**Comandos útiles de la terminal**

**cd** Nos permite movernos dentro de carpetas. Con **cd …** podemos salir de las carpetas.



**mkdir** Para crear carpetas y poner el nombre de la carpeta que vamos a crear.



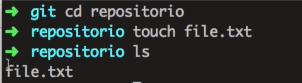
**ls** Para ver las carpetas que tenemos creadas.



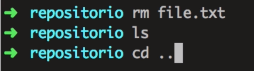
**clear** Para limpiar la terminal. Este nos sirve para hacer scroll e ir al principio de la terminal.



**touch** Nos permite crear archivos.



**rm** Para borrar archivos escribiendo el nombre del archivo después del comando.



**rm -rf** Para borrar carpetas escribiendo el nombre de la carpeta después de los comandos.



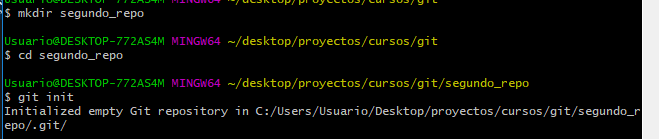
# git-init (Creando repositorios)

# Crear un repositorio

* **git init** nos crea un repositorio de manera local y lo hará en la carpeta donde estamos posicionados o se le puede pasar [nombre\_de\_la\_carpeta] y creará la carpeta con ese nombre.



# CREAR REPOSITORIO DESDE OTRA CARPETA



# ELIMINAR REPOSITORIO

LA CARPETA .GIT ES NUESTRO REPOSITORIO

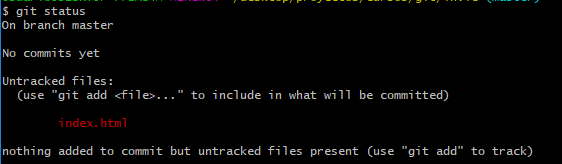
**$ rm -rf .git** para borrar la carpeta oculta y asi borrar el repositorio

# git add | rm | status (Agregando, quitando y viendo el estatus de archivos.

# COMO CREAR ARCHIVOS Y ANADIRLOS NUESTRO DIRECTORIO DE TRABAJO

**Untracked files**: Son archivos que están en nuestro Working Directory,

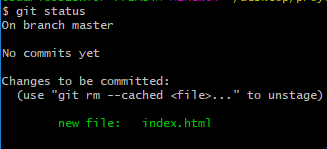
Al colocar el comando **git status** lo que aparezca en rojo es lo que se ha modificado y hay que pasarlo al Staging.



**PASAR EL ARCHIVO AL STAGING**



**Ver el estado**



# QUITAR EL ARCHIVO DE EL STAGING PERO NO LOS HA BORRADO



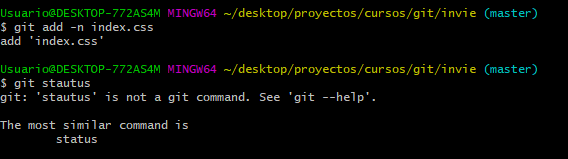
# AGREGAR TODOS LOS ARCHIVOS DEL WORKING DIRECTORY A EL REPOSITORIO .GIT



# QUITAR LOS ARCHIVOS DE STAGING Y BORRARLOS DEL WORKING DIRECTORY (BORRADO FORZOSO)



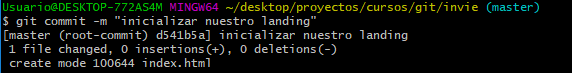
# POSIBLEMENTE AGREGA UN ARCHIVO SIR VE PARA MIRAR SI EXISTE UN ARCHIVO DENTRO DEL DIRECTORIO



**git tag (Etiquetando confirmaciones)**

# git commit (Confirmando cambios)

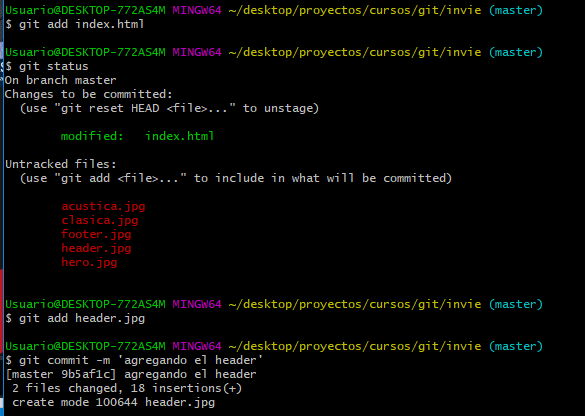
**git commit -m "[description...]" :** Agrega los archivos del **Staging Area** al **Git Repository** con una descripción.



**git commit --amend** concatena nuevos cambios a un commit previo

**SALIR DE LA CONSOLA :wq**

# EJEMPLO DE TODOS LOS COMANDOS



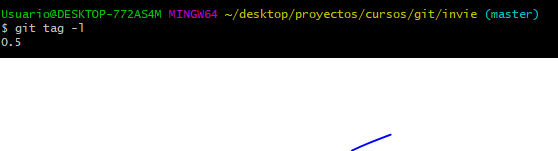
# git tag (Etiquetando confirmaciones)

**git tag**: nos permite agregar etiquetas a nuestros cambios.  
**-a** para la anotación  
**-m** para el mensaje

(Hemos etiquetado el ultimo commit)



**-l** nos muestra la lista de etiquetas

  
**-f** para renombrar  
**-d** para borrar

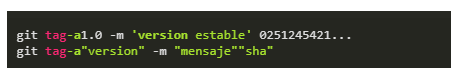
**\*Tipos de tag**

# Colocar tag a una versión anterior de mis commits

**–tag ligera**



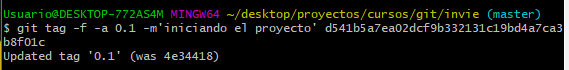
**\*\*tag anotada**



# Borrar un tag

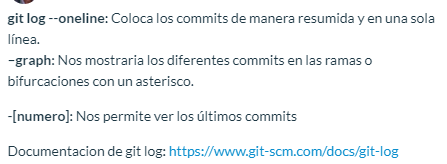


# Actualizar un tag



# git log (Revisando la historia de nuestro proyecto)

Éste se invoca con git log, los logs te muestran los códigos hexadecimáles que crea Sha-1, el autor y la fecha en la que se envió el ese commit.  
Git log se puede personalizar, y de hecho es un comando que se puede personalizar mucho. Para esto podemos usar -oneline que nos ayuda a resumir mucho esos commits.  
Además, podemos ponerle múltiples parámetros, así como consultar solo los logs que nos interesan ver.  
La documentación de git log es súper extensa, y puedes usarla para revisar la historia de nuestro proyecto.

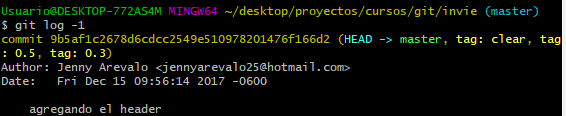












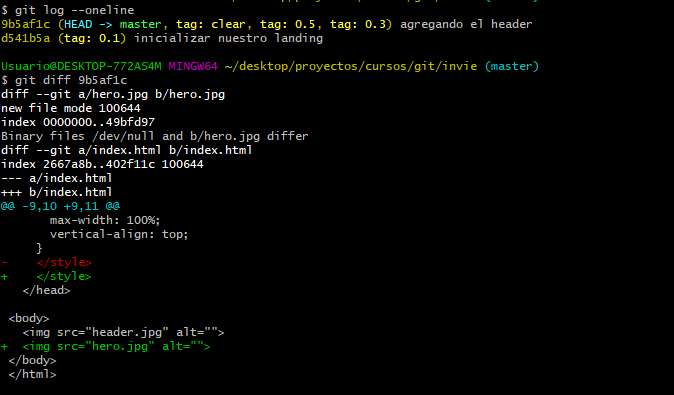


# git diff (Revisando los cambios entre versiones)

# Mostrar la diferencia entre la version actual y el sha ingresado

**-git diff [SHA1]**: Nos muestra las cambios de ese commit.

**rojo**: fueron cambios que se quitaron  
**verde**: se agregaron cosas.

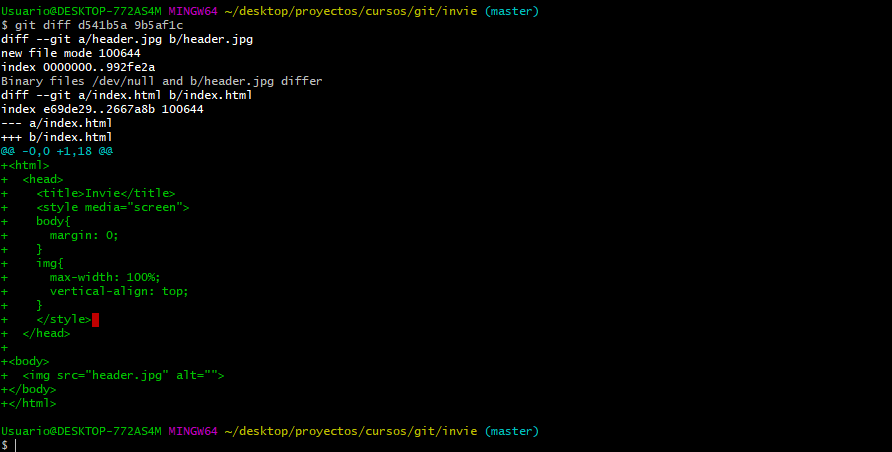


# Mostrar los cambios entre un commit y otro

- **git diff** [sha-1] [sha-1] muestra la diferencia entre los dos sha ingresados

ojo agregar siempre el sha masviejo primero en el comando para ver las diferencia correctamente

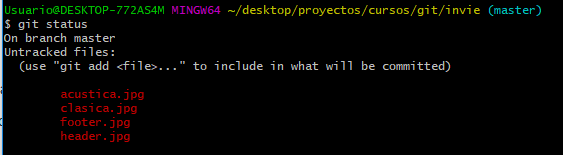




# git reset -- soft

**git reset --soft [SHA1]:** Nos permite quitar los cambios de un commit específico. Deja los archivos en el staging area, listos para hacer un commit.

**NOTA!:** Si quiero modificar el ultimo commit tomar el sha del anterior.



# git reset -- mixed

**git reset --mixed [SHA1]**: nos elimina los cambios, también del staging area.

# git reset – hard

**git reset --hard [SHA1]**: Nos elimina los cambios incluso del working directory, es el más peligroso de todos porque podemos perder parte de nuestro trabajo.



**NOTA**



**Para eliminar todo copiar el utlimo SHA Y SI QUIERO RECUPERARLO UTILIZAR EL PRIMER SHA**

**git branch (Múltiples variantes del repositorio)**

# Desafío: Configurar otro editor de texto

**git config --global core.editor “nano”**

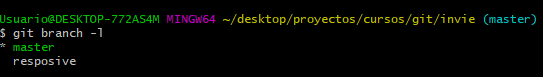


# git branch (Múltiples variantes del repositorio)





**git branch -l**Lista todos las ramas que existen.









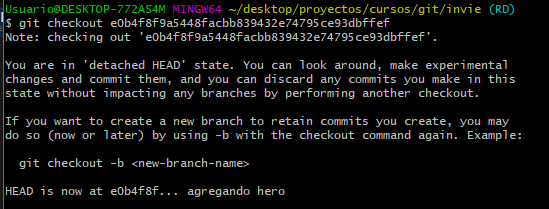


# git checkout (Moviéndonos entre ramas y versiones)

**git checkout [nombre]**: Nos permite mover entre ramas no vamos a borrar nada. Acá es donde podemos movernos en el tiempo.



**git checkout [sha1]**: Nos permite mover entre commits, no vamos a borrar nada. Acá es donde podemos movernos en el tiempo.



**git checkout -b [nombre\_rama]:** Nos permite crear una nueva rama y navegar en ella sin necesidad de usar branch

NOTA:

Rama para corregir errores : hotfix

Rama para desarrollo: develop

# git merge (Mezclando ramas y resolviendo conflictos)

**git merge [rama]**: Nos permite mezclar los cambios realizados en dicha rama con la rama en la que estamos.



# git rebase (Reescribe la historia de tu proyecto)

**git rebase**: hace prácticamente lo mismo que *merge*, cambiamos la historia de nuestro proyecto sin crear bifurcaciones del proyecto. Es mejor usar **merge**  
Usar solo git rebase de manera local.

-i: de manera interactiva, nos abrira el editor que tengamos definido en la configuración de git.

# git stash (Guardando cambios temporalmente)

¿Qué tal si aún no estás listo para confirmar ningún cambio? **Stash** es un estado que tenemos como intermedio. Para esto debemos ir a alguna de nuestras ramas y usando el comando **git stash** que nos permite hacer cambios, pero no confirmarlos.