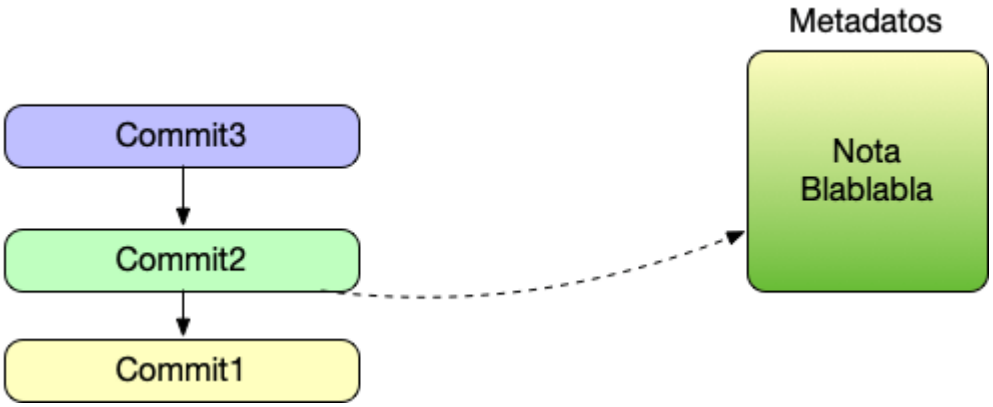
Tabla de Contenidos

Cuando usamos Git, casi siempre pensamos en commits, ramas (*branches*) y etiquetas (*tags*). Son las piezas básicas con las que llevamos el control de un proyecto. Pero Git tiene algún que otro “as bajo la manga” que poca gente conoce. Uno de ellos es Git Notes.

Las *notes* nos permiten añadir información a un commit sin modificarlo. Es decir: puedes dejar comentarios, enlaces o anotaciones en tu historial sin tener que tocar el mensaje del commit ni volver a escribir la historia. Una especie de post-it pegado al commit, pero sin ensuciar nada.



¿Qué son exactamente las Git Notes?

Imagina que tienes un commit con un mensaje bastante escueto, del tipo:

```
fix: bug en login
```

El commit está ahí, ya lo compartiste con el equipo, y cambiarlo sería mala idea porque

alterarías el historial. Pero… ¿y si quieres añadir un enlace al ticket que lo explica mejor? Ahí entran las *notes*.

Una *note* es un texto que se guarda aparte, en un espacio especial dentro del repositorio (`refs/notes/commits`). No cambia el commit, ni su hash, ni su contenido. Simplemente queda asociada a él.

¿Cuándo usarlas?

Algunas ideas:

- Para documentar por qué se tomó cierta decisión técnica.
- Para enlazar un bug o tarea de Jira/GitHub.
- Para dejar constancia de quién revisó o aprobó un cambio.
- Para añadir detalles de auditoría o métricas.
  En resumen: cuando quieras enriquecer la historia del proyecto sin tocar los commits originales.

Ejemplo paso a paso

Vamos a crear un repo muy sencillo:

```
git init git-notes-demo
cd git-notes-demo

echo "Hola Mundo" > app.txt
git add app.txt
git commit -m "Primer commit: archivo inicial"
```

```
echo "Hola Mundo v2" > app.txt
git commit -am "Segundo commit: actualización de contenido"

echo "Hola Mundo v3" > app.txt
git commit -am "Tercer commit: nueva versión"
```

El historial queda así:

```
a3f5c1d (HEAD -> main) Tercer commit: nueva versión
7b92e3f Segundo commit: actualización de contenido
d1a4b9a Primer commit: archivo inicial
```

Ahora añadimos notas:

```
git notes add -m "Se crea el archivo inicial del proyecto" d1a4b9a
git notes add -m "Corrección relacionada con el bug #101 en Jira"
7b92e3f
git notes add -m "Revisión aprobada por @ana_dev" a3f5c1d
```

Y si vemos el log con notas: con –show-notes:

```
git log --show-notes
```

Git nos muestra los commits con sus notas asociadas, sin alterar nada.

```
commit a3f5c1d (HEAD -> main)
Author: Tu Nombre <tu@email.com>
Date:   Mon Sep 30 10:32:00 2025

    Tercer commit: nueva versión
Notes:
    Revisión aprobada por @ana_dev
```

```
commit 7b92e3f
Author: Tu Nombre <tu@email.com>
Date:   Mon Sep 30 10:30:00 2025

    Segundo commit: actualización de contenido
Notes:
    Corrección relacionada con el bug #101 en Jira

commit d1a4b9a
Author: Tu Nombre <tu@email.com>
Date:   Mon Sep 30 10:25:00 2025

    Primer commit: archivo inicial
Notes:
    Se crea el archivo inicial del proyecto
```

Compartir las notas

Ojo: las *notes* no se envían al remoto por defecto. Para subirlas:

```
git push origin refs/notes/*
```

Y para bajarlas en otra máquina:

```
git fetch origin refs/notes/*:refs/notes/*
```

Cierre

Las Git Notes son como un cuaderno de apuntes paralelo a la historia de tu proyecto. Te dejan añadir contexto, comentarios y referencias sin meterte en líos de reescritura de commits.

Si trabajas en equipo, pueden ser muy útiles para dejar huellas de revisión, documentación extra o simplemente recordatorios.

- Eclipse GIT commit y como crear un repositorio
- Git Commit y Buenas prácticas
- Java Predicate Not y como usarlo
- Java APIs Diseño y Homogeneidad
- Eclipse Git , Repositorios locales y remotos