Creamos un directorio e inicializamos en el

```
windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5 (mai
n)
$ mkdir prueba-fast-forward-merge

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5 (mai
n)
$ cd prueba-fast-forward-merge

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prue
ba-fast-forward-merge (main)
$ git init
Initialized empty Git repository in C:/Users/windows10/Desktop/DesarrolloDeSoftw
are/actividad5/prueba-fast-forward-merge/.git/
```

Creamos 2 ramas y

```
ba-fast-forward-merge (main)
$ git checkout -b rama1
Switched to a new branch 'rama1'

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prue
ba-fast-forward-merge (rama1)
$ echo "cambio rama1" > archivo.txt

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prue
ba-fast-forward-merge (rama1)
$ git add .
warning: adding embedded git repository: DesarrolloDeSoftware
hint: You've added another git repository inside your current repository.
hint: Clones of the outer repository will not contain the contents of
hint: the embedded repository and will not know how to obtain it.
hint: If you meant to add a submodule, use:
hint:
hint:   git submodule add <url> DesarrolloDeSoftware
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:
hint:   git rm --cached DesarrolloDeSoftware
hint:
hint: See "git help submodule" for more information.
hint: Disable this message with "git config set advice.addEmbeddedRepo false"
warning: in the working copy of 'archivo.txt', LF will be replaced by CRLF the n
ext time Git touches it
```

hacemos cambios en cada una de las 2 ramas y guardamos en commit:

```
windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prue
ba-fast-forward-merge (rama1)
$ git commit -m "Cambio en rama1"
[rama1 (root-commit) 4b43fe8] Cambio en rama1
 2 files changed, 2 insertions(+)
 create mode 160000 DesarrolloDeSoftware
 create mode 100644 archivo.txt

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prue
ba-fast-forward-merge (rama1)
$ git checkout main
error: pathspec 'main' did not match any file(s) known to git

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prue
ba-fast-forward-merge (rama1)
$ git checkout -b rama2
Switched to a new branch 'rama2'

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prue
ba-fast-forward-merge (rama2)
$ echo "cambio rama2" > archivo.txt

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prue
ba-fast-forward-merge (rama2)
$ git add .
warning: in the working copy of 'archivo.txt', LF will be replaced by CRLF the n
ext time Git touches it

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prue
ba-fast-forward-merge (rama2)
$ git commit -m "Cambio en rama2"
[rama2 5bb6ef0] Cambio en rama2
 1 file changed, 1 insertion(+), 1 deletion(-)
```

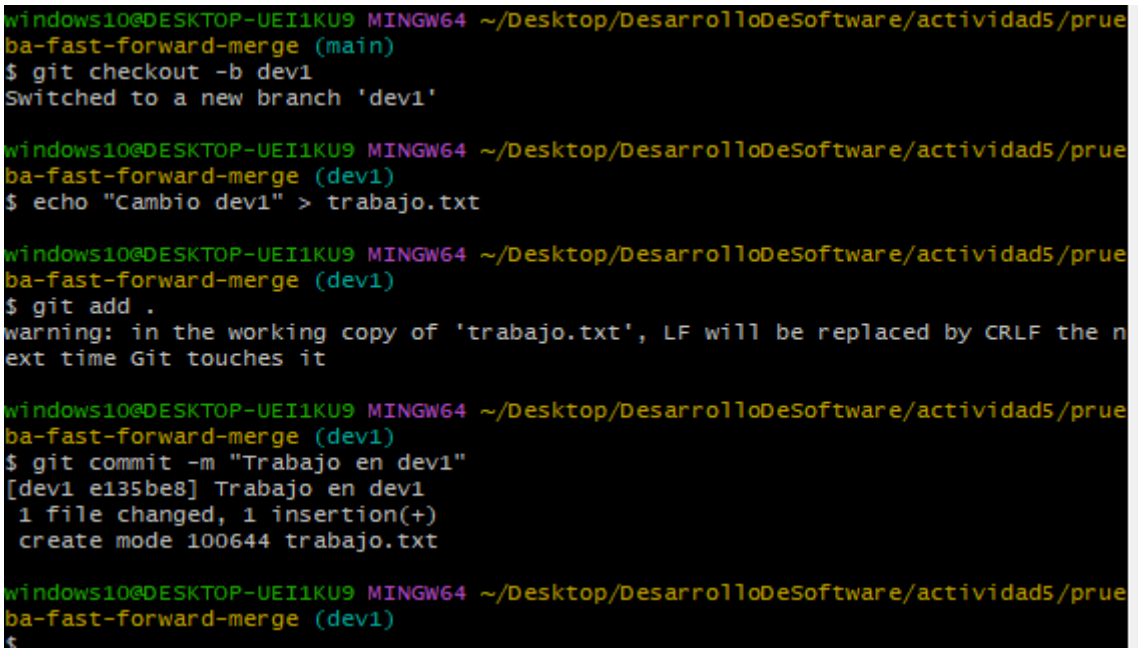
Nos cambiamos al main y fusionamos las ramas

```
ba-fast-forward-merge (main)
$ git checkout main
Already on 'main'

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSo
ba-fast-forward-merge (main)
$ git merge --ff rama2
Updating 4b43fe8..5bb6ef0
Fast-forward
 archivo.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSo
ba-fast-forward-merge (main)
$ git log --graph --oneline
* 5bb6ef0 (HEAD -> main, rama2) Cambio en rama2
* 4b43fe8 (rama1) Cambio en rama1

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSo
```

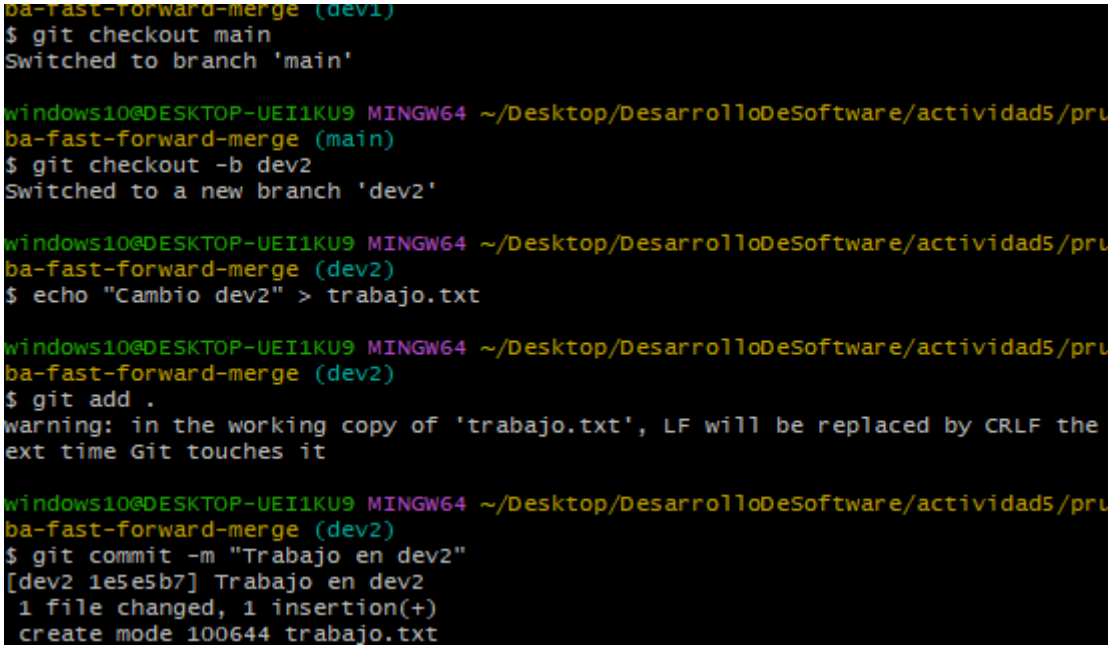
Vemos el historial:

```
windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/Desarr
ba-fast-forward-merge (main)
$ git log --graph --oneline
* 5bb6ef0 (HEAD -> main, rama2) Cambio en rama2
* 4b43fe8 (rama1) Cambio en rama1
```

Creamos nueva rama1 y txt y guardamos el commit



Creamos nueva rama2 y txt y guardamos el commit



Fusiona la rama 1 y forzamos la creación del commit

```
ba-fast-forward-merge (dev2)
$ git checkout main
Switched to branch 'main'

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad
ba-fast-forward-merge (main)
$ git merge --no-ff dev1 -m "Merge de dev1"
Merge made by the 'ort' strategy.
 trabajo.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 trabajo.txt

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad
ba-fast-forward-merge (main)
$ git merge --no-ff dev2 -m "Merge de dev2"
Auto-merging trabajo.txt
CONFLICT (add/add): Merge conflict in trabajo.txt
Automatic merge failed; fix conflicts and then commit the result.

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad
ba-fast-forward-merge (main|MERGING)
```

Mostramos el historial de commits:

```
windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/Desar
ba-fast-forward-merge (main|MERGING)
$ git log --graph --oneline
*   c6f16ce (HEAD -> main) Merge de dev1
|\
| * e135be8 (dev1) Trabajo en dev1
|/
* 5bb6ef0 (rama2) Cambio en rama2
* 4b43fe8 (rama1) Cambio en rama1
```

Pregunta: ¿Cuáles son las principales ventajas de utilizar git merge --no-ff en un proyecto en equipo? ¿Qué problemas podrían surgir al depender excesivamente de commits de fusión?

Ventajas:

. Mantiene el historial claro.

. Sabes qué cambios provinieron de qué rama.

. Facilita el rollback.

Desventajas:

. El historial se llena de muchos commits de merge.

. Puede volverse difícil de leer si el equipo no lo organiza.

Depender excesivamente de **commits de fusión (merge commits)** puede causar varios problemas en el desarrollo de software

Creamos nuevo directorio e inicializamos

```
$ mkdir prueba-merge-conflict

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prue
ba-fast-forward-merge (main|MERGING)
$ cd prueba-merge-conflict

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prue
ba-fast-forward-merge/prueba-merge-conflict (main|MERGING)
$ git init
Initialized empty Git repository in C:/Users/windows10/Desktop/DesarrolloDeSoftw
are/actividad5/prueba-fast-forward-merge/prueba-merge-conflict/.git/
```

Creamos un archivo index.html y un commit

```
ba-fast-forward-merge/prueba-merge-conflict (main)
$ echo "<html><body><h1>Proyecto inicial CC3S2</h1></body></html>" > index.html

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prue
ba-fast-forward-merge/prueba-merge-conflict (main)
$ git add index.html
warning: in the working copy of 'index.html', LF will be replaced by CRLF the ne
xt time Git touches it

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prue
ba-fast-forward-merge/prueba-merge-conflict (main)
$ git commit -m "Commit inicial del index.html en main"
[main (root-commit) 234effd] Commit inicial del index.html en main
 1 file changed, 1 insertion(+)
 create mode 100644 index.html

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prue
```

Creamos una nueva rama y agregamos contenido al archivo y realizamos un commit

```
ba-fast-forward-merge/prueba-merge-conflict (main)
$ git checkout -b feature-update
Switched to a new branch 'feature-update'

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prue
ba-fast-forward-merge/prueba-merge-conflict (feature-update)
$ echo "<p>Contenido de la sección nueva</p>" >> index.html

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prue
ba-fast-forward-merge/prueba-merge-conflict (feature-update)
$ git add index.html
warning: in the working copy of 'index.html', LF will be replaced by CRLF the ne
xt time Git touches it

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prue
ba-fast-forward-merge/prueba-merge-conflict (feature-update)
$ git commit -m "Agrega sección nueva en feature-update"
[feature-update beabb14] Agrega sección nueva en feature-update
 1 file changed, 1 insertion(+)
```

Cambiamos de rama y agregamos un footer al index.html y realizamos un commit

```
windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prue
ba-fast-forward-merge/prueba-merge-conflict (feature-update)
$ git checkout main
Switched to branch 'main'

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prue
ba-fast-forward-merge/prueba-merge-conflict (main)
$ echo "<footer>Contacta aquí example@example.com</footer>" >> index.html

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prue
ba-fast-forward-merge/prueba-merge-conflict (main)
$ git add index.html
warning: in the working copy of 'index.html', LF will be replaced by CRLF the ne
xt time Git touches it

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prue
ba-fast-forward-merge/prueba-merge-conflict (main)
$ git commit -m "Agrega footer en main"
[main dce7019] Agrega footer en main
 1 file changed, 1 insertion(+)
```

Fusionamos y notamos el conflicto

```
windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/activi
ba-fast-forward-merge/prueba-merge-conflict (main)
$ git merge --no-ff feature-update
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

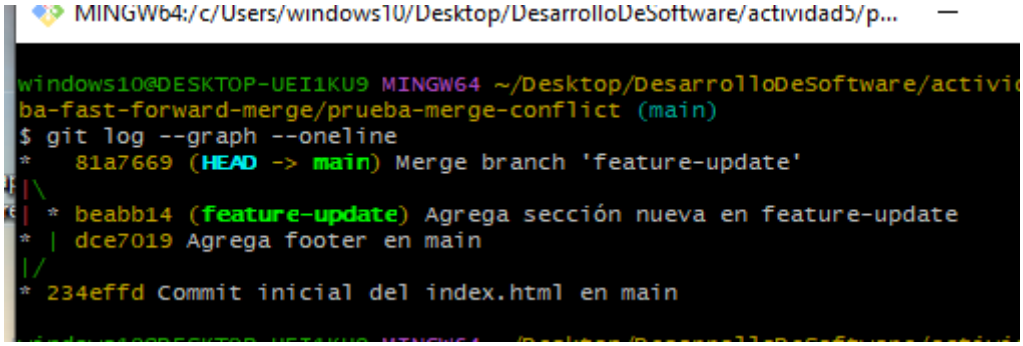
solucionamos

```
ktop > DesarrolloDeSoftware > actividad5 > prueba-fast-forward-merge > prueba-merge-confli
1   <html>
2     <body>
3       <h1>Proyecto inicial CC3S2</h1>
4       <p>Contenido de la sección nueva</p>
5       <footer>Contacta aquí example@example.com</footer>
6     </body>
7   </html>
```

Subimos y realizamos un commit

```
$ git add index.html
warning: in the working copy of 'index.html', LF will be replaced by CRLF
xt time Git touches it

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad
ba-fast-forward-merge/prueba-merge-conflict (feature-update)
$ git commit -m "Agrega sección nueva en feature-update"
[feature-update beabb14] Agrega sección nueva en feature-update
 1 file changed, 1 insertion(+)
```

Verificamos el historial:



PREGUNTAS:
¿Qué pasos adicionales tuviste que tomar para resolver el conflicto?

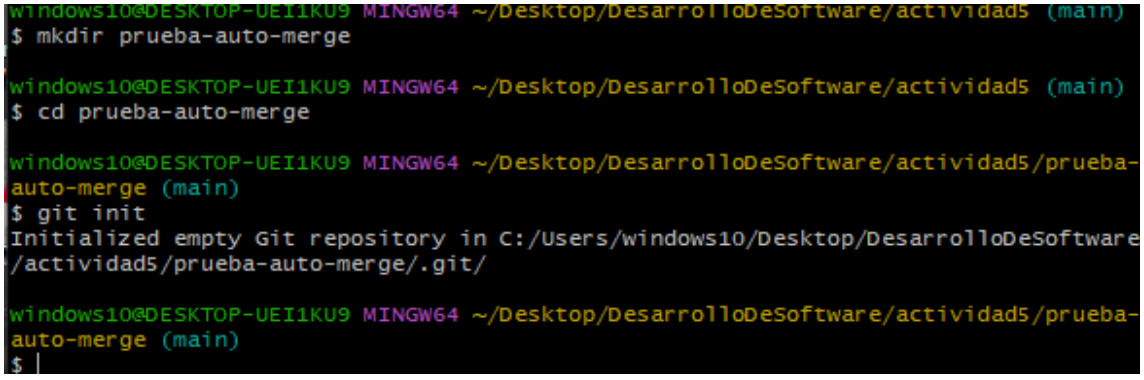
Para resolver el conflicto, primero Git detectó un problema al hacer git merge --no-ff. Abrí el archivo con conflicto (index.html), eliminé las marcas (<<<<<<<, =======, >>>>>>>), combiné los cambios manualmente, guardé el archivo, lo agregué con git add y completé la fusión con git commit.

¿Qué estrategias podrías emplear para evitar conflictos en futuros desarrollos colaborativos? Responde

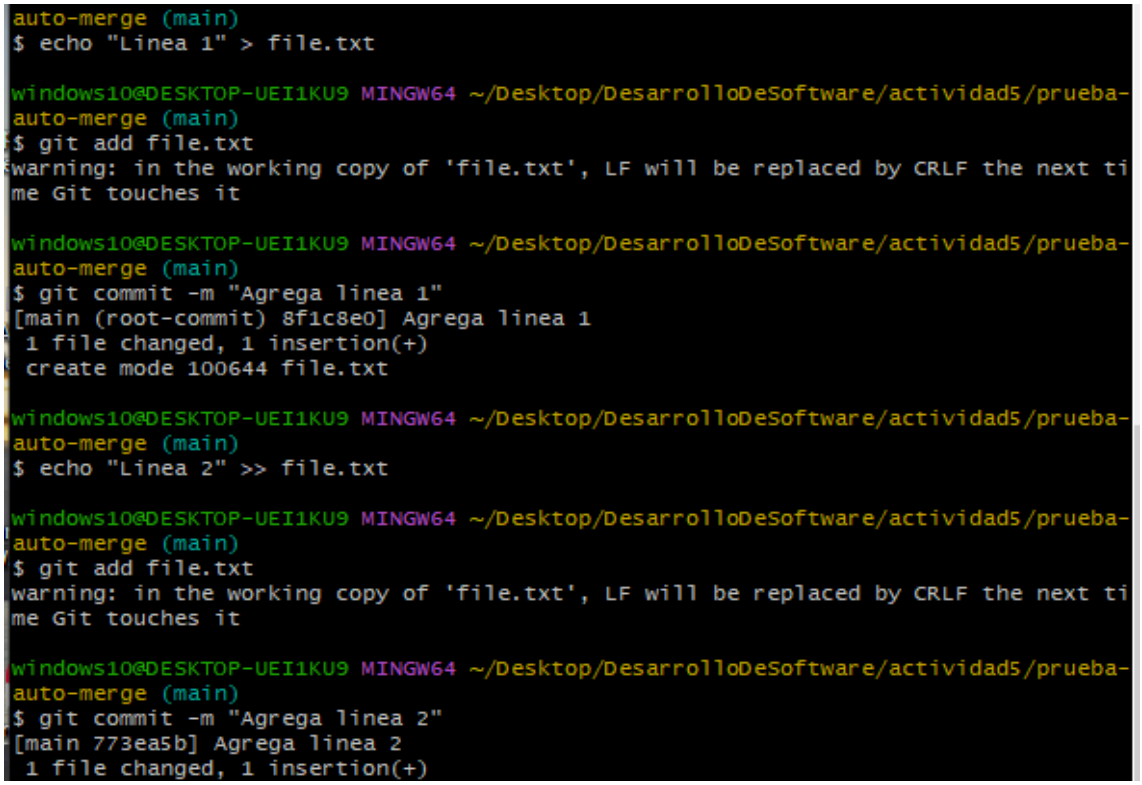
Para evitar conflictos en el futuro, es importante: comunicarse bien con el equipo, dividir las tareas, hacer git pull seguido, trabajar en ramas separadas, hacer commits pequeños y usar revisiones de código para detectar problemas antes de fusionar.

EJERCICIO:

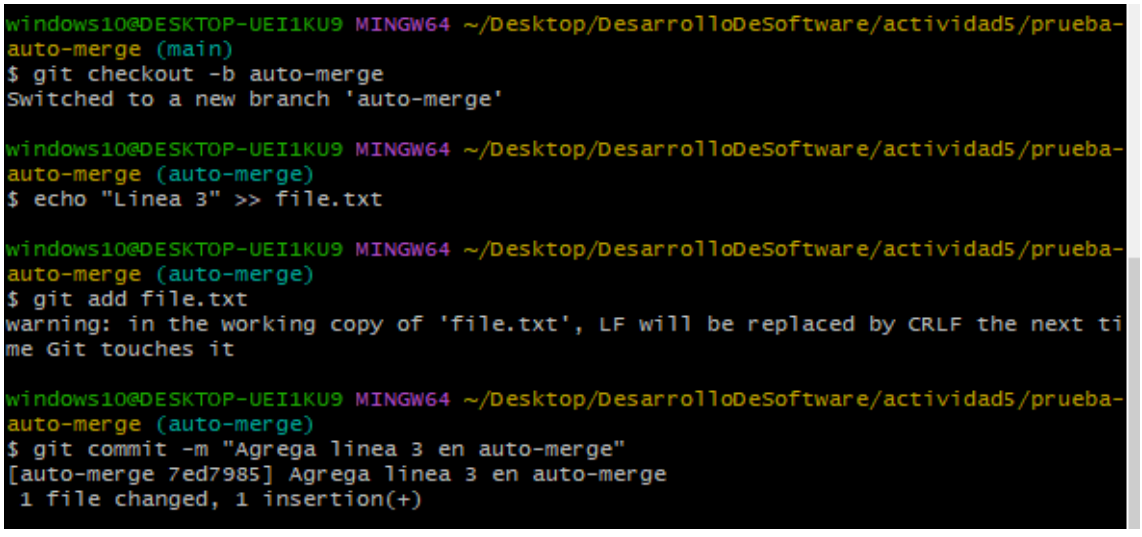
Creamos un directorio e inicializamos en el



Creamos un archivo file.txt que diga línea 1 y hacemos un commit y le agregamos línea 2 y otro commitc



Creamos otra rama y commit



Volvemos al main y hacemos cambio

```
auto-merge (auto-merge)
$ git checkout main
Switched to branch 'main'

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prueba-
auto-merge (main)
$ echo "Footer: Fin del archivo" >> file.txt

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prueba-
auto-merge (main)
$ git add file.txt
warning: in the working copy of 'file.txt', LF will be replaced by CRLF the next ti
me Git touches it

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prueba-
auto-merge (main)
$ git commit -m "Agrega footer en file.txt"
[main 4fd780c] Agrega footer en file.txt
 1 file changed, 1 insertion(+)

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prueba-
auto-merge (main)
```

Fusionamos automerge en main:

```
auto-merge (main)
$ git add file.txt
warning: in the working copy of 'file.txt', LF will be replaced by CRLF the next ti
me Git touches it

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prueba-
auto-merge (main)
$ git commit -m "Agrega footer en file.txt"
[main 4fd780c] Agrega footer en file.txt
 1 file changed, 1 insertion(+)

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prueba-
auto-merge (main)
$ git merge auto-merge
Auto-merging file.txt
CONFLICT (content): Merge conflict in file.txt
Automatic merge failed; fix conflicts and then commit the result.

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/prueba-
auto-merge (main|MERGING)
$ git revert -m 1 HEAD
error: Reverting is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: revert failed
```

Vemos el historial de commits:

```
auto-merge (main|MERGING)
$ git log --graph --oneline
* 4fd780c (HEAD -> main) Agrega footer en file.txt
* 773ea5b Agrega linea 2
* 8f1c8e0 Agrega linea 1
```

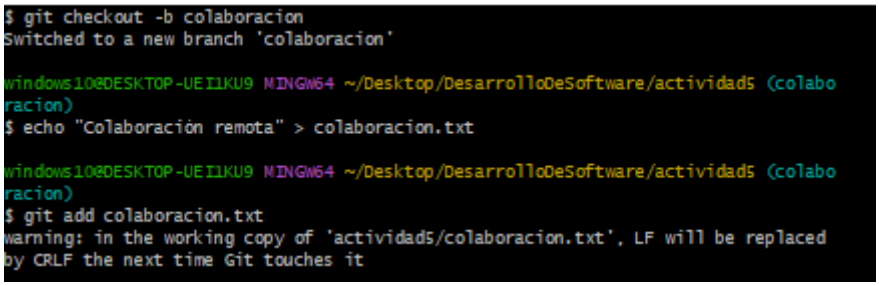
. ¿Cuándo usarías un comando como git revert para deshacer una fusión?

Usarías git revert para deshacer una fusión cuando ya hiciste push o querés mantener el historial intacto, ya que crea un nuevo commit que revierte los cambios sin borrar nada.

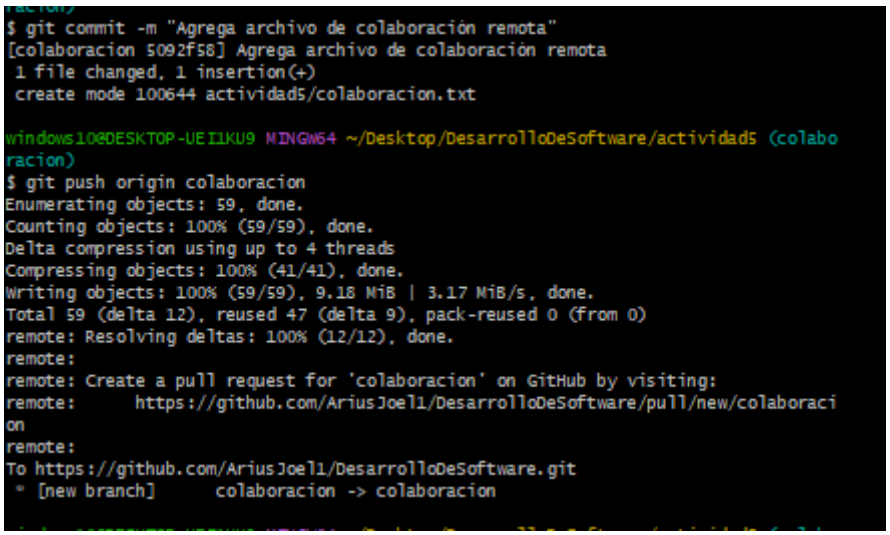
. ¿Qué tan útil es la función de fusión automática en Git?

La fusión automática en Git es muy útil porque ahorra tiempo al combinar ramas sin conflictos, pero si hay cambios en las mismas líneas, requiere intervención manual.

Creamos nueva rama para hacer cambios y realizamos commit



Subimos (pusheamos) al repositorio



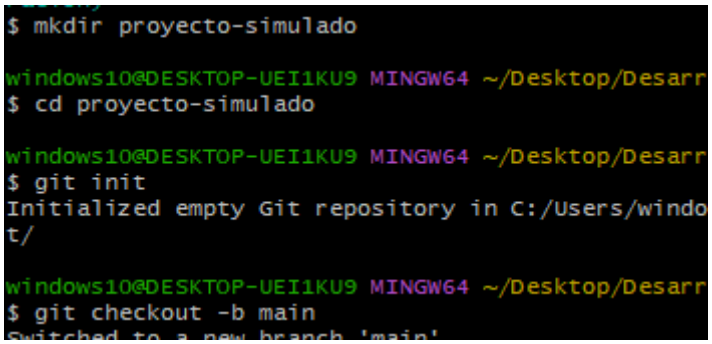
Preguntas:
¿Cómo cambia la estrategia de fusión cuando colaboras con otras personas en un repositorio remoto?

Cuando colaboras con otras personas en un repositorio remoto, la estrategia de fusión se enfoca en usar Pull Requests (PRs) para revisar y aprobar cambios antes de fusionarlos, lo que permite detectar conflictos y evitar errores. Además, es común realizar git pull frecuentemente para mantener el repositorio actualizado y evitar problemas de sincronización.

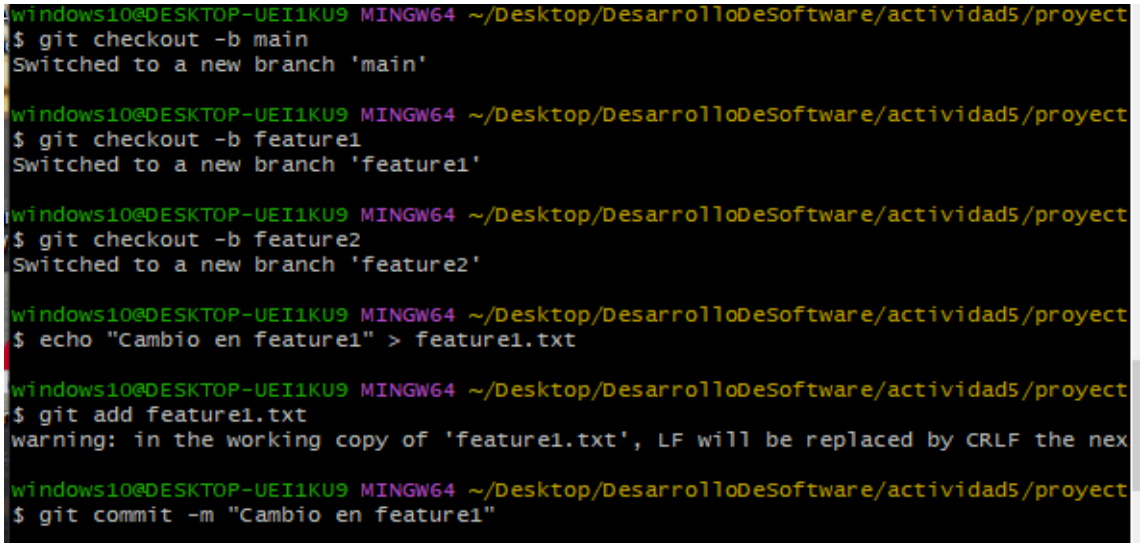
¿Qué problemas comunes pueden surgir al integrar ramas remotas?

Los problemas comunes al integrar ramas remotas incluyen conflictos de fusión cuando dos personas modifican las mismas líneas de código, commits desordenados o incorrectos, y la falta de comunicación entre colaboradores, lo que puede llevar a cambios inconsistentes o incompatibles en el código.

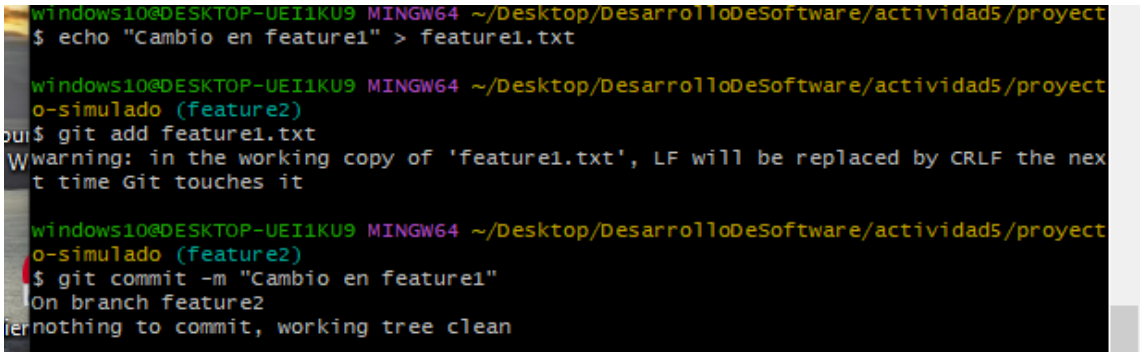
Creamos proyecto con 3 ramas:



Entramos en cada rama:



Realizamos cambios en la rama 1:



Ahora para la rama2

```
o-simulado (feature2)
$ echo "Cambio en feature2" > feature2.txt

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/proyect
o-simulado (feature2)
$ git add feature2.txt
warning: in the working copy of 'feature2.txt', LF will be replaced by CRLF the nex
t time Git touches it

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/proyect
o-simulado (feature2)
$ git commit -m "Cambio en feature2"
[feature2 29a179f] Cambio en feature2
 1 file changed, 1 insertion(+)
 create mode 100644 feature2.txt
```

Cambiamos a la rama main y fusionamos:

```
o-simulado (feature2)
$ git checkout -b main
Switched to a new branch 'main'

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/
o-simulado (main)
$ git merge --no-ff feature2
Already up to date.

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/
o-simulado (main)
$ git checkout -b feature3
Switched to a new branch 'feature3'
```

Creamos un archivo txt que diga cambio en feature3 y lo agregamos y hacemos un commit, luego agregamos y hacemos otro commit

```
windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/proyect
o-simulado (feature3)
$ echo "Cambio en feature3" > feature3.txt

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/proyect
o-simulado (feature3)
$ git add feature3.txt
warning: in the working copy of 'feature3.txt', LF will be replaced by CRLF the nex
t time Git touches it

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/proyect
o-simulado (feature3)
$ git commit -m "Primer commit en feature3"
[feature3 e3c70c4] Primer commit en feature3
 1 file changed, 1 insertion(+)
 create mode 100644 feature3.txt

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/proyect
o-simulado (feature3)
$ echo "Otro cambio en feature3" >> feature3.txt

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/proyect
o-simulado (feature3)
$ git add feature3.txt
warning: in the working copy of 'feature3.txt', LF will be replaced by CRLF the nex
t time Git touches it

windows10@DESKTOP-UEI1KU9 MINGW64 ~/Desktop/DesarrolloDeSoftware/actividad5/proyect
o-simulado (feature3)
$ git commit -m "Segundo commit en feature3"
[feature3 bf5ca9f] Segundo commit en feature3
 1 file changed, 1 insertion(+)
```

Cambiamos nuevamente a la rama principal, fusionamos feature3 con main y revisamos el historial de commits: