

# Control de versiones GIT



# git

Franz Josué Ramírez Villca  
2DAW-A

## Indice

Trabajando con repositorios locales.....	3
Creando nuestra carpeta de trabajo.....	3
Inicializando nuestro proyecto git y haciendo nuestro primer commit.....	3
Trabajando con repositorios remotos.....	8
Trabajando con ramas.....	12
Creando un conflicto al fusionar ramas.....	14
Creando un submodulo y lo obviaremos con .gitignore.....	18

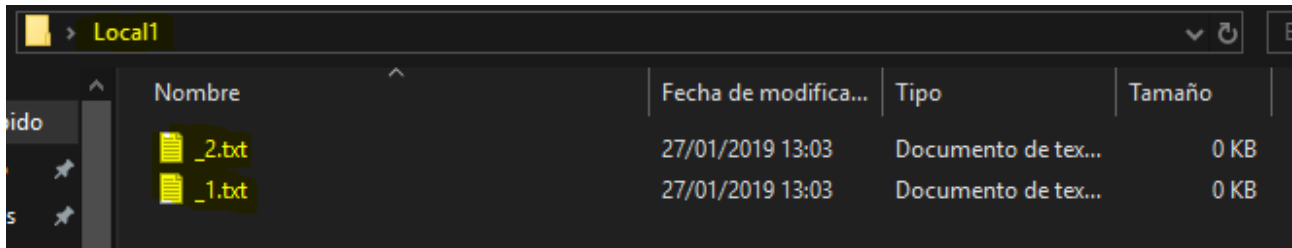
## Trabajando con repositorios locales.

Primeramente iniciaremos con la preparacion de nuestro entorno para poder trabajar con Git.

Obviamente tendremos que tener instalado Git.

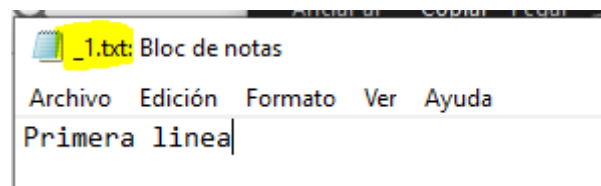
## Creando nuestra carpeta de trabajo.

Creamos nuestra carpeta de la practica, en donde estaremos trabajado con Git.



## Inicializando nuestro proyecto git y haciendo nuestro primer commit.

Aquí añadiremos un pequeño párrafo a nuestro fichero “1” y inicializaremos git en nuestra carpeta y haremos nuestro primer commit.



Aquí ya tenemos inicializado nuestro git y vemos el estado con “git status”, nos aparecen los ficheros aun en el “working directory” que aun no esta “trackeado” por git.

```
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1
$ git init
Initialized empty Git repository in C:/Users/Bomber/Desktop/Local1/.git/

Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git status
On branch master

No commits yet

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

        _1.txt
        _2.txt

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

Ahora haremos un “git add .” para pasar nuestros archivos al “staging area” y volvemos a ver el estado y nos aparece que ya estan siendo “trackeado” por git.

```
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git add .

Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git status
On branch master

No commits yet

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

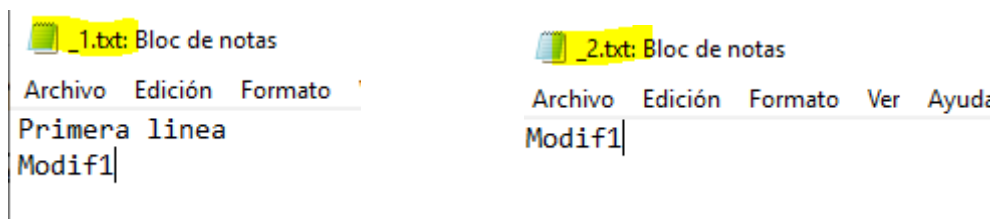
        new file:   _1.txt
        new file:   _2.txt
```

Ahora hacemos el “commit” y lo pasamos al “git repository” donde ya estara listo y “guardado”, vemos que al hacer el git status ya no tenemos ningun fichero sin “trackear”.

```
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git commit -m "Primer commit"
[master (root-commit) 40ee372] Primer commit
2 files changed, 1 insertion(+)
create mode 100644 _1.txt
create mode 100644 _2.txt

Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Ahora añadiremos una nueva linea a nuestros dos ficheros.



Y vemos que el estado ahora nos aparece “modificado”.

```
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   _1.txt
        modified:   _2.txt

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

Añadimos al “Staging area” solo el fichero 1. Y podemos ver que el estado solo nos aparece “trackeado” ese fichero.

```
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git add _1.txt

Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   _1.txt

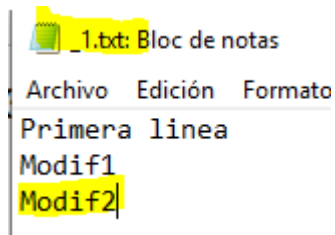
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   _2.txt
```

Ahora veremos la diferencia que hay en el fichero “1” entre el primer commit que y lo que tenemos en el “Staging area”. Se ve claramente que añadimos la nueva línea “Modif1”.

```
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git diff --staged _1.txt
diff --git a/_1.txt b/_1.txt
index fdbe871..98b0b0f 100644
--- a/_1.txt
+++ b/_1.txt
@@ -1,2 @@
-Primera línea
\ No newline at end of file
+Primera línea
+Modif1
\ No newline at end of file
```

Añadimos una nueva línea a nuestro primer fichero “1”.



Ahora viendo es estado, nos aparece nuevamente el fichero “\_1.txt” modificado.

```
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   _1.txt

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

    modified:   _1.txt
    modified:   _2.txt
```

Ahora “eliminaremos” el cambio de nuestro fichero \_1.txt que no esta en el “Staging area” y volveremos al que teníamos.

```
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git checkout -- _1.txt

Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   _1.txt

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

    modified:   _2.txt



Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ cat _1.txt
Primera línea
Modif1
```

## Trabajando con repositorios remotos.

Ahora empezaremos a trabajar con repositorios remotos usando Github, para esto crearemos primeramente nuestro repositorio en este.


### Create a new repository


A repository contains all the files for your project, including the revision history.

Owner:  fjota / Repository name:  


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

Description (optional):

☐  **Public**  
Anyone can see this repository. You choose who can commit.

☒  **Private**  
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore:  | Add a license:  

[Create repository](#)

Ahora vincularemos nuestro proyecto local a este repositorio remoto. Y vemos que esta añadido correctamente.

```
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git remote add origin https://github.com/fjota/Local1.git

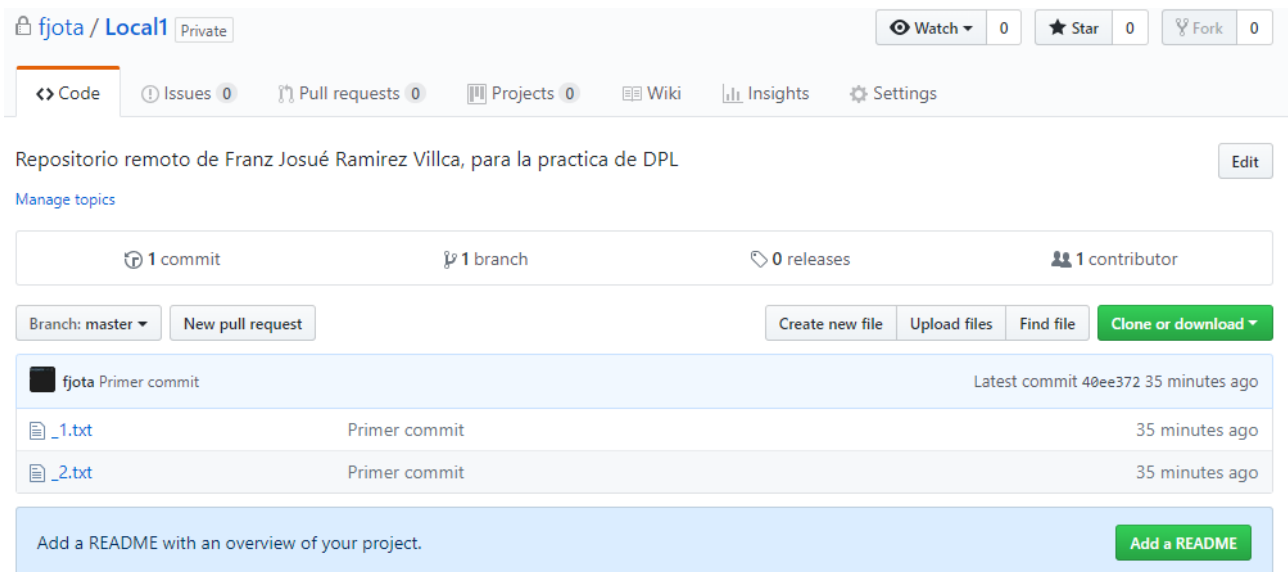
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git remote -v
origin https://github.com/fjota/Local1.git (fetch)
origin https://github.com/fjota/Local1.git (push)
```



Ahora que ya tenemos vinculado nuestro proyecto local con el repositorio remoto, subiremos todos los cambios “commiteados” a este repositorio remoto.

```
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 259 bytes | 259.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/fjota/Local1.git
 * [new branch]      master -> master
```

Y podemos ver en el repositorio que se han subido nuestro ficheros.



The screenshot shows the GitHub interface for a repository named 'fjota / Local1'. The repository is private and has 0 watches, 0 stars, and 0 forks. The main navigation bar includes links for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings. The repository description is 'Repositorio remoto de Franz Josué Ramirez Villca, para la practica de DPL'. It shows 1 commit, 1 branch, 0 releases, and 1 contributor. The 'Branch: master' dropdown is set to 'master', and there is a 'New pull request' button. Below this, there are buttons for 'Create new file', 'Upload files', 'Find file', and a green 'Clone or download' button. The commit history shows a 'Primer commit' by 'fjota' with the latest commit hash '40ee372' made 35 minutes ago. The commit list includes files '\_1.txt' and '\_2.txt', both marked as 'Primer commit' and '35 minutes ago'. At the bottom, there is a blue banner with the text 'Add a README with an overview of your project.' and a green 'Add a README' button.

Crearemos otro repositorio remoto en Github.

## Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

fjota

Repository name \*

Ejemplo1

Great repository names are short and memorable. Need inspiration? How about [musical-umbrella](#).

Description (optional)

Segundo repositorio remoto de Franz josué ramirez Villca, practica de DPL



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None

Add a license: None



Create repository

Vincularemos este nuevo repositorio remoto a nuestro proyecto local, veremos que ahora tenemos el repositorio “Local1” y “Ejemplo1” remotos, añadidos a nuestro proyecto local.

```
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git remote add Ejemplo1 https://github.com/fjota/Ejemplo1.git

Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git remote -v
Ejemplo1      https://github.com/fjota/Ejemplo1.git (fetch)
Ejemplo1      https://github.com/fjota/Ejemplo1.git (push)
origin        https://github.com/fjota/Local1.git (fetch)
origin        https://github.com/fjota/Local1.git (push)
```

Ahora cambiaremos el nombre de “referencia” que tenemos añadido a nuestro repositorio remoto “Ejemplo1” por el de “Ejemplo7”.

```
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git remote rename Ejemplo1 Ejemplo7

Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git remote -v
Ejemplo7      https://github.com/fjota/Ejemplo1.git (fetch)
Ejemplo7      https://github.com/fjota/Ejemplo1.git (push)
origin https://github.com/fjota/Local1.git (fetch)
origin https://github.com/fjota/Local1.git (push)
```

## Trabajando con ramas.

Ahora empezaremos a trabajar con ramas, crearemos una nueva rama llamada “rama1”, nos pasaremos a ella, y realizaremos un “git log” para saber que commits tenemos. Al hacer o crear una nueva rama, nos “copia” el estado de la actual y en la nueva rama dispondremos de todo lo que teníamos hasta el momento.

```
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git branch rama1

Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git checkout rama1
Switched to branch 'rama1'
M       _1.txt
M       _2.txt

Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (rama1)
$ git log --oneline
40ee372 (HEAD -> rama1, origin/master, master) Primer commit
```

Ahora tenemos que hacer un “merge” o fusionaremos con la rama master.

**Nota.-** Como antes de crear la rama aun teniamos cambios en el Stage area y en Working directory, al pasar a la nueva rama tambien se nos pasaros esos cambios con esos estado, de estos mismo en nuestra nueva rama haremos commit.

Haremos un commit de los “cambios que añadimos”, usaremos el commando “git commit -am “mensaje””, esto nos ahorra el paso de hacer el “git add .”.

```
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (rama1)
$ git status
On branch rama1
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   _1.txt

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

        modified:   _2.txt

Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (rama1)
$ git commit -am "Cambio dos en la rama1, para luego fusionar(merge)"
[rama1 906e5ee] Cambio dos en la rama1, para luego fusionar(merge)
 2 files changed, 3 insertions(+), 1 deletion(-)

Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (rama1)
$ git status
On branch rama1
nothing to commit, working tree clean
```

Ahora que ya tenemos un nuevo commit en la “rama1” nos pasaremos a la rama “master” y desde alli haremos un merge.

```
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (rama1)
$ git checkout master
Switched to branch 'master'

Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git merge rama1
Updating 40ee372..906e5ee
Fast-forward
 _1.txt | 3 ++-
 _2.txt | 1 +
 2 files changed, 3 insertions(+), 1 deletion(-)
```

Haciendo un “git log” vemos ahora que tenemos el commit que teníamos en la “rama1”.

```
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git log --oneline
906e5ee (HEAD -> master, rama1) Cambio dos en la rama1, para luego fusionar(merge)
40ee372 (origin/master) Primer commit
```

## Creando un conflicto al fusionar ramas.

Crearemos una nueva rama llamada “rama2”.

```
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git branch rama2

Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (master)
$ git checkout rama2
Switched to branch 'rama2'

Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (rama2)
$ git log --oneline
906e5ee (HEAD -> rama2, rama1, master) Cambio dos en la rama1, para luego fusionar(merge)
40ee372 (origin/master) Primer commit
```

En esta nueva rama (rama2) añadiremos la siguiente línea, para generar conflicto con la rama1, hacemos el commit.

\_1.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

Primera línea

Modif1 - Añadiendo esto en rama2, crearemos un conflicto con rama1

```
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (rama2)
$ git status
On branch rama2
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

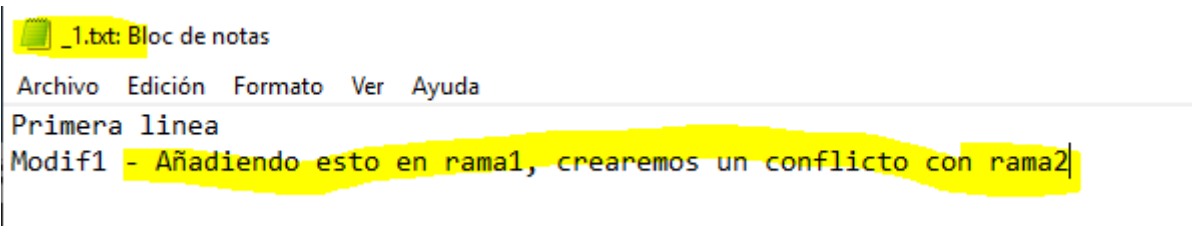
        modified:   _1.txt

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

Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (rama2)
$ git commit -am "Cambio en rama dos, para generar conflicto"
[rama2 9aaf4b3] Cambio en rama dos, para generar conflicto
 1 file changed, 1 insertion(+), 1 deletion(-)

Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (rama2)
$ git status
On branch rama2
nothing to commit, working tree clean
```

Haremos lo mismo en la “rama1” añadiendo un texto distinto.



```
$ git status
On branch rama1
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

       modified:   _1.txt

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

Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (rama1)
$ git commit -am "Creando este cambio en rama1, para generar conflicto"
[rama1 5e52753] Creando este cambio en rama1, para generar conflicto
 1 file changed, 1 insertion(+), 1 deletion(-)

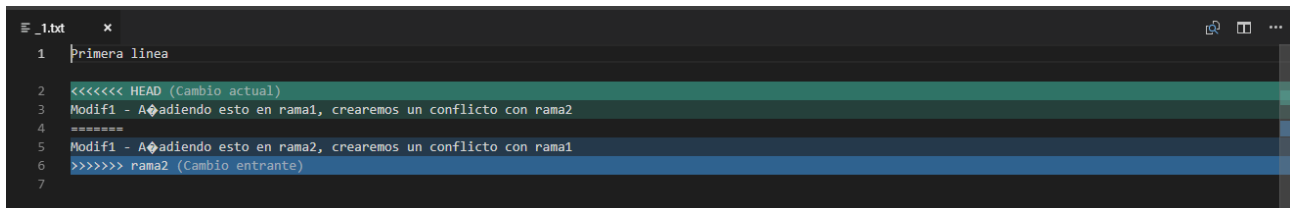
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (rama1)
$ git status
On branch rama1
nothing to commit, working tree clean
```

Ahora hacemos un merge, y vemos que nos salta el conflicto que nosotros provocamos.

```
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (rama1)
$ git merge rama2
Auto-merging _1.txt
CONFLICT (content): Merge conflict in _1.txt
Automatic merge failed; fix conflicts and then commit the result.
```

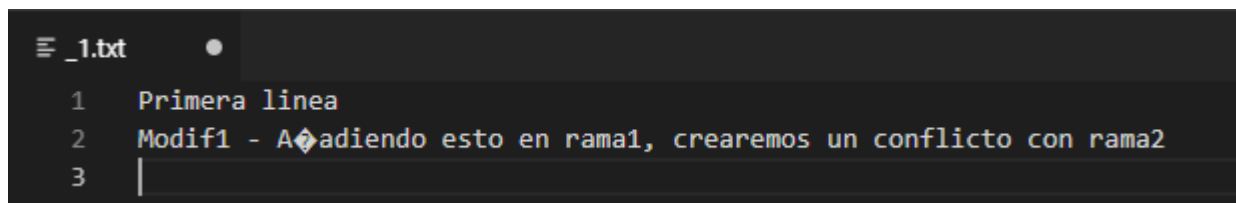


Para resolverlo, nos vamos a un editor de texto, en este caso vsCode y abrimos el fichero que nos genera el conflicto, podemos ver que nos salen las líneas que causan conflictos siendo la primera el cambio que tenemos en nuestra rama actual y la segunda la rama con la que queremos hacer un merge.



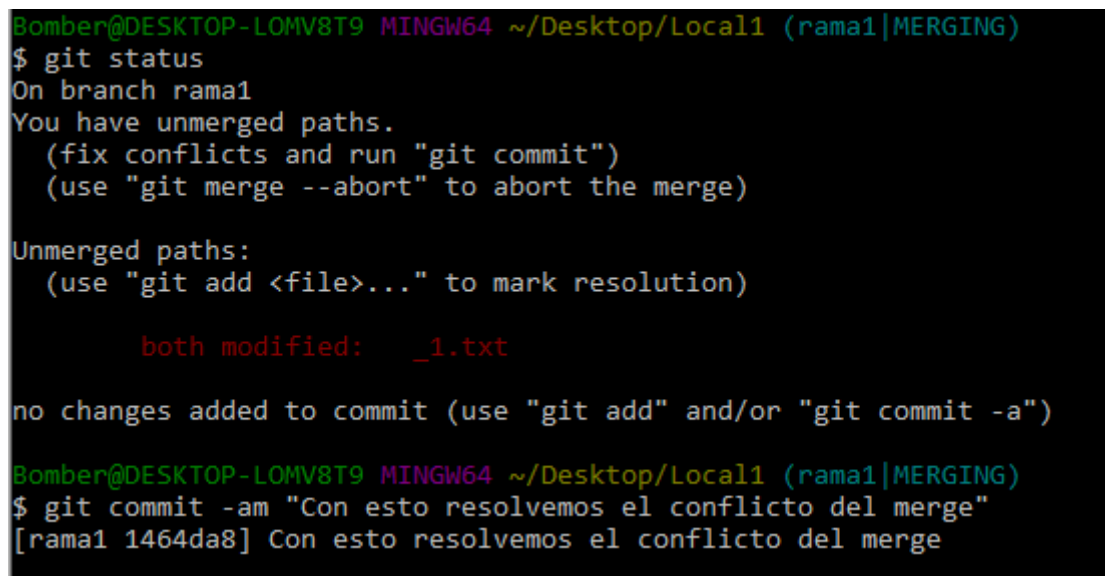
```
1 |Primera linea
2 |<<<<<< HEAD (Cambio actual)
3 |Modif1 - Añadiendo esto en rama1, crearemos un conflicto con rama2
4 |=====
5 |Modif2 - Añadiendo esto en rama2, crearemos un conflicto con rama1
6 |>>>>>> rama2 (Cambio entrante)
7 |
```

Como nos pedía que diéramos prioridad a nuestra rama original, aceptaremos los cambios de la rama1.



```
1 |Primera linea
2 |Modif1 - Añadiendo esto en rama1, crearemos un conflicto con rama2
3 |
```

Ahora para resolver el conflicto, haremos un commit para aceptar los cambios.



```
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (rama1|MERGING)
$ git status
On branch rama1
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:   _1.txt

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

Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (rama1|MERGING)
$ git commit -am "Con esto resolvemos el conflicto del merge"
[rama1 1464da8] Con esto resolvemos el conflicto del merge
```

## Creando un submodulo y lo obviaremos con .gitignore.

Crearemos una carpeta y dentro de esta crearemos un fichero.

Añadiremos a nuestro **.gitignore**.

```
Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (rama1)
$ mkdir sub_modulo && cd $_ && touch .libreria

Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1/sub_modulo (rama1)
$ cd ..

Bomber@DESKTOP-LOMV8T9 MINGW64 ~/Desktop/Local1 (rama1)
$ echo "sub_modulo/" > .gitignore
```