

# **LABORATORIO 1- INTRODUCCIÓN GIT**

CRISTIAN DAVID POLO GARRIDO

Laboratorio 1 correspondiente al primer tercio

Profesor Oscar David Ospina Rodríguez

Docente de cátedra de Ciclos de Vida de Desarrollo de Software

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

DEPARTAMENTO DE CIENCIAS NATURALES

CICLOS DE VIDA DE DESARROLLO DE SOFTWARE

GRUPO 02

Bogotá, Colombia

2025 – 1

# LABORATORIO 1- INTRODUCCIÓN GIT

ESCUELA COLOMBIANA DE INGENIERÍA - CICLOS DE VIDA DE DESARROLLO DE SOFTWARE



- En el presente laboratorio vamos a aprender los manejos básicos de GitHub, esto con el propósito de que entiendas y comiences a trabajar con esta herramienta. Para este laboratorio se trabajará de manera individual la primera parte y con dos integrantes en la segunda parte.

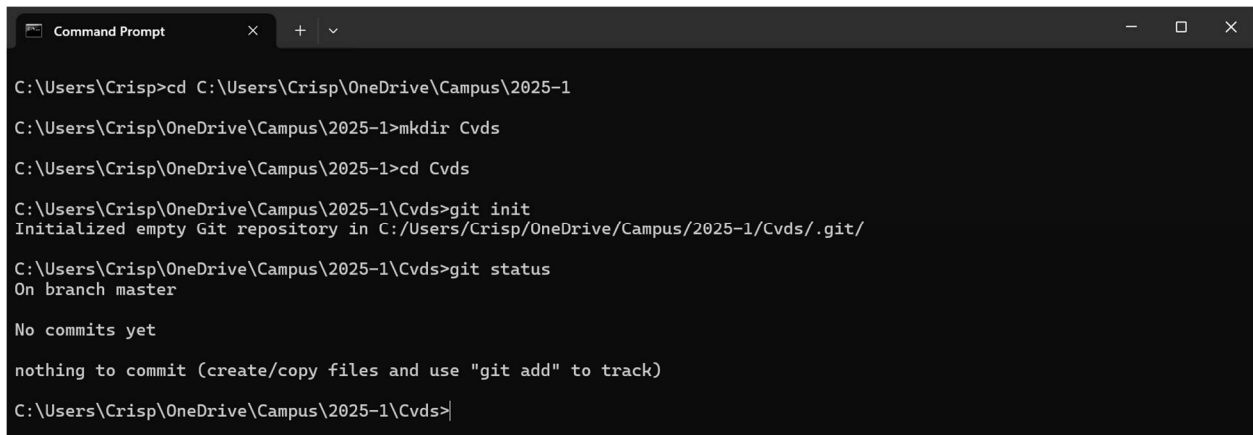
## PARTE I (Trabajo Individual).

1. Crea un repositorio localmente.

Para crear un repositorio local en una máquina Windows, lo primero es descargar e instalar la herramienta Git compatible. Posterior a esto, se debe abrir la terminal, navegar hasta la carpeta en donde deseamos crear el repositorio y allí podremos ir hacia la ruta por medio del comando **cd**. Ya estando sobre la ruta, podemos crear una carpeta en donde alojaremos el repositorio local con el comando **mkdir**, para mi caso, he decidido nombrarla como **Cvds**.

Comprobando la ubicación en donde deseamos alojar nuestro repositorio, hay que inicializarlo con Git dentro del directorio con el comando **git init**.

De esta manera, hemos creado ya un repositorio Git vacío en dicha carpeta. Podemos verificar esta información con el comando **git status**, en donde nos indica la rama en la que nos encontramos, información sobre los commits y los cambios que tenemos localmente que no han sido subidos al remoto.



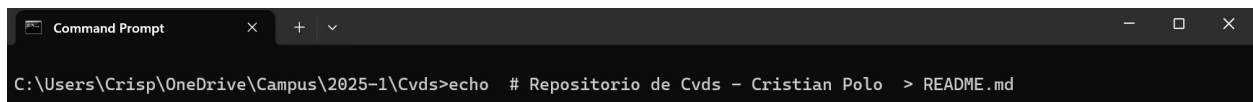
```
Command Prompt
C:\Users\Crisp>cd C:\Users\Crisp\OneDrive\Campus\2025-1
C:\Users\Crisp\OneDrive\Campus\2025-1>mkdir Cvds
C:\Users\Crisp\OneDrive\Campus\2025-1>cd Cvds
C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds>git init
Initialized empty Git repository in C:/Users/Crisp/OneDrive/Campus/2025-1/Cvds/.git/
C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds>git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds>
```

2. Agrega un archivo de ejemplo al repositorio, el **README.md** puede ser una gran opción.

Debemos crear el archivo por medio de la terminal, en este caso he querido ponerle una descripción por lo que lo he hecho con el comando **echo**, posterior a esto la descripción y finalmente colocamos el símbolo **>** con el nombre del archivo que queremos crear.



```
Command Prompt
C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds>echo # Repositorio de Cvds - Cristian Polo > README.md
```

3. Averigua para qué sirve y como se usan estos comandos **git add** y **git commit -m "mensaje"**

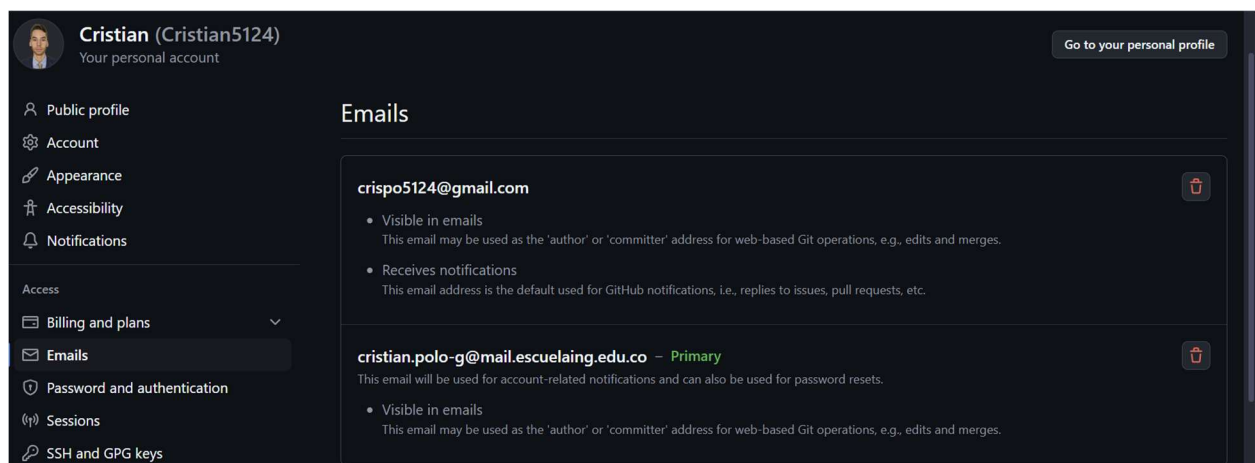
El comando **git add** se usa para agregar cambios al área de preparación, lo que significa que Git los tendrá en cuenta para el próximo commit. Puedes usarlo para agregar archivos específicos o todos los cambios en el directorio actual. Por ejemplo, si modificas un archivo llamado **archivo.txt** y quieres incluirlo en el commit, puedes escribir **git add archivo.txt**. Si deseas agregar todos los cambios en la carpeta en la que te encuentras, puedes usar **git add ..**

Por otro lado, `git commit -m "mensaje"` se utiliza para guardar los cambios de manera definitiva en el historial del repositorio. El mensaje que se coloca entre comillas debe describir brevemente qué cambios se hicieron, ya que esto ayuda a mantener un historial claro del proyecto. Cuando ejecutas este comando después de `git add`, los cambios que estaban en el área de preparación pasan a formar parte del historial del repositorio, creando un nuevo punto de referencia al que puedes volver en el futuro si es necesario.

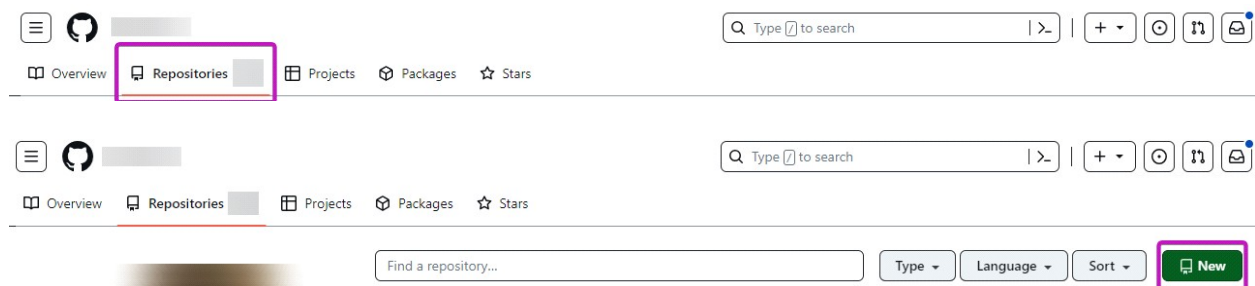
4. Abre una cuenta de github, si ya la tienes, enlazala con el correo institucional.

#### [Como enlazar correos en GitHub](#)

Para ello, me fui a mi cuenta de GitHub y he añadido el correo institucional de la Escuela como correo primario en la barra de configuración de la plataforma.

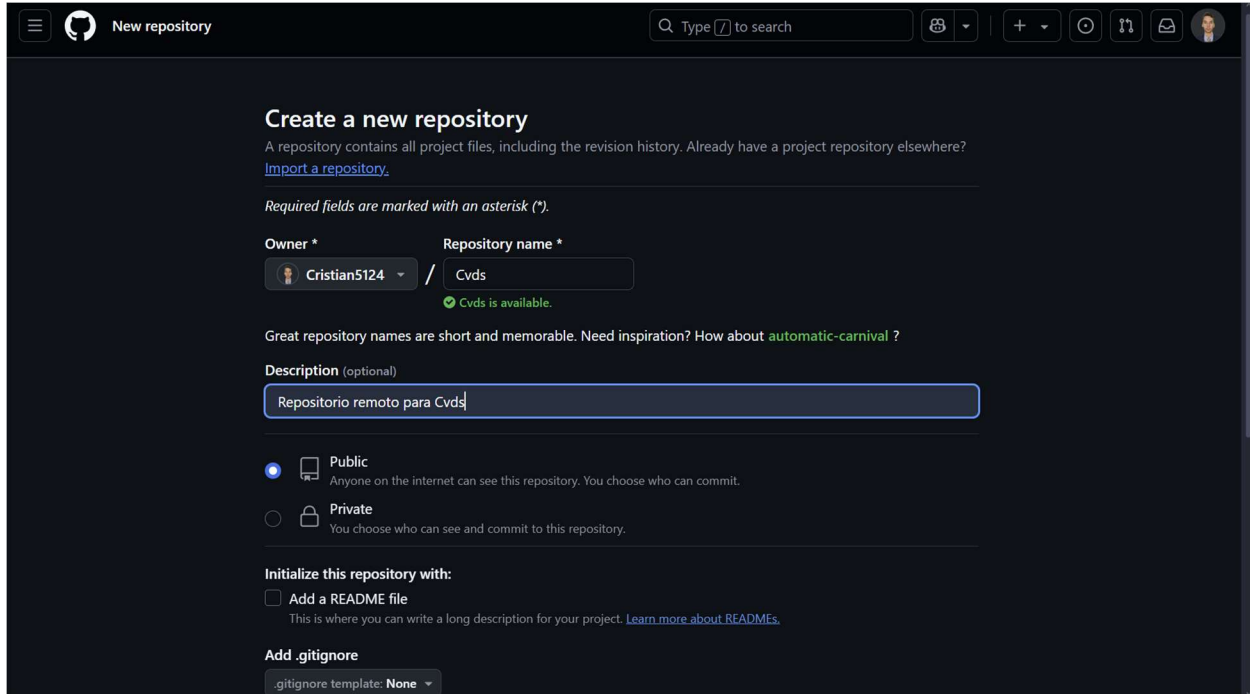


5. Crea un repositorio en blanco (vacío) e GitHub.



Para crear un repositorio en GitHub, primero debemos iniciar sesión en nuestra cuenta y dirigirnos a la pestaña "Repositories". Luego, hacemos clic en el botón "New" para comenzar con la creación. En la nueva página, ingresamos un nombre para el repositorio, asegurándonos de que sea claro y representativo del proyecto. También podemos agregar una descripción

opcional que explique brevemente su propósito. A continuación, elegimos si nuestro repositorio será público, permitiendo que cualquiera lo vea, o privado, restringiendo el acceso solo a personas autorizadas. GitHub también nos da la opción de inicializar el repositorio con un archivo *README*, donde podemos incluir información relevante sobre el proyecto, un archivo *.gitignore* para excluir ciertos archivos del control de versiones y una licencia si lo consideramos necesario. Finalmente, hacemos clic en **Create Repository** para completar el proceso y tener nuestro repositorio listo.



**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 \*

Cristian5124 / Cvds

✓ Cvds is available.

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

Description (optional)

Repositorio remoto para Cvds

☒ 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

6. Configura el repositorio local con el repositorio remoto.

#### [Como subir un proyecto local a github.](#)

El procedimiento comenzó con la inicialización (o reinicialización) de un repositorio Git local mediante el comando `git init` en el directorio "C:/Users/Crisp/OneDrive/Campus/2025-1/Cvds/". Después de esto, se procedió a agregar los archivos al área de preparación utilizando `git add ..`

Una vez preparados los archivos, se realizó el primer commit del proyecto usando `git commit` con el mensaje "first commit", durante el cual se creó un archivo *README.md*. El siguiente paso crucial fue establecer la conexión con el repositorio remoto, lo cual se logró mediante el comando `git remote add origin`, especificando la URL del repositorio en GitHub (<https://github.com/Cristian5124/Cvds.git>).

Para finalizar el proceso, se subieron los cambios al repositorio remoto utilizando *git push -u origin master*. La terminal muestra que esta operación fue exitosa, transfiriendo correctamente los archivos y configurando la rama *master* local para hacer seguimiento de la rama *master* remota. Todo este proceso estableció exitosamente la vinculación entre el repositorio local y el remoto, permitiendo así la futura sincronización del código entre ambos entornos de trabajo.

```
C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds>git init
Reinitialized existing Git repository in C:/Users/Crisp/OneDrive/Campus/2025-1/Cvds/.git/

C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds>git add .

C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds>git commit -m "first commit"
[master (root-commit) e5e7e2c] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md

C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds>git remote add origin https://github.com/Cristian5124/Cvds.git

C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds>git push -u origin master
info: please complete authentication in your browser...
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 265 bytes | 265.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Cristian5124/Cvds.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

7. Sube los cambios, teniendo en cuenta lo que averiguaste en el punto 3. Utiliza los siguientes comandos en el directorio donde tienes tu proyecto, en este orden:

```
git add .

git commit -m "mensaje, lo que hiciste con el archivo"

git push "url repositorio"
```

En el punto anterior he implementado estos cambios.

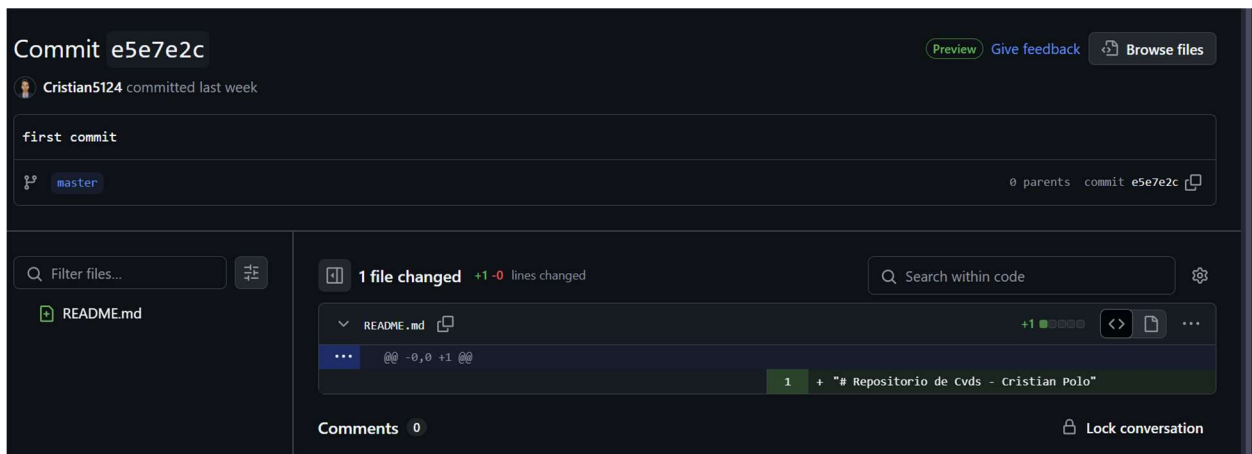
8. Configura el correo en git local de manera correcta [Configurar correo electrónico en GitHub](#)

Para configurar el correo podemos escribir el comando *git config --global user.email "correo"* y para el nombre digitamos *git config --global user.name "nombre"*. De esta manera, cuando hagamos un cambio en el repositorio remoto, aparecerá el nombre y correo de la persona que ha implementado los cambios.

```
C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds>git config --global user.email "cristian.polo-g@mail.escuelaing.edu.co"
C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds>git config --global user.name "Cristian Polo"
```

6. Vuelve a subir los cambios y observa que todo esté bien en el repositorio remoto (en GitHub).

Podemos encontrar el commit subido al repositorio remoto con los cambios realizados junto con el identificador del commit (`e5e7e2c`), archivos modificados y comentarios del commit.



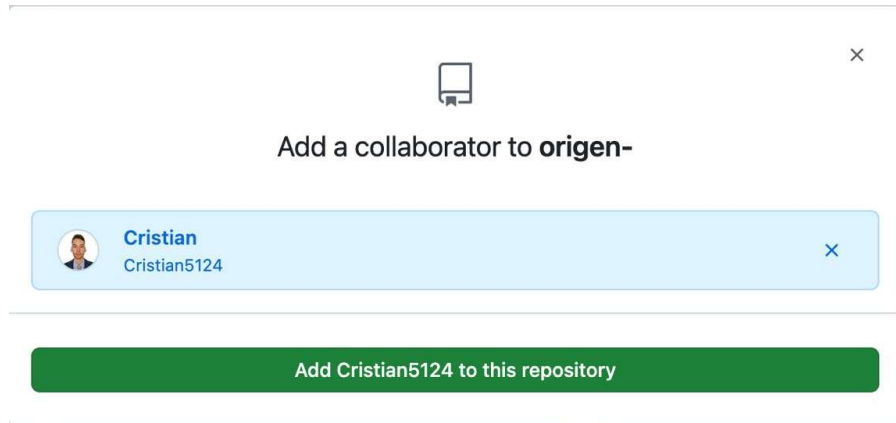
## PARTE II (Trabajo en parejas)

1. Se escogen los roles para trabajar en equipo, una persona debe escoger ser "Owner" o Propietario del repositorio y la otra "Collaborator" o Colaborador en el repositorio.

Para este caso, el propietario del repositorio será Santiago Arteaga y yo, Cristian Polo seré colaborador.

2. El owner agrega al colaborador con permisos de escritura en el repositorio que creó en la parte 1

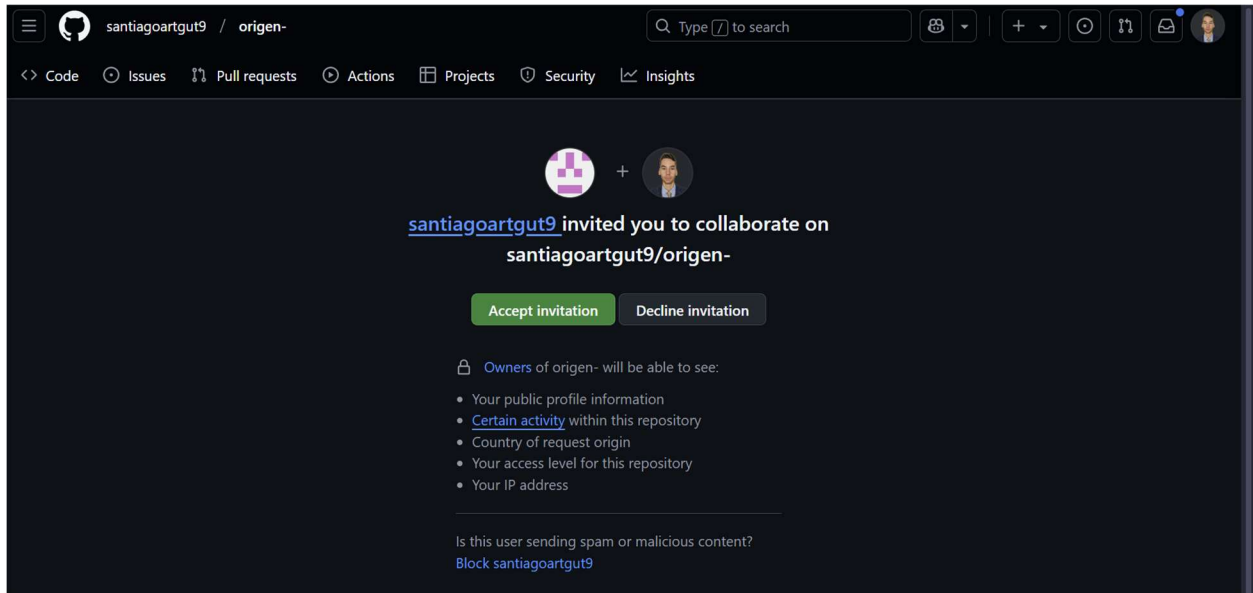
[Invitar colaboradores a un repositorio personal](#)



3. El owner le comparte la url via Teams al colaborador

La url del repositorio es <https://github.com/santiagoartgut9/origen-.git>.

4. El colaborador acepta la invitación al repositorio



Se acepta la invitación de Santiago para tener acceso al repositorio.

5. Owner y Colaborador editan el archivo README.md al mismo tiempo e intentan subir los cambios al mismo tiempo.



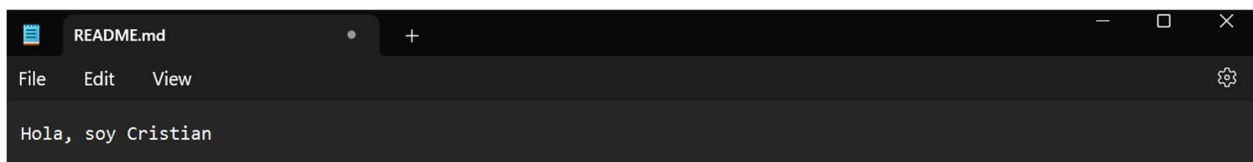
Para ello, nos vamos a la carpeta en donde queremos clonar la información de nuestro repositorio remoto y modificamos el archivo existente, en este caso, *pruebapro.txt*.

Para clonar el repositorio debemos dar el comando *git clone* seguido del vínculo del repositorio:

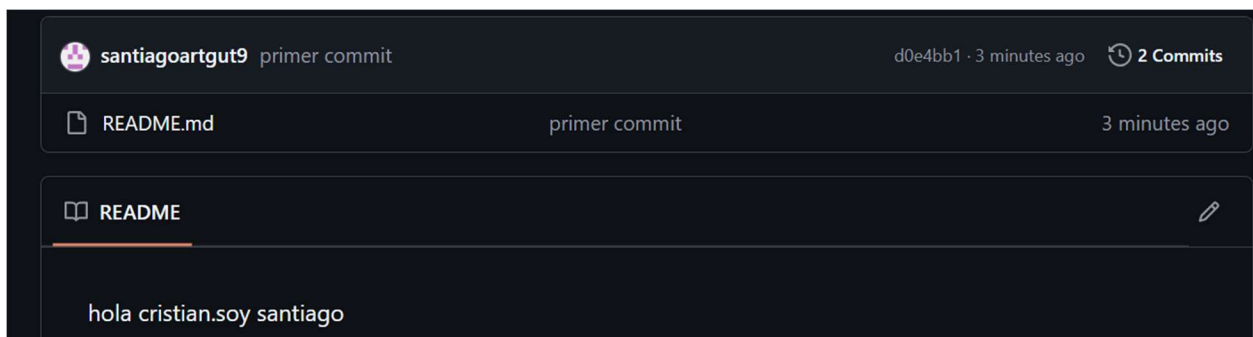
```
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01> git clone https://github.com/santiagoartgut9/origen-.git
Cloning into 'origen-'\...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01> |
```

Una vez verificado que contamos con los archivos locales, procedemos a modificar el archivo README.md.

```
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01> cd .\origen-
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> notepad .\README.md
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> |
```



Por otra parte, mi compañero ha modificado el mismo archivo, pero con la información “hola Cristian.soy santiago”, ha hecho el commit con nombre “primer commit” y ha subido los cambios al repositorio remoto.



Por mi parte, después de haber modificado el archivo, hice un commit con ese cambio de nombre “Commit de Cristian” y he subido de igual manera los cambios hechos por mí al repositorio.

```

PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01> cd .\origen-
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> notepad .\README.md
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> git add .
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> git commit -m "Commit de Cristian"
[master 5489b04] Commit de Cristian
1 file changed, 1 insertion(+)
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> git push -u origin master
To https://github.com/santiagoartgut9/origen-.git
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/santiagoartgut9/origen-.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> |

```

## 6. ¿Qué sucedió?

Lo que ocurrió fue un error de conflictos ya que cuando Santiago subió sus cambios, automáticamente yo pasé a tener la información desactualizada, cuando intento subir mis cambios estoy intentando borrar los cambios hechos por Santiago, es por esto por lo que debemos resolver el conflicto dejando como commit final el de Santiago, el mío o crear uno nuevo en el que la información sea actualizada para ambos. Es decir, el problema es que la información del repositorio remoto contiene trabajo que yo no poseo en mi repositorio local ya que Santiago subió la información del *README.md* antes que yo.

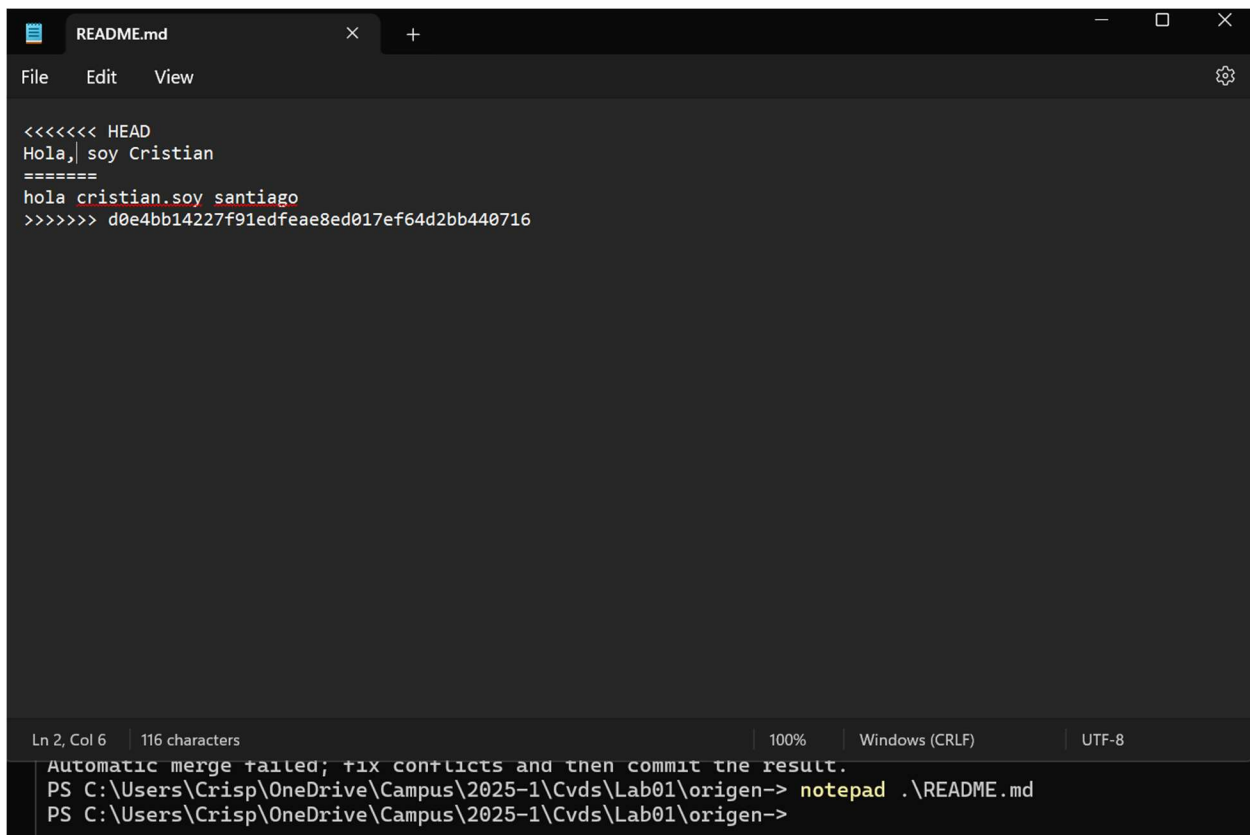
- La persona que perdió la competencia de subir los cambios, tiene que resolver los conflictos, cuando haces pull de los cambios, los archivos tienen los símbolos << == y >> (son normales en la resolución de conflictos), estos conflictos debes resolverlos manualmente. [Como resolver Conflictos GitHub](#)

Lo primero que tengo que hacer es un git pull y posterior a ello abrir el archivo *README.md*.

```

PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 251 bytes | 25.00 KiB/s, done.
From https://github.com/santiagoartgut9/origen-
 996a178..d0e4bb1  master    -> origin/master
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> notepad .\README.md
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> |

```



```

<<<<<< HEAD
Hola, soy Cristian
=====
hola cristian.soy santiago
>>>>>> d0e4bb14227f91edfeae8ed017ef64d2bb440716

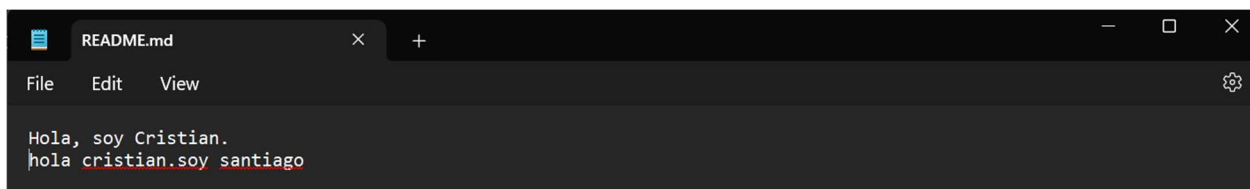
```

Ln 2, Col 6 | 116 characters | 100% | Windows (CRLF) | UTF-8

Automatic merge failed; fix conflicts and then commit the result.  
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> notepad .\README.md  
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen->

Al abrir el archivo, podemos darnos cuenta de que contiene los símbolos mencionados anteriormente y aparecen los commits que ambos hemos intentado subir al repositorio, evidentemente los cambios de Santiago cuentan con el número del commit ya que él si pudo subir los cambios y son estos cambios los que se encuentran en el repositorio remoto.

Para resolver el error, hago un git pull y modifico el *README.md*, en este caso quiero dejar los cambios de ambos de esta manera:

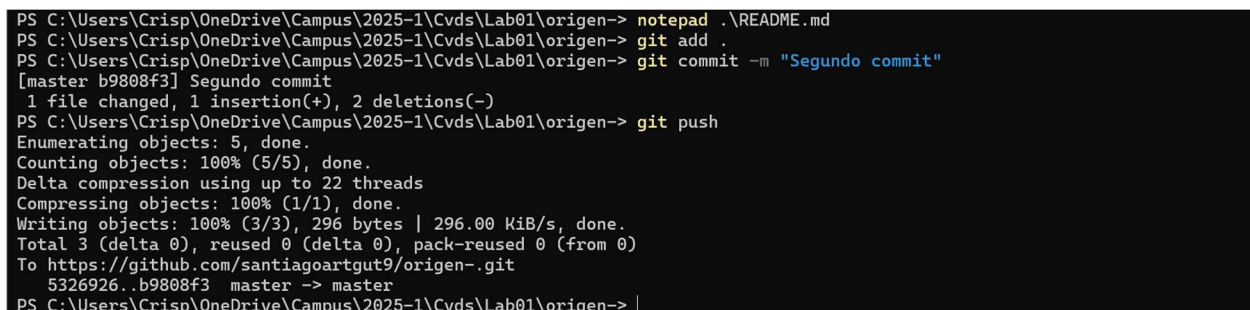


```

Hola, soy Cristian.
hola cristian.soy santiago

```

Hago entonces un commit llamado “Segundo commit” y hago el push.

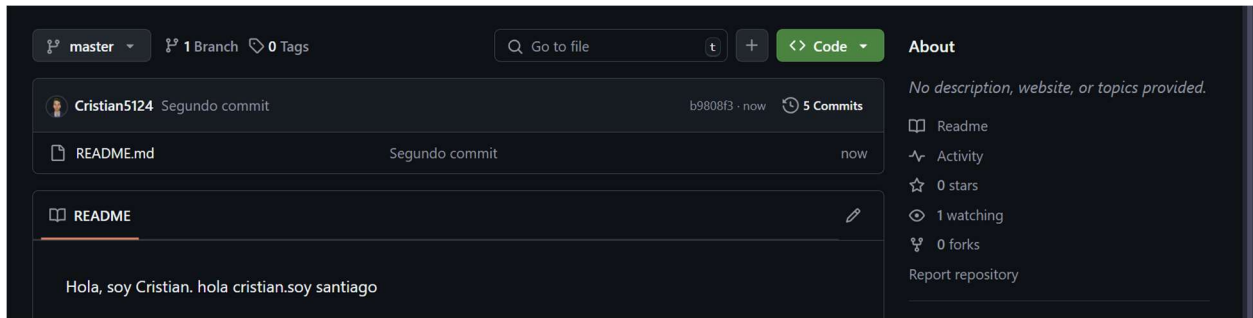


```

PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> notepad .\README.md
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> git add .
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> git commit -m "Segundo commit"
[master b9808f3] Segundo commit
1 file changed, 1 insertion(+), 2 deletions(-)
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 22 threads
Compressing objects: 100% (1/1), done.
Writing objects: 100% (3/3), 296 bytes | 296.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/santiagoartgut9/origen-.git
5326926..b9808f3 master -> master
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen->

```

Finalmente podemos revisar en el repositorio remoto los cambios hechos y la solución a nuestro conflicto ha finalizado:



8. Volver a repetir un cambio sobre el README.md ambas personas al tiempo para volver a tener conflictos.

Para esta ocasión haremos lo contrario, yo he subido mis cambios y están en el repositorio remoto, pero Santiago está trabajando con la versión antigua del archivo `README.md`, por lo que a él le genera este error al intentar modificar el `README.md` agregando información:

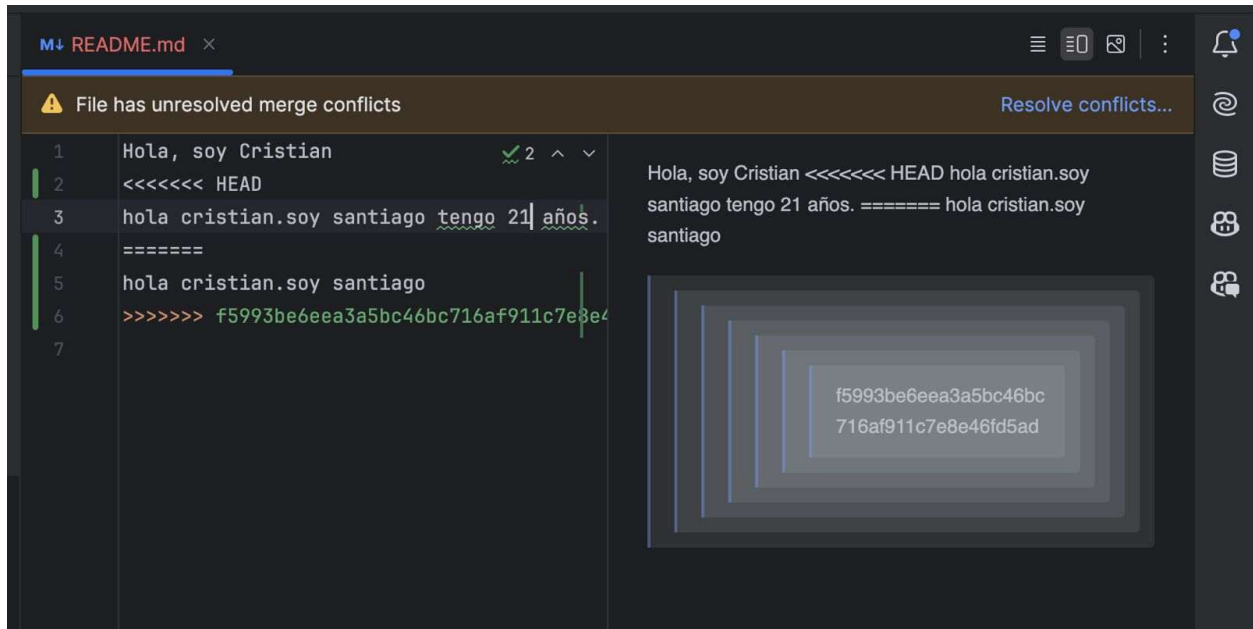
```
Ciclos — nano README.md — 80x24
GNU nano 2.0.6 File: README.md Modified
Hola, soy Cristian
hola cristian.soy santiago tengo 21 años.

(base) santiagoarteaga@Santiagos-MacBook-Pro Ciclos % git commit -m "tercer commit"
[master a04f5f1] tercer commit
7 files changed, 38 insertions(+), 2 deletions(-)
create mode 100644 .DS_Store
create mode 100644 .idea/.gitignore
create mode 100644 .idea/Ciclos.iml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
(base) santiagoarteaga@Santiagos-MacBook-Pro Ciclos % git push
To https://github.com/santiagoartgut9/Ciclos.git
! [rejected] master -> master (fetch first)
error: failed to push some refs to 'https://github.com/santiagoartgut9/Ciclos.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
(base) santiagoarteaga@Santiagos-MacBook-Pro Ciclos %
```

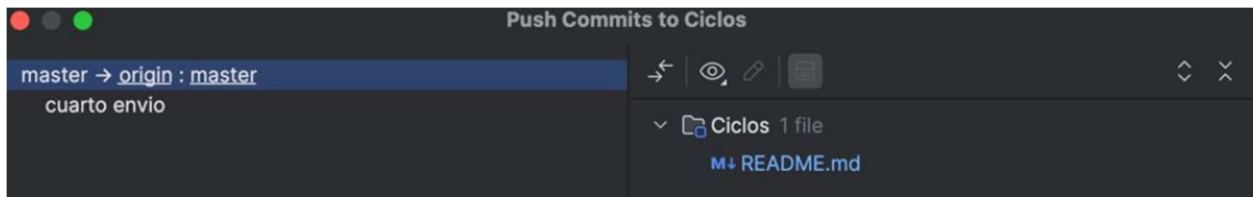
9. Resuelvan el conflicto con IntelliJ si es posible, [Resolver conflictos en IntelliJ](#)

De esta forma ya sabes resolver conflictos directamente sobre los archivos y usando un IDE como IntelliJ, esto te será muy útil en los futuros trabajos en equipo con Git.

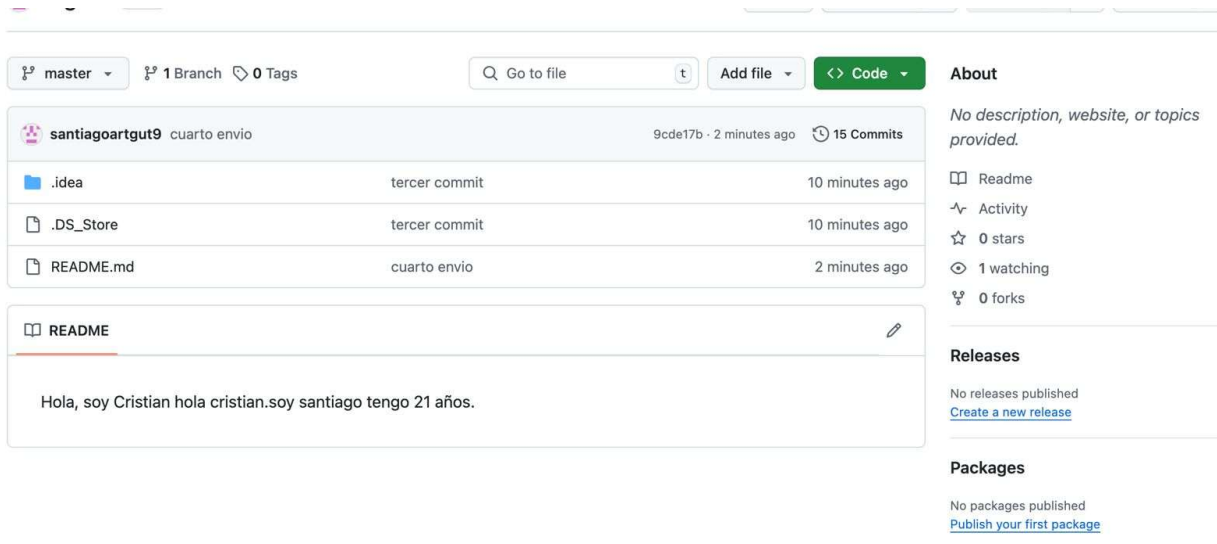
Cuando Santiago abre el archivo en su plataforma de IntelliJ, es muy práctico que la aplicación le muestre los conflictos después de hacer un git pull y con la opción *Resolve conflicts* puede escoger qué cambios dejar en el archivo para hacerle un commit y subirlos para resolver el conflicto:



En este commit, él desea agregar la información de que tiene 21 años, es por esto que ha resuelto el conflicto en IntelliJ y ha subido los cambios en un commit llamado “cuarto envío”.



Finalmente podemos ver los cambios reflejados en el repositorio remoto:



## PARTE III (Trabajo de a parejas)

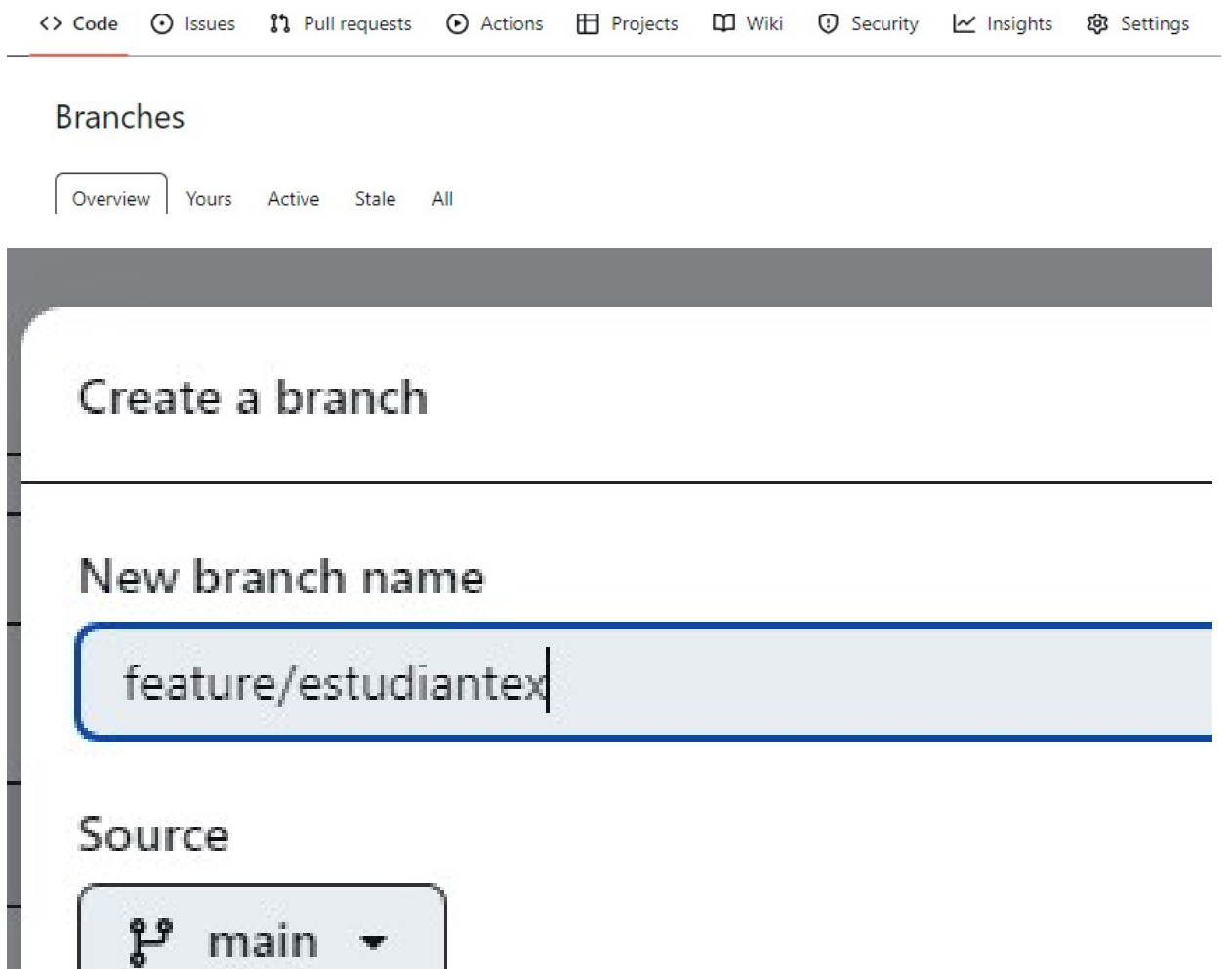
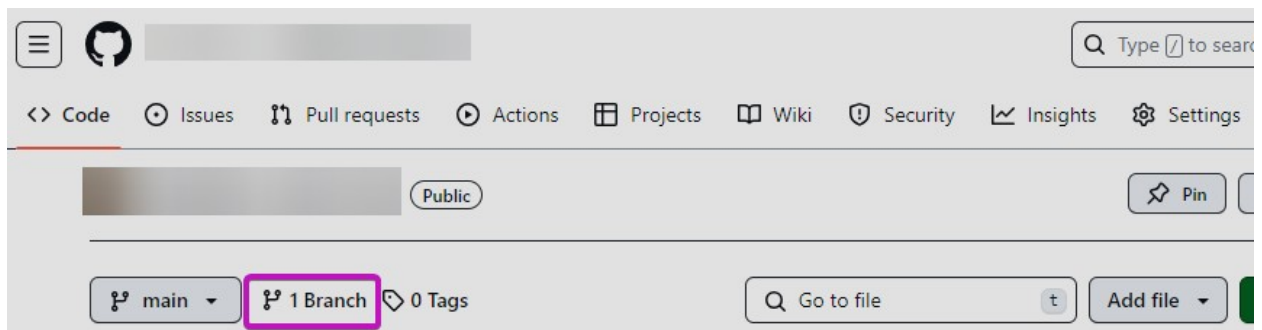
### 1. ¿Hay una mejor forma de trabajar con git para no tener conflictos?

Una mejor forma de trabajar con Git para evitar conflictos es mediante el uso de ramas (branches), donde cada desarrollador puede crear su propia rama para implementar características o correcciones específicas sin afectar la rama principal. Este método permite trabajar de manera aislada, hacer pruebas y, una vez que el código está listo, integrarlo a la rama principal mediante pull requests, facilitando así la revisión de código y reduciendo significativamente la posibilidad de conflictos en proyectos con múltiples colaboradores.

### 2. ¿Qué es y cómo funciona el **Pull Request**?

Un Pull Request es una funcionalidad de sistemas de control de versiones como GitHub que permite a los desarrolladores notificar que han completado una nueva característica o corrección en una rama y solicitar su integración a la rama principal. Funciona como un espacio de colaboración donde otros desarrolladores pueden revisar el código, hacer comentarios y sugerir cambios antes de que estos sean fusionados, asegurando así la calidad del código y la integridad del proyecto.

### 3. Creen una rama cada uno y suban sus cambios



Creamos la rama cada uno dentro del repositorio, en mi caso lo he puesto como nombre *cristian*:



Branches New branch

Overview Yours Active Stale All

Q Search branches...

**Default**

Branch	Updated	Check status	Behind   Ahead	Pull request
master	13 hours ago		Default	...

**Your branches**

Branch	Updated	Check status	Behind   Ahead	Pull request
cristian	now		0   0	...

**Active branches**

Branch	Updated	Check status	Behind   Ahead	Pull request
cristian	now		0   0	...
santiago	3 minutes ago		1   0	...

De esta manera, tenemos una rama para cada uno aparte de la rama principal.

4. Tanto owner como colaborador hacen un cambio en el README.md y hacen un Pull Request (PR) a la rama main/master.

#### [Pull Request - GitHub](#)

**(Recomendación : deben trabajar en equipo y distribuirse de mejor manera quien va a modificar qué, para evitar modificar los mismos archivos al mismo tiempo, si no se editan los mismos archivos la resolución de conflictos es automática)**

Por su parte, Santiago ha realizado los cambios en su rama primero y posterior a ello ha llevado a cabo un commit:



```
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md

(base) santiagoarteaga@Santiagos-MacBook-Pro Ciclos % git commit -m "cuarto envio"
[santiago 5229f74] cuarto envio
1 file changed, 1 insertion(+)
(base) santiagoarteaga@Santiagos-MacBook-Pro Ciclos % git push -u origin santiago

Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 344 bytes | 344.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote: This repository moved. Please use the new location:
```

Jan 29, 2025, 10:31 PM GMT-5

santiago 3 Branches 0 Tags

Go to file Add file Code

This branch is 1 commit ahead of, 1 commit behind master .

Contribute

santiagoartgut9	cuarto envio	5229f74 · 1 minute ago	18 Commits
.idea	tercer commit	13 hours ago	
.DS_Store	tercer commit	13 hours ago	
README.md	cuarto envio	1 minute ago	

README

Hola, soy Cristian hola cristian.soy santiago tengo 21 años. estudio en la escuela colombiana de ingeniería.

No de: provid

Re: Ac: 0 s 1 w 0 f

Releas

No relea Create a

Packa

No pack Publish

Contri

s

Es muy útil que la plataforma nos muestra la diferencia entre la rama que trabajamos y la rama master. Finalmente, Santiago ha hecho un pull request para subir los cambios a la rama principal:

Filters is:pr is:open Labels 9 Milestones 0 New pull request

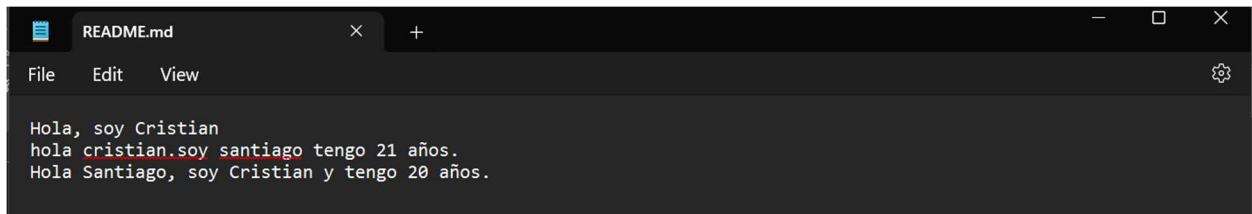
1 Open 0 Closed Author Label Projects Milestones Reviews Assignee Sort

cuarto envio

#1 opened 3 minutes ago by santiagoartgut9

Ahora bien, en mi caso yo he modificado el archivo *README.md* y he hecho un commit para guardar los cambios y subirlos al repositorio:

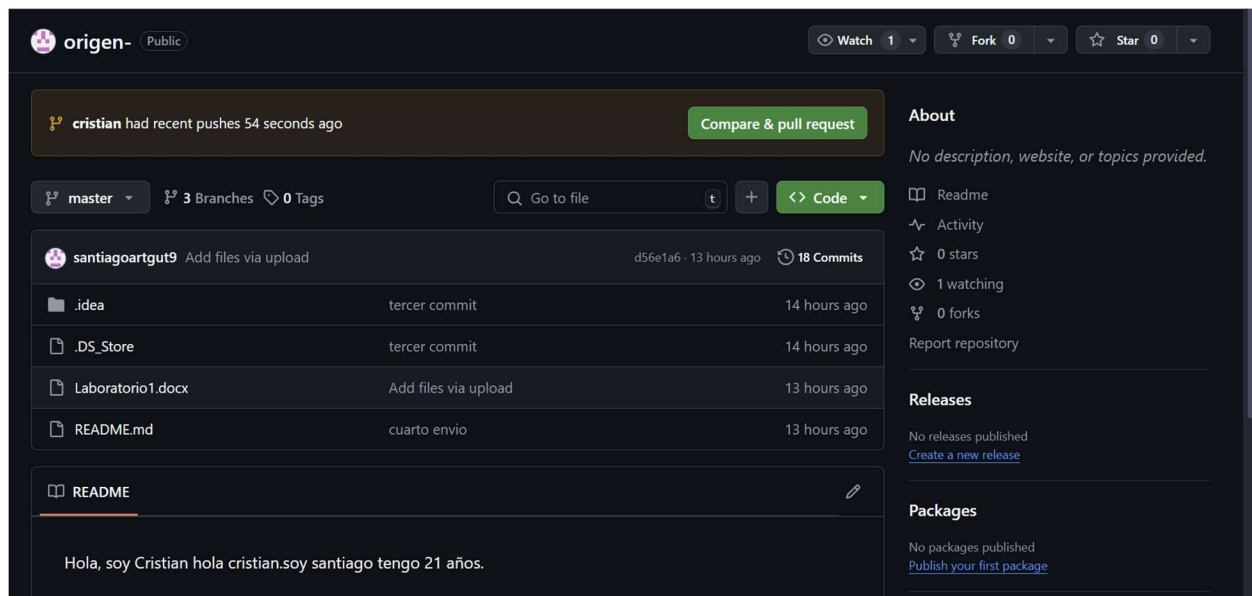
```
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 3 (delta 1), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 324 bytes | 18.00 KiB/s, done.
From https://github.com/santiagoartgut9/origen-
 0b10eb1..5229f74  santiago  -> origin/santiago
Updating f5993be..d56e1a6
Fast-forward
 .DS_Store      | Bin 0 -> 6148 bytes
 .idea/.gitignore | 8 ++++++++
 .idea/Ciclos.iml | 9 ++++++++
 .idea/misc.xml  | 6 ++++++
 .idea/modules.xml | 8 ++++++++
 .idea/vcs.xml   | 6 ++++++
 Laboratoriol.docx | Bin 0 -> 1963891 bytes
 README.md       | 2 +-
8 files changed, 38 insertions(+), 1 deletion(-)
create mode 100644 .DS_Store
create mode 100644 .idea/.gitignore
create mode 100644 .idea/Ciclos.iml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 Laboratoriol.docx
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> notepad .\README.md
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> |
```



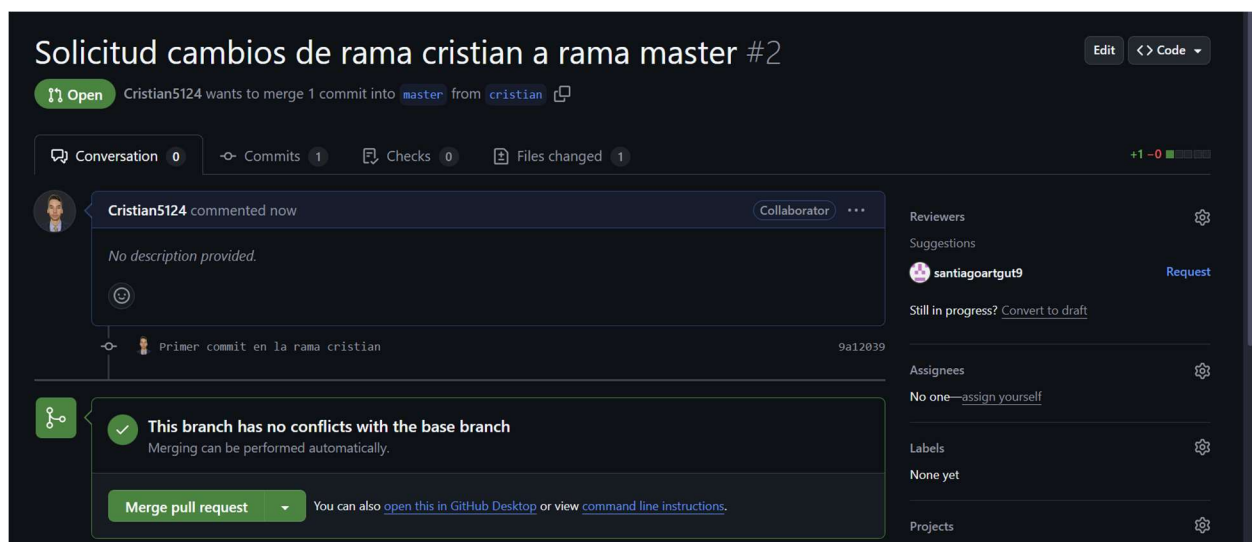
Ahora, hago el commit y el push de los cambios en mi rama:

```
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> git branch
cristian
* master
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> git checkout cristian
M README.md
Switched to branch 'cristian'
Your branch is up to date with 'origin/cristian'.
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> notepad .\README.md
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> git add .
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> git commit -m "Primer commit en la rama cristian"
[cristian 9a12039] Primer commit en la rama cristian
1 file changed, 1 insertion(+)
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> git push -u origin cristian
Enumerating objects: 5, done.
Delta compression using up to 22 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 343 bytes | 343.00 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/santiagoartgut9/origen-.git
 d56e1a6..9a12039  cristian -> cristian
branch 'cristian' set up to track 'origin/cristian'.
PS C:\Users\Crisp\OneDrive\Campus\2025-1\Cvds\Lab01\origen-> |
```

Cuando revisamos el repositorio remoto, nos aparece un mensaje que nos indica que la rama Cristian tiene un push nuevo y que la rama master está desactualizada con respecto a la rama principal.

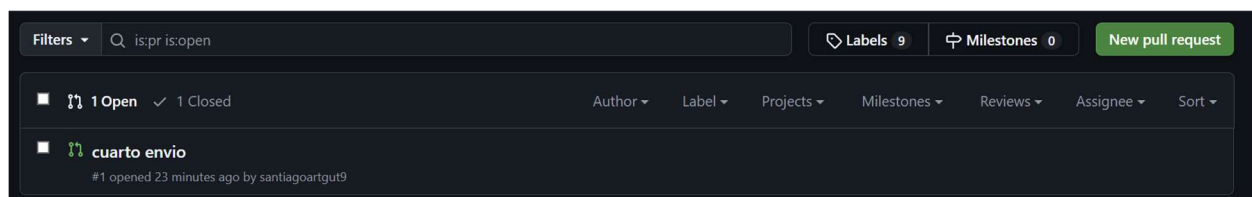


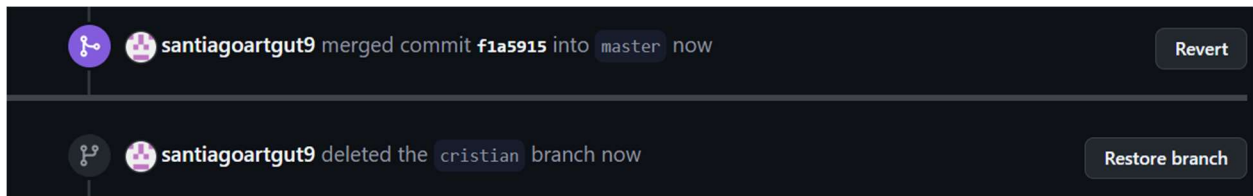
Finalmente, hago el pull request para solicitar la inserción de mis cambios a la rama master:



Ahora bien, la rama master contiene los datos modificados por mí, cuando quiero aceptar el pull request de Santiago, nos dice que efectivamente contamos con un conflicto que debe ser resuelto ya que ambos modificamos el mismo archivo y debemos revisar qué es lo que queremos que permanezca en el archivo.

Cuando Santiago acepta mi pull request, este se elimina automáticamente ya que no hubo conflictos, pero queda el pull request de él que si cuenta con conflictos:

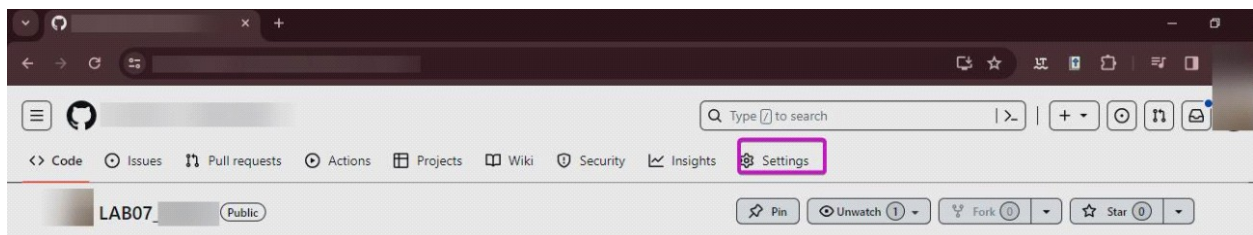




5. Teniendo en cuenta la recomendación, mezclen los cambios a la rama main a través de PR con el check/review/approval del otro compañero (Cuando se hace merge se deberían borrar las ramas en github)

### ***Como Borrar Ramas después de un Pull Request***

Se dirigen a la configuración de su repositorio:



Y en general



Se dirigen al final en el área del pull requests y seleccionan “Automatically delete head branches”

## Pull Requests

When merging pull requests, you can allow any combination of merge commits, squashes, or rebases. If you have linear history requirement enabled on any protected branch, you must be able to rebase.

### ☒ Allow merge commits

Add all commits from the head branch to the base branch with a merge commit.

#### Default commit message

Presented when merging a pull request with merge.

Default message ▾

### ☒ Allow squash merging

Combine all commits from the head branch into a single commit in the base branch.

#### Default commit message

Presented when merging a pull request with squash.

Default message ▾

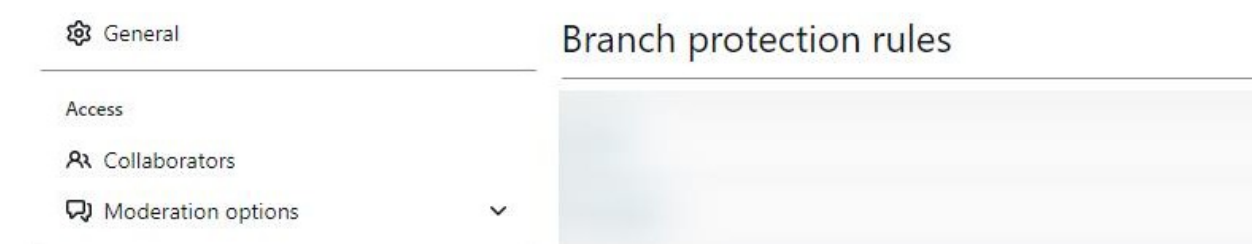
### ☒ Allow rebase merging

Add all commits from the head branch onto the base branch individually.

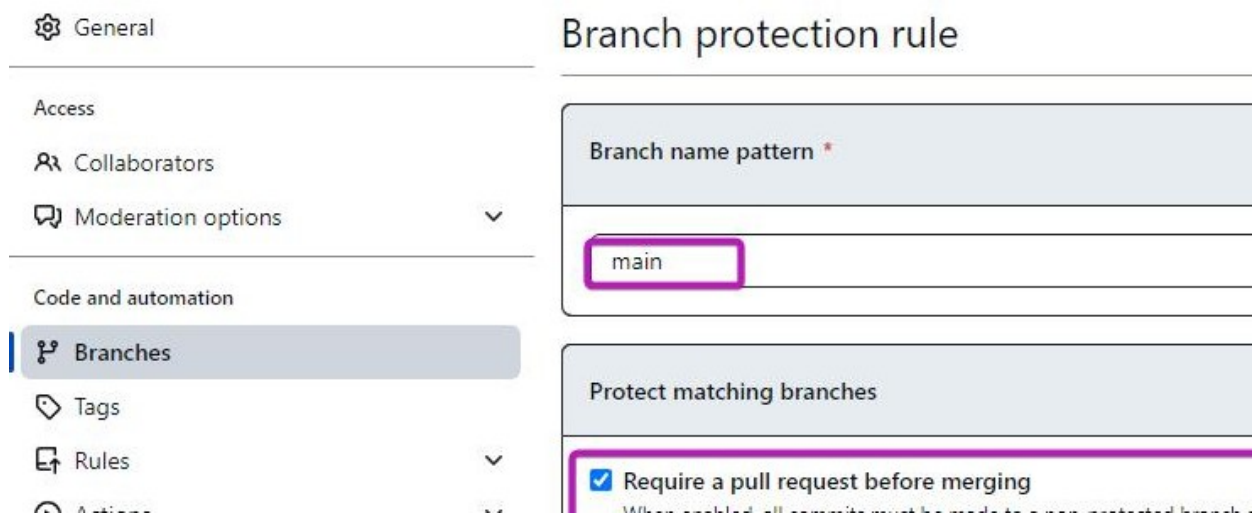
Control how and when users are prompted to update their branches if there are new commits on the base branch.

### **Aprobación Pull Request**

Nos dirigimos (todavía en configuraciones) a Branches, en esta visualizarán donde daremos protección de nuestras ramas, seleccionamos Add rule.



Aquí damos el nombre de nuestra rama (Verificar el nombre tal cual lo tenemos en nuestro repositorio) y seleccionamos la primera opción como se muestra, así estamos requiriendo que cuando se haga ese pull request en nuestra rama se necesita aprobación de otro compañero



Vamos al final y damos clic en Create.

CHOOSE WHICH ENVIRONMENTS MUST BE SUCCESSFULLY DEPLOYED TO BEFORE BRANCHES CAN BE DELETED

---

☐ **Lock branch**  
Branch is read-only. Users cannot push to the branch.

---

☐ **Do not allow bypassing the above settings**  
The above settings will apply to administrators and custom roles with the "bypass" permission.

---

**Rules applied to everyone including administrators**


---

☐ **Allow force pushes**  
Permit force pushes for all users with push access.

Y así protegimos nuestra rama principal, esto se vuelve muy relevante cuando trabajamos en parejas o en equipos, deberían tener un mensaje final que se vea así:

Branch protection rule settings saved.

---

 General **Branch protection rules**

## ENTREGA

- En un README.md colocar lo siguiente:
- Una sección llamada "INTEGRANTES" y allí colocar el listado de los integrantes del taller (máximo 2).
- Una sección llamada "RESPUESTAS" colocar las respuestas a las preguntas.
- La entrega deberá realizarla en Moodle en el espacio correspondiente a su grupo.