

Práctica 06

Contenido

1	Ramas uniones y conflictos	2
1.1	Tipos de merge	2
1.2	Merge Fast-forward	3
1.3	Merge unión automática.....	8
1.4	Merges manuales (con conflictos)	12

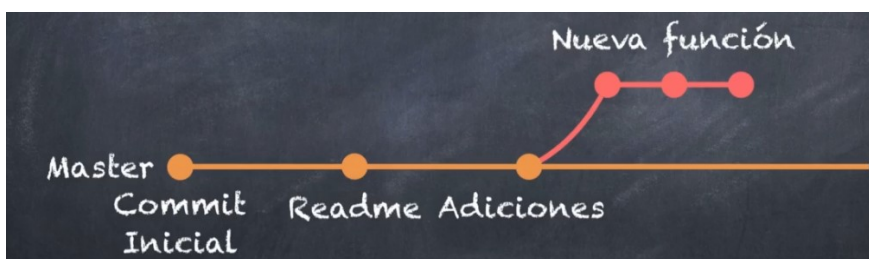
1 Ramas uniones y conflictos

Una rama de un proyecto es una copia en la que empezamos a trabajar y a realizar modificaciones y commits. Desde el punto de vista de git podemos ver cada rama como un camino independiente de commits.

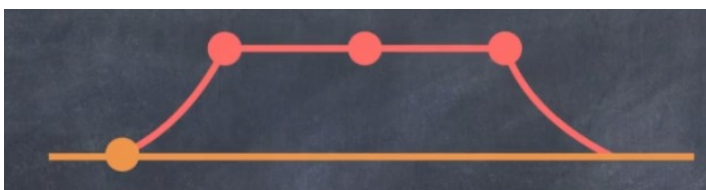
Hasta ahora hemos estado trabajando con sólo una rama que era la rama **master** que en un momento dado vimos como renombrar a **main** por ser más políticamente correcto:



Crear una nueva rama se puede ver visualmente como una bifurcación en el camino actual:



En un momento dado puede que queramos que una rama secundaria se integre en la principal a esto se le llama **unión** o **merge**:



1.1 Tipos de merge

- **Fast-forward**: este tipo de unión se realiza cuando no ha habido cambios en la rama principal y por lo tanto se pueden unir sin problema las dos ramas simplemente agregando los commits de la rama secundaria a la principal:



- **Uniones automáticas:** este tipo de unión se realiza hay cambios en la rama principal pero que no entran en conflicto con los cambios realizados en la rama secundaria, por lo cual git puede realizar la unión de forma automática:

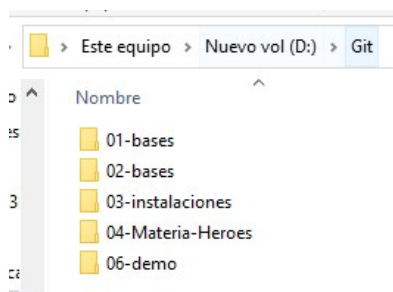


- **Manuales:** en este caso hay modificaciones en la rama principal y en la secundaria que entran en conflicto, por lo que git no puede decidir por sí mismo como hacer la unión y lo que hace es preguntarnos como realizar la unión y a continuación realizar un commit que se denomina **Merge Commit**

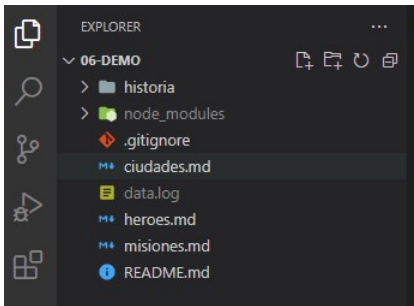


1.2 Merge Fast-forward

Vamos a descomprimir el archivo de recursos 06-demo.zip en la carpeta de git donde estamos subiendo nuestros repositorios. Al final tendremos que tener una nueva carpeta que se llamará 06-demo:



Cerramos la carpeta que tuviéramos abierta en visual studio code y arrastramos esta nueva carpeta 06-demo:

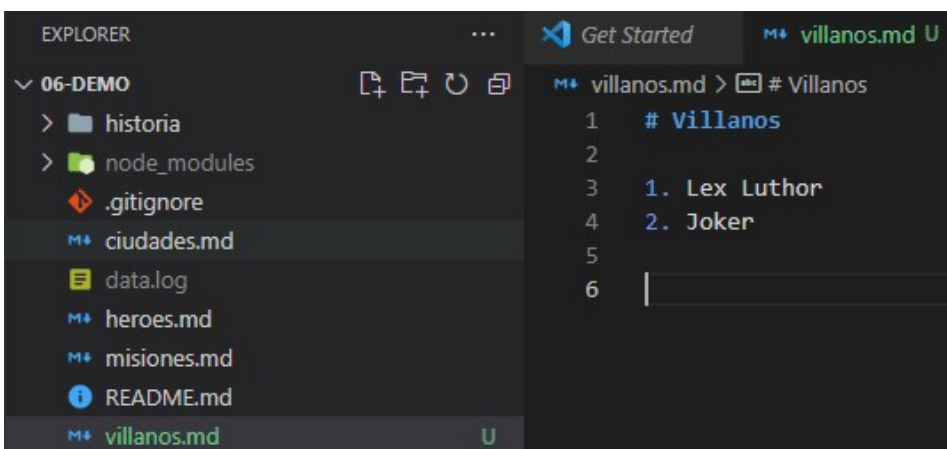


Este repositorio ya tiene la carpeta oculta .git es decir es un repositorio en el que ya se ha inicializado git y además tiene ya una serie de commits realizados y además tiene el archivo .gitignore también creado.

Vamos a ver los commits que tiene el repositorio en este momento:

```
PS D:\Git\06-demo> git lg
* 62c8a10 - (4 years, 7 months ago) Agregando el gitignore - Strider (HEAD -> master)
* ac0d374 - (4 years, 7 months ago) Borramos la historia de batman - Strider
* b4c748c - (4 years, 7 months ago) Cambiamos el nombre de la historia de superman - Strider
* d877f01 - (4 years, 7 months ago) Borrando archivo salvar mundo - Strider
* c9ee153 - (4 years, 7 months ago) Renombrando archivo a salvar-mundo - Strider
* fa3cd3a - (4 years, 7 months ago) Creando el archivo destruir el mundo - Strider
* 4e809d4 - (4 years, 7 months ago) Agregamos a Linterna verde y a Robin - Strider
* 345d7de - (4 years, 7 months ago) Editamos el readme.md - Strider
* 860c6c2 - (4 years, 7 months ago) Agregamos las historias de los heroes - Strider
* bc1a1e5 - (4 years, 7 months ago) Agregamos las ciudades - Strider
* 6b8f60d - (4 years, 7 months ago) Agregamos los heroes - Strider
* da24862 - (4 years, 7 months ago) Agregamos las misiones - Strider
* 88a423d - (4 years, 7 months ago) Se agrego el archivo readme - Strider
```

Supongamos que estamos pensando en añadir al proyecto un apartado de villanos, aunque no estamos seguros de si al final esa parte se quedará en el proyecto o no. De momento vamos a crear un archivo llamados villanos.md con el texto que se ve en la imagen de más abajo:



Vamos a crear una nueva rama que se llame rama-villanos:

Ahora para ver las ramas existentes y en cual estamos ahora mismo ejecutamos el comando:

[Añade aquí una captura del resultado de la ejecución del comando anterior]

El asterisco y el color verde nos indica que estamos en la rama master.

Para cambiar a la nueva rama escribimos:

[Añade aquí una captura del resultado de la ejecución del comando anterior]

Como vemos un mensaje nos indica que nos hemos movido a la rama rama-villanos.

Si hacemos `git lg` vemos lo siguiente:

[Añade aquí una captura del resultado de la ejecución del comando anterior]

Vemos que la cabecera apunta a la rama rama-villanos y que la rama master está en el mismo punto.

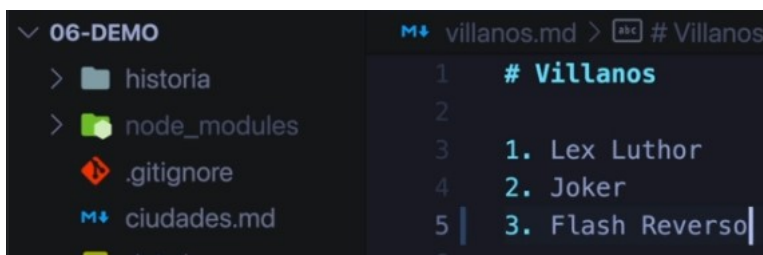
Agregamos el archivo al stage y hacemos commit:

Si hacemos nuevamente `git lg`:

[Añade aquí una captura del resultado de la ejecución del comando anterior]

Vemos que el Head apunta a la rama rama-villanos y que la rama master se ha quedado un commit atrás.

Añadimos un nuevo villano al archivo de villanos:



```
06-DEMO  villanos.md > # Villanos
> historia
> node_modules
.gitignore
ciudades.md
data.log
1 # Villanos
2
3 1. Lex Luthor
4 2. Joker
5 3. Flash Reverso
```

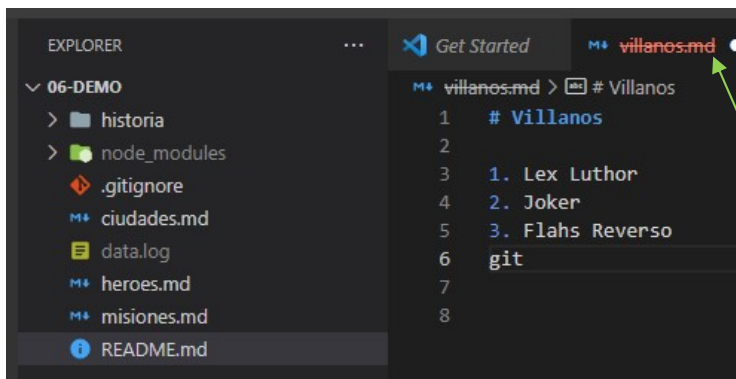
Hacemos un nuevo commit con este cambio usando la forma que sube el archivo al escenario y hace el commit a la vez:

Hacemos nuevamente `git lg`:

[Añade aquí una captura del resultado de la ejecución del comando anterior]

Vemos que la rama master se encuentra ya dos commits por detrás de la rama-villanos.

Si ahora cambiamos a la rama master:



Vemos que el archivo villanos.md desaparece ya que en la rama master no está, y además visual studio code nos indica que la versión que teníamos abierta de villanos.md corresponde a un archivo borrado y por eso nos lo muestra en color rojo y tachado.

Si vuelvo a la rama rama-villanos el archivo volverá a aparecer:

Ahora decidimos que los cambios implementados en esta funcionalidad de villanos son correctos y por lo tanto queremos integrarlos en la rama principal que es la rama master. Al hacer la unión de ramas hay que tener en cuenta que rama tenemos seleccionada, ya que la rama seleccionada recibirá los cambios de la otra rama.

Si hacemos `git branch`:

[Añade aquí una captura del resultado de la ejecución del comando anterior]

Vemos que ahora mismo está seleccionada la rama rama-villanos y de hacer la unión ahora los cambios de la rama master pasarían a la rama-villanos y lo que queremos es agregar los cambios de la rama rama-villanos en la rama master. Por lo tanto cambiamos de rama:

Hacemos la unión o merge:

[Añade aquí una captura del resultado de la ejecución del comando anterior]

Vemos que se ha realizado un merge fast-forward que es la mejor opción porque implica que no ha habido conflictos y git ha podido hacer la unión sin ningún tipo de problemas.

También nos indica que al hacer el merge se incluyó el archivo villanos.md y 7 modificaciones.

Si hacemos una vez más `git lg`:

[Añade aquí una captura del resultado de la ejecución del comando anterior]

Vemos que después del merge las dos ramas están en el mismo punto

Como hemos terminado el desarrollo que estábamos haciendo en la rama rama-villanos y ya hemos unido su funcionalidad con la rama master, no necesitamos ya la rama rama-villanos. Vamos por lo tanto a borrar esta rama:

[Añade aquí una captura del resultado de la ejecución del comando anterior]

Si hubiera cambios en una rama que queremos borrar que no han sido unidos a otra rama, git nos avisará por si estamos borrando la rama por error. En ese caso podríamos forzar el borrado de la rama añadiendo el parámetro `-f`:

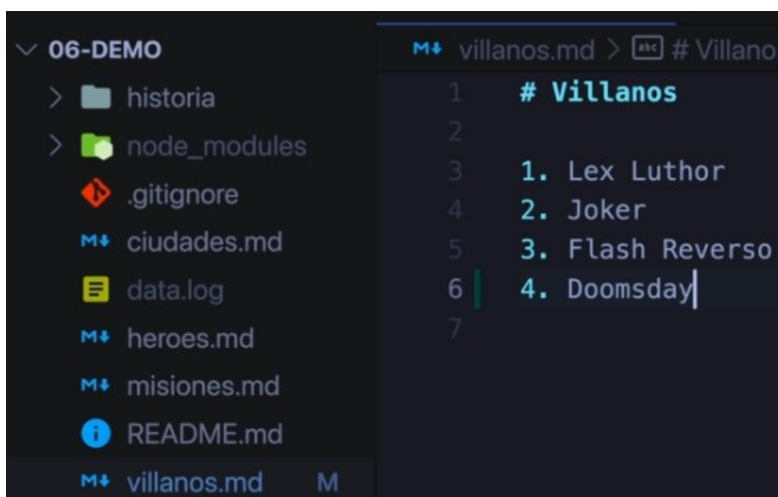
1.3 Merge unión automática

Vamos a trabajar en unas modificaciones sobre el archivo de villanos y para ello vamos a crear una nueva rama. En el apartado anterior vimos como crear una rama y posteriormente movernos a ella usando por tanto dos comandos distintos. Vamos a ver ahora como crear una rama y movernos a la misma con un solo comando:

```
git checkout -b rama-villanos
```

[Añade aquí una captura del resultado de la ejecución del comando anterior]

A continuación modificamos el archivo villanos.md y guardamos los cambios:

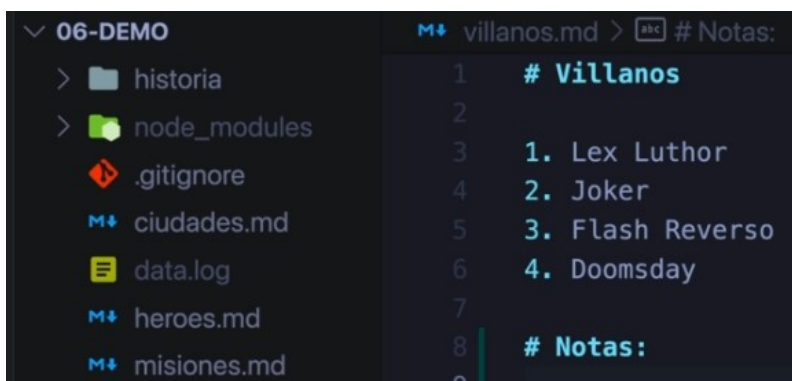


```
06-DEMO
> historia
> node_modules
.gitignore
ciudades.md
data.log
heroes.md
misiones.md
README.md
villanos.md M

villanos.md > # Villanos
1 # Villanos
2
3 1. Lex Luthor
4 2. Joker
5 3. Flash Reverso
6 4. Doomsday
7
```

Hacemos commit de estos cambios:

Hacemos otra modificación en el archivo villanos.md:



```
06-DEMO
> historia
> node_modules
.gitignore
ciudades.md
data.log
heroes.md
misiones.md

villanos.md > # Notas:
1 # Villanos
2
3 1. Lex Luthor
4 2. Joker
5 3. Flash Reverso
6 4. Doomsday
7
8 # Notas:
9
```

Y realizamos un nuevo commit:

Hacemos git lg:

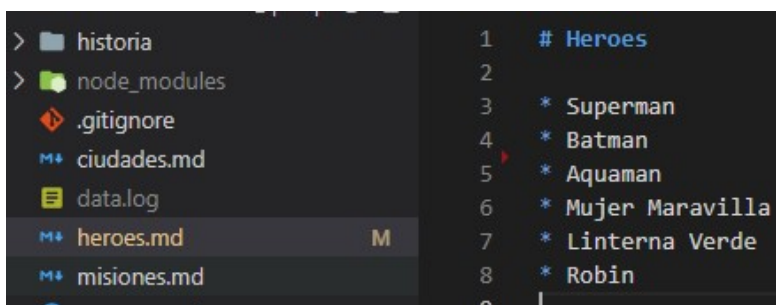
[Añade aquí una captura del resultado de la ejecución del comando anterior]

Vemos que la rama master está dos commits por detrás de rama-villanos.

Ahora nos piden que hagamos un cambio en la rama master ya que hay un error en el archivo de heroes ya que Daredevil es un héroe de Marvel y queremos tener sólo heroes del universo DC en ese archivo.

Nos cambiamos a la rama master:

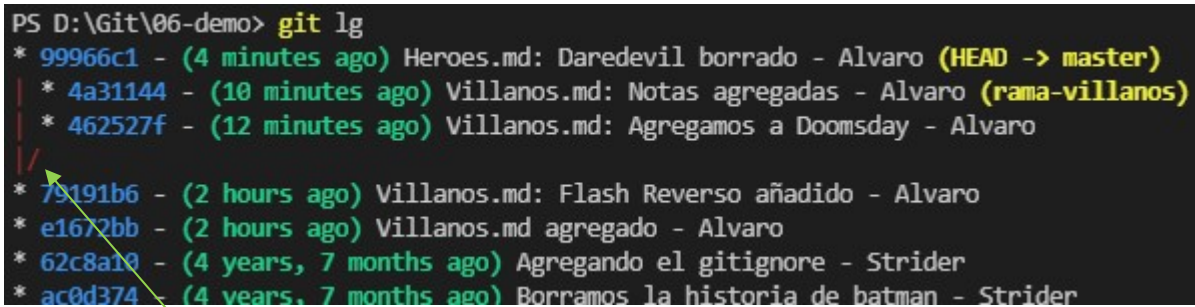
Borramos Daredevil del archivo de heroes y guardamos los cambios:



```
1 # Heroes
2
3 * Superman
4 * Batman
5 * Aquaman
6 * Mujer Maravilla
7 * Linterna Verde
8 * Robin
```

Hacemos commit de este cambio:

Si hacemos `git lg`:



```
PS D:\Git\06-demo> git lg
* 99966c1 - (4 minutes ago) Heroes.md: Daredevil borrado - Alvaro (HEAD -> master)
* 4a31144 - (10 minutes ago) Villanos.md: Notas agregadas - Alvaro (rama-villanos)
* 462527f - (12 minutes ago) Villanos.md: Agregamos a Doomsday - Alvaro
* 79191b6 - (2 hours ago) Villanos.md: Flash Reverso añadido - Alvaro
* e1672bb - (2 hours ago) Villanos.md agregado - Alvaro
* 62c8a10 - (4 years, 7 months ago) Agregando el gitignore - Strider
* ac0d374 - (4 years, 7 months ago) Borrarnos la historia de batman - Strider
```

Vemos que la cabecera está en la rama master y que la rama rama-villanos está un commit por detrás. Además visualmente vemos que hay dos ramas.

Vamos a realizar la unión de estas dos ramas:

Queremos unir los cambios de rama-villanos a la rama master por lo que comprobamos si estamos en ella con `git branch`:

[Añade aquí una captura del resultado de la ejecución del comando anterior]

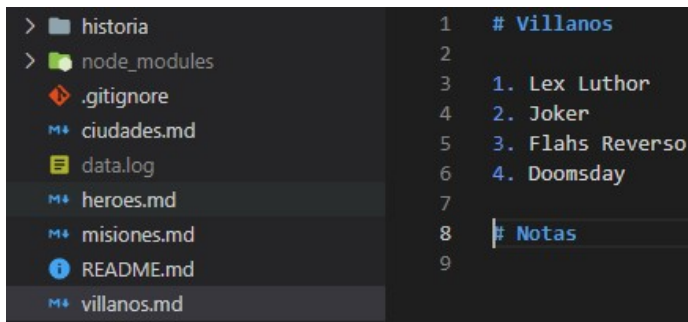
En esta ocasión ya nos encontrábamos en la rama master ya que estuvimos trabando en ella modificando el archivo de heroes.

Hacemos el merge:

[Añade aquí una captura del resultado de la ejecución del comando anterior]

Vemos que git indica que ha realizado la unión usando la estrategia 'recursive' que es a la que estamos llamando en este apartado unión automática, en la cual ha habido cambios en la rama master pero no hay conflictos y git ha podido hacer la unión sin preguntarnos nada.

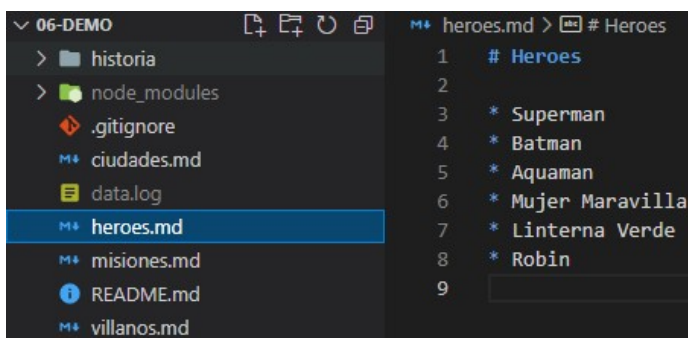
Después de hacer la unión podemos ver que tenemos el archivo de villanos con los 4 villanos y el apartado de notas:



```
> historia
> node_modules
.gitignore
ciudades.md
data.log
heroes.md
misiones.md
README.md
villanos.md
```

```
1 # Villanos
2
3 1. Lex Luthor
4 2. Joker
5 3. Flahs Reverse
6 4. Doomsday
7
8 # Notas
9
```

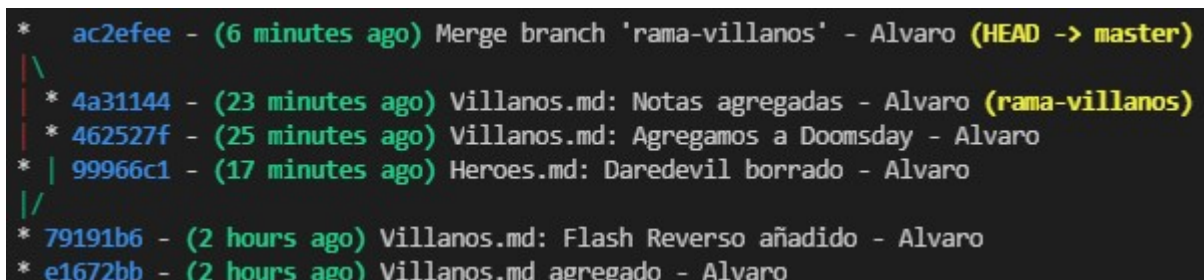
Y que en el archivo héroes no tenemos a Daredevil:



```
06-DEMO
> historia
> node_modules
.gitignore
ciudades.md
data.log
heroes.md
misiones.md
README.md
villanos.md
```

```
heroes.md > # Heroes
1 # Heroes
2
3 * Superman
4 * Batman
5 * Aquaman
6 * Mujer Maravilla
7 *Linterna Verde
8 * Robin
9
```

Si hacemos una vez más `git lg`:



```
* ac2efee - (6 minutes ago) Merge branch 'rama-villanos' - Alvaro (HEAD -> master)
/
* 4a31144 - (23 minutes ago) Villanos.md: Notas agregadas - Alvaro (rama-villanos)
* 462527f - (25 minutes ago) Villanos.md: Agregamos a Doomsday - Alvaro
* | 99966c1 - (17 minutes ago) Heroes.md: Daredevil borrado - Alvaro
|/
* 79191b6 - (2 hours ago) Villanos.md: Flash Reverse añadido - Alvaro
* e1672bb - (2 hours ago) Villanos.md agregado - Alvaro
```

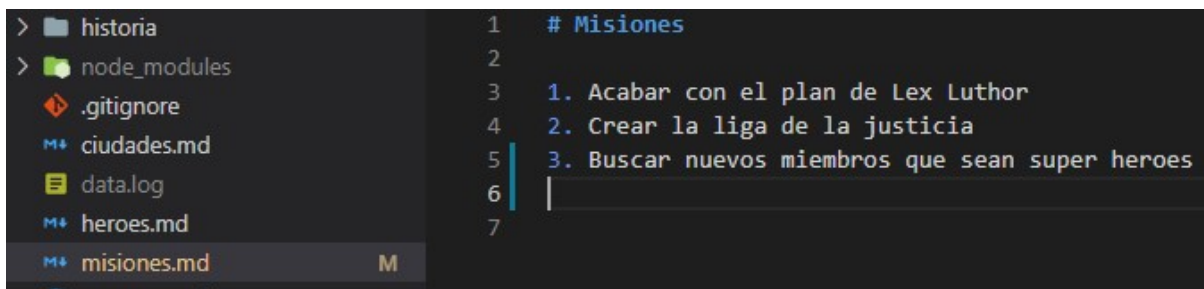
Vemos los dos caminos de commits y como se unieron en el último commit realizado al llevar a cabo la unión de las dos ramas.

1.4 Merges manuales (con conflictos)

Vamos a crear una nueva rama que se va a llamar rama-conflicto:

[Añade aquí una captura del resultado de la ejecución del comando anterior]

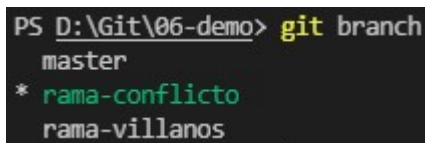
Vamos a modificar el archivo misiones.md y guardar los cambios:



```
> historia
> node_modules
.gitignore
ciudades.md
data.log
heroes.md
misiones.md M

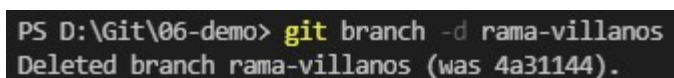
1 # Misiones
2
3 1. Acabar con el plan de Lex Luthor
4 2. Crear la liga de la justicia
5 3. Buscar nuevos miembros que sean super heroes
6
7
```

Comprobamos que estamos en la rama rama-conflicto:



```
PS D:\Git\06-demo> git branch
master
* rama-conflicto
rama-villanos
```

Si nos aparece la rama-villanos del apartado anterior como en la imagen la borramos ya que para este apartado no la necesitamos:



```
PS D:\Git\06-demo> git branch -d rama-villanos
Deleted branch rama-villanos (was 4a31144).
```

Hacemos commit de los cambios realizados en el archivo misiones.md:

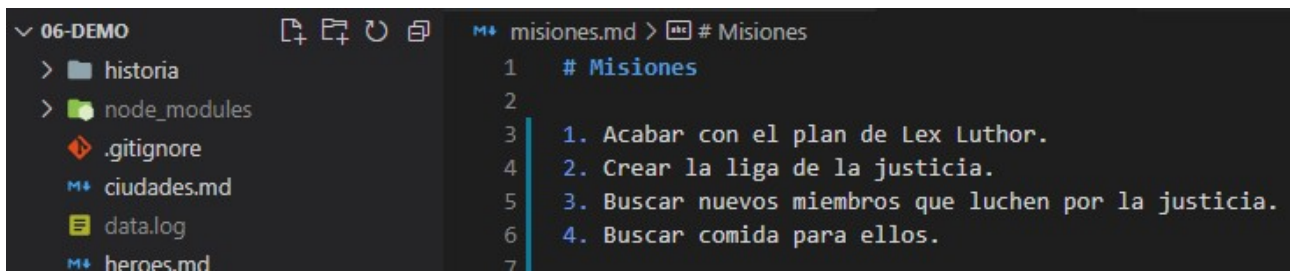
Miramos el estado actual:

[Añade aquí una captura del resultado de la ejecución del comando anterior]

Vamos a regresar a la rama master y realizar cambios en el mismo archivo villanos.md:



```
PS D:\Git\06-demo> git checkout master
Switched to branch 'master'
```



```
06-DEMO
├── historia
├── node_modules
├── .gitignore
├── ciudades.md
├── data.log
├── heroes.md
└── misiones.md

# Misiones
1 # Misiones
2
3 1. Acabar con el plan de Lex Luthor.
4 2. Crear la liga de la justicia.
5 3. Buscar nuevos miembros que luchen por la justicia.
6 4. Buscar comida para ellos.
7
```

(Nota: se ha cambiado la línea 3 y añadido la línea 4, además **se han añadido puntos en todas las líneas**)

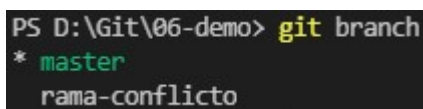
Hacemos commit de estos cambios:

Comprobamos el estado actual:

[Añade aquí una captura del resultado de la ejecución del comando anterior]

Ahora mismo tenemos dos ramas que tienen el mismo archivo modificado, con modificaciones diferentes en cada una de las ramas.

Vamos a realizar la unión o merge. Nos aseguramos que estamos en la rama master ya que queremos traer los cambios de la rama rama-conflicto a la rama master:



```
PS D:\Git\06-demo> git branch
* master
  rama-conflicto
```

Realizamos el merge:

[Añade aquí una captura del resultado de la ejecución del comando anterior]

Git nos muestra un error. Nos dice que ha intentado hacer el merge automático (Auto-merging) pero que no es posible realizarlo porque hay conflictos que hay que resolver de manera manual.

Si abrimos el archivo misiones.md veremos algo así (si no nos aparece algo parecido puede deberse a que lo teníamos abierto con anterioridad, en ese caso debemos cerrarlo y volverlo a abrirlo):

```
1  # Misiones
2
3  Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
4  <<<<<<< HEAD (Current Change)
5  1. Acabar con el plan de Lex Luthor.
6  2. Crear la liga de la justicia.
7  3. Buscar nuevos miembros que luchen por la justicia.
8  4. Buscar comida para ellos.
9  =====
10 1. Acabar con el plan de Lex Luthor
11 2. Crear la liga de la justicia
12 3. Buscar nuevos miembros que sean super heroes
13 >>>>>> rama-conflicto (Incoming Change)
```

Vemos las dos versiones del archivo. En color verde nos indica que es la versión actual ya que es la de la rama en la que estamos. En color azul nos marca la versión de la rama rama-conflicto y nos dice que es la versión entrante (incoming).

La forma de resolver el conflicto de forma manual es comparar visualmente ambas versiones e ir editando el archivo hasta dejar la versión que queramos. Por ejemplo podemos dejar una mezcla de las dos versiones con las 4 líneas pero quitando los puntos:

```
> historia
> node_modules
> .gitignore
> ciudades.md
> data.log
> heroes.md
> misiones.md

1  # Misiones
2
3  1. Acabar con el plan de Lex Luthor
4  2. Crear la liga de la justicia
5  3. Buscar nuevos miembros que luchen por la justicia
6  4. Buscar comida para ellos.
7
8
```

Guardamos los cambios.

Si ahora hacemos `git s`:

[Añade aquí una captura del resultado de la ejecución del comando anterior]

Vemos que git nos marca el conflicto en la unión de las dos versiones del fichero con dos U en color rojo.

Si lo vemos con la versión `git status`:

```
PS D:\Git\06-demo> git status
On branch master
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:   misiones.md

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

Vemos que git nos da más información y nos dice que tenemos un conflicto que tenemos que resolver y hacer luego commit. También nos dice que podemos deshacer el merge usando `git merge --abort`

Como ya hemos arreglado el conflicto de forma manual editando el fichero con las dos versiones sólo nos falta hacer el commit:

[Añade aquí una captura del resultado de la ejecución del comando anterior]

Si hacemos `git s` nuevamente vemos que ya se resolvió el conflicto:

[Añade aquí una captura del resultado de la ejecución del comando anterior]

Y si miramos el estado actual:

```
PS D:\Git\06-demo> git lg
* 037c571 - (4 minutes ago) Unión con rama-conflicto - Alvaro (HEAD -> master)
| \
| * 716bed8 - (37 minutes ago) Misiones.md actualizado - Alvaro (rama-conflicto)
* | b1ea060 - (29 minutes ago) Misiones.md: Actualizado en la rama master - Alvaro
| /
* ac2efee - (3 hours ago) Merge branch 'rama-villanos' - Alvaro
| \
| * 4a31144 - (4 hours ago) Villanos.md: Notas agregadas - Alvaro
* | 462527f - (4 hours ago) Villanos.md: Agregamos a Doomsday - Alvaro
* | 99966c1 - (3 hours ago) Heroes.md: Daredevil borrado - Alvaro
```


Vemos que ya tenemos el commit en el que hemos unido la rama master con la rama rama-conflicto.

Como ya no necesitamos la rama rama-conflicto la borramos:

[Añade aquí una captura del resultado de la ejecución del comando anterior]