

## **Creación y utilización de repositorios en GitHub con comandos de GIT**

Jonathan Alejandro Yacuma Rivera

Universidad de Cundinamarca Extensión Chía

Programación II

William Alexander Matallana Porras

Febrero 20, 2025

## Tabla de contenido

1.introducción.....	5
2.Objetivo General.....	5
3.Objetivos Específicos.....	5
4.Desarrollo.....	6
4.1.Instalación de los programas.....	6
4.2.Creación de un repositorio local.....	6
4.2.1. git config –list.....	7
4.2.2. git config --global user.emal.....	7
4.2.3. git config –global user.name.....	8
4.2.4. git config --global --unset user.email.....	9
4.2.5. git config –global --unset user.name.....	9
4.2.6 git clone.....	9
4.3 Creación de un proyecto Java en IntelliJ IDEA.....	10
4.4.Creacion de un repositorio GitHub.....	11
4.5.Comandos GIT para subir los datos del repositorio local al repositorio GitHub.....	13
4.5.1. git init.....	13

4.5.2. git status.....	13
4.5.3. git add . ....	13
4.5.4.git commit -m “mensaje de un cambio”.....	14
4.5.5. git push origin <b>main</b> .....	14
5.Ramas.....	15
5.1.Creacion de una rama.....	15
5.1.1. git branch.....	16
5.1.2. git switch -c <b>nombre de la rama</b> .....	16
5.1.3. git switch <b>nombre de la rama</b> .....	16
5.1.4. git push origin <b>nombre de la rama</b> .....	16
5.2 Combinación de ramas.....	17
5.2.1 git pull origin <b>nombre de la rama</b> .....	18
5.3 Eliminación o Cambios de ramas.....	18
5.3.1 git branch -D <b>nombre de la rama</b> .....	18
5.3.2 git branch -r.....	19
5.3.3 git push origin --delete <b>nombre de la rama</b> .....	19
5.3.4 git log.....	20
5.3.5git revert.....	20

5.3.6 git reflog.....	21
5.3.7 git log –oneline.....	21
5.3.8 git merge.....	22
5.3.9 git rebase.....	22
7.Consultas.....	22
8.Conclusiones.....	16
9.Referencias Bibliográficas.....	16

## 1.Introducción

En este informe se explicará con detalle todos los pasos y comandos GIT que fueron requeridos para la elaboración de un repositorio GitHub relacionado con cualquier proyecto creado con el IDE de IntelliJ IDEA, todo el seguimiento de instrucciones vendrá acompañado con la explicación de su función más su ejemplo gráfico (captura de pantalla del paso o comando GIT).

## 2.Objetivo General

Aprender a como llevar a cabo la carga de proyectos a los repositorios GitHub con el fin de compartir a un grupo o ser visto por la comunidad, todo esto bajo comandos GIT.

## 3.Objetivos Específicos

- Conocer los pasos básicos de carga de un repositorio.
- Reconocer los comandos GIT para conectar los proyectos de IntelliJ IDEA con GitHub.
- Saber cómo utilizar los repositorios compartidos.

## 4.Desarrollo

### 4.1.Instalación de los programas

Para comenzar el seguimiento de instrucciones, se debe de instalar el programa GIT que permite crear clones locales de proyectos hechos por el usuario, o también de un grupo o entidad el cual va estar vinculada.

Después se requerirá instalar un JDK (Java Development Kit) u otro lenguaje de programación que requiera el usuario, de igual manera, se necesita instalar un IDE para la creación de proyectos con el lenguaje de programación ya establecido, en este caso al ser Java una de las mejores opciones es el IDE de IntelliJ IDEA.

Para saber si tanto GIT como JDK están instalados correctamente, se escribirá en el cmd:

**:git --version** (programa GIT)

**:java --version** (JDK)

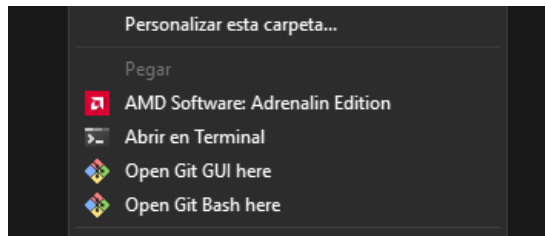
```
C:\Users\JONATHAN YACUMA> git --version
git version 2.48.1.windows.1

C:\Users\JONATHAN YACUMA>java --version
java 21.0.6 2025-01-21 LTS
Java(TM) SE Runtime Environment (build 21.0.6+8-LTS-188)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.6+8-LTS-188, mixed mode, sharing)
```

### 4.2.Creación de un repositorio local

Para crear un repositorio primero se creará una carpeta en el cual guarda todos los cambios locales que hará en el proyecto.

Después con clic derecho se dirigirá a “mostrar más opciones” y dará clic en “Open Git Bash here” donde entrará al programa GIT.



#### 4.2.1. git config --list

Permite ver todas las configuraciones que tiene el GIT.

```
JONATHAN YACUMA@LAPTOP-P5QBJV84 MINGW64 ~/Desktop/Mi proyecto (master)
$ 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
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
```

#### 4.2.2. git config --global user.email

Conecta el repositorio al correo electrónico que será utilizado, adicionalmente se digitará un git config--list para confirmar el proceso.

```

JONATHAN YACUMA@LAPTOP-P5QBJV84 MINGW64 ~/Desktop/Mi proyecto (master)
$ git config --global user.email jyacuma@ucundinamarca.edu.co

JONATHAN YACUMA@LAPTOP-P5QBJV84 MINGW64 ~/Desktop/Mi proyecto (master)
$ 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.email=jiyacuma@ucundinamarca.edu.co ←
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true

```

#### 4.2.3.git config --global user.name

Conecta el nombre de usuario GitHub al repositorio.

```

JONATHAN YACUMA@LAPTOP-P5QBJV84 MINGW64 ~/Desktop/Mi proyecto (master)
$ git config --global user.name JYacuma

JONATHAN YACUMA@LAPTOP-P5QBJV84 MINGW64 ~/Desktop/Mi proyecto (master)
$ 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.email=jiyacuma@ucundinamarca.edu.co
user.name=JYacuma ←
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true

```



#### 4.2.4. git config --global --unset user.email

Elimina el vínculo del correo electrónico con el repositorio.

```
JONATHAN YACUMA@LAPTOP-P5QBJV84 MINGW64 ~/Desktop/Mi proyecto (master)
$ git config --global --unset user.email jyacuma@ucundinamarca.edu.co
```

#### 4.2.5. git config --global --unset user.name

Elimina el vínculo del usuario de GitHub con el repositorio.

```
JONATHAN YACUMA@LAPTOP-P5QBJV84 MINGW64 ~/Desktop/Mi proyecto (master)
$ git config --global --unset user.name JYacuma
```

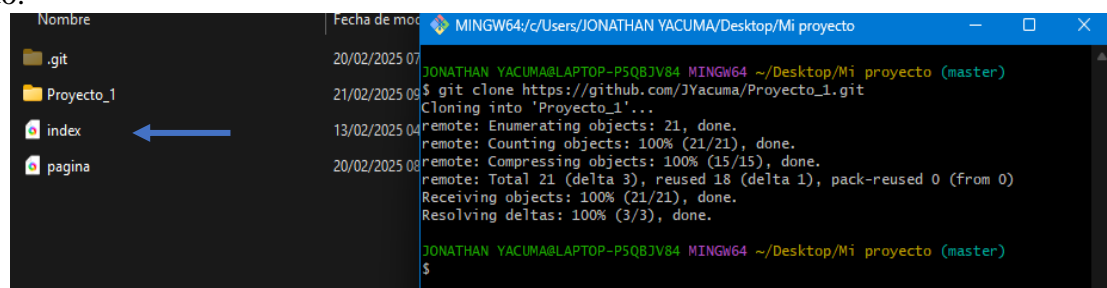
Al hacer los dos gits anteriores, no se verán reflejados al insertar git config --list

```
JONATHAN YACUMA@LAPTOP-P5QBJV84 MINGW64 ~/Desktop/Mi proyecto (master)
$ 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
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
```

#### 4.2.6. git clone

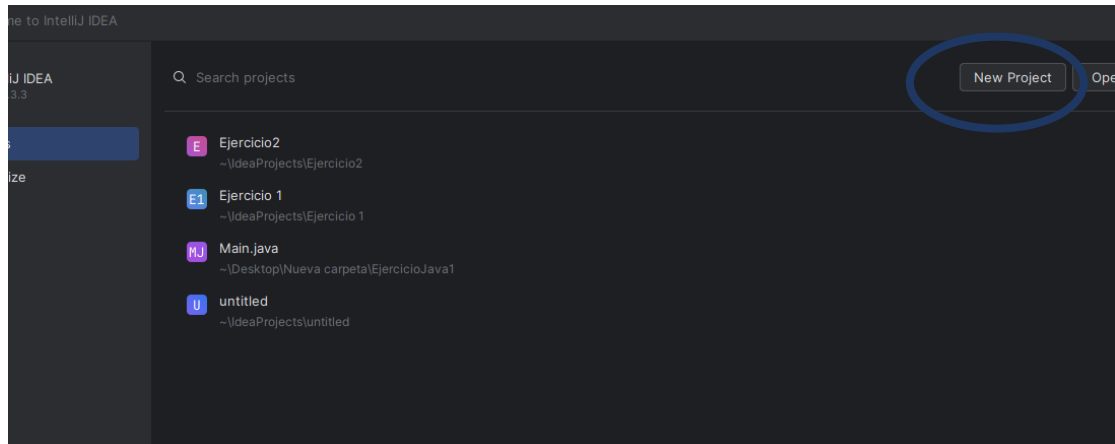
Permite descargar un archivo Java al repositorio local, esto a través del link

copiado.

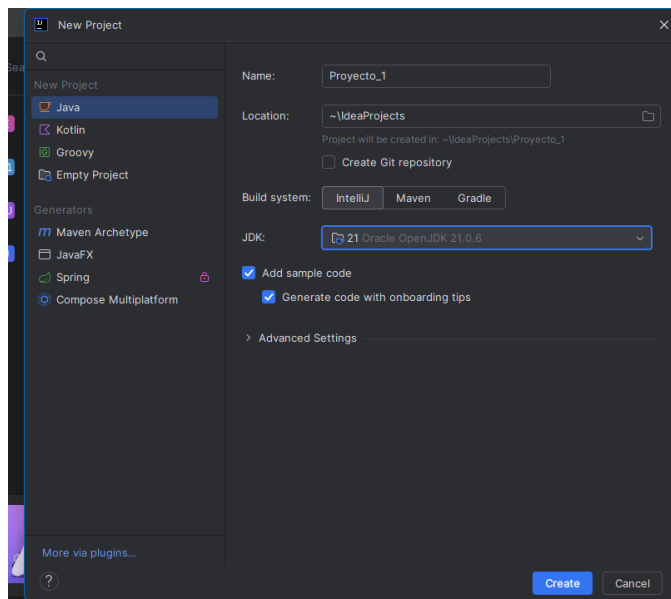


#### 4.3. Creación de un proyecto Java en IntelliJ IDEA.

Se da clic en **New Project** para crear un nuevo proyecto.



Se inserta el nombre de proyecto y se confirma que este con lenguaje de programación requerido, en este caso Java, se da clic en **Create** para confirmar.



Al ser creado el nuevo proyecto se realizará cualquier programa java que sea requerido, todo lo que se haga se guardara de manera local gracias al comando GIT que hizo.



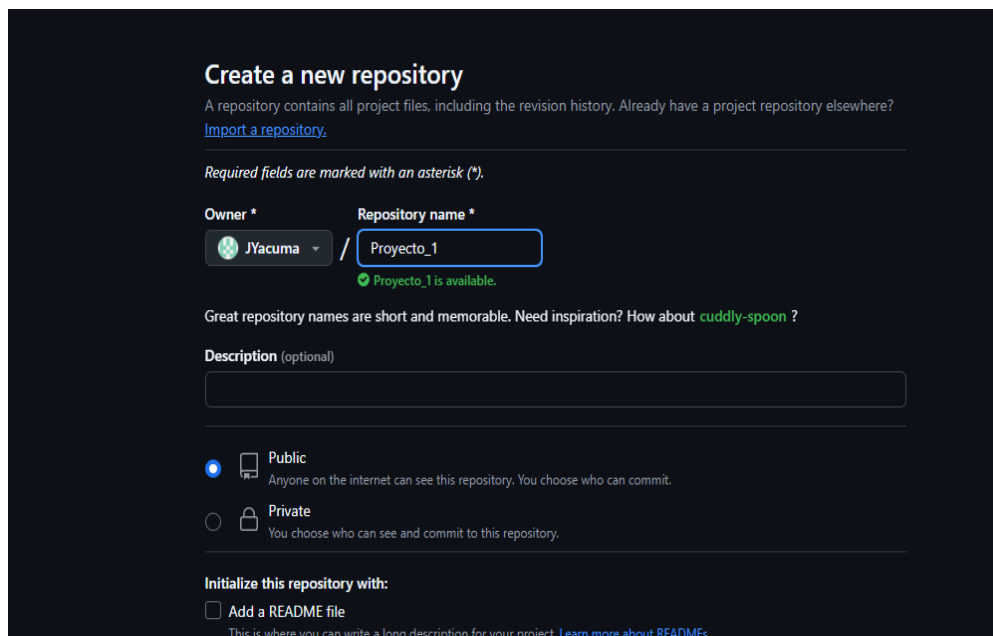
```

Main.java
To Run code, press Mayús F10 or click the ▶ icon in the gutter.
3 public class Main {
4     public static void main(String[] args) {
5
6         System.out.println("Se creo un nuevo proyecto");
7
8     }
9 }
10

```

#### 4.4. Creación de un repositorio GitHub.

Para creación del repositorio, en su cuenta GitHub creada se da clic en “**New**”, después de esto se le sara un nombre al repositorio y se seleccionara si quiere que sea de carácter público o privado, después de esto se da clic en la opción “**Create repository**”.



### Create a new repository

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

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

Owner \* JYacuma / Repository name \* Proyecto\_1  
✔ Proyecto\_1 is available.

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

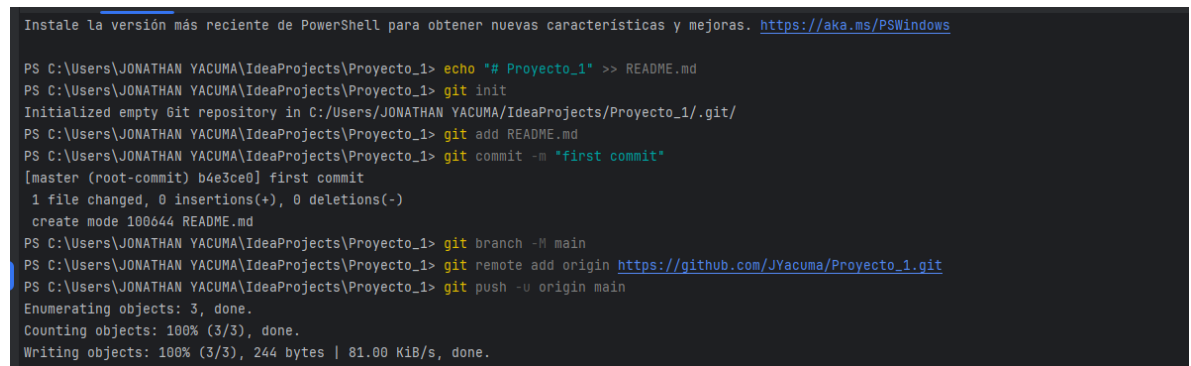
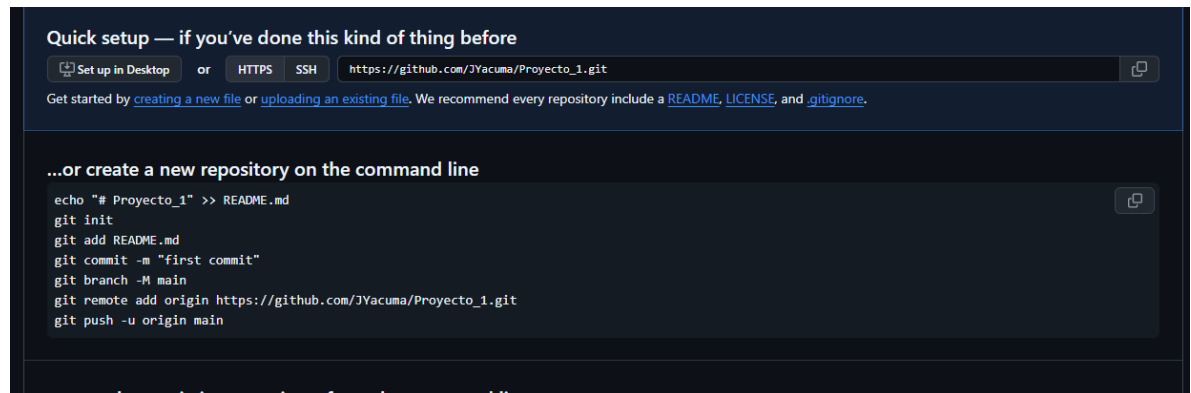
Description (optional)

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

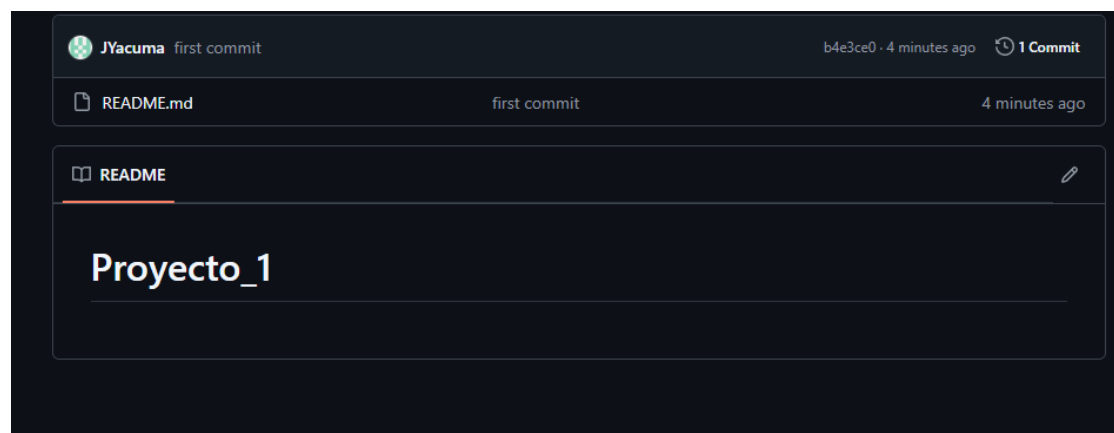
☐ **Private**  
 You choose who can see and commit to this repository.

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

Luego de esto aparecerá una línea de código que se deberá copiar en la terminal del proyecto creado en IntelliJ IDEA, con la función de establecer una conexión del repositorio local al repositorio de GitHub.



Al ser insertados este comando y recargada la página de GitHub, se mostrará la base de repositorio, con esto hecho, a continuación, se seguirá siempre una serie de comandos GIT para subir cualquier dato cambiado en el repositorio local.



#### 4.5. Comandos GIT para subir los datos del repositorio local al repositorio GitHub

Todos los comandos GIT se insertarán en el terminal del proyecto de IntelliJ IDEA.

##### 4.5.1. git init

Inicia un nuevo repositorio,

```
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1> git init
Reinitialized existing Git repository in C:/Users/JONATHAN YACUMA/IdeaProjects/Proyecto_1/.git/
```

##### 4.5.2. git status

Mostrara todos los archivos nuevos que no se han establecido en el repositorio GitHub.

```
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1> 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/
        src/

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1>
```

##### 4.5.3. git add .

Se establecerán una conexión de estos cambios con el repositorio GitHub, más no se subirán.

```
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1> git add .
```

#### 4.5.4.git commit -m “mensaje de un cambio”

Con este comando se confirmará y se mostraran todos los datos subidos al repositorio GitHub.

```
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1> git commit -m "Se agrego un cambio"
[main 1a603ec] Se agrego un cambio
4 files changed, 46 insertions(+)
create mode 100644 .gitignore
create mode 100644 .idea/.gitignore
create mode 100644 .idea/vcs.xml
create mode 100644 src/Main.java
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1>
```

#### 4.5.5. git push origin **main**

Tiene la función de especificar a que rama del proyecto quiere subir estos datos, en este caso como no es un trabajo colaborativo, se podrá subir a la rama **main**, que seria la rama principal del repositorio GitHub.

```
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1> git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (8/8), 1.04 KiB | 152.00 KiB/s, done.
Total 8 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/JYacuma/Proyecto\_1.git
b4e3ce0..1a603ec main -> main
```

Con estos comandos GIT se establece todos los cambios que fueron subidos del repositorio local al GitHub, al recargar la página se mostraran los cambios y las líneas de código que fueron subidas a la rama **main**.



## 5. Ramas

El uso de ramas es comúnmente utilizado para subdividir el proyecto en varias partes, esto se debe a que al poder un proyecto grande, se necesita especificar que función cumple cada subdivisión, para corregir los errores y confirmarlos a subirlos a la rama principal.

### 5.1. Creación de una rama

Para estar en uso compartido en GitHub, se debe aceptar la invitación a través del correo electrónico conectado a la cuenta, después de esto se realizarán unos comandos GIT para poder subir la información a una de las ramas creadas.

### 5.1.1. git branch

Sirve para ver en que rama está ubicado los datos.

```
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1> git branch
* main
```

### 5.1.2. git switch -c nombre de la rama

Crea una nueva rama, en donde se utilizará como borrador antes de combinarla con la rama **main**.

```
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1> git switch -c Tipografia
Switched to a new branch 'Tipografia'
```

### 5.1.3. git switch nombre de la rama

Cambia de la rama **main** a la nueva rama y se confirma con git branch.

```
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1> git switch Tipografia
M      .idea/vcs.xml
M      src/Main.java
Switched to branch 'Tipografia'
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1> git branch
* Tipografia
  main
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1>
```

### 5.1.4. git push origin nombre de la rama

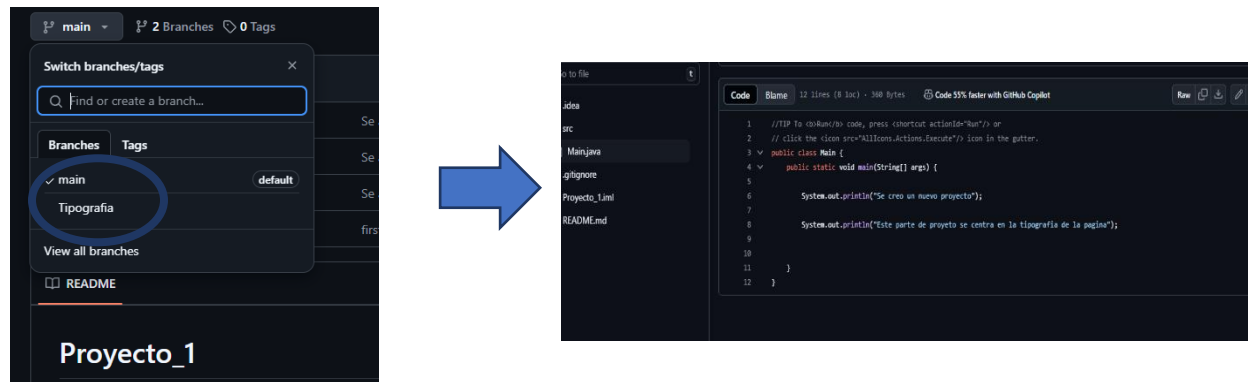
Cumple la función de publicar los datos que se crearon en la rama.

```
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1> git push origin Tipografia
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'Tipografia' on GitHub by visiting:
remote:   https://github.com/JYacuma/Proyecto\_1/pull/new/Tipografia
remote:
To https://github.com/JYacuma/Proyecto\_1.git
 * [new branch]      Tipografia -> Tipografia
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1>
```



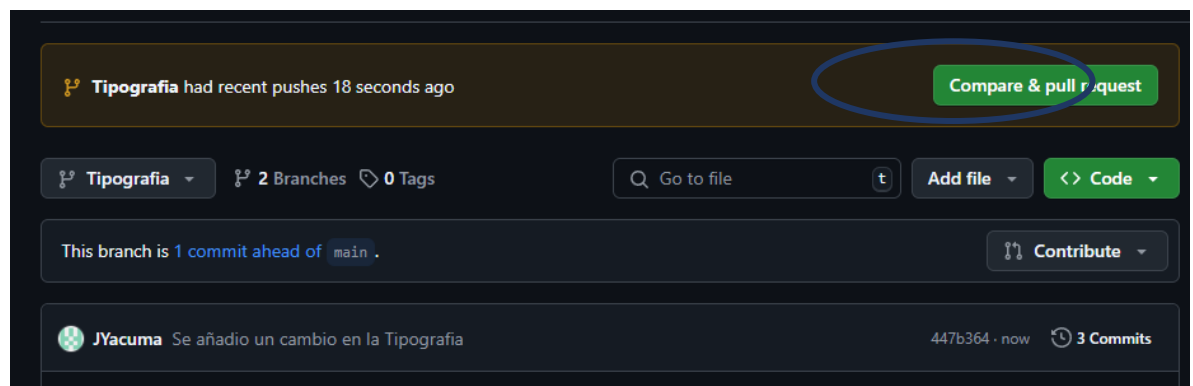
Después de estos pasos, se utiliza de nuevo los comandos vistos: `git add.` , `git commit -m comentario sobre que va hacer en la rama,` `git push origin nombre de la rama.`

Al hacer esto, se observarán los cambios en el repositorio GitHub.

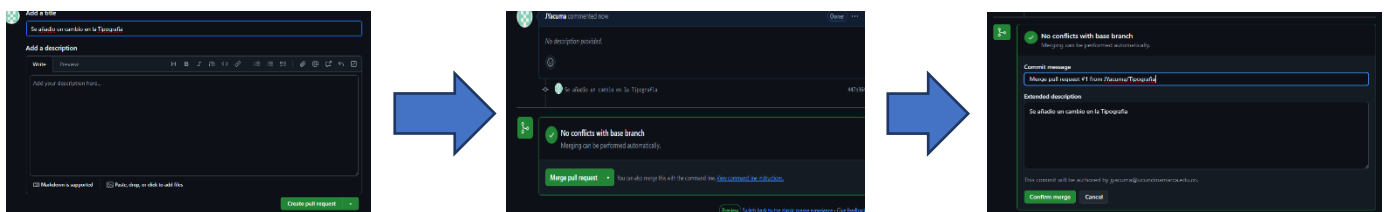


## 5.2 Combinación de ramas

Después de haber subido todos los datos a las ramas creadas, se combinarán con la rama **main**, se dará clic en el botón **Compare y pull request** en el repositorio GitHub.



Consecuentemente se confirma en cada botón verde para combinar las ramas



### 5.2.1 git pull origin nombre de la rama

Sube todos los datos de las ramas del repositorio local a la rama mencionada, en este caso la rama **main**.

```
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1> git pull origin main
From https://github.com/JYacuma/Proyecto_1
* branch      main      -> FETCH_HEAD
Already up to date.
```

## 5.3 Eliminación de Ramas

Se mostraran comandos GIT que permiten eliminar ramas tanto de manera local como remota, de igual manera, como revertir cambios.

### 5.3.1 git branch -D nombre de la rama

Permite borrar ramas en el repositorio local, la D tiene como sentido forzar el cierre de la rama, en este caso se aplicó git branch para observar el registro de la rama **Tipografía** antes de ser borrada.

```
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1> git branch
Tipografia
* main
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1> git branch -D Tipografia
Deleted branch Tipografia (was 88e2dba).
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1> git branch
* main
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1>
```

### 5.3.2 git branch -r

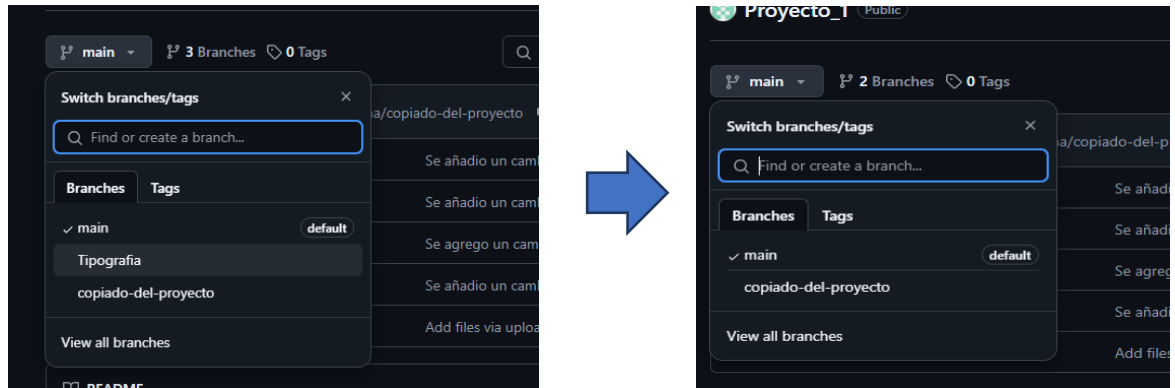
Al introducir este comando nos deja ver que ramas remotas están en el repositorio GitHub, en este caso solo la rama **main** esta en ambos repositorios, mientras que la rama **Tipografía** solo se encuentra en GitHub.

```
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1> git branch -r
  origin/HEAD -> origin/main
  origin/Tipografia
  origin/main
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1>
```

### 5.3.3 git push origin --delete nombre de la rama

Elimina las ramas remotas, es decir, permite eliminar las ramas del repositorio GitHub desde la terminal del repositorio local, se digita **git branch -r** para comprobar que se elimino la rama remota.

```
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1> git branch -r
  origin/HEAD -> origin/main
  origin/Tipografia
  origin/main
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1> git push origin --delete Tipografia
To https://github.com/JYacuma/Proyecto_1.git
- [deleted]          Tipografia
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1> git branch -r
  origin/HEAD -> origin/main
  origin/main
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1>
```



### 5.3.4 git log

Permite ver los cambios confirmados, esto incluye quien fue el autor y el tiempo donde realizó esta confirmación.

```
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1> git log
commit 88e2dba278ad7871e1ea26f3e8713730a8e6e8b5 (HEAD -> main, origin/main, origin/HEAD)
Merge: 1a603ec 447b364
Author: JYacuma <jyacuma@ucundinamarca.edu.co>
Date: Thu Feb 20 14:45:04 2025 -0500

    Merge pull request #1 from JYacuma/Tipografia

    Se añadió un cambio en la Tipografia

commit 447b3644ac8b5e7e6a7e20494e66b245df8f0f51
```

### 5.3.5 git revert

Servirá para deshacer cambios en el historial de confirmaciones de un commit, esto sin eliminar el historial.

### 5.3.6 git reflog

Muestra el registro de referencias GIT, esto se refiere a que muestra de manera detallada las acciones realizadas en el reposito local.

```
JONATHAN YACUMA@LAPTOP-P5QBJV84 MINGW64 ~/IdeaProjects/Proyecto_1 (Practica)
$ git reflog
88e2dba (HEAD -> Practica, origin/Practica, main) HEAD@{0}: checkout: moving from main to Practica
88e2dba (HEAD -> Practica, origin/Practica, main) HEAD@{1}: pull: Fast-forward
1a603ec HEAD@{2}: checkout: moving from Tipografia to main
88e2dba (HEAD -> Practica, origin/Practica, main) HEAD@{3}: pull origin main: Fast-forward
447b364 HEAD@{4}: commit: Se a adio un cambio en la Tipografia
1a603ec HEAD@{5}: checkout: moving from main to Tipografia
1a603ec HEAD@{6}: checkout: moving from Tipografia to main
1a603ec HEAD@{7}: checkout: moving from main to Tipografia
1a603ec HEAD@{8}: commit: Se agrego un cambio
b4e3ce0 HEAD@{9}: Branch: renamed refs/heads/master to refs/heads/main
b4e3ce0 HEAD@{11}: commit (initial): first commit

JONATHAN YACUMA@LAPTOP-P5QBJV84 MINGW64 ~/IdeaProjects/Proyecto_1 (Practica)
$
```

### 5.3.7 git log --oneline

Este tiene la funci n de mostrar todos los commit de manera resumida, dando que la persona vea de manera clara el historial de confirmaciones.

```
JONATHAN YACUMA@LAPTOP-P5QBJV84 MINGW64 ~/IdeaProjects/Proyecto_1 (Practica)
$ git log --oneline
88e2dba (HEAD -> Practica, origin/Practica, main) Merge pull request #1 from JYacuma/Tipografia
447b364 Se a adio un cambio en la Tipografia
1a603ec Se agrego un cambio
b4e3ce0 first commit

JONATHAN YACUMA@LAPTOP-P5QBJV84 MINGW64 ~/IdeaProjects/Proyecto_1 (Practica)
$
```

### 5.3.8 git merge

Este comando fusiona dos ramas de un proyecto en una sola.

```
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1> git merge Practica_2  
Already up to date.
```

### 5.3.9 git rebase

Esta se utiliza para integrar una rama del proyecto a otra, dando que cambie la base del commit.

```
PS C:\Users\JONATHAN YACUMA\IdeaProjects\Proyecto_1> git rebase Practica_2  
Current branch Practica is up to date.
```

## 7.Consultas

- Como se sincroniza una carpeta vacía a un repositorio GitHub ya existente

## 8.Conclusiones

El programa GIT, IntelliJ IDEA, JDK, GitHub son unas herramientas que nos ayudan en el entorno de programación para el aprendizaje personal como para la aplicación de trabajos empresariales.

Todo este seguimiento de pasos nos ayuda en la base de utilización de repositorios, y a su vez para el aprendizaje de como se comportan estos programas, esto con su debida estructuración para el uso personal o compartido.

## 9.Referencias bibliográficas

Git SCM. (s.f.). *Git documentation: Git commands*. Git. Recuperado el 20 de febrero de 2025, de <https://git-scm.com/doc>

GitHub. (2023, 15 de marzo). Guía de inicio rápido en GitHub.  
<https://docs.github.com/es/get-started/quickstart>

Se uso el 25% de IA en este documento.