



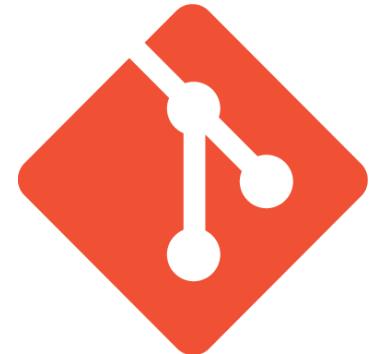
git



¿QUÉ ES GIT?

Linus Torvalds : Creador del sistema operativo Linux y creador de Git

Sistema de control de versiones, que nos ayuda a registrar los cambios que se realizan en uno o un paquete de archivos





→ VCS

VENTAJAS

Realizar copias de seguridad

Histórico, trazabilidad

Resolver conflictos

Qué es el control de versiones

Los sistemas de control de versiones son una categoría de herramientas de software que ayudan a un equipo de software a gestionar los cambios en el código fuente a lo largo del tiempo. El software de control de versiones realiza un seguimiento de todas las modificaciones en el código en un tipo especial de base de datos. Si se comete un error, los desarrolladores pueden ir atrás en el tiempo y comparar las versiones anteriores del código para ayudar a resolver el error, al tiempo que se minimizan las interrupciones para todos los miembros del equipo.

Qué es el control de versiones

El control de versiones protege el código fuente tanto de las catástrofes como del deterioro casual de los errores humanos y las consecuencias accidentales.



EL PROCESO DE TU TESIS



Tesis



Tesis final



Tesis final este
si



Tesis final este
si si si



Tesis final 2



Tesis final final



Tesis final listo



Tesis final por fin



Tesis final por fin
eso espero
the end



Tesis finalisimo



Tesis ultimo



Tesis ultimo
ahora si



Tesis ultimo de
los ultimos



Tesis ultimo
final ok



Doc-Copia2.txt



Doc-CopiaFinal.txt



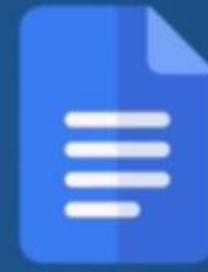
Doc-Copia.txt



Doc-FinalDelFinal.txt



Git



Doc.txt

Cambio Actual



Cambio 2

Cambios 1



Doc-Copia3.txt



Doc-Definitiva.txt

Sistemas de control de versiones local

Es ideal para evitar determinados errores, como es el caso de sobrescribir un archivo por accidente, ya que cuenta con una base de datos, donde recoge los registros de todos los cambios realizados sobre un archivo.

Control de versiones local: Compuesto por el estado actual y archivo de versiones anteriores.

Sistemas de control de versiones centralizados

Más de un cliente trabaja en un único servidor que contiene todos los archivos versionados.

Si el servidor se cae nadie puede trabajar y guardar los cambio. Si el disco duro con la base de datos central se corrompe, la única solución es contar con copias de seguridad. El mismo caso en el sistema de control local, si no trabajas con copias de seguridad, se arriesga a perderlo.

Control de versiones centralizado: Compuesto por el archivo de versiones y el archivo actual.

Sistemas de control de versiones distribuidos

Puedes colaborar con gente simultáneamente dentro del mismo proyecto; permitiendo establecer varios flujos de trabajo.

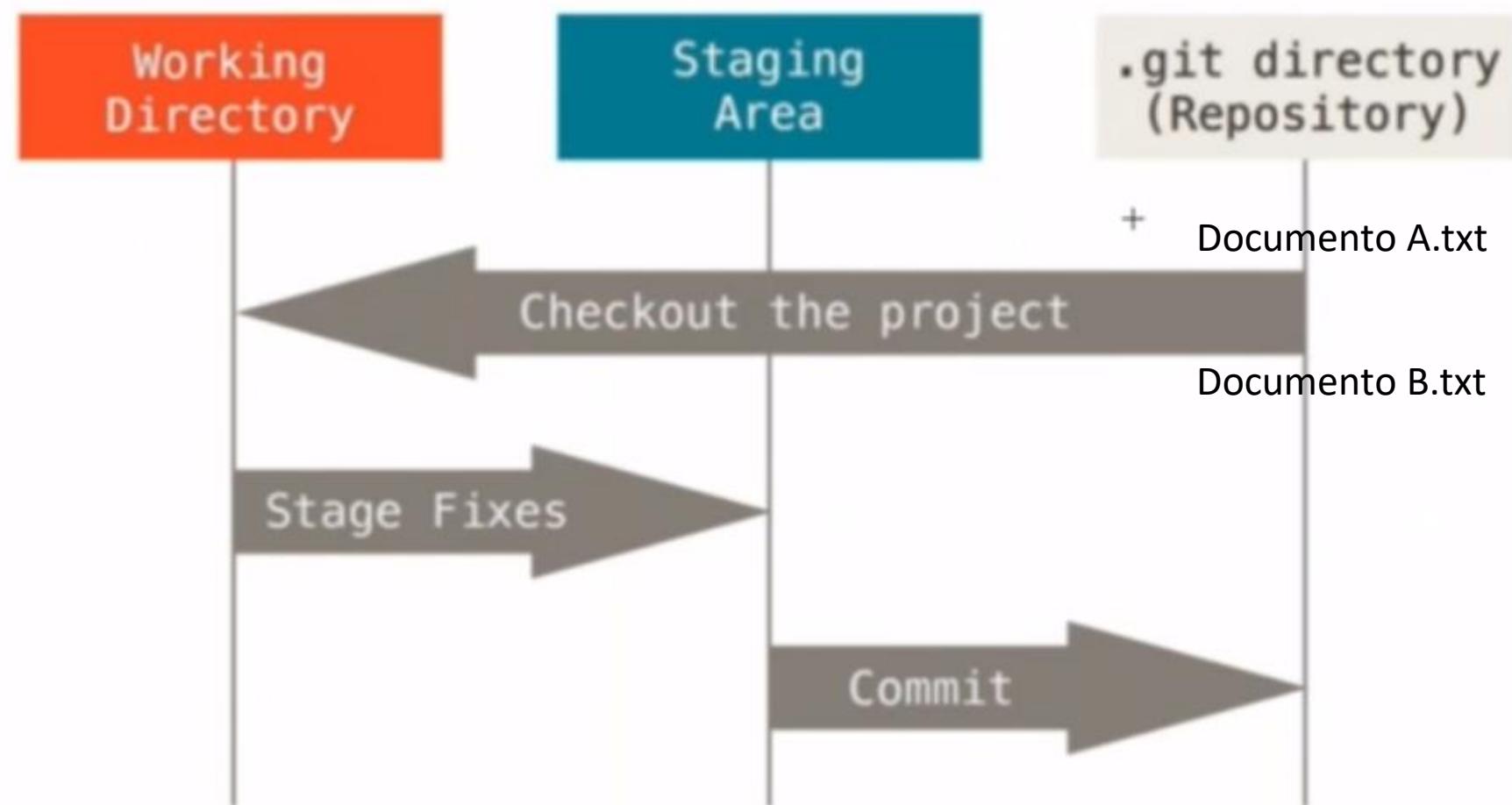
El servidor cuenta con la última versión de los archivos, el cliente se descarga para trabajar una réplica completa.

De esta forma, si un servidor cae, cualquier clientes puede copiar en el servidor la réplica descargada y restaurarlo.

En resumen cada instantánea hace una copia de seguridad completa de todos los datos.

3 Stages of Git

El directorio de trabajo, el área de preparación y el directorio git (Repositorio).

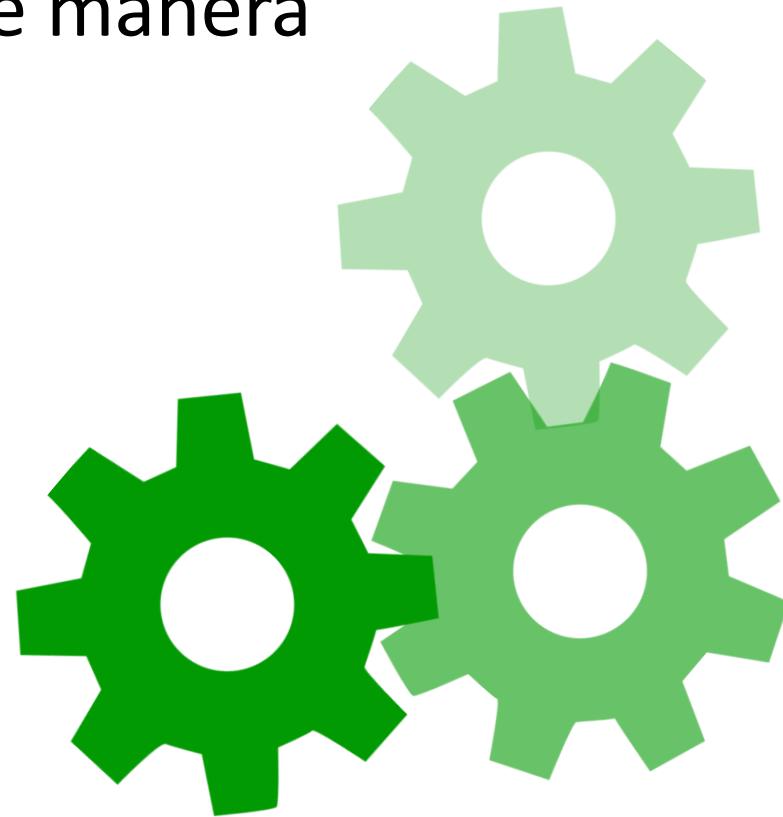


El directorio de trabajo es donde se agregan, borran y editan los archivos. Luego, los cambios son preparados (indexados) en el área de preparación. Después de que confirmes tus cambios, la instantánea de los cambios se guardará en el directorio git.

Working directory

El directorio de trabajo

En este estado editamos y trabajamos en nuestros proyectos ,aun sin usar Git, es decir de manera local

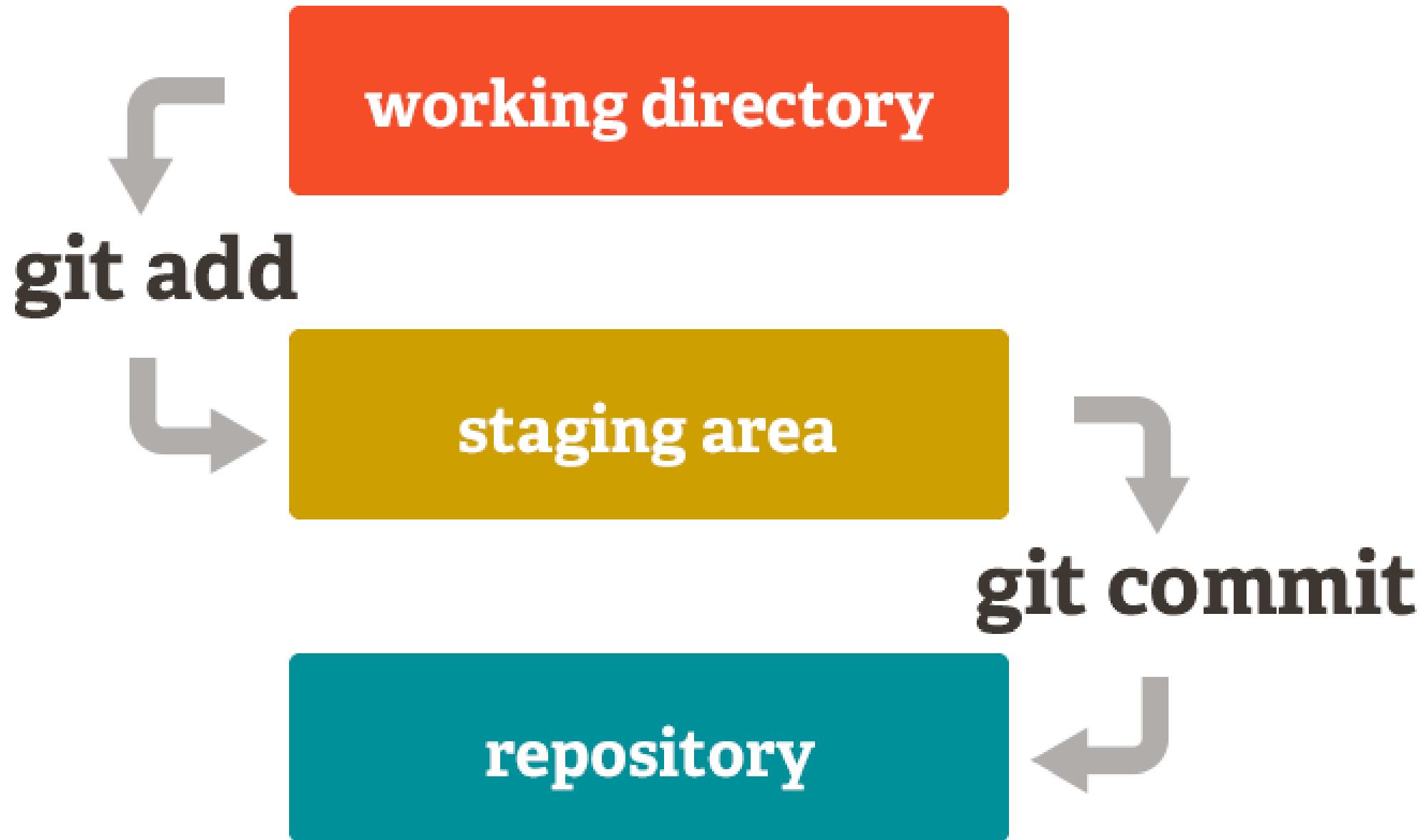


Staging Area

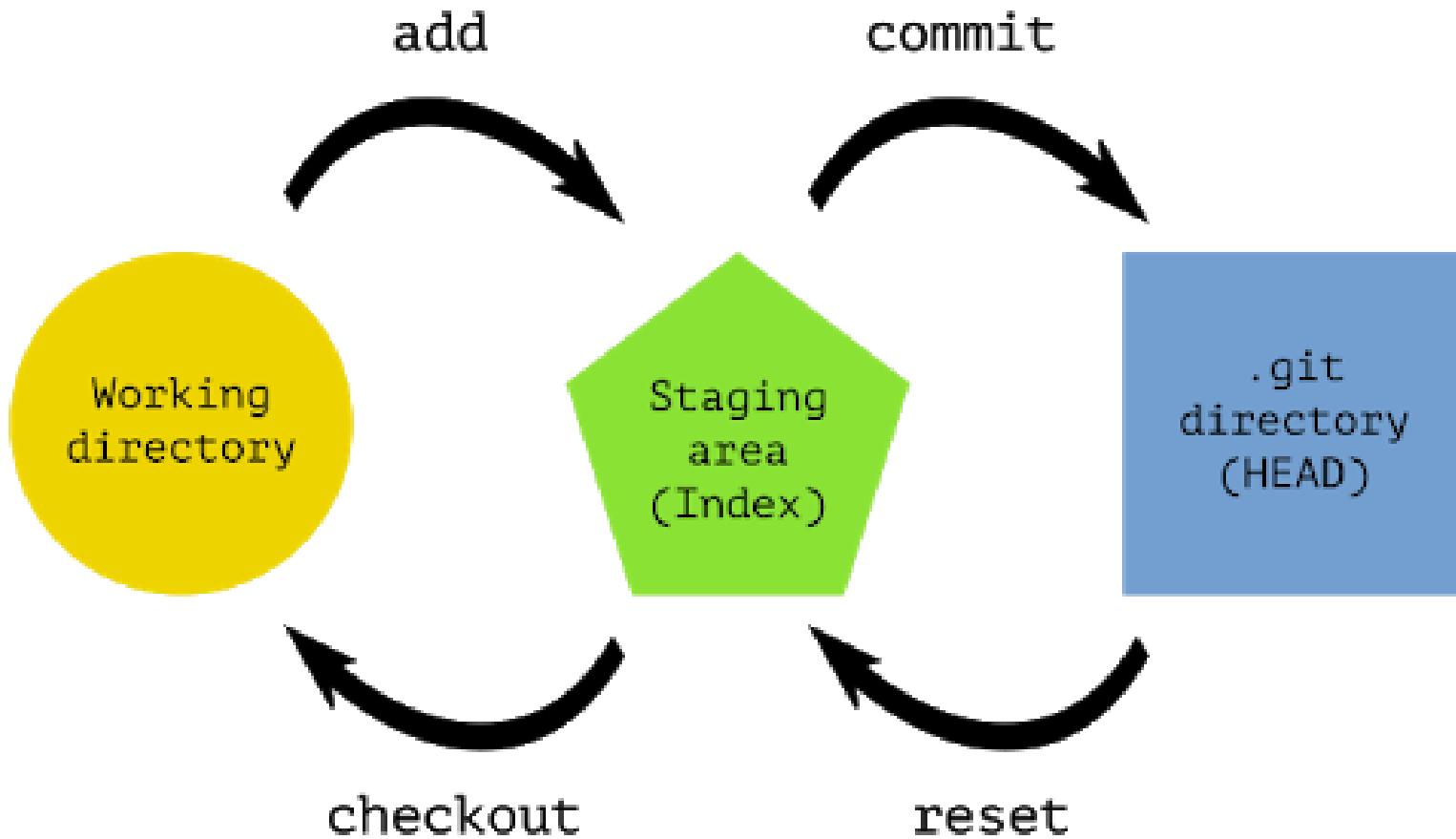
En este estado es donde elegimos que archivos están listos y cumplen al 100% con nuestros parámetros y se deciden cuales no están listos para pasar al ultimo estado.

Repository

En este ultimo estado nosotros usaremos el control de versiones de Git, donde todo nuestro proceso será guardado atreves de “**commits**”



Flujo de trabajo en Git



Descargar Git

Volvemos a
las 10:15am

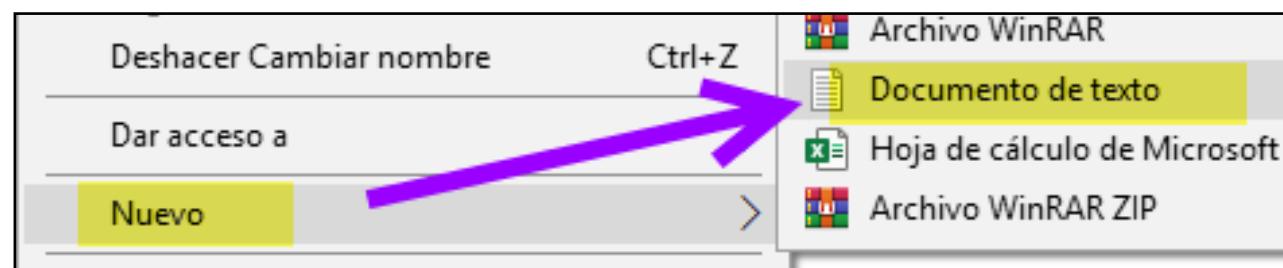
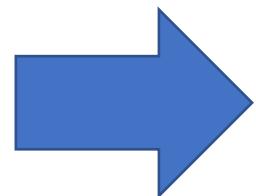
The screenshot shows the official Git download page at <https://git-scm.com/downloads>. The page features the Git logo and the tagline "distributed-is-the-new-centralized". On the left, there's a sidebar with links for About, Documentation, Downloads (selected), GUI Clients, Logos, and Community. The main content area has a "Downloads" heading with sections for macOS, Windows, and Linux/Unix. Below this, it says "Older releases are available and the [Git source repository](#) is on GitHub." To the right, there's a "GUI Clients" section with a link to "View GUI Clients →". On the far right, there's a "Logos" section with a link to "View Logos →". A purple arrow points from the text "Volvemos a las 10:15am" to the "Latest source Release" section, which displays "2.31.1" and a link to "Release Notes (2021-03-26)".

<https://git-scm.com/downloads>

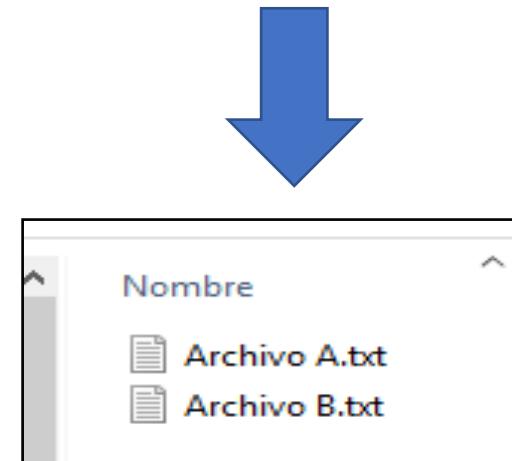
Creamos
carpeta



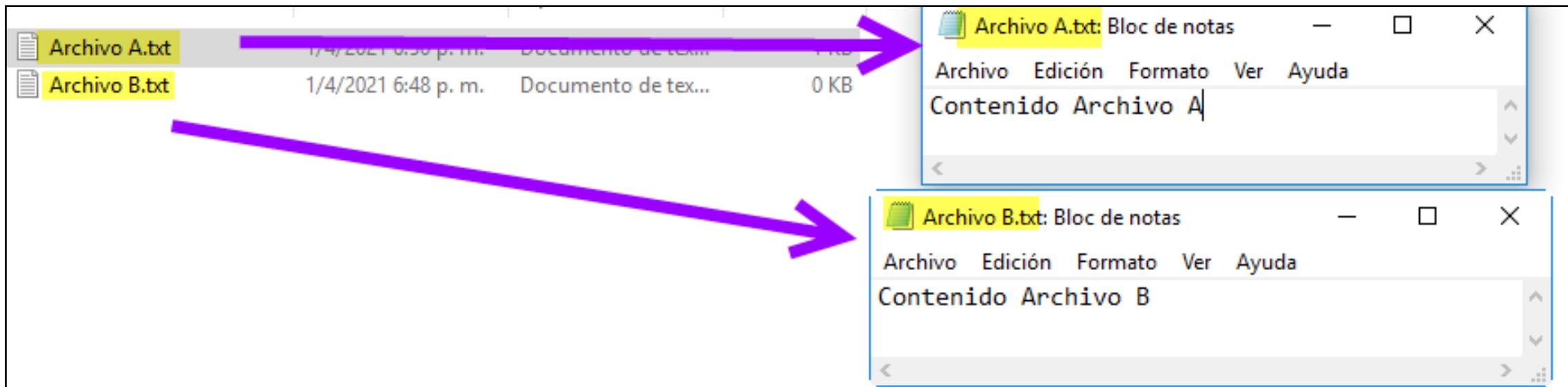
Abrimos la
carpeta



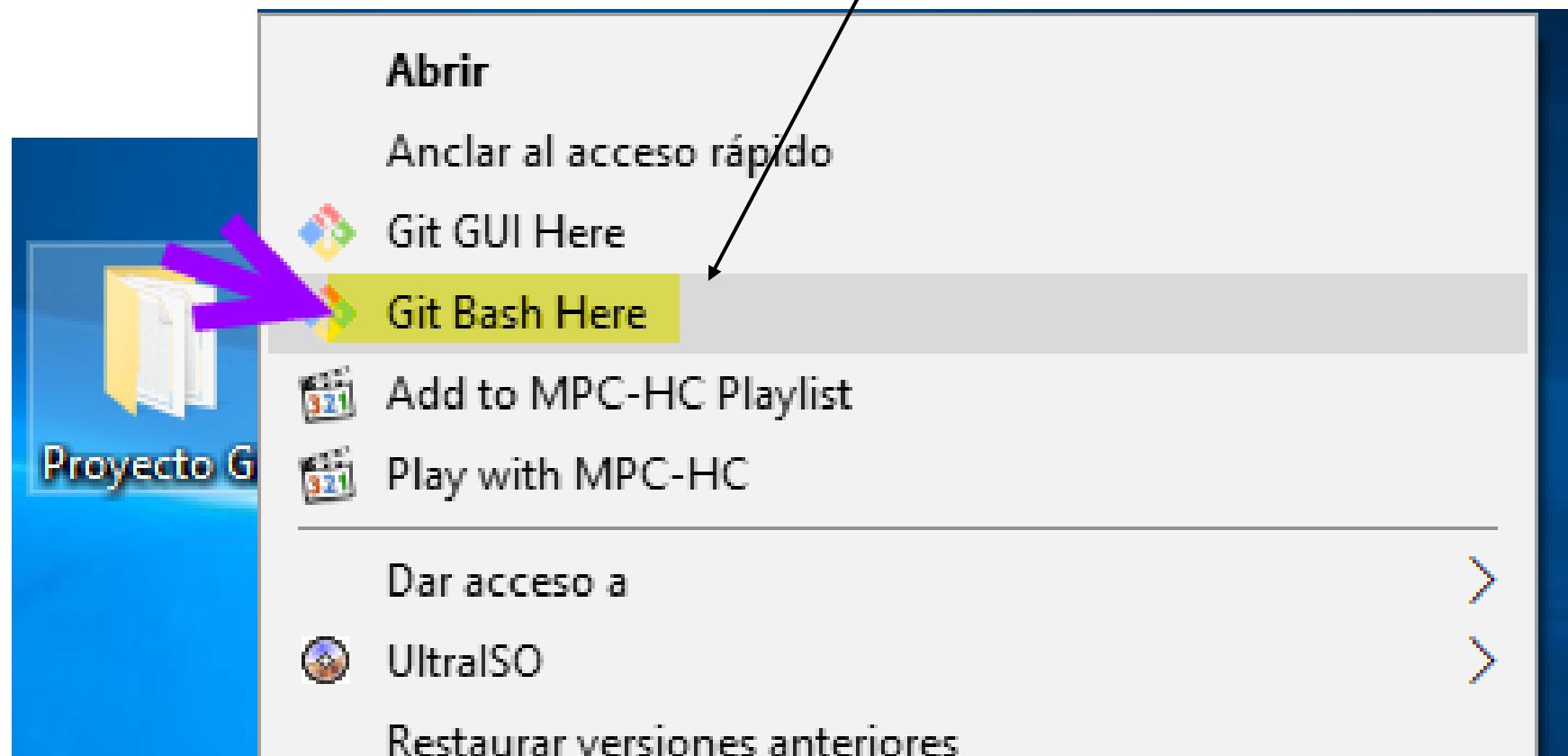
Dentro de la carpeta
creamos dos archivos
de texto



Agregamos contenido con el blog de notas a los archivos



Le damos clic derecho a la carpeta que contiene los dos archivos y seleccionamos la opción



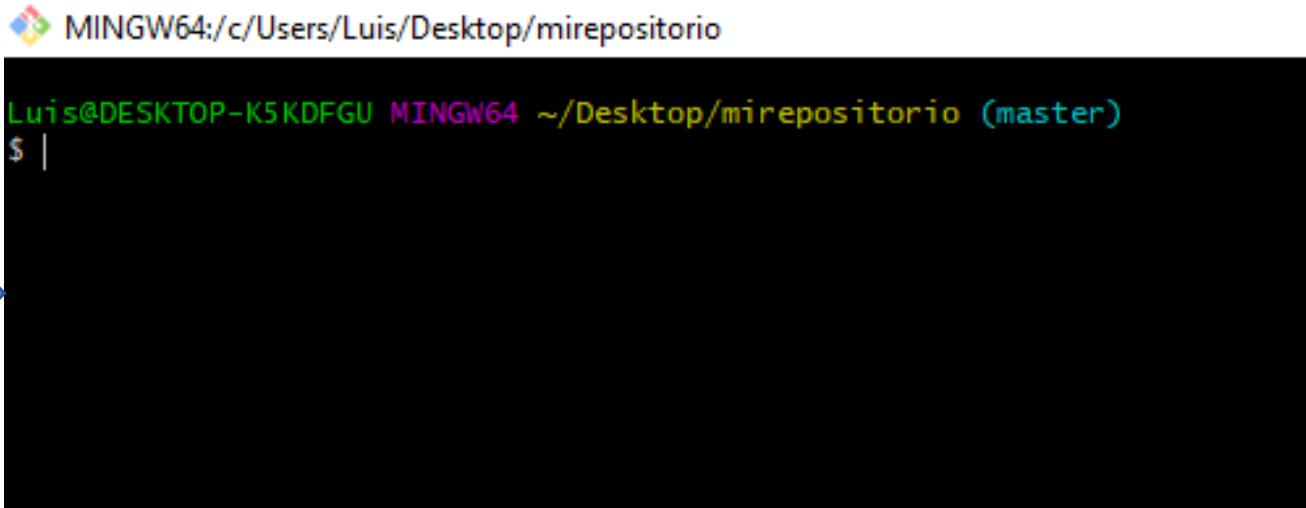
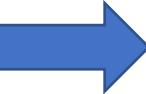
git --help

Este comando permite ver los principales comandos de git

```
$ git --help ←  
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]  
          [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]  
          [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]  
          [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]  
          [--super-prefix=<path>] [--config-env=<name>=<envvar>]  
          <command> [<args>]  
  
These are common Git commands used in various situations:  
  
start a working area (see also: git help tutorial)  
  clone      Clone a repository into a new directory  
  init       Create an empty Git repository or reinitialize an existing one  
  
work on the current change (see also: git help everyday)  
  add        Add file contents to the index  
  mv         Move or rename a file, a directory, or a symlink  
  restore    Restore working tree files  
  rm         Remove files from the working tree and from the index  
  sparse-checkout Initialize and modify the sparse-checkout  
  
examine the history and state (see also: git help revisions)  
  bisect    Use binary search to find the commit that introduced a bug  
  diff      Show changes between commits, commit and working tree, etc  
  grep      Print lines matching a pattern  
  log       Show commit logs  
  show      Show various types of objects  
  status    Show the working tree status  
  
grow, mark and tweak your common history  
  branch   List, create, or delete branches  
  commit   Record changes to the repository  
  merge    Join two or more development histories together  
  rebase   Reapply commits on top of another base tip  
  reset   Reset current HEAD to the specified state  
  switch  Switch branches  
  tag     Create, list, delete or verify a tag object signed with GPG  
  
collaborate (see also: git help workflows)  
  fetch   Download objects and refs from another repository  
  pull    Fetch from and integrate with another repository or a local branch  
  push    Update remote refs along with associated objects  
  
'git help -a' and 'git help -g' list available subcommands and some  
concept guides. See 'git help <command>' or 'git help <concept>'  
to read about a specific subcommand or concept.  
See 'git help git' for an overview of the system.
```

clear

Este comando permite
borrar la pantalla



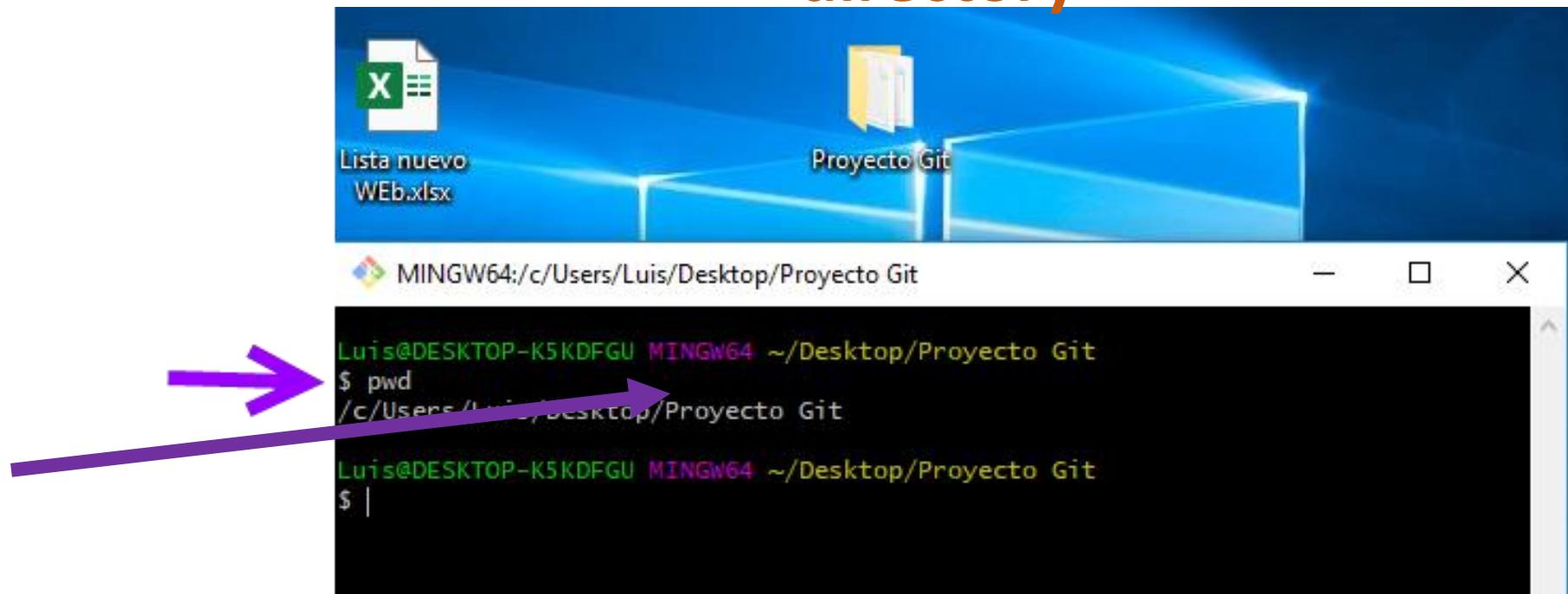
A screenshot of a terminal window titled "MINGW64:/c/Users/Luis/Desktop/mirepositorio". The window shows the command "Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)" followed by a prompt "\$ |". The terminal window has a black background and white text.

history -c

Este comando permite borrar el historial de los
comandos que seleccionamos cuando movemos
flecha arriba flecha abajo

Puedes saber en qué directorio estás a través del comando `pwd`, que significa “(muestra) imprime el directorio de trabajo” (print working directory)

Se nos abre la consola y escribimos **pwd**
Para saber la carpeta de trabajo **working directory**



Si escribimos el siguiente comando podemos ver los archivos dentro de la carpeta con nuestro proyecto

ls

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git
$ ls
'Archivo A.txt'  'Archivo B.txt'
```

git init

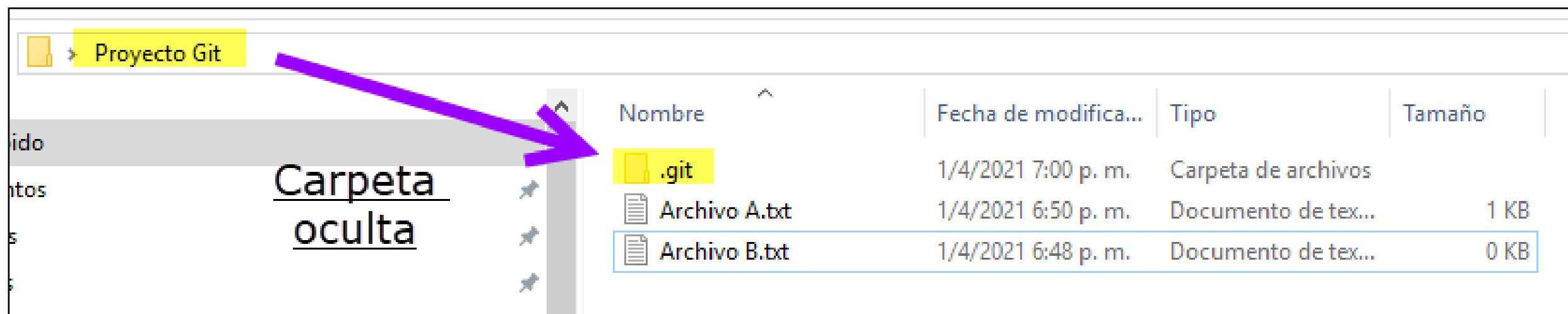
Este comando permite crear un nuevo repositorio de git

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git
$ git init
Initialized empty Git repository in C:/Users/Luis/Desktop/Proyecto Git/.git/

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$
```

El comando **git init crea un nuevo repositorio de Git**. Puede utilizarse para convertir un proyecto existente y sin versión en un repositorio de Git o inicializar un nuevo repositorio vacío. La mayoría de los demás comandos de Git no se encuentran disponibles fuera de un repositorio inicializado, por lo que este suele ser el primer comando que se ejecuta en los proyectos nuevos.

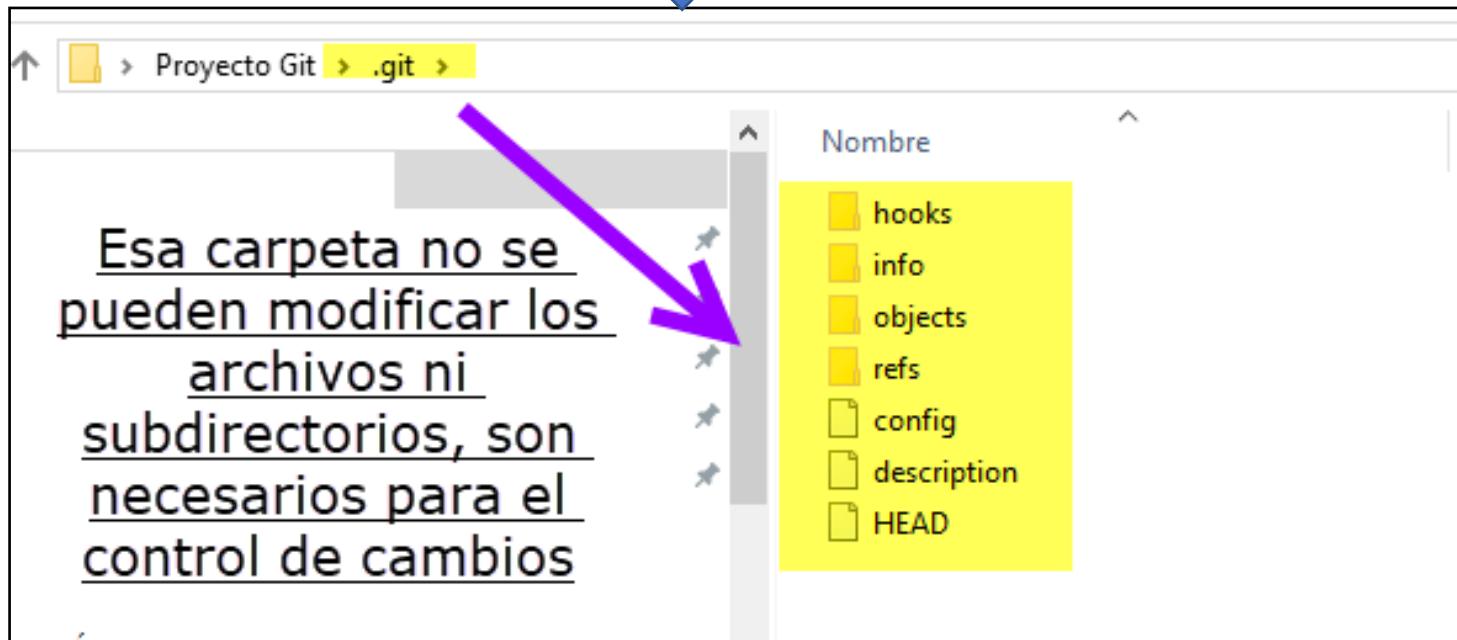
Al ejecutar git init, se crea un subdirectorio de .git en el directorio de trabajo en uso, que contiene todos los metadatos de Git necesarios para el nuevo repositorio. Estos metadatos incluyen subdirectorios de objetos, referencias y archivos de plantilla. También se genera un archivo HEAD que apunta a la confirmación extraída.



Proyecto Git

Nombre	Fecha de modifica...	Tipo	Tamaño
.git	1/4/2021 7:00 p. m.	Carpeta de archivos	
Archivo A.txt	1/4/2021 6:50 p. m.	Documento de tex...	1 KB
Archivo B.txt	1/4/2021 6:48 p. m.	Documento de tex...	0 KB

Carpeta
oculta



↑ Proyecto Git > .git >

Esa carpeta no se
pueden modificar los
archivos ni
subdirectorios, son
necesarios para el
control de cambios

Nombre
hooks
info
objects
refs
config
description
HEAD

git status

Este comando permite ver el estado del directorio de trabajo, visualizar los archivos con los que estamos trabajando y si están agregados al área de trabajo

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git status
On branch master
  ↗
No commits yet

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

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

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
|$
```

git add

El comando git add añade un cambio del directorio de trabajo en el entorno de ensayo.

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git add "Archivo A.txt"
```

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$
```

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git status
On branch master
```

```
No commits yet
```

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

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

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$
```

Sin embargo, **git add** no afecta al repositorio de manera significativa: en realidad, los cambios no se registran hasta que ejecutas git commit

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git add "Archivo B.txt"
```

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git status
On branch master
```

```
No commits yet
```

```
Changes to be committed:
```

```
(use "git rm --cached <file>..." to unstage)
```

```
new file: Archivo A.txt
```

```
new file: Archivo B.txt
```

Junto con estos comandos, también necesitarás git status para ver el estado del directorio de trabajo y el entorno de ensayo.

git add debe llamarse cada vez que se modifica un archivo

```
git add archivo.txt
```

El archivo ahora estará en el área de ensayo, preparado para formar parte del **historial del proyecto**.

Si se han hecho cambios en varios archivos, se pueden añadir todos juntos al **área de ensayo** con el siguiente código:

```
git add .
```

Los comandos `git add`, `git status` y `git commit` se usan en combinación para guardar una instantánea del estado de un proyecto de Git.

git config

Git trae una herramienta llamada git config, que te permite obtener y establecer variables de configuración que controlan el aspecto y funcionamiento de Git.

```
$ git config --global user.name "[name]"
```

Establece el nombre que desea esté anexado a sus transacciones de commit

```
$ git config --global user.email "[email address]"
```

Establece el e-mail que desea esté anexado a sus transacciones de commit

```
$ git config --global color.ui auto
```

Habilita la útil colorización del producto de la línea de comando

git config –global user.email “correo del usuario”

Este comando permite identificarnos dentro de git es decir
Establece el e-mail que desea esté anexado a sus transacciones de
commit

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git config --global user.email "luis73web@gmail.com"
```

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ |
```



git config --global user.name "usuario"

Este comando permite establecer el nombre que desea esté anexado a sus transacciones de commit

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git config --global user.name "luis"
```

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ |
```

```
Luis@DESKTOP-K5KDFGU MINGW6  
$ git add "Archivo B.txt"
```

git commit

Este comando permite grabar los cambios del área de ensayo al repositorio

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)  
$ git commit  
[master (root-commit) b07ca28] MI PRIMER COMMIT  
 2 files changed, 2 insertions(+)  
 create mode 100644 Archivo A.txt  
 create mode 100644 Archivo B.txt
```

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
```

Al ingresar al editor seleccionamos la tecla **i** para editar .

PODEMOS ESCRIBIR CUALQUIER MENSAJE EJEMPLO MI PRIMER COMMIT

Y podemos grabar el seleccionando la teclas **control + c** y luego escribir **:wq**

Después del comando **git commit**, ingresó al editor, así que primero presione **i** y luego comience a escribir. Después de enviar su mensaje, pulse **Ctrl + c** y luego **:wq**

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Mi proyecto web (master)
$ git commit
[master (root-commit) 94afa58] ta es mi primer commit
2 files changed, 2 insertions(+)
 create mode 100644 Archivo A.txt
 create mode 100644 Archivo B.txt
```



Editor de texto

En vim(editor de texto de Linux), puedes pulsar i para empezar a introducir el texto y guardar pulsando las teclas **control + c** y posteriormente escribe :wq y enter , esto hará commit con el mensaje que escribiste. En su estado actual, para salir sin comprometerse, puede hacer :q en lugar del :wq como se ha mencionado anteriormente.

Si hacemos un git status vemos que ya no hay archivos que salvar y con git log observamos el primer commit que hicimos

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git status
On branch master
nothing to commit, working tree clean

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git log
commit b07ca2840673747dd274ffed546c926c6b8c6743 (HEAD -> master)
Author: luis <luis73web@gmail.com>
Date:   Fri Apr 2 13:02:52 2021 -0600

    MI PRIMER COMMIT

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ |
```



MINGW64:/c/Users/Luis/Desktop/Proyecto Git



On branch master

nothing to commit, working tree clean

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)

\$ git status

On branch master

nothing to commit, working tree clean

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)

\$ git log

commit 2ba6939290797ad681519be106a8d0845fef6a03 (HEAD -> master)

Author: luis <luis73web@gmail.com>

Date: Fri Apr 30 11:19:09 2021 -0600

nothing to commit, working tree clean



último
cambio

commit b07ca2840673747dd274ffed546c926c6b8c6743

Author: luis <luis73web@gmail.com>

Date: Fri Apr 2 13:02:52 2021 -0600



MI PRIMER COMMIT

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)

\$ |

```
> GIT INIT  
> GIT STATUS  
> GIT ADD XXXX  
> GIT ADD .  
> GIT COMMIT  
> GIT COMMIT -m XXXX  
> GIT CONFIG --GLOBAL user.email XXXX  
> GIT CONFIG --GLOBAL user.name XXXX  
> GIT LOG
```

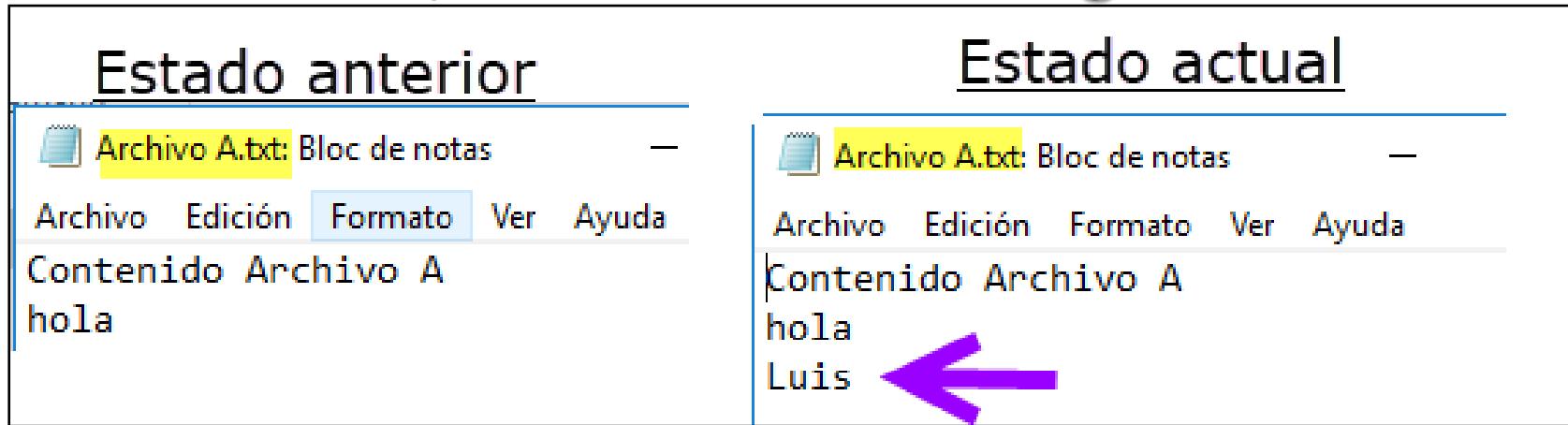
PROTEGER LOS CAMBIOS (INCLUIRLOS EN EL HEAD, NO EN EL REPOSITORIO REMOTO)

CONFIGURA TU INSTALACIÓN DE GIT

VER INFORMACIÓN SOBRE LOS COMMITS

git checkout -- <fichero>

Este comando permite deshacer los cambios en local antes de ejecutar el comando **git add**



```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (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:   Archivo A.txt

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

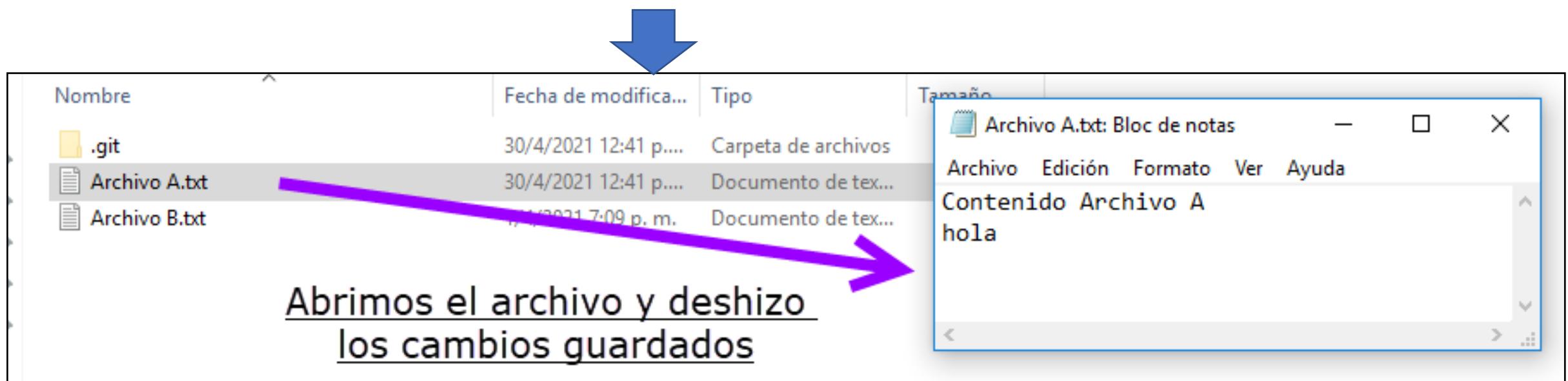
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$
```

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (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:   Archivo A.txt

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

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git checkout -- "Archivo A.txt"
```

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ |
```



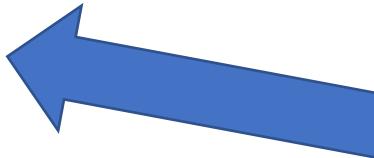
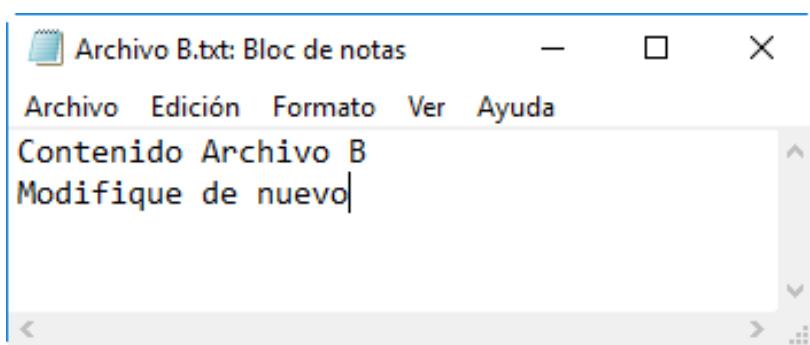
```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git status ←
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   Archivo B.txt
```

Al escribir git status nos muestra el mensaje que deshizo el cambio
(restore)

git diff [archivo]

Este comando sirve para ver las diferencias con los archivos anteriores de los archivos que **no se han añadido al área de ensayo ósea estando en local**. Este comando muestra exactamente los cambios que se han realizado dentro de los archivos. Se suele utilizar después de ver los cambios que ha habido en los archivos con el comando status. Puedes emplear git diff [archivo] para precisar la muestra.

git diff -- "Archivo B.txt"



Borre del archivo 27 de Mayo y agregue el texto
Modifique de nuevo

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (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:   Archivo B.txt

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

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$
```

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git diff -- "Archivo B.txt"
diff --git a/Archivo B.txt b/Archivo B.txt
index 1030f1c..853115e 100644
--- a/Archivo B.txt
+++ b/Archivo B.txt
@@ -1,2 +1,2 @@
-Contenido Archivo B
-27 de Mayo
+ No newline at end of file
+Modifique de nuevo
+ No newline at end of file

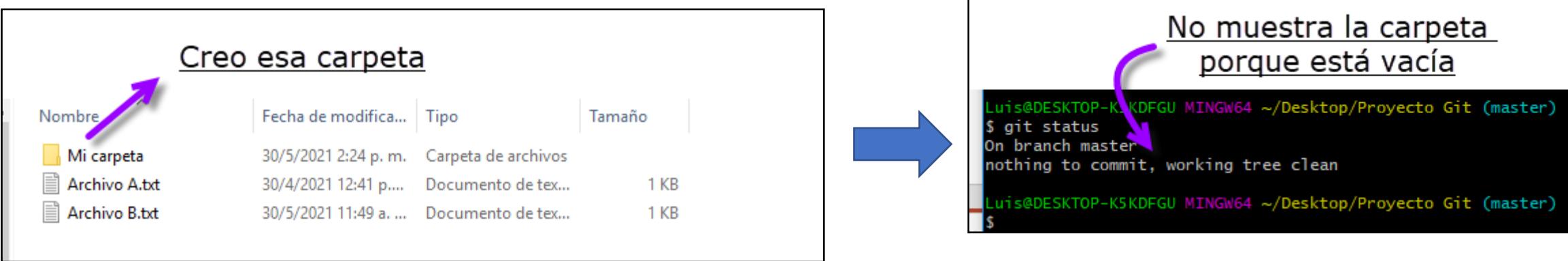
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ |
```

Como estaba anteriormente el contenido del archivo

Se agrego ese nuevo texto

.gitignore

Este archivo permite escribir los nombres de los archivos y carpetas que deseamos sean ignorados por git



nota

Las carpetas vacías son ignoradas por default por git

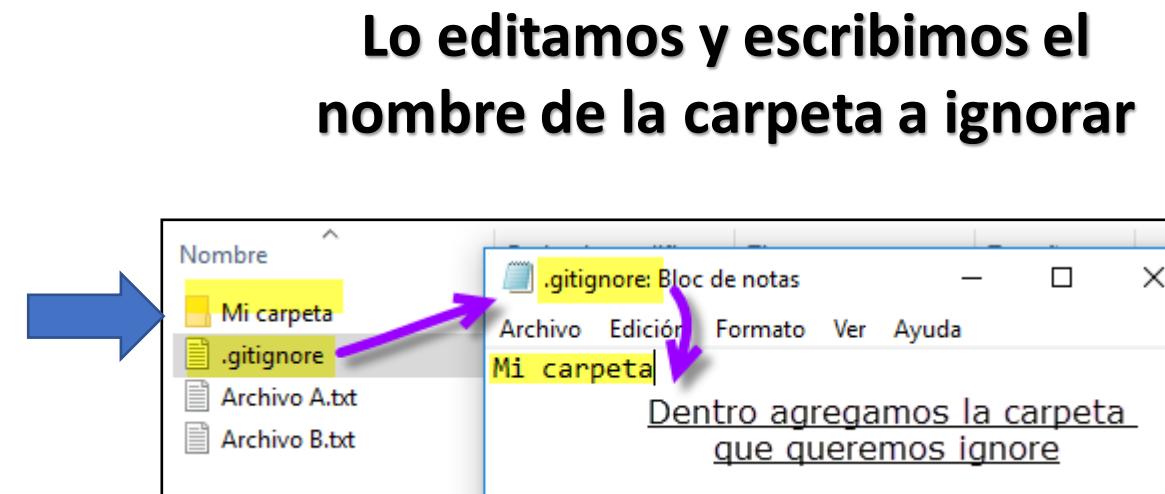
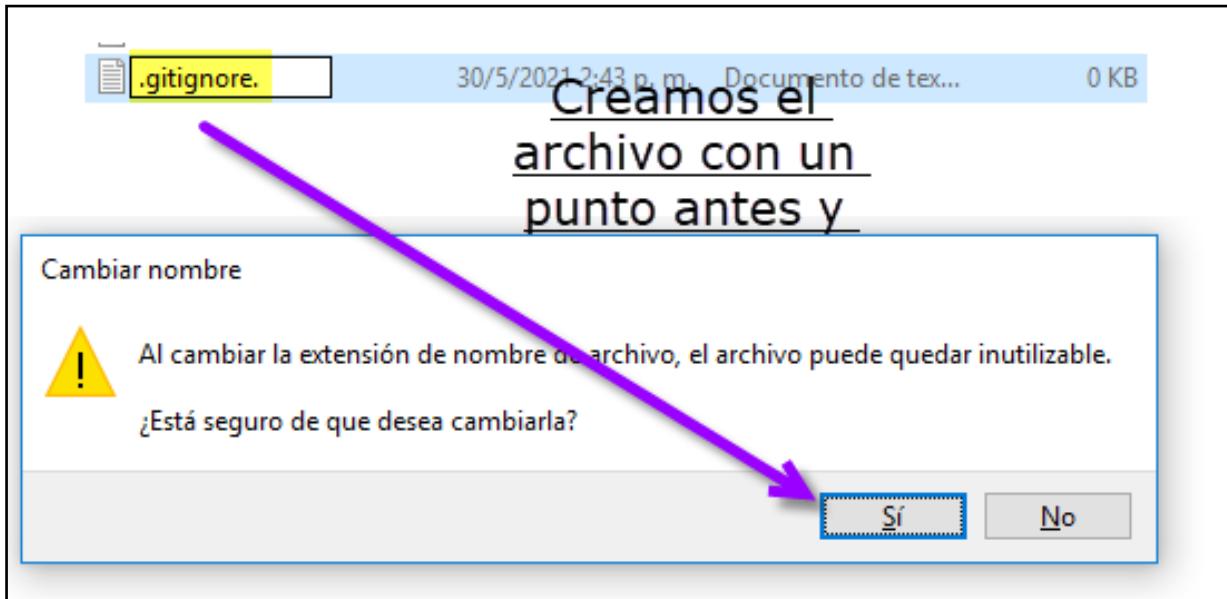
Proyecto Git > Mi carpeta

**Creo un archivo
dentro de la
carpeta**

Nombre
 Archivo C.txt

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git status ←
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Mi carpeta/ ←
nothing added to commit but untracked files present (use "git add" to tr
```

**Como ya hay
archivo dentro de
la carpeta ya se
muestra con git
status**



```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (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)

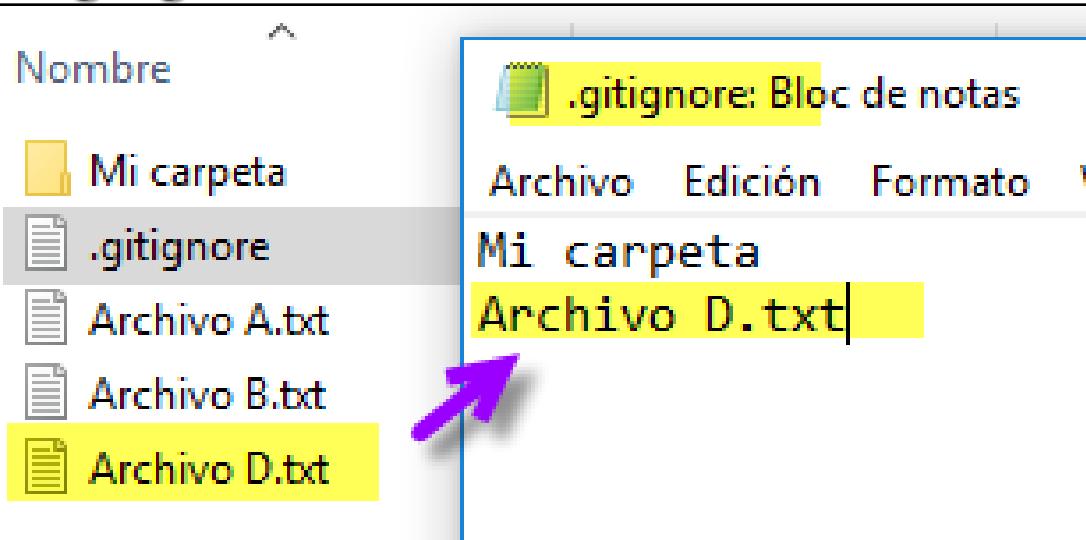
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ |
```

Ya no se ve la carpeta solo el archivo gitignore

Nombre	Fecha de modifica...
Mi carpeta	30/5/2021 2:28 p. m.
.gitignore	30/5/2021 2:41 p. m.
Archivo A.txt	30/4/2021 12:41 p....
Archivo B.txt	30/5/2021 11:49 a. ...
Archivo D.txt	30/5/2021 2:51 p. m.

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    Archivo D.txt
nothing added to commit but untracked files present (use "git add" to track)
```

Editamos el archivo .gitignore y agregamos el nuevo Archivo D.txt



**Ya no se ve el archivo
Es decir se ignoran carpetas y
archivos**

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (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
```

git commit -m “mensaje”

Este comando sirve para hacer un commit sin entrar al editor vim

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (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@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git add .gitignore ←
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .gitignore ←
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git commit -m "Esta es una prueba de comit sin entrar al editor vim"
[master 7cc22d9] Esta es una prueba de comit sin entrar al editor vim
  1 file changed, 2 insertions(+)
  create mode 100644 .gitignore

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$
```

Agregamos el archivo al área de estado

Hacemos commit sin necesidad de pasar por el editor Vim

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git log
commit 7cc22d913365e15353245f08f7618839116f04f9 (HEAD -> master)
Author: luis <luis73web@gmail.com>
Date:   Sun May 30 15:01:52 2021 -0600
      Esta es una prueba de comit sin entrar al editor vim

commit aeeaca2c47df717240343a94fb88999163156548b
Author: luis <luis73web@gmail.com>
Date:   Sun May 30 14:25:58 2021 -0600

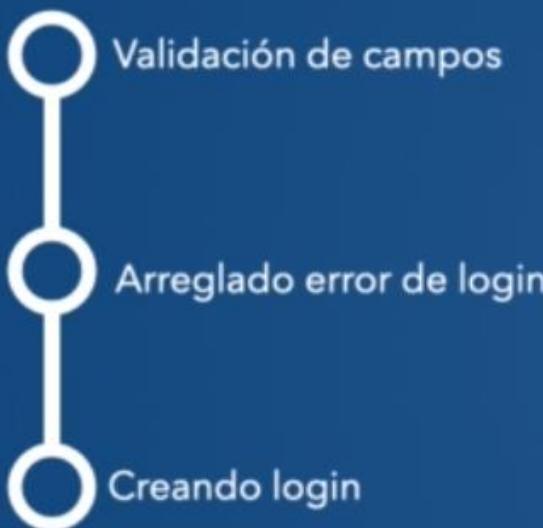
      CAMBIOS DE FICHERO B

commit 5660753f8b60a537425545874e0cee000722e6aa
Author: luis <luis73web@gmail.com>
Date:   Sun May 30 11:47:18 2021 -0600
      Se agregaron 5 lineas de texto hasta hola de nuevo

commit b43d4f2089fd171852ced19da65ce11df770380
Author: luis <luis73web@gmail.com>
Date:   Sun May 30 11:14:50 2021 -0600

      GUARDE EL 30 DE MAYO TEXTO HOLA MUNDO
```

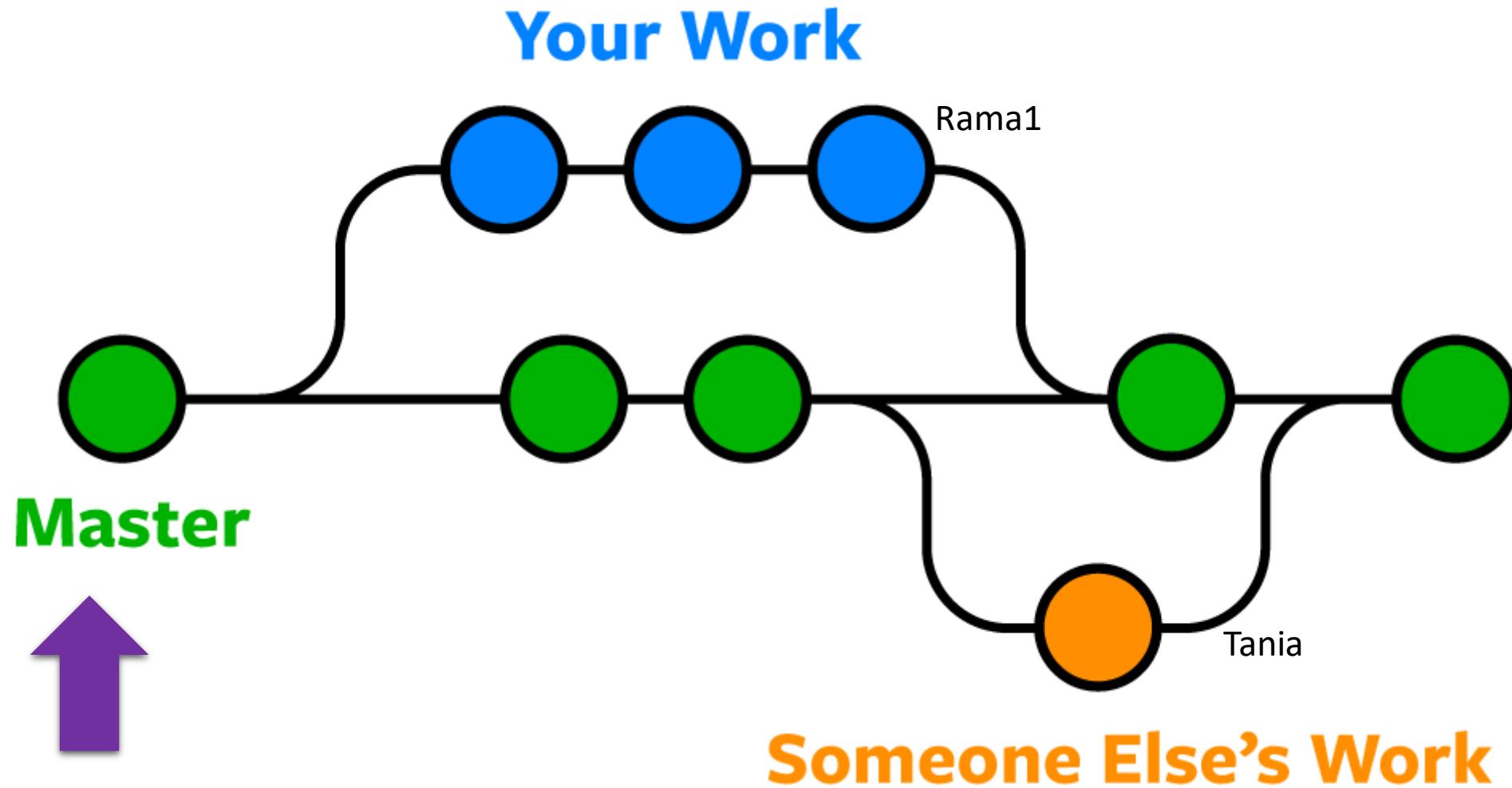
Ramas en Git



`git checkout -b registro`

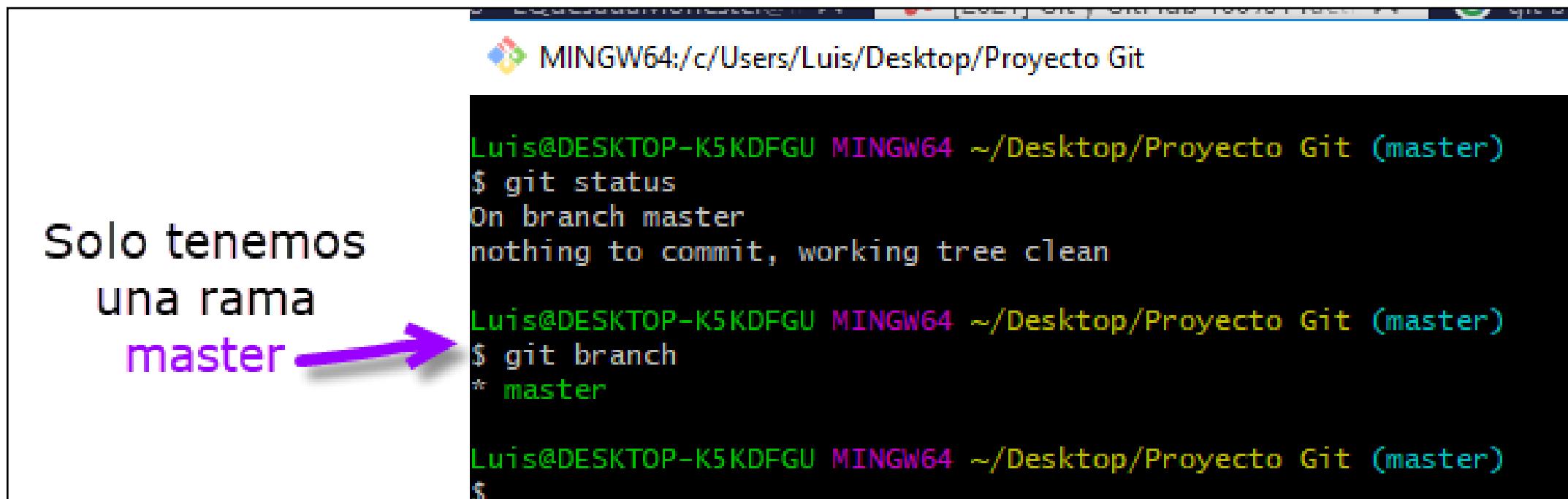


Trabajar con Ramas en Git



git branch

Este comando sirve para listar las ramas que tenemos, Por ejemplo, si quieres listar todas las ramas presentes en el repositorio, el comando debería verse así:



A screenshot of a terminal window titled "MINGW64;/c/Users/Luis/Desktop/Proyecto Git". The window shows the following command-line session:

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git status
On branch master
nothing to commit, working tree clean

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git branch
* master

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
```

The text "Solo tenemos una rama master" is overlaid on the left side of the terminal window, with a purple arrow pointing towards the "master" branch name in the "git branch" output.

nota

master esta en verde porque es la rama en la que estamos

git branch nombre de la rama

Este comando sirve para crear una nueva rama con el nombre que le hemos dado

Creamos la
rama 1
Pero aún
estamos con la
rama master
activa

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git branch ramal

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git branch
* master
  ramal
```

Git checkout nombre de la rama

Este comando sirve para cambiarnos a otra rama con el nombre que le hemos dado

con este
comando nos
cambiamos a
la rama **ramal**

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git checkout ramal
Switched to branch 'ramal'

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (ramal)
$ git branch
  master
* ramal
```

Nombre	Fecha de modifica...
Mi carpeta	30/5/2021 2:28 p. m.
.gitignore	30/5/2021 2:55 p. m.
Archivo A.txt	30/4/2021 12:41 p....
Archivo B.txt	30/5/2021 11:49 a. ...
Archivo D.txt	30/5/2021 2:51 p. m.
Archivo E.txt	30/5/2021 9:58 p. m.

Creamos un
nuevo archivo

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (ramal)
$ git status
On branch ramal
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Archivo E.txt

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

Hacemos
commit para
registrar el
cambio del
archivo creado
en la rama 1

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (rama1)
$ git add .

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (rama1)
$ git commit -m "Creación del Archivo E en la rama rama1"
[rama1 738fcfdc] Creación del Archivo E en la rama rama1
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Archivo E.txt
```

Nos cambiamos a la rama master observa que en la carpeta ya no esta el Archivo E

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (rama1)
$ git checkout master
Switched to branch 'master'

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$
```

Nombre

-  Mi carpeta
-  .gitignore
-  Archivo A.txt
-  Archivo B.txt
-  Archivo D.txt

Con git log podemos ver el último comit que hicimos a la rama master

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
$ git log
commit 7cc22d913365e15353245f08f7618839116f04f9 (HEAD -> master)
Author: luis <luis73web@gmail.com>
Date:   Sun May 30 15:01:52 2021 -0600
      Esta es una prueba de comit sin entrar al editor vim

commit aeeaca2c47df717240343a94fb88999163156548b
Author: luis <luis73web@gmail.com>
Date:   Sun May 30 14:25:58 2021 -0600

CAMBIOS DE FICHERO B

commit 5660753f8b60a537425545874e0cee000722e6aa
Author: luis <luis73web@gmail.com>
Date:   Sun May 30 11:47:18 2021 -0600
      Se agregaron 5 lineas de texto hasta hola de nuevo

commit b43d4f2089fd171852ced19da65ce11df770380
Author: luis <luis73web@gmail.com>
Date:   Sun May 30 11:14:50 2021 -0600
```

Nos cambiamos a la rama rama 1

Con el comando git log vemos el último commit

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (master)
```

```
$ git checkout ramal  
Switched to branch 'ramal'
```

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (ramal)
```

```
$ git log  
commit 738fcfdc825d48e98b442445ac47e4d7d3fa7c746 (HEAD -> ramal)  
Author: luis <luis73web@gmail.com>  
Date:   Sun May 30 22:08:36 2021 -0600
```

Creación del Archivo E en la rama ramal

```
commit 7cc22d913365e15353245f08f7618839116f04f9 (master)
```

```
Author: luis <luis73web@gmail.com>  
Date:   Sun May 30 15:01:52 2021 -0600
```

Esta es una prueba de comit sin entrar al editor vim

```
commit aeeaca2c47df717240343a94fb88999163156548b
```

```
Author: luis <luis73web@gmail.com>  
Date:   Sun May 30 14:25:58 2021 -0600
```

CAMBIOS DE FICHERO B

```
> GIT LOG  
> GIT CHECKOUT -- xxxx  
> GIT DIFF xxxx  
-> GIT BRANCH  
> GIT BRANCH xxxx  
> GIT CHECKOUT xxxx
```



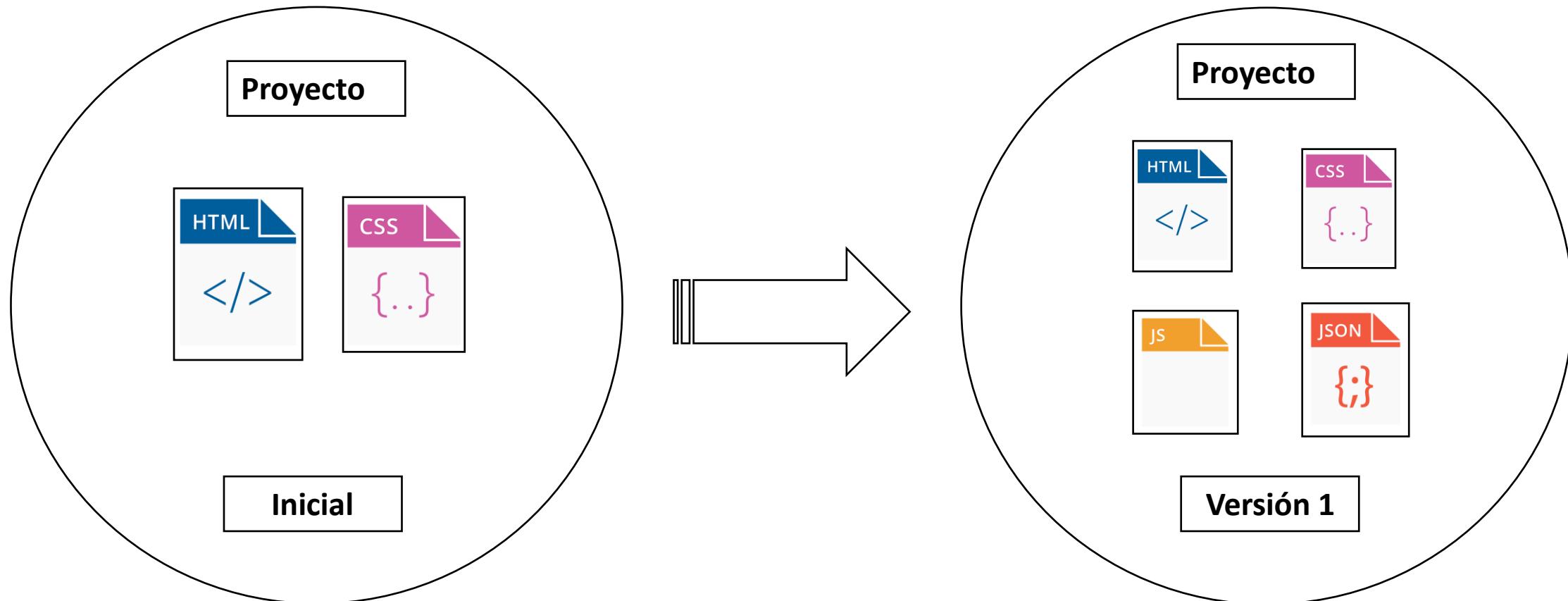
LISTAR RAMAS
CREAR RAMA
CAMBIAR DE RAMA

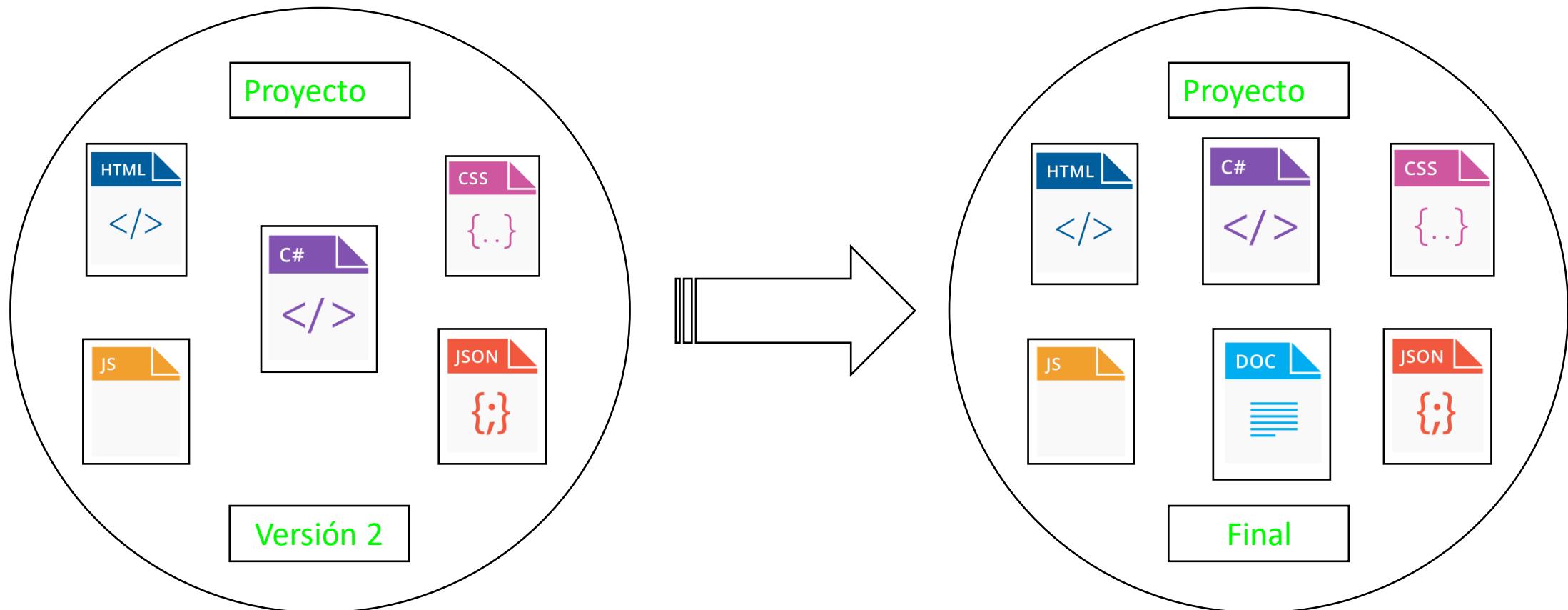
flujo de trabajo

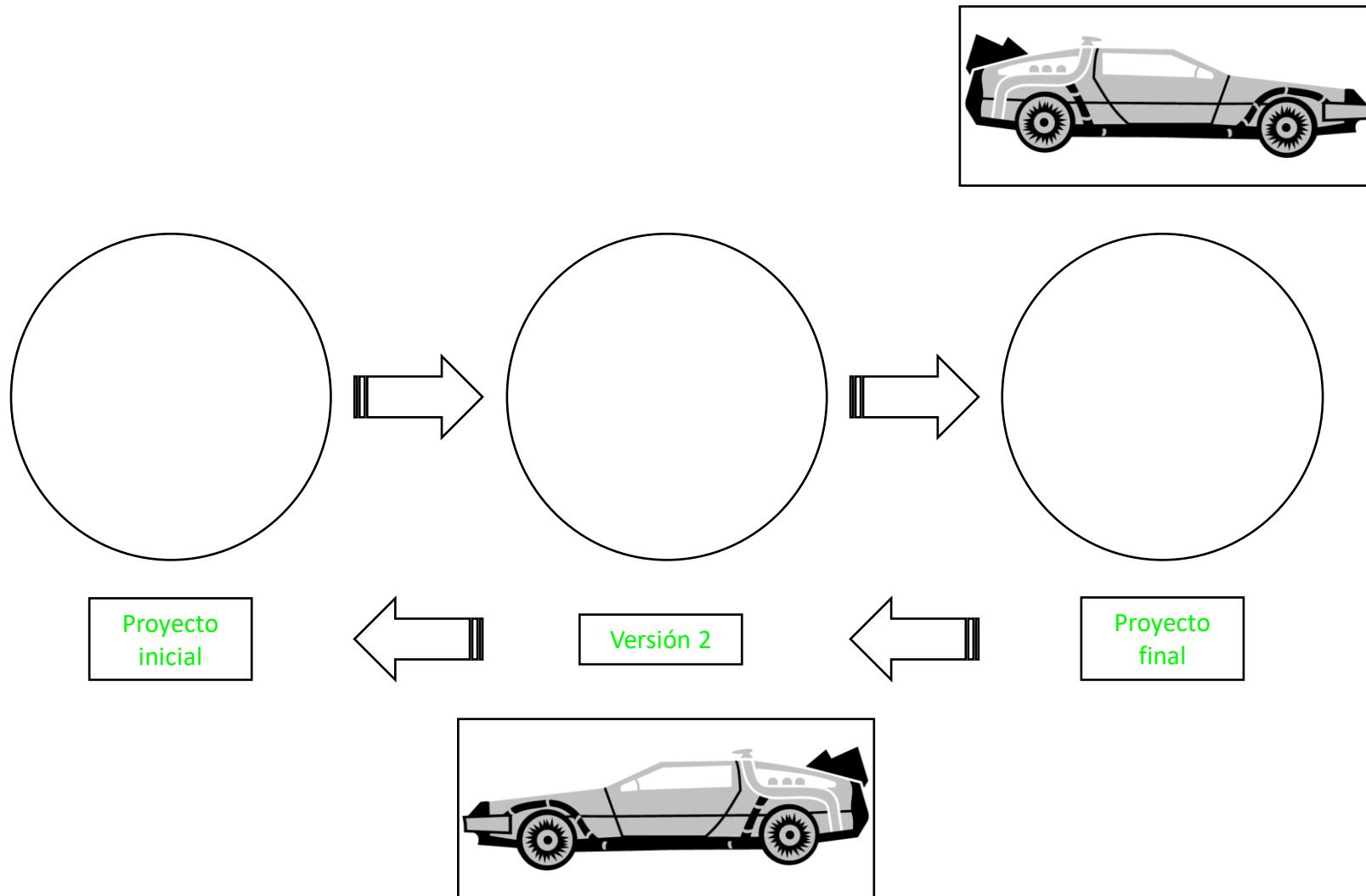
Tu repositorio local esta compuesto por tres "árboles" administrados por git. El primero es tu `Directorio de trabajo` que contiene los archivos, el segundo es el `Index` que actua como una zona intermedia, y el último es el `HEAD` que apunta al último commit realizado.



PROYECTOS

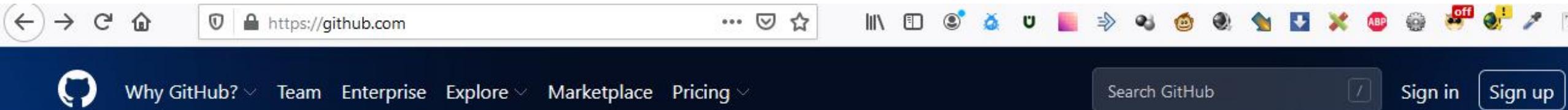






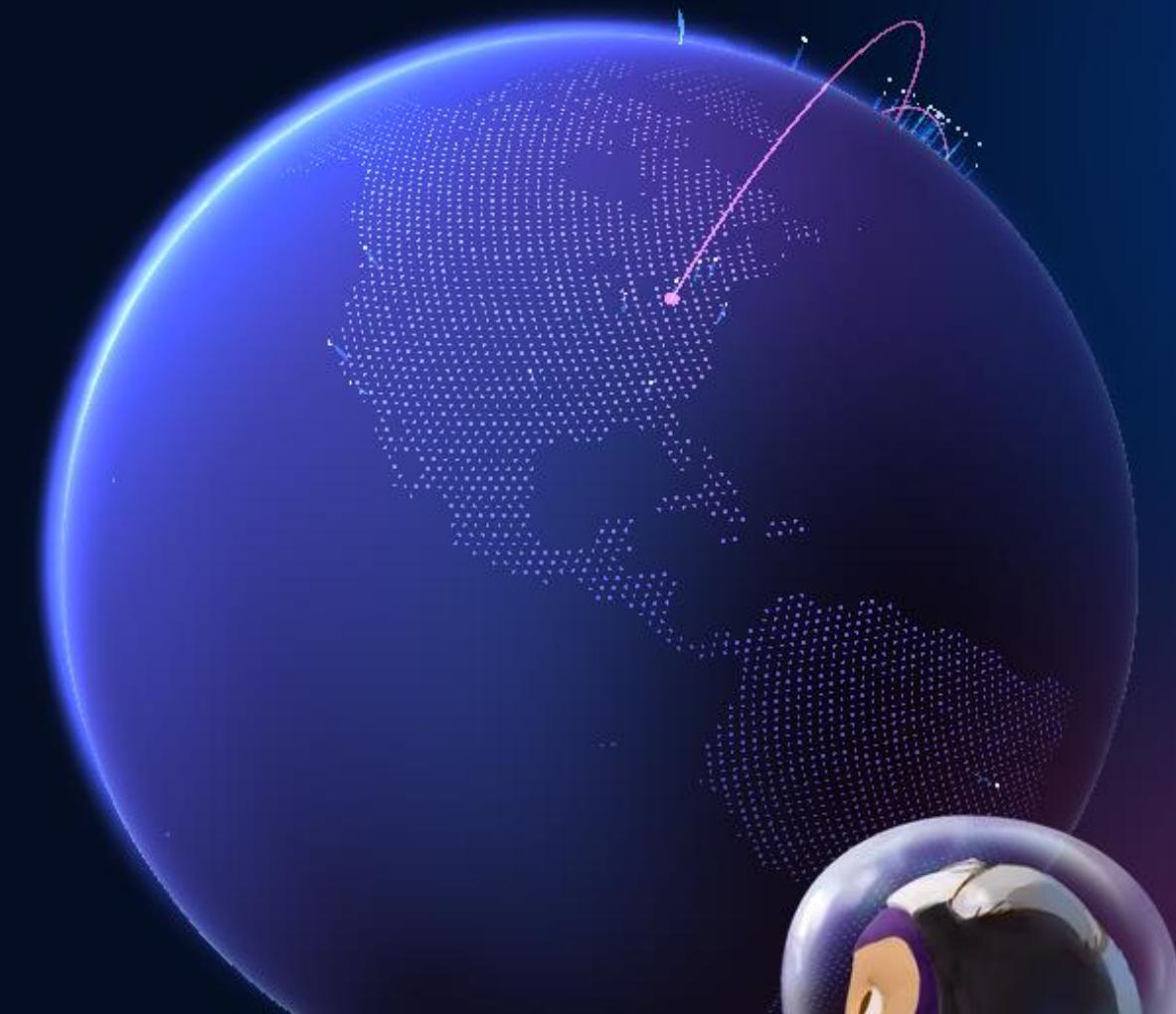


GitHub

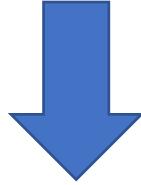


Where the world builds software

Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.

[Sign up for GitHub](#)

GitHub es un servicio basado en la nube que aloja un sistema de control de versiones (VCS) llamado Git. Éste permite a los desarrolladores colaborar y realizar cambios en proyectos compartidos, a la vez que mantienen un seguimiento detallado de su progreso.



Aloja más de 100 millones de repositorios

Planes

Free

The basics for individuals and organizations

- Unlimited public/private repositories
- 2,000 automation minutes/month
Free for public repositories
- 500MB of Packages storage
Free for public repositories
- Community support

\$ 0 per month

[Join for free](#)

MOST POPULAR

Team

Advanced collaboration for individuals and organizations

← Everything included in Free, plus...

- Protected branches
- Multiple reviewers in pull requests
- Draft pull requests
- Code owners
- Required reviewers
- Pages and Wikis
- 3,000 automation minutes/month
Free for public repositories
- 2GB of Packages storage
Free for public repositories
- Web-based support

\$ 4 per user/month

[Continue with Team](#)

Enterprise

Security, compliance, and flexible deployment

← Everything included in Team, plus...

- Automatic security and version updates
- SAML single sign-on
- Advanced auditing
- Github Connect
- 50,000 automation minutes/month
Free for public repositories
- 50GB of Packages storage
Free for public repositories

EXCLUSIVE ADD-ONS

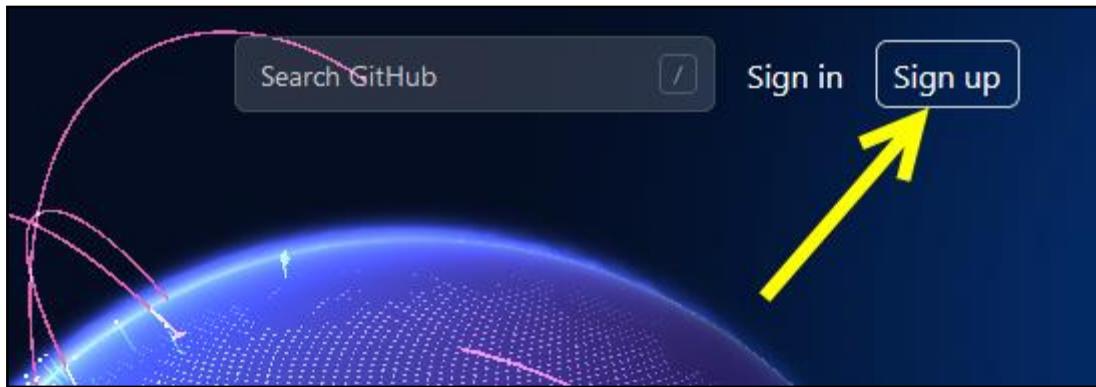
- Token, secret, and code scanning
- Premium support

\$ 21 per user/month

[Contact Sales](#)

[Start a free trial](#)

CREAR CUENTA EN GITHUB



Username *

 ⚠

Username luis73 is not available.
luis73-design, luis73-max, or luis7377 are available.

Ya se esta usando por otro usuario

Username *

 ✓

luis73web is available.

luis73web2@gmail.com ✓

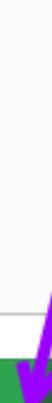
Password *

✓

Verify your account



Resolvemos el
captcha



Create account

By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails.

Welcome to GitHub

Woohoo! You've joined millions of developers who are doing their best work on GitHub. Tell us what you're interested in. We'll help you get there.

What kind of work do you do, mainly?

Software Engineer
I write code

Student
I go to school

Product Manager

UX & Design



Learn to code



Learn Git and GitHub



Host a project (repository)



Create a website with GitHub Pages



Collaborating with my team



Find and contribute to open source

I am interested in:

github X html X css X javascript X

languages, frameworks, industries

We'll connect you with communities and projects that fit your interests.

For example: activitypub security pov-ray

Complete setup



Please verify your email address

Before you can contribute on GitHub, we need you to verify your email address.

An email containing verification instructions was sent to luis73web2@gmail.com.

[Resend verification email](#)

[Change your email settings](#)

Llenamos las estadísticas si queremos

Chequeamos el Correo y aceptamos

 Principal

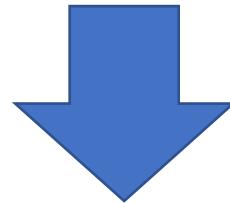
 Social

 Promociones **22 nuevos**
Coggle, Scribd, Lucidchart.com...

 GitHub

Nuevo

[GitHub] Please verify your email address. - Almost done, @luis73web! To complete your GitHub sign up, we just need to verify your email address.



[GitHub] Please verify your email address. Recibidos x



GitHub <noreply@github.com>
para mí ▾



Almost done, @luis73web!

To complete your GitHub sign up, we just need to verify your email address: luis73web2@gmail.com.

 Verify email address



https://github.com/join/get-started

... 🌐 ⭐

to... / Pull requests Issues Marketplace Explore

our email was verified.

Signed in as luis73web

Set status

Your profile

Your repositories

Your codespaces

Your projects

Your stars

Your gists

Upgrade

Feature preview

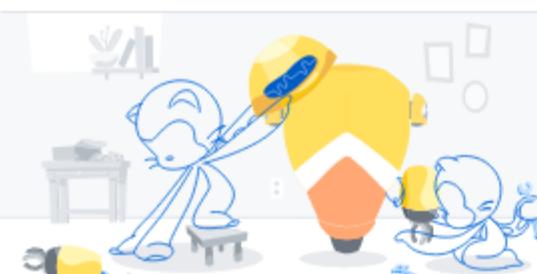
Help

Settings

Sign out

What do you want to do first?

Every developer needs to configure their environment, so let's get your GitHub experience optimized for you.

 Start a new project

Start a new repository or bring over an existing repository to keep contributing to it.

 Collaborate with your team

Improve the way your team works together and get access to more features with an organization.

 Learn how to use GitHub

Get started with an "Introduction to GitHub" course in our Learning Lab.

[Create a repository](#)

[Create an organization](#)

[Start Learning](#)



Search or jump to... / Pull requests Issues Marketplace Explore

Overview Repositories Projects Packages

Find a repository... Type Language Sort New

luis73web doesn't have any public repositories yet.

No hemos creado ningún repositorio

Crearemos un nuevo repositorio

luis73web

Edit profile

Joined 19 hours ago

Owner *



luis73web

Repository name *

mirepositorio



Great repository names are
Description (optional)

mirepositorio is available.

Need inspiration? How about [automatic-funicular](#)?

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.

Le damos un nombre
al repositorio

Los repositorios pueden ser
públicos o privados.

Los privados solo el creador lo
puede usar en el público
cualquier persona puede usarlo y
hacer commit siempre que el
propietario lo permita

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](#).

Add .gitignore

Choose which files not to track from a list of templates. [Learn more](#).

Choose a license

A license tells others what they can and can't do with your code. [Learn more](#).

Create repository

[Set up in Desktop](#)

or

[HTTPS](#)[SSH](#)<https://github.com/luis73web/mirepositorio.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 "# mirepositorio" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git branch -M main  
git remote add origin https://github.com/luis73web/mirepositorio.git  
git push -u origin main
```

Copiamos esas instrucción y la ejecutamos en nuestra terminal para subir mis archivos a github

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

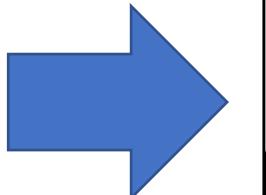
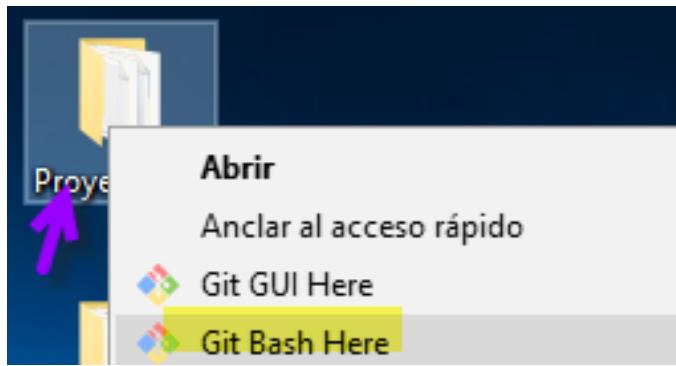
```
git remote add origin https://github.com/luis73web/mirepositorio.git  
git branch -M main  
git push -u origin main
```

...or import code from another repository

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

[Import code](#)

En mi pc busco la carpeta del mi proyecto y llamo la terminal



```
MINGW64:/c/Users/Luis/Desktop/Proyecto Git
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (rama1)
$ git remote add origin https://github.com/luis73web/mirepositorio.git

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (rama1)
$ git push -u origin master
```

Pegamos la primera instrucción y le damos enter

Pegamos la segunda instrucción y cambiamos main por la rama **master**

Connect to GitHub

GitHub

Sign in

[Sign in with your browser](#)

or

Personal Access Token

[Sign in](#)

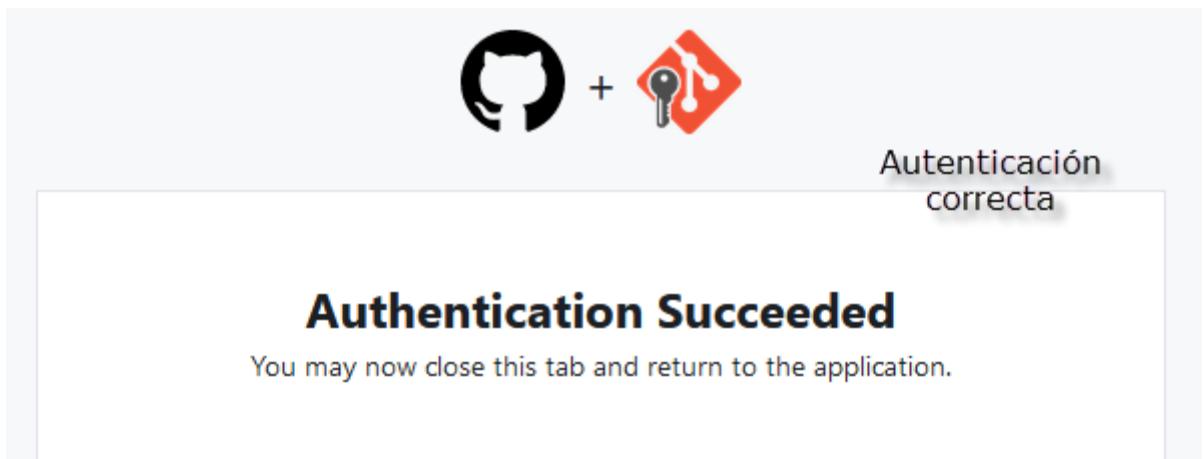
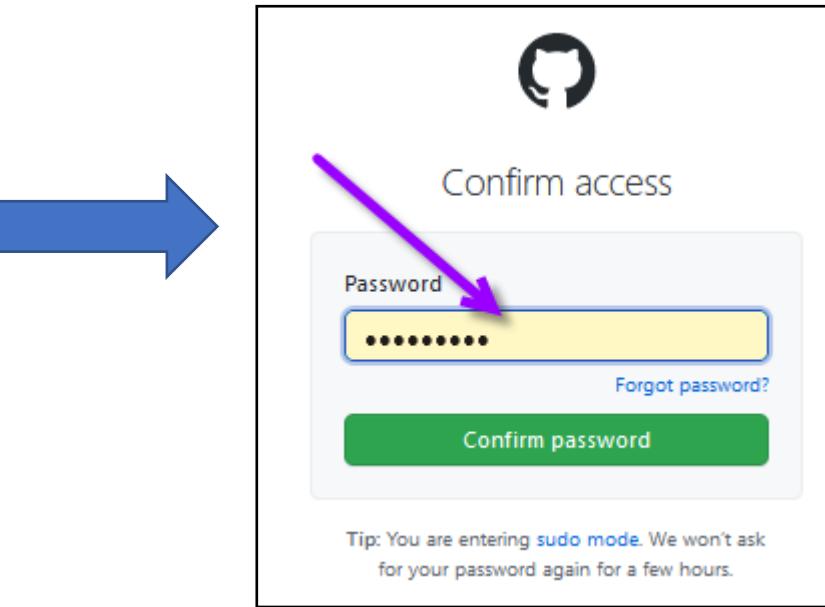
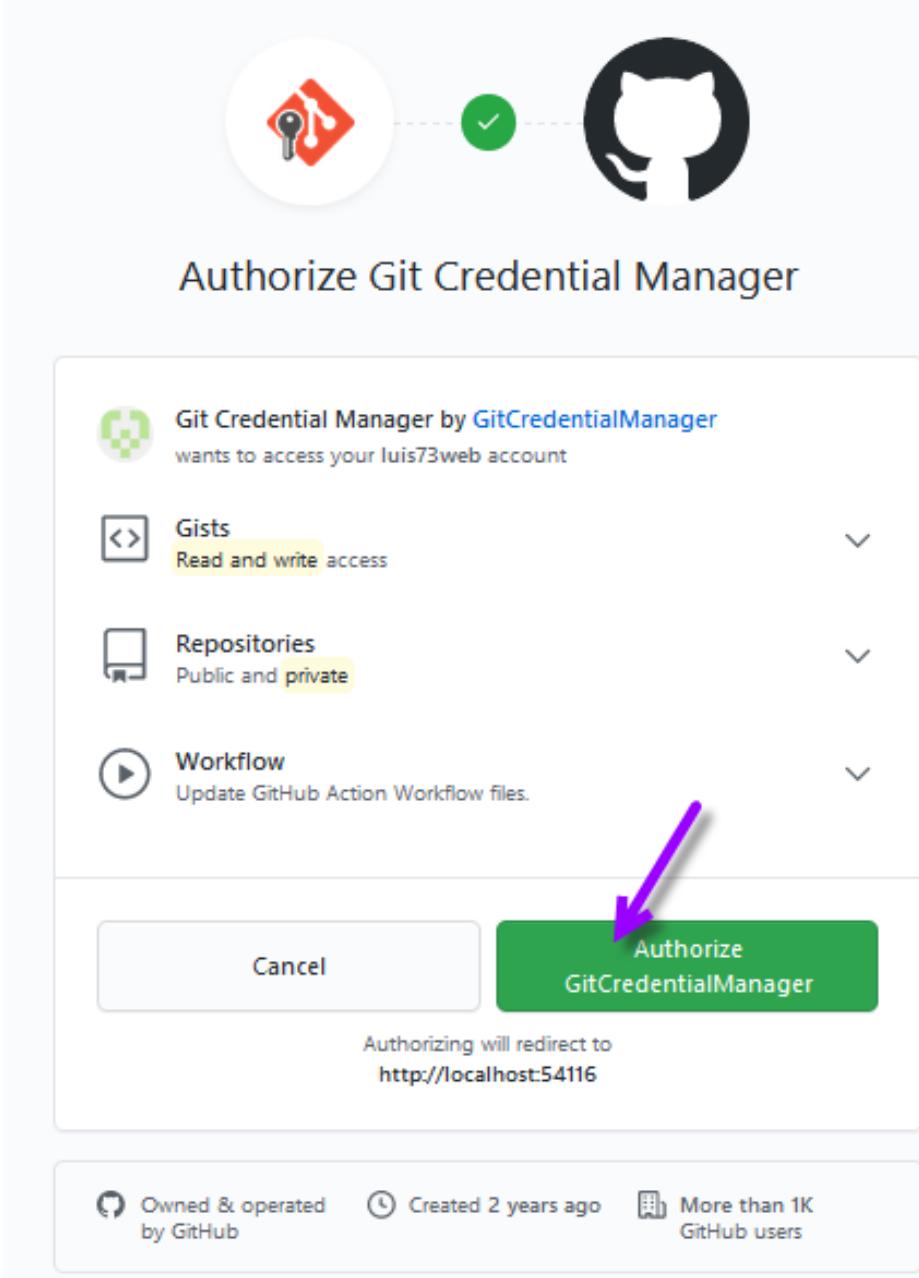
Don't have an account? [Sign up](#)



Eliminar conexión remota SOLO si hay problema de conexión

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Mi proyecto (master)
$ git remote -v
origin  https://github.com/lqm73/Mi-proyecto-web.git (fetch)
origin  https://github.com/lqm73/Mi-proyecto-web.git (push)

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Mi proyecto (master)
$ git remote rm origin
```



En la terminal se muestra que ha subido todos los archivos

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (rama1)
$ git push -u origin master
Enumerating objects: 25, done.
Counting objects: 100% (25/25), done.
Delta compression using up to 4 threads
Compressing objects: 100% (19/19), done.
Writing objects: 100% (25/25), 2.35 KiB | 141.00 KiB/s, done.
Total 25 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/luis73web/mirepositorio.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/Proyecto Git (rama1)
$ |
```

The screenshot shows a GitHub repository page for 'luis73web/mirepositorio'. A purple arrow points from the repository name in the browser tab down to the repository name in the header. Another purple arrow points from the 'Code' button in the navigation bar down to the 'Code' dropdown in the top right. A third purple arrow points from the 'Issues' button in the navigation bar down to the commit list. A fourth purple arrow points from the 'Code' section in the bottom right down to the 'Add a README' button. A large gray callout box on the left contains the text: 'Vemos los archivos y carpetas si hubieran que se han subido al repositorio'.

luis73web/mirepositorio

https://github.com/luis73web/mirepositorio 80%

Search or jump to... Pull requests Issues Marketplace Explore

luis73web / mirepositorio

<> **Code** Issues Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags Go to file Add file Code

Vemos los archivos y carpetas si hubieran que se han subido al repositorio

luis Esta es una prueba de comit sin entrar al editor vim 7cc22d9 2 days ago 8 commits
.gitignore Esta es una prueba de comit sin entrar al editor vim 2 days ago
Archivo A.txt nothinwpw to commit, working tree clean last month
Archivo B.txt CAMBIOS DE FICHERO B 2 days ago

Add a README

 master

 1 branch

 0 tags

[Go to file](#)

[Add file](#)

 [Code](#)

luis Esta es una prueba de comit sin entrar al editor vim

7cc22d9 2 days ago

 8 commits

 .gitignore

Esta es una prueba de comit sin entrar al editor vim

2 days ago

 Archivo A.txt

nothinwpw to commit, working tree clean

Archivo de
texto con
información
del proyecto

last month

 Archivo B.txt

CAMBIOS DE FICHERO B

2 days ago

Help people interested in this repository understand your project by adding a README.

[Add a README](#)



Lenguaje MarkDown

Es un lenguaje que se utiliza para dar formato al texto, se lo aplicaremos al archivo Readme

A screenshot of a Google search results page. The search bar at the top contains the query "markdown syntax". Below the search bar, there are navigation links for "Todo", "Imágenes", "Videos", "Noticias", "Libros", and more. A purple arrow points to the first search result, which is a link to "Basic Syntax | Markdown Guide" on www.markdownguide.org.

Markdown	HTML	Rendered Output
This text is ***really important***.	Copiamos This text is really important.	This text is <i>really important</i> .
This text is __really important__.	This text is really important.	This text is <i>really important</i> .

The image shows a GitHub interface with two main sections: an editor on the left and a preview on the right.

Editor (Left):

- Path: mirepositorio / README.md in master
- Buttons: <> Edit new file (highlighted), Preview
- Content:

```
1 # mirepositorio
2
3 Este repositorio es para realizar pruebas de GitUp
4 Junto a los estudiantes
5
6 Uso ***Markdownt***.
```
- A purple arrow points from the word "Pegar" to the line "Uso ***Markdownt***."

Preview (Right):

- Buttons: <> Edit new file, Preview (highlighted)
- Content:

mirepositorio

Este repositorio es para realizar pruebas de GitUp Junto a los estu

Uso *Markdownt*.
- A purple arrow points to the word "Uso *Markdownt*".

[Edit new file](#)[Preview](#)

mirepositorio

Este repositorio es para realizar pruebas de GitUp Junto a los estudiantes
Uso *Markdownt*.



Commit new file

[Create README.md](#)[Add an optional extended description...](#)

⏪ Commit directly to the `master` branch.

⏪ Create a new branch for this commit and start a pull request. [Learn more about pull requests](#)

[Commit new file](#)[Cancel](#)

Ya se agregó al repositorio

The screenshot shows a GitHub repository page for 'luis73web / mirepositorio'. The 'Code' tab is selected. The repository has 1 branch and 0 tags. The commit history shows:

- luis73web Create README.md (bbf0e53, now) - 9 commits
- .gitignore (2 days ago)
- Archivo A.txt (last month)
- Archivo B.txt (2 days ago)
- README.md (now)

The README.md file content is:

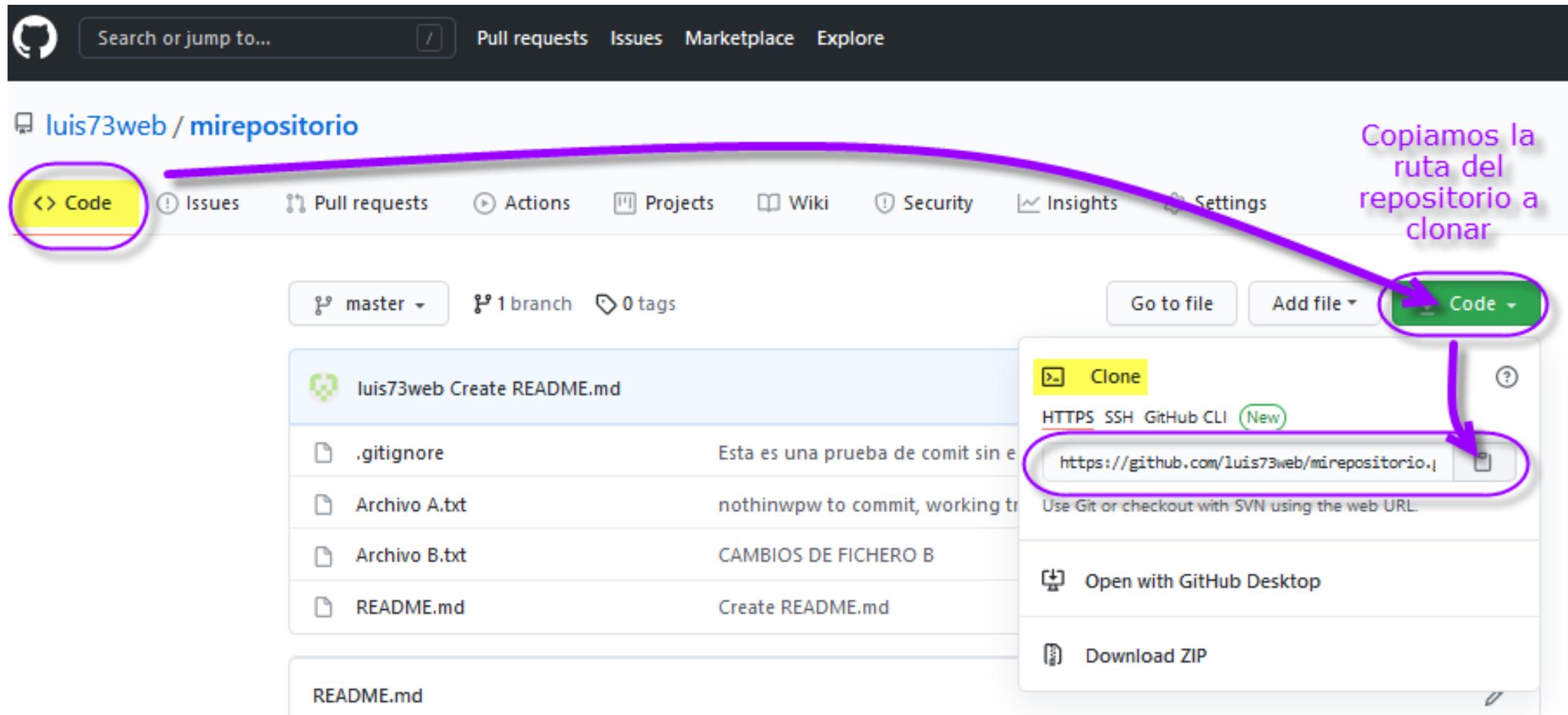
Este repositorio es para realizar pruebas de GitUp Junto a los estudiantes

Uso Markdownt.

A purple arrow points from the top left towards the README.md file, and another purple arrow points from the bottom right towards the 'Uso Markdownt.' text.

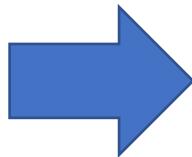
git clone [ruta de repositorio]

Este comando sirve descargar desde github a mi computadora o cualquier máquina el repositorio

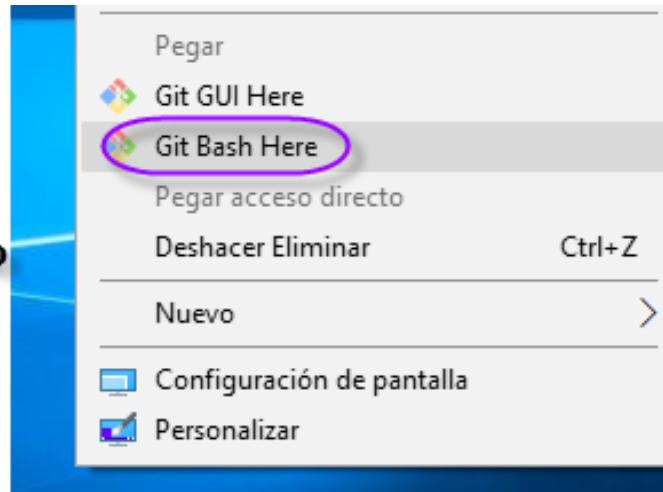




Borro la carpeta de mi repositorio en mi PC



En el escritorio



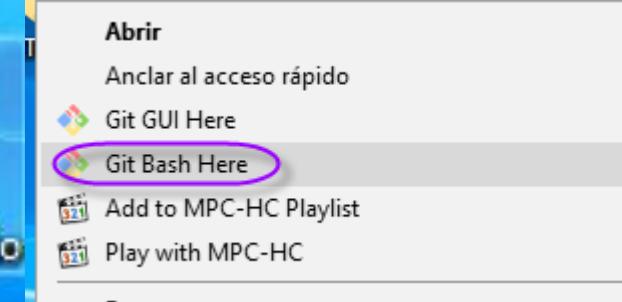
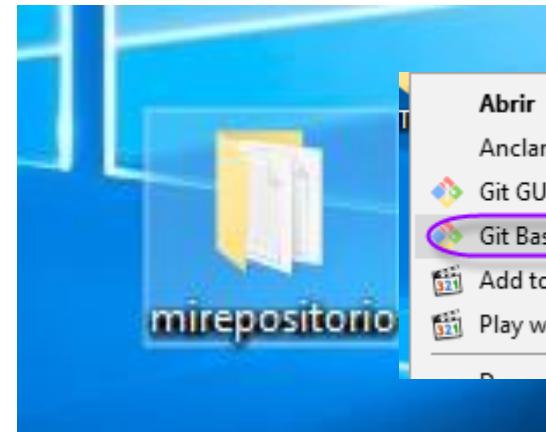
Antepongo el comando git clone y pego la ruta con clic derecho

```
MINGW64:/c/Users/Luis/Desktop
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop
$ git clone https://github.com/luis73web/mirepositorio.git
Cloning into 'mirepositorio'...
remote: Enumerating objects: 28, done.
remote: Counting objects: 100% (28/28), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 28 (delta 3), reused 24 (delta 2), pack-reused 0
Receiving objects: 100% (28/28), done.
Resolving deltas: 100% (3/3), done.

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop
$ |
```

 luis73web / mirepositorio

El repositorio se
restaura con el nombre
que tiene en Git hub



Archivos
restaurados

Todos los
commit
también se
restauran

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ ls
'Archivo A.txt' 'Archivo B.txt' README.md

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ git log
commit bbf0e5397f37264dc27e3caf9ac48243d9c6a28b (HEAD -> master, origin/master,
origin/HEAD)
Author: Luis73web <85135670+luis73web@users.noreply.github.com>
Date:   Tue Jun 1 15:29:57 2021 -0600

    Create README.md

commit 7cc22d913365e15353245f08f7618839116f04f9
Author: Luis <luis73web@gmail.com>
Date:   Sun May 30 15:01:52 2021 -0600

    Esta es una prueba de comit sin entrar al editor vim

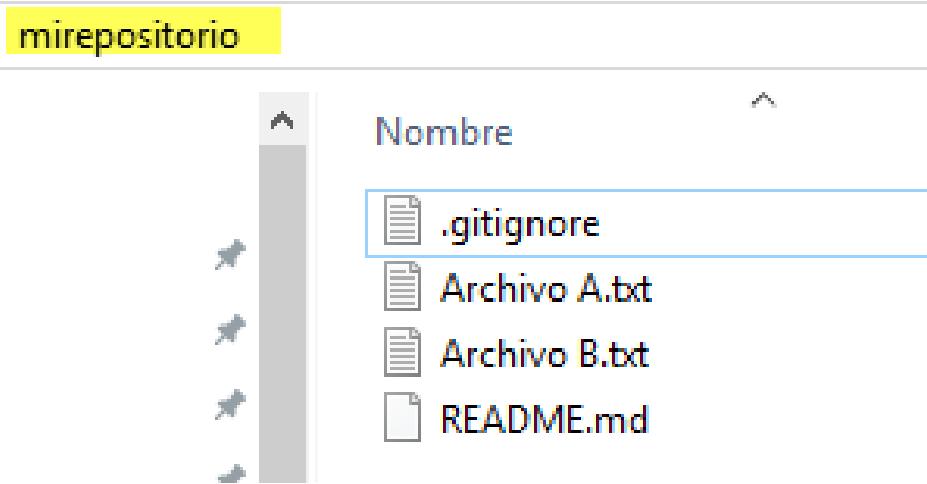
commit aeeaca2c47df717240343a94fb88999163156548b
Author: Luis <luis73web@gmail.com>
Date:   Sun May 30 14:25:58 2021 -0600

    CAMBIOS DE FICHERO B

commit 5660753f8b60a537425545874e0cee000722e6aa
Author: Luis <luis73web@gmail.com>
```

git pull

Este comando sirve para actualizar desde github mi repositorio local



```
MINGW64:/c/Users/Luis/Desktop/mirepositorio
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ git status ←
On branch master
Your branch is up to date with 'origin/master'.
nothing to commit, working tree clean

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ ls ←
'Archivo A.txt' 'Archivo B.txt' README.md

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ |
```

Mi repositorio local

master 1 branch 0 tags

Go to file Add file ▾

Create new file

Upload files

luis BORRE CARPETA

.gitignore Esta es una prueba de comit sin entrar al editor vim

Archivo A.txt nothinwpw to commit, working tree clean

Archivo B.txt CAMBIOS DE FICHERO B

README.md Create README.md

mirepositorio /

README.md

Drag additional files here to add them to your repository

Or choose your files

Archivo X.txt

Commit changes

Add files via upload

Add an optional extended description...

-o Commit directly to the master branch.

! Create a new branch for this commit and start a pull request. Learn more about pull requests.

Commit changes Cancel

Archivo X.txt

Mi repositorio en git hub, se sube un archivo Archivo X.txt

master

1 branch

0 tags

Go to file

Add file

Code



luis73web Add files via upload

bb27d2b now 12 commits

.gitignore

Esta es una prueba de comit sin entrar al editor vim

2 days ago

Archivo A.txt

nothinwpw to commit, working tree clean

last month

Archivo B.txt

CAMBIOS DE FICHERO B

2 days ago

Archvio X.txt

Add files via upload

now

README.md

Create README.md

7 hours ago

README.md



mirepositorio

Este repositorio es para realizar pruebas de GitUp Junto a los estudiantes

Uso *Markdown*.

Actualizamos mi repositorio local desde github

```
nothing to commit, working tree clean

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ ls
'Archivo A.txt'  'Archivo B.txt'  README.md

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 658 bytes | 8.00 KiB/s, done.
From https://github.com/luis73web/mirepositorio
  0636afb..bb27d2b  master      -> origin/master
Updating 0636afb..bb27d2b
Fast-forward
 Archvio X.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 Archvio X.txt

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ ls
'Archivo A.txt'  'Archivo B.txt'  'Archvio X.txt'  README.md

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ |
```

 .gitignore

 Archivo A.txt

 Archivo B.txt

 README.md

 Archvio X.txt

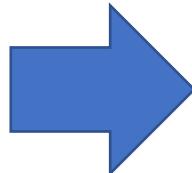
git push

Este comando sirve para actualizar github desde mi repositorio local

Agregue esa
carpeta y adentro
cree un archivo F

Trabajos
.gitignore
Archivo A.txt
Archivo B.txt
Archvio X.txt
README.md

Mi repositorio local



```
MINGW64:/c/Users/Luis/Desktop/mirepositorio
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ git status ←
On branch master
Your branch is up to date with 'origin/master'.

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

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

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ git add . ←
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ git commit -m "Agregue carpeta Trabajos"
[master 6c7269f] Agregue carpeta Trabajos
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Trabajos/Archivo F.txt

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ git status ←
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ git push ←
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 324 bytes | 162.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/luis73web/mirepositorio.git
  bb27d2b..6c7269f  master -> master

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$
```

Actualizamos mi
repositorio remoto
en github desde
repositorio local

luis73web / mirepositorio

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags

luis Agregue carpeta Trabajos

Trabajos

Refrescamos la ventana y podemos observar el último commit y la carpeta agregada

File	Message	Time
.gitignore	Esta es una prueba de comit sin entrar al editor vim	2 days ago
Archivo A.txt	nothinwpw to commit, working tree clean	last month
Archivo B.txt	CAMBIOS DE FICHERO B	2 days ago
Archvio X.txt	Add files via upload	13 minutes ago
README.md	Create README.md	7 hours ago

VOLCAR DE UNA RAMA CUALQUIERA A LA MASTER

Este comando sirve para actualizar github desde mi repositorio local

MINGW64:/c/Users/Luis/Desktop/mirepositorio

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.
```

```
nothing to commit, working tree clean
```

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ git branch
* master
```

Observamos las ramas

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ git branch secundaria
```

Creamos la rama secundaria

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ git branch
* master
secundaria
```

Observamos las ramas ya esta creada

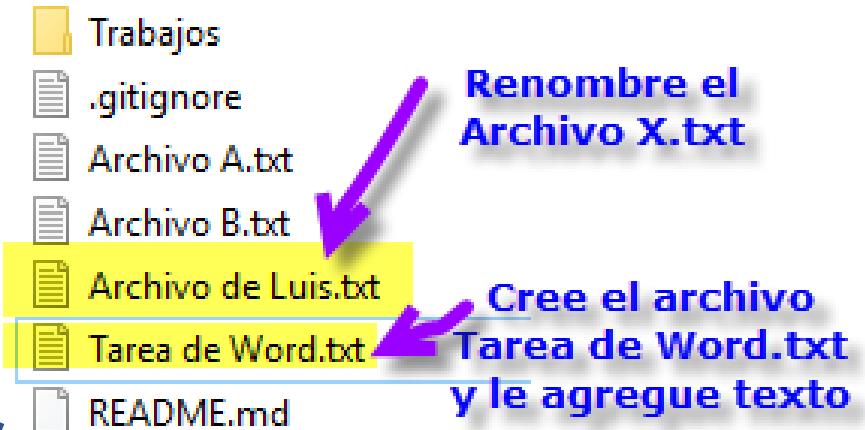
```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ git checkout secundaria
Switched to branch 'secundaria'
```

Nos cambiamos de rama

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (secundaria)
$ |
```

Mi repositorio local

En la rama secundaria hacemos cambios: agrego archivo y renombro otro



```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (secundaria)
$ git status ←
On branch secundaria
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:  Archvio X.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Archivo de Luis.txt
    Tarea de Word.txt

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

Hacemos commit a los cambios

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (secundaria)
$ git add . ←
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (secundaria)
$ git commit -m "Creación de archivo y cambio de nombre al Archivo X.txt"
[secundaria ee55edc] Creación de archivo y cambio de nombre al Archivo X.txt
 2 files changed, 1 insertion(+)
 rename Archvio X.txt => Archivo de Luis.txt (100%)
 create mode 100644 Tarea de Word.txt
```

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (secundaria)
$
```

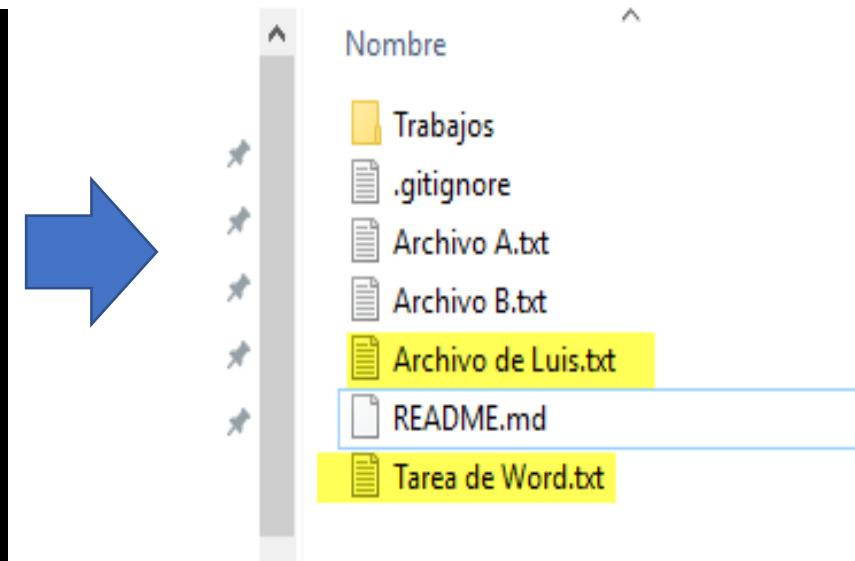
git merge [rama que nos traemos]

Este comando sirve fusionar ramas

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ git merge secundaria ← Estando en la rama master nos
Updating 6c7269f..ee55edc   traemos su contenido de la rama
Fast-forward
 Archvio X.txt => Archivo de Luis.txt | 0
 Tarea de Word.txt           | 1 +
 2 files changed, 1 insertion(+)
 rename Archvio X.txt => Archivo de Luis.txt (100%)
 create mode 100644 Tarea de Word.txt

Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ |
```

mirepositorio



```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ git branch
* master
  secundaria
```

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ git diff secundaria master
```

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ |
```

Podemos ver que
existen las dos
ramas pero al
compararlas ya no
hay diferencias

git branch -d [rama]

Este comando sirve para borrar una rama

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ git branch -d secundaria
Deleted branch secundaria (was ee55edc).
```

```
Luis@DESKTOP-K5KDFGU MINGW64 ~/Desktop/mirepositorio (master)
$ git branch
* master
```

Borramos la rama secundaria



Invitar colaboradores a un repositorio personal

Puedes invitar usuarios para convertir colaboradores de tu repositorio personal.

Si estás utilizando GitHub Free, puedes agregar colaboradores ilimitados en repositorios públicos y privados.

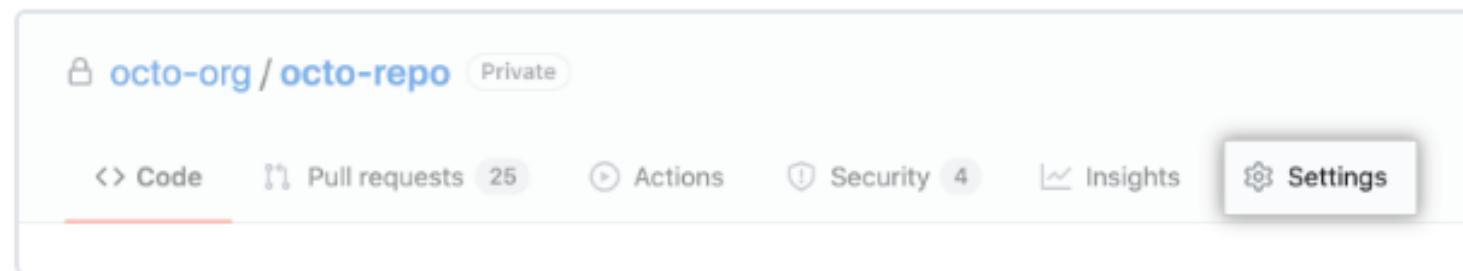
Los repositorios que son propiedad de una organización pueden conceder acceso más pormenorizado. Para obtener más información, consulta "[Permisos de acceso en GitHub](#)".

Las invitaciones pendientes caducarán después de 7 días. Esto restablecerá cualquier licencia sin reclamar.

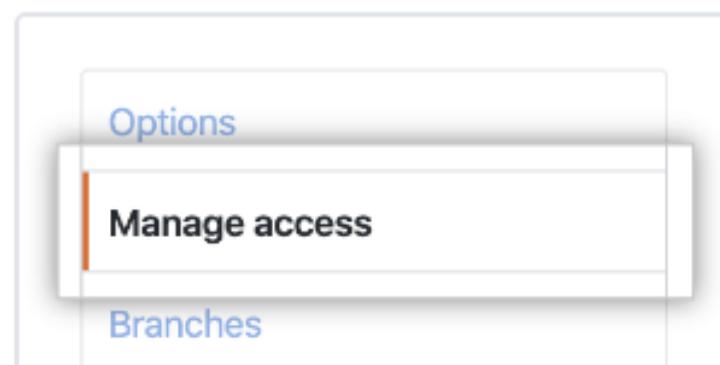
Nota: GitHub limita la cantidad de personas que se pueden invitar a un repositorio dentro de un período de 24 horas. Si excedes este límite, espera 24 horas o crea una organización para colaborar con más personas.

- 1 Solicita el nombre de usuario de la persona que estás invitando como colaboradora.

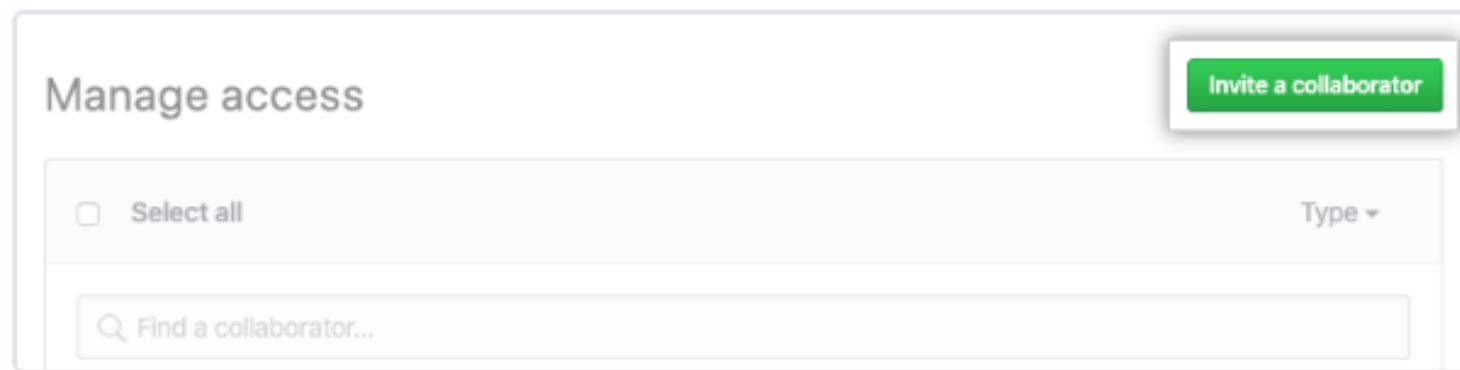
Si todavía no tiene un nombre de usuario, puede registrarse en GitHub. Para obtener más información, consulta "[Registrar una nueva cuenta de GitHub](#)".
- 2 En GitHub, visita la página principal del repositorio.
- 3 Debajo de tu nombre de repositorio, da clic en **Configuración**.



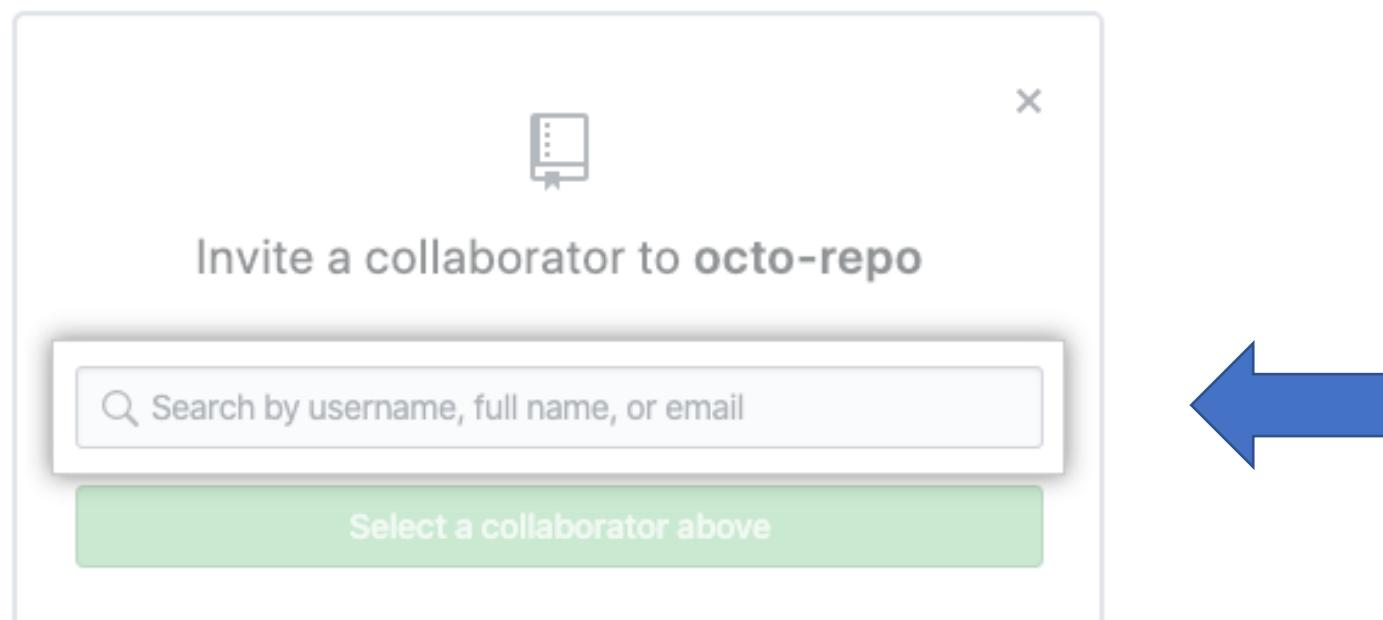
- 4 En la barra lateral izquierda, da clic en **Administrar acceso**.



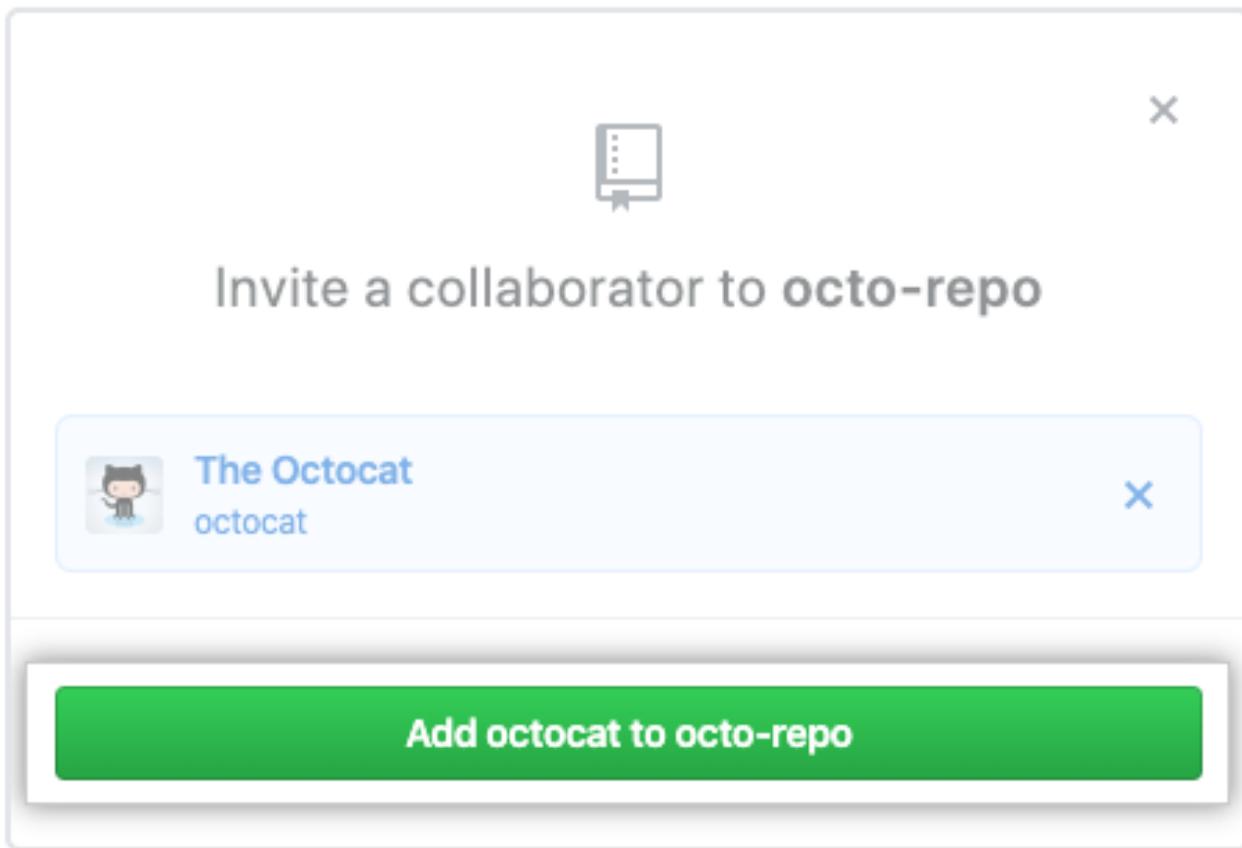
- 5 Da clic en **Invitar un colaborador**.



- 6 En el campo de búsqueda, comienza a teclear el nombre de la persona que quieras invitar, luego da clic en un nombre de la lista de resultados.



- 7 Da clic en Añadir NOMBRE a REPOSITORIO.



- 8 El usuario recibirá un correo electrónico invitándolo al repositorio. Una vez que acepte la invitación, tendrá acceso de colaborador a tu repositorio.

Usuario: **lqm73**

Correo: **luis73web@gmail.com**