

MANUAL GIT y GitHub



Contenido

1)	GIT: ¿Qué es?	4
2)	GIT: Usos.....	4
3)	GIT: revisar versión.....	4
4)	GIT: configuración de usuario	4
5)	GIT: configuración de email	4
6)	GIT: enlazando a VS CODE.....	5
7)	GIT: (core.auto.crlf) configurar salto de línea	6
8)	GIT: (git config -h) revisar todas las configuraciones	7
9)	GIT: (ls) navegar listar todos los archivos de la carpeta.....	7
10)	GIT: (pwd) navegar ver en qué carpeta me encuentro	8
11)	GIT: (clear) limpiar.....	8
12)	GIT: (cd) navegar entrar a carpetas.....	8
13)	GIT: (cd ..) navegar salir de carpeta.....	8
14)	GIT: (mkdir) crear una carpeta	9
15)	GIT: Ejercicio crear y navegar entre carpetas	9
16)	GIT: (git init) inicializar.....	10
17)	GIT: (ls -a) mostrar elementos ocultos	10
18)	GIT: la carpeta. Git.....	11
19)	GIT: flujo de trabajo	11
20)	GIT: (code .)Abrir editor de texto	11
21)	GIT: creando archivo	12
22)	GIT: (git status) Estado de los archivos	13
23)	GIT: (git add) seleccionar archivos a seguir (agregar)	13
24)	GIT: Ejercicio crea archivo y agrégalo (add)	14
25)	GIT: (git add) agregar mas de 2 archivos a la vez	15
26)	Git (cat) revisar contenido de un archivo.....	15
27)	GIT: ¿Qué pasa si modifico un archivo ya agregado?	15
28)	GIT: (git commit -m "text") comprometer un archivo.....	16
29)	GIT: ¿Qué pasa si modifico un archivo ya comprometido?	16
30)	GIT: (git commit) comprometer sin colocar un texto.....	17

31)	GIT: (rm) eliminar archivo	19
32)	GIT: (git rm) eliminar arhivo de la fase stage en un solo paso	20
33)	GIT: (git restored) quitar cambio de etapa de stage	21
34)	Git: (git restore) descartar cambios	21
35)	GIT: (mv) mover/ cambiar nombre archivo	22
36)	GIT: (git mv) mover/ cambiar nombre archivo y poner en fase stage	23
37)	GIT: ignorar archivos	24
38)	Git: (git status -s) mejor git status	25
39)	GIT: (git diff) ver cambios realizados desde GitBash.....	27
40)	Git (git diff --staged) ver cambios en fase stage.....	30
41)	GIT: (git log) ver historial de cambios.....	32
42)	GIT: (git log --oneline) ver historial de cambios	33
43)	Git: Branch (ramas) y merge (uniones)	34
44)	Git: (git branch) revisar en que rama estamos.....	34
45)	Git: (git checkout -b) crear rama	35
46)	Git: (git merge) unir rama.....	37
47)	Enlazar con GitHub.....	37
48)	GIT: Cambiar nombre a rama principal o tronco	39
49)	GIT : subir cambios	39
50)	GIT: revisar cambios.....	44

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 "Nombre apellido "

5) GIT: configuración de email

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

6) GIT: enlazando a VS CODE

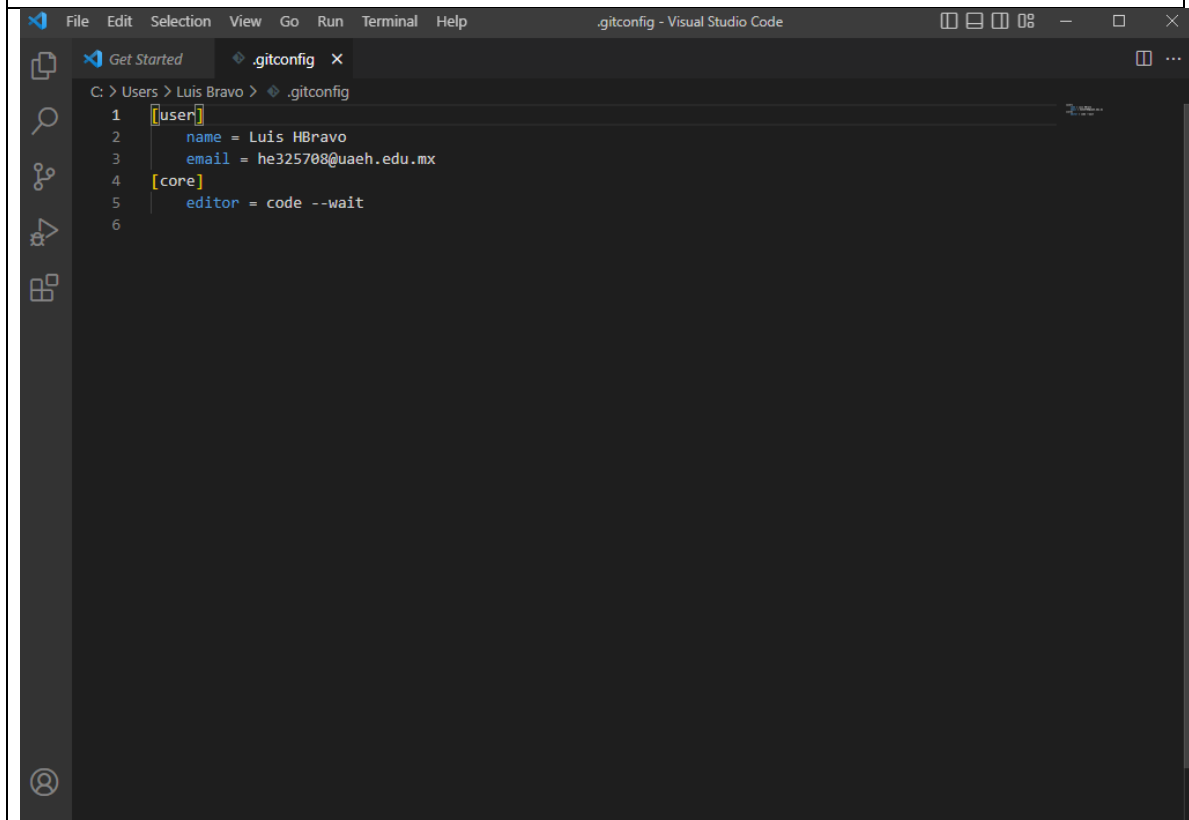
Debes tener instalado VS CODE

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
```

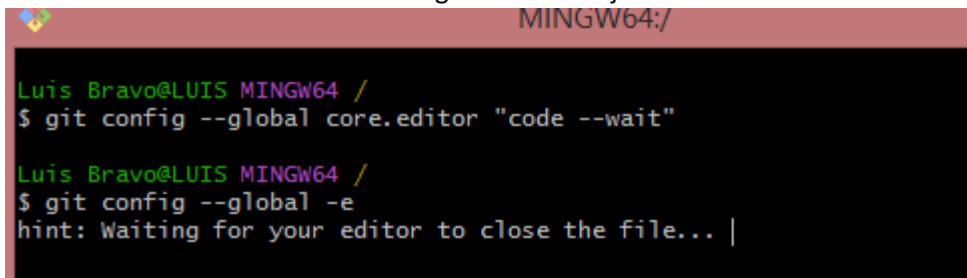
Resultado



The screenshot shows the Visual Studio Code interface with the .gitconfig file open. The file content is as follows:

```
1 [user]  
2   name = Luis HBravo  
3   email = he325708@uaeh.edu.mx  
4 [core]  
5   editor = code --wait  
6
```

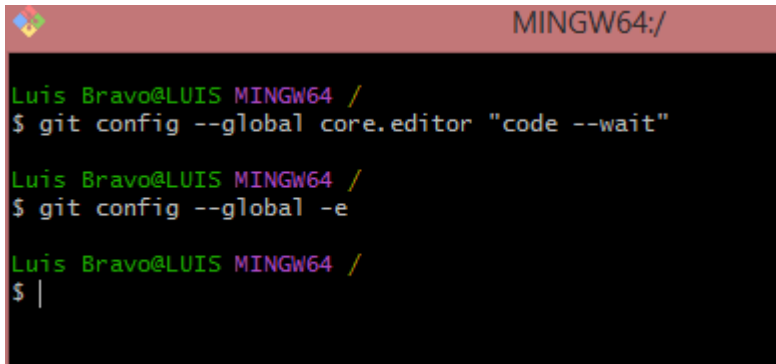
En automático nos abre VS Studio, ya con el nombre y el email que dimos de alta, mientras no cerremos el editor Git mostrara el siguiente mensaje



The screenshot shows a terminal window with the following output:

```
MINGW64:/  
  
Luis Bravo@LUI5 MINGW64 /  
$ git config --global core.editor "code --wait"  
  
Luis Bravo@LUI5 MINGW64 /  
$ git config --global -e  
hint: Waiting for your editor to close the file... |
```

Una vez que cerremos el editor



```
MINGW64:/

Luis Bravo@LUIS MINGW64 /
$ git config --global core.editor "code --wait"

Luis Bravo@LUIS MINGW64 /
$ git config --global -e

Luis Bravo@LUIS MINGW64 /
$ |
```

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	
<pre>\$ 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-pattern] --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-pattern' -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, command line) --show-scope show scope of config (worktree, local, global, system, command) --default <value> with --get, use default value when missing entry</pre>	

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

Código	
\$ ls	
Resultado	
<pre>Luis Bravo@LUI5 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@LUI5 MINGW64 /</pre>	

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

Código
\$ pwd
Resultado
<pre>Luis Bravo@LUIS MINGW64 ~/Desktop \$ pwd /c/Users/Luis Bravo/Desktop</pre>

11) GIT: (clear) limpiar

Código
\$ clear
Resultado
<pre>Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master) \$ </pre>

12) GIT: (cd) navegar entrar a carpetas

Código
\$ cd nombreCarpeta
Resultado
<pre>Luis Bravo@LUIS MINGW64 ~/Desktop \$ cd curs Luis Bravo@LUIS MINGW64 ~/Desktop/curs</pre>

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
<pre>\$ cd .. Luis Bravo@LUIS MINGW64 ~/Desktop \$ </pre>

14) GIT: (mkdir) crear una carpeta

Código
mkdir nombreCarpetaNueva ls -- opcional: para revisar si se creo
Resultado
<pre>Luis Bravo@LUIS MINGW64 ~/Desktop/curs \$ mkdir miweb Luis Bravo@LUIS MINGW64 ~/Desktop/curs \$ ls miweb/</pre>

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
<pre>\$ mkdir workspace \$ cd workspace/ \$ mkdir miweb \$ cd miweb/</pre>
Resultado
<pre>Luis Bravo@LUIS MINGW64 ~/Desktop \$ mkdir workspace Luis Bravo@LUIS MINGW64 ~/Desktop \$ cd workspace/ Luis Bravo@LUIS MINGW64 ~/Desktop/workspace \$ mkdir miweb Luis Bravo@LUIS MINGW64 ~/Desktop/workspace \$ cd miweb/ Luis Bravo@LUIS MINGW64 ~/Desktop/workspace/miweb</pre>

Recuerda por cada línea de código hay que dar ENTER

16) GIT: (git init) inicializar

Código
\$ git init
Resultado
<pre>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) \$</pre>

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

Disco local (C:) > workspace > miweb >				
	Nombre	Fecha de modifica...	Tipo	Tamaño
	.git	18/12/2022 02:31 ...	Carpeta de archivos	

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

Código
\$ ls -a
Resultado
<pre>Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master) \$ ls -a ./ ../ .git/ Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master) \$ </pre>

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

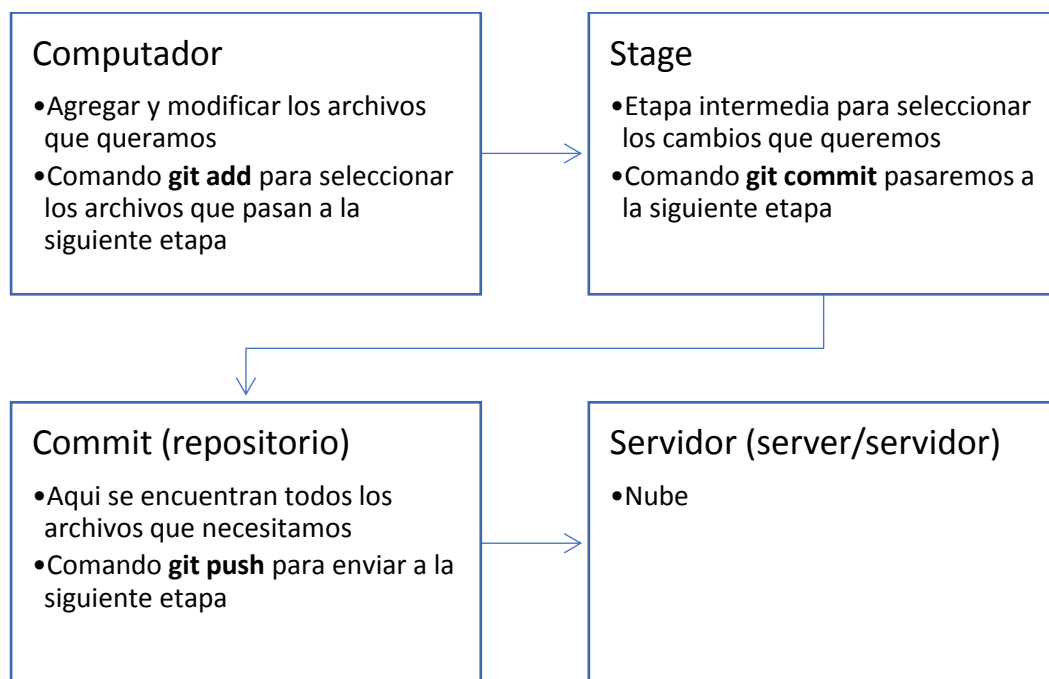
Código
\$ cd .git
Resultado
<pre>Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master) \$ cd .git Luis Bravo@LUIS MINGW64 /c/workspace/miweb/.git (GIT_DIR!) \$ </pre>

18) GIT: la carpeta. Git

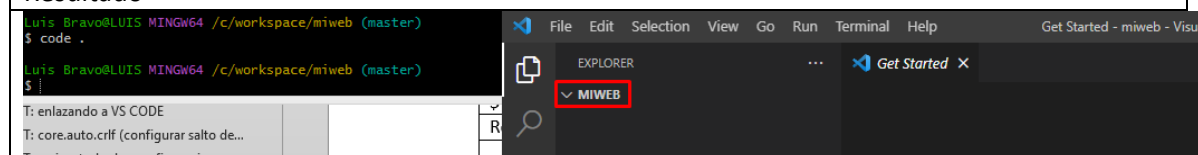
Podemos ver el contenido aplicando el mismo código del punto anterior

Código
\$ ls -a
Resultado
<pre>Luis Bravo@LUIS MINGW64 /c/workspace/miweb/.git (GIT_DIR!) \$ ls -a ./ ../ HEAD config description hooks/ info/ objects/ refs/</pre>

19) GIT: flujo de trabajo



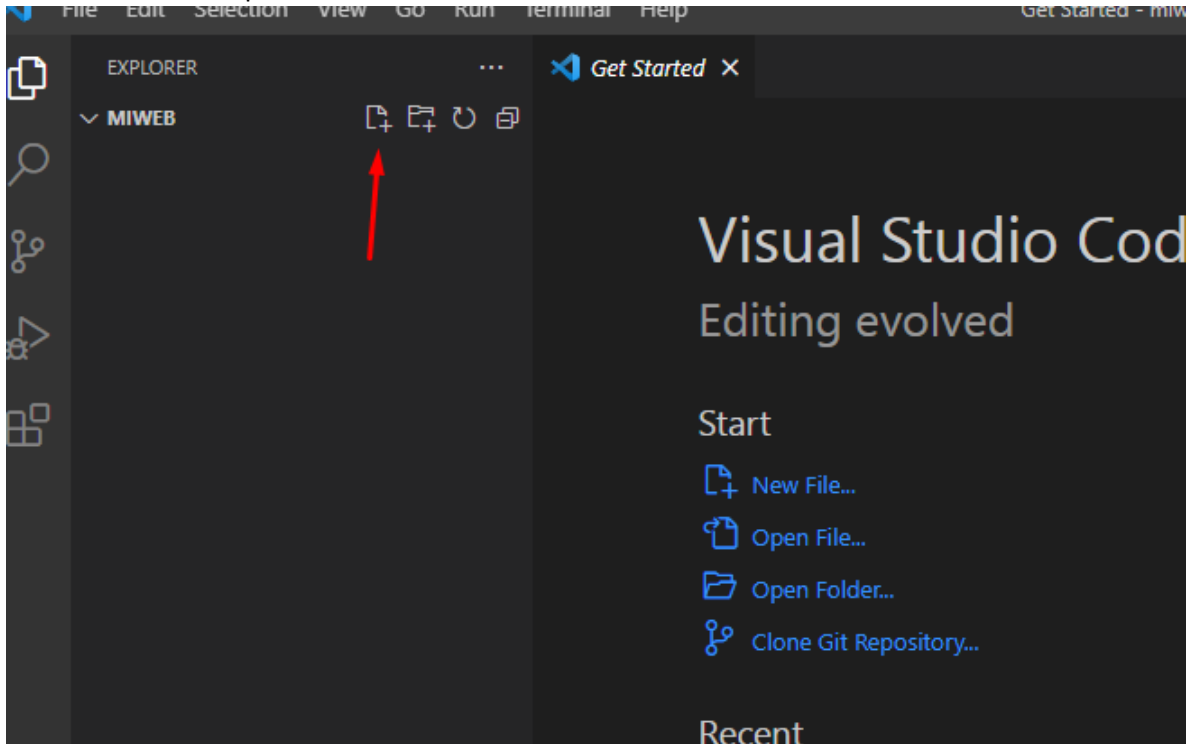
20) GIT: (code .)Abrir editor de texto

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
<pre>Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master) \$ git status On branch master No commits yet Untracked files: (use "git add <file>..." to include in what will be committed) archivo1.txt nothing added to commit but untracked files present (use "git add" to track) Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master) \$ </pre>

Nos dice que no está siguiendo los archivos

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

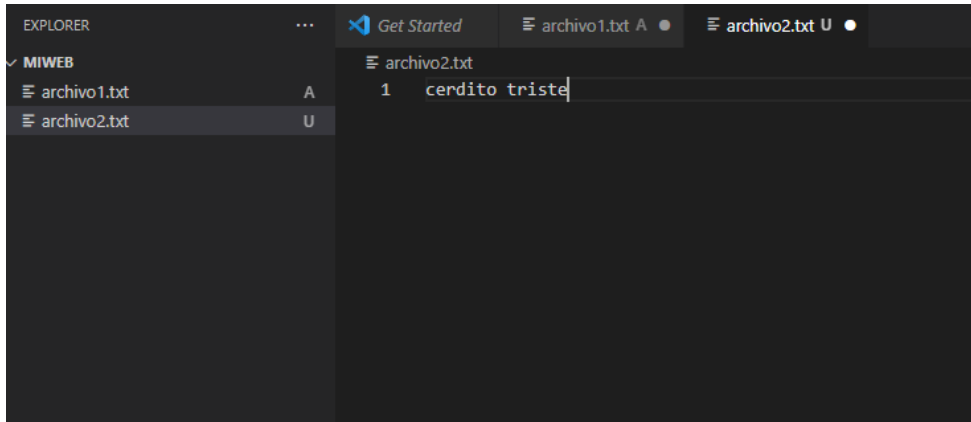
Cuando nosotros usamos git add lo que hacemos es agregar los cambios en el espacio de trabajo, no agregar los archivos, esto se entenderá en temas más adelante a partir de: [GIT: ¿Qué pasa si modifico un archivo ya agregado?](#)

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
<pre>Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master) \$ git add archivo1.txt Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master) \$ git status On branch master No commits yet Changes to be committed: (use "git rm --cached <file>..." to unstage) new file: archivo1.txt Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master) \$ </pre>
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 credito triste, además agrégalo y revisa en status



Usamos status para ver que no está agregada

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

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   archivo1.txt

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

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

Agregamos y usamos status

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

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ status
bash: status: command not found

Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   archivo1.txt
        new file:   archivo2.txt

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

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

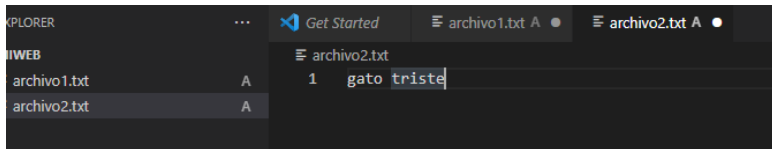
26) Git (cat) revisar contenido de un archivo

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

```
Luis Bravo@LUI5 MINGW64 /c/workspace/miweb (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   archivo1.txt
        new file:   archivo2.txt

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

Luis Bravo@LUI5 MINGW64 /c/workspace/miweb (master)
```

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

```
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

Luis Bravo@LUI5 MINGW64 /c/workspace/miweb (master)
$ git add archivo2.txt

Luis Bravo@LUI5 MINGW64 /c/workspace/miweb (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   archivo1.txt
        new file:   archivo2.txt

Luis Bravo@LUI5 MINGW64 /c/workspace/miweb (master)
$
```

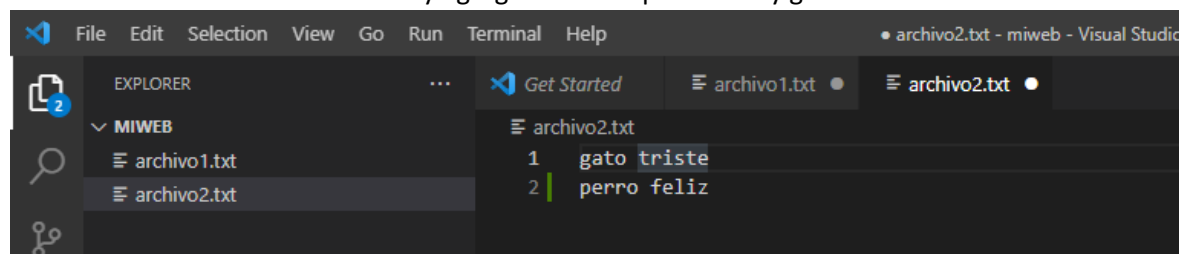
28) GIT: (git commit -m "text") comprometer un archivo

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
<pre>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)</pre>

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.

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

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

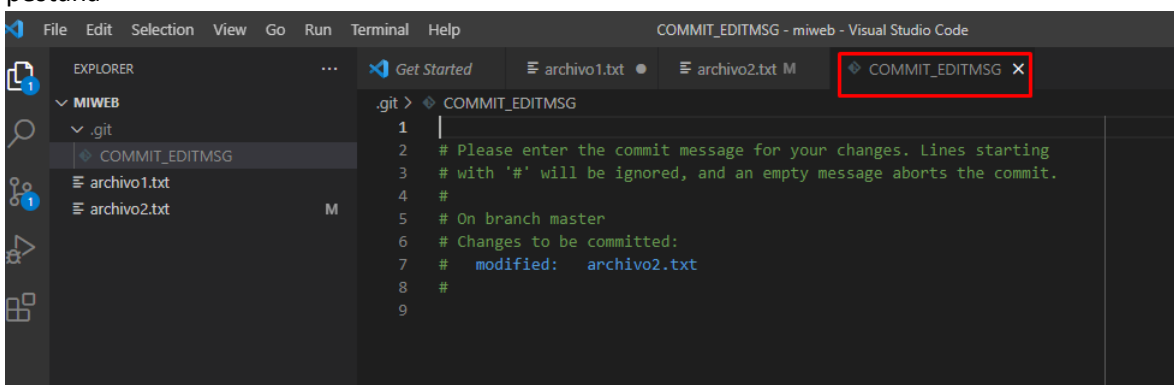
Ahora nos dice que hay cambios listos para comprometerse

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

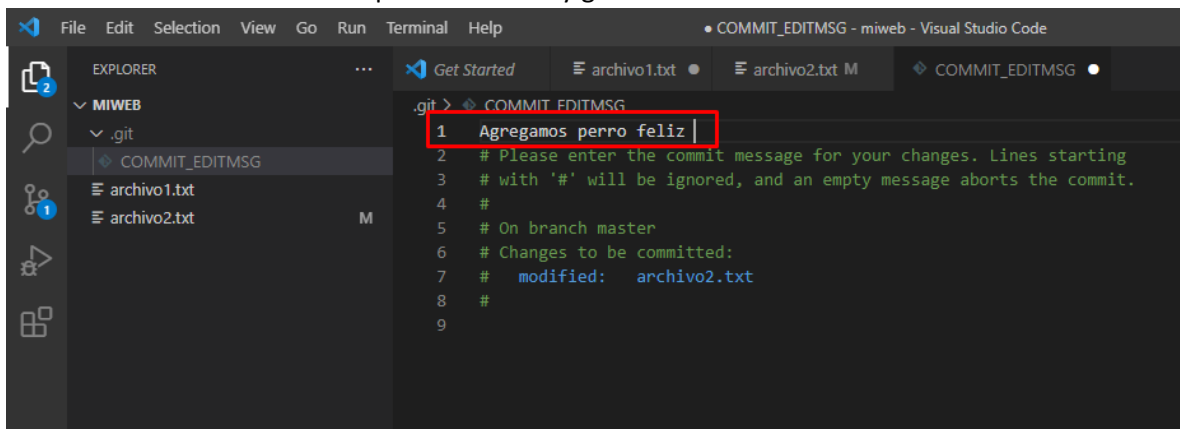
Con el ejemplo anterior

Código
\$ git commit <u>—sin colocar el quion la m, las comillas y el texto</u>
Resultado
<pre>Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master) \$ git commit hint: Waiting for your editor to close the file...</pre>

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



Ahora escribimos los cambios que se hicieron y guardamos

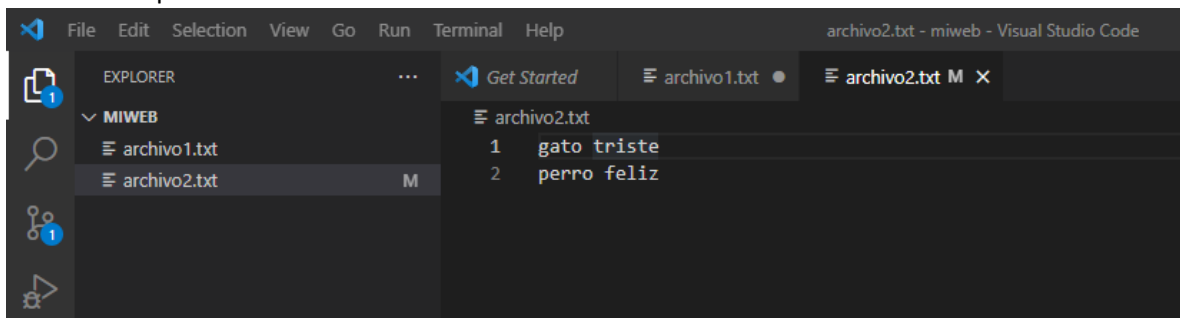


```
File Edit Selection View Go Run Terminal Help
COMMIT_EDITMSG - miweb - Visual Studio Code

EXPLORER
  MIWEB
    .git
      COMMIT_EDITMSG
    archivo1.txt
    archivo2.txt M

1 Agregamos perro feliz |
2 # Please enter the commit message for your changes. Lines starting
3 # with '#' will be ignored, and an empty message aborts the commit.
4 #
5 # On branch master
6 # Changes to be committed:
7 #   modified:   archivo2.txt
8 #
9
```

Cerramos la pestaña

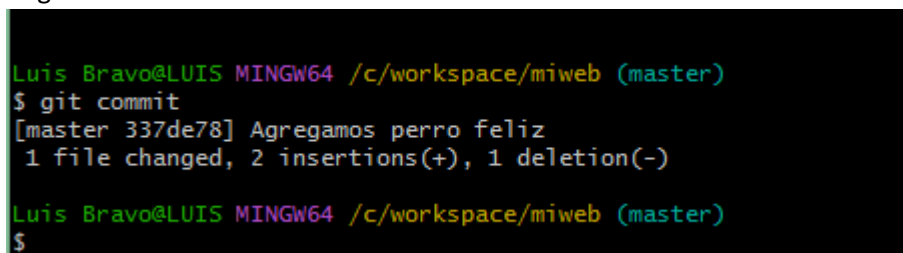


```
File Edit Selection View Go Run Terminal Help
archivo2.txt - miweb - Visual Studio Code

EXPLORER
  MIWEB
    archivo1.txt
    archivo2.txt M

1 gato triste
2 perro feliz
```

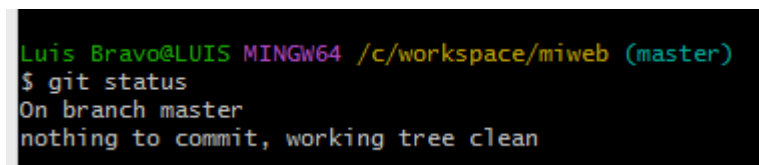
Regresamos a la terminal



```
Luis Bravo@LUI5 MINGW64 /c/workspace/miweb (master)
$ git commit
[master 337de78] Agregamos perro feliz
1 file changed, 2 insertions(+), 1 deletion(-)

Luis Bravo@LUI5 MINGW64 /c/workspace/miweb (master)
$
```

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



```
Luis Bravo@LUI5 MINGW64 /c/workspace/miweb (master)
$ git status
On branch master
nothing to commit, working tree clean
```

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
<pre>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")</pre>

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
<pre>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) \$</pre>

Ahora podemos comprometer dichos cambios

Código
\$ git commit -m "Eliminando archivo 2" --recuerda que el texto depende del nombre del archivo o de lo que hayas borrado
\$ git status
Resultado
<pre>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) \$</pre>

Revisamos cuantos archivos tenemos

Código
\$ ls
Resultado
<pre>Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master) \$ ls archivo1.txt Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master) \$ </pre>

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
<pre>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</pre>

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 código 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
<pre>\$ git restore nombreArchivo.extension \$ git status \$ git ls</pre>
Resultado
<pre>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</pre>

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
<pre>\$ mv nombreActualArchivo.extension nombreNuevoArchivo.extension ls \$ git status</pre>
Resultado
<pre>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")</pre>

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
<pre>\$ git add nombreAntiguoArchivo.extension nombreActualArchivo.extension \$ git status \$ git commit</pre>
Resultado
<pre>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%)</pre>

36) **[GIT: \(git mv\) mover/ cambiar nombre archivo y poner en fase stage](#)**

Regresaremos el nombre del archivo a archivo 1

Código
<pre>\$ git mv nombreActualArchivo.extension nombreNuevoArchivo.extension \$ git status \$ git commit</pre>
Resultado
<pre>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</pre>

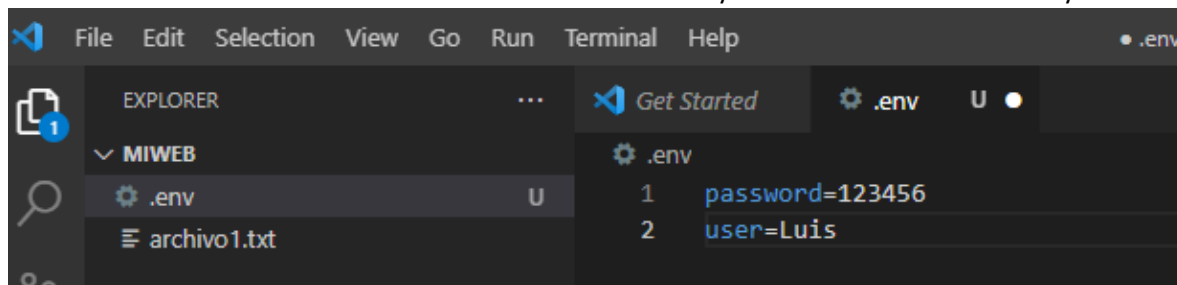
Ahora solo nos falta volver al repositorio o a comprometer el archivo

Código
\$ git commit -m"devolviendo el nombre al archivo"
Resultado
<pre>Luis Bravo@LUI5 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%)</pre>

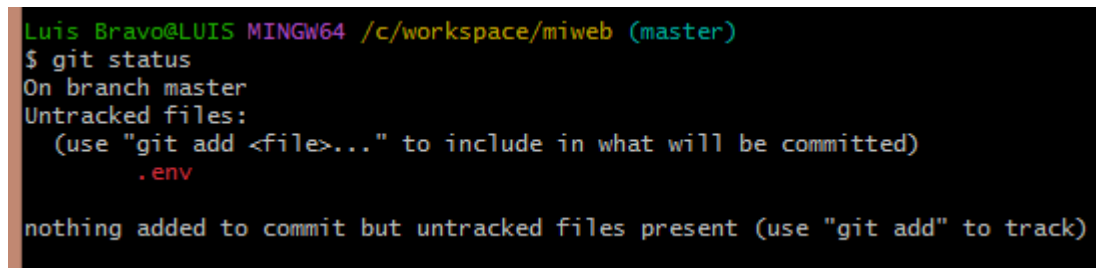
37) GIT: ignorar archivos

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.

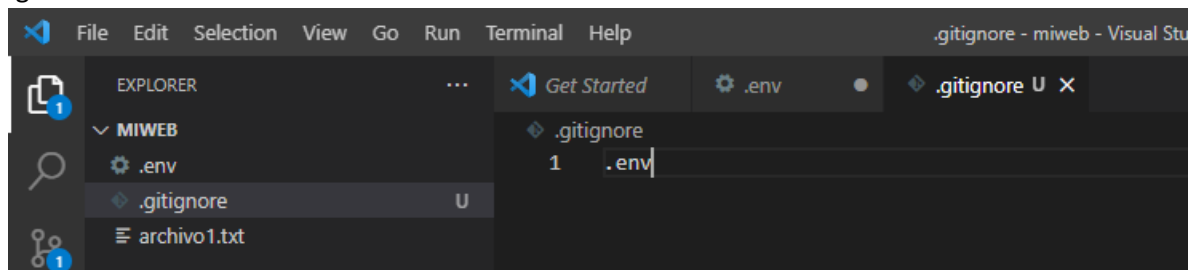
Creamos un nuevo archivo en VS CODE con el nombre .env y escribimos una contraseña y usuario



Revisamos con status



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



Ahora revisamos con estatus

```
Luis Bravo@LUI5 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@LUI5 MINGW64 /c/workspace/miweb (master)
$
```

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

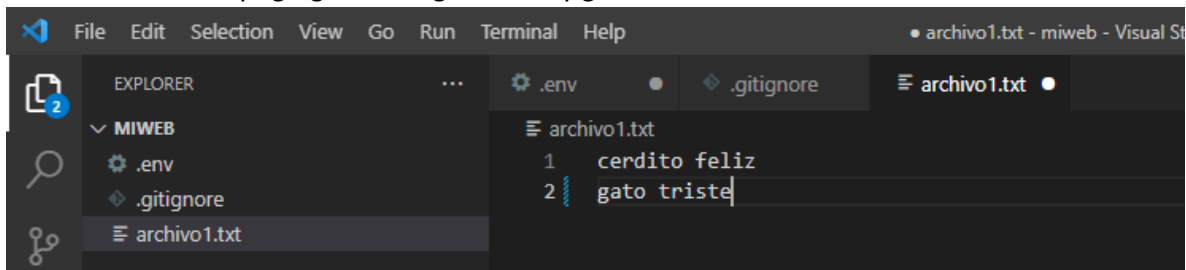
```
Luis Bravo@LUI5 MINGW64 /c/workspace/miweb (master)
$ git add .gitignore

Luis Bravo@LUI5 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

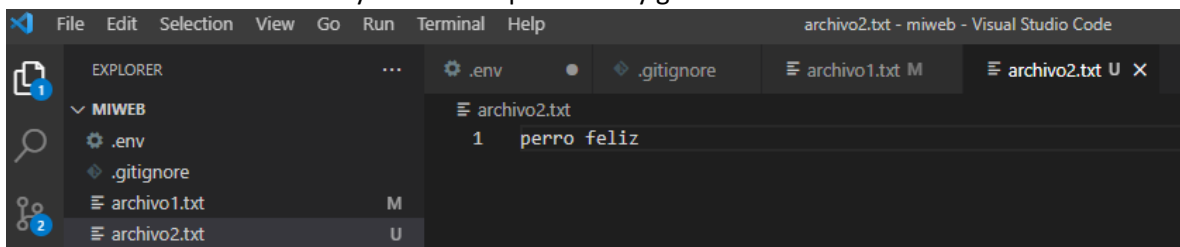
Luis Bravo@LUI5 MINGW64 /c/workspace/miweb (master)
$
```

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 directory)
        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	
<pre>Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master) \$ git status -s M archivo1.txt ?? archivo2.txt</pre>	

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

```
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

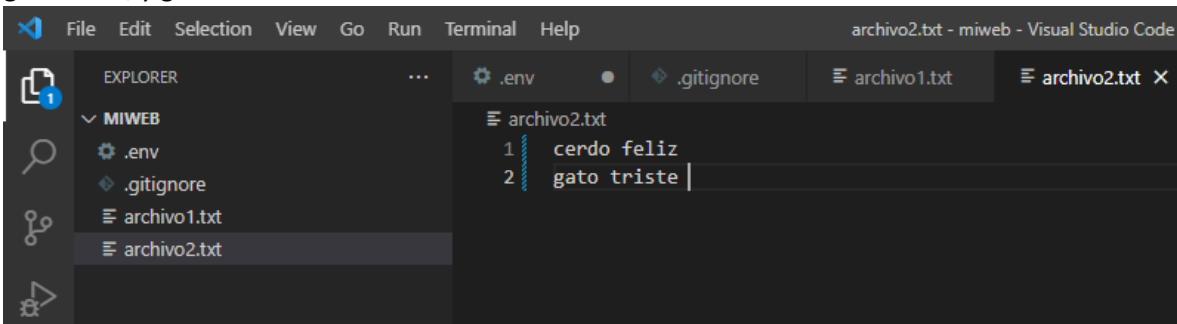
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git status -s

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

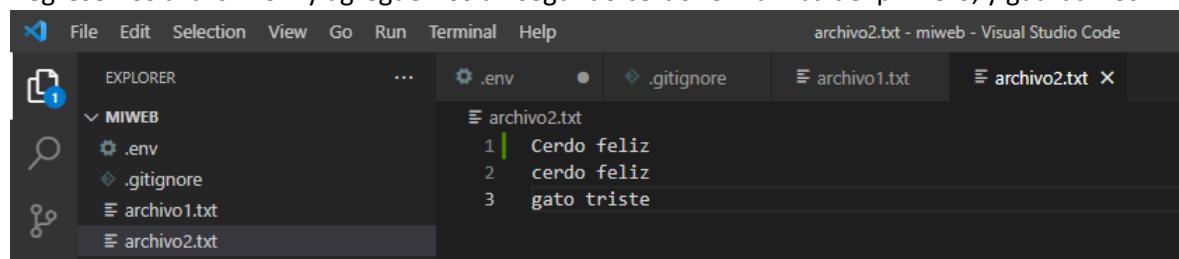
Código
\$ git diff
Resultado
<pre>no changes added to commit (use "git add" and/or "git commit -a") Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master) \$ git diff diff --git a/archivo2.txt b/archivo2.txt index 239e71e..2bf7544 100644 --- a/archivo2.txt +++ b/archivo2.txt @@ -1,2 @@ -perro feliz \ No newline at end of file +cerdo feliz +gato triste \ No newline at end of file</pre>

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 modifiko archivo2"
[master 7738d48] se modifiko 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

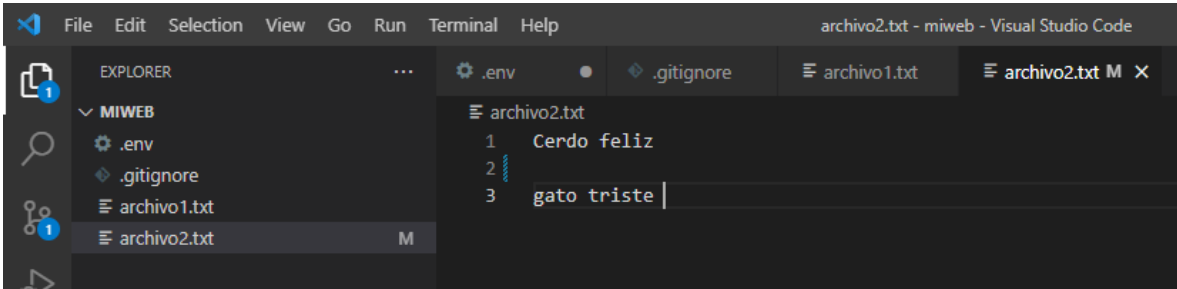
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ |
```

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

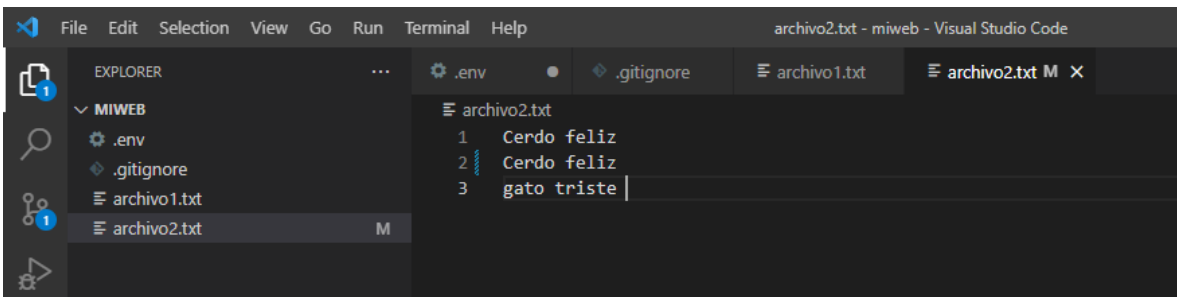
```
Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master)
$ git add archivo2.txt

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(-)
```

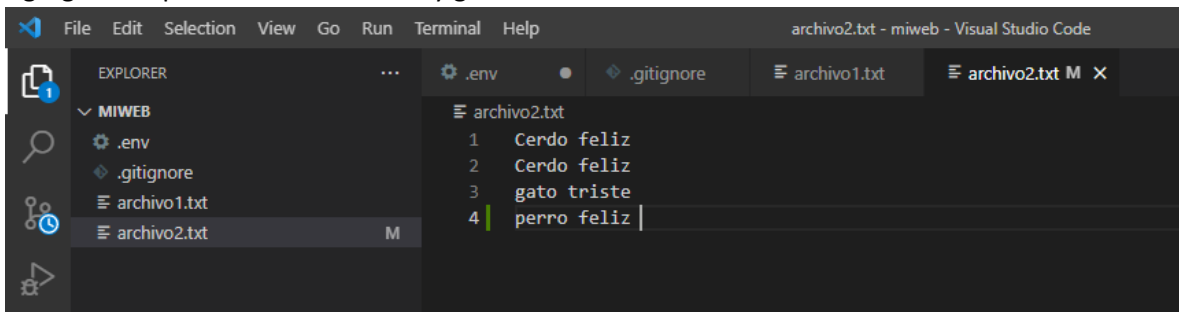
Escribimos una última vez git diff y veamos que no nos sale nada

```
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
<pre>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) \$</pre>

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 letra Q en nuestro teclado

```
Luis Bravo@LUI5 MINGW64 /c/workspace/miweb (master)
$ git log
commit c0fab013ec62ccbcb3a24f638181a95c67bf638 (HEAD -> master)
Author: Luis HBravo <he325708@uaeh.edu.mx>
Date: Sun Dec 18 17:34:44 2022 -0600

    Se modifiko 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 modifiko el archivo 2 otra vez

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

    se modifiko archivo2

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

    mostrando status corto
:...skipping...
commit c0fab013ec62ccbcb3a24f638181a95c67bf638 (HEAD -> master)
Author: Luis HBravo <he325708@uaeh.edu.mx>
Date: Sun Dec 18 17:34:44 2022 -0600

    Se modifiko 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 modifiko el archivo 2 otra vez

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

    se modifiko archivo2
```



```

se modifiko el archivo 2 otra vez

commit 7738d48d77194a86219b790c2bf488e73ba83e
Author: Luis HBravo <he325708@uaeh.edu.mx>
Date: Sun Dec 18 17:27:03 2022 -0600

se modifiko archivo2

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

mostrando status corto

commit 9893203a6846d19c30b9d2e47deaae71fe8b4519
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 modifiko el archivo 2 otra vez 2
03e6e30 se modifiko el archivo 2 otra vez
7738d48 se modifiko 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)
$

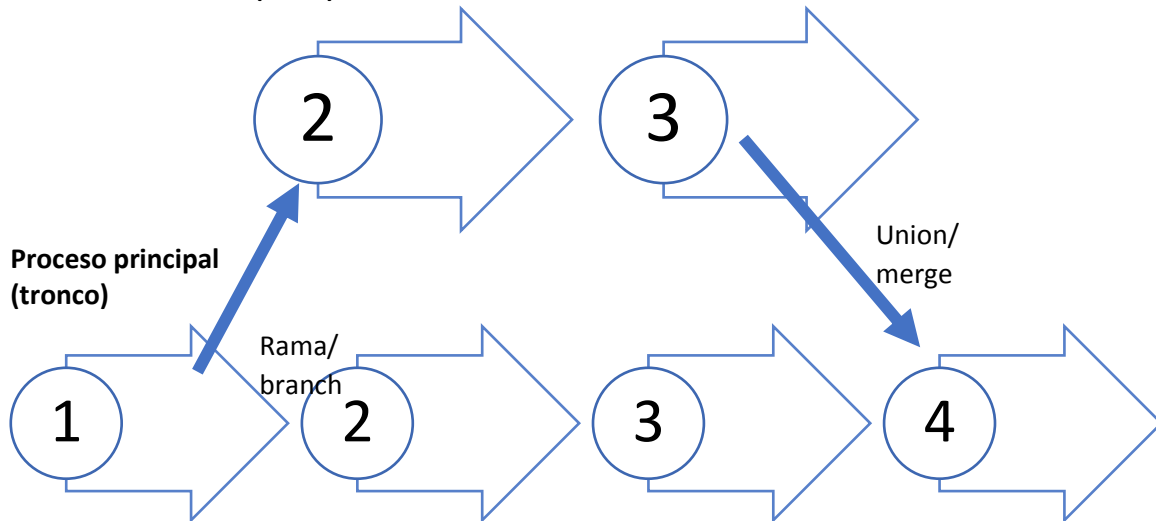
```

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.

Proceso secundario (rama)



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	
<pre>Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master) \$ git branch * master</pre>	

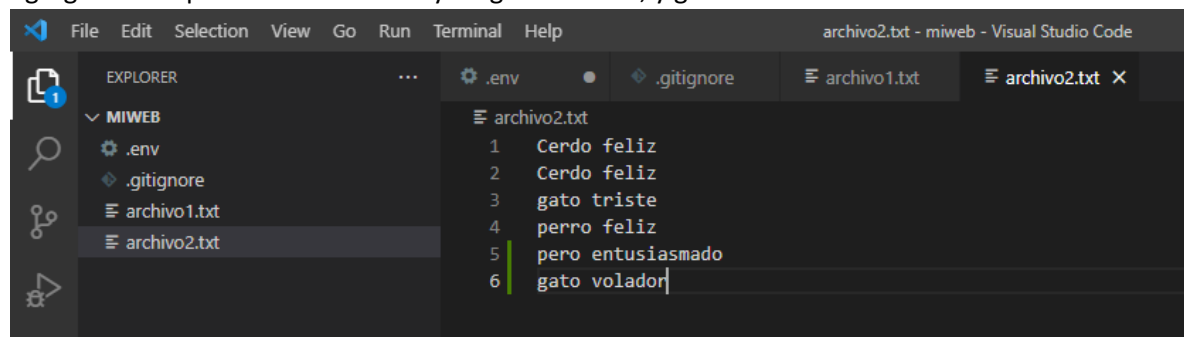
Estamos en la rama principal

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

Ahora crearemos una rama

Código
<pre>\$ git checkout -b nombreRama \$ git branch -- podemos usar branch para revisar que realmente estemos en esa rama creada</pre>
Resultado
<pre>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</pre>

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 modifiko el archivo 2 otra vez 2
03e6e30 se modifiko el archivo 2 otra vez
7738d48 se modifiko 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.

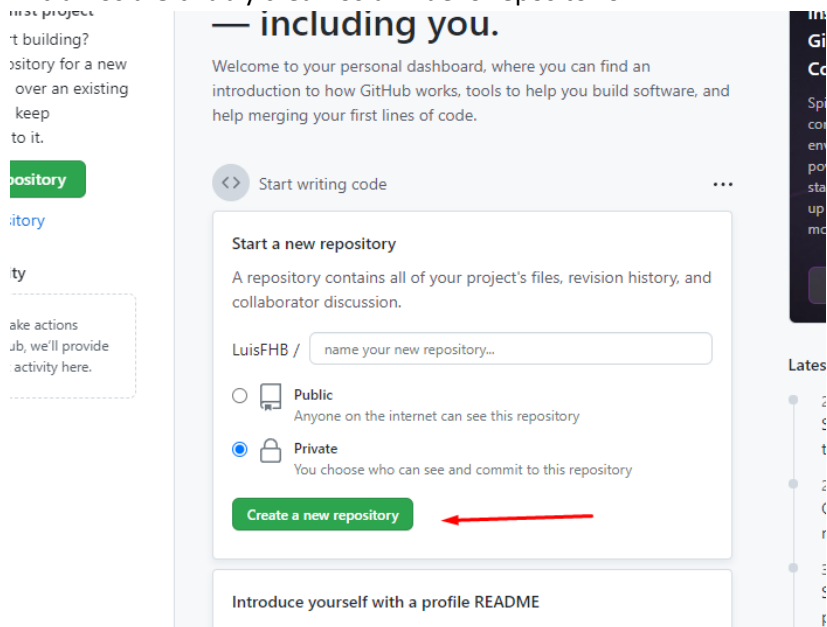
46) Git: (git merge) unir rama

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
<pre>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) \$</pre>

47) Enlazar con GitHub

Entramos a GitHub y creamos un nuevo repositorio




Colocamos nombre, dejamos en público y no movemos nada más

Create a new repository

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

Owner *

 LuisFHB ▾

Repository name *


/ miweb 

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

Description (optional)

☒  **Public**

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

☐  **Private**

You choose who can see and commit to this repository.

Initialize this repository with:



Skip this step if you're importing an existing repository.


☐ **Add a README file**

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


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

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

Quick setup — if you've done this kind of thing before
 Set up in Desktop or **HTTPS** **SSH** 
Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

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

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

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

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

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
<pre>Luis Bravo@LUIS MINGW64 /c/workspace/miweb (master) \$ git branch -m master main Luis Bravo@LUIS MINGW64 /c/workspace/miweb (main)</pre>

49) GIT : subir cambios

Ahora regresamos al navegador y copiamos la siguiente línea

Quick setup — if you've done this kind of thing before

 Set up in Desktop

 or

HTTPS

SSH

https://github.com/LuisFHB/miweb.git 

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

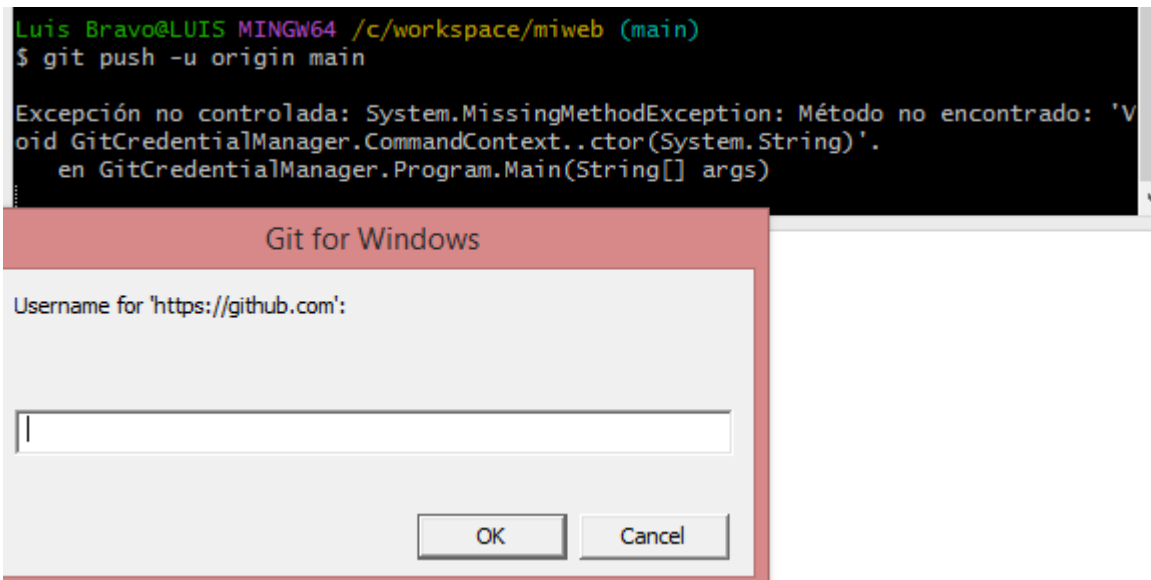
...or create a new repository on the command line

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

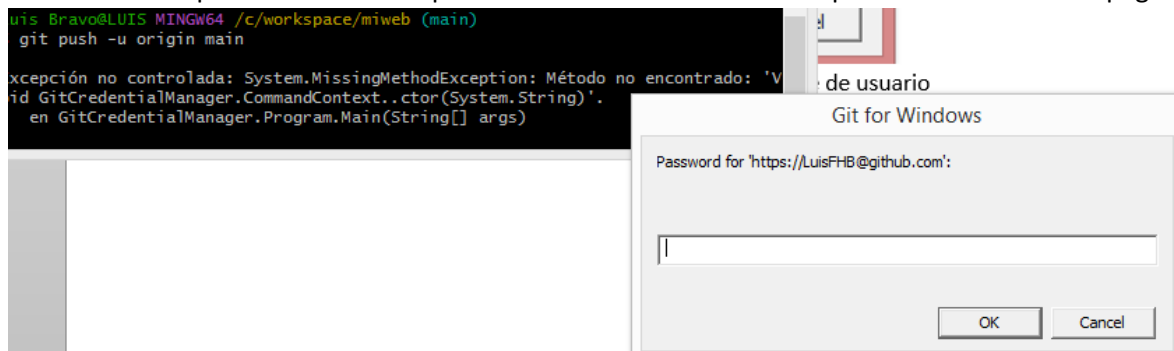


...or push an existing repository from the command line

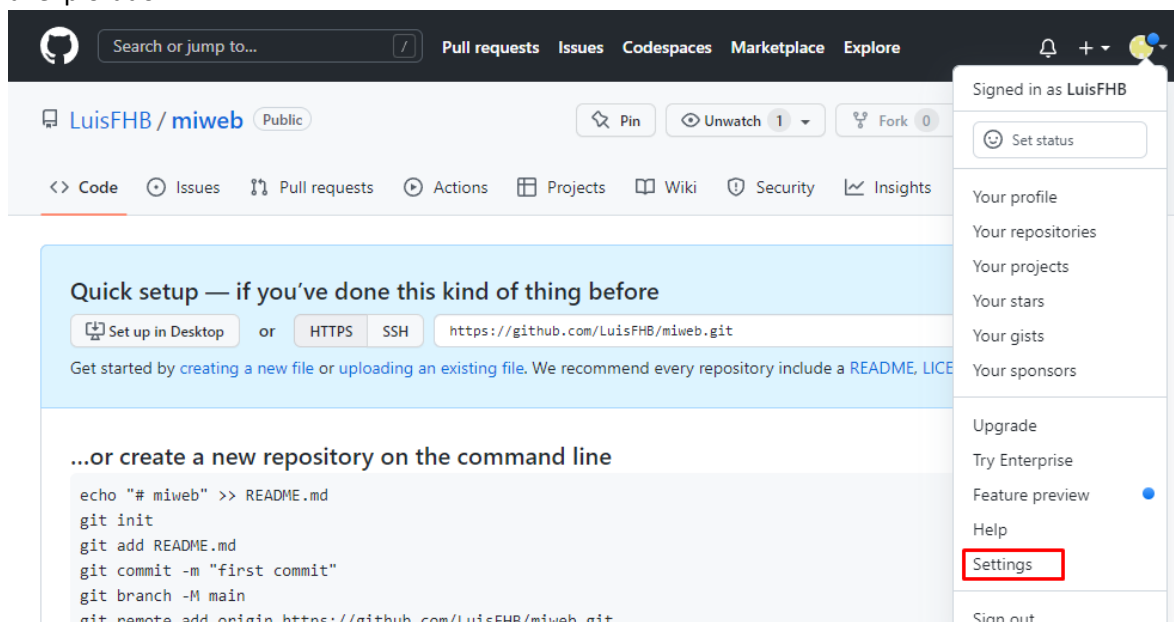
Lo pegamos en GitBash



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

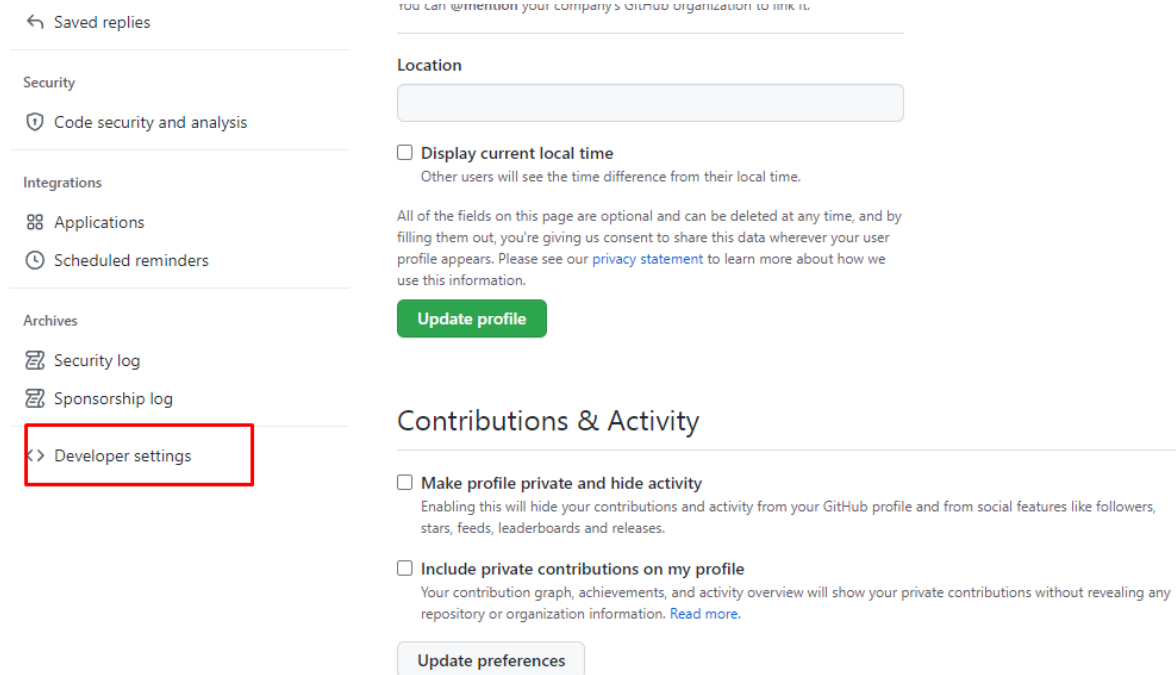


Ahora la contraseña pero esta no es la que usamos para crear la cuenta, la obtenemos así: vamos al explorador



Abrimos setting en otra pestaña

Clic en esa opción de developer settings



The screenshot shows the GitHub Developer Settings page. On the left sidebar, the 'Developer settings' option is highlighted with a red box. The main content area shows the 'Location' field, a checkbox for 'Display current local time', and a green 'Update profile' button. Below this is the 'Contributions & Activity' section with checkboxes for 'Make profile private and hide activity' and 'Include private contributions on my profile', followed by an 'Update preferences' button.

← Saved replies

Security

🔒 Code security and analysis

Integrations

🔗 Applications

🕒 Scheduled reminders

Archives

📄 Security log

📄 Sponsorship log

<> Developer settings

You can @mention your company's GitHub organization to link it.

Location

☐ Display current local time
Other users will see the time difference from their local time.

All of the fields on this page are optional and can be deleted at any time, and by filling them out, you're giving us consent to share this data wherever your user profile appears. Please see our [privacy statement](#) to learn more about how we use this information.

Update profile

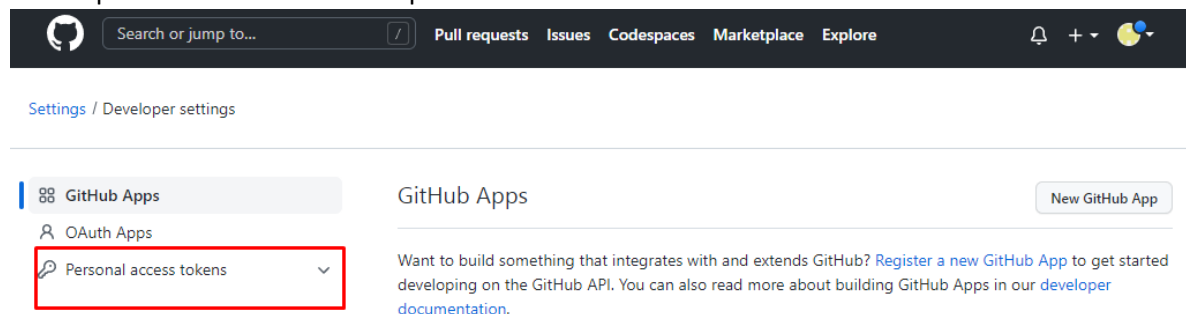
Contributions & Activity

☐ Make profile private and hide activity
Enabling this will hide your contributions and activity from your GitHub profile and from social features like followers, stars, feeds, leaderboards and releases.

☐ Include private contributions on my profile
Your contribution graph, achievements, and activity overview will show your private contributions without revealing any repository or organization information. [Read more.](#)

Update preferences

Clic en personal Access tokens depues en tokens classic



The screenshot shows the GitHub Settings / Developer settings page. The 'Personal access tokens' option in the left sidebar is highlighted with a red box. The main content area shows the 'GitHub Apps' section with a 'New GitHub App' button and a paragraph about building something that integrates with and extends GitHub.

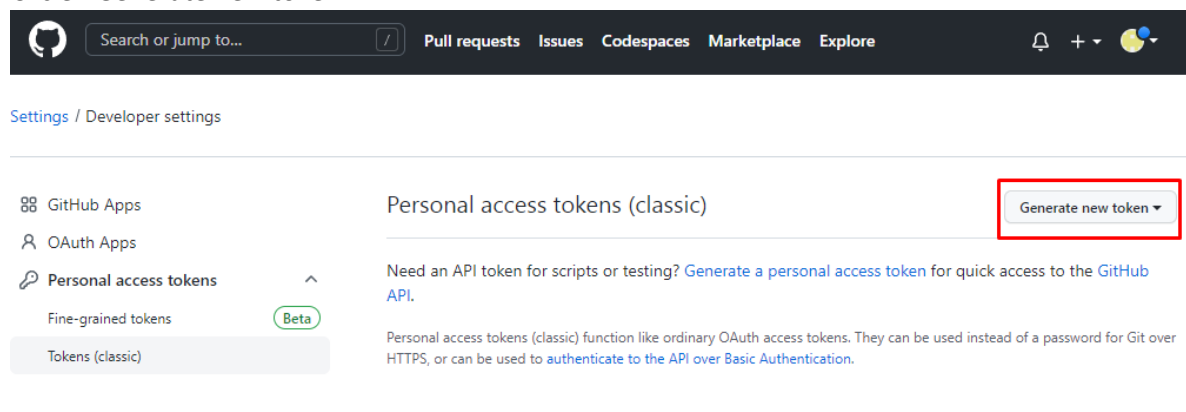
Settings / Developer settings

GitHub Apps

New GitHub App

Want to build something that integrates with and extends GitHub? [Register a new GitHub App](#) to get started developing on the GitHub API. You can also read more about building GitHub Apps in our [developer documentation](#).

Clic en Generate new token



The screenshot shows the GitHub Personal access tokens (classic) page. The 'Generate new token' button in the top right corner is highlighted with a red box. The left sidebar shows the 'Personal access tokens' section with 'Fine-grained tokens' and 'Tokens (classic)' options. The main content area has a heading 'Personal access tokens (classic)' and a paragraph about needing an API token for scripts or testing.

Settings / Developer settings

Personal access tokens (classic)

Generate new token

Need an API token for scripts or testing? [Generate a personal access token](#) for quick access to the [GitHub API](#).

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

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

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

What's this token for?

Expiration *
 The token will expire on Tue, Jan 17 2023

Select scopes
Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo:deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows

☐ read:ssh_signing_key Read public user SSH signing keys

Generate token Cancel

Nos vamos hasta abajo y damos en generate token

CO

Copiamos la contraseña

Personal access tokens (classic) [Generate new token](#) [Revoke all](#)

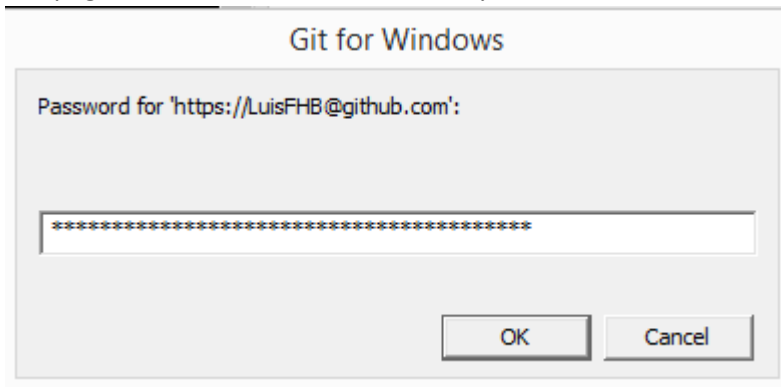
Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

✓ ghp_pOdwUxb21WroxBLuHBVMV8oYAXdPL3wzs1y [Delete](#)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

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

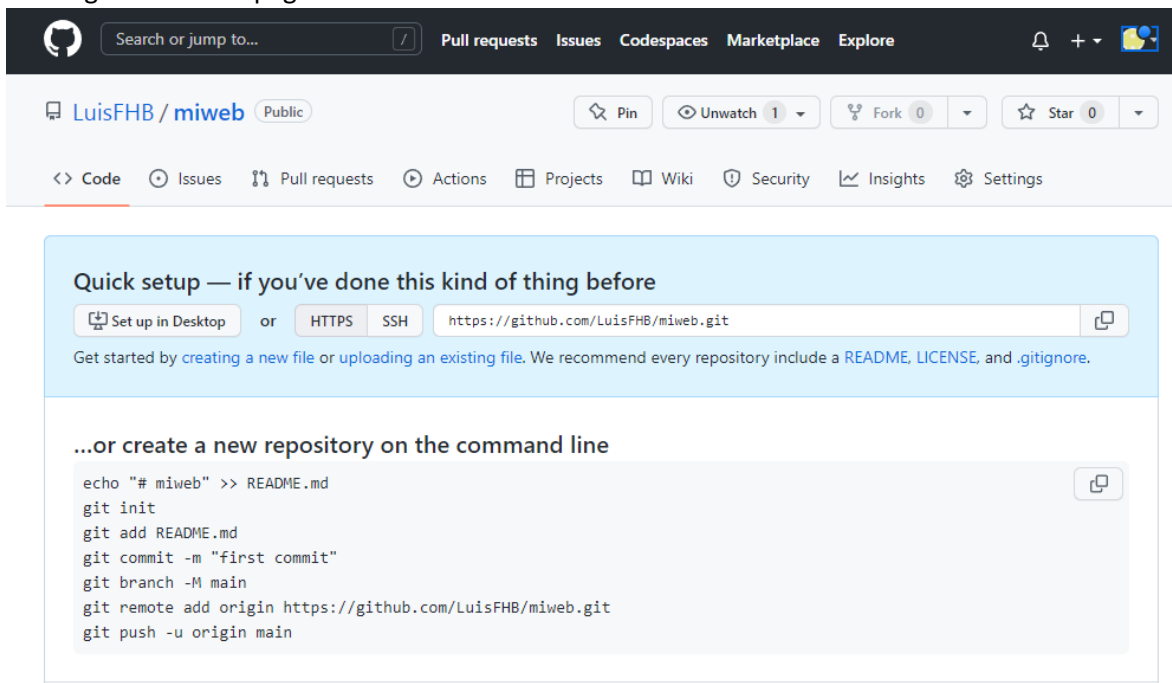
Excepción no controlada: System.MissingMethodException: Método no encontrado: 'Void
GitCredentialManager.CommandContext..ctor(System.String)'.
    en GitCredentialManager.Program.Main(String[] args)

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



Search or jump to...

Pull requests Issues Codespaces Marketplace Explore

LuisFHB / miweb Public

Pin Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Quick setup — if you've done this kind of thing before

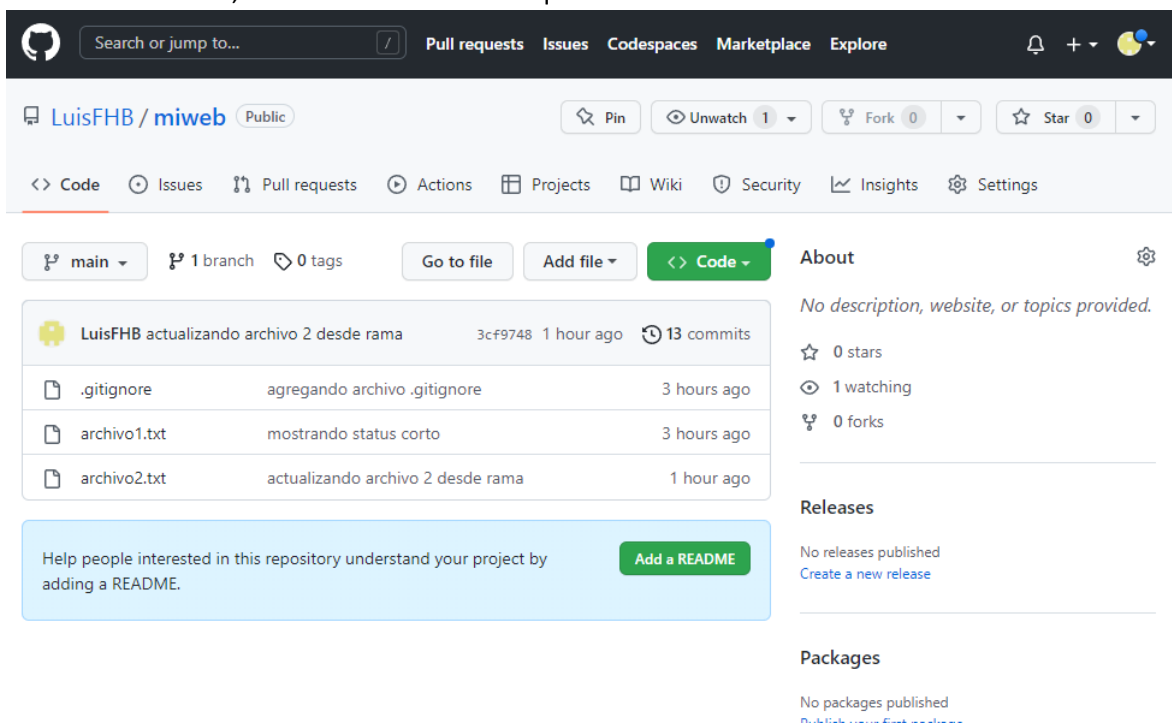
Set up in Desktop or HTTPS SSH <https://github.com/LuisFHB/miweb.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

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

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

Y refrescamos o F5, nos saldrán los archivos que creamos



Search or jump to...

Pull requests Issues Codespaces Marketplace Explore

LuisFHB / miweb Public

Pin Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file <> Code

LuisFHB actualizando archivo 2 desde rama 3cf9748 1 hour ago 13 commits

.gitignore	agregando archivo .gitignore	3 hours ago
archivo1.txt	mostrando status corto	3 hours ago
archivo2.txt	actualizando archivo 2 desde rama	1 hour ago

Help people interested in this repository understand your project by adding a README. [Add a README](#)

About

No description, website, or topics provided.

0 stars

1 watching

0 forks

Releases

No releases published

[Create a new release](#)

Packages

No packages published

[Publish your first package](#)