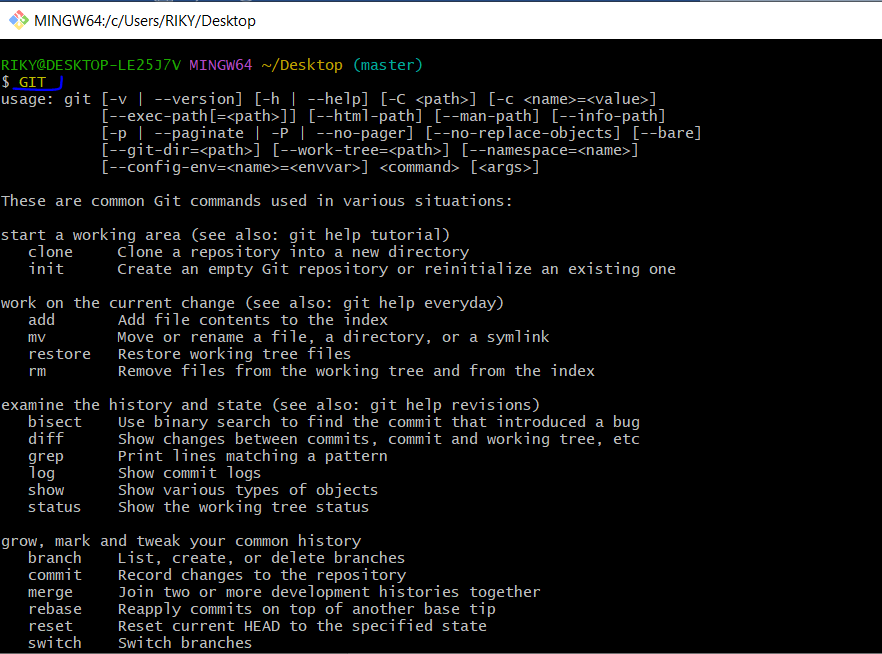
# GIT Y GIT HUB

Git es un control de versiones para la programación, para trabajar con Git es super sencillo. Cuando lo instalemos en Windows no solo se nos instalara git si no que también se nos instalara GIT BASH que es la terminal de GIT

## COMANDOS EN GIT

Como sabemos que Git esta instalado, pies entramos a la terminal y colocamos el comando git y nos saldrá toda la información sobre git



### BORRAR EN GIT

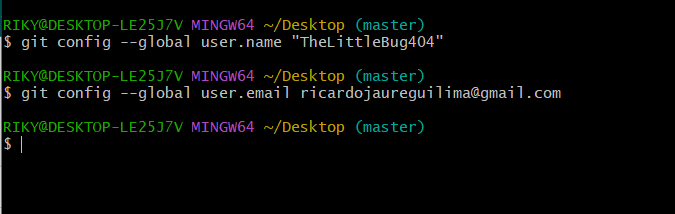
Para esto simplemente usamos el comando clear

### CONFIGURANDO GIT

Para esto simplemente colocaremos el comando git config – global user.name “……” donde en la parte de los …. (puntos) colocamos nuestro nombre

Y para configurar el email colocamos el comando git config –global user.email [ricardojaureguilima@gmail.com](mailto:ricardojaureguilima@gmail.com)

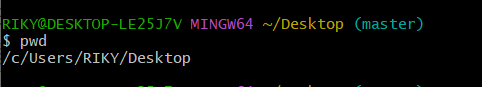
Y para configurar el entorno de desarrollo colocamos git config –global core.editor nombreEditor



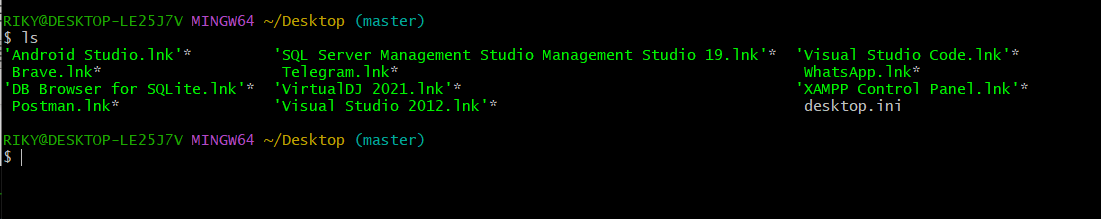
## COMANDOS BASICOS

Antes de empezar con git es necesario saber unos comandos básicos en git par poder manejarlo

* Comando para saber dónde estoy pwd este nos muestra la ruta en donde estamos dentro del git



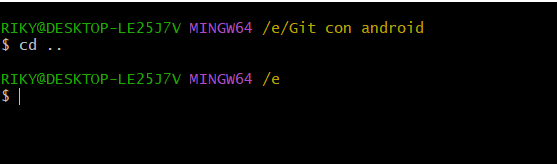
* Comando para mostrar un listado de nuestro directorio donde nos encontramos ls



* Comando para entrar a una carpeta cd observemos que para colocar una cadena debemos colocarlo entre comillas simples



* Comando para salir de una carpeta cd .. observemos que el comando es cd espacio ..

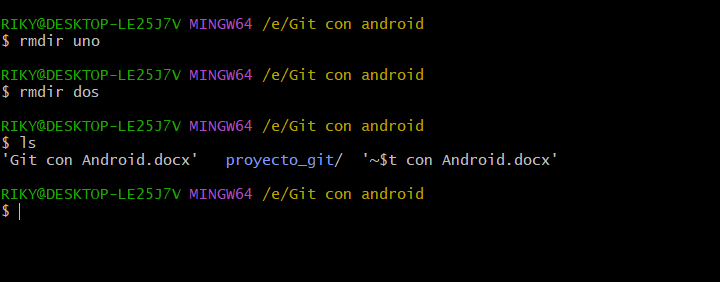


* Comando par poder crear un directorio donde podremos configurar nuestros proyectos el comando será mkdir nombreDirectorio nombreDirectorio observemos que con este comando podemos crear muchos directorios esto solo separándolos con un espacio.



Como podemos ver en la anterior imagen creamos una carpeta en la primer comando, en el segundo comando crearemos2 carpetas y para ver estos solo le colocamos el comando ls

* Comando para borrar un directorio es rmdir nombreDirectorioABorrar

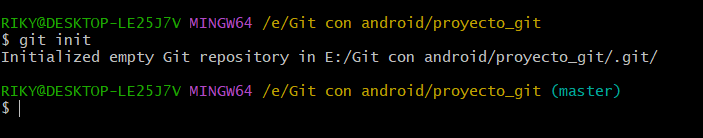


En la imagen anterior primero borramos el directorio uno luego borramos el directorio dos

## CREAREMOS UN ARCHIVO EN NUESTRA CARPETA

En esta ocasión crearemos un archivo en nuestra carpeta que en este caso se llama proyecto Git y lo llamaremos primer archivo esto lo haremos con el fin de entender un poco mas a cerca de git hub

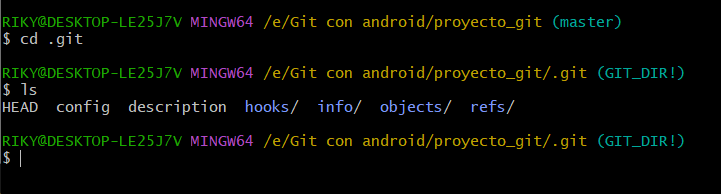
* Primero usaremos el comando git init esto con la finalidad de iniciar git en este archivo



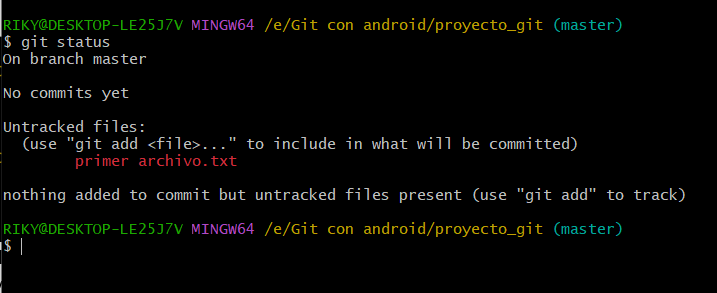
Y como podemos ver colocando el comando ls -a notamos que ya tenemos un archivo de git que esta oculto



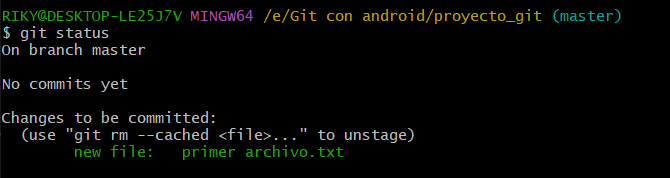
Si accedemos al directorio podremos ver lo siguiente, todo lo que se muestra a continuación es git este se prepara y se configura el solo a través de estos ficheros y es donde guardara la información de todos los cambios que le hemos hecho.



* VER EL ESTADO DEL PROYECTO esto lo haremos con el comando git status



* AÑADIR UN FICHERO AL GIT esto lo haremos con el comando git add nombre\_fichero si añadimos el fichero al directorio git y luego vemos el status nos saldrá de la siguiente forma



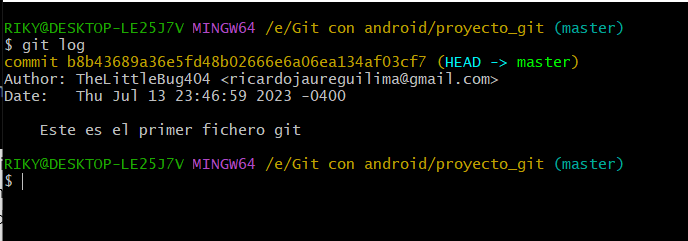
Con esto notamos que ya podemos empezar a trabajar en git

* COMMIT este comando sirve para guardar los cambios que nosotros realizamos en el git esto lo haremos con el comando git commit -m “Acá viene el mensaje” donde -m es el mensaje que nosotros queremos darle al git, este mensaje estará dentro de las comillas, por que le enviamos un mensaje a git esto es para poder ubicarnos sobre que es lo que hicimos dentro del git o darle un mensaje para saber cual es el cambio que realizamos



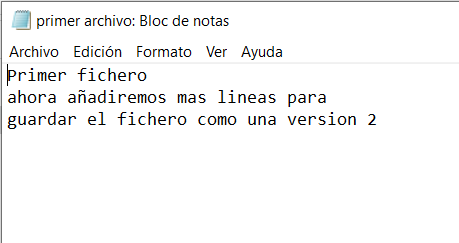
También notemos que si vemos el status ya no nos aparecerá como en los anteriores status que vimos

* COMO VER LOS COMMITS REALIZADOS ANTERIORMENTE para esto usaremos el comando git log

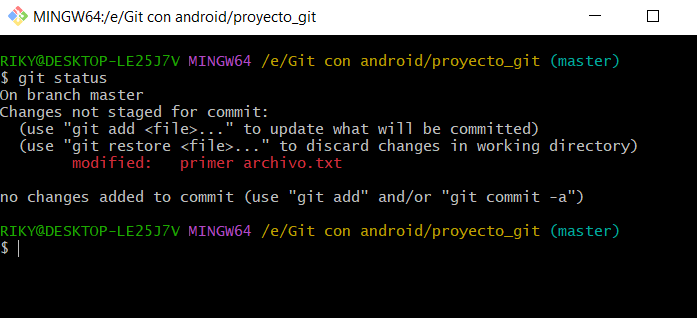


Y como vemos tenemos todos los commit que le hicimos al proyecto que tenemos

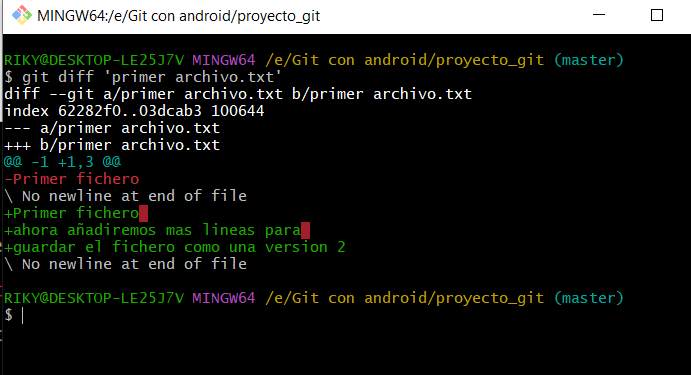
* AHORA MODIFICAREMOS NUESTRO FICHERO
* En este caso primero añadiremos mas líneas a nuestro fichero



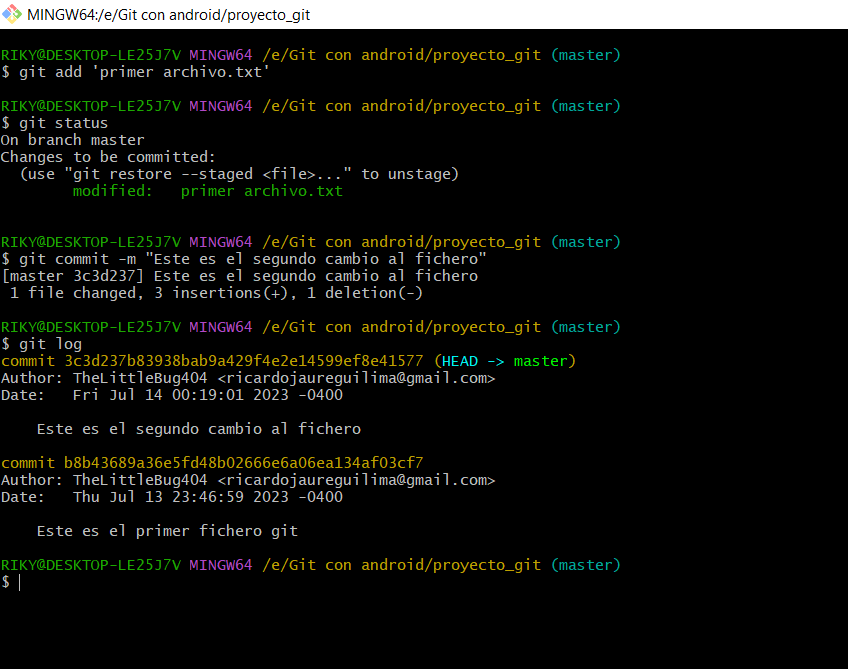
* Luego veremos el status con el comando git status, y como podemos ver a continuación git nos dice que se modifico el fichero



* Para poder ver las diferencias que le hicimos a nuestro fichero colocaremos el comando git diff nombre\_fichero esto nos mostrara todos los cambios que nuestro fichero ha tenido, en la siguiente imagen veremos los cambios que se realizaron en el git



* Si queremos volver a guardar estos cambios recuerda que primero tenemos que añadirlo al repositorio con el comando git add nombre\_fichero luego lo comiteamos con el commit y podemos ver este cambio con el log que nos muestra los commit que hicimos anteriormente



## COMO ACCEDER A LOS COMMIT ANTERIORES

Para esto simplemente usaremos el comando git checkout código\_commit donde el código del commit se puede ver al hacer el comando git log

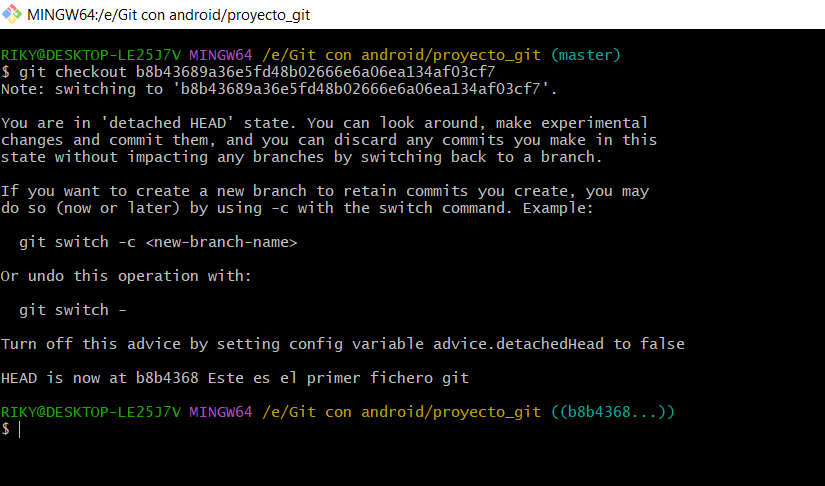


Con esto podremos entrar a cualquier commit que queramos, y al hacer esto volveremos nuestro código en este caso nuestro fichero a la versión que entremos del commit.

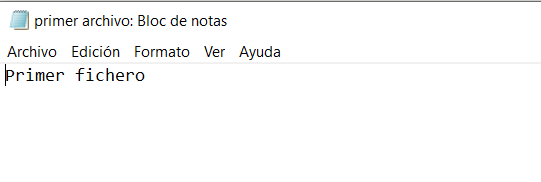
En este caso entraremos a la primera versión del fichero y veremos que las líneas que hicimos en el segundo cambio ya no estarán.

Ejemplo

Primero entraremos a la primera versión que tenemos de nuestro commit



Ahora si vemos el archivo notamos que volvió a ser como en la primera versión



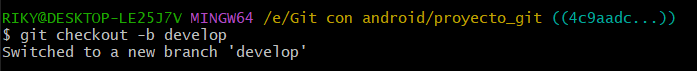
Y Si hacemos de nuevo el comando git log notamos que el segundo commit ha desaparecido



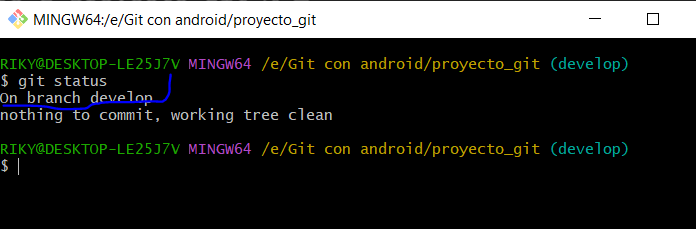
## RAMAS, MERGE Y REBASE

Ahora veremos las ramas en git

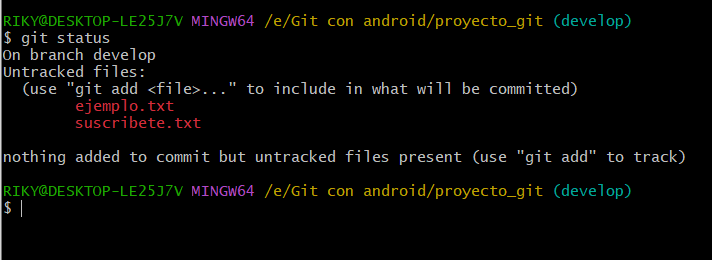
* CREAR UNA NUEVA RAMA Y ENTRAR EN UNA RAMA esto lo haremos con el comando git checkout -b nombre\_rama con este comando crearemos la nueva rama



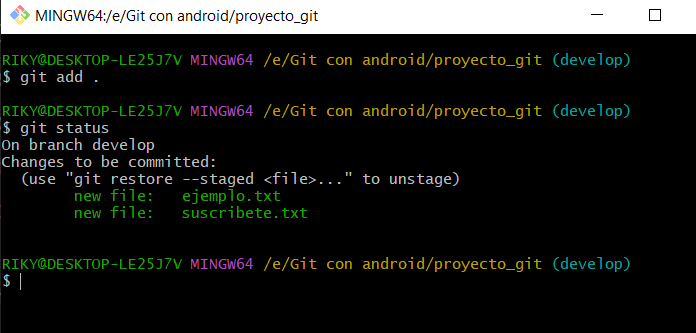
En este caso nuestra rama la llamaremos develop notemos también que no solo nos crea la rama develop si no que también nos lleva a ella si colocamos el comando git status nos damos de cuenta que estamos en la rama develop



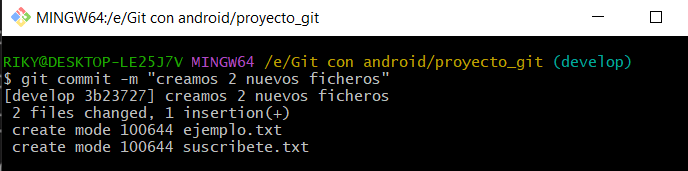
* Para continuar este ejemplo nos crearemos ficheros y luego veremos su status, notaremos que nos mostrara los ficheros que nos creamos que en este caso los ficheros creados son ejemplo y suscribete



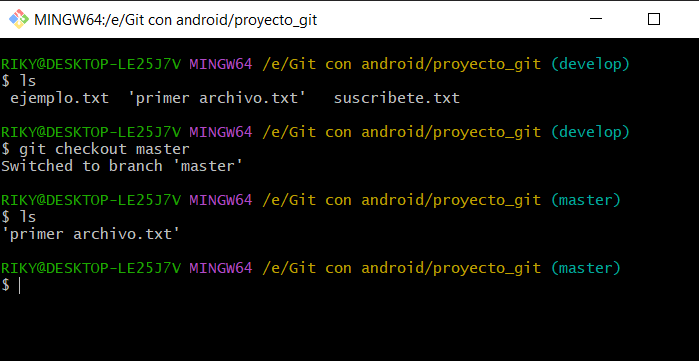
* Añadimos los archivos, esto es muy importante no nos olvidemos añadir los archivos al fichero siempre debemos añadirlos



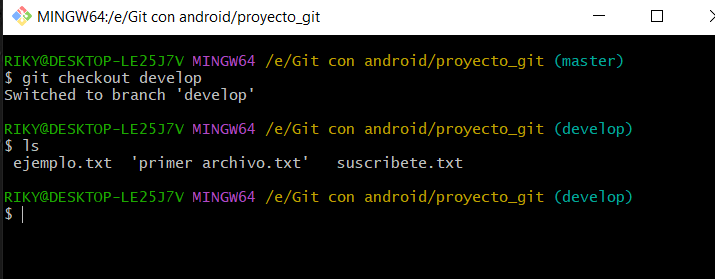
* Ahora los guardamos con el commit



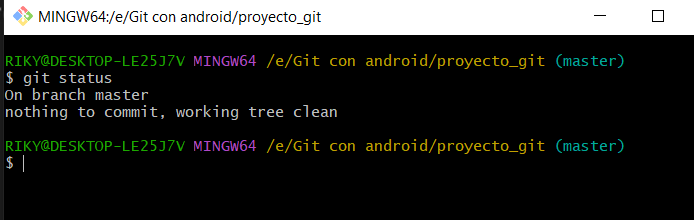
* Ahora volveremos ala rama master y veremos que solo tenderemos el primer fichero que nos creamos recordemos que los otros 2 ficheros están en la rama develop



* Pero si volvemos a la rama develop notamos que están los otros ficheros mas



* COMO TRAEMOS TODO LO QUE TENEMOS EN LA RAMA DEVELOP A LA RAMA MASTER (MERGE)primero nos fijamos el status para saber en que rama estamos para esto notamos que estamos en la rama master



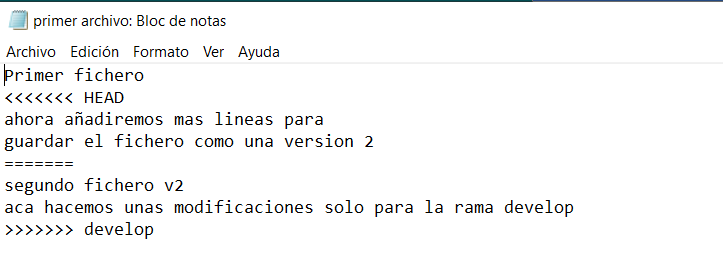
Ahora como estamos en la rama master y es la rama a la que queremos traer todo lo que tiene la rama develop simplemente usamos el comando git merge nombre\_rama donde recordemos que nombre\_rama es la rama desde donde queremos traer nuestros archivo



Como podemos notar nos dará un error esto será por que en la rama master el archivo llamado “primer archivo.txt” esta así:



Pero en la rama develop el archivo “primer archivo.txt” se encuentra de la siguiente forma:



Con esto git se hace un problema por que no sabe que archivo juntar, recordemos que estamos tratando de traer los archivos de la rama develop a la rama master.

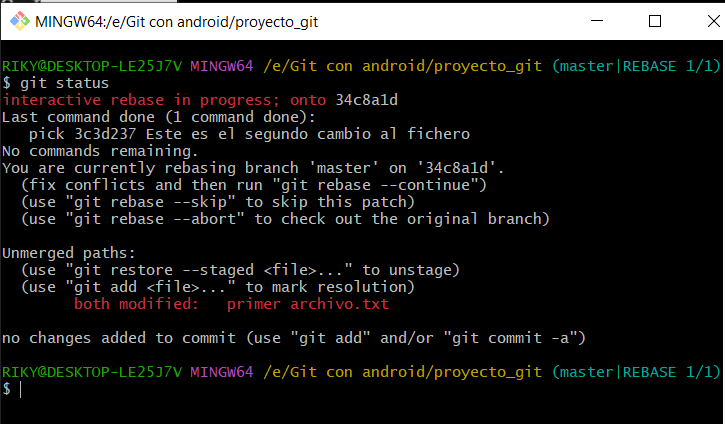
Para esto tenemos 2 soluciones en un IDE que lo veremos mas adelante como hacerlo o colocarl el comando git merge –abort



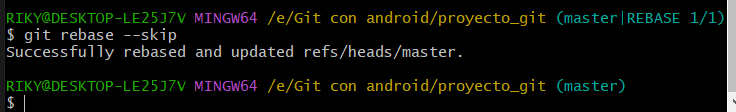
* REBASE: Con esto podemos meter la rama “develop” dentro de la rama “master” recordemos que la rama master siempre estará en el git pero la rama develop es la que nos creamos para este ejemplo, pero esto dará mas prioridad a la rama que queremos meter en este caso develop ya que queremos meter esta rama en master, esto es con el comando git rebase nombre\_rama donde el nombre de rama es la rama que queremos meter a otra rama



Como podemos ver nos muestra que tenemos conflictos al meter la rama develop a master por que nuestro archivo en la rama develop es distinto al de la rama master, si hacemos un status veremos



Para olvidar todo esto o para pasar de esto simplemente colocaremos git rebase –skip con esto le decimos al git que se olvide de los errores del status

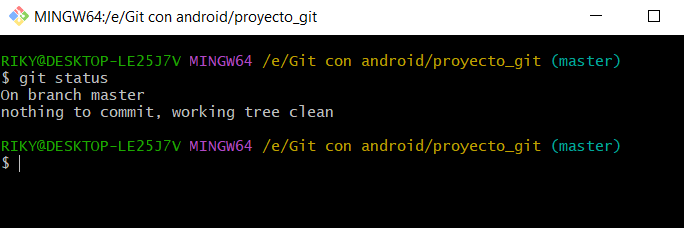


## STASH

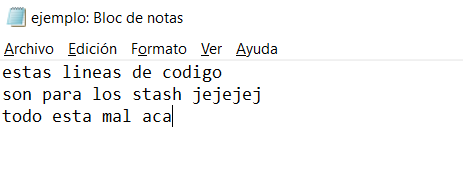
Los stash nos sirven para nos sirven cuando no queremos guardar, ya sabemos que nosotros guardamos con los commit entonces que pasaría si nosotros no queremos guardar una rama dentro de otra pero no queremos llevarnos los cambios de estas rama, con es stash podemos guardar estos códigos o documentos lo guardaremos fuera donde no moleste a nadie pero esta guardado pero cuando volvamos a rama que queremos lo podemos volver a traer.

EJEMPLO

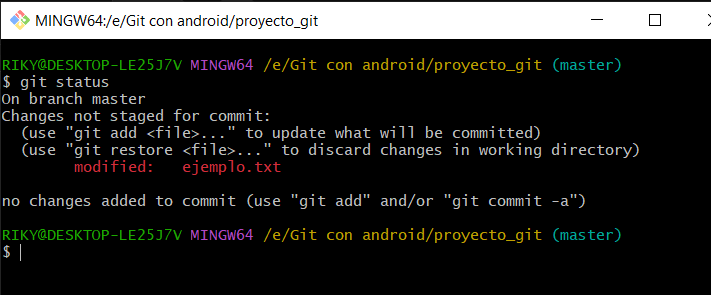
* Primero nos fijamos en que rama estamos



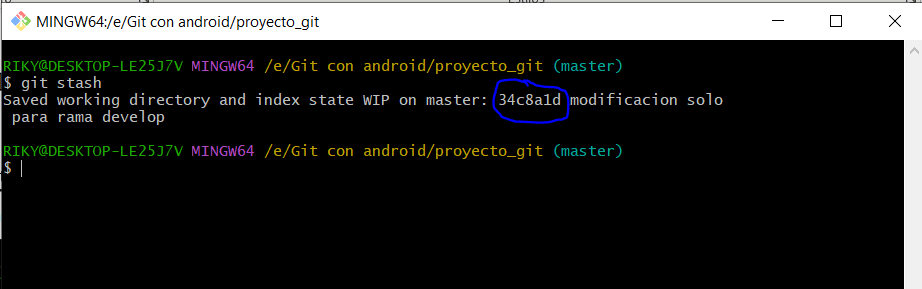
* Ahora entraremos al archivo ejemplo y añadiremos unas líneas de código



* Ahora si vemos el status notamos que nos dice que el archivo ejemplo fue modificado

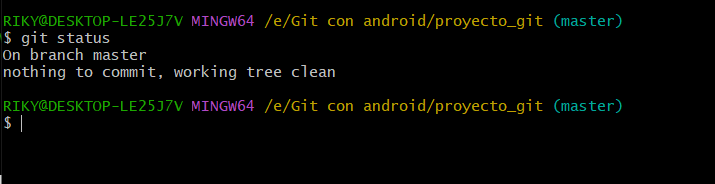


* Pero no quiero guardar estos cambios y tampoco los quiero perder para esto nos sirve el comando git stash además nos dará la dirección donde se guardarán estos cambios que le hicimos al archivo ejemplo.

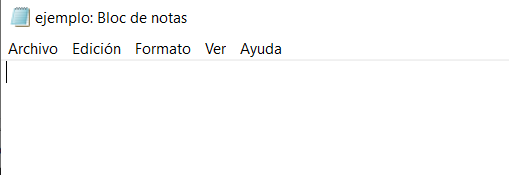


Notemos que para este caso la dirección donde se guardo esta redondeado con azul en la anterior imagen

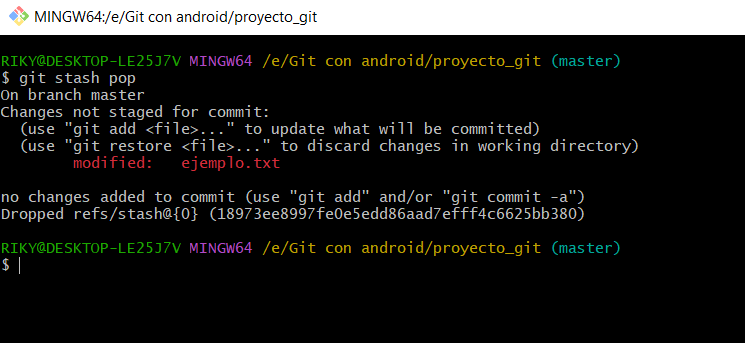
* Ahora si hacemos u git status nos dirá que ya no tenemos nada que guardar



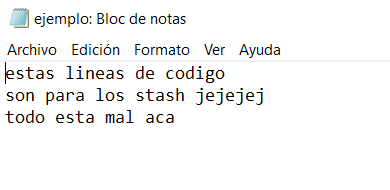
* Si vamos al archivo ejemplo notaremos que ls cambios que hicimos ya no se encuentran



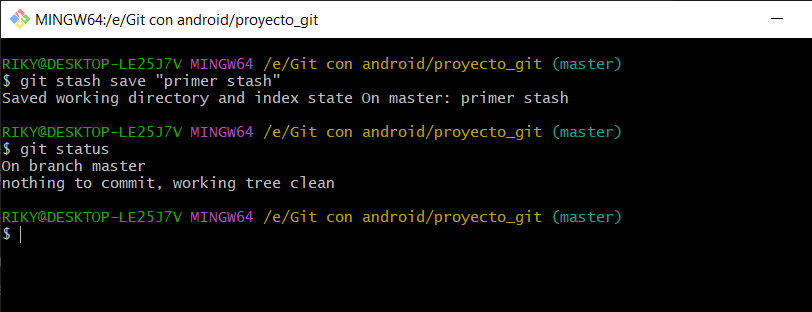
* Pero donde están estos cambios que hicimos con el stash? Si queremos volver a ver los cambios que hicimos con el stash usaremos el comando git stash pop y con esto veremos los cambios que hicimos y además el status nos mostrara que necesitamos hacer un commit



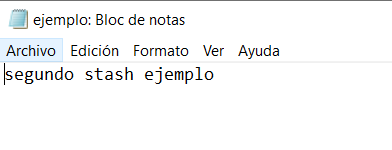
* También notemos que el archivo ejemplo volvió a estar como estaba



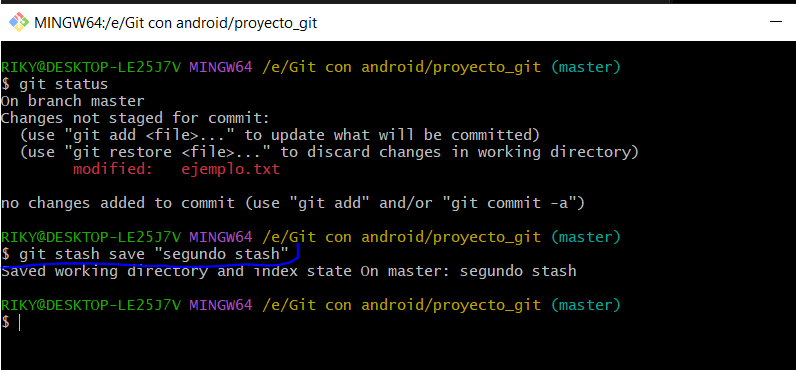
* QUE PASA SI TENEMOS MAS DE UN STASH Y QUEREMOS GESTIONARLO esto simplemente lo hacemos con el comando git stash save ”acá viene un comentario” con esto podemos guardar mas de un stash en el git como se muestra a continuación. Primero guardaremos el primer stash y además si hacemos un status notemos que ya no tenemos que guardar



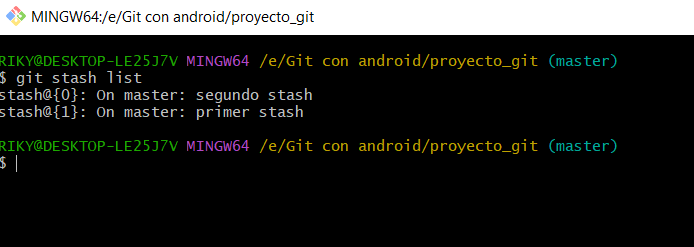
Ahora modificaremos el archivo ejemplo y guardaremos un segundo stash



Guardamos además si hacemos un status notaremos que ya lo tenemos modificado



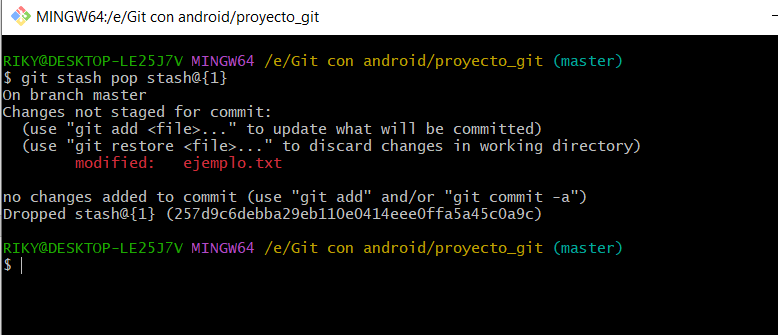
Ahora para ver los stash guardados colocaremos el comando git stash list con esto nos mostrara los stash que nosotros guardamos:



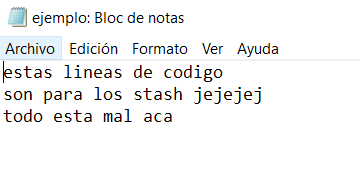
Como podemos entrar al stash que queremos simplemente copiaremos el numero de stash que queremos usar



Y usaremos el comando git stash pop stash\_copiado y listo donde stast\_copiado es el stash que esta con circulo azul en al anterior imagen, en este caso copiaremos el primer stash



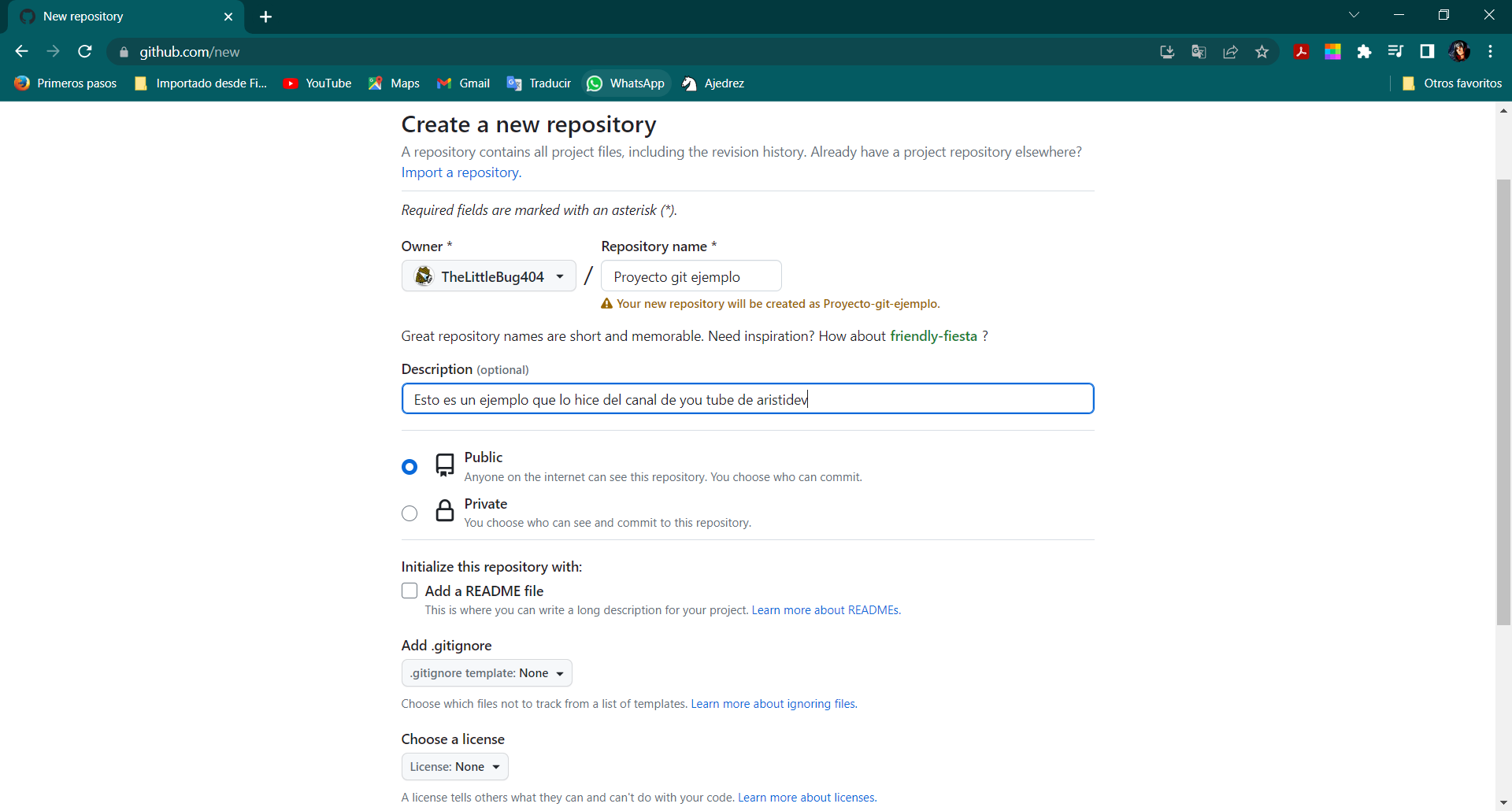
Enseguida notaremos que nos dice que tenemos que guardar y además que si entramos al archivo ejemplo nos volverá aparecer las líneas de este stash

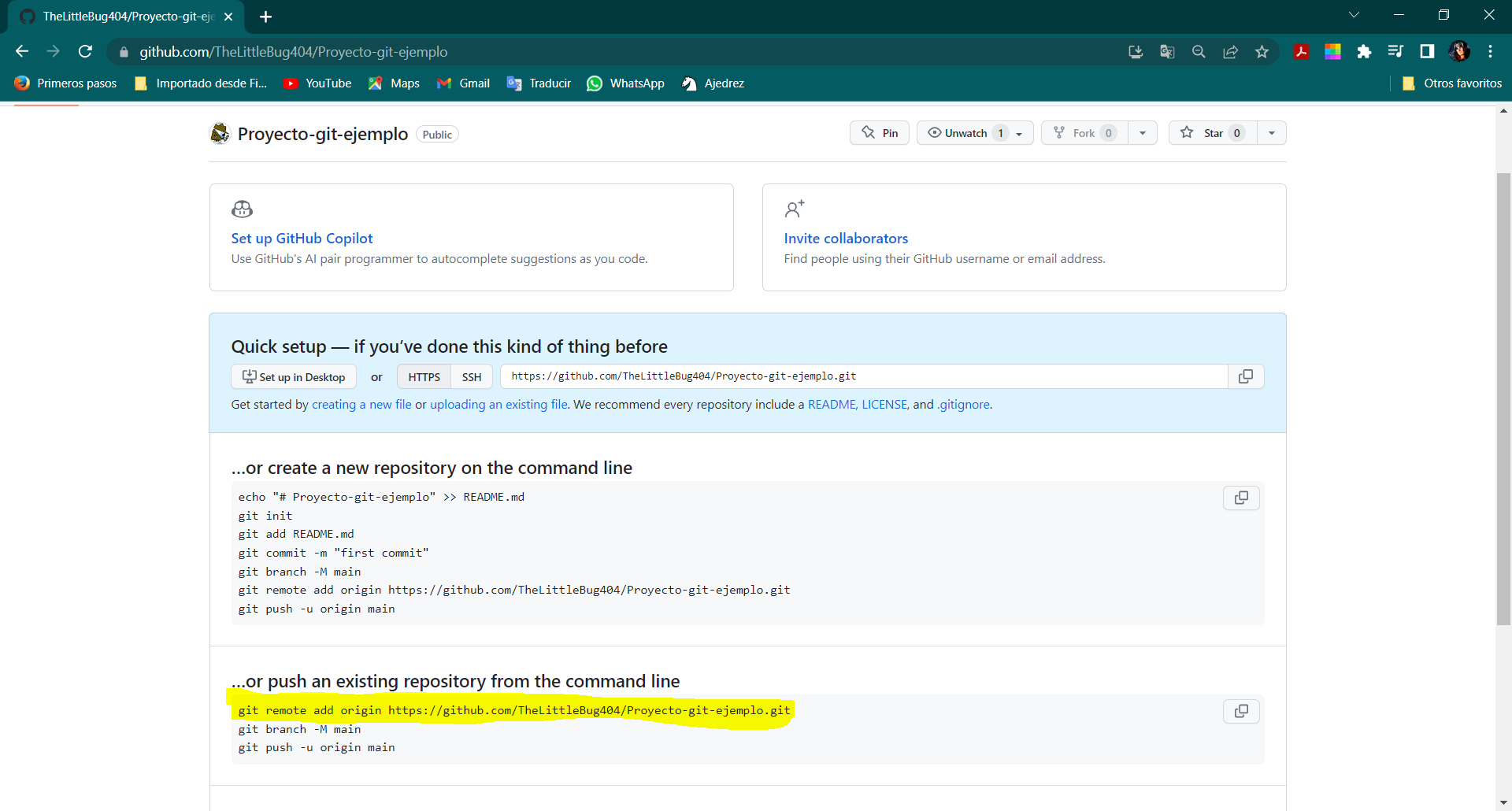


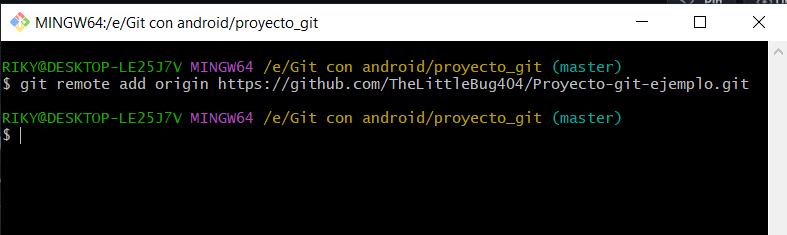
CON ESTO YA ACABAMOS TODO SOBRE EL GIT LOCAL AHORA CONFIGURAREMOS NUESTRO GIT HUB O GIT LAB

# GIT HUB

Creamos un repositorio

Una vez creado copiaremos la siguiente línea de código a nuestra terminal





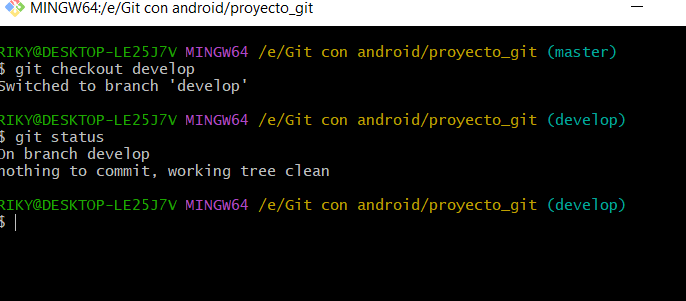
Ahora ya tenemos enganchado el proyecto local que esta en nuestra computadora con el que esta en git hub

## PUSH LO USAREMOS PARA ARRIBA

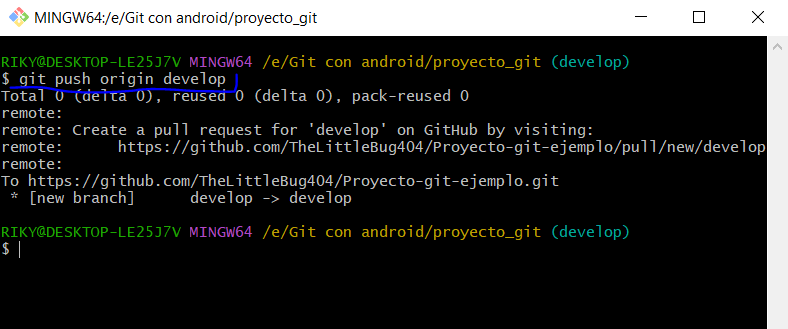
Con este comando podemos subir archivos a la rama que queremos simplemente con el comando git push origin nombre\_rama Pero que pasa si hacemos enter nos pedirá un usuario del git hub (en este caso no me lo pedirá por que ya tenemos una



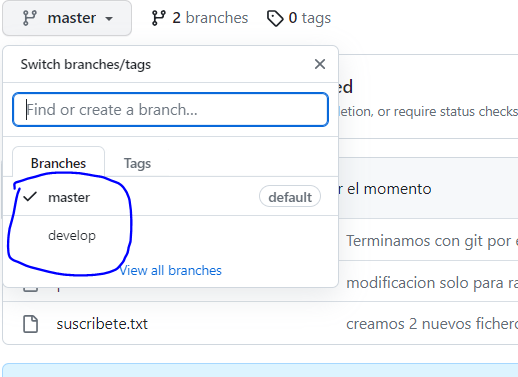
Ahora también subiremos la rama develop para esto primero nos vamos a la rama develop de nuestro ejemplo



Luego lo subiremos al git hub con el siguiente comando git push origin nombre\_rama\_a\_subir y listo donde en este ejemplo el nombre de la rama a subir es develop y listo



Listo con esto ya esta nuestra rama que creamos en el ejemplo ya lo subimos con esto ya tenemos 2 ramas en nuestro caso develop y master



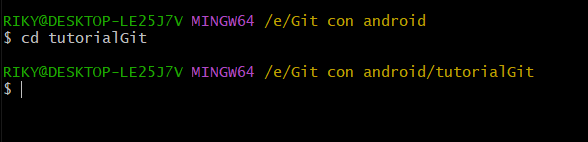
## CLONAR PROYECTO Y TRABAJAR EN OTRO EQUIPO

Para esto primero usaremos git bash nos fijaremos que todos los comandos de Linux, unix, mac funcionaran gracias a esta herramienta

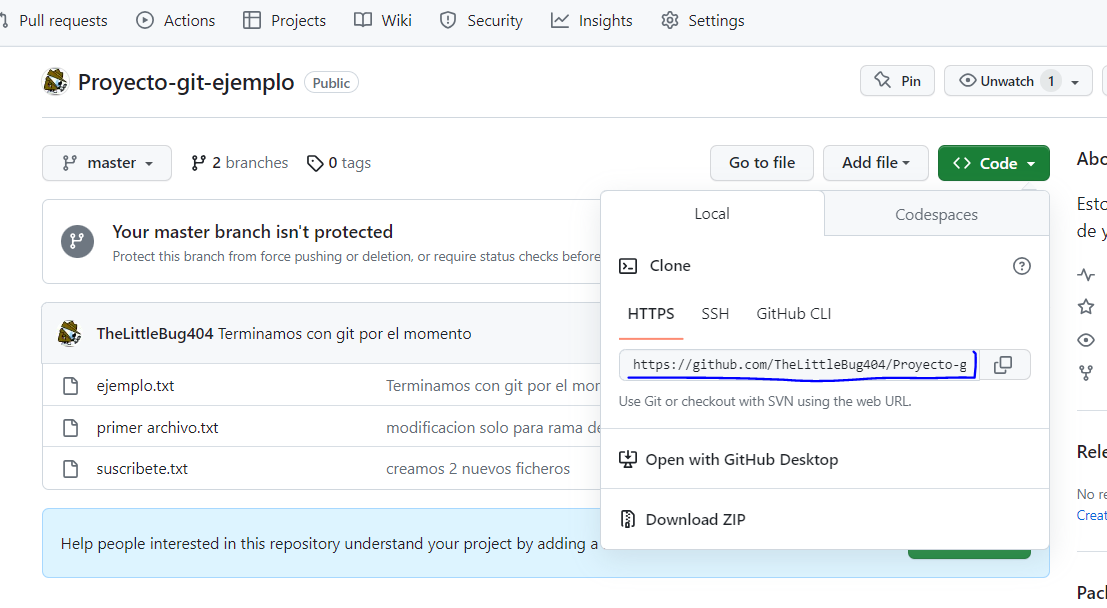
* Nos creamos un nuevo directorio con el comando mkdir



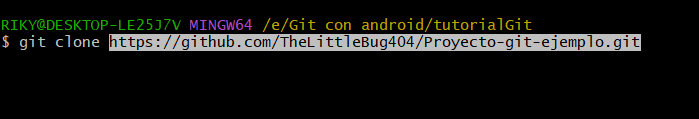
* Luego entraremos al directorio que creamos con el comando cd



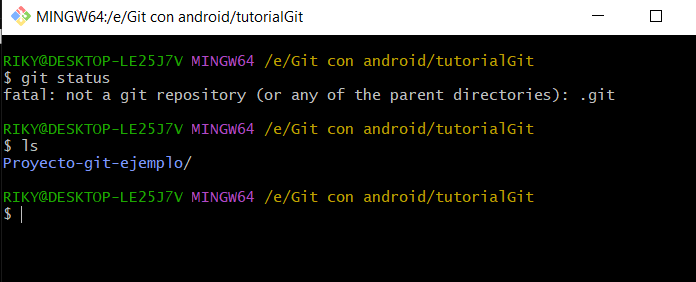
* Para tomar el proyecto de Git Hub y colocarlo en nuestro ordenador primero iremos al git hub vamos al proyecto que queremos clonar, copiamos la ruta del proyecto que queremos clonar



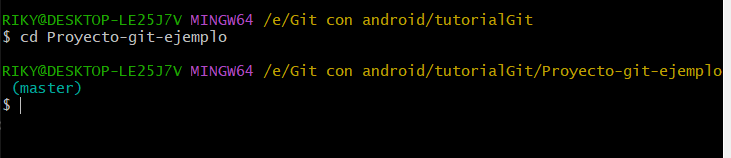
* Luego clonaremos el proyecto con el comando git clone ruta\_copiada, observemos que para pegar en git hub tenemos que hacerlo con click derecho y copiar en acá no sirve control +c



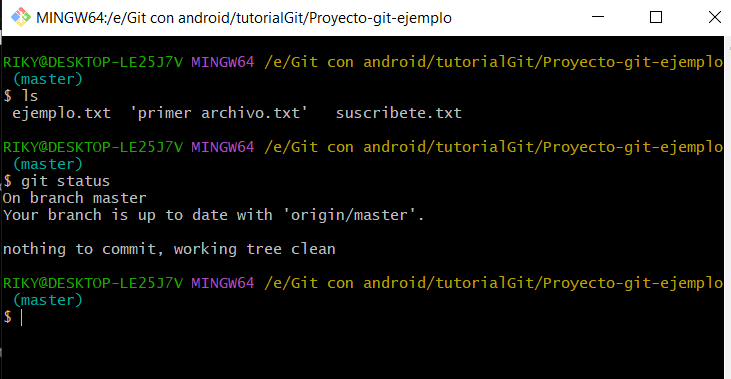
* Lo que conseguimos con esto es acceder al repositorio externo y traernos el contenido y si hacemos un git status notamos que no funciona por que si colocamos ls veremos que nos trajo el proyecto con su directorio



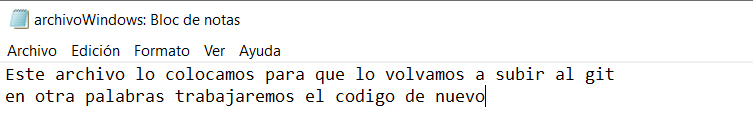
* Así que lo primero que tenemos que hacer es ir a ese directorio con el comando cd



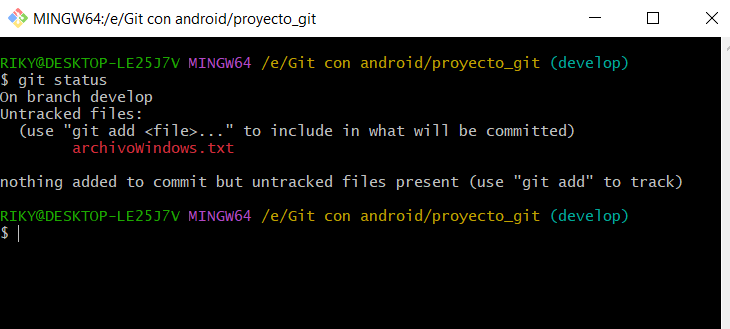
* Ahora si colocamos ls nos fijaremos los ficheros que teníamos y también veremos el status del proyecto



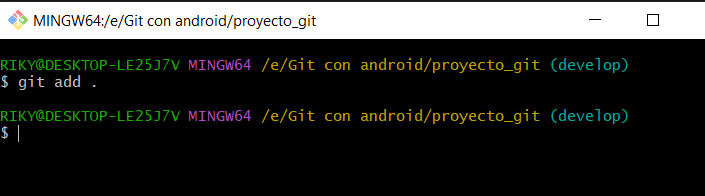
* Ahora con el repositorio clonado nos crearemos un nuevo archivo y lo subiremos de nuevo al git hub



* Git status



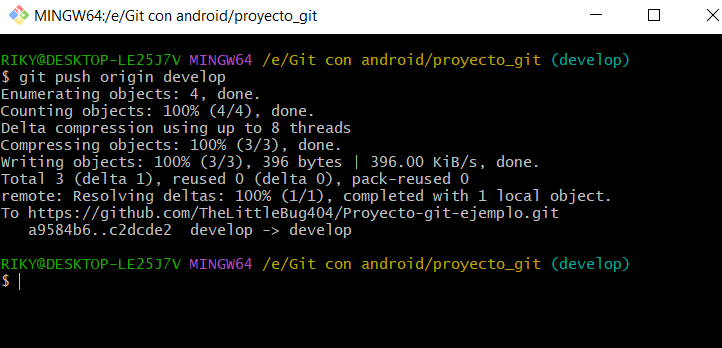
* Git add .



* Git commit



* Subiendo al Git hub con git push origin master en este caso lo subiremos a la rama develop por que estamos en la rama develop

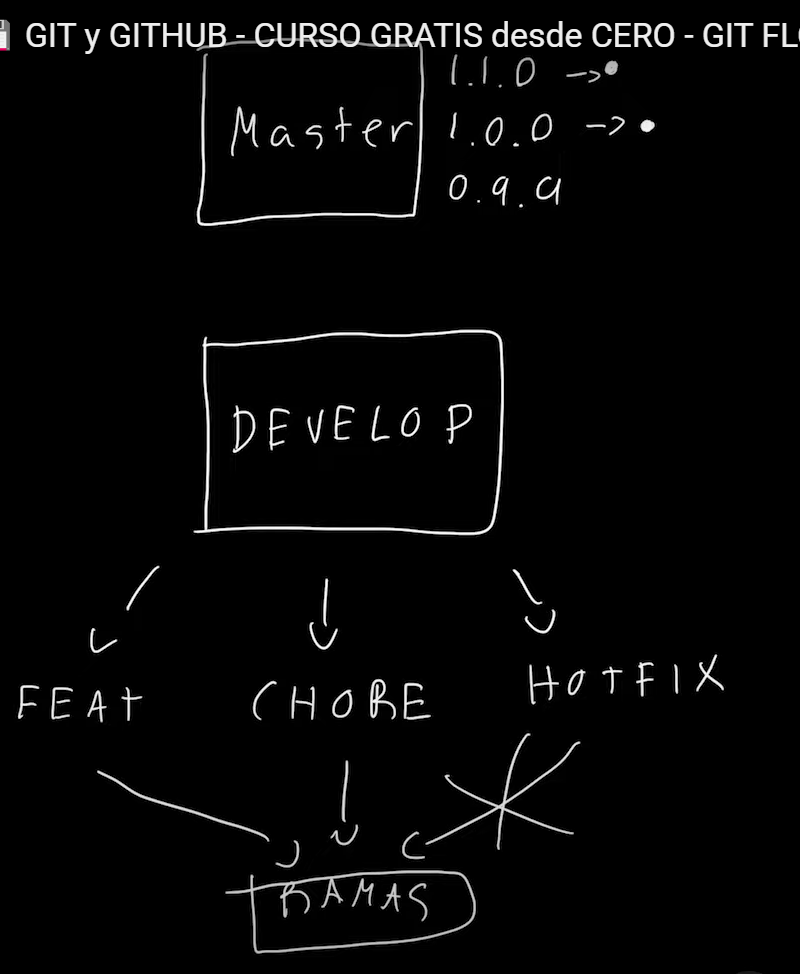


Con el PULL nos traemos los cambios que le hacemos a nuestros ficheros en git

ANTES DE HACER UNPUSH SIEMPRE HACER UN PULL: Significa que nosotros tenemos que traer todo el contenido que esta en el origen traérnoslo ver que no hay conflictos y luego subir lo nuestro.

# GIT FLOW

Git Flow es una metodología para trabajar con las ramas de una forma bastante eficaz pero rápida



Te quedaste en el minuto 57