1. Fichero de configuración global

```
Windows:
 [directorio instalación git]/etc/gitconfig
 C:\ProgramData\Git\config %userprofile%\.gitconfig
Gnu/Linux:
  ~\.gitconfig
```

2. Configuración básica

```
git config --global user.name "Pepitolop34"
git config --global user.email \
    "Pepilop34@iesrodeira.gal"
```

- `--system` *aplica los cambios al fichero de configuración de sistema para todos los usuarios de la máquina.*
- `--local` *aplica los cambios solo para el repo donde se está trabajando (es la opción por defecto)*

7. Eliminar fichero(s)

*Si el fichero no está en la staging area; marca el fichero para eliminación en el siguiente **commit**: (se elimina del repo git y del sistema de archivos).*

```
git rm fichero
git rm log/\*.log     eliminar ficheros con extensión .log dentro del la carpeta log
git rm \*~            eliminar ficheros que terminan en ~
```

*Si el fichero ya está en la **staging area**:*

```
git rm fichero -f
```

*Marcar un fichero que estaba **tracked** como **untracked** (quitarlo de staging area y/o **working directory**)*

```
git rm --cached fichero   eliminar el fichero del repo sin eliminiarlo del sistema de archivos
```

10. Deshacer el último commit creando uno nuevo sin los cambios

Si el fichero no está en la staging

```
git revert HEAD
```

13. No hacer seguimiento de algún fichero o directorio

crear fichero `.gitignore` en carpeta

```
git add .gitignore
git commit -m "Añadir el .gitignore"
```

14. Moverse a otro commit en la misma u otra rama

```
git checkout hashlargo
git checkout hashcorto    g7 dígitos iniciales del hashlargo
git checkout HEAD

git reset --hard hashcorto
```

16. Rama: crear una, cambiar a otra, cambiar nombre

Crear una nueva rama (pero sin pasar a ella):

```
git branch nombreRamaNueva
```

Crear y pasar a una nueva rama

```
git switch -c nombreRama
git checkout -b nombreRama
```

Pasar a otra rama:

```
git switch nombreRama
  git checkout nombreRama
git merge main
```

Cambiar el nombre a una rama (ojo en caso de que haya un upstream)

```
git branch --move nombreRama nombreNuevoRama
```

17. Eliminar una rama

```
git branch -d nombreRama      Usar -D para forzar
```

26. Cherry-pick

Traer un fichero concreto (o varios) de una snapshot o commit concreto a la rama actual

```
git cherry-pick hashpequeñoCommit
git cherry-pick --continue
git cherry-pick --abort
```

*Cambiar el nombre por defecto de la rama principal ("**master**", ahora el nombre está "deprecado") cada vez que cramos un repo git:*

```
git config --global init.defaultBranch main
git config --system --unset init.defaultBranch
```

Configurar editor por defecto (opción larga y opción corta para VSC):

```
git config --global core.editor "\"C:\Users\pepe\AppData\Local\Programs\Microsoft VS Code\bin\code\" --wait"
git config --global core.editor "code --wait"
git config --global core.editor \
 "'C:/Program Files/Notepad++/notepad++.exe' \
  -multiInst -notabbar -nosession -noPlugin"
```

*Usar una cuenta git distinta a la global en un **repo** concreto:*

```
git config user.name "Almudena32"
git config user.email
"Almudena32@iesrodeira.gal"
```

Ver las opciones de configuración actuales y de dónde procede cada una:

```
git config --list --show-origin
git config --show-origin user.name
```

8. Mover fichero(s)

```
git mv ficheroOrigen ficheroDestino
```

Lo anterior equivale a:

```
mv ficheroOrigen ficheroDestino
git rm ficheroOrigen
git add ficheroDestino
```

11. Volver fichero a estado anteiror, el del commit justo anterior tras modificar ficheros

```
git checkout -- nombrefichero
git restore nombrefichero

git reset
```

Volver a un commit justo anterior, conservando staging area y working dir:

```
git reset --soft HEAD~1
```

Volver a un commit justo anterior, conservando solo working dir:

```
git reset --mixed HEAD~1
```

Volver a un commit justo anterior, eliminando staging area y working dir:

```
git reset --hard HEAD~1
```

15. Ver cambios con respecto al último commit

*Para ver qué se ha cambiado en directorio de trabajo (**working directory**) pero no **staged** (add): (es decir, comparar working directory con index)*

```
git diff
```

*Para ver qué ha cambiado de lo que está en **staged** (add) con respecto al útlmo commit:*

```
git diff --staged
git diff --cached
```

18. Ver cambios con respecto a otra rama

```
git diff nombreRamaB
git diff hashcortoA hashcortoB
```

19. Listar ramas existentes

```
git branch --all
```

Ver el último commit en cada rama:

```
git branch -v
```

Ver las ramas mergeadas o no mergeadas con la rama actual (HEAD):

```
git branch --merged
git branch --no-merged
```

24. Hacer un merge de ramas

```
git merge rama    ó   git merge rama --no-ff
git merge rama --abort
git switch nombreRamaA
```

Ver qué cambios aportará la ramaB (rama que vamos a hacer un merge en la main) a la rama `nombreRamaA`

```
git diff nombreRamaB
git merge nombreRamaB
```

Obtener ayuda (aparece en un navegador web):

```
git help comando
git help config
```

Obtener ayuda (en línea de comandos):

```
git add -h
git config -h
```

3. Inicio de seguimiento de versiones en un directorio

```
git init
git branch -m main

git status
git status -s
git status -uall   Mostrar también untracked dentro de carpetas
git log
```

4. Ver más detalles sobre el log y formateo del log

```
git log
```

*Ver un **patch output** (**diff** entre **commits**), 2 últimas entradas:*

```
git log -p -2
git log --pretty=format:"%h - %an, %ar : %s"
git log --stat hashcorto
git log --graph
git log --graph --pretty=oneline
git log --oneline --decorate
git log --graph --decorate --all --oneline

git log --since=2.weeks
git log -S function_name
git log nombreRama
git log -g    <-- tener en cuenta el reflog
```

Ver historial (log) completo de secuencia de movimientos por las ramas (incluso ramas que no aparecen en `git log`)

```
git reflog --no-abbrev
```

20. Etiquetas - tags

***Etiquetas ligeras** (Lightweight Tags):*
*Etiquetar un **checkpoint***

```
git tag nombreEtiqueta
```

Listar etiquetas creadas

```
git tag
git checkout tags/etiquetanuestra
```

Eliminar una etiqueta

```
git tag -d nombreEtiqueta
```

***Etiquetas anotadas** (Annotated Tags) (llevan más metadatos):*

```
git tag -a nombreEtiqueta -m "Mensaje anotación"
git show nombreEtiqueta
git tag -a nombreEtiqueta hashcortoCommit
```

23, Crear un stash o copia temporal privada

```
git stash      <-- solo guarda archivos tracked
git stash --all   <-- guarda también archivos untracked
git stash -u      <-- guarda también archivos untracked
git stash list
git stash list --all
git stash pop
git stash drop
```

25. Rebase *(ver gráfico al lado)*

Traer una rama a un punto contreto y modificar historial commits

```
git checkout ramaDeLaQueQueremosCogerCambios
git rebase ramaALaQueAplicarCambios
```

`git rebase master`

```
git rebase -i ramaDeLaQueQueremosCogerCambios
git rebase --continue ramaDeLaQueQueremosCogerCambios
git rebase --abort
```

Si queremos hacer fast-forward merge:

```
git checkout ramaALaQueAplicarCambios
git rebase ramaALaQueAplicarCambios
```

5. Añadir/quitar fichero(s)

```
git add fichero
git add .
```

NOTA: git no trackea directorios/carpetas vacías.
***Soluciones** [la b) para mantener el directorio vacío]:*
a) meter un fichero `.gitkeep`
b) meter un fichero `.gitignore` con contenido

```
*
!.gitignore
```

6. Hacer commit

se abre el editor por defecto y añadir comentario que identifique y guardar

```
git commit
git commit -m "añadir comentario que identifique"
```

2 en 1: hacer un add de todos los cambios ya tracked y un commit:

```
git commit -a -m "comentario que identifique"
```

Ver el hash del commit padre o antecesores de un commit dado:

```
git rev-parse hash^
git rev-parse hash^N      ver el commit antecesor N
git rev-parse hash^@      ver todos los commit antecesores
```

Quitar de la staging area un fichero:

```
git restore --staged fichero
```

Listar el contenido de la staging area (=index):

```
git ls-files --stage
```

Listar información de los ficheros:

```
git ls-files --debug
```

9. Crear alias para comandos largos

```
git config --global alias.tree "log --graph \
  --decorate --all --oneline"
git config --global alias.unstage 'reset HEAD --'

git config --global alias.lol "log --graph
 --pretty=format:'%Cred%h%Creset \
   -%C(yellow)%d%Creset %s %Cgreen(%cr) \
    %C(bold blue)<%an>%Creset' --abbrev-commit"
```

12. Deshacer ciertos cambios

*Cambiar el mensaje del último **commit** (si no ha habido cambios y no haber hecho un **push**):*

```
git commit --amend
```

*Añadir ficheros olvidados al crear el último commit (evitar si se hizo un **push**):*

```
git commit -m 'Commit con mis olvidos'
git add ficheros_olvidados
git commit --amend
```

*Quitar fichero de **staged area** antes de commit:*

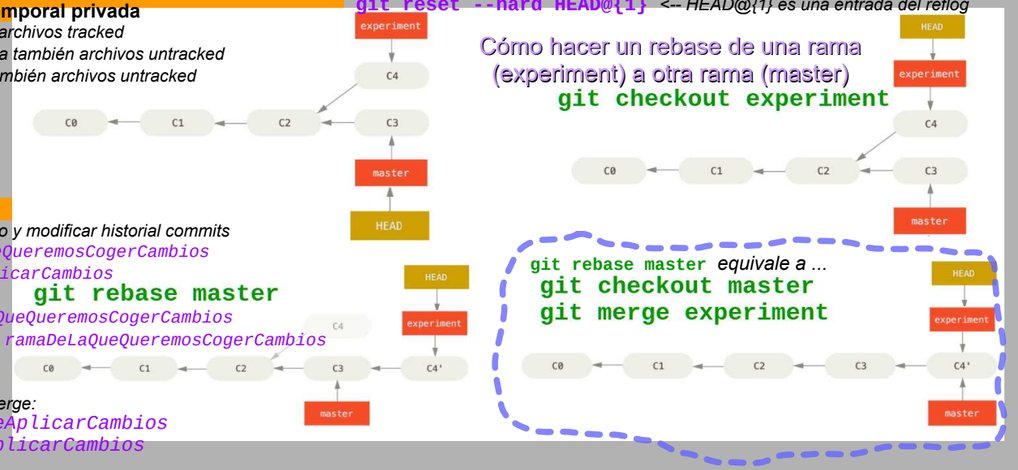
```
git reset HEAD fichero
git restore --staged fichero
```

21. Volver a un commit de una rama "eliminando" los commits intermedios entre el actual y el commit al que queremos volver de la misma rama

```
git reset --hard hashcortoCommitAlQueQueremosVolver
```

22. Volver a un commit "eliminado"

```
git reset --hard hashcortoCommitAlQueQueremosVolver
git reset --hard HEAD@{1}  <-- HEAD@{1} es una entrada del reflog
```



Subir un repo local a Github (GH) y descargar repos

El nombre por defecto del repo remoto es `origin`*; se puede cambiar:*

```
git remote add origin \
    git@github.com:usuariogithub/proyecto
git remote add nombreRepo \      <- pueden existir varios
    https://github.com/usuariogithub2/proyecto.git
```

Listar los repo remotos usados:

```
git remote
git remote -v      (ver URLs)
git branch -vv     (ver tracking branches)
git remote show
git ls-remote
```

Eliminar un repo remoto:

```
git remote remove nombreRepo
```

Renombrar un repo remoto:

```
git remote rename nombreRepo  nuevoNombreRepo
```

La primera vez debemos especificar el nombre de la rama en GitHub

```
git push -u origin nombreRamaCorrespondienteGH
git push   <--Después ya podemos hacer esto
```

Ver información del repo remoto:

```
git remote show origin
```

Acceder al log remoto de una rama:

```
git log origin/master
git log refs/remotes/origin/master
```

Descargar al repo local la info de GH (hacer un sync)

Descargar solo historial de cambios y demás datos ***sin aplicar los cambios a la versión local (sin hacer un merge)****:*

```
git fetch
git fetch nombreRepo
git fetch --all   Descargar de todos los repos remotos
git fetch origin \
   master:refs/remotes/origin/mymaster
```

Descargar historial de cambios y demás datos y ***aplicar los cambios a la versión local (hace un merge)****:*

```
git pull
git pull origin nombreRamaCorrespondienteGH
git pull origin HEAD
```

Si hay conflictos, configuramos el mecanismo por defecto a ***merge****:*

```
git config pull.rebase false
git merge origin nombreRamaCorrespondienteGH
git merge origin/rama \
        --allow-unrelated-histories
git pull
```

Tocamos algo en el código, después de hacer un add y commit, para subirlo al repo remoto:

```
git push origin master
git push
git push origin --tags
git push repoRemoto nombreRamaLocal
git push repoRemoto  \
   nombreRamaLocal:nombreRamaRemota
```

Configurar el mecanismo por defecto de resolución de conflictos a merge a nivel global:

```
git config --global pull.rebase "false"
```

Configurar una tracking branch

Normalmente el nombreRamaA = nombreRamaB, pero no imprescindible:

```
git checkout -b nombreRamaA origin/nombreRamaB
   git checkout --track origin/nombreRama
```

Si la rama nombreRama no existe en local, la copia del orginin:

```
   git checkout nombreRama
```

Clonar un repositorio cualquiera a local

```
git clone git@github.com:usuariogithub/nombreproyecto
```

Dar un nombre distinto al nombre por defecto `origin` *al remote:*

```
git clone -o empresa \
   git@github.com:usuariogithub/nombreproyecto
```

Subir etiquetas al repositorio

El comando git push no transfiere las etiquetas a repos remotos; si queremos que se transfieran:

```
git push origin nombreEtiqueta
git push origin --tags
```

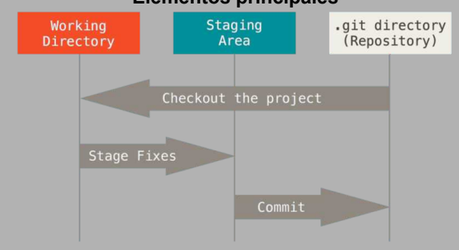
Eliminar etiquetas del repositorio

```
git push origin --delete nombreEtiqueta
git push origin :refs/tags/nombreEtiqueta
```

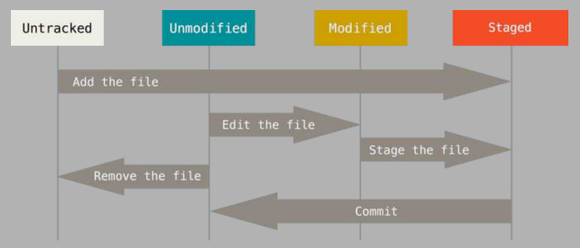
Cambiar el nombre de una rama local y remotamente

```
git branch --move nombreAntiguo nombreNuevo
git push --set-upstream origin nombreNuevo
git branch --all
git push origin --delete nombreAntiguo
```

Elementos principales

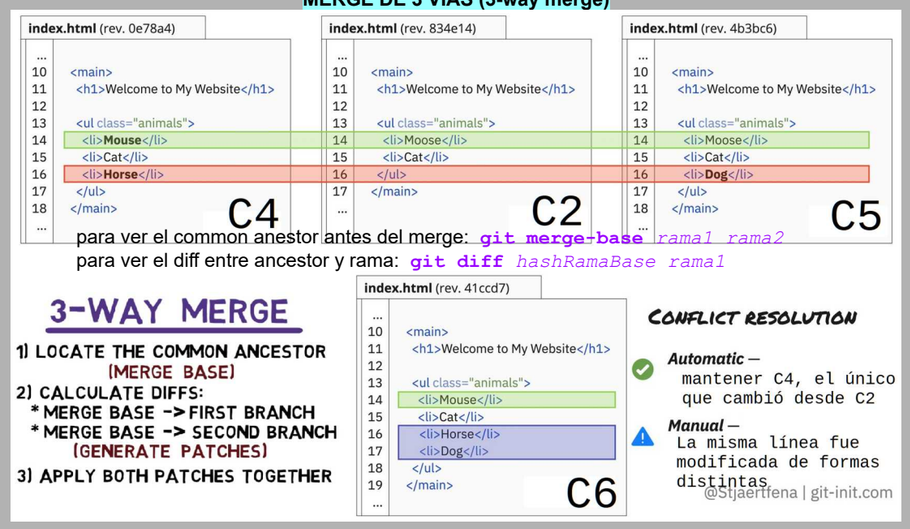


CICLO DE VIDA DE UN FICHERO



Glosario				
repository/repo	branch	untracked file	detached HEAD state	commit
working tree	local branch	tracked file	conflict	stage
working directory	remote branch	unmodified file	production branch	branch
staging area/index	remote-traking branch	modified file	hotfix	merge
checkout	tracking branch	ignored file	workflow	rebase
snapshot	upstream branch	staged file	author	fast-forward
commit	HEAD	unstaged file	commiter	clone
commit history	tag	merge-conflicted file	tagger	stash
merge commit	lightweight tag		checksum/hash	pull
unreachable commit	annotated tag	alias	commit hash	fetch
commit amending	3-way merge	merge base	resolve	push
	fast-forward merge	common ancestor	diverge	track files

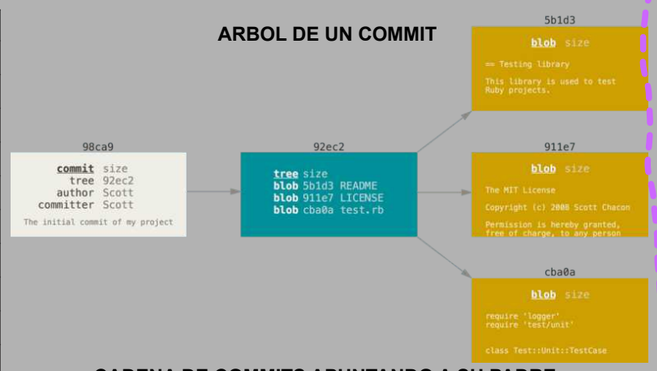
MERGE DE 3 VÍAS (3-way merge)



para ver el common anestor antes del merge: `git merge-base rama1 rama2`

para ver el diff entre ancestor y rama: `git diff hashRamaBase rama1`

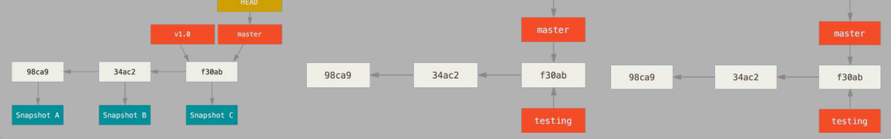
ARBOL DE UN COMMIT



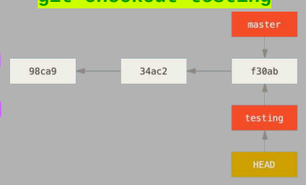
CADENA DE COMMITS APUNTANDO A SU PADRE

Ejemplos de como funcionan HEAD, las ramas y los commit

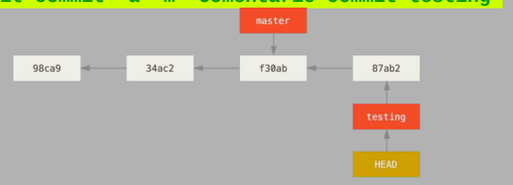
RAMA E HISTORIAL DE COMMITs — **HEAD apunta a la rama actual** — `git branch testing`



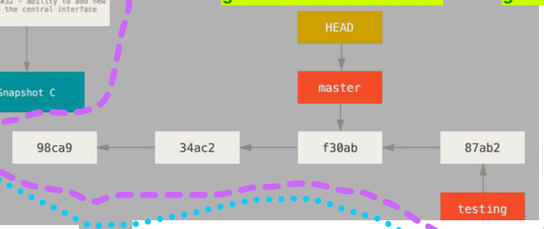
`git checkout testing`



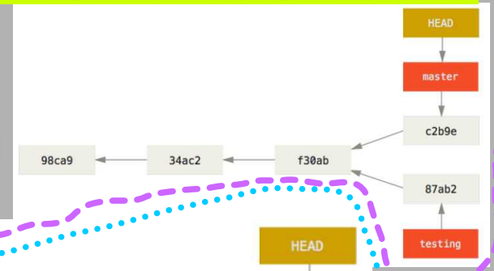
`git commit -a -m 'Comentario commit testing'`



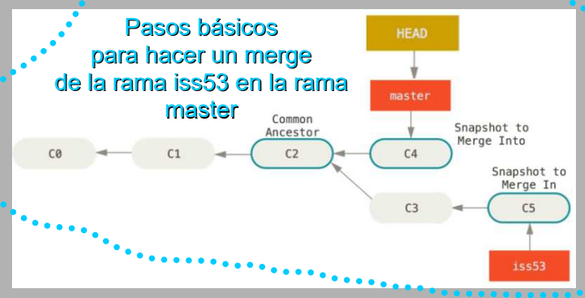
`git checkout master`



`git commit -a -m 'Comentario commit master'`



`git checkout master`



`git merge iss53`