MANUAL GIT y GitHub

Ing. Luis Fco. Hdez. Bravo

Contenido

1) GIT: ¿Qué es?

Git se ha convertido en el estándar mundial para el control de versiones. Entonces, ¿qué es exactamente?

Git es un sistema de control de versiones distribuido, lo que significa que un clon local del proyecto es un repositorio de control de versiones completo. Estos repositorios locales plenamente funcionales permiten trabajar sin conexión o de forma remota con facilidad. Los desarrolladores confirman su trabajo localmente y, a continuación, sincronizan su copia del repositorio con la copia en el servidor. Este paradigma es distinto del control de versiones centralizado, donde los clientes deben sincronizar el código con un servidor antes de crear nuevas versiones.

El sistema es muy parecido al símbolo de sistema y al igual que este, debemos dar enter para ejecutar cada línea de código.

2) GIT: Usos

- Historial de código
- Almacenar código
- Trabajar en equipo
- Saber cuándo se introduce un error

3) GIT: revisar versión

Código
`$ git --version`
Resultado
`git version 2.39.0.windows.1`

4) GIT: configuración de usuario

Código
`$ git config --global user.name "Nompre apellido "`

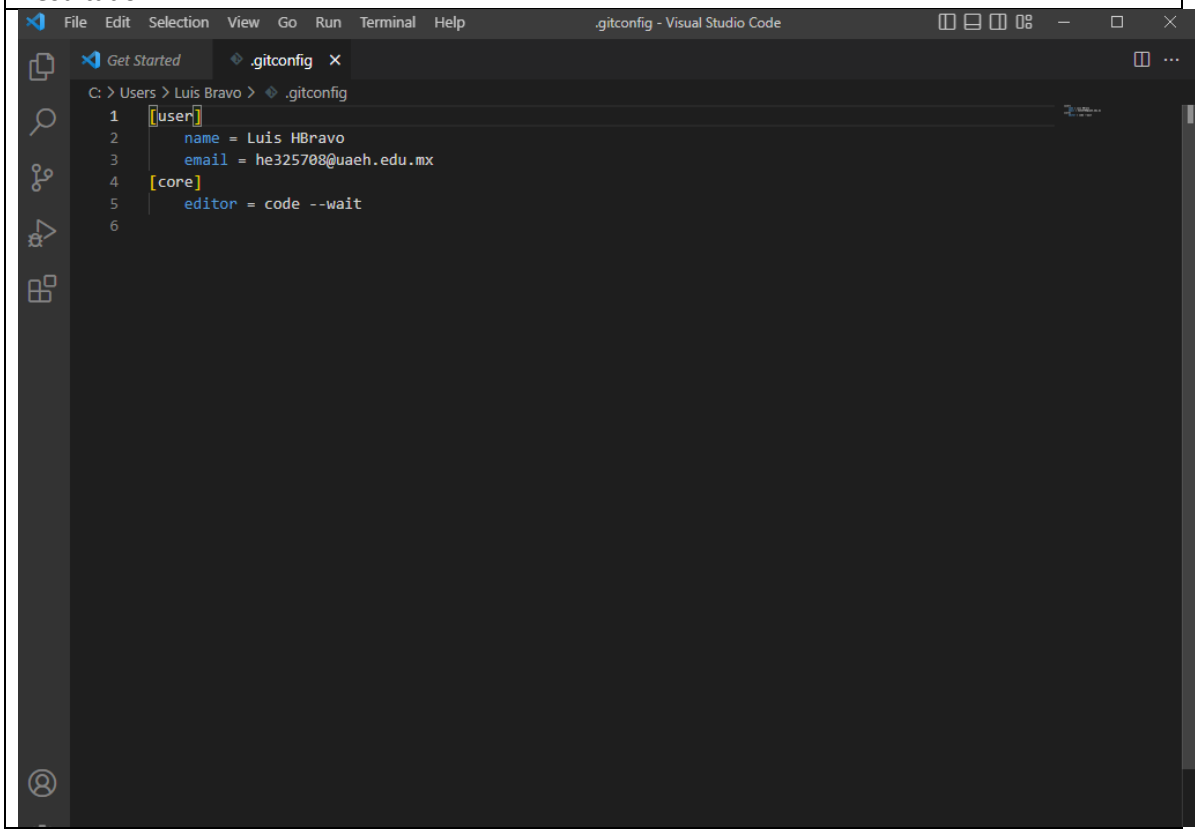
5) GIT: configuración de email

Código
$ git config --global user.email he325708@uaeh.edu.mx

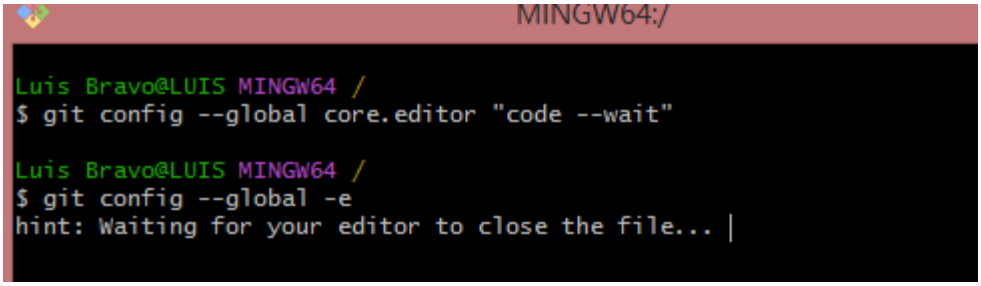
6) GIT: enlazando a VS CODE

Debes tener instalado VS CODE
Ahora, configuraremos VS CODE como nuestro editor de texto

Código
`$ git config --global core.editor "code --wait"` --presionamos enter `$ git config --global -e` --presionamos enter
Resultado



En automático nos abre VS Studio, ya con el nombre y el email que dimos de alta, mientras no cerremos el editor Git mostrara el siguiente mensaje



Una vez que cerremos el editor

Volverá a este estado

7) GIT: (core.auto.crlf) configurar salto de línea

Windows
$ git config --global core.autocrlf true
Mac o linux
$ git config --global core.autocrlf input

8) GIT: (git config –h) revisar todas las configuraciones

Código

```
$ git config -h
```

Resultado

```
$ git config -h
usage: git config [<options>]

Config file location
    --global              use global config file
    --system              use system config file
    --local               use repository config file
    --worktree            use per-worktree config file
    -f, --file <file>     use given config file
    --blob <blob-id>      read config from given blob object

Action
    --get                 get value: name [value-pattern]
    --get-all             get all values: key [value-pattern]
    --get-regexp          get values for regexp: name-regex [value-pattern]
    --get-urlmatch        get value specific for the URL: section[.var] URL
    --replace-all         replace all matching variables: name value [value-patt
ern]
    --add                 add a new variable: name value
    --unset               remove a variable: name [value-pattern]
    --unset-all           remove all matches: name [value-pattern]
    --rename-section      rename section: old-name new-name
    --remove-section      remove a section: name
    -l, --list            list all
    --fixed-value         use string equality when comparing values to 'value-pa
ttern'
          ...                     ...
ttern'
    -e, --edit            open an editor
    --get-color           find the color configured: slot [default]
    --get-colorbool       find the color setting: slot [stdout-is-tty]

Type
    -t, --type <type>     value is given this type
    --bool                value is "true" or "false"
    --int                 value is decimal number
    --bool-or-int         value is --bool or --int
    --bool-or-str         value is --bool or string
    --path                value is a path (file or directory name)
    --expiry-date         value is an expiry date

Other
    -z, --null            terminate values with NUL byte
    --name-only           show variable names only
    --includes            respect include directives on lookup
    --show-origin         show origin of config (file, standard input, blob, com
mand line)
    --show-scope          show scope of config (worktree, local, global, system,
 command)
    --default <value>     with --get, use default value when missing entry
```

9) GIT: (ls) navegar listar todos los archivos de la carpeta

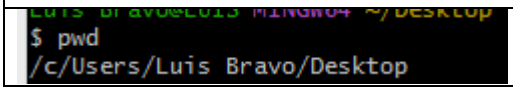
Código

```
$ ls
```

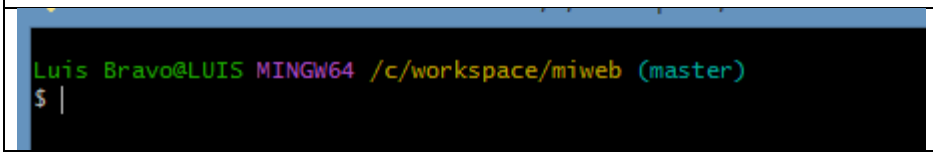
Resultado

```
Luis Bravo@LUIS MINGW64 /
$ ls
LICENSE.txt         cmd/   git-bash.exe*  proc/          unins000.exe*
ReleaseNotes.html   dev/   git-cmd.exe*   tmp/           unins000.msg
bin/                etc/   mingw64/       unins000.dat   usr/

Luis Bravo@LUIS MINGW64 /
```

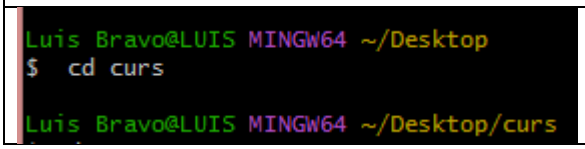
10) GIT: (pwd) navegar ver en qué carpeta me encuentro

Código
$ pwd
Resultado


11) GIT: (clear) limpiar

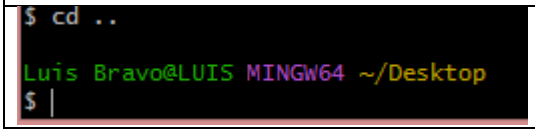
Código
$ clear
Resultado


12) GIT: (cd) navegar entrar a carpetas

Código
$ cd nombreCarpeta
Resultado


Tip: puedes escribir las primeras o primera letra del nombre de la carpeta y presionar TAB para autocompletar

13) GIT: (cd ..) navegar salir de carpeta

Código
$ cd ..
Resultado


14) GIT: (mkdir) crear una carpeta

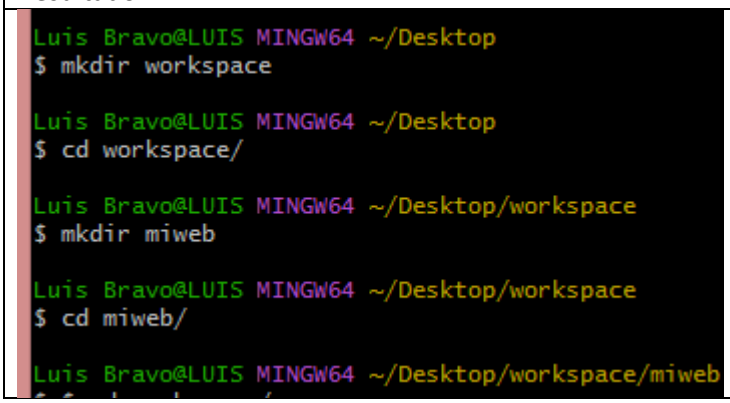
Código
mkdir nombreCarpetaNueva ls -- opcional: para revisar si se creo
Resultado


15) GIT: Ejercicio crear y navegar entre carpetas

Crea una carpeta desde GITBASH en el disco local C llamala workspace y dentro de ella una carpeta llamada mi web

Código

$ mkdir workspace
$ cd workspace/
$ mkdir miweb
$ cd miweb/

Resultado


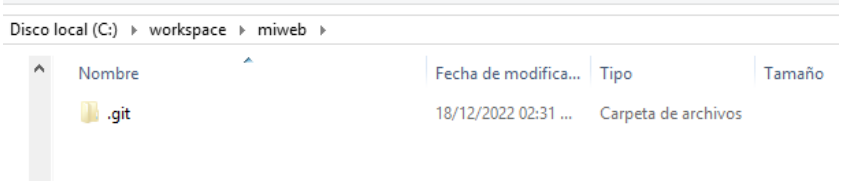
Recuerda por cada línea de código hay que dar ENTER

16) GIT: (git init) inicializar

Código
$ git init
Resultado

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb
$ git init
Initialized empty Git repository in C:/workspace/miweb/.git/

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$
```

La carpeta que se creó esta oculto en Windows podemos activar las opciones para verla



17) GIT: (ls –a) mostrar elementos ocultos

Código
$ ls -a
Resultado

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ ls -a
./  ../  .git/

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$
```

Para ingresar a una carpeta oculta usamos el mismo procedimiento que vimos en GIT: navegar entrar a carpetas, siempre colocando el punto

Código
$ cd .git
Resultado

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ cd .git

Luis Bravo@LUIS MINGW64 /c/workspace/miweb/.git (GIT_DIR!)
$
```

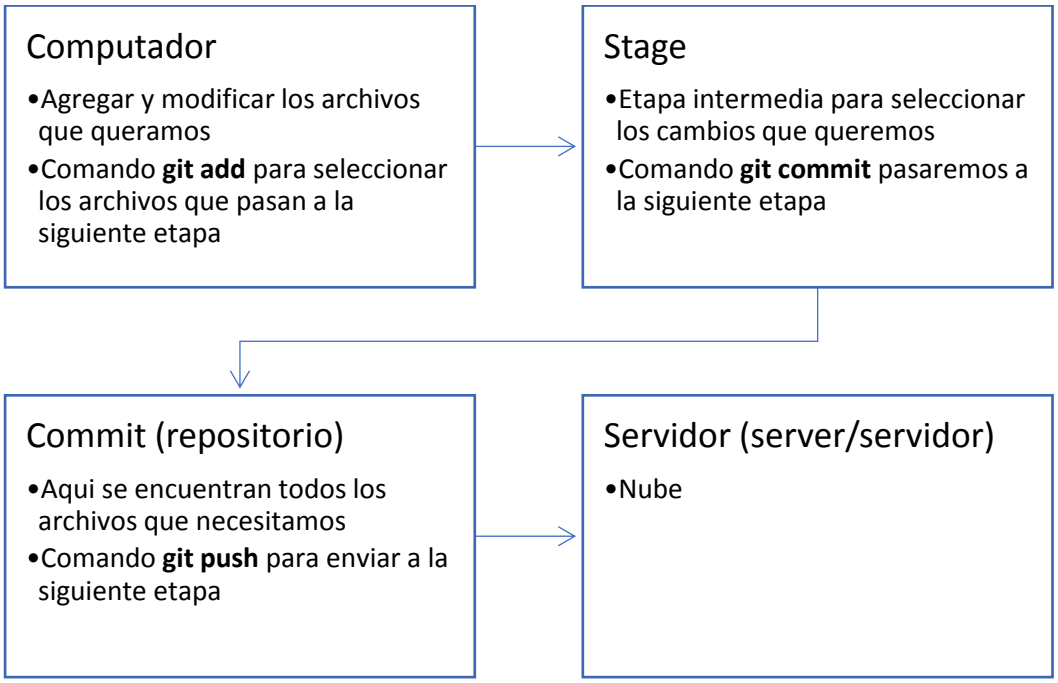
18) GIT: la carpeta. Git

Podemos ver el contenido aplicando el mismo código del punto anterior

Código
$ ls -a
Resultado

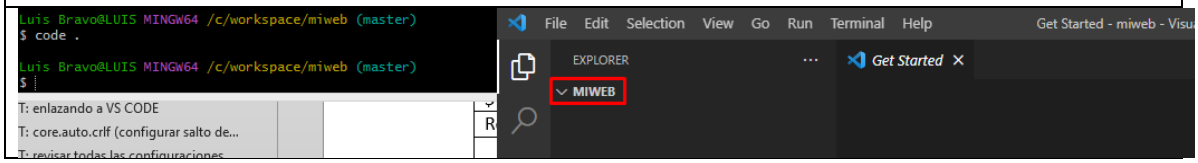
```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb/.git (GIT_DIR!)
$ ls -a
./  ../  HEAD  config  description  hooks/  info/  objects/  refs/
```

19) GIT: flujo de trabajo



20) GIT: (code .)Abrir editor de texto

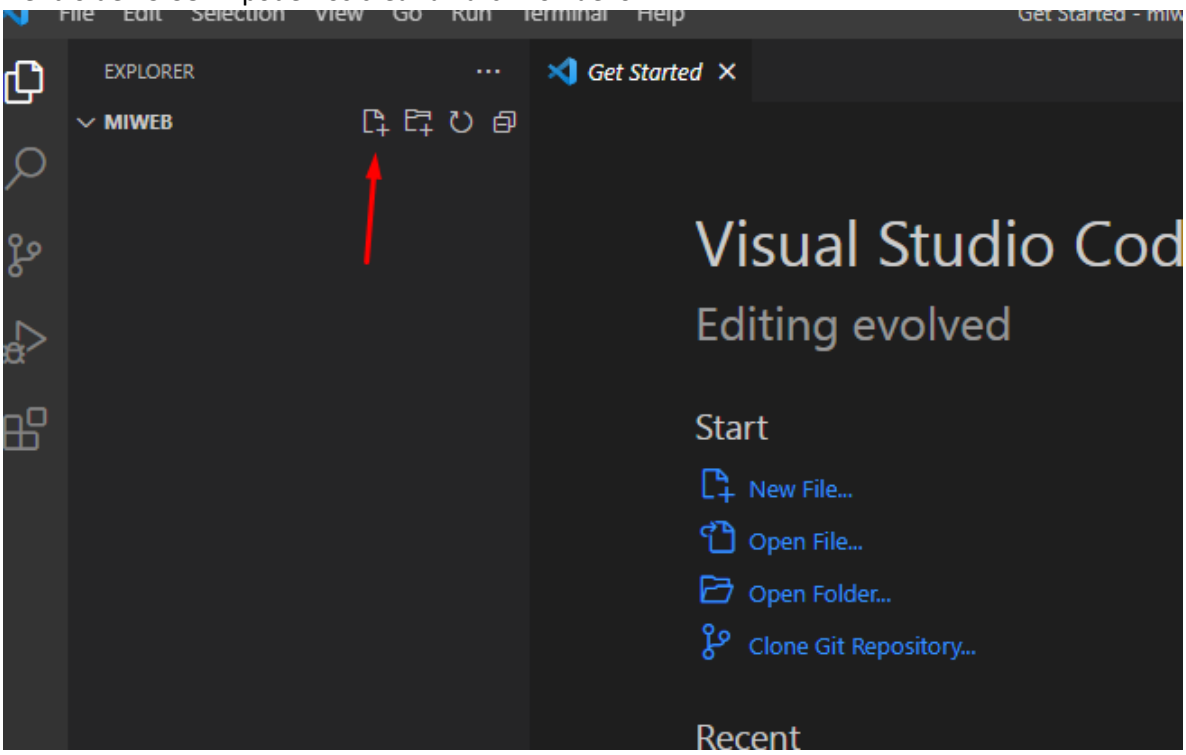
Código
$ code.
Resultado



Observemos que el VS CODE nos aparece la carpeta en la que trabajamos

21) GIT: creando archivo

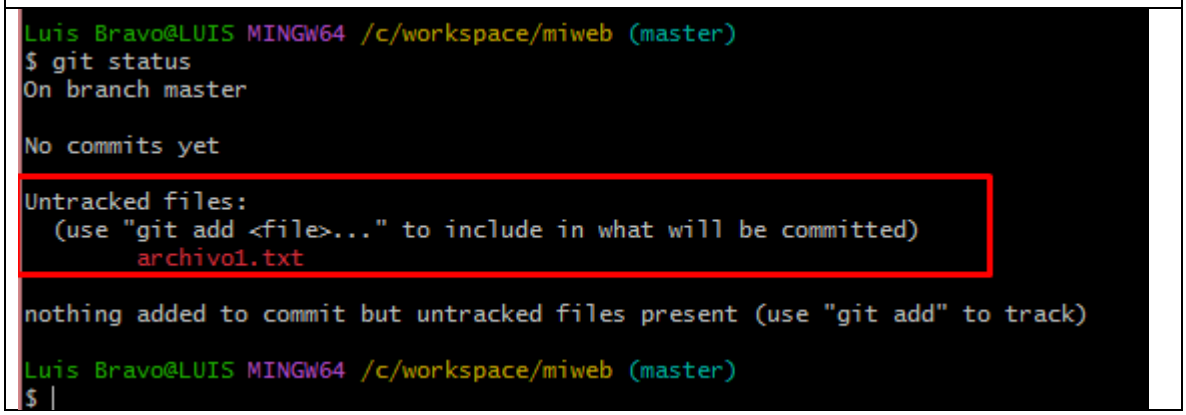
Dentro de VS CODE podemos crear un archivo nuevo



Podemos colocar el nombre y extensión que necesitemos (java, javascript, etc), en este caso dejaremos nombre archivo1.txt y dentro escribiremos la frase cerdito feliz

22) GIT: (git status) Estado de los archivos

Código
$ git status
Resultado



Nos dice que no está siguiendo los archivos

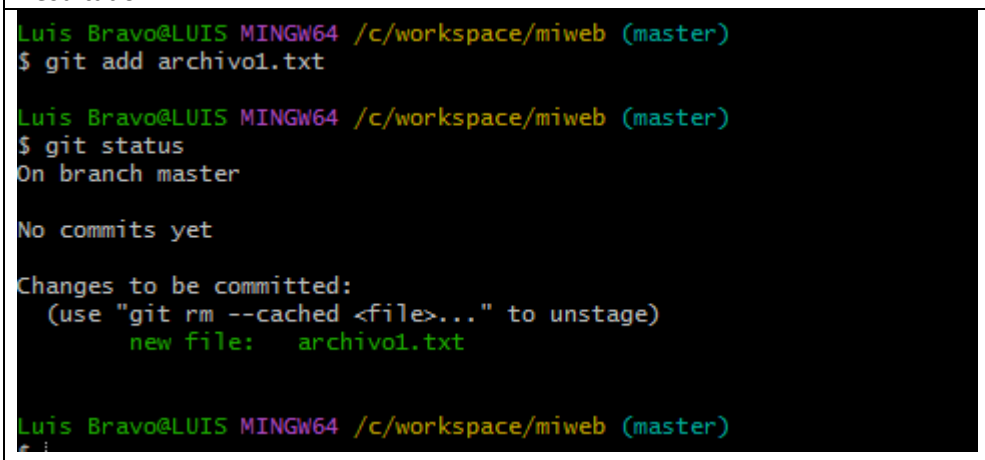
23) GIT: (git add) seleccionar archivos a seguir (agregar)

Cuando nosotros usamos git add lo que hacemos es agregar los cambios en el espacio de trabajo, no agregar los archivos, esto se entenderá en temas más adelante a partir de: GIT: ¿Qué pasa si modifico un archivo ya agregado?

Código

$ git add nombreArchivo --agrega solo el archivo solicitado
$ git add *.txt --agrega los archivos con esa extensión
$ git add . – al poner el punto decimos que agregue todo (no recomendable podemos agregar alguno que no necesitemos)

Resultado

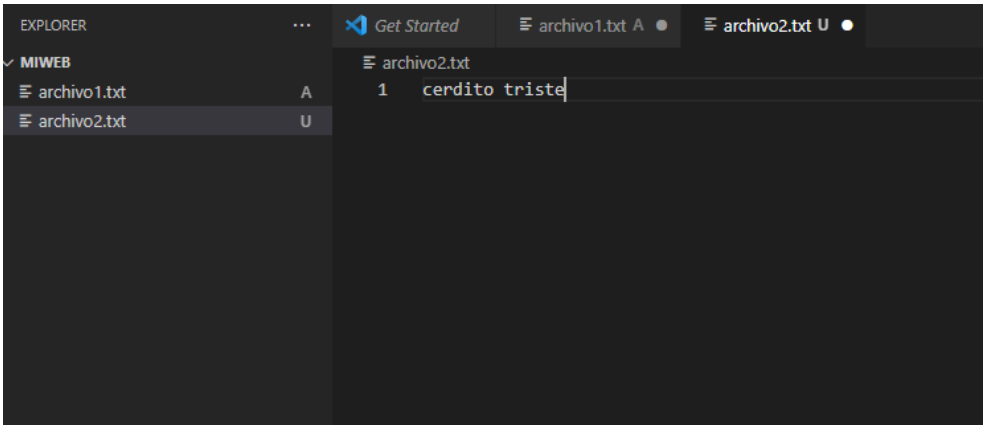


Como podemos ver usamos git status para verificar que se agregó el archivo1.txt y que se puede comprometer (commit)

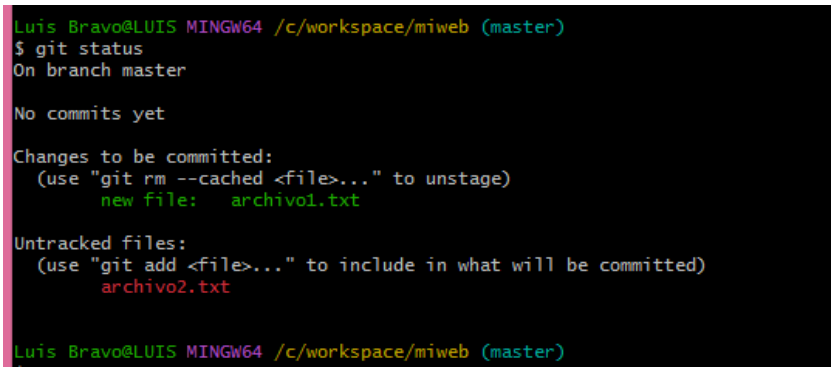
Tip: puedes escribir las primeras o primera letra del nombre de la carpeta y presionar TAB para autocompletar

24) GIT: Ejercicio crea archivo y agrégalo (add)

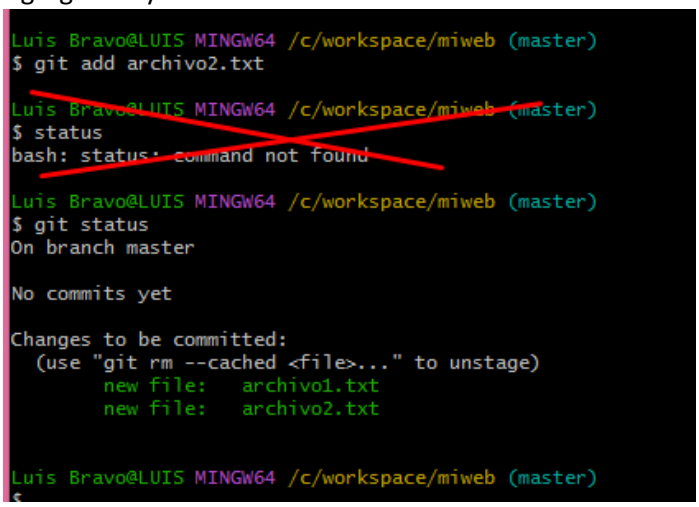
Realiza un segundo archivo2.txt con el texto cerdito triste, además agrégalo y revisa en status



Usamos status para ver que no está agregada



Agregamos y usamos status



En la imagen podemos ver que no se colocó git status, primero status marcando un error.
En el segundo intento podemos ver que nos muestra que el archivo dos se agrego

25) GIT: (git add) agregar mas de 2 archivos a la vez

Código
$ git add nombreArchivo1.extension nombreArchivo2.extension nombreArchivo3.extension

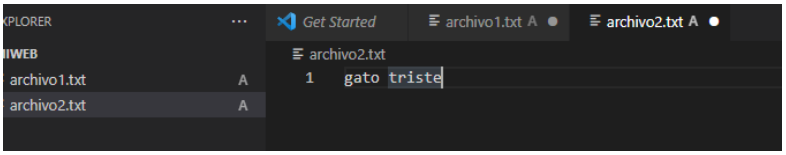
Al final de cada extensión debemos dejar un espacio

26) Git (cat) revisar contenido de un archivo

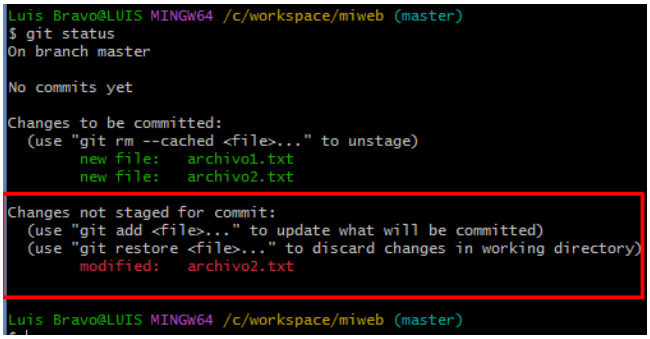
Código
$ cat nombreArchivo.Extension

27) GIT: ¿Qué pasa si modifico un archivo ya agregado?

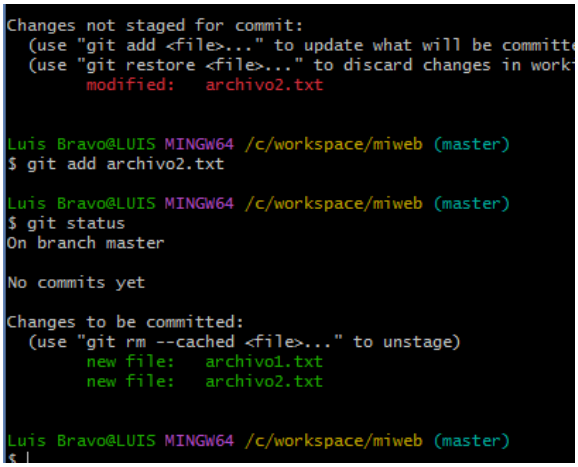
Vamos a VS CODE y modifiquemos el archivo 2, cambiemos de cerdito triste a gato triste y guardamos (CTRL+S)



Al aplicar status



Ahora nos marca que se modificó el archivo 2, para solucionarlo agregamos (add) otra vez el archivo 2



28) GIT: (git commit –m “text”) comprometer un archivo

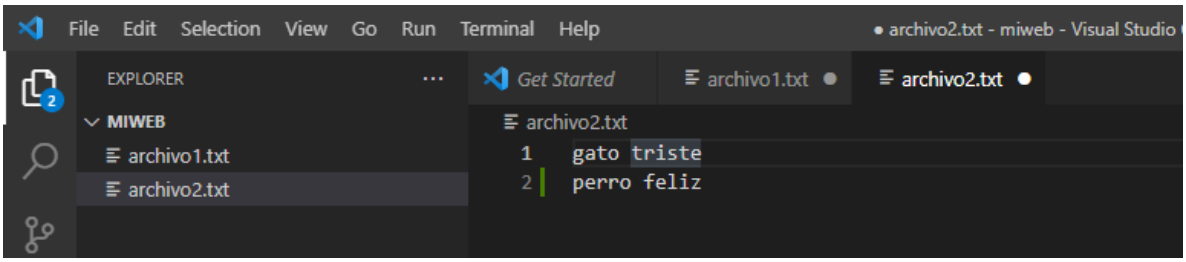
Ahora estamos en la fase para comprometer los archivos, es importante mandar un mensaje al hacer el compromiso, pues servirá de referencia para saber qué cambios o arreglos se hicieron

Código
$ git commit –m “commit inicial” $ git status
Resultado

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git commit -m "commit inicial"
[master (root-commit) 7ffad3f] commit inicial
 2 files changed, 1 insertion(+)
 create mode 100644 archivo1.txt
 create mode 100644 archivo2.txt

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status
On branch master
nothing to commit, working tree clean

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
```

29) GIT: ¿Qué pasa si modifico un archivo ya comprometido?

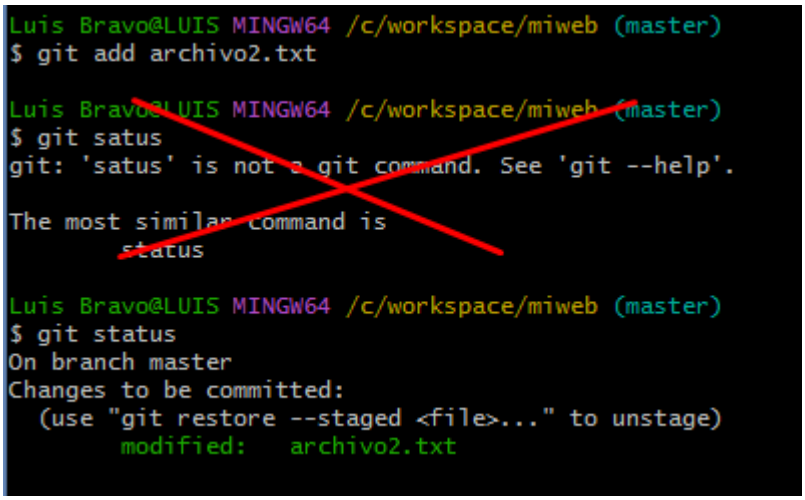
Nos vamos a VS CODE al archivo 2 y agregaremos un perro feliz y guardamos



Revisamos el status

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status
On branch master
nothing to commit, working tree clean

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working dir
        modified:   archivo2.txt

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

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
```

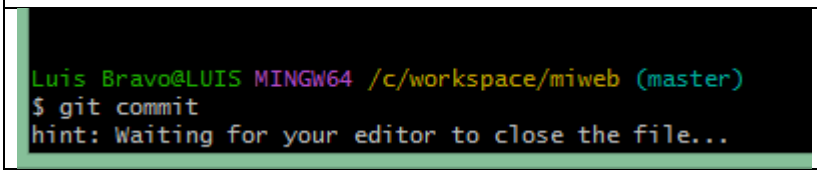
Y nos sale que se ha modificado el archivo 2, a partir de aquí realizamos lo mismo que vimos en: GIT: ¿Qué pasa si modifico un archivo ya agregado?, agregamos el archivo y revisamos status.



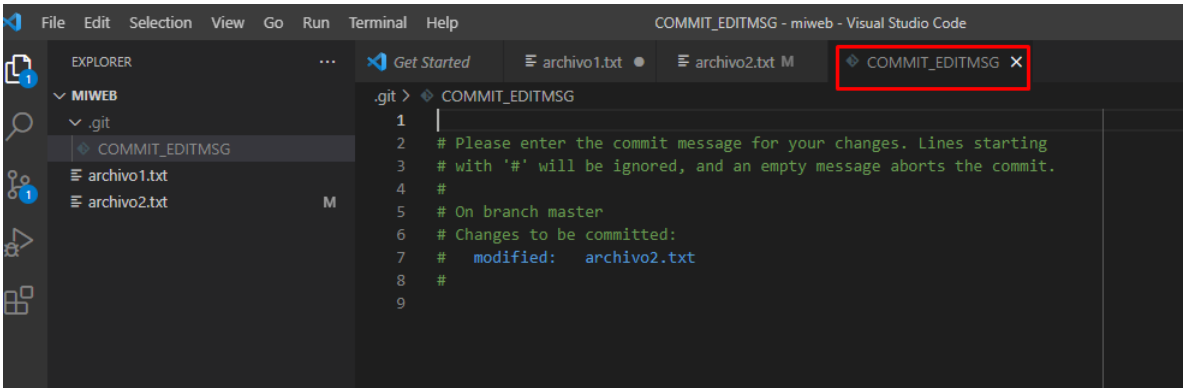
Ahora nos dice que hay cambios listos para comprometerse

30) GIT: (git commit) comprometer sin colocar un texto

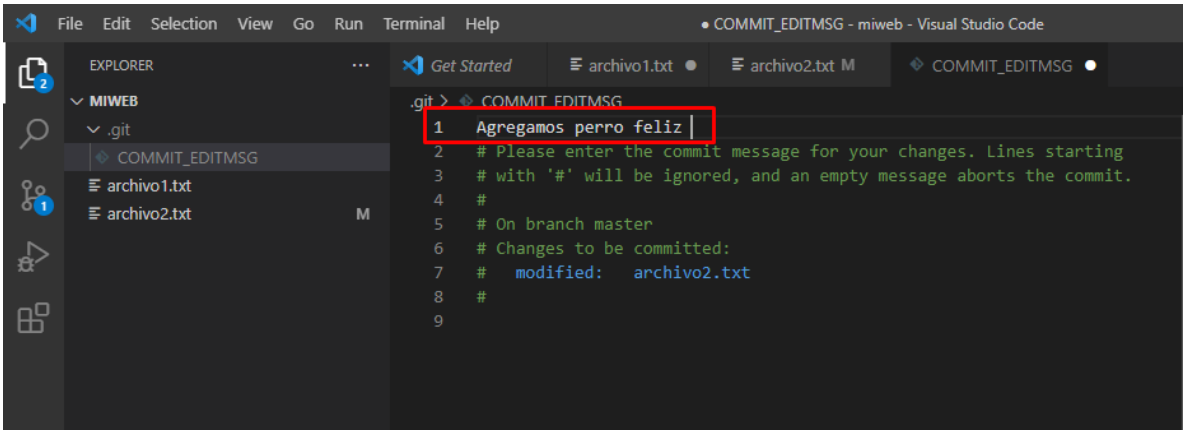
Con el ejemplo anterior

Código
$ git commit –sin colocar el guion la m, las comillas y el texto
Resultado


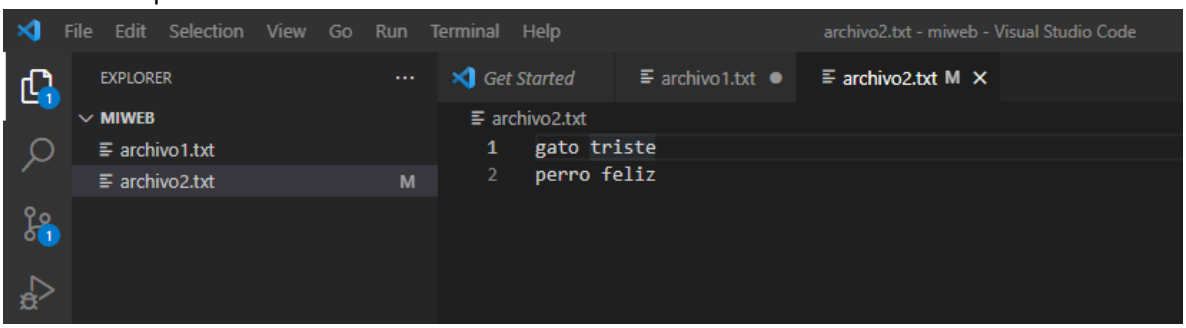
En automático se abrirá nuestro editor de codigo en este caso VS CODE y nos abre una nueva pestaña



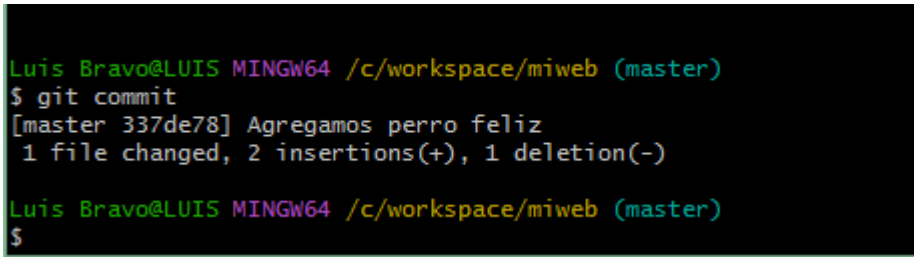
Ahora escribimos los cambios que se hicieron y guardamos



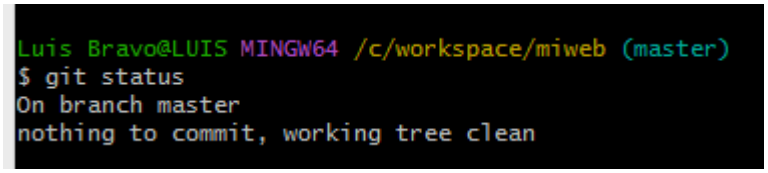
Cerramos la pestaña



Regresamos a la terminal



En automático nos coloca el texto que agregamos en nuestro editor de código, revisamos con status



Ahora ya no hay nada que comprometer

31) GIT: (rm) eliminar archivo

Vamos a eliminar el archivo 2

Código
$ rm nombreArchivo.Extension $ git status
Resultado

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ rm archivo2.txt

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    archivo2.txt

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

Ahora nos dice que dicho cambio de este archivo no se encuentra en la etapa de stage (GIT: flujo de trabajo), lo siguiente que se debe agregar dicho cambio

Código
$ git add nombreArchivo. Extension $ git status
Resultado

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git add archivo2.txt

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    archivo2.txt


Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
```

Ahora podemos comprometer dichos cambios

Código
$ git commit –m “Elminando archivo 2” --recuerda que el texto depende del nombre del archivo o de lo que hayas borrado $ git status
Resultado

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git commit -m "Eliminando archivo 2"
[master 06f57ca] Eliminando archivo 2
 1 file changed, 2 deletions(-)
 delete mode 100644 archivo2.txt

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status
On branch master
nothing to commit, working tree clean

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$
```

Revisamos cuantos archivos tenemos

Código
$ ls
Resultado

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ ls
archivo1.txt

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$
```

Y en efecto el archivo 2 se elimino

32) GIT: (git rm) eliminar arhivo de la fase stage en un solo paso

Ahora vamos a eliminar el archivo 1 en un solo paso

Código
$ git rm nombreArchivo. Extension $ git status
Resultado

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git rm archivo1.txt
rm 'archivo1.txt'

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    archivo1.txt
```

Ahora el cambio está listo para ser comprometido

33) GIT: (git restored) quitar cambio de etapa de stage

Para quitar un cambio de la etapa de stage usamos el siguiente codigo que el mismo GitBash nos da

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    archivo1.txt
```

Código
$ git restore –staged nombreArchivo.extension $ git status
Resultado

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git restore --staged archivo1.txt

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    archivo1.txt

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

Ahora este archivo ya no se encuentra en nuestra etapa de stage

34) Git: (git restore) descartar cambios

Si revisamos con ls nos daremos cuenta de que ya no tenemos ningún archivo

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ ls

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$
```

Por lo cual usaremos el siguiente código tal cual nos lo da el Gibas

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    archivo1.txt

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

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
```

Código

$ git restore nombreArchivo.extension
$ git status
$ git ls

Resultado

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git restore archivo1.txt

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status
On branch master
nothing to commit, working tree clean

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ ls
archivo1.txt
```

Ahora podemos ver que en status nos marca que no hay nada que comprometer o regresar al repositorio y con ls que recuperamos nuestro archivo

35) GIT: (mv) mover/ cambiar nombre archivo

Código

$ mv nombreActualArchivo.extension nombreNuevoArchivo.extension
ls
$ git status

Resultado

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ mv archivo1.txt archivo.txt

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ ls
archivo.txt

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    archivo1.txt

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

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

Podemos observar que se hizo el cambio de nombre pero a su vez en status nos marca que se borró el archivo 1 y que ahora tenemos el que solo se llama archivo hay que agregar los cambios realizados

Código

$ git add nombreAntiguoArchivo.extension nombreActualArchivo.extension
$ git status
$ git commit

Resultado

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git add archivo1.txt archivo.txt

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        renamed:    archivo1.txt -> archivo.txt


Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git commit -m "renombrando archivo"
[master 6c22d1e] renombrando archivo
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename archivo1.txt => archivo.txt (100%)
```

36) GIT: (git mv) mover/ cambiar nombre archivo y poner en fase stage

Regresaremos el nombre del archivo a archivo 1

Código

$ git mv nombreActualArchivo.extension nombreNuevoArchivo.extension
$ git status
$ git commit

Resultado

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git mv archivo.txt archivo1.txt

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        renamed:    archivo.txt -> archivo1.txt
```

Ahora solo nos falta volver al repositorio o a comprometer el archivo

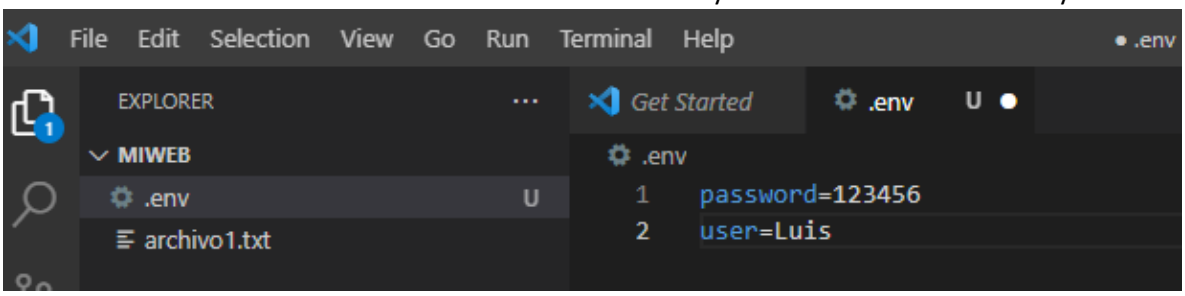
Código
`$ git commit -m"devolviendo el nombre al archivo"`
Resultado

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git commit -m"devolviendo el nombre al archivo"
[master 4e0f46b] devolviendo el nombre al archivo
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename archivo.txt => archivo1.txt (100%)
```

37) GIT: ignorar archivos

Para que no se incluyan en nuestros repositorios (fase commit: GIT: flujo de trabajo), como pueden ser variables de entornos como bases de datos y que esta no se suba por error.

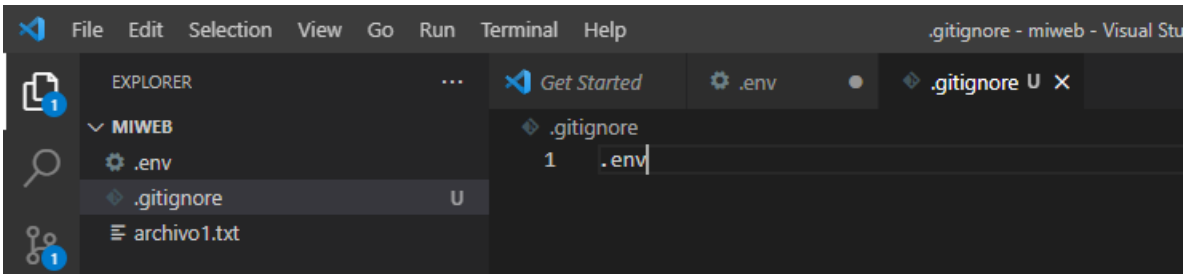
Creamos un nuevo archivo en VS CODE con el nombre .env y escribimos una contraseña y usuario



Revisamos con status

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .env

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

Creamos otro archivo que se llamara .gitignore y dentro escribimos el nombre del archivo a ignorar



Ahora revisamos con estatus

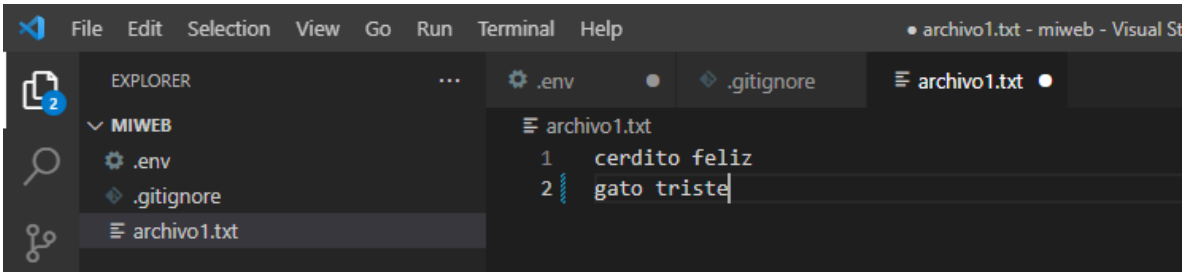
```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

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

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$
```

Agregamos (add) y comprometemos (commit) el archivo .gitignore

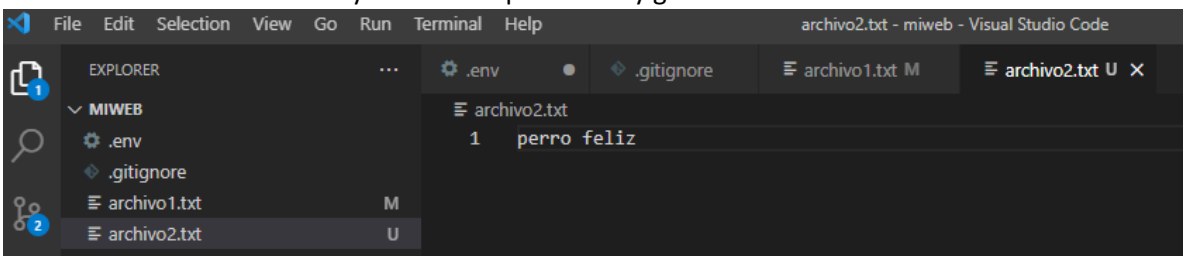
```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git add .gitignore

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git commit -m"agregando archivo .gitignore"
[master 9893203] agregando archivo .gitignore
 1 file changed, 1 insertion(+)
 create mode 100644 .gitignore

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$
```

38) Git: (git status –s) mejor git status

Vamos al archivo 1 y agregamos un gato triste y guardamos



Y otra vez creamos archivo 2 y escribimos perro feliz y guardamos



Escribimos git status

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working director
        modified:   archivo1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        archivo2.txt

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

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$
```

Con esto nos sale toda esta información, ahora intentemos esto

Código
$ git status -s
Resultado

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status -s
 M archivo1.txt
?? archivo2.txt
```

La M roja indica que se modificó y que se debe comprometer y los signos de interrogación que se debe agregar, primero agreguemos el archivo2 y usemos otra vez status-s

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git add archivo2.txt

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status -s
 M archivo1.txt
A  archivo2.txt
```

Ahora sale una A que significa que ya se agregó, agreguemos el archivo 1

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git add archivo1.txt

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git satus -s
git: 'satus' is not a git command. See 'git --help'.

The most similar command is
        status

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status -s
M  archivo1.txt
A  archivo2.txt
```

Ahora la M sale de color verde lo que indica que se agregó la modificación y como los 2 están en verde están listos para comprometerse

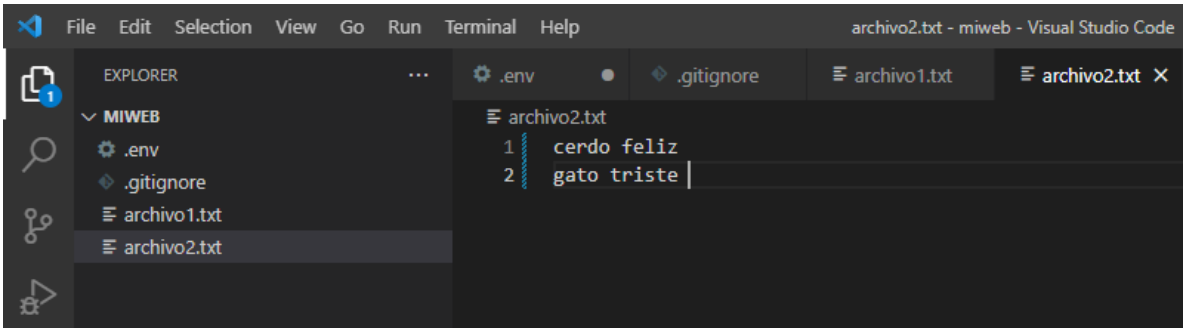
Ing. Luis Fco. Hdez. Bravo

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git commit -m "mostrando status corto"
[master 597276c] mostrando status corto
 2 files changed, 2 insertions(+)
 create mode 100644 archivo2.txt

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status -s

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$
```

Finalmente al aplicar status – s, no aparece nada porque ya todo está comprometido

39) GIT: (git diff) ver cambios realizados desde GitBash

Vamos a VS CODE y modifiquemos el archivo 2 quitemos perro feliz, y agreguemos cerdito feliz y gato triste, y guardamos



Revisamos status

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   archivo2.txt

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

Ahora usaremos

Código
$ git diff
Resultado

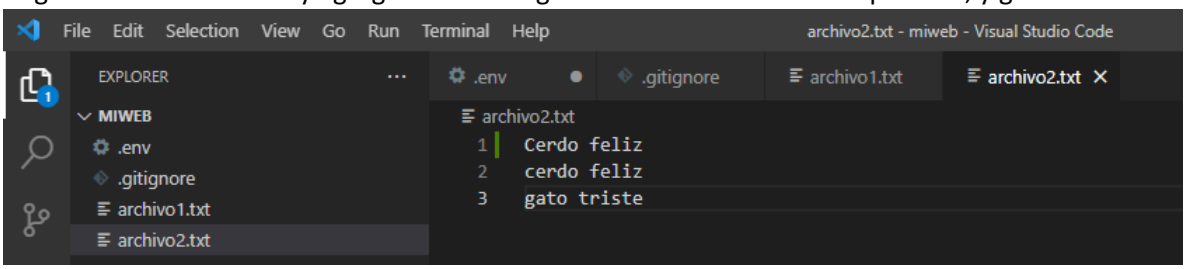
```
no changes added to commit (use "git add" and/or "git commit -a")

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git diff
diff --git a/archivo2.txt b/archivo2.txt
index 239e71e..2bf7544 100644
--- a/archivo2.txt
+++ b/archivo2.txt
@@ -1 +1,2 @@
-perro feliz
\ No newline at end of file
+cerdo feliz
+gato triste
\ No newline at end of file
```

Podemos ver que nos dice que se eliminó perro feliz y se agregó cerdo feliz y gato triste, agreguemos y comprometamos

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git add archivo2.txt
```

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git commit -m "se modifico archivo2"
[master 7738d48] se modifico archivo2
 1 file changed, 2 insertions(+), 1 deletion(-)
```

Regresemos al archivo 2 y agreguemos un segundo cerdo feliz arriba del primero, y guardamos



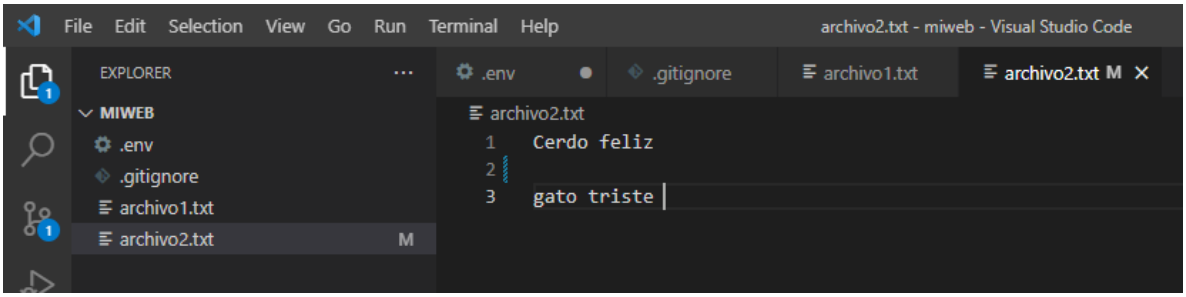
Usamos git diff

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git diff
diff --git a/archivo2.txt b/archivo2.txt
index 2bf7544..6a935f2 100644
--- a/archivo2.txt
+++ b/archivo2.txt
@@ -1,2 +1,3 @@
+Cerdo feliz
 cerdo feliz
 gato triste
\ No newline at end of file

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$
```

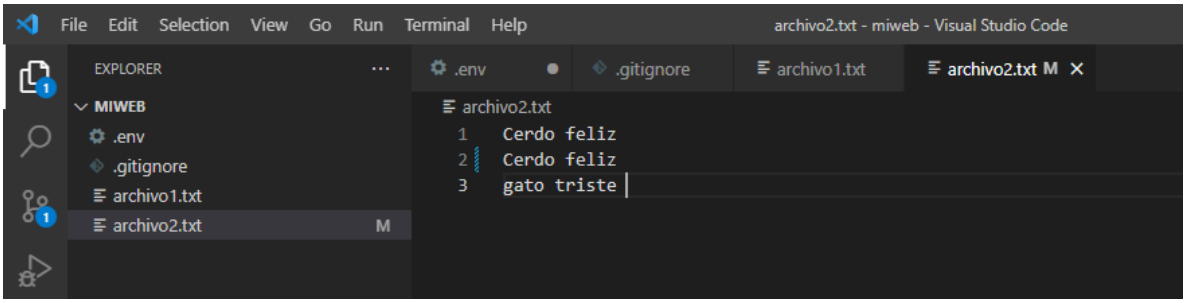
Nos sale ahora + cerdo feliz porque fue el que se agregó con el salto de línea volvemos a agregar y modificar

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git add archivo2.txt

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git commit -m "se modifico el archivo 2 otra vez"
[master 03e6e30] se modifico el archivo 2 otra vez
 1 file changed, 1 insertion(+)
```

Volvemos al archivo 2 eliminamos el segundo cerdo feliz guardamos



Y volvemos a escribir cerdo feliz



Ahora usamos git diff

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git diff
diff --git a/archivo2.txt b/archivo2.txt
index 6a935f2..1429f38 100644
--- a/archivo2.txt
+++ b/archivo2.txt
@@ -1,3 +1,3 @@
 Cerdo feliz
-cerdo feliz
+Cerdo feliz
 gato triste
\ No newline at end of file

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
```

Nos sale que quitamos el segundo cerdo feliz y que lo volvimos a escribir, agregamos y comprometemos

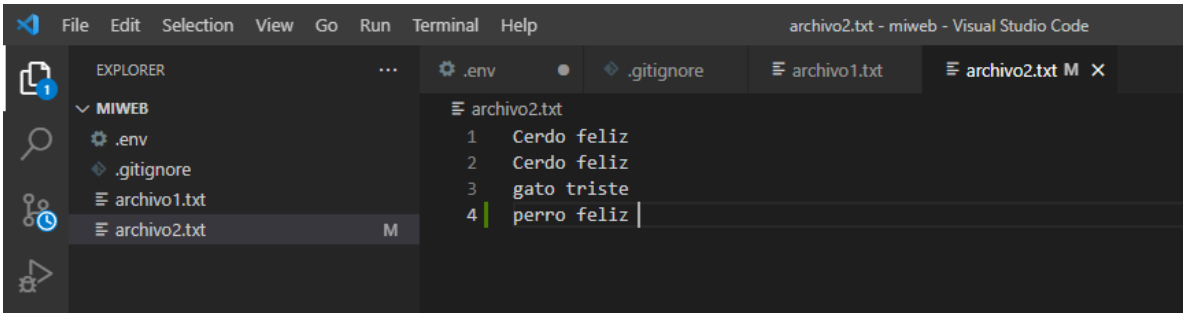
```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git add archivo2.txt

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git commit -m "Se modifico el archivo 2 otra vez 2"
[master c0fab01] Se modifico el archivo 2 otra vez 2
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Escribimos una última vez git diff y veamos que no nos sale nada

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git diff

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$
```

40) Git (git diff --staged) ver cambios en fase stage

Agreguemos perro feliz al archivo 2 y guardemos



Agreguemos y usaremos git diff

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git add archivo2.txt

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git diff

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$
```

No nos sale nada , ahora agregemos –staged

Código
$ git diff --staged
Resultado

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git diff --staged
diff --git a/archivo2.txt b/archivo2.txt
index 1429f38..8a068aa 100644
--- a/archivo2.txt
+++ b/archivo2.txt
@@ -1,3 +1,4 @@
 Cerdo feliz
 Cerdo feliz
-gato triste
\ No newline at end of file
+gato triste
+perro feliz
\ No newline at end of file

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$
```

Ahora podemos ver los cambios en la etapa o fase stage

41) GIT: (git log) ver historial de cambios

Con esto veremos todos los cambios que se fueron registrando, junto con el autor, correo y fecha exacta, para salir presionamos la leta Q en nuestro teclado

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git log
commit c0fab013ec62ccbccb3a24f638181a95c67bf638 (HEAD -> master)
Author: Luis HBravo <he325708@uaeh.edu.mx>
Date:   Sun Dec 18 17:34:44 2022 -0600

    Se modifico el archivo 2 otra vez 2

commit 03e6e3064c23d24eaf285a439cca71b64db76e61
Author: Luis HBravo <he325708@uaeh.edu.mx>
Date:   Sun Dec 18 17:30:09 2022 -0600

    se modifico el archivo 2 otra vez

commit 7738d48d8d77194a86219b790c2bf488e73ba83e
Author: Luis HBravo <he325708@uaeh.edu.mx>
Date:   Sun Dec 18 17:27:03 2022 -0600

    se modifico archivo2

commit 597276cffe8245e22756cc48289563998e2712d8
Author: Luis HBravo <he325708@uaeh.edu.mx>
Date:   Sun Dec 18 17:11:24 2022 -0600

    mostrando status corto
:...skipping...
commit c0fab013ec62ccbccb3a24f638181a95c67bf638 (HEAD -> master)
Author: Luis HBravo <he325708@uaeh.edu.mx>
Date:   Sun Dec 18 17:34:44 2022 -0600

    Se modifico el archivo 2 otra vez 2

commit 03e6e3064c23d24eaf285a439cca71b64db76e61
Author: Luis HBravo <he325708@uaeh.edu.mx>
Date:   Sun Dec 18 17:30:09 2022 -0600

    se modifico el archivo 2 otra vez

commit 7738d48d8d77194a86219b790c2bf488e73ba83e
Author: Luis HBravo <he325708@uaeh.edu.mx>
Date:   Sun Dec 18 17:27:03 2022 -0600

    se modifico archivo2
```

```
    se modifico el archivo 2 otra vez

commit 7738d48d8d77194a86219b790c2bf488e73ba83e
Author: Luis HBravo <he325708@uaeh.edu.mx>
Date:   Sun Dec 18 17:27:03 2022 -0600

    se modifico archivo2

commit 597276cffe8245e22756cc48289563998e2712d8
Author: Luis HBravo <he325708@uaeh.edu.mx>
Date:   Sun Dec 18 17:11:24 2022 -0600

    mostrando status corto

commit 9893203a6846d19c30b9d2e47deaee71fe8b4519
Author: Luis HBravo <he325708@uaeh.edu.mx>
Date:   Sun Dec 18 17:01:05 2022 -0600

    agregando archivo .gitignore

commit 2dfe25e2d89c5148d66593030a938ce0d19182d1
Author: Luis HBravo <he325708@uaeh.edu.mx>
Date:   Sun Dec 18 16:56:08 2022 -0600

    guarde por error

commit 4e0f46bc73f6942dba447f838dede90667a907e9
Author: Luis HBravo <he325708@uaeh.edu.mx>
Date:   Sun Dec 18 16:46:12 2022 -0600

    devolviendo el nombre al archivo

commit 6c22d1e45f4c08d009e5637345a90e93f35fdc11
Author: Luis HBravo <he325708@uaeh.edu.mx>
Date:   Sun Dec 18 16:39:41 2022 -0600

    renombrando archivo

commit 06f57cac2ab3be7fee02ae9d21d6d0d392388055
```

42) GIT: (git log --oneline) ver historial de cambios

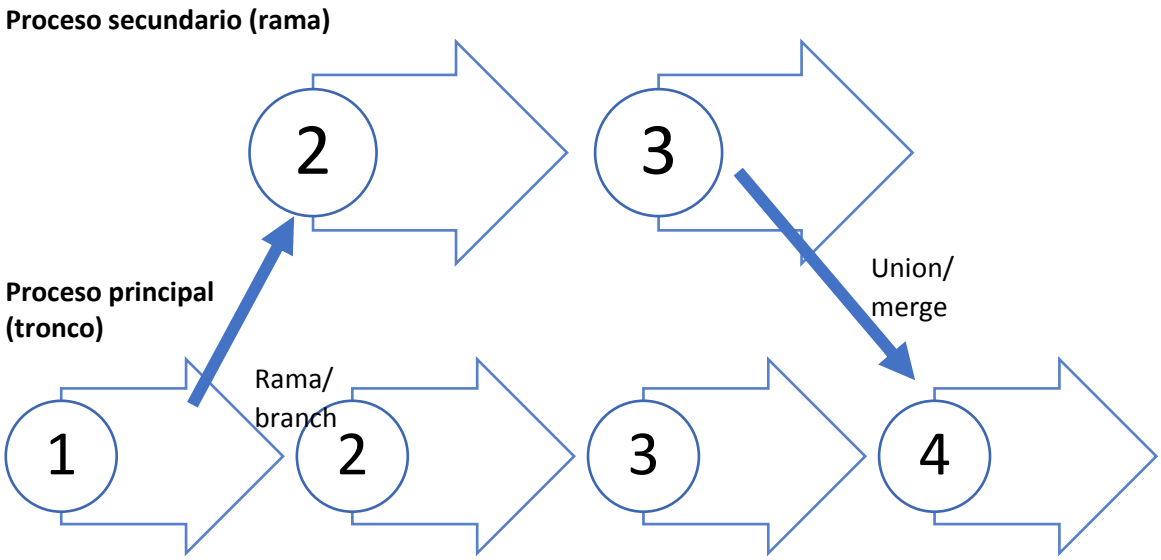
A diferencia de la versión anterior con esta opción nos saldrá un historial resumido

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git log --oneline
c0fab01 (HEAD -> master) Se modifico el archivo 2 otra vez 2
03e6e30 se modifico el archivo 2 otra vez
7738d48 se modifico archivo2
597276c mostrando status corto
9893203 agregando archivo .gitignore
2dfe25e guarde por error
4e0f46b devolviendo el nombre al archivo
6c22d1e renombrando archivo
06f57ca Eliminando archivo 2
337de78 Agregamos perro feliz
7ffad3f commit inicial

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$
```

Solo tendremos al inicio de cada acción un código para identificar ese punto en el historial, notemos que el historial va de abajo hacia arriba

43) Git: Branch (ramas) y merge (uniones)

Imaginemos ahora que trabajamos de la mano con otros desarrolladores, entonces no podemos trabajar de una forma lineal como lo hicimos hasta ahora, si no que hay que crear una rama (branch) y después unir (merge) para poder trabajar nuestra parte del proyecto.



44) Git: (git branch) revisar en que rama estamos

Vamos a revisar el status y asegurarnos que no hay nada que agregar o comprometer

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status
On branch master
nothing to commit, working tree clean

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$
```

Código
$ git branch
Resultado

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git branch
* master
```

Estamos en la rama principal

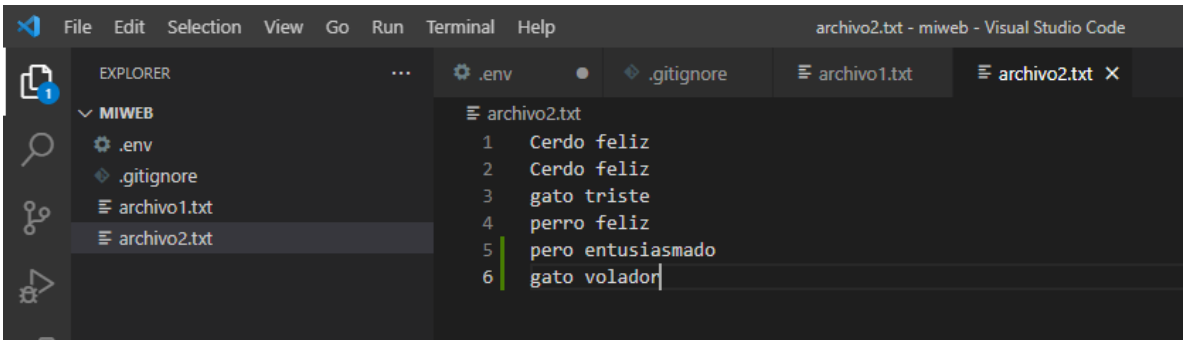
45) Git: (git checkout –b) crear rama

Ahora crearemos una rama

Código
$ git cheout – b nombreRama $ git branch -- podemos usar branch para revisar que realmente estemos en esa rama creada
Resultado

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git checkout -b ramab
Switched to a new branch 'ramab'

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (ramab)
$ git branch
  master
* ramab
```

Ahora podemos hacer cambios en los archivos vamos a VS CODE y modificamos el archivo 2, agregamos un perro entusiasmado y un gato volador, y guardamos



Revisamos status nos marcara la modificación que hicimos, agregamos y comprometemos

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (ramab)
$ git status
On branch ramab
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:   archivo2.txt

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

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (ramab)
$ git add archivo2.txt

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (ramab)
$ git commit -m "actualizando archivo 2 desde rama"
[ramab 3cf9748] actualizando archivo 2 desde rama
 1 file changed, 3 insertions(+), 1 deletion(-)
```

Revisamos en el historial con git log –oneline

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (ramab)
$ git log --oneline
3cf9748 (HEAD -> ramab) actualizando archivo 2 desde rama
21a4966 (master) guardando otra vez archivo2
c0fab01 Se modifico el archivo 2 otra vez 2
03e6e30 se modifico el archivo 2 otra vez
7738d48 se modifico archivo2
597276c mostrando status corto
9893203 agregando archivo .gitignore
2dfe25e guarde por error
4e0f46b devolviendo el nombre al archivo
6c22d1e renombrando archivo
06f57ca Eliminando archivo 2
337de78 Agregamos perro feliz
7ffad3f commit inicial

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (ramab)
$
```

Podemos ver que estamos en el último cambio realizado en la rama porque nos aparece la palabra (HEAD) y que se realizó en la ramab, limpiamos y usamos cat para ver el contenido

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (ramab)
$ cat archivo2.txt
Cerdo feliz
Cerdo feliz
gato triste
perro feliz
pero entusiasmado
gato volador
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (ramab)
$
```

Ahora nos cambiaremos a la rama master y revisaremos con cat

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ cat archivo2.txt
Cerdo feliz
Cerdo feliz
gato triste
perro feliz
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$
```

Y observemos que los cambios siguen como antes de empezar a trabajar este capítulo, con esto ahora sabemos que podemos tener un mismo archivo con más, menos o distintas líneas de código.

46) Git: (git merge) unir rama

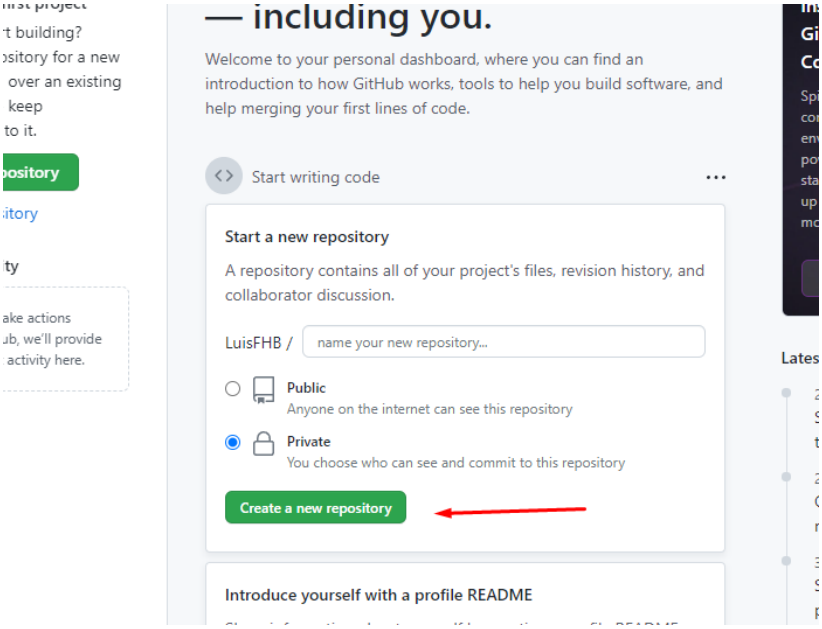
Para unir una rama con el tronco o proceso principal (Git: Branch (ramas) y merge (uniones)), debemos primero encontrarnos en la rama principal (master o main depende de la versión y sistema operativo)

Código
$ git merge nombreRama $ cat nombreArchivo.Extension –opcional para revisar que si se hicieron los cambios
Resultado

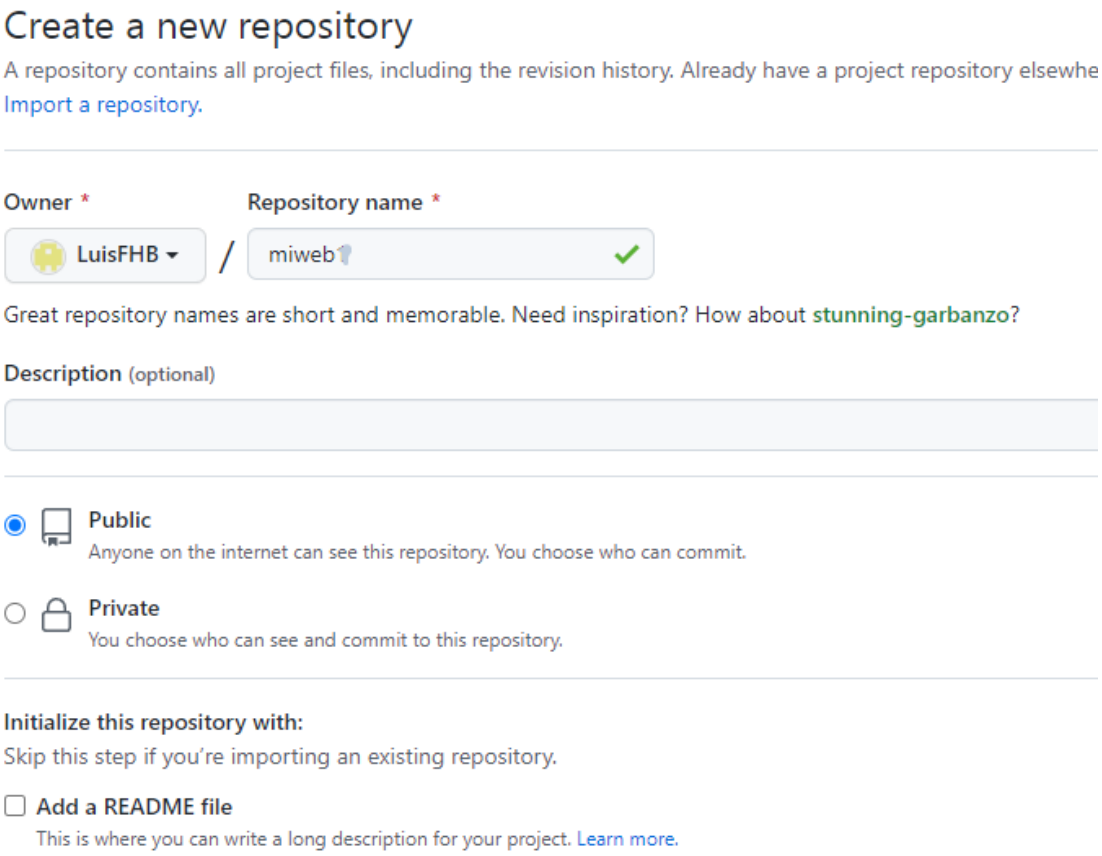
```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git merge ramab
Updating 21a4966..3cf9748
Fast-forward
 archivo2.txt | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ cat archivo2.txt
Cerdo feliz
Cerdo feliz
gato triste
perro feliz
pero entusiasmado
gato volador
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
```

47) Enlazar con GitHub

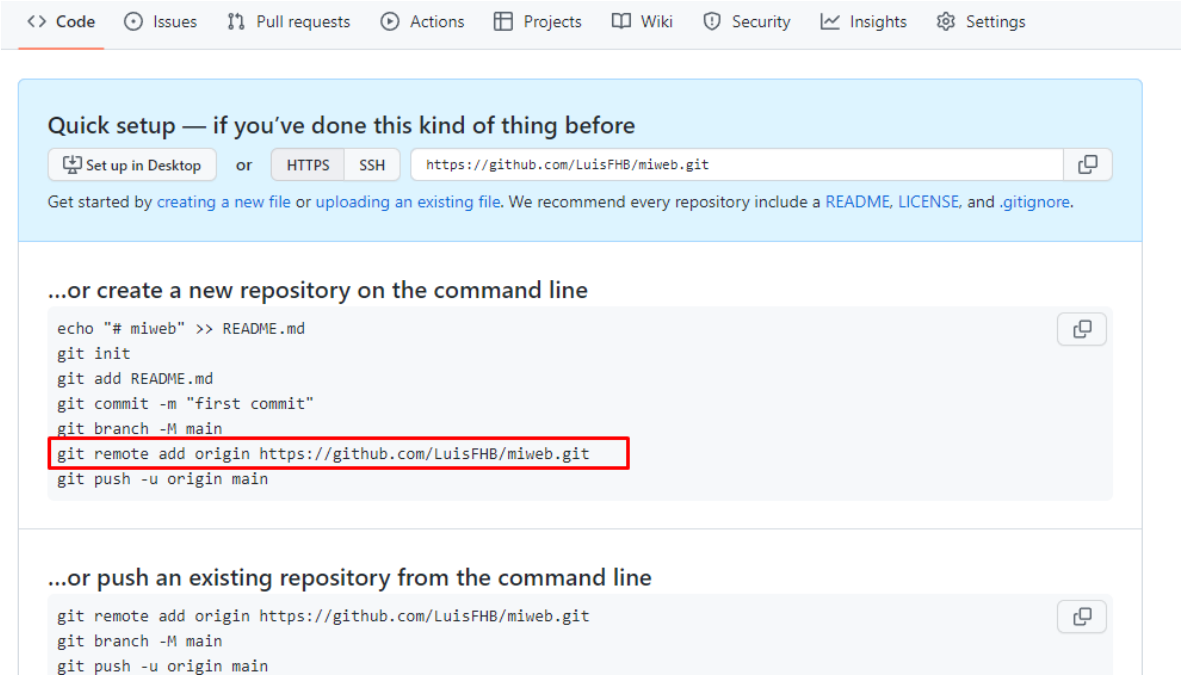
Entramos a GitHub y creamos un nuevo repositorio



Colocamos nombre, dejamos en público y no movemos nada más



Nos aparecerá un código como este y copiaremos el que se encuentra en este punto



Vamos a GitBash y pegamos la línea

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git remote add origin https://github.com/LuisFHB/miweb.git

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$
```

Con esto conectamos al servidor

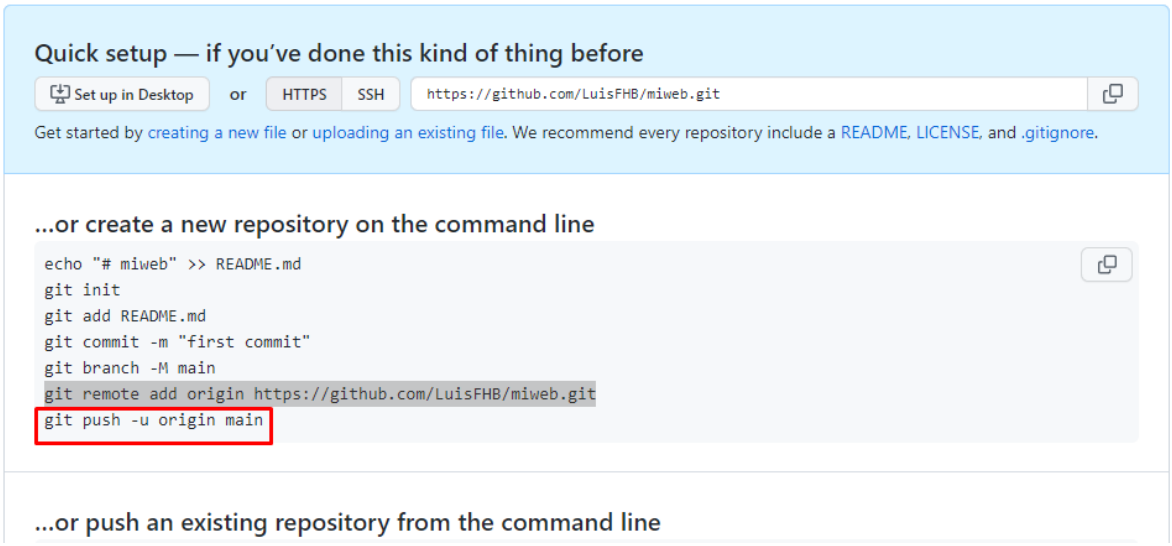
48) GIT: Cambiar nombre a rama principal o tronco

Si observamos en la página de Git Hub utilizan el nombre main para la rama principal

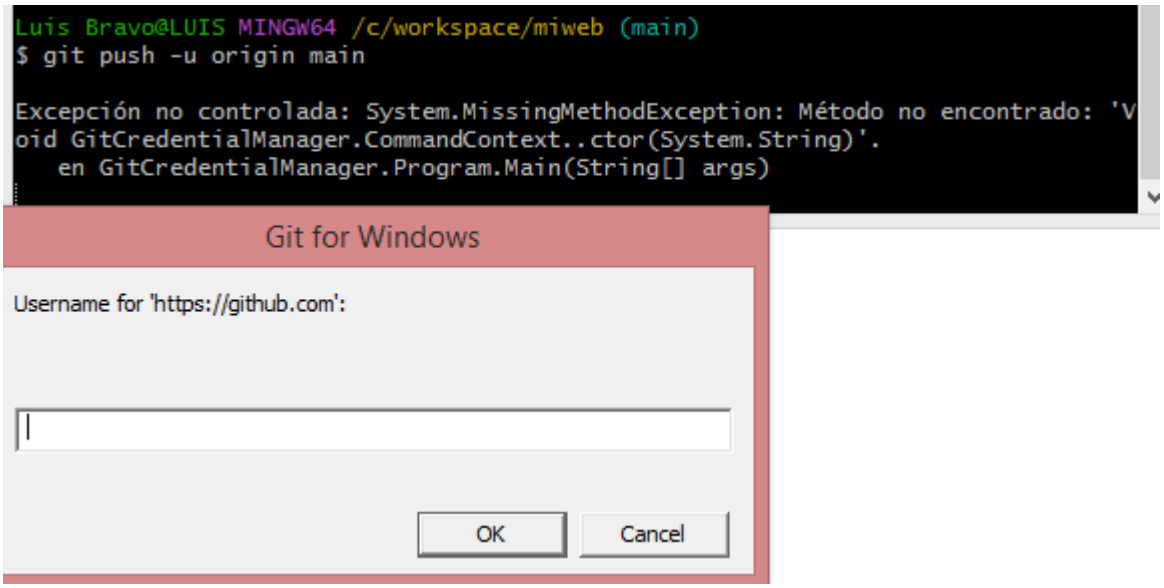
Código
$ git branch -m nombreActual nombreNuevo
Resultado
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master) $ git branch -m master main Luis Bravo@LUIS MINGW64 /c/workspace/miweb (main)

49) GIT : subir cambios

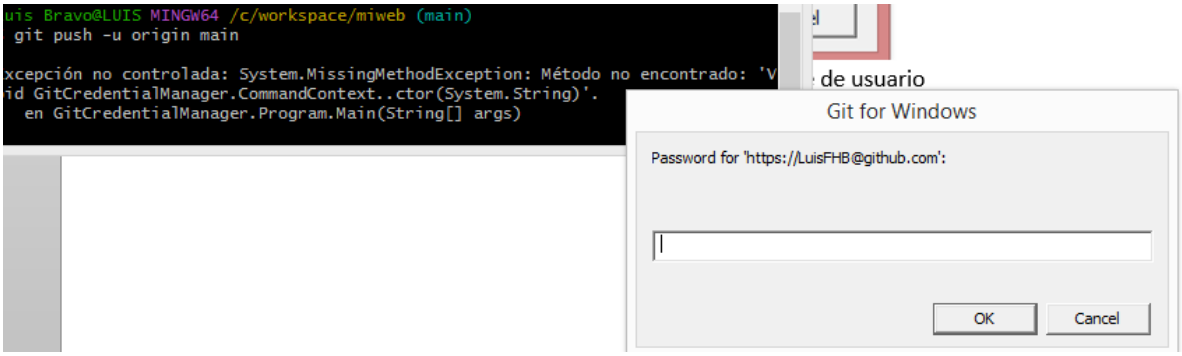
Ahora regresamos al navegador y copiamos la siguiente línea



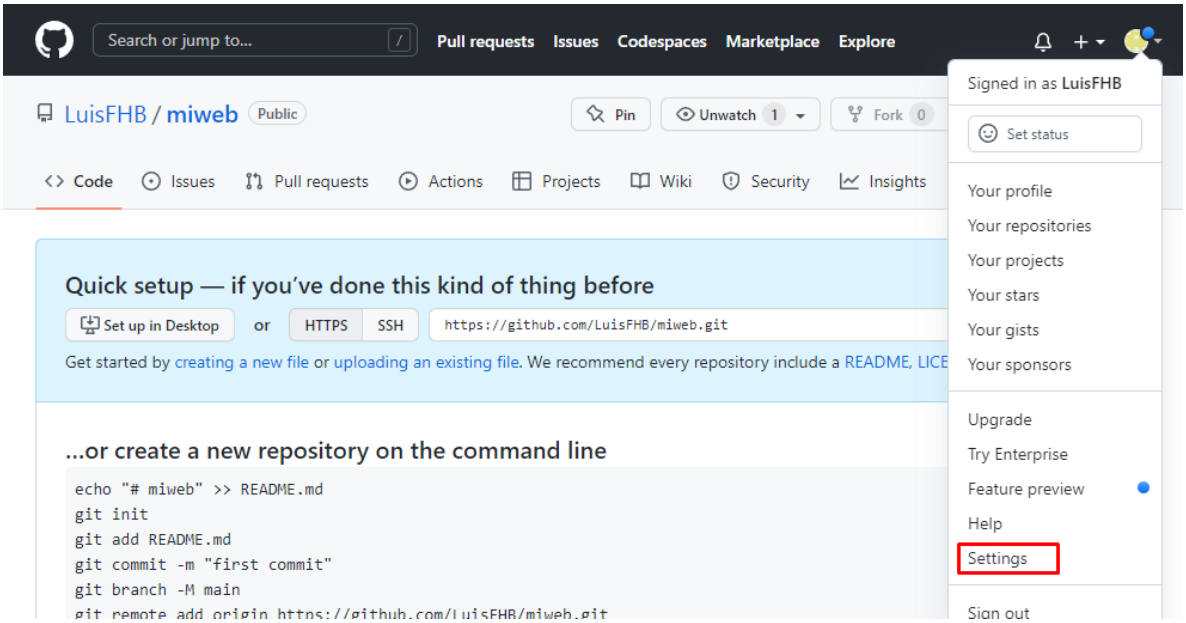
Lo pegamos en GitBash



Nos saldrá esta pantalla donde nos pide nuestro nombre de usuario que dimos de alta en la página

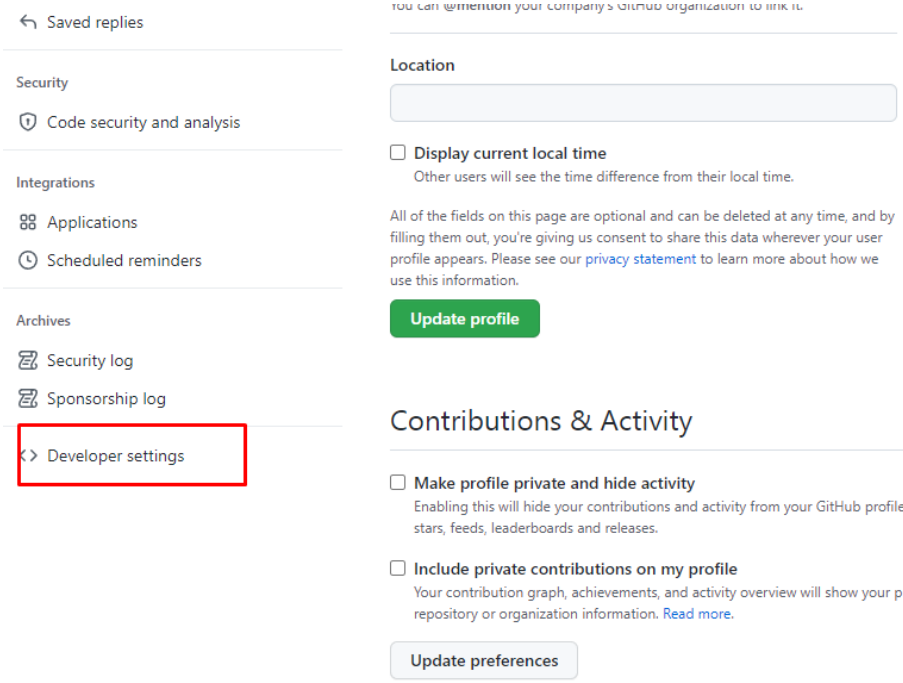


Ahora la contraseña pero esta no es la que usamos para crear la cuenta, la obtenemos así: vamos al explorador

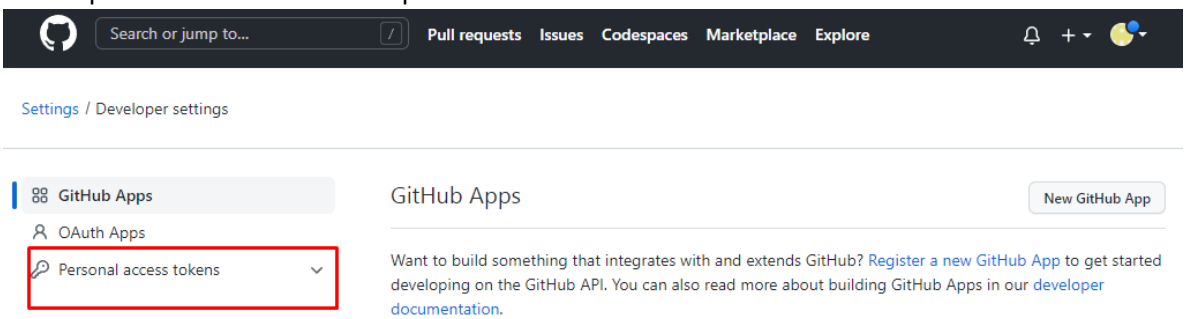


Abrimos setting en otra pestaña

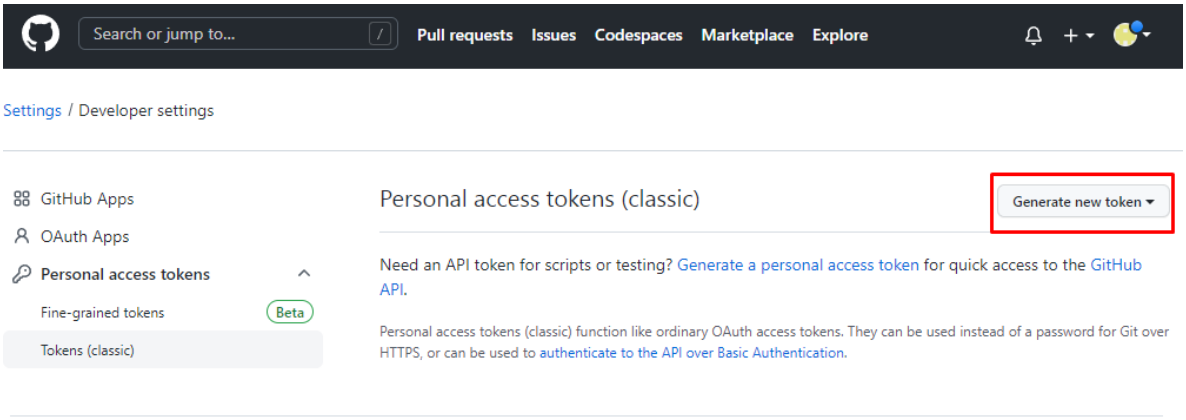
Clic en esa opción de developer settings



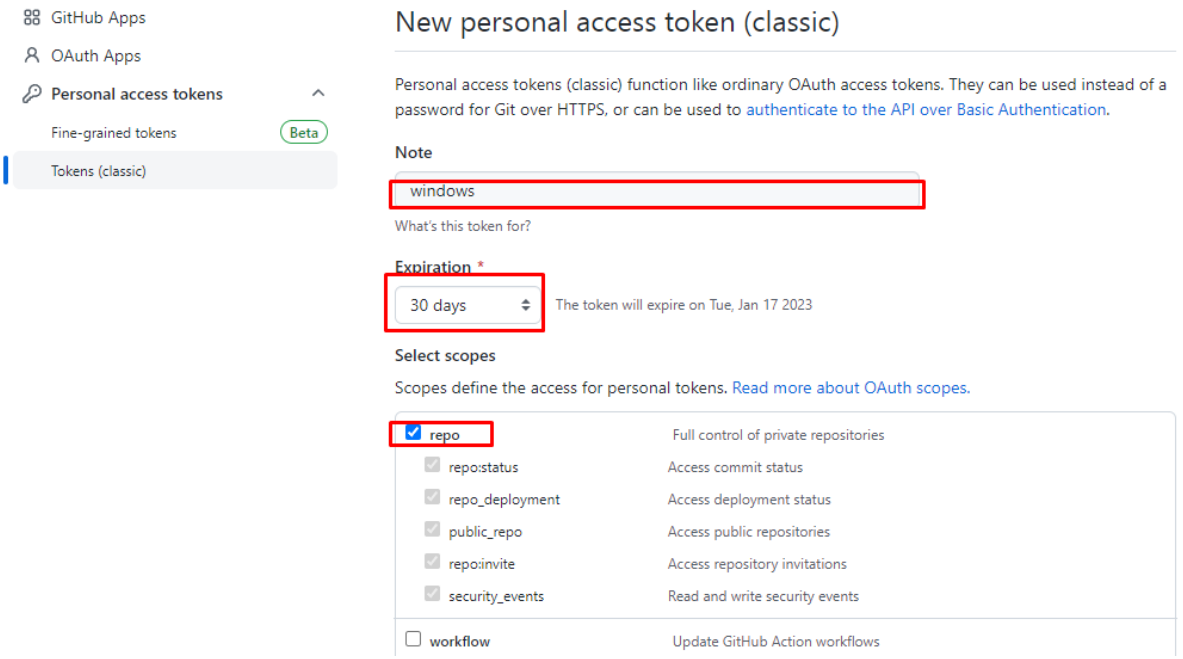
Clic en personal Access tokens depues en tokens classic



Clic en Generate new token

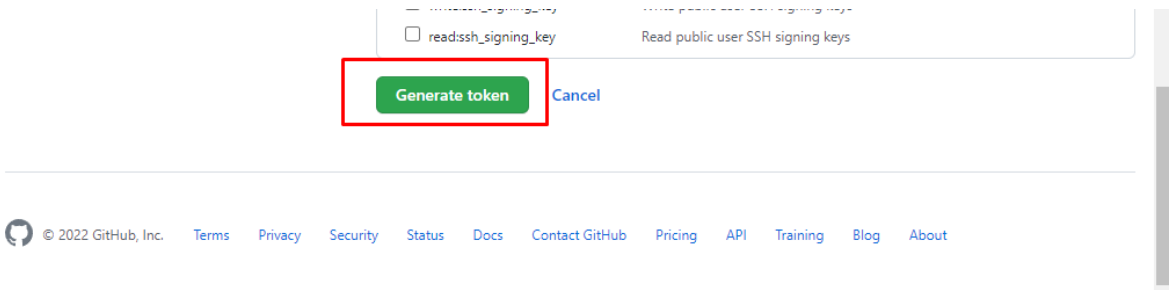


Le ponemos nombre, el tiempo de expiración y seleccionamos todas las opciones del repositorio

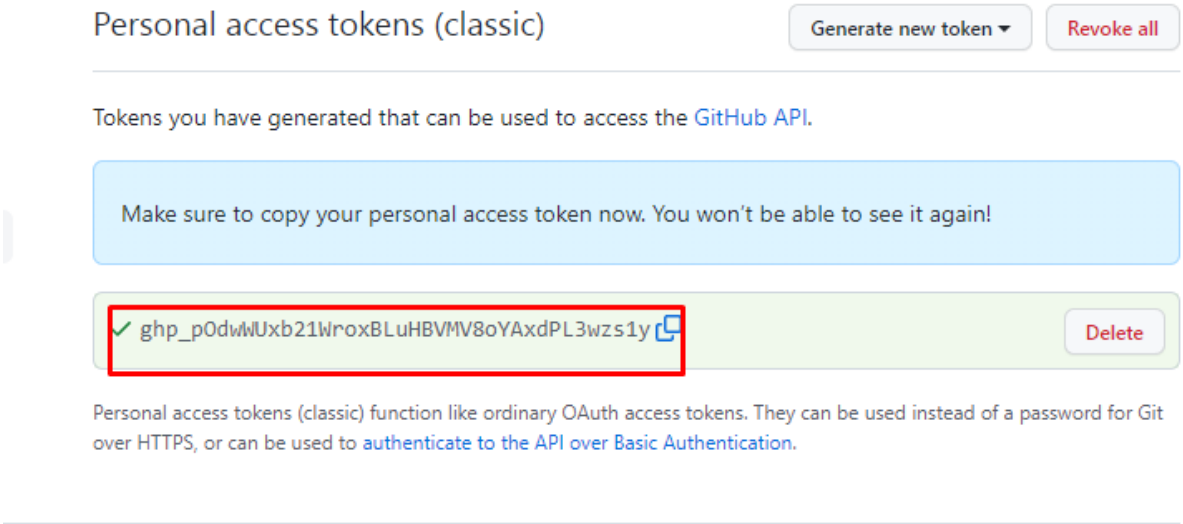


Nos vamos hasta abajo y damos en generate token

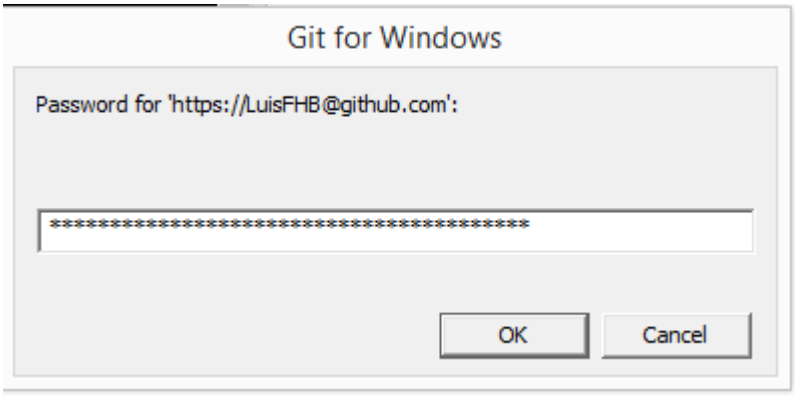
co



Copiamos la contraseña



Y la pegamos en la ventana de Git Bash y damos ok

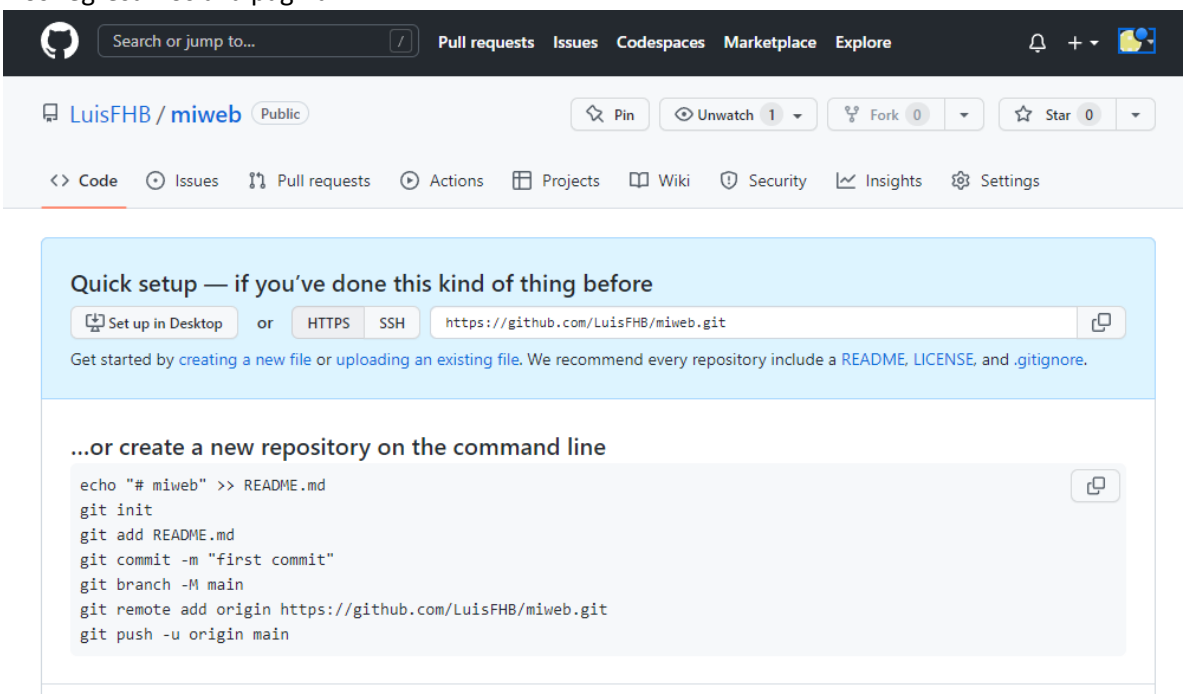


Y listo

```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (main)
$ git push -u origin main

Excepción no controlada: System.MissingMethodException: Método no encontrado: 'V
oid GitCredentialManager.CommandContext..ctor(System.String)'.
   en GitCredentialManager.Program.Main(String[] args)

Excepción no controlada: System.MissingMethodException: Método no encontrado: 'V
oid GitCredentialManager.CommandContext..ctor(System.String)'.
   en GitCredentialManager.Program.Main(String[] args)
Enumerating objects: 37, done.
Counting objects: 100% (37/37), done.
Delta compression using up to 4 threads
Compressing objects: 100% (24/24), done.
Writing objects: 100% (37/37), 3.07 KiB | 285.00 KiB/s, done.
Total 37 (delta 7), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (7/7), done.
To https://github.com/LuisFHB/miweb.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (main)
$
```

50) GIT: revisar cambios

Nos regresamos a la pagina



Y refrescamos o F5, nos saldrán los archivos que creamos