**Git Bash**

# Configuración inicial de Git Bash.

* Usuario para un repositorio:

\>git config user.name “EdgarVera80”

* Email

\>git config user.email “[edgar.vera80@gmail.com](mailto:edgar.vera80@gmail.com)”

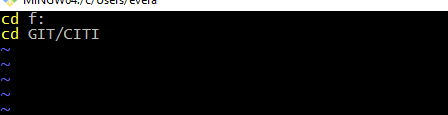
* Para que la configuración sea global (todos los repositorios):

\>git config --global user.name “evera”

* Cambiar el directorio donde inicia

Abrir (Crear sin no existe) el archivo **.bashrc.**

Escribir el directorio donde se desea que incie.



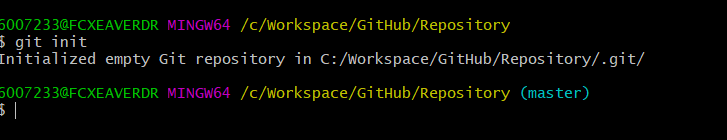
# Deshabilitar el warning LF.

\>git config core.autocrlf false

# Crear repositorio local

Hacer un directorio repositorio.

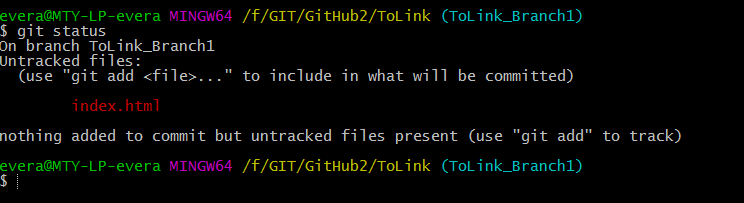
Nos movemos al directorio que queremos convertir en repositorio y se ingresa el comando \>git init, esto nos creará la línea **master**.



# Status.

Muestra el estado actual de un repositorio, es decir, si existen cambios pendientes.

Para el siguiente ejemplo se creó un archivo llamado index.html. y se muestra como **Untracked** (Cambios que no han sido registrados)



# Add.

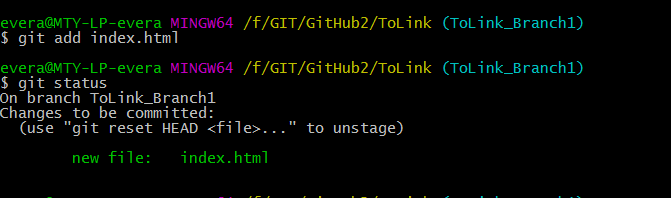
Envía los archivos marcados como **Untracked** al **Staging Area** (Index) para que estén listos para el **Commit**.

* Para un solo archivo.

\>git add **Nombre\_de\_archivo**

* Todos los archivos.

\>git add .



# Reset.

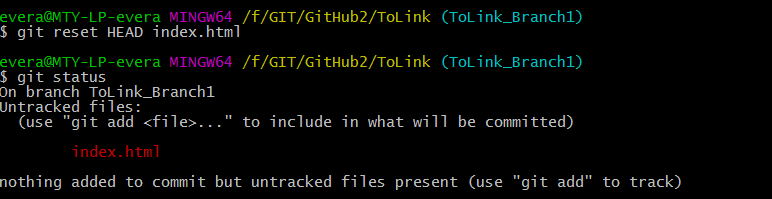
Quita el o los archivos del **Staging Area** (Index). Usar HEAD para indicar el último commit.

* Para un solo archivo.

\>git reset HEAD **Nombre\_de\_archivo**

* Todos los archivos.

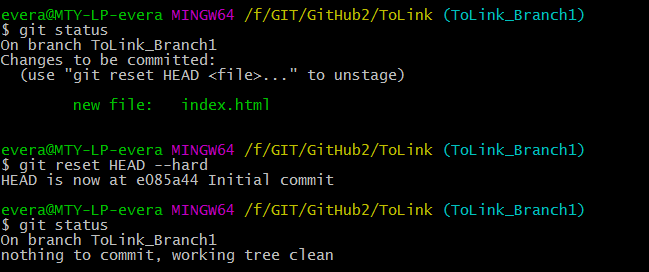
\>git reset HEAD .



# Reset --hard.

Quita todos los archivos del **Staging Area** (Index) y los regresa a su estado original, es decir, antes de cualquier modificación. Si el archivo es nuevo y aún no se ha hecho ningún Commit, éste será eliminado. Usar HEAD si no es la rama Master.

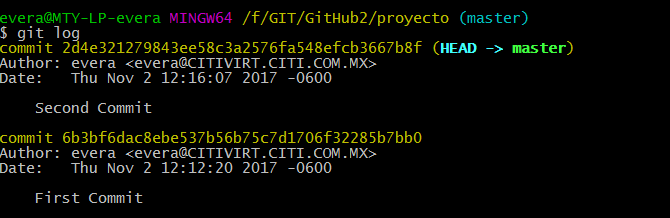
\>git reset HEAD --hard



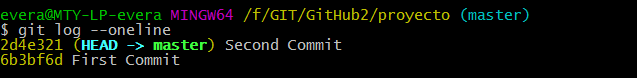
# Historial de Commits (Log).

Muestra el historial de Commits que se han realizado.

\>git log

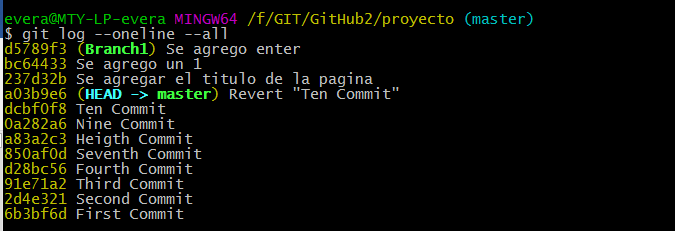


\>git log –oneline



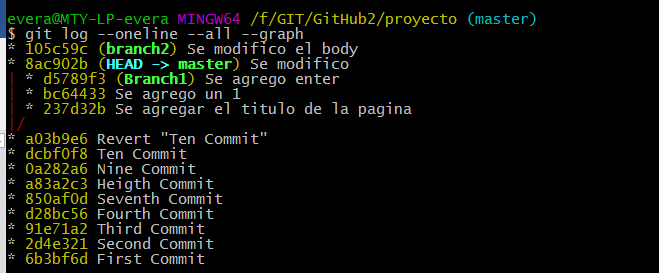
**Ver el historial de todos los branches**

\>git log –oneline –all



**Ver al grafo de los Commits**

\>git log --oneline --all --graph



# Diferencias (diff).

Muestra las diferencias de un archivo modificado antes de ser enviado al Staging Area con respecto al original o al que esté en el Staging Area.

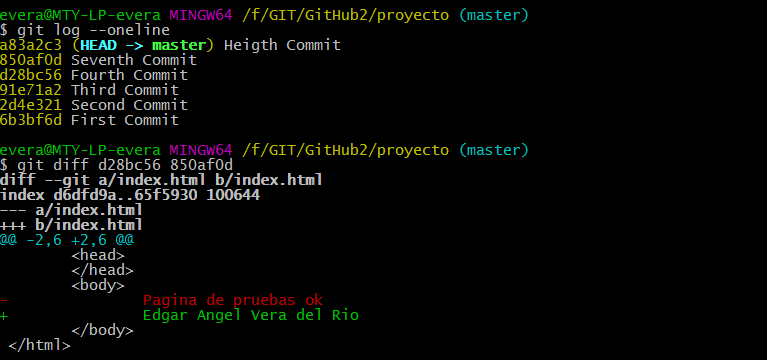
En el ejemplo las líneas con el signo “+” (Verde) indican lo que se ha agregado con signo “-” (Rojo) lo que se ha quitado.



**Diferencia entre dos Commits**

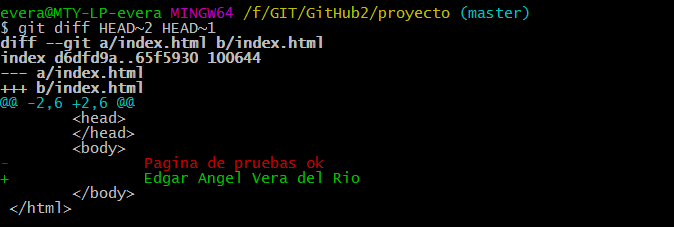
\>git diff **commit\_mas\_antigüo commit\_mas\_actual**

\> git diff d28bc56 850af0d



# Utilizando **HEAD** para referirnos al último Commit más el número de commits anterior a éste.

# \>git diff HEAD~2 HEAD~1

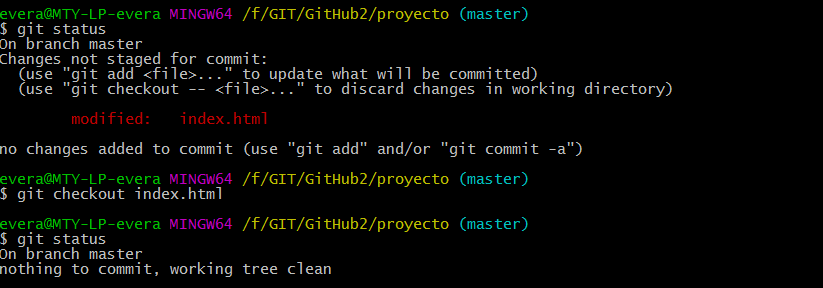


# Checkout

Switchea branches o restaura archivos modificados con la última versión del repositorio.

* Restaurar un archivo: Restaura un archivo local a la versión del último **Commit**, siempre y cuando el archivo modificado no se haya enviado al **Staging Area**.

\>git checkout -- **Nombre\_del\_archivo**



* Restaurar al último commit:

\>git checkout HEAD code/transferFormApps/src/mx/com/citi/transfer/controller/RegisterFormController.java

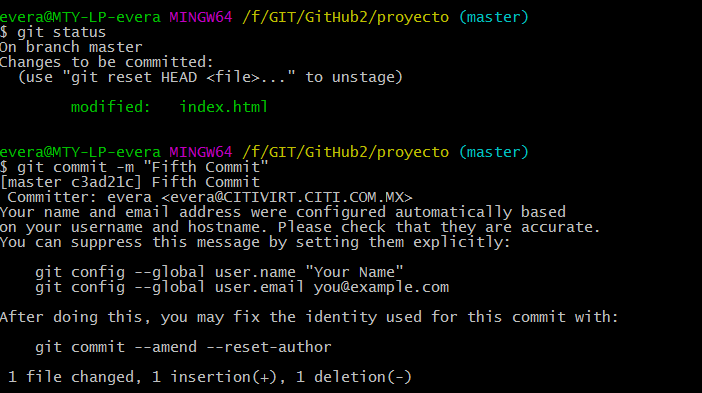
# Commit

Permite hacer un commit de los cambios al branch local.

\>git commit

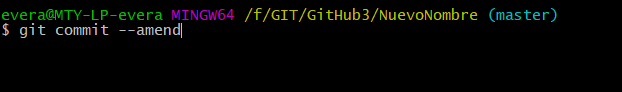
-m : Permite agregar un mensaje directo en la línea de comandos.

-a : Permite saltarse el Staging Area.

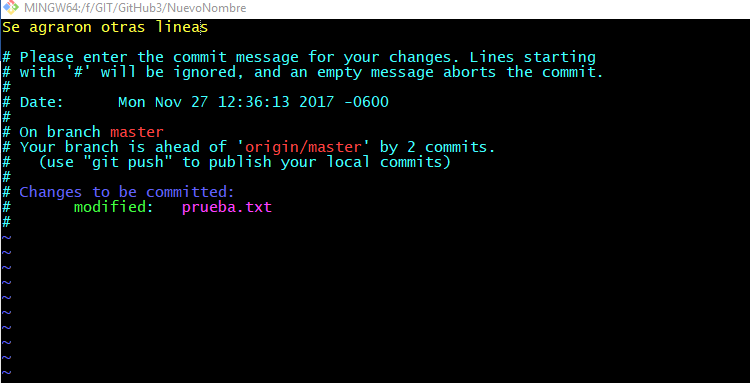


# Agregar cambios al último Commit

\>git commit –amend



Abre el archivo como el mensaje del último commit



# Regresar a un Commit anterior.

\>git reset **Commit\_code**

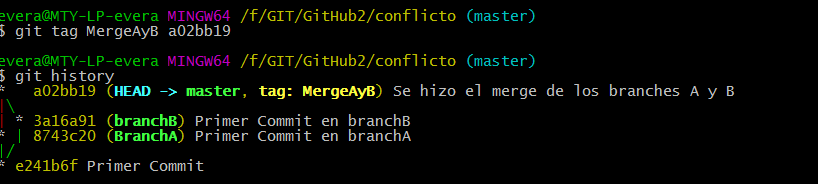


# Revertir un commit.

\>git revert **Commit\_Code**

# Tag para un Commit.

\>git tag nombre\_a\_usar Commit\_code

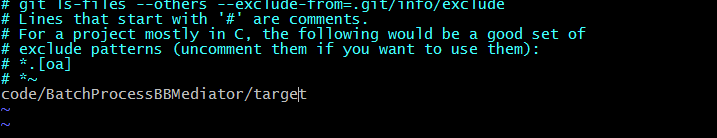


# Ignore (exclude).

Permite agregar archivo para que sean ignorados.

\>vi .git/info/exclude

Ejemplo de exclusión del directorio **target**.

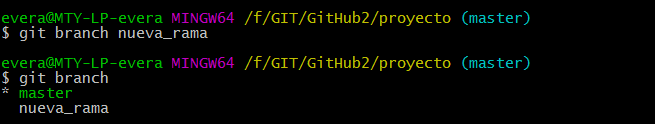


# O crear .gitignore

# Ramas (branch).

**Creación**

\>git branch **Nombre\_rama**



**Visualizar las ramas existentes**

* **Locales**

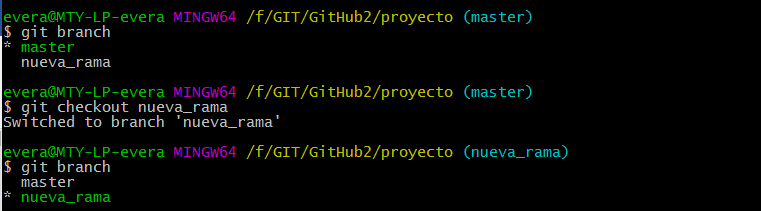
\>git branch

* **Remotas**

\>git branch -a

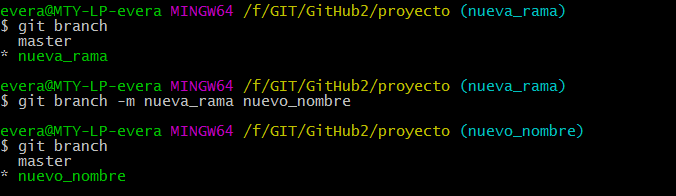
# Cambiarse de Rama (checkout)

\>git checkout **rama\_a\_cambiar**



# Cambiar de nombre a una rama

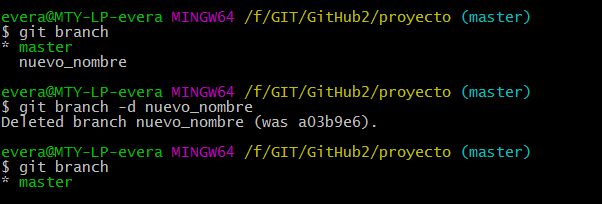
\>git branch -m **nombre\_actual nuevo\_nombre**



# Eliminar rama

\>git branch -d **rama\_a\_eliminar**

\>git branch -D **rama\_a\_eliminar** (Forzar la eliminación)

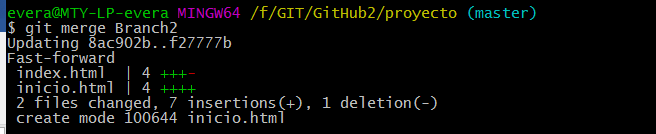


# Merge

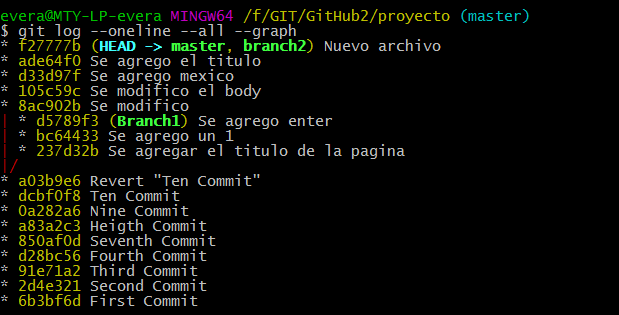
Se debe mover a la rama a la cual se le integrarán los cambios (Rama destino)

\>git merge rama\_origen

En este caso el Merge se resolvió por Fast-Forward debido a que la rama Master no se tocó y simplemente se sobre escribe con la rama.

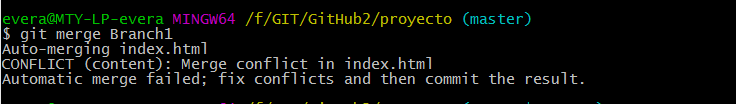


El grafo ahora muestra

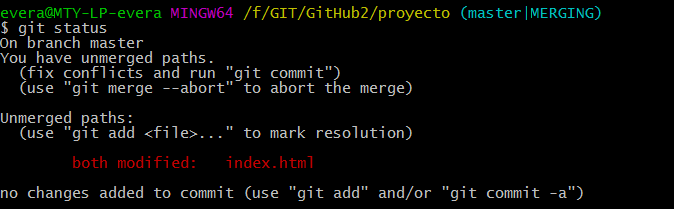


**Conflictos**

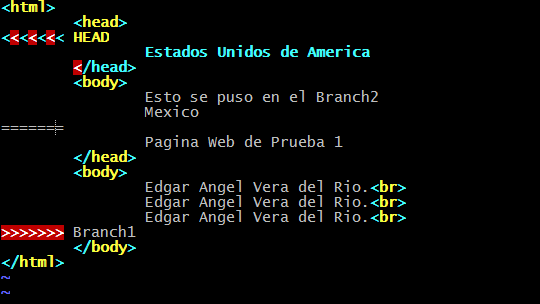
Cuando existen conflictos en el Merge se muestra lo siguiente:



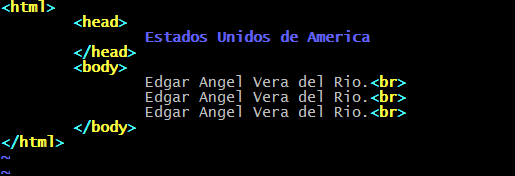
El estatus muestra lo siguiente:



En el archivo con el conflicto se muestra lo siguiente, donde “<<<<<< HEAD” indica lo que pertenece en el branch actual, y debajo de “=============” lo que pertenece al branch origen hasta “>>>>>>>>>”.



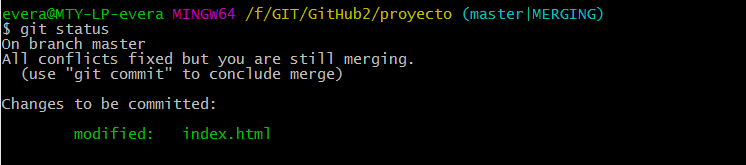
Corregimos el archivo



Agregamos al Staging Area

\>git add .

Nos muestra que los conflictos se han corregido, pero aún no se ha hecho el Commit.



Se hace el Commit

\>git commit -m “Mensaje”



**Deshacer el Merge**

\>git merge --abort

# Merge con checkout y sólo a ciertos archivos

1 Mover se al branch al que se desea hacer merge

\>git checkout [branch] [Archivos]

ejemplo

\>git checkout remotes/origin/feature/TRANSFERDATA-5489-UAT-2.0 code/TransferAppsWs/.

# Alias.

Especificar un Alias para una instrucción.

\>git config –global alias.history ‘log --oneline --all --graph'



**Ver los Alias creados.**

\>git config –global --get-regexp alias



**Eliminar un Alias.**

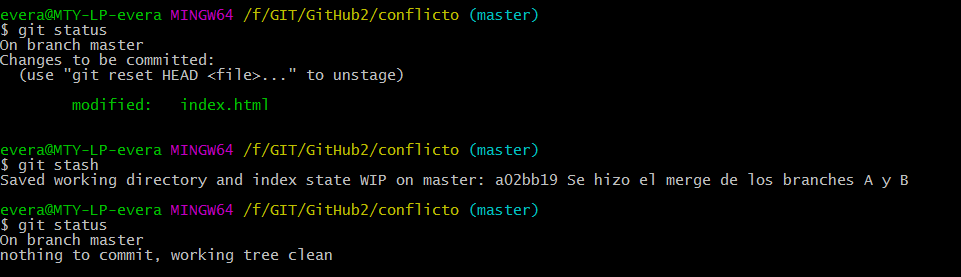
\>git config –global –unset alias.**nombre\_alias\_a\_eliminar**



# Stash (Area temporal).

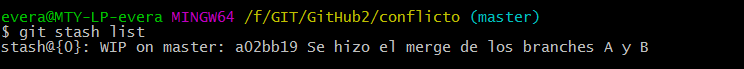
Permite hacer un guardado temporal de los cambios (Esconde los cambios pendientes de commit). Útil para cambiarse de rama sin hacer commit a los cambios actuales.

\>git stash



**Ver el Stash**

\>git stash list



**Restaurar del Stash (No quita el stash de la pila)**

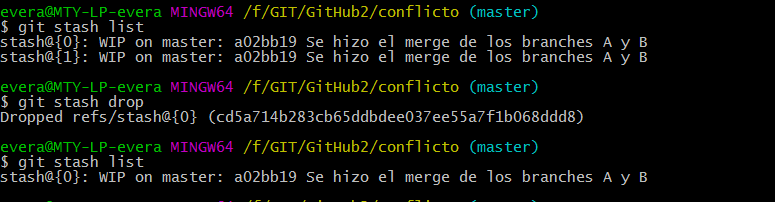
\>git stash apply Restaura todo.

\>git stash apply stash{0} Restaura un stash específico.



**Eliminar el último Stash (de la pila)**

\>git stash drop



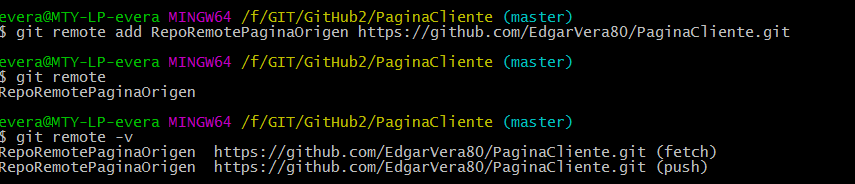
**Restaurar y Eliminar el último Stash (de la pila)**

\>git stash pop

# Remotos

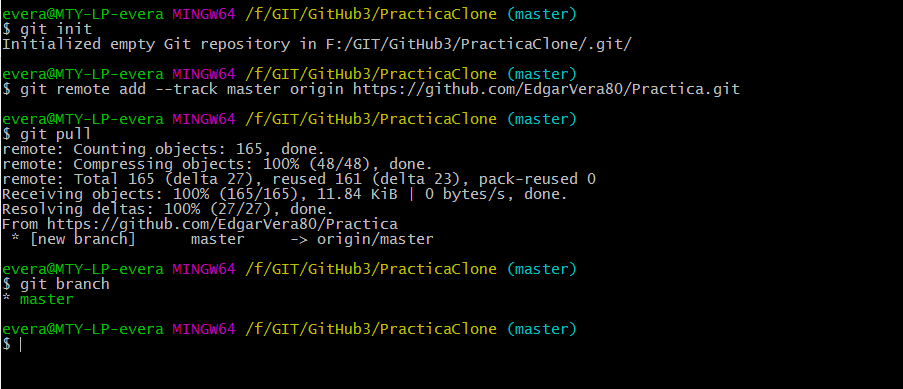
Ligar con un Repositorio Remoto

\>git remote add RepoRemotePaginaOrigen <https://github.com/EdgarVera80/PaginaCliente.git>



Ligando el master local con el master remoto.

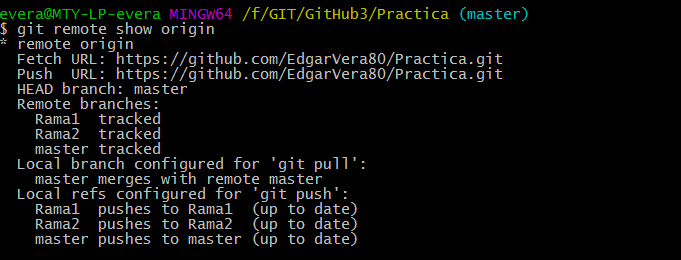
\>git remote --track master origin https://github.com/EdgarVera80/Practica.git



# Show

Muestra información del repositorio remoto.

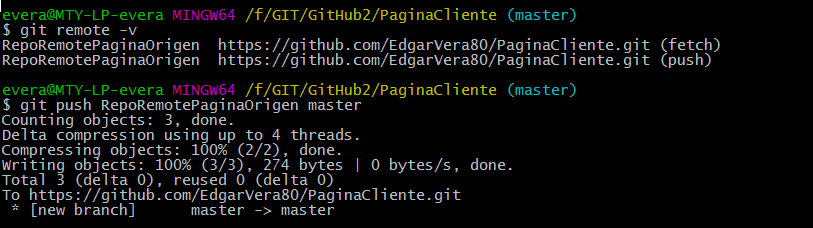
\>git remote show origin



# Push

Envía los datos al repositorio remoto.

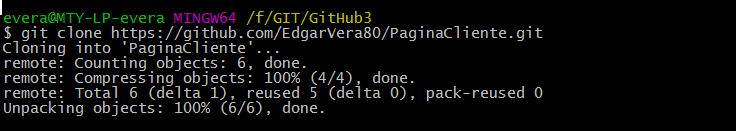
\>git push **Nombre\_Repositorio\_Remoto** **Nombre\_del\_Branch**



# Clone

Clona un repositorio remoto a un directorio local.

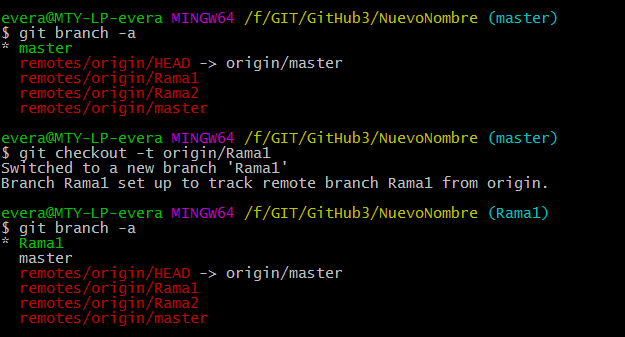
\>git clone **Dirección\_del\_repositorio** <Nuevo\_nombre\_opcional>



# Descarga una rama (Branch)

\>git checkout -t <Branch\_remoto>

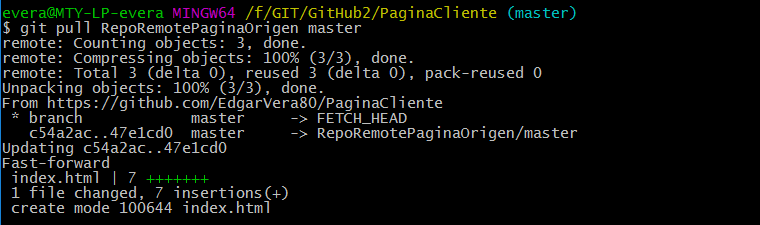
t: Si no existe el branch local lo crea.



# Pull.

Trae los cambios del repositorio remoto y los fusiona con el repositorio local.

\>git pull **Nombre\_Repositorio\_Remoto** **Nombre\_del\_Branch**



Fetch

Trae toda la información de otros branches,

# Quitar Proxy (Branch)

\>git config --global --unset http.proxy