**GIT LOCAL**

Es un software de control de versiones diseñado con el propósito de llevar un registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

**Casi todas las operaciones son locales**

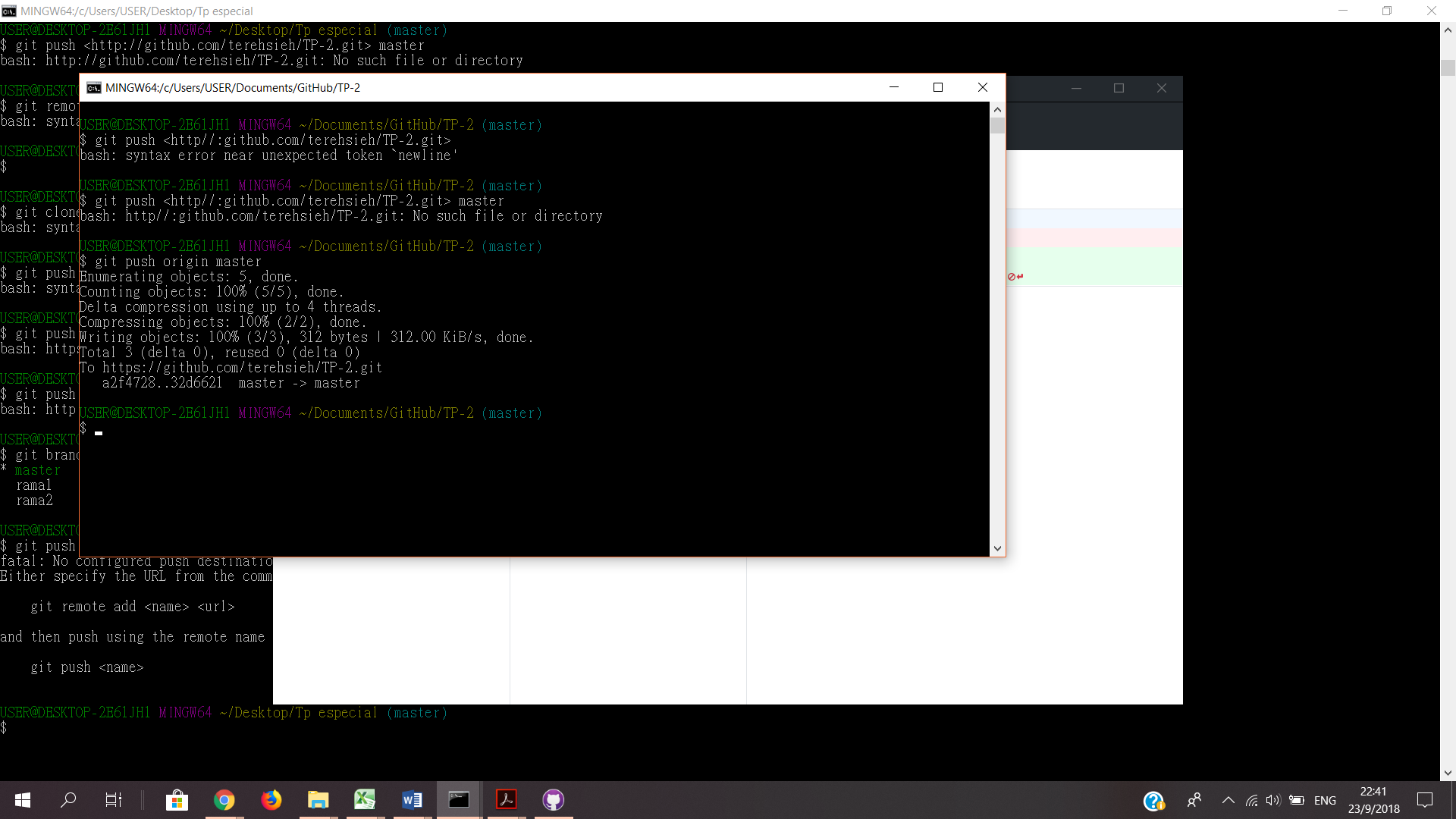
* Tienes toda la historia del proyecto ahí mismo, en tu disco local, la mayoría de las operaciones parecen prácticamente inmediatas, no necesita conectarse a un servidor.
* Si quieres ver los cambios introducidos en un archivo entre la versión actual y la de hace un mes, Git hace un *cálculo de diferencias localmente*, en lugar de tener que pedirle a un servidor remoto que lo haga u obtener una versión antigua desde la red.

Comando *git diff --base* <**nombre archivo**>: para ver conflictos entre archivos.

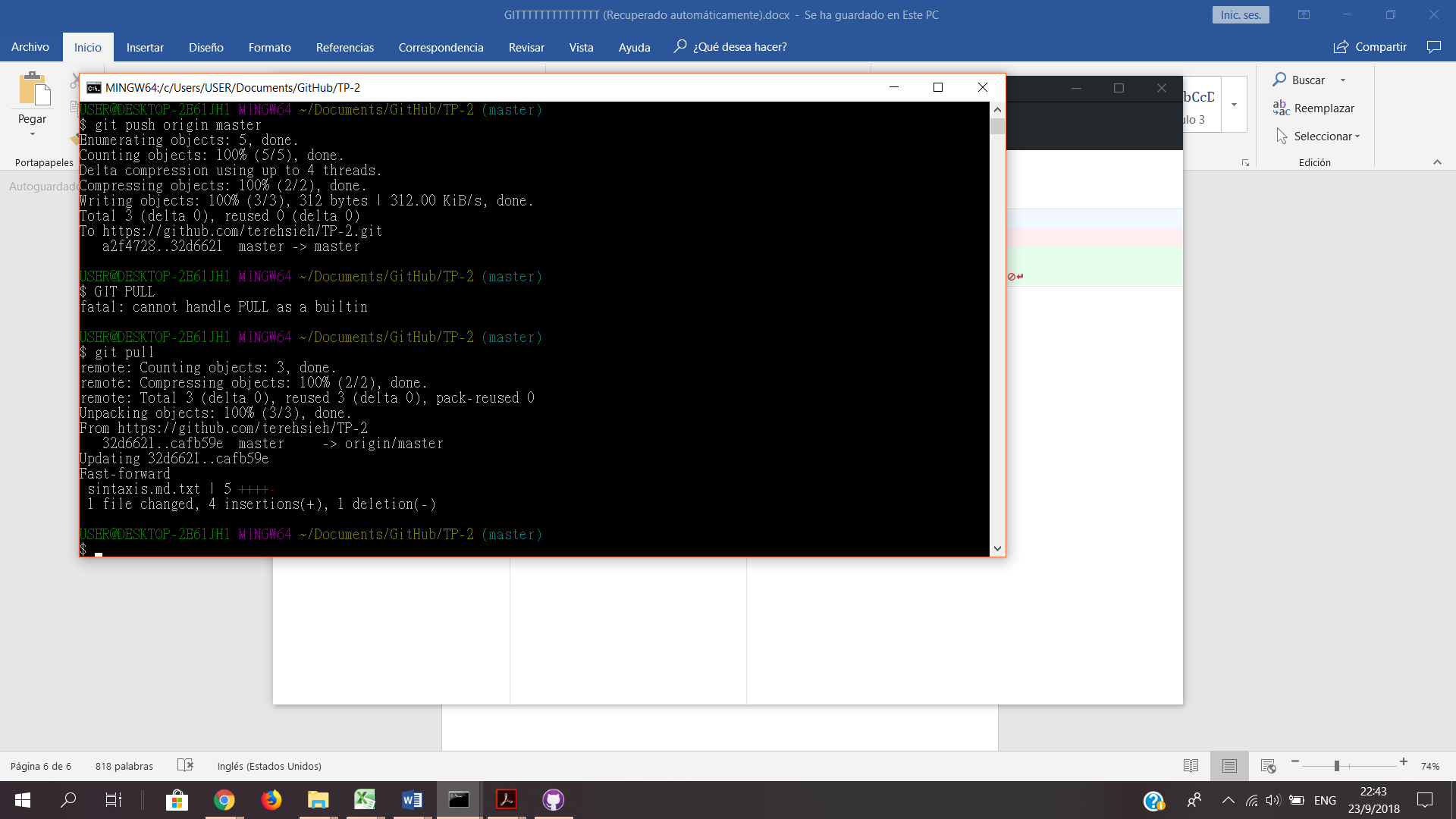
Comando *git diff <source-branch> <target-branch*>: para ver conflictos entre distintas ramas.

* Puedes trabajar tranquilamente en tu repositorio local y luego subir los cambios.

El comando *git push origin master,* envía los cambios que se han hecho en la rama principal de los repertorios remotos que están asociados con el directorio que está trabajando.



Y para poder fusionar todos los cambios que se han hecho en el repositorio local trabajando, el comando que se usa es *git pull*



**Los Tres Estados**

* CONFIRMADO (committed): los datos están almacenados de manera segura en tu base de datos local.
* MODIFICADO (modified):has modificado el archivo pero todavía no lo has confirmado a tu base de datos.
* PREPARADO (staged): has marcado un archivo modificado en su versión actual para que vaya en tu próxima confirmación.

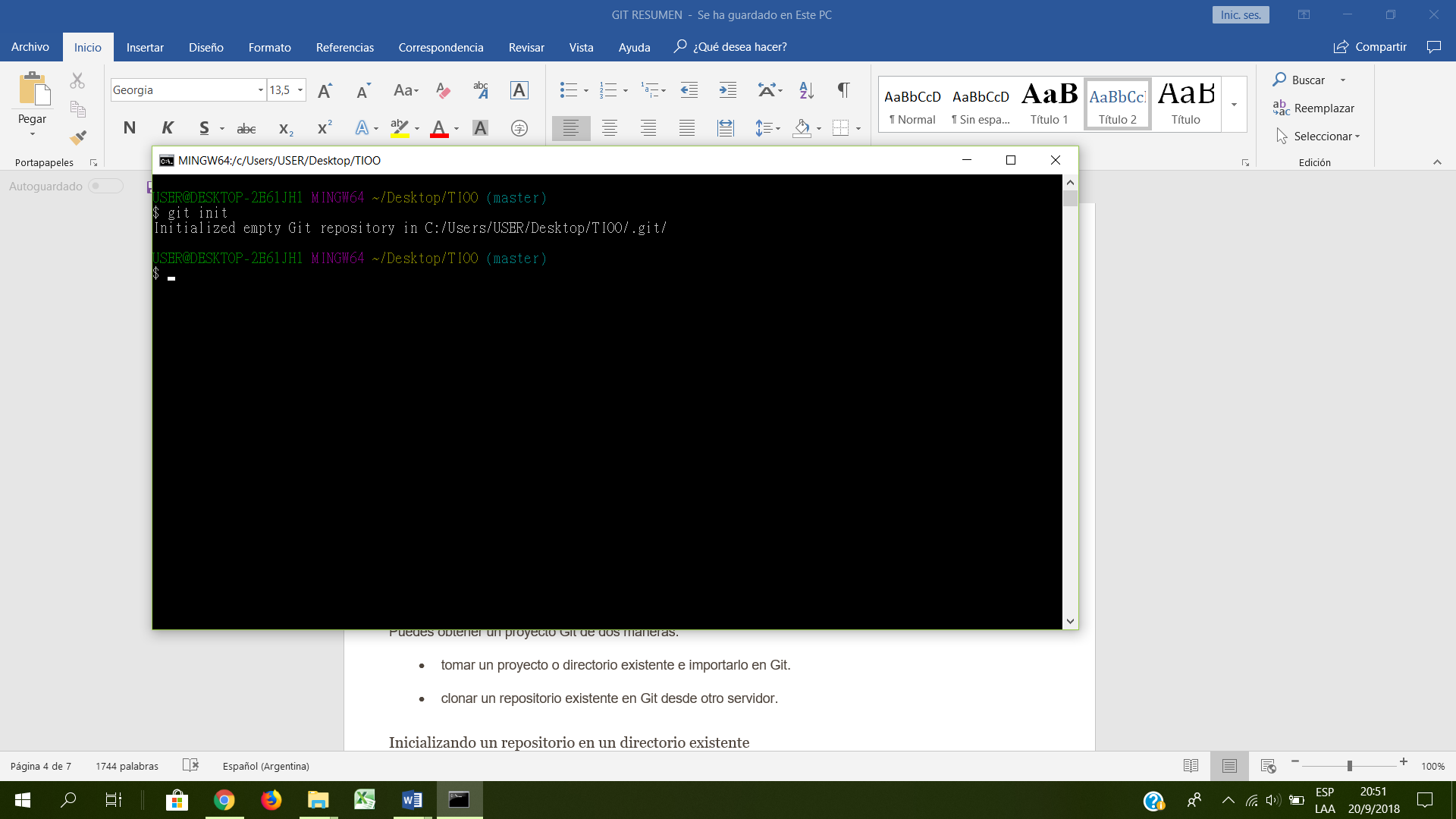
**FUNDAMENTOS BÁSICOS DE GIT**

**Obteniendo un repositorio Git**

Puedes obtener un proyecto Git de dos maneras:

1. tomar un proyecto o directorio existente e importarlo en Git.
2. clonar un repositorio existente en Git desde otro servidor.

### **Inicializando un repositorio en un directorio existente**

Si estás empezando a seguir un proyecto existente en Git, debes ir al directorio del proyecto y usar el siguiente comando:  *git init*

Esto crea un subdirectorio nuevo llamado .git, el cual contiene todos los archivos necesarios del repositorio – un esqueleto de un repositorio de Git. Todavía no hay nada en tu proyecto que esté bajo seguimiento.

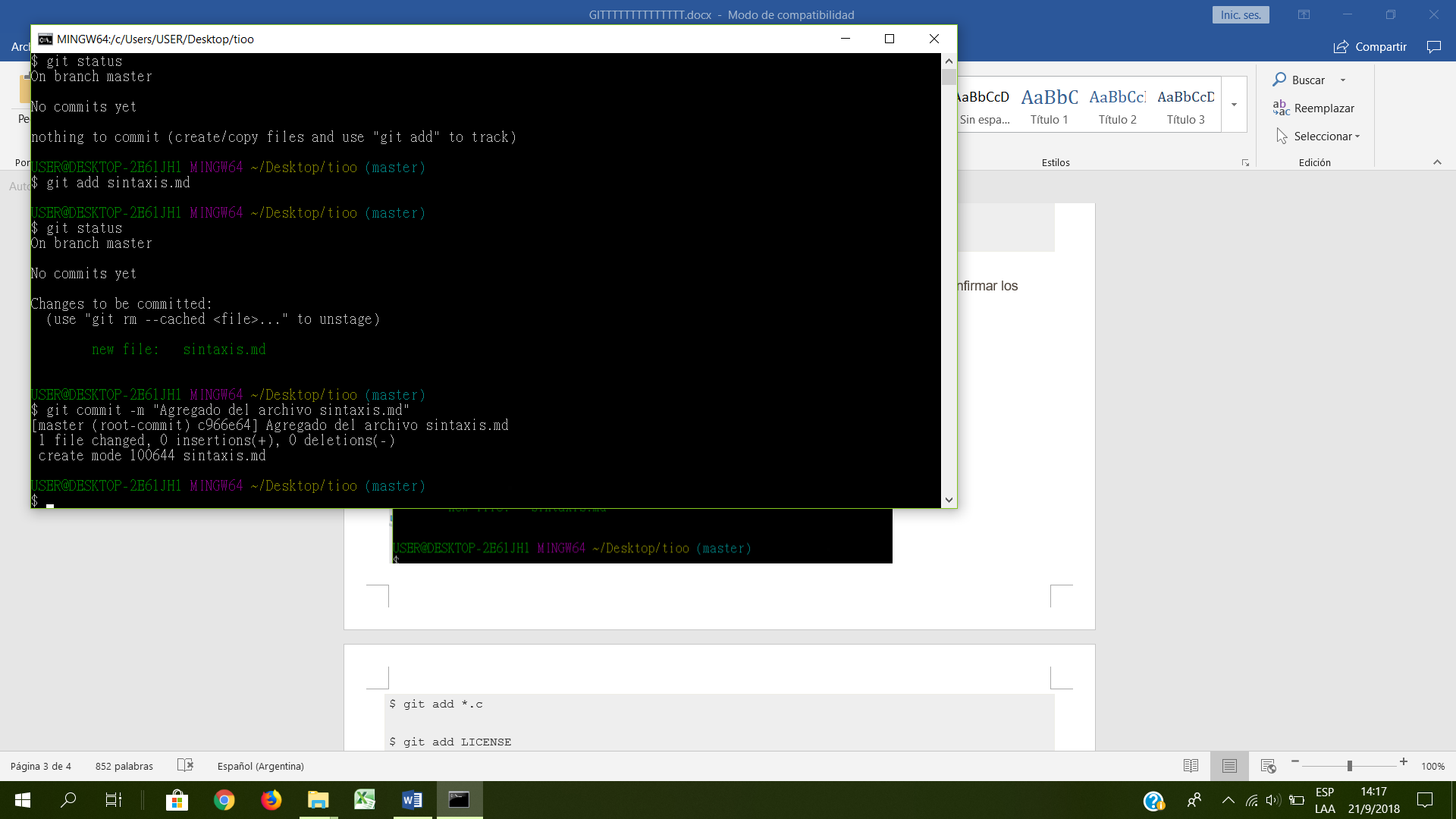
Luego ponemos: *git add* ***nombre del archivo***

Para especificar qué archivos quieres controlar, seguidos de un *git commit* para confirmar los cambios:

Al agregar el archivo, se convierte en verde new file: sintaxis.md (preparado para ser confirmado). En este momento, tienes un repositorio de Git con archivos bajo seguimiento y una confirmación inicial.

Al confirmar los cambios puedes agregar mensaje, de la cual es una breve descripción de cuáles fueron los cambios hechos, para que otros colaboradores puedan saber ¿qué hiciste? Y ¿para qué?

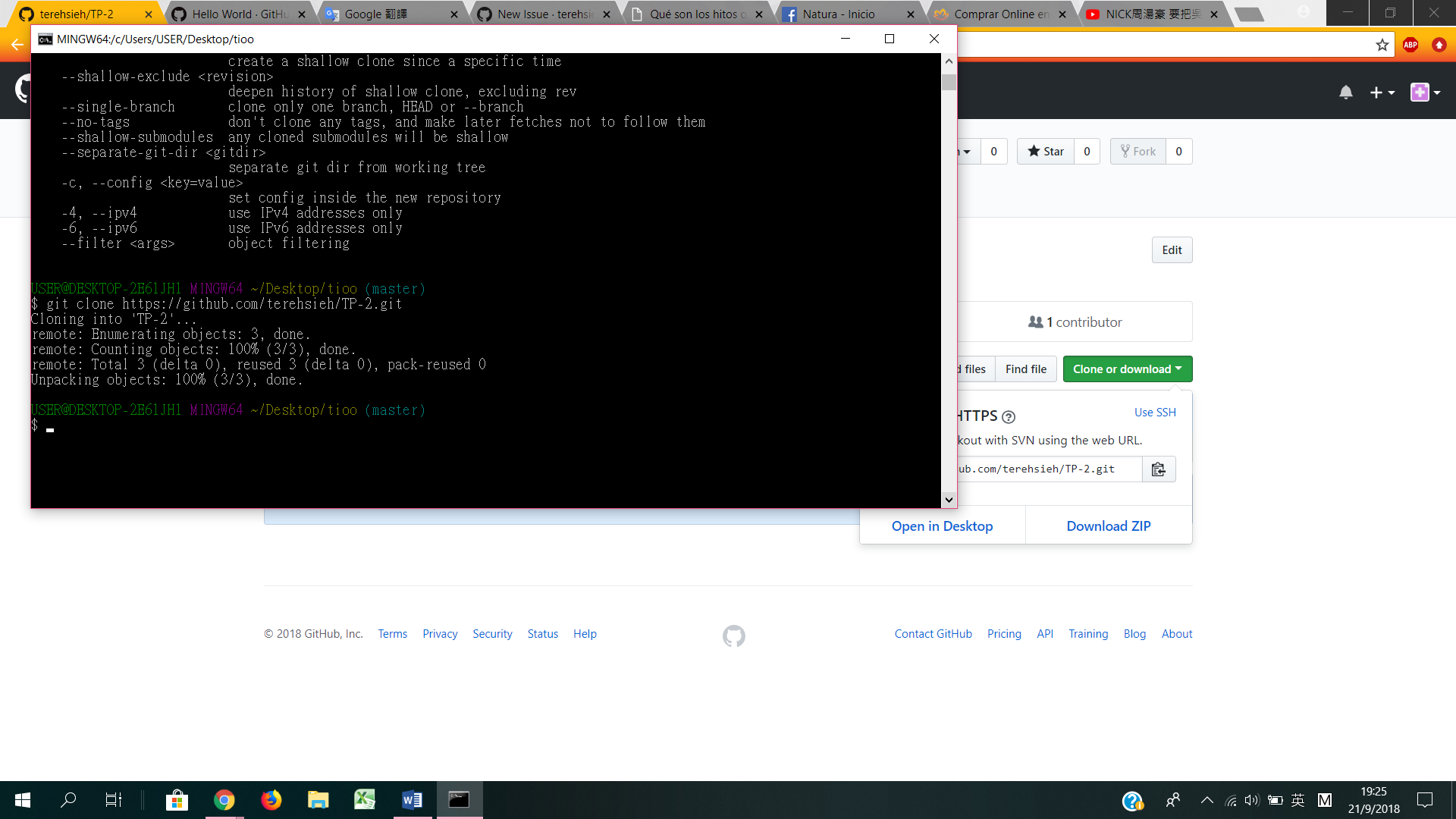
Ej: *git commit -m* “**mensaje descriptivo**”



### **Clonando un repositorio existente**

Si deseas obtener una copia de un repositorio Git existente — por ejemplo, un proyecto en el que te gustaría contribuir — el comando que necesitas es *git clone [url]*

Ej: $ git clone https://github.com/terehsieh/TP-2.git

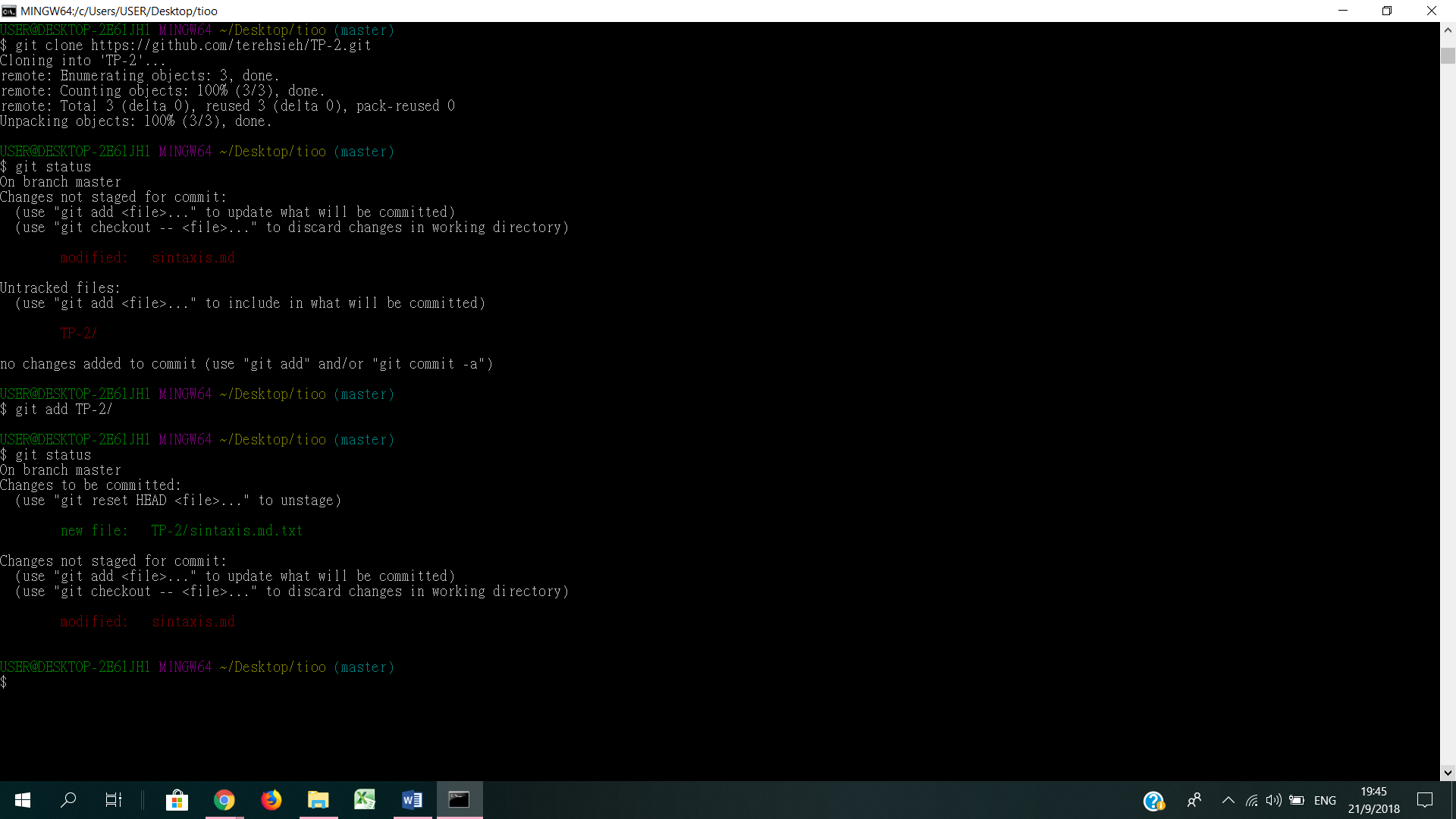


## **Guardando cambios en el Repositorio**

Cada archivo de tu repositorio puede tener dos estados:

* Rastreados (tracked files): Todos aquellos archivos que estaban en la última instantánea del proyecto.
* Sin rastrear: cualquier otro archivo en tu directorio de trabajo que no estaba en tu última instantánea y que no están en el área de preparación (*staging area*).

Cuando CLONAS por primera vez un repositorio TODOS ESTAS RASTREADOS.

El comando *git status:* te permite conocer en qué estado se encuentra el proyecto y en que rama(branch) se encuentra.

Si agrego un archivo aparece -----“Untracked files” (Archivos no rastreados), son archivos que no tenías en el commit anterior.

Y aparecen en color rojo.

Ej: modified: sintaxis.md

TP -2/

### **Rastrear Archivos Nuevo**

Para agregarlos tengo q poner

*git add* **nombre del archivo**

Este comando puede recibir tanto una ruta de archivo como de un directorio; si es de un directorio, el comando añade recursivamente los archivos que están dentro de él.

### **Preparar Archivos Modificados**

*git add. git add* es un comando que cumple varios propósitos - lo usas para empezar a RASTREAR archivos nuevos, PREPARAR archivos, y hacer otras cosas como MARCAR COMO RESUELTO ARCHIVOS en conflicto por combinación.

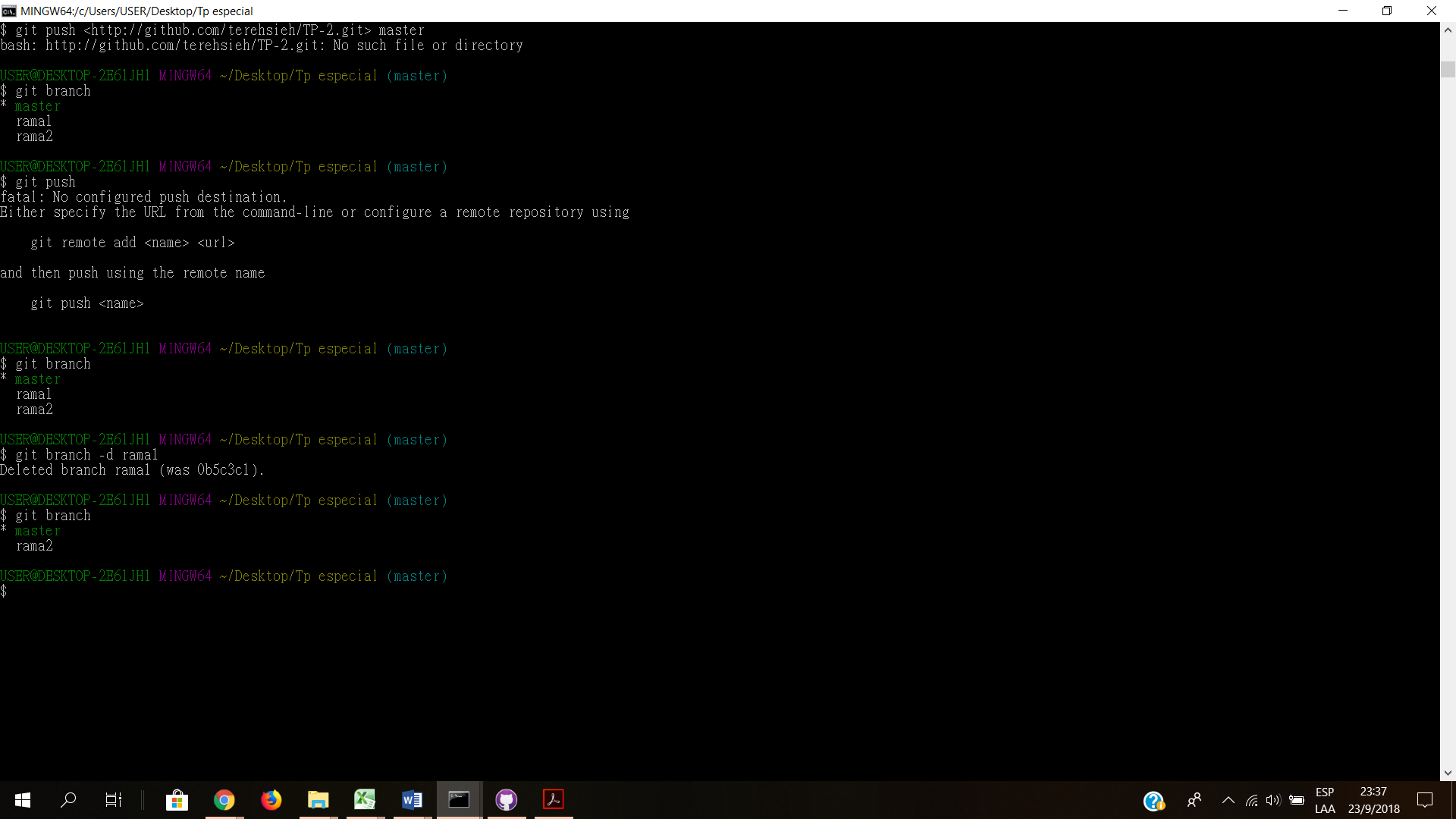
# **Ramificaciones en Git**

Una rama Git es simplemente un apuntador móvil apuntando a una de esas confirmaciones. La rama por defecto de Git es la rama *master*. Con la primera confirmación de cambios que realicemos, se creará esta rama principal *master* apuntando a dicha confirmación. En cada confirmación de cambios que realicemos, la rama irá avanzando automáticamente.

### Crear una Rama Nueva

¿Qué sucede cuando creas una nueva rama? Bueno…​, simplemente se crea un nuevo apuntador para que lo puedas mover libremente.

Para crear una rama: *git branch* ***nombre de la rama***

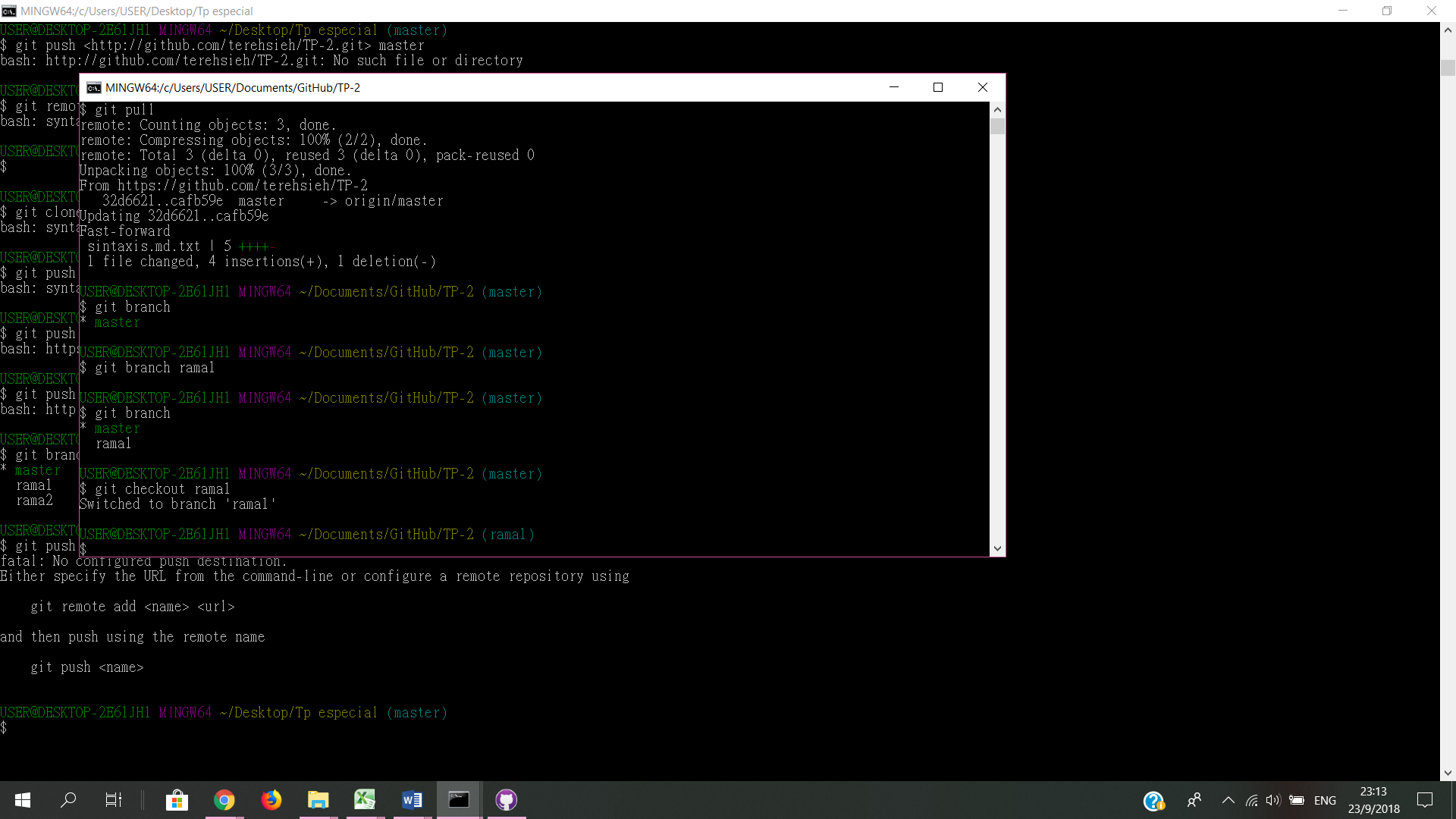
Ej: *git branch rama1*

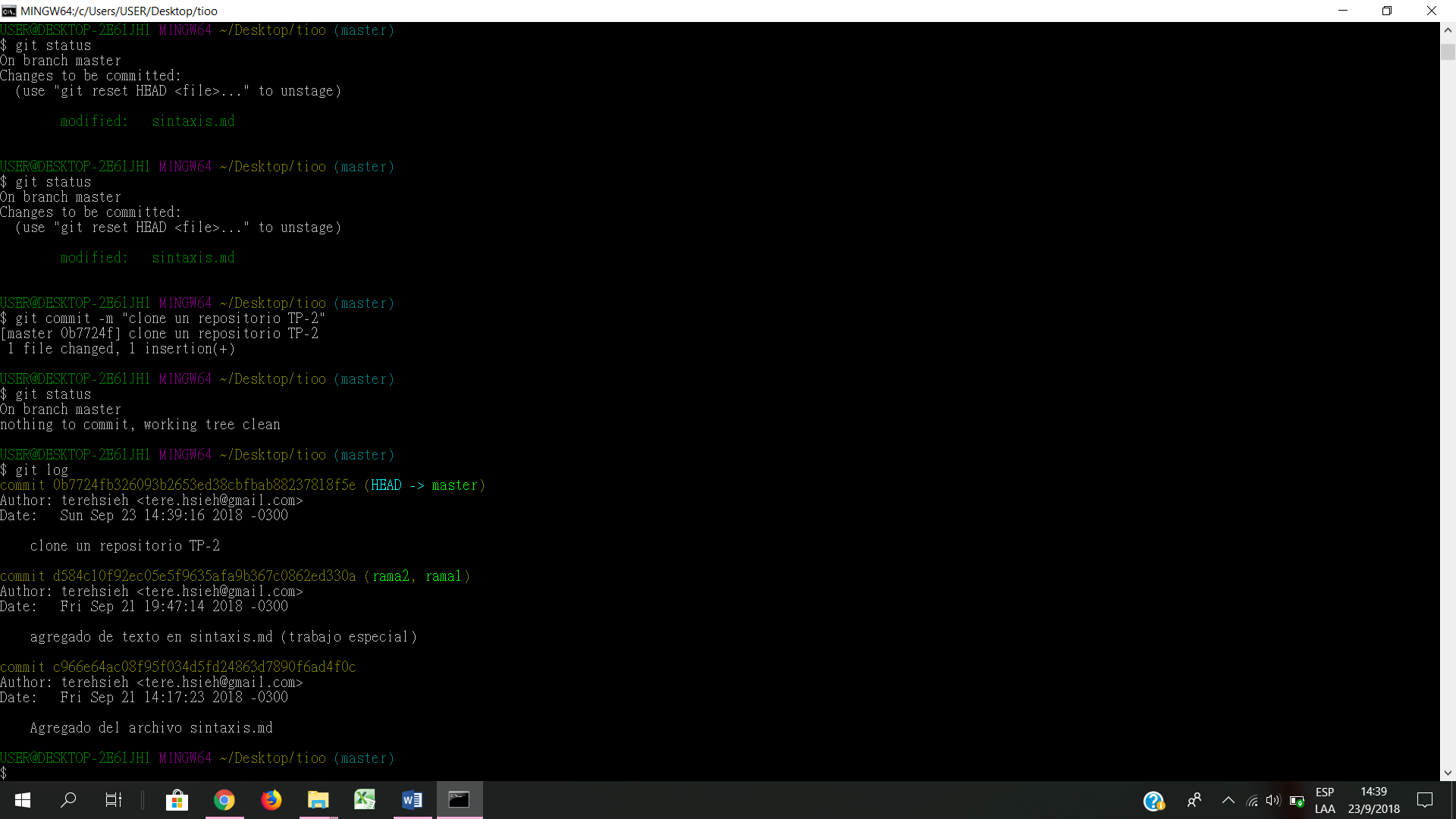
Eliminar una rama: *git branch -d* ***nombre de la rama***

Para saltar de una rama a otra, tienes que utilizar el comando:

*git checkout* ***nombre de la rama***.

La rama en el cual estamos trabajando se encuentra especificado el final de color celeste.

En nuestro ejemplo (master), y en la ultima de la foto aparece al final (rama1).



Con *git log*: proporciona una lista cronológica de cambios realizados detalladamente sobre el proyecto; e indica en qué rama está trabajando, con el apuntador HEAD🡪 **nombre de la rama.**

El comando *git merge* ***nombre de la rama:*** incorpora los cambios a la rama master para ponerlos en producción.

