

Despliegue de aplicaciones web

Tutorial de Git + GitFlow



Actualizado Septiembre 2024


Licencia




Reconocimiento – NoComercial - Compartir Igual (BY-NC-SA): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

 Importante

 Atención

 Interesante

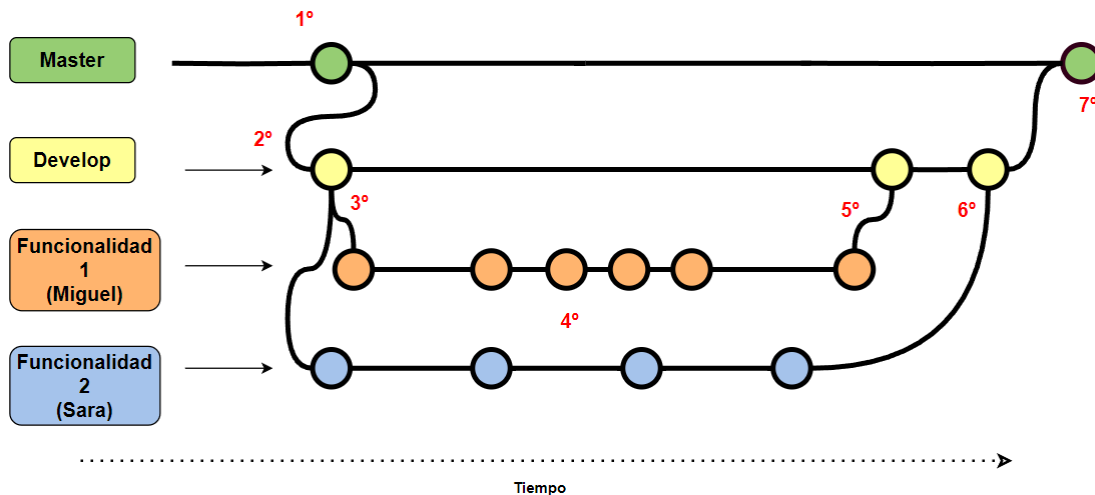
ÍNDICE DE CONTENIDO

1. Introducción	4
2. Preparación del Proyecto y Entorno	4
3. ¿Cómo trabajar en equipo?	22
4. Pull Request	33
5. Git Stash	36
6. Conflictos	38
7. Gitignore	43
8. Reset y Revert	46
9. Bibliografía	50
10. Autores (en orden alfabético)	50

1. INTRODUCCIÓN

A continuación, se mostrará de forma práctica ejemplos de uso de GIT según todo el contenido visto en el documento anterior, de esta forma, se podrá ver de forma práctica cómo hacer uso de GIT.

Vamos a trabajar con el GitFlow que se propuso y que sigue el siguiente diagrama:



Vamos a suponer que estamos en un proyecto de desarrollo de una aplicación Web y en la cual van a trabajar 2 desarrolladores, uno de ellos será el jefe de proyecto:

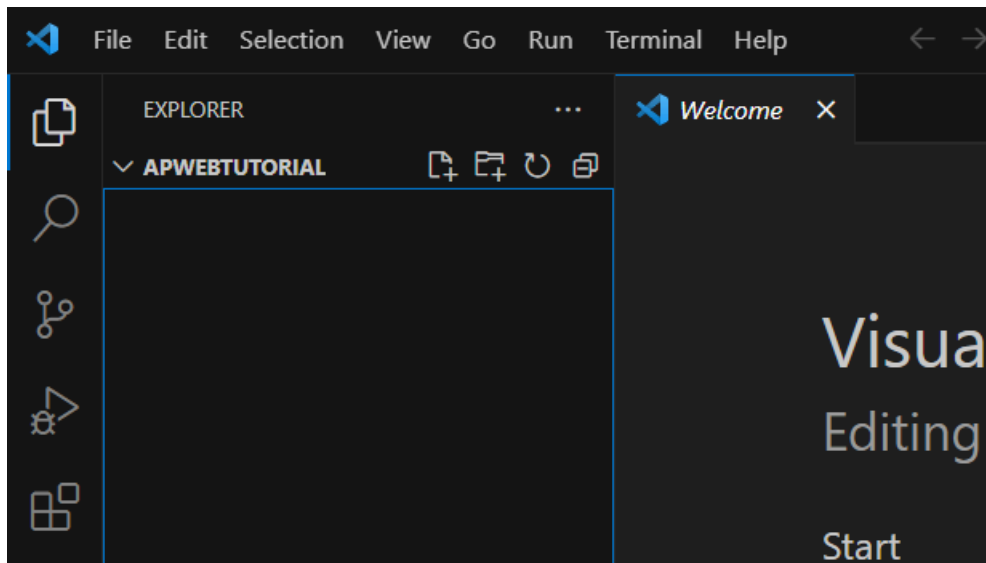
- Vamos a simular que nosotros somos los dos desarrolladores, por ello, vamos a tener que utilizar dos equipos, utilizaremos nuestra máquina virtual de ubuntu que hicimos en la práctica 2 y 3 del primer tema y nuestro equipo anfitrión.
- La máquina anfitrión será el desarrollador 1 y la máquina de ubuntu será el desarrollador 2.
- Por lo tanto, instala GIT en ambos equipos y ejecuta los comandos necesarios para configurar el email y el nombre en ambos equipos (para el desarrollador 1 pon tu nombre y para el desarrollador 2 invéntate uno).

2. PREPARACIÓN DEL PROYECTO Y ENTORNO

1º Lo primero de todo es tener instalado GIT en el equipo.

2º El jefe de proyecto (desarrollador 1) es el que va a crear el proyecto, por lo tanto, crea una carpeta con el nombre “apWebTutorial”.

3º Abrimos el proyecto con Visual Studio Code, ya que vamos a trabajar con este IDE.



4º Abrimos el terminal de VSCode e inicializamos el repositorio.

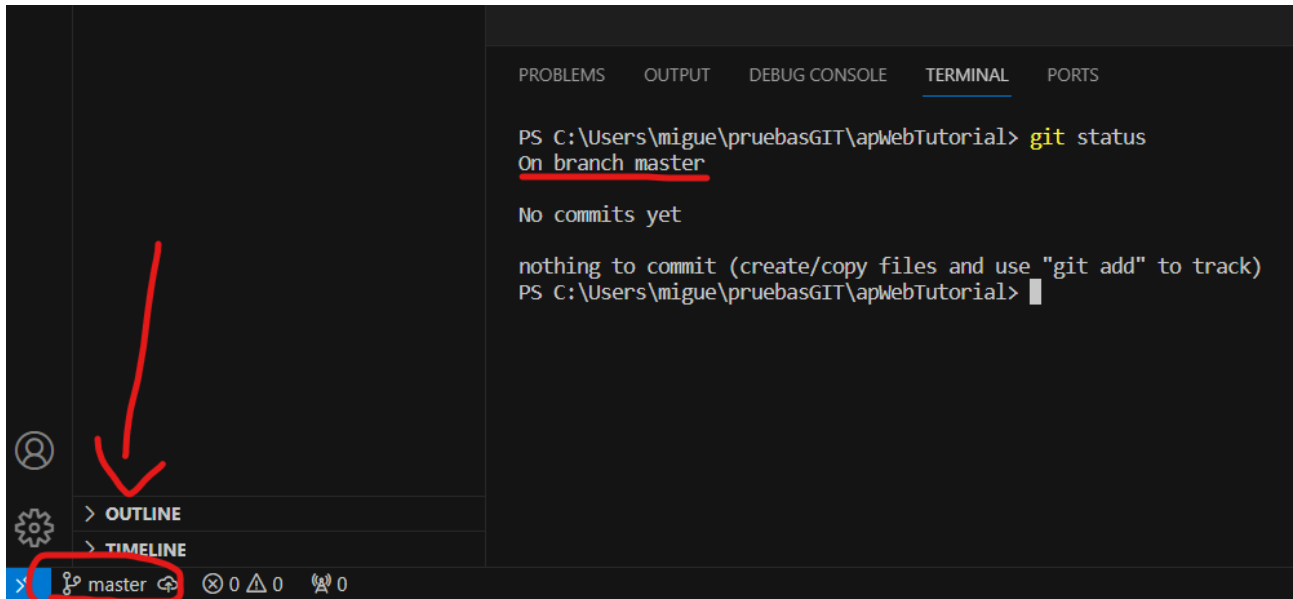
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\miguel\pruebasGIT\apWebTutorial> git init
Initialized empty Git repository in C:/Users/miguel/pruebasGIT/apWebTutorial/.git/
PS C:\Users\miguel\pruebasGIT\apWebTutorial> 
```

El repositorio Git está representado por una única carpeta oculta llamada .git, esta carpeta contiene TODA la información del repositorio local.

Este equipo > Disco local (C:) > Usuarios > miguel > pruebasGIT > apWebTutorial			
	Nombre	Fecha de modificación	Tipo
al	.git	20/09/2023 8:44	Carpeta de arcl

5º Vamos a ejecutar “git status” para ver en que ramas nos encontramos y ver que como no hemos hecho nada todavía está todo limpio y no hay nada que podamos hacer commit.



The screenshot shows the VS Code interface. On the left, the Explorer sidebar is visible with a red arrow pointing to the 'master' branch in the status bar. The status bar at the bottom left shows 'master' with a branch icon. The terminal on the right shows the output of the 'git status' command:

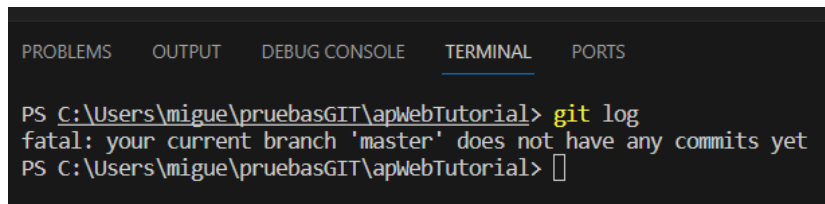
```
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
PS C:\Users\miguel\pruebasGIT\apWebTutorial>
```

En VSCode podemos ver en qué rama nos encontramos en cada momento desde la esquina inferior izquierda.

Incluso si probamos a ver el log de la rama, veremos que tampoco hay nada todavía:



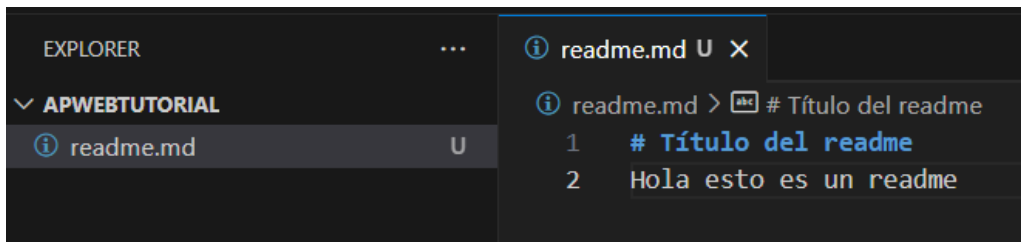
The screenshot shows the terminal output of the 'git log' command:

```
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git log
fatal: your current branch 'master' does not have any commits yet
PS C:\Users\miguel\pruebasGIT\apWebTutorial>
```

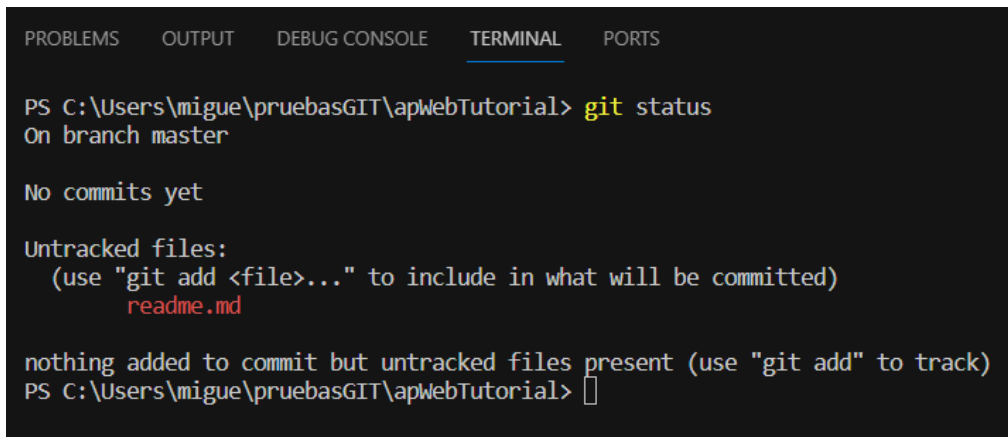
6º Ahora, nos encontraríamos con el siguiente escenario:



7º Vamos a hacer nuestro primer commit, para ello, vamos a crear un archivo llamado "readme.md" con las siguientes líneas:

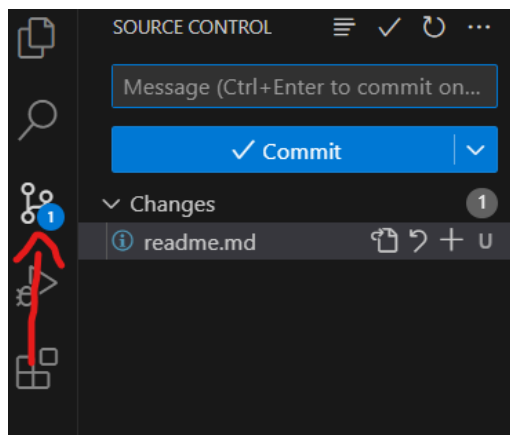


Guarda y vuelve a probar "git status"



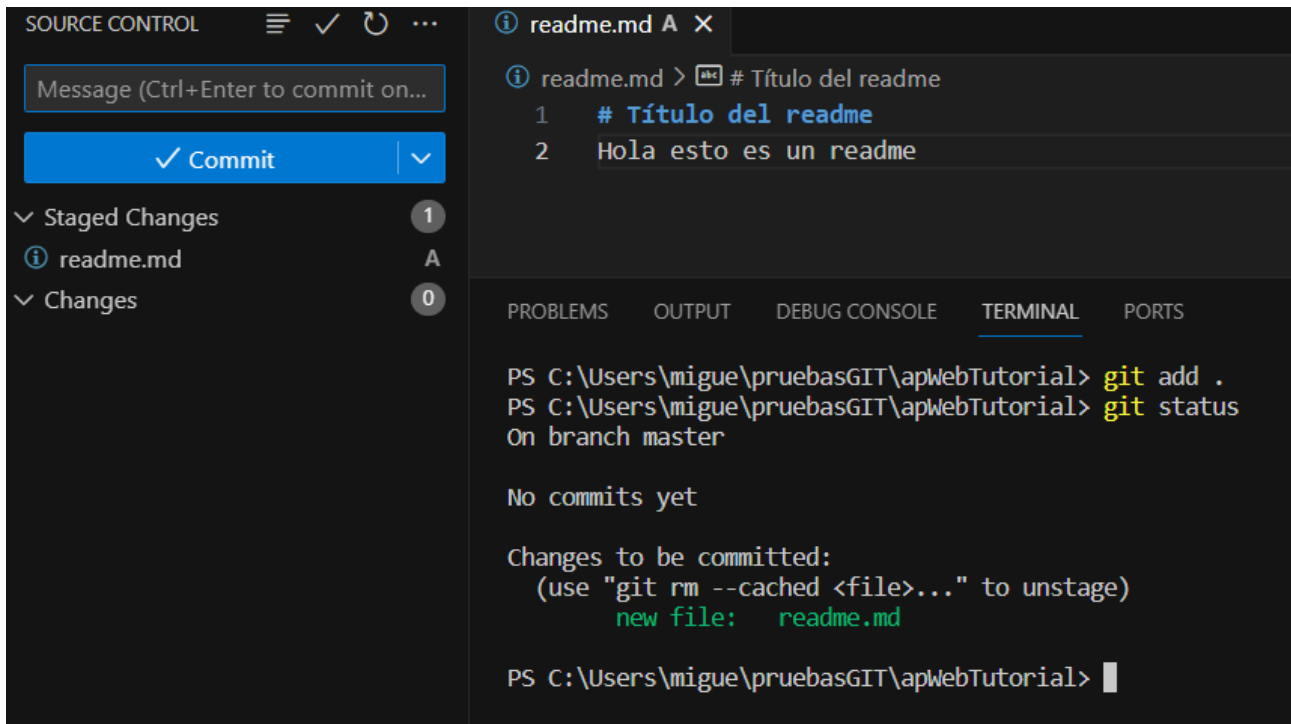
Como vemos hemos creado un archivo que está en el área de trabajo, pero todavía no se ha añadido al Stage Area.

Por otro lado, VSCode tiene una vista que me indica los cambios que han habido:



Incluso si te fijas, te permite hacer un add y un commit de forma visual que incluirá todos esos cambios del listado, pero no vamos a utilizarlo. Lo recomendable es aprender primero a utilizar GIT desde la consola.

Vamos a añadir el archivo al Stage Area con el comando add:



The screenshot shows the VS Code interface. On the left, the 'SOURCE CONTROL' panel displays 'readme.md' under 'Staged Changes'. The main editor shows the content of 'readme.md' with a title and a greeting. The terminal at the bottom shows the following commands and output:

```
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git add .
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git status
On branch master

No commits yet

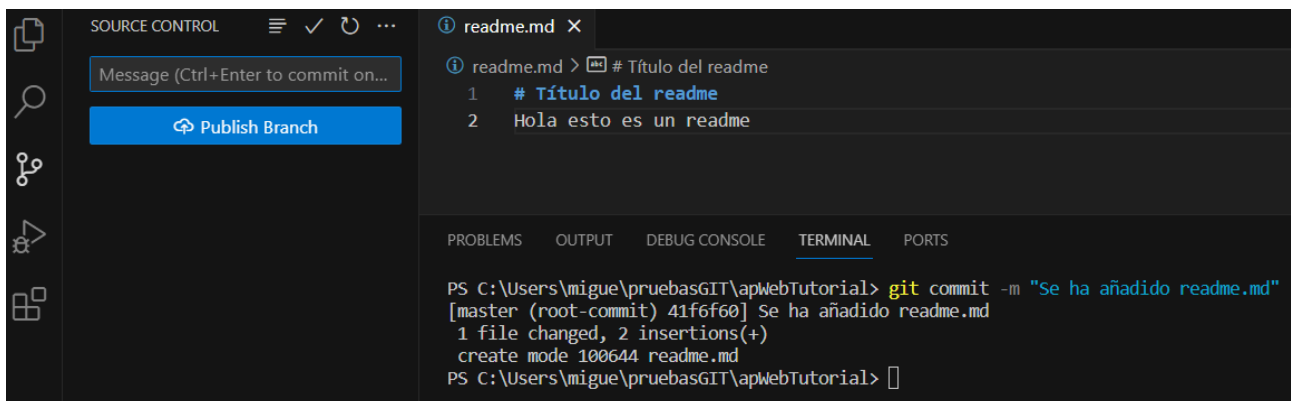
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   readme.md

PS C:\Users\miguel\pruebasGIT\apWebTutorial>
```

Varias cosas:

- He utilizado **git add .** : yo recomiendo hacerlo así, porque cuando hay muchos archivos no vamos a ir uno por uno.
- Vemos cómo al hacer **git status** ya está el fichero en verde.
- Vemos cómo en la vista de VSCode ha metido el fichero en Staged Changes.

Ahora ya estamos preparados para hacer commit de este cambio:



The screenshot shows the VS Code interface after the commit. The 'SOURCE CONTROL' panel now shows 'readme.md' as committed. The terminal shows the following commands and output:

```
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git commit -m "Se ha añadido readme.md"
[master (root-commit) 41f6f60] Se ha añadido readme.md
1 file changed, 2 insertions(+)
create mode 100644 readme.md
PS C:\Users\miguel\pruebasGIT\apWebTutorial>
```

Como vemos, en este caso, ya se ha quedado registrado el cambio en nuestro repositorio local. En la vista de VSCode vemos que ya no hay nada.


```
PS C:\Users\migue\pruebasGIT\apWebTutorial> git status
On branch master
nothing to commit, working tree clean
PS C:\Users\migue\pruebasGIT\apWebTutorial> git log
commit 41f6f6079996bbb8a25d3ea30bea22227327a0e (HEAD -> master)
Author: Miguel Mira Flor <[REDACTED]>
Date:   Wed Sep 20 09:00:30 2023 +0200

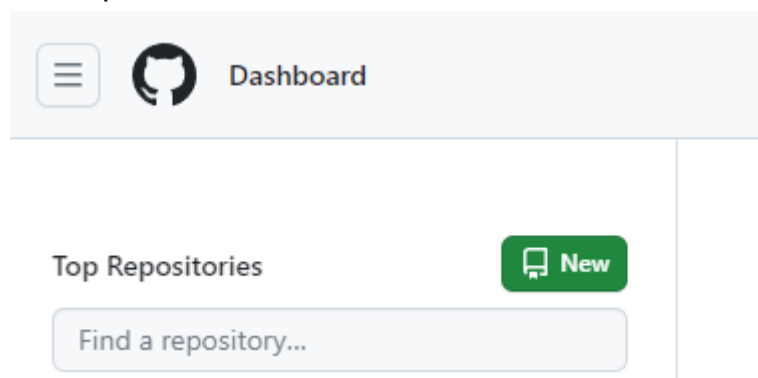
    Se ha añadido readme.md
PS C:\Users\migue\pruebasGIT\apWebTutorial> █
```

El status me dice que está todo limpio, no hay nada pendiente, y el log me indica el commit que hay registrado junto con su hash, además, sale la fecha, hora y el comentario.

8º En este escenario el proyecto está en un repositorio local, eso significa que el desarrollador 2 no tiene forma de poder partir del mismo repositorio. Para ello, primero necesitamos subirlo a un repositorio remoto, de esa forma, el desarrollador 2 se lo podrá descargar.

Vamos a crear el repositorio remoto, para ello, necesitaremos una cuenta de GitHub. Regístrate con un email, esta será la cuenta del desarrollador 1.

Inicia sesión y crea un repositorio nuevo:




Create a new repository

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

Required fields are marked with an asterisk (*).

Owner *

 miguelmf1510

Repository name *

apWebTutorial

✓ apWebTutorial is available.

Great repository names are short and memorable. Need inspiration? How about [cuddly-octo-palm-tree](#) ?

Description (optional)

☐

Public

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

☒

Private

You choose who can see and commit to this repository.

Initialize this repository with:

☐

Add a README file

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

Add .gitignore

.gitignore template: None

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

Choose a license

License: None

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

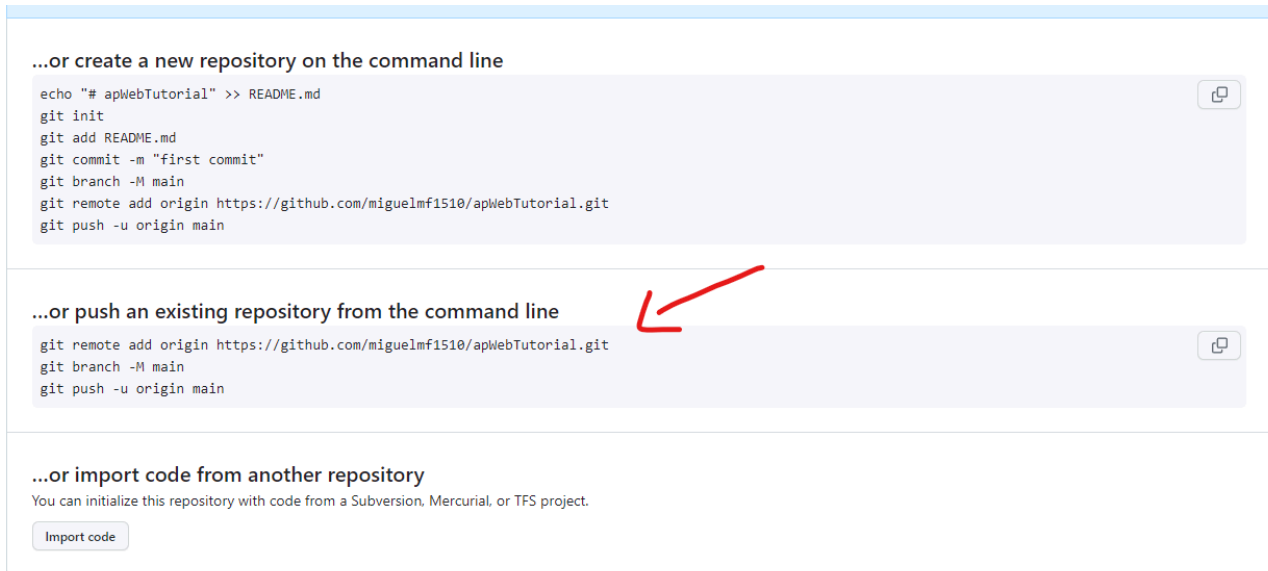


You are creating a private repository in your personal account.

Create repository

El nombre del repositorio tiene que ser el mismo que la carpeta del proyecto, no es obligatorio, pero recomendable para evitar confusiones. Además, vamos a hacerlo privado para que nadie externo pueda acceder a no ser que lo invitemos nosotros.

El repositorio ya estará creado y nos aparecerán unas instrucciones para enlazar este repositorio con uno local, como nosotros ya tenemos uno local creado, seguiremos las siguientes instrucciones pero con la particularidad de que nosotros hemos llamado a la rama "master", GitHub suele llamarla "main".



...or create a new repository on the command line

```
echo "# apWebTutorial" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/miguelmf1510/apWebTutorial.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/miguelmf1510/apWebTutorial.git
git branch -M main
git push -u origin main
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

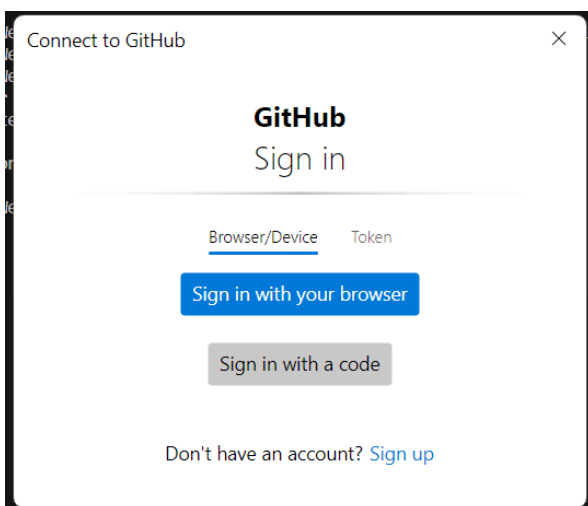
[Import code](#)

Vamos a ello:

```
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git remote add origin https://github.com/miguelmf1510/apWebTutorial.git
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git branch -M master
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git push -u origin master
```

Importante: recuerda cambiar “main” por “master”.

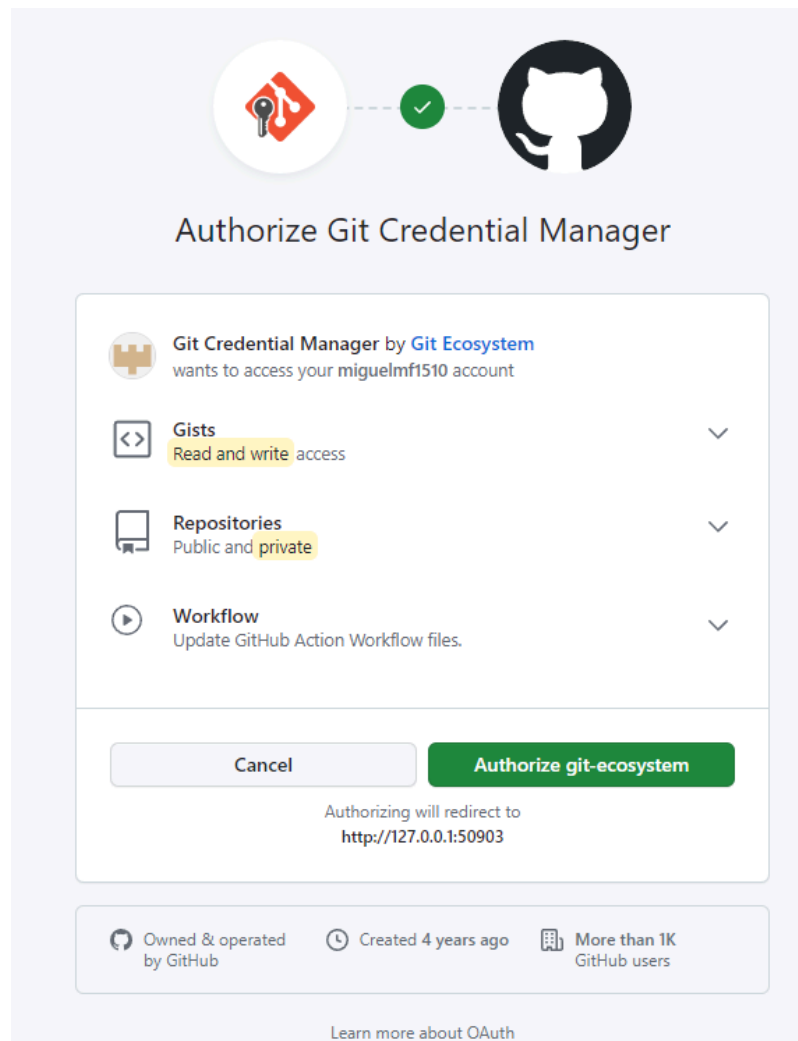
En resumen, lo que hemos hecho es añadir el remote a nuestro repositorio GitHub para que quede enlazada y estamos subiendo la rama master al repositorio remoto.



Cuando pongas el último comando, te aparecerá la autenticación con GitHub:

Elige la primera opción “Sign in with your browser”.

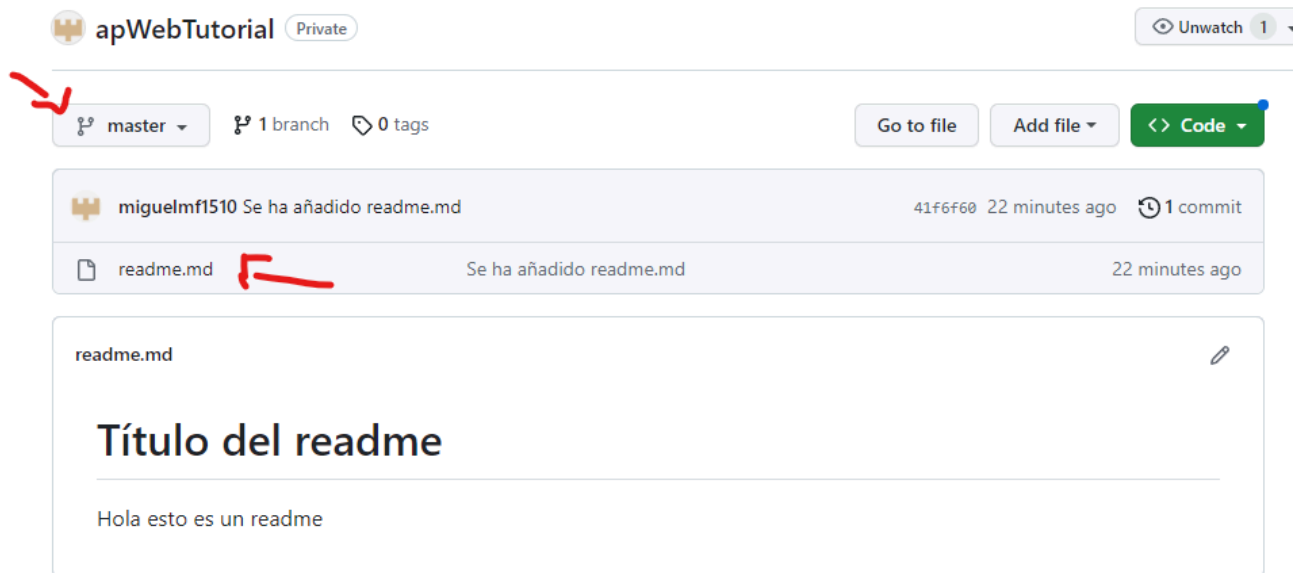
Y en la captura siguiente, autoriza.



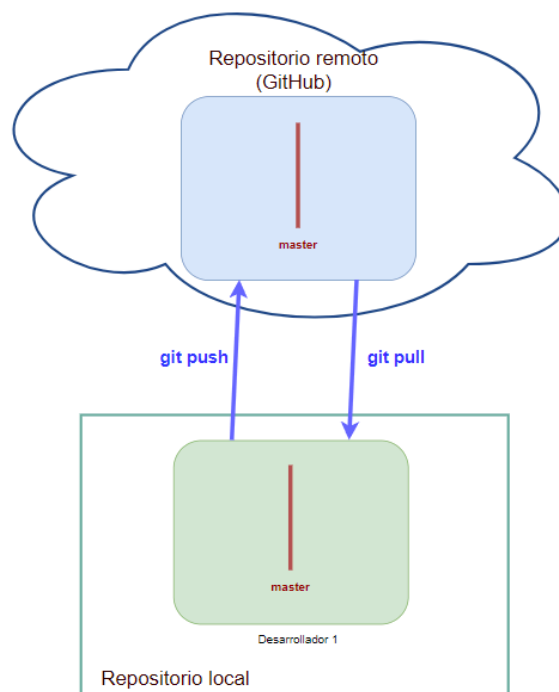
Al autenticarse, verás que ya finaliza la subida de la rama correctamente:

```
PS C:\Users\migue\pruebasGIT\apWebTutorial> git push -u origin master
info: please complete authentication in your browser...
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 266 bytes | 266.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/miguelmf1510/apWebTutorial.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
PS C:\Users\migue\pruebasGIT\apWebTutorial> 
```

Si refrescas vuelves a entrar a tu repositorio remoto, verás que ya podrás ver la rama y el archivo que hemos creado:



Ahora nos encontramos con el siguiente escenario:



Si ejecutamos el comando “git branch” con la opción -a (o --all) veremos que tenemos la rama master en local y en remoto:

```
PS C:\Users\migue\pruebasGIT\apWebTutorial> git branch -a
* master
remotes/origin/master
PS C:\Users\migue\pruebasGIT\apWebTutorial> |
```

Antes de seguir, vamos a crear el fichero **.gitignore**, es importante que lo tengamos creado, posteriormente lo modificaremos, simplemente crea el archivo vacío:

```
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git status
On branch master
Your branch is up to date with 'origin/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)
```

Ahora habría que añadirlo y hacer commit:

```
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git add .
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git commit -m ".gitignore añadido"
[master 74c231c] .gitignore añadido
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 .gitignore
PS C:\Users\miguel\pruebasGIT\apWebTutorial> █
```

Y subimos los cambios para que se apliquen también en el repositorio remoto:

```
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 291 bytes | 291.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/miguelmf1510/apWebTutorial.git
41f6f60..74c231c master -> master
PS C:\Users\miguel\pruebasGIT\apWebTutorial> █
```

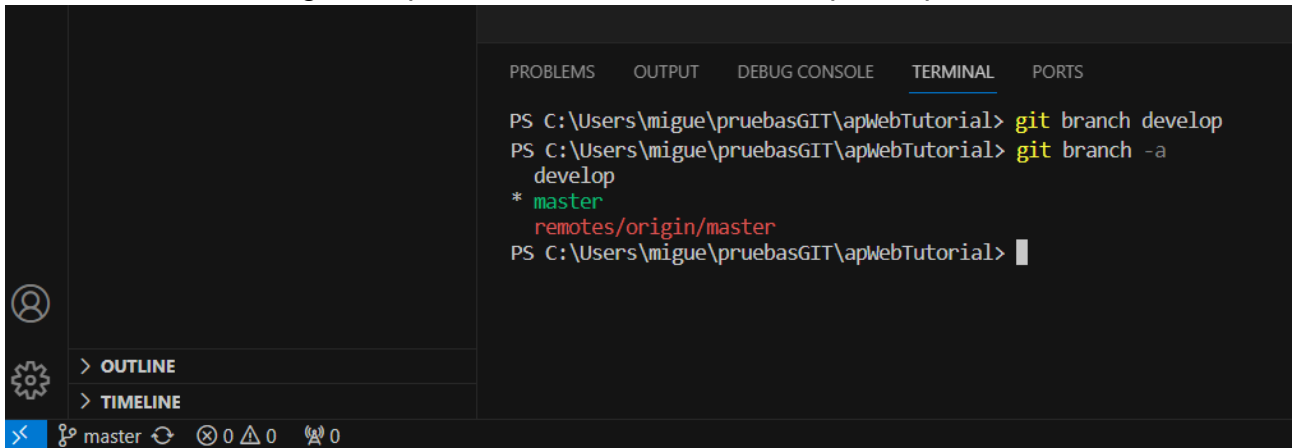
(Si no te funciona git push en este paso, utiliza "git push origin master")

9º Seguimos, como el GitFlow que utilizamos me indica que necesitamos trabajar con una segunda rama develop, vamos a crearla:

Podemos hacerlo de dos formas:

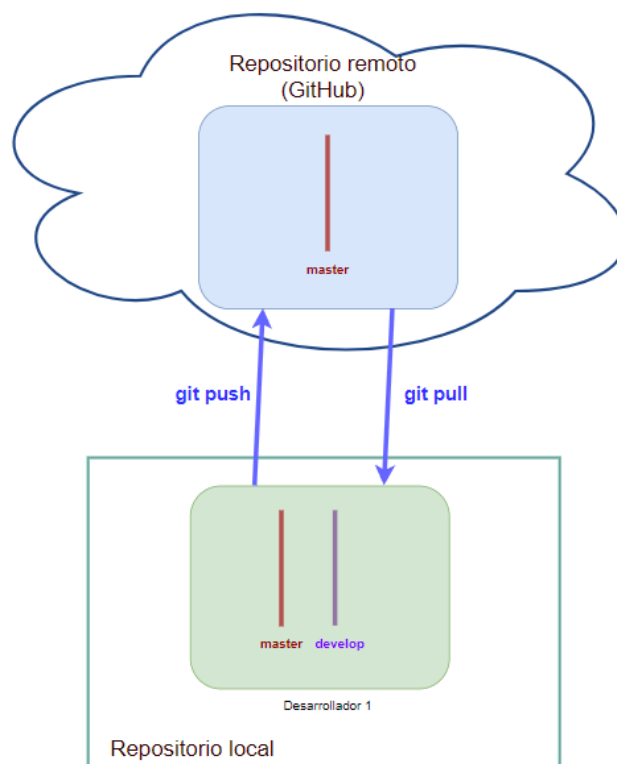
- `git branch <nombre rama>`: crear una nueva rama.
- `git checkout -b <nombre rama>`: crea la rama y se cambia a ella directamente.

Yo suelo utilizar la segunda, pero vamos a hacerlo en dos pasos para ver unos detalles:

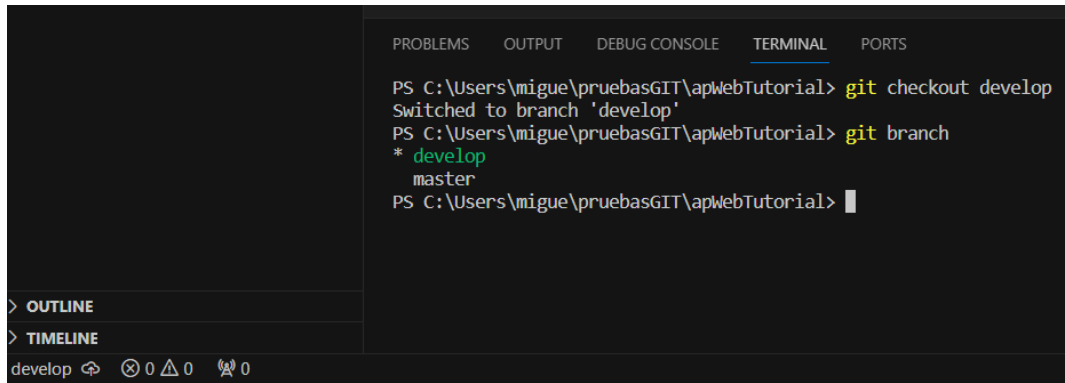


```
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git branch develop
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git branch -a
* master
remotes/origin/master
PS C:\Users\miguel\pruebasGIT\apWebTutorial>
```

Como vemos, se ha creado la rama en local, seguimos en master como indica el * y el color verde (y en la esquina inferior izquierda), pero la rama NO está en el repositorio remoto todavía, tendríamos el siguiente escenario:



Nos cambiamos a la nueva rama:



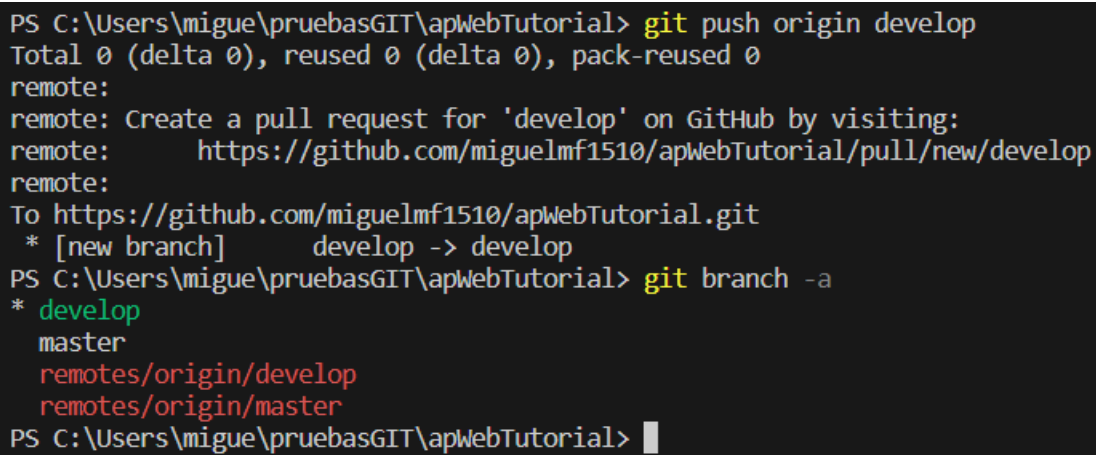
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\migue\pruebasGIT\apWebTutorial> git checkout develop
Switched to branch 'develop'
PS C:\Users\migue\pruebasGIT\apWebTutorial> git branch
* develop
  master
PS C:\Users\migue\pruebasGIT\apWebTutorial>

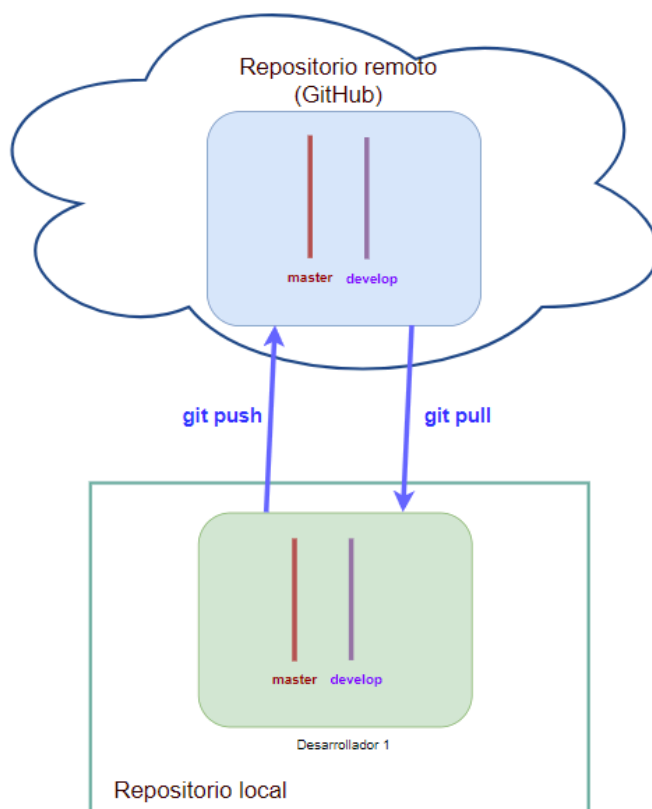
> OUTLINE
> TIMELINE
develop 0 0 0
```

Vemos que efectivamente hemos cambiado de rama.

Y ahora, subimos la rama haciendo simplemente “git push origin develop”.



```
PS C:\Users\migue\pruebasGIT\apWebTutorial> git push origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/miguelmf1510/apWebTutorial/pull/new/develop
remote:
To https://github.com/miguelmf1510/apWebTutorial.git
 * [new branch]      develop -> develop
PS C:\Users\migue\pruebasGIT\apWebTutorial> git branch -a
* develop
  master
  remotes/origin/develop
  remotes/origin/master
PS C:\Users\migue\pruebasGIT\apWebTutorial>
```

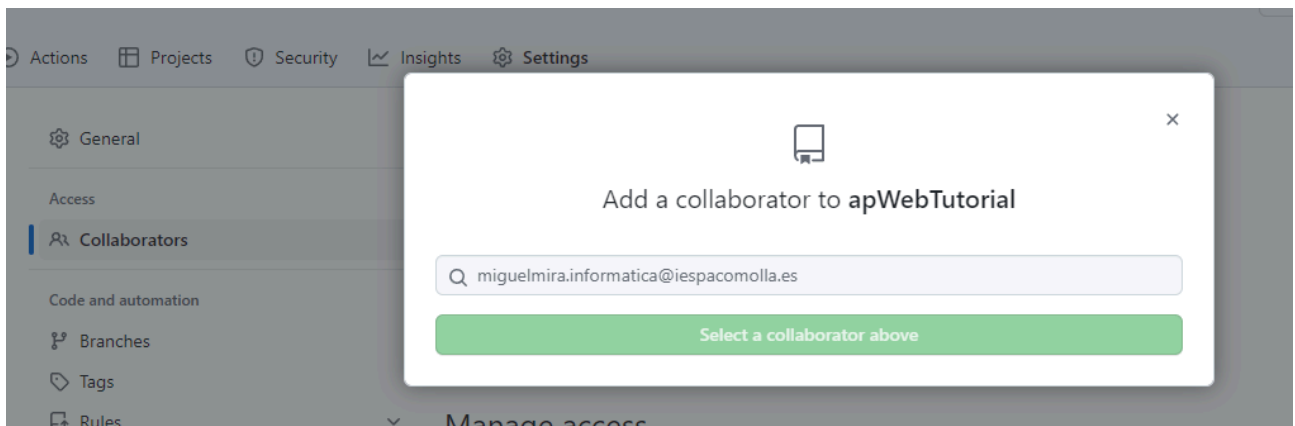
Ahora sí, el entorno ya está listo para que se puedan unir el resto de desarrolladores. (esto no es obligatorio, es decir, una vez esté el repositorio remoto se pueden unir los desarrolladores, pero a mi particularmente me gusta dejar el entorno bien preparado antes de que se empiecen a unir).

9º Esto no ha terminado, ahora nos toca preparar al desarrollador 2 y hacer que pueda trabajar también en el proyecto y por lo tanto, en este repositorio. (*Vamos a trabajar con la máquina virtual*).

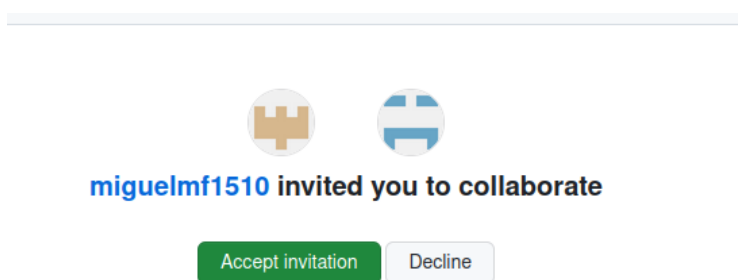
Primero, vamos a crear una nueva cuenta en GitHub para este segundo desarrollador.

Una vez tengamos ya las dos cuentas, desarrollador 1 y 2, vamos a invitar al desarrollador 2 como colaborador al repositorio remoto, para ello, ve a la cuenta del desarrollador 1, entra en el repositorio "apWebTutorial" y en "Settings".

Luego ve a "Collaborators" e invita el desarrollador 2:



Y acepta la invitación desde el email que te habrá llegado:



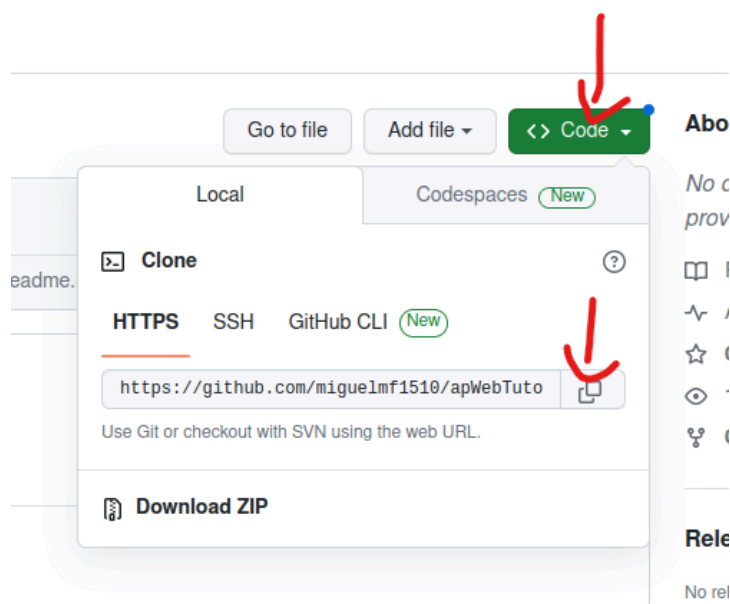
Mis desarrolladores son los siguientes:

Desarrollador 1: miguelmf1510

Desarrollador 2: mmfpacomolla

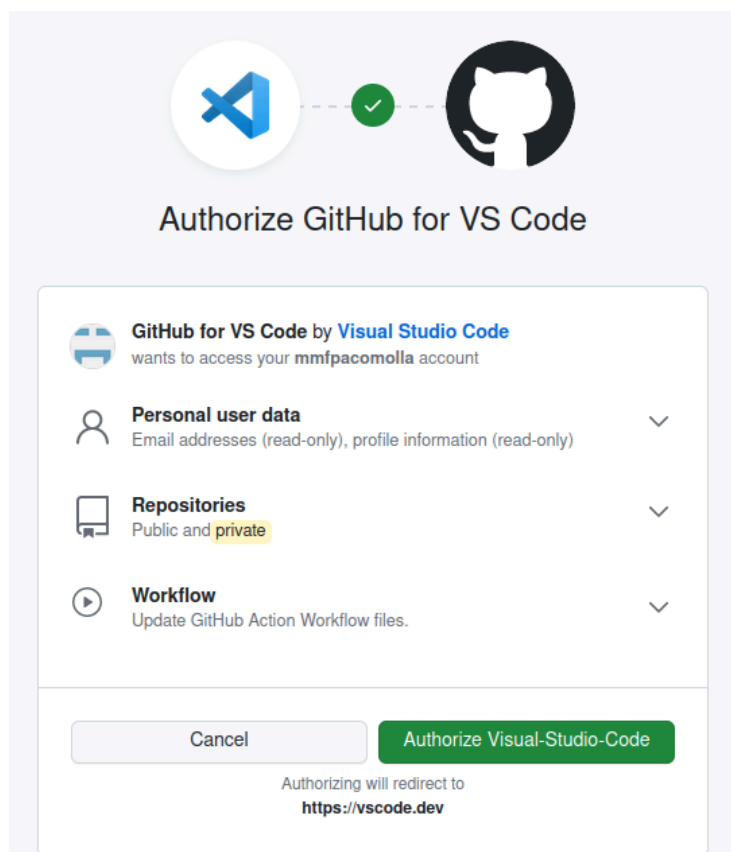
10º En este punto, el proyecto ya está empezado, es decir, ha sido creado e inicializado por el desarrollador 1, por lo tanto, el desarrollador 2 no necesita crear un nuevo repositorio local, simplemente va a CLONAR el repositorio remoto en local, de esa forma, el desarrollador 2 partirá desde el proyecto conforme en el estado que se encontraba en el momento de hacer la clonación.

Para ello, desde el equipo del desarrollador 2, si vamos a GitHub, podemos copiar el comando de clonado:



Vamos ahora a abrir VSCode (instálalo si no lo tienes en la máquina virtual), vamos al directorio donde queramos clonar el proyecto, y escribimos “git clone + la url del repositorio que hemos copiado” y lo ejecutamos:

Aquí vendrá ahora la autenticación y volvemos a autorizar todo:



```
• miguel@miguel-VirtualBox:~/pruebasGit$ git clone https://github.com/miguelmf1510/apWebTutorial.git
Clonando en 'apWebTutorial'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Recibiendo objetos: 100% (3/3), listo.
• miguel@miguel-VirtualBox:~/pruebasGit$
```

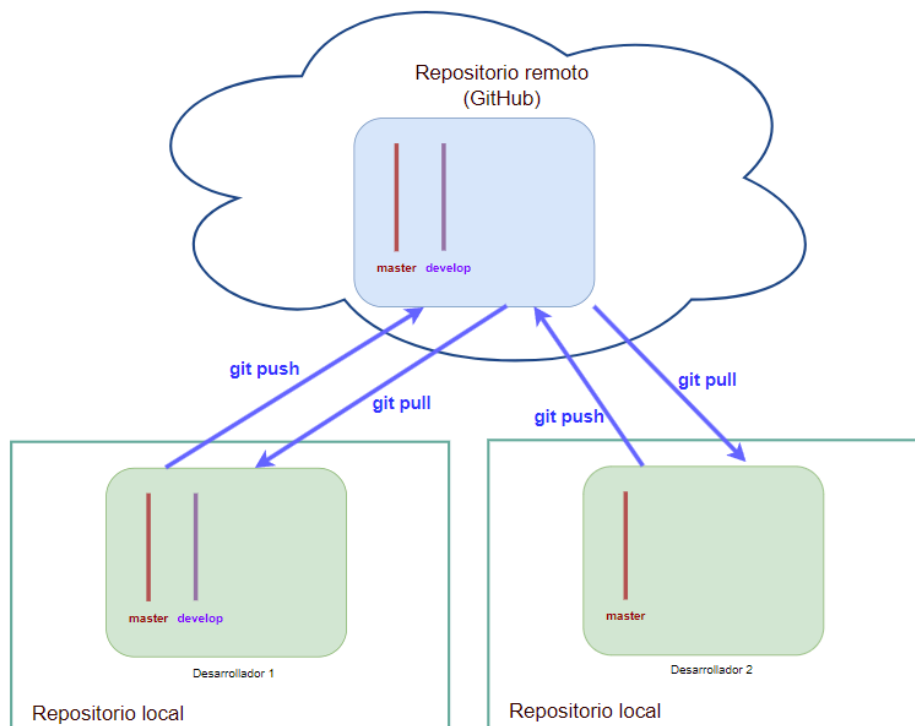
Y listo!, como vemos ya habremos clonado el repositorio.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
• miguel@miguel-VirtualBox:~/pruebasGit$ ls
apWebTutorial
• miguel@miguel-VirtualBox:~/pruebasGit$ cd apWebTutorial/
• miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ ls -la
total 16
drwxrwxr-x 3 miguel miguel 4096 sep 20 10:12 .
drwxrwxr-x 3 miguel miguel 4096 sep 20 10:11 ..
drwxrwxr-x 8 miguel miguel 4096 sep 20 10:12 .git
-rw-rw-r-- 1 miguel miguel  43 sep 20 10:12 readme.md
• miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$
```

Vamos ahora a abrir el proyecto con VSCode, una vez abierto, vamos a visualizar las ramas:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
• miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/develop
  remotes/origin/master
• miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$
```

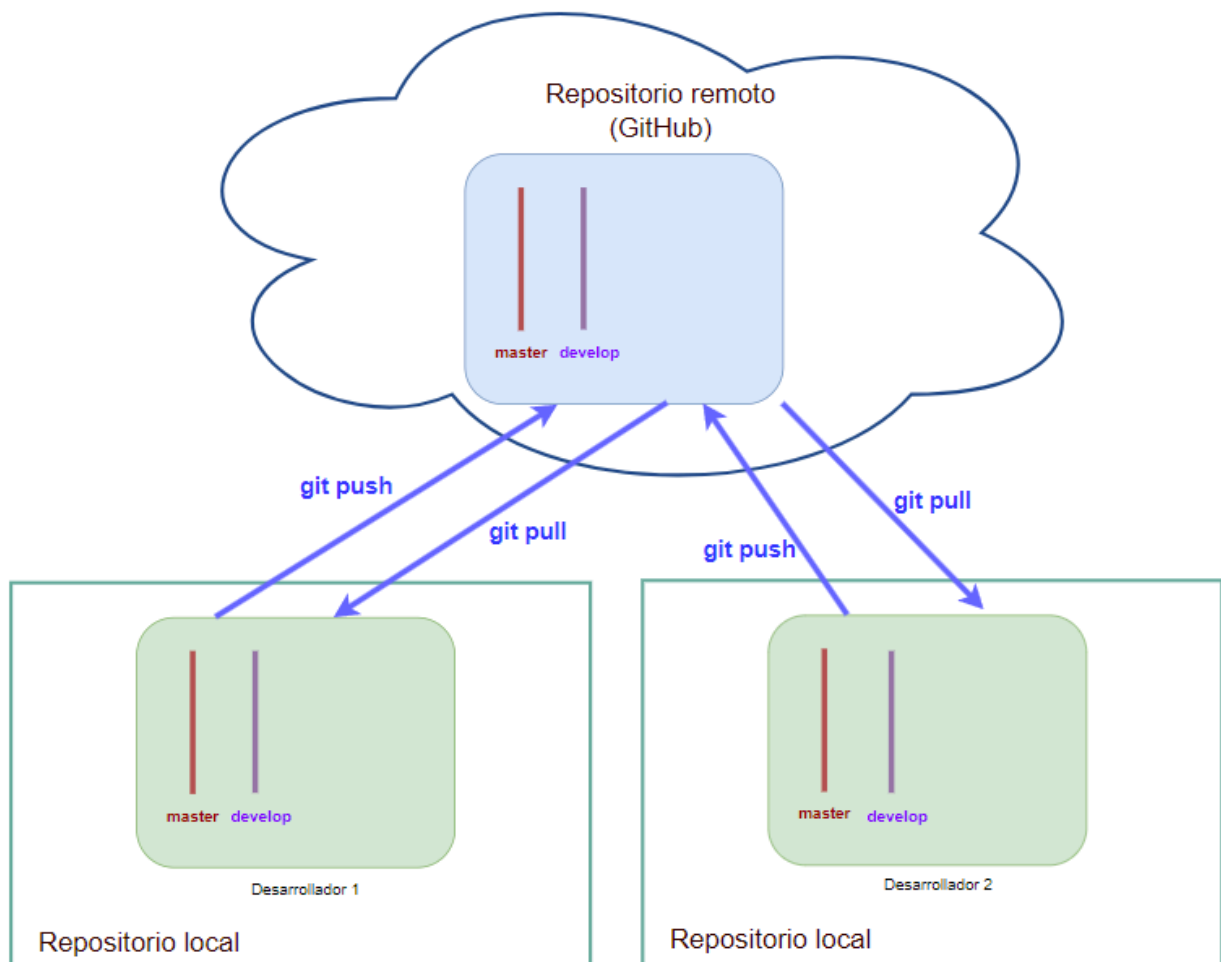
Como puedes observar, nos encontramos en la rama master y únicamente se ha descargado en nuestro repositorio local la rama master, esto pasa siempre que clonamos, se descarga únicamente la rama master, el escenario quedaría ahora así:



Vamos a descargar la rama develop, para ello simplemente nos cambiamos a la rama con checkout:

```
● miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git checkout develop
Rama 'develop' configurada para hacer seguimiento a la rama remota 'develop' de 'origin'.
Cambiado a nueva rama 'develop'
● miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git branch -a
* develop
  master
  remotes/origin/HEAD -> origin/master
  remotes/origin/develop
  remotes/origin/master
○ miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$
```

Para descargar ramas, simplemente nos tenemos que cambiar a ellas, siempre y cuando nos aparezca la rama en ese listado (`git branch -a`), si no, deberemos ejecutar primer `git fetch` para actualizar cambios que no estén reflejados (lo probaremos más adelante).



Ahora sí, ya tenemos el proyecto listo y a los dos desarrolladores preparados para empezar a desarrollar. Si quisiéramos añadir más colaboradores, sería repetir estos últimos pasos que hemos visto.

3. ¿CÓMO TRABAJAR EN EQUIPO?

A continuación, vamos a hacer algunas pruebas con este entorno que hemos creado, vamos a hacer que el desarrollador 1 haga la página de Login y el desarrollador 2 haga la página de Registro, y vamos a ver cómo fusionar ambos trabajos.

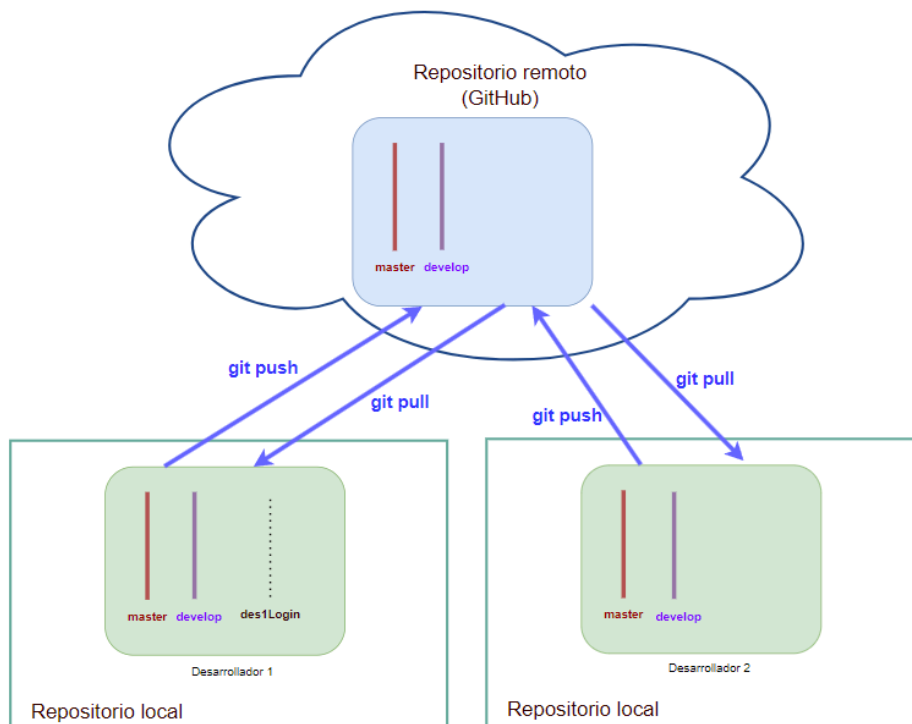
Vamos al desarrollador 1, estamos trabajando actualmente en la rama develop.

```
PS C:\Users\migue\pruebasGIT\apWebTutorial> git branch
* develop
  master
PS C:\Users\migue\pruebasGIT\apWebTutorial> 
```

Como dijimos, NO SE TRABAJA directamente en esta rama, si no, lo que vamos a hacer es crear una rama para desarrollar el Login:

```
PS C:\Users\migue\pruebasGIT\apWebTutorial> git checkout -b des1Login
Switched to a new branch 'des1Login'
PS C:\Users\migue\pruebasGIT\apWebTutorial> git branch -a
* des1Login
  develop
  master
remotes/origin/develop
remotes/origin/master
PS C:\Users\migue\pruebasGIT\apWebTutorial> 
```

Hemos creado y nos hemos cambiado a la rama, en estos momentos, esta rama se encuentra solo en local, hasta que no hagamos un push, esta rama no pasará al repositorio remoto:



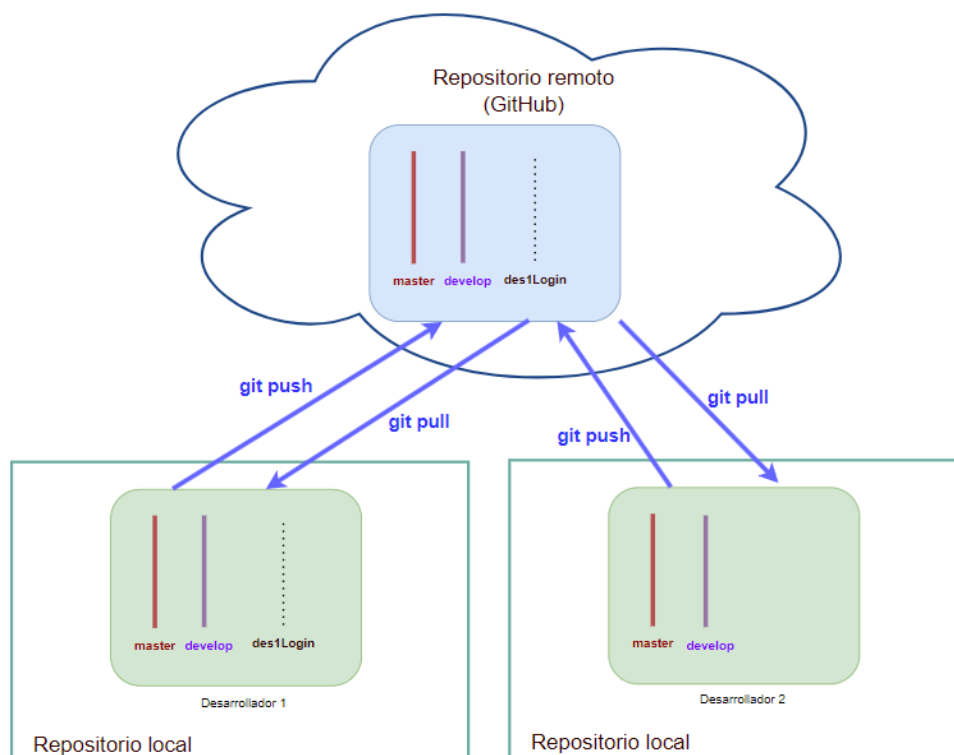
También comentar que la nueva rama comenzará con una copia exacta del proyecto de la rama desde donde la hemos creado. Prueba a cambiar entre develop y des1Login y verás que el contenido de la carpeta apWebTutorial no cambia, ya que tienen lo mismo.

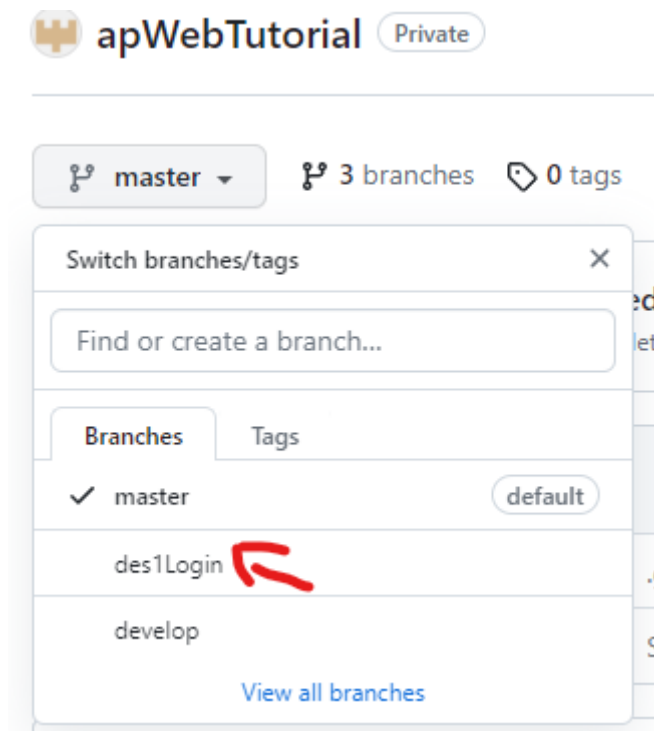
En este punto, podemos hacer 2 cosas, o bien subir directamente la rama para tenerla ya en remoto, o empezar a desarrollar y una vez tengamos algún commit subir los cambios (de esta forma, se sube la rama y los cambios hechos).

A mi particularmente me gusta tener sincronizado el local y el remoto, si no, es posible que empecemos a desarrollar y a hacer commits y que se nos olvide ir subiendo los cambios, si pasa esto, hay un apagón y se estropea nuestro ordenador, perderemos los cambios. Por lo tanto, yo recomiendo ir subiendo de vez en cuando los cambios, al estar en nuestra propia rama, no hay peligro, el resto del proyecto está aislado.

Por lo tanto, vamos a subir la rama aunque no tenga nada:

```
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git push origin des1Login
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'des1Login' on GitHub by visiting:
remote:   https://github.com/miguelmf1510/apWebTutorial/pull/new/des1Login
remote:
To https://github.com/miguelmf1510/apWebTutorial.git
 * [new branch]      des1Login -> des1Login
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git branch -a
* des1Login
develop
master
remotes/origin/des1Login
remotes/origin/develop
remotes/origin/master
PS C:\Users\miguel\pruebasGIT\apWebTutorial> █
```





Vamos a hacer un inciso, vámonos ahora al desarrollador 2, y vamos a listar todas las ramas:

```
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git branch -a
* develop
  master
  remotes/origin/HEAD -> origin/master
  remotes/origin/develop
  remotes/origin/master
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$
```

¿Qué ha pasado con **des1Login**? aunque esté en el repositorio remoto, git branch -a NO envía una petición al repositorio para traer las ramas, si no que el repositorio local tiene información almacenado del remoto, por lo tanto, no hay una sincronización directa de los cambios en el remoto, debemos hacerla manual.

Para actualizar esa información, debemos hacer uso de “**git fetch**” (OJO: git fetch no descarga los cambios de las ramas, eso se hace con git pull).

```
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git fetch
Desde https://github.com/miguelmf1510/apWebTutorial
* [nueva rama]      des1Login -> origin/des1Login
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git branch -a
* develop
  master
  remotes/origin/HEAD -> origin/master
  remotes/origin/des1Login
  remotes/origin/develop
  remotes/origin/master
  virtualBox:~/pruebasGit/apWebTutorial$
```

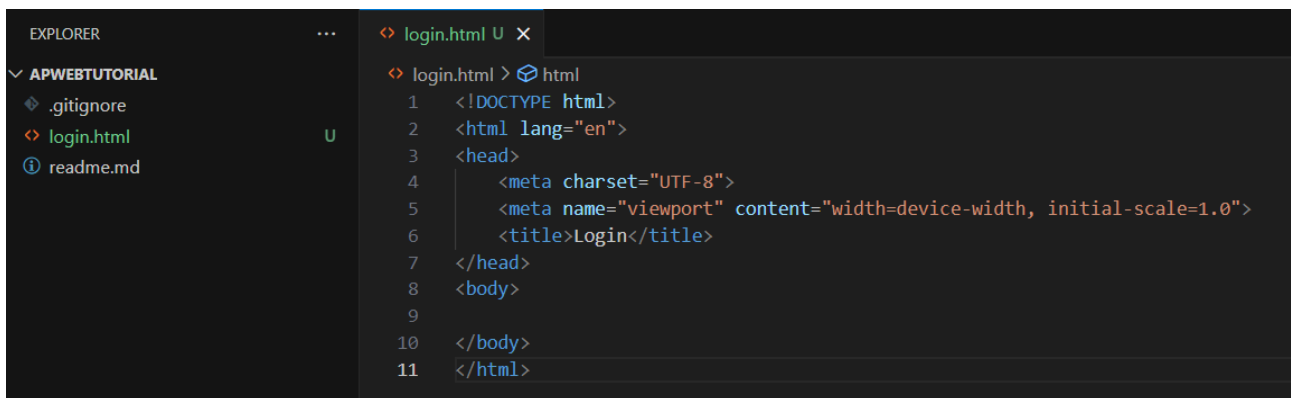

Ahora sí, vemos como al actualizar la información, ya podemos ver que en el remoto hay una nueva rama, pudiendo incluso cambiarnos a ella, pero esto no es lo habitual, vamos a dejar a cada desarrollador con su rama.

Vamos a continuar, volvemos al desarrollador 1, y nos encontrábamos en la nueva rama.

Vamos entonces a desarrollar el Login, aquí ya no hay una regla específica de cuándo hacer commits, no hay que terminar la funcionalidad por completo para poder hacer commit, pero tampoco vamos a hacer un commit cada vez que añadamos una línea de código.

- Al final, es estudiar la tarea a realizar, si es muy sencilla, con un solo commit valdría ya que los cambios han sido mínimos.
- Si la tarea es más difícil, se puede dividir en subtareas e ir haciendo commit conforme se vaya cumpliendo cada subtarea.

Subtarea 1: crear el fichero con un código plantilla:

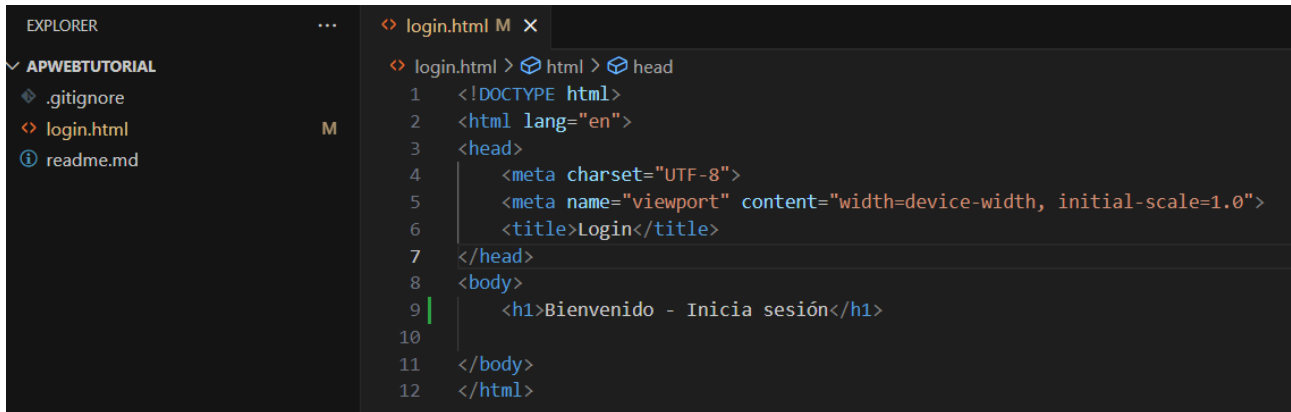
A screenshot of a code editor interface. On the left, the 'EXPLORER' sidebar shows a file tree for 'APWEBTUTORIAL' with files '.gitignore', 'login.html' (highlighted with an orange icon), and 'readme.md'. The main editor area shows the content of 'login.html' with line numbers 1 through 11. The code is an HTML template with a doctype, lang attribute, charset, viewport, and title 'Login'.

Salta en verde cuando es un nuevo archivo, cuando se modifique un archivo saldrá en naranja.

Hago un commit ya que he cumplido la primera tarea:

```
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git add .
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git commit -m "Crear login.html"
[des1Login cddfd3c] Crear login.html
1 file changed, 11 insertions(+)
create mode 100644 login.html
PS C:\Users\miguel\pruebasGIT\apWebTutorial> █
```

Subtarea 2: Crea el título de la página de login

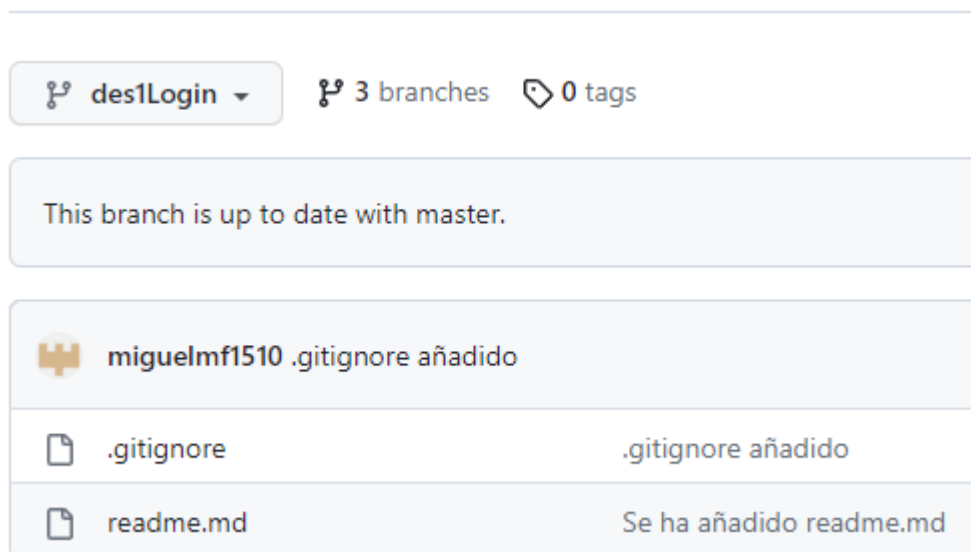


```
login.html M x
login.html > html > head
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Login</title>
7 </head>
8 <body>
9   <h1>Bienvenido - Inicia sesión</h1>
10
11 </body>
12 </html>
```

Como vemos, ahora sale naranja, ya que es un archivo que ya ha sido añadido al repositorio local desde un commit y volvemos a hacer cambios en él.

```
PS C:\Users\migue\pruebasGIT\apWebTutorial> git add .
PS C:\Users\migue\pruebasGIT\apWebTutorial> git commit -m "Añadir título página login"
[des1Login 90375f6] Añadir título página login
1 file changed, 1 insertion(+)
PS C:\Users\migue\pruebasGIT\apWebTutorial> 
```

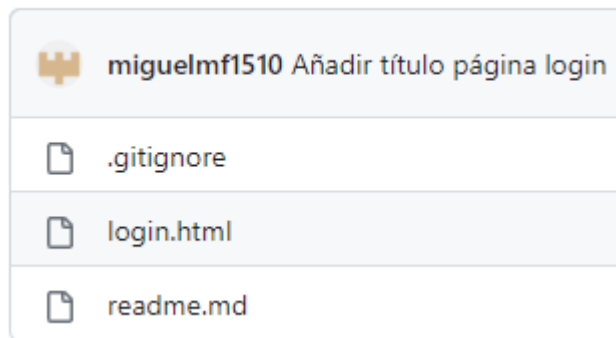
Vamos un momento a GitHub, entra en la rama “des1Login” para ver los archivos que hay:



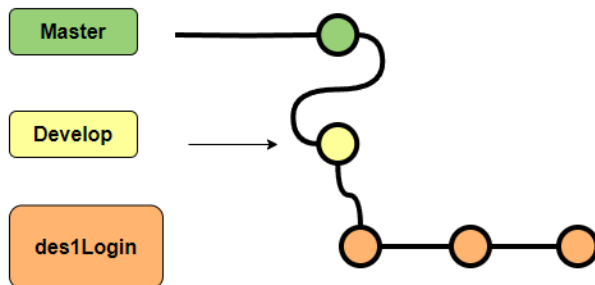
Como puedes ver, NO está el archivo login.html, porque todo lo que hemos hecho anteriormente ha sido en local, por lo tanto, vamos a subir los cambios:

```
PS C:\Users\migue\pruebasGIT\apWebTutorial> git push origin des1Login
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 826 bytes | 826.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/miguelmf1510/apWebTutorial.git
 74c231c..90375f6  des1Login -> des1Login
PS C:\Users\migue\pruebasGIT\apWebTutorial> █
```

Si refrescas la página de GitHub, verás que ahora sí aparecen los cambios.

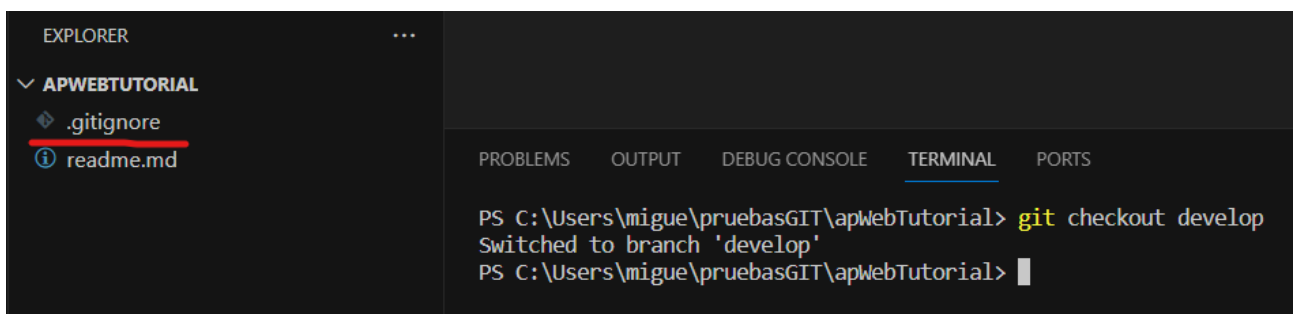


Bien, ahora nos encontramos en el siguiente escenario:



La rama des1Login ha ido avanzando, se ha ido desarrollando parte de la funcionalidad, que de momento únicamente se encuentra en esta rama.

Prueba a cambiar de rama y observa como el fichero login.html desaparece tanto en develop como en master, eso significa que estos cambios realizados están únicamente en la rama des1Login.



Pero si vuelves a des1Login, vuelve a aparecer.

Esto es el potencial de GIT, la posibilidad de tener en un mismo proyecto varios avances, de esa forma, un avance no entra en conflicto con el otro y podemos recuperar versiones anteriores del proyecto.

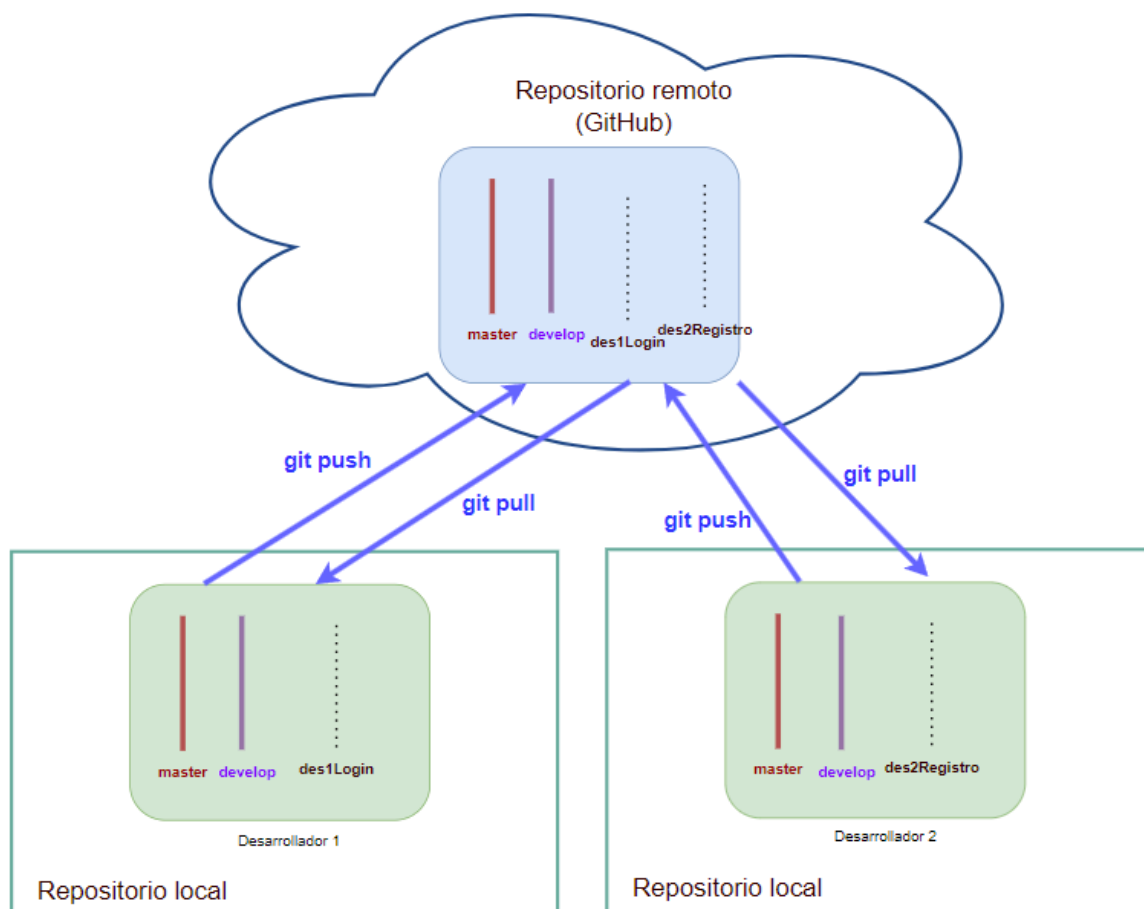
Ahora, si quisiéramos, podríamos eliminar la rama des1Login y seguir por donde nos habíamos quedado antes de crearla, como si nada hubiera pasado. Imagínate que te has equivocado y has roto el proyecto estando en tu rama, no pasa nada, vuelve a develop y elimínala.

IMPORTANTE: para poder moverte de una rama a otra, no deben haber cambios pendientes sin commit, si no, no dejará moverte.

Vale, ahora vámonos al otro desarrollador, ya que él debe empezar su tarea de Registro.

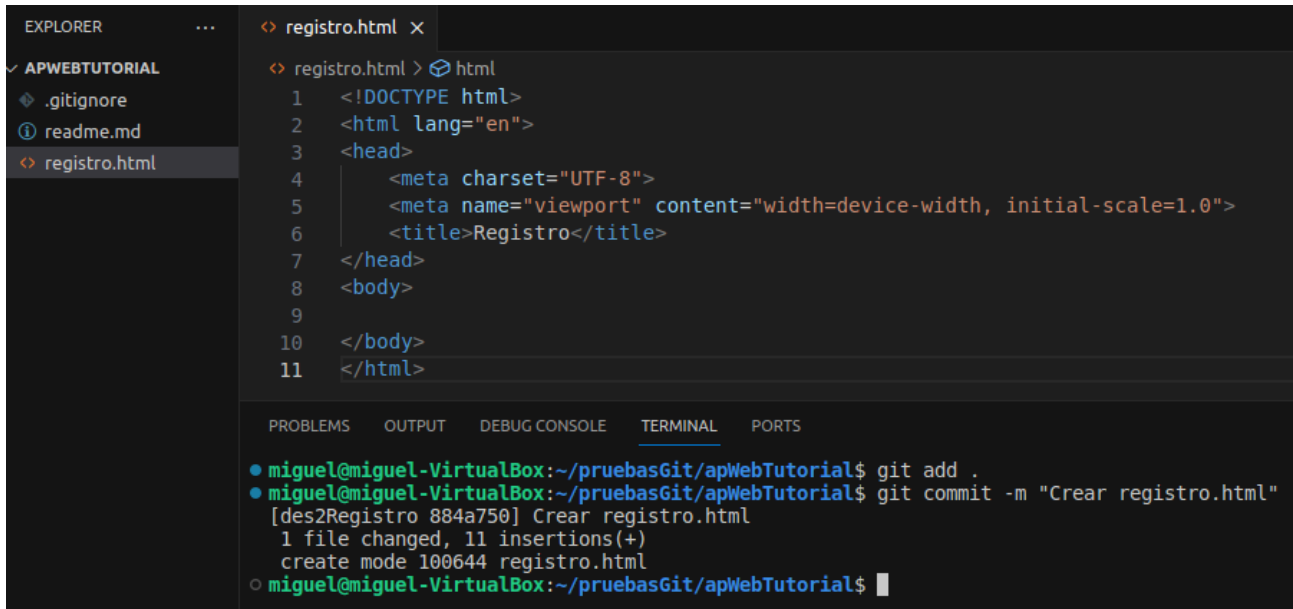
Desde develop, creamos la rama des2Registro y la subimos al repositorio.

```
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git branch
* develop
  master
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git checkout -b des2Registro
Cambiado a nueva rama 'des2Registro'
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git push origin des2Registro
Total 0 (delta 0), reusados 0 (delta 0), pack-reusados 0
remote:
remote: Create a pull request for 'des2Registro' on GitHub by visiting:
remote:   https://github.com/miguelf1510/apWebTutorial/pull/new/des2Registro
remote:
To https://github.com/miguelf1510/apWebTutorial.git
 * [new branch]      des2Registro -> des2Registro
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$
```



Ahora ya podríamos empezar a desarrollar el registro. Como vemos login.html no nos aparecerá, porque como hemos dicho eso se encuentra en la rama des1Login, posteriormente uniremos ambos trabajos.

Ahora vamos a hacer lo mismo, creamos el registro.html plantilla, hacemos commit y luego ponemos un título, hacemos otro commit y subimos los cambios:

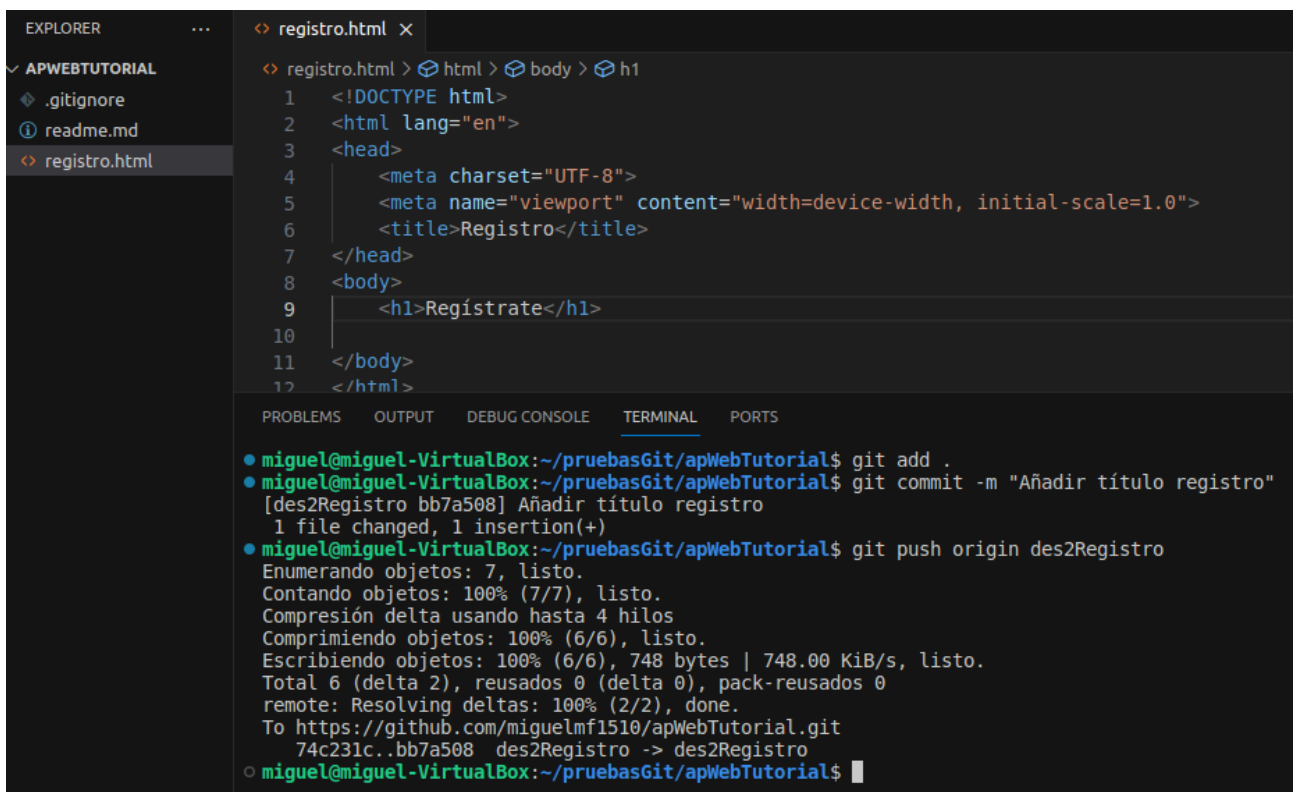


The screenshot shows the VS Code interface with the Explorer sidebar on the left showing the file structure of 'APWEBTUTORIAL' with files .gitignore, readme.md, and registro.html. The main editor shows the content of 'registro.html' with the following code:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Registro</title>
</head>
<body>
</body>
</html>
```

Below the editor, the TERMINAL tab is active, showing the following commands and output:

```
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git add .
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git commit -m "Crear registro.html"
[des2Registro 884a750] Crear registro.html
1 file changed, 11 insertions(+)
create mode 100644 registro.html
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$
```

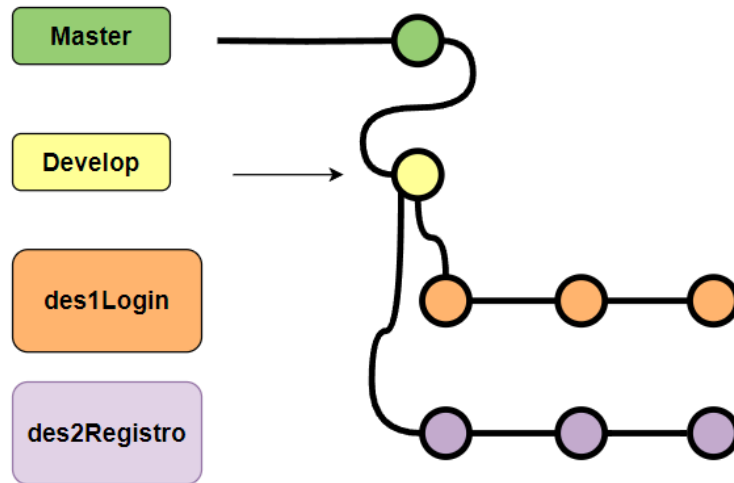


The screenshot shows the VS Code interface with the Explorer sidebar on the left showing the file structure of 'APWEBTUTORIAL' with files .gitignore, readme.md, and registro.html. The main editor shows the content of 'registro.html' with the following code:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Registro</title>
</head>
<body>
  <h1>Regístrate</h1>
</body>
</html>
```

Below the editor, the TERMINAL tab is active, showing the following commands and output:

```
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git add .
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git commit -m "Añadir título registro"
[des2Registro bb7a508] Añadir título registro
1 file changed, 1 insertion(+)
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git push origin des2Registro
Enumerando objetos: 7, listo.
Contando objetos: 100% (7/7), listo.
Compresión delta usando hasta 4 hilos
Comprimiendo objetos: 100% (6/6), listo.
Escribiendo objetos: 100% (6/6), 748 bytes | 748.00 KiB/s, listo.
Total 6 (delta 2), reusados 0 (delta 0), pack-reusados 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/miguelmf1510/apWebTutorial.git
74c231c..bb7a508 des2Registro -> des2Registro
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$
```



Ahora nos encontramos en este punto, cada desarrollador tiene su rama y el proyecto ha avanzado por caminos distintos, es hora de juntar ambos desarrollos.

Primero, se juntan los desarrollos en develop, se comprueba que esté bien y por último, si queremos que el proyecto salga a producción, se llevarían los cambios a master.

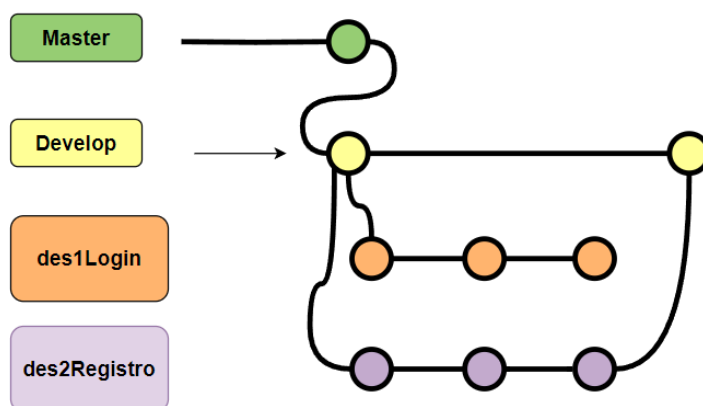
Supongamos que el desarrollador 2 ha terminado ya el registro, por lo tanto, lo que deberá hacer es mezclar los cambios con la rama develop, para ello, hay que cambiarse a la rama develop para traer los cambios, es decir, a la hora de hacer “merge”, se hace desde la rama a la que queremos llevar los cambios:

```

• miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git checkout develop
Cambiado a rama 'develop'
Tu rama está actualizada con 'origin/develop'.
• miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git branch
  des2Registro
* develop
  master
• miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git merge des2Registro
Actualizando 74c231c..bb7a508
Fast-forward
 registro.html | 12 ++++++++
 1 file changed, 12 insertions(+)
 create mode 100644 registro.html
• miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$

```

Y ya estaría, develop ya tiene los cambios de des2Registro:



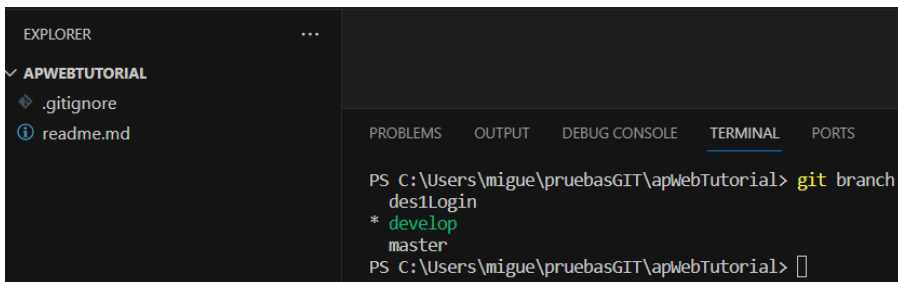
PEROO, estos cambios se han hecho en local, ve a GitHub y compruébalo si quieres, verás cómo develop no tiene registro.html.

Para ello, estando en develop con los nuevos cambios, los subimos:

```
● miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git push origin develop
Total 0 (delta 0), reusados 0 (delta 0), pack-reusados 0
To https://github.com/miguelmf1510/apWebTutorial.git
74c231c..bb7a508 develop -> develop
○ miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$
```

Ahora sí, ya están los cambios en remoto.

Vale, ahora vamos a llevar los cambios del desarrollador 1 a develop también haciendo los mismo pasos, pero ahora hay algo diferente vamos a verlo:



```
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git branch
des1Login
* develop
master
PS C:\Users\miguel\pruebasGIT\apWebTutorial>
```

La rama develop local del desarrollador 1, todavía no tiene los cambios del desarrollador 2, porque están en remoto.

Por lo tanto, llegado a

este punto, yo recomiendo la siguiente **norma**:

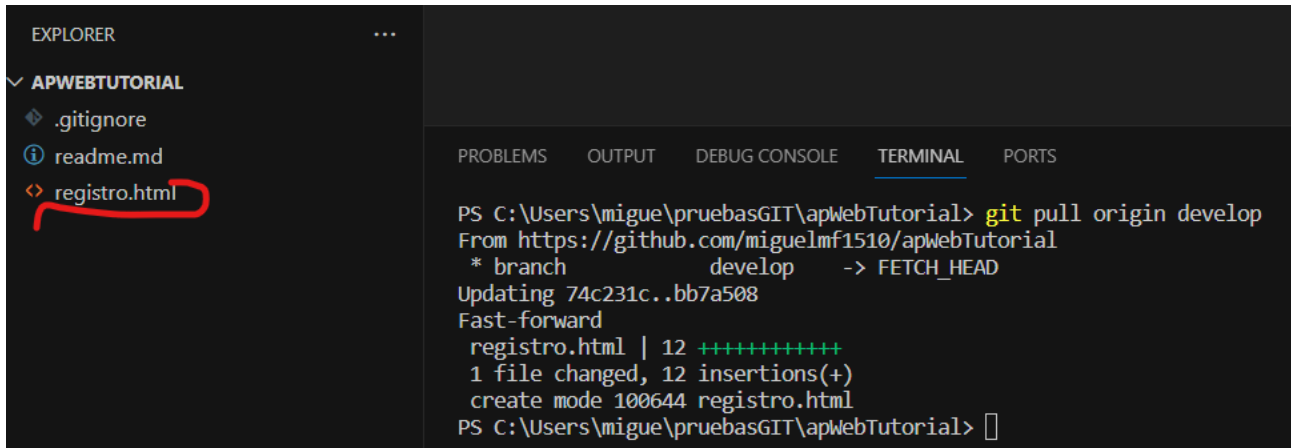
- **Ejecuta “git pull” antes de hacer merge con una rama.**
- De esa forma, nos descargaremos primero los posibles cambios que haya en esa rama.
- Y después, podremos hacer tranquilamente un merge.

Como estamos en ese caso, develop en remoto tiene más cosas que develop en local, pues vamos a primero traernos esos cambios, si hacemos un **diff** con develop de remoto, veremos que efectivamente hay cambios que todavía no tenemos en local:

```
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git diff origin/develop
diff --git a/registro.html b/registro.html
deleted file mode 100644
index 42249cb..0000000
--- a/registro.html
+++ /dev/null
@@ -1,12 +0,0 @@
<!DOCTYPE html>
<html lang="en">
<head>
- <meta charset="UTF-8">
- <meta name="viewport" content="width=device-width, initial-scale=1.0">
- <title>Registro</title>
</head>
<body>
- <h1>Regístrate</h1>
-
</body>
</html>
\ No newline at end of file
```

(tendrás que hacer git fetch, antes del diff si quieres comprobar esto).

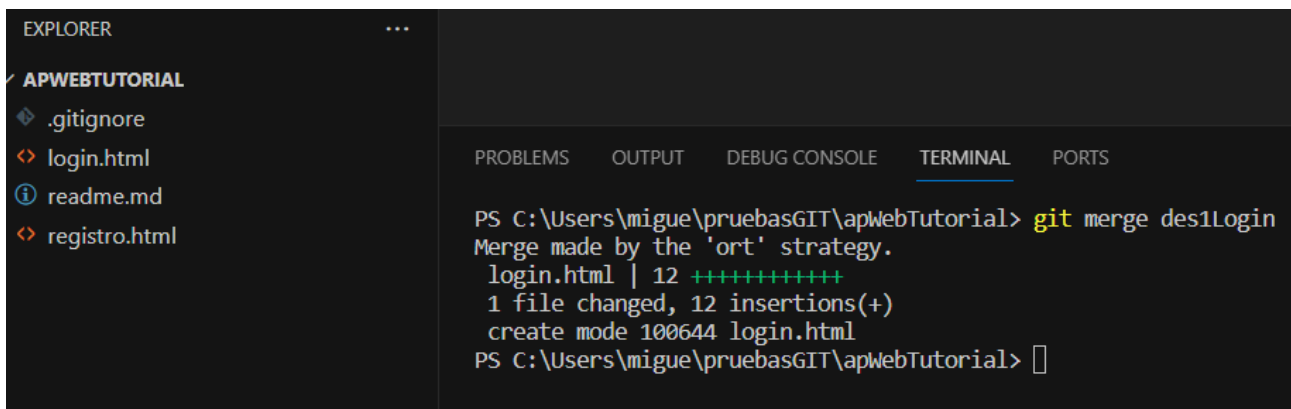
Vale, pues vamos a traernos esos cambios:



The screenshot shows the VS Code interface. On the left, the Explorer pane displays the file structure of the 'APWEBTUTORIAL' project, with 'registro.html' highlighted by a red circle. On the right, the Terminal pane shows the output of a 'git pull' command. The output indicates that the 'develop' branch was fetched and merged into the local 'develop' branch, resulting in 12 insertions in 'registro.html'.

```
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git pull origin develop
From https://github.com/miguelmf1510/apWebTutorial
 * branch                develop      -> FETCH_HEAD
Updating 74c231c..bb7a508
Fast-forward
 registro.html | 12 ++++++++
 1 file changed, 12 insertions(+)
 create mode 100644 registro.html
PS C:\Users\miguel\pruebasGIT\apWebTutorial>
```

Ya tenemos los cambios de la funcionalidad del registro, ahora vamos a mezclarlos con la funcionalidad del Login. Ahora sí, hacemos merge con la rama des1Login:

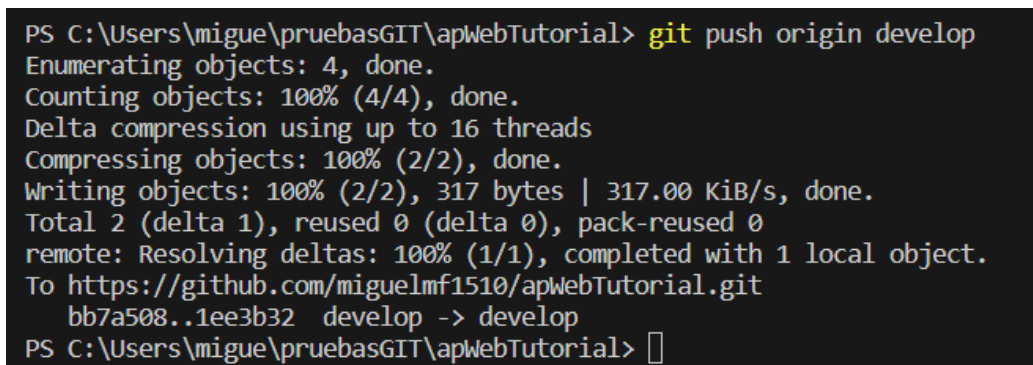


The screenshot shows the VS Code interface. On the left, the Explorer pane displays the file structure of the 'APWEBTUTORIAL' project, with 'login.html' and 'registro.html' highlighted. On the right, the Terminal pane shows the output of a 'git merge' command. The output indicates that the 'des1Login' branch was merged into the local 'develop' branch, resulting in 12 insertions in 'login.html'.

```
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git merge des1Login
Merge made by the 'ort' strategy.
 login.html | 12 ++++++++
 1 file changed, 12 insertions(+)
 create mode 100644 login.html
PS C:\Users\miguel\pruebasGIT\apWebTutorial>
```

Y ya está, no ha habido conflicto en el merge ya que se ha trabajado en ficheros distintos, luego haremos una prueba de conflicto.

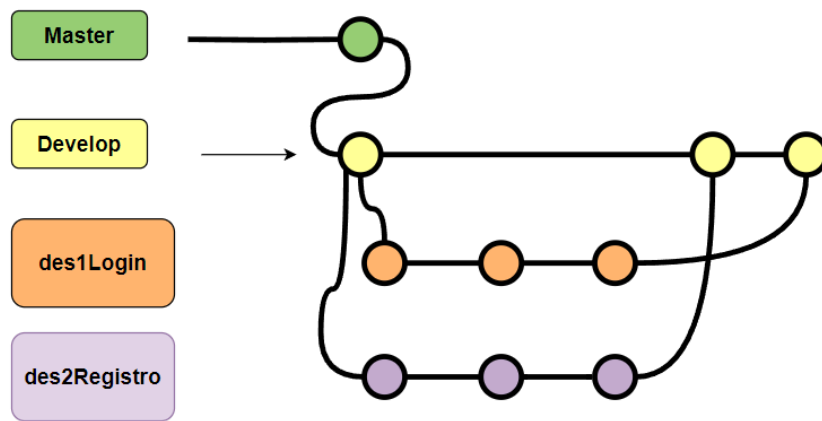
Simplemente nos queda llevar este cambio al remoto ya que este merge se ha hecho de forma local (puedes ir a GitHub y ver que no está el login.html):



The screenshot shows a terminal window with the output of a 'git push' command. The output indicates that the local 'develop' branch was pushed to the remote 'develop' branch, resulting in 1 local object being pushed.

```
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git push origin develop
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 317 bytes | 317.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/miguelmf1510/apWebTutorial.git
 bb7a508..1ee3b32 develop -> develop
PS C:\Users\miguel\pruebasGIT\apWebTutorial>
```

Y ahora si, funcionalidades unificadas correctamente en develop!!



Y ahora otra **norma**, ve al desarrollador 2, mira en qué rama estás y el contenido del proyecto. No está login.html ¿Verdad?, efectivamente, ya que esa unificación la hemos hecho en el ordenador del desarrollador 1 y los cambios se han subido al remoto, pero en ningún momento hemos descargado esos cambios en el desarrollador 2. Por lo tanto, la norma será:

- **Antes de empezar una nueva funcionalidad desde develop, realizar un “git pull”.**
- De esta forma, sabes que vas a empezar una nueva funcionalidad partiendo del proyecto completamente actualizado hasta el momento.

```

APWEBTUTORIAL
├── .gitignore
├── login.html
├── README.md
└── registro.html

miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git pull
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 8 (delta 2), reused 8 (delta 2), pack-reused 0
Desempaquetando objetos: 100% (8/8), 1.05 KiB | 536.00 KiB/s, listo.
Desde https://github.com/miguelmf1510/apWebTutorial
    bb7a508..1ee3b32 develop    -> origin/develop
    74c231c..90375f6 des1Login -> origin/des1Login
Actualizando bb7a508..1ee3b32
Fast-forward
 login.html | 12 ++++++++
 1 file changed, 12 insertions(+)
 create mode 100644 login.html
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$

```

Ahora si, el desarrollador 2 ya podría crear una nueva rama y empezar una nueva funcionalidad.

4. PULL REQUEST

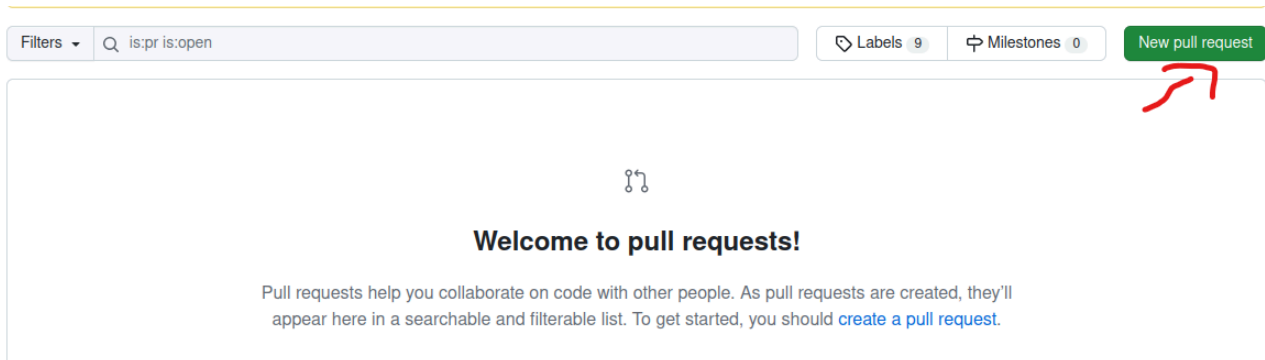
Las funcionalidades las tenemos ahora en develop, pero ¿Qué pasa con master?.

El paso de llevar el código a master significa que es para llevarlo a producción, es decir, código que está listo para publicarse y utilizarse por los clientes, por lo tanto, este momento es mucho más crítico.

Para ello, hay una acción llamada “Pull Request” que consiste en realizar una petición al jefe de proyecto para hacer que develop unifique los cambios con master, es decir, hacer un merge entre develop y master.

Esta petición se realiza para pedir permiso al jefe de proyecto, este hará las revisiones oportunas y podrá aceptarla o rechazarla.

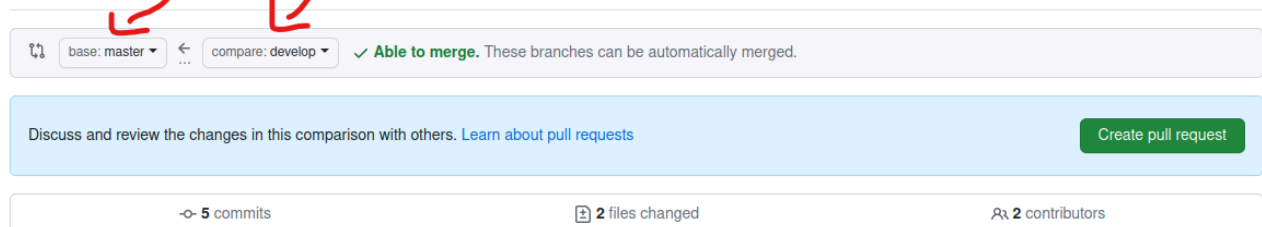
Por lo tanto, después de haber desarrollado estas dos funcionalidades (Login y Registro), vamos a llevarlas a producción, para ello, vamos a hacer que el desarrollador 2 es quién quiere llevar el código a producción, para ello, accede al GitHub del desarrollador 2 y a la sección “Pull requests”:



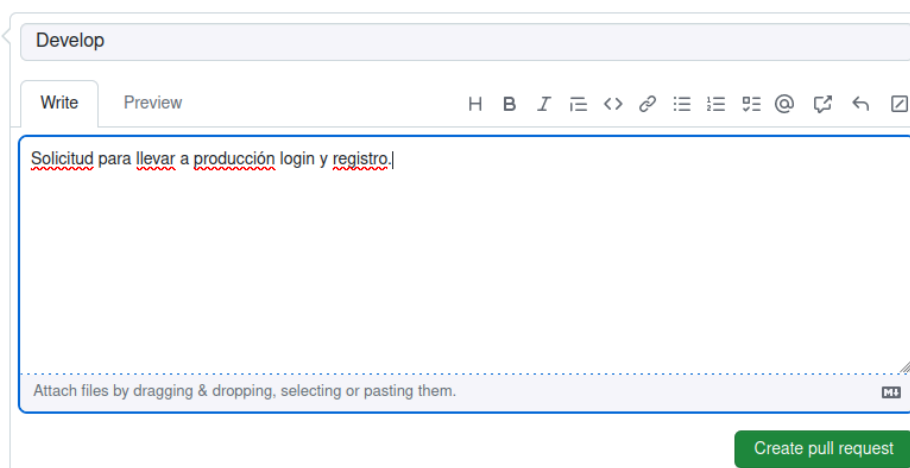
Click en nuevo pull request e indicamos las ramas a mezclar, en este caso, queremos llevar los cambios de develop a master:

Comparing changes

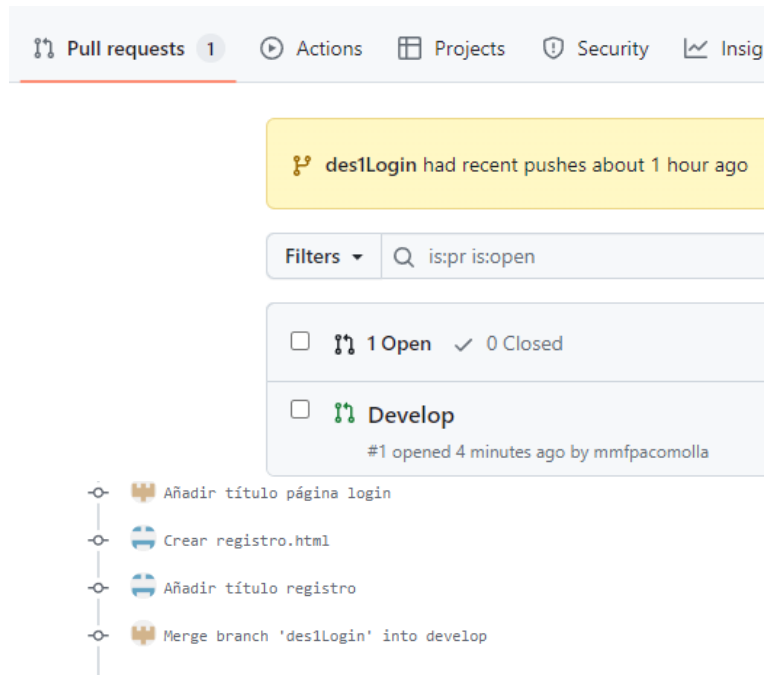
Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).



Se añade un comentario explicando la solicitud:

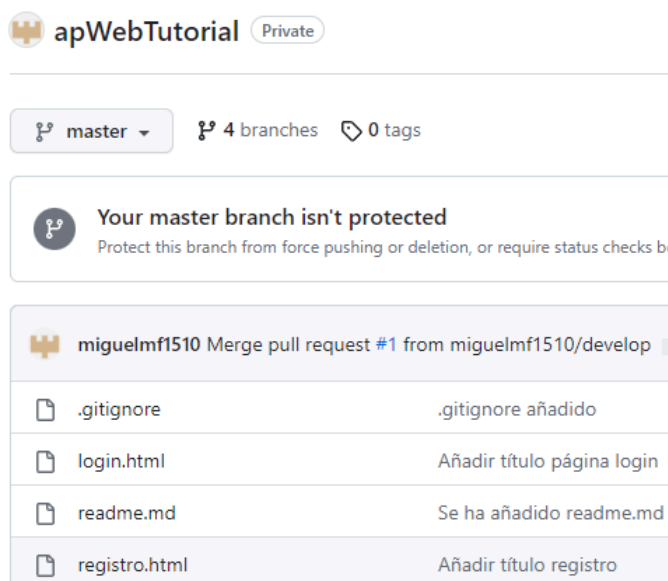
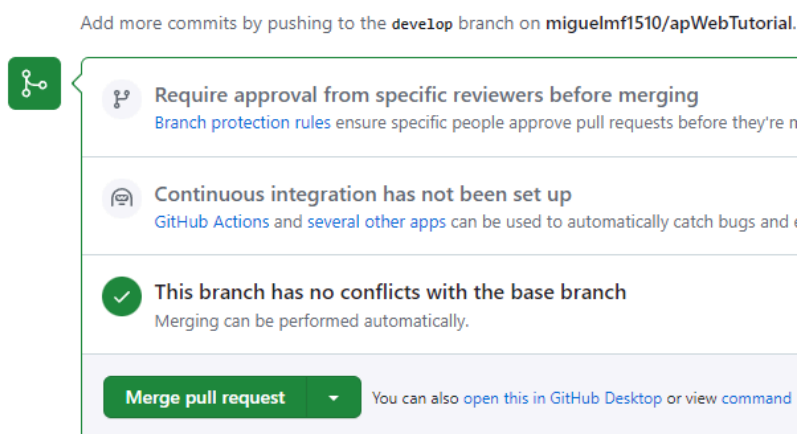


Ahora nos iríamos al GitHub del desarrollador 1, él es propietario del repositorio, por lo tanto, es él quien debe aceptar esta petición, vamos a “Pull requests”, seleccionamos el que hay creado:



Y un poco más abajo, click en “Merge pull request”:

Ya aparecerá resuelto y si vas a la rama master en GitHub, verás que ya tiene todos los cambios:



Ahora, si quisiéramos de forma local tener la rama master actualizada, podríamos hacer un pull en la rama master de cada repositorio local y traer los cambios.

Hasta aquí se ha visto un ciclo completo del GitFlow, ahora os queda practicar mucho, y eso consiste en ir probando e investigando.

5. GIT STASH

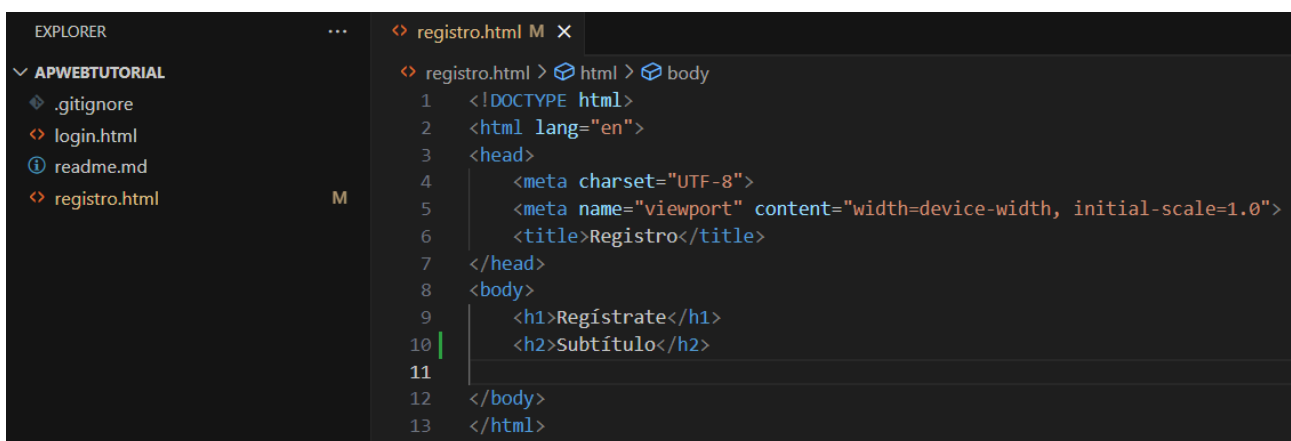
Vamos a ver esta acción, imagínate que eres el desarrollador 1, empiezas una funcionalidad y a mitad de esta funcionalidad te piden que arregles urgentemente un bug, eso significa dejar de lado la funcionalidad para ponerte a corregir el bug, vamos a ello:

Crea una nueva rama llamada “des1Func1”:

```
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git branch
des1Login
* develop
master
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git pull origin develop
From https://github.com/miguelmf1510/apWebTutorial
* branch          develop    -> FETCH_HEAD
Already up to date.
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git checkout -b des1Func1
Switched to a new branch 'des1Func1'
PS C:\Users\miguel\pruebasGIT\apWebTutorial> 
```

Fíjate los pasos que he seguido, primero compruebo que estoy en develop para empezar la nueva rama, después, actualizo develop por si hubiera habido algún cambio (en este caso no lo había) y finalmente creo y me cambio a la rama.

Ahora vamos a realizar algún cambio, en el registro por ejemplo, vamos a poner un subtítulo:



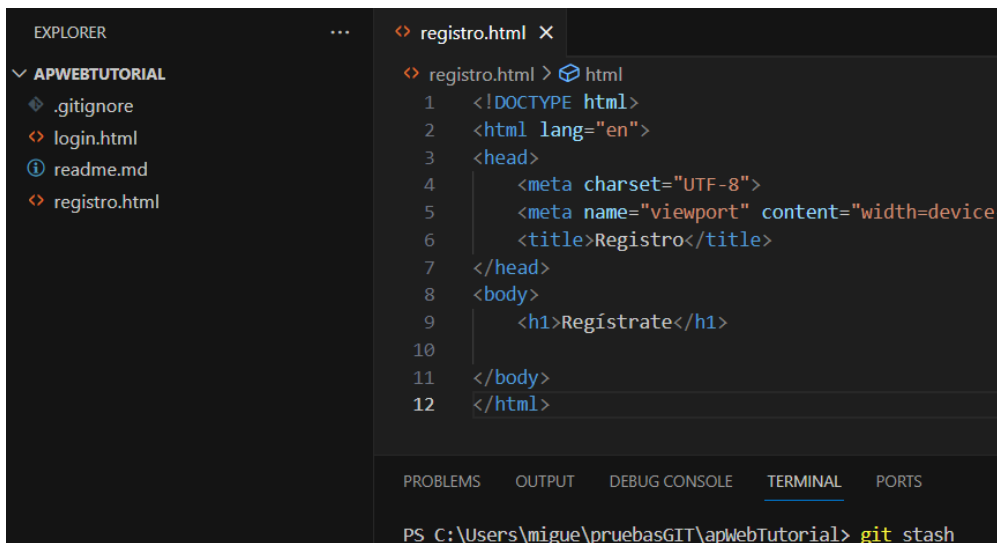
```
EXPLORER
...
v APWEBTUTORIAL
  .gitignore
  login.html
  readme.md
  registro.html

registro.html M x
registro.html > html > body
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Registro</title>
7 </head>
8 <body>
9   <h1>Regístrate</h1>
10  <h2>Subtítulo</h2>
11
12 </body>
13 </html>
```

Y ahora resulta que en este punto, me avisan de que debo corregir un importante bug en el login, y por lo tanto, tengo que dejar de hacer lo que estoy haciendo.

Eso consiste en irse a develop y dejar el trabajo pendiente.

Pero tengo el trabajo a medias y no quiero hacer commit todavía ¿Qué hago? git stash, dejo en un segundo plano el trabajo pendiente y después lo recupero:



```

EXPLORER
  APWEBTUTORIAL
    .gitignore
    login.html
    readme.md
    registro.html

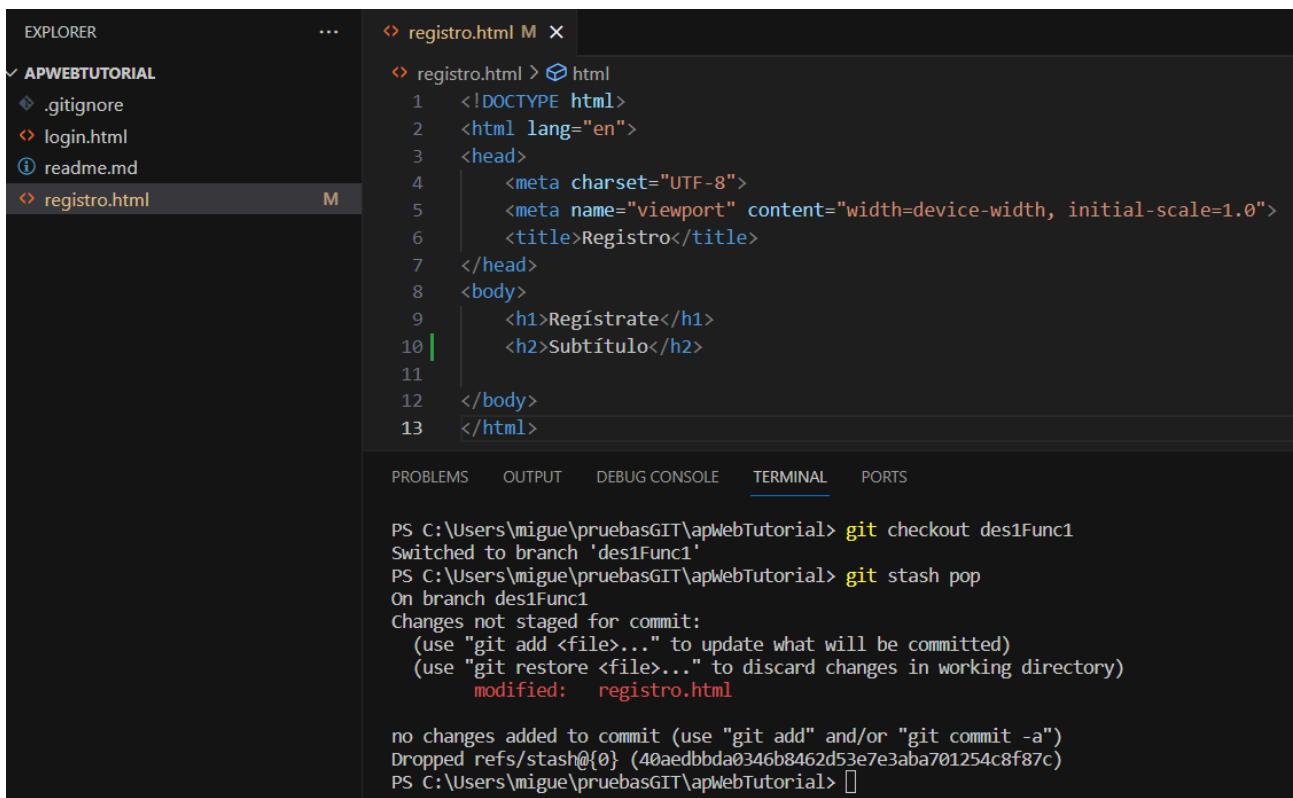
registro.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Registro</title>
7 </head>
8 <body>
9   <h1>Regístrate</h1>
10
11 </body>
12 </html>

TERMINAL
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git stash
  
```

Al hacer git stash todo el trabajo que tenía sin confirmar (commit) ha desaparecido, por lo tanto, ya está el entorno limpio para poder cambiarme a develop.

```
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git checkout develop
```

Me cambio a develop, realizo todo lo que tenga que realizar, y cuando termine, vuelvo de nuevo a des1Func1 y recupero los cambios con “git stash pop”:



```

EXPLORER
  APWEBTUTORIAL
    .gitignore
    login.html
    readme.md
    registro.html M

registro.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Registro</title>
7 </head>
8 <body>
9   <h1>Regístrate</h1>
10  <h2>Subtítulo</h2>
11
12 </body>
13 </html>

TERMINAL
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git checkout des1Func1
Switched to branch 'des1Func1'
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git stash pop
On branch des1Func1
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:   registro.html

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (40aedbbda0346b8462d53e7e3aba701254c8f87c)
PS C:\Users\miguel\pruebasGIT\apWebTutorial> 
  
```

Si quieres ignorar esta funcionalidad, puedes hacer git stash, cambiarte a develop y eliminar la rama.

6. CONFLICTOS

Los conflictos se producen cuando dos desarrolladores trabajan sobre el mismo archivo y mismas líneas (es posible trabajar en un mismo archivo y que no hayan conflictos).

Vamos a provocar un conflicto y vamos a ver cómo se solucionan.

Para ello, el desarrollador 1 y 2 van a trabajar sobre el mismo fichero, concretamente con login.html, por lo tanto, vamos a crear la rama “des1Conflicto” y “des2Conflicto”, cada una la creará el desarrollador en cuestión, primero el desarrollador 1:

```
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git branch
des1Login
* develop
master
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git pull origin develop
From https://github.com/miguelmf1510/apWebTutorial
* branch          develop    -> FETCH_HEAD
Already up to date.
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git checkout -b des1Conflicto
Switched to a new branch 'des1Conflicto'
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git push origin des1Conflicto
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'des1Conflicto' on GitHub by visiting:
remote:   https://github.com/miguelmf1510/apWebTutorial/pull/new/des1Conflicto
remote:
To https://github.com/miguelmf1510/apWebTutorial.git
* [new branch]    des1Conflicto -> des1Conflicto
```

Haz lo mismo pero ahora en el desarrollador 2.

Ahora vamos al desarrollador 1 añade estos cambios:

```
<> login.html M ●
<> login.html > html > body
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Login</title>
7  </head>
8  <body>
9      ...<h1>Bienvenido/a de nuevo - Inicia sesión</h1>
10     ...<h2>Subtitulo</h2>
11
12
13     ...<p>Si tienes problemas para iniciar sesión contacta con el administrador</p>
14 </body>
15 </html>
```

Ahora vamos al desarrollador 2 y añade estos cambios:

```
<h1>INICIO de Sesión - bienvenido</h1>
```

Ahora, procederemos a lo mismo que hicimos en el punto 3, confirmaremos los cambios, subiremos los cambios y los llevaremos a develop, siempre con el siguiente orden y comandos.

Vamos a suponer que el desarrollador 1 ha terminado antes, por lo tanto, vamos a seguir los siguientes pasos:

- Add+Commit
- Push de des1Conflicto
- Cambio a develop
- Pull para asegurar que no hay cambios nuevos y si los hay se descargan.
- Merge de des1Conflicto a develop
- Push de develop

```
PS C:\Users\migue\pruebasGIT\apWebTutorial> git add .
PS C:\Users\migue\pruebasGIT\apWebTutorial> git commit -m "Cambios de login.html"
[des1Conflicto be4a01d] Cambios de login.html
 1 file changed, 5 insertions(+), 2 deletions(-)
PS C:\Users\migue\pruebasGIT\apWebTutorial> git push origin des1Conflicto
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 513 bytes | 513.00 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 https://github.com/miguelf1510/apWebTutorial.git
 1ee3b32..be4a01d  des1Conflicto -> des1Conflicto
PS C:\Users\migue\pruebasGIT\apWebTutorial> git checkout develop
Switched to branch 'develop'
PS C:\Users\migue\pruebasGIT\apWebTutorial> git pull origin develop
From https://github.com/miguelf1510/apWebTutorial
 * branch                develop      -> FETCH_HEAD
Already up to date.
PS C:\Users\migue\pruebasGIT\apWebTutorial> git merge des1Conflicto
Updating 1ee3b32..be4a01d
Fast-forward
 login.html | 7 +++++--
```

Ahora vamos al desarrollador 2, realizamos primero estos pasos:

- Add + Commit
- Push de des2Conflicto
- Cambio a develop


```

● miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git add .
● miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git commit -m "Cambios login.html"
[des2Conflicto 0193f56] Cambios login.html
 1 file changed, 1 insertion(+), 1 deletion(-)
● miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git push origin des2Conflicto
Enumerando objetos: 5, listo.
Contando objetos: 100% (5/5), listo.
Compresión delta usando hasta 4 hilos
Comprimiendo objetos: 100% (3/3), listo.
Escribiendo objetos: 100% (3/3), 359 bytes | 359.00 KiB/s, listo.
Total 3 (delta 2), reusados 0 (delta 0), pack-reusados 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/miguelf1510/apWebTutorial.git
   lee3b32..0193f56  des2Conflicto -> des2Conflicto
● miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git checkout develop
Cambiado a rama 'develop'
Tu rama está actualizada con 'origin/develop'.
○ miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$

```

Ahora deberían hacerse los siguientes pasos:

- Pull para asegurar que no hay cambios nuevos y si los hay se descargan.
- Merge de des2Conflicto a develop
- Push de develop

Pero van a cambiar un poco porque va a haber un conflicto, vamos a realizar el primero, que es el pull, de esta forma nos traeremos los cambios, que en este caso sí los hay, son los que ha realizado el desarrollador 1, vamos a descargarlos y verlos:

```

 7  </head>
 8  <body>
 9      <h1>Bienvenido/a de nuevo - Inicia sesión</h1>
10      <h2>Subtitulo</h2>
11
12
13      <p>Si tienes problemas para iniciar sesión contacta con el administrador</p>
14  </body>
15  </html>

```

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
● miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git pull origin develop
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 3 (delta 1), pack-reused 0
Desempaquetando objetos: 100% (3/3), 493 bytes | 493.00 KiB/s, listo.
Desde https://github.com/miguelf1510/apWebTutorial
 * branch          develop      -> FETCH HEAD
   lee3b32..be4a01d  develop      -> origin/develop
Actualizando lee3b32..be4a01d
Fast-forward
 login.html | 7 +++++-
 1 file changed, 5 insertions(+), 2 deletions(-)
○ miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$

```

Efectivamente, ya tenemos los cambios. Ahora en el segundo paso (merge) es donde viene el conflicto ya que nos vamos a traer de des2Conflicto los cambios que hemos hecho, como coinciden las líneas, cuando realice el merge, se producirá el conflicto, vamos a ello:


```

8  <body>
  Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
9  <<<<<< HEAD (Current Change)
10  <h1>Bienvenido/a de nuevo - Inicia sesión</h1>
11  <h2>Subtitulo</h2>
12
13  Resolve in Merge Editor
14  <p>Si tienes problemas para iniciar sesión contacta con el administrador</p>

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git merge des2Conflicto
Auto-fusionando login.html
CONFLICTO (contenido): Conflicto de fusión en login.html
Fusión automática falló; arregle los conflictos y luego realice un commit con el resultado.
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$

```

Ya lo tenemos, en el momento se produce un conflicto, se abre el editor por defecto de GIT, en mi caso y espero que en el vuestro es VSCode.

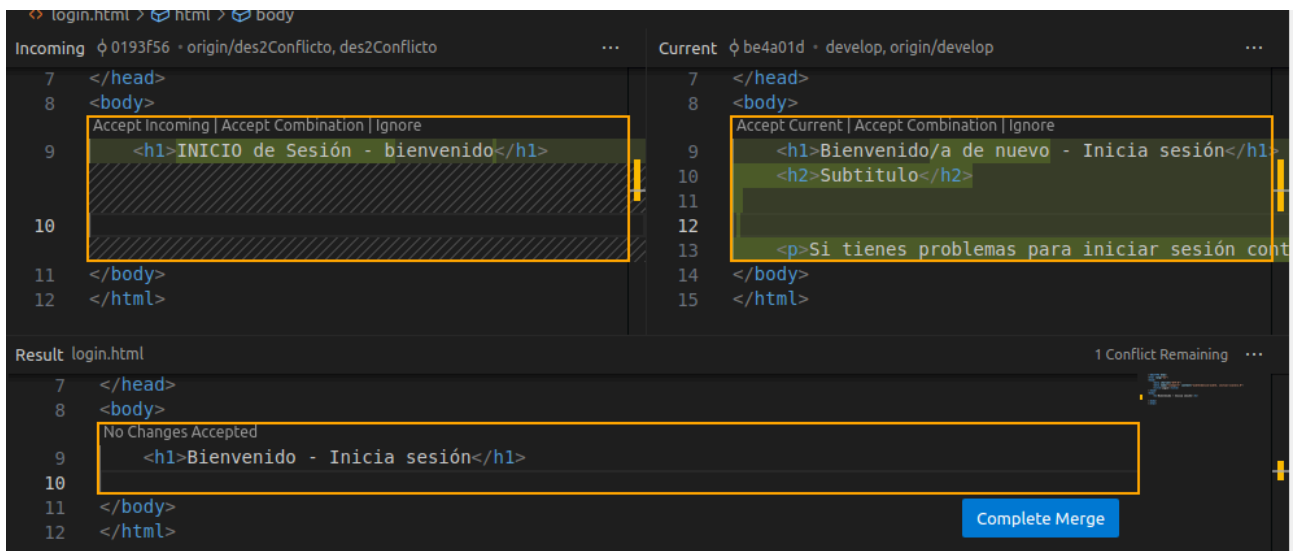
```

<> login.html ! x
<> login.html > html > body > ?
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Login</title>
7  </head>
8  <body>
  Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
9  <<<<<< HEAD (Current Change)
10  <h1>Bienvenido/a de nuevo - Inicia sesión</h1>
11  <h2>Subtitulo</h2>
12
13
14  <p>Si tienes problemas para iniciar sesión contacta con el administrador</p>
15  =====
16  <h1>INICIO de Sesión - bienvenido</h1>
17
18  >>>>>> des2Conflicto (Incoming Change)
19  </body>
20  </html>
  Resolve in Merge Editor

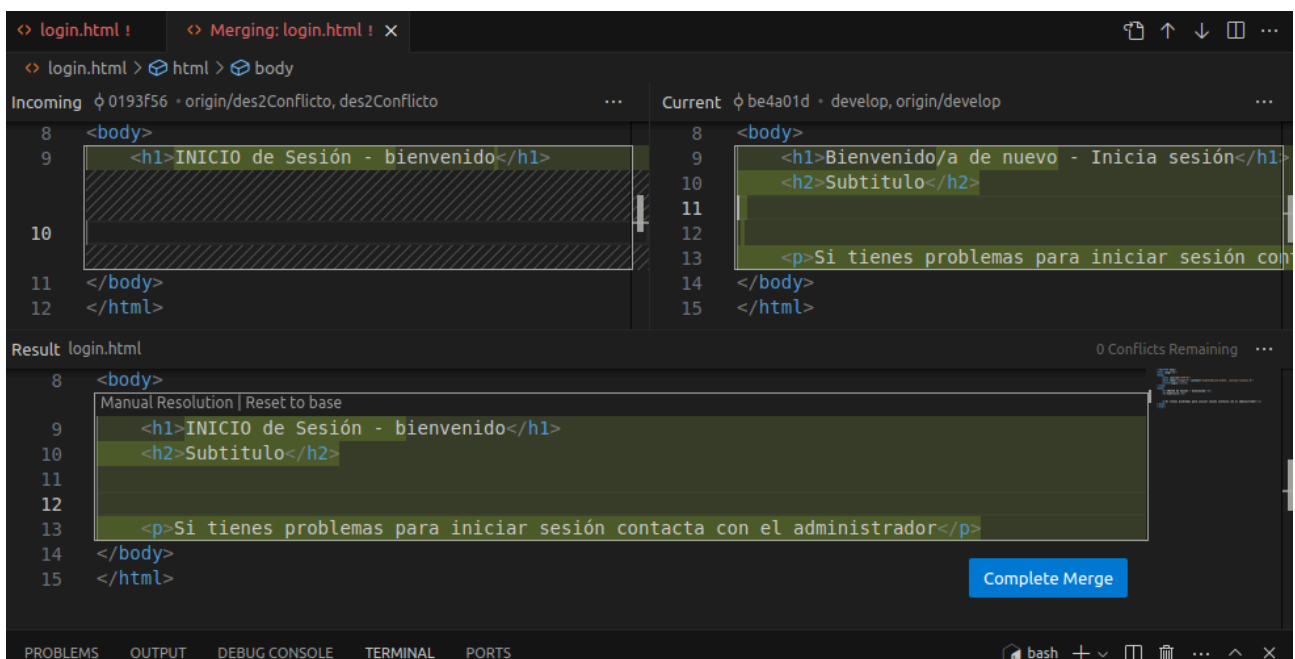
```

Vamos a explicar esta ventana:

- En verde, aparecen los cambios que actualmente tenía la rama develop y en azul los cambios que vienen de la rama des2Conflicto.
- Aquí ambos desarrolladores deberían ponerse en contacto para solucionar el conflicto y ver cómo dejar el fichero final. Podríamos modificar directamente este archivo dejándolo como nos interesa.
- VSCode da la opción de una herramienta para resolver el conflicto, vamos a utilizarla, haz click en "Resolve in Merge Editor".



Aquí se puede observar ambos cambios, arriba a la izquierda los cambios que vienen de la rama des2Conflicto, a la derecha los cambios que están en develop y bajo la salida que queremos obtener, simplemente copia y pega en el fichero Result cómo quieres que quede definitivamente:



Lo dejamos así, ya que los desarrolladores han decidido que así está bien. Pulsamos en “Complete Merge”.

Ya podremos visualizar el archivo correctamente y ahora sí, deberíamos ir al paso de Push de Develop, pero, cuando hay conflicto, como editamos el archivo para solucionarlo, el archivo se queda como modificado, por lo tanto, hay que confirmar el cambio realizado, por lo tanto, si hay conflicto hay que hacer:

- Add + commit indicando la solución del conflicto
- Push de develop

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git add .
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git commit -m "Solución conflicto merge des2Conflicto con develop"
[develop 409cb67] Solución conflicto merge des2Conflicto con develop
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$ git push origin develop
Enumerando objetos: 7, listo.
Contando objetos: 100% (7/7), listo.
Compresión delta usando hasta 4 hilos
Comprimiendo objetos: 100% (3/3), listo.
Escribiendo objetos: 100% (3/3), 391 bytes | 391.00 KiB/s, listo.
Total 3 (delta 2), reusados 0 (delta 0), pack-reusados 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/miguelmf1510/apWebTutorial.git
   be4a01d..409cb67  develop -> develop
miguel@miguel-VirtualBox:~/pruebasGit/apWebTutorial$
```

Y ya hemos terminado, ya has podido ver otro ciclo de desarrollo junto con un conflicto.

Faltaría llevar los cambios a master con un pull request si queremos publicar estos cambios realizados.

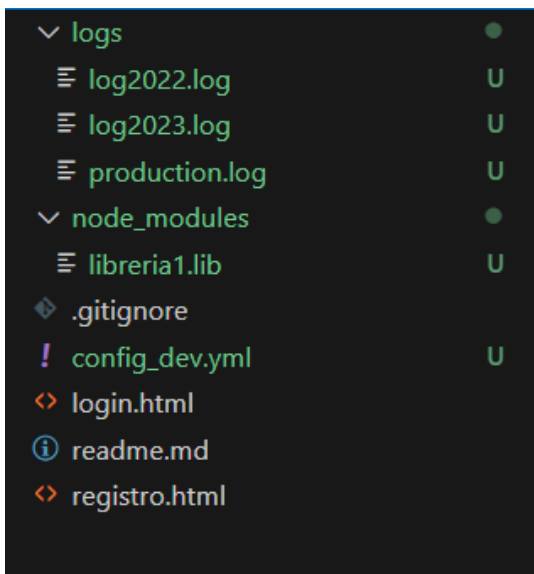
7. GITIGNORE

Vamos a hacer alguna prueba muy básica con .gitignore.

Como se ha comentado en el documento de teoría, el 90% de las veces por no decir el 100% vamos a trabajar con frameworks, estos frameworks ya te van a dar una archivo .gitignore plantilla, y si no, GitHub te ofrece plantillas de este tipo de archivo dependiendo del proyecto, por lo tanto, no es habitual estar modificando el archivo .gitignore. Aún así, siempre es bueno conocer el funcionamiento.

Primero, estando en el desarrollador 1, como ya hemos terminado el tutorial, da igual con el desarrollador que trabajemos, y también da igual si trabajamos directamente en develop, simplemente vamos a hacer alguna prueba.

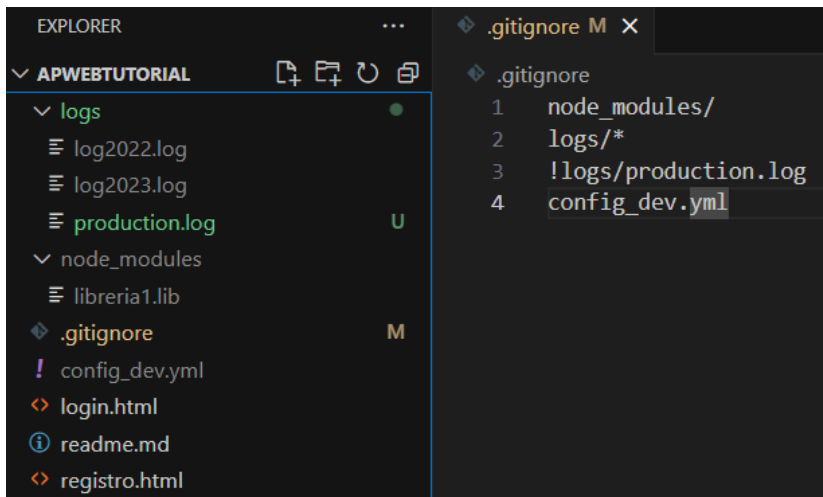
Para ello, crea los siguientes directorios y archivos (los que están en verde):



Como puedes observar, al crearlos, lógicamente se están detectando como nuevos cambios y te los marca en verde para añadirlos y poder hacer un commit de ellos.

Pero no queremos eso, lo que queremos es que esos elementos NO se guarden en el repositorio y mucho menos que se puedan subir al repositorio remoto. Por lo tanto, tenemos que ignorarlos para hacerle ver a GIT que no existen.

Vamos a ello, modificamos el .gitignore y añadimos las siguientes reglas:



1. Con la primera línea, ignoramos toda la carpeta `node_modules` y su contenido.

2. Con la siguiente, ignoramos toda la carpeta y todo su contenido excepto si posteriormente hay alguna excepción.

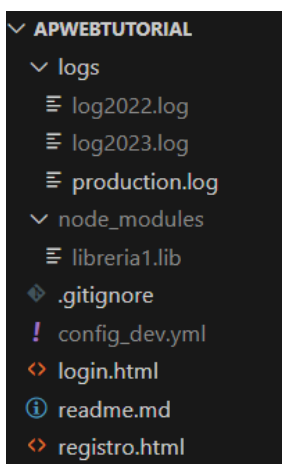
3. Indicamos que no queremos que ignore concretamente ese archivo ya que en la regla anterior lo estamos indicando. Es muy importante el asterisco (*), es la diferencia entre la línea 1 y 2, si quisiéramos poner alguna excepción en `node_modules`, no podríamos, ya que sin asterisco es más restrictivo.

4. Indicamos concretamente el archivo a ignorar.

Ahora yo podríamos confirmar los cambios, hacer un commit y subirlos:

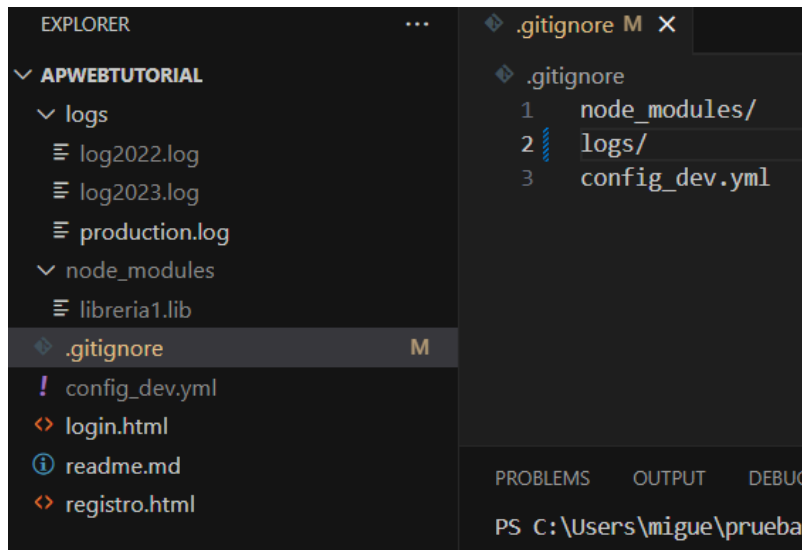
```
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git add .
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git commit -m "Cambios en .gitignore"
```

```
PS C:\Users\miguel\pruebasGIT\apWebTutorial> git push origin develop
Enumerating objects: 10, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 16 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 762 bytes | 381.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/miguelmf1510/apWebTutorial.git
409cb67..3a3e670 develop -> develop
```



Como vemos ya estaría, y los archivos ignorados aparecen con un color más oscuro para diferenciarlos.

Ahora supongamos que queremos ignorar el archivo `production.log`, pero ya lo hemos añadido al repositorio local y remoto ¿Qué podemos hacer?

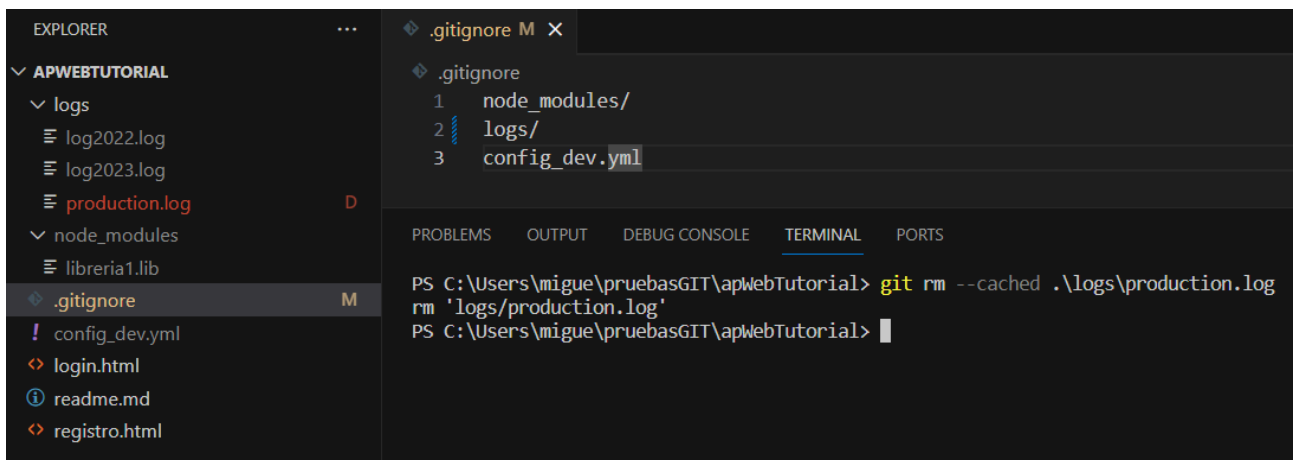


Primero cambiar las reglas del gitignore, en este caso, estoy diciendo que sí quiero que ignore todo lo que haya en logs/, pero como production.log ya se había añadido, no se puede quitar de esta forma.

Por lo tanto, además de realizar lo anterior, yo recomiendo utilizar los comandos:

- Para directorios: `git rm -r --cached <directorio>`
- Para ficheros: `git rm --cached <fichero>`

Ejecutamos el comando concretamente para ese fichero y como puedes observar, git lo marca como rojo, como si lo estuviera eliminado, realmente es como si lo estuviéramos eliminado para git.



Ya ya estaría, añadimos, confirmamos los cambios y los subimos:

```
git add .
git commit -m "ignorar production.log"

git push origin develop
```

Nota: si véis que el color de production.log sigue igual, no os preocupéis, a veces a VSCode le cuesta cargar los colores, pero si váis a GitHub veréis que ya no aparecerá allí, o si cerráis VSCode y volvéis abrir ya saldrán con el color oscuro.

8. RESET Y REVERT

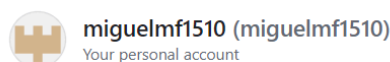
Hay dos formas principales de deshacer cambios en git: uno es usando git reset y el otro es usando git revert. Utilizar git reset no es recomendable ya que podemos perder información.

Para hacer estas pruebas, vamos a utilizar un nuevo repositorio, así estará todo limpio y quedará más claro, además, os voy a enseñar otra forma de crear un repositorio.

El repositorio de los tutoriales anteriores lo creamos primero en local y luego lo enlazamos con el remoto, ahora vamos hacerlo al revés. Vamos a crearlo directamente en GitHub y luego simplemente lo clonamos donde lo queramos utilizar.

Antes de crear el repositorio, vamos a cambiar una configuración, GitHub por defecto utiliza el nombre “main” para la rama principal de los repositorios que se crean, vamos a

cambiarlo ya que nosotros estamos utilizando el nombre “master”, para ello, ve a “Settings” en tu cuenta y al apartado “Repositories”, ahí podrás cambiar el nombre.



- Public profile
- Account
- Appearance
- Accessibility
- Notifications

Repository default branch

Choose the default branch for your new personal repository workflows, or because your integrations still require "master" branch name on individual repositories. [Learn more about](#)

master


Update

create a new repository

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

Required fields are marked with an asterisk (*).

Owner * Repository name *

 miguelmf1510 / appWebTutorial2

Repository names are short and memorable. Need inspiration? How about [super-octo-palm-tree](#) ?

Description (optional)



Public

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



Private

You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file

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

Choose a .gitignore

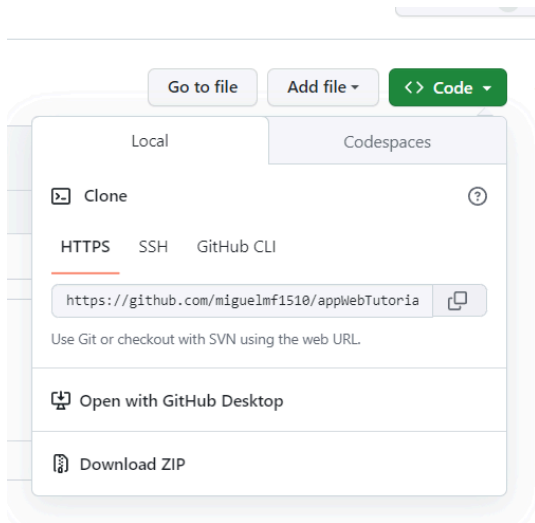
Ignore template: Laravel

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

Ahora sí, crea un nuevo repositorio llamado “apWebTutorial2”:

En esta ocasión, vamos a marcar “Add a README file”, para que genere un README file por defecto, además, vamos a hacer que cree un .gitignore con plantilla “Laravel” para que de paso, veáis un ejemplo.

(Laravel es un framework que utilizaréis en la asignatura de Entorno Servidor).



Quando el repositorio esté creado, vamos a copiar la URL para clonarlo.

En nuestro equipo, accedemos al terminal de VSCode y navegamos hasta la carpeta donde queramos clonar el proyecto y hacemos un git clone:

```
PS C:\Users\miguel\pruebasGIT> git clone https://github.com/miguelmf1510/appWebTutorial2.git
Cloning into 'appWebTutorial2'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

Ya estaría el repositorio preparado (ahora tendríamos que crear la rama develop, subirla, etc.. pero como vamos a hacer unas pruebas rápidas, vamos a trabajar directamente en master). Si hacemos un “git log”, veremos que ya hay un commit, este commit se hace inicialmente cuando creamos el repositorio ya que se han añadido varios ficheros.

Vamos a crear varios archivos y hacer varios commits:

- Crea el archivo1.txt y git add . + git commit -m “Commit archivo1.txt”
- Crea el archivo2.txt y git add . + git commit -m “Commit archivo2.txt”
- Crea el archivo3.txt y git add . + git commit -m “Commit archivo3.txt”
- Crea el archivo4.txt y git add . + git commit -m “Commit archivo4.txt”


```

PS C:\Users\migue\pruebasGIT\appWebTutorial2> git log
commit 9d44043094e94ee35df3b9e2b4a0c356dcc9d082 (HEAD -> master)
Author: Miguel Mira Flor <miguelf92@gmail.com>
Date: Thu Sep 21 09:01:54 2023 +0200

    Commit archivo4.txt

commit 85ce862b70b2c131f94ba371fcec2801c62675c7
Author: Miguel Mira Flor <miguelf92@gmail.com>
Date: Thu Sep 21 09:01:45 2023 +0200

    Commit archivo3.txt

commit 2f4481d39c27527fd83026ce39d758fb38dfa180
Author: Miguel Mira Flor <miguelf92@gmail.com>
Date: Thu Sep 21 09:01:35 2023 +0200

    Commit archivo2.txt

commit 3538047390c9c2739159dc7f9c0604d803455d07
Author: Miguel Mira Flor <miguelf92@gmail.com>
Date: Thu Sep 21 09:01:25 2023 +0200

    Commit archivo1.txt

commit c7d01e7f65e75d662751326cb0515fdc0c76ac88 (origin/master, origin/HEAD)
Author: miguelmf1510 <79716245+miguelf1510@users.noreply.github.com>
Date: Wed Sep 20 19:55:28 2023 +0200

    Initial commit

```

Deberás tener un log como este, vamos a hacer alguna prueba:

Vamos primero con **git reset**, con este comando podemos eliminar todos los commits hasta el que indiquemos, eso hará que se borren todos los cambios realizados, por ejemplo, vamos a suponer que no queríamos hacer los cambios del archivo3 y archivo4, y que nos da igual perder los cambios, por lo tanto, ejecutamos reset hasta el commit del archivo2, eso hará que se borren todos los commits hasta el commit del archivo2 (este no se incluye):

```

PS C:\Users\migue\pruebasGIT\appWebTutorial2> git reset --hard 2f4481d39c27527fd83026ce39d758fb38dfa180
HEAD is now at 2f4481d Commit archivo2.txt

```

Veremos cómo desaparecen los archivos 3 y 4, ya que hemos eliminado sus commits.

```

PS C:\Users\migue\pruebasGIT\appWebTutorial2> git log
commit 2f4481d39c27527fd83026ce39d758fb38dfa180 (HEAD -> master)
Author: Miguel Mira Flor <miguelf92@gmail.com>
Date: Thu Sep 21 09:01:35 2023 +0200

    Commit archivo2.txt

commit 3538047390c9c2739159dc7f9c0604d803455d07
Author: Miguel Mira Flor <miguelf92@gmail.com>
Date: Thu Sep 21 09:01:25 2023 +0200

    Commit archivo1.txt

commit c7d01e7f65e75d662751326cb0515fdc0c76ac88 (origin/master, origin/HEAD)
Author: miguelmf1510 <79716245+miguelf1510@users.noreply.github.com>
Date: Wed Sep 20 19:55:28 2023 +0200

    Initial commit

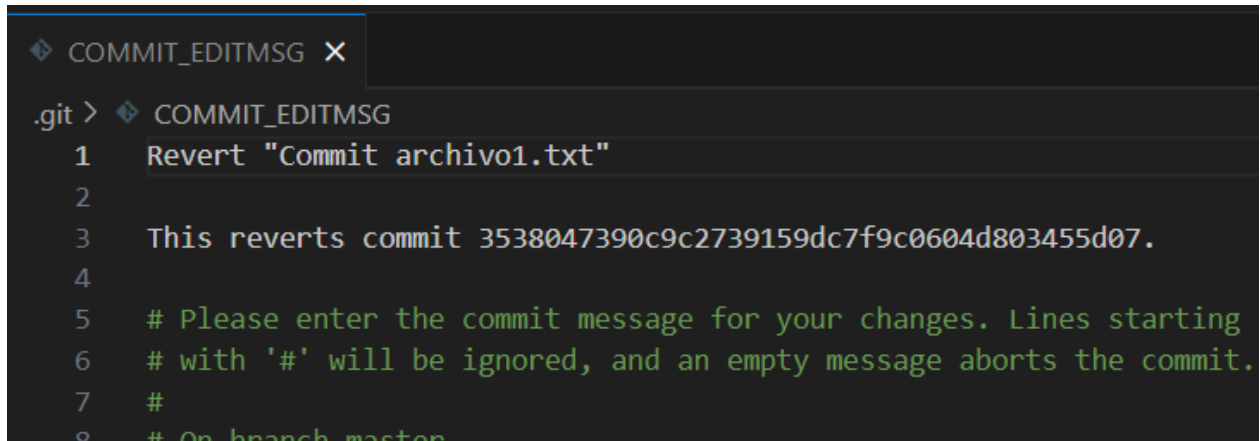
```

Con **git revert**, únicamente “eliminamos” los cambios del commit especificado, es decir, imaginémonos que no queríamos hacer el archivo1 pero el archivo2 sí está bien, pues

puedo deshacer ese commit, internamente el commit seguirá estando por si en un futuro necesitara acceder a él, no es como el reset que lo elimina directamente y crea un nuevo commit con estos cambios, por ello tenemos que añadir un mensaje:

```
PS C:\Users\miguel\pruebasGIT\appWebTutorial2> git revert 3538047390c9c2739159dc7f9c0604d803455d07
hint: Waiting for your editor to close the file... █
```

Esto abrirá el editor por defecto para que modifiquemos el mensaje que quedará almacenado, puedes modificarlo si lo prefieres y una vez cerremos el editor, quedará finalizado el revert:



```

COMMIT_EDITMSG X
.git > COMMIT_EDITMSG
1  Revert "Commit archivo1.txt"
2
3  This reverts commit 3538047390c9c2739159dc7f9c0604d803455d07.
4
5  # Please enter the commit message for your changes. Lines starting
6  # with '#' will be ignored, and an empty message aborts the commit.
7  #
8  # On branch master

```

```

PS C:\Users\miguel\pruebasGIT\appWebTutorial2> git revert 3538047390c9c2739159dc7f9c0604d803455d07
[master f156e5c] Revert "Commit archivo1.txt"
 1 file changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 archivo1.txt
PS C:\Users\miguel\pruebasGIT\appWebTutorial2> git log
commit f156e5cbcc22b81a5e9d518f10069a624516ec25 (HEAD -> master)
Author: Miguel Mira Flor <miguelf92@gmail.com>
Date: Thu Sep 21 09:07:46 2023 +0200

    Revert "Commit archivo1.txt"

    This reverts commit 3538047390c9c2739159dc7f9c0604d803455d07.

commit 2f4481d39c27527fd83026ce39d758fb38dfa180
Author: Miguel Mira Flor <miguelf92@gmail.com>
Date: Thu Sep 21 09:01:35 2023 +0200

    Commit archivo2.txt

commit 3538047390c9c2739159dc7f9c0604d803455d07
Author: Miguel Mira Flor <miguelf92@gmail.com>
Date: Thu Sep 21 09:01:25 2023 +0200

    Commit archivo1.txt

commit c7d01e7f65e75d662751326cb0515fdc0c76ac88 (origin/master, origin/HEAD)
Author: miguelmf1510 <79716245+miguelf1510@users.noreply.github.com>
Date: Wed Sep 20 19:55:28 2023 +0200

    Initial commit

```

Como vemos, el archivo1.txt nos habrá desaparecido pero el commit del archivo1.txt sigue estando, lo que se ha hecho es crear un nuevo commit deshaciendo los cambios realizados en el commit del archivo1.txt.

9. BIBLIOGRAFÍA

- [1] <https://github.com/pedroprieto/curso-github/> - Gestión de Git para la tarea docente - curso de Pedro Prieto, profesor del IES Mare Nostrum de Alicante
- [2] <http://logongas.es/doku.php?id=clase:daw:daw:start>
- [3] <https://pedroprieto.github.io/categories/>
- [4] <https://git-scm.com/book/es/v2/Fundamentos-de-Git-Trabajar-con-Remotos>
- [5] Hoja de referencia de GitHub
https://training.github.com/downloads/es_ES/github-git-cheat-sheet.pdf
- [6] https://corriol.github.io/sxe/UD01/1_arquitectura_de_xarxa_tcpip.html
- [7] <https://www.blai.blog/2018/12/tipos-de-red-en-virtualbox.html>
- [8] <https://sites.google.com/site/wikiredespro/sistemas-operativos-ii/crear-red-entre-maquina-virtual-y-maquina-host-en-virtualbox>
- [9] Desplegament d'aplicacions web. Institut Obert de Catalunya
- [10] Libro de Git <https://git-scm.com/book/es/v2/>
- [11] Git CheatSheets <https://training.github.com/>

10. AUTORES (EN ORDEN ALFABÉTICO)

A continuación ofrecemos en orden alfabético el listado de autores que han hecho aportaciones a este documento.

- David Folgado De la Rosa
- Miguel Mira Flor