

Sistema de control de versiones

Actividades iniciales



git

Javier Soler Cantó
2 DAW - Despliegue de Aplicaciones

1. Configurar Git definiendo vuestro nombre de usuario, correo electrónico y el editor por defecto (por ejemplo Visual Studio Code).

```
javi — -zsh — 80x24
[javi@MacBook-Pro ~ % git config --global user.name "javisolercanto" ]
[javi@MacBook-Pro ~ % git config --global user.email "jsolercanto11@gmail.com" ]
[javi@MacBook-Pro ~ % git config --global core.editor "code --wait" ]
[javi@MacBook-Pro ~ % git config --list ]
credential.helper=osxkeychain
user.name=javisolercanto
user.email=jsolercanto11@gmail.com
core.editor=code --wait
javi@MacBook-Pro ~ %
```

Accedemos a la configuración de Git y su entorno con **git config --global**. Ahora para poder cambiar el nombre usuario utilizaremos **user.name**, para el email usaremos **user.mail**, para que se habrá nuestro editor favorito **core.editor**.

2. Crear un repositorio con el nombre actividadesPrimeraClaseDAW y mostrar el contenido por consola (incluyendo el que sea oculto)

```
actividadesPrimeraClaseDAW — -zsh — 80x24
[javi@MacBook-Pro actividadesPrimeraClaseDAW % git init ]
Initialized empty Git repository in /Users/javi/Documents/DAW/DA/T1/E1/actividadesPrimeraClaseDAW/.git/
[javi@MacBook-Pro actividadesPrimeraClaseDAW % ls -la ]
total 0
drwxr-xr-x  3 javi  staff   96 10 sep 21:19 .
drwxr-xr-x  3 javi  staff   96 10 sep 21:19 ..
drwxr-xr-x  9 javi  staff  288 10 sep 21:19 .git
javi@MacBook-Pro actividadesPrimeraClaseDAW %
```

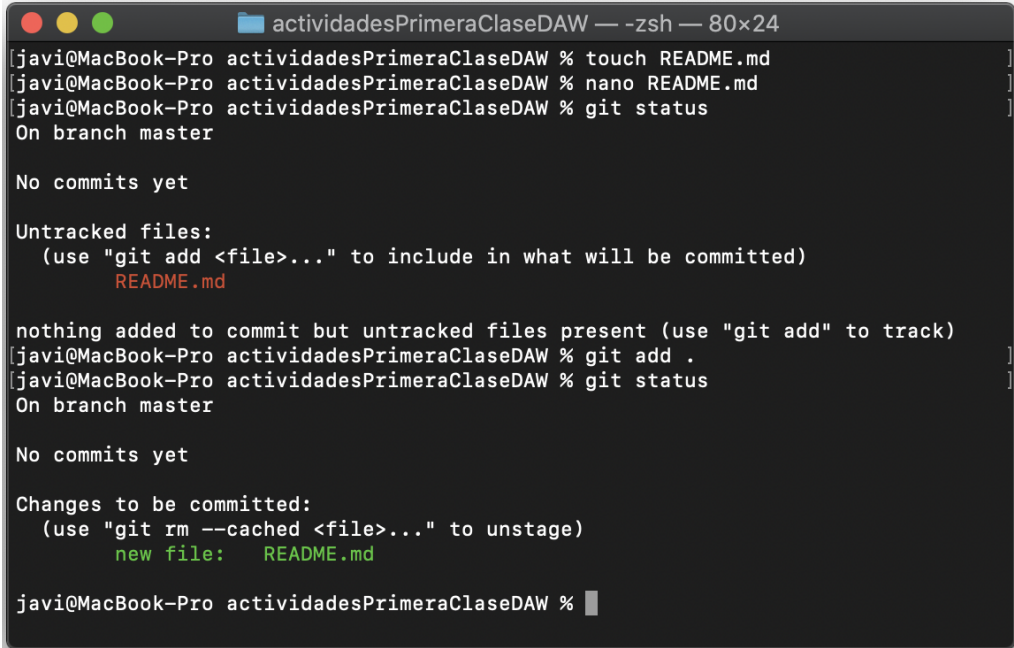
Para crear el repositorio podemos crear la carpeta desde el gestor de archivos y luego utilizar el comando **git init** o usar directamente **git init folder/** para crear la carpeta e inicializar el repositorio git directamente. Para mostrar el contenido de la carpeta **ls -la** donde el segundo parámetro nos permite mostrar los archivos ocultos y los permisos sobre ellos.

3. Crear un fichero README.md en el que incluiremos lo siguiente:

Este repositorio contiene una recopilación de actividades muy sencillas realizadas en la primera clase sobre Sistemas de control de versiones.

1. Configuración de algunas preferencias
2. Instrucciones para la creación de un primer repositorio
3. Añadir un archivo al repositorio moviéndolo de la stage area a la zona local comprobando el estado.

Comprobar el estado de dicho fichero. Tras ello, añadirlo a la stage area y, una vez realizado este paso, volver a comprobar el estado del repositorio.



```
actividadesPrimeraClaseDAW — zsh — 80x24
[javi@MacBook-Pro actividadesPrimeraClaseDAW % touch README.md
[javi@MacBook-Pro actividadesPrimeraClaseDAW % nano README.md
[javi@MacBook-Pro actividadesPrimeraClaseDAW % git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  README.md

nothing added to commit but untracked files present (use "git add" to track)
[javi@MacBook-Pro actividadesPrimeraClaseDAW % git add .
[javi@MacBook-Pro actividadesPrimeraClaseDAW % git status
On branch master

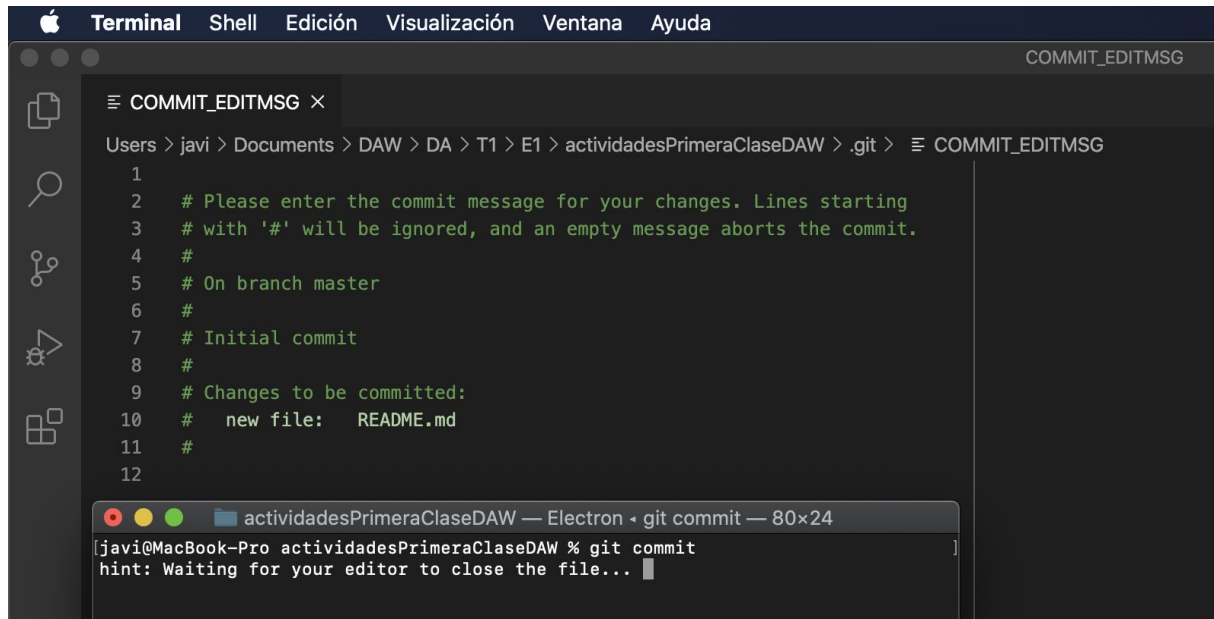
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
  new file:   README.md

javi@MacBook-Pro actividadesPrimeraClaseDAW %
```

Para saber en qué estado se encuentra nuestra mesa de trabajo usaremos el comando **git status**. Para poder añadirlo al área de stage sin confirmar el cambio usaremos el comando **git add .**

4. Realizar un commit con los cambios realizados en el ejercicio anterior. El commit lo ejecutaremos sin incluir el mensaje en la misma instrucción para provocar que se nos abra el editor que hemos configurado en la actividad 1.

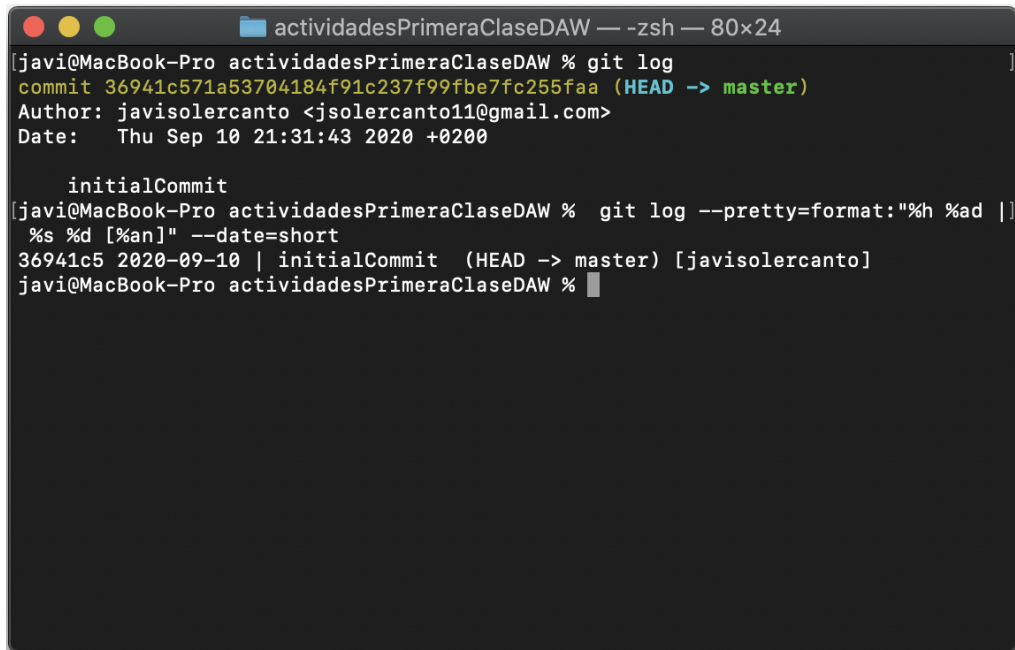


```
Users > javi > Documents > DAW > DA > T1 > E1 > actividadesPrimeraClaseDAW > .git > COMMIT_EDITMSG
1
2 # Please enter the commit message for your changes. Lines starting
3 # with '#' will be ignored, and an empty message aborts the commit.
4 #
5 # On branch master
6 #
7 # Initial commit
8 #
9 # Changes to be committed:
10 #   new file:   README.md
11 #
12
```

```
javi@MacBook-Pro actividadesPrimeraClaseDAW % git commit
hint: Waiting for your editor to close the file...
```

Para realizar el commit usaremos el comando **git commit** , si lo utilizamos sin el parámetro **-m** se nos abrirá nuestro editor seleccionado para escribir el mensaje de commit y si lo guardamos confirmaremos el commit. Para usar el parámetro **-m** debemos usarlo de la siguiente manera: **git commit -m “mensaje”**.

5. Mostrar la traza de log de git. Averiguar de qué forma podemos hacerlo para que en una línea aparezca el hash del commit, el autor, la fecha y hora del cambio.



```
actividadesPrimeraClaseDAW — zsh — 80x24
[javi@MacBook-Pro actividadesPrimeraClaseDAW % git log
commit 36941c571a53704184f91c237f99fbe7fc255faa (HEAD -> master)
Author: javisolerconto <jsolerconto11@gmail.com>
Date: Thu Sep 10 21:31:43 2020 +0200

    initialCommit
[javi@MacBook-Pro actividadesPrimeraClaseDAW % git log --pretty=format:"%h %ad |
%s %d [%an]" --date=short
36941c5 2020-09-10 | initialCommit (HEAD -> master) [javisolerconto]
javi@MacBook-Pro actividadesPrimeraClaseDAW %
```

Para mostrar la traza debemos usar el comando **git log** y gracias al parámetro **--pretty=format** podemos formatear la salida a nuestro gusto utilizando los parámetros que queremos que muestre como, por ejemplo, **%h** para el hash.

git log --pretty=format:"%h %ad | %s %d [%an]" --date=short

6. Averiguar de qué forma podemos deshacer el último commit por línea de comandos para que:
- Se deshaga el último commit pero el cambio se quede en el stage area
 - Se deshaga completamente el cambio del último commit y no exista ni en el stage area ni en nuestros ficheros locales.

```
actividadesPrimeraClaseDAW — -zsh — 80x24

nothing added to commit but untracked files present (use "git add" to track)
[javi@MacBook-Pro actividadesPrimeraClaseDAW % git add .]
[javi@MacBook-Pro actividadesPrimeraClaseDAW % git status]
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   reset.txt

[javi@MacBook-Pro actividadesPrimeraClaseDAW % git commit -m "reset"]
[master 4bcb774] reset
 1 file changed, 2 insertions(+)
 create mode 100644 reset.txt
[javi@MacBook-Pro actividadesPrimeraClaseDAW % git status]
On branch master
nothing to commit, working tree clean
[javi@MacBook-Pro actividadesPrimeraClaseDAW % git reset --soft HEAD~1]
[javi@MacBook-Pro actividadesPrimeraClaseDAW % git status]
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   reset.txt

javi@MacBook-Pro actividadesPrimeraClaseDAW %
```

```
[javi@MacBook-Pro actividadesPrimeraClaseDAW % git commit -m "resetHard"]
[master b9bbdfe] resetHard
 1 file changed, 2 insertions(+)
 create mode 100644 reset.txt
[javi@MacBook-Pro actividadesPrimeraClaseDAW % git status]
On branch master
nothing to commit, working tree clean
[javi@MacBook-Pro actividadesPrimeraClaseDAW % git reset --hard HEAD~1]
HEAD is now at 51d045a hard-soft
[javi@MacBook-Pro actividadesPrimeraClaseDAW % git status]
On branch master
nothing to commit, working tree clean
javi@MacBook-Pro actividadesPrimeraClaseDAW %
```

Para deshacer el último commit y se quede en el área de stage debemos usar el comando **git reset --soft** pero, si lo que queremos es revertirlo completamente debemos usar el comando **git reset --hard**