



**git**

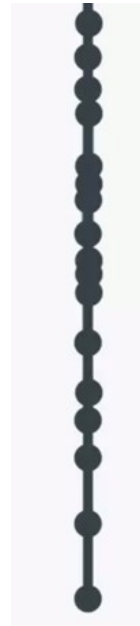
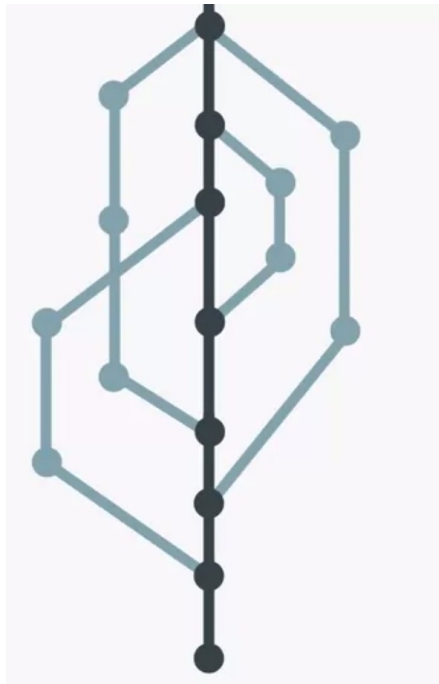
**Introducción**

**Basic merging**

**Administrando branches**

## Introducción

- Ya hemos visto como crear ramas, pero estas en algún momento tiene que ser unidas a la rama principal para tener un proyecto terminado
- Esta acción es conocida como merging

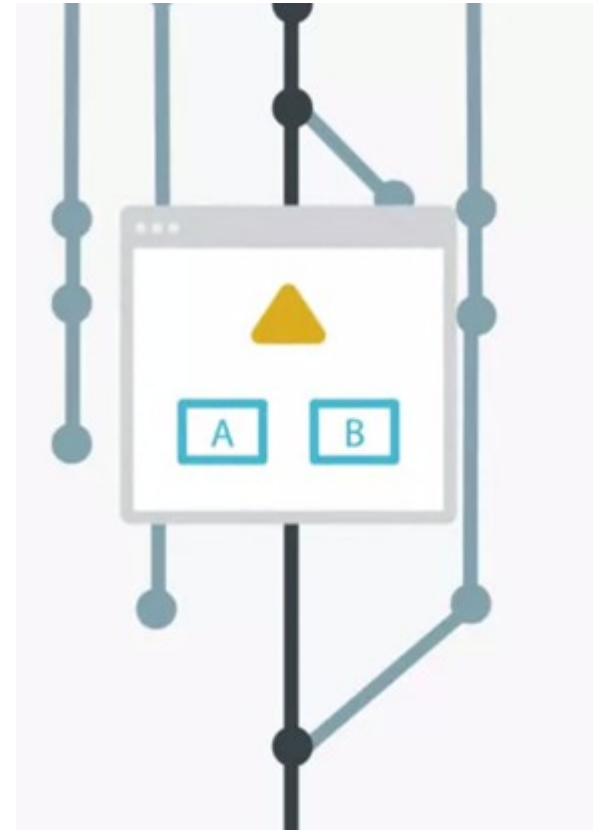


## Introducción

- Cuando trabajamos con ramas, el proyecto tiene dos estados diferentes (branching > merge)
- De esta forma combina todos los commits de la rama en un solo.
- Las ramas no desaparecen cuando realizamos merging, por lo que continuarán en el historial de nuestro repositorio
- Cuando realizamos merging de varias ramas tenemos que realizarlo en el orden adecuado y poniendo atención a los ficheros que han cambiado en ambas ramas
- Cuando el mismo fichero ha sido cambiado en dos ramas diferentes se produce un merge conflict

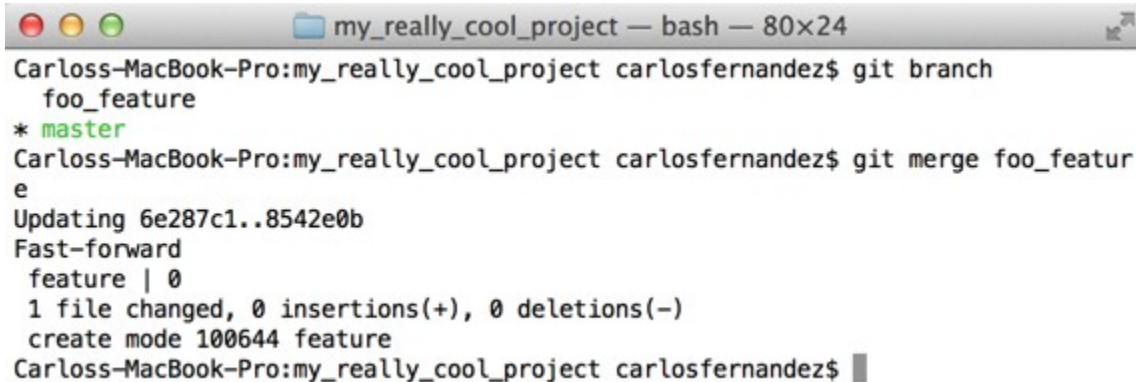
## Introducción

- Cuando sucede un merge conflict hay que tomar una decisión
- Algunos SCV te preguntan que acción tomar para todos los conflictos
- Git en cambio resolverá sólo la mayoría de los conflictos automáticamente
- En algunas ocasiones te preguntará que hacer para solucionarlas
- Git hace el proceso de merging menos doloroso generalmente



## Basic Merging

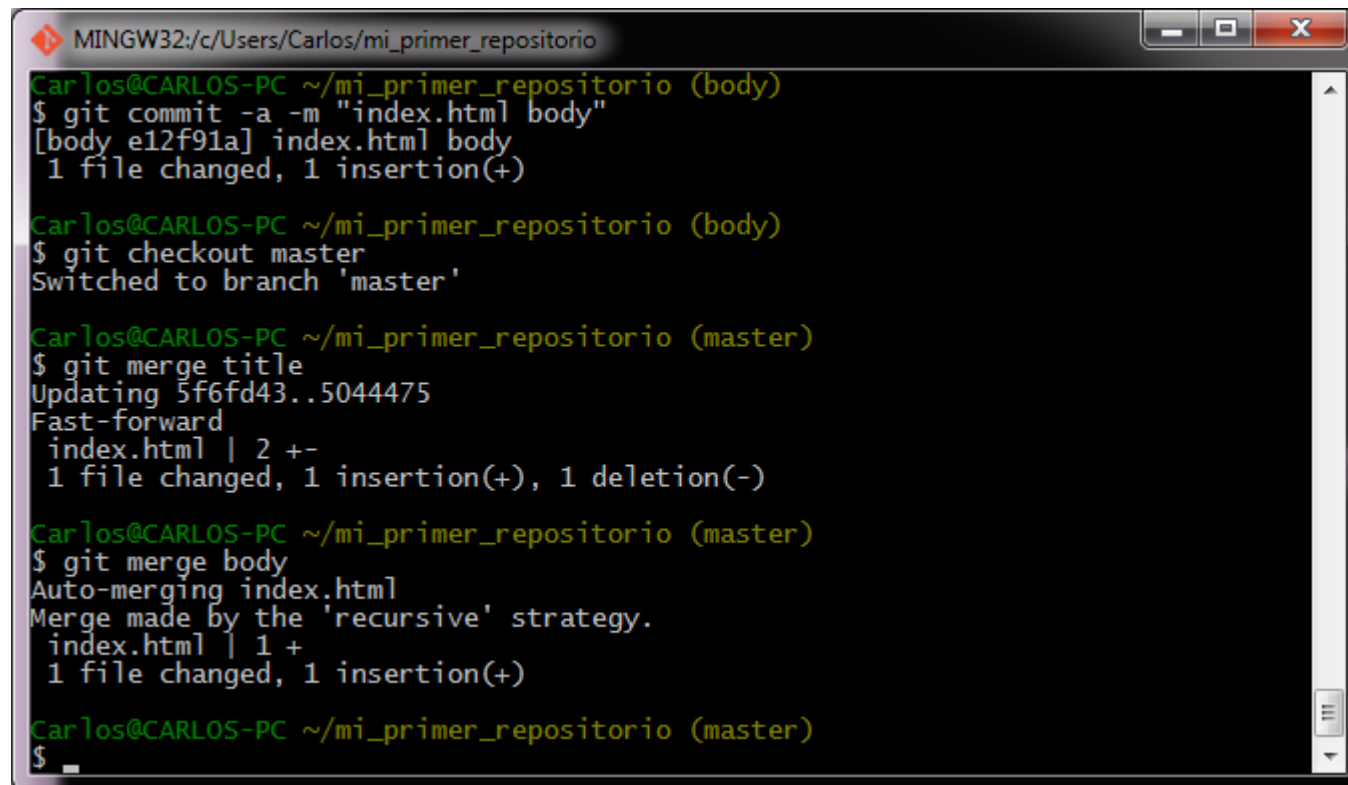
- Si unes dos ramas que que no han modificado los mismos ficheros, merging es un proceso rápido y sencillo
- Para unir dos ramas utilizamos el comando `git merge nombre_rama`

A screenshot of a macOS terminal window titled "my\_really\_cool\_project — bash — 80x24". The terminal shows the following sequence of commands and output:

```
Carloss-MacBook-Pro:my_really_cool_project carlosfernandez$ git branch
foo_feature
* master
Carloss-MacBook-Pro:my_really_cool_project carlosfernandez$ git merge foo_featur
e
Updating 6e287c1..8542e0b
Fast-forward
 feature | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 feature
Carloss-MacBook-Pro:my_really_cool_project carlosfernandez$
```

## Basic Merging

- Veremos un ejemplo de git intentando resolver un conflicto de forma automática



```
MINGW32/c/Users/Carlos/mi_primer_repositorio
Carlos@CARLOS-PC ~/mi_primer_repositorio (body)
$ git commit -a -m "index.html body"
[body e12f91a] index.html body
1 file changed, 1 insertion(+)

Carlos@CARLOS-PC ~/mi_primer_repositorio (body)
$ git checkout master
Switched to branch 'master'

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git merge title
Updating 5f6fd43..5044475
Fast-forward
 index.html | 2 +-
1 file changed, 1 insertion(+), 1 deletion(-)

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git merge body
Auto-merging index.html
Merge made by the 'recursive' strategy.
 index.html | 1 +
1 file changed, 1 insertion(+)

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$
```

## Basic Merging

### Ejercicio

1. Crea un fichero llamado home.html con el siguiente contenido

```
<!doctype html>  
  
<html>  
  
    <head>  
  
        <title></title>  
  
    </head>  
  
<body>  
  
</body>  
  
</html>
```

2. Añadelo a la rama maestra con add y commit
3. Crea dos nuevas, una rama llamada body1 y otra body2



## Basic Merging

### Ejercicio

4. Cambiate a la rama body1 y añade en una nueva línea en el fichero home.html entre las etiquetas body el siguiente párrafo “hello world”

```
<body>
```

```
    <p>hello world</p>
```

```
</body>
```

5. Actualiza los cambios en la rama body1 (add y commit)
6. Vuelve a la rama maestra y realiza un merge con la rama body1 ¿Qué sucede?
7. Cambiate a la rama body2 y edita el fichero home.html y añade en la misma línea entre las etiquetas body el siguiente texto

```
<body>
```

```
    <p>bye bye world</p>
```

```
</body>
```

## Conflictos de Merging

- ¿Que ha sucedido ahora? Aunque la mayoría de las veces no tendremos problemas al hacer merging, tendremos que solucionar conflictos en los que git no pueda por sí solo
- Cuando esto suceda git nos avisará y mostrará en el fichero que ha sucedido el conflicto que líneas están dando los problemas

```
<<<<<< HEAD
```

```
Esto es el fichero1, hello world
```

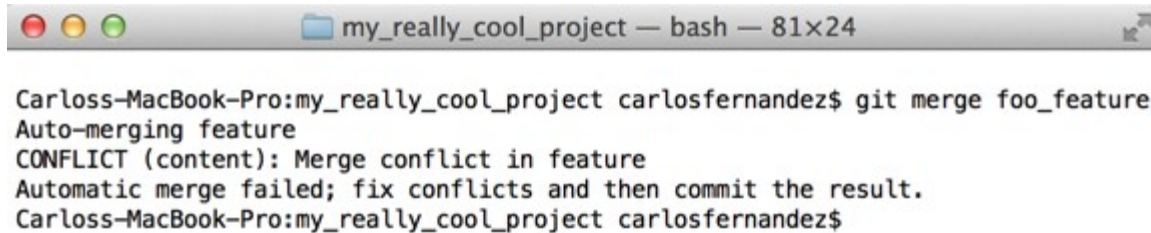
```
=====
```

```
Esto es el fichero1, main page
```

```
>>>>>> rama_cualquiera
```

## Conflictos de Merging

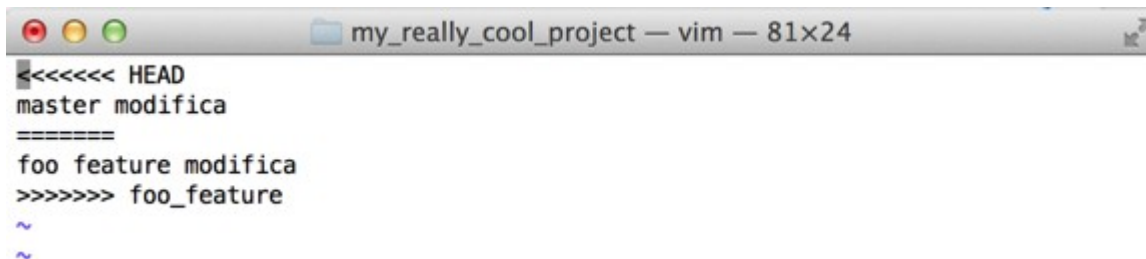
- En este ejemplo hemos modificado el mismo fichero en dos ramas diferentes



```
my_really_cool_project — bash — 81x24

Carloss-MacBook-Pro:my_really_cool_project carlosfernandez$ git merge foo_feature
Auto-merging feature
CONFLICT (content): Merge conflict in feature
Automatic merge failed; fix conflicts and then commit the result.
Carloss-MacBook-Pro:my_really_cool_project carlosfernandez$
```

- Git no puede solucionarlo, pero en este caso ha fallado, si editamos el fichero que generó el conflicto veremos lo siguiente



```
my_really_cool_project — vim — 81x24
<<<<<<< HEAD
master modifica
=====
foo feature modifica
>>>>>>> foo_feature
~
~
```

## Conflictos de Merging

Solucionar conflicto

- Una opción es borrar los comentarios y unir los dos cambios

Esto es el fichero1, hello world

Esto es el fichero1, main page

Una mejor solución será decidir cual de las dos habría que dejar

Esto es el fichero1, hello world

## Conflictos de Merging

Solucionar conflicto

- Una vez que hemos solucionado manualmente el conflicto pasaremos el fichero al *staging area* con el comando `git add`
- Después realizaremos *commiting* con el comando `git commit`
- Dejaremos un mensaje indicativo de que hemos solucionado el conflicto en el fichero

## Conflictos de Merging

### Ejercicio

Cada vez que se soluciona un conflicto, se hace un tipo especial de commit merge, algunas personas no les gusta que esto se guarde en el historial, Esto podríamos solucionarlo con la técnica “rebasing”.

Hay otras operaciones como reset y revert que igualmente nos ayudan a resolver problemas.

Es posible, cuando se realiza un git merge, sobrecribir todos los cambios conflictivos con el de la nueva rama y esto podría ser peligroso para la integridad del proyecto.

Algunas operaciones avanzadas con Git las puedes encontrar aquí:

[Git Checkout, Reset, Revert](#)

[Git Merge, Rebase](#)

Una guía resumen:

[Git simple guide](#)

Y una repositorio con ejercicios:

[git\\_training](#)

## Webgrafia

<http://git-scm.com/>