

DAM

Contornos de desenvolvimento

UD2 **INSTALACIÓN Y USO DE ENTORNOS DE DESARROLLO: RAMIFICACIONES CON GIT**

PROFESORA | CRISTINA PUGA FDEZ

UNIDADE | UD2

TRIMESTRE | 1

MÓDULO | CD

Tabla de contenido

1.	Introducción ramificaciones.....	3
2.	Creación de ramas.....	3
3.	Eliminar ramas.....	8
4.	Fusionar	9
5.	Conflictos en las fusiones	10
6.	Gestión de ramas.....	13
7.	Ramas remotas.....	14
	ANEXO I: Materiales	15

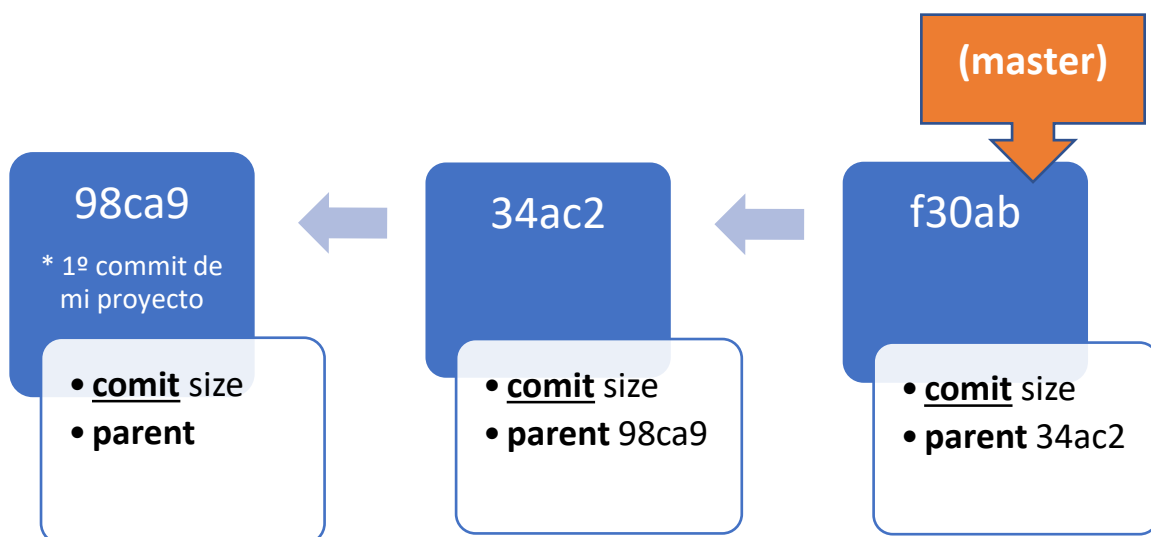
1. Introducción ramificaciones

Cualquier sistema de control de versiones moderno tiene algún mecanismo para soportar el uso de ramas. Cuando hablamos de ramificaciones, significa que tú has tomado la rama principal de desarrollo (master) y a partir de ahí has continuado trabajando sin seguir la rama principal de desarrollo. En muchos sistemas de control de versiones este proceso es costoso, pues a menudo requiere crear una nueva copia del código, lo cual puede tomar mucho tiempo cuando se trata de proyectos grandes.

La forma en la que Git maneja las ramificaciones es increíblemente rápida, haciendo así de las operaciones de ramificación algo casi instantáneo, al igual que el avance o el retroceso entre distintas ramas, lo cual también es tremendamente rápido.

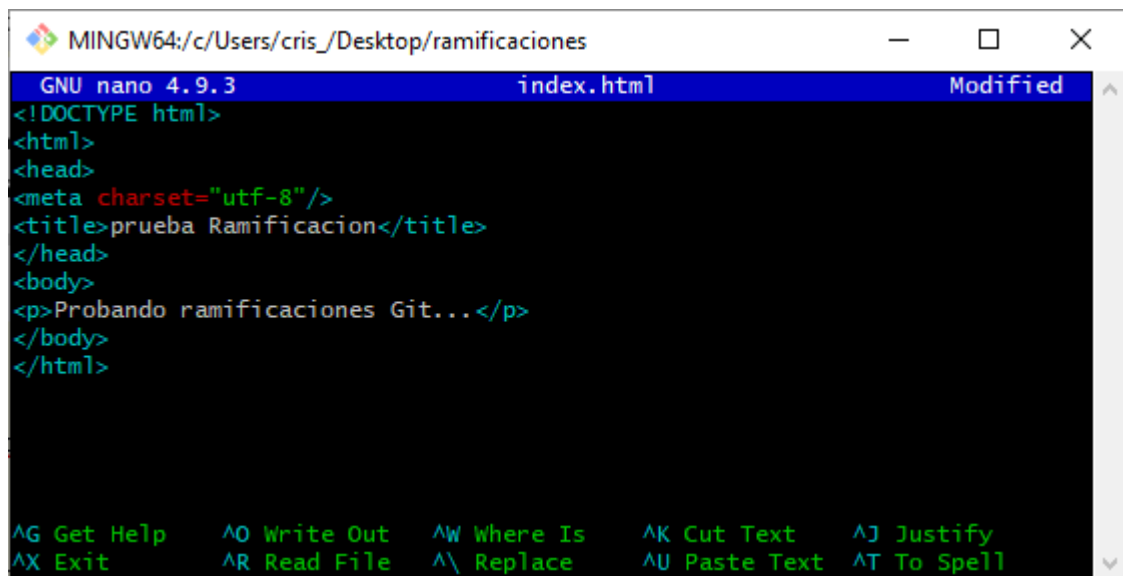
2. Creación de ramas

Cuando creamos una rama nueva simplemente creamos un nuevo apuntador. Para poder llegar a esta situación realizamos dos commits:



Vamos a crear un fichero index.html:

```
nano index.html
```



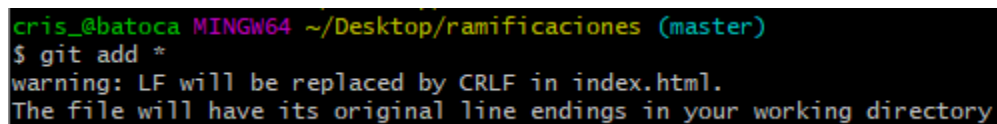
The screenshot shows a terminal window with the title bar "MINGW64:/c/Users/cris/Desktop/ramificaciones". Inside, the GNU nano 4.9.3 editor is open, editing the file "index.html". The file content is as follows:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"/>
<title>prueba Ramificacion</title>
</head>
<body>
<p>Probando ramificaciones Git...</p>
</body>
</html>
```

At the bottom of the editor, there is a status bar with various keyboard shortcuts: ^G Get Help, ^O Write Out, ^W Where Is, ^K Cut Text, ^J Justify, ^X Exit, ^R Read File, ^\ Replace, ^U Paste Text, and ^T To Spell.

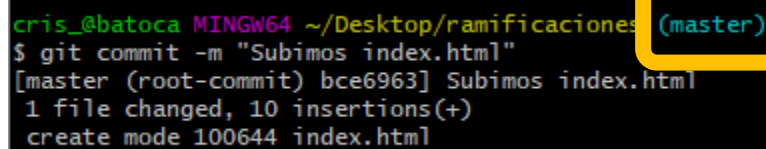
Añadimos el archivo y hacemos commit:

```
$git add *
```



The terminal shows the user "cris_batoca" in the "MINGW64" environment at the directory "~/Desktop/ramificaciones" on the "master" branch. They run the command "\$ git add *". The output is a warning: "warning: LF will be replaced by CRLF in index.html. The file will have its original line endings in your working directory".

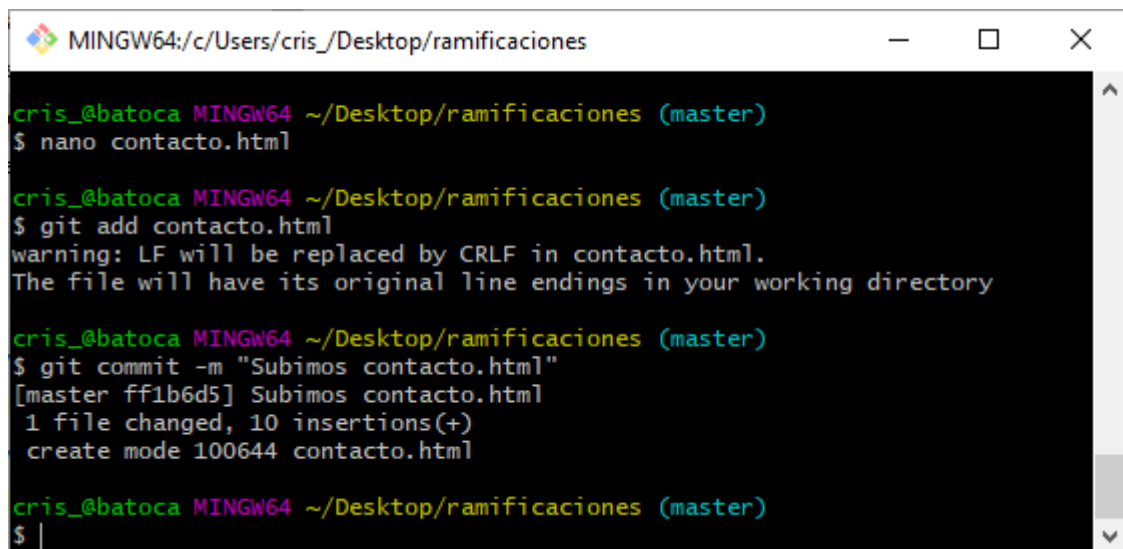
```
$git commit -m "Subimos index.html"
```



The terminal shows the user "cris_batoca" in the "MINGW64" environment at the directory "~/Desktop/ramificaciones" on the "master" branch. They run the command "\$ git commit -m 'Subimos index.html'". The output is: "[master (root-commit) bce6963] Subimos index.html", "1 file changed, 10 insertions(+)", and "create mode 100644 index.html". The "(master)" text in the prompt is highlighted with a yellow box.

Como podemos ver, estamos trabajando sobre la rama **(master)**.

Creamos ahora una nueva página web contacto.html y realizamos commit.



```
MINGW64:/c/Users/cris_/Desktop/ramificaciones

cris_batoca MINGW64 ~/Desktop/ramificaciones (master)
$ nano contacto.html

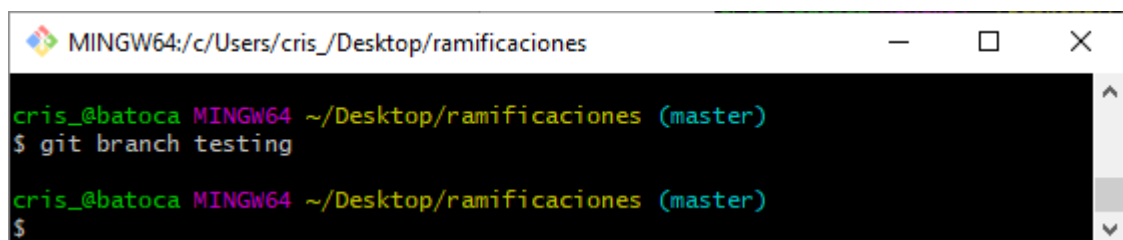
cris_batoca MINGW64 ~/Desktop/ramificaciones (master)
$ git add contacto.html
warning: LF will be replaced by CRLF in contacto.html.
The file will have its original line endings in your working directory

cris_batoca MINGW64 ~/Desktop/ramificaciones (master)
$ git commit -m "Subimos contacto.html"
[master ff1b6d5] Subimos contacto.html
1 file changed, 10 insertions(+)
create mode 100644 contacto.html

cris_batoca MINGW64 ~/Desktop/ramificaciones (master)
$
```

Ahora vamos a crear una nueva rama llamada **testing**:

```
$git branch testing
```




```
MINGW64:/c/Users/cris_/Desktop/ramificaciones

cris_batoca MINGW64 ~/Desktop/ramificaciones (master)
$ git branch testing

cris_batoca MINGW64 ~/Desktop/ramificaciones (master)
$
```

Esto creará un nuevo apuntador apuntando a la misma confirmación donde estés actualmente.

Podemos comprobar con el comando **git log** los códigos y ramas que tenemos:



```
MINGW64:/c/Users/cris_/Desktop/ramificaciones

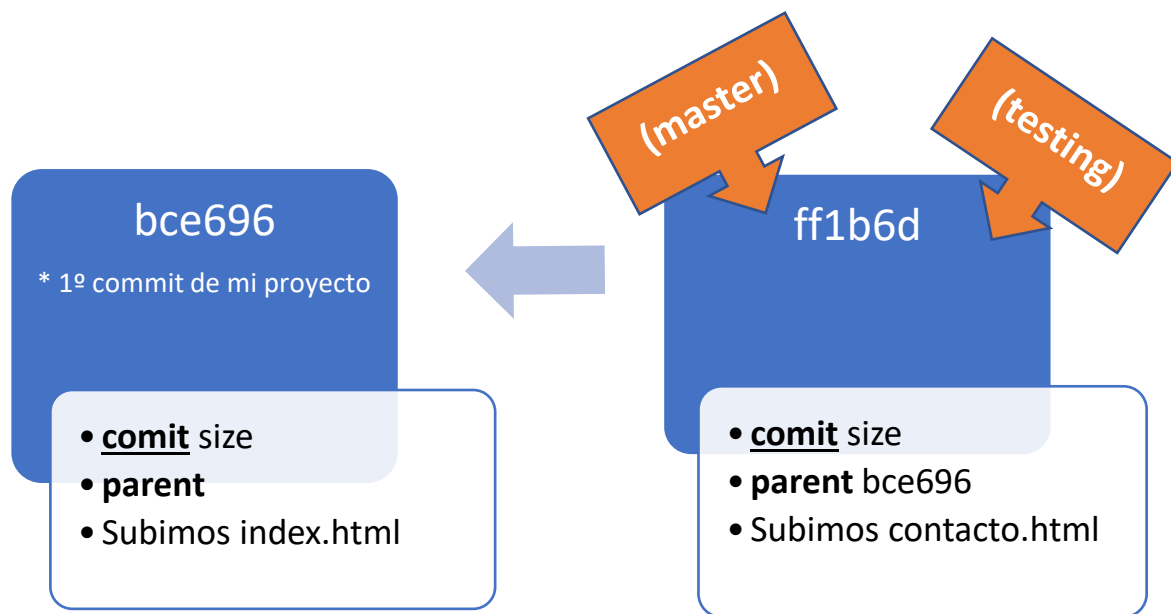
cris_batoca MINGW64 ~/Desktop/ramificaciones (master)
$ git log
commit ff1b6d5105d5aae4d851280596c8b709f4d2a6c2 (HEAD -> master, testing)
Author: Cristina <crispf@ies.net>
Date: Sat Oct 24 11:35:10 2020 +0200

    Subimos contacto.html

commit bce69638fff399493445d3822ed8a3165ceeee61
Author: Cristina <crispf@ies.net>
Date: Sat Oct 24 11:28:20 2020 +0200

    Subimos index.html

cris_batoca MINGW64 ~/Desktop/ramificaciones (master)
$
```



Pero seguimos en la rama master, para cambiarnos a la rama testing debemos ejecutar:

```
git checkout [rama]
```

```
MINGW64:/c/Users/cris/Desktop/ramificaciones
cris@batoca MINGW64 ~/Desktop/ramificaciones (master)
$ git checkout testing
Switched to branch 'testing'
cris@batoca MINGW64 ~/Desktop/ramificaciones (testing)
$ |
```

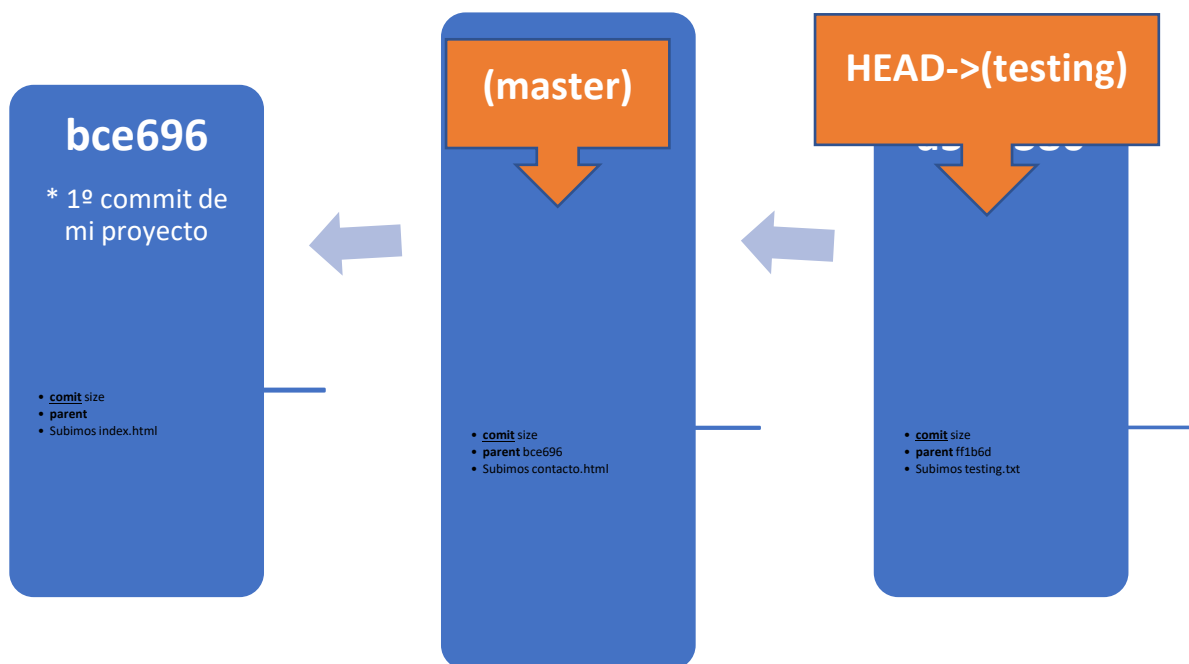
Para comprobar el significado de las ramas, vamos a crear un nuevo archivo en la rama testing y comitearlo:

```
MINGW64:/c/Users/cris_/Desktop/ramificaciones
cris_batoca MINGW64 ~/Desktop/ramificaciones (testing)
$ nano testing.txt

cris_batoca MINGW64 ~/Desktop/ramificaciones (testing)
$ git add testing.txt
warning: LF will be replaced by CRLF in testing.txt.
The file will have its original line endings in your working directory

cris_batoca MINGW64 ~/Desktop/ramificaciones (testing)
$ git commit -m "Añadiendo testing.txt"
[testing a58e380] Añadiendo testing.txt
1 file changed, 1 insertion(+)
create mode 100644 testing.txt

cris_batoca MINGW64 ~/Desktop/ramificaciones (testing)
$
```



Observamos algo interesante: la rama testing avanza, mientras que la rama master permanece en la confirmación donde estaba cuando lanzaste el comando git checkout para saltar.

Volvamos ahora a la rama master y visualizamos el contenido:

```
$git checkout master
$ls
```

```

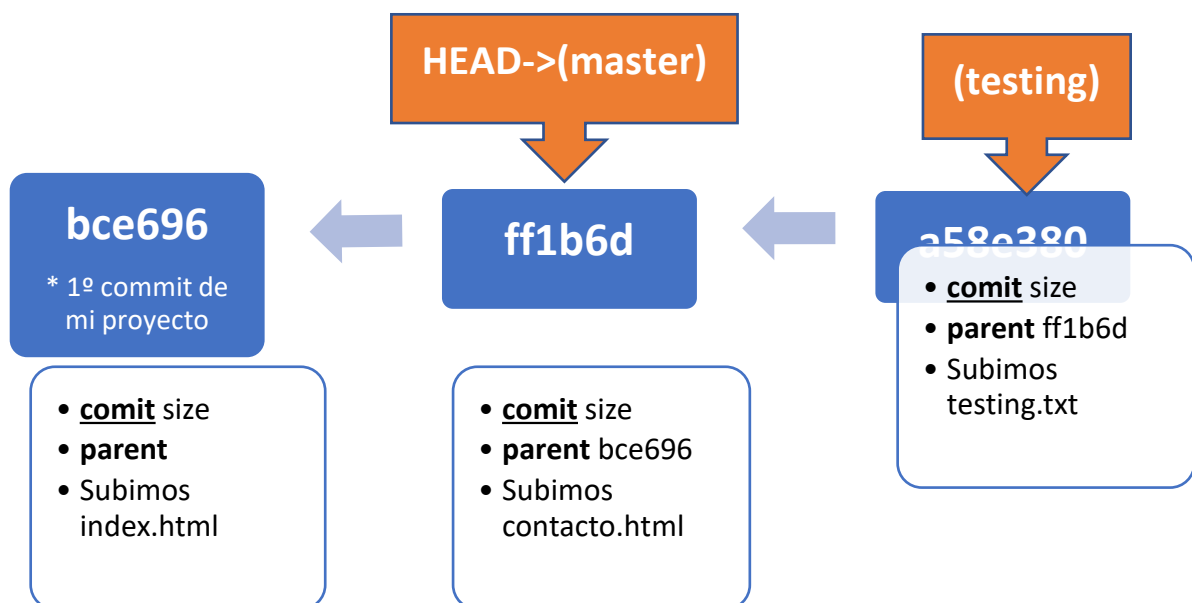
MINGW64:/c/Users/cris_/Desktop/ramificaciones
cris_batoca MINGW64 ~/Desktop/ramificaciones (testing)
$ git checkout master
Switched to branch 'master'

cris_batoca MINGW64 ~/Desktop/ramificaciones (master)
$ ls
contacto.html  index.html

cris_batoca MINGW64 ~/Desktop/ramificaciones (master)
$ |

```

Este comando realiza dos acciones: Mueve el apuntador HEAD de nuevo a la rama master, y revierte los archivos de tu directorio de trabajo; dejándolos tal y como estaban en la última instantánea confirmada en dicha rama master y por lo tanto no vemos el fichero testing.txt.



3. Eliminar ramas

Para eliminar ramas usamos el siguiente comando: **git Branch -D [rama]**

```
$git Branch -D [rama]
```

```

MINGW64:/c/Users/cris_/Desktop/ramificaciones
cris_batoca MINGW64 ~/Desktop/ramificaciones (master)
$ git branch -D testing
Deleted branch testing (was a58e380).

```


4. Fusionar

Imagina que estás trabajando en un proyecto y tienes un par de confirmaciones (commit) ya realizadas. Decides trabajar en el problema #53, según el sistema que tu compañía utiliza para llevar el seguimiento de los problemas. Para ello tienes que crear una nueva rama. Trabajas en el sitio web y confirmas los cambios.

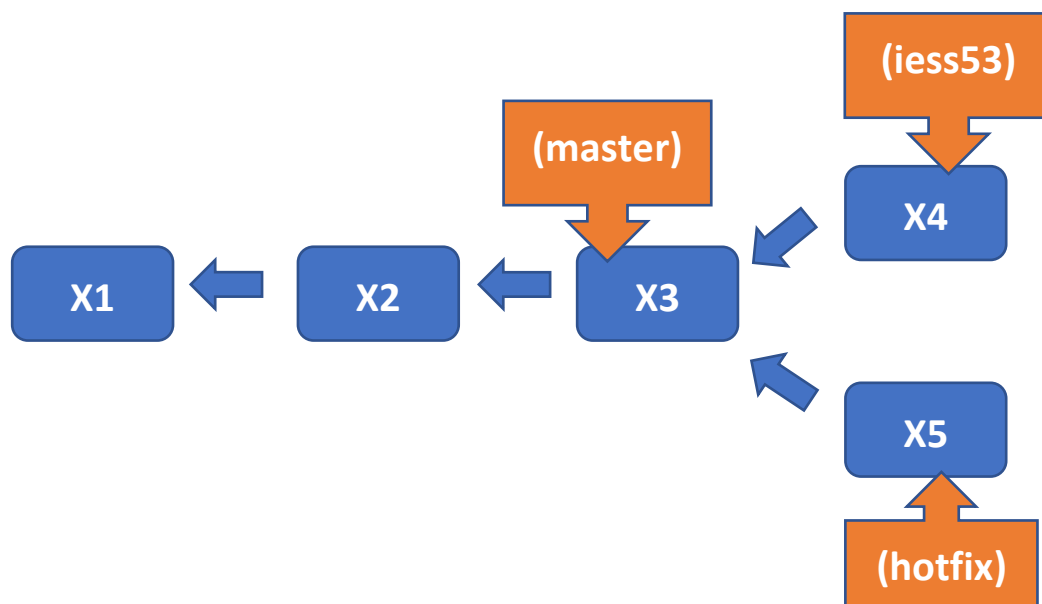
Entonces, recibes una llamada avisándote de otro problema urgente en el sitio web y debes resolverlo inmediatamente (hotfix).

Al usar Git, no necesitas mezclar el nuevo problema con los cambios que ya habías realizado sobre el problema #53; ni tampoco perder tiempo revirtiendo esos cambios para poder trabajar sobre el contenido que está en producción. Basta con saltar de nuevo a la rama master y continuar trabajando a partir de allí.

Hay que tener en cuenta que **para cambiar de rama tenemos que tener todo confirmado.**

Después de crear la rama, creas un fichero nuevo y confirmas los cambios:

Nos encontramos en este punto, con tres ramas:



Después de realizar las pruebas oportunas, comprobar que la solución es correcta, debemos incorporar los cambios a la rama master para ponerlos en producción. Esto se hace con el comando

git merge [rama] desde la rama master. Por lo tanto debemos ejecutar:

```
$git checkout master  
$git merge hotfix
```

Tras haber resuelto el problema urgente que había interrumpido tu trabajo, puedes volver a donde estabas. Pero antes, es importante borrar la rama hotfix, ya que no la vamos a necesitar más, puesto que apunta exactamente al mismo sitio que la rama master.

```
$git branch -d hotfix
```

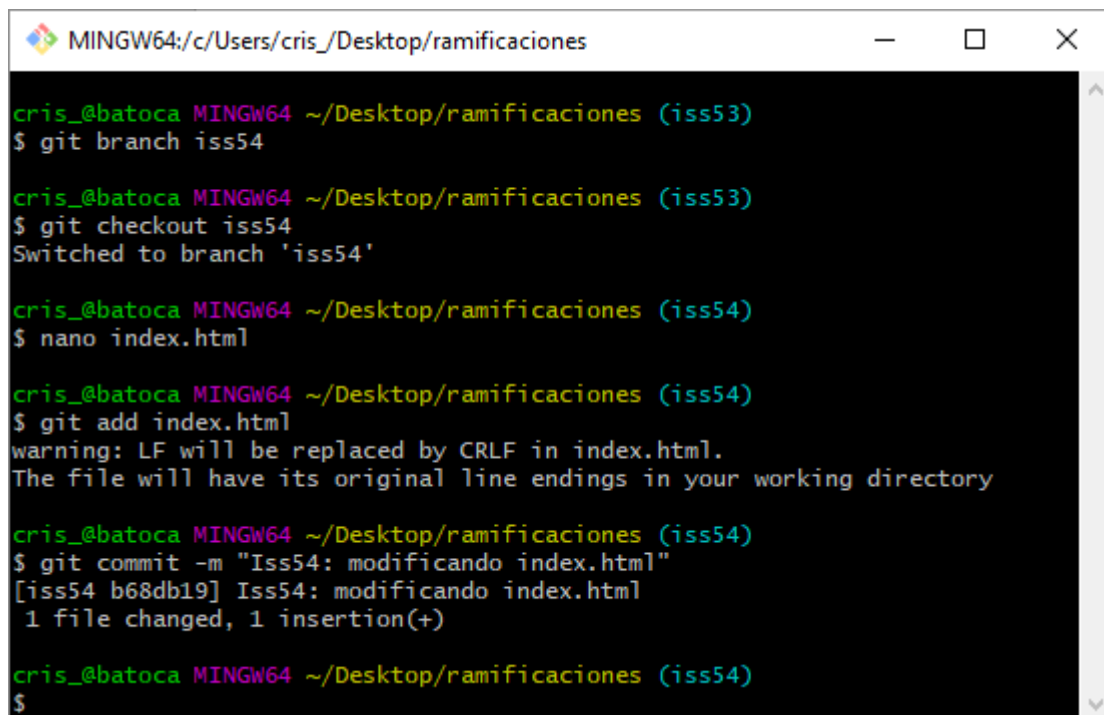
Podemos ver las ramas que nos quedan ejecutando directamente el comando **git Branch**.

5. Conflictos en las fusiones

En algunas ocasiones, los procesos de fusión no suelen ser fluidos. Si hay modificaciones dispares en una misma porción de un mismo archivo en dos ramas distintas que pretendes fusionar, Git no será capaz de fusionarlas directamente.

Para comprobar esto vamos a crear dos nuevas ramas que las asociamos a las tareas iss54 y iss55.

Empezamos creando la rama iss54, modificamos el fichero index.html, y realizamos confirmación de los cambios:



```
MINGW64:/c/Users/cris_/Desktop/ramificaciones

cris@batoca MINGW64 ~/Desktop/ramificaciones (iss53)
$ git branch iss54

cris@batoca MINGW64 ~/Desktop/ramificaciones (iss53)
$ git checkout iss54
Switched to branch 'iss54'

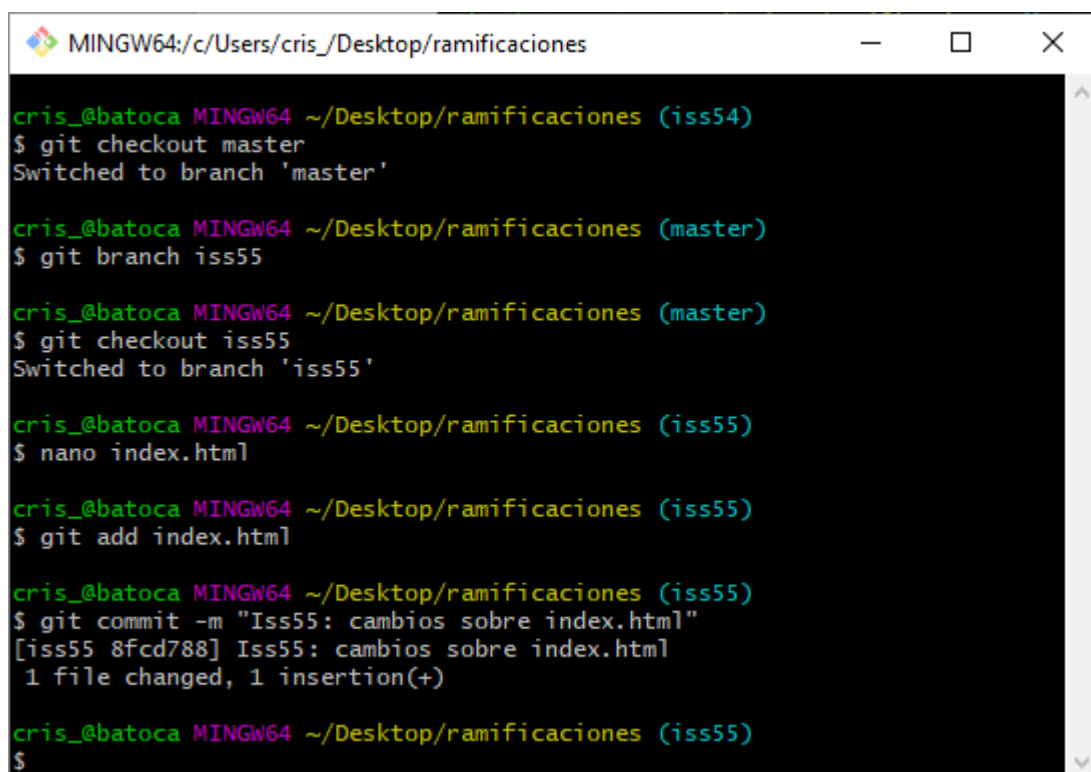
cris@batoca MINGW64 ~/Desktop/ramificaciones (iss54)
$ nano index.html

cris@batoca MINGW64 ~/Desktop/ramificaciones (iss54)
$ git add index.html
warning: LF will be replaced by CRLF in index.html.
The file will have its original line endings in your working directory

cris@batoca MINGW64 ~/Desktop/ramificaciones (iss54)
$ git commit -m "Iss54: modificando index.html"
[iss54 b68db19] Iss54: modificando index.html
1 file changed, 1 insertion(+)

cris@batoca MINGW64 ~/Desktop/ramificaciones (iss54)
$
```

Creemos y nos cambiamos de rama a las iss55 desde la rama master y modificamos el fichero index.html también. Por supuesto, a continuación confirmamos los cambios mediante commit.



```
MINGW64:/c/Users/cris_/Desktop/ramificaciones

cris@batoca MINGW64 ~/Desktop/ramificaciones (iss54)
$ git checkout master
Switched to branch 'master'

cris@batoca MINGW64 ~/Desktop/ramificaciones (master)
$ git branch iss55

cris@batoca MINGW64 ~/Desktop/ramificaciones (master)
$ git checkout iss55
Switched to branch 'iss55'

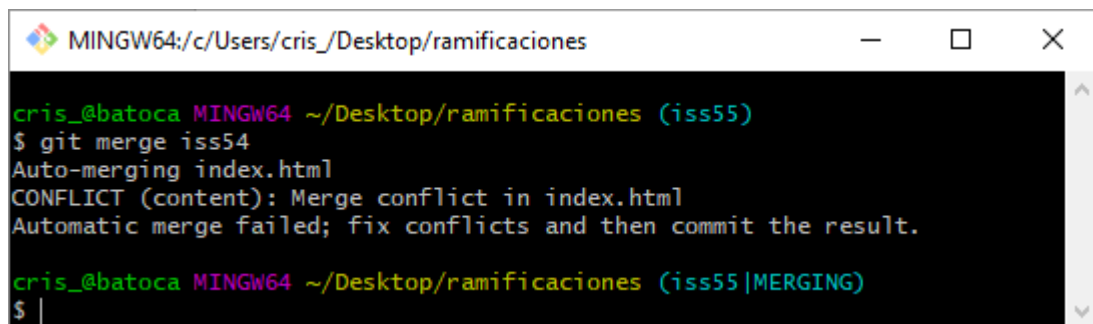
cris@batoca MINGW64 ~/Desktop/ramificaciones (iss55)
$ nano index.html

cris@batoca MINGW64 ~/Desktop/ramificaciones (iss55)
$ git add index.html

cris@batoca MINGW64 ~/Desktop/ramificaciones (iss55)
$ git commit -m "Iss55: cambios sobre index.html"
[iss55 8fcd788] Iss55: cambios sobre index.html
1 file changed, 1 insertion(+)

cris@batoca MINGW64 ~/Desktop/ramificaciones (iss55)
$
```

Una vez modificados los ficheros index.html en las dos ramas intentamos combinarlas (comando **merge**). Vamos a realizar el merge de la rama iss54 desde la iss55:



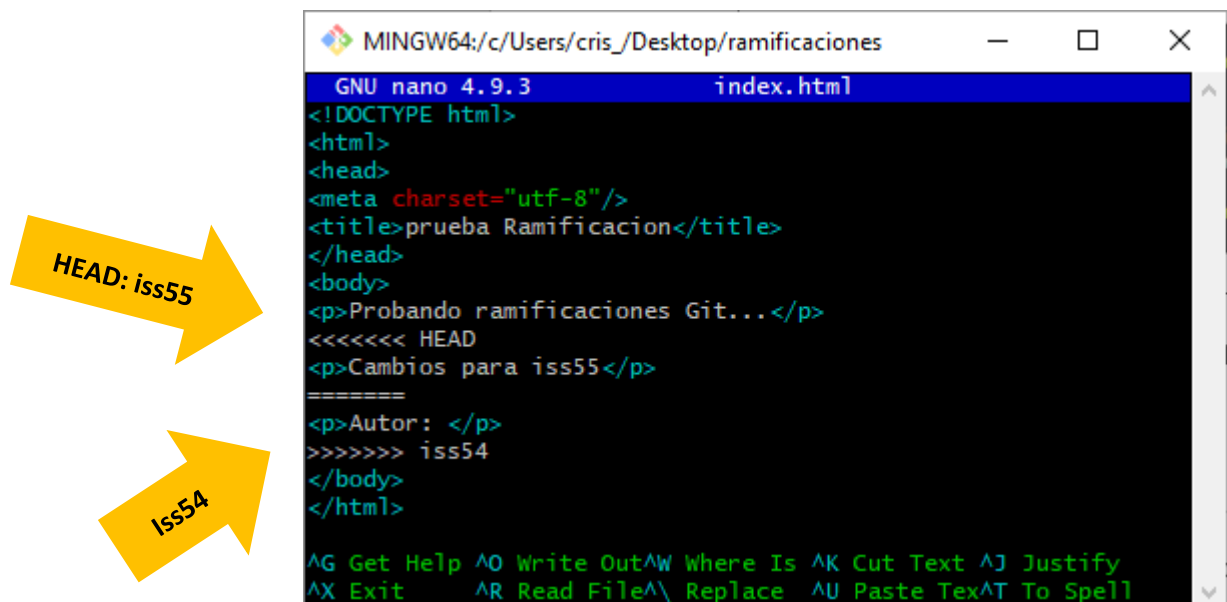
```

MINGW64:/c/Users/cris_/Desktop/ramificaciones
cris@batoca MINGW64 ~/Desktop/ramificaciones (iss55)
$ git merge iss54
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.

cris@batoca MINGW64 ~/Desktop/ramificaciones (iss55|MERGING)
$
  
```

Comprobamos que nos da un error. Como hay modificaciones dispares en una misma porción de un mismo archivo en las dos ramas distintas que pretendes fusionar, Git no será capaz de fusionarlas directamente.

Git no crea automáticamente una nueva fusión confirmada (merge commit), sino que hace una pausa en el proceso, esperando a que tú resuelvas el conflicto. Para eso tenemos que entrar directamente en el archivo index.html que nos indica. El archivo conflictivo contendrá algo como:



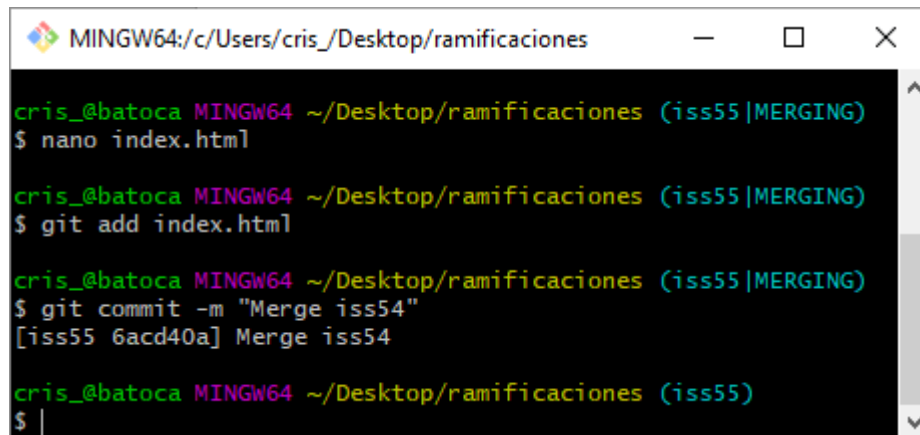
```

GNU nano 4.9.3 index.html
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"/>
<title>prueba Ramificacion</title>
</head>
<body>
<p>Probando ramificaciones Git...</p>
<<<<<<< HEAD
<p>Cambios para iss55</p>
=====
<p>Autor: </p>
>>>>>>> iss54
</body>
</html>

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell
  
```

Para resolver el conflicto, has de elegir manualmente el contenido de uno o de otro lado.

Esta corrección contiene un poco de ambas partes y se han eliminado completamente las líneas <<<<<<<, ===== y >>>>>>>. Tras resolver todos los bloques conflictivos, has de lanzar comandos **git add** para marcar cada archivo modificado.

A terminal window titled 'MINGW64:/c/Users/cris_/Desktop/ramificaciones' showing a series of Git commands and their output. The user is in the 'iss55' branch, which is in a MERGING state. The commands executed are: 'nano index.html', 'git add index.html', and 'git commit -m "Merge iss54"'. The output shows the commit hash '6acd40a' and the message 'Merge iss54'.

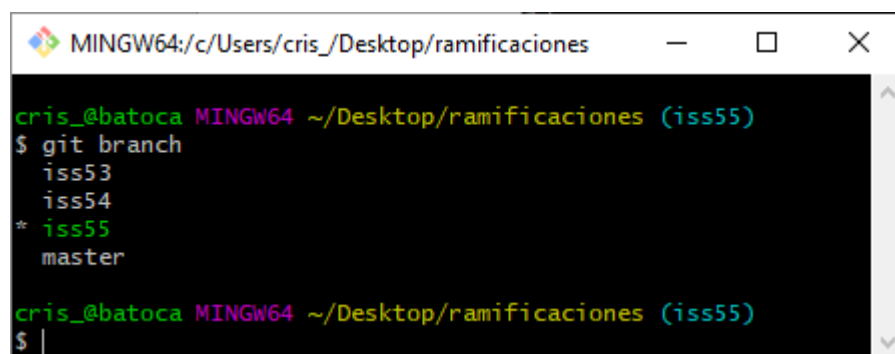
```
MINGW64:/c/Users/cris_/Desktop/ramificaciones
cris_batoca MINGW64 ~/Desktop/ramificaciones (iss55|MERGING)
$ nano index.html
cris_batoca MINGW64 ~/Desktop/ramificaciones (iss55|MERGING)
$ git add index.html
cris_batoca MINGW64 ~/Desktop/ramificaciones (iss55|MERGING)
$ git commit -m "Merge iss54"
[iss55 6acd40a] Merge iss54
cris_batoca MINGW64 ~/Desktop/ramificaciones (iss55)
$
```

Después de esto nos desaparecerá el mensaje de MERGING.

Si en lugar de resolver directamente prefieres utilizar una herramienta gráfica, puedes usar el comando **git mergetool**, el cual arrancará la correspondiente herramienta de visualización y te permitirá ir resolviendo conflictos con ella.

6. Gestión de ramas

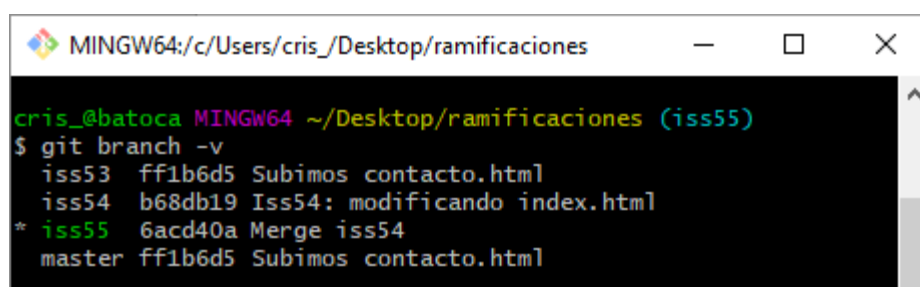
El comando **git branch** tiene más funciones que las de crear y borrar ramas. Si lo lanzas sin parámetros, obtienes una lista de las ramas presentes en tu proyecto:

A terminal window titled 'MINGW64:/c/Users/cris_/Desktop/ramificaciones' showing the output of the 'git branch' command. It lists the branches: 'iss53', 'iss54', 'iss55' (marked with an asterisk as the current branch), and 'master'.

```
MINGW64:/c/Users/cris_/Desktop/ramificaciones
cris_batoca MINGW64 ~/Desktop/ramificaciones (iss55)
$ git branch
  iss53
  iss54
* iss55
  master
cris_batoca MINGW64 ~/Desktop/ramificaciones (iss55)
$
```

Para ver la última confirmación de cambios en cada rama, puedes usar el comando

```
$git branch -v
```

A terminal window titled 'MINGW64:/c/Users/cris_/Desktop/ramificaciones' showing the output of the 'git branch -v' command. It displays the branch names along with their commit hashes and the last commit message: 'iss53 ff1b6d5 Subimos contacto.html', 'iss54 b68db19 Iss54: modificando index.html', 'iss55 6acd40a Merge iss54' (marked with an asterisk), and 'master ff1b6d5 Subimos contacto.html'.

```
MINGW64:/c/Users/cris_/Desktop/ramificaciones
cris_batoca MINGW64 ~/Desktop/ramificaciones (iss55)
$ git branch -v
  iss53 ff1b6d5 Subimos contacto.html
  iss54 b68db19 Iss54: modificando index.html
* iss55 6acd40a Merge iss54
  master ff1b6d5 Subimos contacto.html
```

Otra opción útil para averiguar el estado de las ramas, es filtrar y mostrar solo aquellas que han sido fusionadas (o que no lo han sido) con la rama actualmente activa.

```
$git branch --merged
```

Para mostrar todas las ramas que contienen trabajos sin fusionar, puedes utilizar el comando:

```
$git branch --no-merged
```

Esto nos muestra la otra rama del proyecto. Debido a que contiene trabajos sin fusionar, al intentar borrarla con `git branch -d`, el comando nos dará un error.

7. Ramas remotas

Las ramas remotas son referencias al estado de las ramas en tus repositorios remotos. Las ramas remotas funcionan como marcadores, para recordarte en qué estado se encontraban tus repositorios remotos la última vez que conectaste con ellos.

Supongamos que tienes un servidor Git en tu red, en `git.ourcompany.com`. Si haces un clon desde ahí, Git automáticamente lo denominará `origin`, traerá (pull) sus datos, creará un apuntador hacia donde esté en ese momento su rama `master` y denominará la copia local `origin/master`. Git te proporcionará también tu propia rama `master`, apuntando al mismo lugar que la rama `master` de `origin`; de manera que tengas donde trabajar.

ANEXO I: Materiales

I. Textos de apoyo o de referencia

- Manuais San Clemente:
[https://manuais.iessanclemente.net/index.php/Control de versiones con Git y GitHub](https://manuais.iessanclemente.net/index.php/Control%20de%20versiones%20con%20Git%20y%20GitHub)
- Libro Git Pro: <https://git-scm.com/book/es/v2>

II. Recursos web

- Página oficial de Git: <https://git-scm.com>
- Página oficial de Bit bucket: <https://bitbucket.org/>

III. Recursos didácticos

- Apuntes en el aula virtual.
- Ordenador personal, con navegador web y conexión a internet.
- Software para elaboración de documentos de texto.