
Git Cheat Sheet

By Stf

¿Qué es Git?

Git es un **sistema de control de versiones**.

Lo que significa que va a controlar los cambios en el desarrollo de cualquier tipo de software, permitiendo conocer el estado actual de un proyecto, los cambios que se le han realizado, las personas que intervinieron en ellos, etc.

Descarga: <https://git-scm.com/>

Configuración: Abrir “Cmd” o “Git Bash” (Windows) /
Terminal (Linux/Mac)

```
git config --global user.name "Estefania Aguilar" (Enter)
```

git config user.name (mostrará el nombre de usuario ingresado)

```
git config --global user.email correo@correo.com (Enter)
```

git config user.email (hará lo mismo con el mail)

Inicializando un repositorio en un directorio existente

Si empezás el seguimiento de Git en un proyecto existente, necesitás ir a su ubicación y escribir:

```
git init
```

Con este primer comando creamos un repositorio local Git en nuestro directorio que guardará los cambios de nuestro proyecto. Este subdirectorio .git, como se llama, no se ve pero está. Si presionan “Ctrl + h” en Linux o “Vista> Elementos ocultos” en Windows pueden ver para creer.

En adelante, cualquier cambio que se produzca en los archivos del directorio, GIT lo notará pero no hará seguimiento hasta que se lo indiqués explícitamente.

Nuestra herramienta para saber en qué estado están los archivos es el comando `git status`

Los comandos en acción:

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git init
Initialized empty Git repository in /home/estefania/Escritorio/proyecto/.git/
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ touch index.html
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git status
En la rama master
```

Commit inicial

Archivos sin seguimiento:

(use «git add <archivo>...» para incluir en lo que se ha de confirmar)

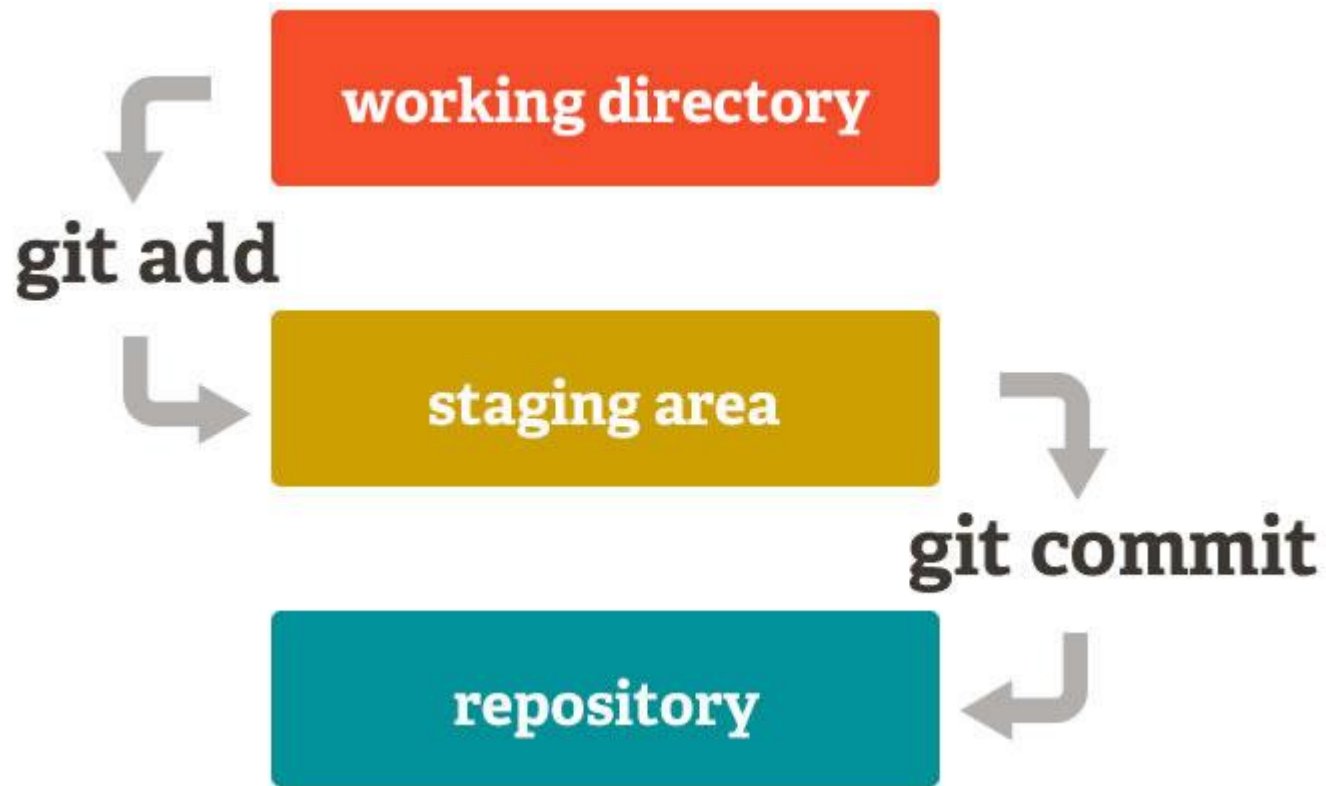
index.html

no se ha agregado nada al commit pero existen archivos sin seguimiento (use «git add» para darle seguimiento)

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ █
```

El flujo de trabajo básico en Git consiste en:

1. Modificar los archivos en el Directorio de trabajo.
 2. Preparar los archivos, añadiéndolos al área de preparación.
 3. Confirmar los cambios, para esto necesita que los archivos estén en el área de preparación, y almacena una versión permanente en el directorio de Git.
-



Como vimos en la secuencia, todos los cambios que hagamos en nuestro directorio deberán ser “agregados” al Staging (o área de preparación) de GIT:

```
git add [filename] o git add .
```

Aclaración: “git add” es un comando multiuso, sirve tanto para agregar los archivos por primera vez al Stage como para agregar sus cambios.

Y luego confirmas los cambios, lo que toma los archivos tal y como están en el área de preparación y almacena esa copia instantánea de manera permanente en el repositorio de GIT con

```
git commit -m “Commit message”
```

Archivos sin seguimiento:

(use «git add <archivo>...» para incluir en lo que se ha de confirmar)

index.html

no se ha agregado nada al commit pero existen archivos sin seguimiento (use «git add» para darle seguimiento)

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git add index.html
```

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git status
```

En la rama master

Commit inicial

Cambios para hacer commit:

(use «git rm --cached <archivo>...» para sacar del stage)

nuevo archivo: index.html

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git commit -m "agregamos index.html"
```

```
[master (root-commit) 24e813e] agregamos index.html
```

```
1 file changed, 0 insertions(+), 0 deletions(-)
```

```
create mode 100644 index.html
```

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ █
```

En adelante, todos los cambios que se le hagan a los archivos “stageados” aparecerán en el estado “modified” con `git status`.

Para agregar esos cambios sólo debemos repetir `git add` y, si con ellos queremos hacer una versión del proyecto, `git commit`.

¿Cómo eliminar archivos del Staging area?

```
git rm --cached [filename]
```

¿Qué pasa si tengo archivos que no quiero incluir en el versionado?

Muy simple, creamos un archivo “.gitignore” que contenga los nombres de aquellos archivos que no queremos incluir.

Veamos cómo hacerlo...

The screenshot shows the Sublime Text editor interface. The title bar indicates the file path is `~/Escritorio/proyecto/.gitignore (proyecto) - Sublime Text (UNREGISTERED)`. The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. On the left, the 'FOLDERS' sidebar shows a project named 'proyecto' containing files `.gitignore`, `index.html`, and `readme.md`. The main editor area has three tabs: `index.html`, `readme.md`, and `.gitignore`. The `.gitignore` tab is active, showing the following content:

```
1 *.psd
2 readme.md
```

The status bar at the bottom of the editor shows 'Line 2, Column 10', 'Tab Size: 4', and 'Plain Text'. Below the editor, a terminal window displays the output of the `git status` command:

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git status
En la rama master
Archivos sin seguimiento:
  (use «git add <archivo>...» para incluir en lo que se ha de confirmar)

    .gitignore

no se ha agregado nada al commit pero existen archivos sin seguimiento (use «git
add» para darle seguimiento)
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$
```

Aclaración: con `*.psd` indicamos que no incluya TODOS los archivos con extensión psd (algo frecuente a la hora de diseñar sobre un layout).

Como se ve en el ejemplo, el archivo “readme.md” existe en el directorio (al igual que los psd), pero Git los ignora al tener sus nombres en tal archivo.

Commits!

👉 Ver historial de commits: numeración, autor, correo, fecha, hora y mensaje:

```
git log
```

👉 Ver historial de commits: numeración, autor, correo, fecha, hora, mensaje, archivos incluidos y tipo de cambios de cada uno (líneas borradas y/o añadidas):

```
git log --stat
```

👉 Ver historial de commits: numeración (reducida) y mensaje:

```
git log --oneline
```


👉 Ver los cambios recientes de un archivo con respecto al último commit:

```
git diff
```

👉 Ver los cambios recientes de un archivo con respecto a cualquier commit:

```
git diff numeración [filename]
```

👉 Ver los cambios entre commits:

```
git diff numeración numeración
```

 Regresar a una versión anterior:

git checkout numeración

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git log --oneline
382392b sumamos parrafos
30568dc estructura básica de html
24e813e agregamos index.html
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git checkout 30568dc
Note: checking out '30568dc'.
```

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using `-b` with the checkout command again. Example:

```
git checkout -b <new-branch-name>
```

```
HEAD se encuentra en 30568dc... estructura básica de html
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git log --oneline
30568dc estructura básica de html
24e813e agregamos index.html
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ █
```

Ramas!

Una rama es una copia idéntica de nuestro proyecto donde podemos probar nuevas funcionalidades en nuestro código sin afectar a la versión original.

Las ramas posibilitan el desarrollo colaborativo.

🔍 Ver todas las ramas que tenemos:

git branch

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git log --oneline
30568dc estructura básica de html
24e813e agregamos index.html
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git branch
* (HEAD detached at 30568dc)
  master
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$
```

En nuestro ejemplo habíamos regresado a un commit pasado, entonces aquí al ejecutar “git branch” nos muestra dos ramas: la rama master, que es la rama principal, y una nueva rama (donde estamos ahora) con una copia del código de la versión 30568dc.

 Cambiar de rama:

git checkout [nombre_rama]

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git branch
* (HEAD detached at 30568dc)
  master
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git checkout master
Previous HEAD position was 30568dc... estructura básica de html
Switched to branch 'master'
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git branch
* master
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$
```

Si volvemos a ejecutar “git log” reaparecerá el commit “sumamos párrafos” porque estamos en la versión original. Pruébenlo!

Todavía no está claro, vamos con otro ejemplo...

 Crear rama:

```
git branch [nombre_rama]
```

ó

```
git checkout -b [nombre_rama]
```

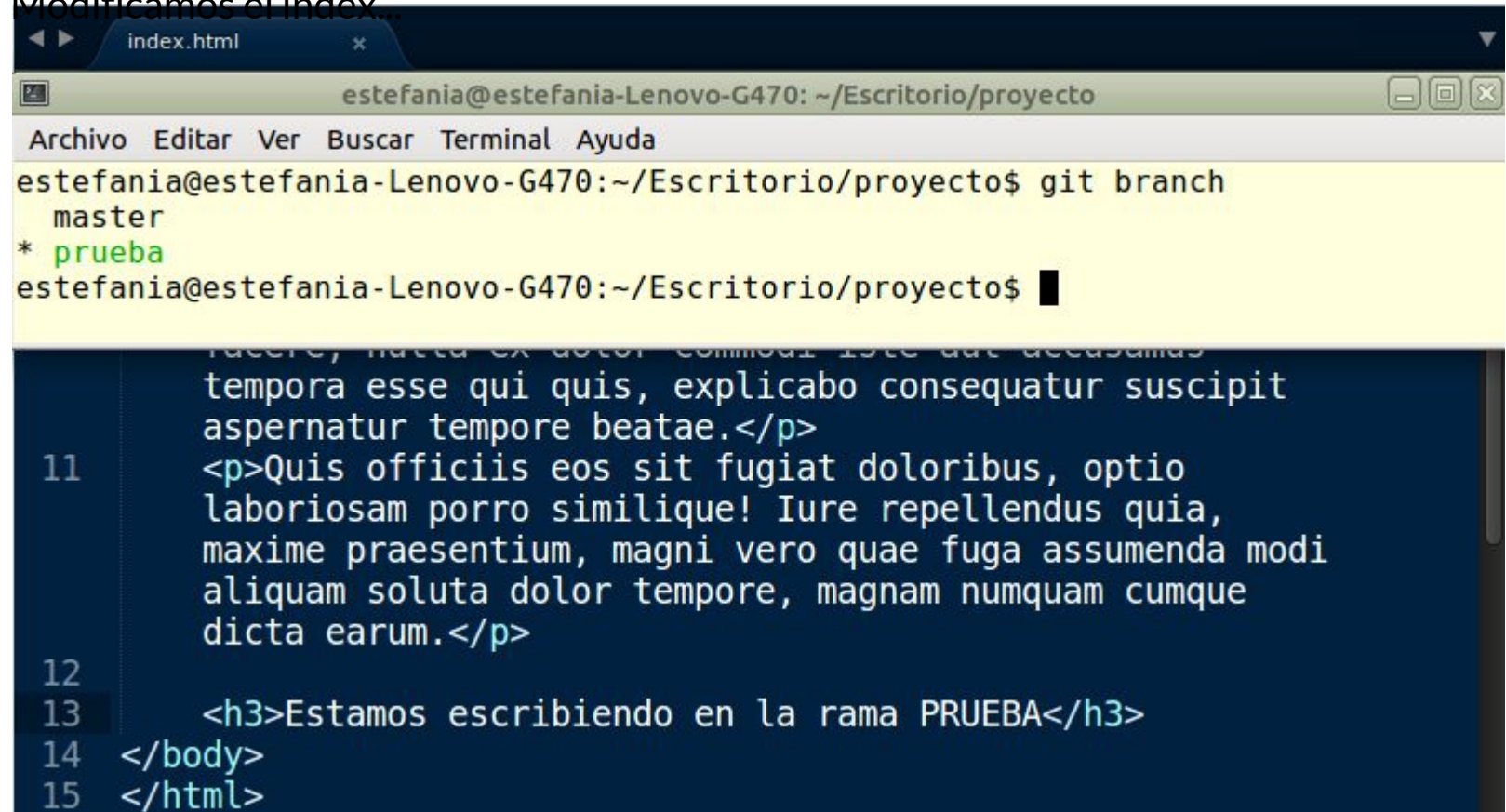
(creamos rama y nos posicionamos directamente ahí)

 Eliminar rama:

```
git branch -d [nombre_rama]
```


Previamente creamos la rama “Prueba” y nos ubicamos ahí con los comandos anteriores.

Modificamos el index...



The image shows a web browser window at the top with the tab 'index.html' and a terminal window below it. The terminal window title is 'estefania@estefania-Lenovo-G470: ~/Escritorio/proyecto'. The terminal menu bar includes 'Archivo', 'Editar', 'Ver', 'Buscar', 'Terminal', and 'Ayuda'. The terminal output shows the command 'git branch' being executed, listing 'master' and the current branch '* prueba'. Below this, the terminal shows the editing of 'index.html' with line numbers 11 through 15. The content of the file includes HTML tags for paragraphs and a heading.

```
estefania@estefania-Lenovo-G470: ~/Escritorio/proyecto$ git branch
master
* prueba
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$
```

```
11      tempora esse qui quis, explicabo consequatur suscipit
      aspernatur tempore beatae.</p>
12
13      <p>Quis officiis eos sit fugiat doloribus, optio
      laboriosam porro similique! Iure repellendus quia,
      maxime praesentium, magni vero quae fuga assumenda modi
      aliquam soluta dolor tempore, magnam numquam cumque
      dicta earum.</p>
14
15      <h3>Estamos escribiendo en la rama PRUEBA</h3>
16
17  </body>
18
19  </html>
```

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git status
```

En la rama prueba

Cambios no preparados para el commit:

(use «git add <archivo>...» para actualizar lo que se confirmará)

(use «git checkout -- <archivo>...» para descartar cambios en el directorio de trabajo)

modificado: index.html

no hay cambios agregados al commit (use «git add» o «git commit -a»)

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git add index.html
```

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git commit -m "cambios en PRUEBA"
```

[prueba 9991e13] cambios en PRUEBA

1 file changed, 2 insertions(+)

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ █
```

Incluso podemos generar nuevos archivos!

```
[prueba 9991e13] cambios en PRUEBA
```

```
1 file changed, 2 insertions(+)
```

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ touch prueba.html
```

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git status
```

En la rama prueba

Archivos sin seguimiento:

(use «git add <archivo>...» para incluir en lo que se ha de confirmar)

prueba.html

no se ha agregado nada al commit pero existen archivos sin seguimiento (use «git add» para darle seguimiento)

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git add prueba.html
```

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git commit -m "nuevo archivo prueba.html"
```

```
[prueba 4f8c8ea] nuevo archivo prueba.html
```

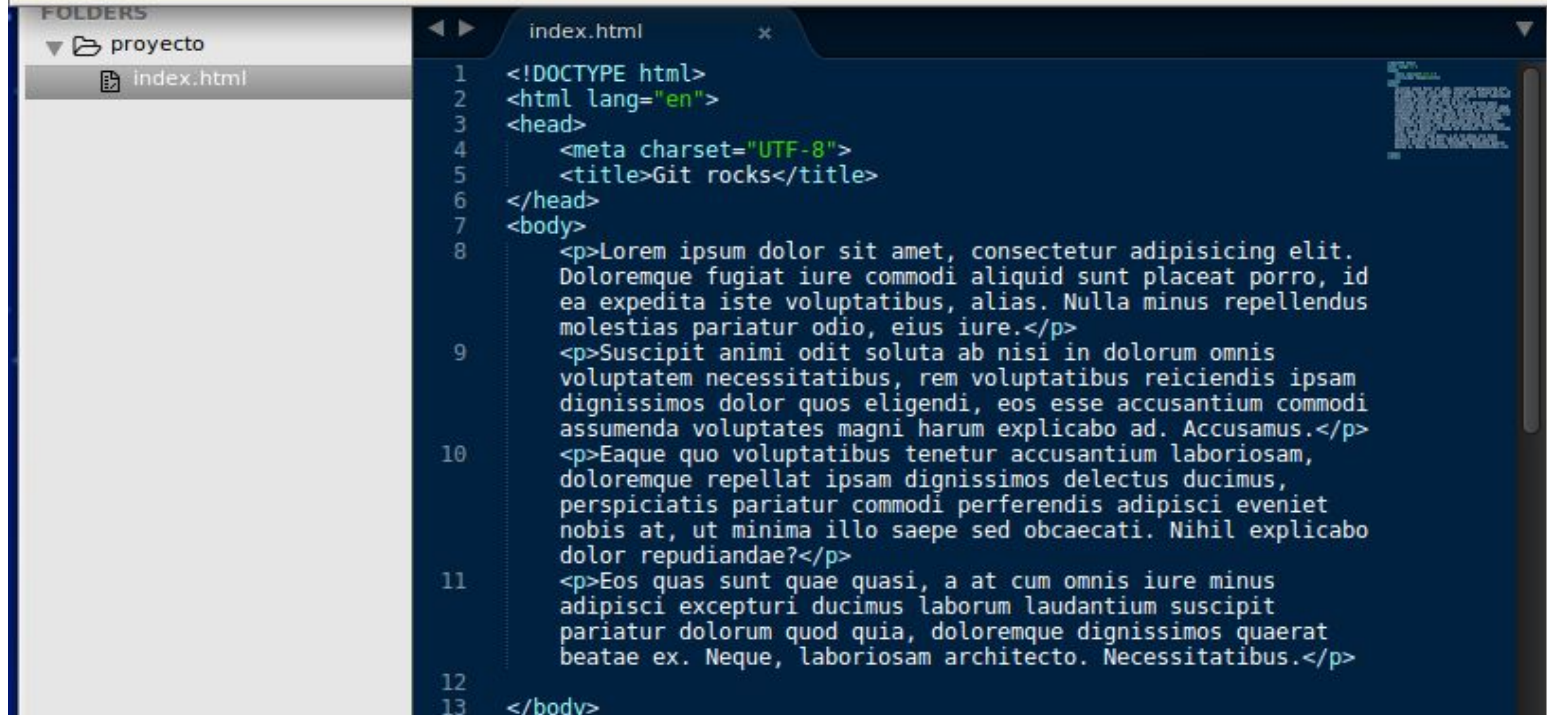
```
1 file changed, 0 insertions(+), 0 deletions(-)
```

```
create mode 100644 prueba.html
```

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$
```

¿Qué pasará si volvemos a MASTER?

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git checkout master
Switched to branch 'master'
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$
```



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Git rocks</title>
6 </head>
7 <body>
8   <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.
  Doloremque fugiat iure commodi aliquid sunt placeat porro, id
  ea expedita iste voluptatibus, alias. Nulla minus repellendus
  molestias pariatur odio, eius iure.</p>
9   <p>Suscipit animi odit soluta ab nisi in dolorum omnis
  voluptatem necessitatibus, rem voluptatibus reiciendis ipsam
  dignissimos dolor quos eligendi, eos esse accusantium commodi
  assumenda voluptates magni harum explicabo ad. Accusamus.</p>
10  <p>Eaque quo voluptatibus tenetur accusantium laboriosam,
  doloremque repellat ipsam dignissimos delectus ducimus,
  perspiciatis pariatur commodi perferendis adipisci eveniet
  nobis at, ut minima illo saepe sed obcaecati. Nihil explicabo
  dolor repudiandae?</p>
11  <p>Eos quas sunt quae quasi, a at cum omnis iure minus
  adipisci excepturi ducimus laborum laudantium suscipit
  pariatur dolorum quod quia, doloremque dignissimos quaerat
  beatae ex. Neque, laboriosam architecto. Necessitatibus.</p>
12
13 </body>
```


Tanto el <h3> del index como el archivo “prueba.html” se esfumaron de mi editor de código ya que recuerden que fueron cambios hechos en la rama prueba.

Entonces...

¿Qué pasa si yo esos cambios los quiero unir en mi versión original (master)?

```
git merge [rama_con_los_cambios]
```

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git branch
* master
  prueba
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git merge prueba
Updating eb3e693..4f8c8ea
Fast-forward
 index.html | 2 ++
 prueba.html | 0
 2 files changed, 2 insertions(+)
 create mode 100644 prueba.html
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$
```

| | |
|-------------|--|
| index.html | 1 <!DOCTYPE html> |
| prueba.html | 2 <html lang="en"> |
| | 3 <head> |
| | 4 <meta charset="UTF-8"> |
| | 5 <title>Git rocks</title> |
| | 6 </head> |
| | 7 <body> |
| | 8 <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Doloremque fugiat iure commodi aliquid sunt placeat porro, id ea expedita iste voluptatibus, alias. Nulla minus repellendus molestias pariatur odio, eius iure.</p> |
| | 9 <p>Suscipit animi odit soluta ab nisi in dolorum omnis voluptatem necessitatibus, rem voluptatibus reiciendis ipsam dignissimos dolor quos eligendi, eos esse accusantium commodi assumenda voluptates magni harum explicabo ad. Accusamus.</p> |
| | 10 <p>Eaque quo voluptatibus tenetur accusantium laboriosam, doloremque repellat ipsam dignissimos delectus ducimus, perspiciatis pariatur commodi perferendis adipisci eveniet nobis at, ut minima illo saepe sed obcaecati. Nihil explicabo dolor repudiandae?</p> |
| | 11 <p>Eos quas sunt quae quasi, a at cum omnis iure minus adipisci excepturi ducimus laborum laudantium suscipit pariatur dolorum quod quia, doloremque dignissimos quaerat beatae ex. Neque, laboriosam architecto. Necessitatibus.</p> |
| | 12 |
| | 13 <h3>Estamos escribiendo en la rama PRUEBA</h3> |
| | 14 |
| | 15 </body> |
| | 16 </html> |

Y si los cambios ya los tengo en MASTER puedo eliminar la rama PRUEBA!

```
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git branch
* master
  prueba
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git branch -d prueba
Eliminada la rama prueba (era 4f8c8ea)
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ git branch
* master
estefania@estefania-Lenovo-G470:~/Escritorio/proyecto$ □
```

Listo!



**Gracias por
llegar al final!**

Dudas y/o good vibes:
silestagui@gmail.com
