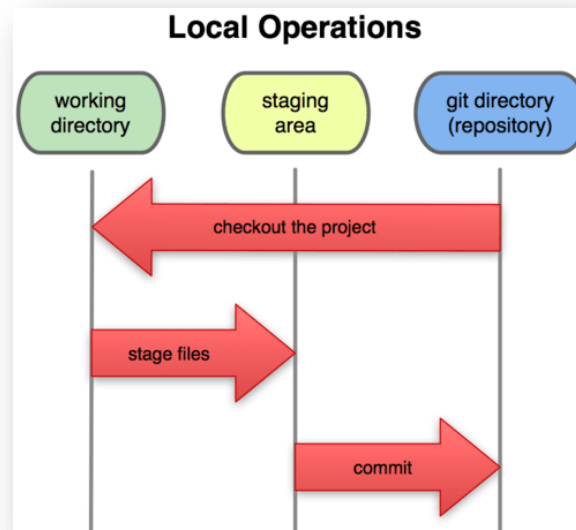


1. MANUAL DE USO PARA GIT

Punto del libro 1.3



Primero clono el archivo, luego trabajo sobre él, lo preparo antes de subirlo (stage files) y luego lo confirmo en otro momento (o a la vez) (commit).

Donde se guarda en el **Working Directory** mis archivos se llama **/mirepo**

En el **repositorio**, que es una BD y un sistema de archivos de metadatos (no lo tocamos, lo usamos desde el software) en este repo, hay una carpeta oculta llamada **/miRepo/.git**, mejor no tocar.

El **stating area** no es físico, es un fichero donde marca los archivos que están listos para subir.

Cualquier cambio que yo haga antes de la preparación, puedo deshacerlo volviendo a la foto anterior que está en el repo.

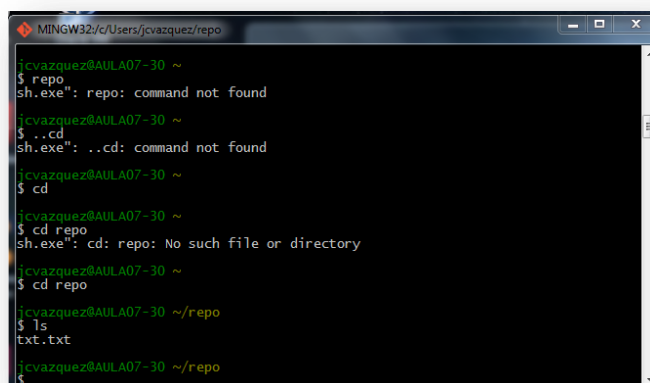
1.1 Ejemplos con Git

Abro el ejecutable **Gitbush**, guardo una carpeta en algún sitio del pc local, para controlar. En nuestro caso **repo**, tengo que mover la posición de la línea de comando, uso **pwd** (donde estoy, directorio), **ls** (lo que hay en la carpeta, los uso para moverme y saber donde estoy y como llegar a la carpeta creada (**repo**), si lo pongo en usuario puedo escribir directamente

```
cd/repo
```

SIEMPRE tengo que empezar poniendo la dirección de mi carpeta que estoy controlando, para que todos los comandos actúen sobre esa carpeta, se pondrá dentro de la línea un texto (master)

..ahora le hago un listado con **ls**,



```
MINGW32/c/Users/jcvazquez/repo
jcvazquez@AULA07-30 ~
$ repo
sh.exe": repo: command not found
jcvazquez@AULA07-30 ~
$ ..cd
sh.exe": ..cd: command not found
jcvazquez@AULA07-30 ~
$ cd
jcvazquez@AULA07-30 ~
$ cd repo
sh.exe": cd: repo: No such file or directory
jcvazquez@AULA07-30 ~
$ cd repo
jcvazquez@AULA07-30 ~/repo
$ ls
txt.txt
jcvazquez@AULA07-30 ~/repo
$
```

Escribo "git init" para iniciar el repositorio en Git, y decirle al programa que me cree un repositorio y que comience a gestionar esa carpeta. Ahora le digo que me controle mi fichero txt.

git add txt.txt



```
jcvazquez@AULA07-30 ~
$ cd
jcvazquez@AULA07-30 ~
$ cd repo
sh.exe": cd: repo: No such file or directory
jcvazquez@AULA07-30 ~
$ cd repo
jcvazquez@AULA07-30 ~/repo
$ ls
txt.txt
jcvazquez@AULA07-30 ~/repo
$ git init
Initialized empty Git repository in c:/Users/jcvazquez/repo/.git/
jcvazquez@AULA07-30 ~/repo (master)
$ git add txt.txt
jcvazquez@AULA07-30 ~/repo (master)
$
```

Ahora está en el estado staging , puedo escribir **git status** para ver como está.

Ahora pulso **git commit -m "Añado txt"** Se suele poner un comentario entre "" , me saldrá los cambios realizados, (he subido el archivo). Tenemos que configurar un nombre de usuario para saber quien lo hace. **git config --global user.name "your name"**. Debe de aparecer después del commit , 1 file changed, 1 insertion(+). Si ahora le doy a **git status** , me sale que no hay cambios por realizar.

Modifico el txt, hago status, y me dice que lo he modificado, ahora lo preparo y luego lo subo, primero add y luego commit. (se pueden poner patrones, *para todos etc..).

En qué estado estoy?, preparado, antes del commit y luego subido cuando hago commit

(pulsar doble tab te pone lo que falta en el comando)

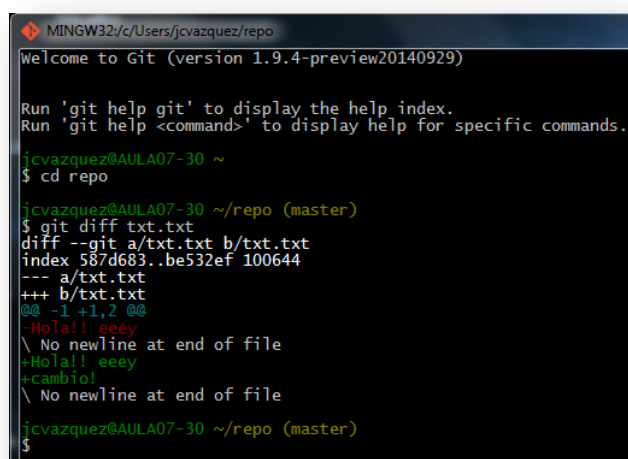
Si no pongo comentario, le doy a la i, cuando salga la pantalla fea, para el modo edición, pongo comentario pulso (esc) y escribo al pie de ventana **:wq!**. Y estoy en el modo edición para escribir el comentario.

31/10/2014

Para volver a entrar (si he salido del programa) siempre pongo en la consola `cd repo`, en nuestro caso el repositorio que creamos previamente.

Vamos a modificar el txt, desde el archivo, a continuación miro en la consola,

git diff nombrefichero.txt



```
MINGW32/c/Users/jcvazquez/repo
Welcome to Git (version 1.9.4-preview20140929)

Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

jcvazquez@AULA07-30 ~
$ cd repo

jcvazquez@AULA07-30 ~/repo (master)
$ git diff txt.txt
diff --git a/txt.txt b/txt.txt
index 587d683..be532ef 100644
--- a/txt.txt
+++ b/txt.txt
@@ -1,2 @@
-Holal! eeey
\ No newline at end of file
+cambio! eeey
\ No newline at end of file

jcvazquez@AULA07-30 ~/repo (master)
$
```

Me muestra las diferencias entre mi archivo y el que está en el repositorio (nube), aunque lo hemos cambiado aún no lo he subido.

Si pulsamos **history** me sale el historial de los comandos que estamos usando.

Pongo **git status** para ver el estado,

Sigo con mi ejemplo, si me doy cuenta que el trabajo que he hecho no sirve, siempre puedo hacer un **git checkout – nombrefichero.txt**, descarta los cambios que he hecho en el archivo y me trae la última versión confirmada (commit) del repositorio.

Cambio de nuevo el fichero.txt (se puede hacer directamente desde consola con **vim**), hago un **git status**, luego un **git add nombrefichero.txt**. ver la siguiente ilustración.

```
jcvazquez@AULA07-30 ~/repo (master)
$ git add
Nothing specified, nothing added.
Maybe you wanted to say 'git add .'?

jcvazquez@AULA07-30 ~/repo (master)
$ git add txt.txt

jcvazquez@AULA07-30 ~/repo (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   txt.txt

jcvazquez@AULA07-30 ~/repo (master)
$
```

Si quisiera deshacer lo que he “preparado” y quiero volver al estado “despreparado”, hago;

git reset HEAD nombrefichero.txt

Luego hago **git status** , veo que no he perdido cambios, sólo he desmarcado el fichero preparado, a despreparado. Podría subirlo directamente sin prepararlo primero, para eso añado un **-a** en el commit.

git commit -a -m “otro commit”

Si he subido pero se me ha olvidado algo en ese último commit, y no quiero que tenga dos “fotos” diferentes en la misma versión,

git commit --amend -a -m “otro commit”

Se añade cosas a la última versión sin que parezca que lo he subido en dos veces.

04/11/2014

Iniciar git bush y buscar la ruta del repo (**cd repo**).

Hoy subiremos los archivos o bajaremos desde github (trabajamos en repositorios remotos). Me voy a la página github.com para crear un repositorio. Puedo crear un nuevo repo desde la pestaña de usuario (arriba) *new repository*.

El nombre que le damos es repoclase, luego en gitbush nos situamos en una carpeta para traer el nuevo repositorio a nuestro pc. (Esa carpeta es mejor llamarla igual que el repo, aunq no hace falta crearla, se crea al bajarla)

mkdir Repoclase

Para copiar ahora la carpeta, (una vez situado en gitbush),

git clone http://github.com/Hasimov/Repoclase

```
jc vazquez@AULA07-30 ~/Repoclase
$ git clone http://github.com/Hasimov/Repoclase
Cloning into 'Repoclase'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
Checking connectivity... done.
```

Para hacer un commit, en local puedo hacer lo que quiera, tenemos dos copias (pc y github) y los commit del pc se hacen locales (no cambian en github, remoto),

git add pruebaword.docx

commit

```
jc vazquez@AULA07-30 ~/Repoclase/Repoclase (master)
$ git add pruebaword.docx

jc vazquez@AULA07-30 ~/Repoclase/Repoclase (master)
$ git commit pruebaword.docx -m "Subo el word"
[master f155330] Subo el word
Committer: Hasimov <jc vazquez@AULA07-30.aulas.triana>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 pruebaword.docx
```

Para subirlo a la nube **git push origin master**, origin es el remoto, me pedirá el usuario y la clave.

Para actualizar lo que hay en github, puedo editar el readme desde la web y abajo tengo el commit. Ahora la versión web está más actualizada, por lo que ahora uso el ;

git pull origin master

para traer lo nuevo a la carpeta local, puedo comprobar los cambios con,

cat readme.md

Los repositorios son diferentes, el local y el remoto, los add y commit son independientes entre ellos, hasta que los suba o baje con los push y pull.

10/11/2014

Para trabajar **colaborativamente** con Github hay dos maneras,

1. Doy permisos a todos, arriesgado.
2. Como hay varios repo, cuando yo quiera contribuir a algo, me hago una copia del repo y cuando crea que el commit es válido para todos envío una solicitud al administrador que valora si es aceptable, si es así se actualizará para todos los usuarios.

Con el fork, me hago una copia local del repo, cuando considere que se puede integrar mi parte hago un **pull request** (petición al administrador)