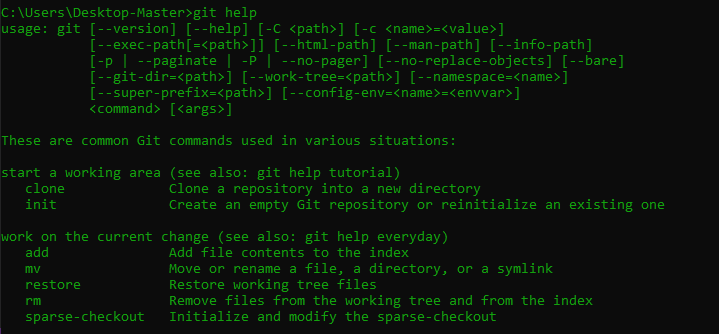
**Comandos Git**

Git es un sistema de control de versiones

**git --version**: muestra la versión de git que tenemos instalado

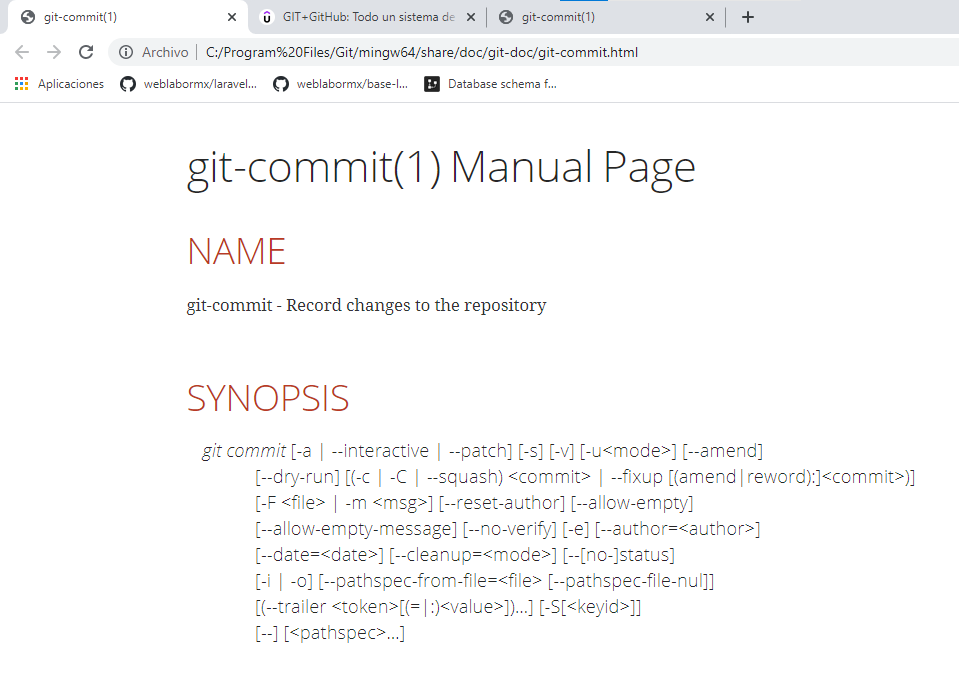


**git help**: nos da una ayuda, de los comandos de git



**git help commit**: en las versiones más recientes de git, abre en el navegador el manual de comando que hemos solicitado la ayuda





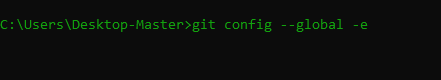
**git config --global user.name “Nombre de usuario”**: es para decirle a git quienes somos

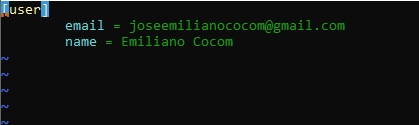


**git config --global user.email “Email de la cuenta de github”**: debemos tener una cuenta de github u otro servicio de la nube, para almacenar repositorios, ese es el correo que debemos ingresar



**git config --global -e**: muestra la configuración del usuario y alias si existe, para salir de la vista se utiliza :qa o para escribir y salir :wq!

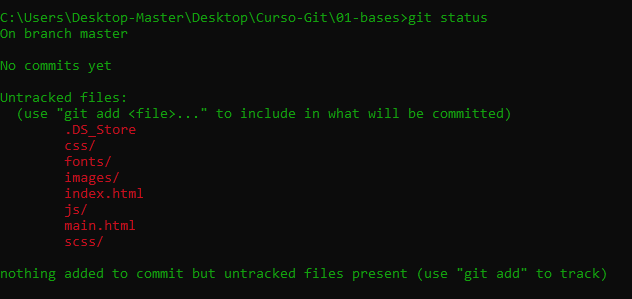




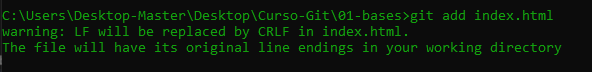
**git init**: inicializa el repositorio



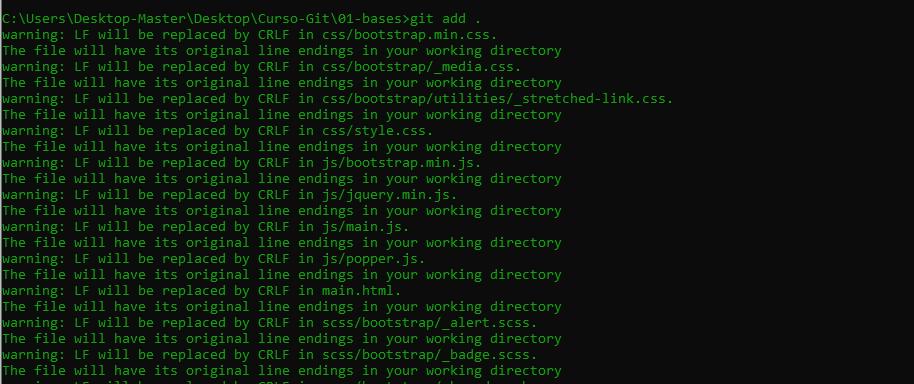
**git status**: muestra el status de los archivos que tienen o no seguimiento y cambios



**git add nombre\_del\_archivo**: agregar un archivo, para ser versionado



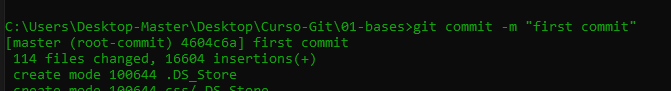
**git add .** : agrega todo los archivos para que sean versionados



**git reset nombre\_del\_archivo** : sirve para quitar del stage, un archivo que no queramos versionar

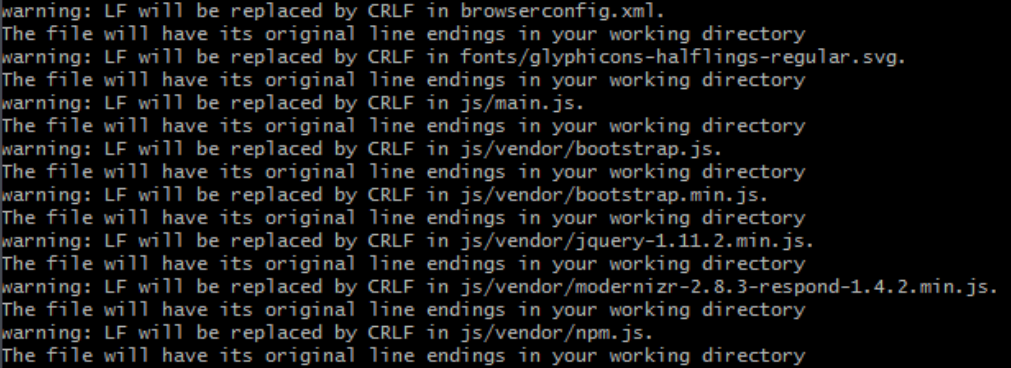


**git commit -m “mensaje para identificar cambios”** : se utiliza para confirmar cambios

****

**git config core.autocrlf true** : Soluciona un problema de CRLF, que es la interpretación de carácter

Fotografía del posible error



**git checkout -- .** : reconstruye el proyecto (archivos y carpetas) hasta el último commit, pero solo los archivos que se le esta dando seguimiento



**git branch** : nos muestra la rama en la cual estamos trabajando



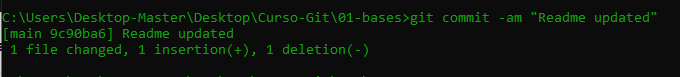
**git branch -m master main** : cambia el nombre de la rama master por main



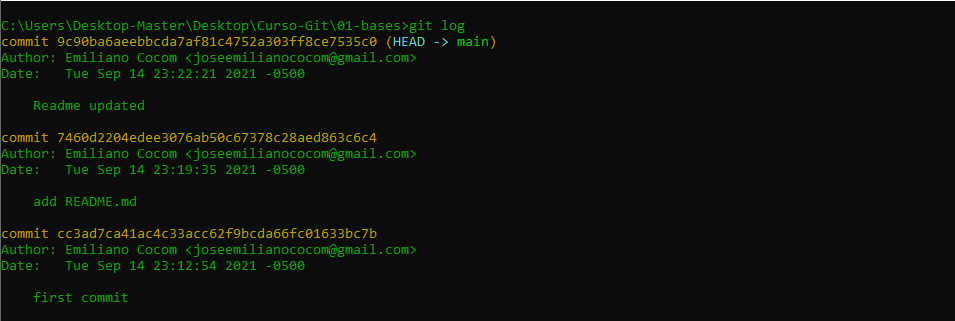
**git config --global init.defaultBranch main** : establece como configuración global que cada vez que se inicialice un repositorio la rama principal sea main



**git commit -am “mensaje del cambio”** : agregar cambios y confirmarlos, en un solo comando a de add y m de message, solo funciona si el archivo ha sido agregado con anterioridad, de lo contrario solo se ejecutar el commit con los cambios en los archivos versionados



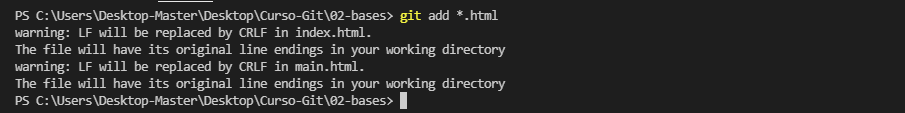
**git log** : muestra todos los log del repositorio, nos muestra todos los commits realizados



**git reset nombre\_del\_archivo\_uno nombre\_del\_archivo\_dos**: una forma de bajar del stage a varios archivos



**git add \*.extensión\_del\_archivo** : agrega al stage o escenario todos los archivos con la extensión indicada

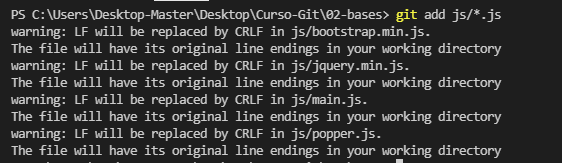


**git reset \*.extension\_del\_archivo** : baja del stage o escenario todos los archivos con la extensión indicada



**Otra forma de agregar varios archivos**

**git add carpeta/\*.extension\_del\_archivo** : agrega todos los archivos que esten dentro de la carpeta con la extensión indicada

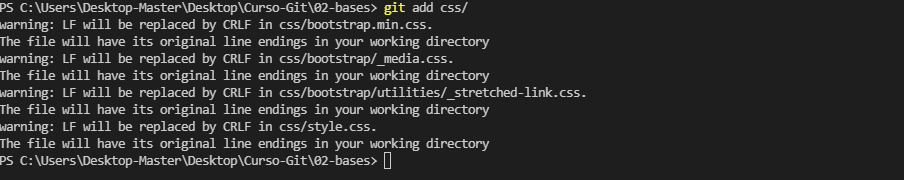
****

**Nota**: si un directorio esta vacío git no lo toma en cuenta, a menos que este contenga algún archivo, en caso de requerir que una carpeta tenga un control de versiones se debe agregar un archivo .gitkeep

**Otra forma de agregar el archivo .gitkeep**

****

**git add nombre\_de\_la\_carpeta/** : agrega todos los archivos que se encuentren dentro de la carpeta



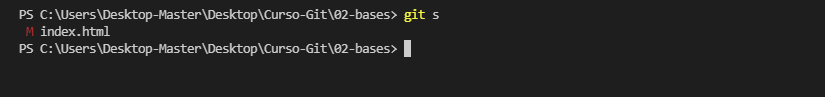
**git status --short** : muestra el estatus de los archivos de manera simplificada



**Alias**

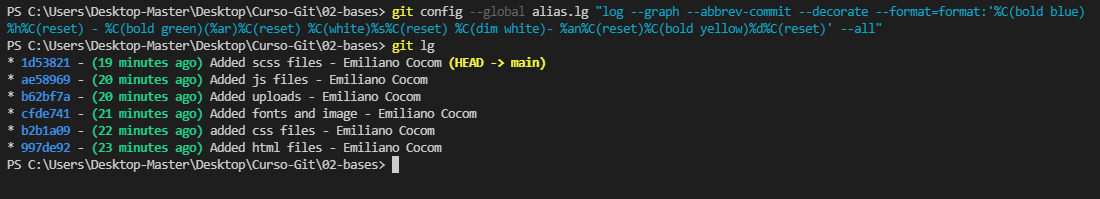
**git config --global alias.s “status –short”** : crea un alias de git status –short a simplemente git s

****

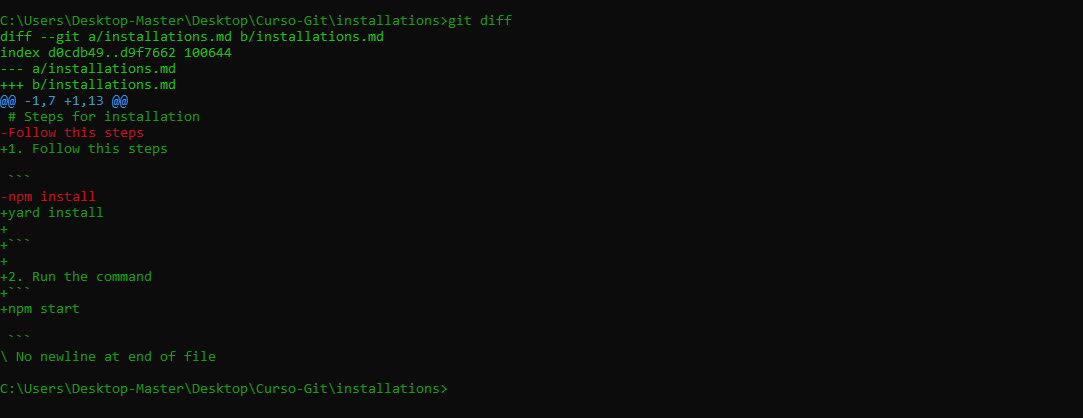
****

**git config --global alias.lg "log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset) %C(white)%s%C(reset) %C(dim white)- %an%C(reset)%C(bold yellow)%d%C(reset)' --all"**

crea un alias que nos muestra la información de la siguiente manera

****

**git diff** : nos muestra los cambios que no se han subido al stage

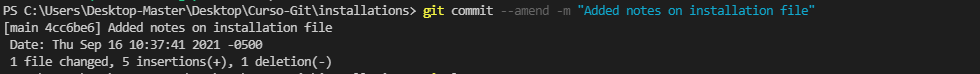


**Nota:** Si ya se han subido los cambios al stage git diff no mostrara estos cambios. Tenemos que ejecutar otro comando git diff –staged

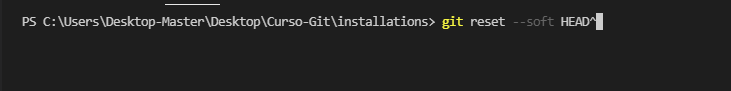
**git diff --staged** : muestra los que se han subido al stage



**git commit --amend -m “mensaje del commit”**: sirve para cambiar el mensaje del último commit

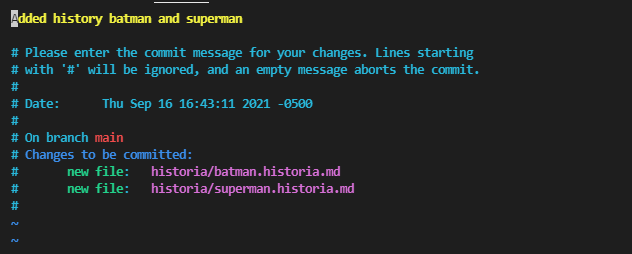


**git reset --soft HEAD^** : Elimina el ultimo commit

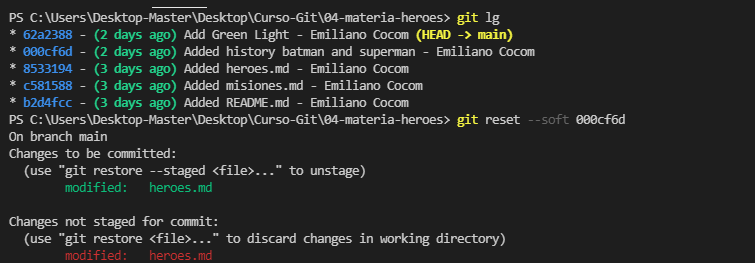


**git commit --amend** : con este comando se puede editar todo la información del último commit

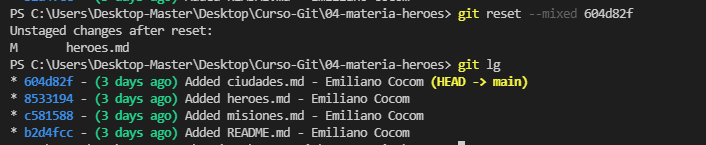




**git reset --soft hash** : Elimina el commit que le pasemos el hash



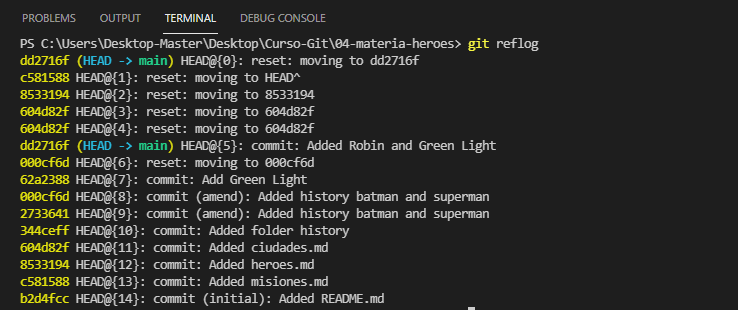
**git reset --mixed hash** : saca todo del stage y los cambios quedan listos para volver a añadirlos.



**git reset --hard hash** : Elimina todos los cambios que hemos realizado hasta el hash del commit



**git reflog** : es la referencia o un log de las referencias de todo lo que ha sucedido en orden cronológico



**git mv destruir-mundo.md salvar-mundo.md** : mueve el archivo y lo renombra



**git rm salvar-mundo.md** : elimina un archivo desde git



**git reset --hard** : este comando es como hacer <git chekout -- .> muy parecidos



**Rama o Branch**

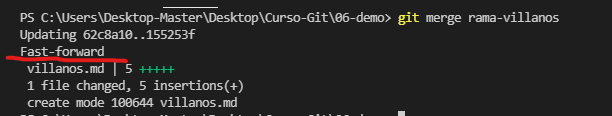
**git branch nombre\_de\_la\_rama** : crea una nueva rama o Branch, de preferencia nombrar la rama con un nombre que identifique en que se va trabajar



**git checkout nombre\_del\_branch** : sirve para cambiar de Branch

****

**git merge branch** : este comando nos sirve para hacer merge con otras ramas, para hacer merge, nos tenemos que posicionar en la rama en la cual queremos agregar los cambios, en este caso unimos los cambios de la rama-villanos con master, el mensaje Fast-forward, significa que git logró identificar cada uno de los cambios y no existe ningún conflicto

****

**git branch -d nombre\_del\_branch** : Elimina el branch

****

**git branch -d rama-villanos -f** : eliminar un branch de manera forzada

****

**git checkout -b nuevo-branch** : nos crea y mueve al nuevo branch

****

**git merge rama -m “commit message”** : agregar mensaje personalizado al hacer merge

**git tag nombre\_del\_tag** : Un tag es un punto especifico en un commit,



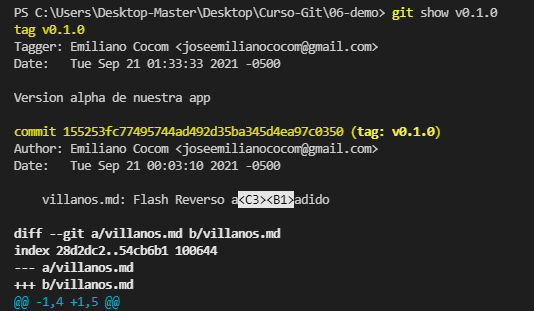
**git tag -d nombre-del\_tag** : elimina el tag



**Crear un tag de forma semántica (recomendado)**



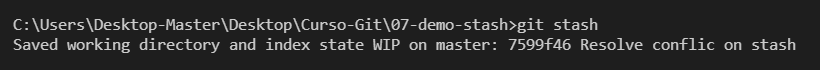
**git show nombre\_tag** : Muestra de manera detallada la información del tag



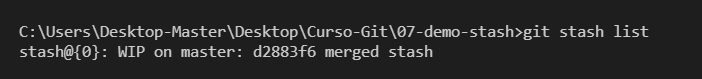
**Git stash**

El stash es básicamente un lugar donde podemos guardar información de manera temporal y luego recuperarlo.

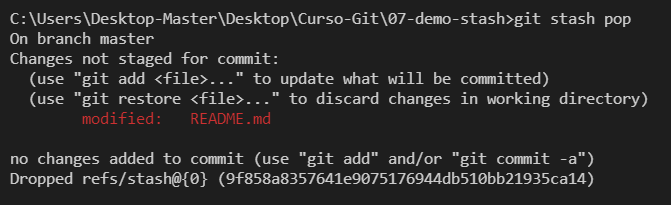
**git stash** : guarda los cambios que tengamos actualmente, sin hacer commit, estos cambios se guardan en el stash para posteriormente recuperarlos.



**git stash list** : muestra el listado de stash que tengamos



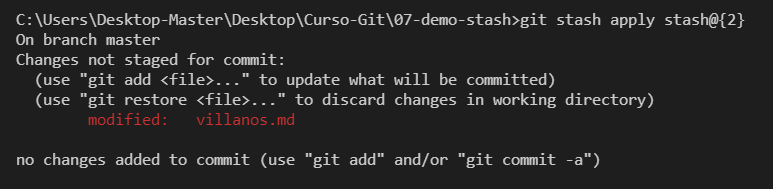
**git stash pop** : recupera el último stash agregado



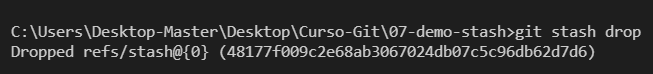
**git stash clear** : Elimina los stash que tengamos almacenados



**git stash apply nombre\_stash** : Se utiliza para aplicar un stash, identificándolo por el nombre



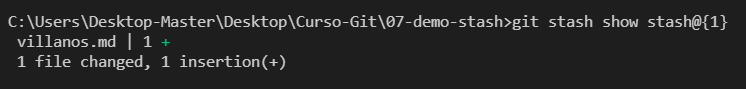
**git stash drop** : elimina el primer stash



**git stash drop nombre\_stash** : elimina el stash con el nombre que le indicamos



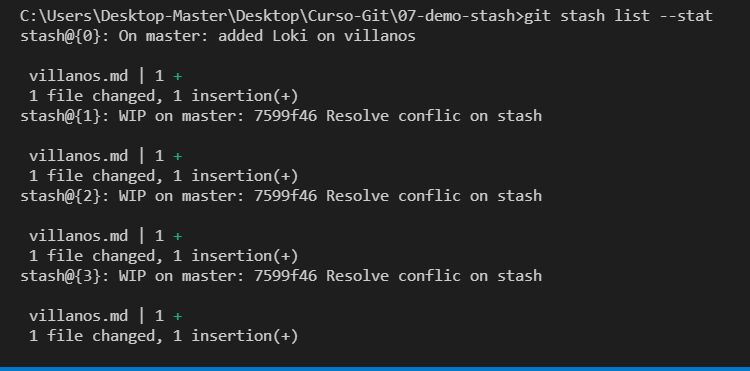
**git stash show nombre\_stash** : muestra los detalles del stash



**git stash save “message”** : agrega el stash con mensaje personalizado



**git stash list --stat** : Muestra el detalle de todos los stash

****

**Git rebase**

El rebase nos sirve para:

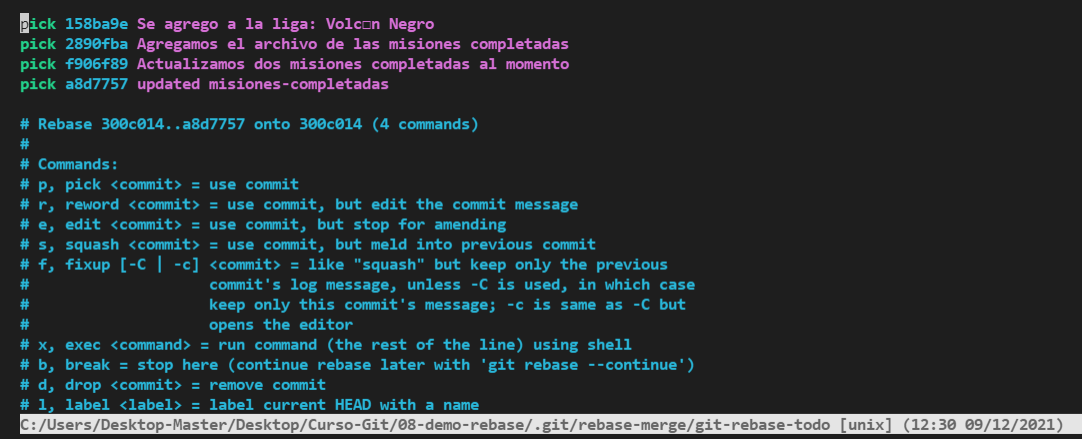
* Ordenar commits
* Corregir mensajes de los commits
* Unir commits
* Separar commits

**git rebase nombre\_branch** : integra los commits que se tengan antes del punto de separación.



**git rebase -i HEAD~4** : Nos retorna una vista de edición de los commits que le indiquemos





squash : se utiliza para fusionar dos commits y convertirlos en uno solo

reword: se utiliza para renombrar los commits

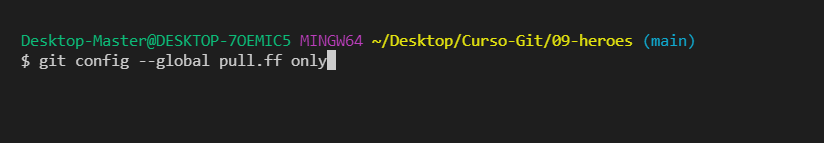
**git rebase --continue** : terminar el proceso del rebase interactivo



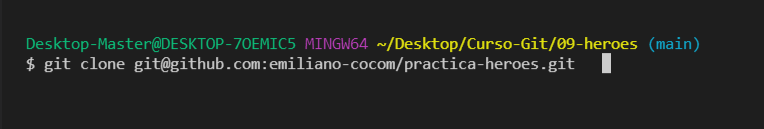
**GitHub**

**git push --tag:** Sube a repositorio remoto todos los tags que tengamos en local

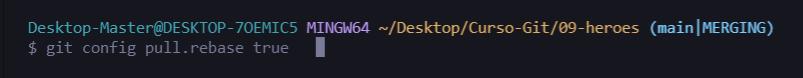
**git config --global pull.ff only:** Este comando configura git para que solo se fusionen los cambios cuando sea Fast-forward



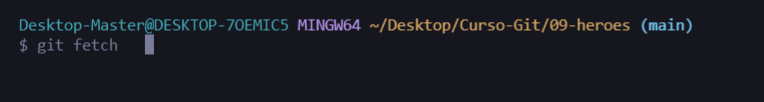
**git clone remote-repository:** clona el repositorio a nuestra computadora de manera local



**git config pull.rebase true:** declara la configuración por defecto del repositorio a utilizar el método rebase, si queremos que se apliquen en todos los repositorios debemos utilizar el “**git config --global pull.rebase true**”



**git fetch:** Nos sirve para descargar los cambios, pero sin aplicarlo, para actualizar las referencias



**git pull:** Descarga los cambios y realiza un merge con la rama local

