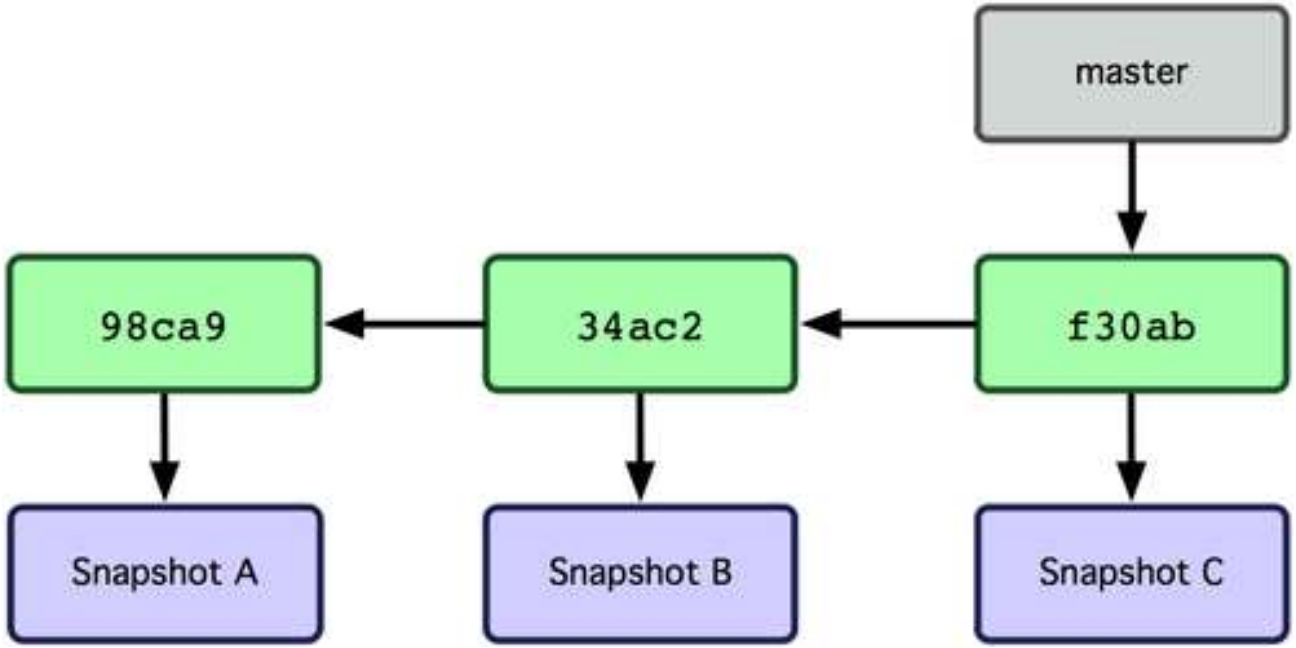
Ramas

¿Qué es una Rama?

- Una rama es un puntero a un commit.
  - Este puntero se reasigna al crear nuevos commits.
- En GIT suele existir una rama principal llamada **master**.



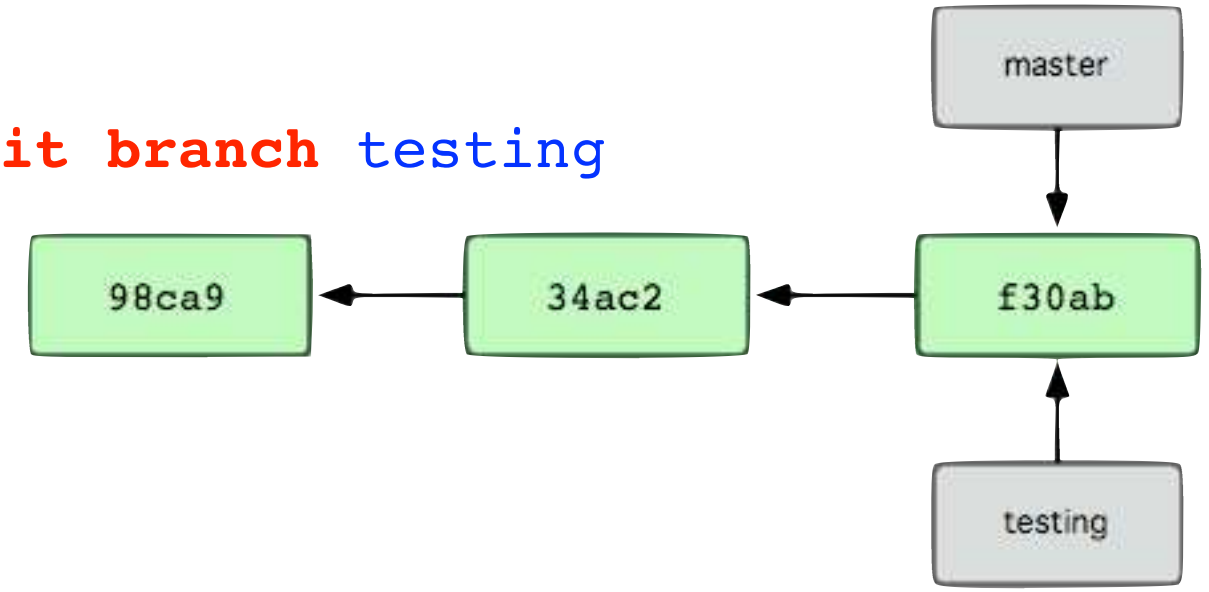
*de Scott Chanson: http://git-scm.org/book/

- Para crear nuevas ramas:

```
$ git branch <nombre>
```

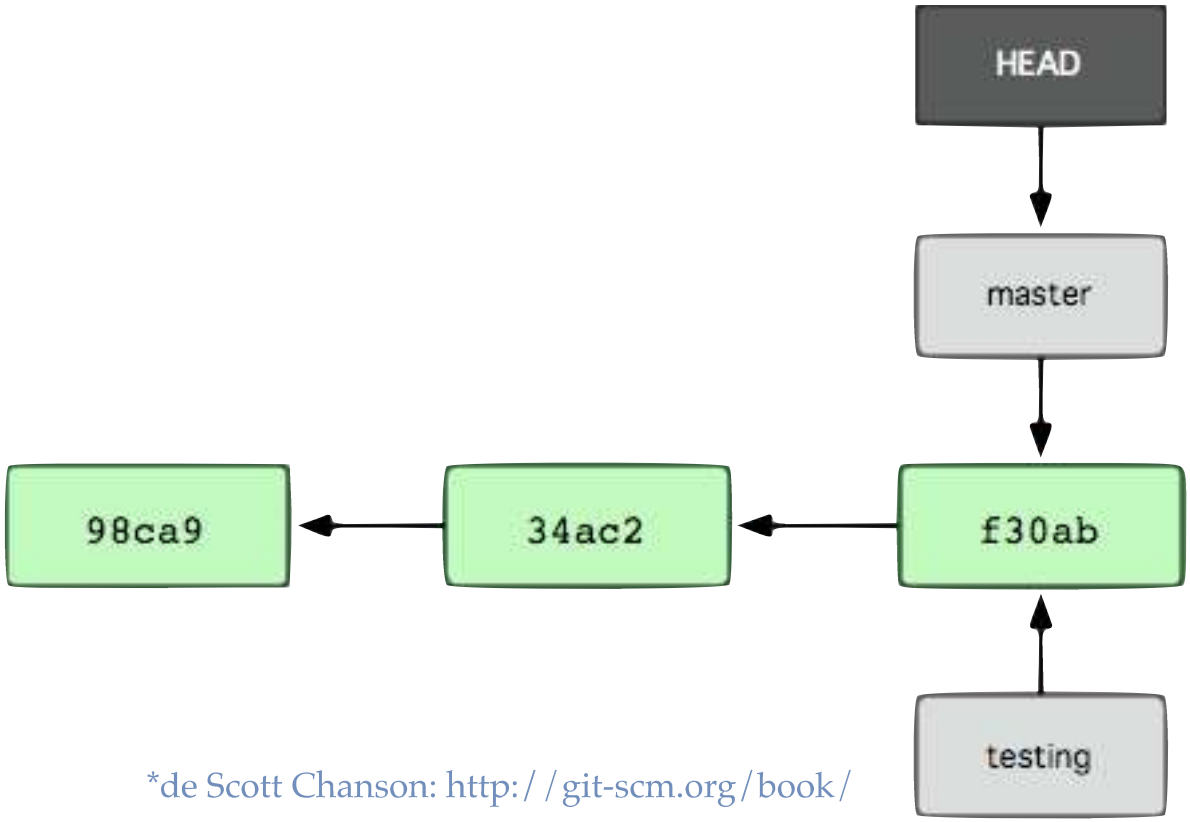
  - Se crea un nuevo puntero.

```
$ git branch testing
```



*de Scott Chanson: http://git-scm.org/book/

- Existe una referencia llamada **HEAD** que apunta a la rama actual
  - que apunta al commit sobre el que trabajamos.
    - los ficheros del directorio de trabajo y del staging area están basados en este commit.
  - Los nuevos commits se añaden al commit apuntado por **HEAD**.



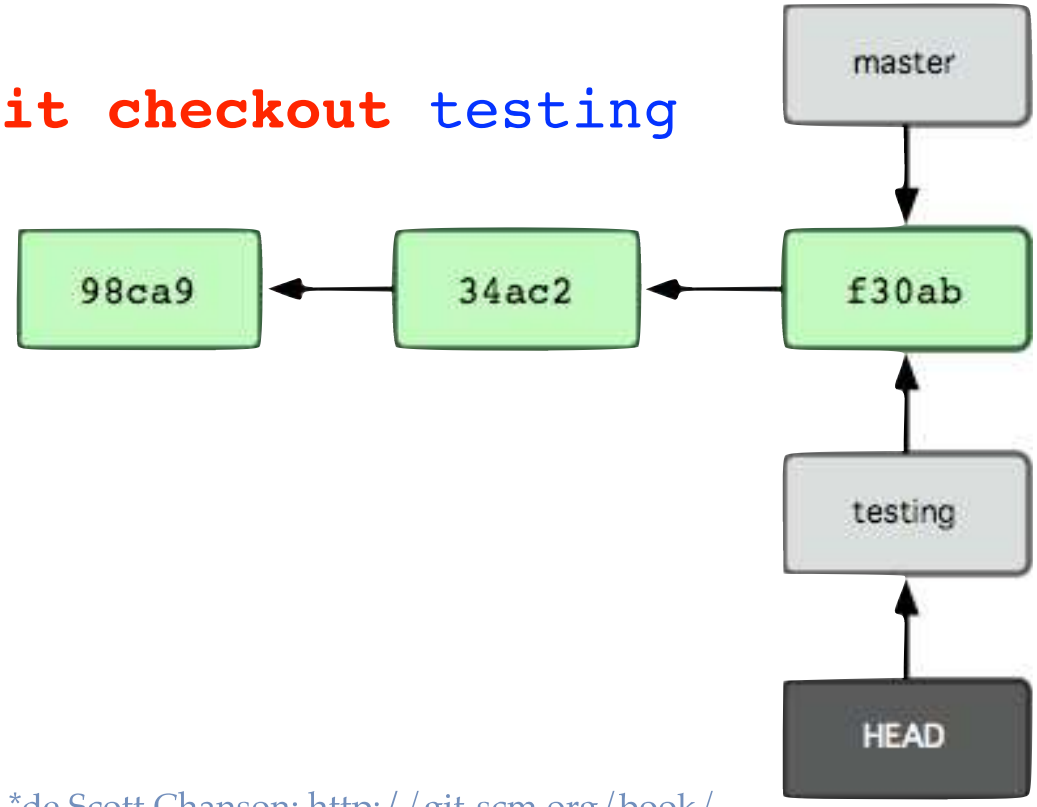
*de Scott Chanson: http://git-scm.org/book/

- Para cambiar de rama actual:

```
$ git checkout <nombre_rama>
```

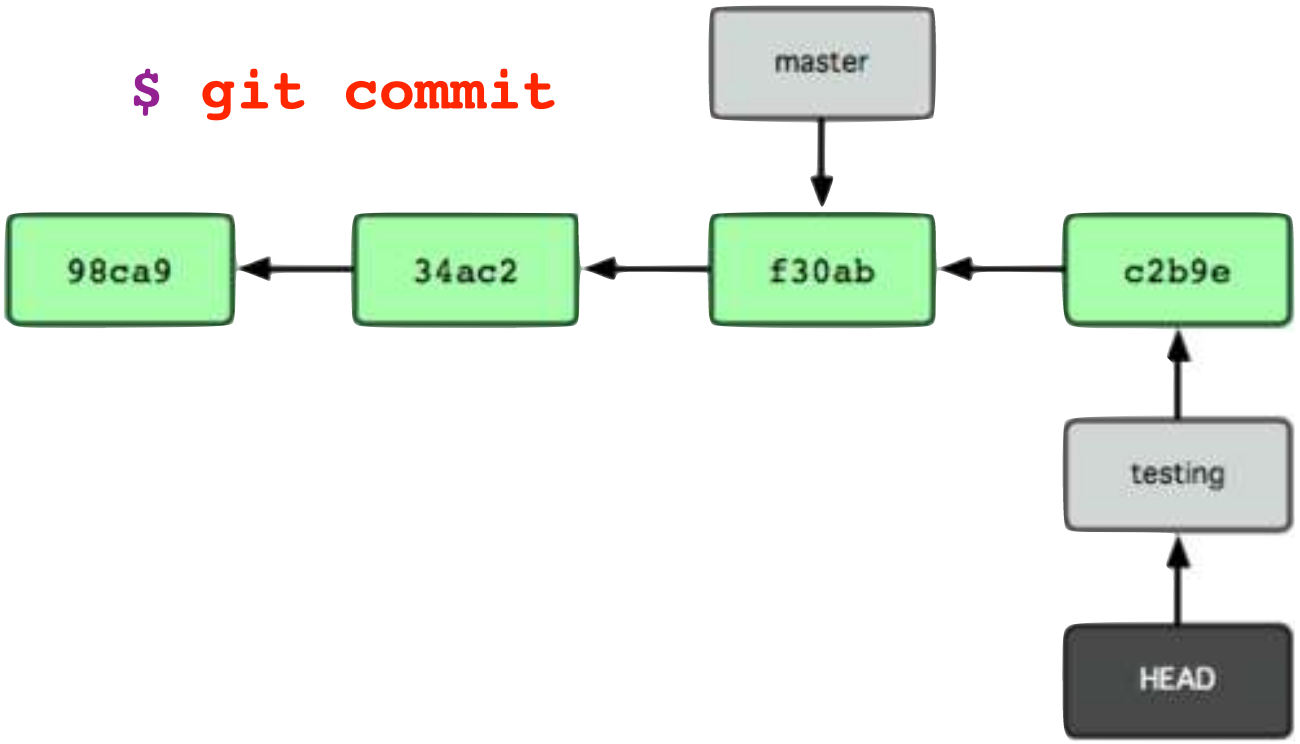
  - HEAD apuntará a la nueva rama.
  - Se actualiza el directorio de trabajo y el staging area.

```
$ git checkout testing
```



*de Scott Chanson: http://git-scm.org/book/

- Al hacer commit avanza la rama apuntada por **HEAD**.

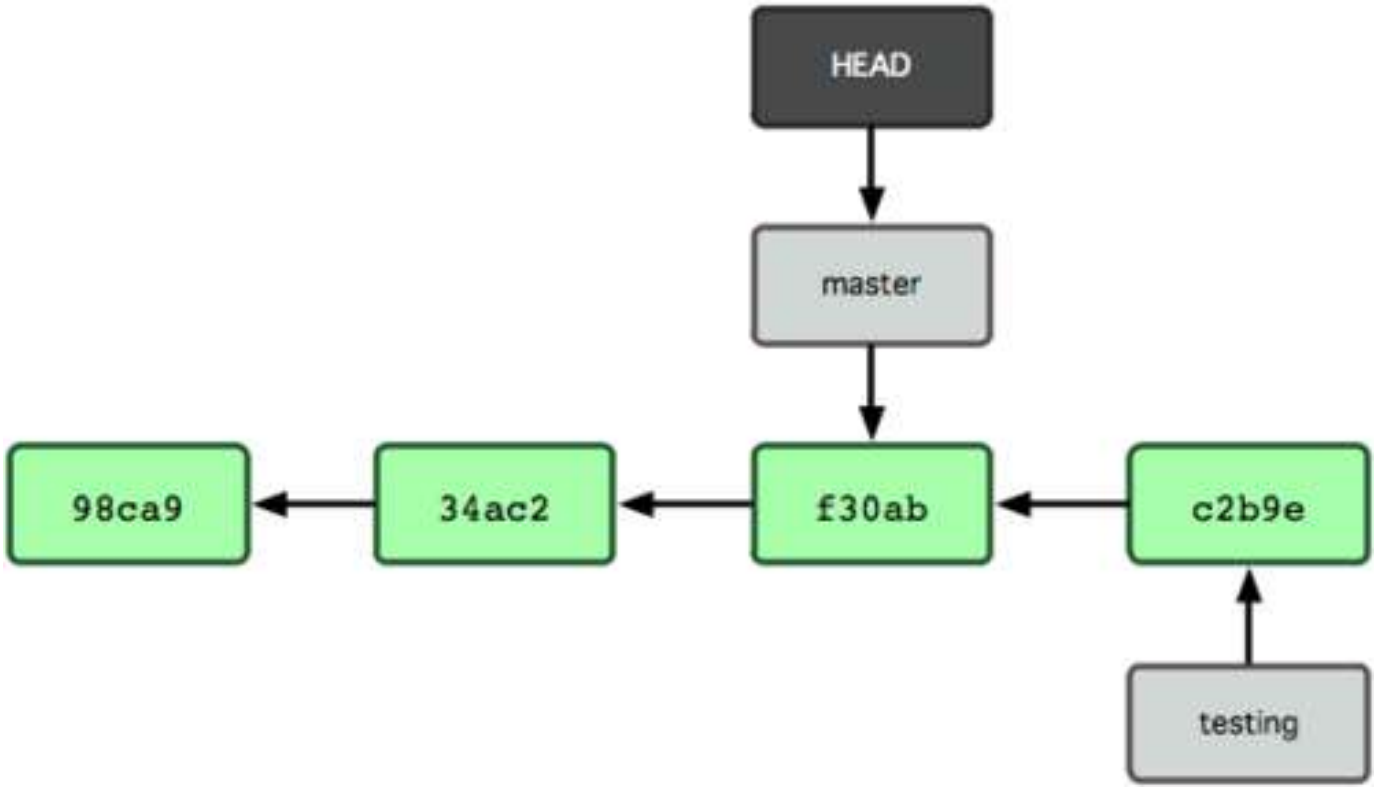


*de Scott Chanson: http://git-scm.org/book/

- Para volver a la rama master:

```
$ git checkout master
```

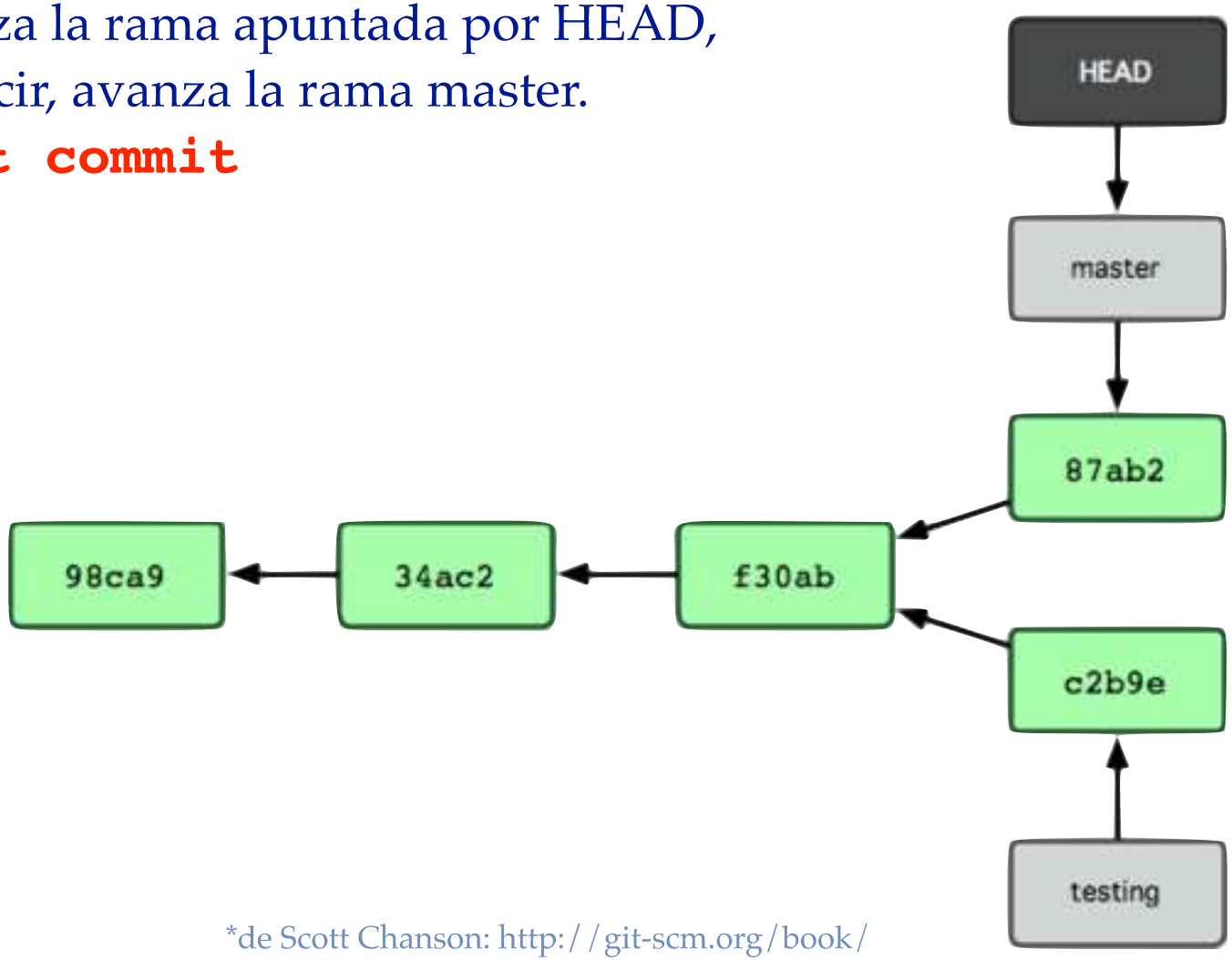
  - HEAD apuntará a la nueva rama.
  - Se actualiza el directorio de trabajo y el staging area.



*de Scott Chanson: http://git-scm.org/book/

- Si hacemos un nuevo commit:
  - avanza la rama apuntada por HEAD, es decir, avanza la rama master.

```
$ git commit
```



*de Scott Chanson: http://git-scm.org/book/

Crear Ramas y Cambiar de Rama

- Para crear una rama nueva en el punto de trabajo actual:

```
$ git branch <nombre>
```

- Para crear una rama nueva situada en un commit dado:

```
$ git branch <nombre> <commit>
```

- El comando **git checkout -b** permite crear una rama y cambiarse a ella con un solo comando:

```
$ git checkout -b <nombre>
```

Cambiar de Rama

- Para cambiar de rama:

```
$ git checkout <nombre>
```

- La opción **-b** se usa para crear una rama y cambiarse a ella:

```
$ git checkout -b <nombre>
```

- Checkout falla si existen cambios en los ficheros del directorio de trabajo o en el staging area que no están incluidos en la rama a la que nos queremos cambiar.
  - Podemos forzar el cambio usando la opción **-f**.
    - perderemos los cambios realizados
  - Podemos usar la opción **-m** para que nuestros cambios se mezclen con la rama que queremos sacar
    - Si aparecen conflictos, los tendremos que solucionar.

Más usos de git checkout

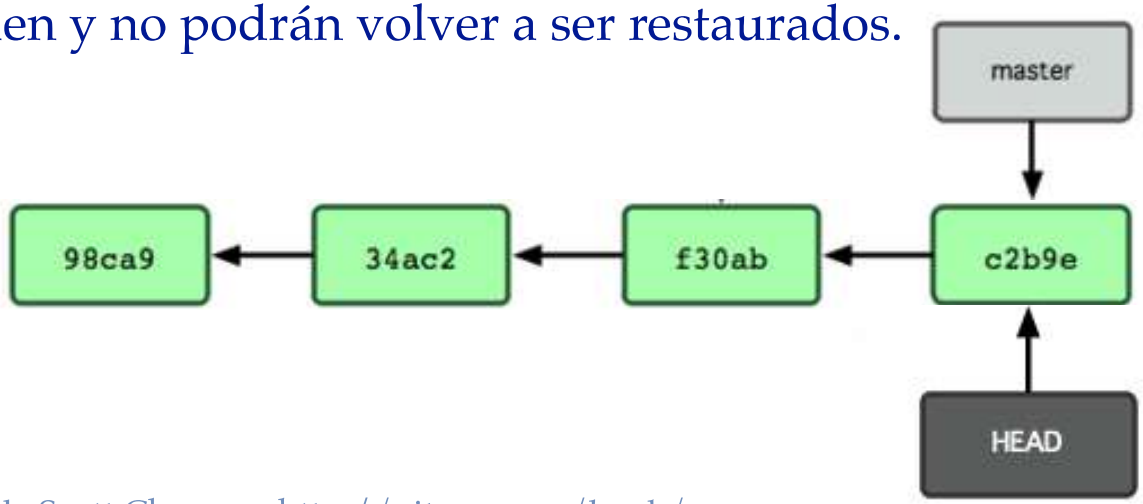
- Deshacer todos los cambios staged en el directorio de trabajo:

```
$ git checkout .
```

- Deshace los cambios stagged de <fichero> en directorio de trabajo:

```
$ git checkout -- <fichero>
```

- Los cambios se pierden y no podrán volver a ser restaurados.

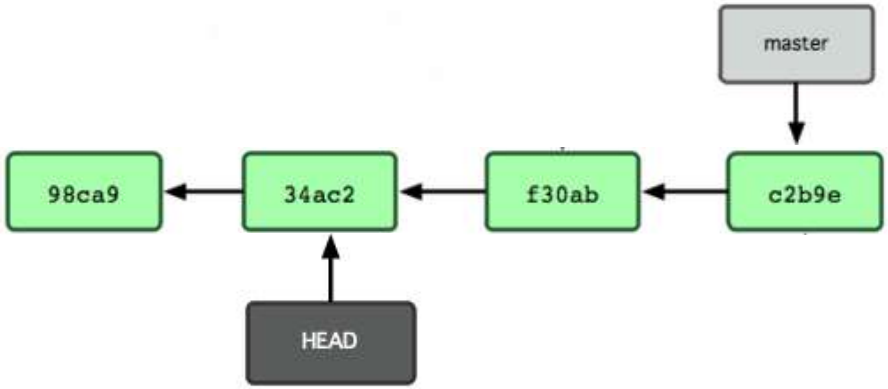


*de Scott Chanson: http://git-scm.org/book/

Detached HEAD Branch

- Es una rama apuntada por HEAD, pero que no tiene un nombre de rama apuntándole.
- Aparecen cuando se realiza checkout directamente sobre el identificador sha1 de un commit.

```
$ git checkout 34ac2
```



- Suele hacerse para inspeccionar en un momento dado los ficheros en un punto de la historia, pero no queremos crear una rama que no vamos a usar.
- En cualquier momento podemos crear una rama con nombre ejecutando:

```
$ git checkout -b <nombre_rama>
```

Integrar Ramas

- Para incorporar en la rama actual los cambios realizados en otra rama:

```
$ git merge <rama>
```

- Internamente GIT analiza la historia de commits para calcular como hacer la integración de las ramas.
  - Puede hacer un fast forward, una mezcla recursiva, ...

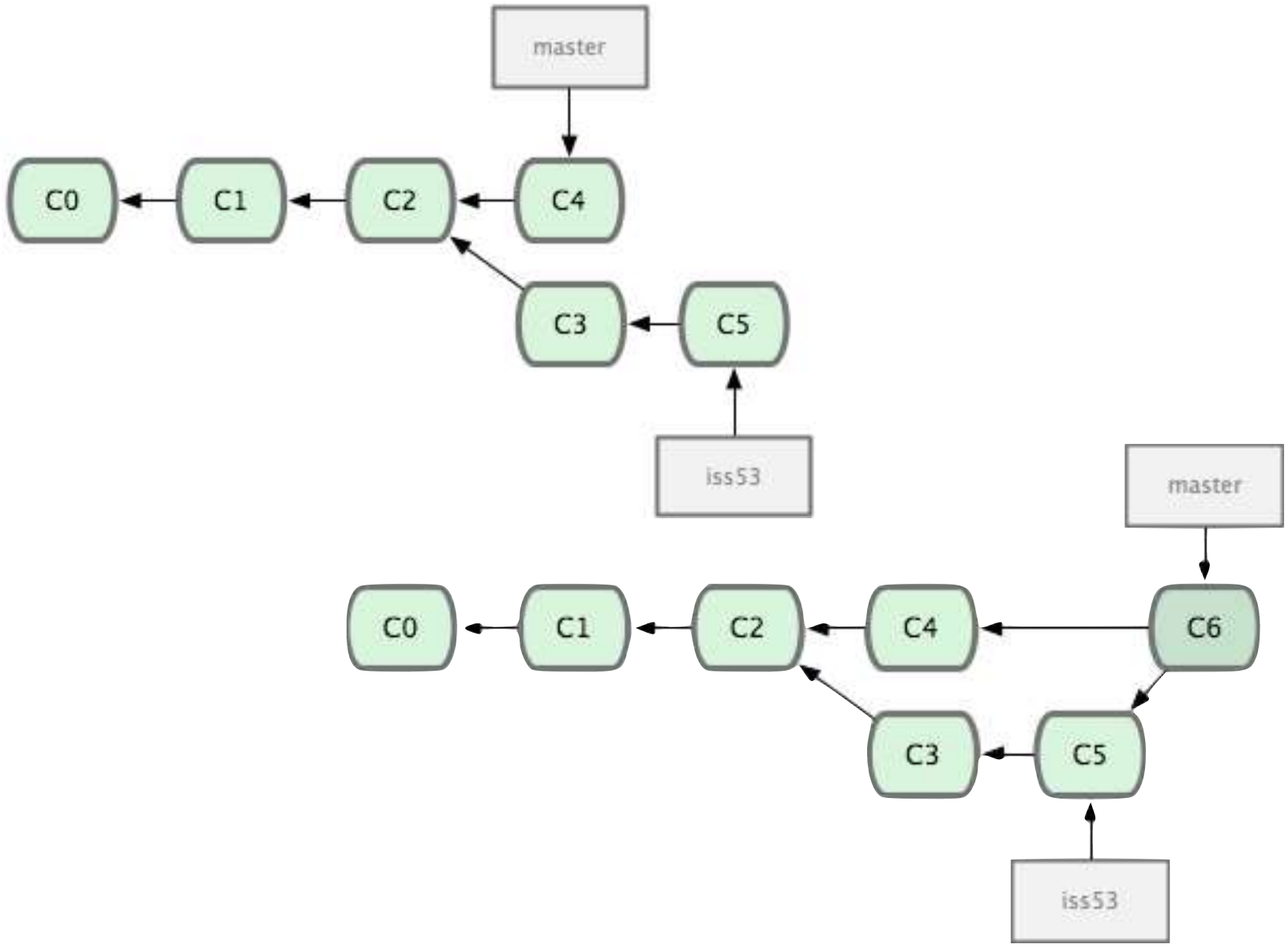
- Ejemplo:

```
$ git checkout master
```

  - Estamos en la rama master.

```
$ git merge iss53
```

  - Incorporamos los cambios hechos en la rama iss53 en la rama master.



*de Scott Chanson: http://git-scm.org/book/

Conflictos

- Al hacer el **merge** pueden aparecer conflictos
  - si las dos ramas han modificado las mismas líneas de un fichero.
- Si hay conflictos:
  - No se realiza el commit.
  - Las zonas conflictivas se marcan:

```
<<<<<<< HEAD:index.html
  <div id="footer">contact : email.support@github.com</div>
=======
  <div id="footer">contact us at support@github.com</div>
>>>>>>> iss53:index.html
```

  - El comando **git status** lista los ficheros con conflictos como **unmerged**.
- Para resolver el conflicto hay que:
  - editar el fichero para resolver el conflicto.
  - y después ejecutar **git add** y **git commit**.

Borrar Ramas

- Una ver terminado el trabajo con una rama
  - se borrar con el comando:

```
$ git branch -d <nombre>
```

  - Lo que se elimina es el puntero al commit.

- Si la rama a borrar no ha sido integrada con otra rama
  - se muestra un mensaje de error y no se borra.
  - Para borrar la rama independientemente de si ha sido integrada o no, usar la opción **-D** en vez de **-d**.

```
$ git branch -D <nombre>
```

Listar Ramas

- Para listar las ramas existentes:

```
$ git branch
```

- Ejemplo:

```
$ git branch
  iss53
* master
  testing
```

  - La rama activa se marca con un asterisco.

- Opciones:
  - **-r** muestra ramas remotas
  - **-a** muestra todas las ramas (locales y remotas)
  - **-v** muestra el último commit de la rama.
  - **--merged** muestra las ramas que ya se han mezclado con la la rama actual.
  - **--no-merged** muestra las ramas que aun no se han mezclado con la rama actual.