git – Gestión de Cambios

1.COMBINANDO BRANCHES CON “MERGE”

1.1. Vete a la rama principal del repositorio y comprueba el contenido del fichero readme.md

```
alumno@sriserver:~$ git checkout master
Already on 'master'
alumno@sriserver:~$ git branch
* master
alumno@sriserver:~$ nano readme.md
```

```
  GNU nano 6.2
Estoy modificando el título
```

1.2 Fusiona la rama feature1

Ahora vamos a recuperar el contenido de la rama “feature1” e incluirlo (fundirlo, empujarlo) en la rama “master”.

```
alumno@sriserver:~$ git checkout master
Switched to branch 'master'
alumno@sriserver:~$ git branch
  feature1
  feature2
* master
alumno@sriserver:~$ git merge feature1
Updating 61be3f2..bc9a80b
Fast-forward
 readme.md | 1 +
 1 file changed, 1 insertion(+)
```

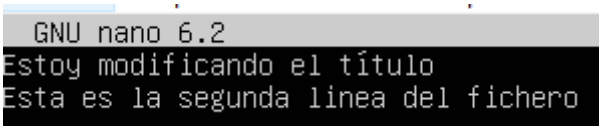
La primera línea indica que el estado del repositorio se ha actualizado desde el commit "8e16db9 hasta el commit "9603786"

La línea "Fast-forward" indica que se realizó un "fast-forward merge" en lugar de un "merge" regular. Esto ocurre cuando no hay conflictos entre los cambios de ambas ramas y Git puede hacer la fusión automáticamente.

La línea "readme.md | 1 +" indica que el archivo "readme.md" se ha modificado en el proceso de fusión y que se ha agregado una línea nueva al archivo.

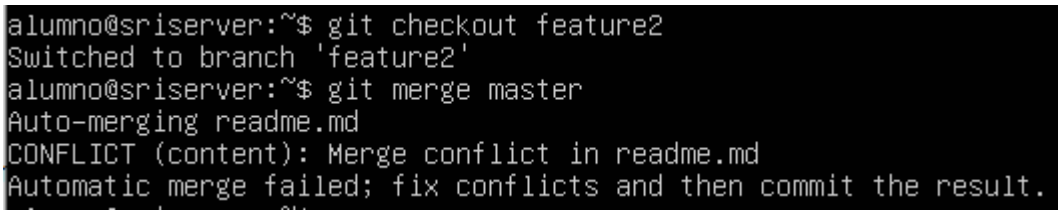
La línea "1 file changed, 1 insertion(+)" indica que se ha modificado un archivo y se ha agregado una línea. Especifica que un archivo fue modificado, cuántos cambios se realizaron y si el número es positivo o negativo. El signo "+" indica que se han agregado líneas y el signo "-" indica que se han eliminado líneas.

Abre el fichero y observa el resultado.

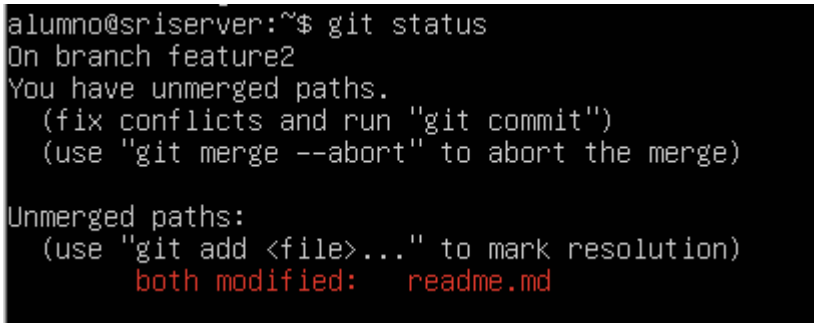


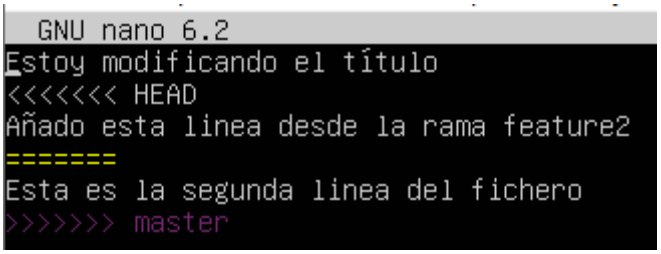
1.3 Fusiona la rama feature2

Ahora veremos qué ocurre cuando dos desarrolladores trabajan a la vez sobre el mismo fichero. Y aquí es donde es importante entender como git gestiona los cambios. Git no interpreta los contenidos; sólo ve líneas que han cambiado, se han añadido o borrado desde el último commit. Y en esta práctica hemos modificado la misma línea del mismo fichero (la segunda). Podríamos simplemente hacer "git merge feature2" pero somos desarrolladores serios (¿verdad?), y un desarrollador serio no coge y sin más mete su trabajo en la rama principal. Lo que hace un desarrollador serio es tomar lo que hay en la rama principal, e incorporarlo en su rama propia (retrofit).



1.4. Muestra el estado del repositorio y del fichero readme.md.





¿Qué observas?

Que git muestra un conflicto que no sabe como resolver y nos pide tomar una acción y realizar un commit.

2.RESOLUCIÓN DE CONFLICTOS

2.1 Modifica el fichero:

```
  GNU nano 6.2
Titulo Nuevo
feature1
feature2
```

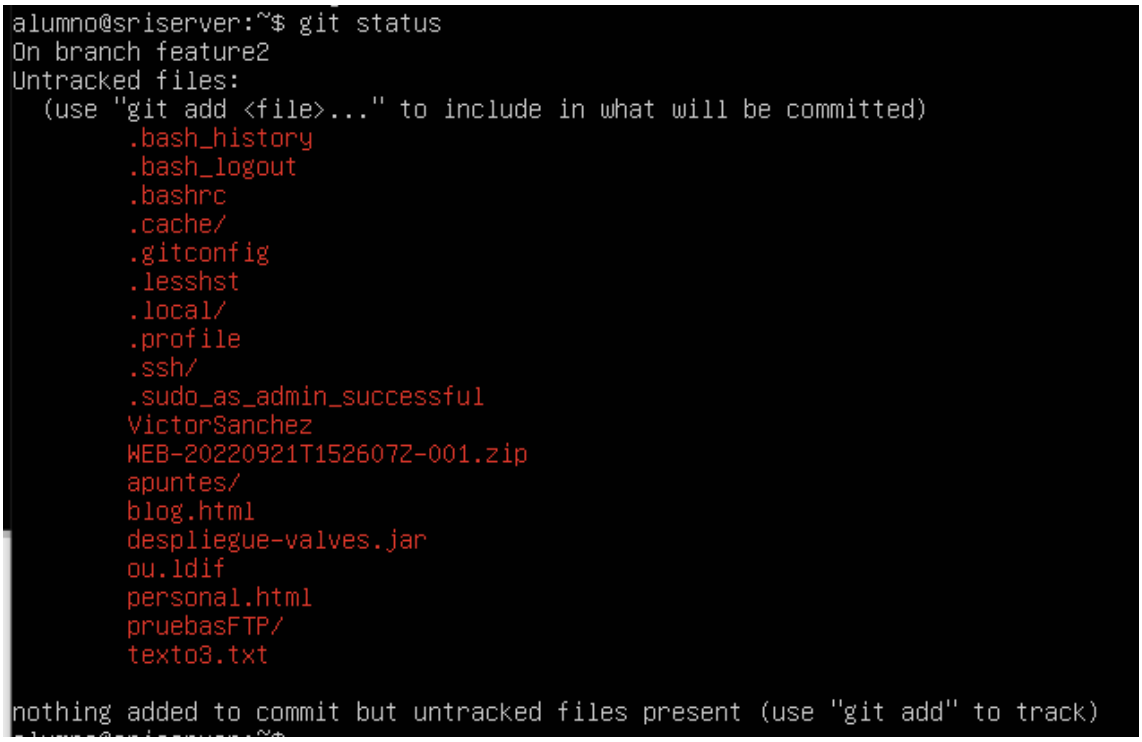
Añade el fichero al INDEX y realiza el commit.

¿qué ha ocurrido al hacer el commit?

Muestra un merge de ‘master’ a ‘feature2’
Nos muestra un mensaje con el siguiente conflicto

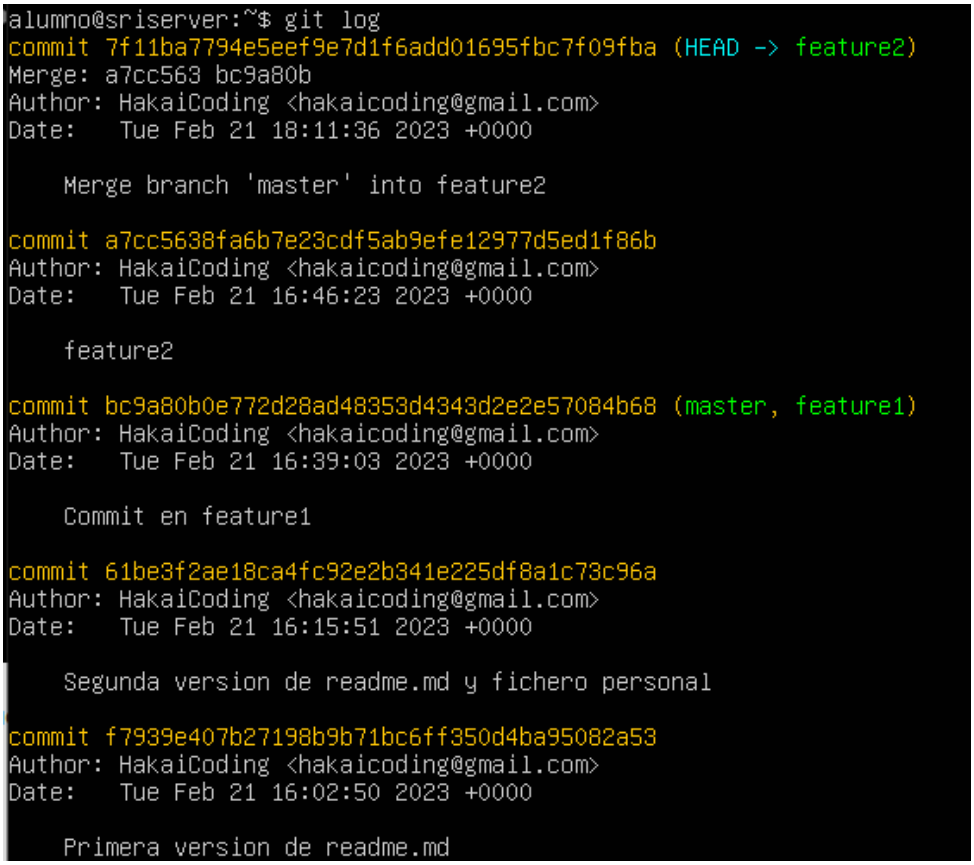
```
Merge branch 'master' into feature2

# Conflicts:
#       readme.md
#
# It looks like you may be committing a merge.
# If this is not correct, please run
#       git update-ref -d MERGE_HEAD
# and try again.


# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch feature2
# All conflicts fixed but you are still merging.
#
# Changes to be committed:
#       modified:   readme.md
#
# Untracked files:
#       .bash_history
#       .bash_logout
#       .bashrc
#       .cache/
#       .gitconfig
#       .lesshst
#       .local/
#       .profile
#       .ssh/
#       .sudo_as_admin_successful
#       VictorSanchez
#       WEB-20220921T152607Z-001.zip
```

2.2 Observa y comenta el estado del repositorio



Estamos situadios en la rama ‘feature2’ y gut status nos dice que no hay ningun archivo pendiente de commit.

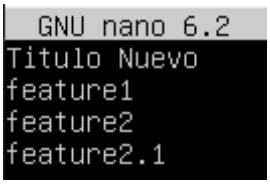
2.3 ¿Qué información te da el log del repositorio?



3.RESOLUCIÓN DE ERRORES

3.1 Avanzamos en feature2

Imaginemos que seguimos trabajando en la feature2. Editemos de nuevo el fichero asegurándonos primero que seguimos en la rama feature2 y añadimos una nueva línea feature2.1



3.2 Cambio de Ramas

Ahora trataremos de ir a la rama principal

```
alumno@sriserver:~$ git checkout master
error: Your local changes to the following files would be overwritten by checkout:
        readme.md
Please commit your changes or stash them before you switch branches.
Aborting
```

¿Qué ha ocurrido? ¿Por qué ha abortado git?
El mensaje indica que hay cambios locales en el repositorio que entran en conflicto con la rama que estás intentando cambiar.

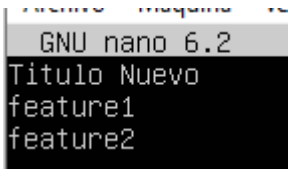
En este caso, se indica que hay cambios sin confirmar en el archivo "readme.md" que se perderían si cambias a la rama "master.

El mensaje sugiere que debes confirmar tus cambios o guardarlos antes de cambiar de rama.

3.3 Aparcamos el Conflicto (stash)

Ahora tenemos dos opciones: borrar los cambios o dejarlos de lado en un reservorio (stash).

```
alumno@sriserver:~$ git stash
Saved working directory and index state WIP on feature2: 7f11ba7 Merge branch 'master' into feature2
```



3.4 Vuelta la Rama Principal

Ahora volvamos a la rama principal (main), que de momento sólo contiene los cambios de la rama “feature1”

```
alumno@sriserver:~$ git checkout master
Switched to branch 'master'
```

```
  GNU nano 6.2
Estoy modificando el título
Esta es la segunda linea del fichero
```

```
alumno@sriserver:~$ git merge feature2
Updating bc9a80b..7f11ba7
Fast-forward
 readme.md | 5 +++--
 1 file changed, 3 insertions(+), 2 deletions(-)
```

Abrimos el fichero readme.md y comprobamos que sólo tiene los cambios de la rama “feature1”

```
  GNU nano 6.2
Titulo Nuevo
feature1
feature2
```

3.5 Listado del stash

Para listar los cambios que están aparcados

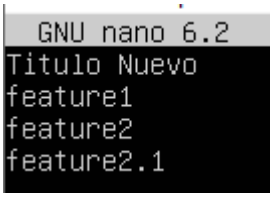
```
alumno@sriserver:~$ git stash list
stash@{0}: WIP on feature2: 7f11ba7 Merge branch 'master' into feature2
```

3.6 Resolución del stash

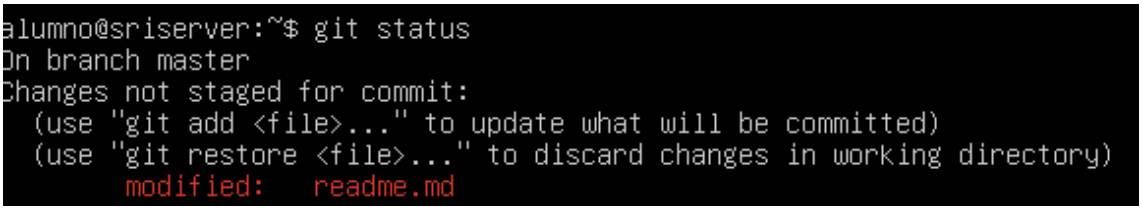
Para aplicar los cambios pendientes en el stash

```
alumno@sriserver:~$ git stash apply
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   readme.md
```

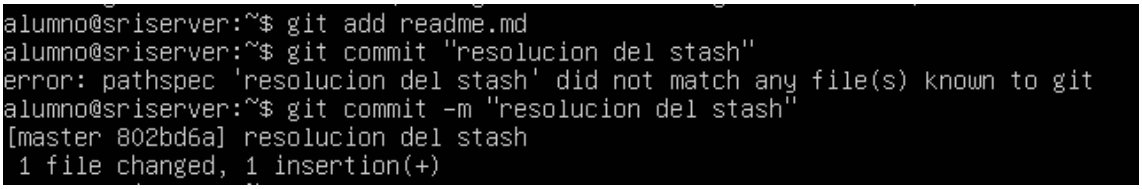
Abrimos el fichero readme.md y comprobamos que se ha añadido la línea que originaba el conflicto:



Sin embargo, si preguntamos a git en qué estado está nos dirá que los cambios no han sido salvados porque estamos en el reservorio (stash).



Para formalizar el cambio necesitamos añadir el fichero al índice y hacer el commit:



3.7 Descarte del stash

Imagina que no quieres quedarte con el cambio de feature2.1. ¿qué tendrías que hacer?

Puedes usar el comando git stash drop para eliminar el stash de forma permanente, lo que significa que todos los cambios que contenía se perderán.

4.GESTIONAR VERSIONES DE FICHEROS

4.1 Descartar los Cambios

```
alumno@sriserver:~$ git checkout readme.md
Updated 0 paths from the index
```

Si abrimos el fichero veremos que el cambio que estaba aparcado (stash) ha sido eliminado y el fichero está en el último estado estable de la rama del repositorio:

```
  GNU nano 6.2
Titulo Nuevo
feature1
feature2
```

4.2 Recuperar la Versión de otra Rama

También podemos recuperar la version estable del fichero pero en otra rama

```
alumno@sriserver:~$ git checkout feature1 -- ./readme.md
```

```
  GNU nano 6.2
Estoy modificando el título
Esta es la segunda linea del fichero
```

4.3. Sacar un fichero del índice

Para sacar un fichero del índice hay que ejecutar el comando

```
alumno@sriserver:~$ git restore ./readme.md
alumno@sriserver:~$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .bash_history
        .bash_logout
        .bashrc
        .cache/
        .gitconfig
        .lesshst
        .local/
        .profile
        .ssh/
        .sudo_as_admin_successful
        VictorSanchez
        WEB-20220921T152607Z-001.zip
        apuntes/
        blog.html
        despliegue-valves.jar
        ou.ldif
        personal.html
        pruebasFTP/
        texto3.txt
```

4.4. Recuperar la versión de master

Para descartar los cambios y quedarnos con la versión de la rama principal ejecutamos:

```
alumno@sriserver:~$ git checkout ./readme.md
```

```
  GNU nano 6.2                                          readme.md
Titulo Nuevo
feature1
feature2
feature2.1
```

4.5. Limpiar el Reservorio

En el stash todavía están los cambios salvados en la lista:

```
alumno@sriserver:~$ git stash
```

Tenemos dos opciones:

Si queremos añadir un item desde el stash y quitarlo de la lista, necesitamos ejecutar:
$ git stash pop Y no git stash apply ya que este comando aplica el cambio pero no lo saca del stash, mientras que git stash pop además de aplicar el cambio (si no hay conflicto) y lo saca del stash.

Podemos también sacar un item del stash sin aplicar los cambios, usando el comando git stash drop indicando qué cambio quitar si hubiera varios