

Practica Git & GitHub

Este documento describe la práctica del módulo de “git, github y sourcetree” del Bootcamp de IA.

A continuación se describen los ejercicios propuestos y como hemos resuelto cada uno de ellos:

EJERCICIO 1

Se deberá crear un repositorio y realizar una serie de operaciones **desde la consola de comandos** sobre el mismo para posteriormente subir el repositorio a Github.

Se deberá entregar a través del formulario de prácticas indicando la URL del repositorio. En el repositorio, deberá existir un archivo readme.md con las respuestas a las siguientes preguntas:

- ¿Qué comando utilizaste en el paso 11? ¿Por qué?
- ¿Qué comando o comandos utilizaste en el paso 12? ¿Por qué?
- El merge del paso 13, ¿Causó algún conflicto? ¿Por qué?
- El merge del paso 19, ¿Causó algún conflicto? ¿Por qué?
- El merge del paso 21, ¿Causó algún conflicto? ¿Por qué?
- ¿Qué comando o comandos utilizaste en el paso 25?
- El merge del paso 26, ¿Podría ser fast forward? ¿Por qué?
- ¿Qué comando o comandos utilizaste en el paso 27?
- ¿Qué comando o comandos utilizaste en el paso 28?
- ¿Qué comando o comandos utilizaste en el paso 29?
- ¿Qué comando o comandos utilizaste en el paso 30?
- ¿Qué comando o comandos usaste en el paso 32?
- ¿Qué comando o comandos usaste en el punto 33?

Los pasos a ejecutar son los siguientes (los pasos en negrita indican que hay una pregunta asociada):

1. Crear un repositorio en GitHub y clónalo en tu equipo

Hacemos login en <https://github.com> con nuestra cuenta, si no tenemos cuenta la creamos. En mi caso, usare la cuenta que tengo: dcanmen

The first screenshot shows the GitHub homepage with the URL 'github.com' in the address bar and a red box around it. The top navigation bar includes 'Product', 'Solutions', 'Resources', 'Open Source', 'Enterprise', and 'Pricing'. The 'Sign in' button in the top right corner is also highlighted with a red box.

The second screenshot shows the 'Sign in to GitHub' page. It features a form for 'Username or email address' and 'Password', with links for 'Forgot password?' and 'Sign in'. Below the form are links for 'Sign in with a passkey', 'New to GitHub?', and 'Create an account'. The 'Sign in' button is highlighted with a red box.

The third screenshot shows the user's GitHub dashboard under the 'dcanmen' profile. The dashboard includes sections for 'Top repositories', 'Ask Copilot', and 'Home'. A 'New' button is visible in the top right of the repository list. A modal window for 'GitHub Copilot' is open in the center-right of the screen, with its 'Open Copilot' button highlighted with a red box.

Creamos un repositorio nuevo vacío, podemos inicializarlo con el fichero README.md para facilitar la documentación acerca de este repositorio.

En el menú superior, pulsamos sobre el desplegable con un “+” y pulsamos en “new repository”

This screenshot shows the same GitHub dashboard as the previous one. In the top right corner, there is a vertical menu with a '+' icon. The 'New repository' option is highlighted with a red box.

Y aparecerá un menú para configurar el repositorio.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Required fields are marked with an asterisk (*).

Owner * **Repository name ***

dcanmen / **github**   

github is available.

Great repository names are short and memorable. Need inspiration? How about [potential-eureka](#) ?

Description (optional)

Public Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file
This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore

.gitignore template: **None** 

Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license

License: **None** 

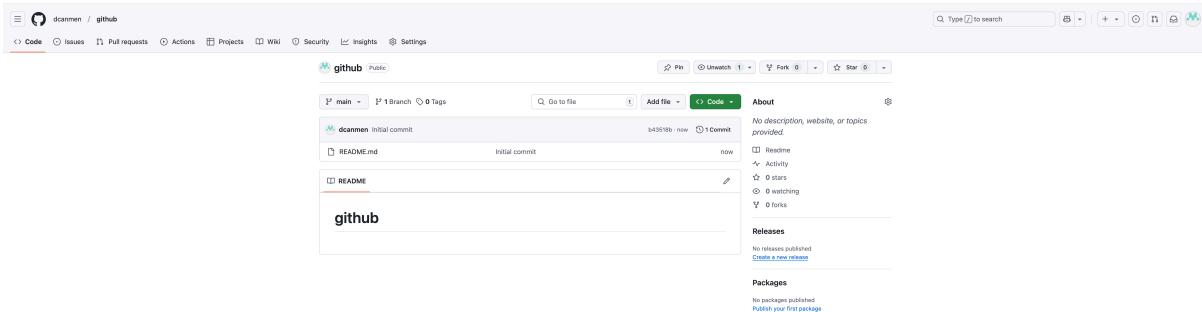
A license tells others what they can and can't do with your code. [Learn more about licenses](#).

This will set **main** as the default branch. Change the default name in your [settings](#).

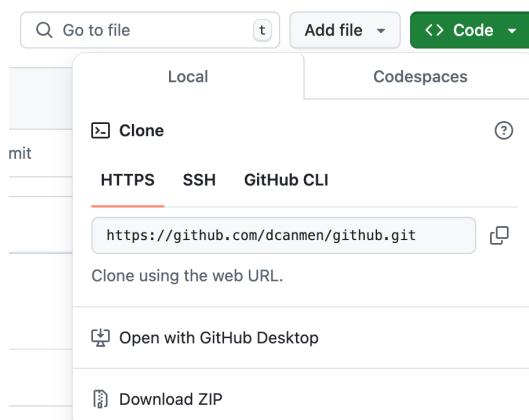
ⓘ You are creating a public repository in your personal account.

Create repository

Pulsamos el botón “Create repository” y ya tendremos listo nuestro repositorio de github para comenzar a trabajar.



Para poder clonar el repositorio en nuestro ordenador y trabajar en local, necesitamos copiar la URL del repositorio primero. Para ello, pulsamos el botón “Code” y aparece un pequeño menú con algunas opciones e información.



Copiamos la URL que aparece en la pestaña “Local” y en la sección “HTTPS”.

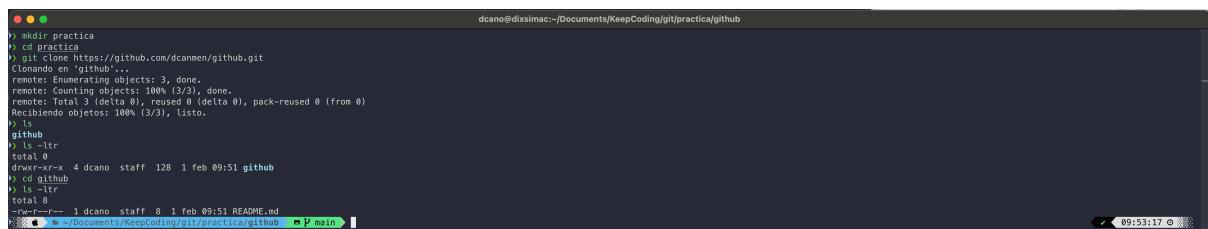
Ahora nos vamos a nuestra terminal en el ordenador y ejecutamos el comando de clonado de repositorio, en la carpeta que queramos utilizar para almacenar el repositorio en mi caso (~/Documents/KeepCoding/git/practica). (NOTA: Se presupone que tenemos instalado el cliente de git en nuestro ordenador y que sabemos manejar la consola de comandos de nuestro ordenador).

```
$ git clone https://github.com/dcanmen/github.git
```



A terminal window showing the command \$ git clone https://github.com/dcanmen/github.git being run. The output shows the cloning process, including object enumeration and receiving objects. The final directory listing shows a new folder named 'github' containing a file named 'README.md'.

En la imagen siguiente se puede ver que nos ha creado una carpeta con el nombre de nuestro repositorio y que dentro está nuestro fichero README.md. (NOTA: En mi caso tengo configurado la consola de comandos de mi mac para que me identifique en que rama del repositorio estoy trabajando)



A terminal window showing the command \$ git clone https://github.com/dcanmen/github.git being run again. The output shows the cloning process. The final directory listing shows a new folder named 'github' containing a file named 'README.md'. The current working directory is shown as ~/Documents/KeepCoding/git/practica/github/main.

2. Crear un archivo git-nuestro.md con el contenido:

```
Git nuestro  
Git nuestro que estas en los repos  
Comprimidos sean tus commits  
Venga a nosotros tu log  
En el local como en el remote  
Danos hoy nuestro pull de cada dia  
Perdona nuestros conflictos  
Como tambien perdonamos los de otros geeks  
No nos dejes caer en detached HEAD  
y libranos de SVN  
git commit --amend
```

En nuestro terminal de comandos, ejecutamos el siguiente comando:

```
$ nano git-nuestro.md
```

Se abre el editor nano, copiamos el texto anterior y guardamos el fichero.

```
> nano git-nuestro.md
> cat -ltr
total 16
-rw-r--r-- 1 dcano staff 8 1 feb 09:51 README.md
-rw-r--r-- 1 dcano staff 311 1 feb 09:59 git-nuestro.md
> cat git-nuestro.md
Git-nuestro
Git nuestro que estas en los repos
Comprinidos seis tus commits
Venga a nosotros tu los
En el mundo de los remotes
Damos hoy nuestro pull de cada dia
Perdona nuestros conflictos
Como tambien perdonamos los de otros geeks
No nos dejes caer en detached HEAD
Y no te pierdas SNAKE
git commit -am "d"
> main ?!
```

Si queremos saber el estado de lo que estamos haciendo en nuestro repositorio, podemos ejecutar el comando:

\$ git status

```
> git status
En la rama main
Tu rama esta actualizada con 'origin/main'.
Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que sera confirmado)
    git-nuestro.md
no hay nada agregado al commit pero hay archivos sin seguimiento presentes (usa "git add" para hacerles seguimiento)
> main ?!
```

Esto nos indica que tenemos un fichero llamado “git-nuestro.md” en nuestra área de “**working**”.

3. Añadir git-nuestro.md al staging área

Para mover el archivo “git-nuestro.md” del área “**working**” al área “**staging**”, tenemos que ejecutar el comando:

\$ git add git-nuestro.md

```
> git add git-nuestro.md
> git status
En la rama main
Tu rama esta actualizada con 'origin/main'.
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del area de stage)
    nuevos archivos: git-nuestro.md
> main +!
```

Como vemos, se ha movido al área de staging y ha cambiado de color a verde, lo que indica que está listo para ser subido al repositorio en la rama actual (main).

4. Mover lo que hay en el staging area al repositorio

El comando que tenemos que utilizar para mover el archivo “git-nuestro.md” del staging área al área de repositorio es el siguiente:

\$ git commit -m <comentario o indicaciones de que estamos subiendo>

```
> git commit -m "Initial commit"
[main 0d71cb7] Initial commit
  1 file changed, 11 insertions(+)
  create mode 100644 git-nuestro.md
> git status
En la rama main
Tu rama esta adelantada a 'origin/main' por 1 commit.
  (usa "git push" para publicar tus commits locales)
nada para hacer commit, el arbol de trabajo està limpio
> main +!
```

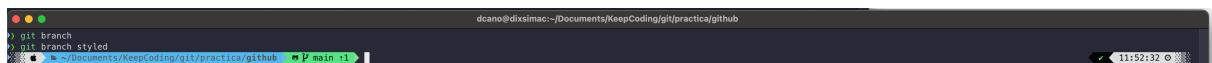
Utilizamos el comando (git status) para comprobar el estado de los cambios realizados y en qué estado están.

NOTA: Podemos ver que el comando status nos indica que estamos 1 cambio por encima de nuestro repositorio de github.com y que podemos sincronizarlo usando el comando “git push”. Es decir, nuestro repositorio local está más actualizado que nuestro repositorio remoto.

5. Crear una rama llamada “styled”

Para crear una nueva rama usamos el comando:

```
$ git branch styled
```



A terminal window titled "git branch". The command \$ git branch styled is entered. The output shows the new branch "styled" has been created. The terminal is located at dcano@dixsimac:~/Documents/KeepCoding/git/practica/github. The status bar shows 11:52:32 O.

6. Listar las ramas que hay en el repositorio

El comando que debemos utilizar para listar las ramas existentes:

```
$ git branch
```



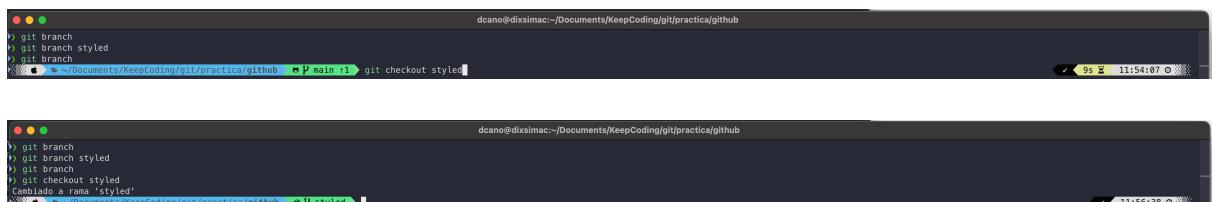
Two terminal windows. The top one shows the command \$ git branch and lists "main" and "styled". The bottom one shows the command \$ git branch again, with "styled" highlighted in green, indicating it is the active branch. Both terminals are at dcano@dixsimac:~/Documents/KeepCoding/git/practica/github. The status bar shows 11:54:07 O.

Y como podemos observar nuestra rama (branch) activa es “main” y hay una nueva rama llamada “styled”

7. Moverse a la rama “styled”

Para movernos a la rama “styled” desde “main” debemos ejecutar el comando:

```
$ git checkout styled
```



Two terminal windows. The top one shows the command \$ git checkout styled and the message "Cambiado a rama 'styled'". The bottom one shows the command \$ git branch again, with "styled" highlighted in green, indicating it is now the active branch. Both terminals are at dcano@dixsimac:~/Documents/KeepCoding/git/practica/github. The status bar shows 11:56:38 O.

Nos muestra un mensaje que nos dice que hemos cambiado a la rama “styled”
(NOTA: En mi terminal se muestra en que rama de git estoy trabajando, esto es una configuración especial de mi terminal y que me permite entender en todo momento donde me encuentro)

8. Comprobar que se está en la rama correcta

Para verificar que se está en la rama adecuada usamos el comando:

```
$ git branch
```



```
dcano@diximac:~/Documents/KeepCoding/git/practica/github
* git branch
* git branch styled
* git branch
* git checkout styled
Cambiado a rama 'styled'
dcano@diximac:~/Documents/KeepCoding/git/practica/github ~ p styled git branch
git branch
```

```
git branch
* main
* styled
[END]
```

Podemos ver que nos marca la rama activa con el asterisco y un color diferente.

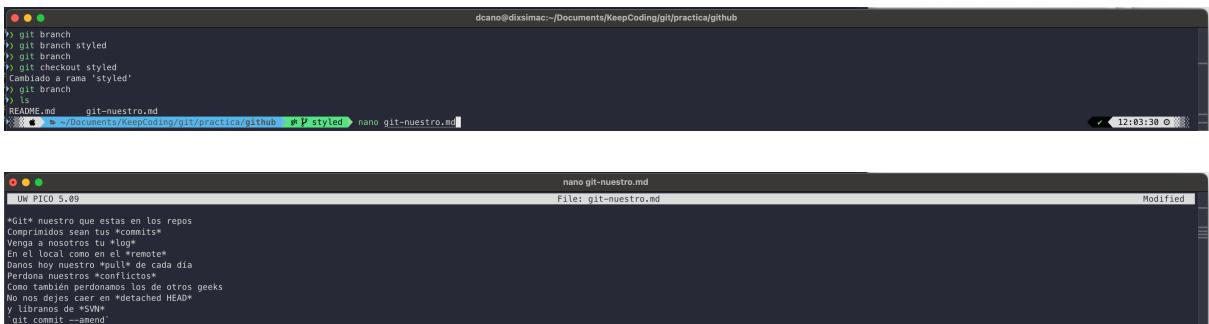
9. Modificar en el archivo git-nuestro.md

Modificamos el archivo git-nuestro.md con el contenido que aparece a continuación:

```
*Git* nuestro que estás en los repos
Comprimidos sean tus *commits*
Venga a nosotros tu *log*
En el local como en el *remote*
Danos hoy nuestro *pull* de cada día
Perdona nuestros *conflictos*
Como también perdonamos los de otros geeks
No nos dejes caer en *detached HEAD*
y libráanos de *SVN*
`git commit --amend`
```

Para ello, editamos el fichero con el editor nano, ejecutando el siguiente comando:

```
$ nano git-nuestro.md
```



```
dcano@diximac:~/Documents/KeepCoding/git/practica/github
* git branch
* git branch styled
* git checkout styled
Cambiado a rama 'styled'
* git branch
* git branch
RENAME: git-nuestro.md
dcano@diximac:~/Documents/KeepCoding/git/practica/github ~ p styled nano git-nuestro.md
12:03:30
```

```
nano git-nuestro.md
File: git-nuestro.md
Modified

*Git* nuestro que estás en los repos
Comprimidos sean tus *commits*
Venga a nosotros tu *log*
En el local como en el *remote*
Danos hoy nuestro *pull* de cada día
Perdona nuestros *conflictos*
Como también perdonamos los de otros geeks
No nos dejes caer en *detached HEAD*
y libráanos de *SVN*
`git commit --amend`
```

Y mostramos el fichero modificado.

```
* cat git-nuestro.md
*Git* nuestro que estás en los repos
Comprimidos sean tus *commits*
Venga a nosotros tu *log*
En el local como en el *remote*
Danos hoy nuestro *pull* de cada día
Perdona nuestros *conflictos*
Como también perdonamos los de otros geeks
No nos dejes caer en *detached HEAD*
y libráanos de *SVN*
`git commit --amend`
```

10. Añadir los cambios al staging area y luego pasarlos al repositorio

Para ello ejecutamos los siguientes comandos:

```
$ git add git-nuestro.md
```

```
> git add git-nuestro.md
> git status
En la rama styled
  Cambios a ser confirmados:
    (usa "git add -A <archivo...>" para sacar del área de stage)
      modificados: git-nuestro.md
```

12:18:37

```
$ git commit -m "Nueva version de git-nuestro.md"
```

```
> git commit -m "New version of git-nuestro.md"
[styled 0b27c6e] New version of git-nuestro.md
  1 file changed, 9 insertions(+), 10 deletions(-)
> git status
En la rama styled
  nada para hacer commit, el árbol de trabajo está limpio
```

12:12:29

Después de cada comando podemos ver cómo está el estado de nuestro repositorio para comprobar que todo se ha ejecutado correctamente.

11. Deshacer el último commit (perdiendo los cambios realizados en el working copy)

Si utilizamos el comando “git reflog” podemos identificar todos los commits realizados en nuestro repositorio.

```
> git reflog
> ~Documents/KeepCoding/git/practica/github > p styled !1
```

git reflog

18s 10:43:53

Para deshacer el último commit, debemos usar el comando (git reset):

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

Pero utilizando los atributos mostrados en el comando anterior, posicionamos HEAD en el padre de nuestro actual commit y volvemos al commit anterior deshaciendo los cambios realizados a partir de ese commit padre, lo que significa que perdemos todo nuestro trabajo en el working copy a partir de dicho commit padre.

```
> git reset --hard HEAD~1
HEAD está ahora en 0d71cb7 Initial commit
> ~Documents/KeepCoding/git/practica/github > p styled
```

10:52:40

```
commit 0d71cb7de177198b968d183ff71a3706c37493 (HEAD -> styled, main)
Author: Diego Cane <diego.cano@gmail.com>
Date:  Sat Feb 1 10:12:41 2025 +1000

  Initial commit

commit b4351b84f437cc1a16347ad9c992cd399754 (origin/main, origin/HEAD)
Author: dcammen <3843232864dcammen@users.noreply.github.com>
Date:  Sat Feb 1 09:38:30 2025 +1000

  Initial commit

(END)
```

21s 10:53:20

Si usamos los comandos “git log” y “git reflog” podemos ver el estado actual de nuestro repositorio y todos los comandos ejecutados.

12. Rehacer el último commit (el que acabamos de deshacer)

Si utilizamos el comando “git reflog” podemos ver toda la historia de los commits y resets realizados en el repositorio. Para ello, ejecutamos el comando y buscamos cual es el identificador del punto al que queremos volver. En nuestro caso es justamente el anterior.

```
git reflog
```



```
0b71cbf (HEAD -> styled, main) HEAD@{0}: reset: moving to HEAD@-1
0b71cbf (main) HEAD@{1}: commit: New version of git-nuestro.md
0b71cbf (HEAD -> styled, main) HEAD@{2}: checkout: moving from main to styled
0b71cbf (HEAD -> styled, main) HEAD@{3}: commit: Initial commit
b435180 (origin/main, origin/HEAD) HEAD@{4}: clone: from https://github.com/dcammen/github.git
(END)
```

Ejecutamos el comando:

```
$ git reset --hard <id>
```

```
git reset --hard b027c6e
```



```
HEAD esta ahora en b027c6e New version of git-nuestro.md
(END)
```

```
git reflog
```



```
0b71cbf (HEAD -> styled) HEAD@{0}: reset: moving to b027c6e
0b71cbf (main) HEAD@{1}: reset: moving to HEAD@-1
0b27c6e (HEAD -> styled) HEAD@{2}: commit: New version of git-nuestro.md
0b71cbf (main) HEAD@{3}: checkout: moving from main to styled
0b71cbf (main) HEAD@{4}: commit: Initial commit
b435180 (origin/main, origin/HEAD) HEAD@{5}: clone: from https://github.com/dcammen/github.git
(END)
```

13. Hacer un merge con ‘main’ (styled absorbe a main)

Como queremos absorber la rama “main” por la rama “styled”, debemos confirmar que la rama activa es “styled”, para ello ejecutamos el comando:

```
$ git branch
```

```
git branch
```



```
main
* styled
(END)
```

Vemos que tenemos 2 ramas y que la activa es “styled”

Ejecutamos el comando merge:

```
$ git merge main
```

```
git merge main
```



```
la rama actualizada.
(END)
```

14. Volver a la rama main

Para volver a otra rama existente, ejecutamos el comando:

```
$ git checkout <nOMBRE_rama>
```

```
git checkout main
```



```
Cambiando a rama 'main'.
(Fuente de datos actualizada a 'origin/main' por 1 commit.
  Usa "git push" para publicar tus commits locales)
(END)
```

Y podemos comprobar que estamos en la rama main, con el comando:

```
$ git branch
```

```
git branch
```



```
* main
  styled
(END)
```

15. Crear una nueva rama llamada “htmlify”

Para crear una nueva rama usamos el comando:

```
$ git branch htmlify
```



Comprobamos que tenemos las 3 ramas.



16. Cambiar a la rama htmlify

Para movernos a la rama “htmlify” debemos ejecutar el comando:

```
$ git checkout htmlify
```



Confirmamos que estamos en htmlify con el comando “git branch”

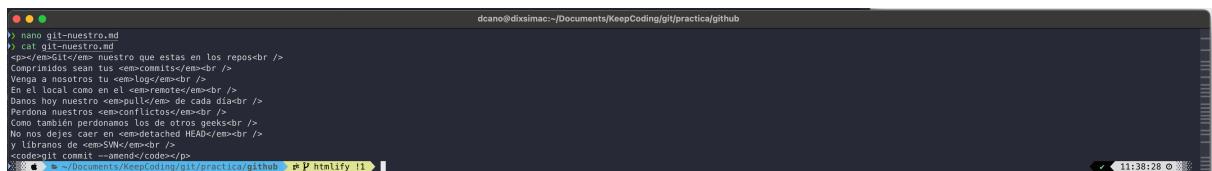
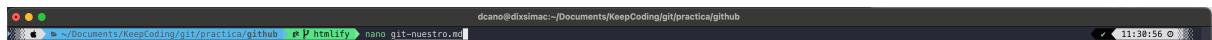


17. Modificar en el archivo git-nuestro.md

Modificamos el archivo git-nuestro.md como se muestra a continuación:

```
<p><em>Git</em> nuestro que estas en los repos<br />
Comprimidos sean tus <em>commits</em><br />
Venga a nosotros tu <em>log</em><br />
En el local como en el <em>remote</em><br />
Danos hoy nuestro <em>pull</em> de cada dia<br />
Perdona nuestros <em>conflictos</em><br />
Como tambien perdonamos los de otros geeks<br />
No nos dejes caer en <em>detached HEAD</em><br />
y libranos de <em>SVN</em><br />
<code>git commit --amend</code></p>
```

Para ello usamos el editor “nano” y guardamos los cambios.



18. Hacer un commit

Para hacer un commit, primero pasamos el fichero git-nuestro.md al área de staging y posteriormente hacemos el commit dentro de la rama htmlify.

```
git add git-nuestro.md
git status
En la rama htmlify
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    modificados:   git-nuestro.md

git commit -m "Cambios en git-nuestro dentro rama htmlify"
[htmlify Ideabot] Cambios en git-nuestro dentro rama htmlify
  1 file changed, 10 insertions(+), 11 deletions(-)
  git status
En la rama htmlify
nada para hacer commit, el árbol de trabajo está limpio
$ cd ~/Documents/KeepCoding/git/practicas/github & git htmlify
```

19. Hacer un merge de “htmlify” en “styled” (styled absorbe a htmlify)

Primero cambiamos a la rama “styled” y la hacemos la rama activa, y luego ejecutamos el comando “merge”.

```
! git checkout styled
  Cambiado a rama 'styled'
! git branch
* styled
  main
  htmlify
(END)
```

Ahora ejecutamos el comando:

```
$ git merge htmlify
```

```
git merge-himself
Auto-fusionando git-nuestro.md
COMFLICTO (contenido): Conflicto de fusión en git-nuestro.md
Fusión automática falló. Resuelve los conflictos y luego realiza un commit con el resultado.
Y...  
git add -u  
git commit -m "Resolví los conflictos"
```

Vemos que tenemos conflicto en el fichero git-nuestro.md, porque el fichero ha sido modificado en ambas ramas con contenido diferente en las mismas líneas.

20. Si hay conflictos, deberemos resolverlos quedándonos con el contenido de la rama “styled”.

Como hemos visto anteriormente existen conflictos. Para resolverlo tenemos que editar el fichero git-nuestro.md y en nuestro caso nos quedamos con el contenido que proviene de la rama styled.

```
MacBook-Pro:~/Documents/DeepCoding/git/practico/github/p4styled$ nano git-nuestro.md
```

```
nano git-nuestro.md
File: git-nuestro.md

UW PICO 5.89

>-----< HEAD
Dile nostro que estas en los repos
Comprimidos sean tus *commits*
Venga a nosotros tu *log*
En el local como en el <remote>
Damos hoy nuestro <em>pull</em> de cada dia
Permita nuestros <em>conflictos</em>
Como tambien perdonamos los de otros geeks
No nos dejes caer en *detached HEAD*
y libranos de <em>svN</em>.
git commit --amend.
```

```
<-----< emGit/>
<-----< emGit/nuestro que estas en los repos;br />
<-----< emGit/comprimidos sean tus commits;br />
Venga a nosotros tu <em>log</em>br />
En el local como en el <em>remote</em>br />
Damos hoy nuestro <em>pull</em> de cada dia;br />
Permita nuestros <em>conflictos</em>br />
Como tambien perdonamos los de otros geeks;br />
No nos dejes caer en <em>detached HEAD</em>br />
y libranos de <em>svN</em>br />
<code>git commit --amend</code>
```

```
git cat git-nuestro.md
Comunidad que estás en los repos
(Comprando, sellando "commits"
Venga a nosotros tu *log*
En el local como en el *remote*
Danos hoy nuestro *pull* de cada día
Perdona nuestros *conflictos*
Como también perdonamos los de otros geeks
No nos dejes caer en *detached HEAD*
y libranos de *SWN*.
git commit --amend
```

Movemos el fichero final al area de staging y hacemos commit, de esta forma se resuelve el conflicto.

```
> git add git-nuestro.md
En la rama styled
  Cambios no staged para el commit:
    (usa "git add <archivos...>" para actualizar lo que será confirmado)
    (usa "git restore <archivos...>" para descartar los cambios en el directorio de trabajo)
      modificados: README.md
sin cambios agregados al commit (usa "git add" y/o "git commit -a")
> nano git-nuestro.md
> ls -al
total 16
drwxr-xr-x  3 dcano  staff  100  2 feb 11:48 .
drwxr-xr-x  5 dcano  staff  160  2 feb 12:15 ..
drwxr-xr-x 14 dcano  staff  448  2 feb 17:52 .git
-rw-r--r--@  1 dcano  staff  2743  2 feb 11:55 README.md
-rw-r--r--@  1 dcano  staff  318  2 feb 17:48 git-nuestro.md
> git commit
[styled bb2aaabb] resolvemos el conflicto de union de las ramas htmlify y styled.
  1 file changed, 78 insertions(+), 1 deletion(-)
> git status
En la rama styled
  nada para hacer commit, el árbol de trabajo está limpio
  ➜  ~ /Documents/KeepCoding/git/practica/github  ⌘ P styled 17:55:05
```

21. Desde “main”, hacer un merge con “styled”

Primero cambiamos a la rama “main” y desde esta hacemos un “merge” con “styled” ejecutando los siguientes comandos:

```
$ git checkout main
```

```
$ git merge styled
```

```
> git checkout main
Combiendo rama main
Tu rama está adelantada a 'origin/main' por 1 commit,
  (usa "git push" para publicar tus commits locales)
> git merge styled
Actualizando 0d71cb7..bb2aaabb
Fast forward
  README | 79 ++++++-----+
  git-nuestro.md | 28 ++++++-----
  2 files changed, 88 insertions(+), 11 deletions(-)
> git status
En la rama main
Tu rama está adelantada a 'origin/main' por 5 commits.
  (usa "git push" para publicar tus commits locales)
nada para hacer commit, el árbol de trabajo está limpio
  ➜  ~ /Documents/KeepCoding/git/practica/github  ⌘ P main 17:58:53
```

22. Crear una rama “title” y cambiarse a esa rama

Para crear una rama nueva ejecutamos el comando:

```
$ git branch title
```

Y para cambiarnos

```
$ git checkout title
```

```
> git branch title
> git checkout title
Cambiado a rama title
  ➜  ~ /Documents/KeepCoding/git/practica/github  ⌘ P title 18:06:25
```

23. Añadir un título (a tu gusto) al archivo git-nuestro.md y hacer un commit.

Para añadir un titulo al fichero git-nuestro.md, lo editamos con nano y lo guardamos.

```
> nano git-nuestro.md
nano git-nuestro.md
File: git-nuestro.md
Modified
#Git Nuestro
#GIS nuestro que estás en los repos
Comprisiones tean tus *commits*
Venga a nosotros tu *log*
En el local como en el *remote*
Dame hoy nuestro *pull* de cada dia
Pero no nuestros *rebase*
Como tu queremos los de otros geeks
No nos dejes caer en *detached HEAD*
y librarnos de *SVN*
*git commit --amend*
```

```
> nano git-nuestro.md
> cat git-nuestro.md
`#Git Nuestro
`#git nuestro que estás en los repos
Comprimido cada vez *commits*
Venga a nosotros tu *log*
En el local como en el *remote*
Damos hoy nuestro *pull* de cada día
Permita nuestros *conflictos*
Como la piedra rompiendo los de otros geeks
No nos dejéis caer en *detached HEAD*
y librando --*SIN*
`git commit --amend`
```

Añadimos el fichero modificado al staging área y realizamos un commit.

```
$ git add git-nuestro.md
$ git commit -m "Añadimos un título al fichero git-nuestro.md en la rama title"
```

```
> git status
En la rama title
Cambiados no confirmados para el commit:
  (usa "git add <archivo>..." para actualizar lo que será confirmado)
  (usa "git restore <archivo>..." para descartar los cambios en el directorio de trabajo)
    modificados:   git-nuestro.md
sin cambios agregados al commit (usa "git add" y/o "git commit -a")
> git add git-nuestro.md
> git status
En la rama title
Cambiados a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    modificados:   git-nuestro.md
> git commit -m "Añadimos un título al fichero git-nuestro.md en la rama title"
[el log se muestra]
  1 file changed, 2 insertions(+)
> git status
En la rama title
nada más hacer commit, el árbol de trabajo está limpio
```

24. Volver a la rama main

Para volver a la rama main ejecutamos el comando:

```
$ git checkout main
```

Y comprobamos que la rama activa es main con el comando:

```
$ git branch
```

```
> git checkout main
Cambiado a rama 'main'
Tu rama actual está adelantada a 'origin/main' por 5 commits.
  (usa "git push" para publicar tus commits locales)
> git branch
  .htmlify
* main
  styled
  title
  [ONG]
```

25. Dibujar el diagrama

Vamos a usar el comando

```
$ git log --graph
```

Para dibujar el diagrama de las ramas y commits del repositorio

```

  ↵ [~/Documents/KeenCoding/git/practica/github] P main :5 git log --graph
* commit bb2a08667f20b55971a524ea0d9aa0c8fb8663 (HEAD -> main, styled)
| Author: Diego Cano <diego.cano@gmail.com>
| Date:  Sun Feb 2 17:54:12 2025 +0100
|   resolvemos el conflicto de union de las ramas htmlify y styled.
|
| * commit b8e1fc15c8d02a1125317eb654a70b2e938
| | Merge: 8b27c6a1dea80
| | Author: Diego Cano <diego.cano@gmail.com>
| | Date:  Sun Feb 2 17:58:59 2025 +0100
| |   Arreglamos los conflictos de hacer el merge de la rama htmlify, quedandonos con la version de styled
|
| * commit 1de8a8e8d5da75ad289d6ded59a7876e141ef04 (htmlify)
| | Author: Diego Cano <diego.cano@gmail.com>
| | Date:  Sun Feb 2 11:48:27 2025 +0100
| |   Cambios en git-nuestro dentro rama htmlify
|
| * commit bb27c6a15c8d02a1125317eb654a70b2e938
| | Merge: 8b27c6a1dea80
| | Author: Diego Cano <diego.cano@gmail.com>
| | Date:  Sat Feb 1 12:12:20 2025 +0100
| |   New version of git-nuestro.md
|
| * commit 8d71cb7d4c1719a8c96d837fb71a3706c37493
| | Author: Diego Cano <diego.cano@gmail.com>
| | Date:  Sat Feb 1 10:12:41 2025 +0100
| |   Initial commit
|
| * commit b43518bb4f437ced1eb6347ad9c092b5cd399754 (origin/main, origin/HEAD)
| | Author: dcannen <38432586@dcannenusers.noreply.github.com>
| | Date:  Sat Feb 1 09:30:30 2025 +0100
| |   Initial commit
|
(End)

```

Podemos utilizar unos modificadores de este comando que nos ayudan a visualizar de forma más efectiva este gráfico. Para ello usamos el siguiente comando:

```
$ git log --graph --decorate --pretty=oneline
```

```

  ↵ [~/Documents/KeenCoding/git/practica/github] P main :5 git log --graph --decorate --pretty=oneline
* bb2a08667f20b55971a524ea0d9aa0c8fb8663 (HEAD -> main, styled) resolvemos el conflicto de union de las ramas htmlify y styled.
| b8e1fc15c8d02a1125317eb654a70b2e938 Arreglamos los conflictos de hacer el merge de la rama htmlify, quedandonos con la version de styled
| * 1de8a8e8d5da75ad289d6ded59a7876e141ef04 (htmlify) Cambios en git-nuestro dentro rama htmlify
| * 8b27c6a1dea8052fffe0940a510836c4913bf19e6f47 New version of git-nuestro.md
|
| * 8d71cb7d4c1719a8c96d837fb71a3706c37493 Initial commit
|
| * b43518bb4f437ced1eb6347ad9c092b5cd399754 (origin/main, origin/HEAD) Initial commit
|
(End)

```

26. Hacer un merge “no fast-forward” de “title” en “main” (main absorbe a title)

Para hacer un merge “no fast-forward” de la rama “title” en la rama “main” ejecutaremos el siguiente comando:

```
$ git merge --no-ff title
```

```

  ↵ [~/Documents/KeenCoding/git/practica/github] P main :7 git merge --no-ff title
Merge made by the 'ort' strategy.
git-nuestro.md | 2 ++
 1 file changed, 2 insertions(+)
|+
En la rama main
Tu rama está adelantada a 'origin/main' por 7 commits.
(usa "git push" para publicar tus commits locales)

nada para hacer commit, el árbol de trabajo está limpio
`_ git log
commit dd343c3ecfcf3697e6330ad2b0b10481506611523 (HEAD -> main)
Merge: bb2a08667f20b55971a524ea0d9aa0c8fb8663
Author: Diego Cano <diego.cano@gmail.com>
Date:  Sun Feb 2 18:36:16 2025 +0100

  Merge branch 'title'

commit be0256d455c0b273a3140f002ad2f5b2ddfb47 (title)
Author: Diego Cano <diego.cano@gmail.com>
Date:  Sun Feb 2 18:12:42 2025 +0100

  Añadimos un titulo al fichero git-nuestro.md en la rama title

commit bb2a08667f20b55971a524ea0d9aa0c8fb8663 (styled)
Author: Diego Cano <diego.cano@gmail.com>
Date:  Sun Feb 2 17:54:12 2025 +0100

  resolvemos el conflicto de union de las ramas htmlify y styled.

commit b8e1fc15c8d02a1125317eb654a70b2e938
Merge: 8b27c6a1dea8052fffe0940a510836c4913bf19e6f47
Author: Diego Cano <diego.cano@gmail.com>
Date:  Sun Feb 2 17:59:59 2025 +0100

  Arreglamos los conflictos de hacer el merge de la rama htmlify, quedandonos con la version de styled

commit 1de8a8e8d5da75ad289d6ded59a7876e141ef04 (htmlify)
Author: Diego Cano <diego.cano@gmail.com>
Date:  Sun Feb 2 11:48:27 2025 +0100

  Cambios en git-nuestro dentro rama htmlify

commit 8b27c6a1dea8052fffe0940a510836c4913bf19e6f47
Author: Diego Cano <diego.cano@gmail.com>
Date:  Sat Feb 1 12:12:20 2025 +0100

  New version of git-nuestro.md

commit 8d71cb7d4c1719a8c96d837fb71a3706c37493
Author: Diego Cano <diego.cano@gmail.com>
Date:  Sat Feb 1 10:12:41 2025 +0100

  Initial commit

commit b43518bb4f437ced1eb6347ad9c092b5cd399754 (origin/main, origin/HEAD)
Author: dcannen <38432586@dcannenusers.noreply.github.com>
Date:  Sat Feb 1 09:30:30 2025 +0100

```

Lo que acabamos de hacer es desde la rama main, se ha creado un commit nuevo que absorbe el trabajo realizado en la rama title.

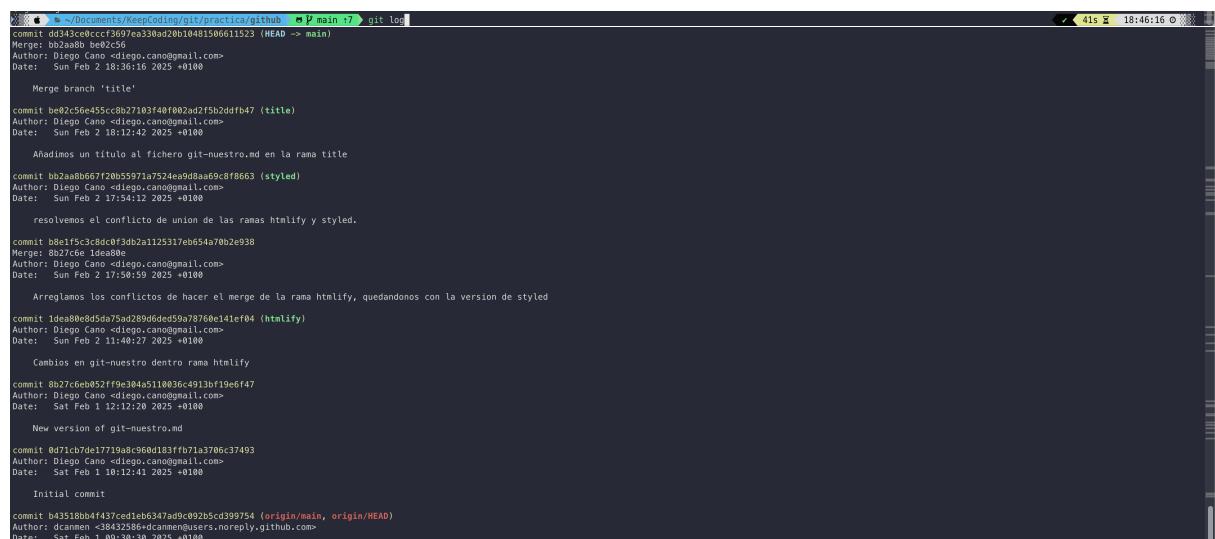
27. Deshacer el merge (sin perder los cambios del working copy)

Para poder deshacer el merge actual sin perder los cambios en el working copy ejecutamos el comando:

```
$ git reset HEAD~1
```

Este comando mueve el head a una posición anterior, que es justamente el último commit antes de realizar el merge.

Revisamos el log para ver posteriormente que hemos hecho con el reset.



```
git log
commit 0d71cb7de1779abc60d183ff71a3796c37493
Author: Diego Cano <diego.cano@gmail.com>
Date:   Sat Feb 1 10:12:41 2025 +0100

    Initial commit

commit b43518b04f437ced1eb6347ad9c092b5cd399754 (origin/main, origin/HEAD)
Author: dcanmen <38432586-dcanmen@users.noreply.github.com>
Date:   Sat Feb 1 09:38:13B 2025 +0100

    Cambios en git-nuestro dentro de rama htmlify

Commit 8b276eb0852f9e384511803634913bf19e6f47
Author: Diego Cano <diego.cano@gmail.com>
Date:   Sat Feb 1 12:12:20 2025 +0100

        Arreglamos los conflictos de hacer el merge de la rama htmlify, quedandonos con la version de styled

commit 1dea0ed5da5ad280ddded59a7876be141ef04 (htmlify)
Author: Diego Cano <diego.cano@gmail.com>
Date:   Sun Feb 2 17:54:12 2025 +0100

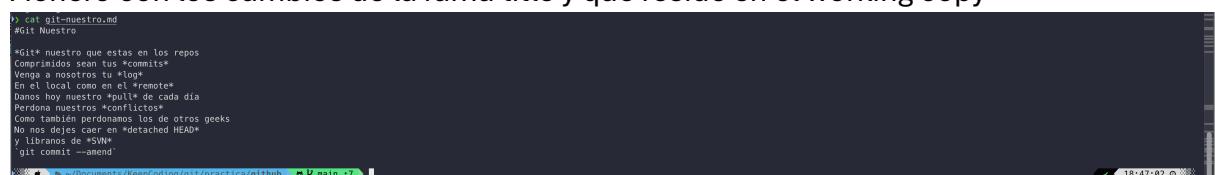
    resolvemos el conflicto de union de las ramas htmlify y styled.

commit bb2a280d7f0305971a7234e0ed5da59c8f8663 (styled)
Author: Diego Cano <diego.cano@gmail.com>
Date:   Sun Feb 2 18:17:42 2025 +0100

    Añadimos un tifnero al fichero git-nuestro.md en la rama title

commit bb2a280d7f0305971a7234e0ed5da59c8f8663 (styled)
Author: Diego Cano <diego.cano@gmail.com>
Date:   Sun Feb 2 18:17:42 2025 +0100
```

Fichero con los cambios de la rama title y que reside en el working copy



```
# cat git-nuestro.md
#Git Nuestro

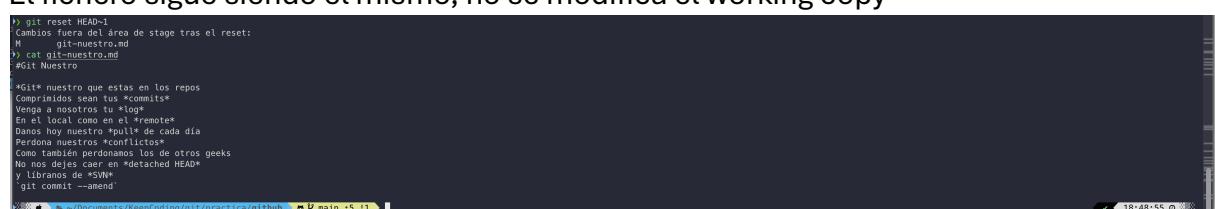
*Git* nuestro que estas en los repos
Comprimidos sean tus *commits*
Venga a nosotros tu *log*
En el canal como tu *remote*
Danos hoy nuestro *pull* de cada dia
Perdona nuestros *conflictos*
Como tambien perdonamos los de otros geeks
No nos dejes caer en *detached HEAD*
y libranos de *SNW*
git commit --amend
```

Ejecutamos el comando reset



```
$ git reset HEAD~1
Cambios fuera del área de stage tras el reset:
M     git-nuestro.md
```

El fichero sigue siendo el mismo, no se modifica el working copy



```
$ git reset HEAD~1
Cambios fuera del área de stage tras el reset:
M     git-nuestro.md
$ cat git-nuestro.md
#Git Nuestro

*Git* nuestro que estas en los repos
Comprimidos sean tus *commits*
Venga a nosotros tu *log*
En el canal como tu *remote*
Danos hoy nuestro *pull* de cada dia
Perdona nuestros *conflictos*
Como tambien perdonamos los de otros geeks
No nos dejes caer en *detached HEAD*
y libranos de *SNW*
git commit --amend
```

Revisamos el log y vemos que no están los commits de la rama title.

```

commit bb2a68d097f20b559715324ea9dd5ad9c8f8f8663 (HEAD -> main, styled)
Author: Diego Cano <diego.cano@gmail.com>
Date:  Sun Feb 2 17:54:18 2023 +0100

    resolvemos el conflicto de union de las ramas htmlify y styled.

commit b8e1f5c3cbdrf3eb2a1125317eb654a70b2e938
Merge: 8b27f6e 1dea@0
Author: Diego Cano <diego.cano@gmail.com>
Date:  Sun Feb 2 17:58:59 2023 +0100

    Arreglamos los conflictos de hacer el merge de la rama htmlify, quedandnos con la version de styled

commit idea80e8d5da75ad280ddded59a7676ae141ef04 (htmlify)
Author: Diego Cano <diego.cano@gmail.com>
Date:  Sat Feb 2 11:40:27 2023 +0100

    Cambios en git-nuestro dentro ram htmlify

commit 8b27f6e0953ff9e30a5110826e4913bf13e6f47
Author: Diego Cano <diego.cano@gmail.com>
Date:  Sat Feb 1 12:12:26 2023 +0100

    New version of git-nuestro.md

commit 0d71c7d0e17719ab9c96d6183fffb71a3706c37493
Author: Diego Cano <diego.cano@gmail.com>
Date:  Sat Feb 1 10:12:41 2023 +0100

    Initial commit

commit b43518bb4f437ced1eb6347ad9c092b5cd399754 (origin/main, origin/HEAD)
Author: dcammen <384325286-dcammenusers.noreply.github.com>
Date:  Sat Feb 1 09:36:30 2023 +0100

    Initial commit

[END]

```

28. Descartar los cambios

Para entender el comando que debemos ejecutar, primero ejecutamos el comando:

\$ git status

Este nos dice que ficheros están modificados y en qué área (working o staging)

```

$ git status
# On branch main
# Your branch is ahead of 'origin/main' by 5 commits.
#   (use "git push" to publish your commits locally)
#
# Changes not staged for commit:
#   (use "git add ..." to stage these changes)
#     modified:   git-nuestro.md
#
# no changes added to commit (use "git add" and/or "git commit -a")

```

Como vemos el fichero modificado está en el área working copy, para poder descartar los cambios del fichero usamos el comando:

\$ git restore git-nuestro.md

Pero vamos a realizar un cat del fichero antes y después del comando para comparar los resultados.

```

$ cat git-nuestro.md
`git Nuestro

`git nuestro que estas en los repos
Comprimidos sean tus `commits`
`Venga a nosotros tu `log
`En el local como en el *remote*
`Damos hoy nuestro *pull* de cada dia
`Perdonos nuestros *conflictos*
`Como familia perdonamos los de otros geeks
`No nos dejes caer en *detached HEAD*
`y librarnos de *SN*
`git commit --amend

`$ git restore git-nuestro.md
`$ cat git-nuestro.md
`$git nuestro que estas en los repos
Comprimidos sean tus `commits*
`Venga a nosotros tu `log
`En el local como en el *remote*
`Damos hoy nuestro *pull* de cada dia
`Perdonos nuestros *conflictos*
`Como familia perdonamos los de otros geeks
`No nos dejes caer en *detached HEAD*
`y librarnos de *SN*
`git commit --amend`


```

Como vemos el título añadido al fichero git-nuestro en la rama title desaparece de la copia local (working copy) y lo reestablece con la versión del commit HEAD de la rama main actual.

29. Eliminar la rama “title”

Para eliminar una rama tenemos que estar en otra rama diferente, y que esta tenga visibilidad de la misma para poder borrarla. El comando para borrar una rama es:

```
$ git branch -D title
```

Antes podemos comprobar las ramas que tenemos y cual es la activa con el comando “git branch” y una vez ejecutado el comando hacer lo mismo.



```
git branch
```

Output:
* main
 styled
 title
 [END]

git branch -D title
Borrada la rama title (era be02c56).
git branch
* main
 styled
 [END]

30. Rehacer el merge que hemos deshecho

Para rehacer un merge que hemos deshecho utilizaremos el comando:

```
$ git reflog
```



```
git reflog
```

Output:
bb2aa0b HEAD -> main, styled HEAD@{0}: reset: moving to HEAD-1
dd343ce HEAD@{1}: merge title: Merge made by the 'ort' strategy.
bb2aa0b HEAD@{2}: checkout: moving from title to main
bb2aa0b HEAD@{3}: commit: Cambios en git-nuestro dentro de la rama title
bb2aa0b HEAD -> main, styled HEAD@{4}: checkout: moving main to title
bb2aa0b HEAD -> main, styled HEAD@{5}: merge styled: Fast-forward
0d71cb HEAD@{6}: checkout: moving from styled to main
bb2aa0b HEAD -> main, styled HEAD@{7}: commit: resolvemos el conflicto de union de las ramas htmlify y styled.
bb2aa0b HEAD@{8}: checkout: moving from styled a los resultados de hacer el merge de la rama htmlify, quedandnos con la version de styled
bb27dc HEAD@{9}: checkout: moving from htmlify to styled
0de80e [htmlify] HEAD@{10}: commit: Cambios en git-nuestro dentro rama htmlify
0d71cb HEAD@{11}: checkout: moving from main to htmlify
0d71cb HEAD@{12}: checkout: moving from styled to main
0d71cb HEAD@{13}: merge styled: Fast-forward
0d71cb HEAD@{14}: reset: moving to HEAD-1
bb27dc HEAD@{15}: commit: New version of git-nuestro.md
0d71cb HEAD@{16}: checkout: moving from main to styled
0d71cb HEAD@{17}: commit: Initial Commit
ba3519b (origin/main, origin/HEAD) HEAD@{18}: clone: from https://github.com/dcanmen/github.git
[END]

Con el que obtenemos los id's de los commits realizados dentro del repositorio con toda la historia. Solo tenemos que identificar cual es el id que tenemos que utilizar para ejecutar el comando:

```
$ git reset --hard <id>
```

En nuestro caso tenemos que utilizar el segundo id: dd343ce



```
cat git-nuestro.md
```

Git nuestro que estas en los repos
Comprimidos seis tus *commits*
Venga a nosotros tu *log*
En el local como en el *remote*
Danos hoy nuestro *pull* de cada dia
Perdona nuestros *conflictos*
Como tambien perdonamos los de otros geeks
No nos dejes caer en *detached HEAD*
y libramos de *SVN*
`git commit --amend`

```
git reset --hard dd343ce  
HEAD esté ahora en dd343ce Merge branch 'title'  
cat git-nuestro.md  
#Git Nuestro

*Git* nuestro que estas en los repos  
Comprimidos seis tus *commits*  
Venga a nosotros tu *log*  
En el local como en el *remote*  
Danos hoy nuestro *pull* de cada dia  
Perdona nuestros *conflictos*  
Como tambien perdonamos los de otros geeks  
No nos dejes caer en *detached HEAD*  
y libramos de *SVN*  
`git commit --amend`


```

Hemos comprobado que el fichero git-nuestro.md ahora tiene el título.

Vemos que pinta tiene ahora el historio de acciones:

```

git reflog
d0d41c (HEAD main) HEAD@{0}: reset: moving to dd343ce
bb2aa8b (styled) HEAD@{1}: reset: moving to HEAD@{0}
dd343ce (HEAD -> main) HEAD@{2}: merge: title: Merge made by the 'ort' strategy.
bb2aa8b (styled) HEAD@{3}: checkout: moving from title to main
be02c56 HEAD@{4}: commit: Anadimos un título al fichero git-nuestro.md en la rama title
bb2aa8b (styled) HEAD@{5}: checkout: moving from main to title
bb2aa8b (styled) HEAD@{6}: merge: Fast-Forward
0d71cb7 HEAD@{7}: checkout: moving from styled to main
bb2aa8b (styled) HEAD@{8}: commit: resolvemos el conflicto de union de las ramas htmlify y styled.
b8e1f5c HEAD@{9}: commit (merge): Arreglamos los conflictos de hacer el merge de la rama htmlify, quedandonos con la version de styled
0d71cb7 HEAD@{10}: checkout: moving from main to styled
1dea80e (htmlify) HEAD@{11}: commit: Cambios en git-nuestro dentro rama htmlify
0d71cb7 HEAD@{12}: checkout: moving from main to htmlify
0d71cb7 HEAD@{13}: checkout: moving from styled to main
0d71cb7 HEAD@{14}: reset: moving to 0d71cb7
0d71cb7 HEAD@{15}: checkout: moving to HEAD@{0}
bb27f6e HEAD@{16}: commit: New version of git-nuestro.md
0d71cb7 HEAD@{17}: checkout: moving from main to styled
0d71cb7 HEAD@{18}: commit: Initial commit
b43510b (origin/main, origin/HEAD) HEAD@{19}: clone: from https://github.com/dcammen/github.git
[END]

```

Vemos que el HEAD de la rama main, apunta al merge que hicimos anteriormente.

31. Volver a main y eliminar el resto de ramas

Para borrar el resto de ramas, tenemos que conocer cuáles son y para ello ejecutamos el comando:

```
$ git branch
```

```

git branch
* main
* styled
* htmlify
[END]

```

Ahora ejecutamos el comando para borrar las ramas:

```
$ git branch -D htmlify
$ git branch -D styled
```

```

git branch -D htmlify
Eliminada la rama htmlify (era 1dea80e).
git branch -D styled
Eliminada la rama styled (era bb2aa8b).
[END]

```

32. Volver al commit inicial cuando se creó el poema

Para poder volver al commit inicial necesitamos conocer el <id> del commit cuando se creó el poema, para ello ejecutamos el comando:

```
$ git reflog
```

```

git reflog
d0d41c (HEAD main) HEAD@{0}: reset: moving to dd343ce
bb2aa8b HEAD@{1}: reset: moving to HEAD@{0}
dd343ce (HEAD -> main) HEAD@{2}: merge: title: Merge made by the 'ort' strategy.
bb2aa8b (styled) HEAD@{3}: checkout: moving from title to main
be02c56 HEAD@{4}: commit: Anadimos un título al fichero git-nuestro.md en la rama title
bb2aa8b (styled) HEAD@{5}: checkout: moving from main to title
bb2aa8b (styled) HEAD@{6}: merge: Fast-Forward
0d71cb7 HEAD@{7}: checkout: moving from styled to main
bb2aa8b (styled) HEAD@{8}: commit: resolvemos el conflicto de union de las ramas htmlify y styled.
b8e1f5c HEAD@{9}: commit (merge): Arreglamos los conflictos de hacer el merge de la rama htmlify, quedandonos con la version de styled
0d71cb7 HEAD@{10}: checkout: moving from main to styled
1dea80e (htmlify) HEAD@{11}: commit: Cambios en git-nuestro dentro rama htmlify
0d71cb7 HEAD@{12}: checkout: moving from main to htmlify
0d71cb7 HEAD@{13}: checkout: moving from styled to main
0d71cb7 HEAD@{14}: reset: moving to 0d71cb7
0d71cb7 HEAD@{15}: checkout: moving to HEAD@{0}
bb27f6e HEAD@{16}: commit: New version of git-nuestro.md
0d71cb7 HEAD@{17}: checkout: moving from main to styled
0d71cb7 HEAD@{18}: commit: Initial commit
b43510b (origin/main, origin/HEAD) HEAD@{19}: clone: from https://github.com/dcammen/github.git
[END]

```

Entonces ejecutamos el comando, con el id que hemos identificado:

```
$ git reset --hard <id>
```

```

git reset --hard 0d71cb7
HEAD está ahora en 0d71cb7 Initial commit
[END]

```

Podemos comprobar el fichero git-nuestro.md en el working copy y ver si tiene título y los tags especiales añadidos en styled.

```
1) cat git-nuestro.md
Git nuestro
Git nuestro que estas en los repos
Comprimidos sean tus commits
Venga a nosotros tu log
En el local como en el remote
Danos hoy nuestro pull de cada dia
Perdona nuestros conflictos
Como tambien perdonamos los de otros geeks
No nos dejes caer en detached HEAD
y librarnos de SWN
git commit --amend
```

Y podemos ver que la versión que tenemos es la inicial.

33. Volver al estado final, cuando pusimos título al poema

Utilizamos el comando “git reflog” para identificar el commit al que queremos volver.

```
git reflog
bd71cb7 (HEAD -> main) HEAD@{0}: reset: moving to bd71cb7
dd343ce HEAD@{1}: reset: moving to dd343ce
bd71cb7 (HEAD -> main) HEAD@{2}: reset: moving to HEAD@{1}
bd312c9 HEAD@{3}: merge title! Merge made by the 'ort' strategy.
bd2a8ab HEAD@{4}: checkout: moving from title to main
be02c56 HEAD@{5}: commit: Añadimos un título al fichero git-nuestro.md en la rama title
bd2a8ab HEAD@{6}: checkout: moving from main to title
bd2a8ab HEAD@{7}: checkout: moving from title to main
bd17c97 (HEAD -> main) HEAD@{8}: checkout: moving from styled to main
bd2a8ab HEAD@{9}: commit: resolvemos el conflicto de unión de las ramas htmlify y styled.
bd8ef5c HEAD@{10}: commit (merge): Arreglamos los conflictos de hacer el merge de la rama htmlify con la versión de styled
bd27c6e HEAD@{11}: checkout: moving from htmlify to styled
bd27c6e HEAD@{12}: checkout: moving from styled to main
bd27c6e HEAD@{13}: checkout: moving from main to htmlify
bd71cb7 (HEAD -> main) HEAD@{13}: checkout: moving from main to htmlify
bd71cb7 (HEAD -> main) HEAD@{14}: checkout: moving from styled to main
bd27c6e HEAD@{15}: reset: moving to bd27c6e
bd27c6e HEAD@{16}: reset: moving to HEAD@{1}
bd27c6e HEAD@{17}: commit: New version of git-nuestro.md
bd71cb7 (HEAD -> main) HEAD@{18}: checkout: moving from main to styled
bd71cb7 (HEAD -> main) HEAD@{19}: commit: Initial commit
ba35180 (origin/main, origin/HEAD) HEAD@{20}: clone: from https://github.com/dcamnen/github.git
[END]
```

Ahora ejecutamos el comando:

```
$ git reset --hard <id>
```

Y comprobamos que estamos en el commit cuando añadimos el título al poema comprobando el fichero git-nuestro.md de nuestro working copy.

```
* cat git-nuestro.md
Git nuestro
Git nuestro que estas en los repos
Comprimidos sean tus commits
Venga a nosotros tu log
En el local como en el remote
Danos hoy nuestro pull de cada dia
Perdona nuestros conflictos
Como tambien perdonamos los de otros geeks
No nos dejes caer en detached HEAD
y librarnos de SWN
git commit --amend
? git reset --hard dd343ce
HEAD está ahora en dd343ce Merge branch 'title'
? cat git-nuestro.md
#Git Nuestro

*Git* nuestro que estas en los repos
Comprimidos sean tus *commits*
Venga a nosotros tu *log*
En el local como en el *remote*
Danos hoy nuestro *pull* de cada dia
Perdona nuestros *conflictos*
Como tambien perdonamos los de otros geeks
No nos dejes caer en *detached HEAD*
y librarnos de *S*
```

34. Crear los siguientes tags:

inicial: en el primer commit

styled: modificación del paso 10

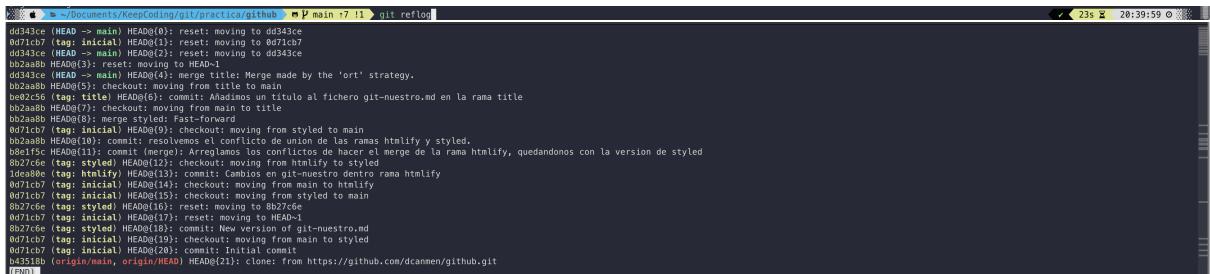
htmlify: modificación del paso 17-18

title: modificación del paso 30

Para crear los tags, vamos a identificar los commits que corresponden a estos pasos (git reflog) y usaremos el siguiente comando:

```
$ git tag <nombre_tag> <id_commit>
```

Una vez ejecutados todos los comandos para todos los tags, podemos comprobar como se han creado los tags.



```
git reflog
```

dd343ce HEAD@{main} HEAD@{0}: reset: moving to dd343ce
dd343ce HEAD@{main} HEAD@{1}: merge: moving to dd343ce
dd343ce (HEAD -> main) HEAD@{2}: reset: moving to dd343ce
bb2aa8b HEAD@{3}: reset: moving to HEAD-1
dd343ce (HEAD -> main) HEAD@{4}: merge title: Merge made by the 'ort' strategy.
bb2aa8b HEAD@{5}: checkout: moving to main
bb2aa8b HEAD@{6}: commit: Añadimos el título al fichero git-nuestro.md en la rama title
bb2aa8b HEAD@{7}: checkout: moving from main to title
bb2aa8b HEAD@{8}: merge styled: Fast-forward
0d71cb7 (tag: inicial) HEAD@{9}: checkout: moving from styled to main
0d71cb7 (tag: inicial) HEAD@{10}: checkout: moving from main to styled
bb276e (tag: htmlify) HEAD@{11}: commit (merge): Arreglamos los conflictos de hacer el merge de las ramas htmlify y styled.
bb276e (tag: htmlify) HEAD@{12}: checkout: moving from htmlify to styled
1de880e (tag: htmlify) HEAD@{13}: commit: Cambios en git-nuestro dentro rama htmlify
0d71cb7 (tag: inicial) HEAD@{14}: checkout: moving from htmlify to main
0d71cb7 (tag: inicial) HEAD@{15}: checkout: moving from styled to main
bb276e (tag: styled) HEAD@{16}: reset: moving to bb276e
0d71cb7 (tag: inicial) HEAD@{17}: reset: moving to HEAD-1
bb276e (tag: styled) HEAD@{18}: commit: New version of git-nuestro.md
0d71cb7 (tag: inicial) HEAD@{19}: commit: Cambio de nombre de main a styled
0d71cb7 (tag: inicial) HEAD@{20}: commit: Initial commit
b43518b (origin/main, origin/HEAD) HEAD@{21}: clone: from https://github.com/dcammen/github.git
[END]

35. Ir al tag htmlify

Para movernos al tag htmlify usaremos el siguiente comando:

```
$ git checkout <tag_name>
```



```
git checkout htmlify  
Nota: cambiando a 'htmlify'.  
Te encuentras en estado 'detached HEAD'. Puedes revisar por aqui, hacer cambios experimentales y hacer commits, y puedes descartar cualquier commit que hayas hecho en este estado sin impactar a tu rama realizando otro checkout.  
Si quieras crear una nueva rama para mantener los commits que has creado, puedes hacerlo (ahora o despues) usando -c con el comando checkout. Ejemplo:  
    git switch -c nombre-de-nueva-rama  
O deshacer la operación con:  
    git switch -  
Desactiva este aviso poniendo la variable de config advice.detachedHead en false.  
HEAD está ahora en 1de880e Cambios en git-nuestro dentro rama htmlify
```

36. Vuelve a la rama main

Para volver a la rama main usaremos el siguiente comando:

```
$ git checkout main
```



```
git checkout main  
La posición previa de HEAD era 1de880e Cambios en git-nuestro dentro rama htmlify  
Cambiado a rama 'main'.  
Tu rama está actualizada a 'origin/main' por 7 commits.  
Tus "git push" para publicar tus commits locales.
```

37. Sube a GitHub todas las ramas y todos los tags

Antes de realizar el push del repositorio local al remoto, debemos recrear las ramas ya que las hemos borrado durante el ejercicio.

Para ello, ejecutamos los comandos:

```
$ git checkout <tag_name>  
$ git branch <tag_name>
```

Y cuando finalizamos volvemos a main y comprobamos el estado de la información que queremos subir al repositorio remoto.

```
> git checkout styled
Nota: cambiando a 'styled'.

Te encuentras en estado 'detached HEAD'. Puedes revisar por aqui, hacer cambios experimentales y hacer commits, y puedes descartar cualquier commit que hayas hecho en este estado sin impactar a tu rama realizando otro checkout.

Si quieres crear una nueva rama para mantener los commits que has creado, puedes hacerlo (ahora o despues) usando -c con el comando checkout. Ejemplo:

git switch -c <nombre-de-nueva-rama>

0 deshacer la operación con:

git switch -

Desactiva este aviso poniendo la variable de config advice.detachedHead en false

HEAD esté ahora en bb27c6e New version of git-nuestro.md
> git branch styled
> git branch
> git checkout htmlify
La posición previa de HEAD era bb27c6e New version of git-nuestro.md
HEAD esté ahora en ide80e Cambios en git-nuestro dentro rama htmlify
> git branch htmlify
> git checkout title
La posición previa de HEAD era ide80e Cambios en git-nuestro dentro rama htmlify
HEAD esté ahora en be02c56 Añadimos un título al fichero git-nuestro.md en la rama title
> git branch title
> git checkout main
La posición previa de HEAD era be02c56 Añadimos un título al fichero git-nuestro.md en la rama title
Cambiado a rama 'main'
Tus cambios se adelantaron a 'origin/main' por 7 commits.
(usa "git push" para publicar tus commits locales)
> git branch
  htmlify
* main
  styled
  title
[END]
```

Para sincronizar la información local con el repositorio remoto, ejecutamos los siguientes comandos:

```
$ git push origin --all
$ git push origin --tags
```