

# Git Básico

---

**Arturo Silvelo**

Try New Roads

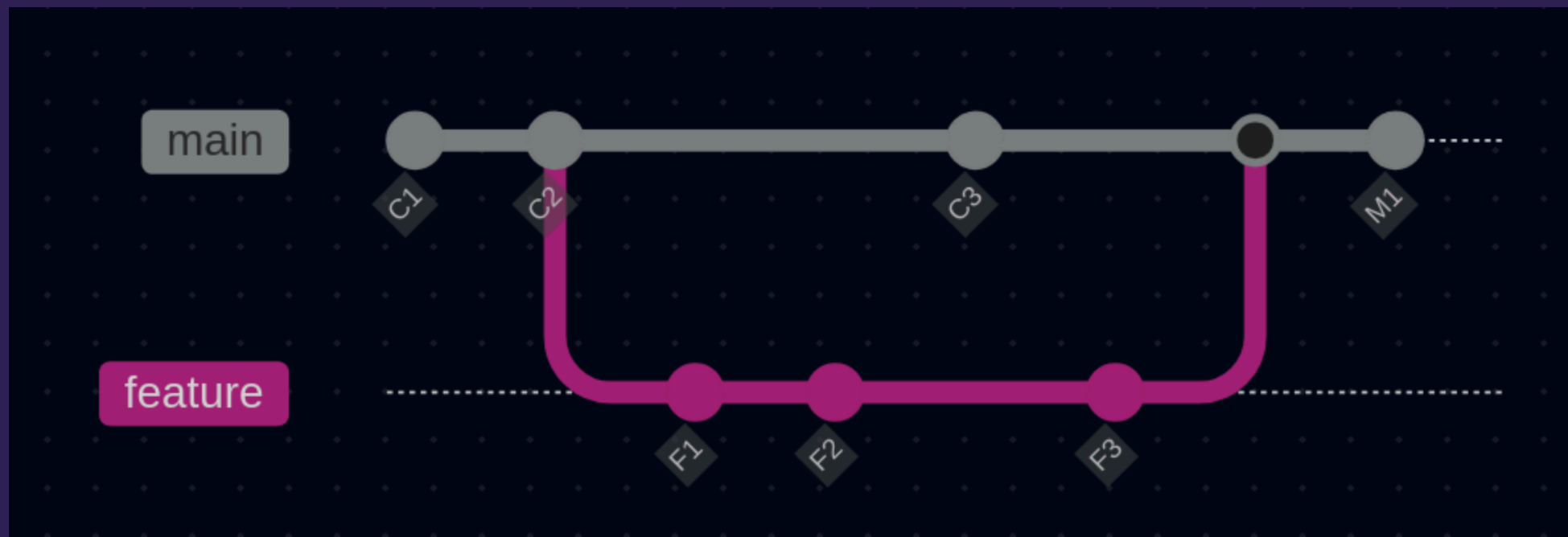
# Ramas

## ¿Qué es una rama?

---

Una **rama** (branch) en Git es una línea de desarrollo independiente que permite:

- Trabajar en nuevas características sin afectar el código principal
- Experimentar con cambios de forma segura
- Colaborar en paralelo con otros desarrolladores
- Mantener un historial limpio y organizado



## Ventajas de usar ramas

---

- **Aislamiento:** Los cambios en una rama no afectan otras ramas
- **Experimentación:** Puedes probar ideas sin riesgo
- **Colaboración:** Múltiples desarrolladores pueden trabajar simultáneamente
- **Organización:** Cada característica o bug fix tiene su propia rama
- **Historial limpio:** Facilita el seguimiento de cambios

## Crear una rama

---

Para crear una nueva rama se pueden usar diferentes comandos

- Crear solo la rama (sin cambiar a ella)

```
git branch feature
```

Para cambiar a la rama creada:

```
git switch feature  
# ó  
git checkout feature
```

- Crear la rama y cambiar a ella:

```
git switch -c feature  
git checkout -b feature
```

- Crear rama desde un commit específico:

```
git branch feature abc1234  
git switch -c feature abc1234
```

## Listar ramas

---

- Ver ramas locales:

```
git branch
```

La rama actual tendrá un asterisco al inicio.

- Ver todas las ramas (locales y remotas):

```
git branch -a
```



- Ordenar por fecha de modificación:

```
git branch --sort=-committerdate
```

- Ver información detallada:

```
git branch -v
```

# Trabajando Con Ramas

## Escenario inicial

---

Tu equipo está trabajando en un proyecto y necesitan manejar múltiples tareas simultáneamente.

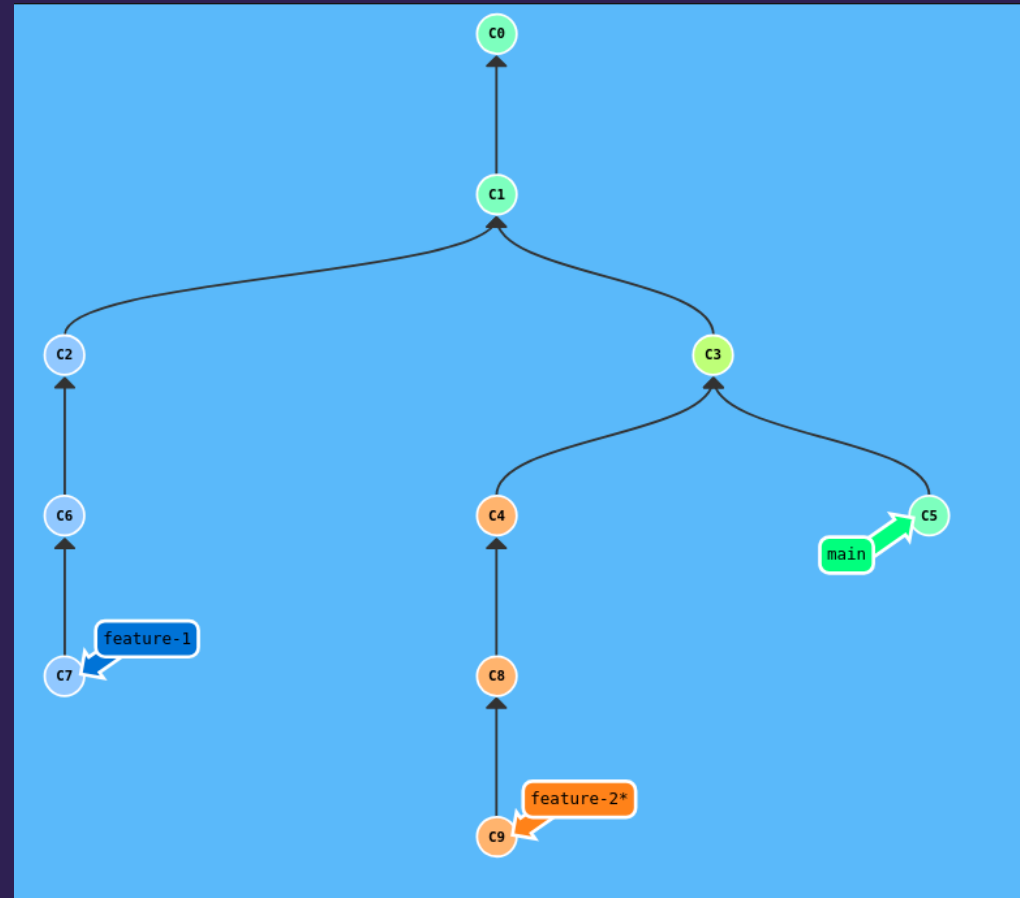
### Tareas a realizar:

1. **Nueva característica:** Te asignan desarrollar `feature-1`
2. **Bug crítico:** Durante el desarrollo encuentras un error que necesita solución inmediata
3. **Desarrollo paralelo:** Un compañero debe crear `feature-2` al mismo tiempo
4. **Segundo bug:** Se reporta otro error en la aplicación principal mientras todos trabajan

## Cargar el nivel

---

1. **Learn Git Branching**
2. Escribimos `import level`
3. Copiamos el fichero `ejercicios_1.json` y lo pegamos.



# Fusionar Ramas

## ¿Qué es fusionar (merge)?

---

**Fusionar** es el proceso de **integrar los cambios** de una rama en otra rama.

- Los commits de la rama origen se incorporan a la rama destino
- Se crea un historial unificado
- Permite combinar el trabajo de diferentes desarrolladores
- Mantiene la trazabilidad de los cambios

- Comando básico de fusión

```
git merge <rama-origen>
```

Ejemplo:

```
# 1. Cambiar a la rama destino  
git switch main  
  
# 2. Fusionar la rama feature  
git merge feature-1
```

 **Importante:** Siempre debes estar en la rama destino antes de hacer merge



## Tipos de merge

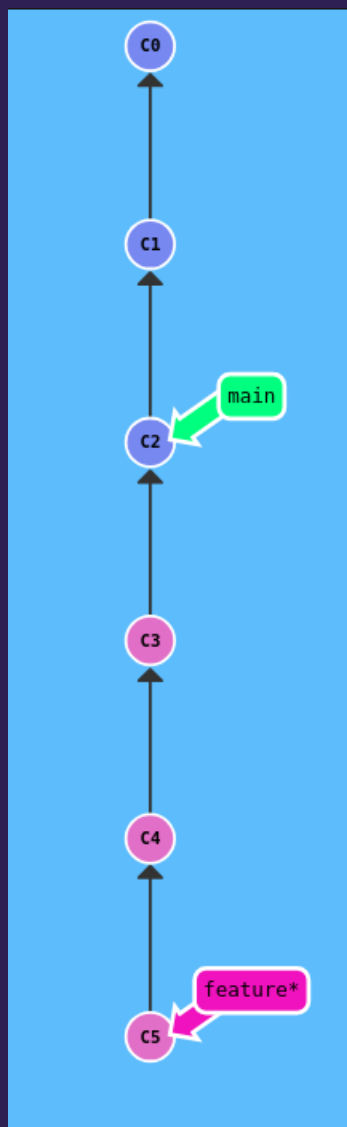
---

## Fast-forward

---

- Cuando no hay commits nuevos en la rama destino
- Simplemente mueve el puntero hacia adelante

```
git merge --ff-only my-branch
```



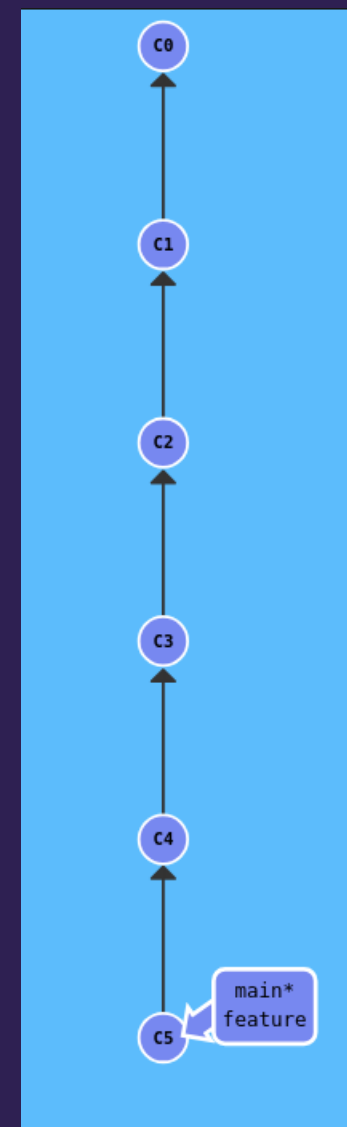
```
# Creamos la nueva rama
git switch -c feature

# Añadimos cambios
git commit
git commit
git commit

# Volvemos a la rama principal
git switch main

# Fusionamos las ramas
git merge --ff-only feature

# ó
git merge feature
```

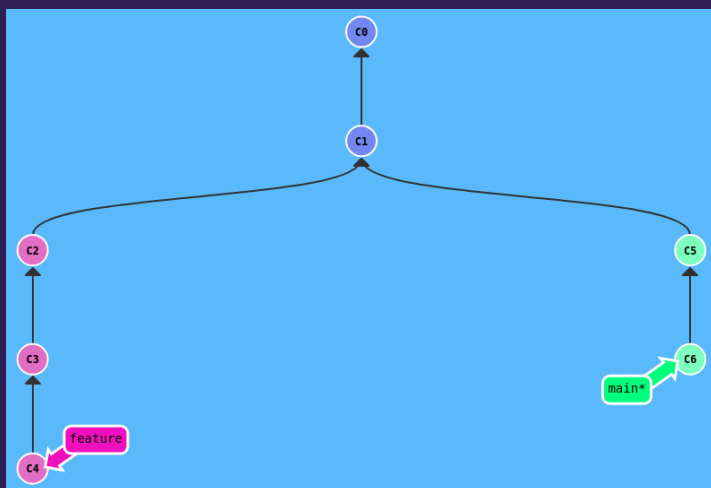


## No Fast-forward

---

- Cuando ambas ramas tienen commits nuevos
- Crea un commit de fusión que une ambas historias
- Preserva la estructura de ramas en el historial

```
git merge --no-ff my-branch
```



```
# Creamos la nueva rama  
git switch -c feature
```

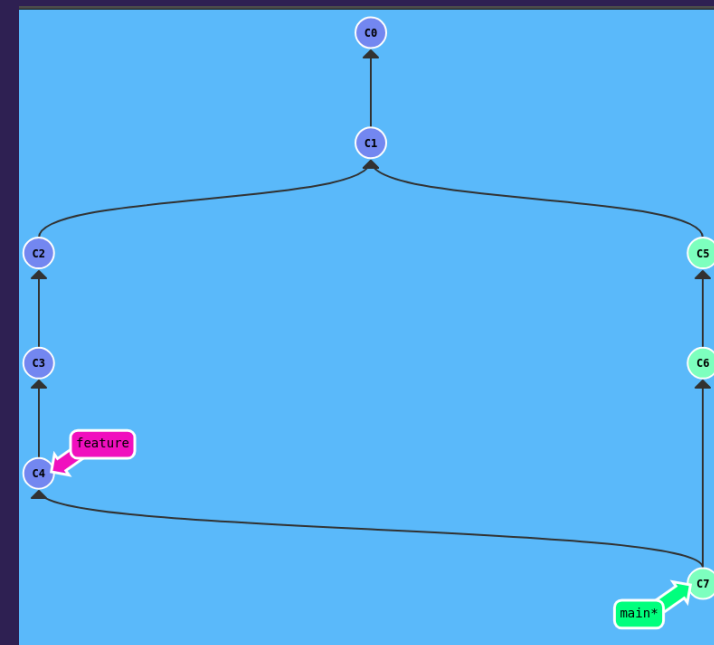
```
# Añadimos cambios  
git commit  
git commit  
git commit
```

```
# Volvemos a la rama principal  
git switch main
```

```
# Añadimos cambios  
git commit  
git commit  
git commit
```

```
# Fusionamos las ramas  
git merge --no-ff feature
```

```
# ó  
git merge feature
```

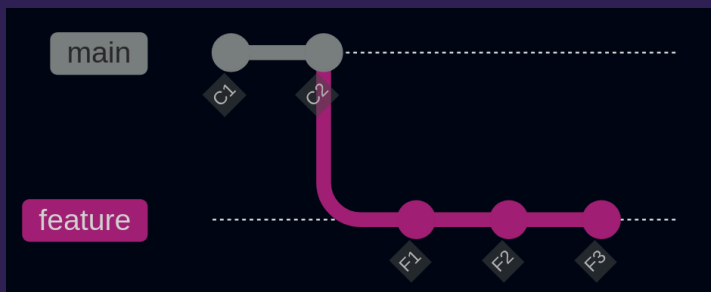


## Squash Merge

---

- Combina todos los commits de una rama en un solo commit
- Crea un historial más limpio y lineal
- Útil cuando la rama tiene muchos commits pequeños o experimentales

```
git merge --squash feature  
git commit -m "Add complete feature X"
```

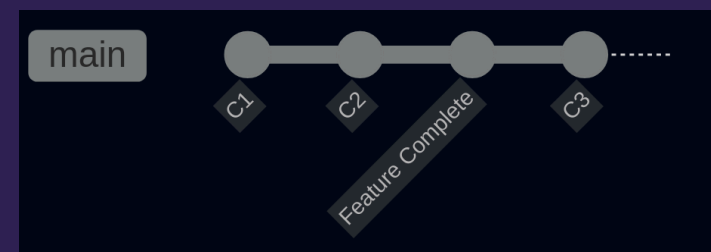


```
# Creamos la nueva rama
git switch -c feature

# Añadimos múltiples commits
git commit -m "F1"
git commit -m "F2"
git commit -m "F3"

# Volvemos a main
git switch main

# Squash merge: combina todos en uno
git merge --squash feature
git commit -m "Feature Complete"
git commit -m "C3"
```



## Personalizar mensajes de merge

---

Puedes modificar el mensaje de un merge usando `--edit` y `--no-commit`:

- `--edit`: Abre el editor para modificar el mensaje automático
- `--no-commit`: Realiza el merge sin crear el commit automáticamente





## Comportamiento por tipo de merge

---

### Fast-forward merge:

<code>git merge --edit feature</code>	#	✗	No funciona (no hay commit de merge)
<code>git merge --no-commit feature</code>	#	✓	Evita fast-forward, deja cambios en staging

## No Fast-forward merge:

```
git merge --edit --no-ff feature      #  Abre editor para mensaje  
git merge --no-commit --no-ff feature #  Prepara merge sin commitear
```

## Squash merge:

```
git merge --squash feature      # ☒ Siempre requiere commit manual  
git merge --squash --edit feature # ☒ Redundante (ya controlas el mensaje)
```

# Ejemplos prácticos

- git merge --edit --no-ff feature

```
silvelo@silvelo-Inspiron-7559 ~/..../course-example 📁 master > git status -s
silvelo@silvelo-Inspiron-7559 ~/..../course-example 📁 master > git switch -c feature
Switched to a new branch 'feature'
silvelo@silvelo-Inspiron-7559 ~/..../course-example 📁 feature > touch feature
silvelo@silvelo-Inspiron-7559 ~/..../course-example 📁 feature > git add feature
silvelo@silvelo-Inspiron-7559 ~/..../course-example 📁 feature > git commit -m 'Add feature'
[feature b2f33c0] Add feature
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 feature
silvelo@silvelo-Inspiron-7559 ~/..../course-example 📁 feature > git switch main
fatal: invalid reference: main
silvelo@silvelo-Inspiron-7559 ~/..../course-example 📁 feature > git switch master
Switched to branch 'master'
silvelo@silvelo-Inspiron-7559 ~/..../course-example 📁 master > git merge --edit --no-ff feature
Merge made by the 'ort' strategy.
 feature | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 feature
```

GNU nano 7.2

```
Mi merge personalizado
# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.
```

- `git merge --no-commit --no-ff feature`

```
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > git switch -c feature
Switched to a new branch 'feature'
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 feature > touch feature-file
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 feature > git add feature-file
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 feature > git commit -m 'My feature file'
[feature 4872d76] My feature file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 feature-file
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 feature > git switch master
Switched to branch 'master'
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > git status -s
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > git merge --no-co
--no-commit      --no-continue
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > git merge --no-commit --no-ff feature
Automatic merge went well; stopped before committing as requested
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > git status -s
A   feature-file
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > git commit -m 'My merge'
[master a4da5ca] My merge
```

- git merge --squash feature

```
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > git switch -c feature
Switched to a new branch 'feature'
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 feature > touch feature-45
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 feature > git add feature-45
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 feature > git commit -m 'feature-45'
[feature d6f5943] feature-45
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 feature-45
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 feature > git switch master
D   feature
Switched to branch 'master'
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > git merge --squash feature
Updating a4da5ca..d6f5943
Fast-forward
Squash commit -- not updating HEAD
 feature-45 | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 feature-45
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > git status -s
D feature
A feature-45
silvelo@silvelo-Inspiron-7559 ~/..📁../course-example 📁 master > git commit -m 'Complete Features'
[master 305e9f5] Complete Features
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 feature-45
```

```
commit 305e9f52d42aba00ed2441a30aa0ab5885bc21fd (HEAD -> master)
Author: silvelo <arturo.silvelo@gmail.com>
Date: Mon Sep 22 12:57:01 2025 +0200
```

Complete Features

```
commit a4da5ca3a65915ad9984394e5d58052bff59e1bc
Merge: ec218dc 4872d76
Author: silvelo <arturo.silvelo@gmail.com>
Date: Mon Sep 22 12:53:13 2025 +0200
```

My merge

```
commit 4872d7696e238be1ea5edacea1cd66cafd9985f6
Author: silvelo <arturo.silvelo@gmail.com>
Date: Mon Sep 22 12:52:16 2025 +0200
```

My feature file

```
commit ec218dccbf796b019623e57e0ce545eff296a7fb
Merge: 309800d b2f33c0
Author: silvelo <arturo.silvelo@gmail.com>
Date: Mon Sep 22 12:48:06 2025 +0200
```

Mi merge personalizado

```
commit b2f33c01d2ca03c44c90aef992291eb83c9ef3d6
Author: silvelo <arturo.silvelo@gmail.com>
Date: Mon Sep 22 12:44:39 2025 +0200
```

# Conflictos de Merge



## ¿Qué es un conflicto?

---

Un **conflicto** ocurre cuando Git no puede fusionar automáticamente los cambios porque:

- Dos ramas modificaron las **mismas líneas** en el mismo archivo
- Una rama modificó un archivo que otra rama eliminó
- Ambas ramas crearon archivos con el **mismo nombre**

Git necesita **tu ayuda** para decidir qué cambios mantener.

# ¿Cuándo ocurren los conflictos?

---

## Escenarios comunes:

- Dos desarrolladores editan la misma función
- Cambios en la misma línea de código
- Modificaciones simultáneas en archivos de configuración
- Refactorización que afecta las mismas secciones

## Git puede fusionar automáticamente:

- Cambios en líneas diferentes del mismo archivo
- Cambios en archivos diferentes
- Adición de nuevas líneas sin solapamiento

## Identificar un conflicto

---

Cuando ocurre un conflicto, Git te muestra:

```
$ git merge feature
Auto-merging file.txt
CONFLICT (content): Merge conflict in file.txt
Automatic merge failed; fix conflicts and then commit the result.
```

### Verificar estado:

```
git status
# On branch main
# You have unmerged paths.
# Unmerged paths:
#   both modified:   file.txt
```

## Anatomía de un conflicto

---

Git marca los conflictos en el archivo con marcadores especiales:

```
<<<<<<< HEAD
Código de la rama actual (main)
=====
Código de la rama que se está fusionando (feature)
>>>>>>> feature
```

## Ejemplo real:

```
function saludar() {  
<<<<<<< HEAD  
    return "Hola Mundo";  
=====  
    return "Hello World";  
>>>>>>> feature  
}
```