

GIT

GIT COMANDOS:

ANTES DE NADA, EN EL PROYECTO QUE ESTEMOS CREANDO, LE DAMOS A BOTON DERECHO Y CLICKAMOS GIT BASH HERE, PARA QUE AL INICIAR GIT, LO HAGAMOS EN EL PROYECTO DESEADO.

- **GIT INIT →** INICIAR EL SEGUIMIENTO DE GIT EN NUESTRO DIRECTORIO DE TRABAJO
- **GIT STATUS -S →** NOS CONFIRMA QUE TODO LO QUE ESTAMOS GUARDANDO EN ESA CARPETA GIT LO ESTA PROCESANDO

```
MINGW64: c:/Users/Maria/Desktop/Proyecto web empresa A
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git init
Initialized empty Git repository in C:/Users/Maria/Desktop/Proyecto web empresa A/.git/

Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git status -s
?? css/
?? index.html
?? javascript/

Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$
```

SI NOS FIJAMOS, CUANDO PONE INTERROGACIONES ES QUE NINGUNO ESTA SIENDO SOMETIDO A SEGUIMIENTO POR GIT

- **GIT CONFIG -- GLOBAL USER.EMAIL "EMAIL"**

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git config --global user.email "mariapuentebenito93@hotmail.com"
```

- **GIT CONFIG -- GLOBAL USER.NAME "NOMBRE"**

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git config --global user.name "Maria"
```

- **GIT ADD →** AÑADIR EL SEGUIMIENTO EN NUESTRO AREA DE ENSAYO

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git add index.html

Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git status -s
A index.html
?? css/
?? javascript/
```

AL HABER DADO EL SEGUIMIENTO AL ARCHIVO INDEX, NOS APARECE UNA A EN NUESTRO ARCHIVO YA QUE SE LE ESTA HACIENDO EL SEGUIMIENTO

- **GIT ADD . →** AÑADE EL SEGUIMIENTO DE TODOS NUESTROS ARCHIVOS AL AREA DE ENSAYO

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git add .

Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git status -s
A  css/estilos.css
A  javascript/archivo.js
```

COMO VEMOS AHORA NOS APARECE EL SEGUIMIENTO DE TODOS NUESTROS ARCHIVOS

- **GIT COMMIT -m "" →** TRASLADA EL ARCHICO O ARCHIVOS QUE TENGAMOS EN EL AREA DE ENSAYO AL REPOSITORIO LOCAL, EN EL QUE OBTENEMOS TODOS LOS RESPALDOS

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git commit -m "Comienzo del proyecto"
[master (root-commit) 0aa2442] Comienzo del proyecto
1 file changed, 7 insertions(+)
create mode 100644 index.html
```

AQUÍ ESTAMOS CREANDO UN RESPALDO DE NUESTRO INDEX.HTML

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git status -s
?? css/
?? javascript/
```

AHORA YA NOS INFORMA DE LOS ARCHIVOS QUE NO TIENEN SEGUIMIENTO NINGUNO POR QUE EL QUE YA TENGA COPIA DEL RESPALDO YA ESTA EN EL REPOSITORIO LOCAL

- **GIT COMMIT -am →** AUNA COMMIT Y ADD CUANDO HEMOS REALIZADO CAMBIOS EN VARIOS ARCHIVOS Y QUEREMOS CREAR EL RESPALDO DE TODOS

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git commit -am "Creado el parrafo y el color del titulo"
[master 65af42c] Creado el parrafo y el color del titulo
2 files changed, 6 insertions(+)
```

- **GIT LOG - --oneline →** NOS DA TODAS LAS COPIAS CON SU DESCRIPCION Y SU CODIGO CORRESPONDIENTE

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git log --oneline
e424390 (HEAD -> master) Agregado titulo y primer parrafo
0aa2442 Comienzo del proyecto
```

- ```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git reset --hard 0aa2442
HEAD is now at 0aa2442 Comienzo del proyecto
```

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git status -s
?? css/
?? javascript/

Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git log --oneline
0aa2442 (HEAD -> master) Comienzo del proyecto
```

```
[master e5e0608] He creado el parrafo y el color del titulo
Date: Sun Jun 11 13:06:13 2023 +0200
2 files changed, 6 insertions(+)

Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$
```

```
He creado el parrafo y el color del titulo

Please enter the commit message for your changes. Lines starting
with '#' will be ignored, and an empty message aborts the commit.
#
Date: Sun Jun 11 13:06:13 2023 +0200
#
On branch master
#
Changes to be committed:
modified: css/estilos.css
modified: index.html
#
~
~
~
~
~
~
~
~
~
~
```

[illegible]

```
[master ca4ad41] Creado el parrafo y el color del titulo
Date: Sun Jun 11 13:06:13 2023 +0200
2 files changed, 6 insertions(+)
```

## GIT SITUACIONES:

YA HEMOS GUARDADO EN NUESTRO REPOSITORIO LA INSTANTANEA DE NUESTRO INDEX.HTML, Y AHORA HACEMOS UNA MODIFICACION DENTRO DE NUESTRO ARCHIVO, SI VOLVEMOS A MIRAR EL ESTADO EN GIT NOS APARECE LO SIGUIENTE :

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git status -s
M index.html
?? css/
?? javascript/
```

NOS INDICA QUE ES UN CAMBIO QUE DE MOMENTO NO HA SIDO RESPALDADO POR GIT

PARA VOLVER A CREAR EL RESPALDO VOLVEMOS A HACER LOS SIGUIENTES COMANDOS

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git add index.html

Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git commit -m "Agregado titulo y primer parrafo"
[master e424390] Agregado titulo y primer parrafo
1 file changed, 6 insertions(+), 3 deletions(-)
```

POR EJEMPLO, CUANDO YO HE HECHO VARIAS MODIFICACIONES Y SE QUE FUNCIONAN QUIERO DIRECTAMENTE CREAR UN RESPALDO, EN VEZ DE HACER ESTE COMANDO

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git status -s
AM css/estilos.css
M index.html
A javascript/archivo.js

Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git add .

Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git status -s
A css/estilos.css
M index.html
A javascript/archivo.js

Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git commit -m "Creado el titulo en HTML y el css modificado con color de fondo"
[master 24a7032] Creado el titulo en HTML y el css modificado con color de fondo
3 files changed, 6 insertions(+)
create mode 100644 css/estilos.css
create mode 100644 javascript/archivo.js

Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git log --oneline
24a7032 (HEAD -> master) Creado el titulo en HTML y el css modificado con color de fondo
27ae09a Agregada la forma correcta de archivo HTML
0aa2442 Comienzo del proyecto
```

DEBERIAMOS HACER ESTE COMANDO QUE AUNA ADD Y COMMIT EN UNO

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git commit -am "Creado el parrafo y el color del titulo"
[master 65af42c] Creado el parrafo y el color del titulo
 2 files changed, 6 insertions(+)

Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git log --oneline
65af42c (HEAD -> master) Creado el parrafo y el color del titulo
24a7032 Creado el titulo en HTML y el css modificado con color de fondo
27ae09a Agregada la forma correcta de archivo HTML
0aa2442 Comienzo del proyecto
```

## GITHUB CON GIT

### Creación de un repositorio:

UNA VEZ NOS REGISTRAMOS EN GITHUB WEB, TENEMOS UNA PESTAÑA EN LA QUE NOS PONDRÁ NEW, CLICKAMOS Y NOS DA LA OPCION DE CREAR UN NUEVO REPOSITORIO.


EN ESTE, AÑADIMOS EL NOMBRE QUE QUEREMOS DARLE, UNA DESCRIPCION DE LO QUE HACE NUESTRO REPOSITORIO, Y SI QUEREMOS QUE SEA PUBLICO O PRIVADO.

(RECORDANDO QUE EL NOMBRE SIEMPRE SERÁ, NUESTRO NOMBRE DE USUARIO/ EL NOMBRE QUE LE ASIGNEMOS)

### Create a new repository

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

Owner \*

 Mariapiente ▾

Repository name \*

CursoGit

✓ CursoGit is available.

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

Description (optional)

Se trata de pruebas que realizo mientras aprendo Git y Github

☒  **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.

AHORA NOS VA A DAR UN COMIENZO RAPIDO

NOS DICE QUE TENEMOS DOS OPCIONES:

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

 Set up in Desktop or ☐ HTTPS ☒ SSH <https://github.com/Mariapiente/CursoGit.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 "# CursoGit" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Mariapiente/CursoGit.git
git push -u origin main
```



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

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



LA PRIMERA ES CREAR UN NUEVO REPOSITORIO CON LINEA DE COMANDOS

Y LA SEGUNDA, (QUE ES LA QUE NOSOTROS VAMOS A UTILIZAR) ES QUE VAMOS A SUBIR UN REPOSITORIO EXISTENTE CON LINEA DE COMANDOS

LA ULTIMA OPCION, NOS DICE QUE PODEMOS IMPORTAR DE UN REPOSITORIO, A OTRO REPOSITORIO

### ...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

## Subir nuestro repositorio de Git a GitHub

SI NOS FIJAMOS, PODEMOS VER QUE NOS DAN TRES LINEAS DE COMANDO PARA ENLAZAR LOS REPOSITORIOS

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

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

LA PRIMERA, AÑADE TODO NUESTRO REPOSITORIO EN GIT A NUESTRO REPOSITORIO EN GITHUB

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$ git remote add origin https://github.com/Mariapuerto/CursoGit.git
```

LA SEGUNDA, LO QUE HACE ES CREAR RAMAS

LA TERCERA, AÑADE LAS MODIFICACIONES QUE SE VAYAN CREANDO

LA SEGUNDA, ES LA PRIMERA QUE DEBEMOS HACER PARA CREAR LA RAMA MASTER EN GITHUB Y PODER SUBIR DESPUES NUESTRO ENLACE (PRIMERA LINEA DE CODIGO). POR ÚLTIMO PONDREMOS LA TERCERA LINEA DE CODIGO. Y EL RESULTADO ES:

The screenshot shows the GitHub interface for a repository named 'CursoGit' by user 'Mariapuerto'. The repository is public and has a 'main' branch. The commit history shows a single commit 'Creado el parrafo y el color del titulo' by 'ca4ad41' 1 hour ago, with 4 commits in total. The file list includes 'css', 'javascript', and 'index.html', all created 1 hour ago. A button 'Add a README' is visible at the bottom.

| File       | Commit Message                                                  | Commit Hash | Time       |
|------------|-----------------------------------------------------------------|-------------|------------|
| css        | Creado el parrafo y el color del titulo                         | ca4ad41     | 1 hour ago |
| javascript | Creado el titulo en HTML y el css modificado con color de fondo |             | 1 hour ago |
| index.html | Creado el parrafo y el color del titulo                         |             | 1 hour ago |

SI DENTRO DE GITHUB EN REMOTO, HACEMOS ALGUN CAMBIO Y QUEREMOS TRAERLO A LOCAL CON EL COMANDO GIT PULL LO ACTUALIZAMOS TAMBIEN EN NUESTRO ORDENADOR

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (aleatoria)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 719 bytes | 179.00 KiB/s, done.
From https://github.com/Mariapuerto/CursoGit
 e39f6b6..71ecf18 aleatoria -> origin/aleatoria
Updating e39f6b6..71ecf18
Fast-forward
 codigos.html | 4 ++--
 1 file changed, 2 insertions(+), 2 deletions(-)
```

## Creación de versiones

SI YO CREO QUE EL PROYECTO CREADO YA ES VIABLE PARA QUE GENTE LO PUEDA VER Y DESCARGARSE, AUNQUE SEA LA VERSION 1.0 LO HARÉ DE LA SIGUIENTE MANERA:

CREAMOS EL TAG DANDOLE UN NOMBRE CARACTERÍSTICO, Y ENTRE COMILLAS PONEMOS UNA DASCRIPCION PARA NOSOTROS

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (aleatoria)
$ git tag 12/06/23v1 -m "Version 1.0 proyecto"
```

AL CREAR EL TAG, GITHUB NO NOS HARA NADA, SIMPLEMENTE AL MIRAR EL LO - -ONELINE NOS APARECERA QUE LO HEMOS CREADO

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (aleatoria)
$ git log --oneline
71ecf18 (HEAD -> aleatoria, tag: 12/06/23v1, origin/aleatoria) Creado el parrafo
de codigo
e39f6b6 Creado HolaMundo en Java
02cf084 Creado html de codigos
81085ce Creado titulo y primer parrafo
0d6b27b Creada pagina de estilos y enlazada con el html
9bea016 Creados html Index y Codigos
a0f12f7 creado el html de codigos
```

TRAS ESTE PASO, LO QUE VAMOS A HACER ES EL SIGUIENTE COMANDO PARA SUBIR EL TAG A GITHUB

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (aleatoria)
$ git push --tags
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 179 bytes | 179.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Mariapuerto/CursoGit.git
 * [new tag] 12/06/23v1 -> 12/06/23v1
```



PODREMOS VER EN GITHUB QUE HEMOS SUBIDO EL TAG

12/06/23v1 ...

🕒 19 minutes ago 🔗 71ecf18 📦 zip 📦 tar.gz

## Trabajar con diferentes ramas

CUANDO EN GITHUB TRAS EL COMANDO “GIT BRANCH -M MAIN” HEMOS CREADO LA RAMA MAIN, Y HEMOS IDO AÑADIENDO TODO A ESA MISMA RAMA, TODAS LAS MODIFICACIONES, CREACION DE TAGS ETC ETC .. SE HAN HECHO UNICA Y EXCLUSIVAMENTE EN ESA RAMA

PERO, POR EJEMPLO, SI QUEREMOS CREAR UNA RAMA DISTINTA PARA PODER SUBIR MODIFICACIONES SIN AFECTAR A LA RAMA PRINCIPAL, LO HAREMOS CON EL MISMO COMANDO, PERO CAMBIANDO EL “MAIN” POR EL NOMBRE QUE QUERAMOS ASIGNARLE A LA RAMA EN LA QUE VAMOS A TRABAJAR NOSOTROS.

UNA VEZ CREADA, CUANDO SUBAMOS CAMBIOS LO HAREMOS CON GIT PUSH -U ORIGIN (NOMBRE DE LA NUEVA RAMA)

CADA VEZ QUE QUERAMOS TRABAJAR CON ALGUNA TENDREMOS QUE UTILIZAR EL COMANDO GIT SWITCH (NOMBRE DE LA RAMA) o GIT CHECKOUT (NOMBRE DE LA RAMA)

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (aleatoria)
$ git switch main
Switched to a new branch 'main'
branch 'main' set up to track 'origin/main'.
```

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (main)
$ git checkout aleatoria
Switched to branch 'aleatoria'
Your branch is up to date with 'origin/aleatoria'.
```

SI NOS FIJAMOS, NOS PONE ENTRE PARENTESIS QUE ESTAMOS TRABAJANDO EN ALEATORIA

Y AL HACER ESE COMANDO NOS APARECERA MAIN:

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (main)
$ git pull
Already up to date.
```

## Cambiar nombre a una rama existente

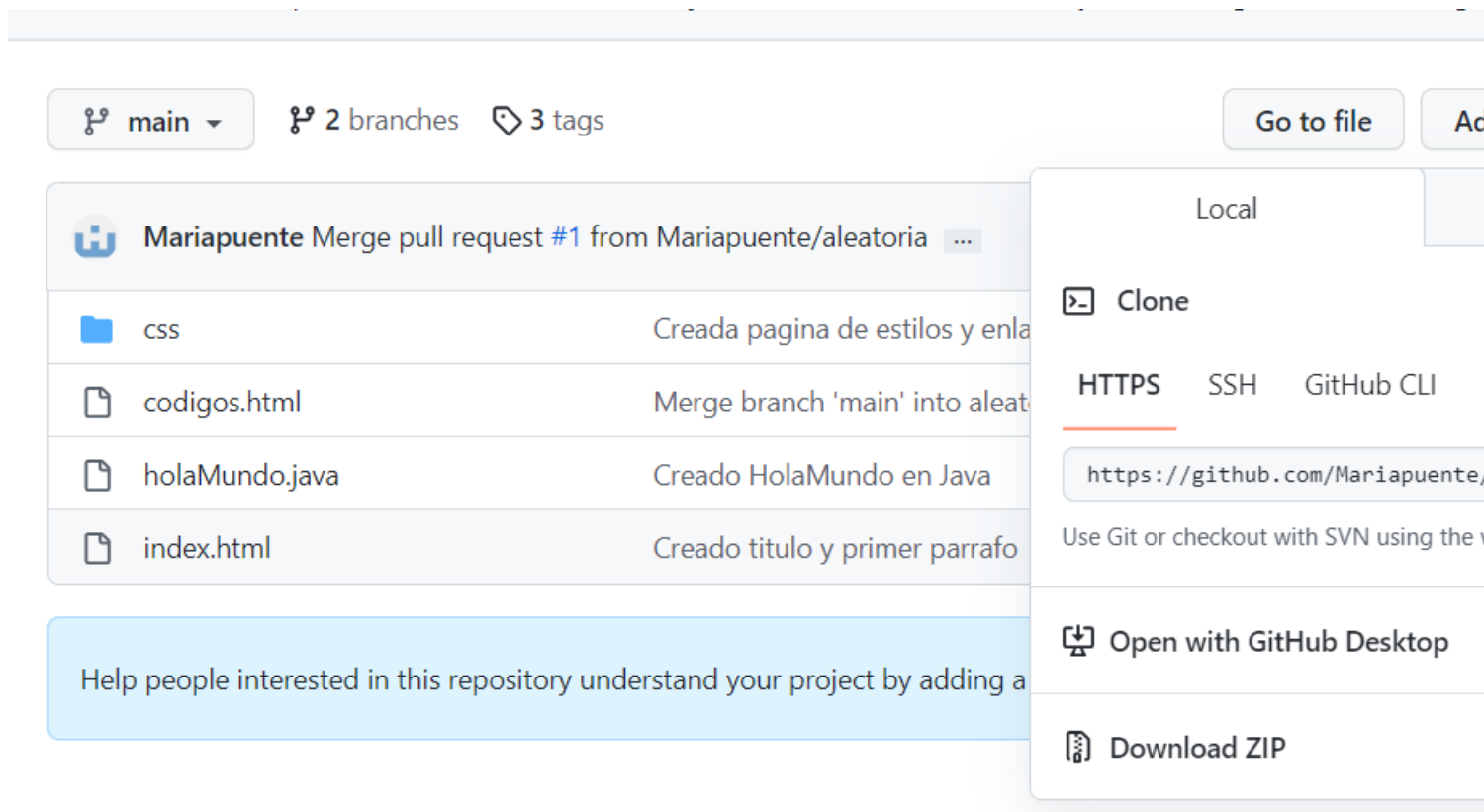
PARA CAMBIARLE EL NOMBRE A UNA RAMA EXISTENTE HAREMOS EL SIGUIENTE COMANDO

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (main)
$ git branch main -M master

Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (master)
$
```

## Clonar el repositorio tras perdida en local

SI HEMOS PERDIDO TODO EL TRABAJO EN NUESTRO ORDENADOR, TENEMOS LA SUERTE DE TENERLO EN GITHUB (REMOTO) POR LO QUE, PARA VOLVER A TENERLO EN LOCAL, HAREMOS LO SIGUIENTE:



COPIAMOS EL ENLACE QUE NOS APARECE TRAR PULSAR CODE.

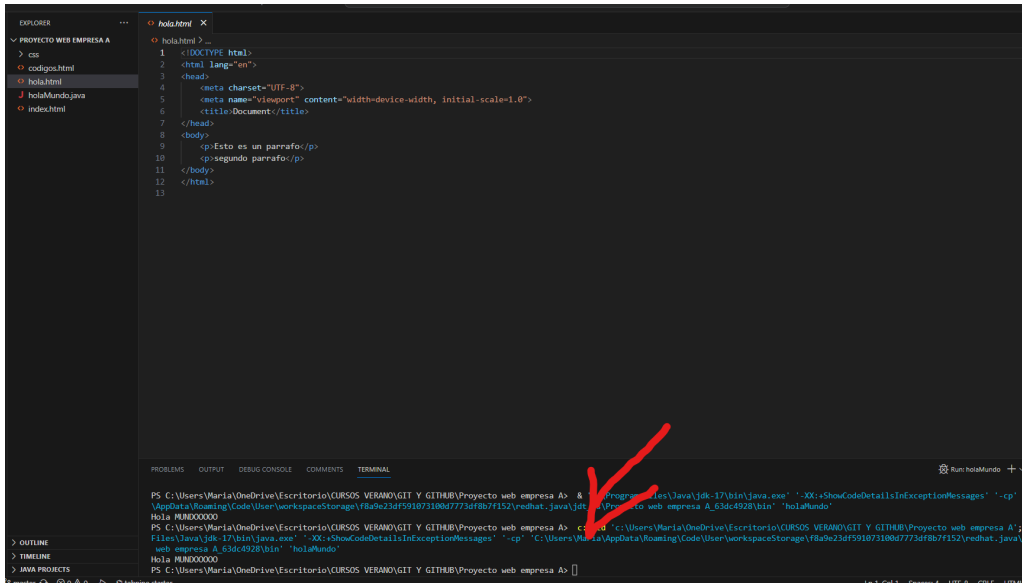
Y EN GIT USAMOS COMANDO GIT CLONE (ENLACE COPIADO)

```
Maria@Meery MINGW64 ~/Desktop/Proyecto web empresa A (main)
$ git clone https://github.com/Mariapuerto/CursoGit.git
Cloning into 'CursoGit'...
remote: Enumerating objects: 39, done.
remote: Counting objects: 100% (39/39), done.
remote: Compressing objects: 100% (32/32), done.
remote: Total 39 (delta 12), reused 23 (delta 3), pack-reused 0
Receiving objects: 100% (39/39), 6.03 KiB | 6.03 MiB/s, done.
Resolving deltas: 100% (12/12), done.
```

Y AHORA YA PODEMOS COMPROBAR QUE VOLVEMOS A TENER EL PROYECTO EN NUESTRO ORDENADOR

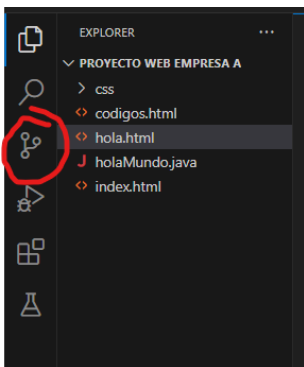
# GITHUB CON VS CODE

HAY QUE TENER CLARO QUE TODOS LOS COMANDOS VISTOS ANTERIORMENTE, LOS PODREMOS USAR EN LA CONSOLA DE VS CODE.

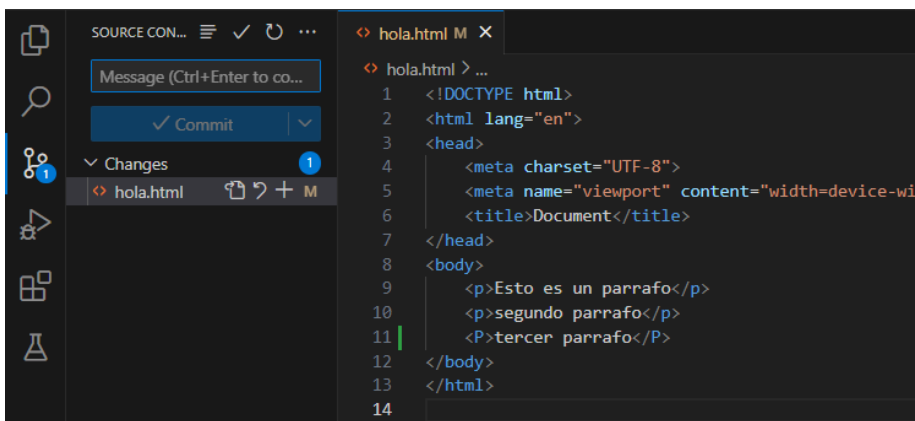


CUANDO ESTEMOS EN UN PROYECTO QUE QUEREMOS SUBIR A GIT, INICIAREMOS GIT DENTRO DEL MISMO.

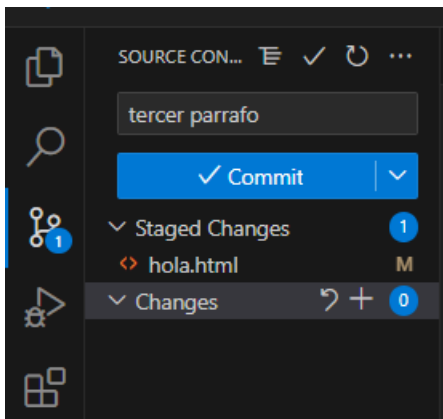
TENEMOS LA OPCION DE UTILIZAR TODOS LOS COMANDOS VISTOS ANTERIORMENTE O DE LA SIGUIENTE MANERA



DENTRO DE ESTA PARTE DEL VS CODE, PODEMOS SUBIR, MODIFICAR RAMAS Y FICHEROS Y SINCRONIZARLOS DIRECTAMENTE CON NUESTRO REPOSITORIO EN REMOTO.

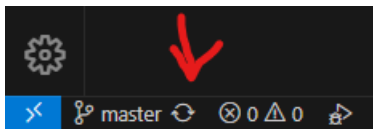


AQUÍ PODEMOS VER, QUE EN CUANTO MODIFICAMOS ALGO EN NUESTRO ARCHIVO, DIRECTAMENTE VS NOS LO DICE. AQUÍ ÚNICAMENTE TENDRIAMOS QUE HACER COMMIT Y SINCRONIZARLO CON NUESTRO REPOSITORIO REMOTO.



PONEMOS EL TITULO DEL COMMIT Y CLICKAMOS. Y SI VAMOS A GITHUB PODREMOS VER LOS CAMBIOS QUE HEMOS REALIZADO.

CUANDO HAGAMOS CAMBIOS DIRECTAMENTE EN REMOTO, TENEMOS QUE TENER EN CUENTA QUE NO SE NOS GUARDA EN NUESTRO REPOSITORIO LOCAL, POR LO TANTO, EN VS DIRECTAMENTE TENDREMOS QUE HACER LO SIGUIENTE.



SI DAMOS A SINCRONIZAR, NOS APARECERAN LOS CAMBIOS REALIZADOS TANTO EN REMOTO COMO EN LOCAL.