

# Prácticas guiadas con git y github



Luisa Parra García  
2023/2024

<b>Práctica 1: Nuestro primer repositorio</b>	<b>3</b>
Paso 1: Configuración Git.	3
Paso 2: Crear un nuevo proyecto en local.	3
Paso 3. El primer commit.	3
<b>Práctica 2. Trabajamos en local</b>	<b>5</b>
Añadimos ficheros al repositorio	5
Descartar cambios en el repositorio	7
Eliminar ficheros del repositorio	8
<b>Práctica 3: Histórico de operaciones en Git.</b>	<b>8</b>
<b>Práctica 4: El fichero .gitignore</b>	<b>10</b>
<b>Práctica 5: Cargando versiones antiguas</b>	<b>11</b>
Borrar último commit. Reset vs Revert	11
Checkout	12
<b>Práctica 6. Conectándome a un repositorio remoto.</b>	<b>15</b>
Subir al remoto tu repositorio local	15
Clonar tu repositorio remoto en local	16
<b>Práctica 7: Actualización del repositorio local</b>	<b>16</b>
Git fetch + git merge	18
Git pull	19
<b>Práctica 8: Actualización del repositorio remoto</b>	<b>21</b>
Actualización simple	21
Actualización del repositorio remoto no actualizado en local	22
Actualización del repositorio remoto con conflictos	25
<b>Práctica 9 Ramificaciones en git</b>	<b>29</b>
Paso 1: Crear una rama	29
Paso 2. Realizamos cambios en la rama y confirmamos	29
Paso 3: Fusionamos las ramas	30
Paso 4: Enviamos cambios al repositorio remoto	31

# Práctica 1: Nuestro primer repositorio

## Paso 1: Configuración Git.

Después de la instalación de git, como mínimo debemos configurar el nombre y el email en la aplicación:

```
git config --global user.name "Tu nombre aquí"
git config --global user.email "tu_email_aquí@example.com"
```

Comprueba tu configuración con el siguiente comando:

```
git config --global -list
```

Si necesitas **ayuda** con git, con `git help`, obtendremos ayuda.

## Paso 2: Crear un nuevo proyecto en local.

Sitúate en la carpeta donde tengas todos tus repositorios. Con `git init` y el nombre del proyecto, se nos creará una carpeta con el mismo nombre y en su interior podemos objetar que se crea una subcarpeta oculta `.git`.



```
MINGW64:/c/Users/Luisa/repos
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos
$ git init prueba01
Initialized empty Git repository in C:/Users/Luisa/repos/prueba01/.git/
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos
$ ls -la prueba01/
./ ../ .git/
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos
$ |
```

## Paso 3. El primer commit.

Creamos un archivo en el proyecto con algo de contenido, por ejemplo README.md en el contenido #Prueba Git

Si ejecutamos `git status`, veremos que estamos en la rama master y que tenemos un fichero untraked.

```

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ echo "#Prueba Git" >> README.md

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md

nothing added to commit but untracked files present (use "git add" to track)

```

Lo primero que tenemos que hacer es enviar el fichero al staging area, para ello usaremos git add. Si volvemos a ejecutar git status, veremos que ya está preparado para enviarlo a nuestro repositorio local mediante un commit:

```
git commit -m "Primer commit"
```

Al hacer git status, veremos que el staging area está vacía y no hay ningún commit pendiente.

```

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git add README.md

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git status
On branch master

No commits yet

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

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git commit -m "Primer commit"
[master (root-commit) f6b36a7] Primer commit
1 file changed, 1 insertion(+)
create mode 100644 README.md

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git status
On branch master
nothing to commit, working tree clean

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ |

```

## Práctica 2. Trabajamos en local

### Añadimos ficheros al repositorio

Vamos a modificar el fichero README.md. Si queremos hacer un commit, primero tendremos que volver a añadir los cambios al staging area, con `git -a`

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ echo "#Fichero inicial del proyecto" >> README.md

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git add README.md

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git commit -m "Modificación README.md"
[master 048e6cb] Modificaci|n README.md
1 file changed, 1 insertion(+)

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ |
```

Si quiero añadir todos los ficheros a la staging area, puedo usar la opción `git -a .`

```

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html
        styles.css

nothing added to commit but untracked files present (use "git add" to track)

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git add .

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   index.html
        new file:   styles.css

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$

```

Vamos a modificar, esos dos ficheros. Siguiendo los pasos anteriores deberíamos hacer:

```
git -a .
```

```
git commit -m "mensaje"
```

Pero estas instrucciones se pueden abreviar utilizando:

```
git commit -am "mensaje"
```

```

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   index.html
        new file:   styles.css

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html
        modified:   styles.css

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git commit -am "Index y hoja de estilos"
[master 90e1056] Index y hoja de estilos
2 files changed, 3 insertions(+)
create mode 100644 index.html
create mode 100644 styles.css

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$

```

## Descartar cambios en el repositorio

Cómo podemos sacar un archivo del staging area y descartar sus cambios en el directorio de trabajo. Nos valdremos de los comandos:

```
git reset HEAD <nombre del fichero> //para sacar del staging area
```

```
git checkout -- <nombre del fichero> // para descartar cambios en el directorio de trabajo.
```

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ cat index.html
<html>
<head></head>
</html>

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git reset HEAD index.html
Unstaged changes after reset:
M       index.html

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ cat index.html
<html>
<head></head>
</html>

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git checkout -- index.html

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ cat index.html
<html>
</html>

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ |
```

## Eliminar ficheros del repositorio

Imaginemos que uno de los archivos que ya están en el repositorio de git tras un commit no lo queremos allí. Podemos usar el comando rm de remove y el nombre del archivo:

```
git rm index.html
```

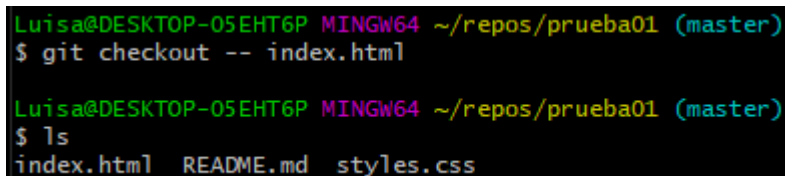


```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git rm index.html
rm 'index.html'

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    index.html

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ ls
README.md  styles.css
```

Si nos damos cuenta que nos hemos equivocado antes de hacer el commit podemos revertirlo:



```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git checkout -- index.html

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ ls
index.html  README.md  styles.css
```

Sí definitivamente queremos eliminarlo, después de hacer el borrado, hacemos un commit y ya desaparecerá de nuestro repositorio.

## Práctica 3: Histórico de operaciones en Git.

Si queremos ver los commit que se han hecho, utilizaremos un el comando git log, con git help log podemos ver todas las opciones que existen. Aquí tenéis una muestra de una vista compacta y colorida:

```
git log --oneline --graph --decorate --color
```

Y otra más detallada

```
git log --graph --decorate --color
```



```

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git log --oneline --graph --decorate --color
* 90e1056 (HEAD -> master) Index y hoja de estilos
* 048e6cb Modificación README.md
* f6b36a7 Primer commit

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git log --graph --decorate --color
* commit 90e105601a77f49d6d9fcad855f8e72ca5124b1f (HEAD -> master)
| Author: luisaparra-info <luisaparra.info@gmail.com>
| Date: Tue Oct 6 11:33:07 2020 +0200
|
| Index y hoja de estilos
|
* commit 048e6cb73f8fcec75f8c9fce8fb37e9eae60a3d
| Author: luisaparra-info <luisaparra.info@gmail.com>
| Date: Fri Oct 2 20:31:25 2020 +0200
|
| Modificación README.md
|
* commit f6b36a7de2cc0b3bee4a0d814d4e8f8bceb29c9c
| Author: luisaparra-info <luisaparra.info@gmail.com>
| Date: Fri Oct 2 20:06:44 2020 +0200
|
| Primer commit

```

Si en algún momento quieres repasar los ficheros modificados en cada commit puedes hacerlo con la opción -p:

```

commit 048e6cb73f8fcec75f8c9fce8fb37e9eae60a3d
Author: luisaparra-info <luisaparra.info@gmail.com>
Date: Fri Oct 2 20:31:25 2020 +0200

    Modificación README.md

diff --git a/README.md b/README.md
index 3fa9c62..8a79cf9 100644
--- a/README.md
+++ b/README.md
@@ -1,2 @@
-#Prueba Git
+#Fichero inicial del proyecto

commit f6b36a7de2cc0b3bee4a0d814d4e8f8bceb29c9c
Author: luisaparra-info <luisaparra.info@gmail.com>
Date: Fri Oct 2 20:06:44 2020 +0200

    Primer commit

diff --git a/README.md b/README.md
new file mode 100644
index 0000000..3fa9c62
--- /dev/null
+++ b/README.md
@@ -0,0 +1 @@
+#Prueba Git
(END)

```

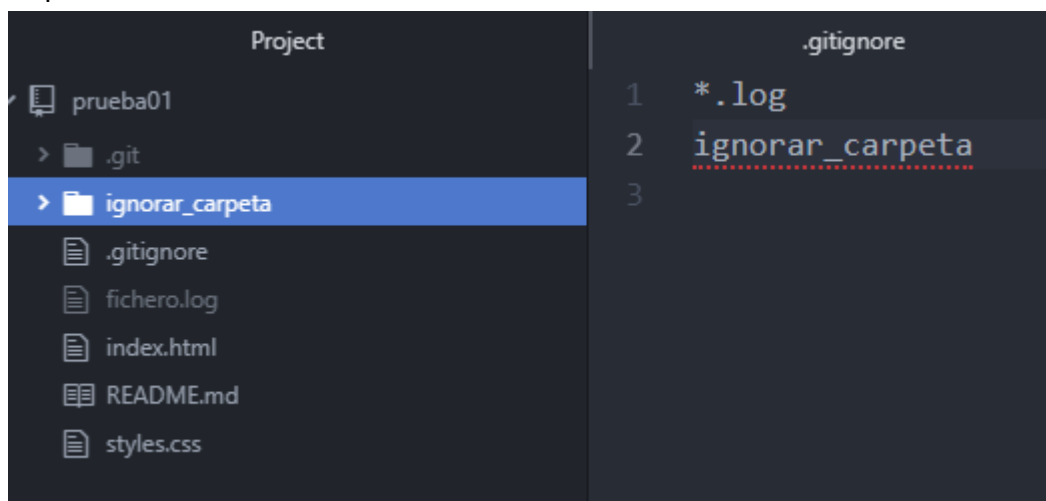
## Práctica 4: El fichero .gitignore

Git tiene una herramienta imprescindible casi en cualquier proyecto, el archivo ".gitignore", que sirve para decirle a Git qué archivos o directorios completos debe ignorar y no subir al repositorio de código.

Únicamente se necesita crear un archivo especificando qué elementos se deben ignorar y, a partir de entonces, realizar el resto del proceso para trabajar con Git de manera habitual.

Existe una herramienta online que se llama ***gitignore.io***. Básicamente permite escribir en un campo de búsqueda los nombres de todas las herramientas, sistemas, frameworks, lenguajes, etc. que puedas estar usando. Seleccionas todos los valores y luego generas el archivo de manera automática.

Vamos a crear un archivo .gitignore que ignore los ficheros \*.log y los contenidos de una carpeta en concreto.



Ahora podemos hacer pruebas para comprobar que cada de lo que añadamos a la carpeta\_ignorar, ni ningún fichero \*.log se tendrán en cuenta en un commit.

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ touch ignorar_carpeta/ignorar.txt
```

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ touch fichero.log
```

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git status
On branch master
nothing to commit, working tree clean
```

## Práctica 5: Cargando versiones antiguas

### Borrar último commit. Reset vs Revert

Muchas veces nos sucede que realizamos un commit y después nos damos cuenta que no deberíamos haberlo hecho y necesitamos eliminarlo. ¿Cómo podemos eliminar el commit?

#### Si no hemos subido el commit a nuestro repositorio remoto

`git reset --hard HEAD~1` → Se borran todos los commits posteriores al que indicamos

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git rm styles.css
rm 'styles.css'

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git commit -am "elimino styles"
[master eb9f7eb] elimino styles
1 file changed, 1 deletion(-)
delete mode 100644 styles.css

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ ls
fichero.log  ignorar_carpeta/  index.html  README.md

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git reset --hard HEAD~1
HEAD is now at 0e2b477 Revert "elimino styles"

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ ls
fichero.log  ignorar_carpeta/  index.html  README.md  styles.css

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$
```

`git reset --soft HEAD~1` → No perdemos los cambios de los commits posteriores.

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git reset --soft HEAD~1

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ ls
fichero.log  ignorar_carpeta/  index.html  README.md

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    deleted:    styles.css
```

Si hemos subido el commit a nuestro repositorio remoto

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git rm styles.css
rm 'styles.css'

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git commit -am "elimino styles.css"
[master 69ff45d] elimino styles.css
1 file changed, 1 deletion(-)
delete mode 100644 styles.css

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ ls
fichero.log  ignorar_carpeta/  index.html  README.md

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git revert HEAD
hint: Waiting for your editor to close the file...
[master 0f10a17] Revert "elimino styles.css"
1 file changed, 1 insertion(+)
create mode 100644 styles.css

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ ls
fichero.log  ignorar_carpeta/  index.html  README.md  styles.css

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ |
```

## Checkout

`git checkout` te permite ante situaciones donde hayas modificado cosas en el área de trabajo, pero aún no las hayas pasado al área de preparación (con `git add`), volver a la versión exacta del repositorio de dicho archivo, es decir, deshacer los cambios realizados. Para ello, simplemente tienes que hacer lo siguiente:

```
git checkout NOMBRE_DEL_ARCHIVO
```

Si quieres hacer esto mismo pero para todos los archivos, utilizarás:

```
git checkout -- .
```

Si además de modificaciones, has hecho inserciones tienes que ejecutar el siguiente comando:

```
git clean -df
```

```

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html
        modified:   styles.css

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        function.php

no changes added to commit (use "git add" and/or "git commit -a")

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git che
checkout      cherry      cherry-pick

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git checkout -- .

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        function.php

nothing added to commit but untracked files present (use "git add" to track)

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git clean -df
Removing function.php

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git status
On branch master
nothing to commit, working tree clean

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ |

```

Vamos a recuperar la versión de nuestro proyecto en el primer commit que hicimos. Si hacemos `git log` obtenemos lo siguiente:

```

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git log --oneline
0e2b477 (HEAD -> master) Revert "elimino styles"
c3aa53c elimino styles
0f10a17 Revert "elimino styles.css"
69ff45d elimino styles.css
5e20092 Revert "elimino styles.css"
490b82b elimino styles.css
2f46a02 gitignore
1a56821 borrar fichero
85830b2 fichero para borrar
90e1056 Index y hoja de estilos
048e6cb Modificación README.md
f6b36a7 Primer commit

```

Por lo tanto `f6b36a7` es el identificador de mi primer commit. Utilizando:  
`git checkout f6b36a7` , has solicitado cargar en tu directorio de trabajo esta instantánea del proyecto. Pero puedes volver en cualquier momento al punto en el que te encontrabas haciendo `git checkout master`.

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git checkout f6b36a7
Note: switching to 'f6b36a7'.

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 switching back to a branch.

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

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at f6b36a7 Primer commit

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 ((f6b36a7...))
$ ls
fichero.log  ignorar_carpeta/  README.md

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 ((f6b36a7...))
$ git checkout master
Previous HEAD position was f6b36a7 Primer commit
Switched to branch 'master'

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ ls
fichero.log  ignorar_carpeta/  index.html  README.md  styles.css

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ |
```

## Práctica 6. Conectándome a un repositorio remoto.

### Subir al remoto tu repositorio local

Primero vamos a crear el repositorio remoto, en nuestro caso en github, con el mismo nombre que el repositorio local.



Owner \* Repository name \*

luisaparra-info / prueba01 ✓

Great repository names are short and snappy. Need inspiration? How about fictional-meme?

Después, ejecutando los comandos de la siguiente imagen, añadimos el 'origin' como el repositorio recién creado y subimos el contenido del local al 'origin'

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git remote add origin https://github.com/luisaparra-info/prueba01.git

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git remote -v
origin https://github.com/luisaparra-info/prueba01.git (fetch)
origin https://github.com/luisaparra-info/prueba01.git (push)

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git push -u origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 436 bytes | 145.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/luisaparra-info/prueba01.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ |
```

## Clonar tu repositorio remoto en local

Supongamos ahora que tenemos un repositorio en remoto y queremos clonarlo en local para trabajar en ese proyecto.

Para seguir con el ejemplo, vamos a borrar nuestro repositorio actual en local con todo el contenido del directorio de trabajo y posteriormente vamos a clonar el repositorio que tenemos en github.

1º Borramos nuestro repositorio en local,

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos
$ rm -rf prueba01
```

2º Clonamos nuestro repositorio remoto

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos
$ git clone http://github.com/luisaparra-info/prueba01.git
Cloning into 'prueba01'...
warning: redirecting to https://github.com/luisaparra-info/prueba01.git/
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 6 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), 416 bytes | 9.00 KiB/s, done.

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos
$ cd prueba01/

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ ls -a
./ ../ .git/ .gitignore index.html README.md styles.css

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ |
```

## Práctica 7: Actualización del repositorio local

Lo primero que vamos a hacer es desde github hacer una modificación del repositorio en remoto, para ello vamos a editar el fichero README.txt y vamos a hacer un commit desde github, actualizado la rama master.



prueba01 / README.md Cancel

<> Edit file

Preview changes

```
1 #Prueba Git
2 #Fichero inicial del proyecto
3 Modificación hecha desde el remoto
.
```



### Commit changes

Actualización de README.md desde github

Add an optional extended description...

☒ Commit directly to the `master` branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes

Cancel

Si nos fijamos tanto en el estado del repositorio local, como remoto, vemos que no coinciden:

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git log --oneline
bacdf39 (HEAD -> master, origin/master, origin/HEAD) primer commit
```

master

Commits on Oct 11, 2020

Actualización de README.md desde github



luisaparra-info committed 4 minutes ago

Commits on Oct 10, 2020

primer commit



luisaparra-info committed 11 hours ago

## Git fetch + git merge

Si utilizamos git fetch, tan sólo recuperaremos la información del repositorio remoto y la ubicará en una rama oculta del repositorio local (FETCH\_HEAD), pero no hará el 'merge'. Así que tendremos que hacerlo manualmente si queremos que ambos repositorios estén sincronizados.

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git fetch
warning: redirecting to https://github.com/luisaparra-info/prueba01.git/
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 735 bytes | 31.00 KiB/s, done.
From http://github.com/luisaparra-info/prueba01
   bacdf39..02b9ea9  master       -> origin/master
```

Como vemos el contenido del fichero es diferente:

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ cat README.md
#Prueba Git
#Fichero inicial del proyecto

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git checkout FETCH_HEAD
Note: switching to 'FETCH_HEAD'.

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 switching back to a branch.

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

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 02b9ea9 Actualizaci|n de README.md desde github

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 ((02b9ea9...))
$ cat README.md
#Prueba Git
#Fichero inicial del proyecto
Modificación hecha desde el remoto

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 ((02b9ea9...))
$
```

Para fusionar la rama oculta con la rama local necesitamos hacer un git merge:

```

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 ((02b9ea9...))
$ git checkout master
Previous HEAD position was 02b9ea9 Actualizaci|n de README.md desde github
Switched to branch 'master'
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
(use "git pull" to update your local branch)

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git merge FETCH_HEAD
Updating bacdf39..02b9ea9
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git log --oneline
02b9ea9 (HEAD -> master, origin/master, origin/HEAD) Actualización de README.md
desde github
bacdf39 primer commit

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ cat README.md
#Prueba Git
#Fichero inicial del proyecto
Modificación hecha desde el remoto

```

## Git pull

El comando git pull es un atajo para evitar realizar las dos acciones anteriores en un solo paso. Cuando hacemos git pull, estamos bajando todos los cambios en el remoto en la rama que estamos trabajando.

Vamos a hacer una prueba de cómo revertir los cambios que hemos hecho hasta ahora tanto el remoto como en local:

```

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git revert 02b9ea9

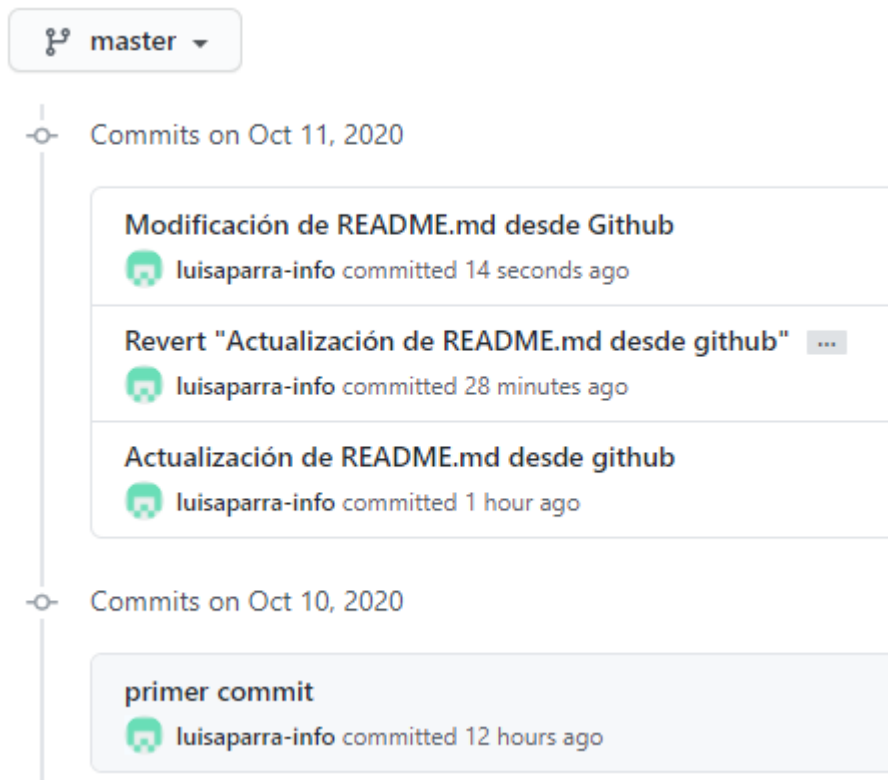
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git push
warning: redirecting to https://github.com/luisaparra-info/prueba01.git/
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 356 bytes | 178.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To http://github.com/luisaparra-info/prueba01.git
 02b9ea9..b95b799 master -> master

```

En el siguiente enlace puedes ver como realizar este tipo de acciones:

<https://midu.dev/como-deshacer-el-ultimo-commit-git/>

Ahora partiremos desde el principio, volviendo a realizar una modificación en remoto, la misma que anteriormente:



Y mediante pull vamos a actualizar nuestro repositorio local:

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ cat README.md
#Prueba Git
#Fichero inicial del proyecto

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git pull
warning: redirecting to https://github.com/luisaparra-info/prueba01.git/
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 738 bytes | 61.00 KiB/s, done.
From http://github.com/luisaparra-info/prueba01
   b95b799..dd66ebe master    -> origin/master
Updating b95b799..dd66ebe
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
```

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ cat README.md
#Prueba Git
#Fichero inicial del proyecto
Modificación hecha desde el remoto
```

# Práctica 8: Actualización del repositorio remoto

## Actualización simple

Vamos a modificar algunos ficheros en local de nuestro repositorio y vamos a hacer un push, es este primer ejemplo, suponemos que sólo nosotros estamos modificando el repositorio remoto.

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

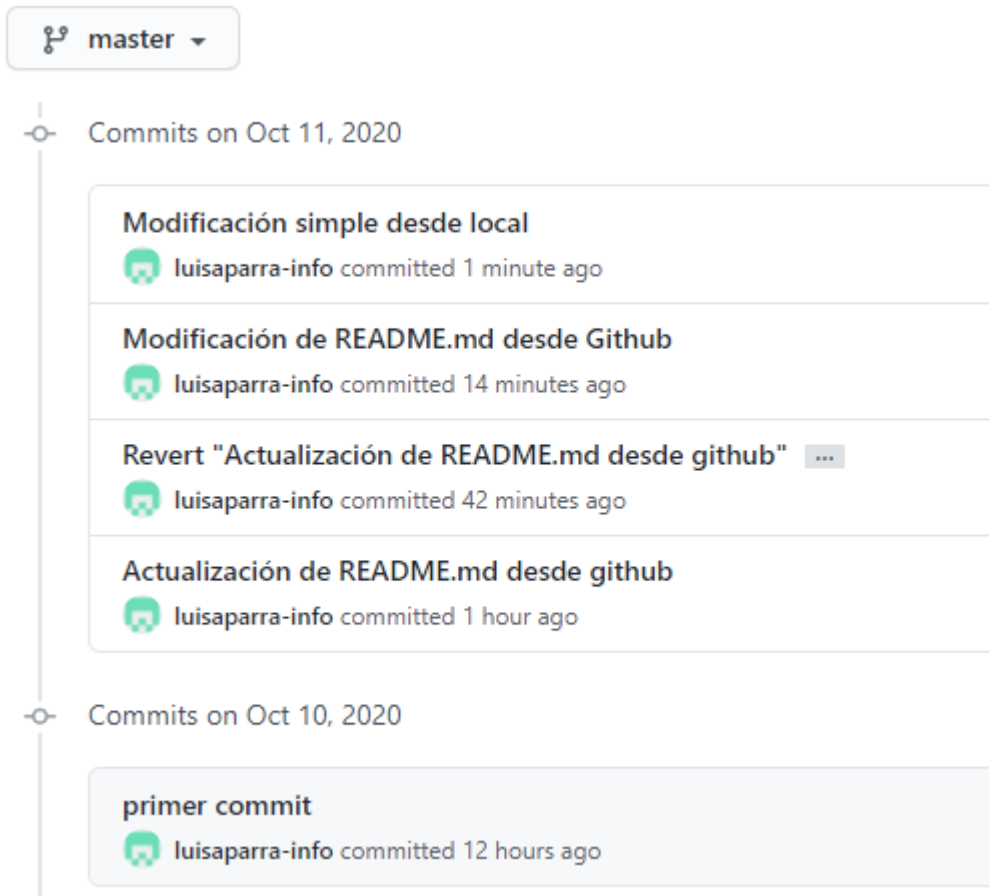
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md
        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git commit -am "Modificación simple desde local"
[master 0fc4247] Modificaci|n simple desde local
 2 files changed, 2 insertions(+)

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git push
warning: redirecting to https://github.com/luisaparra-info/prueba01.git/
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 485 bytes | 242.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To http://github.com/luisaparra-info/prueba01.git
 dd66ebe..0fc4247 master -> master
```

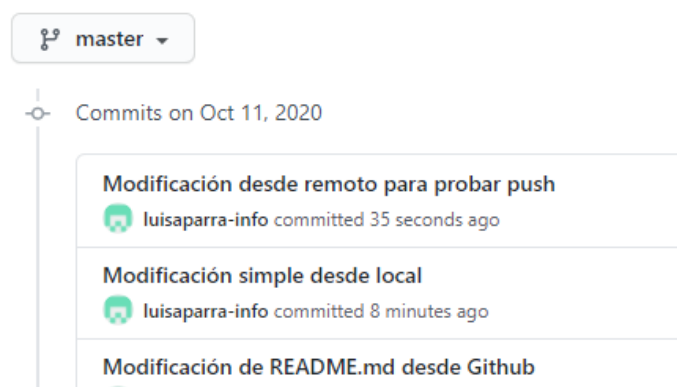
En la siguiente captura vemos como se reflejan los cambios en el remoto.



## Actualización del repositorio remoto no actualizado en local

Supongamos que alguien más está colaborando con nosotros en el proyecto y que ha hecho modificaciones en el repositorio, que aún nosotros no tenemos en nuestro local. Realmente, a no ser que estemos utilizando un entorno visual con git, a simple vista, no sabremos si eso se ha producido. Veamos qué ocurre si intentamos actualizar el remoto, cuando no tenemos la última versión del mismo.

1º. Modificamos el repositorio en remoto



2º Modificamos el repositorio en local ficheros distintos.

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git commit -am 'Modificación en local para probar push'
[master 55ff049] Modificaci|n en local para probar push
 1 file changed, 1 insertion(+)
```

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git log --oneline
55ff049(HEAD -> master) Modificación en local para probar push
0fc4247 (origin/master, origin/HEAD) Modificación simple desde local
dd66ebe Modificación de README.md desde Github
b95b799 Revert "Actualización de README.md desde github"
02b9ea9 Actualización de README.md desde github
bacdf39 primer commit
```

Cuando intentamos hacer push, ocurre lo siguiente:

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git push
warning: redirecting to https://github.com/luisaparra-info/prueba01.git/
To http://github.com/luisaparra-info/prueba01.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'http://github.com/luisaparra-info/prueba01.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Esto ocurre porque no tenemos el repositorio actual actualizado, por lo tanto debemos hacer un pull o fetch antes de actualizar el remoto.

```

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git pull
warning: redirecting to https://github.com/luisaparra-info/prueba01.git/
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 735 bytes | 28.00 KiB/s, done.
From http://github.com/luisaparra-info/prueba01
   0fc4247..9f73309  master    -> origin/master
hint: Waiting for your editor to close the file...
Merge made by the 'recursive' strategy.
 README.md | 1 +
 1 file changed, 1 insertion(+)

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git push
warning: redirecting to https://github.com/luisaparra-info/prueba01.git/
Enumerating objects: 9, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 580 bytes | 290.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To http://github.com/luisaparra-info/prueba01.git
   9f73309..845e2ae  master -> master

```

Veamos ahora que los dos últimos commits están tanto en local como en remoto:

```

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git log --oneline
845e2ae (HEAD -> master, origin/master, origin/HEAD) Merge branch 'master' of http://github.com/luisaparra-info/prueba01 into master
55ff049 Modificación en local para probar push
9f73309 Modificación desde remoto para probar push
0fc4247 Modificación simple desde local
dd66ebe Modificación de README.md desde Github
b95b799 Revert "Actualización de README.md desde github"
02b9ea9 Actualización de README.md desde github
bacdf39 primer commit

```

 master ▼



Commits on Oct 11, 2020

Merge branch 'master' of <http://github.com/luisaparra-info/prueba01> i... ...

...nto master


luisaparra-info committed 4 minutes ago

Modificación en local para probar push


luisaparra-info committed 9 minutes ago

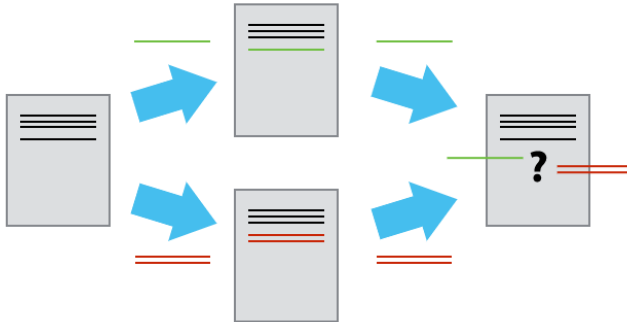
Modificación desde remoto para probar push


luisaparra-info committed 12 minutes ago



## Actualización del repositorio remoto con conflictos

Pongámonos en la situación de que un colaborador de nuestro repositorio ha modificado el fichero index.html y ha subido su versión al repositorio remoto. Posteriormente nosotros tenemos que modificar ese mismo fichero y subirlo también al repositorio. Al subirlo, nos dará un conflicto en el merge final que tendremos que resolver.



Veamos los pasos que tenemos que seguir para reproducir esta situación:

1º. Modificamos el fichero index.html en el repositorio remoto.

### Modificación index.html remoto - prueba conflictos

🔑 master

👤 luisaparra-info committed 38 seconds ago Verified

📄 Showing 1 changed file with 3 additions and 1 deletion.

4 index.html		
...	...	@@ -1,4 +1,6 @@
1	1	<html>
2	-	<head></head>
	2	+ <head>
	3	+ <meta charset="utf-8">
	4	+ </head>
3	5	<body></body>
4	6	</html>

2º Modificamos el mismo fichero en local

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git commit -am "Modificación index.html local - prueba conflictos"
[master 70f730d] Modificaci|n index.html local - prueba conflictos
1 file changed, 3 insertions(+), 1 deletion(-)

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git log -p -1
commit 70f730dd560fd51e3fb8bfb5470e7001bf003899 (HEAD -> master)
Author: luisaparra-info <luisaparra.info@gmail.com>
Date: Sun Oct 11 11:23:23 2020 +0200

    Modificación index.html local - prueba conflictos

diff --git a/index.html b/index.html
index 60e3e88..c5572f1 100644
--- a/index.html
+++ b/index.html
@@ -1,4 +1,6 @@
<html>
<head></head>
- <body></body>
+ <body>
+ <header></header>
+ </body>
</html>
```

Veamos que si queremos hacer un push de nuestra rama en local, nos dirá que no estamos actualizados, ya que hay modificaciones pendientes de bajar.

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git push origin master
warning: redirecting to https://github.com/luisaparra-info/prueba01.git/
To http://github.com/luisaparra-info/prueba01.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'http://github.com/luisaparra-info/prueba01.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Por lo tanto haremos un pull para actualizarnos:

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git pull origin master
warning: redirecting to https://github.com/luisaparra-info/prueba01.git/
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 723 bytes | 28.00 KiB/s, done.
From http://github.com/luisaparra-info/prueba01
 * branch          master      -> FETCH_HEAD
   845e2ae..46ff5cb master     -> origin/master
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

El pull ha funcionado, pero no del todo, ya tenemos conflictos pendientes de resolver antes de nos deje hacer el merge:

```

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master|MERGING)
$ git status
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 1 different commits each, respectively.
(use "git pull" to merge the remote branch into yours)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")

```

Ahora tenemos que resolver los conflictos y hacer un commit, para así poder hacer el push al remoto.

Pero ¿cómo sabemos dónde está el conflicto? Pues git lo marca de la siguiente manera:

```

1  <html>
2  <<<<<< HEAD
3  <head></head>
4  <body>
5  <header></header>
6  </body>
7  =====
8  <head>
9    <meta charset="utf-8">
10   </head>
11  <body></body>
12  >>>>>> 46ff5cbcdeab31787bcd3faad265be0131e03ec5

```

Lo que me quiere decir que yo en mi HEAD tengo una versión y la que viene pull tiene otra versión y me las marca. Ahora tengo que quedarme con la versión que me interesa o una mezcla de las dos, que es el caso.

Por lo tanto, editamos el fichero hasta que quede de la siguiente manera:

```

<html>
<head>
  <meta charset="utf-8">
</head>
<body>
<header></header>
</body>
</html>

```

Ya podemos hacer el commit y el push de nuestros cambios al remoto:

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master|MERGING)
$ git commit -am "Resolución de conflictos en index.html"
[master fc6e5e1] Resolución de conflictos en index.html
```


```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git push origin master
warning: redirecting to https://github.com/luisaparra-info/prueba01.git/
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 672 bytes | 224.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To http://github.com/luisaparra-info/prueba01.git
 46ff5cb..fc6e5e1 master -> master
```

Éste sería el estado del repositorio en local:


```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git log --oneline
fc6e5e1 (HEAD -> master, origin/master, origin/HEAD) Resolución de conflictos en index.html
70f730d Modificación index.html local - prueba conflictos
46ff5cb Modificación index.html remoto - prueba conflictos
845e2ae Merge branch 'master' of http://github.com/luisaparra-info/prueba01 into master
55ff049 Modificación en local para probar push
9f73309 Modificación desde remoto para probar push
0fc4247 Modificación simple desde local
dd66ebe Modificación de README.md desde Github
b95b799 Revert "Actualización de README.md desde github"
02b9ea9 Actualización de README.md desde github
bacdf39 primer commit

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ cat index.html
<html>
<head>
  <meta charset="utf-8">
</head>
<body>
<header></header>
</body>
</html>
```


Y este el estado del remoto:

 master ▾

Commits on Oct 11, 2020

Resolución de conflictos en index.html  
 luisaparra-info committed 4 minutes ago

Modificación index.html local - prueba conflictos  
 luisaparra-info committed 22 minutes ago

Modificación index.html remoto - prueba conflictos  
 luisaparra-info committed 27 minutes ago

## Práctica 9 Ramificaciones en git

Una rama (branch) es una bifurcación en la línea de tiempo del proyecto que nos permite crear una copia paralela para desarrollar cambios sin afectar la versión estable (por defecto la rama master).

Supongamos que nos mandan una tarea nueva y es hacer la página aboutus.html de nuestra página web y para ello queremos partir de la última versión de "master". Una vez creada la rama, haremos las modificaciones necesarias y una vez que comprobamos que hace lo que nosotros queramos, subiremos las modificaciones a la rama master.

### Paso 1: Crear una rama

Como vemos en la siguiente imagen, en este repositorio sólo existe la rama master, por lo tanto vamos a crear una nueva rama llamada LPG\_aboutus.html\_20201012, es interesante que durante el trabajo colaborativo se establezca un protocolo en la nomenclatura de las ramas de los diferentes colaboradores para facilitar un búsqueda.

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git branch
* master

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git branch LPG_aboutus.html_20201012

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git branch
  LPG_aboutus.html_20201012
* master
```

Como vemos, la rama donde nos encontramos, es la rama marcada con un \*. Vamos ahora a hacer un checkout para posicionarnos en la rama recién creada.

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git checkout LPG_aboutus.html_20201012
Switched to branch 'LPG_aboutus.html_20201012'

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (LPG_aboutus.html_20201012)
$ |
```

### Paso 2. Realizamos cambios en la rama y confirmamos

Vamos a hacer cambios, empezaremos por crear el fichero y posteriormente introduciremos contenido.

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (LPG_aboutus.html_20201012)
$ git status
On branch LPG_aboutus.html_20201012
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  aboutus.html

nothing added to commit but untracked files present (use "git add" to track)
```

Confirmamos los cambios:

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (LPG_aboutus.html_20201012)
$ git commit -m "Nueva pagina aboutus.html"
[LPG_aboutus.html_20201012 2beba58] Nueva pagina aboutus.html
1 file changed, 10 insertions(+)
create mode 100644 aboutus.html
```

### Paso 3: Fusionamos las ramas

Una vez aquí tenemos dos opciones, subir la rama al remoto o no. Para esta práctica, dejaremos la rama sólo en local y sólo subiremos los cambios resultantes de fusionarnos con master.

Si en este punto listamos el contenido de master y de nuestra rama, vemos que son distintos.

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (LPG_aboutus.html_20201012)
$ ls
aboutus.html index.html README.md styles.css

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (LPG_aboutus.html_20201012)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ ls
index.html README.md styles.css
```

Necesitamos por lo tanto fusionar nuestro contenido con master. Para ello, sobre la rama master, hacemos un git merge con nuestra rama.

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git merge LPG_aboutus.html_20201012
Updating fc6e5e1..2beba58
Fast-forward
 aboutus.html | 10 ++++++++
1 file changed, 10 insertions(+)
create mode 100644 aboutus.html

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ ls
aboutus.html index.html README.md styles.css
```

Como vemos, master, ya contiene los cambios realizados en nuestra rama.

## Paso 4: Enviamos cambios al repositorio remoto

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git log --oneline
2beba58 (HEAD -> master, LPG_aboutus.html_20201012) Nueva pagina aboutus.html
fc6e5e1 (origin/master, origin/HEAD) Resolución de conflictos en index.html
70f730d Modificación index.html local - prueba conflictos
46ff5cb Modificación index.html remoto - prueba conflictos
845e2ae Merge branch 'master' of http://github.com/luisaparra-info/prueba01 into master
55ff049 Modificación en local para probar push
9f73309 Modificación desde remoto para probar push
0fc4247 Modificación simple desde local
dd66ebe Modificación de README.md desde Github
b95b799 Revert "Actualización de README.md desde github"
02b9ea9 Actualización de README.md desde github
bacdf39 primer commit

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git push origin master
warning: redirecting to https://github.com/luisaparra-info/prueba01.git/
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 393 bytes | 196.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To http://github.com/luisaparra-info/prueba01.git
   fc6e5e1..2beba58 master -> master

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git log --oneline
2beba58 (HEAD -> master, origin/master, origin/HEAD, LPG_aboutus.html_20201012) Nueva pagina aboutus.html
fc6e5e1 Resolución de conflictos en index.html
70f730d Modificación index.html local - prueba conflictos
46ff5cb Modificación index.html remoto - prueba conflictos
845e2ae Merge branch 'master' of http://github.com/luisaparra-info/prueba01 into master
55ff049 Modificación en local para probar push
9f73309 Modificación desde remoto para probar push
0fc4247 Modificación simple desde local
dd66ebe Modificación de README.md desde Github
b95b799 Revert "Actualización de README.md desde github"
02b9ea9 Actualización de README.md desde github
bacdf39 primer commit

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ |
```

## Práctica 10: Las ramas en remoto

### Subir mi rama a remoto y actualizarla

Supongamos que queremos hacer una modificación que durará por un largo tiempo y que puede que nos interese que algún colaborador pueda verla. Necesitamos, por tanto, subirla al remoto.

Como se puede ver en la siguiente imagen, hemos procedido igual que en la práctica anterior, creando la rama, posicionándonos sobre ella y confirmando los cambios pertinentes.



```

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git branch LPG_modificacion_frontend_20201013

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git checkout LPG_modificacion_frontend_20201013
Switched to branch 'LPG_modificacion_frontend_20201013'

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (LPG_modificacion_frontend_20201013)
$ git status
On branch LPG_modificacion_frontend_20201013
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (LPG_modificacion_frontend_20201013)
$ git commit -am 'Modificación frontend'
[LPG_modificacion_frontend_20201013 d411324] Modificaci|n frontend
1 file changed, 3 insertions(+), 1 deletion(-)

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (LPG_modificacion_frontend_20201013)
$ |

```

Vamos ahora a subir la rama al remoto:

`git push -u origin <nombre rama>`

```

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (LPG_modificacion_frontend_20201013)
$ git push -u origin LPG_modificacion_frontend_20201013
warning: redirecting to https://github.com/luisaparra-info/prueba01.git/
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 346 bytes | 69.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'LPG_modificacion_frontend_20201013' on GitHub
by visiting:
remote:   https://github.com/luisaparra-info/prueba01/pull/new/LPG_modificaci
on_frontend_20201013
remote:
To http://github.com/luisaparra-info/prueba01.git
* [new branch]      LPG_modificacion_frontend_20201013 -> LPG_modificacion_fron
tend_20201013
Branch 'LPG_modificacion_frontend_20201013' set up to track remote branch 'LPG_m
odificacion_frontend_20201013' from 'origin'.

```

Como podemos ver, en github aparece la nueva rama:

All branches	
<code>master</code>	Updated 15 hours ago by luisaparra-info
<code>LPG_modificacion_frontend_20201013</code>	Updated 9 minutes ago by luisaparra-info



La manera de proceder para ir actualizando la rama es, modificar y confirmar los cambios en local y subirlos al origen. Una vez que demos por finalizada la tarea, procederemos igual que en la práctica 9, fusionando nuestra rama con master y subiéndolo al repositorio.

Cuando hablemos de workflow en git, veremos que se producen eliminación de ramas ( en local: `git branch -d rama_a_borrar` y en remoto: `git push origin --delete rama_a_borrar` ), pero lo veremos en entorno gráfico.

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (LPG_modificacion_frontend_20201013)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git merge LPG_modificacion_frontend_20201013
Updating 2beba58..d411324
Fast-forward
 index.html | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)

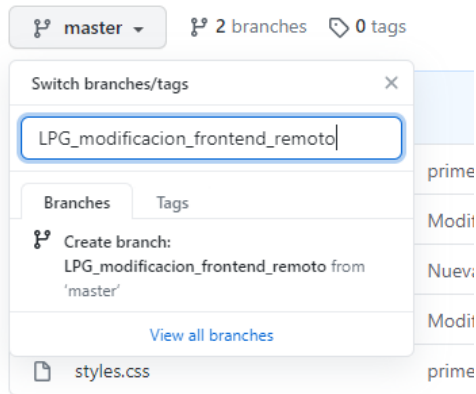
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ cat index.html
<html>
<head>
  <meta charset="utf-8">
</head>
<body>
<header>
  <h1>Mi web</h1>
</header>
</body>
</html>

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git push origin master
warning: redirecting to https://github.com/luisaparra-info/prueba01.git/
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To http://github.com/luisaparra-info/prueba01.git
 2beba58..d411324 master -> master

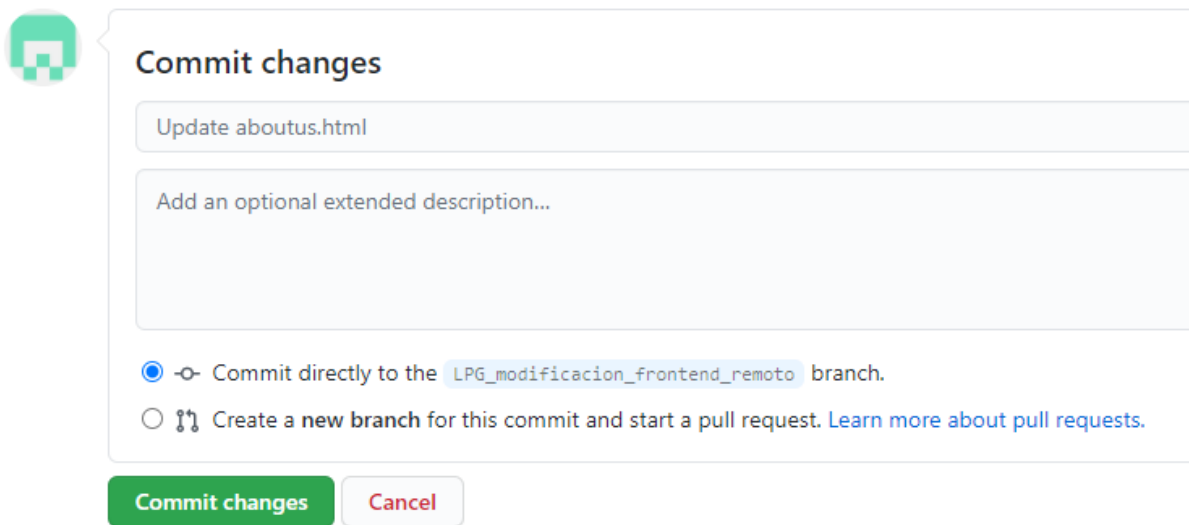
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ |
```

## Bajar rama del remoto y trabajar en ella

Vamos a crear una nueva rama en el remoto:



Y vamos a hacer alguna modificación y la vamos a confirmar:



Si listamos las ramas que tenemos en local, vemos que no tenemos la rama que acabamos de crear en remoto:

```
Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git branch -a
  LPG_aboutus.html_20201012
  LPG_modificacion_frontend_20201013
* master
remotes/origin/HEAD -> origin/master
remotes/origin/LPG_modificacion_frontend_20201013
remotes/origin/master
```

Por lo tanto vamos a bajarla para poder trabajar en ella:

```

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git pull
warning: redirecting to https://github.com/luisaparra-info/prueba01.git/
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 713 bytes | 39.00 KiB/s, done.
From http://github.com/luisaparra-info/prueba01
* [new branch]      LPG_modificacion_frontend_remoto -> origin/LPG_modificacion_frontend_remoto
Already up to date.

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git branch -a
  LPG_aboutus.html_20201012
  LPG_modificacion_frontend_20201013
* master
remotes/origin/HEAD -> origin/master
remotes/origin/LPG_modificacion_frontend_20201013
remotes/origin/LPG_modificacion_frontend_remoto
remotes/origin/master

```

Si observamos sólo la tenemos como remota, no hay ninguna rama local con ese nombre, pero al hacer checkout sobre ella, ya si se nos crea una rama local con ese nombre:

```

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (master)
$ git checkout LPG_modificacion_frontend_remoto
Switched to a new branch 'LPG_modificacion_frontend_remoto'
Branch 'LPG_modificacion_frontend_remoto' set up to track remote branch 'LPG_modificacion_frontend_remoto' from 'origin'.

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (LPG_modificacion_frontend_remoto)
$ git branch -a
  LPG_aboutus.html_20201012
  LPG_modificacion_frontend_20201013
* LPG_modificacion_frontend_remoto
master
remotes/origin/HEAD -> origin/master
remotes/origin/LPG_modificacion_frontend_20201013
remotes/origin/LPG_modificacion_frontend_remoto
remotes/origin/master

Luisa@DESKTOP-05EHT6P MINGW64 ~/repos/prueba01 (LPG_modificacion_frontend_remoto)

```

La manera de proceder será igual que en casos anteriores, podemos:

- Añadir modificaciones en local
- Subir modificaciones a remoto
- Descargarnos modificaciones de remoto
- Resolver conflictos si fuese necesario.