GIT

(utiliza git)

github
SOCIAL CODING

(utiliza git)

git - ?¿ (1)

- git es un sistema de control de versiones distribuído (scvd) escrito en C
- un sistema de control de versiones permite la creación de una 'historia' (aka **repository**) de un conjunto de archivos determinado, y en la que cada momento de la historia se guarda el estado de los archivos dicho momento, incluye la funcionalidad de revertir los archivos a un estado anterior de la 'historia'
- web oficial: https://git-scm.com/

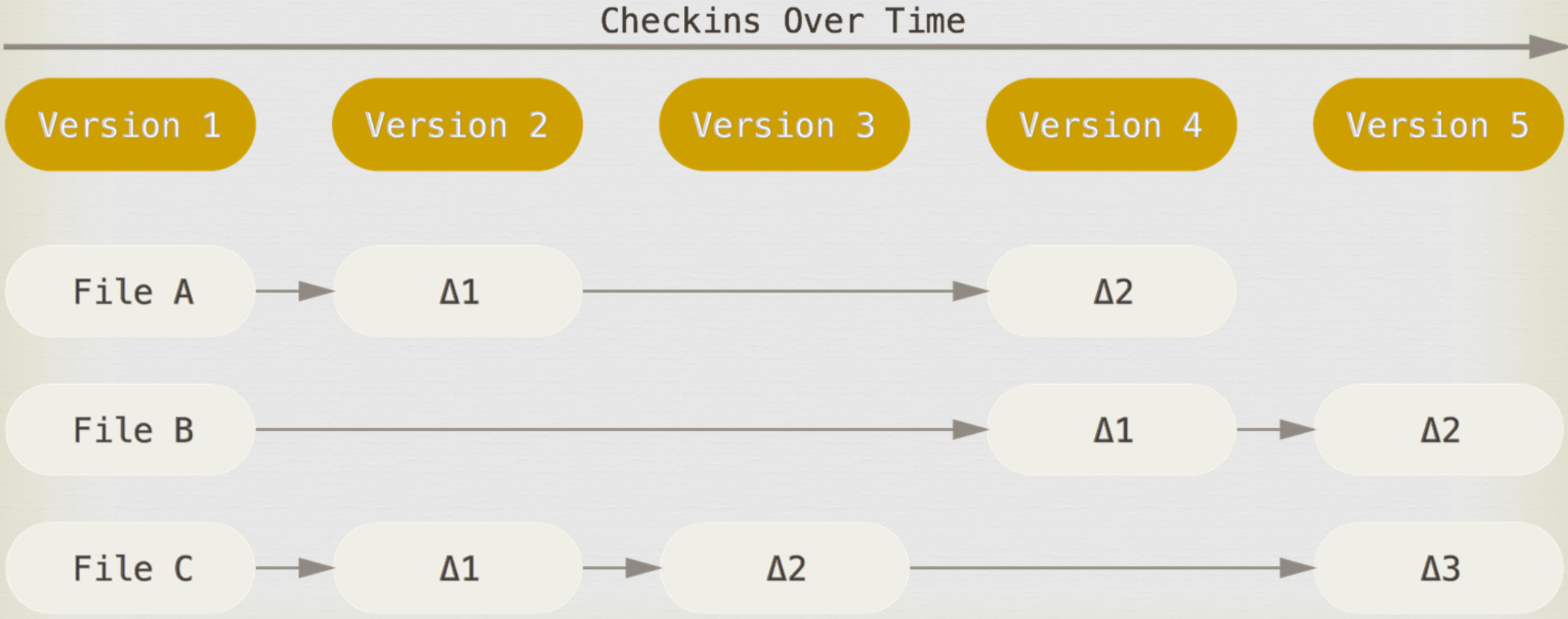
git - ?¿ (2)

- dicho de otra forma: Git es un programita (escrito por Linus Torvalds, todo sea dicho) que:
- te permite administrar-gestionar los cambios que haces en un proyecto
- ver que ha ocurrido
- juntar versiones distintas
- retroceder a un estado anterior (a hace una semana, por ej.)

difs. con otros SCV (1)

(SCV: sistema de control de versiones)

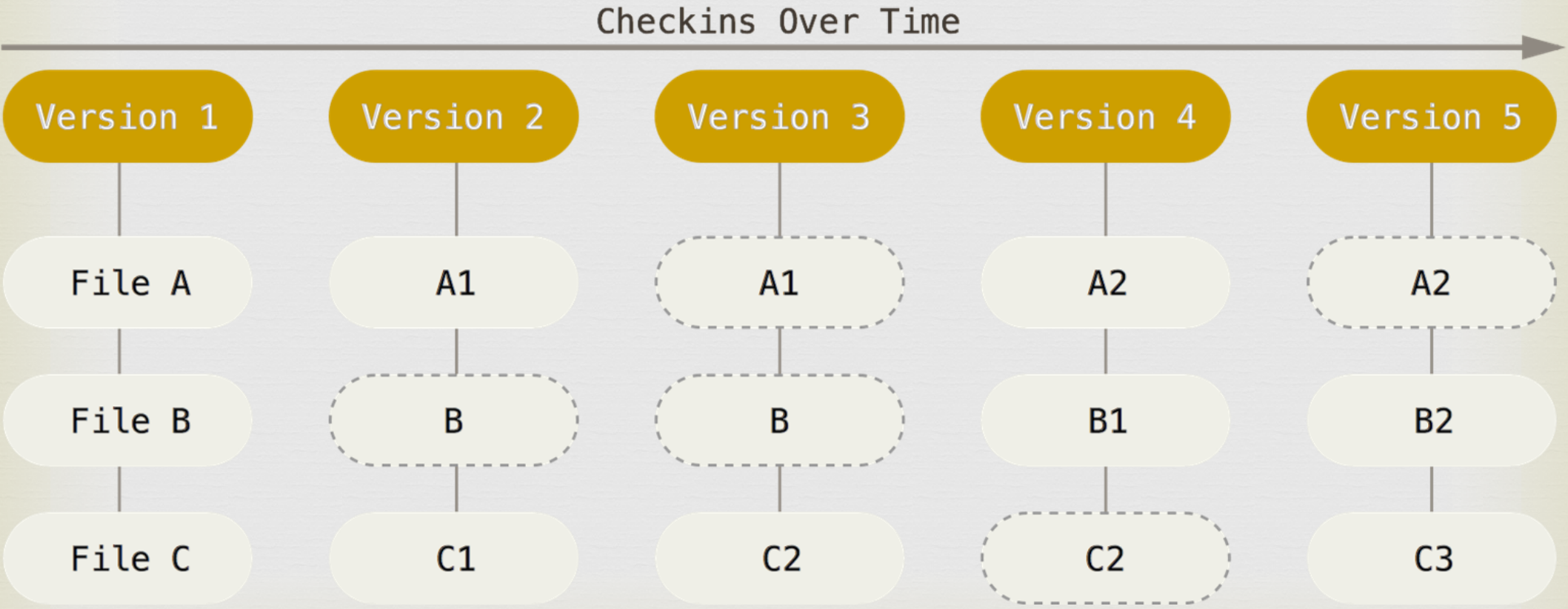
- otros SCV (Subversion, Bazaar, etc.), guardan los cambios hechos en los ficheros



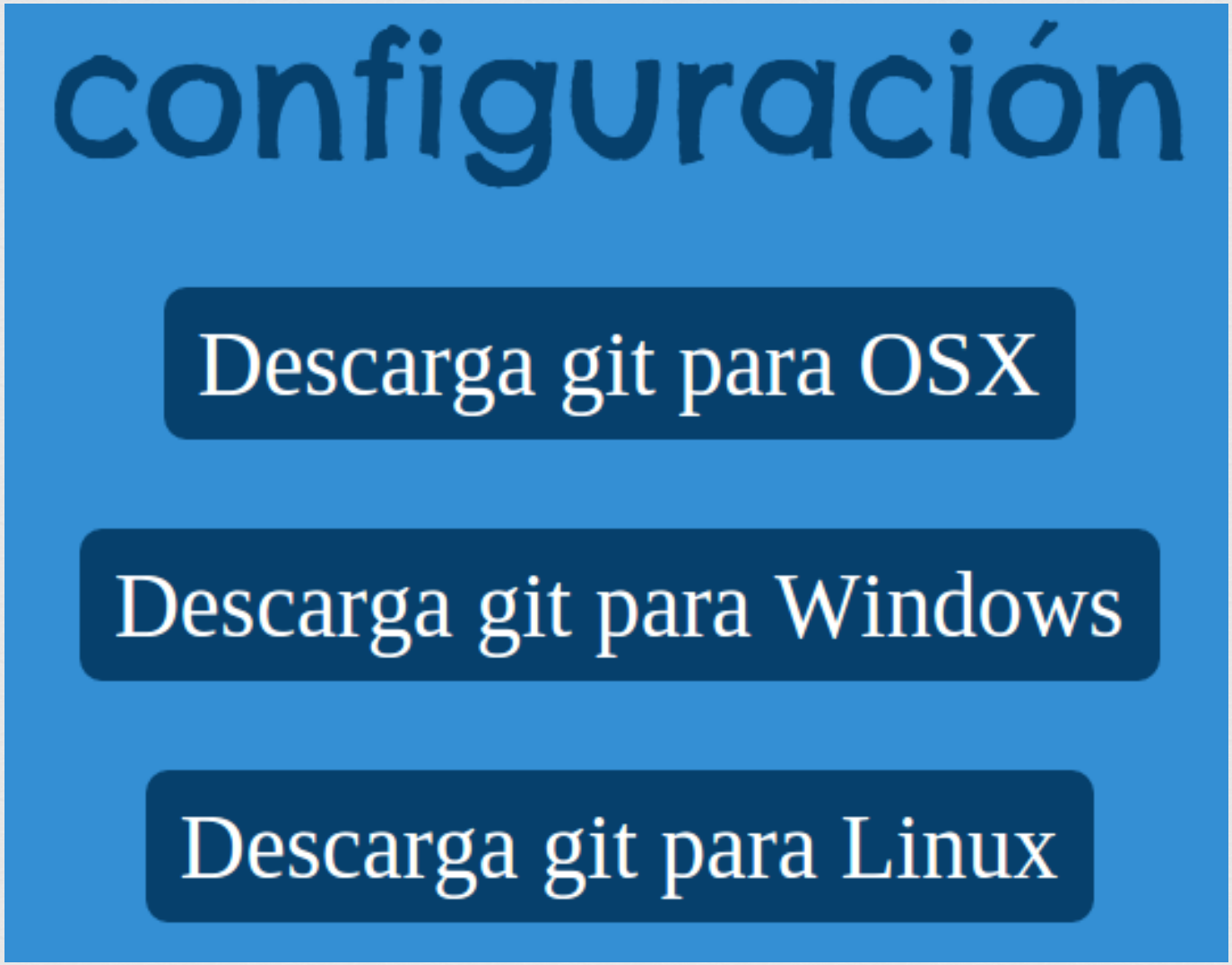
difs. con otros SCV (2)

(SCV: sistema de control de versiones)

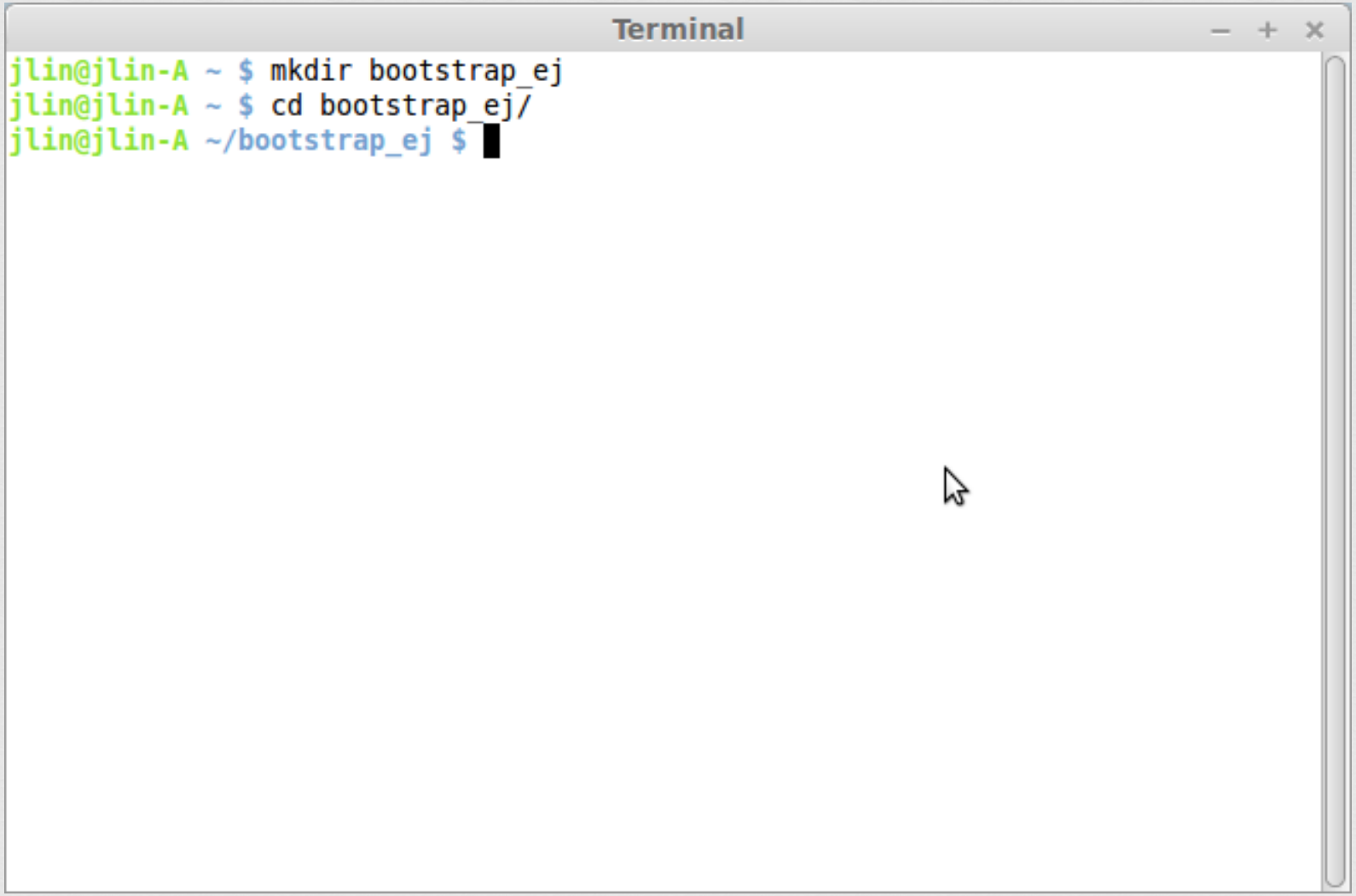
- git mantiene una imagen de todo en cada versión, lo punteado son links, no nuevas copias



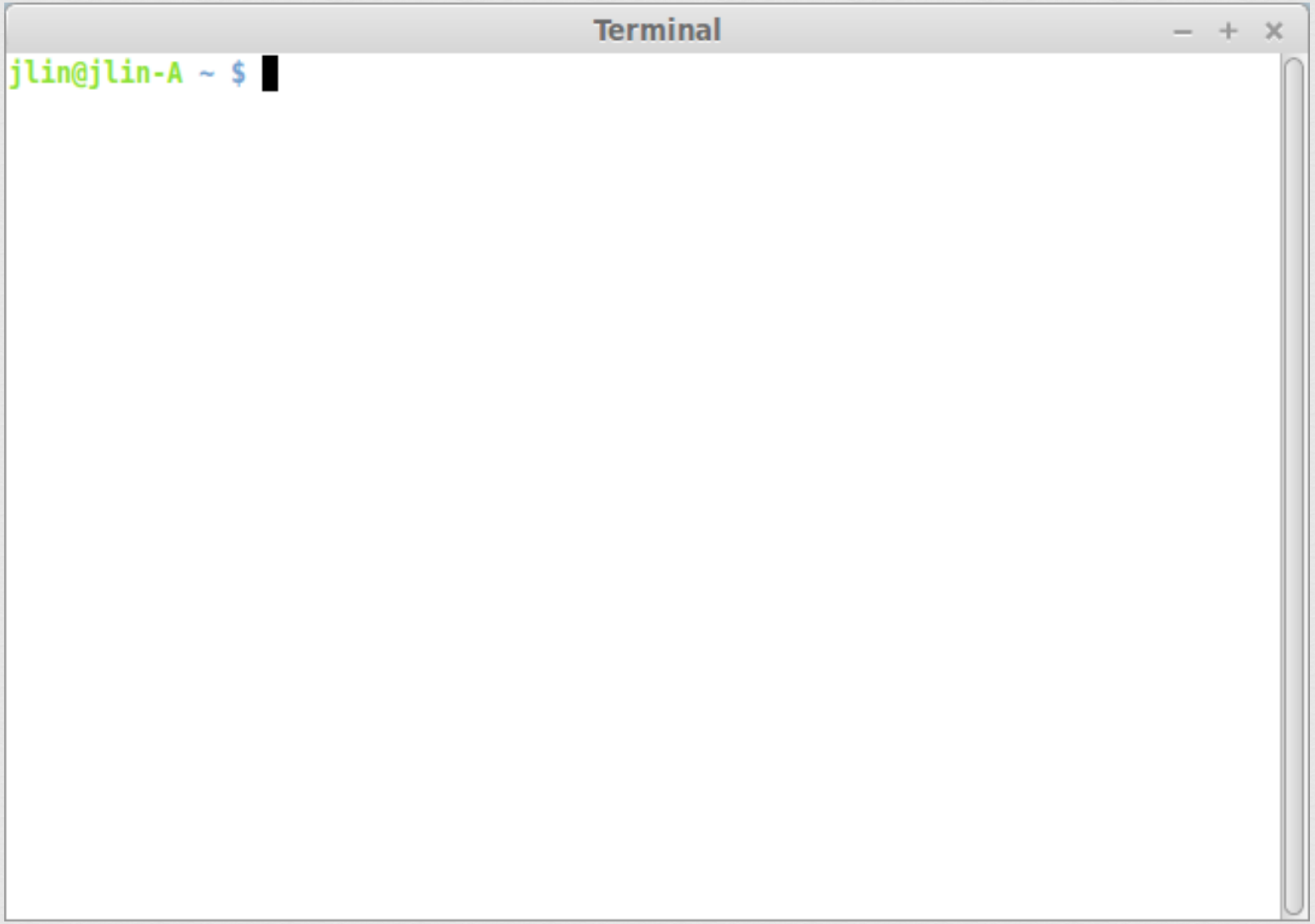
primero descargar e instalar



creamos carpeta de proyecto



a la consola...



git init

Esto crea un nuevo subdirectorio llamado .git que contiene todos los archivos necesarios del repositorio - un esqueleto de un repositorio Git. Todavía no hay nada en el proyecto que ponemos bajo seguimiento.

git init

Así creamos un repositorio (local)...

creamos un fichero...

creamos un fichero...

Edítalo con el programa que quieras...

Terminal

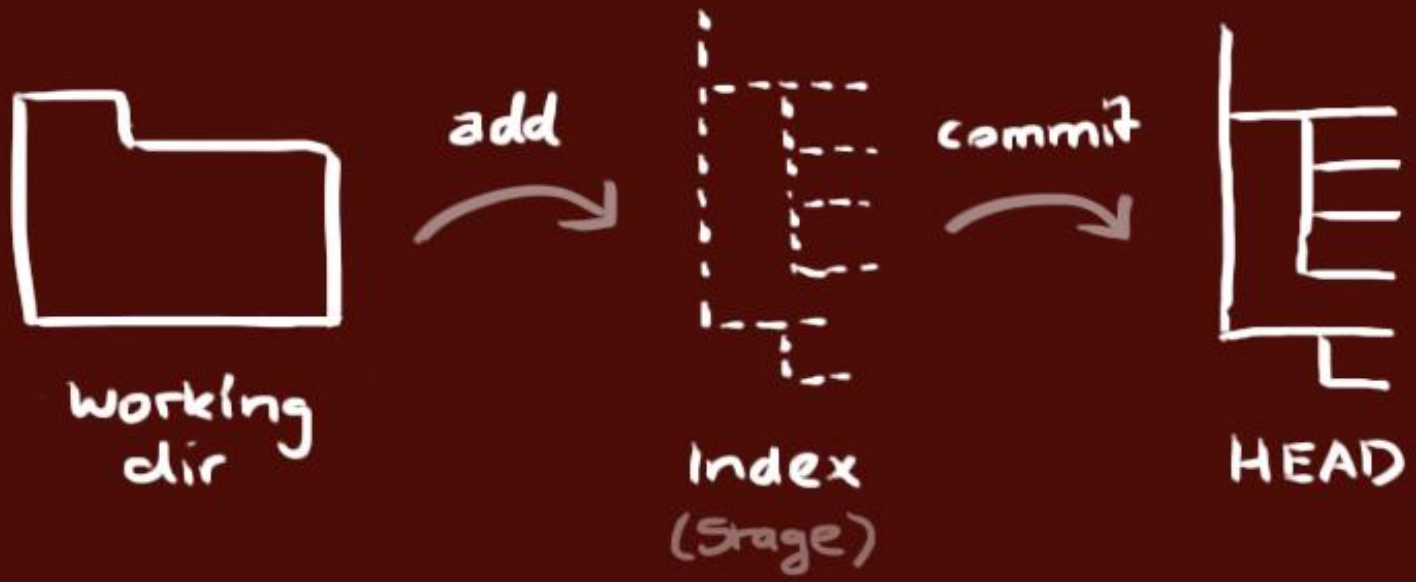
```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>inicial</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/
css/bootstrap.min.css">
    <style>
        body {
            background-color: #9a7;
        }

        .container-fluid {
            background-color: white;
        }
    </style>
</head>

<body>
    <p>Código base para una página con bootstrap, los css y los js</p>
                                                                6,15          Top
```

git add

Terminal

```
jlin@jlin-A ~/bootstrap_ej $ ls
index.html
jlin@jlin-A ~/bootstrap_ej $ # añadimos los ficheros html que tenemos para ser controlados por git
jlin@jlin-A ~/bootstrap_ej $ # en este caso solamente tenemos uno
jlin@jlin-A ~/bootstrap_ej $ git add *.html
jlin@jlin-A ~/bootstrap_ej $
```

Tu repositorio local esta compuesto por tres "árboles" administrados por git. El primero es tu `Directorio de trabajo` que contiene los archivos, el segundo es el `Index` que actua como una zona intermedia, y el último es el `HEAD` que apunta al último commit realizado.



git status

- Muestra el estado actual del árbol de ficheros que tenemos creado. Dicho de otra manera, te dice si los ficheros añadidos para ser vigilados están actualizados hasta su último cambio.

git status

- Vemos que falta por guardar (index.html)

```
Terminal
jlin@jlin-A ~/bootstrap_ej $ ls
index.html
jlin@jlin-A ~/bootstrap_ej $ # añadimos los ficheros html que tenemos en la carpeta para que git haga su seguimiento
jlin@jlin-A ~/bootstrap_ej $ # en este caso solamente tenmos uno
jlin@jlin-A ~/bootstrap_ej $ git add *.html
jlin@jlin-A ~/bootstrap_ej $
jlin@jlin-A ~/bootstrap_ej $ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   index.html

jlin@jlin-A ~/bootstrap_ej $ # con git status vemos que ficheros están bajo seguimiento y su estado
jlin@jlin-A ~/bootstrap_ej $
```

git commit -m “mensaje”

- Guardar los archivos añadidos mediante el comando ‘add’ en una nueva versión (de ahí que git sea un SCV, Sistema Control de Versiones), o en un momento de la historia.
- Cada commit es una imagen de lo que hacemos, y se puede recuperar posteriormente o juntar con otras versiones.

commit

- Realizamos un commit y se guardan los ficheros añadidos mediante el comando add en una

```
jlin@jlin-A ~/bootstrap_ej $ git commit -m "mi primer commit, plantilla inicial de bootstrap
[master (root-commit) 4181374] mi primer commit, plantilla inicial de bootstrap
 1 file changed, 29 insertions(+)
 create mode 100644 index.html
```

git status

- Si modificamos el fichero con el comando ‘git status’ vemos que el index.html ha sido modificado. Si queremos guardarlo primero haremos un ‘git add’ y después un ‘git commit’

```
jlin@jlin-A ~/bootstrap_ej $ # modificamos el archivo .html
jlin@jlin-A ~/bootstrap_ej $ vim index.html
jlin@jlin-A ~/bootstrap_ej $ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
jlin@jlin-A ~/bootstrap_ej $ # y vemos que git ve que ha habido un cambio
```

add + commit

Terminal

```
jlin@jlin-A ~/bootstrap_ej $ git commit -m "guardo en la 'rama principal' los ficheros guardados con el commmit"
On branch master
Changes not staged for commit:
        modified:   index.html

no changes added to commit
jlin@jlin-A ~/bootstrap_ej $ # como ves no hay nada que guardar (commit)
jlin@jlin-A ~/bootstrap_ej $ # si quiero guardar lo modificado tengo que hacer un 'git add' otra vez
jlin@jlin-A ~/bootstrap_ej $ git add *.html
jlin@jlin-A ~/bootstrap_ej $ git commit -m "he hecho una modificacion"
[master 783d6c3] he hecho una modificacion
 1 file changed, 1 insertion(+), 1 deletion(-)
jlin@jlin-A ~/bootstrap_ej $ # ahora si que ha funcionado
jlin@jlin-A ~/bootstrap_ej $ git status
On branch master
nothing to commit, working directory clean
jlin@jlin-A ~/bootstrap_ej $ # lo ves?
```

.gitignore

- Algunas veces deseamos que algunos archivos o directorios no sean incluídos en tu repositorio Git. Si los agregas al archivo .gitignore, Git comenzará a ignorarlos desde ese momento.

.gitignore

Terminal

```
jlin@jlin-A ~/bootstrap_ej $ ls
index.html
jlin@jlin-A ~/bootstrap_ej $ ls -la
total 16
drwxr-xr-x  3 jlin jlin 4096 may 15 23:08 .
drwxr-xr-x 85 jlin jlin 4096 may 15 23:08 ..
drwxr-xr-x  8 jlin jlin 4096 may 15 23:14 .git
-rw-r--r--  1 jlin jlin  769 may 15 23:08 index.html
jlin@jlin-A ~/bootstrap_ej $ touch .gitignore # creo el fichero .gitignore
jlin@jlin-A ~/bootstrap_ej $ vim .gitignore # lo edito, editalo con el editor de textos que quieras
```

editando .gitignore

Terminal

```
# para ignorar los ficheros acabados en .swp añado la siguiente linea
*.swp

#para ignorar los acabados en .o y .a
*.[oa]

# los que tu quieras que no se monitoreen aquí debajo...
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
                                                  3,0-1         All
```

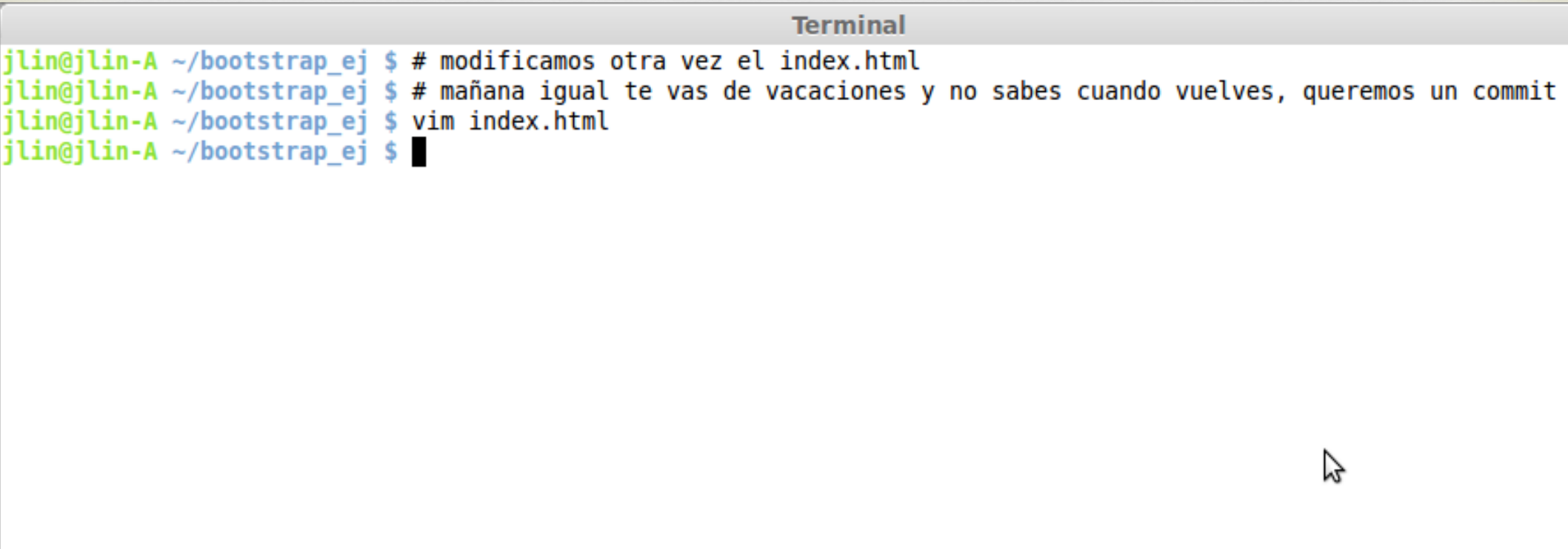
añadiendo .gitignore...

Terminal

```
jlin@jlin-A ~/bootstrap_ej $ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore

nothing added to commit but untracked files present (use "git add" to track)
jlin@jlin-A ~/bootstrap_ej $ git add .gitignore
jlin@jlin-A ~/bootstrap_ej $ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   .gitignore

jlin@jlin-A ~/bootstrap_ej $ # añadimos el .gitignore al proyecto git
jlin@jlin-A ~/bootstrap_ej $ git commit -m "añadido .gitignore"
[master d1e742e] añadido .gitignore
 1 file changed, 7 insertions(+)
 create mode 100644 .gitignore
jlin@jlin-A ~/bootstrap_ej $
```

modificaciones (1) - diff

- Hacer una modificación nos obliga a hacer un ‘add’ y un commit si queremos guardarlo



modificaciones (2) - diff

- Modificando el index.html...

Terminal

```
            background-color: #9a7;
        }

        .container-fluid {
            background-color: white;
        }
    </style>
</head>

<body>
    <p>Código base para una página con bootstrap, los css y los js</p>
    <h1>un h1</h1>
    <table>
        <thead>
                <tr>
                        <th>encabez</th>
                </tr>
        </thead>
        <tbody>
                <tr>
                        <td>cambio</td>
                </tr>
  </table>
```

34,4

git diff

Terminal

```
jlin@jlin-A ~/bootstrap_ej $ vim index.html
jlin@jlin-A ~/bootstrap_ej $
jlin@jlin-A ~/bootstrap_ej $ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
jlin@jlin-A ~/bootstrap_ej $ # pero yo quiero saber exactamente que ha cambiado!!
jlin@jlin-A ~/bootstrap_ej $ git diff
diff --git a/index.html b/index.html
index dada785..64659b8 100644
--- a/index.html
+++ b/index.html
@@ -21,6 +21,17 @@
 <body>
     <p>Código base para una página con bootstrap, los css y los js</p>
     <h1>un h1</h1>
+    <table>
+       <thead>
+               <tr>
+                       <th>encabez</th>
+               </tr>
+       </thead>
+       <tbody>
+               <tr>
+                       <td>cambio</td>
+               </tr>
+    </table>
     <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
     <script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/js/bootstrap.min.js"></script>
 </body>
jlin@jlin-A ~/bootstrap_ej $ # vemos todas las lineas con +, significa que las he añadido nuevas
```

git add .

Terminal

```
jlin@jlin-A ~/bootstrap_ej $ touch ejemplo.o
jlin@jlin-A ~/bootstrap_ej $ ls -la
total 20
drwxr-xr-x  3 jlin jlin 4096 may 15 23:33 .
drwxr-xr-x 85 jlin jlin 4096 may 15 23:27 ..
-rw-r--r--  1 jlin jlin    0 may 15 23:33 ejemplo.o
drwxr-xr-x  8 jlin jlin 4096 may 15 23:33 .git
-rw-r--r--  1 jlin jlin  182 may 15 23:21 .gitignore
-rw-r--r--  1 jlin jlin  891 may 15 23:27 index.html
jlin@jlin-A ~/bootstrap_ej $ git commit -m "he añadido una tabla"
On branch master
Changes not staged for commit:
        modified:   index.html

no changes added to commit
jlin@jlin-A ~/bootstrap_ej $ git add .
jlin@jlin-A ~/bootstrap_ej $ # con esto añado todos los ficheros en la nueva version
jlin@jlin-A ~/bootstrap_ej $
```

teniendo .gitignore...

Terminal

```
jlin@jlin-A ~/bootstrap_ej $ # añadimos a ejemplo.o
jlin@jlin-A ~/bootstrap_ej $ git add ejemplo.o
The following paths are ignored by one of your .gitignore files:
ejemplo.o
Use -f if you really want to add them.
fatal: no files added
jlin@jlin-A ~/bootstrap_ej $ # pero como en .gitignore le hemos dicho que no lo mire nos lo dice
```

añadimos otro fichero...

Terminal

```
jlin@jlin-A ~/bootstrap_ej $ # añadimos a ejemplo.o
jlin@jlin-A ~/bootstrap_ej $ git add ejemplo.o
The following paths are ignored by one of your .gitignore files:
ejemplo.o
Use -f if you really want to add them.
fatal: no files added
jlin@jlin-A ~/bootstrap_ej $ # pero como en .gitignore le hemos dicho que no lo mire nos lo dice
jlin@jlin-A ~/bootstrap_ej $
jlin@jlin-A ~/bootstrap_ej $ touch navbar.html # añado un fichero navbar.html
jlin@jlin-A ~/bootstrap_ej $
jlin@jlin-A ~/bootstrap_ej $ git add .
jlin@jlin-A ~/bootstrap_ej $ # así hemos añadido todos los ficheros en la carpeta local en el proyecto
```

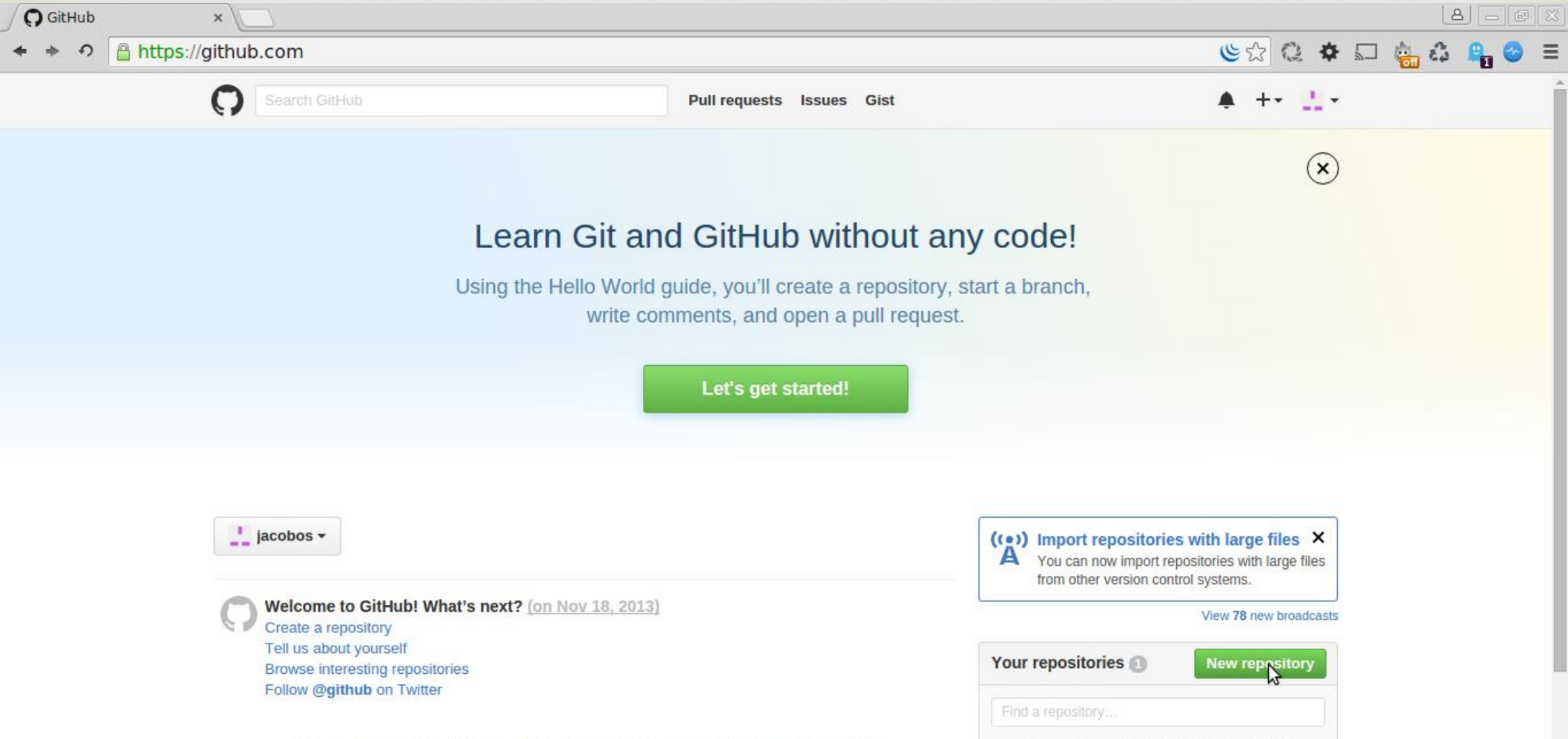
otro add y un commit

Terminal

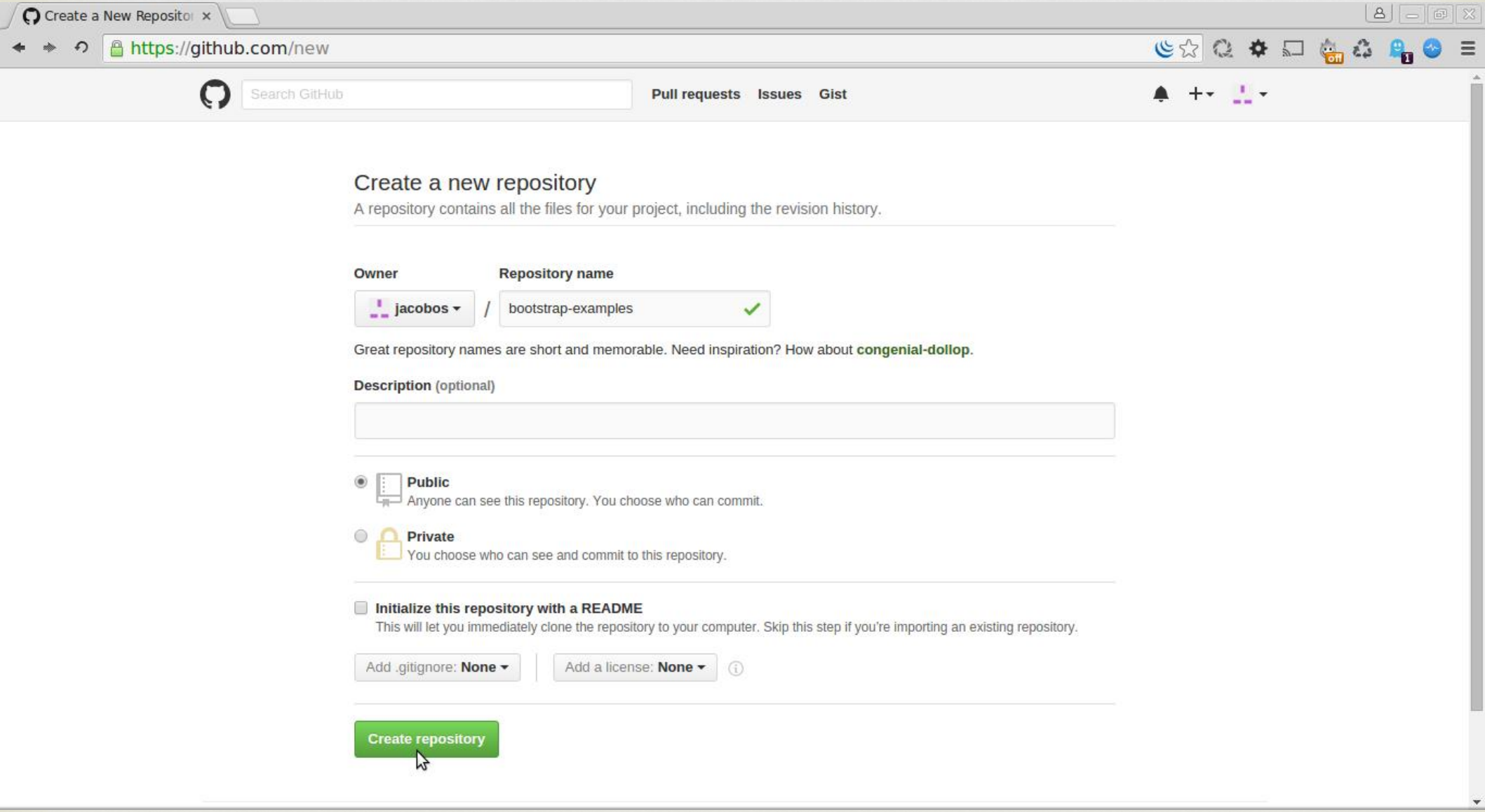
```
jlin@jlin-A ~/bootstrap_ej $ git add .
jlin@jlin-A ~/bootstrap_ej $ # así hemos añadido todos los ficheros en la carpeta local en el proyecto
jlin@jlin-A ~/bootstrap_ej $ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   index.html
        new file:   navbar.html

jlin@jlin-A ~/bootstrap_ej $ # nos muestra que hemos añadido uno nuevo
jlin@jlin-A ~/bootstrap_ej $
jlin@jlin-A ~/bootstrap_ej $ git commit -m "he añadido todos los ficheros nuevos, solo teniamos navbar.html"
[master e786e52] he añadido todos los ficheros nuevos, solo teniamos navbar.html
 2 files changed, 11 insertions(+)
 create mode 100644 navbar.html
jlin@jlin-A ~/bootstrap_ej $
```

y ahora lo subimos a github...

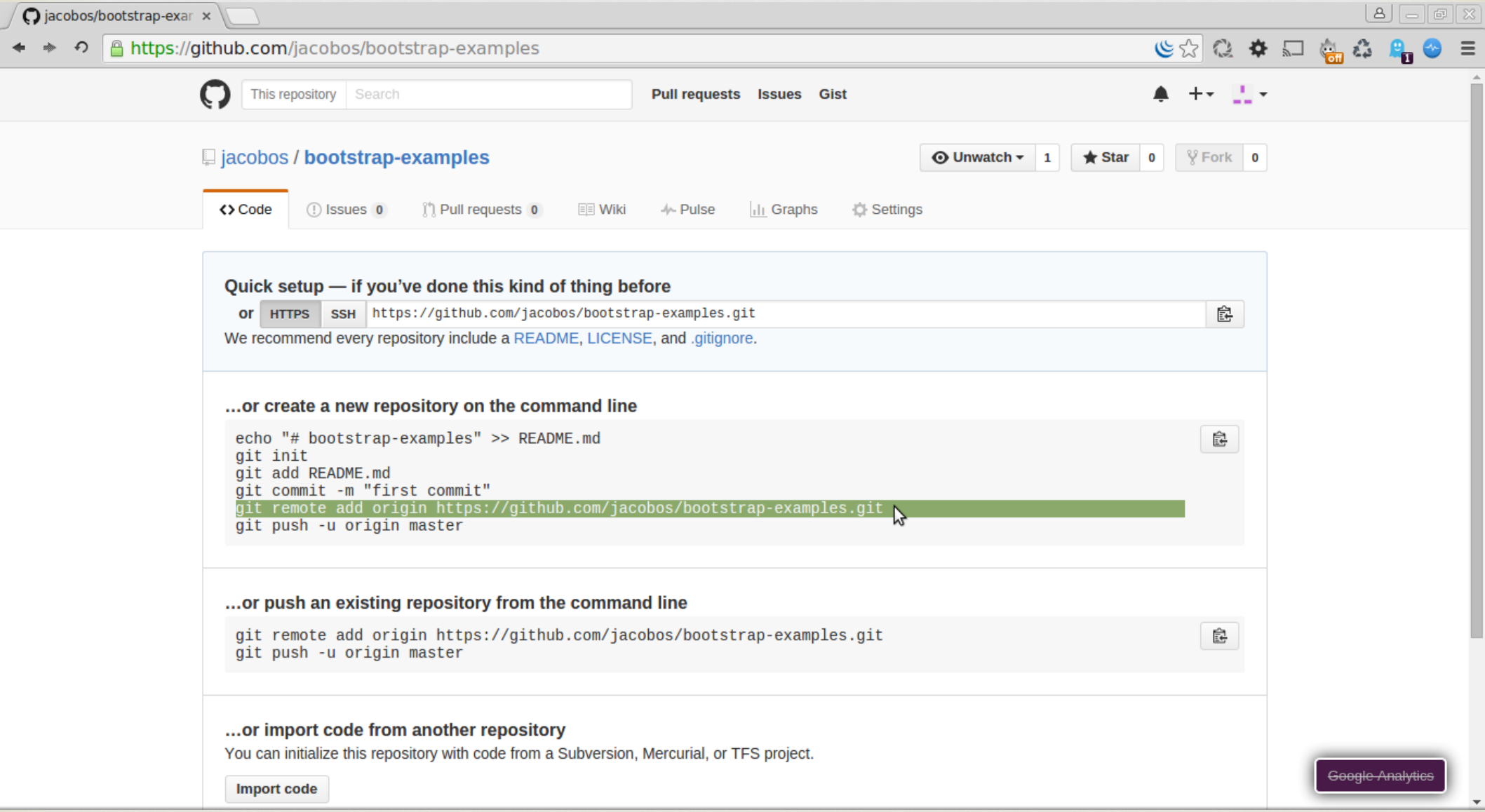
- Hasta este momento todo ha sido solamente git, creamos un repositorio nuevo



creamos un repositorio

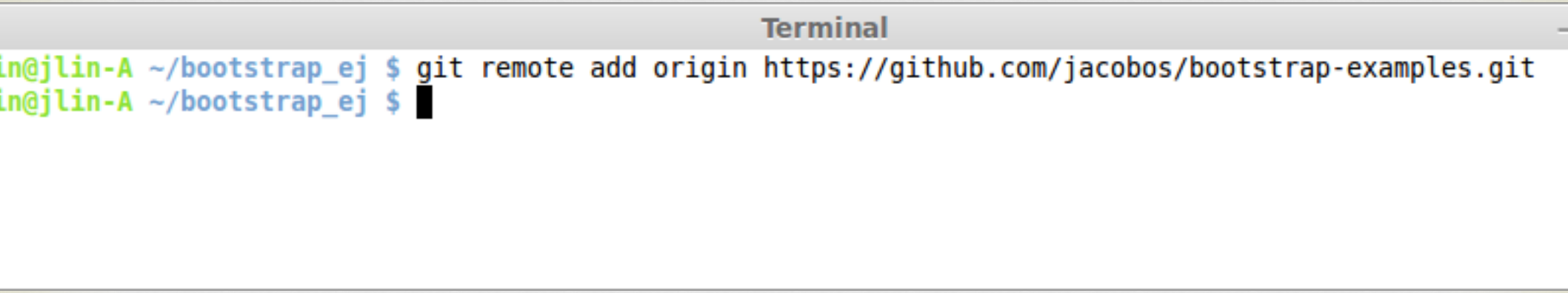


git remote add origin



git remote add origin

- Con este comando le decimos cuál es el repositorio remoto (el link que aparece) con el que queremos compartir los datos que tenemos en el repositorio local



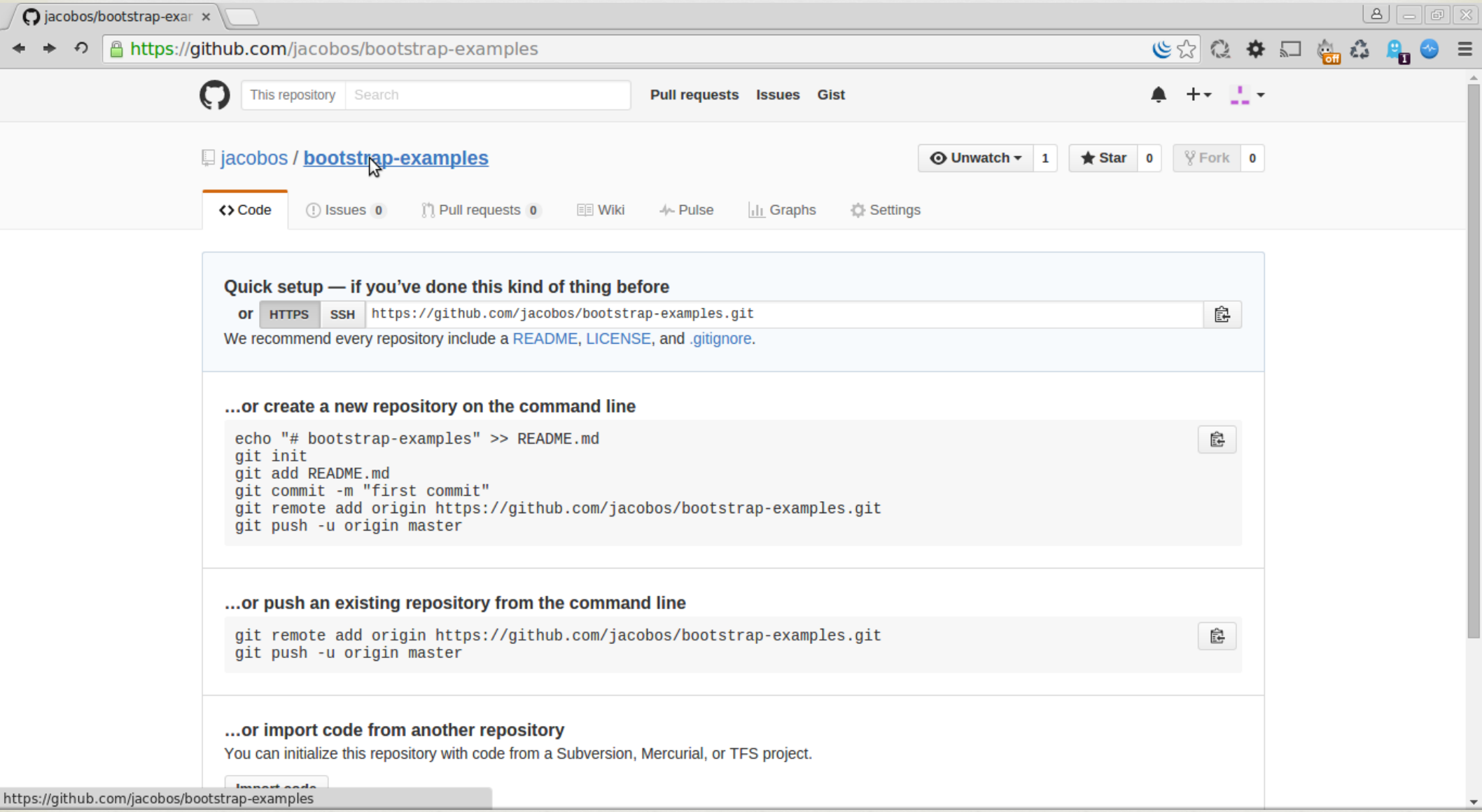
git push -u origin master

- Con este comando guardamos lo que tenemos en local en el remoto, en la rama principal (master)

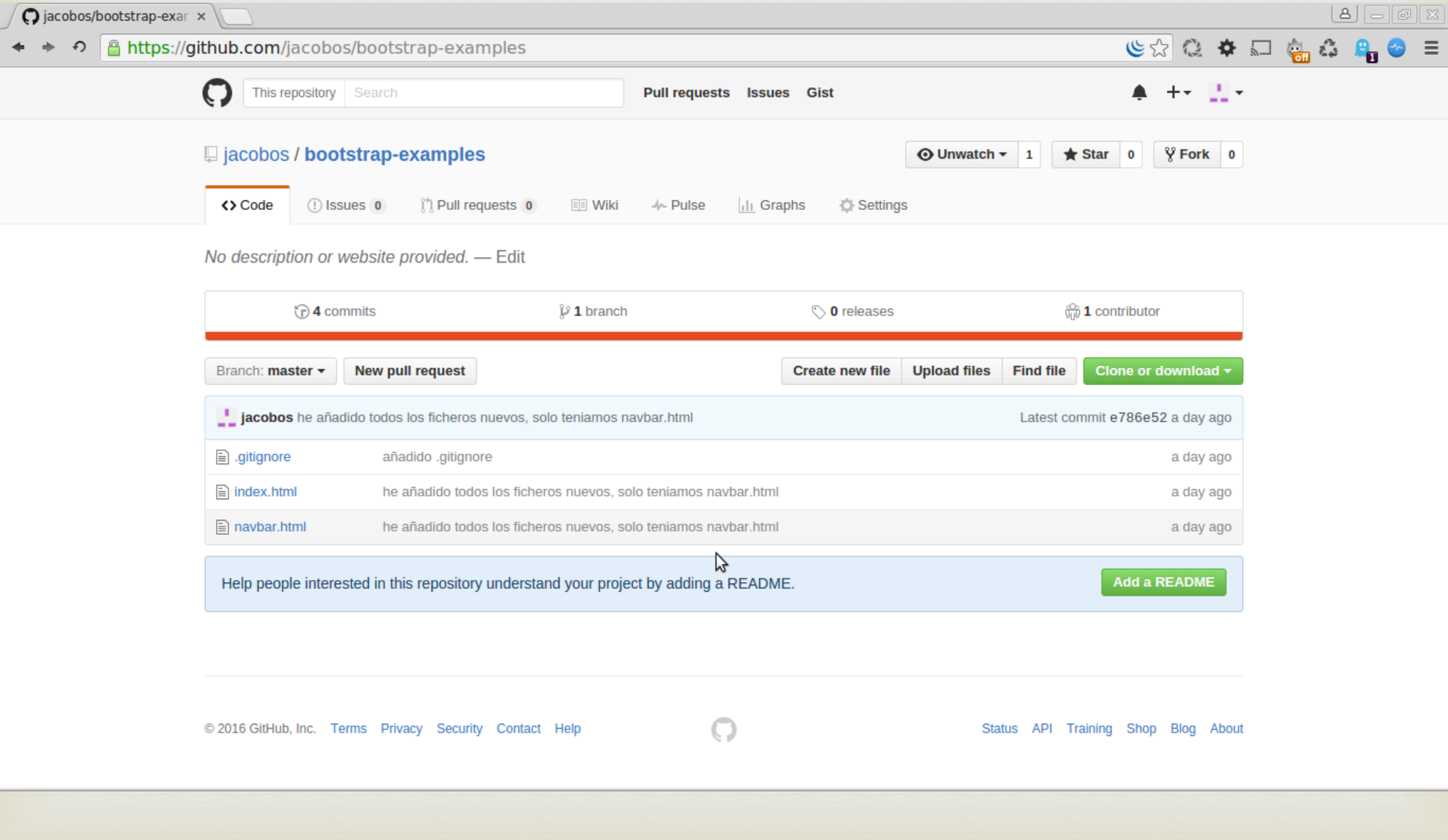
Terminal

```
jlin@jlin-A ~/bootstrap_ej $ git remote add origin https://github.com/jacobos/bootstrap-examples.git
jlin@jlin-A ~/bootstrap_ej $ git status
On branch master
nothing to commit, working directory clean
jlin@jlin-A ~/bootstrap_ej $ git push -u origin master
Username for 'https://github.com': [redacted]
Password for 'https://[redacted]@github.com':
Counting objects: 13, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (10/10), done.
Writing objects: 100% (13/13), 1.60 KiB | 0 bytes/s, done.
Total 13 (delta 2), reused 0 (delta 0)
To https://github.com/jacobos/bootstrap-examples.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
jlin@jlin-A ~/bootstrap_ej $
```

vamos al repositorio...



y vemos los ficheros subidos...



git clone <.git>

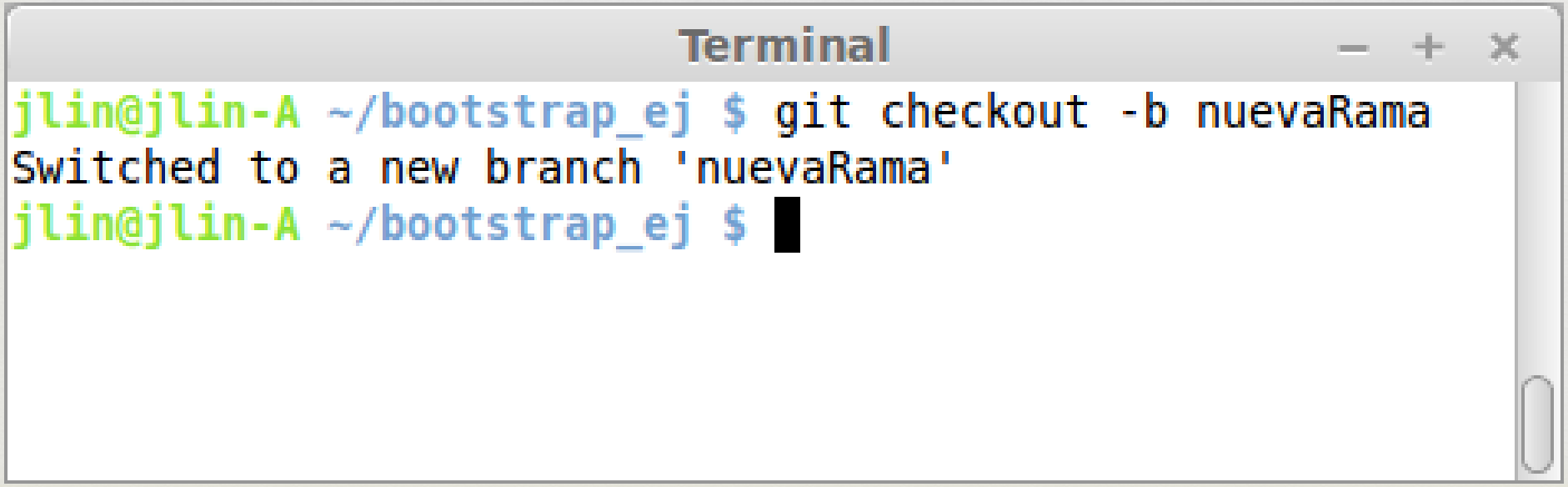
- Para clonar un repositorio entero en local
  - Git clone https: ·…/../repo.git (mira el ejemplo en la imagen)

Branches

- Por ahora hemos visto que en el proyecto actual en el que te encuentres (con una carpeta .git) se van guardando versiones del cógido (git commit…)
- Pero a su vez tenemos 'branches', ramas en castellano. Podemos tener subproyectos dentro del mismo repositorio. A partir de la branch original (la **master**) se crean desarrollos nuevos (nuevos ficheros, pequeñas variaciones...) pero toas se encuentran en el mismo repositorio (una carpeta con su .git)

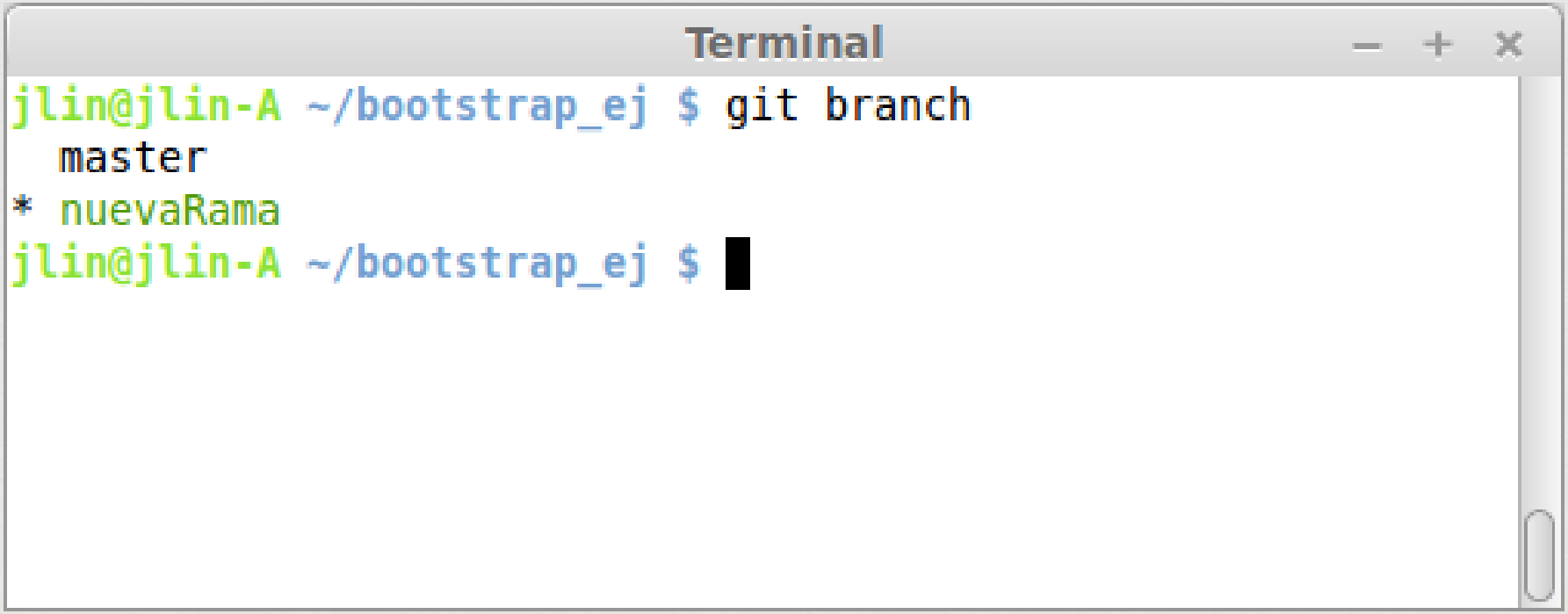
git checkout -b nuevaRama

- Con este comando creamos una nueva branch (en este caso llamada nuevaRama)
- Automáticamente cambiamos a dicho proyecto



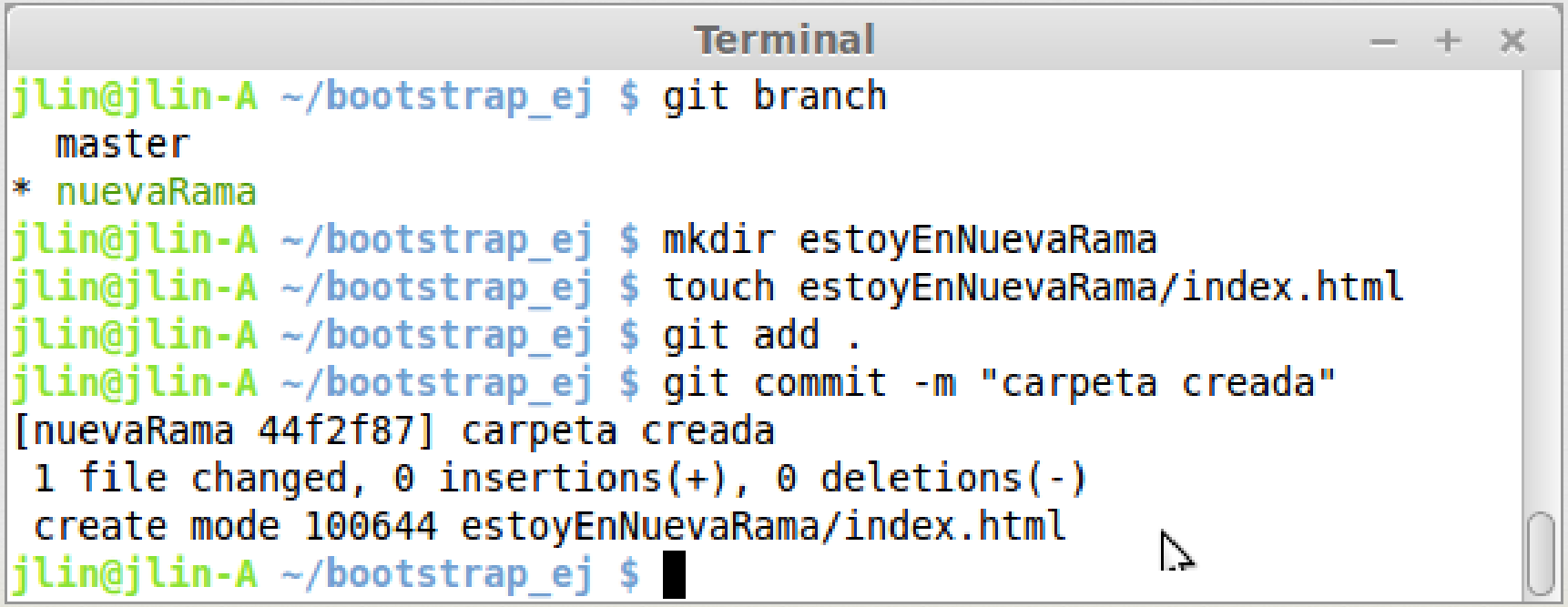
git branch

- Con este comando vemos las ‘branches’ creadas, y con un * se muestra en la que nos encontramos



Branch, ejemplo

- Creo una carpeta llamada estoyEnNuevaRama y dentro un html en la branch nuevaRama, add&commit



Branch, ejemplo

git checkout master

- Con este comando vamos a la branch master (miremos las diferencias)

```
Terminal
jlin@jlin-A ~/bootstrap_ej $ git branch
  master
* nuevaRama
jlin@jlin-A ~/bootstrap_ej $ ls
ejemplo.o          gitting.pdf  kraken.txt   README.md
estoyEnNuevaRama   index.html   navbar.html
jlin@jlin-A ~/bootstrap_ej $ # ahora volvamos a la 'rama' master
jlin@jlin-A ~/bootstrap_ej $ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
jlin@jlin-A ~/bootstrap_ej $ ls
ejemplo.o  gitting.pdf  index.html  kraken.txt  navbar.html  README.md
jlin@jlin-A ~/bootstrap_ej $ # miramos los ficheros y vemos que la carpeta est
oyEnNuevaRama no existe, es otro proyecto pero dentro del mismo repositorio
```

Branch, ejemplo

- Volvemos y vemos que funciona como lo hemos pensado :)

Terminal

```
jlin@jlin-A ~/bootstrap_ej $ git branch
* master
  nuevaRama
jlin@jlin-A ~/bootstrap_ej $ git checkout nuevaRama
Switched to branch 'nuevaRama'
jlin@jlin-A ~/bootstrap_ej $ ls
ejemplo.o          gitting.pdf  kraken.txt   README.md
estoyEnNuevaRama   index.html   navbar.html
jlin@jlin-A ~/bootstrap_ej $ # vemos que vuelve a estar la carpeta
jlin@jlin-A ~/bootstrap_ej $ # el resto esta guardado en el .git, ya se encarg
a git de hacer lo que toque para que funcione
```

git branch -d nombreRama

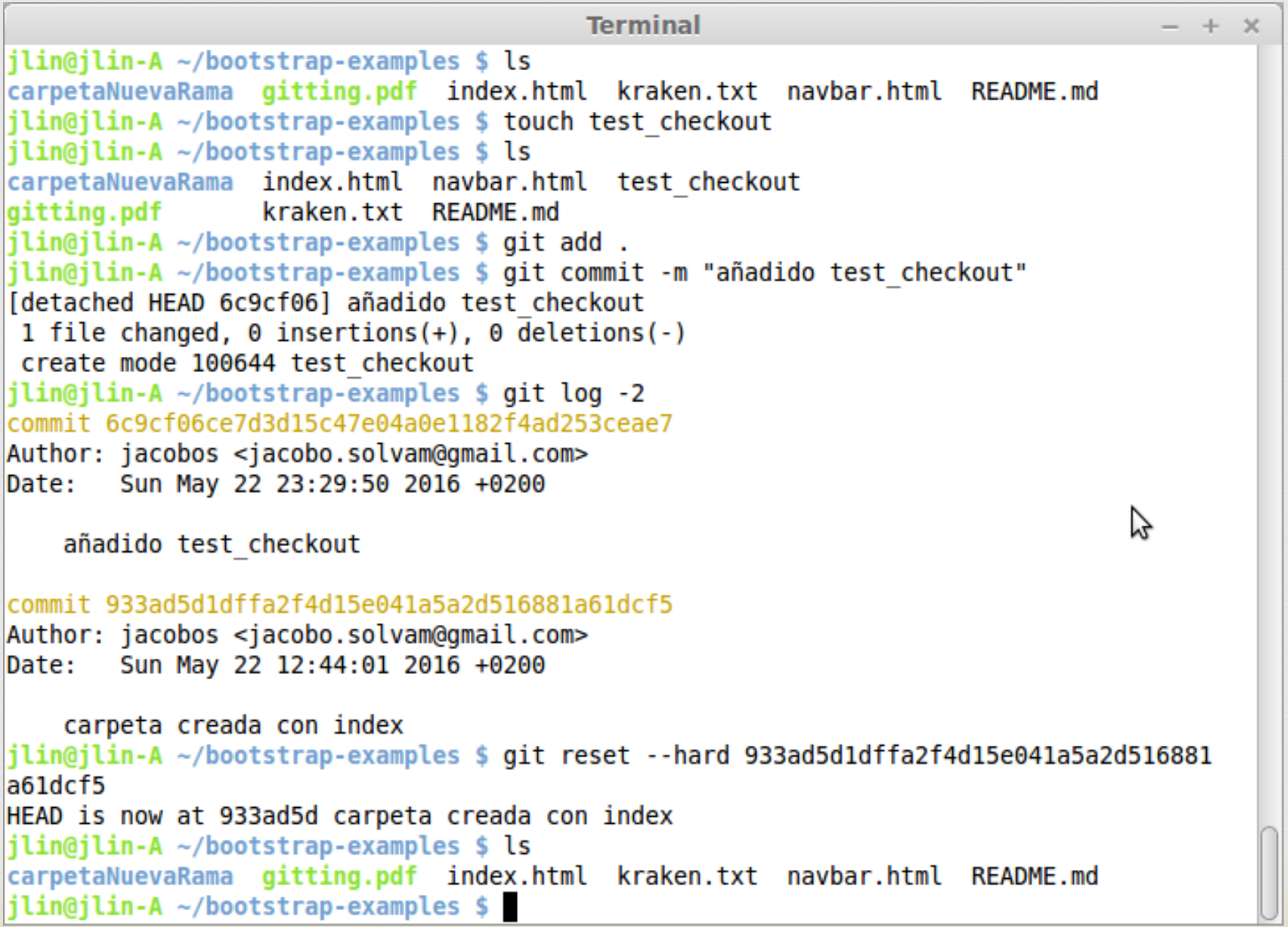
- Si por alguna razón queremos borrar una branch, con este comando la borraríamos del proyecto (no podemos estar dentro de dicha branch para borrarlo), si ponemos -D en lugar de -d si hay cambios que no están en la rama principal de la que desciende el proyecto los borraría sin preguntar.

git branch -D nombreRama

Terminal

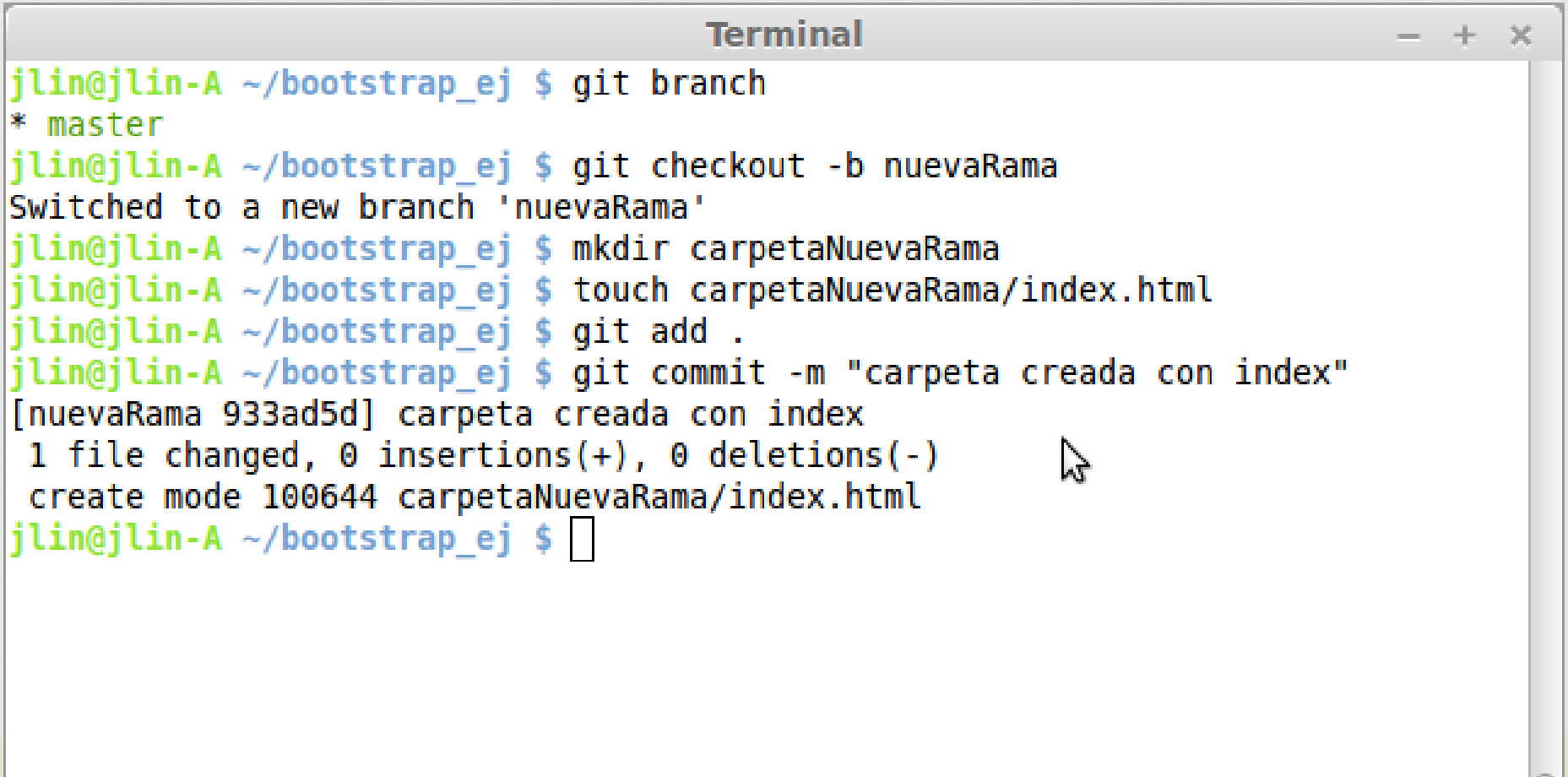
```
jlin@jlin-A ~/bootstrap_ej $ git branch
  master
* nuevaRama
jlin@jlin-A ~/bootstrap_ej $ git branch -d nuevaRama
error: Cannot delete the branch 'nuevaRama' which you are currently on.
jlin@jlin-A ~/bootstrap_ej $ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
jlin@jlin-A ~/bootstrap_ej $ git branch -d nuevaRama
error: The branch 'nuevaRama' is not fully merged.
If you are sure you want to delete it, run 'git branch -D nuevaRama'.
jlin@jlin-A ~/bootstrap_ej $ # como hay cambios que no hemos juntado con el pr
oyecto original, nos pide la D mayúscula, para asegurarse que lo queremos
jlin@jlin-A ~/bootstrap_ej $ git branch -D nuevaRama
Deleted branch nuevaRama (was 44f2f87).
jlin@jlin-A ~/bootstrap_ej $ git branch
* master
jlin@jlin-A ~/bootstrap_ej $
```

Ejemplo para volver a una versión anterior con → git reset –hard numDeCommit



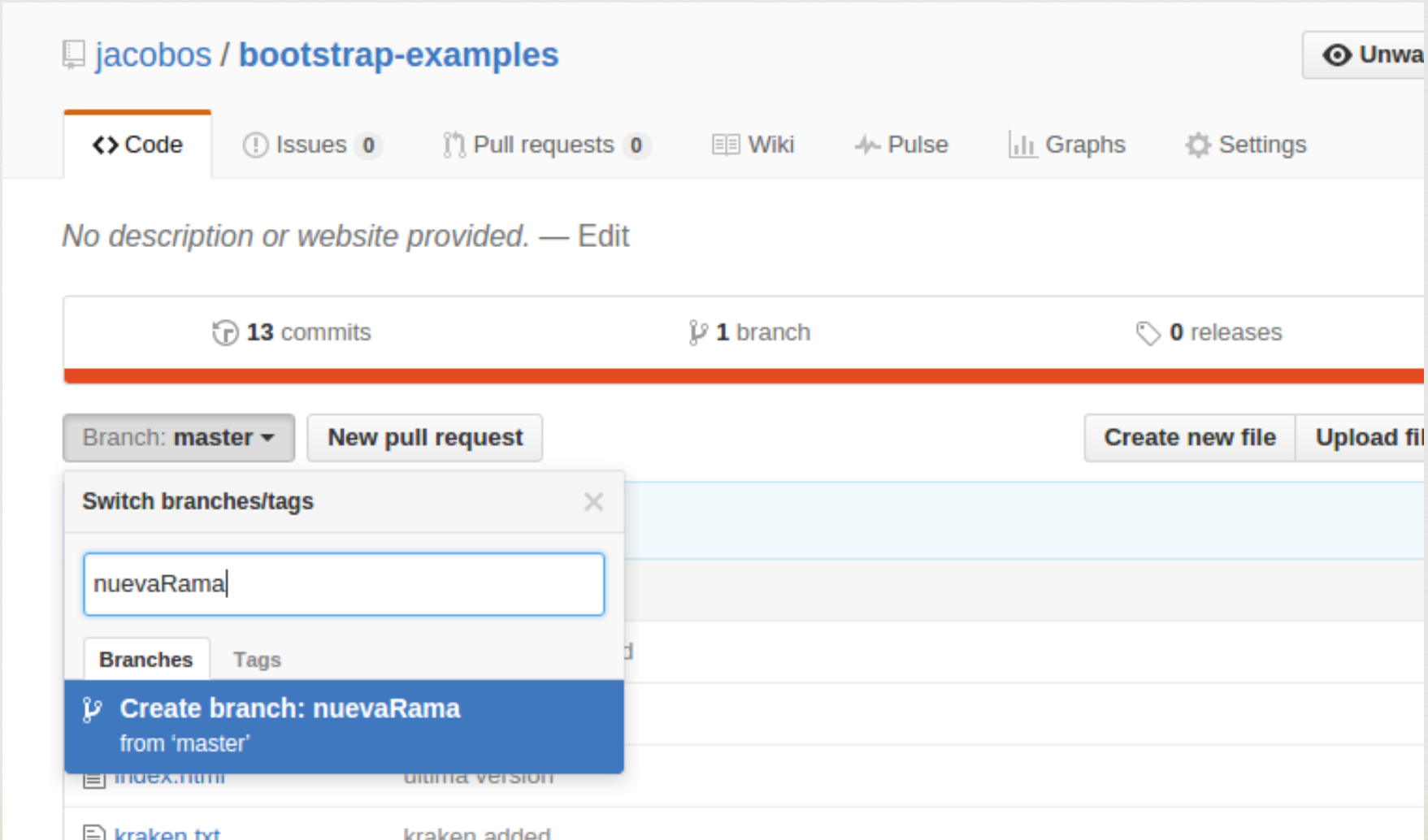
Para subir a un repo remoto

- Creo una nueva, con una carpeta y un fichero dentro



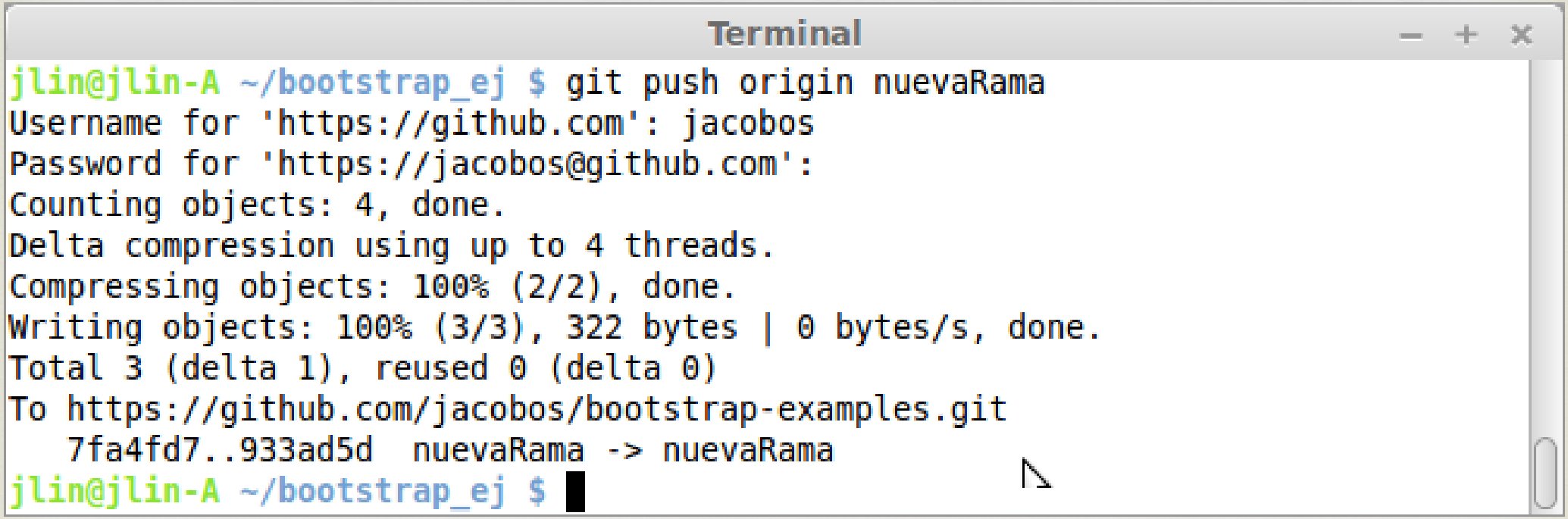
y para subir a remoto...

- Creo la nueva ‘branch’ en github

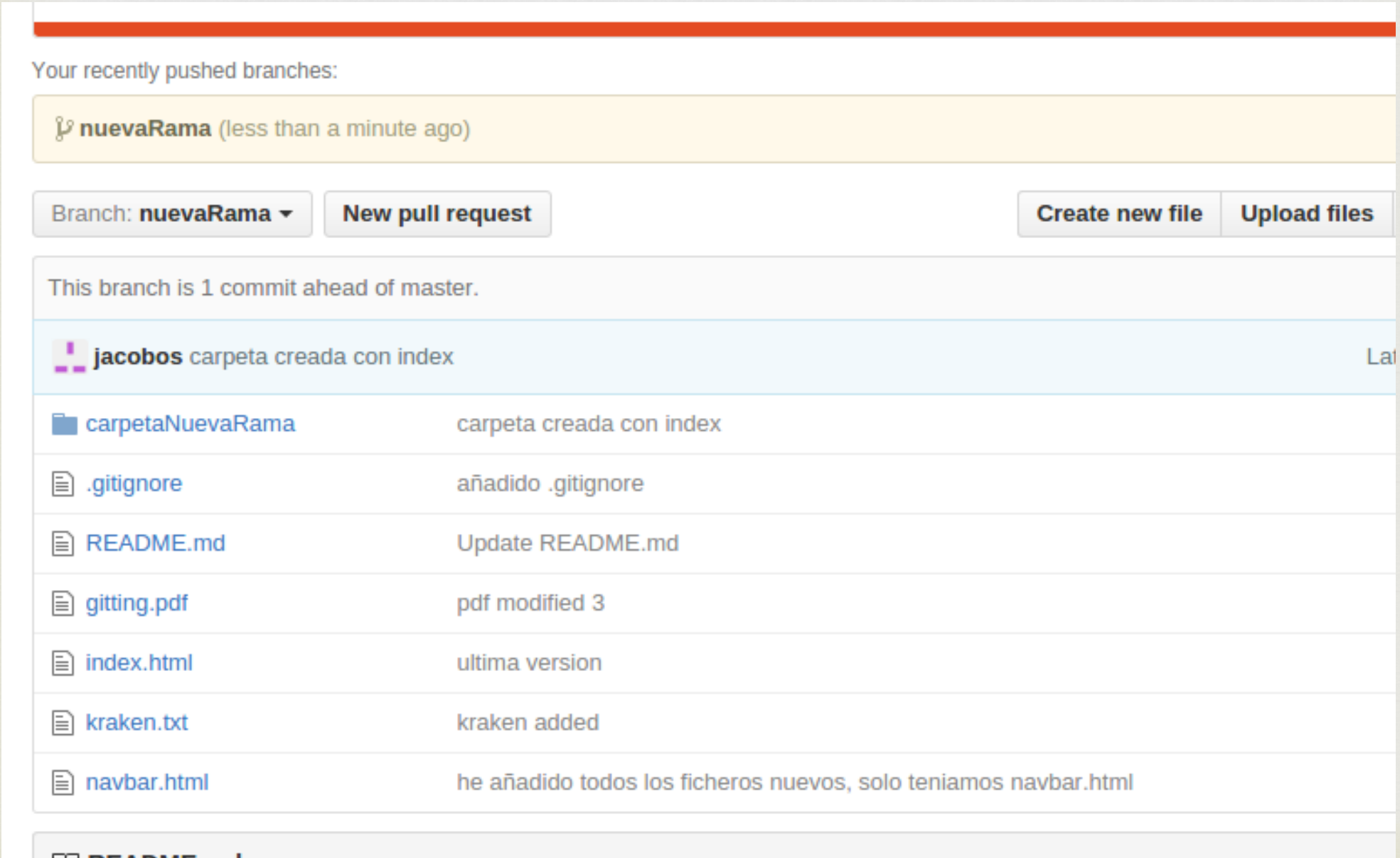


git push origin <branch>

- Y lo subimos, de otra manera solo lo tendremos en local ;), solo hay que poner el nombre del branch

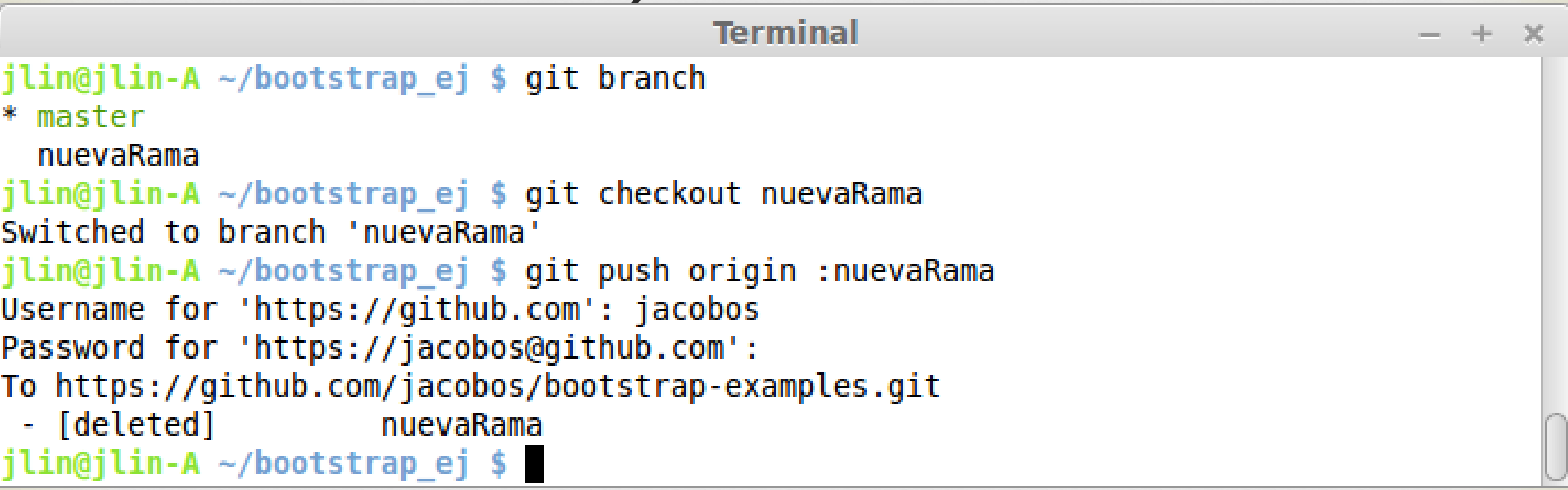


y vemos que se ha subido...



git push origin :<branch>

- Con este comando borramos una branch remota, abajo un ejemplo (haz el test, pero recuerda que se borra del todo)



git push <origin> <master>

Si dos personas intentan subir sus ficheros modificados al mismo repositorio el segundo no lo podrá hacer porqué dejaría una versión no ‘estable’. El segundo primero tendrá que descargar las versiones, unirlas y una vez hecho esto podrá juntar la suya y subirlo en una nueva versión.

git merge <branch>

- Si queremos fusionar dos branches distintas se hace con este comando
- Fusiona la branch actual (git branch) con la que le digamos en el parámetro que le pasamos

git merge en acción

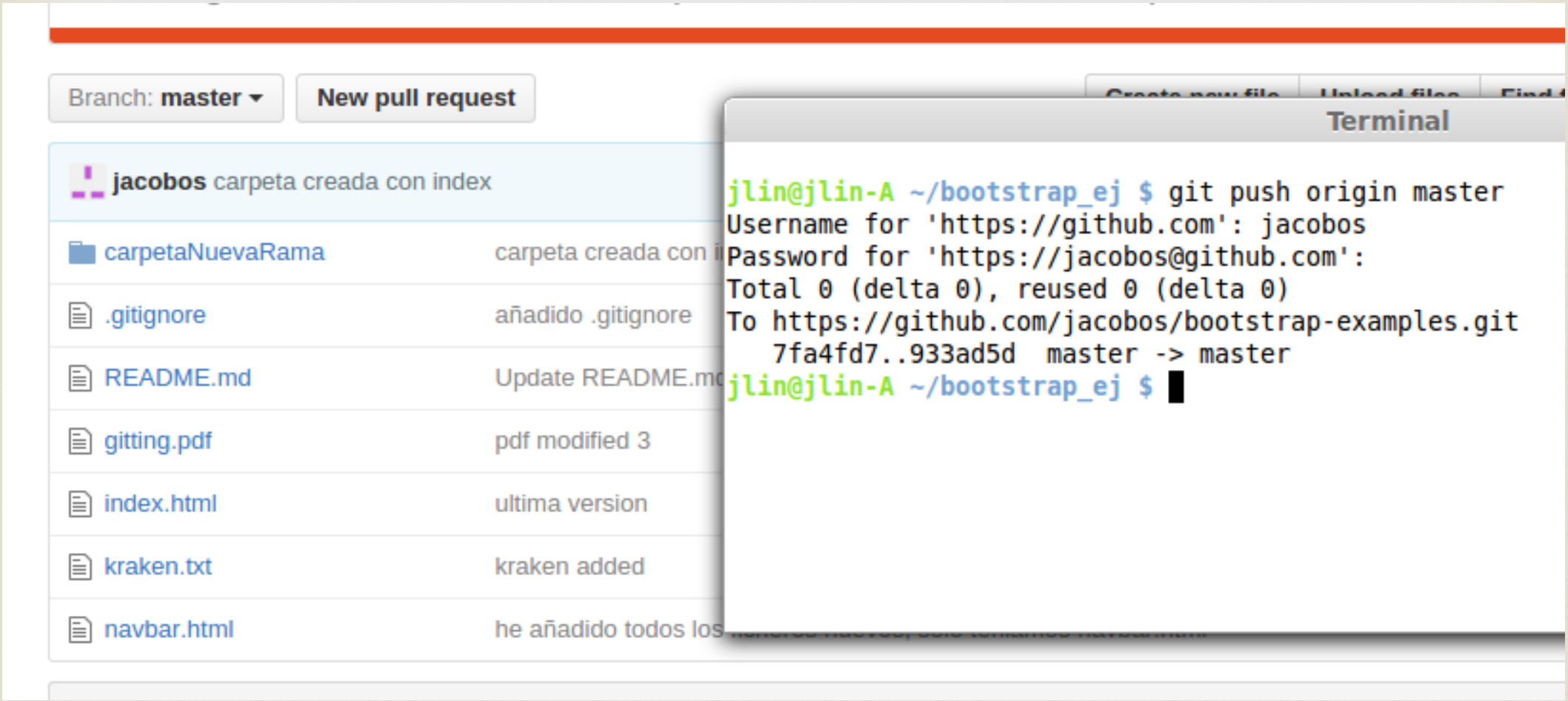
- Vemos cuál es la branch actual y la juntamos con la branch nuevaRama (nuevaRama sigue siendo una branch)

Terminal

```
jlin@jlin-A ~/bootstrap_ej $ git branch
* master
  nuevaRama
jlin@jlin-A ~/bootstrap_ej $ git merge nuevaRama
Updating 7fa4fd7..933ad5d
Fast-forward
 carpetaNuevaRama/index.html | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 carpetaNuevaRama/index.html
jlin@jlin-A ~/bootstrap_ej $ ls
carpetaNuevaRama  gitting.pdf  kraken.txt   README.md
ejemplo.o         index.html   navbar.html
jlin@jlin-A ~/bootstrap_ej $
```

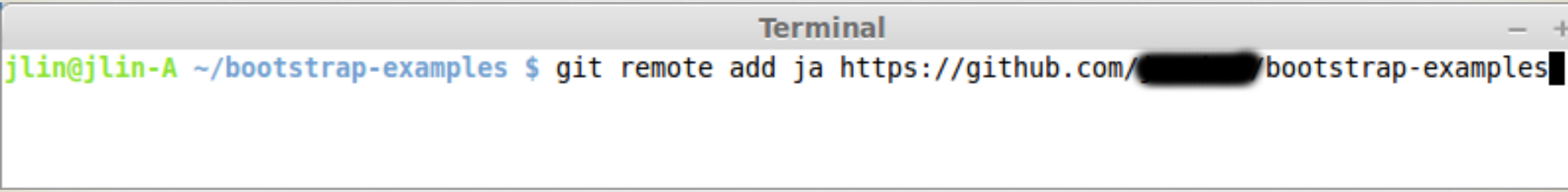
y subimos el cambio

- (a la izquierda vemos el resultado tras hacer el push)



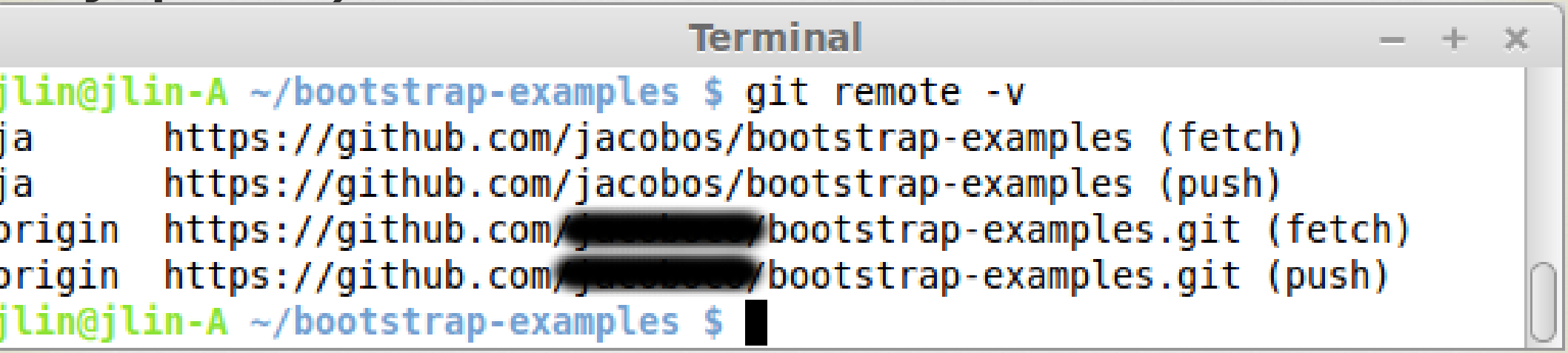
git remote add

- Añadiendo un repositorio remoto para poder compartir proyectos, en este caso le ponemos al local el nombre “ja”



git remote -v

- Este comando nos muestra todos los repos remotos con los que poder trabajar (los comandos fetch y push)



git remote show <repo>

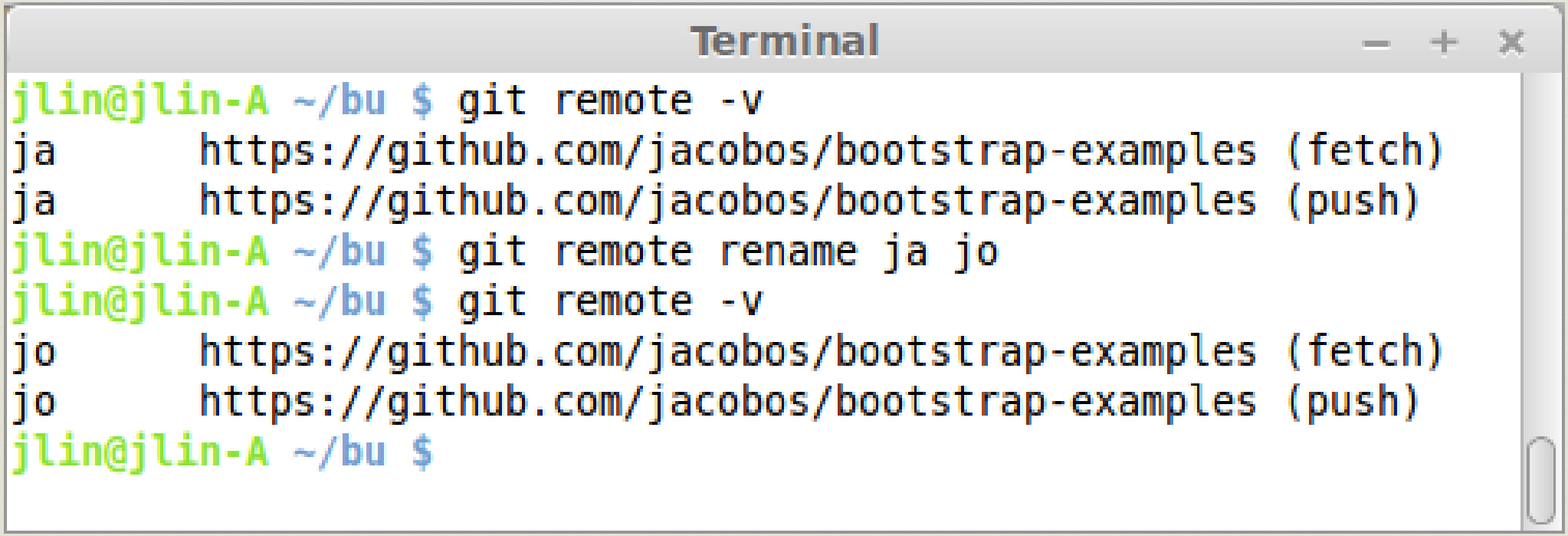
- Con este comando vemos el estado del repositorio (recuerda que antes le hemos asignado a uno el nombre ‘ja’

Terminal

```
jlin@jlin-A ~/bu $ git remote -v
ja      https://github.com/jacobos/bootstrap-examples (fetch)
ja      https://github.com/jacobos/bootstrap-examples (push)
jlin@jlin-A ~/bu $ git remote show ja
* remote ja
  Fetch URL: https://github.com/jacobos/bootstrap-examples
  Push  URL: https://github.com/jacobos/bootstrap-examples
  HEAD branch: master
  Remote branches:
    master    tracked
    nuevaRama tracked
  Local branch configured for 'git pull':
    master merges with remote master
  Local ref configured for 'git push':
    master pushes to master (up to date)
```

git remote rename nom_orig nom_dest

- Para renombrar un repositorio remoto (el alias que le hemos puesto) por otro lo haríamos con este comando.



git remote add y remove

- git remote add <nombre> <repo> → añade al repositorio en el que nos encontramos uno remoto con el que trabajar
- git remote -v → muestra info
- git remote rm <nombre> → lo elimina

Terminal

```
jlin@jlin-A ~/bu $ git remote add ja https://github.com/jacobos/bootstrap-examples
jlin@jlin-A ~/bu $ git remote -v
ja      https://github.com/jacobos/bootstrap-examples (fetch)
ja      https://github.com/jacobos/bootstrap-examples (push)
jlin@jlin-A ~/bu $ git remote rename ja jo
jlin@jlin-A ~/bu $ git remote -v
jo      https://github.com/jacobos/bootstrap-examples (fetch)
jo      https://github.com/jacobos/bootstrap-examples (push)
jlin@jlin-A ~/bu $ git remote rm jo
jlin@jlin-A ~/bu $ git remote -v
jlin@jlin-A ~/bu $
```

git fetch <remote>

- Para descargar datos de un proyecto remoto se hace con fetch
- git fetch <remote>
- Este comando no junta versiones, solo descarga el proyecto o rama que le pidas

Ejemplo de fetch

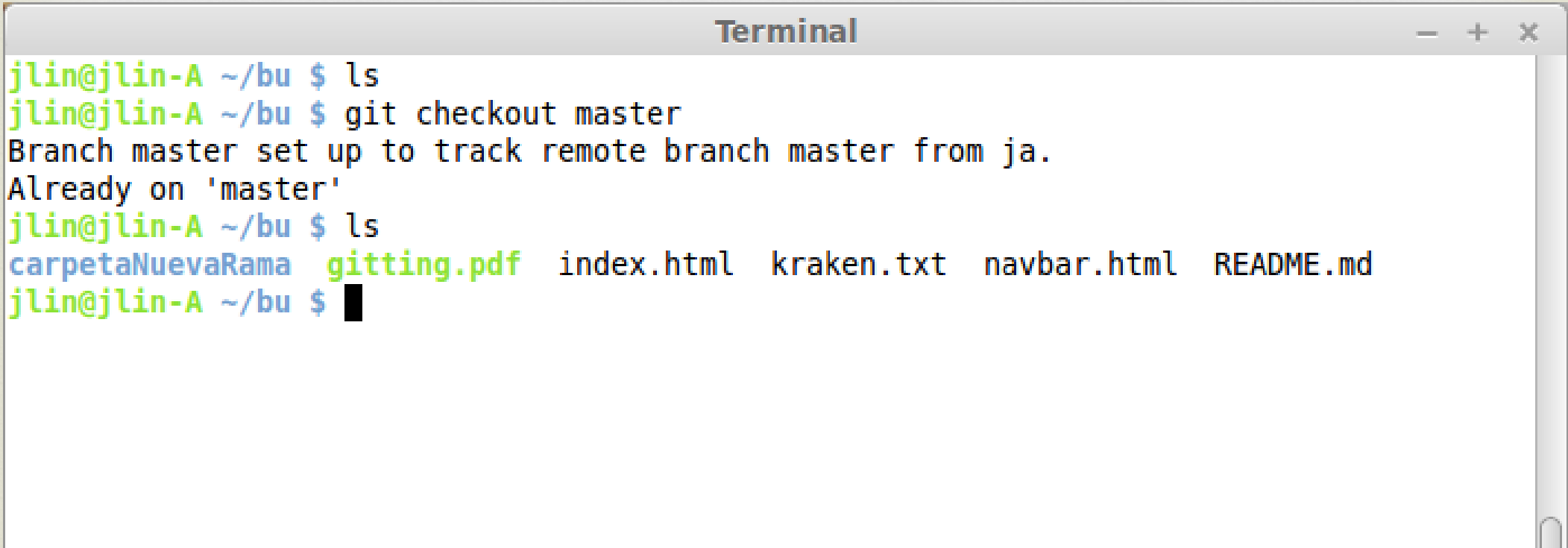
- Aquí vemos un ejemplo. Añadimos un repo remoto (remote add) y después lo descargamos con fetch.

Terminal

```
jlin@jlin-A ~ $ mkdir bu
jlin@jlin-A ~ $ cd bu
jlin@jlin-A ~/bu $ git init
Initialized empty Git repository in /home/jlin/bu/.git/
jlin@jlin-A ~/bu $ git remote add ja https://github.com/jacobos/bootstrap-examples
jlin@jlin-A ~/bu $ git fetch ja
remote: Counting objects: 44, done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 44 (delta 15), reused 43 (delta 14), pack-reused 0
Unpacking objects: 100% (44/44), done.
From https://github.com/jacobos/bootstrap-examples
 * [new branch]      master     -> ja/master
 * [new branch]      nuevaRama  -> ja/nuevaRama
jlin@jlin-A ~/bu $
```

Ejemplo de fetch

- Tras hacer el fetch con el checkout recuperamos la rama que nos interesa, antes está solamente alamacenado en la carpeta .git

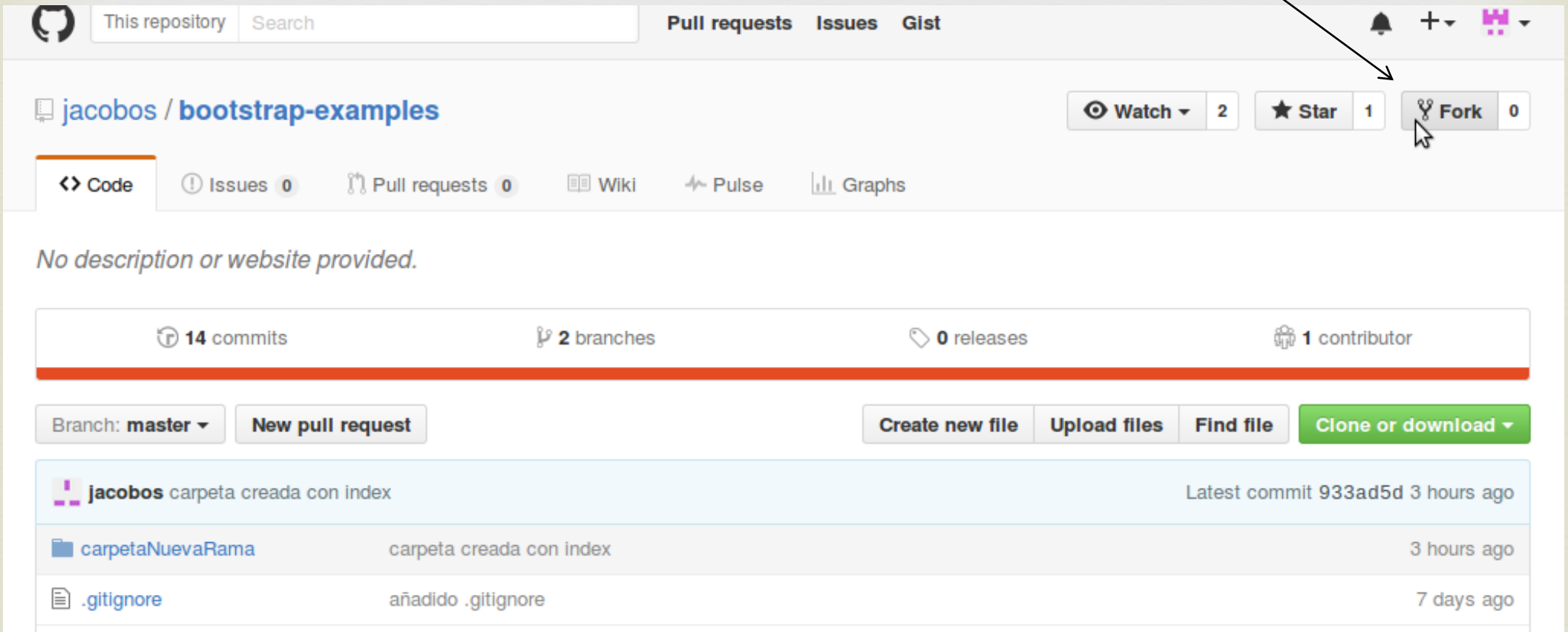


Forks

- Si queremos contribuir a un proyecto de github o bitbucket (esto no es parte de git) utilizaremos lo que se conoce como un fork.
- Un fork es una copia de un repositorio, tras hacerlo podemos compartir nuestros cambios (hacer peticiones de cambio) con las personas que administren el repositorio original (todas las branches).

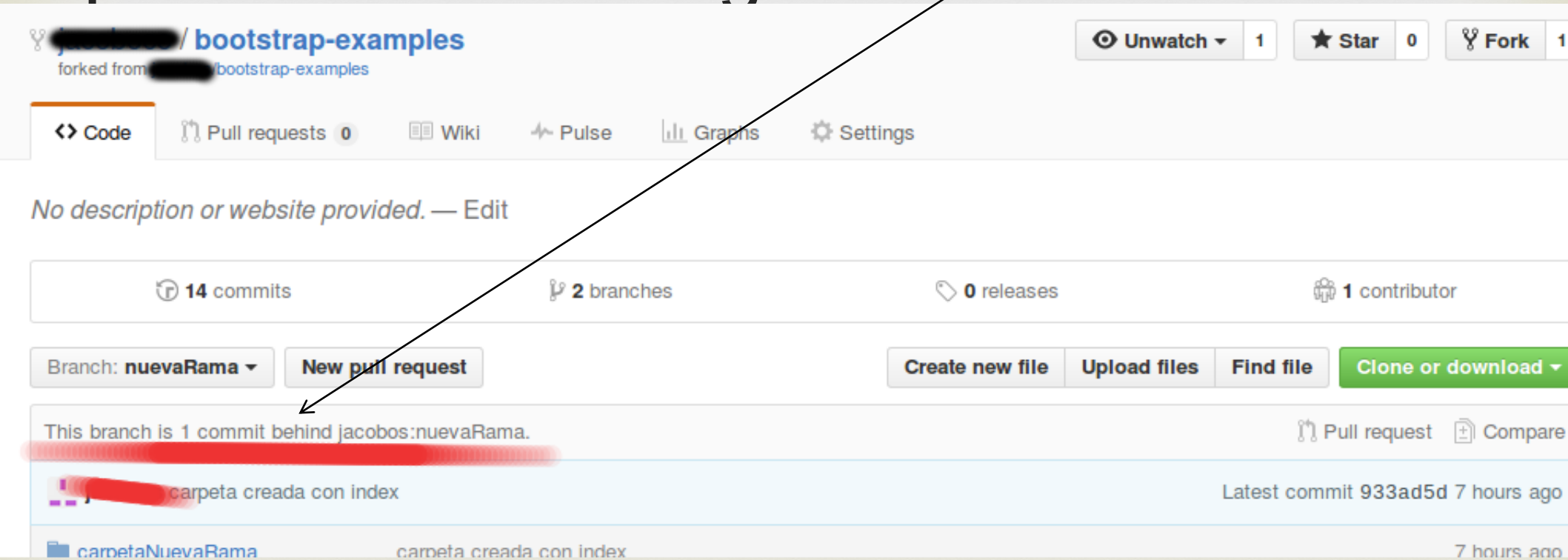
Haciendo un fork...

- Vas al proyecto en cuestión que te interese y le das al botón de fork

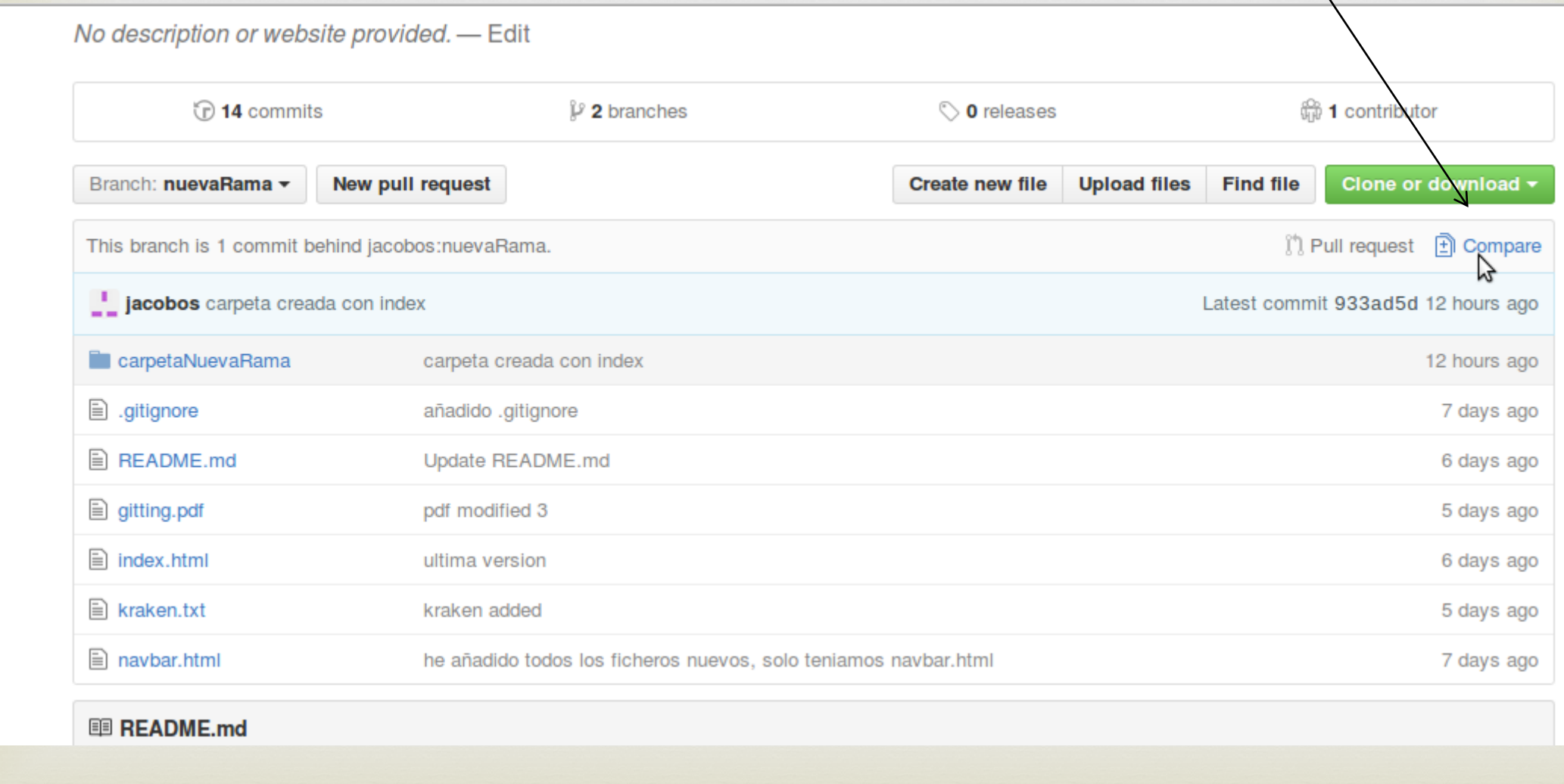


Pero quieres estar actualizado

- Hemos hecho un fork, y al mirarlo vemos que estamos un commit por detrás del original

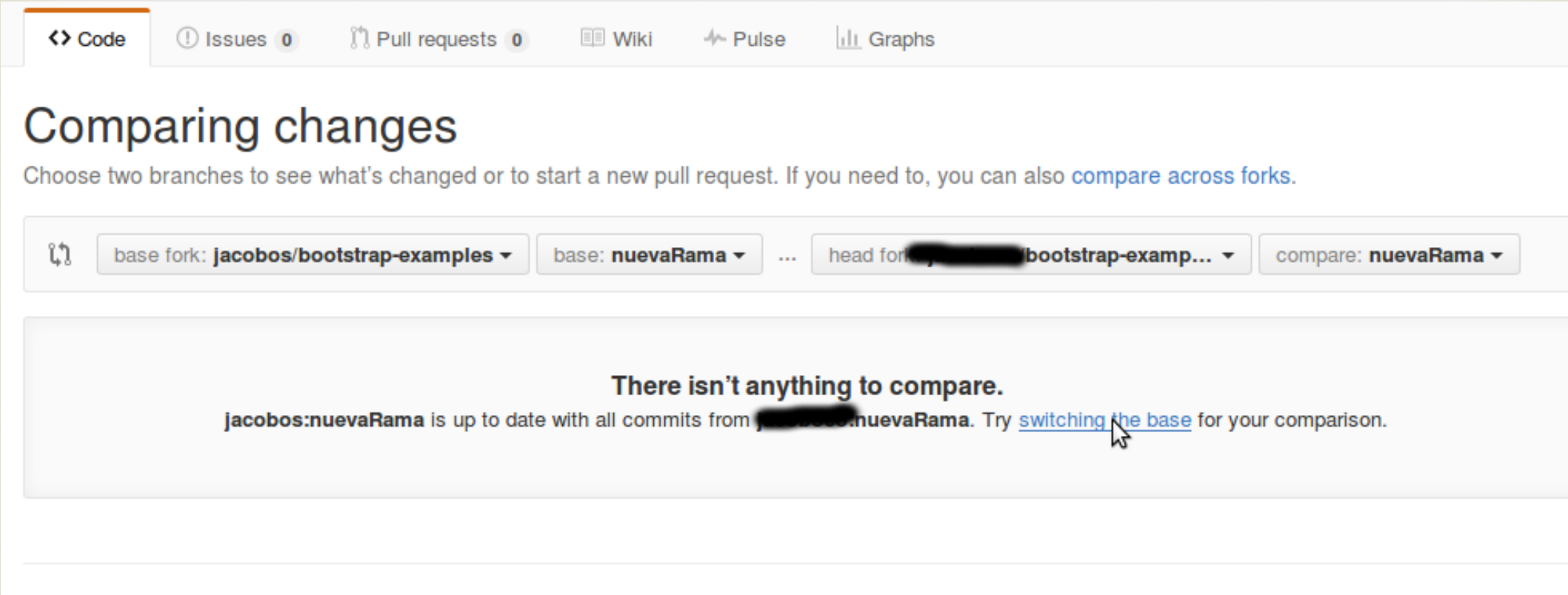


En github podemos usar compare

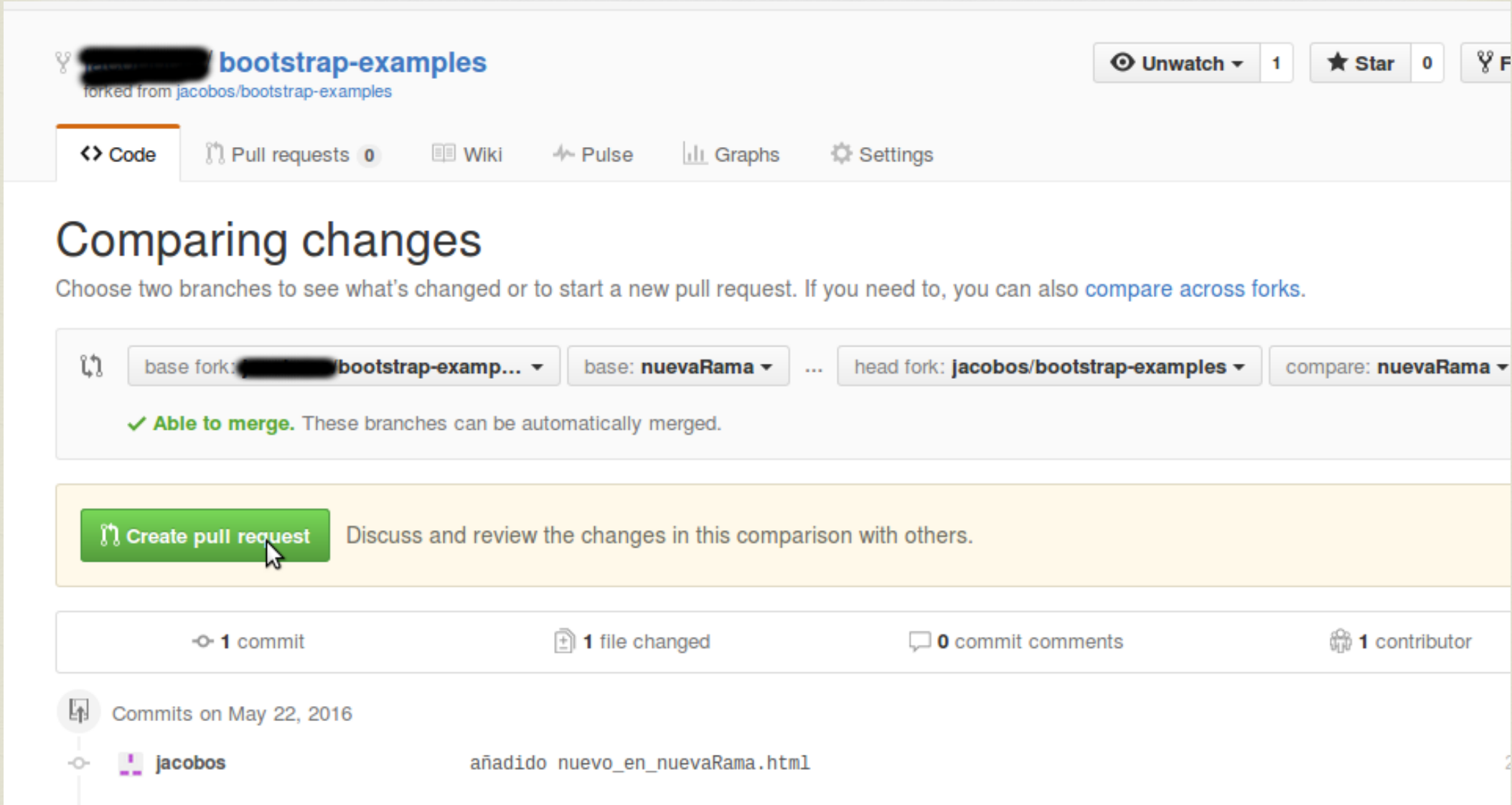


Switching the base

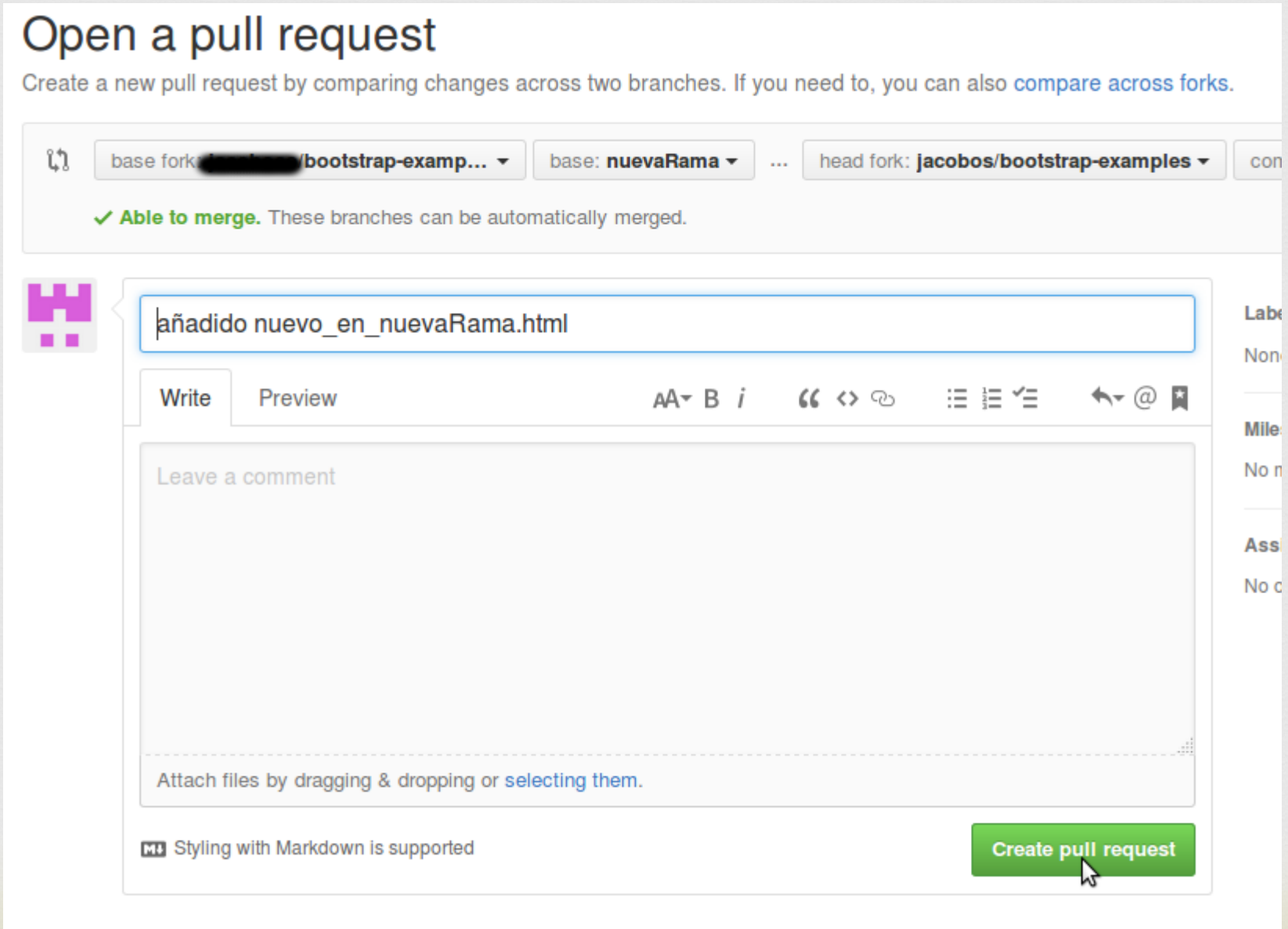
- Utilizando esta opción modifica el origen y el destino de la comparación (sabemos que algo esta cambiado aunque aquí no se vea)



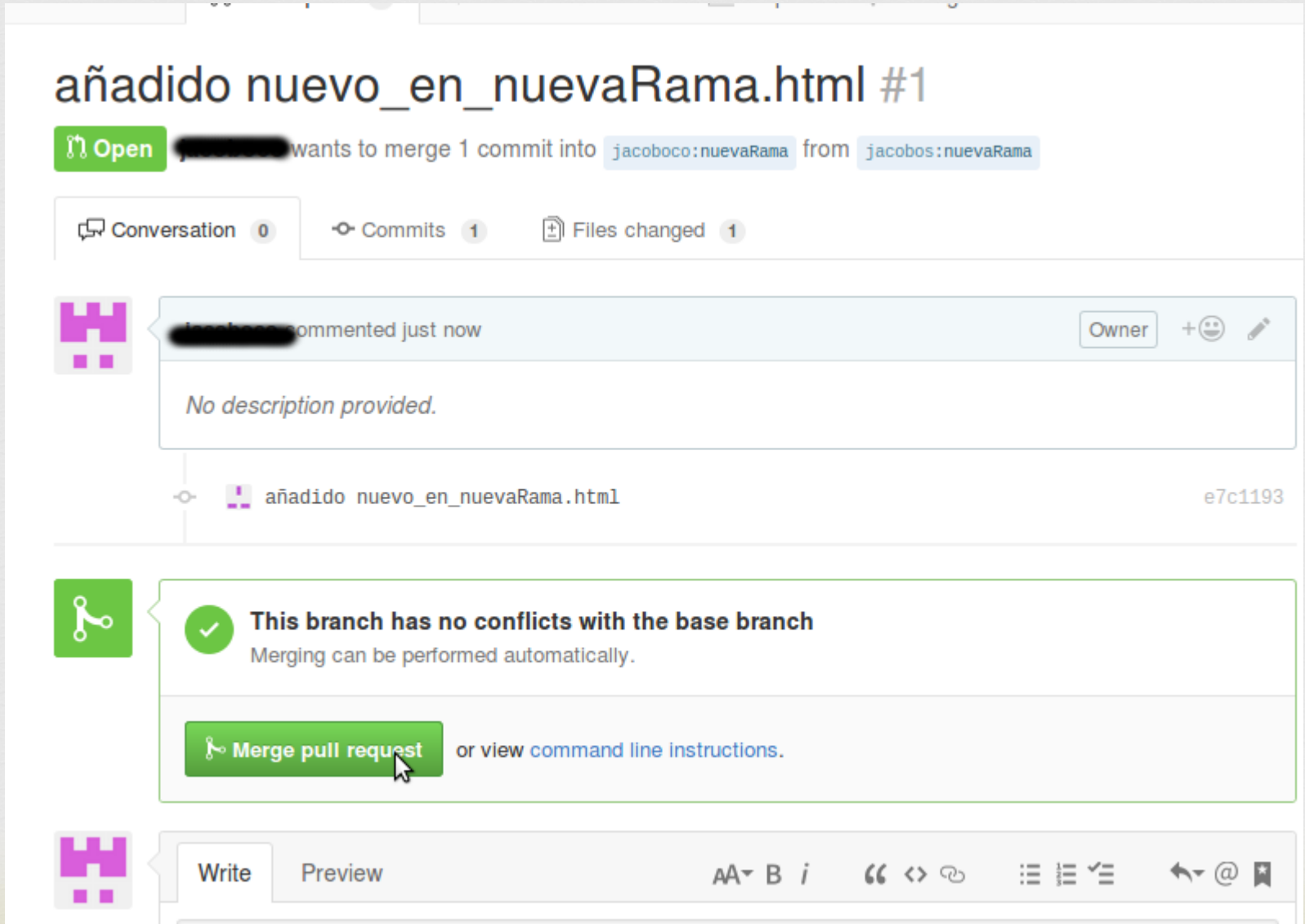
Create pull request (1)



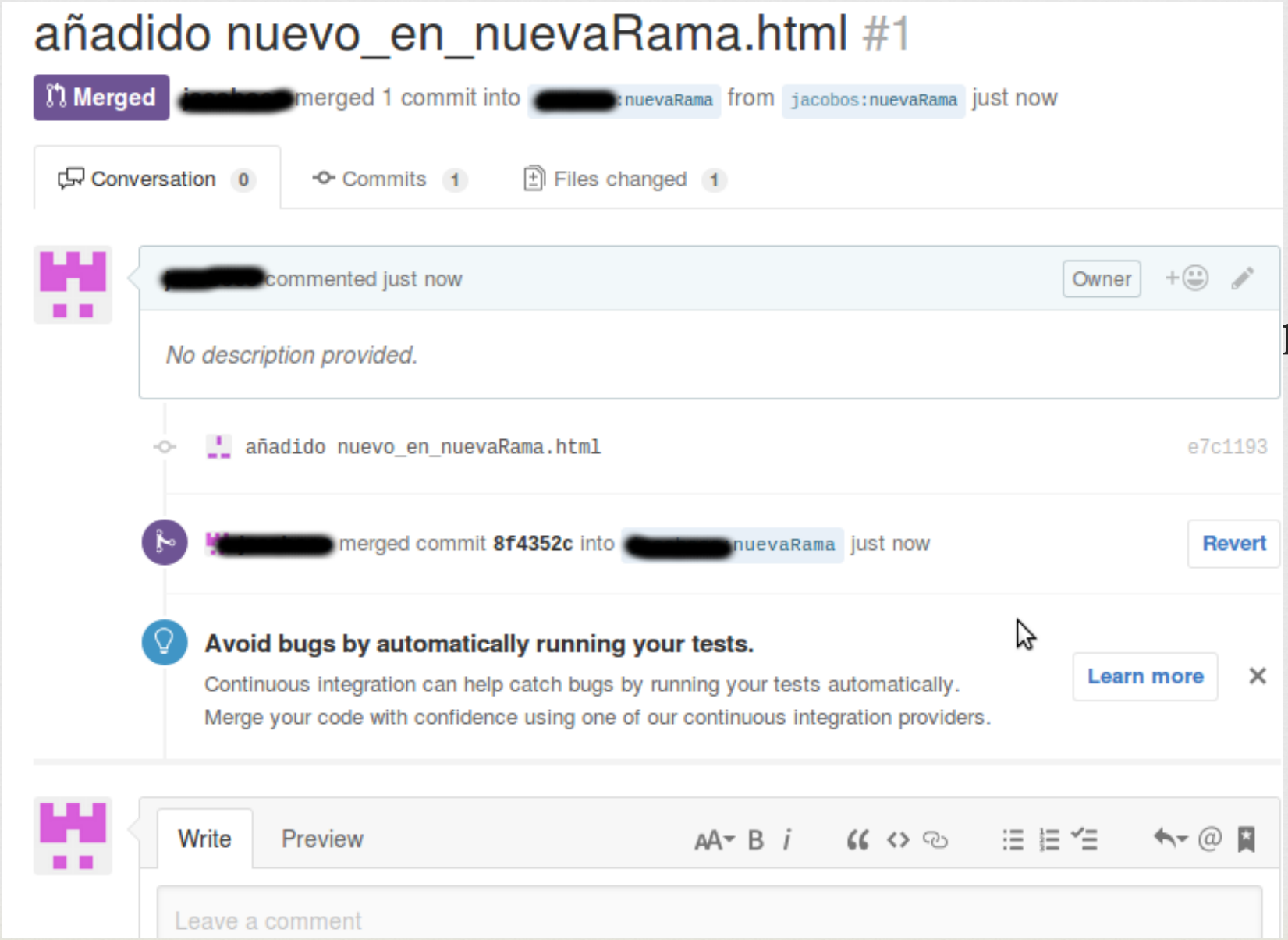
Create pull request (2)



Create pull request (3)



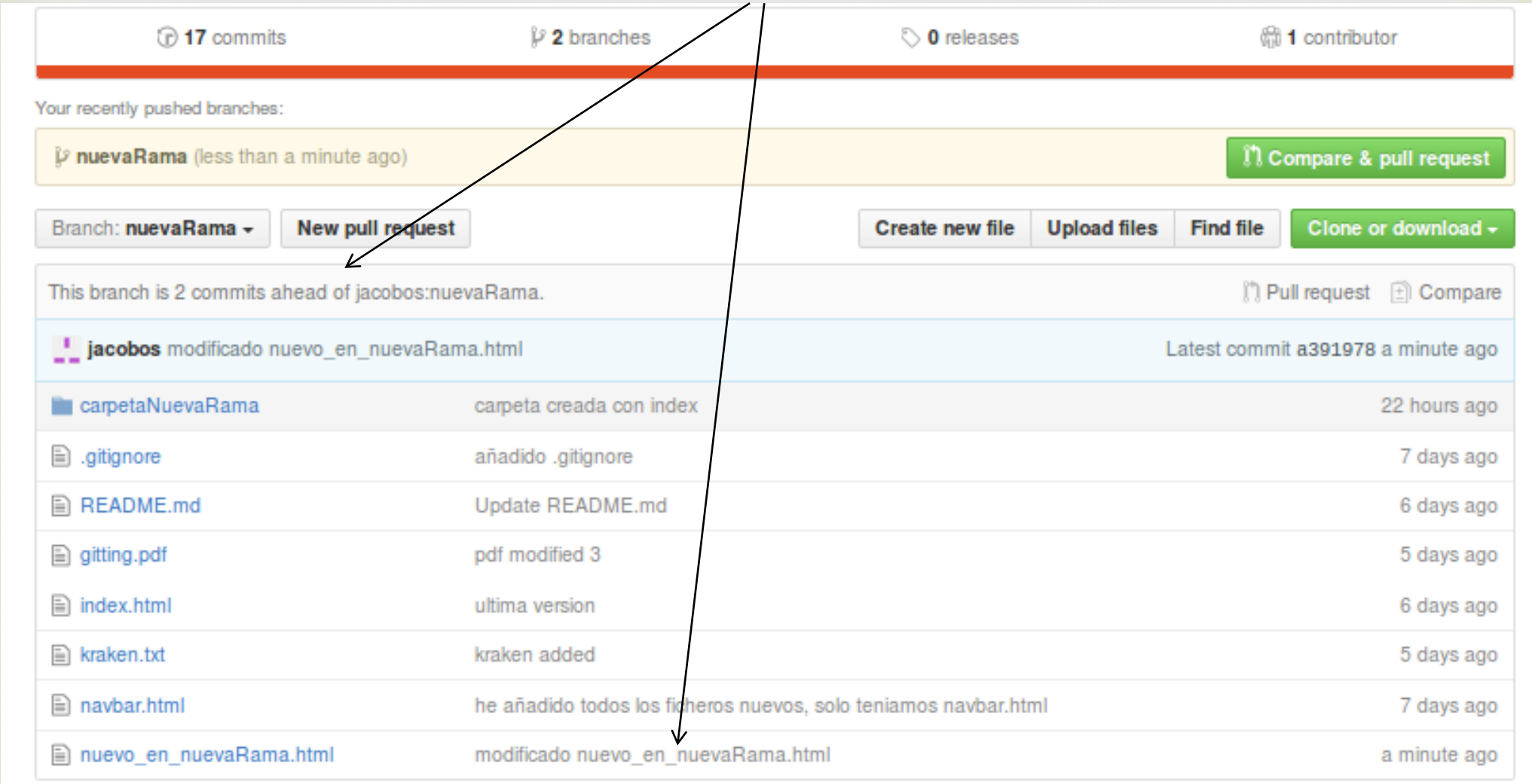
Create pull request (4)



n a la

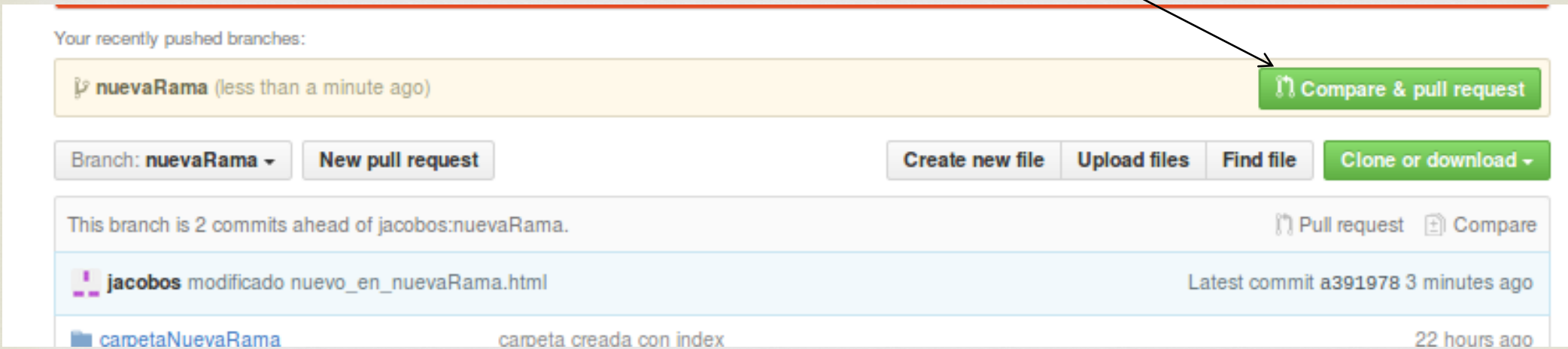
Queremos pedir que actualicen el nuevaRama original

Vemos que vamos por delante del nuevaRama original, dos commits por delante



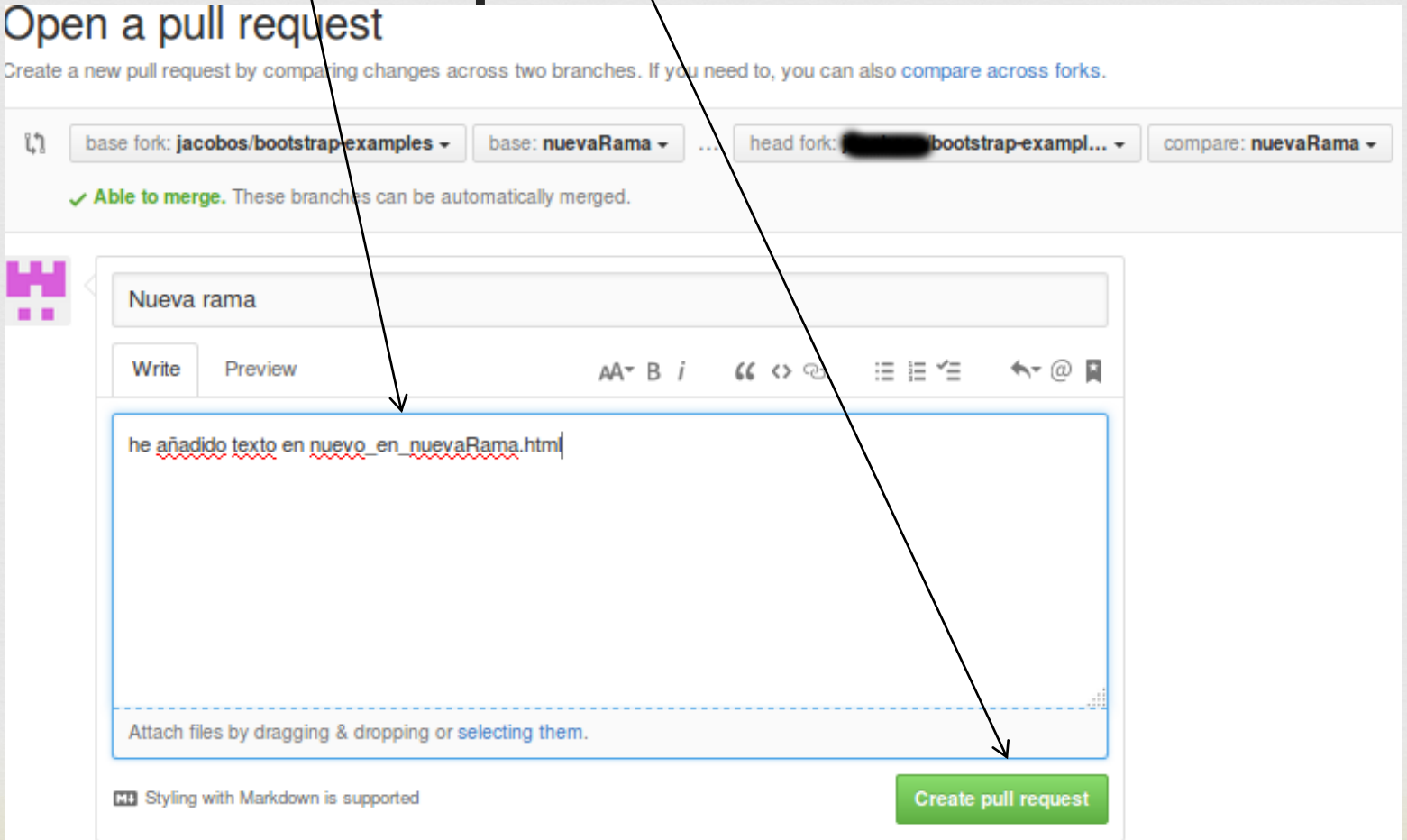
Compare & pull request

- Vamos a hacer un pull request

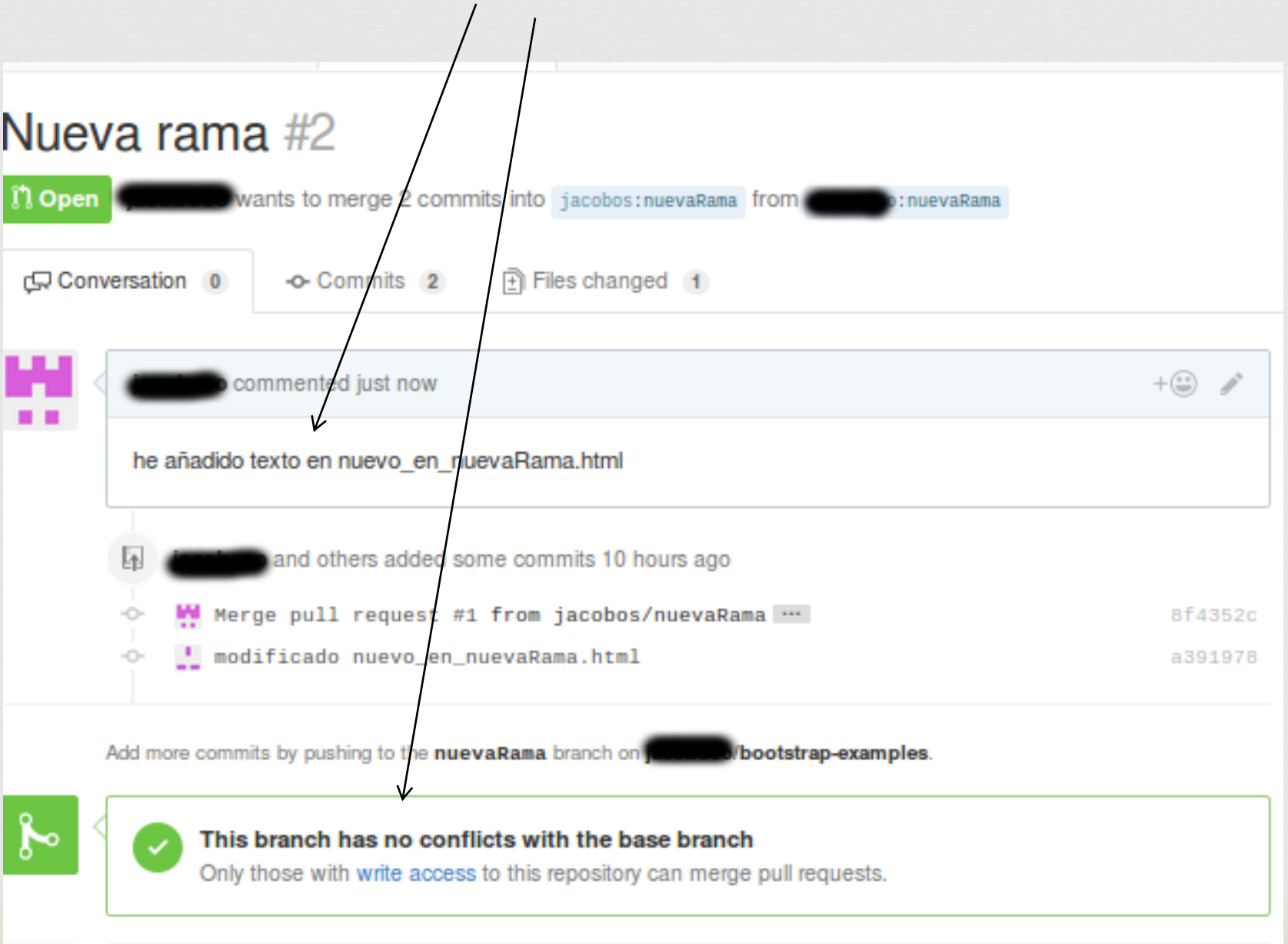


Create pull request

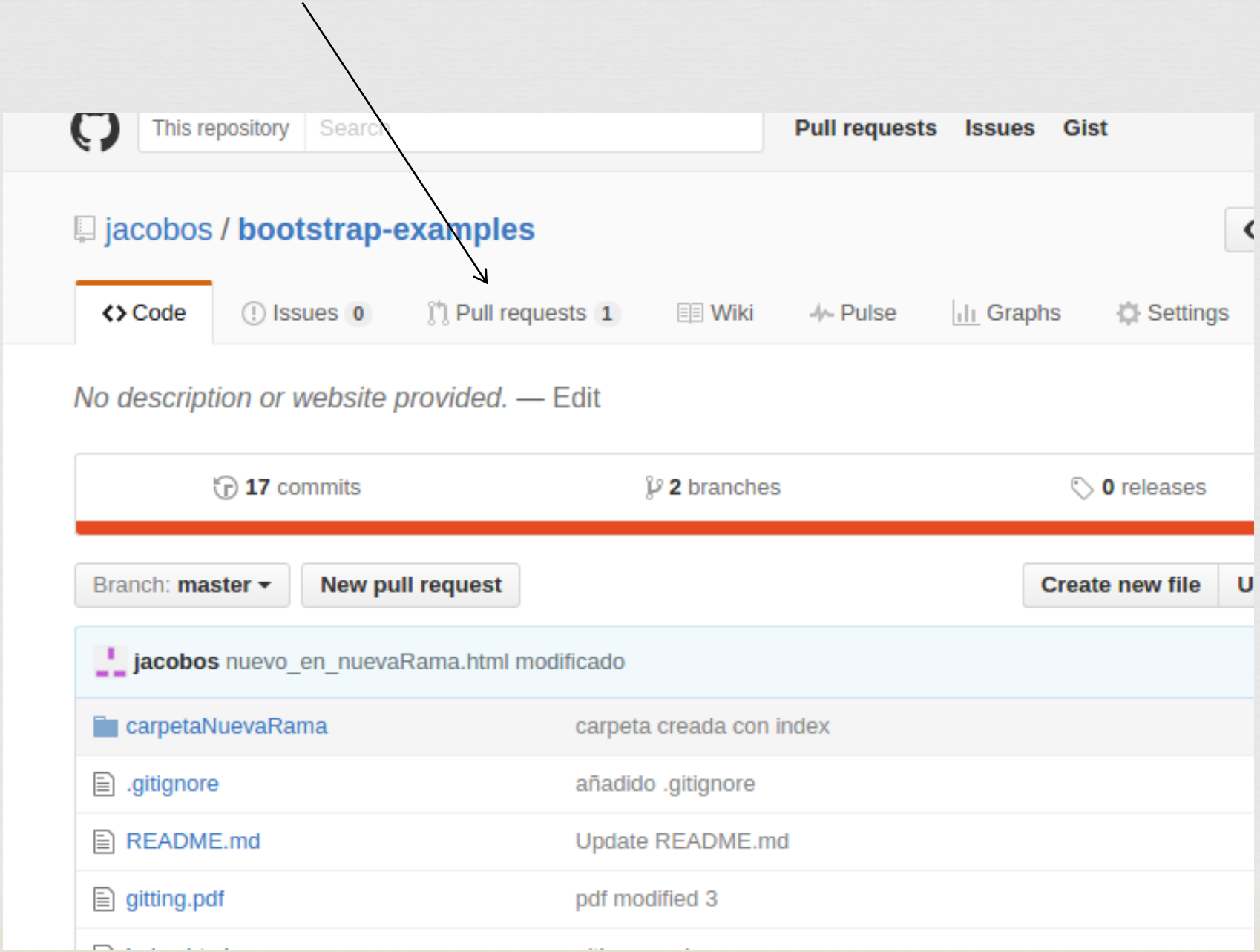
- Añadimos el texto que queramos y hacemos la petición



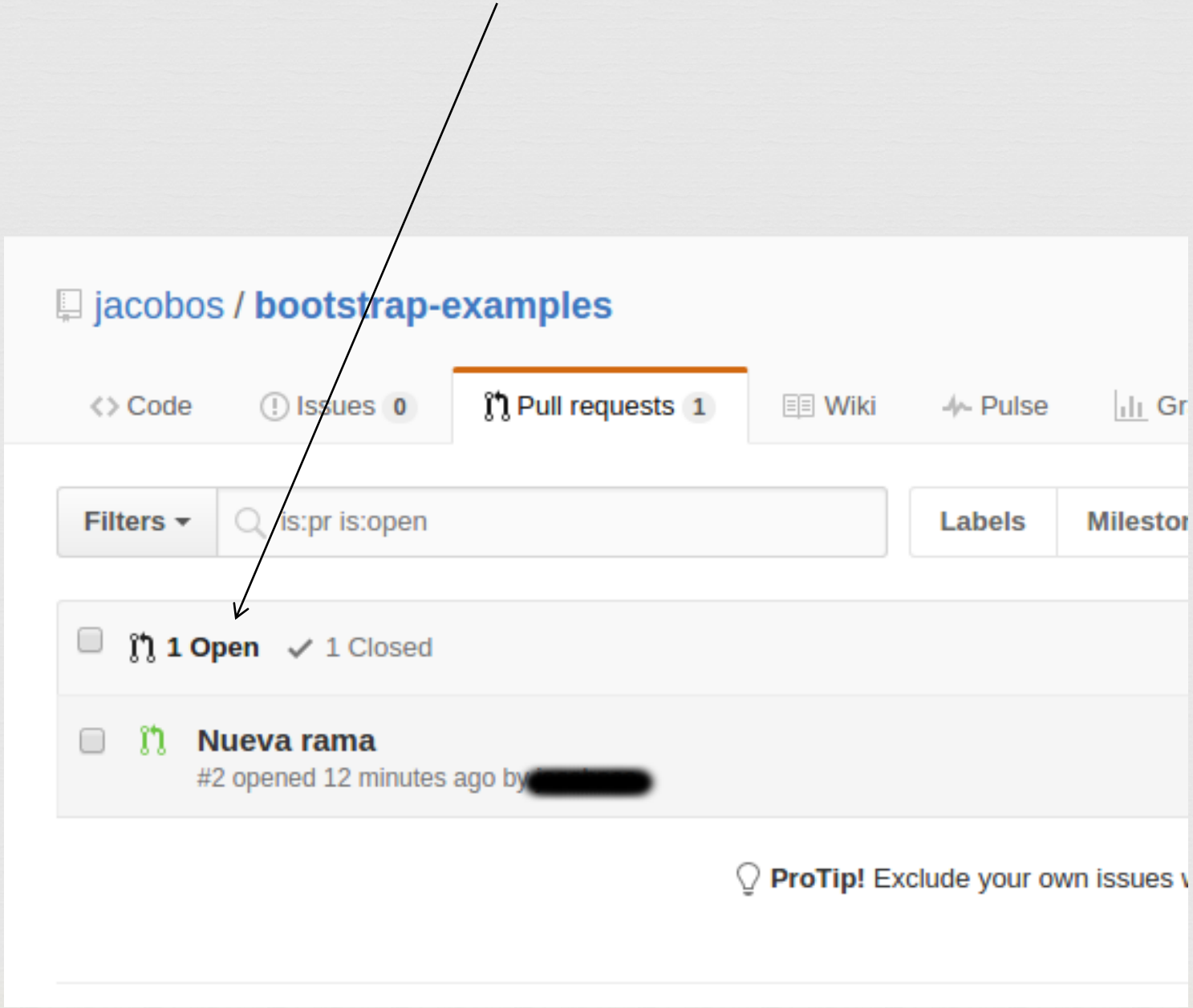
Vemos el resultado de la petición



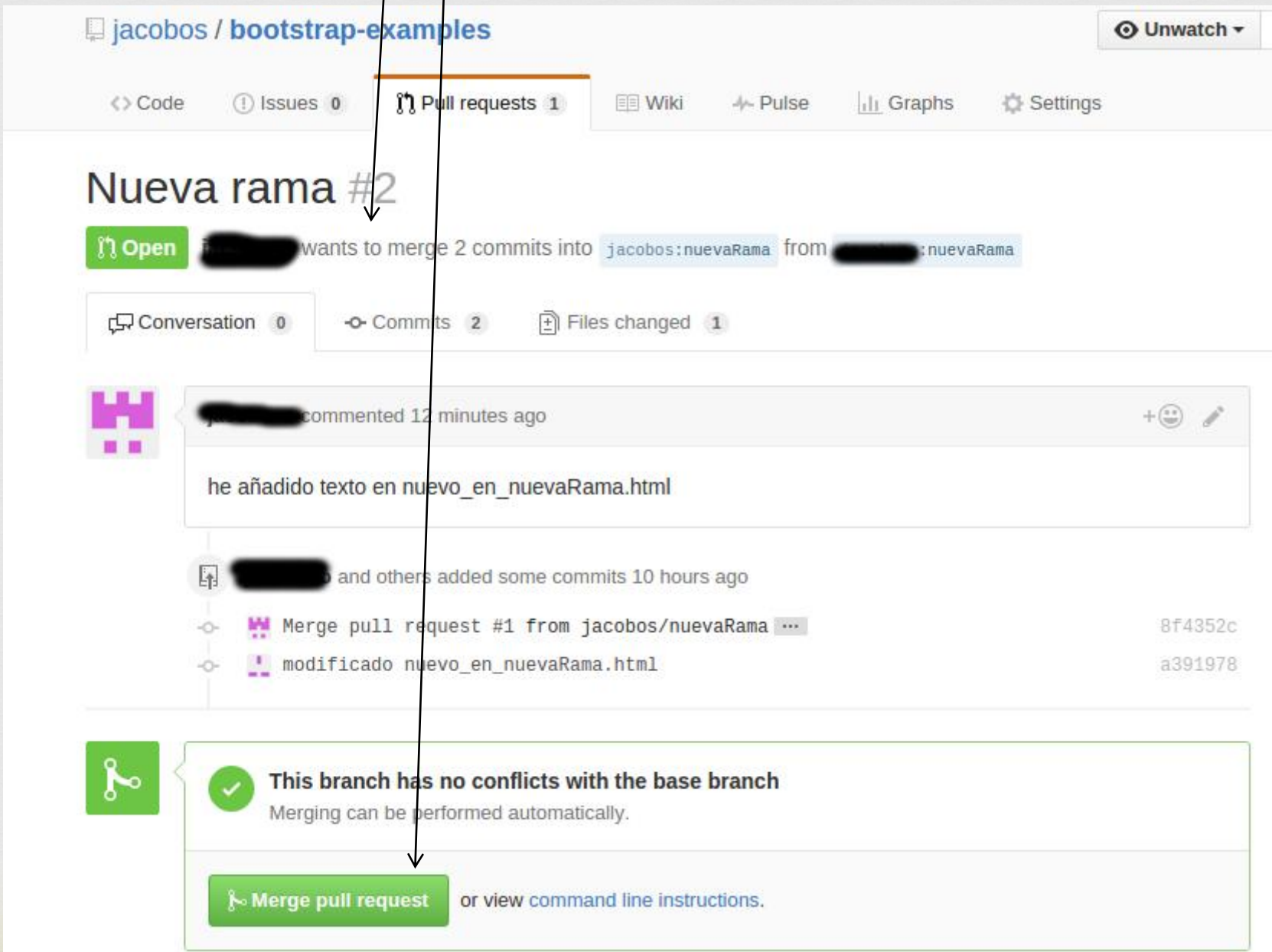
Y vemos las pull requests



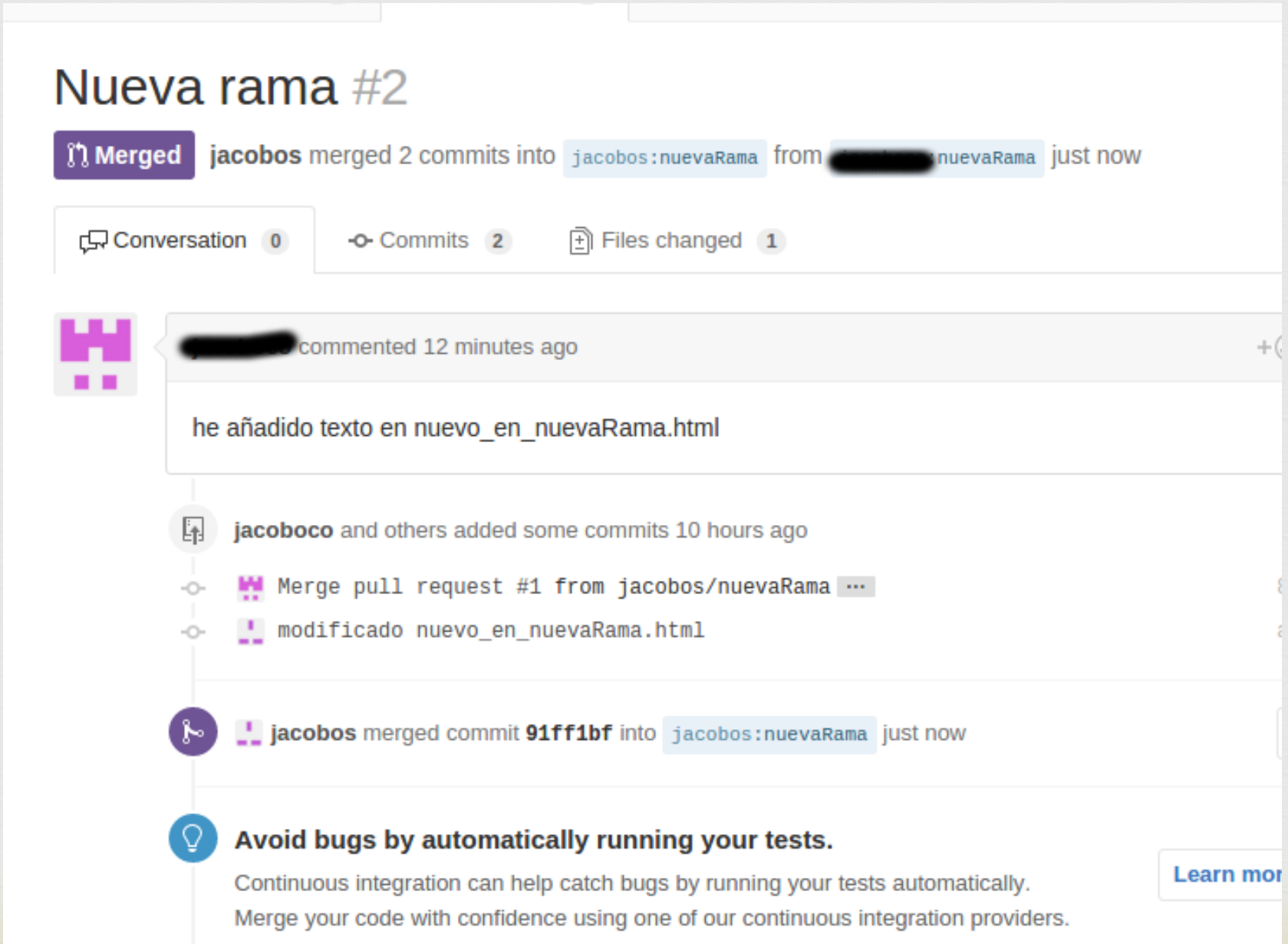
Y hay una abierta, por aceptar



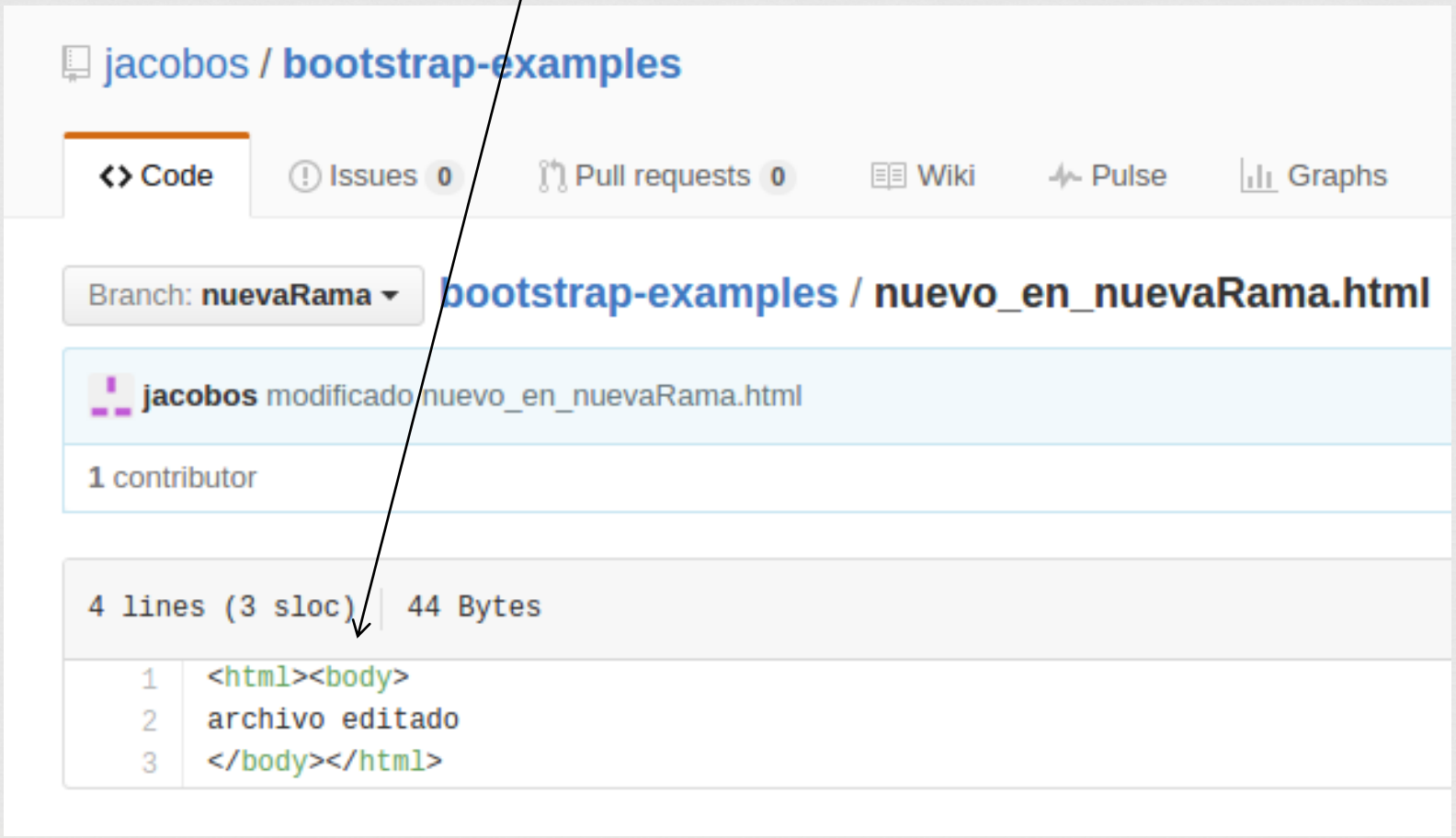
Vemos la petición del usuario



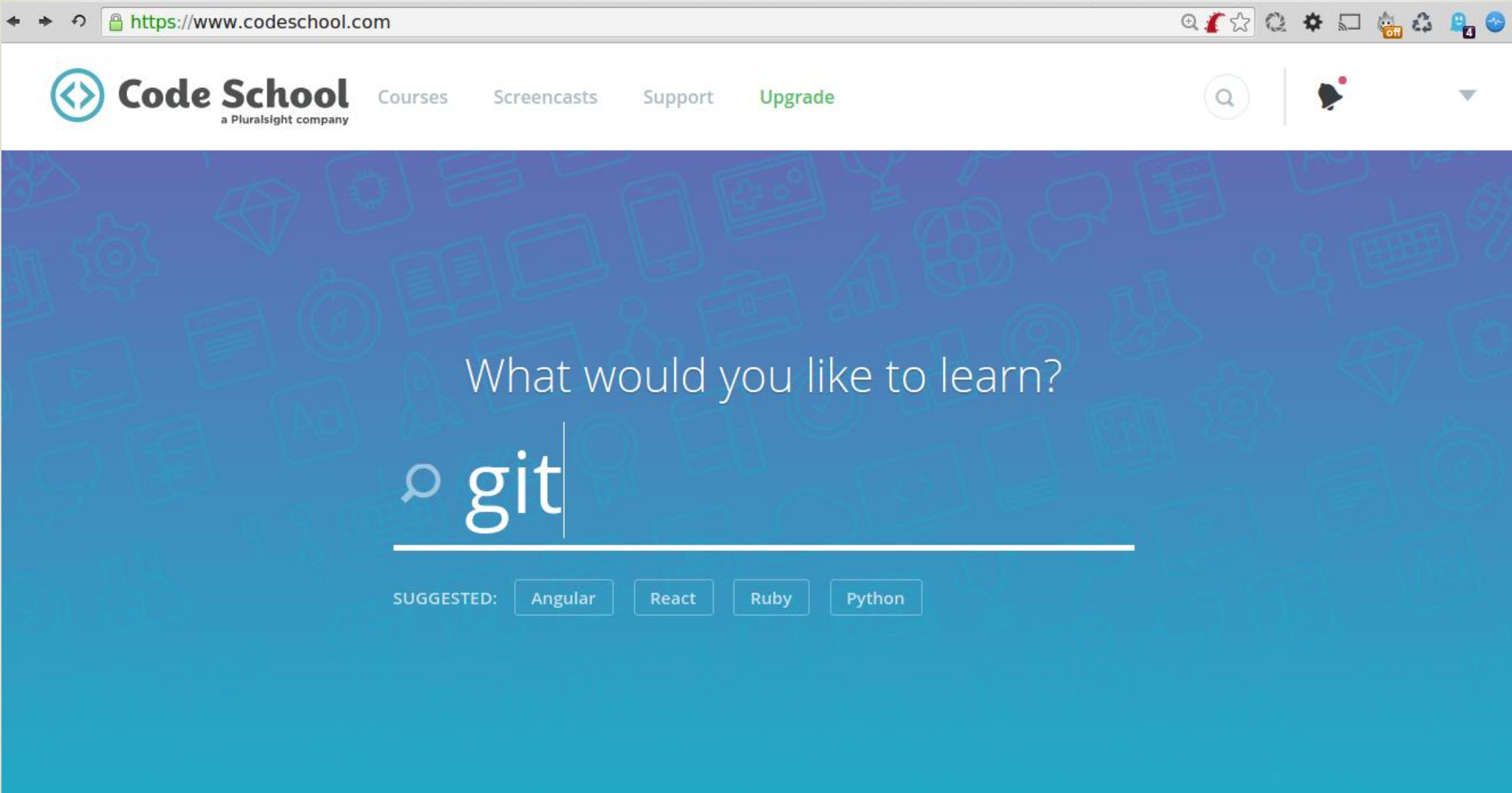
La aceptamos y se guarda



Y lo vemos...


jacobos / bootstrap-examples
Code
Issues 0
Pull requests 0
Wiki
Pulse
Graphs
Branch: nuevaRama
bootstrap-examples / nuevo_en_nuevaRama.html
jacobos modificado nuevo_en_nuevaRama.html
1 contributor
4 lines (3 sloc) 44 Bytes
1 <html><body>
2 archivo editado
3 </body></html>

un cursillo… (hay +)



Links

- Página oficial
- Set up git (github)
- Understanding git
- Understanding git(2)
- Git, guía sencilla
- Tutorial de git
- Git courses (codeschool)
- Bitbucket