## **Tomado de** [**https://www.diegocmartin.com/tutorial-git/**](https://www.diegocmartin.com/tutorial-git/)

## **Conceptos básicos de Git**

* Git se basa en snapshots (instantáneas) del código en un estado determinado, que viene dado por el autor y la fecha.
* Un Commit es un conjunto de cambios guardados en el repositorio de Git y tiene un identificador SHA1 único.
* Las ramas (branches) se pueden pensar como una línea de tiempo a partir de los commit. Hay siempre como mínimo una rama principal o predefinida llamada master o main.
* Head es el puntero al último commit en la rama activa.
* Remote se refiere a sitios que hospedan repositorios remotos como GitHub.

## **Workflow de Git**

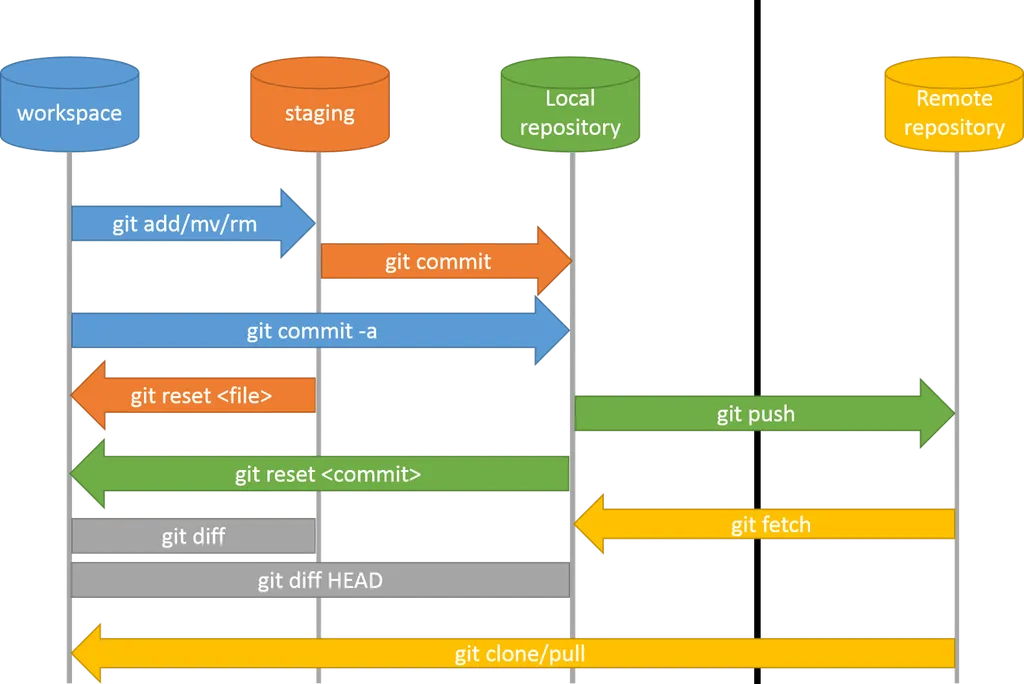


Imagen obtenida de blog.podrezo.com

Si trabajamos en local (comenzamos en la imagen por la izquierda), inicializamos el directorio de trabajo (working directory). Podemos trabajar (editar ficheros) en el directorio de trabajo.

Con el comando Git add enviamos los cambios a staging, que es un estado intermedio en el que se van almacenando los archivos a enviar en el commit.

Finalmente con commit lo enviamos al repositorio local.

Si queremos colaborar con otros, con push subimos los archivos a un repo remoto y mediante pull podríamos traer los cambios realizados por otros en remoto hacia nuestro directorio de trabajo.

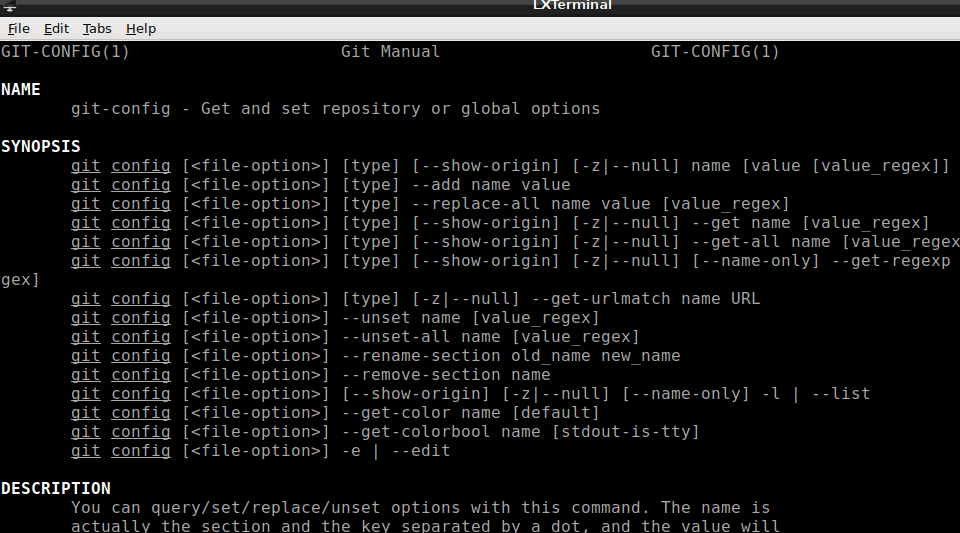
Si comenzamos trabajando en remoto, lo primero que hacemos es un clon de la información en el directorio local.

## **Ayuda de Git**

Con git help en el terminal obtenemos ayuda.

Vamos a mirar la ayuda de configuración con

#git help config

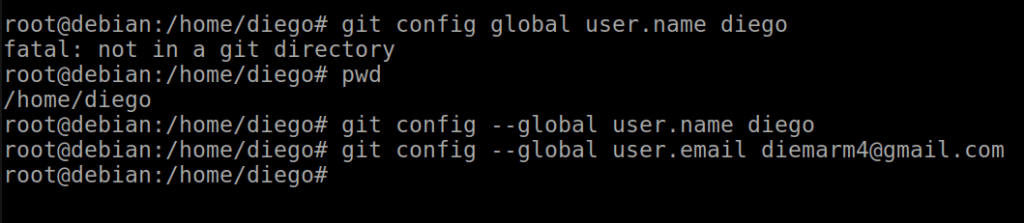


## **Configuración de Git**

Como mínimo debemos configurar el nombre y el email en la aplicación con los siguientes comandos:

git config --global user.name "Tu nombre aquí"

git config --global user.email "tu\_email\_aquí@example.com"



Para comprobar podemos usar:

git config -–global –list

Lo que hace el sistema es crear un archivo de texto llamado .gitconfig, por lo que podemos mostrarlo también con cat:

cat ~/.gitconfig

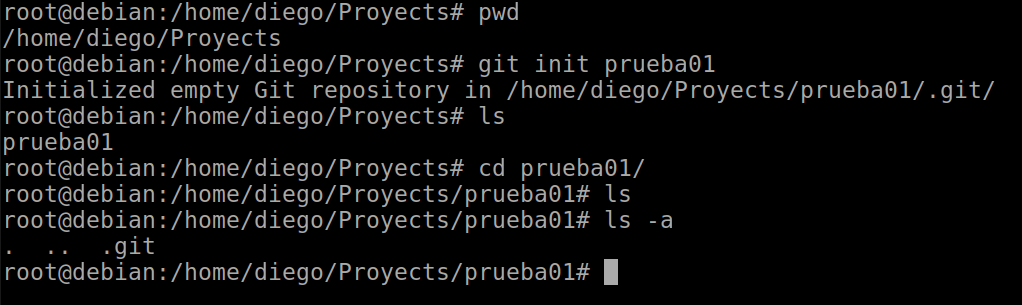
## **Comenzar a trabajar con Git**

### Trabajar en un nuevo proyecto con Git

Nos situamos en la carpeta en la que queremos trabajar. Nos aseguramos con pwd, para saber dónde estamos.

Ahora con git init y el nombre del proyecto, creamos un nuevo proyecto:

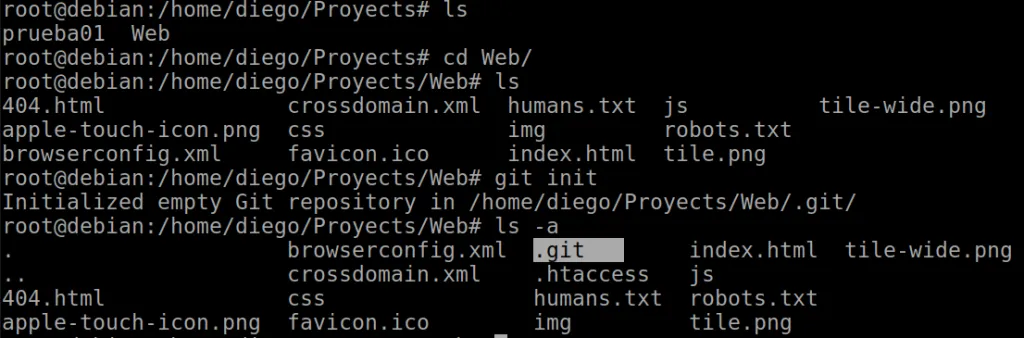
git init prueba01



Esto crea una carpeta con el mismo nombre y en su interior, podemos observar con ls -a que se crea una subcarpeta oculta .git para el control y gestión de la herramienta.

### Trabajar en un proyecto existente con Git

Creamos una nueva carpeta y un archivo dentro de ella, nos situamos en ella (en la carpeta) y ejecutamos simplemente git init.

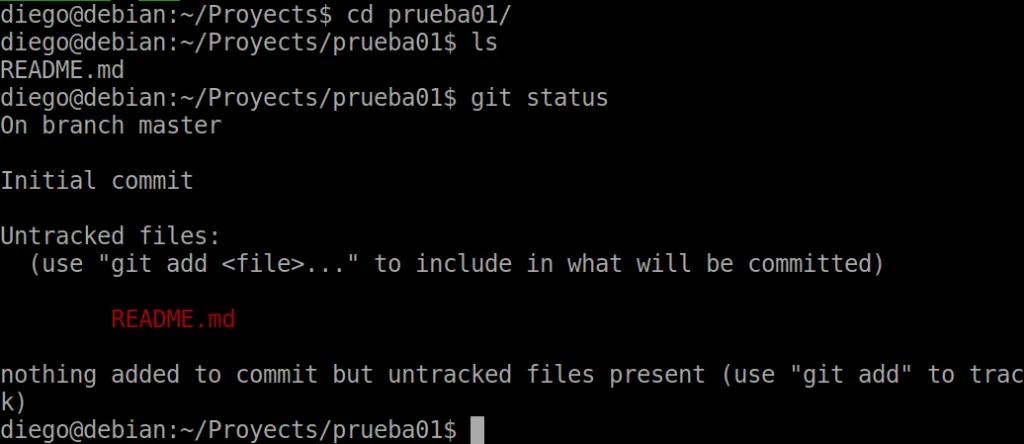


### Primer commit

Ahora vamos a crear algún archivo en el primero de los proyectos. Yo he creado un archivo README.md con el contenido: #Demo de Git.

Ahora, en la carpeta del proyecto ejecutamos

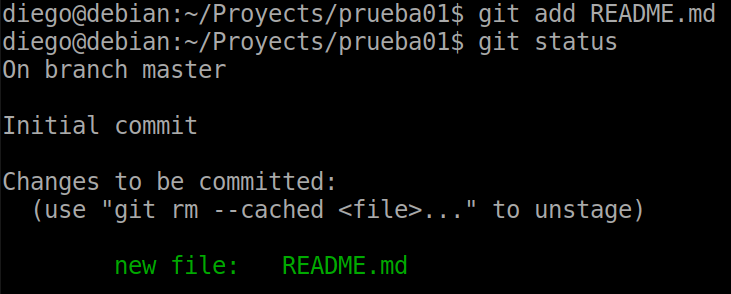
git status



Podemos ver los detalles. Estamos ubicados en la rama máster o main y que tenemos un archivo que no ha sido agregado aún.

Lo primero que vamos a hacer es agregarlo al staging, también llamado Git Index, mediante el comando git add:

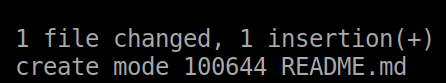
git add README.md



Ahora vemos que el archivo ya está esperando a ser enviado al repo mediante commit.

Al ejecutar commit no especificamos qué archivos se enviarán. Se envían todos los que estén en staging. Vamos a agregar la opción -m para agregar un mensaje al commit, por ejemplo “commit inicial”:

git commit -m “commit inicial”



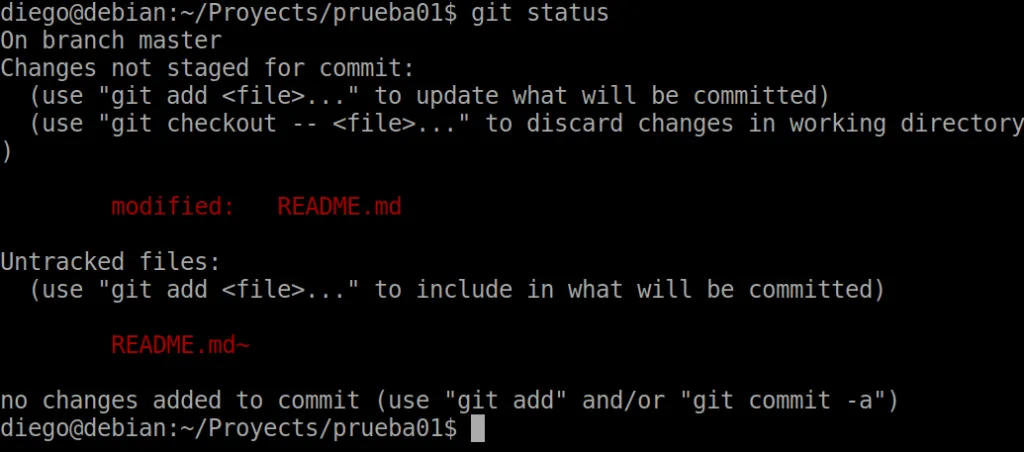
Veremos un mensaje similar a este:

Si hacemos un status ahora veremos que el área de staging está vacía y no hay ningún commit pendiente.

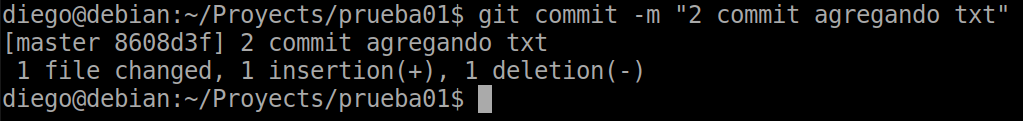
## **Trabajando con Git en local**

Ahora vamos a agregar más contenido al archivo README y hacemos un status.

Veremos que ahora el sistema nos indica que hay cambios no enviados al staging:



Con el mismo procedimiento de antes, agregamos (add) y hacemos un nuevo commit:



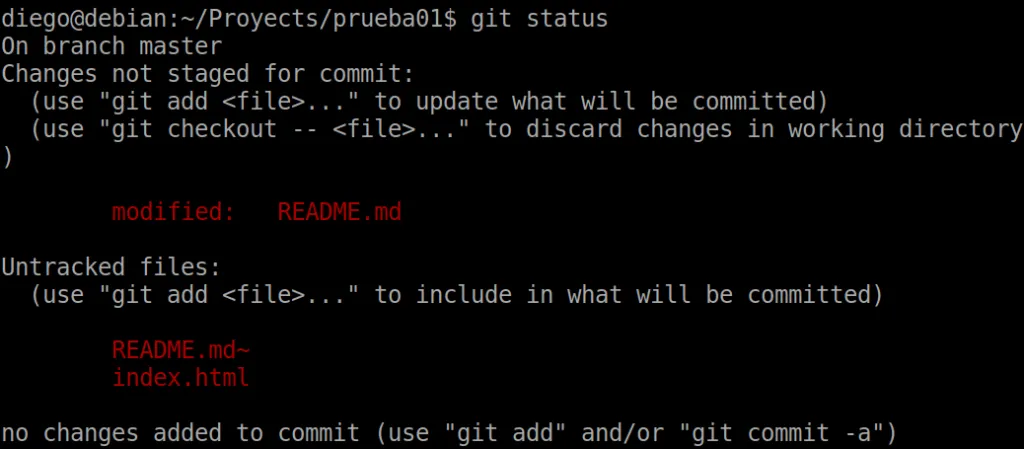
También podemos hacer el add y el commit en un solo paso con:

git commit -a

Tal como nos recomienda el propio Git en la captura de arriba. Si queremos también agregar el mensaje quedaría

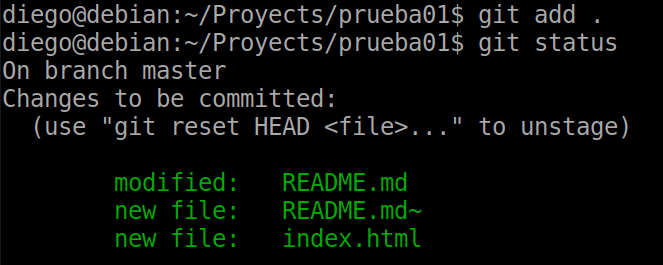
git commit -am “msj”

Podemos hacer algunos cambios más y agregar otro archivo, por ejemplo, un index.html al proyecto.

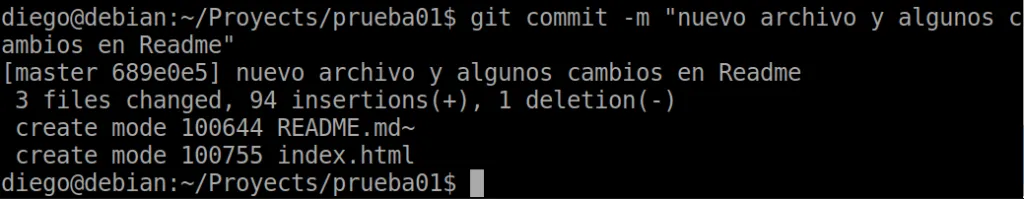


En este caso vamos a agregar los archivos modificados o nuevos al staging de forma recursiva con

git add .

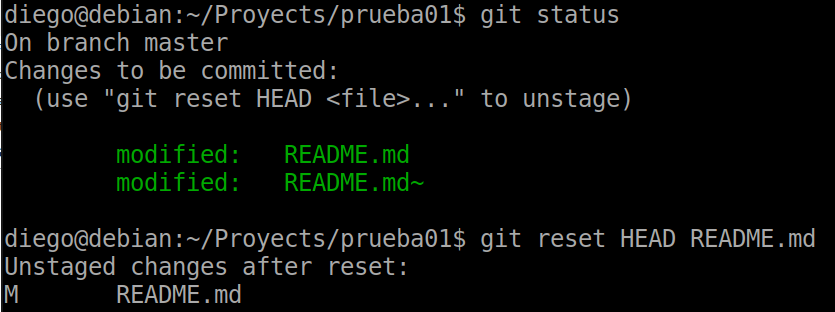


Un nuevo commit:

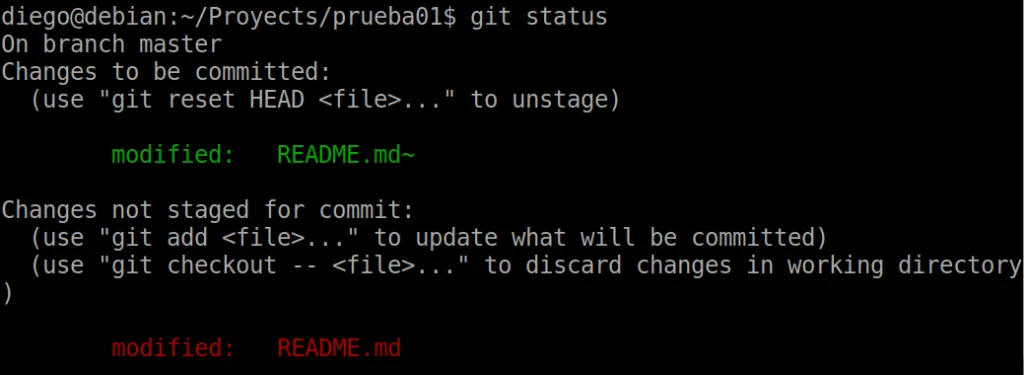


Vamos a volver a hacer algún cambio y lo agregamos al staging. Si nos fijamos en la captura de la anterior operación add, el sistema nos indica que podemos sacar un archivo del stage con git reset HEAD y el nombre del archivo. Lo probamos con README:

git reset HEAD README.md



Un nuevo status nos indicará que hay uno en el staging y otro por agregar:



Como podemos ver, también podemos descartar los cambios en el directorio de trabajo con checkout:

git checkout -- README.md

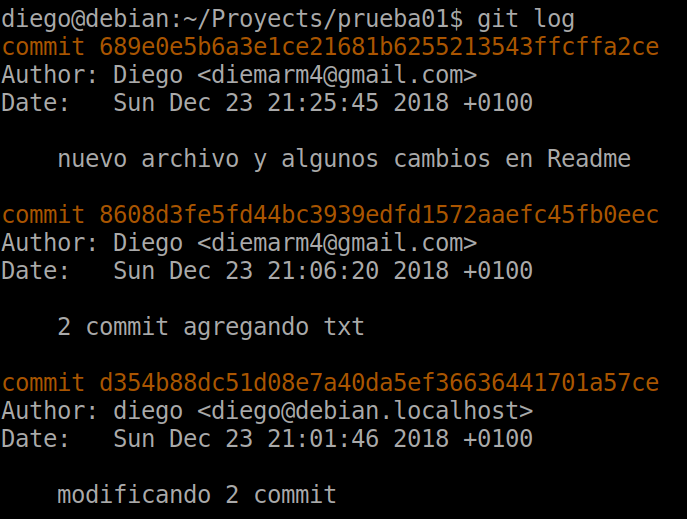
Esto devuelve nuestro archivo readme al estado anterior.

Recuerda que puedes consultar la ayuda para ver más opciones y comandos.

### Histórico de operaciones en Git. El log

Con git log podemos ver el histórico de operaciones que hemos hecho:

git log

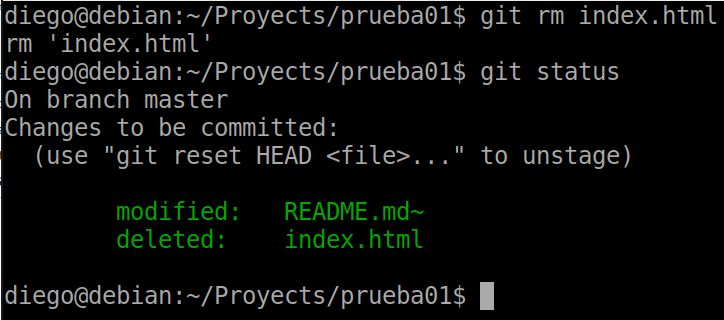


Podemos ver que los commit se han hecho en orden cronológico inverso con los distintos usuarios y que cada uno tiene asociado su clave y su fecha.

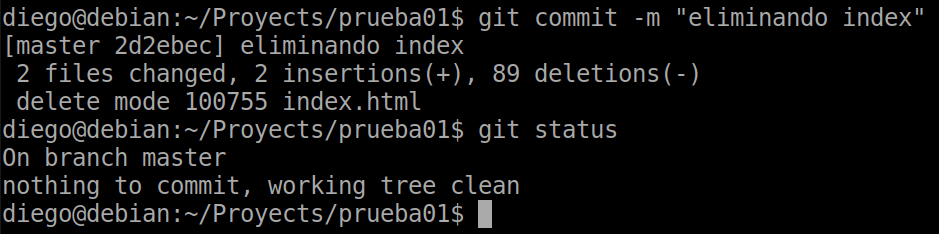
### Eliminar archivos del repositorio de Git

Imaginemos que uno de los archivos que ya están en el repositorio de git tras un commit no lo queremos allí. Podemos usar el comando rm de remove y el nombre del archivo:

git rm index.html

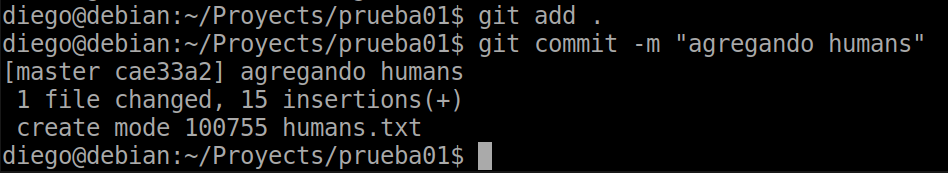


Podemos observar que el status nos indica que el cambio ejecutado para el archivo a eliminar está en el staging. Para eliminarlo debemos hacer un nuevo commit:

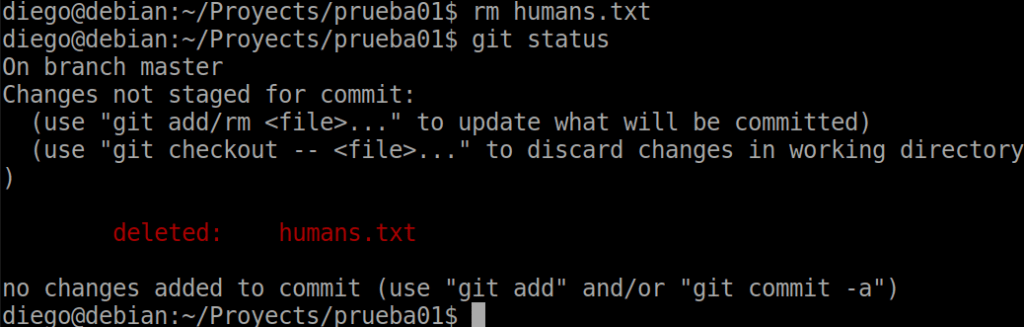


Ahora veremos que el archivo index ya no está en el directorio de trabajo.

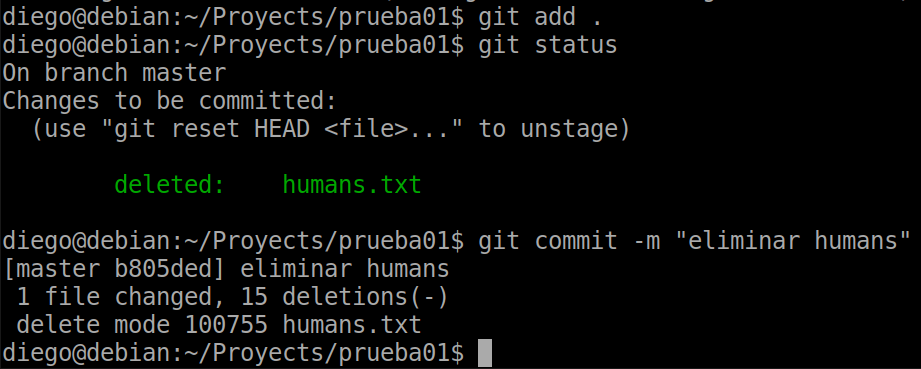
Ahora vamos a agregar un nuevo archivo, lo agregamos al staging y hacemos un commit. Vamos a eliminarlo manualmente desde el sistema de archivos del SO, no desde Git.



Ahora elimino el archivo humans.txt y hacemos un status:



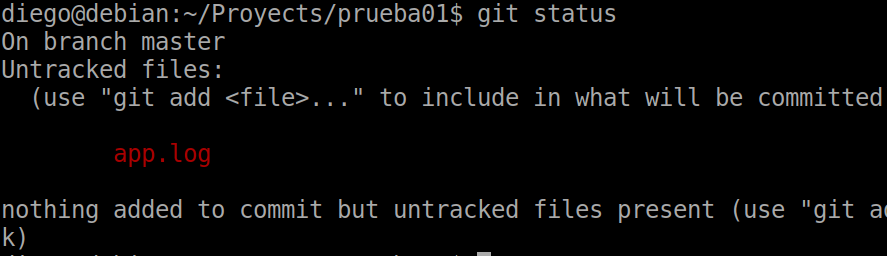
Como la misma pantalla indica, podemos usar add o rm para que definitivamente los cambios se agregan al staging.



En este caso se ha usado un add y tras un commit se ha ejecutado el borrado en el repo.

### Ignorar archivos en Git

Ahora disponemos de un archivo que no queremos que se agregue nunca al repo, por ejemplo un archivo de log.

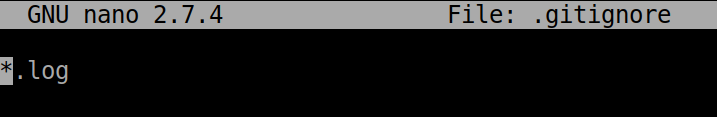


En Linux Debian, con el comando nano podemos editar un archivo en un editor de textos en el terminal.

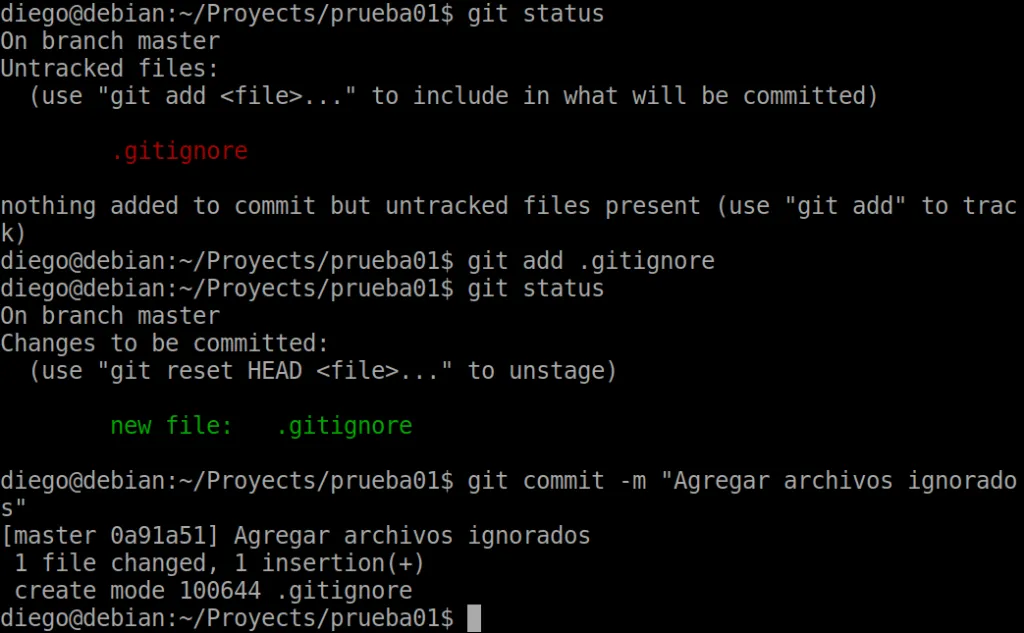
Vamos a editar el archivo .gitignore, en el que especificamos los archivos a ignorar en líneas independientes.

nano .gitignore

Y en el archivo, que aparecerá vacío, agregamos en la primera línea \*.log. Guardamos y salimos.



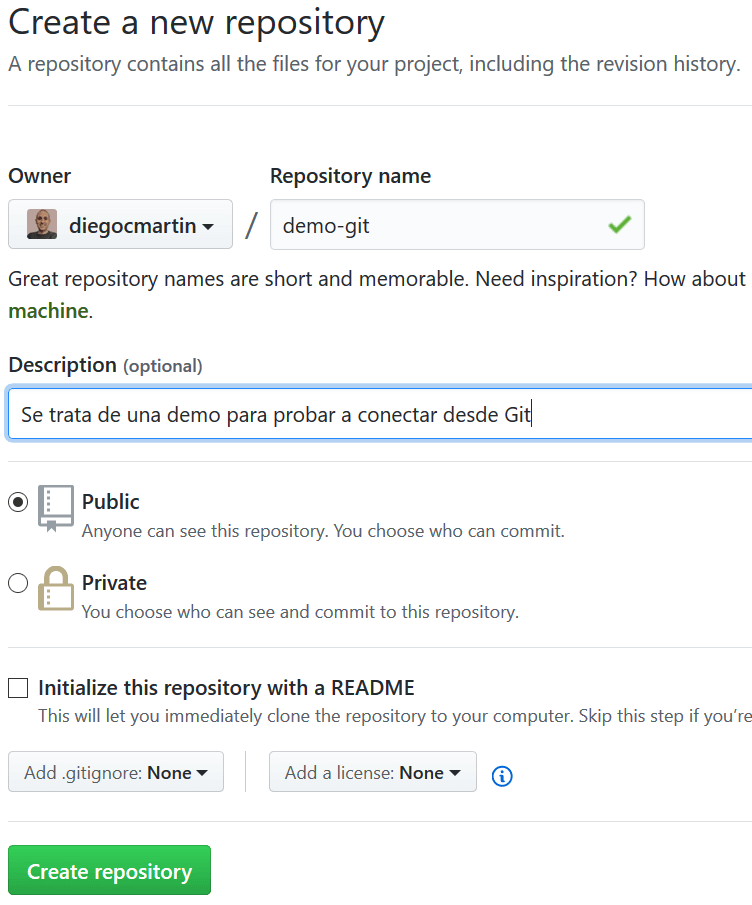
Ahora hacemos un status y veremos el archivo .gitignore para agregar a staging:



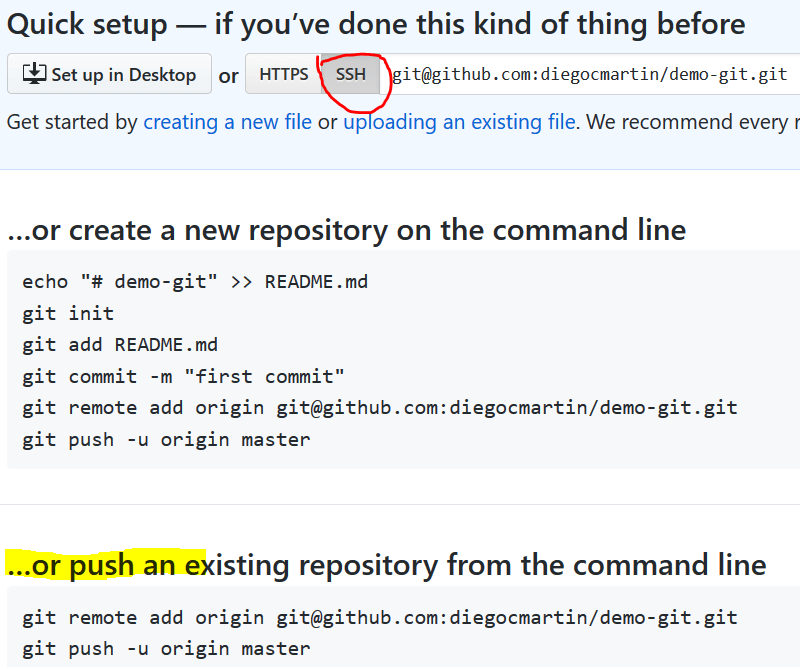
También hemos hecho un nuevo commit.

### Trabajando con un repositorio remoto en GitHub

Primero vamos a crear un nuevo repositorio desde la aplicación web de GitHub con la sesión iniciada.



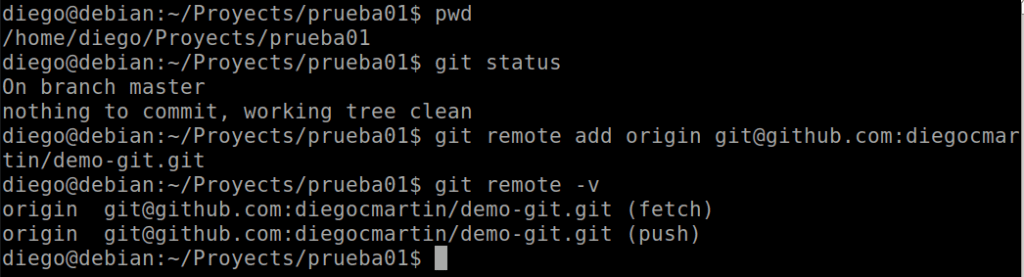
Una vez creado, en la siguiente pantalla hacemos clic en SSH en la parte superior para ver las instrucciones de envío mediante línea de comandos.



Ahora nos movemos al proyecto con el que queremos trabajar. Vemos que el status no hay nada pendiente y ejecutamos el primer comando. Usando después…

git remote -v

El comando remote sirve para ver los repositorios remotos asociados. con la opción -v vemos la [URL](https://www.diegocmartin.com/glosario/url/):



Las URLs que vemos en la captura anterior son de tipo SSH, indicativo de que podríamos enviar. También podemos agregar repositorios de otros usuarios para recibir con el comando git remote add [nombre] [url].

Ejemplo: $ git remote add pb git://github.com/paulboone/ticgit.git

Para recibir los ficheros de uno de estos repositorios de otro usuario usamos el comando fetch seguido del nombre que hemos puesto en el comando anterior.

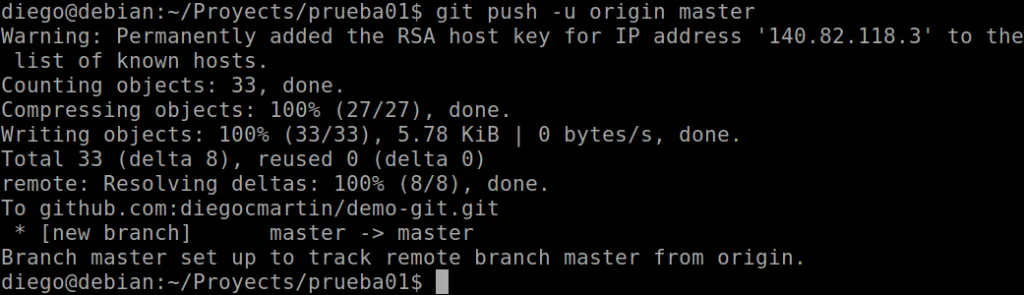
Ejemplo: git fetch pb

Ahora vamos a enviar los archivos en local al repo remoto con push y la opción -u para establecer un enlace entre ellos, especificando también el repositorio remoto (origin) y la rama de trabajo (master o main), quedando:

git push -u origin master

// Si no te funciona prueba con

//git push -u origin main

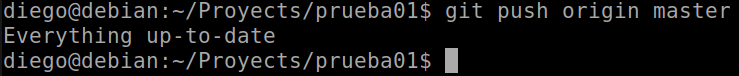


Una vez establecido el enlace no hará falta usar la opción -u, quedando la instrucción para actualizar los cambios en el repositorio remoto de la siguiente manera:

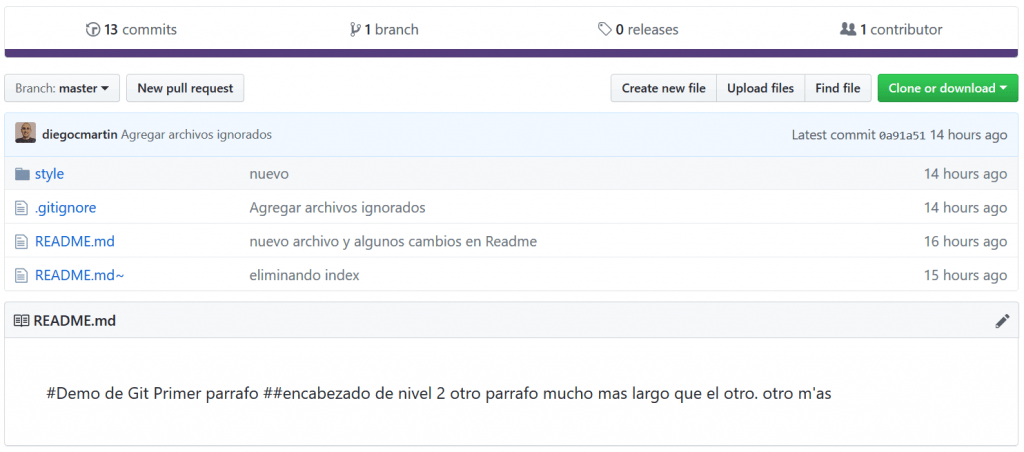
git push origin master

// Si no te funciona prueba con

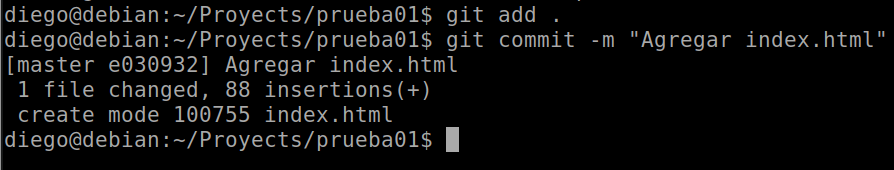
//git push -u origin main



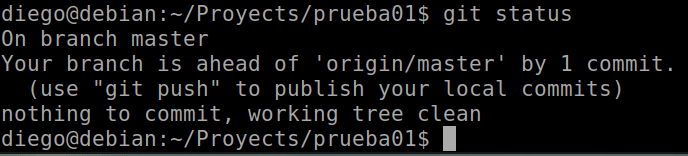
Ahora podemos ver el repositorio actualizado en GitHub:



Ahora vamos a modificar o agregar algún archivo al repo local y actualizar de nuevo el remoto.



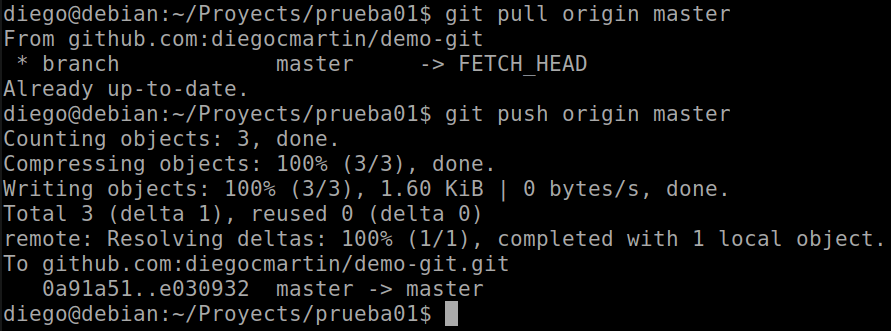
Si vemos el status ahora, este nos indica que hay cambios pendientes de sincronizar con el remoto:



Aunque ahora se esta trabajando sólo, lo lógico es solicitar los últimos cambios mediante un pull, antes de enviar los nuestros con push, para evitar conflictos con otros miembros del equipo. De modo que ejecutamos:

git pull origin master

git push origin master



Ya podemos ver los nuevos cambios en el repositorio remoto.

Al hacer pull, el sistema recupera y trata de unir la rama remota con la local, mientras que con el comando fetch que veíamos antes no.

**repositorio Git - Conectando**

1- Lo primero que hacemos es inicializar el repositorio de Git en la carpeta que queramos trabajar, que en nuestro caso será ~/Desktop/CodigosTemas, con el comando:

git init

2- Después nos vinculamos contra el repositorio remoto:

git remote add origin

[https://github.com/ingcarlosmontenegro/CodigosTemas.git](https://github.com/ingcarlosmontenegro/CodigosTemas.git 3)

[3](https://github.com/ingcarlosmontenegro/CodigosTemas.git 3) - Una vez efectuado el vínculo comprobamos qué objetos hay y las ramas que existen:

git fetch

4 - El siguiente paso será descargar el código:

git pull origin master

5 - Añade archivos al repositorio

Una vez tenemos el repositorio clonado tenemos que añadir todos los archivos y directorios del proyecto al repositorio, es muy fácil, añada o cambia archivos, verifica los cambios con:

git diff

6 - Añade los cambios con:

git add .

7 - Haz el primer commit

El comando anterior solo añade archivos al repositorio, no hace commit de los cambios realizados. Por tanto, debes hacer el primer commit:

git commit -m "Mi primer commit"

8 - Tu primer push

Una vez configurado el repositorio remoto ya puedes hacer el primer push de cambios:

git push origin master

**repositorio Git - Clonando**

1 - Clona un repositorio remoto

Para poder usar Git lo primero que tienes que hacer es abrir la terminal y ubicarte en el directorio donde están ubicadas las fuentes de tu proyecto y clonar un repositorio:

cd <directorio-proyecto>

git clone https://github.com/ingcarlosmontenegro/CodigosTemas.git

Acabas de clonar el repositorio Git a local correspondiente a tu proyecto.

2 - Añade archivos al repositorio

Una vez tenemos el repositorio clonado tenemos que añadir todos los archivos y directorios del proyecto al repositorio, es muy fácil, añada o cambia archivos.

verifica los cambios con:

git diff

añade cambios con:

git add .

3 - Haz el primer commit

El comando anterior solo añade archivos al repositorio, no hace commit de los cambios realizados. Por tanto, debes hacer el primer commit:

git commit -m "Mi primer commit"

4 - Tu primer push

Una vez configurado el repositorio remoto ya puedes hacer el primer push de cambios:

git push origin master

**Como continuar trabajando con Git**

Ahora ya tienes un repositorio Git local, has hecho tu primer commit y también has hecho tu primer push al repositorio alojado en GitHub. ¿Y ahora que?

Ahora debes continuar con el desarrollo de tu proyecto y de vez en cuando hacer commit al repositorio local, de esta forma tendrás un historico de cambios y podrás revertir cualquier cambio que hagas.

Hacerlo es muy fácil, es casi igual que la primera vez, solo que ahora debes pasar la opción -a al hacer commit:

git add .

git commit -a -m "Descripción del commit"

Por otro lado, cada vez que completes alguna característica o corrijas algún bug conviene que hagas push al repositorio remoto, para que el resto de programadores puedan obtener el cambio. El comando a ejecutar es igual que la primera vez:

git push origin master

Ahora si lo que quieres es bajar las actualziaciones del repositior, que es muy pertinetne hacerlo cada vez que inices a trabajar:

git pull origin master

Resolver conflictos

git fetch origin

git reset --hard origin/master

git pull

Fetch descargará todo desde otro repositorio, en este caso, el marcado como "origen".

Restablecer descartará los cambios y volverá a la rama mencionada, "maestro" en el repositorio "origen".

Pull obtendrá todo de un repositorio remoto e integrará.

Manejo de ramas

git branch NombreRama  
git checkout hola

Switched to branch 'hola'

git add .

git checkout master

Switched to branch 'master'

$ git merge hola

Resolver coinflictos

git add //agrehar los archivos.

$ git commit -m "Solucionado el conflicto al fusionar con la rama master"

Check

Git branch

Resumen pasos para manejar git

**Git Local**

1. Cree una carpeta para alojar su repositorio local

Mkdir repositorio

1. Creo el repo en esa carpeta

Cd repositorio

Git init

1. Cree sus archivos, trabájelos y súbalos con git add

Git add .

1. Haga un commit

Commit -m “subir archivos”

**Conectar con repo remoto**

Clonando el repo

1. Clone su repo, recuerde colocar la url de su repo

Git clone <https://github.com/usuariogit/EjemploNuevo.git>

1. Tenga en cuenta que esto crea una nueva carpeta con el repositorio remoto clonado, para nuestro caso EjemploNuevo, con lo cual la información debe quedar de aquí en adelante en esa nueva carpeta.

En este caso vamos a mover todos los archivos creados a la nueva carpeta.

Cd EjemploNuevo

1. Cree sus archivos, trabájelos y súbalos con git add

Git add .

1. Haga un commit

Git Commit -m “subir archivos”

1. Suba lo que tiene en su repositorio local al remoto

Git push origin main