

GIT: MÁS COMANDOS

- Ignorar ficheros del control de versiones.

Creamos 1 fichero

```
echo "fichero 1" > fich1.tmp
```

```
git status
```

Se crea un fichero .gitignore

y se le añade una máscara

```
*.tmp
```

No se agregará nunca al área de staging.

```
git status
```

Incluyo todos los ficheros del Working directory para pasarlos al área de Staging o preparados.

```
git add .
```

```
git status
```

Se hago git add fich1.tmp da el siguiente error:

```
osachace1@DESKTOP-RH7U6TK MINGW64 ~/OneDrive/Escritorio/test/escenario5 (master)
$ git add fich1.tmp
The following paths are ignored by one of your .gitignore files:
fich1.tmp
hint: Use -f if you really want to add them.
hint: Turn this message off by running
hint: "git config advice.addIgnoredFile false"
```

Para añadir .gitignore lo tengo que hacer con

`git add .gitignore.`

- Ver los cambios de un fichero modificado

Me creo un nuevo fichero `readme2.md` y modifico `readme.md`.

Ahora tendré dos advertencias en rojo.

Subo todo al área de Staging

`git add .`

`git commit -m "revisión 3"`

Ahora modifico el fichero `readme.md`

Hacemos `git status` y nos dice que hay un cambio y para ver en que difiere hacemos

`git diff`



```
Lucia@LAPTOP-LV01PEG0 MINGW32 ~/Desktop/test/escenario1 (master)
$ git diff
diff --git a/readme.md b/readme.md
index cef11ee..21460b5 100644
--- a/readme.md
+++ b/readme.md
@@ -1,2 +1,2 @@
 # primera linea d
 ## eun fichero readme.md
 ## eun *fichero* readme.md
Lucia@LAPTOP-LV01PEG0 MINGW32 ~/Desktop/test/escenario1 (master)
$
```

Hace la diferencia entre el Working Directory y el repositorio remoto.

git difftool

me aparece vimdiff. Pero vamos a instalar tkdiff

Copiar tkdiff en la siguiente ruta de git

C:\Program Files\Git\usr\bin

git difftool

ahora nos reconoce tkdiff

- Borrado de ficheros

Borrado de fichero

rm readme.md

git status

la forma de hacerlo debería ser

git rm readme.md

Lo borra y lo pasa directamente al área de staging.

- Movimiento de ficheros

Lo mismo para mv.

Hace dos cosas git considera que se ha borrado un fichero y se ha creado uno nuevo, luego hay que hacer un git add . para guardar los dos cambios. Lo mejor es hacer los siguiente

git mv fich1 fich2

- Mas opciones de git log

--oneline

Devuelve el identificador de la versión y la etiqueta.

```
rosachacel@DESKTOP-RH7U6TK MINGW64 ~/OneDrive/Escritorio/test/escenario1 (master)
$ git log --oneline
3b446d3 (HEAD -> master) nuevo fichero readme3.md
8e3858f mensaje de mi primer commit
```

el id de la versión es un hash, por eso se suele llamar también “hash id”.

HEAD es la etiqueta de la revisión más actual, se va a utilizar para hacer búsquedas: desde HEAD tres versiones atrás, etc.

`git log --pretty="%h %an %ar %s"`

h -> hashid

an -> autor name

ar -> de cuando es el cambio

s -> nombre de la revisión

```
osachacel@DESKTOP-RH7U6TK MINGW64 ~/OneDrive/Escritorio/test/escenario1 (master)
$ git log --pretty=format:"%h %an %ar- %s"
b446d3 David Arroyo 10 minutes ago- nuevo fichero readme3.md
e3858f David Arroyo 2 weeks ago- mensaje de mi primer commit
```

NOTA: format se puede omitir.

- Chequear revisiones

`git show hashid`

```
osachacel@DESKTOP-RH7U6TK MINGW64 ~/OneDrive/Escritorio/test/escenario1 (master)
$ git show e3858f
commit e3858ffe8ddb57e4544f7b38245194129e4e2f
Author: David Arroyo <darroyo3@educa.madrid.org>
Date: Sun Oct 18 22:14:39 2020 +0200

    mensaje de mi primer commit

diff --git a/readme.md b/readme.md
new file mode 100644
index 0000000..a547ece
--- /dev/null
+++ b/readme.md
@@ -0,0 +1 @@
+#Primera linea del fichero markdown

osachacel@DESKTOP-RH7U6TK MINGW64 ~/OneDrive/Escritorio/test/escenario1 (master)
```

Muestra los cambios de una versión respecto a la anterior

Paso de no tener nada /dev/null a tener un fichero.

Para ver las diferencias entre un fichero en el área de staging y el repositorio local

`git diff --staged`