# Iniciar un git

**Git init**

# Configurar un git

**git config --global user.name “Pepe”**

**git config --global user.email “pepe@cpp.com”**

# Comprobar conexión git

**ssh -T git@github.com**

**Git commit –m “initial commit of my project”**

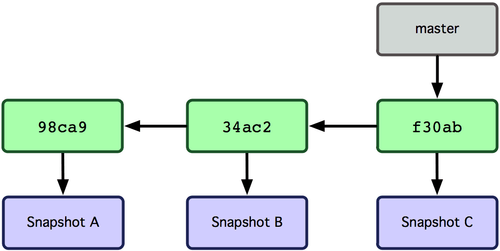
**No conecta con proxy, solución:**

**git config --global http.proxy** [**http://ptmproxy.gmv.es:80**](http://ptmproxy.gmv.es:80)

**git config --global https.proxy https://ptmproxy.gmv.es:80**

Commit --> confirmación de cambios

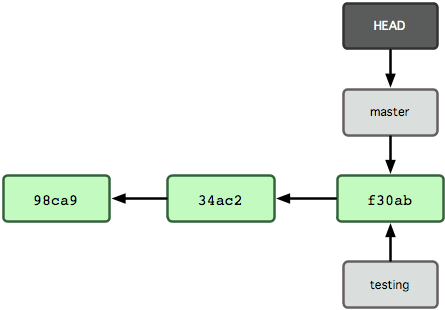
Branch -> Una rama Git es simplemente un apuntador móvil apuntando a una de esas confirmaciones. La rama por defecto de Git es la rama master. Con la primera confirmación de cambios que realicemos, se creará esta rama principal master apuntando a dicha confirmación. En cada confirmación de cambios que realicemos, la rama irá avanzando automáticamente. Y la rama master apuntará siempre a la última confirmación realizada.



Qué sucede cuando creas una nueva rama? Bueno..., simplemente se crea un nuevo apuntador para que lo puedas mover libremente. Por ejemplo, si quieres crear una nueva rama denominada "testing". Usarás el comando git branch:

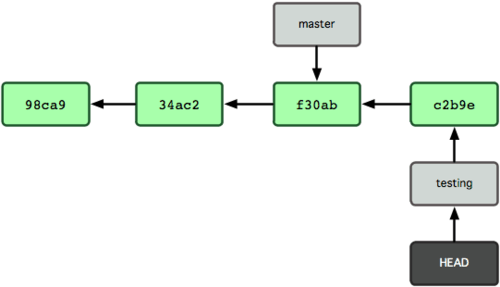
**$ git branch testing**

Esto crea un apuntador donde se está actualmente. Y, ¿cómo sabe Git en qué rama estás en este momento? Pues..., mediante un apuntador especial denominado HEAD. En Git, es simplemente el apuntador a la rama local en la que tú estés en ese momento.

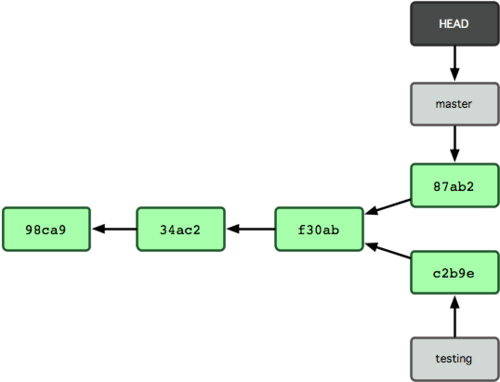


**$ git checkout testing**

Esto mueve el apuntador HEAD a la rama testing.



Si se crea una apuntando a testing y luego se cambia de branch a master y se crea un nuevo archivo el árbol se expandirá.



**$ git merge hotfix**

Mezclaría la rama master con hotfix.

**Git checkout - - hola.cpp** -> vuelve a la versión anterior del archivo.

**Git diff hola.cpp** -> ve los cambios hechos

**Git add hola.cpp** -> añade el cambio para hacer su commit, “add .” añade todos los cambios.

**Git commit –m “initial commit of my project”** -> mensaje commit

**Git commit –a –m “merge commit”** -> mensaje commit ->confirmación para "agregar" automáticamente los cambios de todos los archivos conocidos (es decir, todos los archivos que ya figuran en el índice) y para automatizar el borrado de los archivos del índice que se han eliminado del árbol de trabajo, y luego realizar el commit real

**Git stash** -> se usa cuando se necesite cambiar de rama durante un breve tiempo para ponerse a trabajar en otra. Toma el estado del espacio de trabajo, con todas las modificaciones en los archivos bajo control de cambios, y lo guarda en una pila provisional.

**Git stash list** -> se ve el contenido de la pila de guardado rápido.

**Git stash apply** -> se pueden volver a aplicar los últimos cambios recién guardados. Si no se indica ningún grupo concreto, Git asume que se desea reaplicar el grupo de cambios más reciente de entre los guardados en la pila. Si no, **git stash apply stash@{2}**(el que se quiera)

**Git stash drop** -> borra los archivos de la pila, hay que indicar el número de guardado.

**Git commit ->** abre el editor para escribir commit

**Git log** -> muestra el registro de cambios

**Git branch** -> muestra las ramas que hay y en cual estas

**Git branch rama2** -> crearía una rama 2

**Git checkout rama2** -> se mueve a la rama 2

Si se incluye un archivo .gitignore y se añade al git, lo que haya dentro lo ignorará git. Ejemplo proyecto/ main.cpp .gitignore test\_folder.

.gitignore -> si dentro del archivo se escribe: test\_folder, esta carpeta será ignorada.

# Volver a un git anterior

git checkout <id commit>