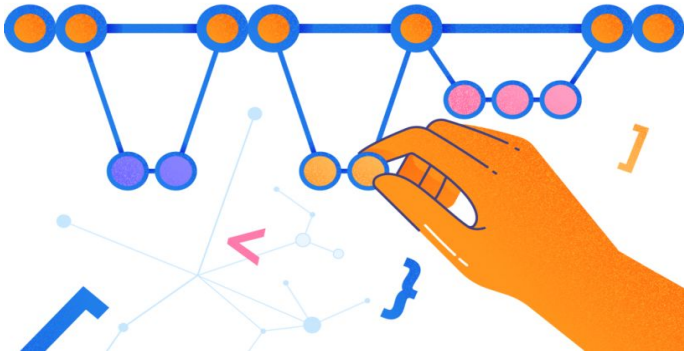
Git - branch, merge y resolución de conflictos



Matías Corvetto
Alejo Fraga
Juan Prina

Branching (uso de ramas)

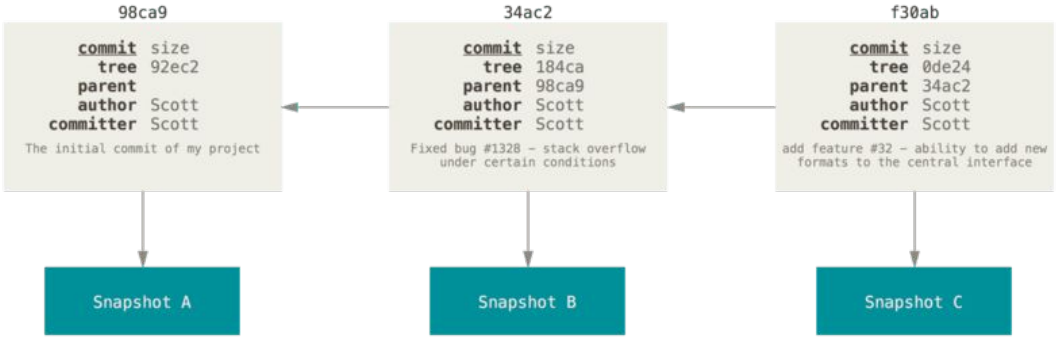
Sencillo, instantáneo y práctico

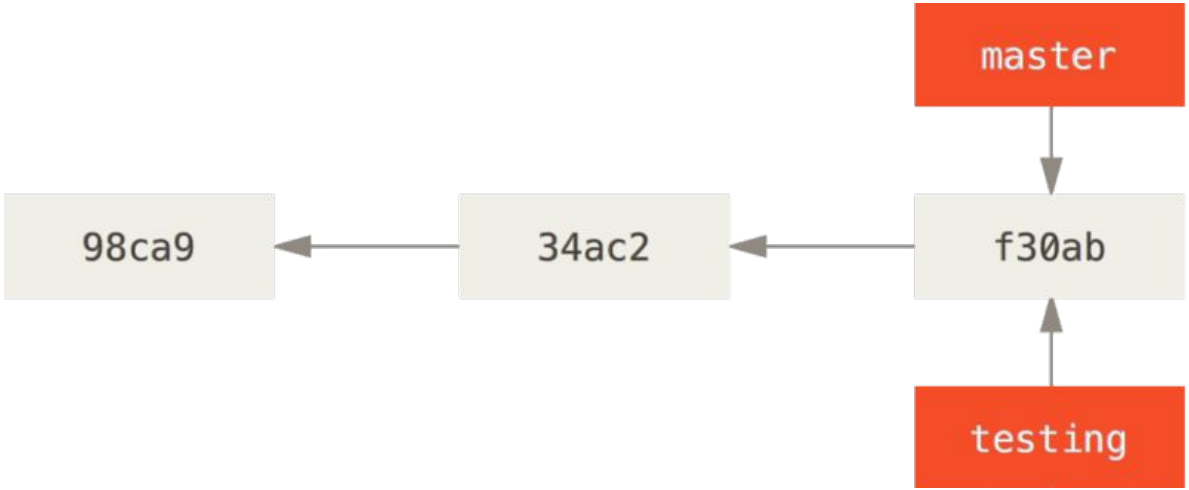
Uno de los aspectos más fuertes de Git es su capacidad para manejar ramificaciones, lo que lo distingue de otros sistemas de control de versiones.

Entender y manejar esta opción te proporciona una poderosa y exclusiva herramienta que puede **cambiar la forma en la que desarrollas**.

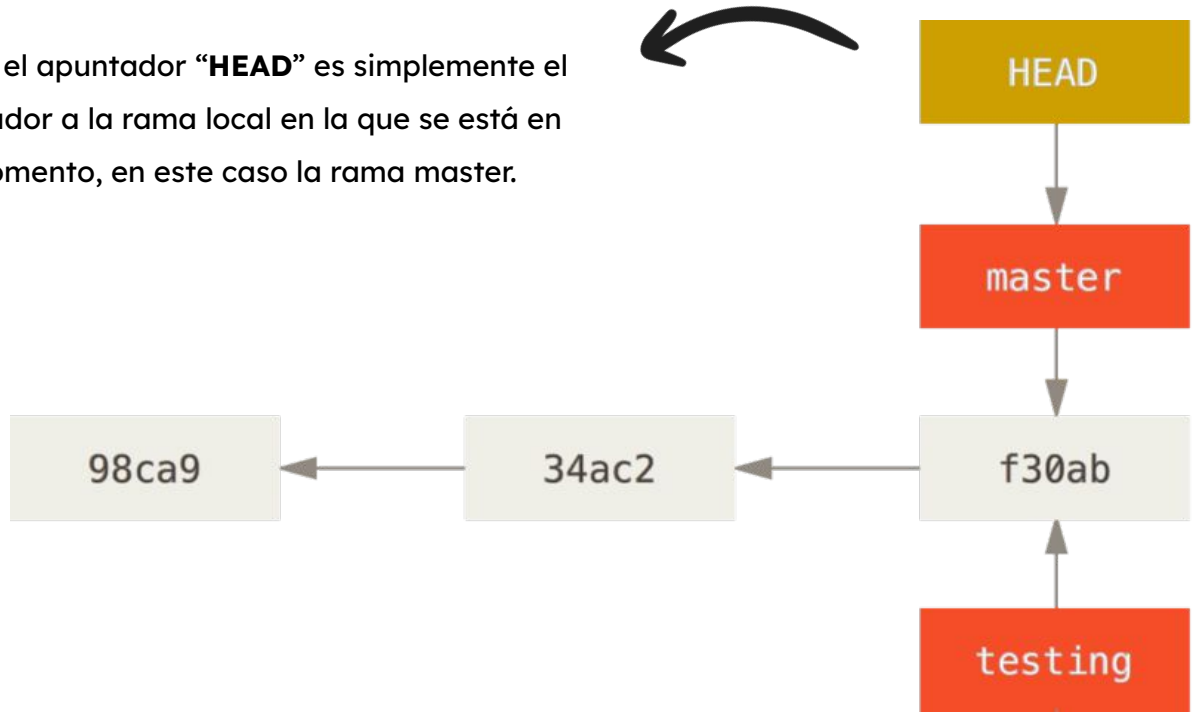
¿Cómo funciona una rama?

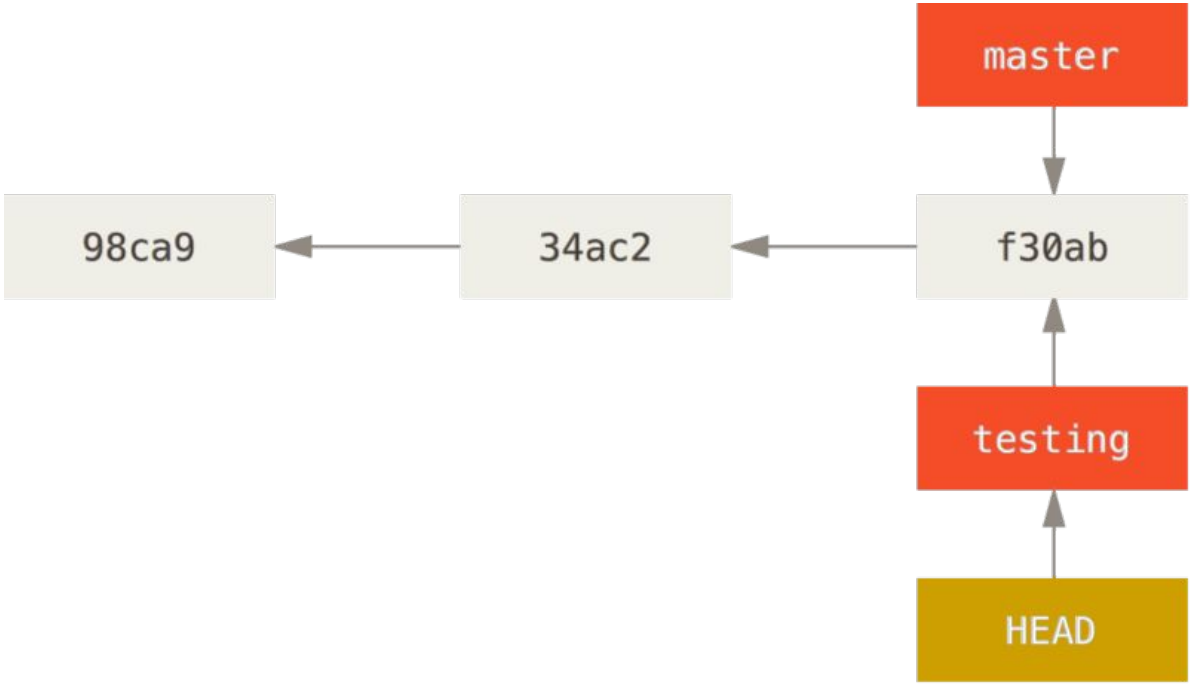
En cada confirmación de cambios (commit), Git guarda una instantánea del trabajo preparado, metadatos y un mensaje explicativo.
Además, incluye uno o más **apuntadores** a los commits que son padres directos de este nuevo commit (un padre solo en situaciones normales y múltiples padres cuando se fusionan (merge) dos o más ramas).

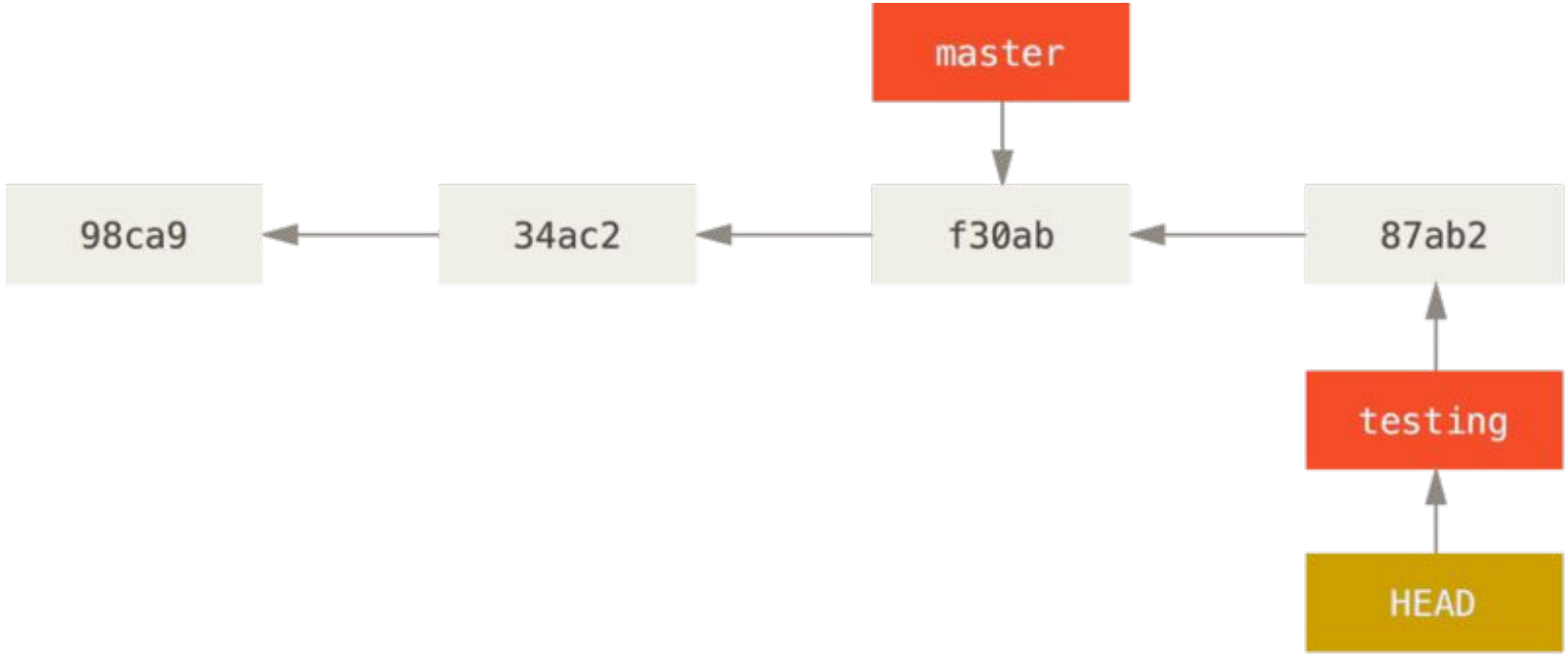


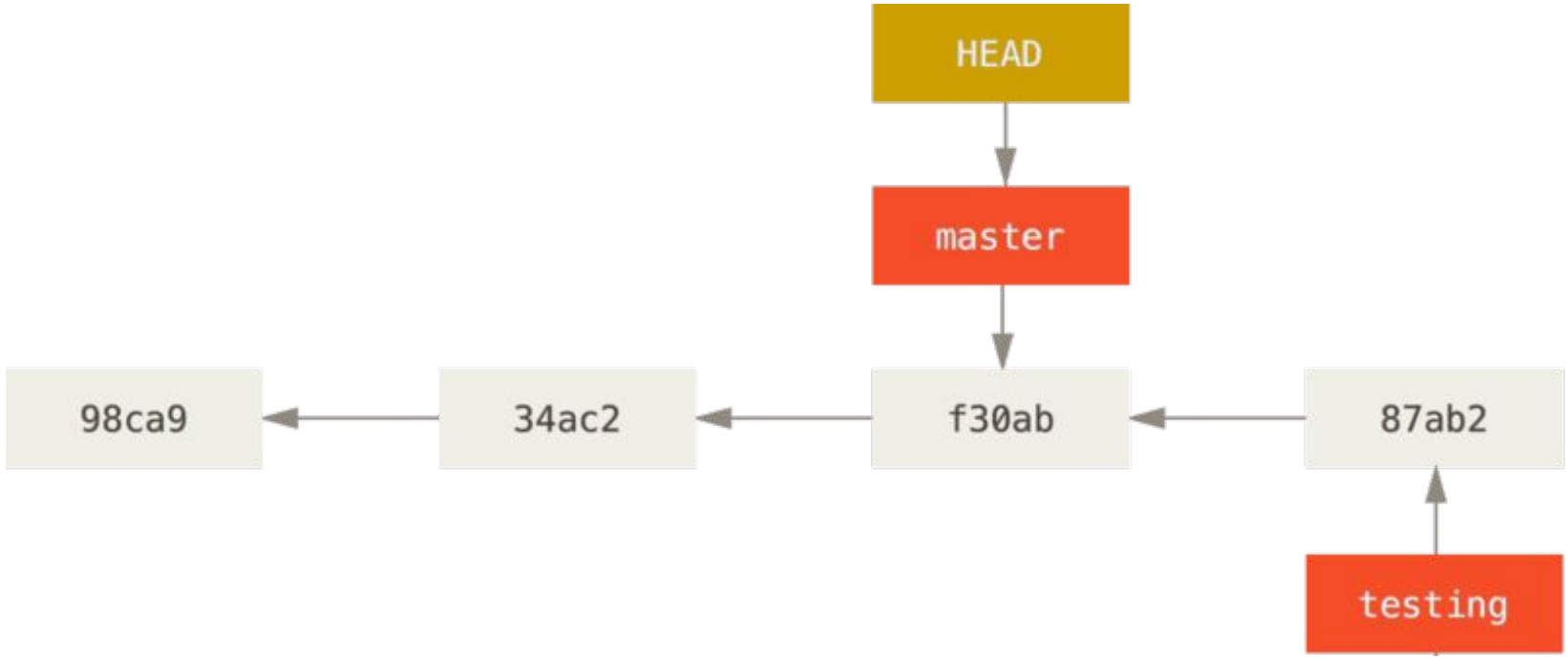

master
98ca9
34ac2
f30ab
testing

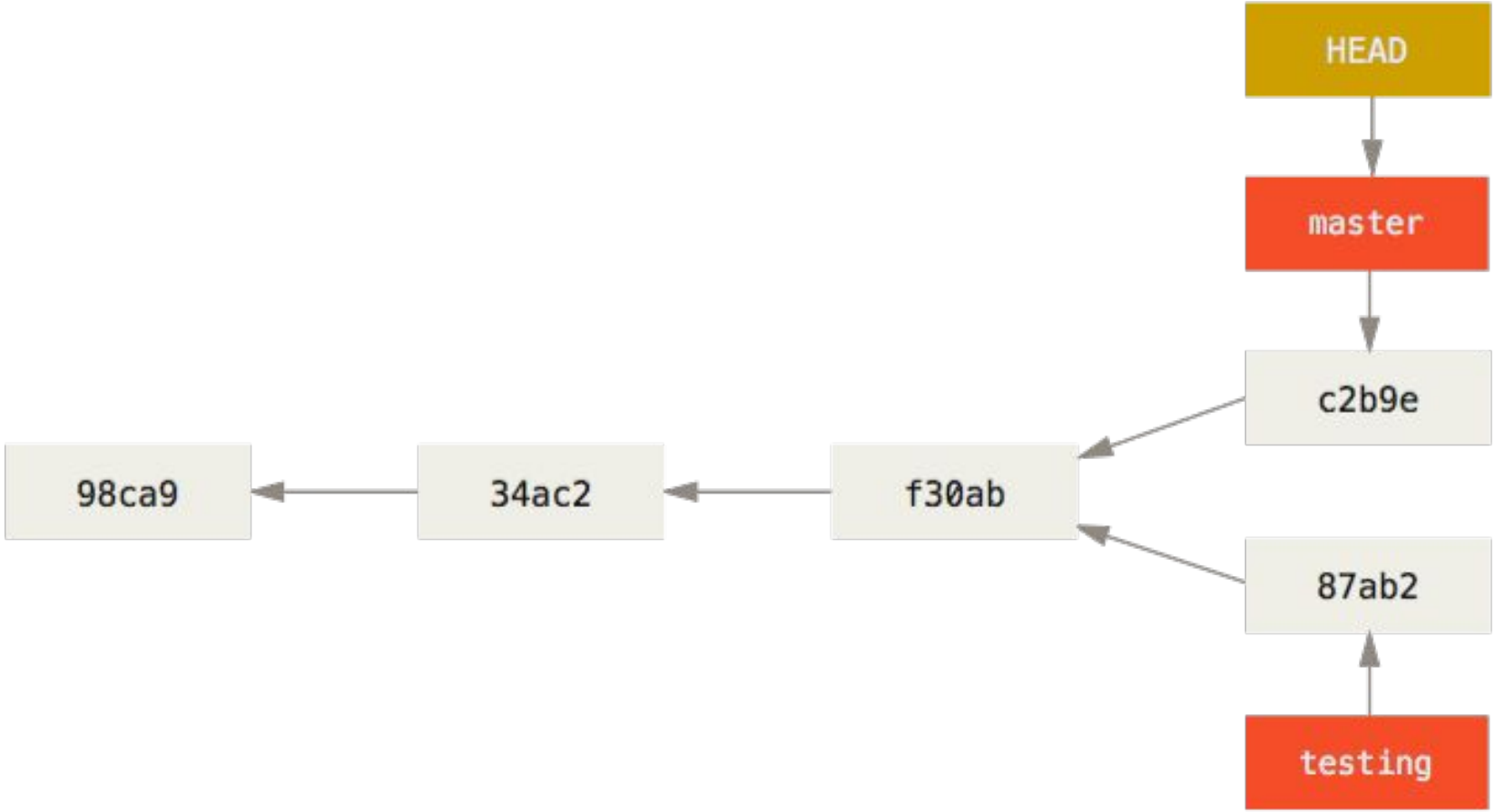
En Git, el apuntador “**HEAD**” es simplemente el apuntador a la rama local en la que se está en ese momento, en este caso la rama master.




master
98ca9
34ac2
f30ab
testing
HEAD


master
98ca9
34ac2
f30ab
87ab2
testing
HEAD


HEAD
master
98ca9
34ac2
f30ab
87ab2
testing


HEAD
master
c2b9e
98ca9
34ac2
f30ab
87ab2
testing

Comandos Útiles

`checkout`

`git checkout <id-commit>` El HEAD apuntará al commit específico con el id <id-commit>

`git checkout <rama>` El HEAD apuntará al extremo de la rama llamada <rama>.

`git checkout -b <rama>` Se crea una nueva rama llamada <rama> y el HEAD pasará a apunta al extremo de esta simultáneamente.

`git checkout -` Se deshace el desplazamiento anterior (switch o checkout) y el HEAD vuelve a la posición anterior.

Comandos Útiles

`switch`.

`git switch <rama>`	El HEAD apuntará al extremo de la rama llamada <rama>.
`git switch -c <rama>`	Se crea una nueva rama llamada <rama> y el HEAD
`git switch -`	Se deshace el desplazamiento anterior (switch o checkout) y el HEAD vuelve a la posición anterior.

Comandos Útiles

`branch`.

```
git branch
```

Enumera todas las ramas de tu repositorio. Es similar a git branch --list.

```
git branch <rama>
```

Crea una nueva rama llamada <rama>.

```
git branch -d <rama>
```

Elimina la rama especificada. Esta es una operación segura, ya que Git evita que elimines la rama si tiene cambios que aún no se han fusionado.

```
git branch -D <rama>
```

Fuerza la eliminación de la rama especificada, incluso si tiene cambios sin fusionar.

Comandos Útiles

```
git branch -m <rama>
```

Cambia el nombre de la rama actual a <rama>.

```
log.
```

```
git log
```

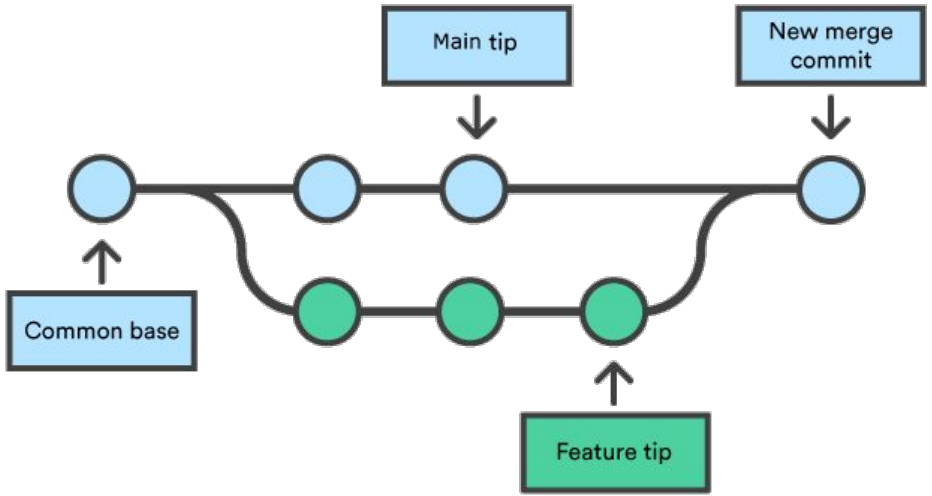
Muestra el historial de commits.

```
git log --oneline
```

Comprime cada commit en una sola línea.

Merge

En el contexto de GitHub y control de versiones, "merge" se refiere a la acción de combinar los cambios realizados en una rama en otra, generalmente en la rama principal.



PASOS:

- ## Creación de la Rama de Trabajo
  Rama en la cual se estará trabajando.

- ## Trabajar en la Rama creada
  Edición de la Rama creada de manera local.

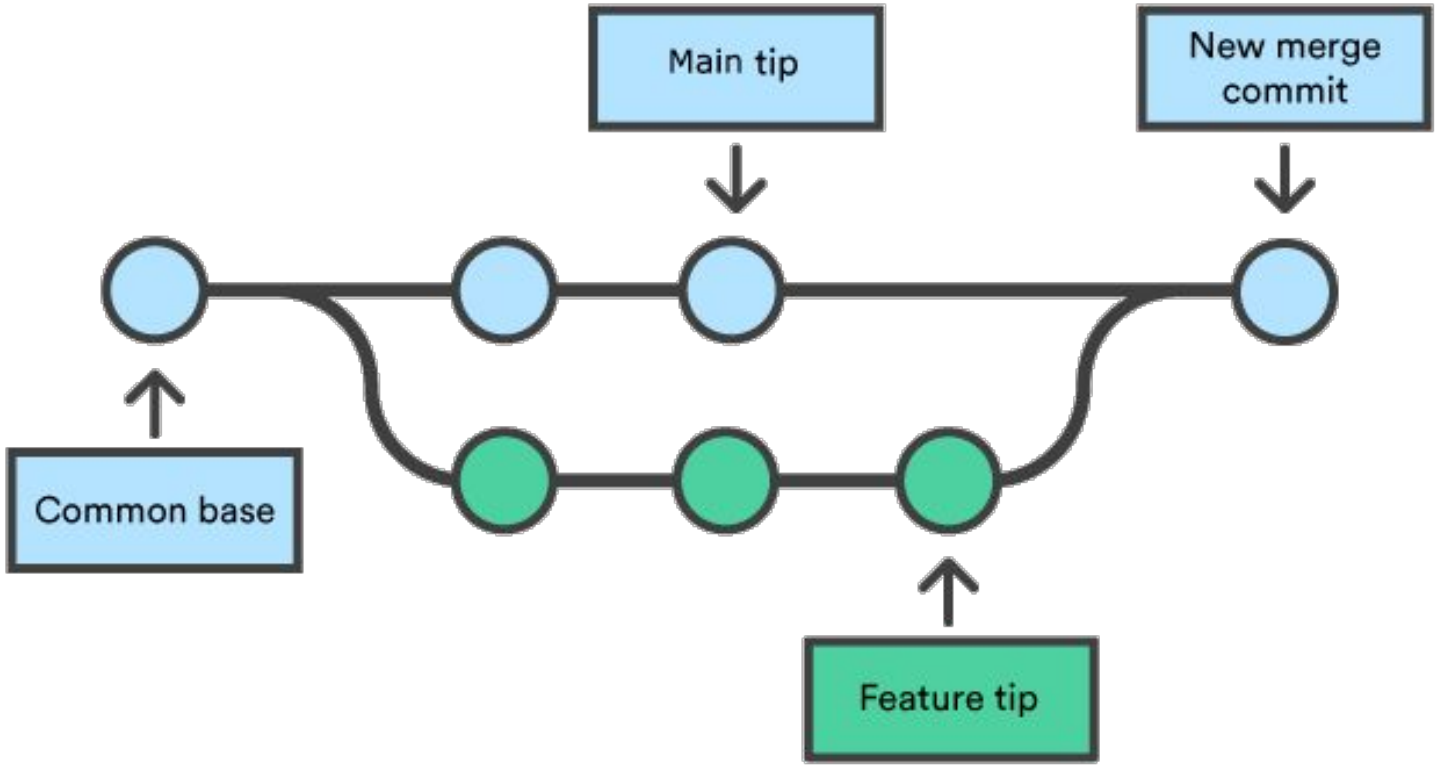
- ## Crear Pull Request / merge manual
  Después de hacer los cambios, se crea una Pull Request a la rama principal. Se detallan los cambios que se van fusionar y permite a otros revisar las modificaciones.
  Merge manual : git merge <rama>

PASOS:

- ## Revisión y Resolución de Conflictos
- ## Aprobación y Merge

  Una vez que el Pull Request fue revisado y aprobado por el equipo, se puede hacer el Merge. Importante proporcionar un mensaje descriptivo que detalle la fusión.
- ## Borrar la Rama de Trabajo

  Paso opcional. Después de que los cambios se hayan incorporado en la rama principal, se puede eliminar la rama de trabajo si ya no es necesaria.


Main tip
New merge commit
Common base
Feature tip

Analicemos la siguiente situación...

Se tienen dos ramas (A y B).

En la rama A se realizan una serie de cambios en el archivo “ejemplo.txt”

En la rama B se realizan otros cambios en el mismo archivo y en la misma área.

Cuando se intente hacer un Merge de las ramas, Git notará que ambas ramas han tocado las mismas líneas del archivo de forma diferente y no sabrá decidir cuál de los dos cambios finalmente se hará.

¿QUÉ SUCEDE? SE PRODUCE UN **CONFLICTO**

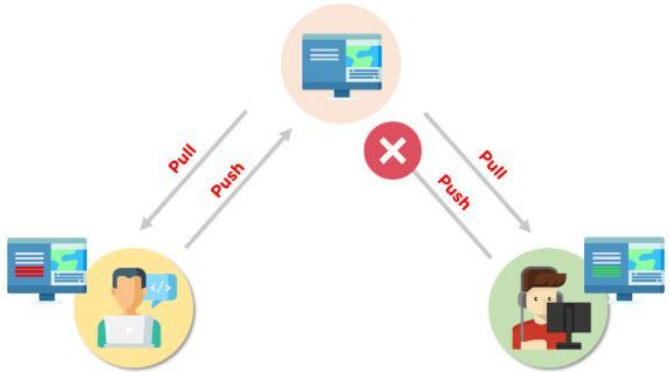
CONFLICTO

Refiere a una situación en la que el sistema no puede determinar automáticamente cómo combinar cambios en dos o más versiones diferentes de un archivo.

Por lo tanto se requiere de la intervención del usuario.

TIPOS DE CONFLICTO

- Conflictos de fusión (merge conflicts)
- Conflictos de rebase (rebase conflicts)
- Conflictos de cherry-pick (cherry-pick conflicts)
- Conflictos por cambios simultáneos en un mismo archivo



CONFLICTO DE FUSIÓN (merge conflict):

- Es el tipo de conflicto más común.
- El escenario mostrado inicialmente es un claro ejemplo.
- Se produce exclusivamente cuando dos o más ramas realizan cambios en el mismo archivo y en la misma área.

RESOLUCIÓN DE CONFLICTOS

Cuando Git detecta un conflicto resaltará los archivos involucrados y colocará marcadores especiales en el archivo para delimitar las diferencias entre las versiones en conflicto.

```
<<<<<<<
(Cambios en la rama actual)
=======
(Cambios en la otra rama)
>>>>>>>
```

RESOLUCIÓN DE CONFLICTOS

1. Identificación
2. Análisis
3. Eliminación de marcadores
4. Confirmación
5. Finalización de la operación

RECOMENDACIONES DE USO

- Planificar y nombrar las ramas de manera descriptiva
- Asignar tareas específicas para cada rama
- Fusionar regularmente para evitar conflictos masivos
- Resolver conflictos uno por uno de forma metódica
- Mantener a tu equipo informado sobre conflictos y resoluciones

Recursos de aprendizaje

- https://learngitbranching.js.org/?locale=es_ARNODEMO

PRÁCTICA EN CLASE