

Introduccion en Git

Juan Sebastian Rodriguez Blanco

Facultad De Ingenieria, Universidad De Cundinamarca

Programación

William Alexander Matallana Porras

17 de noviembre del 2024

INTRODUCCION -----	3
OBJETIVOS -----	3
SOBRE GIT -----	4
CONFIGURACION DE GIT -----	5
COMANDOS BASICOS DENTRO DE GIT -----	9
CREAR Y UTILIZAR UN REPOSITORIO EN GIT-HUB -----	12
RAMAS EN GIT -----	16
REVERSA EN GIT -----	23
ENLAZAR UN REPOSITORIO LOCAL CON UNO REMOTO YA ANTIGUAMENTE CREADO -----	28
REFERENCIAS BIBLIOGRAFICAS -----	29

INTRODUCCION

En este trabajo se va a dar una introducción sobre la herramienta GIT la cual nos permite tener nuestros proyectos en su propio repositorio lo que nos ayuda a guardarlos, compartirlos e incluso poder recuperar una recuperación de una versión anterior de nuestro trabajo , vamos a ver la introducción a la creación de repositorios, que tener en cuenta al momento de crear un repositorio y las funciones que nos brinda git-hub(el servidor en la nube), como puede ser clonar archivos directamente de un link a una carpeta creada en nuestra máquina de trabajo , crear ramas que nos permiten no afectar de manera directa al proyecto principal, entre otras.

OBJETIVOS

GENERAL

- Realizar una introducción al uso de la herramienta git para trabajos de desarrollo de software

ESPECIFICOS

- Identificar las funciones de git en el ambito de desarrollo de software
- Crear un repositorio dentro de git con el fin de verificar su uso y confiabilidad
- Realizar pruebas y análisis de las funciones de git
-

SOBRE GIT

Git es un sistema de control de versiones distribuido desarrollado por Linus Torvalds en 2005 con el propósito de gestionar el desarrollo del kernel de Linux. Su objetivo principal es permitir a los desarrolladores realizar un seguimiento de los cambios en su código, revertir modificaciones cuando sea necesario y trabajar de manera colaborativa sin el riesgo de sobrescribir los cambios de otros colaboradores (Torvalds, 2005).

El uso de Git se basa en comandos que permiten inicializar un repositorio, agregar archivos, confirmar cambios y sincronizar el código con repositorios remotos. Estas funcionalidades facilitan la gestión eficiente de proyectos de software, permitiendo que múltiples desarrolladores trabajen en paralelo sin generar conflictos.

Funciones clave de Git

- **Acceso y monitoreo de cambios** dentro del proyecto, lo que permite revisar el historial de modificaciones y restaurar versiones anteriores si es necesario.
- **Uso de ramas (*branches*)**, las cuales permiten trabajar en nuevas funcionalidades o corregir errores sin afectar el proyecto principal. Esto facilita el desarrollo ágil y la integración continua.
- **Creación de repositorios remotos**, alojados en la nube a través de plataformas como GitHub, GitLab o Bitbucket. Esto permite colaborar con otros desarrolladores, clonar proyectos y mantener copias seguras del código.

Gracias a su eficiencia y flexibilidad, Git se ha convertido en una herramienta estándar en el desarrollo de software moderno, siendo ampliamente utilizada en proyectos de código abierto y en la industria tecnológica

CONFIGURACION DE GIT

- Git config -- list

```
C:\Users\pc>git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=schannel
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=sjjrodriguez
user.email=sjjrodriguez120@gmail.com
```

Este comando nos permite ver las configuraciones de git en nuestro dispositivo y estas son las configuraciones dentro de git-hub

- diff.astextplain.textconv=astextplain

convierte archivos binarios en archivos de texto

- filter.lfs.clean=git-lfs clean -- %f

filter.lfs.smudge=git-lfs smudge -- %f

```
filter.lfs.process=git-lfs filter-process
```

```
filter.lfs.required=true
```

estas líneas de código permite almacenar archivos de gran tamaño almacenando referencias del archivo en vez del archivo en si

```
http.sslbackend=schannel
```

Usa el backend **Schannel** de Windows para manejar SSL, en lugar de OpenSSL.

```
core.autocrlf=true
```

Convierte automáticamente los finales de línea (\n y \r\n) dependiendo del sistema operativo.

```
core.fscache=true
```

Activa la caché del sistema de archivos para mejorar el rendimiento

```
core.symlinks=false
```

Desactiva los enlaces simbólicos en Windows, útil si el sistema no los soporta bien.

```
pull.rebase=false
```

permite usar merge en lugar de rebase al hacer git pull

```
credential.helper=manager
```

Usa el **Git Credential Manager** para almacenar credenciales.

```
credential.https://dev.azure.com.usehttppath=true
```

Permite autenticación basada en rutas para Azure DevOps.

`init.defaultbranch=master`

Define master como el nombre del Branch por defecto en repositorios nuevos (en lugar de main)

Configurar registro dentro del dispositivo local

- `Git config -- global user.name` (nos permite ver que nombre tenemos registrado en el local)
- `Git config -- global user.email` (nos permite ver que correo tenemos registrado en el local)
-

```
C:\Users\pc>git config -- user.name  
sjjrodriguez  
  
C:\Users\pc>git config -- user.email  
sjrodriguez120@gmail.com
```

Eliminar registro anterior del dispositivo local

Al momento de ingresar y ves que no son tus datos de usuario de git para quitarlos usamos los siguientes comandos

- `Git config -- global -- unset user.name` (Eliminar el nombre registrado en nuestro git local)
- `Git config -- global -- unset user.email` (Eliminar el nombre registrado en nuestro git local)

```
C:\Users\pc>git config --global --unset user.name  
  
C:\Users\pc>git config --global --unset user.email
```

- Aplicamos git config --list para comprobar y verificaremos que no aparezcan en la parte final el nombre y email que estaban anteriormente

```
C:\Users\pc>git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=schannel
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master

C:\Users\pc>
```

Registrarse en el dispositivo local

Para registrar nuestro nombre y correo utilizaremos los siguientes comandos

- Git config --global user.name "nombre de la cuenta de git-hun"
(este comando nos sirve para asignar un nombre de usuario)
- Git config --global user.email "correo "
(este comando nos sirve para asignar un nombre de usuario)

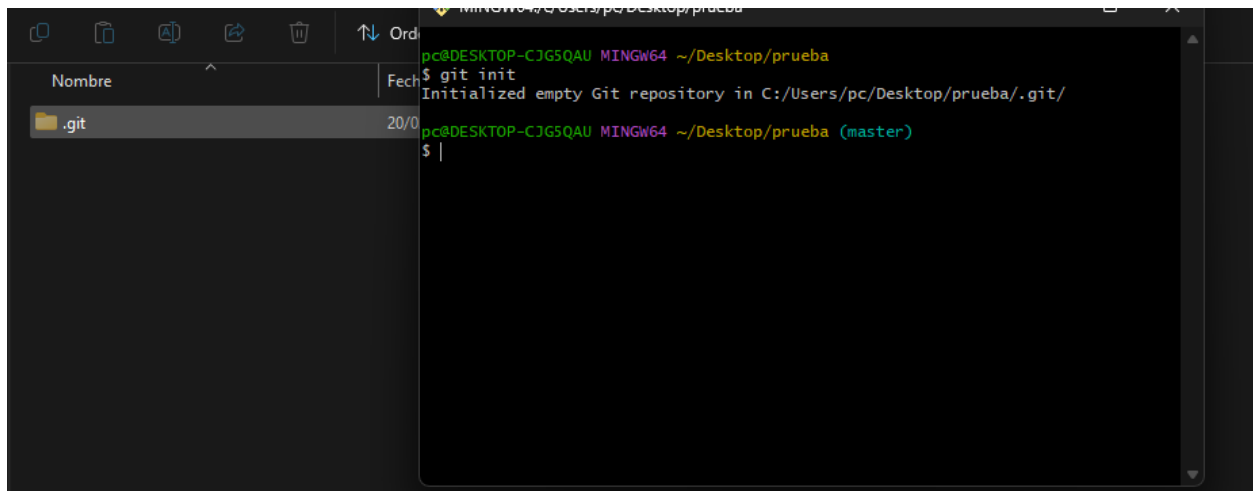
```
C:\Users\pc>git config --global user.name sjjrodriguez
C:\Users\pc>git config --global user.email sjrodriguez120@gmail.com
```


COMANDOS BASICOS DENTRO DE GIT

Son los comandos para crear e iniciar git solo haciendo uso de la git BASH

- Git init

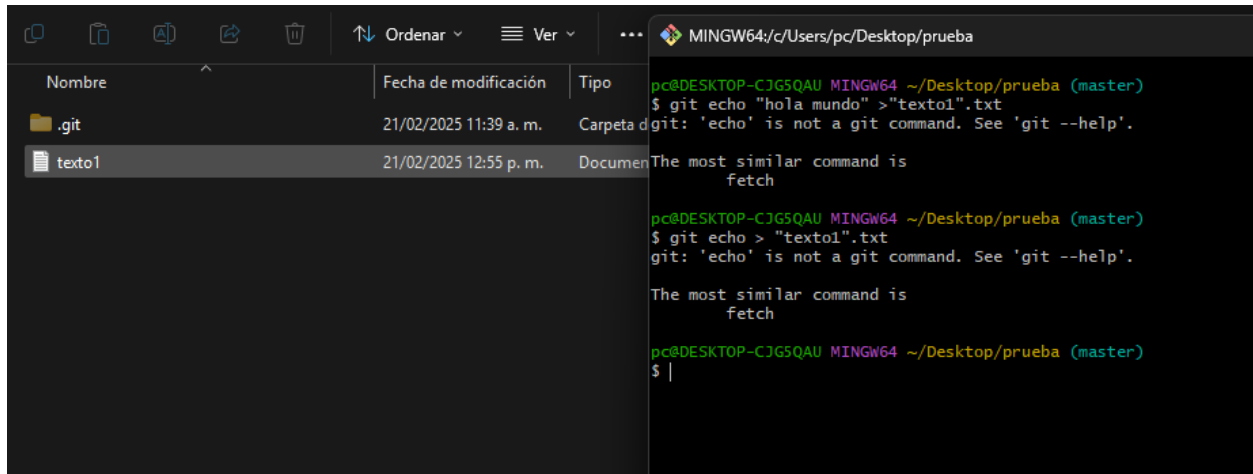
Esta sirve para crear la carpeta git y inicializar un proyecto de forma local



```
pc@DESKTOP-CJG5QAU MINGW64 ~/Desktop/prueba
$ git init
Initialized empty Git repository in C:/Users/pc/Desktop/prueba/.git/
pc@DESKTOP-CJG5QAU MINGW64 ~/Desktop/prueba (master)
$ |
```

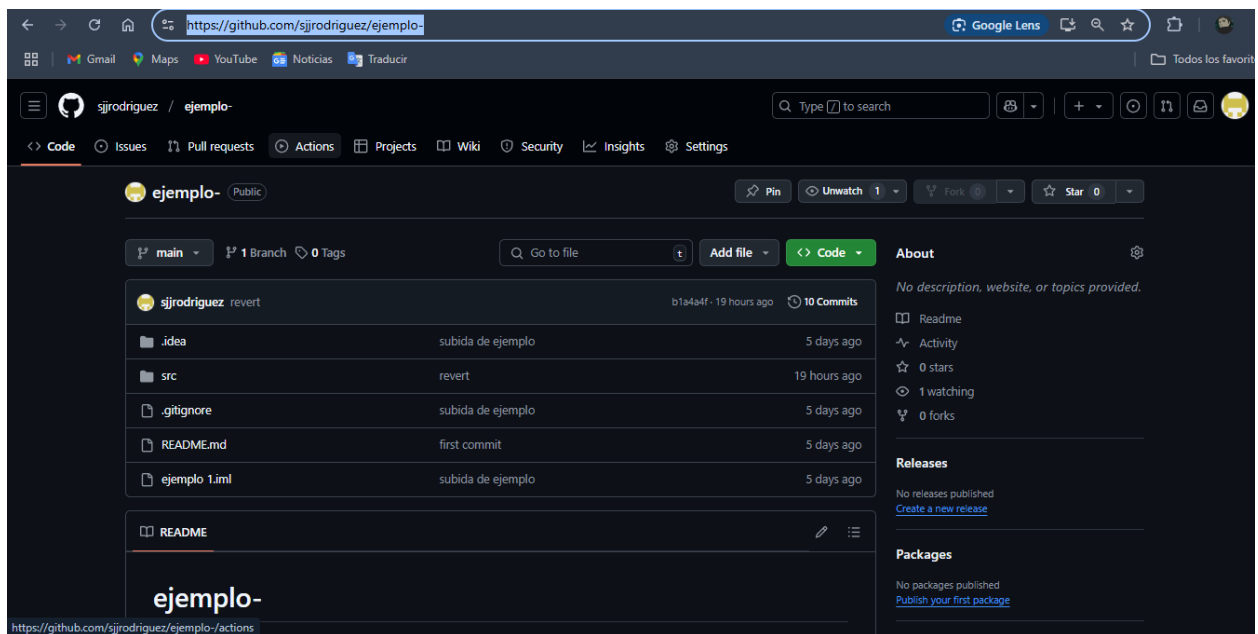
Para crear un archivo desde nuestra git bash usamos el siguiente comando

- Git echo “contenido del archivo” >”nombre archivo”. (txt.java.py...etc)



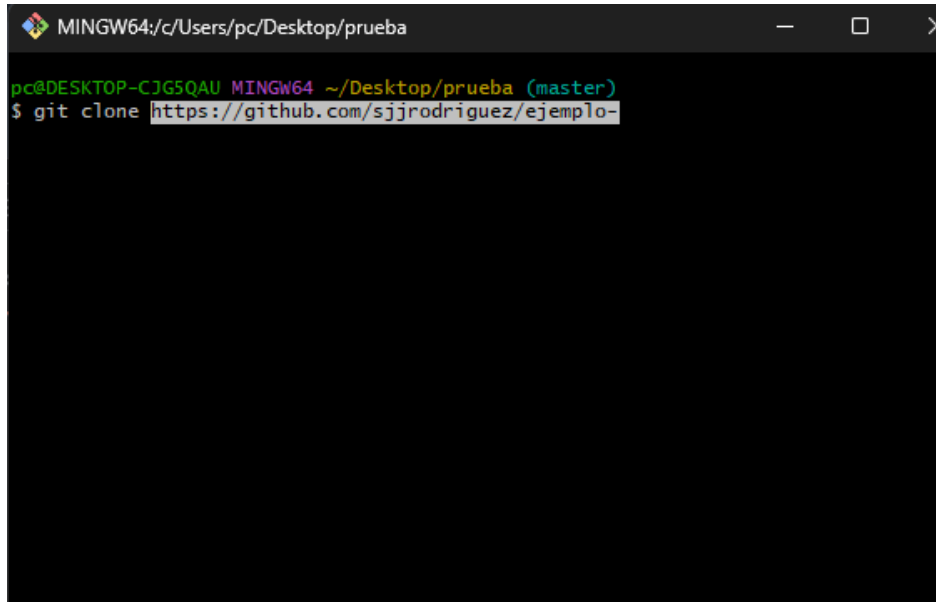
Para clonar repositorios de git-hub utilizamos git clone “link del repositorio git hub”

Por ejemplo tenemos este repositorio dentro de git-hub



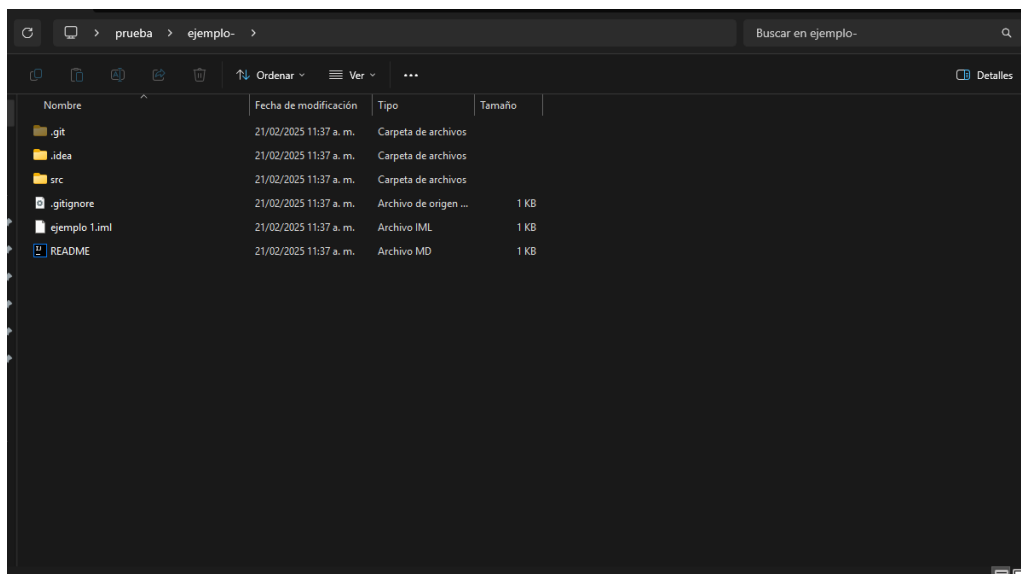
En nuestra git bash vamos a poner el comando

- Git clone “link del repositorio”



A screenshot of a MINGW64 terminal window. The title bar shows the path "MINGW64:/c/Users/pc/Desktop/prueba". The terminal prompt is "pc@DESKTOP-CJG5QAU MINGW64 ~/Desktop/prueba (master)". The command being entered is "\$ git clone https://github.com/sjjrodriguez/ejemplo-".


Y nos clonara todos los archivos del repositorio de git-hub en nuestra carpeta



Para verificar que tengamos al día los archivos del repositorio remoto y el repositorio local utilizamos

CREAR Y UTILIZAR UN REPOSITORIO EN GIT-HUB

- Creamos proyecto en nuestro IDE (entorno de desarrollo integrado)

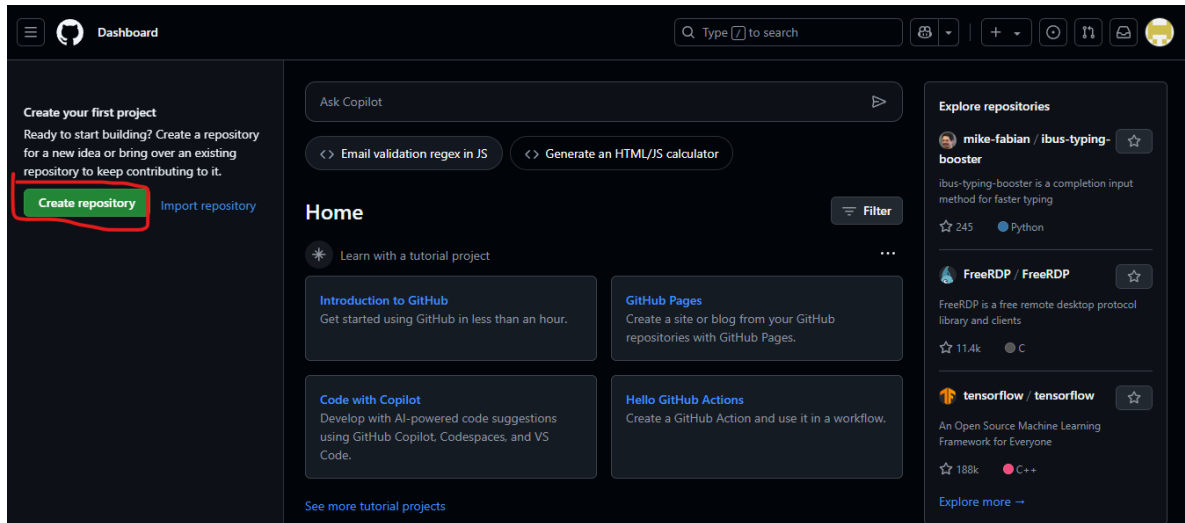


The screenshot shows an IDE window with a file named 'Main.java'. The code is as follows:

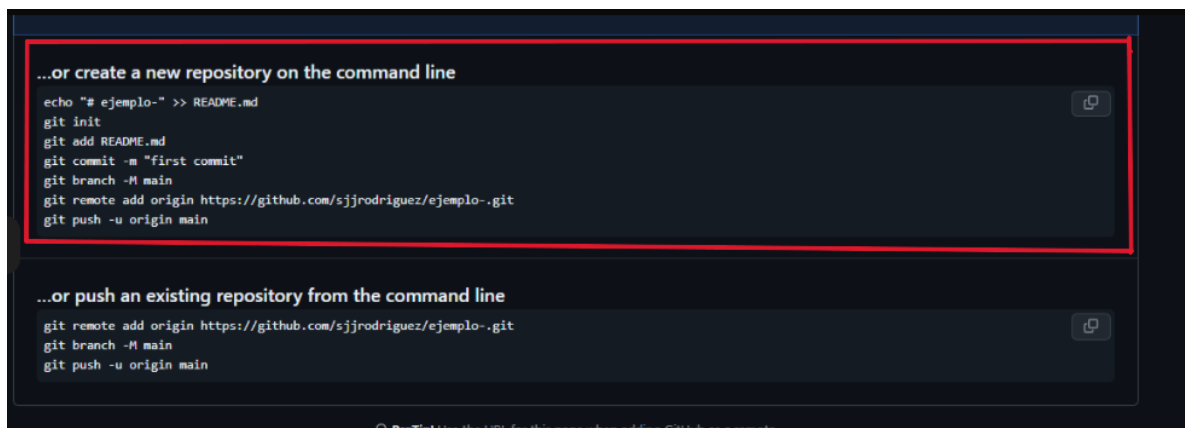
```
1 public class Main {  
2     public static void main(String[] args) {  
3         System.out.println("ejemplo Git");  
4     }  
5 }
```

Below the code editor is a 'Run' panel. It shows the command executed: `"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.3.2.2\lib\idea_rt.jar=54164:C:\ejemplo Git"`. The output is `Process finished with exit code 0`. The status bar at the bottom indicates the file is at `ejemplo 1 > src > Main` with encoding `UTF-8` and 4 spaces.

- Crear un repositorio en tu Dashboard de git-hub opción “créate repository”



- Al crear el repositorio en git-hub te saldrán dos códigos uno te servirá para crear un repositorio desde cero y el otro te va a funcionar en caso de tener un repositorio local por medio de Git y volverlo un repositorio remoto dentro de git-hub, en nuestro caso usaremos el código que nos permite crear un repositorio desde cero



- Este código lo pondremos en la terminal de nuestro proyecto

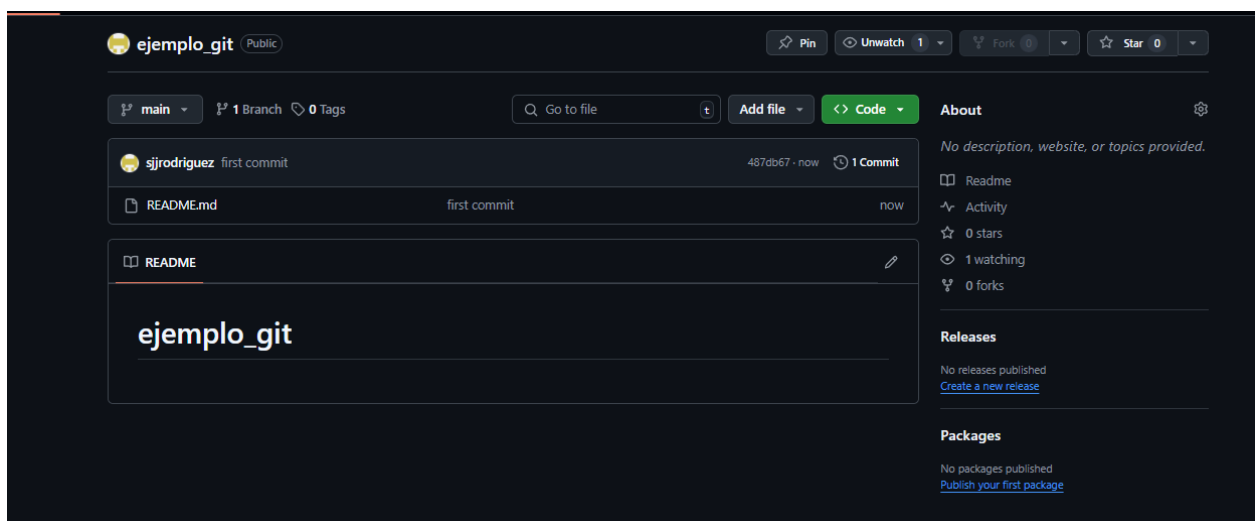


```
1 public class Main {
2     public static void main(String[] args) {
3
4         System.out.printf("ejemplo git");
5     }
6 }
7
8 }
```

Terminal Local x + v

Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 244 bytes | 122.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/sjjrodriguez/ejemplo_git.git
* [new branch] main -> main
branch 'main' set up to track 'origin/main'.
PS C:\Users\pc\IdeaProjects\ejemplo>

- Una vez hecho esto ya encontraremos enlazado nuestro repositorio en git-hub



En este repositorio aun no encontramos los archivos de nuestro programa para eso utilizamos los siguientes comandos

- Git status
- (este nos permite ver los archivos que tenemos en nuestro repositorio local y no en el remoto e viceversa

```

branch main set up to track origin/main .
PS C:\Users\pc\IdeaProjects\ejemplo> git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        .idea/
        ejemplo.iml
        src/

```

- Git add .

(nos permite seleccionar todos los archivos que falten en nuestro repositorio remoto)

```

PS C:\Users\pc\IdeaProjects\ejemplo> git add .
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it

```

- Git commit -m “mensaje”

(nos permite crear un mensaje de actualización que se mostrara en el repositorio con el subíndice -m de mensaje)

```

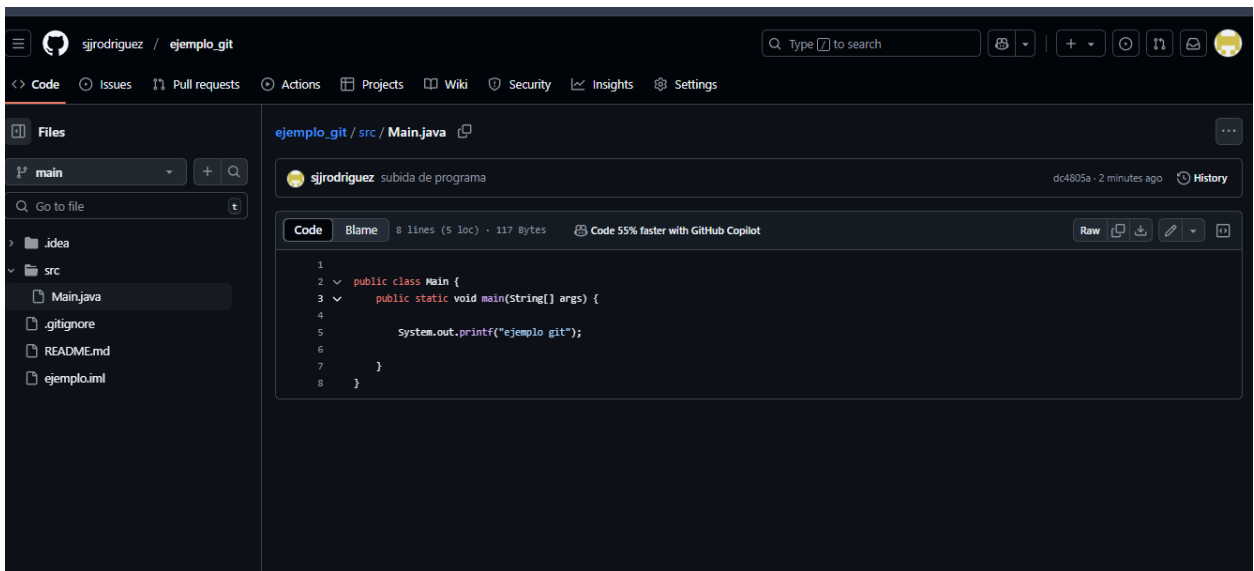
PS C:\Users\pc\IdeaProjects\ejemplo> git commit -m "subida de programa"
[main dc4805a] subida de programa
7 files changed, 71 insertions(+)
create mode 100644 .gitignore
create mode 100644 .idea/.gitignore
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 ejemplo.iml
create mode 100644 src/Main.java

```

- Git push origin main
- (este nos permite subir los archivos directamente a nuestro repositorio de git-hub)

```
create mode 100644 src/main.java
PS C:\Users\pc\IdeaProjects\ejemplo> git push origin main
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 4 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (11/11), 1.67 KiB | 342.00 KiB/s, done.
Total 11 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
```

Una vez hecho lo anterior en nuestro git-hub ya podremos encontrar los archivos en nuestro repositorio



RAMAS EN GIT

Ramas dentro de git-hub las ramas o más conocidas en git-hub como “Branches” nos permiten crear una extensión del código sin afectar directamente a la rama principal “main” a continuación vamos a ver cómo crear y usar las ramas de git-hub.

Para crear una rama usaremos los siguientes comandos

Git switch -c “nombre de la rama” (este nos permite crear una rama con el subíndice -c)


```
PS C:\Users\pc\IdeaProjects\ejemplo> git switch -c rama1
Switched to a new branch 'rama1'
PS C:\Users\pc\IdeaProjects\ejemplo> 
```

Para confirmar que tenemos creados nuestro repositorio usamos

- Git Branch (nos sirve para ver las ramas que tenemos en nuestro repositorio local y en que rama estamos trabajando actualmente)

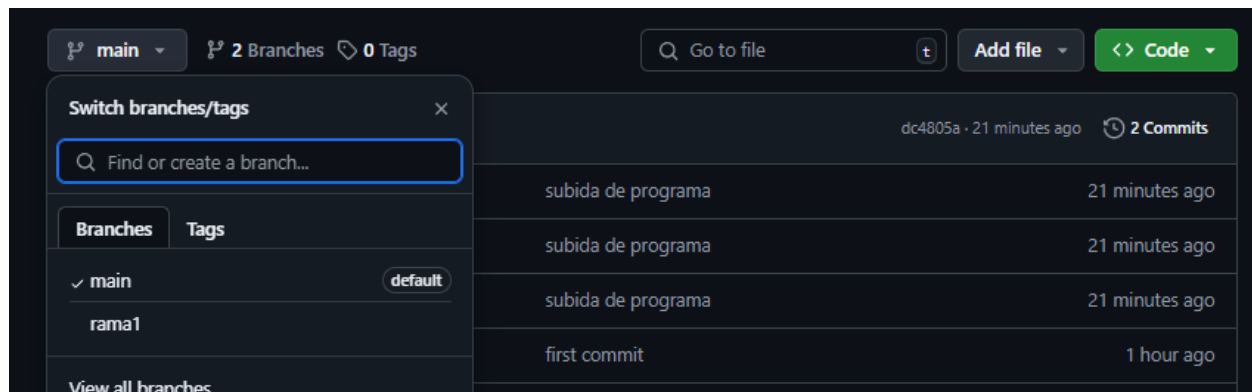
```
PS C:\Users\pc\IdeaProjects\ejemplo> git branch
main
* rama1
PS C:\Users\pc\IdeaProjects\ejemplo> 
```

Para guardar los cambios realizados en nuestro repositorio local utilizamos el siguiente comando

- Git push origin “nombre de la rama creada”

```
PS C:\Users\pc\IdeaProjects\ejemplo> git push origin rama1
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'rama1' on GitHub by visiting:
remote:   https://github.com/sjjrodriguez/ejemplo\_git/pull/new/rama1
remote:
To https://github.com/sjjrodriguez/ejemplo\_git.git
 * [new branch]      rama1 -> rama1
PS C:\Users\pc\IdeaProjects\ejemplo> 
```

Y ya aparecería dentro de nuestro git-hub



- Ahora realizamos un cambio a nuestro programa

```
1 public class Main {  
2     public static void main(String[] args) {  
3         System.out.printf("ejemplo git");  
4         System.out.println("hola mundo");  
5     }  
6 }  
7 }
```

Después de hacer un cambio en nuestro repositorio vamos a utilizar el mismo código para subir los archivos

- Git status
- Git add .
- Git commit -m "mensaje"

Pero en este caso vamos a especificar el nombre de la rama a la cual le vamos a subir los cambios

- git push origin "nombre de la rama"

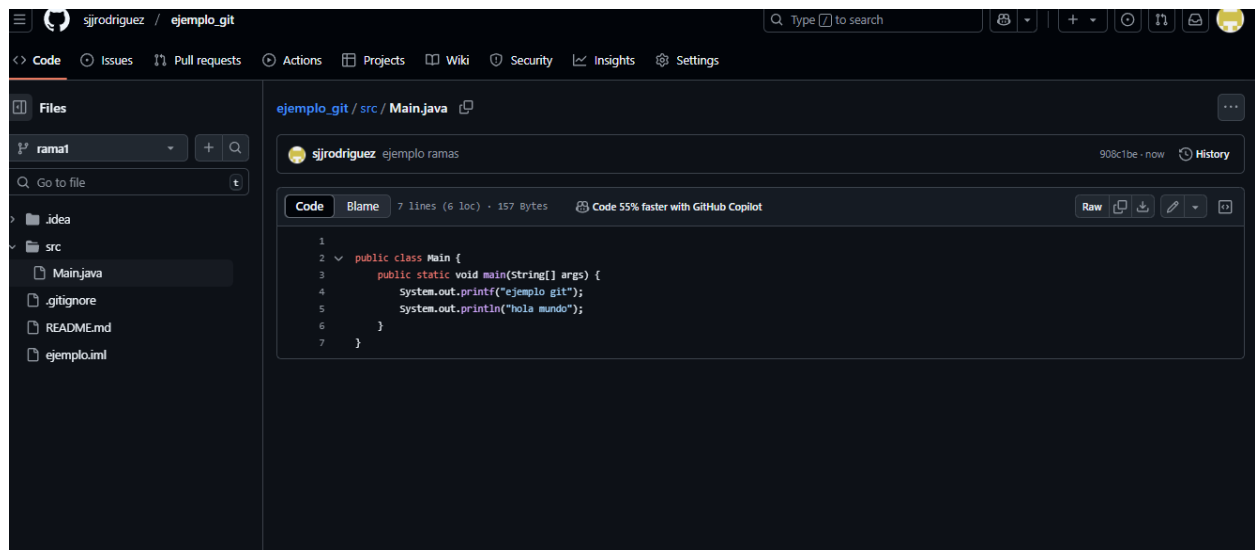
```

PS C:\Users\pc\IdeaProjects\ejemplo> git status
On branch rama1
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/Main.java

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\pc\IdeaProjects\ejemplo> git add .
warning: in the working copy of 'src/Main.java', LF will be replaced by CRLF the next time Git
PS C:\Users\pc\IdeaProjects\ejemplo> git commit -m "ejemplo ramas"
[rama1 908c1be] ejemplo ramas
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 376 bytes | 188.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/sjrodriguez/ejemplo_git.git
   dc4805a..908c1be  rama1 -> rama1
PS C:\Users\pc\IdeaProjects\ejemplo>

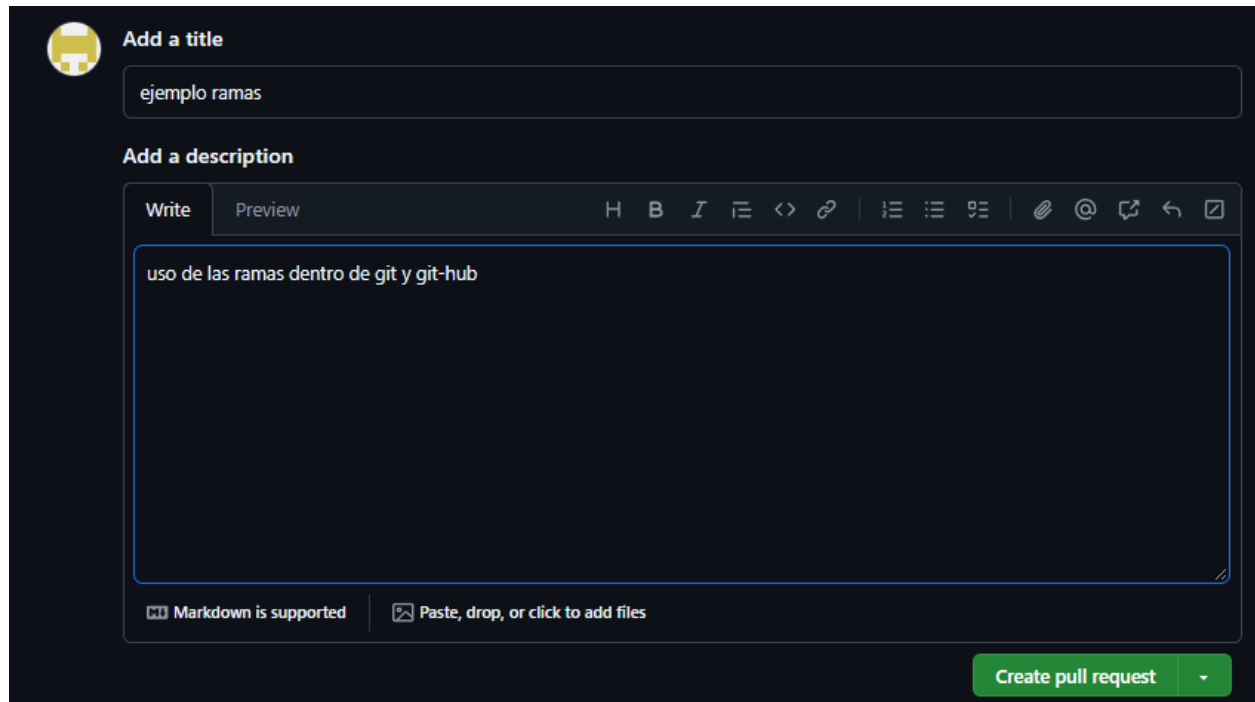
```

En nuestra rama ya encontraremos los cambios



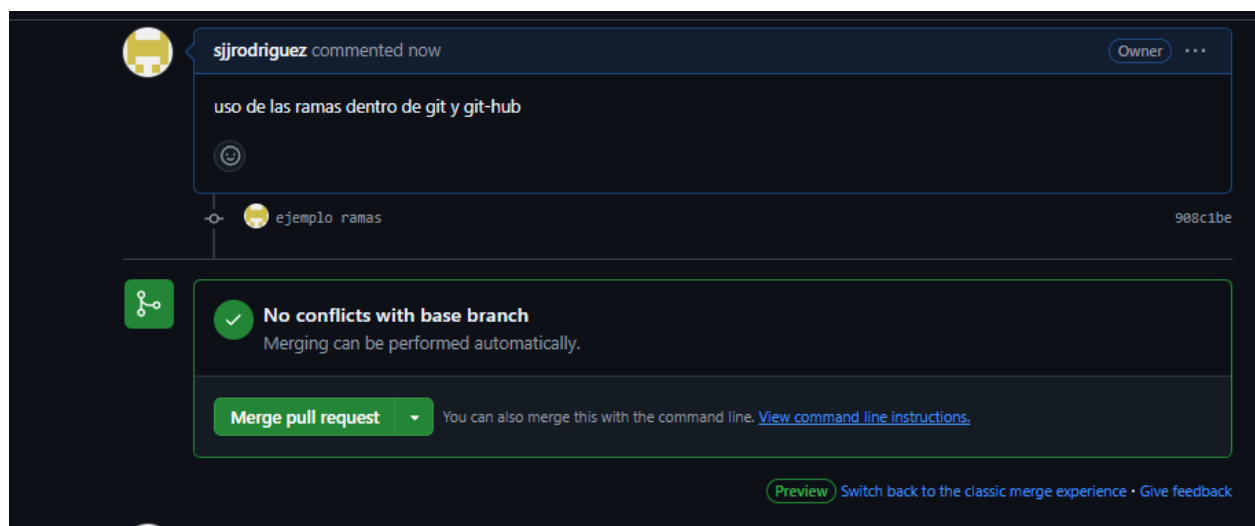
Para subir nuestro trabajo y unir directamente nuestra sub rama a la rama “Main” en nuestro git-hub encontraremos una opción de “compare y pull request” esta en un trabajo colaborativo nos

permite ver los cambios de un compañero y revisarlos antes de combinar y guardar el código de la rama “Main” al momento de mandar el pull request a los colaboradores les va a aparecer esto



The screenshot shows the GitHub interface for creating a pull request. At the top, there's a header with a repository icon and the text "Add a title". Below this is a text input field containing "ejemplo ramas". Underneath is the "Add a description" section, which has a "Write" tab and a "Preview" tab. The "Write" tab is active, showing a rich text editor with the text "uso de las ramas dentro de git y git-hub". The editor has a toolbar with various formatting options like bold, italic, link, and list. At the bottom right of the editor, there's a green button that says "Create pull request".

A nuestros colaboradores o equipo de trabajo les va a aparecer esto una vez que acepten el trabajo de la subrama “rama1” se subirá a la rama principal



The screenshot shows the GitHub interface for merging a pull request. At the top, there's a header with a repository icon and the text "sjjrodriguez commented now". Below this is a text input field containing "uso de las ramas dentro de git y git-hub". Underneath is a section with a green checkmark and the text "No conflicts with base branch". Below this is a green button that says "Merge pull request". At the bottom right, there's a green button that says "Preview" and a link that says "Switch back to the classic merge experience".

Despues de aceptar volveremos a la rama principal y usando el siguiente comando

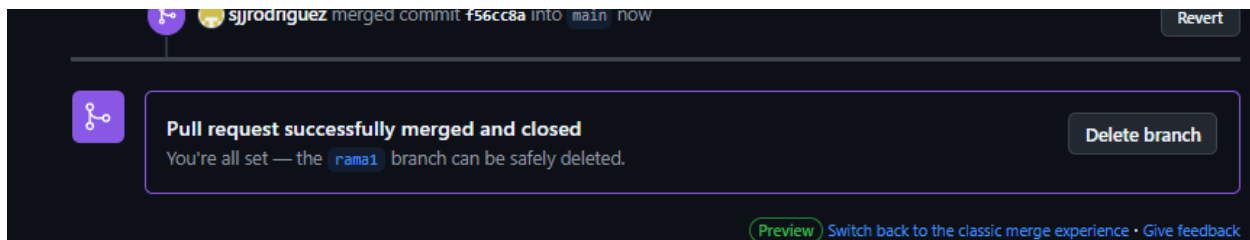
- git switch main (nos permite cambiar la rama que estamos usando)

y así mismo usaremos(git Branch) para verificarlo

```
PS C:\Users\pc\IdeaProjects\ejemplo> git switch main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\pc\IdeaProjects\ejemplo> git branch
* main
  rama1
PS C:\Users\pc\IdeaProjects\ejemplo> |
```

Ahora para eliminar la rama tenemos que eliminarla de manera local y de manera remota

Para eliminar una rama de manera remota una vez que todos los colaboradores acepten va a aparecerá una opción de eliminar la rama “delete Branch”

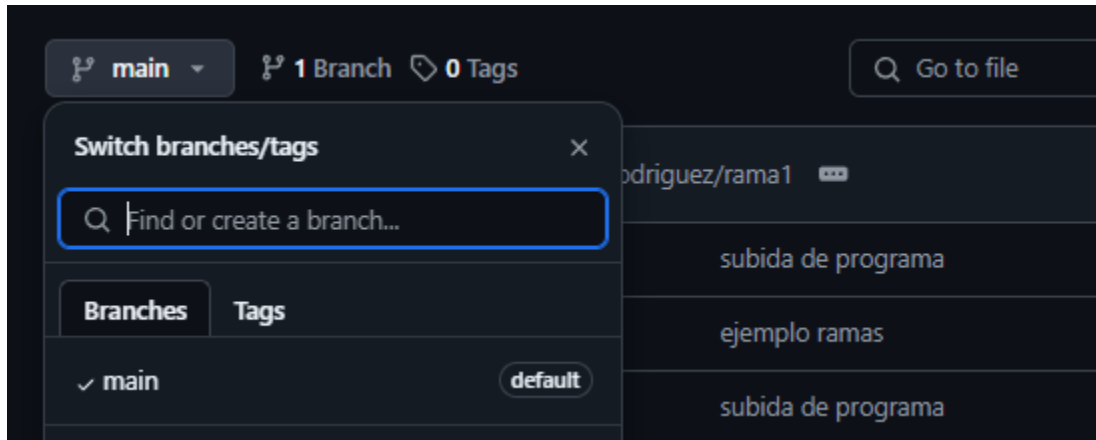


Pero para hacerla de forma remota escribiremos el siguiente comando

- Git push origin - - delete “nombre de la rama”

```
PS C:\Users\pc\IdeaProjects\ejemplo> git push origin --delete rama1
To https://github.com/sjjrodriguez/ejemplo_git.git
- [deleted]          rama1
PS C:\Users\pc\IdeaProjects\ejemplo> |
```

Y notaremos como en nuestro git-hub ya la rama no existe



Para eliminar la rama local solo escribimos el siguiente comando

- `Git Branch -D “nombre de la rama”` (usamos -D mayúscula para forzar el borrado)

Y usamos (`git branch`) para verificarlo

```
PS C:\Users\pc\IdeaProjects\ejemplo> git branch -D rama1
Deleted branch rama1 (was 908c1be).
PS C:\Users\pc\IdeaProjects\ejemplo> git branch
* main
```

Ahora usamos el comando

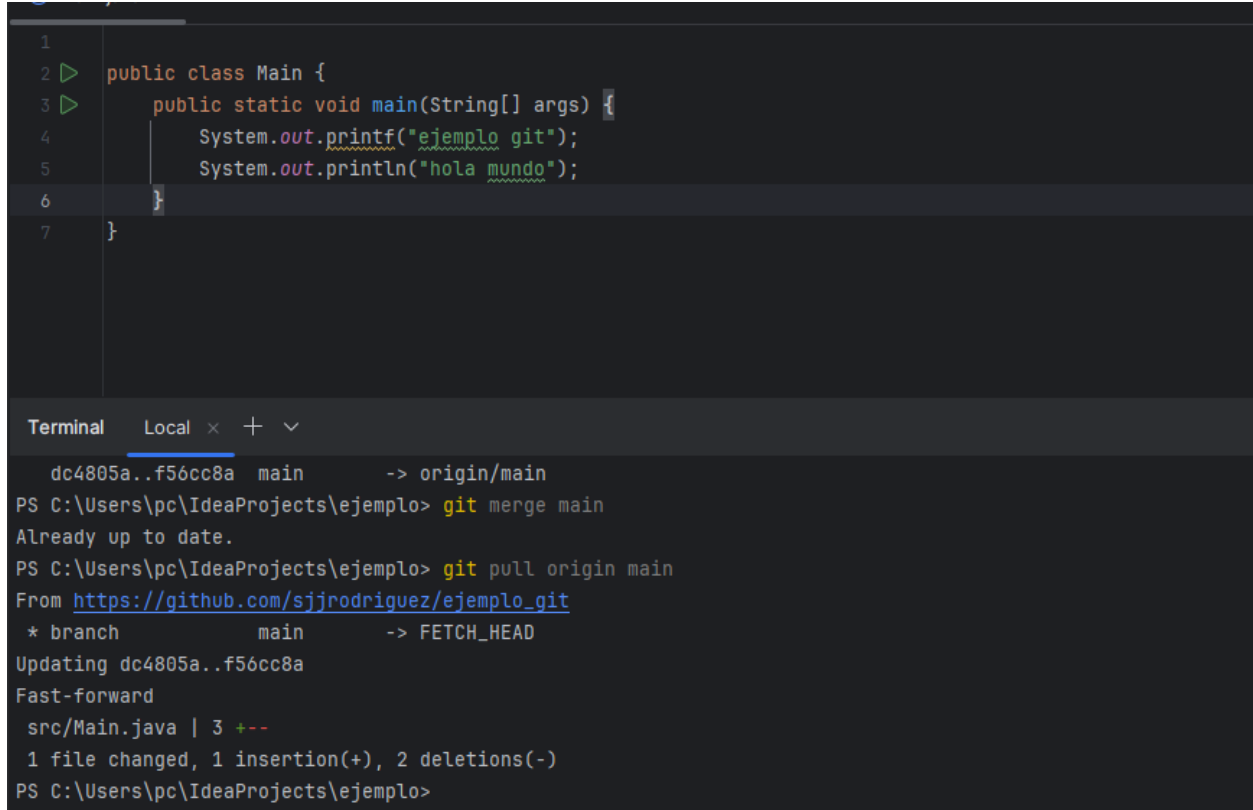
Para combinar ramas de manera local solo

- `Git merge main`

Y para bajar los archivos que esten guardados en la rama main de nuestro repositorio remoto solo usamos

- `git pull origin main`

y nos baja el programa de la rama main



The screenshot shows an IDE with a Java file and a terminal window. The Java file contains the following code:

```
1  
2 public class Main {  
3     public static void main(String[] args) {  
4         System.out.printf("ejemplo git");  
5         System.out.println("hola mundo");  
6     }  
7 }
```

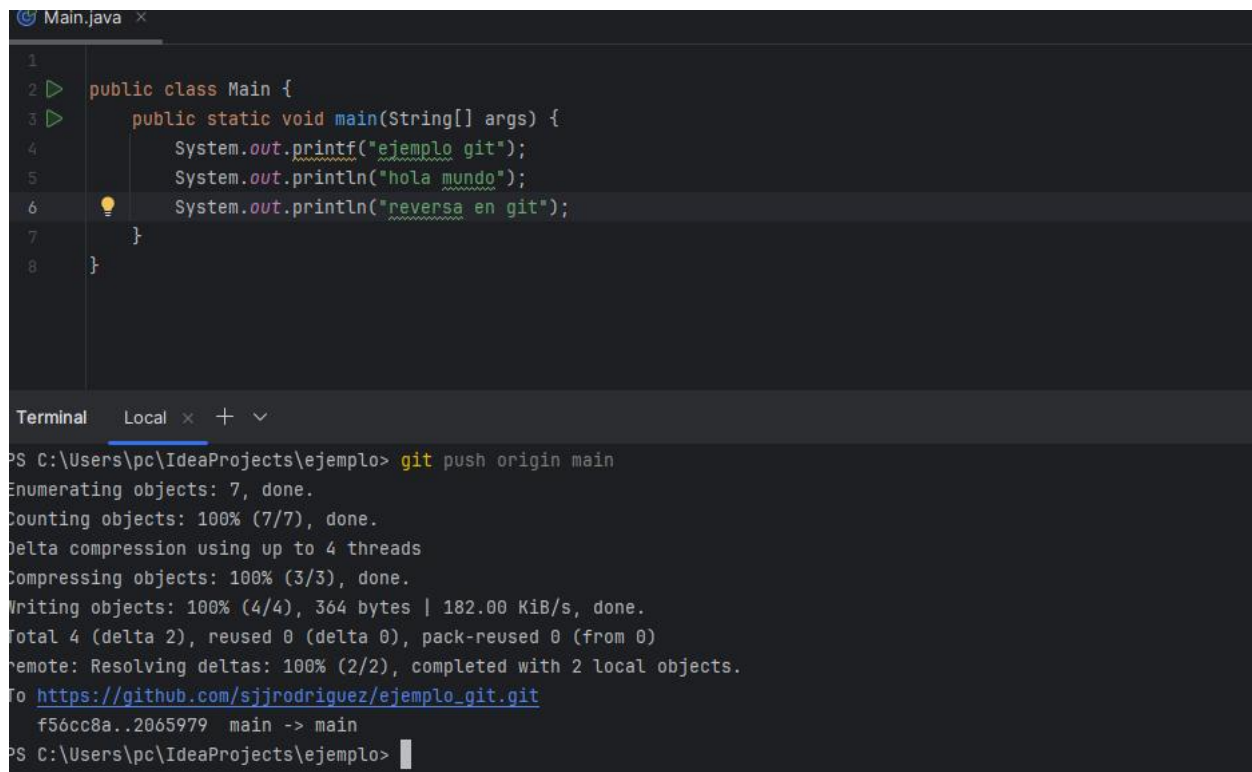
The terminal window shows the following commands and output:

```
dc4805a..f56cc8a  main    -> origin/main  
PS C:\Users\pc\IdeaProjects\ejemplo> git merge main  
Already up to date.  
PS C:\Users\pc\IdeaProjects\ejemplo> git pull origin main  
From https://github.com/sjjrodriguez/ejemplo-git  
* branch      main      -> FETCH_HEAD  
Updating dc4805a..f56cc8a  
Fast-forward  
 src/Main.java | 3 +--  
 1 file changed, 1 insertion(+), 2 deletions(-)  
PS C:\Users\pc\IdeaProjects\ejemplo>
```

REVERSA EN GIT

La reversa en git la aplicamos si queremos devolvernos un paso o varios paso sin que todos los cambios se pierdan , nos sirve como una línea de tiempo para crear y hacer cambios en nuestro programa

Para aplicar vamos a hacer un cambio en el programa y subirlo con un commit



```
1 public class Main {
2     public static void main(String[] args) {
3         System.out.printf("ejemplo git");
4         System.out.println("hola mundo");
5         System.out.println("reversa en git");
6     }
7 }
8 }
```

```
PS C:\Users\pc\IdeaProjects\ejemplo> git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 364 bytes | 182.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/sjjrodriguez/ejemplo_git.git
   f56cc8a..2065979  main -> main
PS C:\Users\pc\IdeaProjects\ejemplo>
```



```
1 public class Main {
2     public static void main(String[] args) {
3         System.out.printf("ejemplo git");
4         System.out.println("hola mundo");
5         System.out.println("reversa en git");
6     }
7 }
8 }
```

Y ahora para devolvernos un paso anterior tenemos que ver la información de los commits ya hechos con algunos de los siguientes comando

- git log (muestra el historial de commits en un repositorio. Es útil para ver qué cambios se han realizado, por quién y cuándo)


```

PS C:\Users\pc\IdeaProjects\ejemplo> git log
commit 2065979d57c6f3da1463caaa5ecb87262522c6b4 (HEAD -> main, origin/main, origin/HEAD)
Author: sjjrodriguez <sjjrodriguez120@gmail.com>
Date:   Fri Feb 21 13:34:24 2025 -0500

    reversa en git

commit f56cc8aa89947a7470355f9854542f60ed9a02d1
Merge: dc4805a 908c1be
Author: sjjrodriguez <sjjrodriguez120@gmail.com>
Date:   Fri Feb 21 13:18:18 2025 -0500

    Merge pull request #1 from sjjrodriguez/rama1

ejemplo ramas

commit 908c1be789e2d3ab0655be6892ea2ba2b6b9c654
Author: sjjrodriguez <sjjrodriguez120@gmail.com>
Date:   Fri Feb 21 13:12:36 2025 -0500

    ejemplo ramas

```

- git reflog , nos sirve para ver los commit de una rama en especifico

```

PS C:\Users\pc\IdeaProjects\ejemplo> git reflog
2065979 (HEAD -> main, origin/main, origin/HEAD) HEAD@{0}: commit: reversa en git
f56cc8a HEAD@{1}: pull origin main: Fast-forward
dc4805a HEAD@{2}: checkout: moving from rama1 to main
908c1be HEAD@{3}: commit: ejemplo ramas
dc4805a HEAD@{4}: checkout: moving from main to rama1
dc4805a HEAD@{5}: commit: subida de programa
487db67 HEAD@{6}: Branch: renamed refs/heads/master to refs/heads/main
487db67 HEAD@{8}: commit (initial): first commit

```

- git log --oneline,nos sirve para ver los commit de una línea en especifico

```

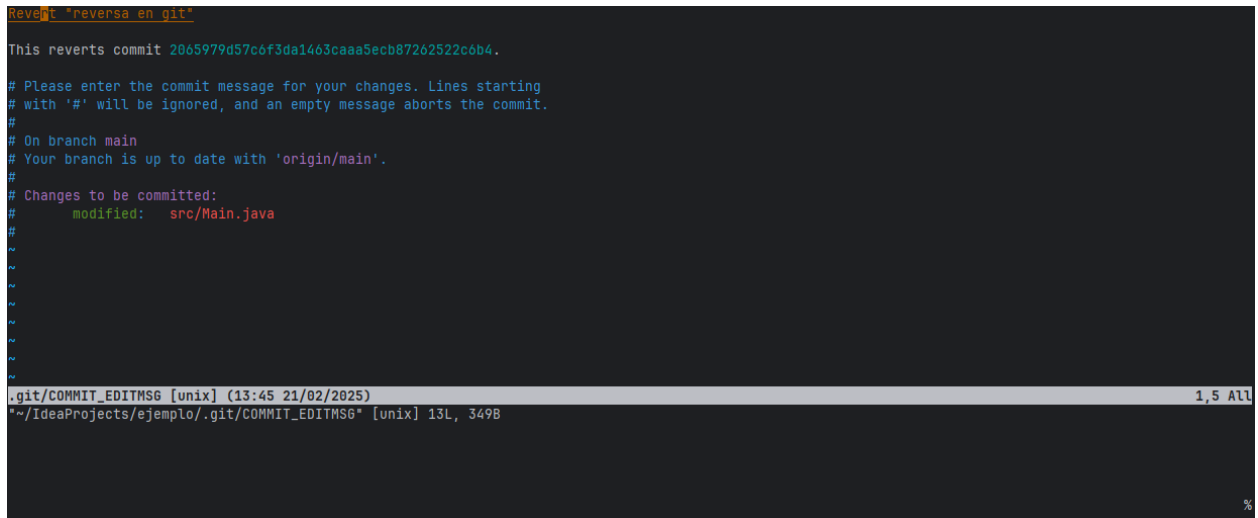
PS C:\Users\pc\IdeaProjects\ejemplo> git log --oneline
2065979 (HEAD -> main, origin/main, origin/HEAD) reversa en git
f56cc8a Merge pull request #1 from sjjrodriguez/rama1
908c1be ejemplo ramas
dc4805a subida de programa
487db67 first commit

```

En este caso usare la información de log como pueden observar nuestro ultimo commit sus 4 primeros dígitos de referencias son 2065 y entonces usamos el comando

- Git reverse 2065/(primeros 4 o más dígitos de referencia del commit)

Y nos aparecerá la siguiente, si no queremos cambiarle el nombre y solo dejar como el nombre del commit eliminado presionamos la tecla “esc” y le damos q y enter



```
ReveGit *reversa en git*

This reverts commit 2065979d57c6f3da1463caaa5ecb87262522c6b4.

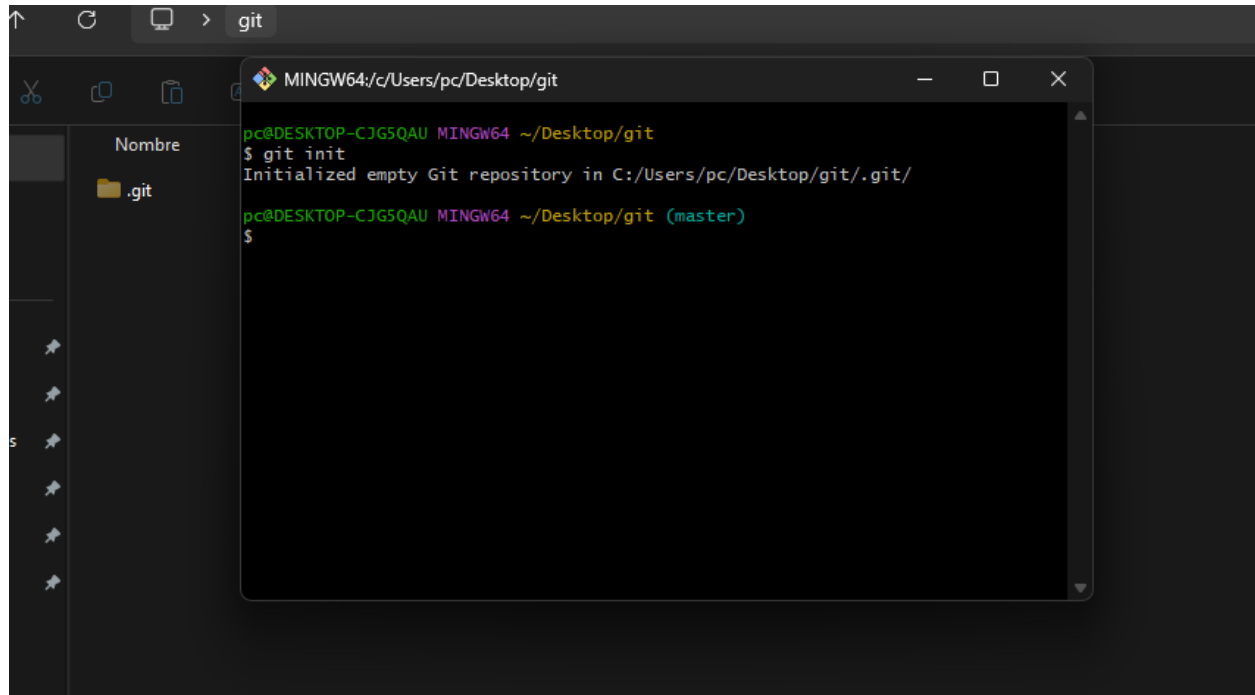
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch main
# Your branch is up to date with 'origin/main'.
#
# Changes to be committed:
#   modified:   src/Main.java
#
~
~
~
~
~
~
~
~
~
~

.git/COMMIT_EDITMSG [unix] (13:45 21/02/2025) 1,5 All
~/IdeaProjects/ejemplo/.git/COMMIT_EDITMSG* [unix] 13L, 349B
```

Si queremos cambiar el texto presionamos la tecla “a” y entre las comillas de arriba escribimos el mensaje que queremos que aparezca luego de esto presionamos “esc” le damos “w” de texto y “q” de salir

ENLAZAR UN REPOSITORIO LOCAL CON UNO REMOTO YA ANTIGUAMENTE CREADO

Creamos un repositorio local



The image shows a file explorer window on the left with a folder named '.git' under the name 'Nombre'. To the right is a terminal window titled 'MINGW64/c/Users/pc/Desktop/git'. The terminal output is as follows:

```
pc@DESKTOP-CJG5QAU MINGW64 ~/Desktop/git
$ git init
Initialized empty Git repository in C:/Users/pc/Desktop/git/.git/

pc@DESKTOP-CJG5QAU MINGW64 ~/Desktop/git (master)
$
```

Verificamos que no tengamos ningún repositorio remoto vinculado con el siguiente comando y si no muestra nada significa que no tenemos ningún repositorio remoto conectado

- Git remote -v (de detalles)

Ahora enlazaremos el repositorio local con el remoto con el siguiente comando y verificamos de nuevo con (git remote -v)

- Git remote add origin “link del repositorio”

```
pc@DESKTOP-CJG5QAU MINGW64 ~/Desktop/git (master)
$ git remote add origin https://github.com/sjjrodriguez/ejemplo-

pc@DESKTOP-CJG5QAU MINGW64 ~/Desktop/git (master)
$ git remote -v
origin  https://github.com/sjjrodriguez/ejemplo- (fetch)
origin  https://github.com/sjjrodriguez/ejemplo- (push)
```

Y así ya estarían enlazados nuestros repositorios nuevamente

REFERENCIAS BIBLIOGRAFICAS

- © 2025 GitHub, Inc.

<https://docs.github.com/es/get-started/start-your-journey/about-github-and-git>

porcentaje de IA (20%, organizar y ampliar texto)