

Git y sus comandos en IntelliJ IDEA

Informe de Programación II

Brayan Esteban Pinzon Torres

Profesor: William Alexander Matallana Porras

Universidad de Cundinamarca

Extensión Chía

301T: Programación II

Fecha de entrega: 20/02/2025

Tabla De Contenido

3. Introducción

4. Objetivos

5. Desarrollo

6. Conclusión

7. Bibliografía

Introducción

Git es un sistema de control de versiones ampliamente utilizado en el desarrollo de software, permitiendo a los desarrolladores gestionar cambios en el código de manera eficiente y colaborativa. IntelliJ IDEA es un entorno de desarrollo integrado (IDE) que facilita la integración con Git y GitHub, ofreciendo herramientas visuales para gestionar repositorios y sincronizar cambios. Este informe tiene como objetivo proporcionar una guía paso a paso para configurar Git, IntelliJ IDEA y GitHub, y subir un repositorio a GitHub de manera correcta.

Objetivos

1. Configurar Git en IntelliJ IDEA.
2. Crear y gestionar un repositorio local con Git.
3. Subir un repositorio local a GitHub.
4. Familiarizarse con los comandos básicos de Git y su uso en IntelliJ IDEA.

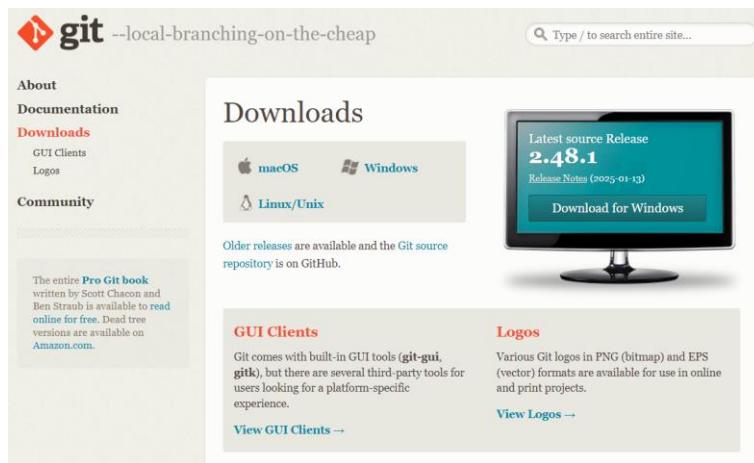
Desarrollo

1. Configuración Inicial de Git

Antes de comenzar, es necesario instalar y configurar Git en tu sistema.

1. Instalar Git:

- Descarga Git desde git-scm.com.



- Elige tu sistema operativo



- Sigue las instrucciones de instalación para tu sistema operativo.

Download for Windows

[Click here to download](#) the latest (2.48.1) 64-bit version of Git for Windows. This is the most recent [maintained build](#). It was released [7 days ago](#), on 2025-02-13.

Other Git for Windows downloads

[Standalone Installer](#)

[32-bit Git for Windows Setup](#).

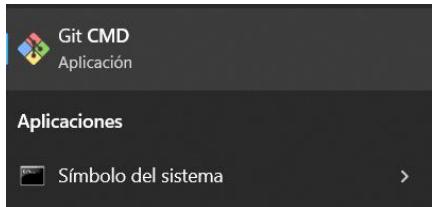
[64-bit Git for Windows Setup](#).

Portable ("thumbdrive edition")

[32-bit Git for Windows Portable](#).

[64-bit Git for Windows Portable](#).

2. Configuración de Git

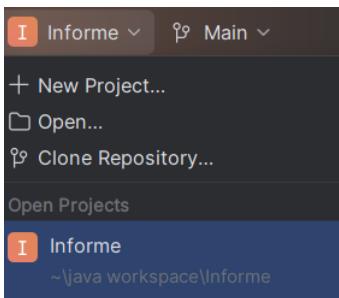


- o **git --version** : para asegurarte que este correctamente instalado y la versión

```
C:\Users\beste>git --version  
git version 2.47.0.windows.1
```

3. Abre IntelliJ IDEA:

- Inicia IntelliJ y abre tu proyecto.



4. Configurar nombre y correo:

- Abre la terminal de IntelliJ IDEA



Paso a Paso: Uso de Todos los Comandos en Orden

1. Configuración Inicial de Git

1. git config --global user.name "Tu Nombre"

- o Configura el nombre asociado a mis commits.

- Ejemplo:

```
git config --global user.name "Brayan Esteban Pinzon Torres"
```

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git config --global user.name "Brayan Esteban Pinzon Torres"
```

2. **git config --global user.email "tu-email@example.com"**

- Configura el correo electrónico asociado a mis commits.
- Ejemplo:

```
git config --global user.email b.esteban2006@mail.com
```

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git config --global user.email b.esteban2006@mail.com
```

¿Para qué sirve?: Estos comandos configuran tu identidad en Git, asociando tus commits con tu nombre y correo electrónico.

3. **git config --list**

- Muestra todas las configuraciones de Git actuales.
- Ejemplo:

```
git config --list
```

```
user.name=Brayan Esteban Pinzon Torres
user.email=b.esteban2006@mail.com
```

2. Crear y Configurar un Repositorio Local

1. **git init**

- Inicializa un nuevo repositorio Git en el directorio actual.

- Ejemplo:

```
git init
```

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git init
Reinitialized existing Git repository in C:/Users/beste/java workspace/miRepositorioJAVA/.git/
```

2. git status

- Muestra el estado actual del repositorio.

- Ejemplo:

```
git status
```

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git status
On branch master

No commits yet

Changes to be committed:
(use "git rm --cached <file>..." to unstage)
  new file:   .gitignore
  new file:   .idea/.gitignore
  new file:   .idea/misc.xml
  new file:   .idea/modules.xml
  new file:   .idea/vcs.xml
  new file:   miRepositorioJAVA.iml
  new file:   src/Main.java
```

3. Crear un archivo de ejemplo

- Para simular cambios en el proyecto.

- Ejemplo:

```
echo "Mi Repositorio Informe" > archivo.txt
```

```
public class Main { new *
    public static void main(String[] args) { new *
        System.out.println("Mi Repositorio Informe");
    }
}
```

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> echo "Mi Repositorio Informe" > archivo.txt
```

4. git add .

- Añade todos los archivos modificados al área de preparación.
- Ejemplo:

```
git add .
```

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git add .
warning: in the working copy of 'src/Main.java', LF will be replaced by CRLF the next time Git touches it
```

5. git commit -m "Primer commit"

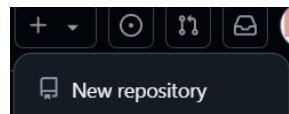
- Guarda los cambios en el historial del repositorio.
- Ejemplo:

```
git commit -m "Primer commit: se añadió archivo.txt"
```

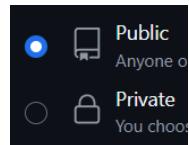
```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git commit -m "Primer commit: se añadió archivo.txt"
[master (root-commit) 3e229d3] Primer commit: se añadió archivo.txt
 8 files changed, 68 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 .idea/.gitignore
 create mode 100644 .idea/misc.xml
 create mode 100644 .idea/modules.xml
 create mode 100644 .idea/vcs.xml
 create mode 100644 archivo.txt
 create mode 100644 miRepositorioJAVA.iml
 create mode 100644 src/Main.java
```

3. Conectar el Repositorio Local con GitHub

1. Inicia sesión en GitHub y crea un nuevo repositorio.



2. Asigna un nombre y elige la privacidad (público o privado).



3. Copia la URL del repositorio para usarla más adelante.

```
https://github.com/coldpastel/MiRepositorioJAVA.git
```

1. **git remote add origin https://github.com/coldpastel/MiRepositorioJAVA.git**

- Conecta el repositorio local con el remoto.
- Ejemplo:

```
git remote add origin https://github.com/coldpastel/MiRepositorioJAVA.git
```

```
git remote add origin https://github.com/coldpastel/MiRepositorioJAVA.git
```

2. git push origin main

- Sube los cambios locales al repositorio remoto.
- Ejemplo:

```
git push origin main
```

3. git branch -M main

- Cambia el nombre de la rama principal de master a main o para renombrar cualquier rama existente

(se pueden seguir esos pasos o en su defecto y para mayor agilidad se puede copiar los comandos de git hub para un repositorio ya creado)

Ejemplo:

```
...or push an existing repository from the command line
git remote add origin https://github.com/coldpastel/MiRepositorioJAVA.git
git branch -M main
git push -u origin main
```

Para luego pegarlo en la terminal de IntelliJ IDEA

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git remote add origin https://github.com/coldpastel/MiRepositorioJAVA.git
PS C:\Users\beste\java workspace\miRepositorioJAVA> git branch -M main
PS C:\Users\beste\java workspace\miRepositorioJAVA> git push -u origin main
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 16 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (12/12), 1.75 KiB | 597.00 KiB/s, done.
Total 12 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/coldpastel/MiRepositorioJAVA.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
PS C:\Users\beste\java workspace\miRepositorioJAVA>
```

De esta manera ya quedaría subido en el repositorio de GitHub

4. **git remote -v**

- Para verificar a qué repositorios remotos está conectado mi repositorio local.

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git remote -v
origin  https://github.com/coldpastel/MiRepositorioJAVA.git (fetch)
origin  https://github.com/coldpastel/MiRepositorioJAVA.git (push)
```

5. Gestión de Ramas

1. **git switch -c NuevaRama**

- Crea una nueva rama y cambia a ella.
- Ejemplo:

```
git switch -c NuevaRama
```

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git switch -c NuevaRama
Switched to a new branch 'NuevaRama'
```

2. **git branch Rama1**

- para crear una rama nueva llamada Rama1

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git branch Rama1
```

3. **Hacer cambios en la nueva rama**

- Para simular cambios en la nueva rama.
- Ejemplo:

```
echo "Nueva funcionalidad" > login.txt
```

```
git add .
```

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> echo "Nueva funcionalidad" > login.txt  
PS C:\Users\beste\java workspace\miRepositorioJAVA> git add .
```

```
git commit -m "Se añadió la funcionalidad de login"
```

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git commit -m "Se añadió la funcionalidad de login"  
[NuevaRama 60e7969] Se añadió la funcionalidad de login  
 2 files changed, 1 insertion(+)  
 create mode 100644 login.txt
```

4. git push origin NuevaRama

- Sube la nueva rama al repositorio remoto.
- Ejemplo:

```
git push origin NuevaRama
```

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git push origin NuevaRama  
Enumerating objects: 8, done.  
Counting objects: 100% (8/8), done.  
Delta compression using up to 16 threads  
Compressing objects: 100% (3/3), done.  
Writing objects: 100% (5/5), 457 bytes | 457.00 KiB/s, done.  
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)  
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.  
remote:  
remote: Create a pull request for 'NuevaRama' on GitHub by visiting:  
remote:     https://github.com/coldpastel/MiRepositorioJAVA/pull/new/NuevaRama  
remote:  
To https://github.com/coldpastel/MiRepositorioJAVA.git  
 * [new branch]      NuevaRama -> NuevaRama
```

5. git fetch --all

- Descarga todos los cambios del repositorio remoto.

- Ejemplo:

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git fetch --all
```

6. **git branch -r**

- Muestra las ramas remotas.
- Ejemplo:

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git branch -r  
origin/NuevaRama  
origin/main
```

7. **git switch main**

- Cambia a la rama principal.
- Ejemplo:

```
git switch main
```

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git switch main  
Switched to branch 'main'  
Your branch is up to date with 'origin/main'.
```

8. **git merge NuevaRama**

- Fusiona los cambios de la rama feature-login con la rama principal.
- Ejemplo:

```
git merge NuevaRama
```

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git merge NuevaRama
Updating 3e229d3..60e7969
Fast-forward
 login.txt      | Bin 0 -> 44 bytes
 src/Main.java | 1 +
 2 files changed, 1 insertion(+)
 create mode 100644 login.txt
```

9. git branch -D NuevaRama

- Elimina la rama local NuevaRama
- Ejemplo:

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git branch -D NuevaRama
Deleted branch NuevaRama (was 60e7969).
```

10. git push origin --delete NuevaRama

- Elimina la rama remota NuevaRama
- Ejemplo:

```
git push origin --delete NuevaRama
```

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git push origin --delete NuevaRama
To https://github.com/coldpastel/MiRepositorioJAVA.git
 - [deleted]           NuevaRama
```

6. Verificación del Historial y Reversión de Cambios

1. git log

- Muestra el historial de commits.
- Ejemplo:

```
git log
```

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git log
commit 60e7969d4f5dd57f0a3813ddd4984c06fa6018e3 (HEAD -> main)
Author: Brayan Esteban Pinzon Torres <b.esteban2006@mail.com>
Date:   Fri Feb 21 14:49:07 2025 -0500

    Se añadió la funcionalidad de login

commit 3e229d3fc68d9945193c98cc260acce99cf204a3 (origin/main)
Author: Brayan Esteban Pinzon Torres <b.esteban2006@mail.com>
Date:   Fri Feb 21 14:04:29 2025 -0500

    Primer commit: se añadió archivo.txt
```

2. git log --oneline

- Muestra el historial de commits en un formato resumido.
- Ejemplo:

```
git log --oneline
```

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git log --oneline
60e7969 (HEAD -> main) Se añadió la funcionalidad de login
3e229d3 (origin/main) Primer commit: se añadió archivo.txt
```

3. git reflog

- Muestra un registro de todas las acciones realizadas en el repositorio.
- Ejemplo:

```
git reflog
```

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git reflog
60e7969 (HEAD -> main) HEAD@{0}: merge NuevaRama: Fast-forward
3e229d3 (origin/main) HEAD@{1}: checkout: moving from NuevaRama to main
60e7969 (HEAD -> main) HEAD@{2}: commit: Se añadió la funcionalidad de login
3e229d3 (origin/main) HEAD@{3}: checkout: moving from main to NuevaRama
3e229d3 (origin/main) HEAD@{4}: Branch: renamed refs/heads/master to refs/heads/main
3e229d3 (origin/main) HEAD@{6}: commit (initial): Primer commit: se añadió archivo.txt
```

4. git revert 60e7969

- Crea un nuevo commit que deshace los cambios de un commit anterior.
- Ejemplo:

```
git revert 60e7969
```

- en este comando tuve un problema ya que me salió lo siguiente:

```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git revert 60e7969
hint: Waiting for your editor to close the file...
```

Cuando esto ocurrió automáticamente git abrió automáticamente mi editor de texto predeterminado (Visual Studio Code)

```
❖ COMMIT_EDITMSG •  
C: > Users > beste > java workspace > miRepositorioJAVA > .git > ❖ COMMIT_EDITMSG  
1 Revert "Se añadió la funcionalidad de login"  
2  
3 This reverts commit 60e7969d4f5dd57f0a3813ddd4984c06fa6018e3.  
4  
5 # Please enter the commit message for your changes. Lines starting  
6 # with '#' will be ignored, and an empty message aborts the commit.  
7 #  
8 # On branch main  
9 # Your branch is ahead of 'origin/main' by 1 commit.  
10 # (use "git push" to publish your local commits)  
11 #  
12 # Changes to be committed:  
13 # deleted:    login.txt  
14 # modified:   src/Main.java  
15 #  
16
```

Lo que hice fue investigar en deepseek (Inteligencia Artificial) y lo que me dijo fue que solo tenia que cambiar el mensaje predeterminado o modificarlo si lo creía necesario y guardarla y cerrar el editor, yo edite el mensaje, lo guarde y lo cerre

```
❖ COMMIT_EDITMSG •  
C: > Users > beste > java workspace > miRepositorioJAVA > .git > ❖ COMMIT_EDITMSG  
1 Revert "Solucion Problema"  
2  
3 This reverts commit 60e7969d4f5dd57f0a3813ddd4984c06fa6018e3.  
4
```

Este fue el resultado en IntelliJIDEA:

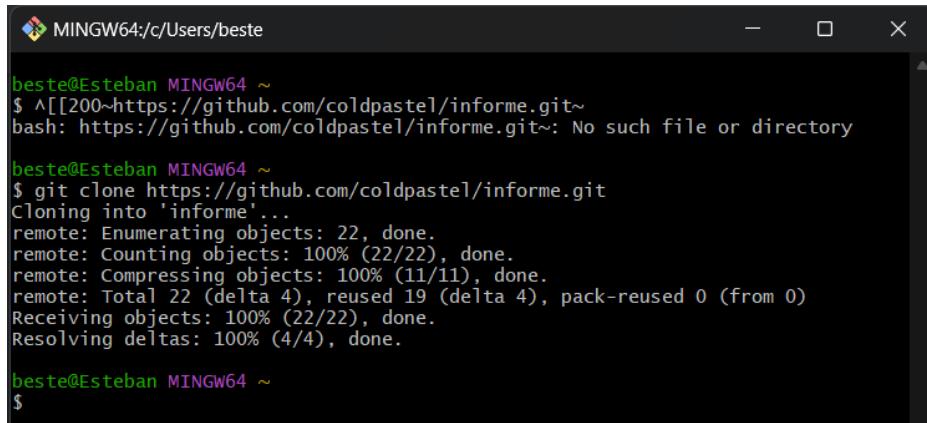
```
PS C:\Users\beste\java workspace\miRepositorioJAVA> git revert 60e7969  
[main 57eb3a2] Revert "Solucion Problema"  
2 files changed, 1 deletion(-)  
 delete mode 100644 login.txt
```

11. clonar un repositorio

- obtener la URL del repositorio que en este caso seria:

<https://github.com/coldpastel/informe.git>

- abrir una terminal, en este caso usare Git Bash
 - o pondre **git clone** y pegare el enlace



```
MINGW64:/c/Users/beste
beste@Esteban MINGW64 ~
$ ^[[200~https://github.com/coldpastel/informe.git~
bash: https://github.com/coldpastel/informe.git~: No such file or directory

beste@Esteban MINGW64 ~
$ git clone https://github.com/coldpastel/informe.git
Cloning into 'informe'...
remote: Enumerating objects: 22, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 22 (delta 4), reused 19 (delta 4), pack-reused 0 (from 0)
Receiving objects: 100% (22/22), done.
Resolving deltas: 100% (4/4), done.

beste@Esteban MINGW64 ~
$
```

Conclusión

Este flujo de trabajo cubre todos los comandos mencionados, desde la configuración inicial de Git hasta la gestión de ramas y la sincronización con un repositorio remoto. Cada comando tiene un propósito específico y se usa en un orden lógico para garantizar un flujo de trabajo eficiente. ¡Ahora estás listo para usar Git como un profesional! 😊

Conclusión

En este informe se exploró el uso de Git, IntelliJ IDEA y GitHub para gestionar proyectos de desarrollo de software de manera eficiente. A través de una guía paso a paso, se cubrieron los comandos básicos de Git, su configuración en IntelliJ IDEA y cómo subir y clonar repositorios en GitHub. Los principales aprendizajes incluyen:

1. Configuración de Git: Comandos como `git config --global user.name` y `git config --global user.email` son esenciales para asociar commits con la identidad del desarrollador, asegurando la trazabilidad del proyecto.
2. Inicialización de repositorios: `git init` y `git status` permiten crear un historial de cambios y verificar el estado de los archivos antes de realizar un commit.
3. Comandos básicos: `git add .`, `git commit -m`, y `git push origin main` facilitan la preparación de cambios, su registro en el historial y su sincronización con GitHub.
4. Gestión de ramas: Comandos como `git branch`, `git branch -D`, `git switch -c`, y `git push origin nombre-de-la-rama` permiten crear, eliminar, cambiar y publicar ramas, mejorando la organización del código.
5. Clonación de repositorios: `git clone` descarga una copia completa de un repositorio remoto, facilitando la colaboración en proyectos existentes.
6. Sincronización: `git pull origin main` y `git push origin main` aseguran que el repositorio local esté actualizado con los cambios remotos, evitando conflictos.
7. Uso de IntelliJ IDEA: La integración con Git y GitHub simplifica tareas como la clonación de repositorios y la creación de ramas, optimizando el flujo de trabajo.

En resumen, el dominio de estas herramientas mejora la organización del código, fomenta la colaboración en equipo y asegura que todos los miembros trabajen con la versión más actualizada del proyecto. Este conocimiento es fundamental para cualquier desarrollador que desee trabajar en entornos profesionales o contribuir a proyectos de código abierto.

Bibliografía

- Chacon, S., & Straub, B. (2014). *Pro Git* (2^a ed.). Apress. <https://git-scm.com/book/es/v2>
- Atlassian. (s.f.). *Glosario de Git: Comandos esenciales*. <https://www.atlassian.com/es/git/glossary#commands>
- GitHub Docs. (s.f.). *Guía de GitHub*. <https://docs.github.com>