

se enumera los videos

1. **introduccion**

Git:

- sistema control de versiones

- permite auditoria de codigo, control de cambios

- es diferente a github, bitbucket, gitlab

Github:

- es como una red social que usa git

1. **instalacion**

https://git-scm.com/

ver la vesion desde el bash como desde la terminal

git --version

1. **Primer Repositorio y Commit**

- configurar las credenciales:

git config --global user.name "rockywgomsa"

git config --global user.email "wgomsa2018gmail.com"

- ver las credenciales:

git config --global -l

- empezar git, se hace una sola vez. y debe aparecer una carpeta oculta .git

git init

- ver el status

git status

- adicionar archivos al git. con esto los archivos se ponen en una zona llamada "stage"

git add .

- confirmar los cambios

git commit -m "algun mensaje"

- ver los cambios que se haya realizado al archivo

git diff

- quitar los cambios, dejar como estaba

git checkout .

- ver el historico de los commit

git log

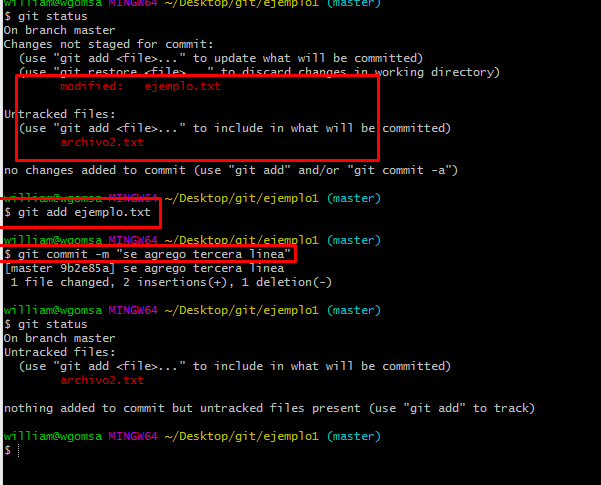
1. **Stage y Logs**

podemos hacer un commit de un archivo que querramos (no del todo)

git add archivo

git commit -m "mensaje"

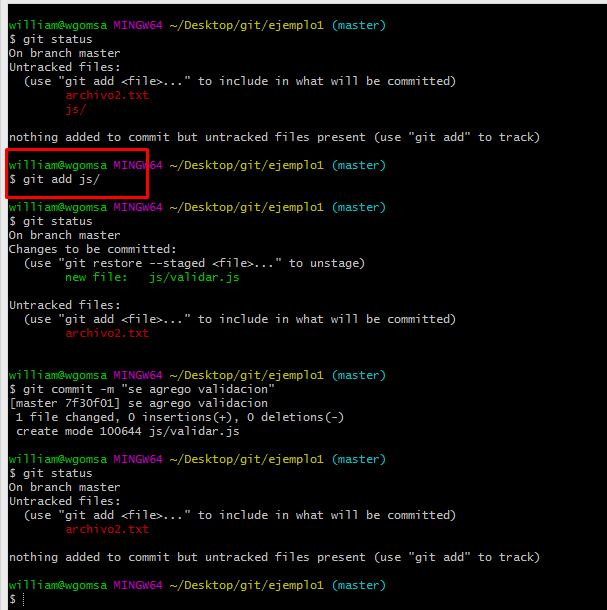
y cuando le hagamos un status, el otro que no se considero seguire pendiente



trabajando sobre un directorio.

git add directorio/

git commit -m "mensaje"

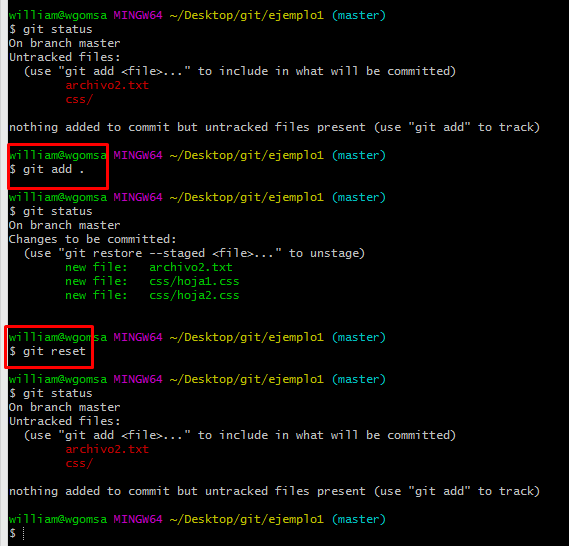


quitar del stage lo que agregue, puede ser todo o un archivo .

todo:

git add .

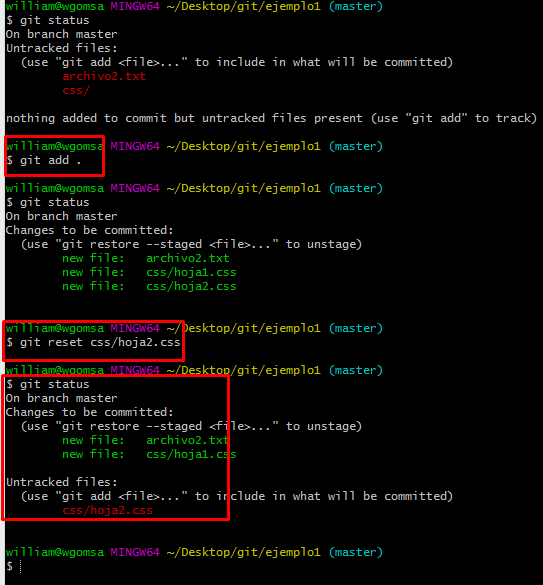
git reset



quitando un solo archivo

git add .

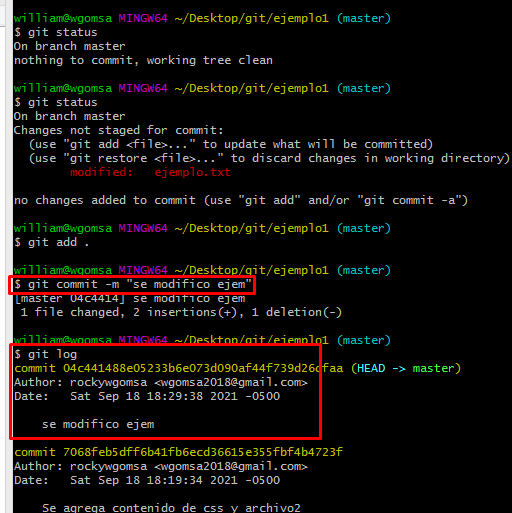
git reset carpeta/archivo



1. **Git Alias**
2. **Git MV/RM/Reset**

si nos equivocamos al poner el nombre del mensaje, se usa amend.

pongo mal el mensaje



rectifico el mensaje

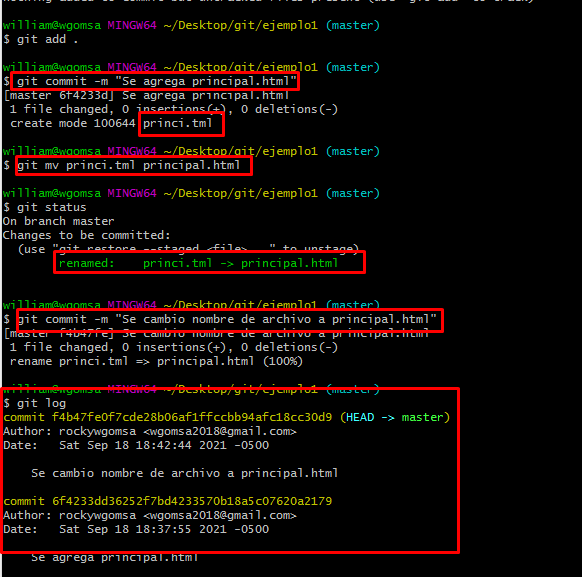
git commit --amend -m "mensaje"

cambiar el nombre del archivo comiteado

git add .

git commit -m "mensajito"

git mv nombre\_archivo nuevo\_nombre\_archivo

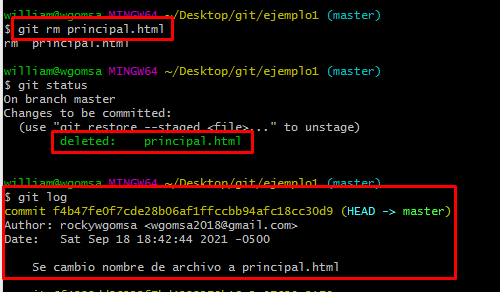


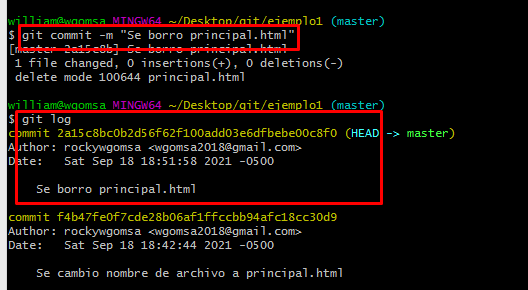
eliminando archivo y luego ir a ese punto donde estaba

git rm archivo

git status

commit



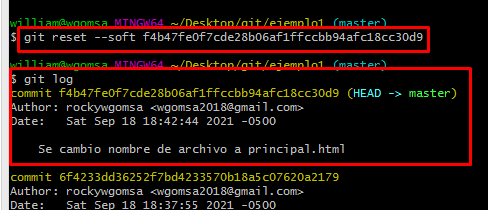


ir al punto donde estaba:

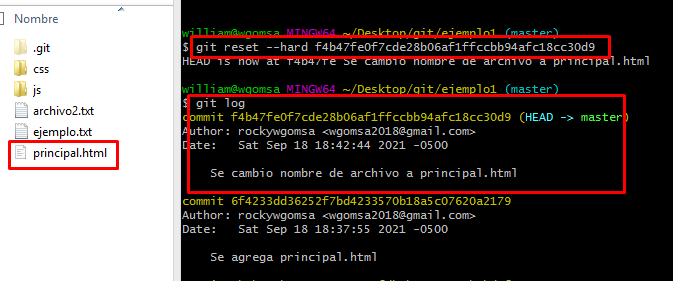
con soft: solo se ubica mas no recupera

con hard: se ubica y recupera

soft:

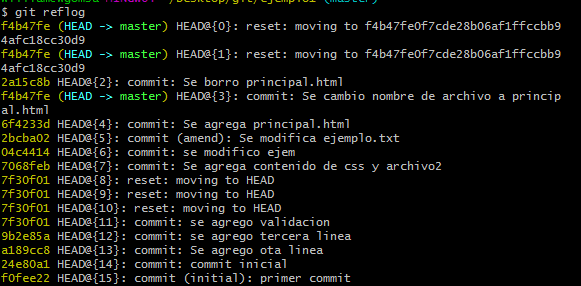


hard:

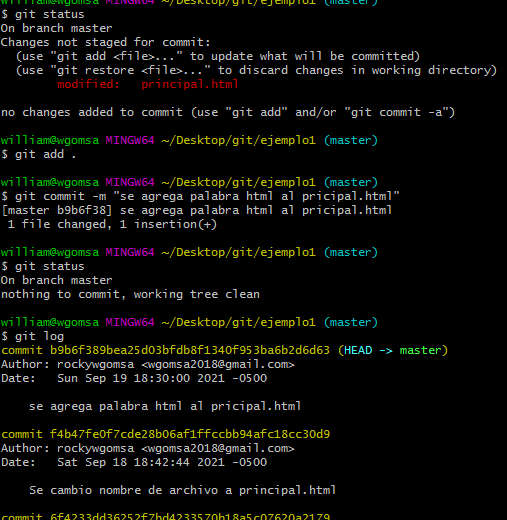


ver los historicos de los commit

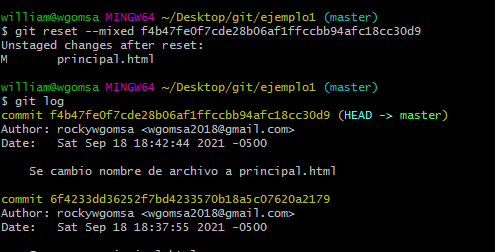
git reflog



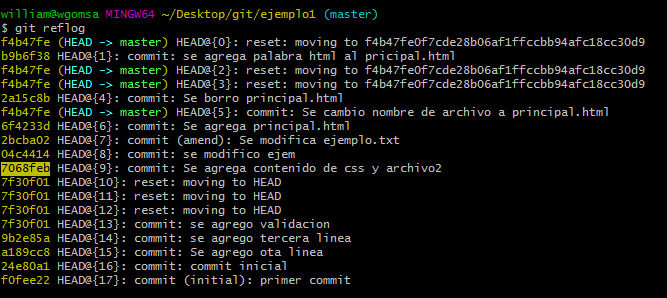
voy agregar palabra html al principal.html y luego me voy a ir a un punto a tras pero sin perder ese cambio. Eso es posible usando el reset con mixed, pero el mixed tambien funciona si lo se pone: git reset --mixed id\_punto



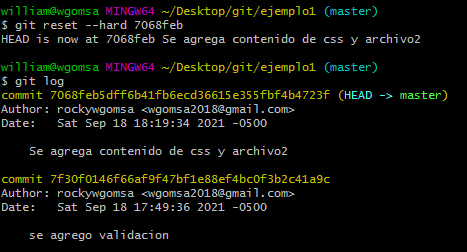
git reset --mixed f4b47fe0f7cde28b06af1ffccbb94afc18cc30d9



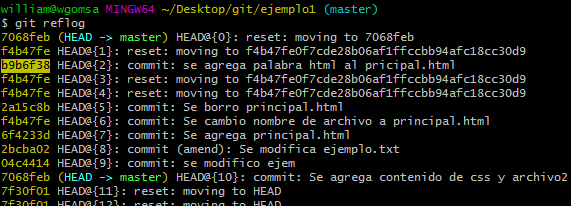
moverse entre puntos trayendo los cambios, es usando el --hard



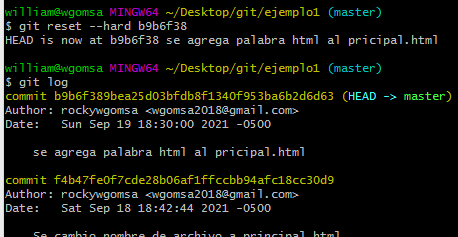
git reset --hard 7068feb



ahora me voy al ultimo commit que hice



git reset --hard b9b6f38



1. **gitignore**

es crear el .gitignore y colocar dentro lo que no quiero commitear

1. **Ramas y Uniones (Merge)**

crear rama:

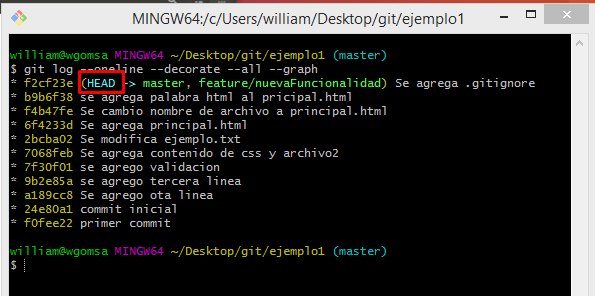
git branch nombre\_rama --> con esta la crea pero no se dirige a esa rama

git chekout -b nombre\_rama --> la crea y se dirige a esa rama

para ver mas resumidamente los cambios:

git log --oneline --decorate --all --graph

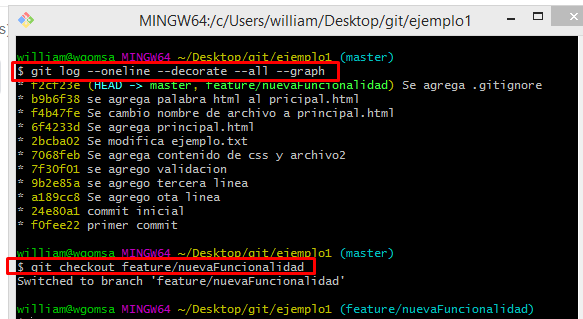
el HEAD me indica que estoy en la master y hay una rama feature/nuevaFuncionalidad



moverse a otra rama

git checkout nombre\_rama

git checkout feature/nuevaFuncionalidad



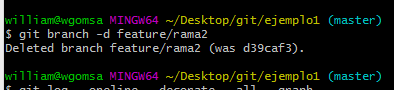
merge entre ramas:

hacer merge de rama1 a rama2: situarse en la rama2 y colocar: git merge rama1

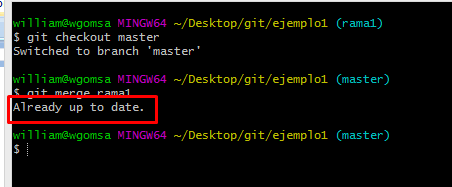
merge de tipo Fast-forward: sin conflictos

eliminar rama, situarse en una rama distinta a la que se va eliminar:

git branch -d nombre\_rama



el mensaje Already up to date es porque no se comiteo a la rama1



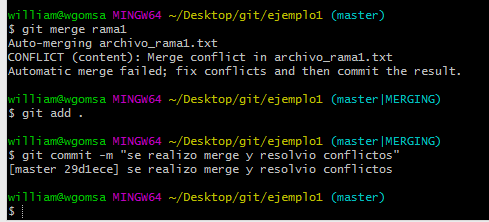
1. **Merge con conflictos**

en la master modificamos un archivo y comiteamos

creamos una rama y en esa rama modificamos el mismo archivo y lo comiteamos

vamos a la master y modificamos de nuevo el mismo archivo

hacemos merge de rama1 a master y saldra conflictos. corregir y luego comitearlo



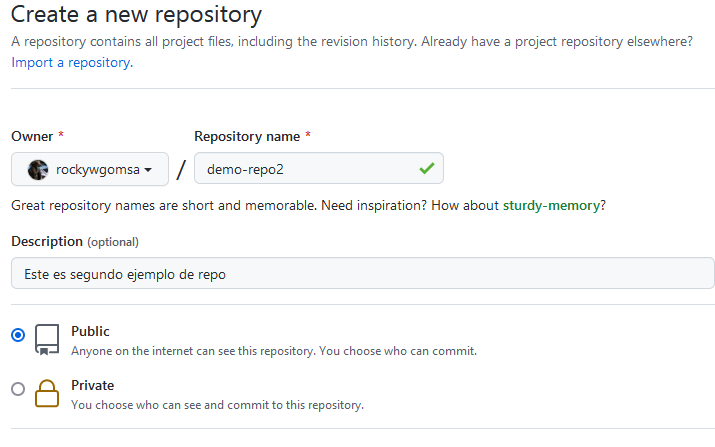
1. **Etiquetas (tags)**
2. **Github (git clone)**

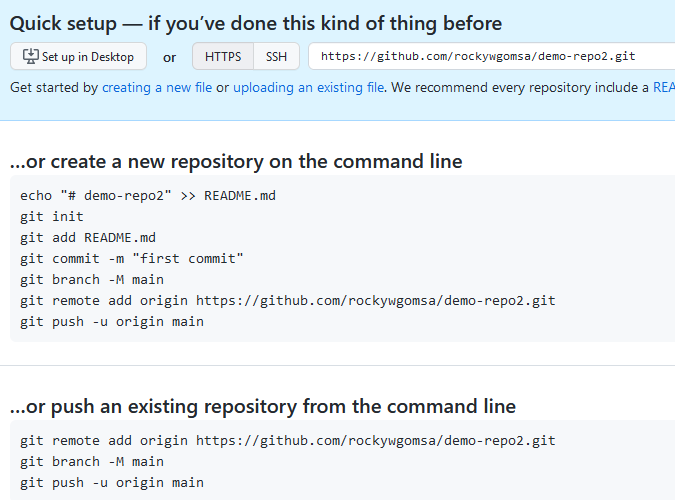
https://github.com/mitocode21

crear una cuenta desde Github

hay 3 escenarios:

11.1. cuando se crea el repo y localmente aun no se ha controlado con git, es decir, no se tiene un .git.





en la pc local tendremos una carpeta sin el .git, para ello abrir git bash y seguir los pasos primera opcion

git init

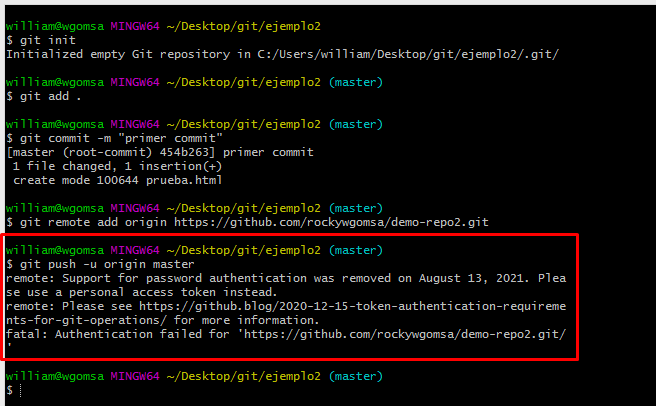
git add .

git commit -m "primer commit"

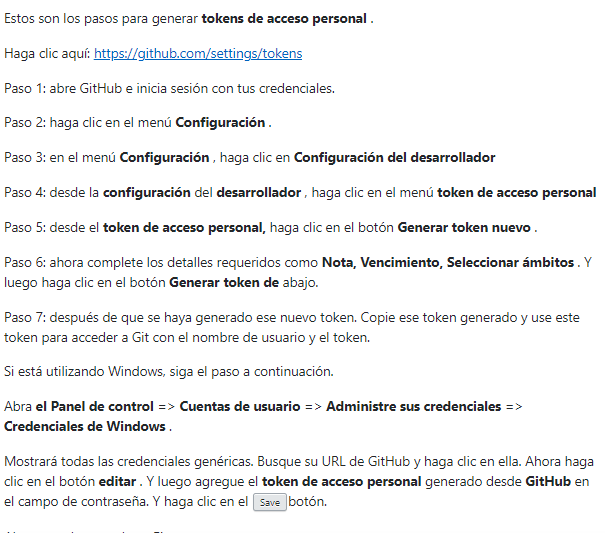
git remote add origin https://github.com/rockywgomsa/demo-repo2.git

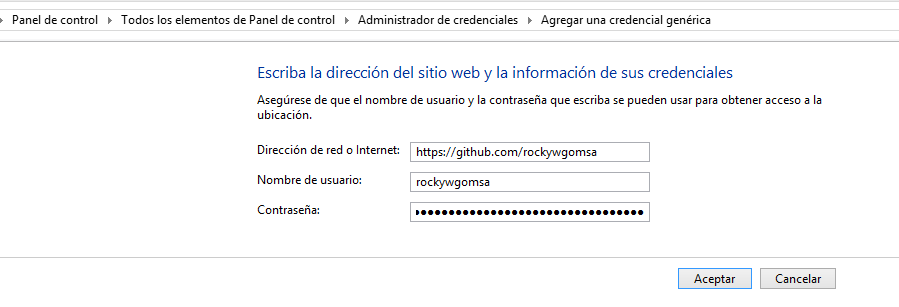
git push -u origin master

salio error porque ahora no es por claves sino por token

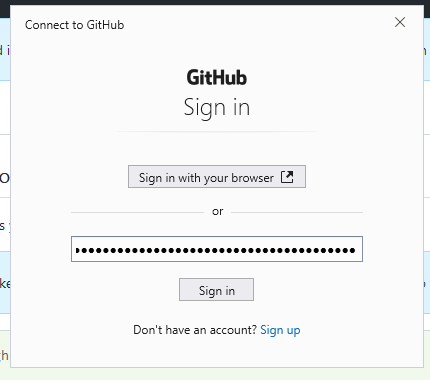


<https://github.com/settings/tokens>

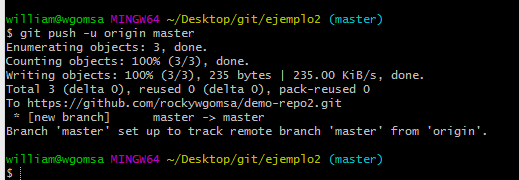




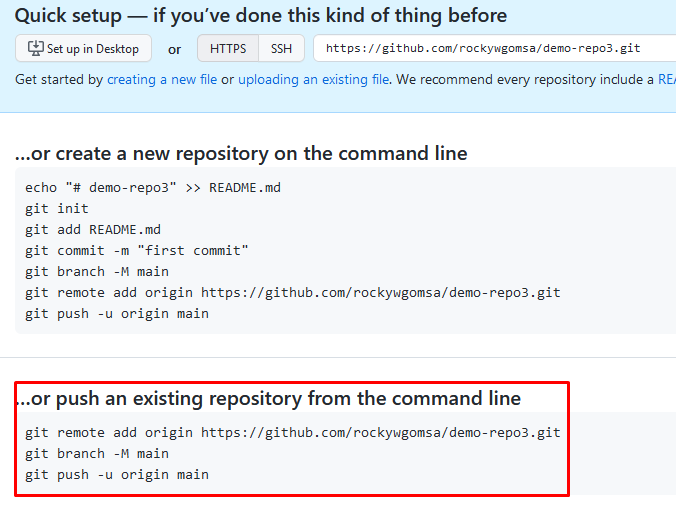
luego de darle : git push -u origin master me salio para logearme con mi token



y finalmente subio el cambio



11.2. cuando se tiene un repo creado y localmente se tiene controlado con git, es decir,tenemos un .git. para nuestro caso la carpeta "ejemplo1". abrir el git bash en esa carpeta



git remote add origin https://github.com/rockywgomsa/demo-repo3.git

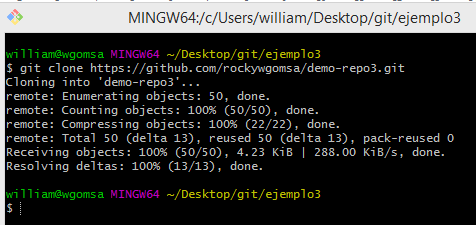
git push -u origin master



11.3. cuando necesitamos clonar algun repo.

en este caso creamos una carpeta "ejemplo3", abrimos git bash y vamos a clonar el demo-repo3.

git clone https://github.com/rockywgomsa/demo-repo3.git



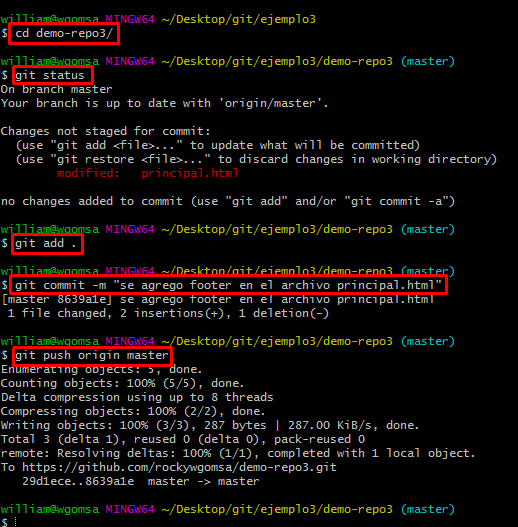
y dentro de este modificamos algún archivo para luego subir, estos son los pasos.

git status

git add .

git commit -m "se agrego footer en el archivo principal.html"

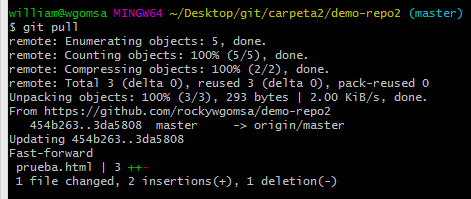
git push origin master ---> aca ya no puse -u origin



1. **git pull**

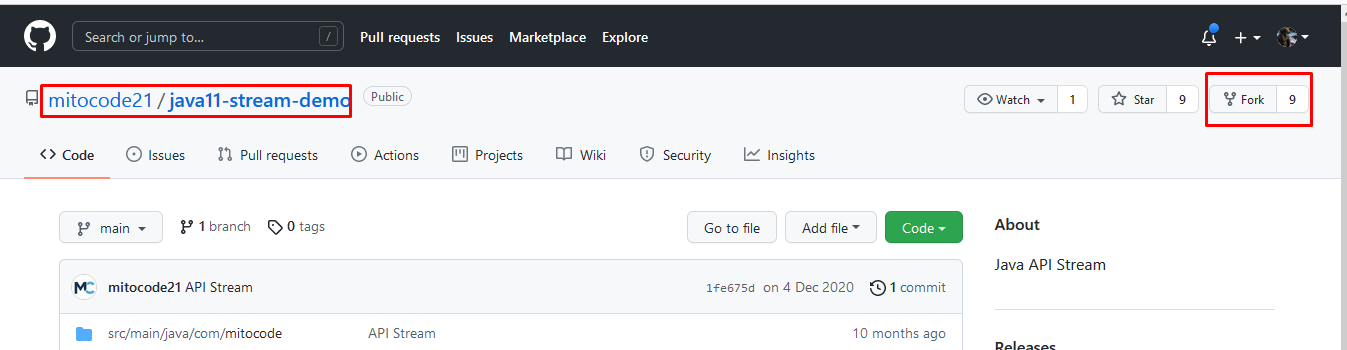
para traer los cambios que se tengan

git pull



1. **fork**

Sirve para copiar un repositorio que sea de nuestro agrado. ir a ese repositorio y darle en el boton fork. y luego podemos clonarlo para hacerle algun cambio.



1. **pull request**

es para sugerir cambios a un repositorio que le dimos fork

1. **Integración con NetBeans, Eclipse, Intellij y Visual Studio**