

Verifica el estado de Git

Incluso si opera con comandos de Git, a menudo es invisible, por lo que es difícil comprender el estado.

En tales casos, `git status` utilice comandos. Puede comprobar el estado de los archivos bajo control de versiones.

Avancemos moviendo nuestras manos.

Como referencia, si lo ejecuta sin confirmar nada todavía, es decir, sin crear ningún archivo o directorio, verá lo siguiente:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  1: zsh  ▾

% git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
% █
```

No commits yet se muestra en esta pantalla . Este es un mensaje de que no se ha creado una confirmación, que es un registro de control de versiones. Esto no se mostrará si se ha realizado alguna confirmación.

De hecho, creamos un nuevo archivo y `git add` realicemos un seguimiento de los cambios en el contenido del archivo mediante comandos.

Crea un archivo llamado `sample_text_file.txt`.

\$ touch sample_text_file.txt

Ejecuta el comando `git status` para comprobar el estado.

\$ git status

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  1: zsh  ▾

% touch sample_text_file.txt
% git status
On branch master

No commits yet

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

    sample_text_file.txt

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

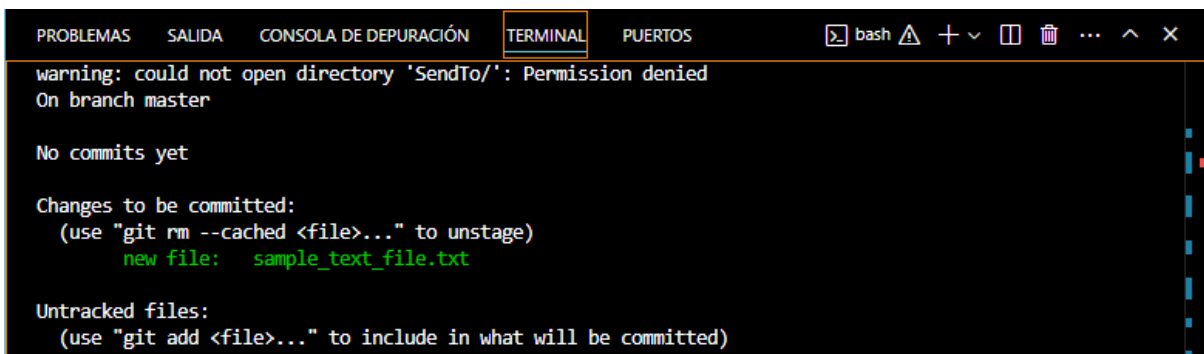
La pantalla muestra `Untracked files:`, lo que confirma que hay archivos recién creados cuyos cambios de contenido no son rastreados por Git.

Añade el archivo `sample_text_file.txt` al área de preparación.

\$ git add sample_text_file.txt

Ejecuta de nuevo el comando `git status` para comprobar el estado.

\$ git status



```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS
warning: could not open directory 'SendTo/': Permission denied
On branch master

No commits yet

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

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

Aparece en la pantalla `Cambios a comprometerse:`. Puede ver que se ha confirmado el cambio para añadir un nuevo archivo.

Crea un commit con un mensaje apropiado.

En este caso, añade un mensaje de confirmación **"Add: add text file as sample"**.

\$ git commit -m "Add: add text file as sample"

Cambia el contenido del archivo `sample_text_file.txt` que acabas de crear.

Utiliza el comando `echo` para añadir "hola mundo" al archivo.

\$ echo "hello world" >> sample_text_file.txt

Ejecute el comando `open` para ver el contenido del archivo.

Al ejecutar el commando `open`, el contenido del archivo se muestra en un editor de texto.

\$ open sample_text_file.txt

(si este no funciona prueba el de abajo). (y al ejecutar si te pregunta con que abrir, elige bloc de notas o VS Code).

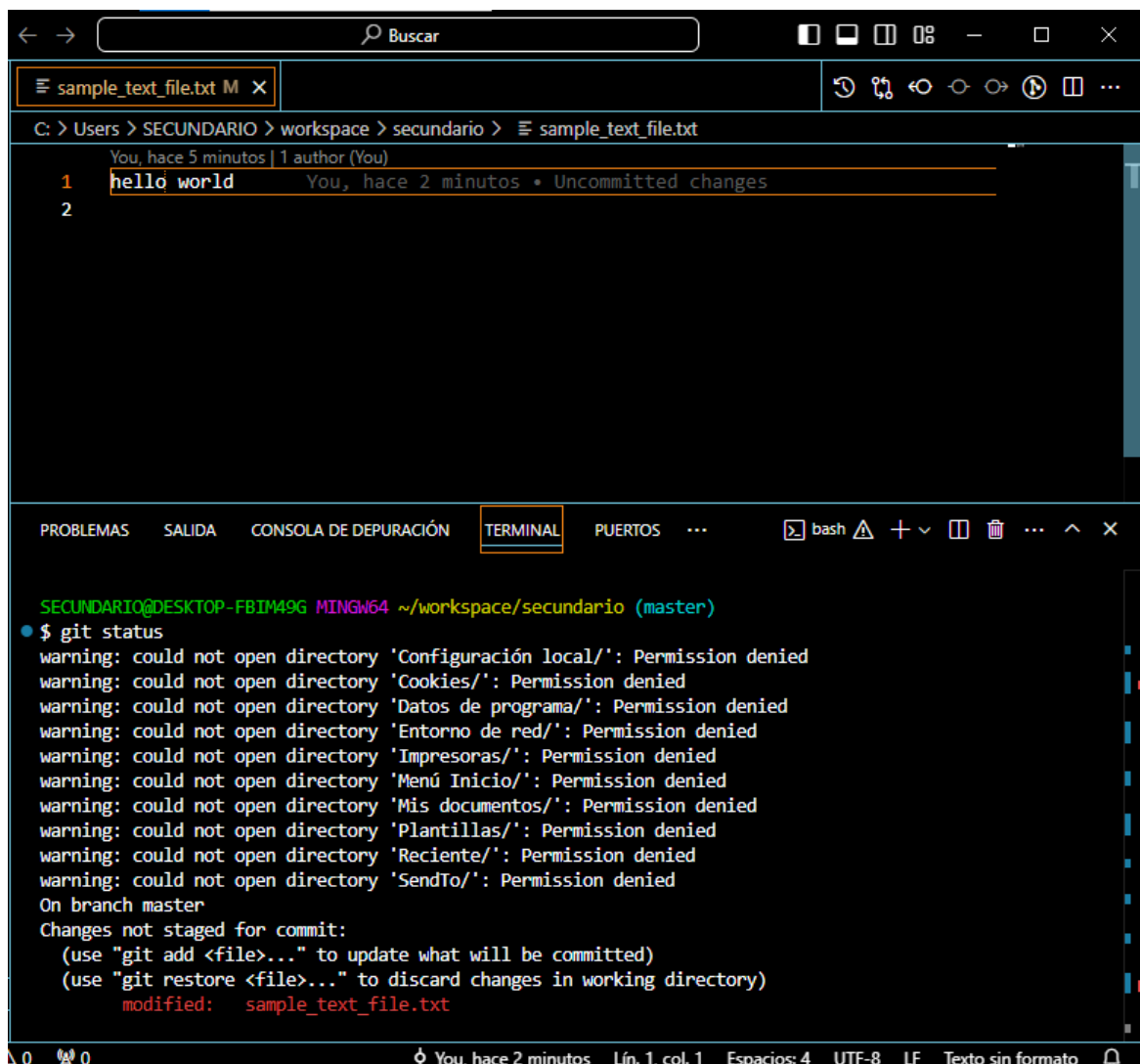
Los usuarios de Windows (WSL) `explorer.exe` ejecuta el comando.

\$ explorer.exe sample_text_file.txt

(y al ejecutar si te pregunta con que abrir, elige bloc de notas o VS Code).

Ejecuta de nuevo el comando `git status` para comprobar el estado.

\$ git status



```

C:\Users\SECUNDARIO\workspace\secundario> open sample_text_file.txt

You, hace 5 minutos | 1 author (You)
1 hello world You, hace 2 minutos • Uncommitted changes
2

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS ...
SECUNDARIO@DESKTOP-FBIM49G MINGW64 ~/workspace/secundario (master)
$ git status
warning: could not open directory 'Configuración local/': Permission denied
warning: could not open directory 'Cookies/': Permission denied
warning: could not open directory 'Datos de programa/': Permission denied
warning: could not open directory 'Entorno de red/': Permission denied
warning: could not open directory 'Impresoras/': Permission denied
warning: could not open directory 'Menú Inicio/': Permission denied
warning: could not open directory 'Mis documentos/': Permission denied
warning: could not open directory 'Plantillas/': Permission denied
warning: could not open directory 'Reciente/': Permission denied
warning: could not open directory 'SendTo/': Permission denied
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:   sample_text_file.txt
```

La pantalla mostrará `Changes not staged for commit:`.

Esto significa que se dan las siguientes condiciones

- Algunos de los archivos versionados en Git han sido modificados.
- Los cambios realizados no se han añadido al área de preparación para ser confirmados.

Añade al área de preparación y comprueba su estado.

```
$ git add sample_text_file.txt
```

```
$ git commit -m "Modify: change text file contents"
```

```
$ 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:   sample_text_file.txt
```

Se confirma el mensaje `nothing to commit. working tree clean.`

Significa "Nada ha cambiado en el archivo que está gestionando, nada que confirmar".



[Registro de cambios Git en el repositorio.](#)

Resumen

El comando `git status` permite comprobar el estado de los archivos bajo control de versiones.

Cómo configurar archivos no administrados por Git

¿Cuándo no gestionar con Git?

En el texto anterior, hemos añadido los archivos que queríamos que estuvieran controlados por versiones como commits en Git.

Sin embargo, hay archivos que no deben añadirse como commits y deben excluirse de la gestión de Git.

Archivos que no necesitan ser compartidos con otros trabajadores

1. La información de registro del entorno local (como los [informes generados en la Consola de macOS](#) o en la terminal de WSL) no es necesaria para otros colaboradores. Los archivos `.DS_Store` utilizados por Finder en macOS y los archivos `desktop.ini` utilizados por el Explorador de Windows son para gestionar la configuración de las carpetas, por lo que no es necesario añadirlos al control de versiones.
2. Tomando como ejemplo el desarrollo en Rails y JavaScript, los ficheros de las librerías Ruby y Node.js que se adquieren por comando con `bundle install` o `yarn install` deben excluirse de la gestión para que no se incluyan en los commits. Estas librerías se actualizan diariamente y son adquiridas por cada trabajador en su propio entorno mediante comando, por lo que no es necesario gestionar los ficheros en el estado en el que se encuentran cuando se ejecuta el commit.

Archivos sobre seguridad

- La información de contraseñas y los archivos relacionados con la seguridad necesarios para la autenticación pueden ser muy peligrosos si están versionados, ya que están expuestos por Git.
- Utilizando Rails como ejemplo, la versión 5.2 o posterior tiene la capacidad de cifrar la información de autenticación utilizando credenciales. Sin embargo, si el archivo

master.key para descifrar la información se publica en Git, la información cifrada se analizará y el cifrado perderá su significado.

- Aparte de la función de credenciales, también puede tratar con un archivo que define información específica del entorno llamado `.env`. Si este archivo también está comprometido, la información de autenticación quedará expuesta.
- Otro peligro es que si consignas un archivo de claves privadas para conexiones SSH a un servidor como GitHub, o un archivo de claves secretas, cualquier persona de dominio público puede acceder y ver o manipular la información.



Los ejemplos de desarrollo en Rails y JavaScript pueden no ser imaginables ahora, pero podrás volver a leerlos y entenderlos cuando los necesites.

Gestión de archivos `.gitignore`

Para registrar un archivo como no gestionado por Git, necesitas incluir el nombre del archivo en cuestión en un archivo de configuración llamado `.gitignore`.

En primer lugar, crea un archivo `.gitignore` utilizando el siguiente comando.

\$ touch .gitignore

Debido a que se trata de archivos de configuración, normalmente no se muestran en la Terminal o en Finder. En la Terminal, para mostrarlos junto con otros archivos, es necesario ejecutar el comando `ls -la` con las opciones adecuadas. En Finder, se puede alternar la visibilidad de los archivos de configuración presionando simultáneamente las teclas `Command + Shift + .` En el Explorador de Windows, puedes alternar la visibilidad de los archivos utilizando la pestaña 'Ver' y marcando o desmarcando la casilla 'Archivos ocultos'.



```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  GITLENS  bash + - [ ] [ ] ... ^ X

SECUNDARIO@DESKTOP-FBIM49G MINGW64 ~ (master)
• $ touch .gitignore

SECUNDARIO@DESKTOP-FBIM49G MINGW64 ~ (master)
• $ ls -la .gitignore
-rw-r--r-- 1 SECUNDARIO 197121 0 sep.  4 09:34 .gitignore

SECUNDARIO@DESKTOP-FBIM49G MINGW64 ~ (master)
o $
```

De hecho, vamos a registrar un archivo en `.gitignore` que no debe ser comprometido.

Crear el archivo `config/master.key` como un archivo temporal aquí.

Ve al directorio de trabajo que creaste para la serie Introducción a Git.

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

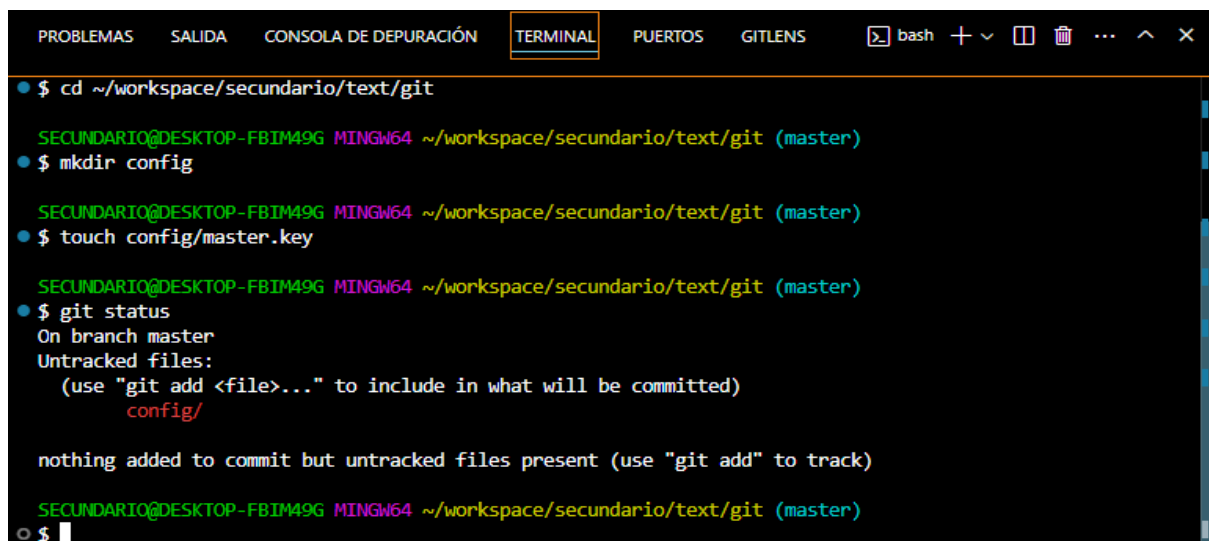
Crea un directorio llamado `config`.

```
$ mkdir config
```

Crea un archivo llamado `master.key` en el directorio `config` y comprueba el estado de Git.

```
$ touch config/master.key
```

```
$ git status
```



```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  GITLENS  bash + - [ ] [X] ... ^ X

$ cd ~/workspace/secundario/text/git

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

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

SECUNDARIO@DESKTOP-FBIM49G MINGW64 ~/workspace/secundario/text/git (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    config/

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

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

El directorio de configuración que contiene el `master.key` puede ser `git add`.

Usa el comando `echo` para añadir la cadena `"config/master.key"` a `.gitignore`.

Esto listará `"config/master.key"` como un archivo no gestionado por Git.

```
$ echo "config/master.key" >> .gitignore
```

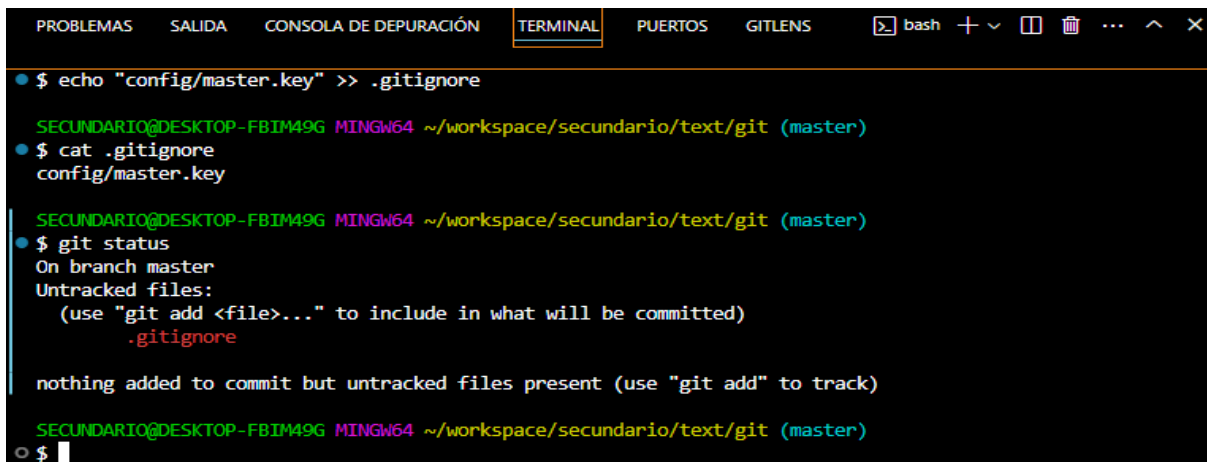
Utilice el comando `cat` para comprobar el contenido del archivo `.gitignore`.

```
$ cat .gitignore
```

El resultado del comando `git status` debería cambiar.

```
$ git status
```

El directorio `config` ha desaparecido, prueba de que ya no está bajo el control de Git.



```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  GITLENS  bash + - [ ] [ ] ... ^ X

$ echo "config/master.key" >> .gitignore

SECUNDARIO@DESKTOP-FBIM49G MINGW64 ~/workspace/secundario/text/git (master)
$ cat .gitignore
config/master.key

SECUNDARIO@DESKTOP-FBIM49G MINGW64 ~/workspace/secundario/text/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)

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

Un archivo `.gitignore` permite a un repositorio local reconocer archivos que no están sujetos al control de Git sin tener que `git add` o `git commit`. Sin embargo, dado que el archivo `.gitignore` se encuentra en el mismo directorio que el archivo `.git`, suele situarse en la misma jerarquía que éste.

Este archivo `.gitignore` en sí mismo está comprometido y gestionado por Git. Esto es para que puedas saber que compartir con otros desarrolladores y qué archivos quieres excluir de la gestión.

También puedes excluir de la gestión de archivos de Git que ya han sido `git added` o `git committed`. Para ello, existe el comando `git reset` y el comando `git rm --cached`.

Experimenta de primera mano el proceso de reconocer los archivos que ya se han añadido al área de preparación como no sujetos a la gestión de Git.

Utiliza `git status` para comprobar el estado de cada trabajo a medida que avanzas.
Primero, crea un archivo `.env` y añádelo al área de preparación.

```
$ touch .env
```

```
$ git add .env
```

```
$ git status
```

`.env` en el archivo `.gitignore`.

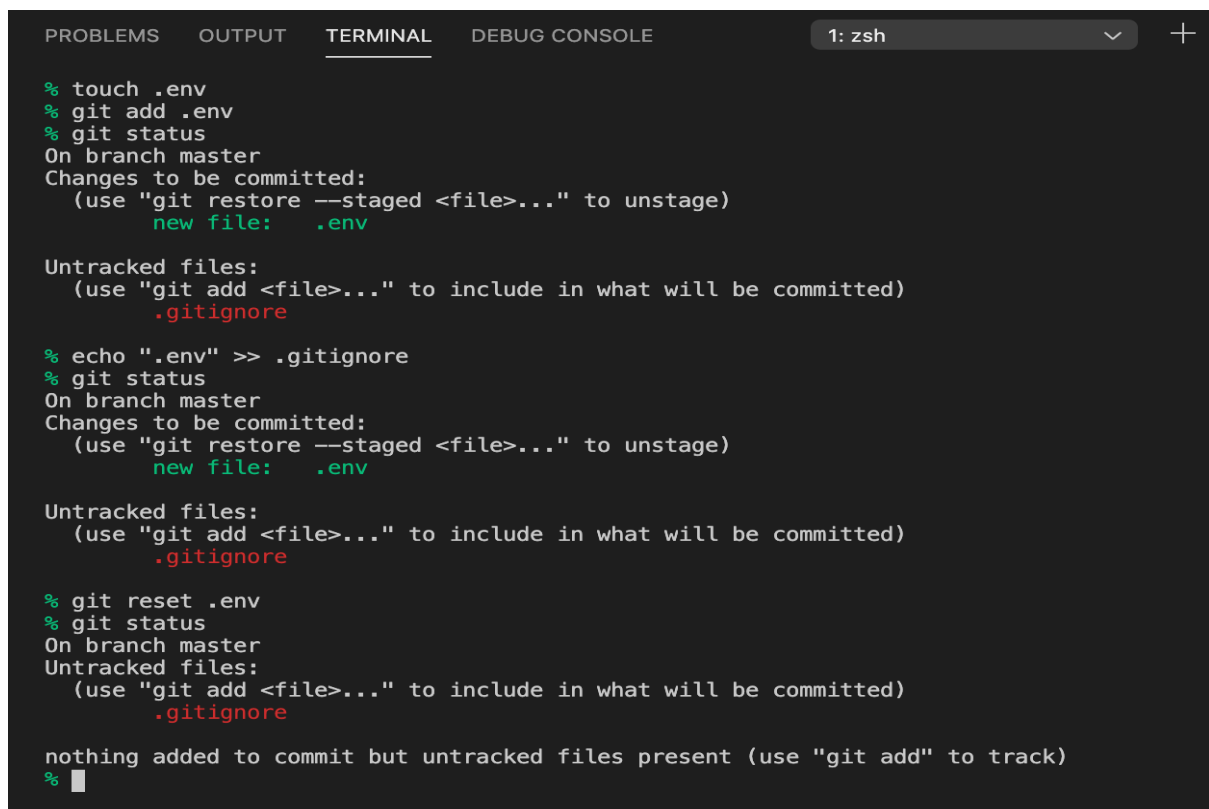
```
$ echo ".env" >> .gitignore
```

```
$ git status
```

Cancele el archivo `.env` del área de preparación.

```
$ git reset .env
```

```
$ git status
```



```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  1: zsh  +

% touch .env
% git add .env
% git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .env

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

% echo ".env" >> .gitignore
% git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .env

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

% git reset .env
% 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)
% 
```

El archivo `.env` ya no está sujeto a la gestión de Git.

Para manejar archivos ya confirmados que ya no están bajo el control de Git, es necesario ejecutar el comando `git rm`. Sin embargo, esto también puede tener un impacto inesperado en repositorios remotos. Comprueba cuidadosamente el significado y el impacto del comando antes de proceder.

Resumen

Los archivos que no deberían ser gestionados por Git incluyen archivos innecesarios como librerías adquiridas y logs, y archivos relacionados con la seguridad.

Puedes hacer que el repositorio reconozca un fichero o directorio como no gestionado por Git describiéndolo en un fichero de configuración llamado `.gitignore`.

Para eliminar un archivo `git add` o `git commit` del control de Git, necesitas eliminarlo del área de preparación y del historial de confirmaciones.

