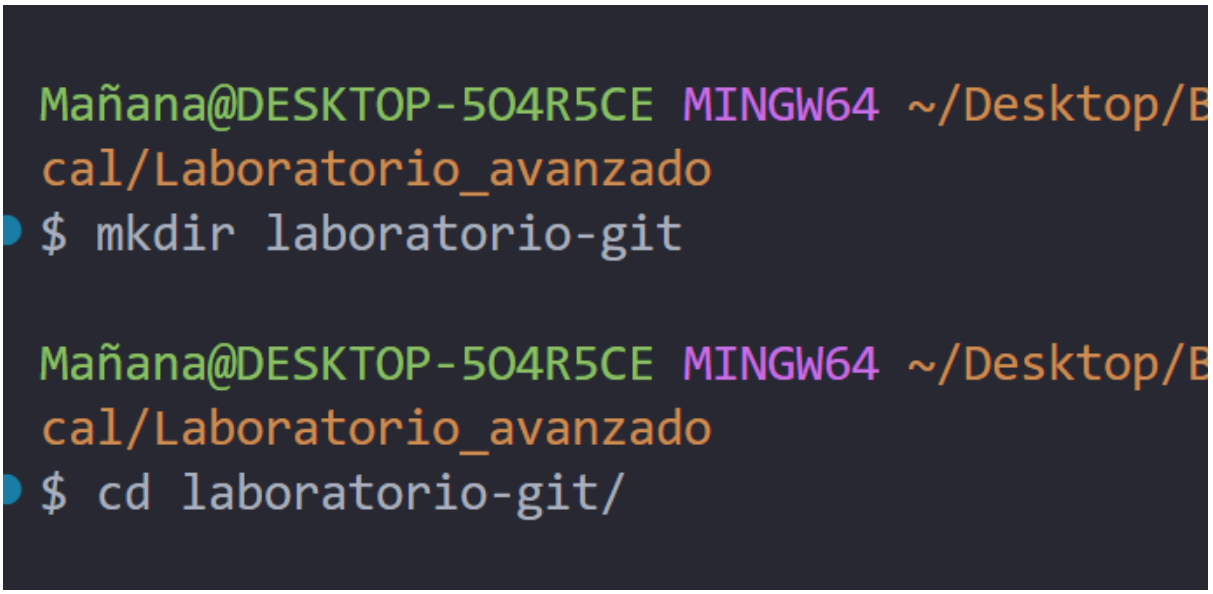
LABORATORIO_AVANZADO

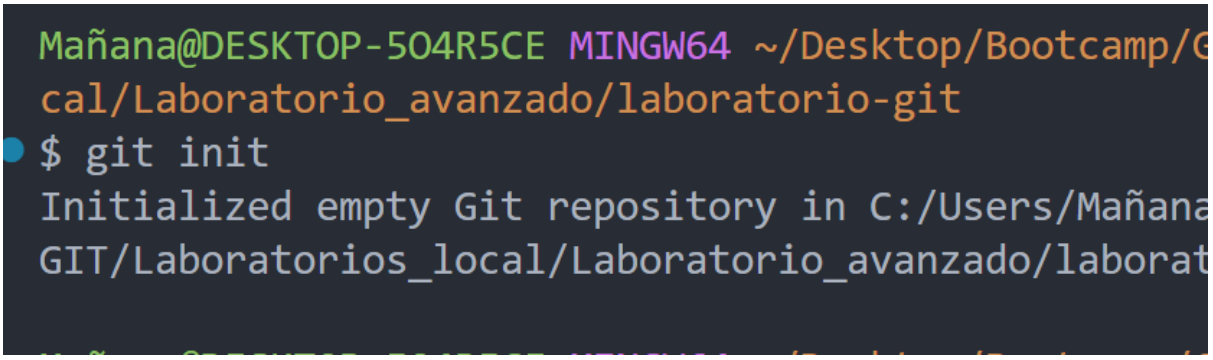
Laboratorio #1 cherry-pick básico

1. Crear un repositorio Git nuevo

Vamos a crear un nuevo directorio y nos metemos en el:



Iniciamos el repositorio con el init:



2. Crea tu primer commit en la rama main

Vamos a crear el primer commit normalmente:

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratori
cal/Laboratorio_avanzado/laboratorio-git (master)
$ echo "Este es el archivo principal" > archivo.txt

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratori
cal/Laboratorio_avanzado/laboratorio-git (master)
$ git add archivo.txt
warning: in the working copy of 'archivo.txt', LF will be replac
 CRLF the next time Git touches it

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratori
cal/Laboratorio_avanzado/laboratorio-git (master)
$ git commit -m "Primer commit en la rma main"
[master (root-commit) 2e224af] Primer commit en la rma main
 1 file changed, 1 insertion(+)
```

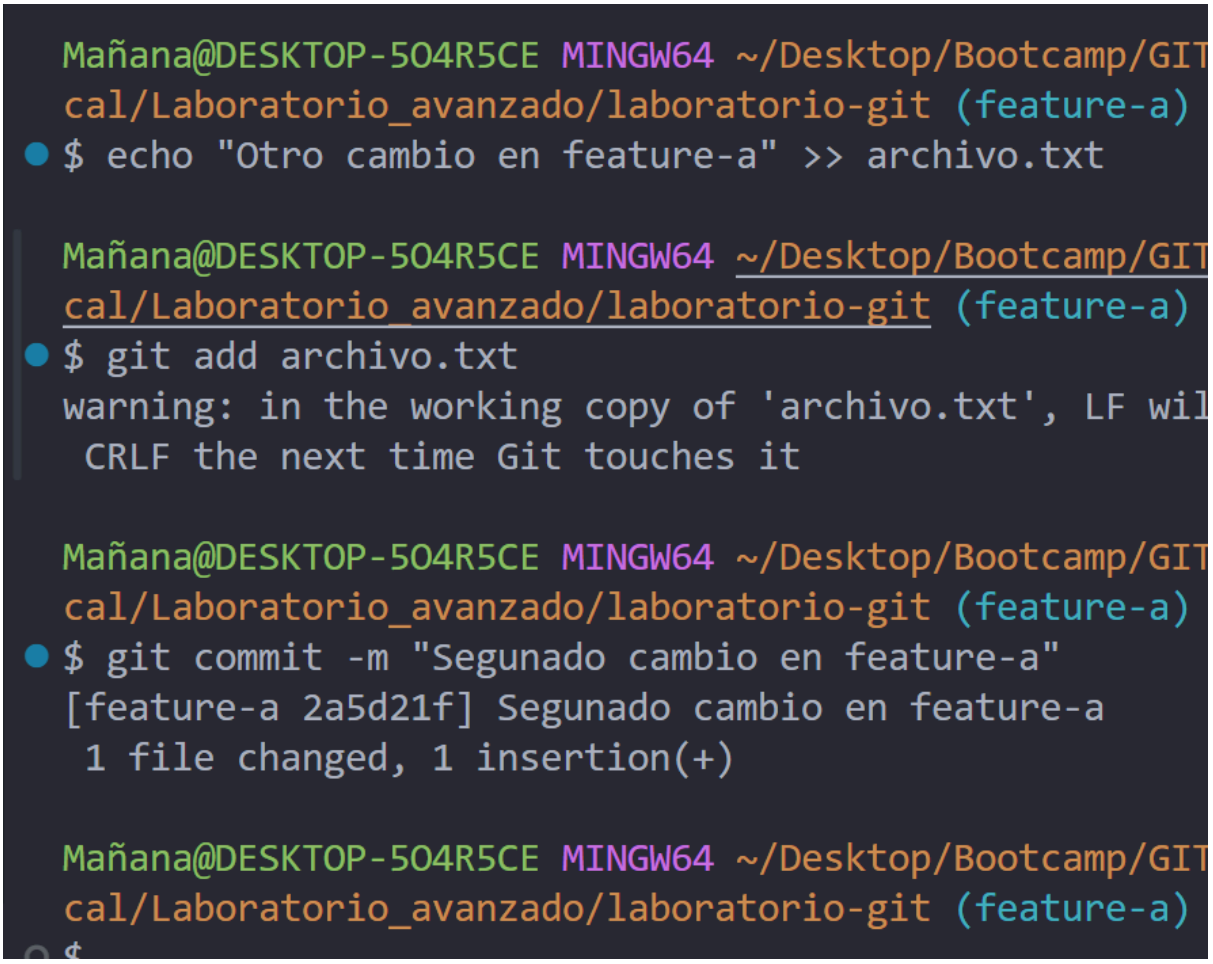
3.Crear una nueva rama y hacer commits

Creamos la rama y guardamos los cambios correctamente:

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_lo
cal/Laboratorio_avanzado/laboratorio-git (master)
$ git checkout -b feature-a
Switched to a new branch 'feature-a'

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_lo
cal/Laboratorio_avanzado/laboratorio-git (feature-a)
$ echo "Este es un cambio en feature-a" >> archivo.txt && git add arc
hivo.txt
warning: in the working copy of 'archivo.txt', LF will be replaced by
 CRLF the next time Git touches it
```

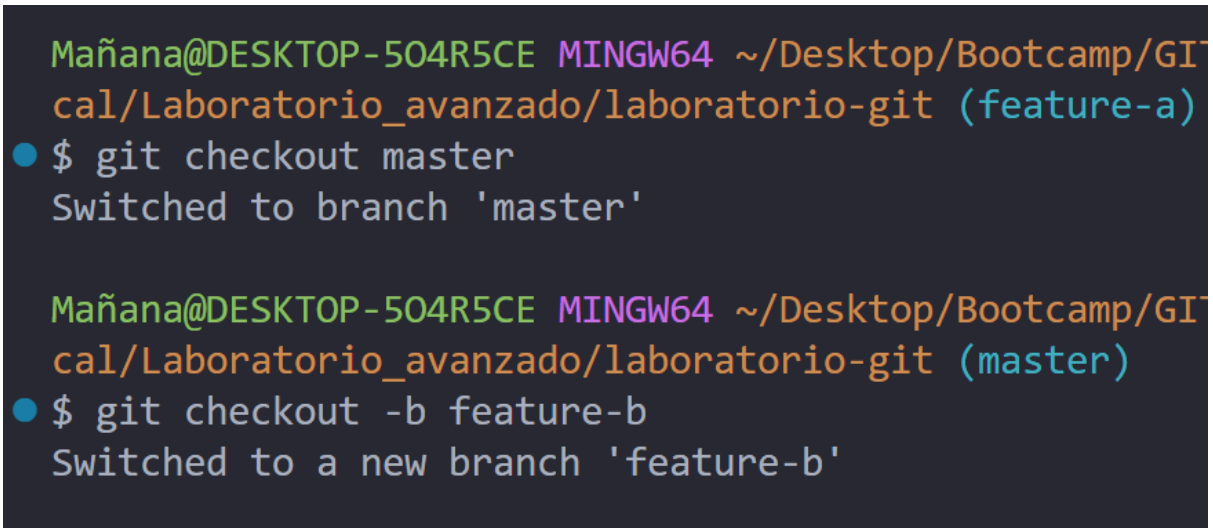
```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp
cal/Laboratorio_avanzado/laboratorio-git (feature
$ git commit -m "Cambio en feature-a"
[feature-a 71db471] Cambio en feature-a
 1 file changed, 1 insertion(+)
```

Ahora vamos a hacer otro **commit** en la misma rama:



4. Crear otra rama y hacer commits

Volvemos a la rama **master** y creamos otra rama:



Y creamos un commit:

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/La
cal/Laboratorio_avanzado/laboratorio-git (feature-b)
$ echo "Este es un cambio en feature-b" >> archivo.txt

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/La
cal/Laboratorio_avanzado/laboratorio-git (feature-b)
$ git add archivo.txt
warning: in the working copy of 'archivo.txt', LF will b
 CRLF the next time Git touches it

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/La
cal/Laboratorio_avanzado/laboratorio-git (feature-b)
$ git commit -m "Cambio en feature-b"
[feature-b 7ce8059] Cambio en feature-b
 1 file changed, 1 insertion(+)
```

5. Practica el comando cherry-pick

Vamos a verificar los commits de la rama **feature-a:**

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/La
cal/Laboratorio_avanzado/laboratorio-git (feature-b)
$ git log --oneline feature-a
2a5d21f (feature-a) Segunado cambio en feature-a
71db471 Cambio en feature-a
2e224af (master) Primer commit en la rma main
```

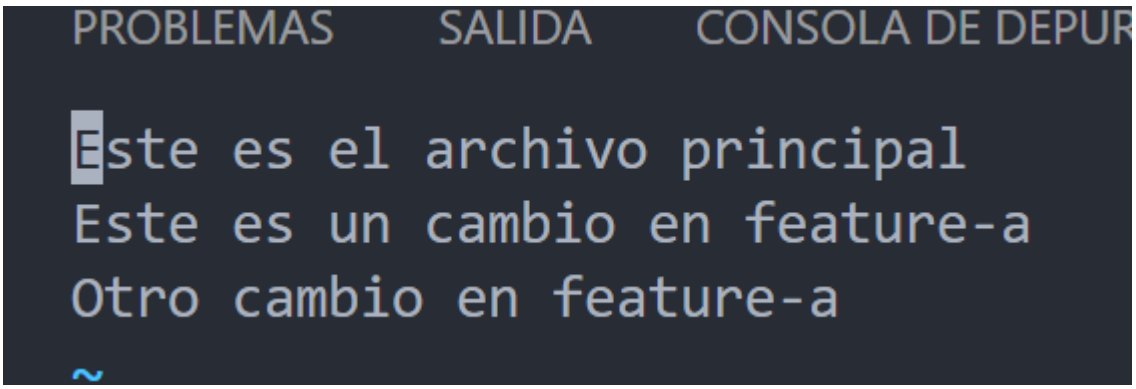
Y debemos recordar un identificador SHA de algun commit
Volvemos al master

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/L
cal/Laboratorio_avanzado/laboratorio-git (feature-b)
$ git checkout master
Switched to branch 'master'
```

Y hacemos un cherry pick

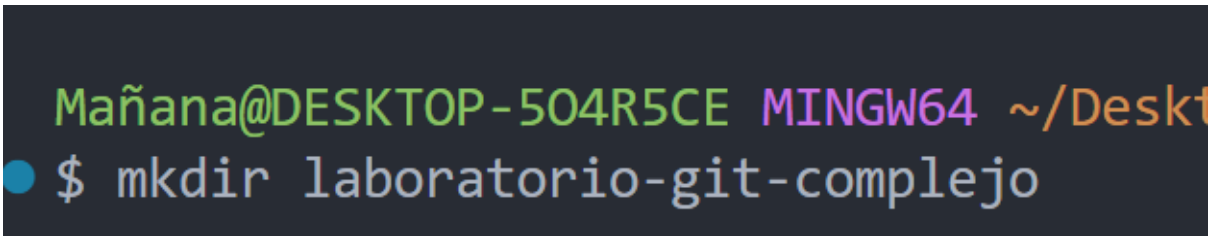
```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorio
cal/Laboratorio_avanzado/laboratorio-git (master|CHERRY-PICKING)
$ git cherry-pick 71db471
[master 8415729] Cambio en feature-a
 Date: Tue Mar 18 09:33:16 2025 +0100
 1 file changed, 1 insertion(+)
```

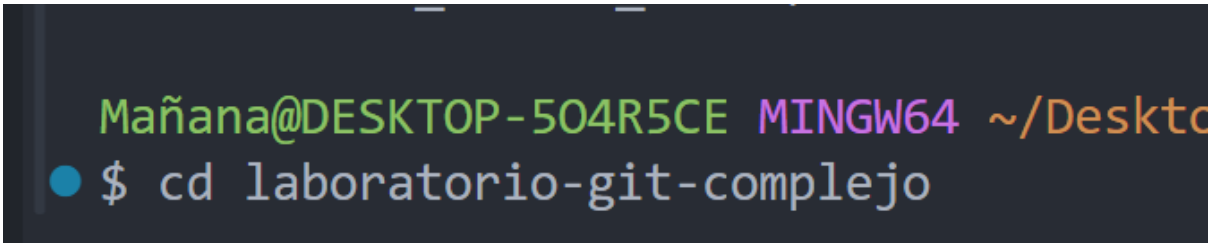
Vamos a entrar en el vim

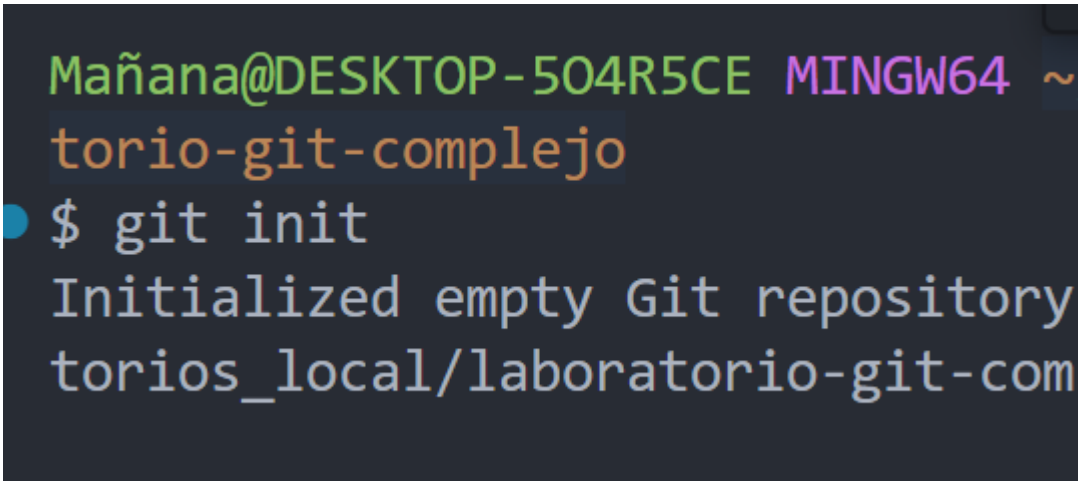


Laboratorio #2 entorno más realista

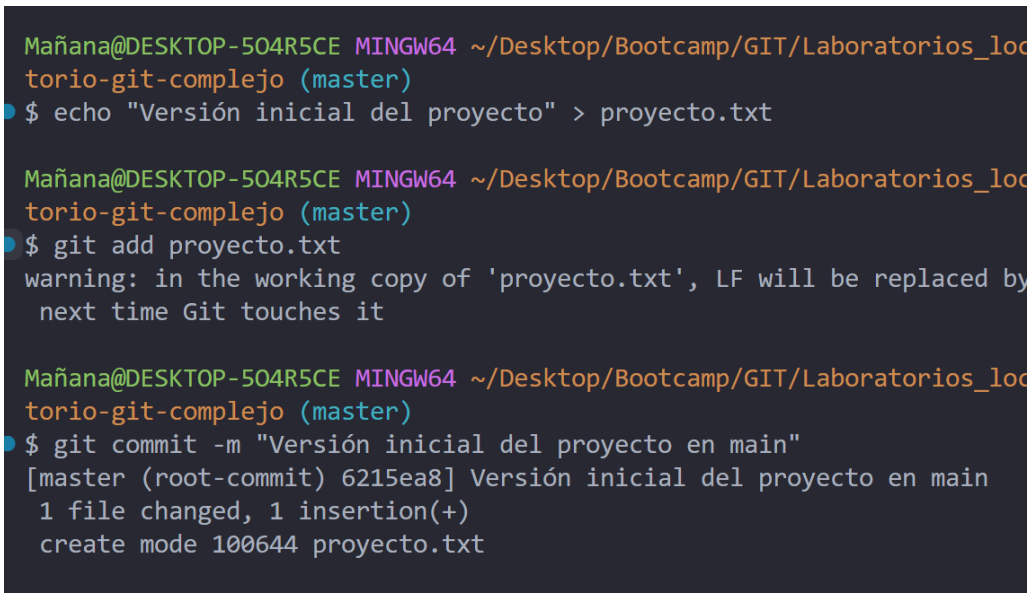
1. Inicializar el repositorio y crear el entorno básico





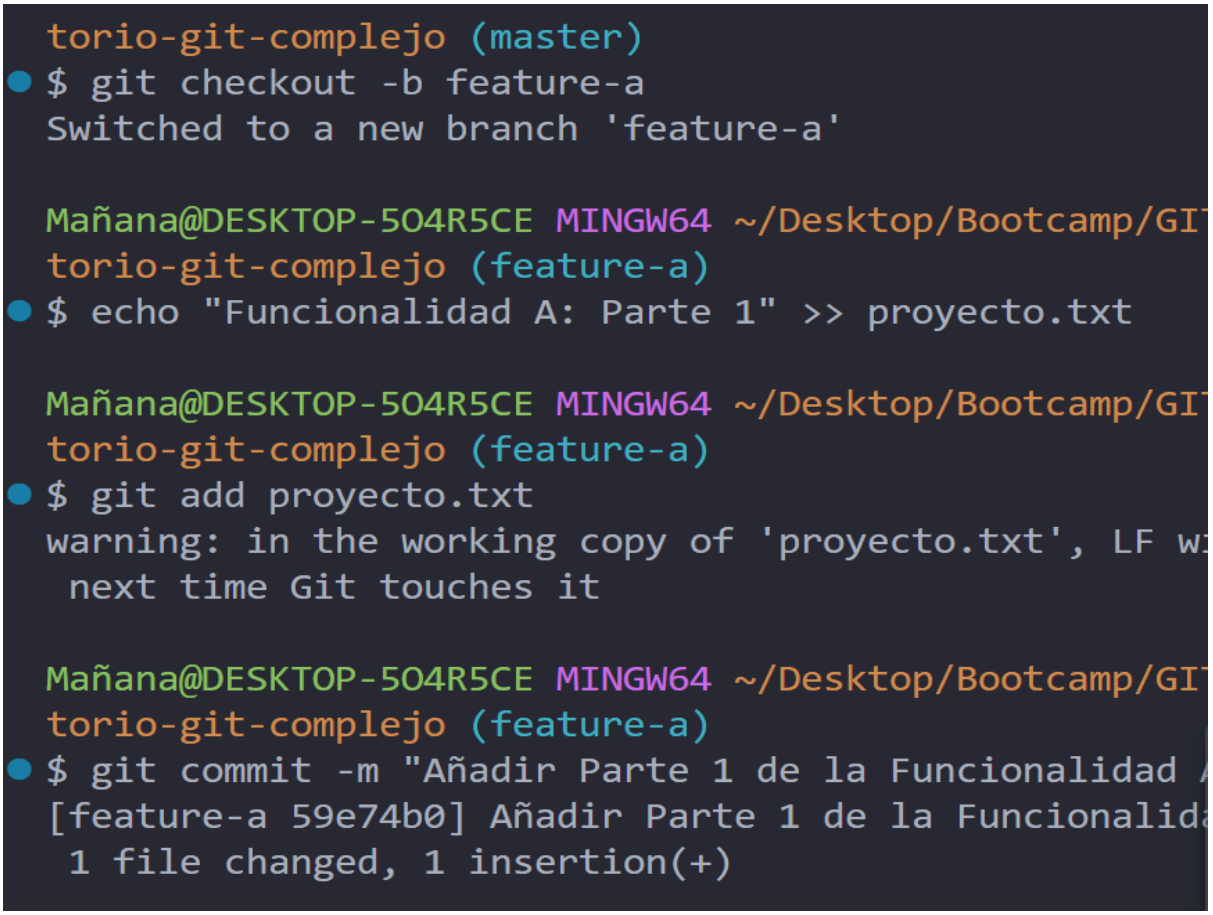


2. Crear la rama principal main y hacer un commit inicial

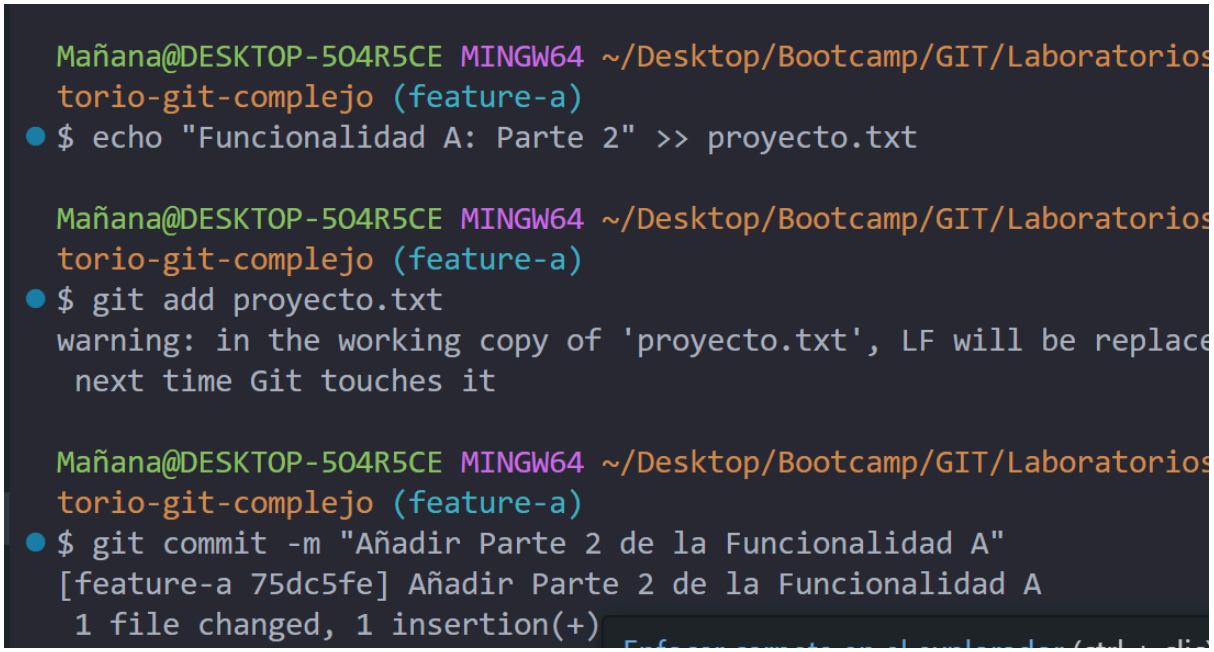


3. Crear y desarrollar la rama feature-a

Creamos la rama **feature-a**:



Modificamos la rama:



4. Crear y desarrollar la rama feature-b

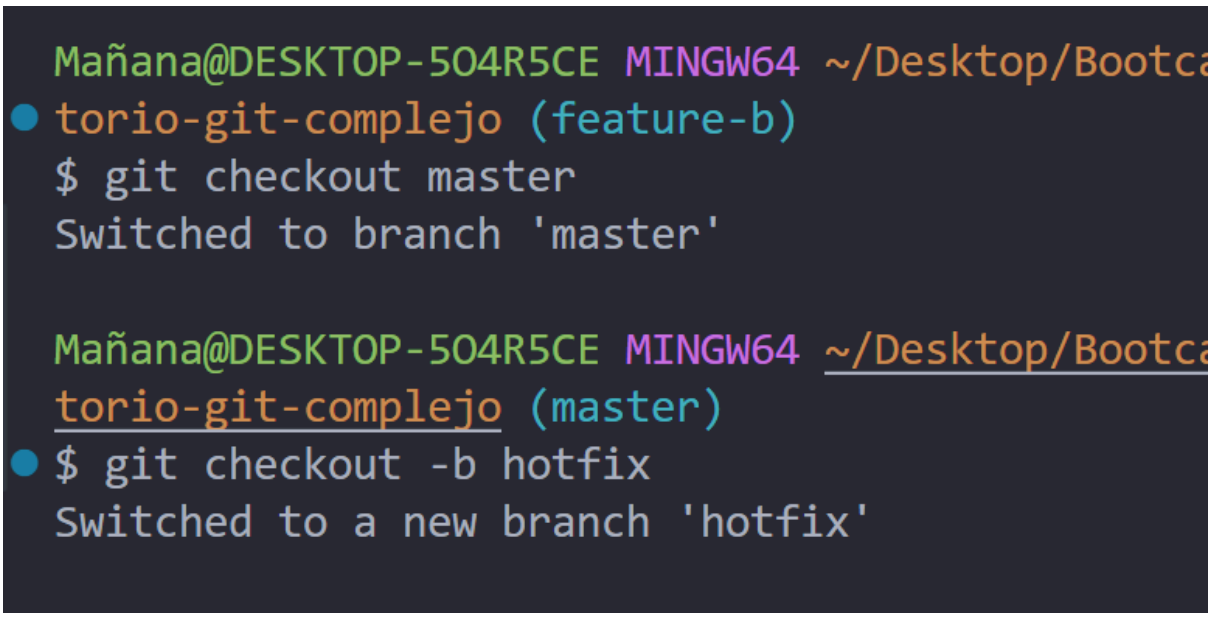
```
torio-git-complejo (master)
$ git checkout -b feature-b
Switched to a new branch 'feature-b'

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laborat
torio-git-complejo (feature-b)
$ echo "Funcionalidad B: Parte 1" >> proyecto.txt

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laborat
torio-git-complejo (feature-b)
$ git add proyecto.txt
warning: in the working copy of 'proyecto.txt', LF will be re
 next time Git touches it

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laborat
torio-git-complejo (feature-b)
$ git commit -m "Añadir Parte 1 de la Funcionalidad B"
[feature-b 4743b6b] Añadir Parte 1 de la Funcionalidad B
 1 file changed, 1 insertion(+)
```

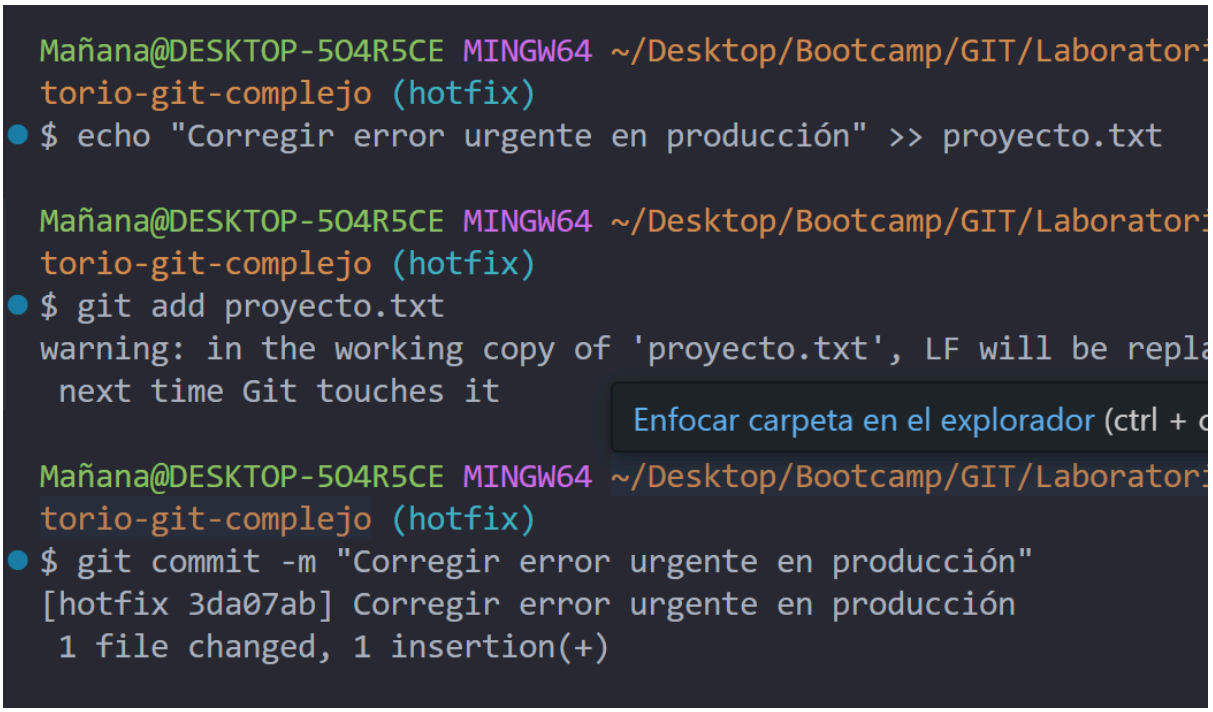
Ahora modificamos la rama **feature-b:**

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laborator
torio-git-complejo (feature-b)
$ echo "Funcionalidad B: Parte 2" >> proyecto.txt

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laborator
torio-git-complejo (feature-b)
$ git add proyecto.txt
warning: in the working copy of 'proyecto.txt', LF will be repl
 next time Git touches it

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laborator
torio-git-complejo (feature-b)
$ git commit -m "Añadir Parte 2 de la Funcionalidad B"
[feature-b c9aef57] Añadir Parte 2 de la Funcionalidad B
 1 file changed, 1 insertion(+)
```

5. Crear una rama de hotfix para resolver problemas urgentes

Volvemos a la **rama main** y creamos la **rama hotfix:**





6. Usar cherry-pick para traer cambios de feature-a y feature-b a hotfix

Vamos a mirar los commits de la rama **feature-a:**

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laborator
orio-git-complejo (hotfix)
$ git log --oneline feature-a
75dc5fe (feature-a) Añadir Parte 2 de la Funcionalidad A
59e74b0 Añadir Parte 1 de la Funcionalidad A
6215ea8 (master) Versión inicial del proyecto en main
```

Vemos que hay un **conflicto:**

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_local/labora
orio-git-complejo (hotfix)
$ git cherry-pick 75dc5fe
Auto-merging proyecto.txt
CONFLICT (content): Merge conflict in proyecto.txt
error: could not apply 75dc5fe... Añadir Parte 2 de la Funcionalidad A
hint: After resolving the conflicts, mark them with
hint: "git add/rm <pathspec>", then run
hint: "git cherry-pick --continue".
hint: You can instead skip this commit with "git cherry-pick --skip".
hint: To abort and get back to the state before "git cherry-pick",
hint: run "git cherry-pick --abort".
hint: Disable this message with "git config set advice.mergeConflict false"
```

Ahora veamos los commits de la rama **feature-b**

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laborat
orio-git-complejo (hotfix|CHERRY-PICKING)
$ git log --oneline feature-b
c9aef57 (feature-b) Añadir Parte 2 de la Funcionalidad B
4743b6b Añadir Parte 1 de la Funcionalidad B
6215ea8 (master) Versión inicial del proyecto en main
```

Vemos también que hay conflictos:

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_local/L
orio-git-complejo (hotfix|CHERRY-PICKING)
$ git cherry-pick 4743b6b
error: Cherry-picking 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: cherry-pick failed
```

7. Resolver conflictos (opcional)
PAZO AZIN

```
Versión inicial del proyecto
<<<<<<< HEAD
Corregir error urgente en producción
=======
Funcionalidad A: Parte 1
Funcionalidad A: Parte 2
>>>>>>> 75dc5fe (Añadir Parte 2 de la Funcionalidad A)
~
~
~
```

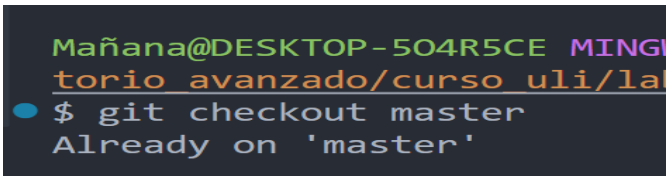
Laboratorio #3 escenario más complejo(cont.)

1. Continuar desde el laboratorio anterior

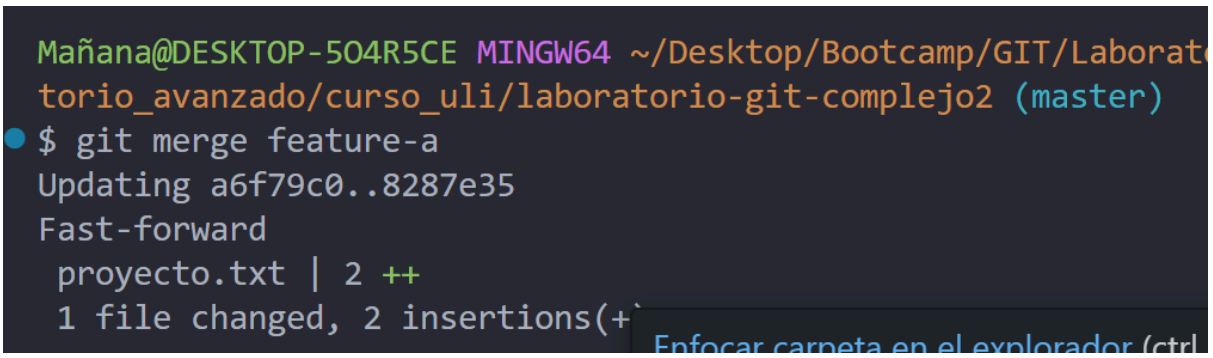
Primero debíamos volver a estar en el entorno configurado anteriormente con las ramas **main , feature-a , feature-b , y hotfix**

2. Fusionar feature-a en main

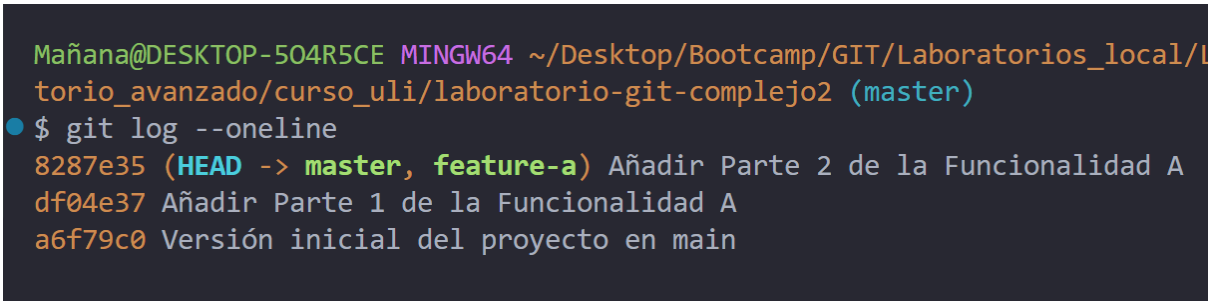
Vamos a la rama **main**



Ahora le hacemos un **merge** desde la rama **master** a la rama **feature-a** y podemos comprobar que no hay conflicto:

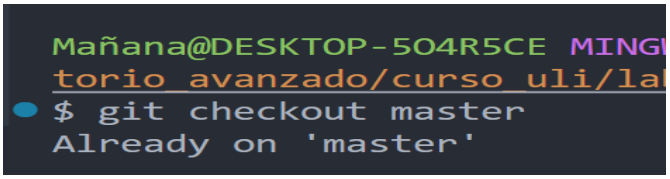


Verificamos el estado de la fusión:

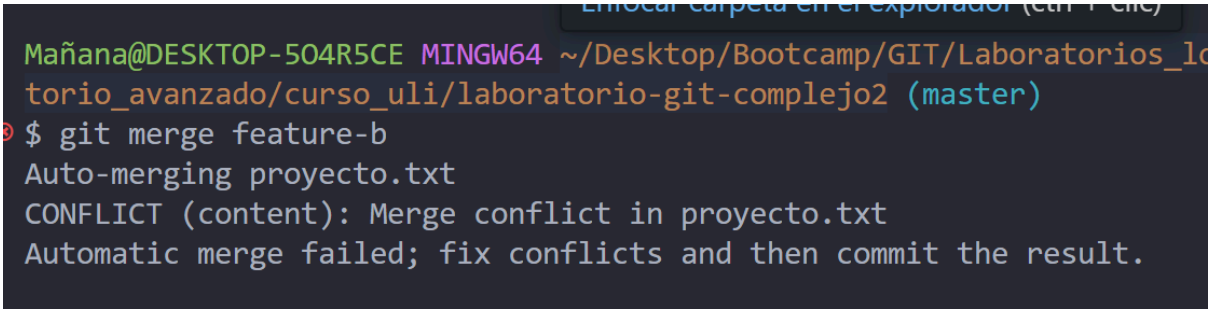


3. Fusionar feature-b en main (simulación de conflicto)

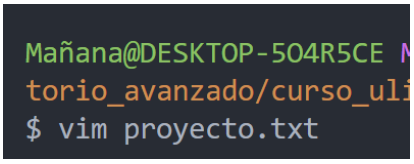
Volvemos a la rama master:

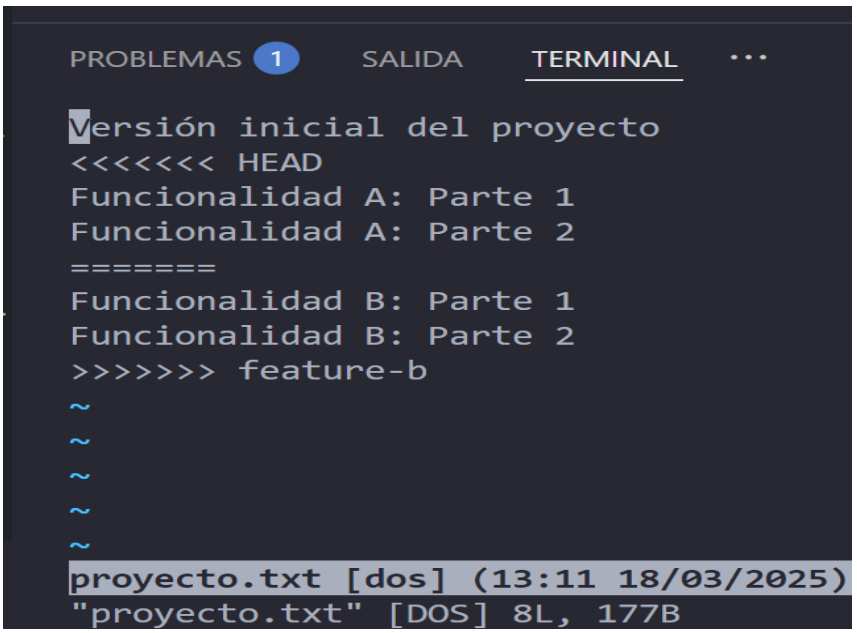


Le hacemos un **merge** y vemos que hay un conflicto esperado:



Abrimos el proyecto.txt con el vim:





Una vez visto el problema lo arreglamos:

```
Versión inicial del proyecto
HEAD
Funcionalidad A: Parte 1
Funcionalidad A: Parte 2

Funcionalidad B: Parte 1
Funcionalidad B: Parte 2
feature-b
~
```

Vemos que se ha realizado correctamente:

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_lo
torio_avanzado/curso_uli/laboratorio-git-complejo2 (master)
$ git log --oneline --graph
*   0400f0b (HEAD -> master) Merge branch 'feature-b'
|\
| * fcb787a (feature-b) Añadir Parte 2 de la Funcionalidad B
| * e9ba7a6 Añadir Parte 1 de la Funcionalidad B
* | 8287e35 (feature-a) Añadir Parte 2 de la Funcionalidad A
* | df04e37 Añadir Parte 1 de la Funcionalidad A
|/
* a6f79c0 Versión inicial del proyecto en main
```

4. Realizar un revert de un commit problemático

Hago un **git log** para obtener el identificador (SHA) del commit que deseo **revertir:**

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorio
torio_avanzado/curso_uli/laboratorio-git-complejo2 (master)
$ git log --oneline
0400f0b (HEAD -> master) Merge branch 'feature-b'
fcb787a (feature-b) Añadir Parte 2 de la Funcionalidad B
e9ba7a6 Añadir Parte 1 de la Funcionalidad B
8287e35 (feature-a) Añadir Parte 2 de la Funcionalidad A
df04e37 Añadir Parte 1 de la Funcionalidad A
a6f79c0 Versión inicial del proyecto en main
```

Ahora ahora hago un git revert SHA y (Esto creará un nuevo commit que revierte los cambios hechos en ese commit específico, sin eliminar el historial de Git)

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_local
torio_avanzado/curso_uli/laboratorio-git-complejo2 (master|REVERTING)
$ git revert --continue
[master 8fd621d] Revert "Añadir Parte 2 de la Funcionalidad B"
 1 file changed, 2 insertions(+)
```

Y verificamos:

```
$ git log --oneline
8fd621d (HEAD -> master) Revert "Añadir Parte 2 de la Funcionalidad B"
0400f0b Merge branch 'feature-b'
fcb787a (feature-b) Añadir Parte 2 de la Funcionalidad B
e9ba7a6 Añadir Parte 1 de la Funcionalidad B
8287e35 (feature-a) Añadir Parte 2 de la Funcionalidad A
df04e37 Añadir Parte 1 de la Funcionalidad A
a6f79c0 Versión inicial del proyecto en main
```

Ahora podemos ver un **nuevo commit** que describe que se ha **revertido el commit**

5. Crear una nueva rama de hotfix a partir de main

Imaginemos que después de **revertir el commit,** necesitamos aplicar un **hotfix urgente.**
Voy a crear una nueva rama **hotfix-v2** a partir de **main :**
Y también haremos una corrección urgente y guardaremos el **commit:**

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_local/Labora
torio_avanzado/curso_uli/laboratorio-git-complejo2 (master)
$ git checkout -b hotfix-v2
Switched to a new branch 'hotfix-v2'

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_local/Labora
torio_avanzado/curso_uli/laboratorio-git-complejo2 (hotfix-v2)
$ echo "Hotfix v2: Corrección urgente después del revert" >> proyecto.txt

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_local/Labora
torio_avanzado/curso_uli/laboratorio-git-complejo2 (hotfix-v2)
$ git add proyecto.txt
warning: in the working copy of 'proyecto.txt', LF will be replaced by CRLF the
 next time Git touches it

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_local/Labora
torio_avanzado/curso_uli/laboratorio-git-complejo2 (hotfix-v2)
$ git commit -m "Hotfix v2: Corrección urgente después del revert"
[hotfix-v2 fc45250] Hotfix v2: Corrección urgente después del revert
 1 file changed, 1 insertion(+)
```

Ahora vamos a fusionar **hotfix-v2** con **main:**

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_loc
torio_avanzado/curso_uli/laboratorio-git-complejo2 (hotfix-v2)
$ git checkout master
Switched to branch 'master'

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_loc
torio_avanzado/curso_uli/laboratorio-git-complejo2 (master)
$ git merge hotfix-v2
Updating 8fd621d..fc45250
Fast-forward
 proyecto.txt | 1 +
 1 file changed, 1 insertion(+)
```

6. Fusionar hotfix en una rama de producción

Vamos a crear una **rama de producción** para simular que todo está preparado para el lanzamiento y la fusionaremos con la **versión corregida y los hotfixes aplicados**:

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_lo
torio_avanzado/curso_uli/laboratorio-git-complejo2 (master)
$ git checkout -b production
Switched to a new branch 'production'

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_lo
torio_avanzado/curso_uli/laboratorio-git-complejo2 (production)
$ git merge master
Already up to date.
```

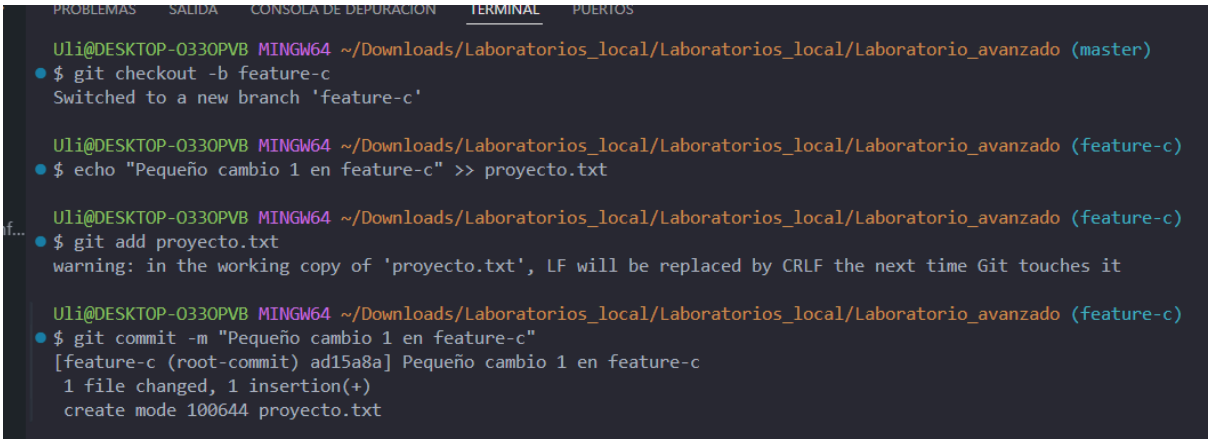
Como **main** ya incluye todas las **fusiones y hotfixes**, ahora la rama **production** está lista para ser lanzada con la versión corregida.

Este flujo simula un entorno de desarrollo con varios equipos trabajando en diferentes características, donde algunos cambios causan problemas y deben revertirse, y otros necesitan ser aplicados de inmediato como hotfixes. Además, se simulan conflictos de fusión, algo muy común en proyectos grandes.

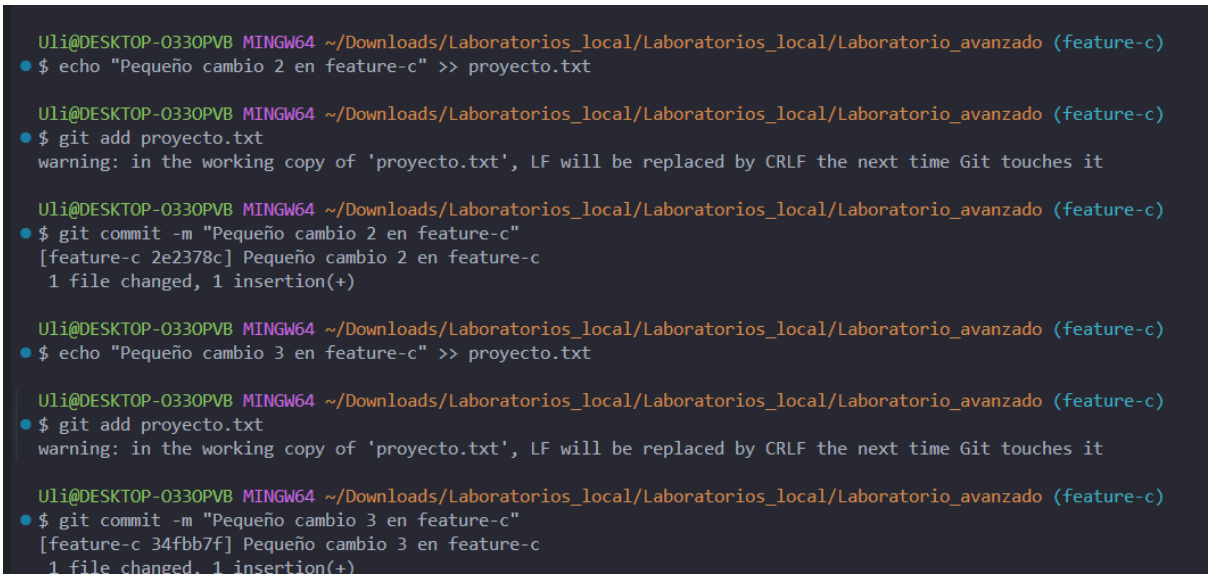
Laboratorio #4 todavía más funcionalidades (cont.)

1. Crear la rama feature-c y hacer varios commits pequeños

Creamos la rama **feature-c** y le hacemos unos commits de desarrollo:



Hacemos otros **commits** como si fueran pequeños cambios:



2. Usar git rebase para actualizar feature-c con la última versión de main

Mientras trabajabamos en **feature-c** , la rama **main** ha recibido nuevas actualizaciones. Para asegurarnos de que el trabajo en **feature-c** esté alineado con la última versión de main , podemos hacer un **rebase**.

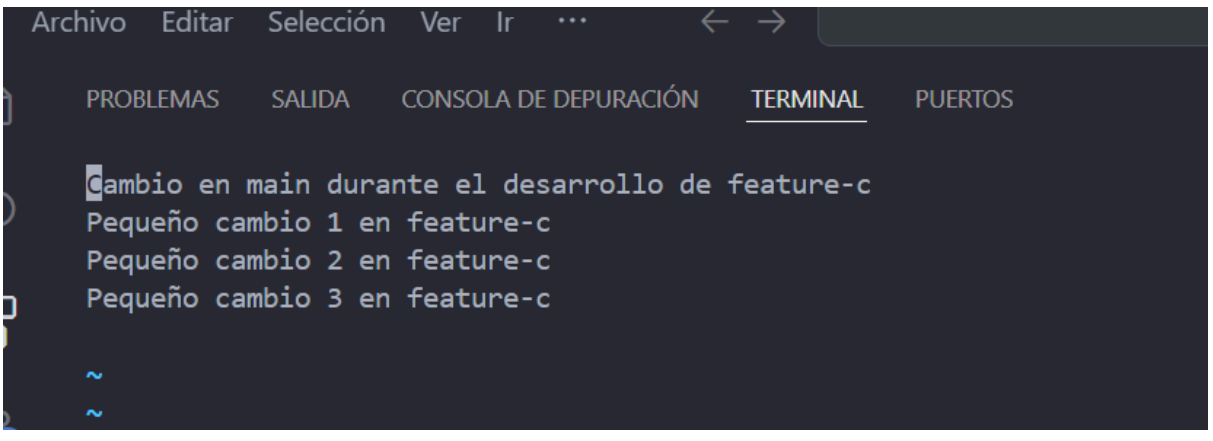
Primero vamos a volver a la rama **master** y nos aseguramos de que hayan cambios nuevos:

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_local/Laboratorio_avanzado/laboratorio-git (feature-c
$ git checkout master
D       archivo.txt
Switched to branch 'master'

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_local/Laboratorio_avanzado/laboratorio-git (master)
$ echo "Cambio en main durante el desarrollo de feature-c">> proyecto.txt

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_local/Laboratorio_avanzado/laboratorio-git (master)
$ git add proyecto.txt
warning: in the working copy of 'proyecto.txt', LF will be replaced by CRLF the next time Git touches it

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_local/Laboratorio_avanzado/laboratorio-git (master)
$ git commit -m "Cambio en main"
[master 8117593] Cambio en main
 1 file changed, 1 insertion(+)
 create mode 100644 proyecto.txt
```

Ahora volvemos a la rama feature-c y hacemos el **rebase** sobre la **última versión de main :**
Podemos ver que detecta conflicto al realizar un **rebase:**

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_local/Laboratorio_avanz
$ git rebase master
Auto-merging proyecto.txt
CONFLICT (add/add): Merge conflict in proyecto.txt
error: could not apply 67178cc... Pequeño cambio 1 en feature-c
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git rebase --abort".
hint: Disable this message with "git config set advice.mergeConflict false"
Could not apply 67178cc... Pequeño cambio 1 en feature-c
```

Vamos a modificar el **proyecto.txt:**



Una vez arreglado vamos a realizar un **merge** para combinar los cambios y a volver a realizar el **rebase** con el **- -continue:**

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_local/Laboratorio_ava
$ git add proyecto.txt

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_local/Laboratorio_ava
$ git commit -m "Archivo corregido"
[detached HEAD 5de7124] Archivo corregido
 1 file changed, 2 insertions(+), 4 deletions(-)

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_local/Laboratorio_ava
$ git merge master
Already up to date.

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_local/Laboratorio_ava
$ git rebase --continue
Successfully rebased and updated refs/heads/feature-c.
```

3. Usar git squash para combinar commits

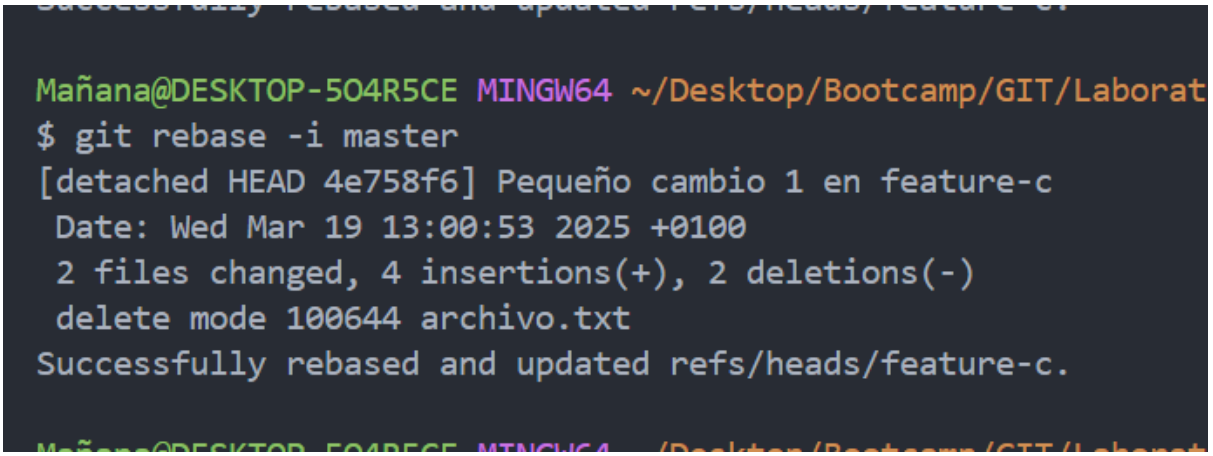
Ahora que **feature-c** está alineada con **main** , es momento de **squash** (comprimir) los commits pequeños en un solo **commit** coherente. Vamos a usar **rebase interactivo** para lograrlo con **git rebase -i**:

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACION  TERMINAL  PUERTOS

pick 6de12d9 Pequeño cambio 1 en feature-c
pick ca284c7 Pequeño cambio 2 en feature-c
pick 5de7124 Archivo corregido
pick e20281b Archivo borrado
```

Ahora para combinar los **commits** en uno solo, vamos a cambiar las líneas **pick** de los commits que queremos comprimir por **squash** (o simplemente s), excepto el primero:

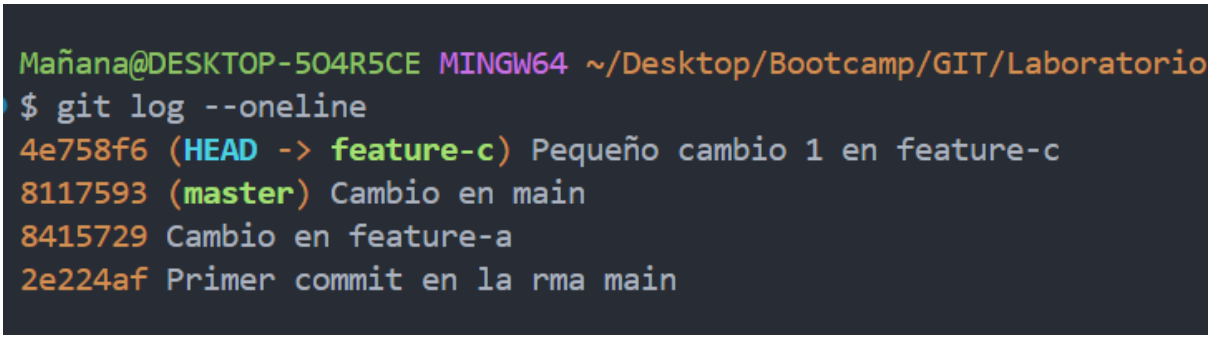
```
pick 6de12d9 Pequeño cambio 1 en feature-c
squash ca284c7 Pequeño cambio 2 en feature-c
squash 5de7124 Archivo corregido
squash e20281b Archivo borrado
```

Guardamos y cerramos el archivo para completar el **rebase interactivo.**

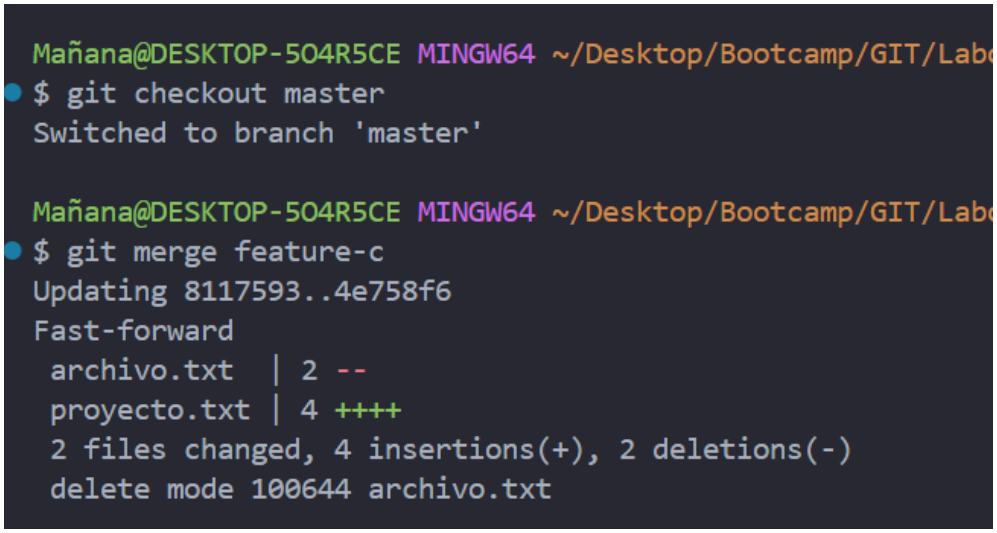


4. Verificar el historial y fusionar la rama en main

Verificamos que todos los commits pequeños se hayan comprimido en uno solo con **git log - -online:**



Cambiamos al **master** y fusionamos con el **merge** las dos ramas y le hacemos un **git log –oneline** para verificar que los cambios se hayan fusionado correctamente y que el historial sea limpio:



Vemos que se ha fusionado correctamente:

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_loc
$ git log --oneline
4e758f6 (HEAD -> master, feature-c) Pequeño cambio 1 en feature-c
8117593 Cambio en main
8415729 Cambio en feature-a
2e224af Primer commit en la rma main
```

5. Flujos de trabajo adicionales: Rebase sobre una rama remota

Por una mala configuración de mi repositorio remoto no me estaba ejecutando el **git fetch** y me enseñaba este error ya que no tenia conectada mi repositorio de github (Este paso no hace falta hacerlo si tienes bien configurado con anterioridad el repositorio remoto):

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios
$ git fetch origin
fatal: 'origin' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
```

Solo tuve que añadirlo a partir del comando **git remote add** y para verificarlo hice el r**emote -v:**

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_local
$ git remote add origin https://github.com/ulippuden99/curso_uli.git

Mañana@DESKTOP-5O4R5CE MINGW64 ~/Desktop/Bootcamp/GIT/Laboratorios_local
$ git remote -v
origin  https://github.com/ulippuden99/curso_uli.git (fetch)
origin  https://github.com/ulippuden99/curso_uli.git (push)
```

Ahora ya podemos hacer el **git fetch** para que se actualice nuestra rama remota:

```
Mañana@DESKTOP-5O4R5CE MINGW64 ~/Deskto
$ git fetch origin
```

Luego podemos **rebasar** nuestra rama local sobre la versión más reciente de la rama remota con **git rebase origin/master** :

NO ME SALE ASI Q ESTO ULTIMO QUE LE DEN

Este flujo de trabajo es común en proyectos de software reales y te ayuda a mantener un historial de commits limpio y fácil de entender.