educación IT
CODE YOUR FUTURE

Git:
Desarrollo colaborativo

Módulo 2

Crear ramas

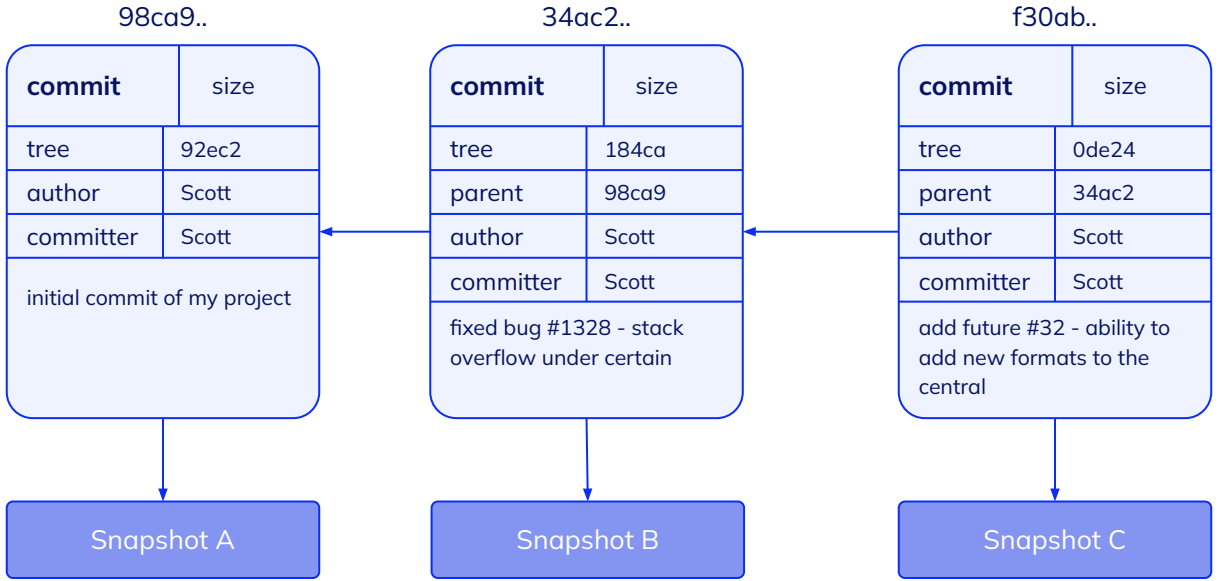
Ramas

Para entender realmente cómo ramifica Git, se debe examinar la forma en que almacena sus datos.

Como vimos anteriormente, Git no los almacena de forma incremental (guardando solo diferencias/deltas), sino que los almacena como una serie de snapshots (copias puntuales de los archivos completos, tal y como se encuentran en ese momento).

En cada confirmación de cambios (`commit`), Git almacena un punto de control que conserva:

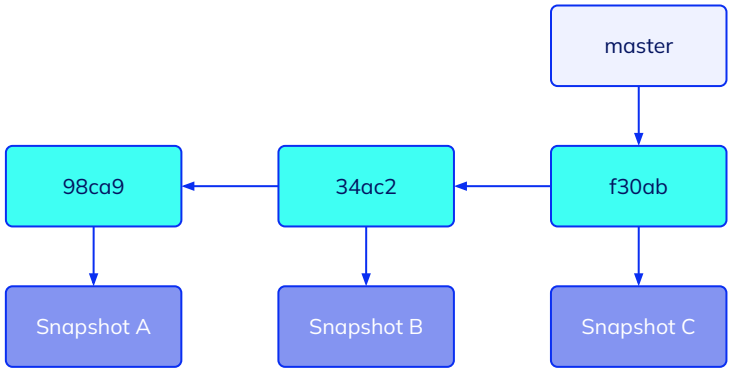
- Un **apuntador** a la copia puntual de los contenidos preparados (**`staged`**).
- **Metadatos** con el autor y el mensaje explicativo.
- Uno o varios **apuntadores a las confirmaciones** (**`commit`**) que sean padres directos de esta (un padre en los casos de confirmación normal, y múltiples padres en los casos de estar confirmando una fusión `-merge-` de dos o más ramas).


98ca9..
commit
size
tree
92ec2
author
Scott
committer
Scott
initial commit of my project
Snapshot A
34ac2..
commit
size
tree
184ca
parent
98ca9
author
Scott
committer
Scott
fixed bug #1328 - stack overflow under certain
Snapshot B
f30ab..
commit
size
tree
0de24
parent
34ac2
author
Scott
committer
Scott
add future #32 - ability to add new formats to the central
Snapshot C

Branch

Una ***branch* o rama** en GIT es simplemente un **apuntador móvil** que apunta a una de esas confirmaciones. **La rama por defecto de Git es la *master***. Con la primera confirmación de cambios que se realice, se creará esta rama principal master que apuntará a ella.

En cada confirmación de cambios que hagamos, la rama irá avanzando automáticamente. **La master apuntará siempre a la última confirmación realizada.**

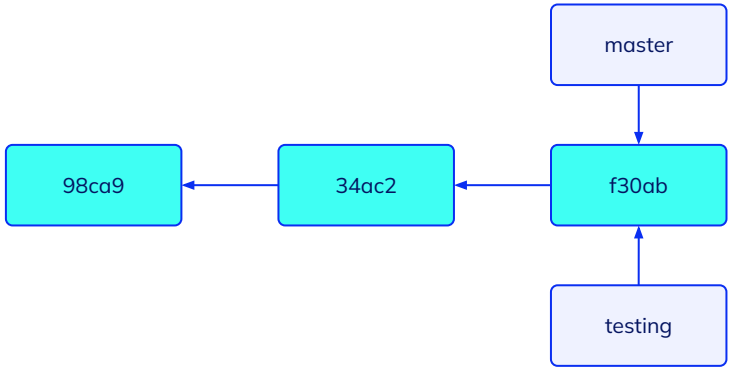


¿Qué sucede cuando creas una nueva rama?

Simplemente **se crea un nuevo apuntador** para que lo puedas mover libremente. Por ejemplo, para crear una nueva rama denominada "testing", usarás el comando **`git branch`**:

```
git branch testing
```

Esto creará un nuevo apuntador a la misma confirmación donde estés:

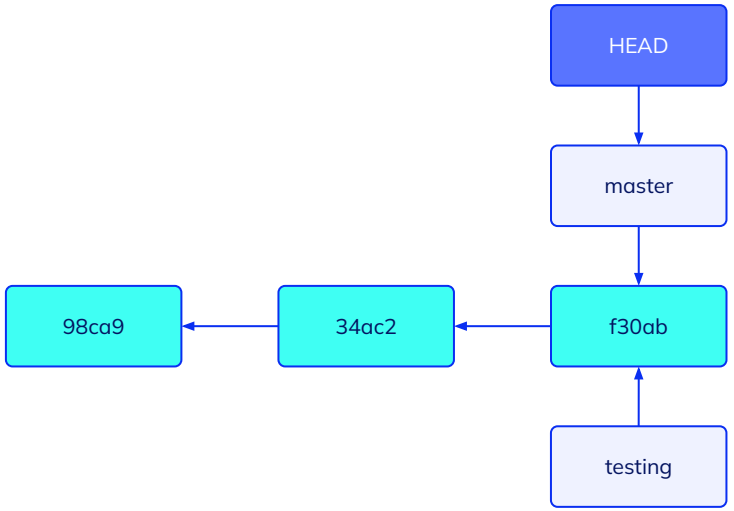


Head

¿Cómo sabe Git en qué rama estás en este momento?

Mediante un apuntador especial denominado `HEAD`. Es preciso comentar que este **`HEAD`** es totalmente distinto al concepto de *HEAD* en otros sistemas de control de cambios como *Subversion* o *CVS*.

En Git, es simplemente el **apuntador a la rama local en la que estés en ese momento.** En este caso, en la rama master, ya que el comando **`git branch`** solamente crea una nueva rama, y no salta a esa rama.



Navegar entre ramas

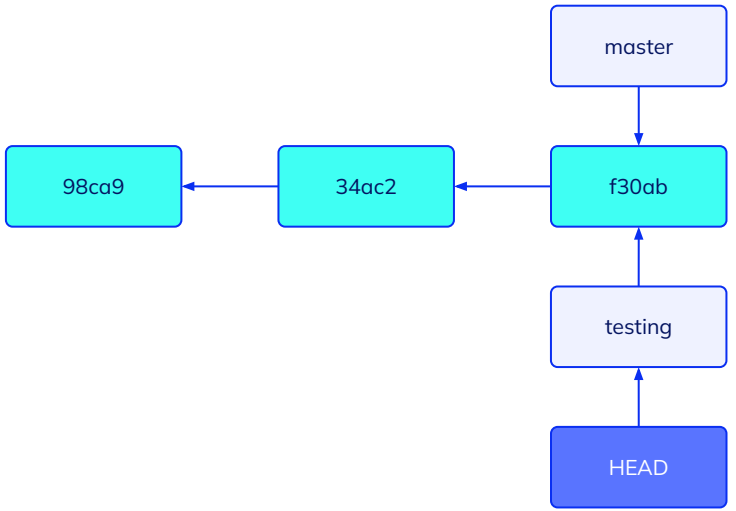
git checkout

Para **saltar de una rama a otra,** tienes que utilizar el comando **`git checkout`**.

Para hacer una prueba se saltará a la rama *testing* recién creada:

```
git checkout testing
```

Esto mueve el apuntador **`HEAD`** a la rama testing:

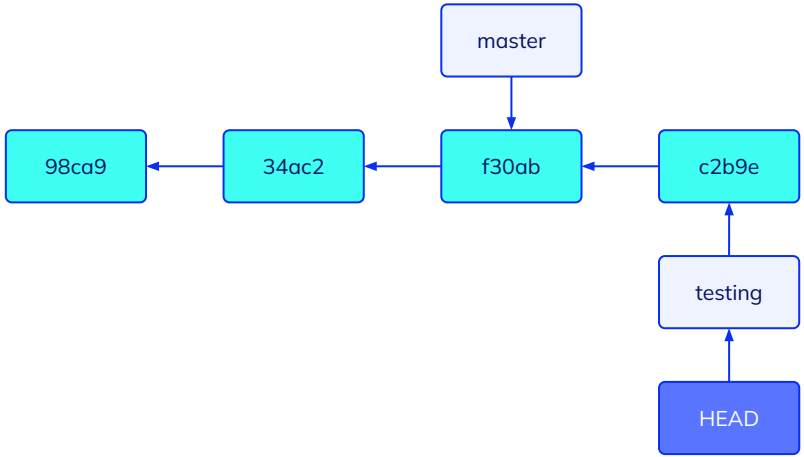


¿Cuál es el significado?

Lo veremos tras realizar otra confirmación de cambios:

```
git commit -a -m 'made a change'
```

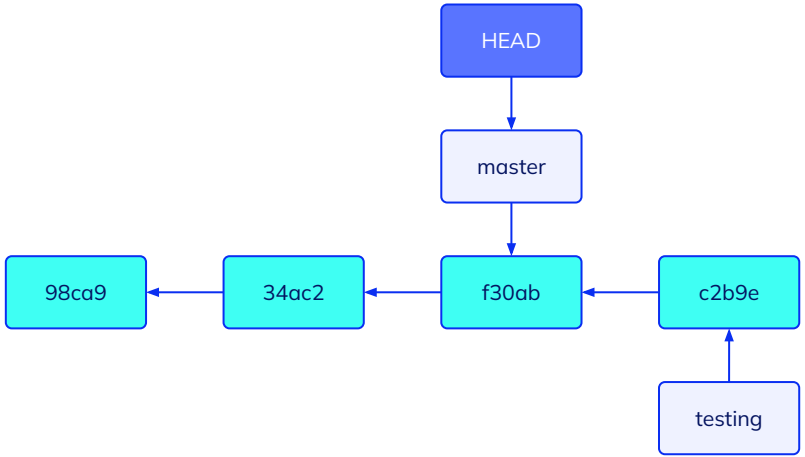
Realizar un checkout a testing



Se observa 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:`***

```
git checkout master
```

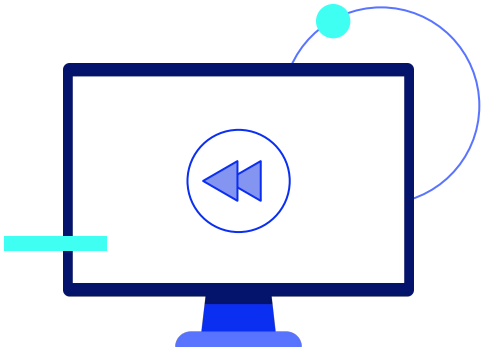
Realizar un checkout a master



El comando **`git checkout`** 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 esa rama ***master***.

Esto supone que los cambios que hagas desde este momento en adelante divergirán de la antigua versión del proyecto. Básicamente, lo que se está haciendo es rebobinar el trabajo que habías hecho temporalmente en la rama ***testing***; de tal forma que puedas avanzar en otra dirección diferente.



¡Sigamos trabajando!