



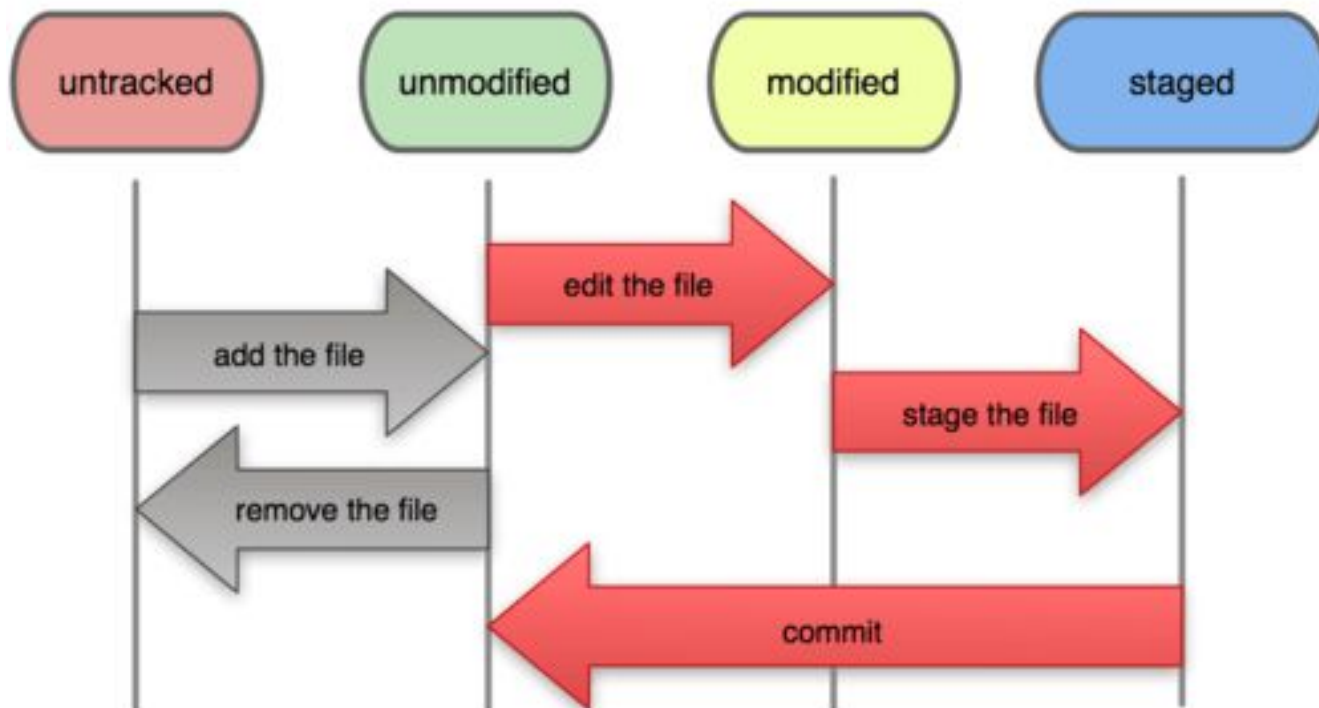
[kokoa]

Git like a boss.

Sistemas de versionamiento de código

Guardando cambios en el repositorio

File Status Lifecycle





Ingñorando archivos.

- | - A menudo se tienen archivos no deseados.
- | - Como por ejemplo archivos temporales, binarios de compilación.
- | Archivos temporales.



Seamos más precisos

Diferencias entre:

ver diferencias para archivos no confirmados:

```
git diff
```

ver diferencias para archivos confirmados:

```
git diff --cached
```

Eliminando archivos.

- | Dejar de seguir archivos:
- | `git rm <file> --cached`
- | Expresiones regulares
- | `*.[exp1, exp2, ..]`

Moviendo archivos

- | `git mv <file>`

- | A diferencia de muchos otros VCSs, Git no hace un seguimiento explícito del movimiento de archivos. Si renombbras un archivo, en Git no se almacena ning ún metadata que indique que lo has renombrado. Sin embargo, Git es suficientemente inteligente como para darse cuenta — trataremos el tema de la detección de movimiento de archivos un poco más adelante.

Histórico de confirmaciones

- | `git log`
- | Sirve para ver el histórico de commits.
- | El comando `git log` proporciona gran cantidad de opciones para mostrarte exactamente lo que buscas. Aquí veremos algunas de las más usadas.
- | `git log -p -2`
- | `git log -p -2 <filename>`
- | `git log -p -2 --word-diff <filename>`
- | `git log -p -2 --word-diff --stat <filename>`
- | `git log --no-merges`
- | `git log --pretty=oneline`
- | `git log --since=2.weeks`
- | `git log --committer='Jony Rodriguez'`



Deshaciendo cosas

- | En cualquier momento puedes querer deshacer algo. Veremos algunas herramientas básicas para deshacer cambios. Ten cuidado, porque no siempre puedes volver atrás después de algunas de estas operaciones. Ésta es una de las pocas áreas de Git que pueden provocar que pierdas datos si haces las cosas incorrectamente.




Modificando el último commit.

- | `git commit --amend`
- | ¡Olvidé confirmar un archivo!
- | `git add <archivo_olvidado>`
- | `git commit --amend`



Deshaciendo la preparación de un archivo

- | Por error agregé un archivo a la fase de preparados
- | `git reset HEAD <filename>`



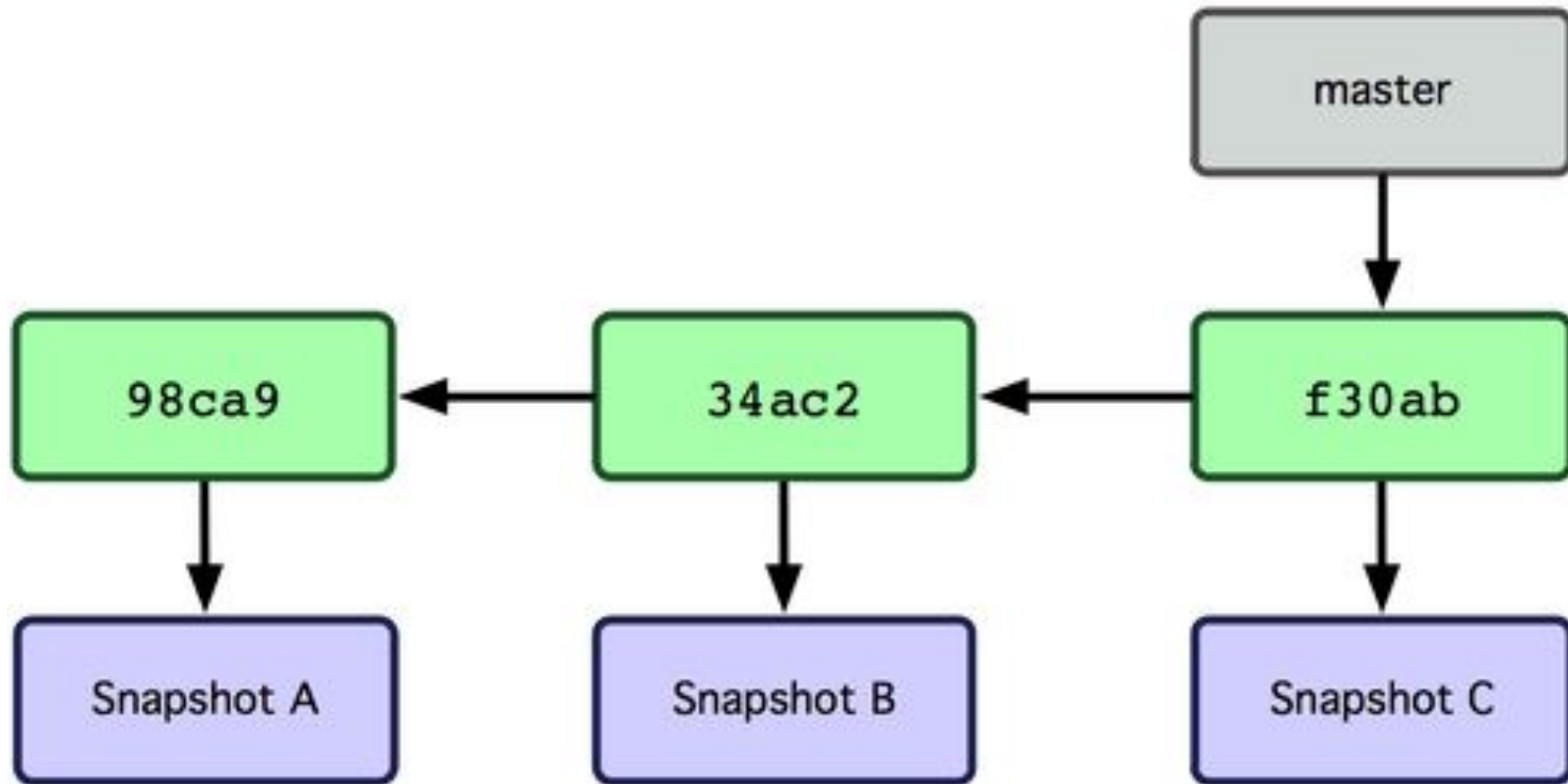
Deshaciendo la modificación de un archivo

- | ¿Qué pasa si te das cuenta de que no quieres mantener las modificaciones que has hecho sobre el archivo? ¿Cómo puedes deshacerlas fácilmente —revertir el archivo al mismo estado en el que estaba cuando hiciste tu última confirmación (o cuando clonaste el repositorio, o como quiera que metieses el archivo en tu **directorio de trabajo**)?
- | `git checkout -- <filename>`
- | #Por favor, ten cuidado.



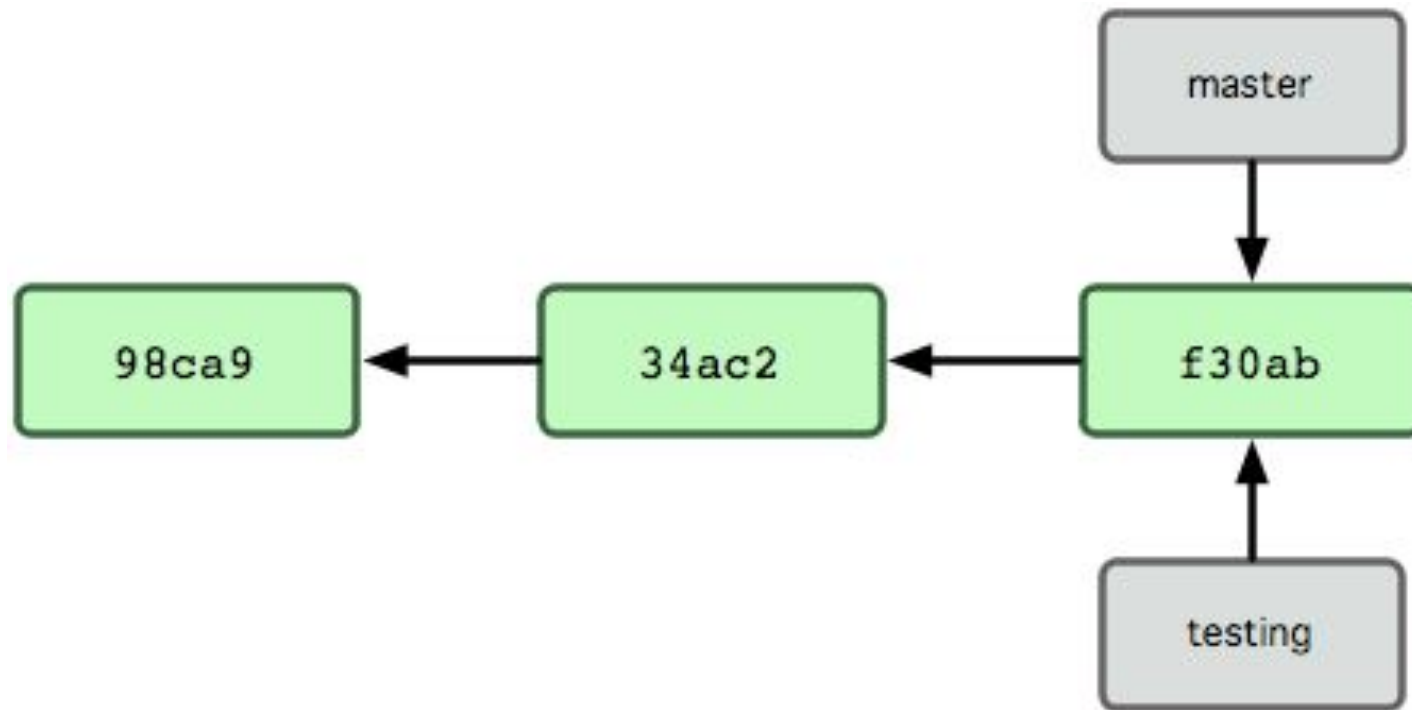
Ramificaciones

Branches, conceptos.



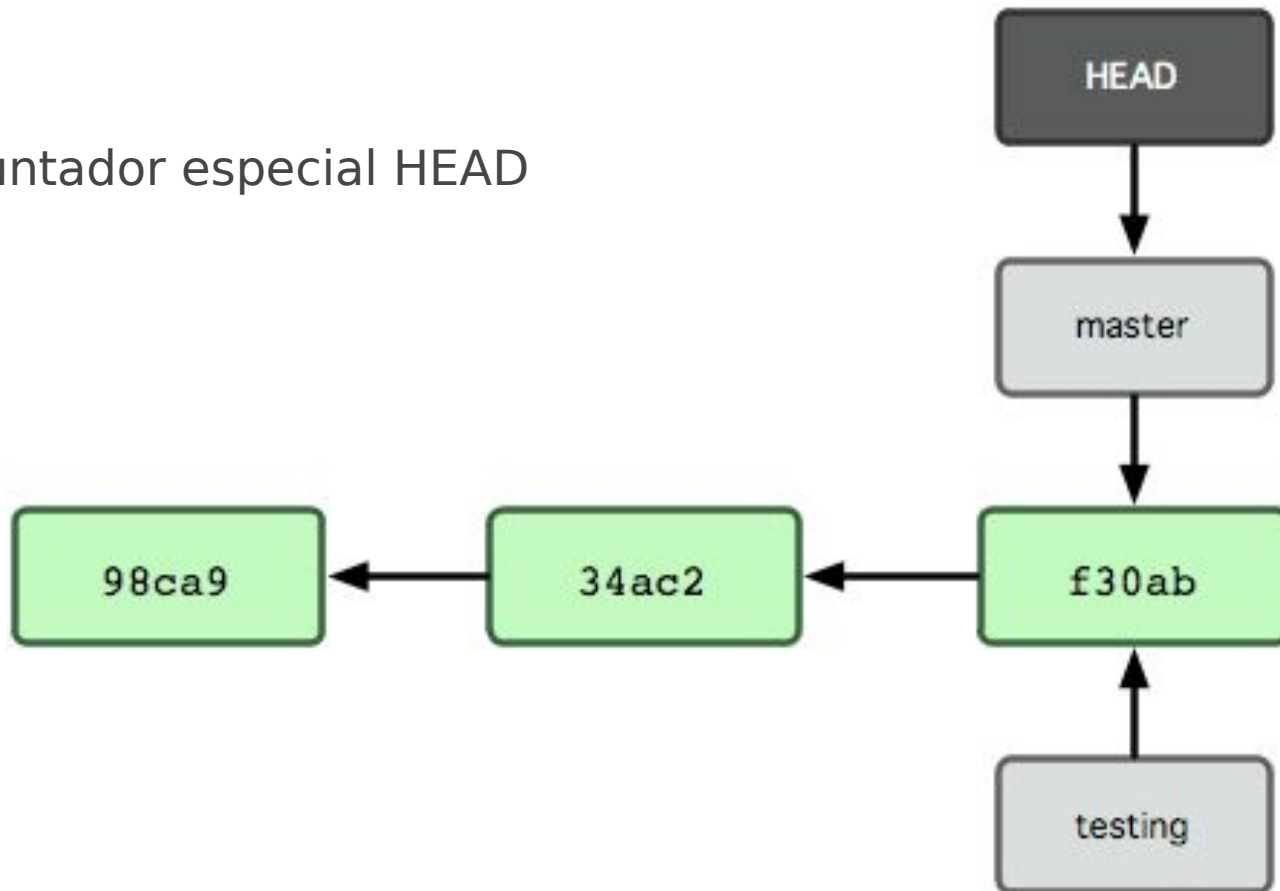
Branches, conceptos.

git branch testing



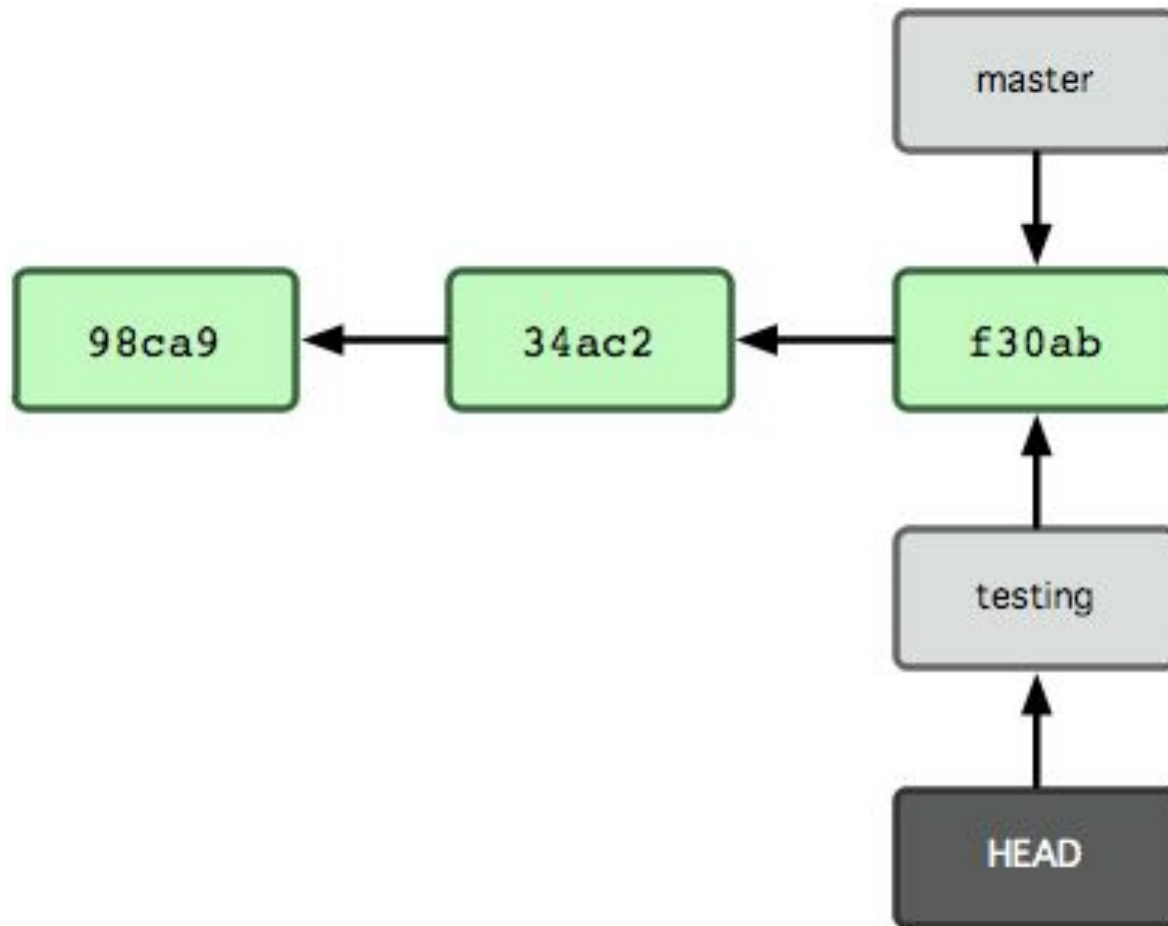
Y, ¿cómo sabe Git en qué rama estás en este momento?

Apuntador especial HEAD

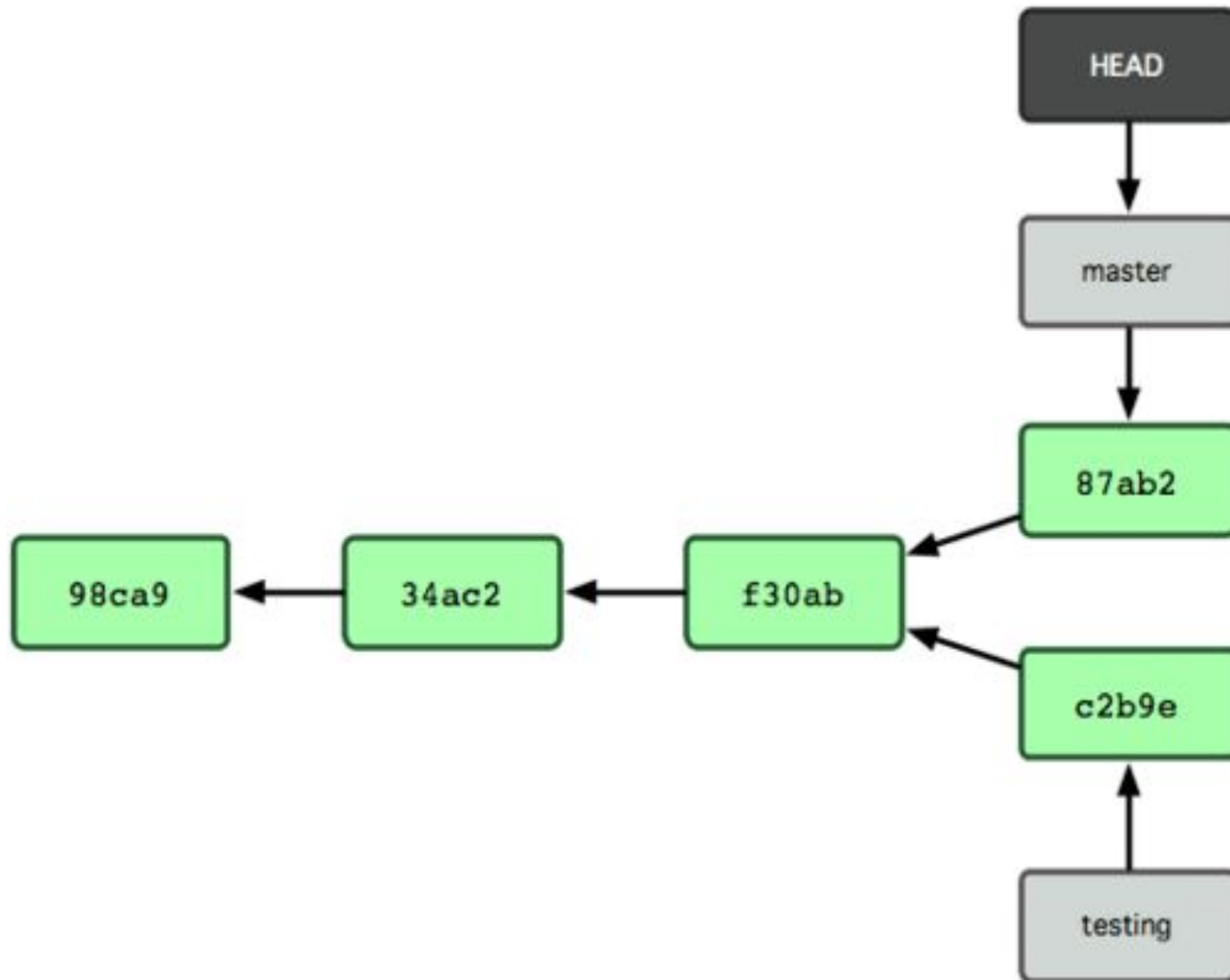


Salto entre ramas.

- | git checkout testing



Vista actual





En otros SCV

- | En los que crear una nueva rama supone el copiar todos los archivos del proyecto a una nueva carpeta adicional. Lo que puede llevar segundos o incluso minutos, dependiendo del tamaño del proyecto. Mientras que en Git el proceso es siempre instantáneo. Y, además, debido a que se almacenan también los nodos padre para cada confirmación, el encontrar las bases adecuadas para realizar una fusión entre ramas es un proceso automático y generalmente sencillo de realizar
- | Alienta a los desarrolladores a usar ramas por las facilidades que ofrecen.



Manejo de errores con ramas.

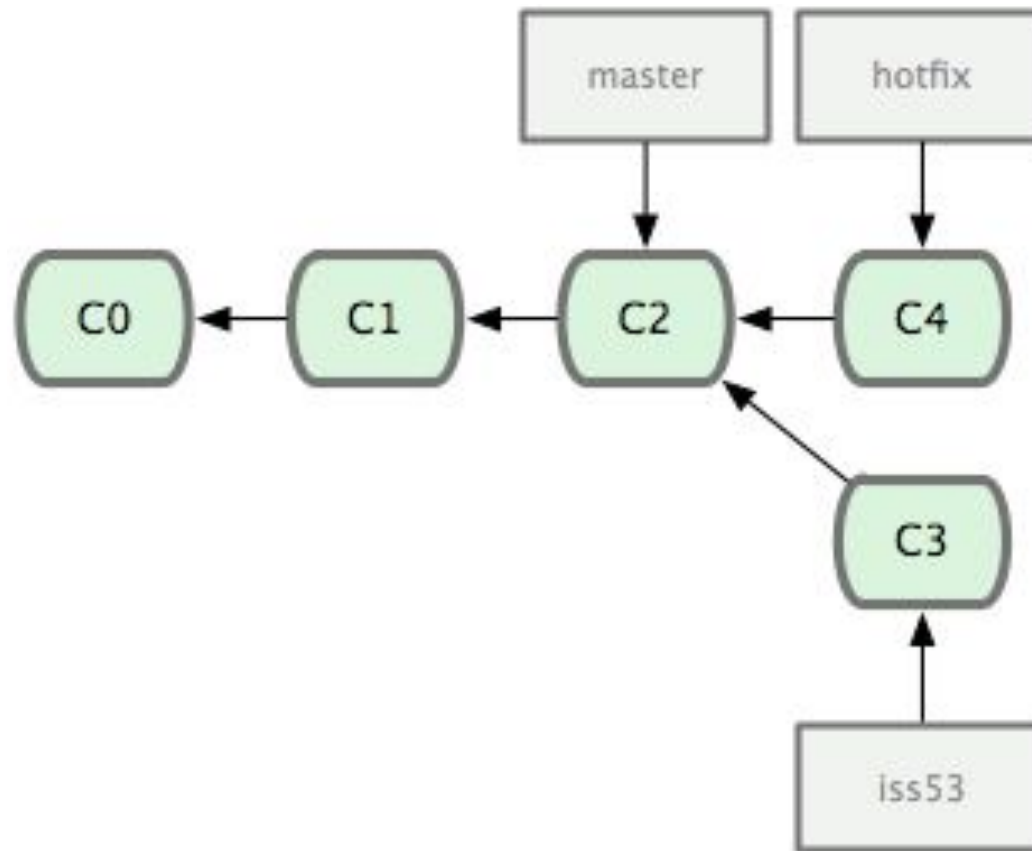
`git checkout -b <branch-name>`

Atajo a:

`git branch <branch-name>`

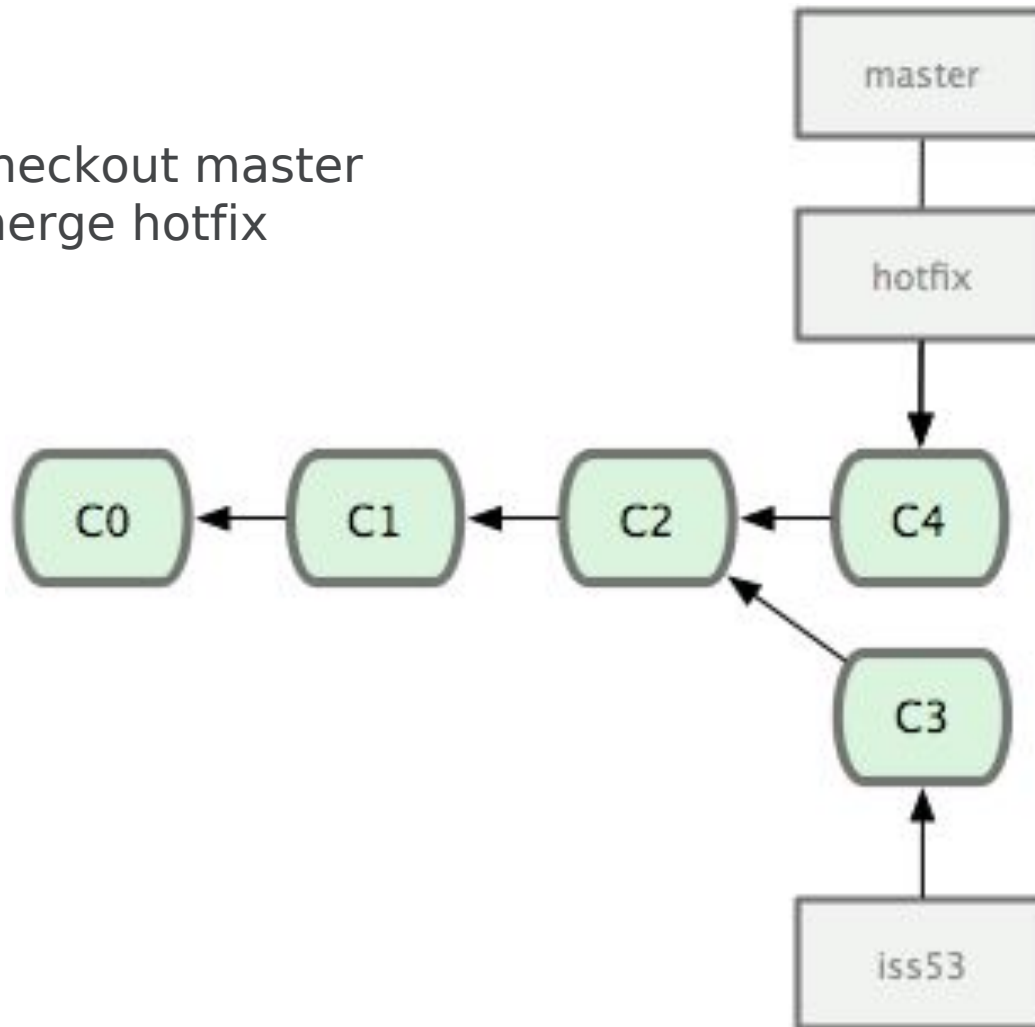
`git checkout <branch-name>`

Rama hotfix basada en la rama master original.



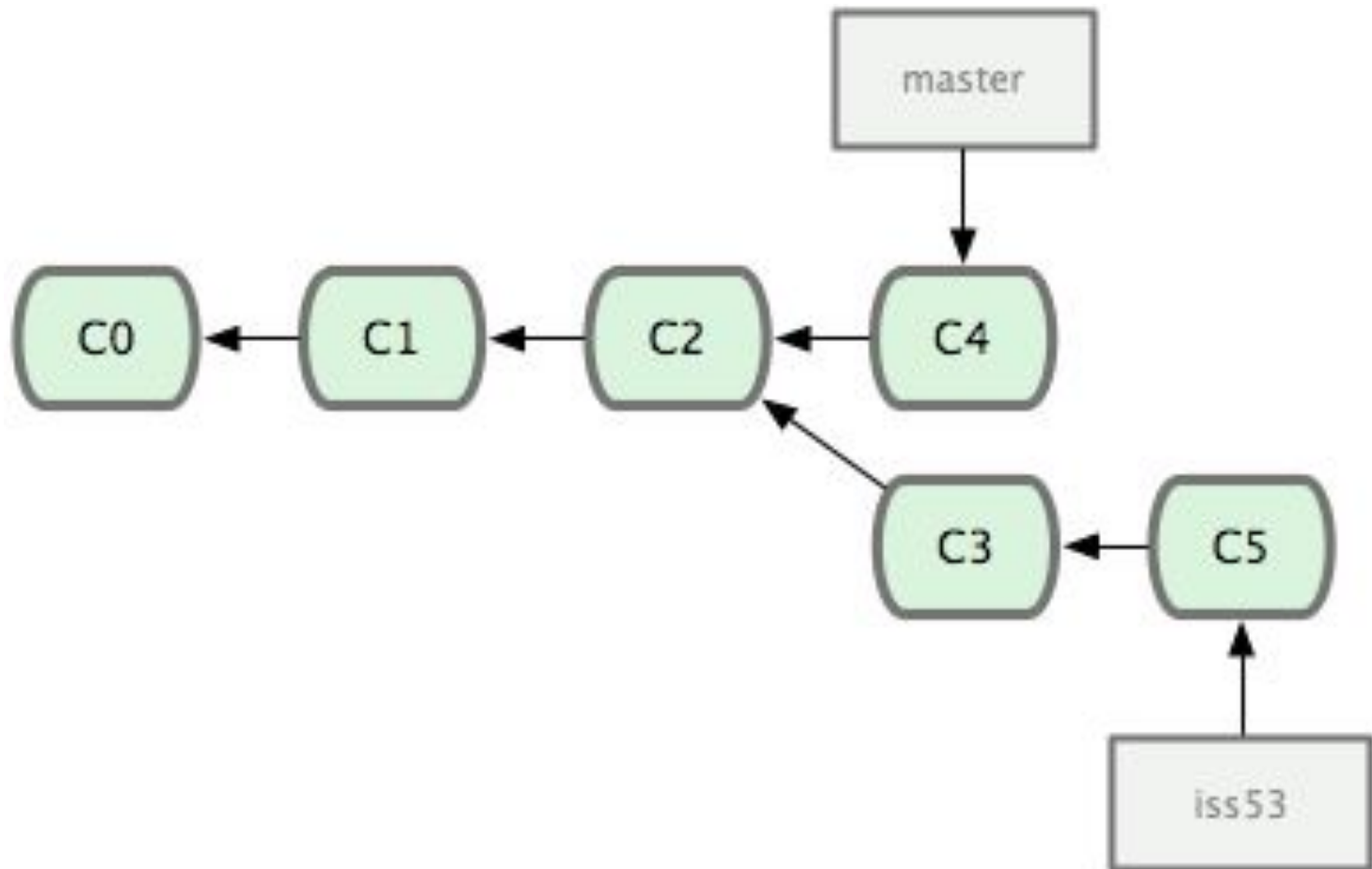
Integrando ramas.

git checkout master
git merge hotfix



Borrando ramas

`git branch -d hotfix`





Git en un servidor.

- | ¿Por qué tener un servidor git?
- | Tener un repositorio fiable.
- | Funcionando 24/7
- | No será necesario utilizar todo un servidor entero para él.
- | Un repositorio remoto es normalmente un repositorio básico mínimo, un repositorio Git sin carpeta de trabajo.



Elección del protocolo.

Protocolo Local

El más básico es el Protocolo Local, donde el repositorio remoto es simplemente otra carpeta en el disco. Se utiliza habitualmente cuando todos los miembros del equipo tienen acceso a un mismo sistema de archivos, como por ejemplo un punto de montaje NFS



El Procotolo SSH

- | Probablemente, SSH sea el protocolo más habitual para Git. Debido a disponibilidad en la mayor parte de los servidores; (pero, si no lo estuviera disponible, además es sencillo habilitarlo). Por otro lado, SSH es el único protocolo de red con el que puedes fácilmente tanto leer como escribir. Los otros dos protocolos de red (HTTP y Git) suelen ser normalmente protocolos de solo-lectura

El Protocolo Git

- | El protocolo Git es un demonio (daemon) especial, que viene incorporado con Git. Escucha por un puerto dedicado (9418), y nos da un servicio similar al del protocolo SSH; pero sin ningún tipo de autenticación.
- | Ventajas.- El protocolo Git es el más rápido de todos los disponibles. Si has de servir mucho tráfico de un proyecto público o servir un proyecto muy grande
- | Desventajas
- | La pega del protocolo Git, es su falta de autenticación.

El protocolo HTTP/S

Por último, tenemos el protocolo HTTP. Cuya belleza radica en la simplicidad para habilitarlo. Basta con situar el repositorio Git bajo la raíz de los documentos HTTP y preparar el enganche (hook) 'post-update' adecuado.

Ventajas

La mejor parte del protocolo HTTP es su sencillez de preparación. Simplemente lanzando unos cuantos comandos, dispones de un método sencillo de dar al mundo entero acceso a tu repositorio Git. En tan solo unos minutos. Además, el protocolo HTTP no requiere de grandes recursos en tu servidor. Por utilizar normalmente un servidor HTTP estático, un servidor Apache estandar puede con un tráfico de miles de archivos por segundo; siendo difícil de sobrecargar incluso con el más pequeño de los servidores.

En un servidor *nix

- | 1- Instalar git-core [en ubuntu: sudo apt-get install git]
- | 2- Crear un usuario git [sudo adduser git]
- | 3- Cambiar a usuario git [su git]
- | 4- Crear el repositorio del proyecto e inicializarlo:
| mkdir proyecto
| cd proyecto
| git init
- | 5- Ir al directorio donde se sirve contenido en apache2 es:
| cd /var/www/html
- | 6- Clonar un bare del repositorio de proyecto que hicimos en el paso 4
| git clone --bare /<ruta>/<al>/proyecto proyecto.git
| cd proyecto.git
- | 7- configurarlo y darle permisos:
| mv hooks/post-update.sample hooks/post-update
| chmod a+x hooks/post-update
- | 8- a partir de ahora tu proyecto está disponible para el mundo desde tu servidor y se lo puede clonar:
| git clone git@<ip_del_servidor>:/var/www/html/proyecto.git



Autenticación SSH en un repo de GitHub.



Buenas prácticas...

- | Estilos de commit.
- | Escribir un buen README.MD