

Registrarse en el staging área

Crear un Fichero

Crear un nuevo archivo en formato **markdown**.

En primer lugar, vaya al directorio git.

```
$ cd ~/workspace/secundario/text/git
```

A continuación, crea un archivo **sample_memo.md**.

```
$ touch sample_memo.md
```

Utilice el comando echo para añadir el texto "## hola mundo" al archivo **sample_memo.md**.

```
$ echo "## hello world" >> sample_memo.md
```

Ahora vamos a comprobar el estado de Git.

Para comprobar el estado de Git, utiliza el comando **git status**.

Ejecute el siguiente comando.

```
$ git status
```

Debería verse como sigue.

```
$ git status
On branch master

No commits yet

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

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

SECUNDARIO@DESKTOP-FBIM49G MINGW64 ~/workspace/secundario/text/git (master)
$
```

La pantalla muestra el mensaje `Archivos no gestionados:` y el nombre del archivo `sample_memo.md` en texto rojo.

Esto significa 'El archivo `sample_memo.md` no está gestionado por Git'.

También aparece el mensaje `"nothing added to commit but untracked files present"`.

Esto significa que "no se ha añadido nada a la confirmación, pero hay archivos sin seguimiento".

El archivo `sample_memo.md` ha sido creado, pero aún no está registrado en el área de preparación y, por lo tanto, no puede ser rastreado en Git.

Registrar el archivo en área de preparación

Para registrar archivos en el área de preparación, utilice el comando `git add`.

\$ git add [nombre del archivo a registrar]

Esta vez, registre el archivo `sample_memo.md`.

\$ git add sample_memo.md

De nuevo, comprueba el estado del archivo con el comando `git status`.

```
• $ git add sample_memo.md
warning: in the working copy of 'sample_memo.md', LF will be replaced by CRLF the next time Git
touches it

SECUNDARIO@DESKTOP-FBIM49G MINGW64 ~/workspace/secundario/text/git (master)
• $ git status
On branch master

No commits yet

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

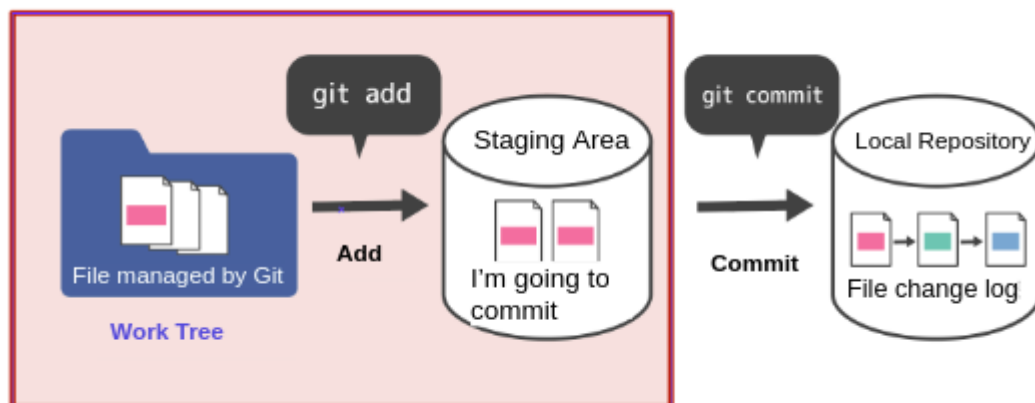
SECUNDARIO@DESKTOP-FBIM49G MINGW64 ~/workspace/secundario/text/git (master)
○ $
```

Aparece el mensaje `Changes to be committed`. Además, el nombre del archivo se ha vuelto verde.

Esto indica que el archivo **sample_memo.md** es ahora rastreado por Git.

Hasta el momento, se han llevado a cabo las siguientes tareas

1. Se ha creado un nuevo archivo en el entorno de trabajo local, el árbol de trabajo.
2. A continuación, el archivo se añadió al área de preparación y se puso a disposición para su confirmación como elemento controlado por Git.



Tenga en cuenta que si un archivo se ha añadido accidentalmente al área de preparación, ejecute el comando `git reset`.

\$ `git reset [Nombre del archivo]`

Para cancelar el archivo `sample_memo.md` que ha añadido al área de preparación, ejecute el siguiente comando.

\$ `git reset sample_memo.md`

Comprueba el estado de Git con el comando `git status`.

\$ `git status`

```
• $ git reset sample_memo.md

SECUNDARIO@DESKTOP-FBIM49G MINGW64 ~/workspace/secundario/text/git (master)
• $ git status
On branch master

No commits yet

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

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

SECUNDARIO@DESKTOP-FBIM49G MINGW64 ~/workspace/secundario/text/git (master)
○ $
```

El archivo `sample_memo.md` está ahora en texto rojo y ha sido restaurado.

Sólo ejecutar `git reset` sin especificar ningún nombre de archivo deshacerá todos los archivos que hayan sido añadidos al área de preparación.

Estos archivos también se pueden añadir de nuevo al área de preparación con el comando `git add`.

Resumen

- Los archivos recién añadidos se muestran como `Untracked files`: en el comando `git status`. Esto se debe a que el archivo no está gestionado por Git.
- El comando `git add` permite que el trabajo realizado en el árbol de trabajo se añada al área de preparación.
- El comando `git reset` te permite deshacer lo que has añadido al área de preparación y eliminarlo de la lista de confirmaciones.



Commit (confirmar) un archivo en el repositorio

Para registrar una versión de tus cambios como commit, necesitas añadirla al área de preparación con el comando `git add` y luego ejecutar el comando `git commit`.

El archivo `sample_memo.md` creado en el texto anterior se ha eliminado del área de preparación con el comando `git reset` y debe añadirse de nuevo al área de preparación.

\$ `git add sample_memo.md`

Al confirmar, debe registrarse el mensaje correspondiente. El mensaje debe describir qué cambios se han realizado. Al describir el mensaje correspondiente, se informará a los demás trabajadores de la intención del cambio.

Existen dos tipos de mensajes de confirmación

1. Cómo escribir en `git commit -m "Contenido del mensaje"`.
2. Cómo entrar utilizando un editor.

Uso del Editor VS Code con Git

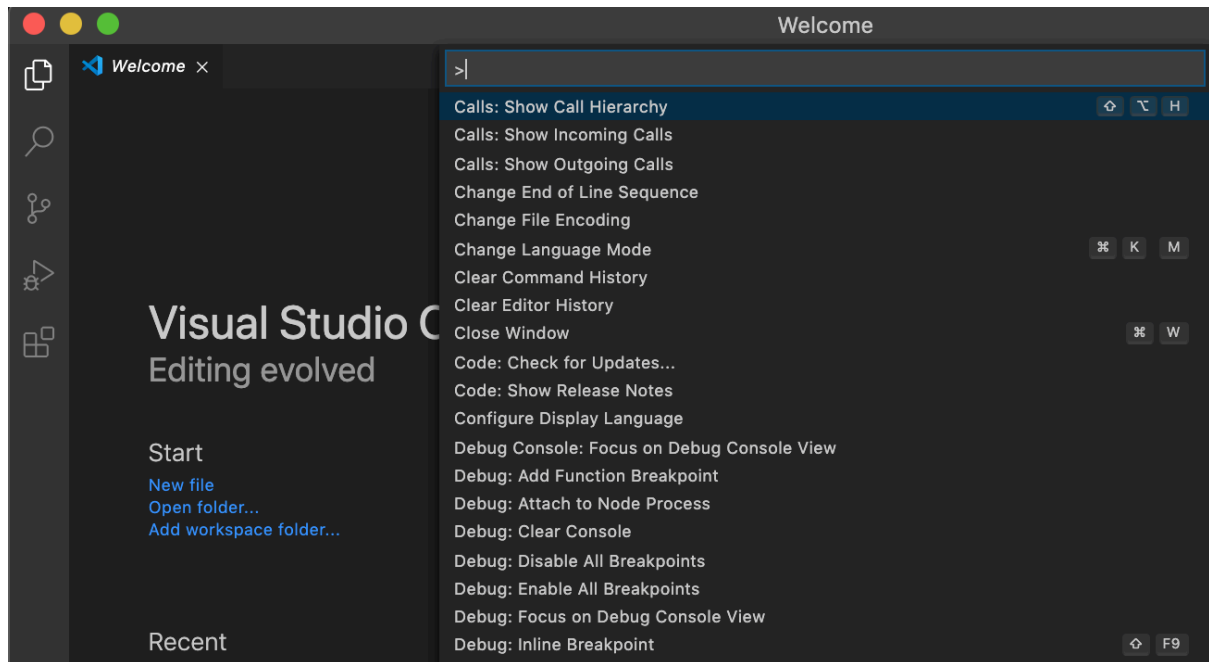
En esta ocasión, vamos a explicar cómo utilizar Visual Studio Code para manejar Git de manera eficiente en tu flujo de trabajo de desarrollo.

1. Configuración Predeterminada en Windows

Cuando instalas VS Code en Windows, este ya viene con una configuración predeterminada que permite interactuar con Git sin problemas. Para realizar un commit, simplemente abre una terminal en VS Code y navega al directorio de tu proyecto. Desde allí, puedes ejecutar comandos Git como `git commit` sin necesidad de configuraciones adicionales.

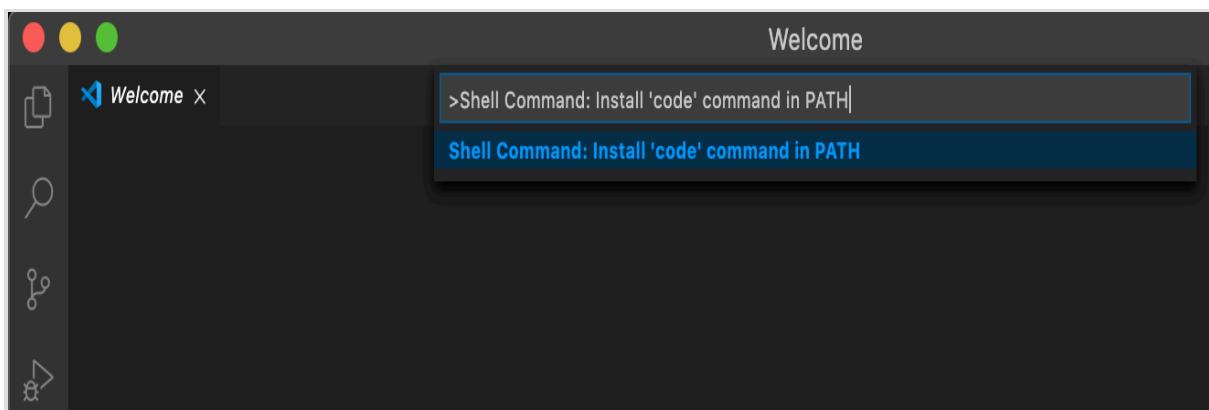
MacOS

Inicie VS Code y Command + Shift + P ejecútalo para mostrar el campo de entrada del comando.



Ingresa el comando a continuación.

Shell Command: Install 'code' command in PATH



Una vez ejecutado, verá la siguiente notificación en la esquina inferior derecha del editor.



git config --local core.editor "code --wait"

Ejecutemos el comando en la terminal .

La ejecución de este comando configurará Git para usar VS Code para ciertas operaciones que requieren un editor de texto.

\$ git config --local core.editor "code --wait"

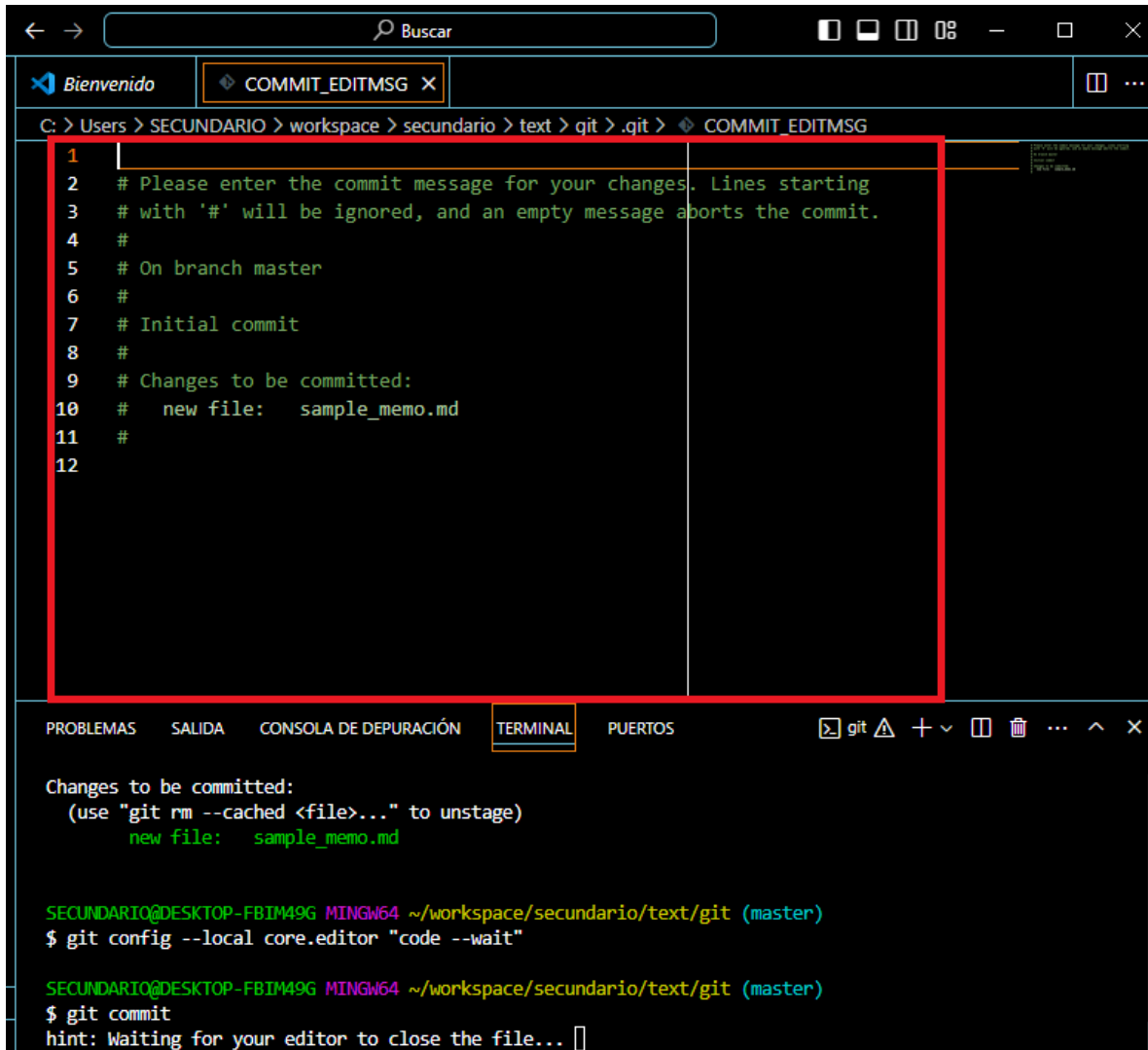
1. `--local` Opción significa realizar esta configuración para el repositorio actual.
2. `core.editor` es un elemento de configuración que especifica el programa que utiliza Git como editor de texto.
3. `code --wait` especifica utilizar VS Code como editor.
4. `--wait` La opción es hacer que Git espere las operaciones mientras VS Code está abierto. Esto suspenderá las operaciones de Git hasta que cierres el editor.

`git config core.editor` Si puede confirmar la visualización del comando como se muestra en la imagen `code --wait`, la configuración estará completa.

Una vez que esto esté completo, podrá realizar operaciones de Git en VS Code, como ingresar mensajes de confirmación.

Para Windows, comience desde aquí.

Escriba `git commit` en el terminal y pulse Enter para abrir la pantalla de edición del mensaje de commit en VS Code.



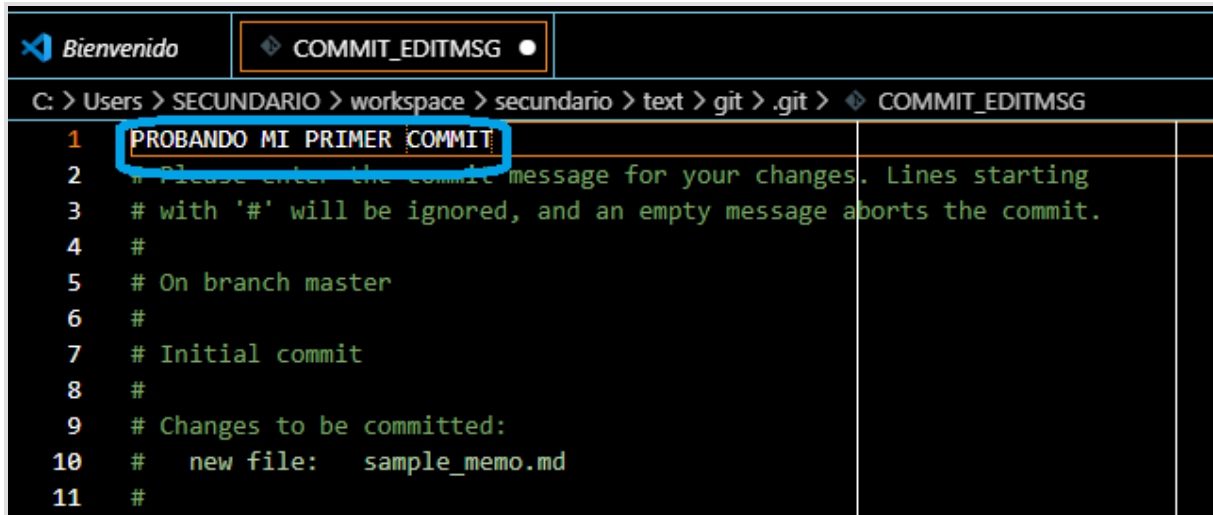
The screenshot shows the Visual Studio Code interface. The top editor pane displays the `COMMIT_EDITMSG` file, which contains a template for a commit message. The bottom pane shows the integrated terminal with the following commands and output:

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

SECUNDARIO@DESKTOP-FBIM49G MINGW64 ~/workspace/secundario/text/git (master)
$ git config --local core.editor "code --wait"

SECUNDARIO@DESKTOP-FBIM49G MINGW64 ~/workspace/secundario/text/git (master)
$ git commit
hint: Waiting for your editor to close the file...
```

Escriba su mensaje de confirmación, `Command + S` presione para guardar en Mac y `Command + W` cierre la ventana para crear una confirmación. En Windows (WSL), por otro lado, puede crear una confirmación de manera similar `Ctrl + S` presionando para guardar y luego para cerrar la ventana. `Ctrl + W`



```

1  PROBANDO MI PRIMER COMMIT
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  #
7  # Initial commit
8  #
9  # Changes to be committed:
10 #   new file:   sample_memo.md
11 #

```

En el trabajo realizado desde el texto anterior hasta este punto, se ha llevado a cabo lo siguiente

1. Se ha creado un nuevo archivo en el entorno de trabajo local, el árbol de trabajo.
2. A continuación, el archivo se añadió a la zona de preparación con el comando `git add` y se puso a disposición de los commits como archivo gestionado por Git.
3. Se ha creado un registro de cambios como commit en el repositorio local utilizando el comando `git commit`.



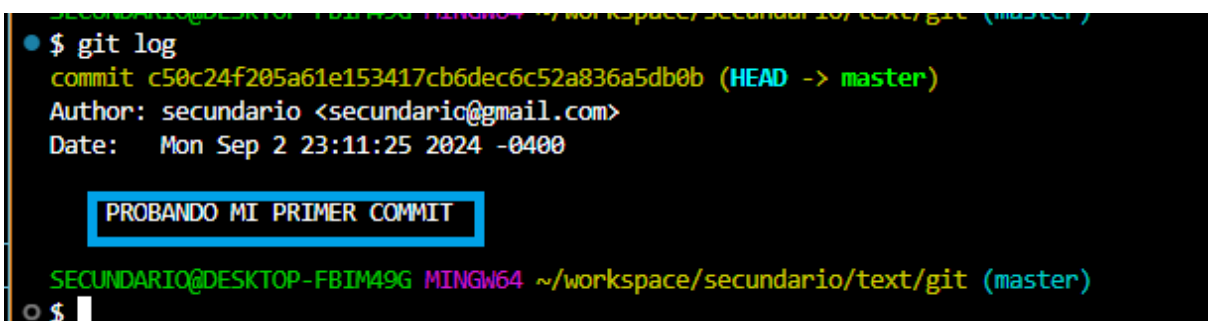
Documentación oficial sobre la operación de confirmación.

[Grabación de cambios Git en el repositorio](#)

Comprueba el historial de confirmación

Para comprobar el historial de confirmaciones, ejecute el comando `git log`

\$ git log



```

SECUNDARIO@DESKTOP-FBIM49G MINGW64 ~/workspace/secundario/text/git (master)
$ git log
commit c50c24f205a61e153417cb6dec6c52a836a5db0b (HEAD -> master)
Author: secundario <secundario@gmail.com>
Date: Mon Sep 2 23:11:25 2024 -0400

    PROBANDO MI PRIMER COMMIT

SECUNDARIO@DESKTOP-FBIM49G MINGW64 ~/workspace/secundario/text/git (master)
$

```

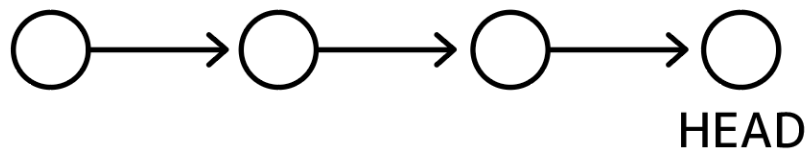
Puedes ver el mensaje de confirmación que acaba de crear.

Junto a la palabra "commit", hay una serie de números y letras. Este es el **commit ID**.

Este ID se genera cada vez que se ejecuta una confirmación y se puede utilizar para desplazarse a un punto de confirmación específico en el tiempo o para deshacer una confirmación.

Además, a la derecha del ID de confirmación, verás **HEAD**.

Este HEAD indica el último commit realizado en el repositorio local, que queda registrado en la carpeta `.git`.



Por ejemplo, si ejecutas un commit 10 veces, habrá 10 IDs de commit que podrás ver en el comando `git log`, pero **HEAD** sólo se muestra en un lugar, lo que significa que es la confirmación más reciente.

Ten en cuenta que para obtener un historial conciso y fácil de leer, utiliza el comando `git log --oneline`.

Una única confirmación se muestra línea a línea.

Además, escribe una clave `git log` para finalizar el comando `.q`

q

En cuanto a la visualización del historial de commits, aquí está la documentación oficial



[Ver el historial de confirmaciones de Git](#)

Resumen

- El mensaje de confirmación puede ser descrito en VS Code editando git config core.editor.
- Los archivos añadidos al área de preparación con git add se registran como cambios con git commit.
- Puedes comprobar el historial de commits en git log.
- HEAD significa el último commit ejecutado.
-

Documentos Oficiales

- [Registrar los cambios Git en el repositorio](#)
- [2. Ver el historial de commits de Git](#)