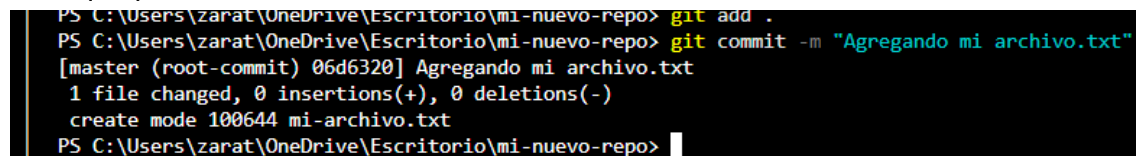
- b. Se va a desplegar de abajo y dentro vamos a poder poner los siguientes comandos para guardar nuestros cambios.



- c. Luego de ingresar los comando `git add .` y `git commit -m` con un comentario "Agregando mi archivo.txt"



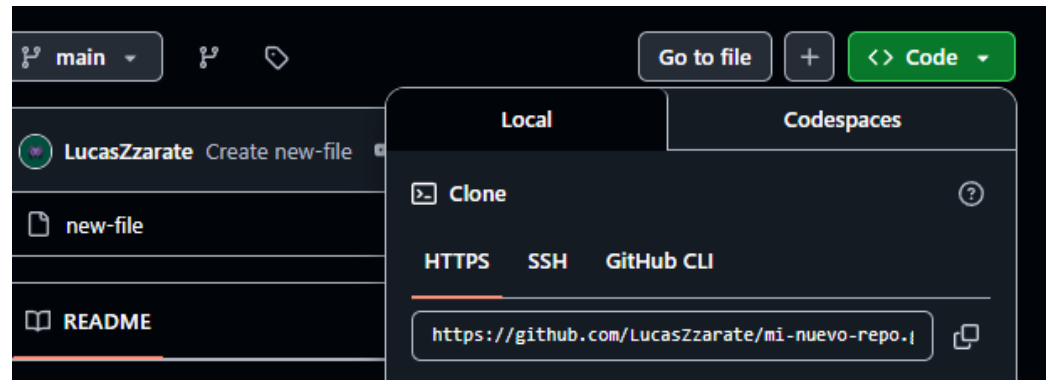
Si sale correctamente pecera las siguientes líneas, indicando que se agregó el commit con su respectivo comentario a la rama master , que hubo un cambio en un archivo (la creación del mismo) y tambien hay hash que pertenece al commit creado.



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



- a. Pero antes de realizar lo siguiente debemos ubicar nuestro repositorio remoto desde el local.  
Entramos a nuestro repositorio remoto en GitHub y copiamos la URL de nuestro repositorio.



Vamos a abrir Git Bash en la carpeta de nuestro repositorio local, y colocar los siguientes comandos.

1. Vamos a ubicar y agregar donde se encuentra nuestro repositorio remoto al local con `git remote add origin <URLcopiada>`

```
zarat@Lucasnotebook MINGW64 ~/OneDrive/Escritorio/mi-nuevo-repo (master)
$ git remote add origin https://github.com/LucasZzarate/mi-nuevo-repo.git
```

2. Ahora pusheamos los cambios, con `git -u origin master`

```
zarat@Lucasnotebook MINGW64 ~/OneDrive/Escritorio/mi-nuevo-repo (master)
$ git remote add origin https://github.com/LucasZzarate/mi-nuevo-repo.git

zarat@Lucasnotebook MINGW64 ~/OneDrive/Escritorio/mi-nuevo-repo (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 236 bytes | 78.00 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/LucasZzarate/mi-nuevo-repo/pull/new/master
remote:
To https://github.com/LucasZzarate/mi-nuevo-repo.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

```
zarat@Lucasnotebook MINGW64 ~/OneDrive/Escritorio/mi-nuevo-repo (master)
$ git push origin master
Everything up-to-date
```

```
zarat@Lucasnotebook MINGW64 ~/OneDrive/Escritorio/mi-nuevo-repo (master)
```

### Creando Branchs

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

1. Estamos en Git Bash parados en la rama master, ahora vamos a crear y cambiar de rama con el siguiente comando. `git checkout -b nueva_rama`, lo que hace este comando es posicionarnos en la rama `nueva_rama` y como no existe, la crea.

```
zarat@Lucasnotebook MINGW64 ~/OneDrive/Escritorio/mi-nuevo-repo (master)
$ git checkout -b nueva_rama
```

Si sale bien.

```
zarat@Lucasnotebook MINGW64 ~/OneDrive/Escritorio/mi-nuevo-repo (maste
$ git checkout -b nueva_rama
Switched to a new branch 'nueva_rama'
```

2. Primero corroboro estar parado en la rama de mi interes, para poder ejecutar la creacion de un nuevo archivo donde queremos.

```
zarat@Lucasnotebook MINGW64 ~/OneDrive/Escritorio/mi-repo-git (nueva_
$ git branch
  master
* nueva_rama

zarat@Lucasnotebook MINGW64 ~/OneDrive/Escritorio/mi-repo-git (nueva_
$ touch "texto.txt"
```

Efectivamente estamos en la nueva rama creada llamada “nueva\_rama”, y creamos un nuevo archivo con el comando `touch`, el nombre del archivo es “texto.txt” formato .txt.

Y ahora para confirmar y subir o pushear los cambios a nuestro repositorio remoto se utiliza “`git add .`” “`git commit -m “`” y “`git push`”

```
zarat@Lucasnotebook MINGW64 ~/OneDrive/Escritorio/mi-repo-git (nueva_rama)
$ git add .

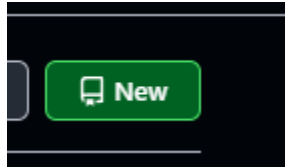
zarat@Lucasnotebook MINGW64 ~/OneDrive/Escritorio/mi-repo-git (nueva_rama)
$ git commit -m "Se agrego un nuevo archivo de tipo .txt"
[nueva_rama 1cba450] Se agrego un nuevo archivo de tipo .txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 texto.txt
```

```
zarat@Lucasnotebook MINGW64 ~/OneDrive/Escritorio/mi-repo-git (nueva_rama)
$ git push --set-upstream origin nueva_rama
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 310 bytes | 155.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'nueva_rama' on GitHub by visiting:
remote:   https://github.com/LucasZzarate/mi-repo-git/pull/new/nueva_rama
remote:
To https://github.com/LucasZzarate/mi-repo-git
 * [new branch]      nueva_rama -> nueva_rama
branch 'nueva_rama' set up to track 'origin/nueva_rama'.
```

### 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 (\*).

## Repository template

No template ▾

Start your repository with a template repository's contents.

Owner \*

 LucasZzarate ▾

Repository name \*

conflict-exercise

✔ conflict-exercise is available.

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

Description (optional)

Es una prueba para el ejercicio 3 del Tp 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 ▾

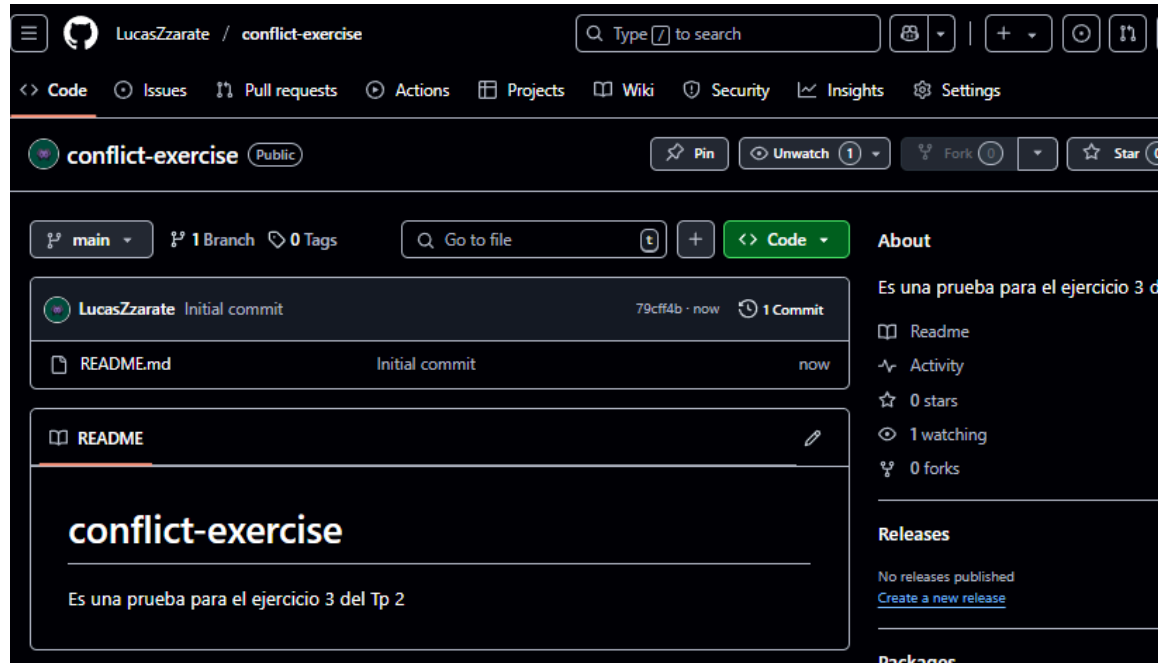
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

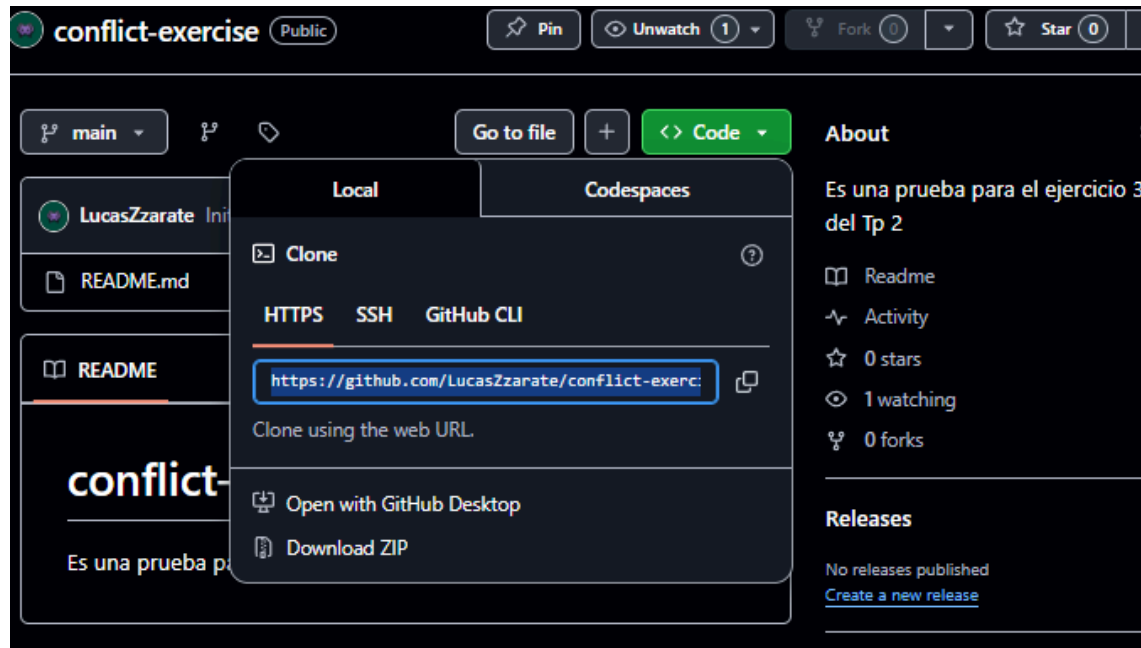


You are creating a public repository in your personal account.

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`

```

zarat@Lucasnotebook MINGW64 ~/OneDrive/Escritorio/conflict-exercise (master)
$ git clone https://github.com/LucasZzarate/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.

```

- Entra en el directorio del repositorio:

- `cd conflict-exercise`

```

zarat@Lucasnotebook MINGW64 ~/OneDrive/Escritorio/conflict-exercise (master)
$ cd conflict-exercise

zarat@Lucasnotebook MINGW64 ~/OneDrive/Escritorio/conflict-exercise/conflict-exercise (main)
$

```

- **Paso 3: Crear una breve rama y editar un archivo.**

- Crear una nueva rama llamada feature-branch:

`git checkout -b feature-branch`

```

PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise> git checkout -b feature-branch
Switched to a new branch 'feature-branch'
PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise>

```

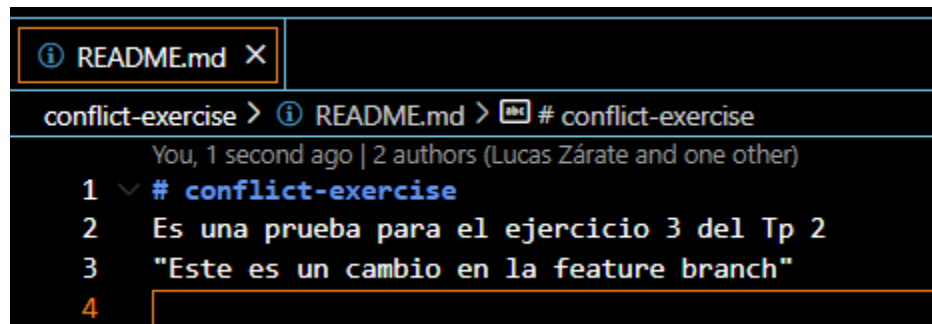
- Abre el archivo README.md en un editor de texto y añade una línea nueva, por ejemplo:

“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"`



```

PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise> git checkout -b feature-branch
Switched to a new branch 'feature-branch'
PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise> git branch
* feature-branch
  main
PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise> git add README.md
PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise> git commit -m "Added a line in feature-branch"
[feature-branch 5b0682d] Added a line in feature-branch
1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise>

```

- **Paso 4: Volver a la rama principal y editar el mismo archivo.**

- Cambia de vuelta a la rama principal (main):

`git checkout main`

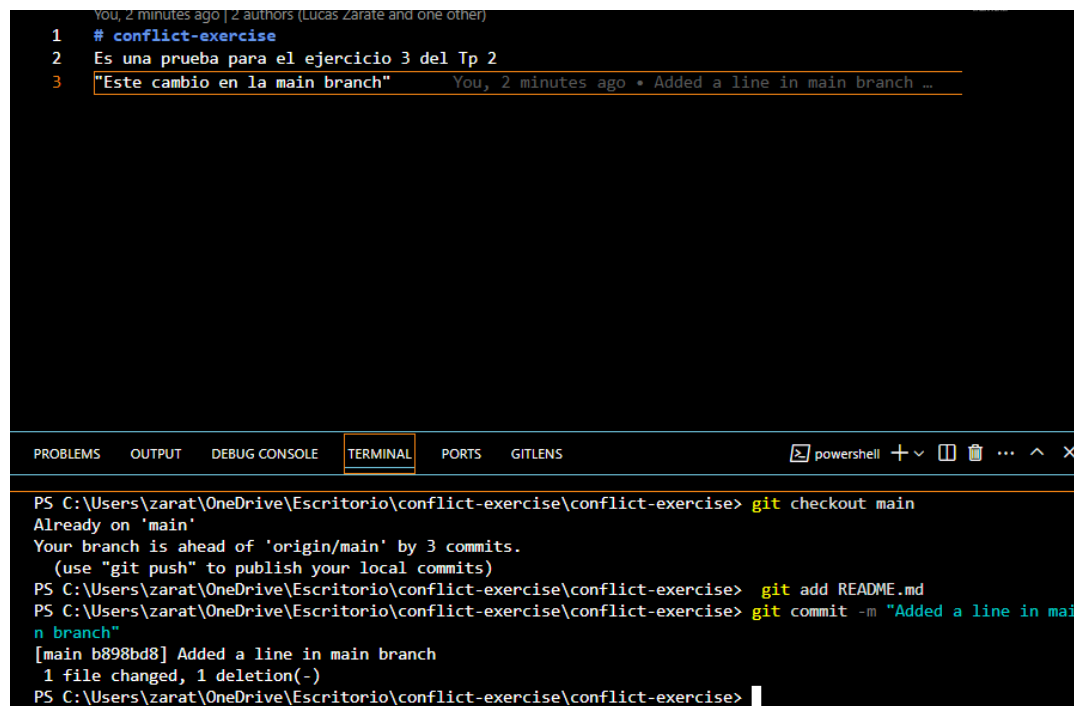
```
PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise> git checkout main
Already on 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise> |
```

- 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”`



The screenshot shows a code editor with a commit history on the left and a terminal window at the bottom. The commit history lists three commits: 1. # conflict-exercise, 2. Es una prueba para el ejercicio 3 del Tp 2, and 3. “Este cambio en la main branch” by You, 2 minutes ago. The terminal window shows the following commands and output:

```
PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise> git checkout main
Already on 'main'
Your branch is ahead of 'origin/main' by 3 commits.
(use "git push" to publish your local commits)
PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise> git add README.md
PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise> git commit -m "Added a line in main branch"
[main b898bd8] Added a line in main branch
1 file changed, 1 deletion(-)
PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise> |
```

- **Paso 5: Hacer un merge y generar un conflicto.**

- Intentar 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.

```

You, 46 seconds ago | 2 authors (Lucas Zárate and one other)
1 # conflict-exercise
2 Es una prueba para el ejercicio 3 del Tp 2
3 "Este cambio es para un conflicto" You, 45 seconds ago • Added a line in a new branch

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS powershell + -

PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise> git merge feature-branch
Already up to date.
PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise> git checkout -b feature-branch2
Switched to a new branch 'feature-branch2'
PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise> git add README.md
PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise> git commit -m "Added a line in a new branch"
[feature-branch2 8f1d6fe] Added a line in a new branch feature-branch
1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise> git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 4 commits.
(use "git push" to publish your local commits)
PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise> git merge feature-branch2
Updating b898bd8..8f1d6fe
Fast-forward
 README.md | 2 +-
1 file changed, 1 insertion(+), 1 deletion(-)

```

- Paso 6: Resolver el conflicto.
  - Abre el archivo README.md en tu editor de texto. Verás algo similar a esto:
 

```

<<<<<<<HEAD
Este es un archivo 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"`

```
PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise> git add README.md
PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise> git commit -m "Resolved merge conflict"
On branch main
Your branch is ahead of 'origin/main' by 5 commits.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
```

- Paso 7: Subir los cambios a GitHub
  - Sube los cambios de la rama main al repositorio remoto en GitHub:

`git push origin main`

```
PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise> git push origin main
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 12 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (13/13), 1.21 KiB | 35.00 KiB/s, done.
Total 13 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/LucasZarate/conflict-exercise.git
79cff4b..8f1d6fe main -> main
```

- También sube la feature-branch si deseas:

`git push origin feature-branch`

```
PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\conflict-exercise> git push origin main
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 12 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (13/13), 1.21 KiB | 35.00 KiB/s, done.
Total 13 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/LucasZarate/conflict-exercise.git
79cff4b..8f1d6fe main -> main
PS C:\Users\zarat\OneDrive\Escritorio\conflict-exercise\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/LucasZarate/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/LucasZarate/conflict-exercise.git
* [new branch]   feature-branch -> 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 la resolución.

