

Introducción



Índice

- ¿Qué es GIT?
- Usando GIT
- Instalando GIT
- Configurando GIT
- Obteniendo ayuda en GIT
- Inicializar repositorio local
- Oh my zsh
- GIT Workflow
- Staging/Preparando ficheros
- Commit
- Eliminando ficheros
- Renombrando ficheros
- Ignorando ficheros
- Short status (status -s)
- Cambios staged/unstaged (diff)
- Herramientas para comparar ficheros y vscode
- Histórico de commits
- Visualización de commits
- *Unstaging*
- Descartando cambios locales

¿Qué es GIT?

- GIT es un *Sistema de Control de Versión*.
 - En inglés con las siglas de CVS (*Control Version System*).
 - Ampliamente utilizado en la actualidad.
 - Permite:

Registrar los cambios realizados al código fuente de un proyecto. Albergándolos en una base de datos especial, llamada **repositorio**. Revisar el **histórico** de cambios del proyecto, observando **quién** los hizo, además de **cuándo** y **por qué**, pudiéndose **revertir** a un estado anterior, si algo ha ido mal.

¿Qué es GIT?

- Sin un CVS:
 - Una nueva versión del código, supone copiar todo el directorio del código anterior a un nuevo directorio.

Mayor inconveniente:

Si existen varios programadores trabajando sobre la misma versión

- se necesita una mezcla manual de los cambios.
- la documentación de los cambios tendrá que realizarse mediante correo o documento aparte.

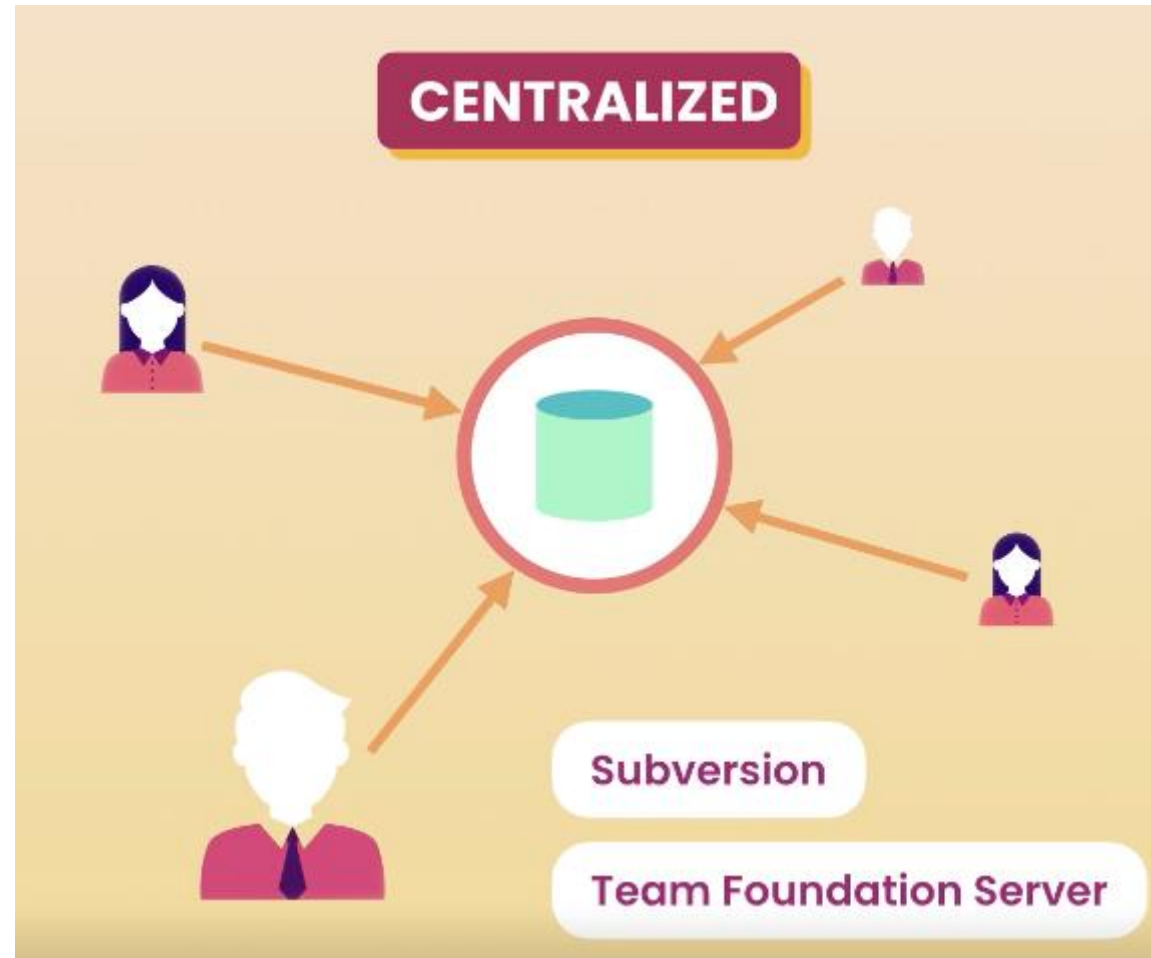
¿Qué es GIT?



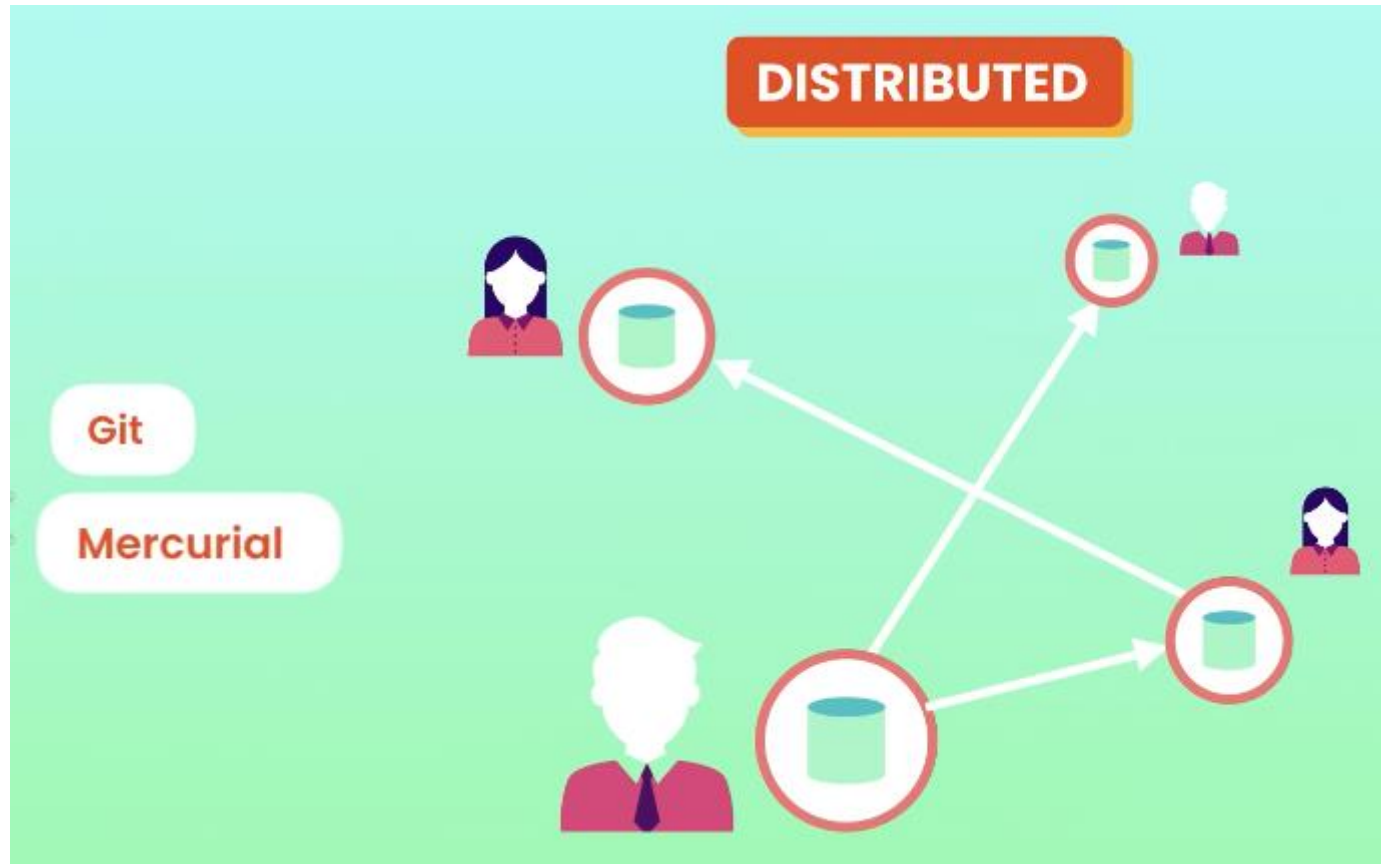
¿Qué es GIT?

- Existen 2 tipos de CVS:
 - Centralizados:
 - Todos los miembros del equipo se conectan a un servidor central para obtener la última copia del código y compartir sus cambios con los demás.
 - Ejemplos: Subversion (SVN) y Microsoft Team Foundation Server (TFS)
 - Problema principal:
 - Fallo del servidor central.
 - Los miembros del equipo no pueden obtener instantáneas ni colaborar hasta que se restablezca el servidor.
 - Distribuidos:
 - Cada miembro del equipo tiene una copia del proyecto con historial de cambios en su máquina. Se guardan instantáneas del proyecto localmente.
 - Si el servidor central falla, los miembros del equipo pueden sincronizar sus cambios directamente entre ellos.

¿Qué es GIT?



¿Qué es GIT?



¿Qué es GIT?

- **GIT es un CVS distribuido y el más utilizado.**
- Características:
 - Libre
 - OpenSource
 - Rápido
 - Escalable
 - Branching/Merging más sencillo.
- Operaciones de branching (ramas) y merging (mezclas) en otros CVS pueden ser muy lentas y dolorosas en comparación con GIT.

¿Qué es GIT?

- Fundamental hoy día para trabajar como programador.
 - En cualquier entrevista de trabajo te preguntarán por los CVS que conoces.
 - Seguir el historial de cambios del proyecto.
 - Trabajar colaborativamente con él con el resto de miembros del equipo.



Usando GIT

- Puedes usar GIT desde:
 - Línea de comandos
 - Editores de código IDEs
 - VSCode panel de GIT y extensiones como GITLens
<https://code.visualstudio.com/>
<https://marketplace.visualstudio.com/items?itemName=eamodio.gitlens>
- GUIs clients
 - SourceTree (Windows, Mac)
 - Github Desktop (Windows, Mac)
 - ... <https://git-scm.com/downloads/guis>
 - GITKraken (Windows, Mac, Linux) <https://www.gitkraken.com/>
 - También destacan:
 - GITKraken Boards (seguimiento de incidencias)
 - GITKraken Timelines (eje de tiempo con hitos de desarrollo)

Usando GIT

- ¿Por qué usar Git en la línea de comandos?
 - Los GUIs presentan limitaciones.
 - Sólo soporte para los comandos git más usados/típicos.
 - Uso en servidores sin entorno gráfico o permisos para lanzar un GUI.

Instalando GIT

- Ve a la página oficial de GIT

<https://git-scm.com/>

Se tiene la documentación oficial, así como los paquetes instaladores para Windows, Mac y Linux.

- Para Linux, instalaremos la última versión estable disponible en el repositorio ppa de Ubuntu de git-core

<https://git-scm.com/download/linux>

#add-apt-repository ppa:git-core/ppa

#apt-get update; apt-get install git

#git --version

```
user@user-VirtualBox:~$ sudo add-apt-repository ppa:git-core/ppa
The most current stable version of Git for Ubuntu.

For release candidates, go to https://launchpad.net/~git-core/+archive/candidate .
Más información: https://launchpad.net/~git-core/+archive/ubuntu/ppa
Pulse [ENTRAR] para continuar o Ctrl+C para cancelar la adición.

Des:1 http://ppa.launchpad.net/git-core/ppa/ubuntu focal InRelease [23,8 kB]
Obj:2 http://es.archive.ubuntu.com/ubuntu focal InRelease
Des:3 http://es.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Obj:4 http://packages.microsoft.com/repos/code stable InRelease
Des:5 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Des:6 http://es.archive.ubuntu.com/ubuntu focal-backports InRelease [101 kB]
Des:7 http://ppa.launchpad.net/git-core/ppa/ubuntu focal/main amd64 Packages [3.024 B]
Des:8 http://ppa.launchpad.net/git-core/ppa/ubuntu focal/main i386 Packages [3.020 B]
Des:9 http://ppa.launchpad.net/git-core/ppa/ubuntu focal/main Translation-en [2.252 B]
Des:10 http://es.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [1.205 kB]
Des:11 http://security.ubuntu.com/ubuntu focal-security/main amd64 DEP-11 Metadata [27,6 kB]
Des:12 http://es.archive.ubuntu.com/ubuntu focal-updates/main i386 Packages [532 kB]
Des:13 http://security.ubuntu.com/ubuntu focal-security/main amd64 c-n-f Metadata [8.688 B]
Des:14 http://security.ubuntu.com/ubuntu focal-security/universe amd64 DEP-11 Metadata [61,0 kB]
Des:15 http://security.ubuntu.com/ubuntu focal-security/universe amd64 c-n-f Metadata [12,4 kB]
Des:16 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 DEP-11 Metadata [2.468 B]
Des:17 http://es.archive.ubuntu.com/ubuntu focal-updates/main Translation-en [258 kB]
Des:18 http://es.archive.ubuntu.com/ubuntu focal-updates/main amd64 DEP-11 Metadata [282 kB]
Des:19 http://es.archive.ubuntu.com/ubuntu focal-updates/main amd64 c-n-f Metadata [14,2 kB]
Des:20 http://es.archive.ubuntu.com/ubuntu focal-updates/universe i386 Packages [632 kB]
Des:21 http://es.archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [855 kB]
Des:22 http://es.archive.ubuntu.com/ubuntu focal-updates/universe Translation-en [181 kB]
Des:23 http://es.archive.ubuntu.com/ubuntu focal-updates/universe amd64 DEP-11 Metadata [351 kB]
Des:24 http://es.archive.ubuntu.com/ubuntu focal-updates/universe amd64 c-n-f Metadata [18,9 kB]
Des:25 http://es.archive.ubuntu.com/ubuntu focal-updates/multiverse amd64 DEP-11 Metadata [944 B]
Des:26 http://es.archive.ubuntu.com/ubuntu focal-backports/universe amd64 DEP-11 Metadata [10,3 B]
Descargados 4.813 kB en 5s (949 kB/s)
Leyendo lista de paquetes... Hecho
```



```
user@user-VirtualBox:~$ sudo apt-get update
Obj:1 http://ppa.launchpad.net/git-core/ppa/ubuntu focal InRelease
Obj:2 http://es.archive.ubuntu.com/ubuntu focal InRelease
Obj:3 http://es.archive.ubuntu.com/ubuntu focal-updates InRelease
Obj:4 http://packages.microsoft.com/repos/code stable InRelease
Obj:5 http://es.archive.ubuntu.com/ubuntu focal-backports InRelease
Obj:6 http://security.ubuntu.com/ubuntu focal-security InRelease
Leyendo lista de paquetes... Hecho
```

```
user@user-VirtualBox:~$ sudo apt-get install git
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  git-man liberror-perl
Paquetes sugeridos:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki
Se instalarán los siguientes paquetes NUEVOS:
  git git-man liberror-perl
0 actualizados, 3 nuevos se instalarán, 0 para eliminar y 18 no actualizados.
Se necesita descargar 7.517 kB de archivos.
Se utilizarán 39,5 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] S
Des:1 http://ppa.launchpad.net/git-core/ppa/ubuntu focal/main amd64 git-man all 1:2.33.0-0ppa1~u
Des:2 http://es.archive.ubuntu.com/ubuntu focal/main amd64 liberror-perl all 0.17029-1 [26,5 kB]
Des:3 http://ppa.launchpad.net/git-core/ppa/ubuntu focal/main amd64 git amd64 1:2.33.0-0ppa1~ubu
Descargados 7.517 kB en 6s (1.333 kB/s)
Seleccionando el paquete liberror-perl previamente no seleccionado.
(Leyendo la base de datos ... 152202 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../liberror-perl_0.17029-1_all.deb ...
Desempaquetando liberror-perl (0.17029-1) ...
Seleccionando el paquete git-man previamente no seleccionado.
Preparando para desempaquetar .../git-man_1%3a2.33.0-0ppa1~ubuntu20.04.1_all.deb ...
Desempaquetando git-man (1:2.33.0-0ppa1~ubuntu20.04.1) ...
Seleccionando el paquete git previamente no seleccionado.
Preparando para desempaquetar .../git_1%3a2.33.0-0ppa1~ubuntu20.04.1_amd64.deb ...
Desempaquetando git (1:2.33.0-0ppa1~ubuntu20.04.1) ...
Configurando liberror-perl (0.17029-1) ...
Configurando git-man (1:2.33.0-0ppa1~ubuntu20.04.1) ...
Configurando git (1:2.33.0-0ppa1~ubuntu20.04.1) ...
Procesando disparadores para man-db (2.9.1-1) ...
```

Instalando GIT

- Para revisar la versión instalada emitimos el comando:
 - `git --version`

```
user@user-VirtualBox:~$ git --version
git version 2.33.0
```

- Recuerda que para eliminar un paquete instalado completamente junto con dependencias debes utilizar:

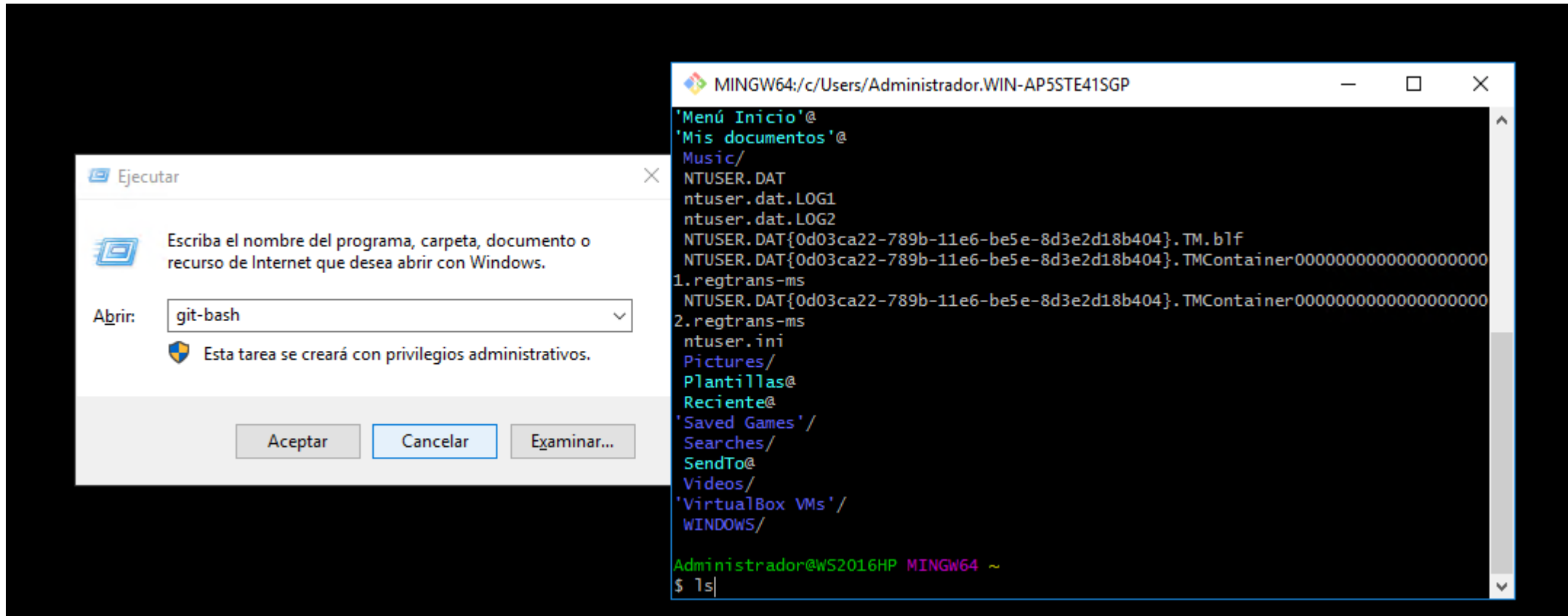
```
apt-get remove paquete # elimina el paquete
apt-get purge paquete # elimina los ficheros
de configuración del paquete en el sistema
apt-get autoremove # elimina dependencias no
en uso
apt-get clean vaciamos la caché de
repositorio local de paquetes
```

Para desinstalar git...

```
sudo dpkg --get-selections | grep git
sudo apt-get remove git
sudo apt-get purge git
sudo apt-get autoremove
sudo apt-get clean
```


Instalando GIT

- En Windows si quieres tener línea de comando git instala git-bash



Configurando GIT

- La primera vez que uses GIT tienes que especificar los siguientes *settings*:
 - Nombre
 - Email
 - Editor por defecto
 - Manejo de terminadores de línea.
- Existen 3 niveles de configuración de GIT en tu máquina:
 - System: afecta a todos los usuarios de la máquina.
 - Global: afecta a todos los repositorios del usuario actual.
 - Local: afecta al repositorio actual, o el repositorio en la carpeta actual. De modo que podemos tener diferentes configuraciones para diferentes repositorios en este nivel.

Configurando GIT

```
git config --global user.name "José M. Martín"
```

```
git config --global user.email jmartejXXX@g.educaand.es
```

Nota: reemplaza valores por tus datos.

Tras introducir nombre y correo para todos los repositorios del usuario actual, pasamos a configurar el editor por defecto.

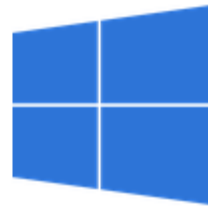
Vamos a utilizar VSCode y configurarlo como el editor por defecto:

<https://code.visualstudio.com/download>

```
git config --global core.editor "code --wait"
```

Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.



↓ Windows
Windows 7, 8, 10

User Installer	64 bit	32 bit	ARM
System Installer	64 bit	32 bit	ARM
.zip	64 bit	32 bit	ARM



↓ .deb
Debian, Ubuntu

↓ .rpm
Red Hat, Fedora, SUSE

.deb	64 bit	ARM	ARM 64
.rpm	64 bit	ARM	ARM 64
.tar.gz	64 bit	ARM	ARM 64

Snap Store

Configurando GIT

Para instalar VSCode desde consola:

```
dpkg -i ~/Descargas/code_1*.deb
```

```
#Si faltara alguna dependencia para dpkg, puedes  
arreglarlo mediante  
apt install -f
```

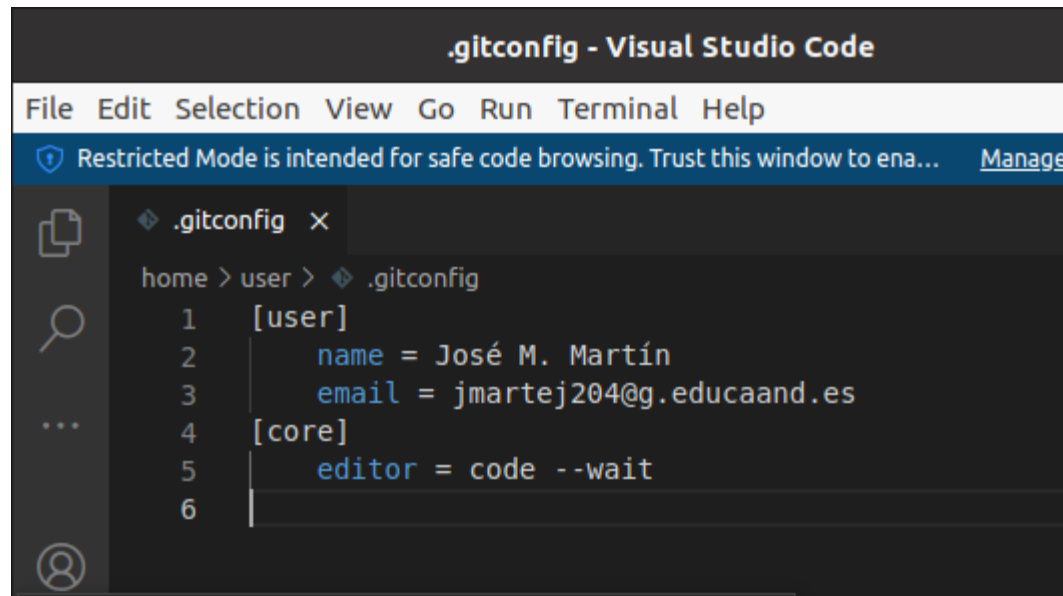
```
#Para probar VSCode puedes lanzarlo desde consola  
mediante:  
code
```

```
#Fíjate que se ha unido al path del sistema  
después de la instalación  
which code  
/usr/bin/code  
echo $PATH  
..:usr/bin:..
```

Configurando GIT

- Para probar la configuración de editor

```
git config --global -e
```



The screenshot shows the Visual Studio Code editor interface with the title bar ".gitconfig - Visual Studio Code". The menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. A blue status bar at the top indicates "Restricted Mode is intended for safe code browsing. Trust this window to ena... Manage". The left sidebar shows icons for Explorer, Search, and Source Control. The main editor area displays the content of the ".gitconfig" file, which is located at "home > user > .gitconfig". The file content is as follows:

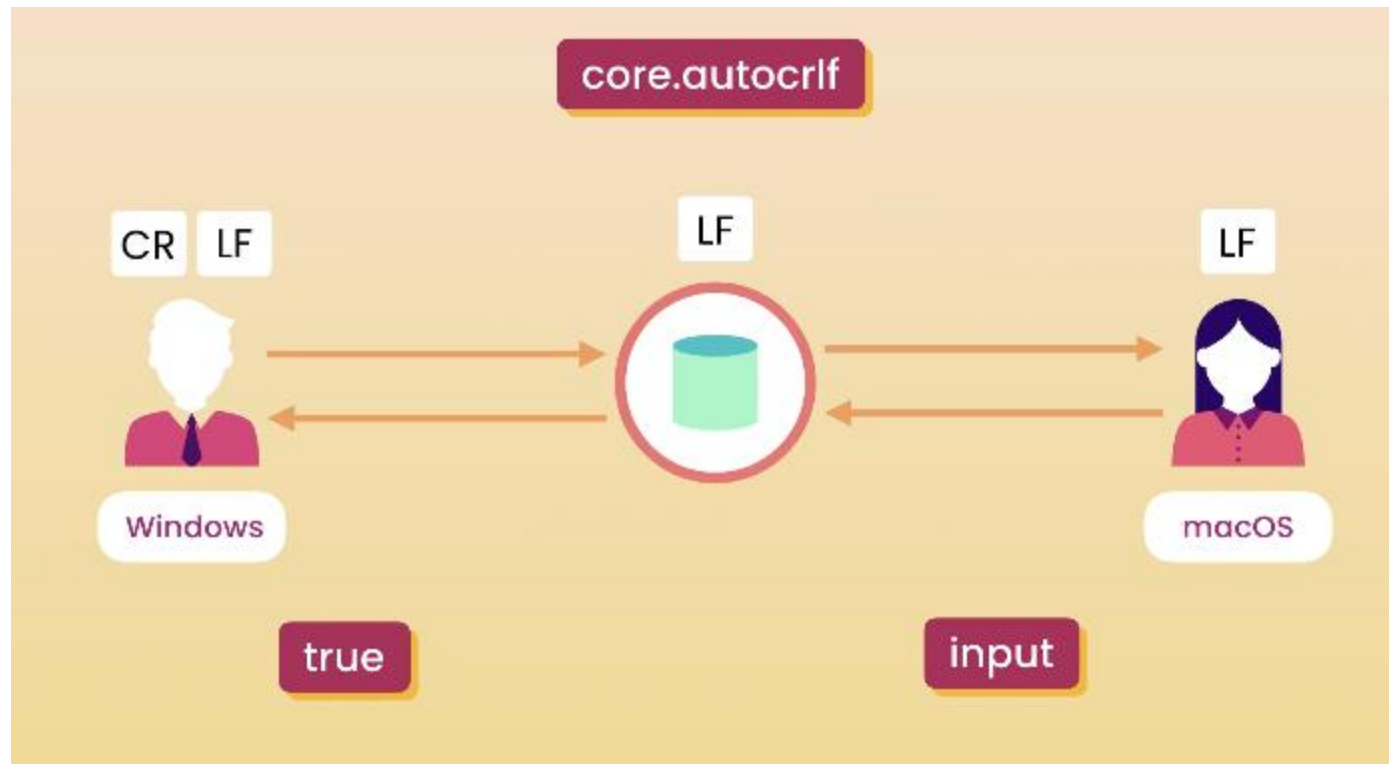
```
1 [user]
2   name = José M. Martín
3   email = jmartej204@g.educaand.es
4 [core]
5   editor = code --wait
6
```

Configurando GIT

- A continuación, vamos a configurar el manejo de los caracteres de final de línea:
 - En Windows el final de línea está formado por 2 caracteres: `\r\n`
 - `\r` retorno de carro, y `\n` salto de línea
 - En Mac/Linux el final de línea está formado por 1 solo carácter: `\n`



Configurando GIT



Configurando GIT

```
#En Windows
```

```
git config --global autocrlf true
```

```
#En Linux/Mac
```

```
git config --global autocrlf input
```

Obteniendo ayuda en GIT

```
#Para obtener ayuda sobre configuración  
git config --help
```

```
#Para obtener ayuda rápida sobre configuración:  
git config -h
```

Cliente de correo Thunderbird

and set repository or global options

SYNOPSIS

```
git config [<file-option>] [--type=<type>] [--fixed-value] [--show-origin] [--show-scope]
[-z|--null] name [value [value-pattern]]
git config [<file-option>] [--type=<type>] --add name value
git config [<file-option>] [--type=<type>] [--fixed-value] --replace-all name value [value-pattern]
git config [<file-option>] [--type=<type>] [--show-origin] [--show-scope] [-z|--null] [--fixed-value] --get name [value-pattern]
git config [<file-option>] [--type=<type>] [--show-origin] [--show-scope] [-z|--null] [--fixed-value] --get-all name [value-pattern]
git config [<file-option>] [--type=<type>] [--show-origin] [--show-scope] [-z|--null] [--fixed-value] [--name-only] --get-regexp name_regex [value-pattern]
git config [<file-option>] [--type=<type>] [-z|--null] --get-urlmatch name URL
git config [<file-option>] [--fixed-value] --unset name [value-pattern]
git config [<file-option>] [--fixed-value] --unset-all name [value-pattern]
git config [<file-option>] --rename-section old_name new_name
git config [<file-option>] --remove-section name
git config [<file-option>] [--show-origin] [--show-scope] [-z|--null] [--name-only] -l |
--list
git config [<file-option>] --get-color name [default]
git config [<file-option>] --get-colorbool name [stdout-is-tty]
git config [<file-option>] -e | --edit
```

DESCRIPTION

You can query/set/replace/unset options with this command. The name is actually the section and the key separated by a dot, and the value will be escaped.

```
user@ubudesk:~$ git config -h
```

```
uso: git config [<opciones>]
```

Ubicación del archivo de configuración

--global	usar archivo de config global
--system	usar archivo de config del sistema
--local	usar archivo de config del repositorio
--worktree	usar archivo de config del árbol de trabajo
-f, --file <archivo>	usar archivo de config especificado
--blob <blob-id>	leer config del objeto blob suministrado

Acción

--get	obtener valor: nombre [patrón de valor]
--get-all	obtener todos los valores: clave [patrón de valor]
--get-regexp	obtener valores para regexp: name-regex [value-pattern]
--get-urlmatch	obtener valor específico para el URL: sección[.var] URL
--replace-all	reemplazar todas las variables coincidentes: nombre valor [valor-patrón]

n]

--add	agregar nueva variable: nombre valor
--unset	eliminar una variable: nombre [patrón de valor]
--unset-all	eliminar todas las coincidencias: nombre [patrón de valor]
--rename-section	renombrar sección: nombre-viejo nombre-nuevo
--remove-section	borrar una sección: nombre
-l, --list	listar todo
--fixed-value	use la igualdad de cadenas al comparar valores con 'patrón de valor'
-e, --edit	abrir el editor
--get-color	encontrar el color configurado: slot [default]
--get-colorbool	encontrar las opciones del color: slot [stdout-es-tty]

Inicializar repositorio local

#Primero creamos un directorio de trabajo, desde línea de comandos:

```
cd
```

```
mkdir proyecto0
```

```
cd proyecto0
```

#A continuación, inicializamos un repositorio vacío en este directorio

```
git init
```

```
ls -a
```

```
nautilus .&
```

```
user@ubudesk:~$ mkdir proyecto0
```

```
user@ubudesk:~$ cd proyecto0/
```

```
Terminal user@ubudesk:~/proyecto0$ git init
```

```
ayuda: usando 'master' como el nombre de la rama inicial. Este nombre de rama predeterminado
```

```
ayuda: está sujeto a cambios. Para configurar el nombre de la rama inicial para usar en todos
```

```
ayuda: de sus nuevos repositorios, reprimiendo esta advertencia, llame a:
```

```
ayuda:
```

```
ayuda: git config --global init.defaultBranch <nombre>
```

```
ayuda:
```

```
ayuda: Los nombres comúnmente elegidos en lugar de 'master' son 'main', 'trunk' y
```

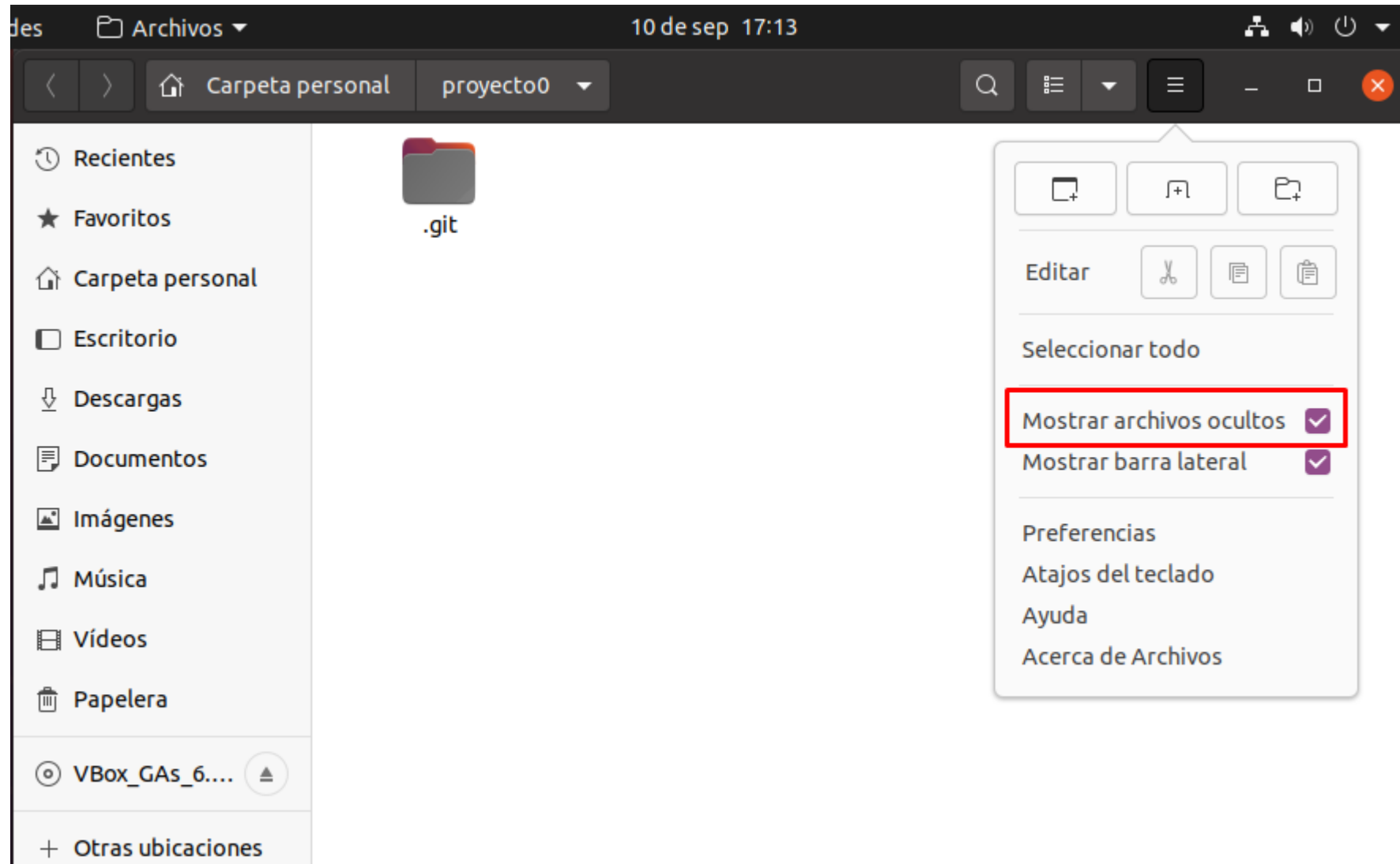
```
ayuda: 'development'. Se puede cambiar el nombre de la rama recién creada mediante este comando:
```

```
ayuda:
```

```
ayuda: git branch -m <nombre>
```

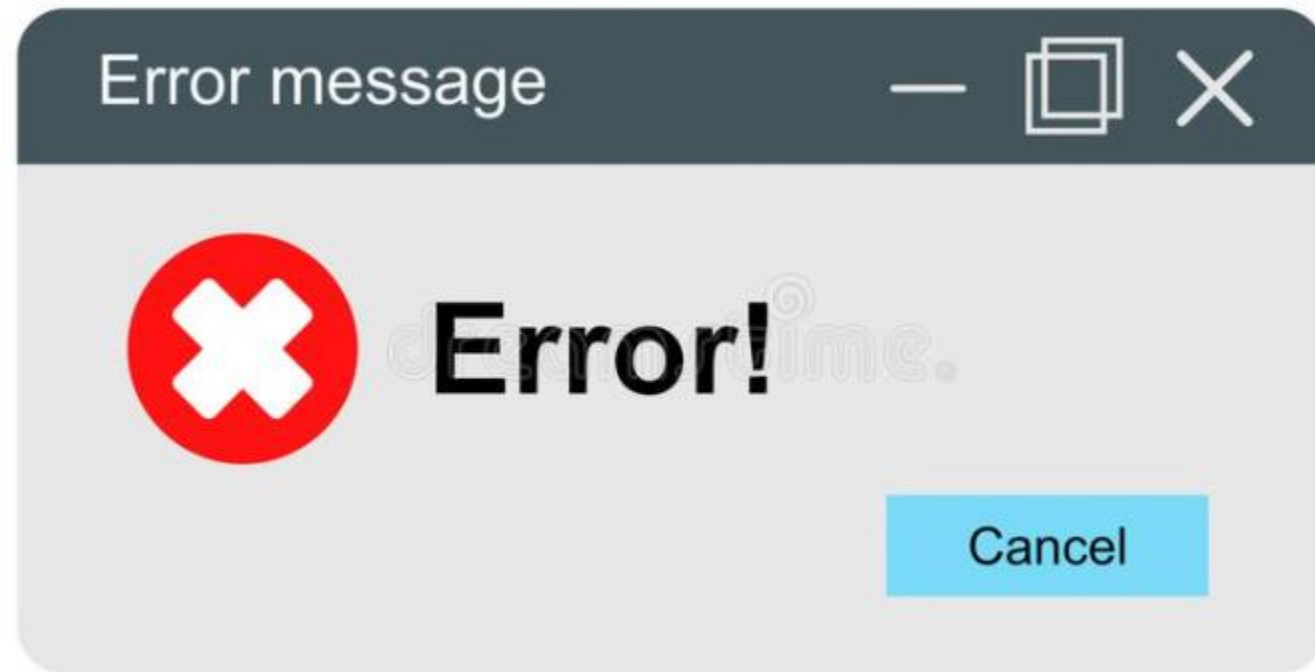
```
Inicializado repositorio Git vacío en /home/user/proyecto0/.git/
```

```
user@ubudesk:~/proyecto0$ ls -a
.  ..  .git
user@ubudesk:~/proyecto0$ nautilus .&
[1] 6332
```



Inicializar repositorio local

- El directorio oculto .git va a contener al repositorio físicamente.
- NOTA IMPORTANTE:
 - NO TOCAR DENTRO NI BORRAR EL DIRECTORIO OCULTO.git











- Recientes
- Favoritos
- Carpeta personal
- Escritorio
- Descargas
- Documentos
- Imágenes
- Música
- Videos
- Papelera

⦿ VBox_GAs_6.... ▲

+ Otras ubicaciones

Nombre

	branches
	hooks
	info
	objects
	refs
	config
	description
	HEAD

```
user@ubudesk:~/proyecto0$ git status
```

```
En la rama master
```

```
No hay commits todavía
```

```
no hay nada para confirmar (crea/copia archivos y usa "git add" para hacerles seguimiento)
```

```
[1]+  Hecho          nautilus .
```

```
user@ubudesk:~/proyecto0$ rm -rf .git
```

```
user@ubudesk:~/proyecto0$ git status
```

```
fatal: no es un repositorio git (ni ninguno de los directorios superiores): .git
```

Oh my zsh

- zsh shell alternativo a bash con plugins especial para uso con git



```
apt install zsh
#Para lanzar el nuevo terminal
zsh
```

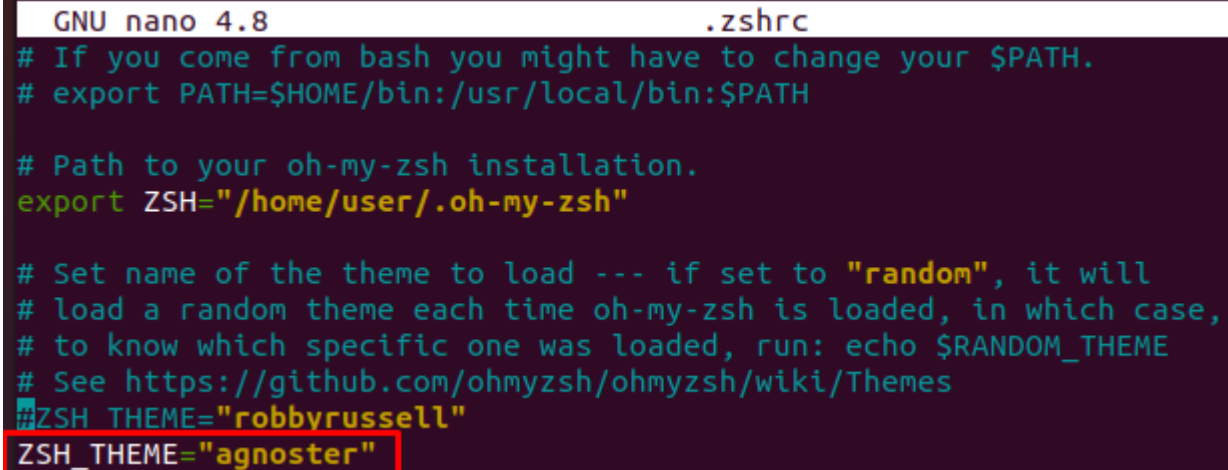


```
sh -c "$(wget -O-  
https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"
```

```
#Nota si cambias a zsh permanentemente y más tarde quieres volver a bash  
chsh -s /bin/bash
```

Oh my zsh

```
#Cambiando al famoso tema agnoster  
cd  
apt-get install fonts-powerline  
nano .zshrc
```

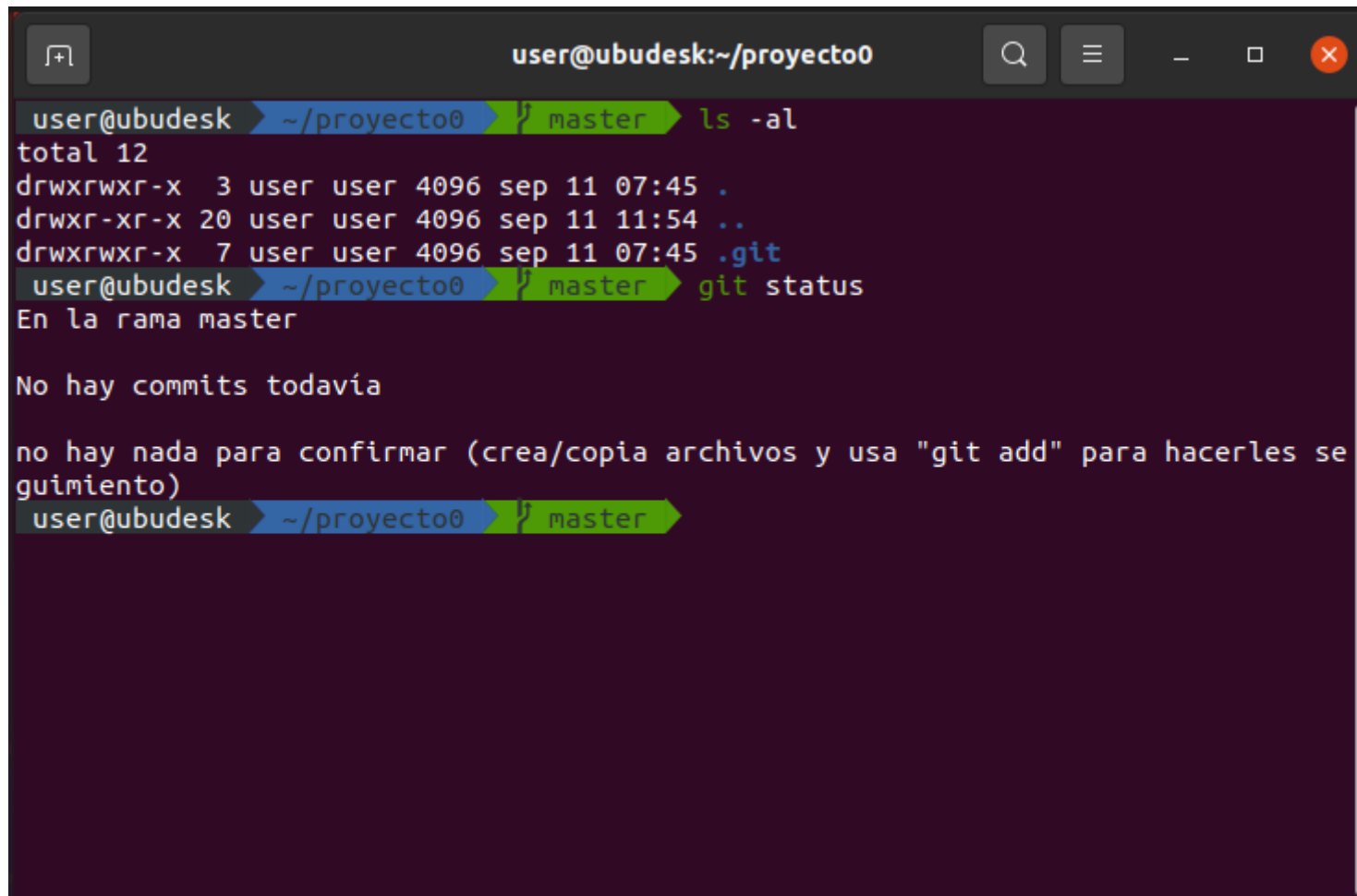


```
GNU nano 4.8 .zshrc  
# If you come from bash you might have to change your $PATH.  
# export PATH=$HOME/bin:/usr/local/bin:$PATH  
  
# Path to your oh-my-zsh installation.  
export ZSH="/home/user/.oh-my-zsh"  
  
# Set name of the theme to load --- if set to "random", it will  
# load a random theme each time oh-my-zsh is loaded, in which case,  
# to know which specific one was loaded, run: echo $RANDOM_THEME  
# See https://github.com/ohmyzsh/ohmyzsh/wiki/Themes  
ZSH_THEME="robyrussell"  
ZSH_THEME="agnoster"
```

Oh my zsh

```
#Instalando un plugin de realzado de sintaxis de zsh-users
cd
git clone https://github.com/zsh-users/zsh-syntax-
highlighting.git
echo "source ${(q-)PWD}/zsh-syntax-highlighting/zsh-syntax-
highlighting.zsh" >> ${ZDOTDIR:-$HOME}/.zshrc
#Observa que la línea source está al final de .zshrc
cat .zshrc
#Actívalo en la sesión de Shell actual
source ./zsh-syntax-highlighting/zsh-syntax-highlighting.zsh
```

Oh my zsh



```
user@ubudesk:~/proyecto0
user@ubudesk > ~/proyecto0 > master > ls -al
total 12
drwxrwxr-x  3 user user 4096 sep 11 07:45 .
drwxr-xr-x 20 user user 4096 sep 11 11:54 ..
drwxrwxr-x  7 user user 4096 sep 11 07:45 .git
user@ubudesk > ~/proyecto0 > master > git status
En la rama master

No hay commits todavía

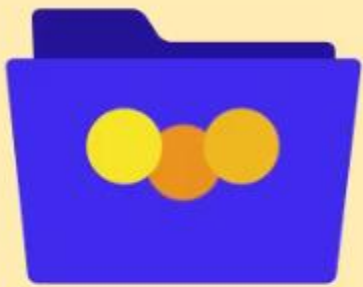
no hay nada para confirmar (crea/copia archivos y usa "git add" para hacerles seguimiento)
user@ubudesk > ~/proyecto0 > master >
```

GIT Workflow

- Veamos ahora el flujo de trabajo con GIT
- Después de inicializar el repositorio se tiene:
 - Directorio de trabajo del proyecto
 - Repositorio del proyecto, es decir, el directorio oculto .git
 - Recuerda NO TOCAR, NO BORRAR .git
 - Además, se tendrá una zona intermedia entre ambos conocida como el área de preparación, *staging área o index*.

WORKFLOW

Staging Area



GIT Workflow

- A medida que se van preparando los ficheros en el directorio de proyecto, éstos pasarán al área de preparación (staging área o index), convirtiéndose en candidatos a instantánea (snapshot) para el repositorio mediante el comando:

```
git add file
```

El área de preparación nos va a permitir revisar los ficheros candidatos a instantánea (snapshot) en el repositorio. De esta forma, si, posteriormente, alguno de los ficheros añadidos no debe entrar en la instantánea basta con retirarlo del área.

La idea del área de preparación es que contenga el fuente que deba ir a un pase de producción.

WORKFLOW



```
git add file1 file2
```

GIT Workflow

- Para tomar una instantánea (snapshot) de los ficheros en el área de preparación (staging area) se emplea el comando git de commit:

```
git commit -m "Commit inicial"
```

El modificador -m permite introducir un mensaje para identificación posterior de la instantánea en el histórico del repositorio.

- Después de este commit inicial ya se tendrá la primera instantánea del fuente del proyecto en el repositorio almacenada.

WORKFLOW



```
git commit -m "Initial commit."
```

GIT Workflow

- **IMPORTANTE:**

Después de realizar un commit el área de preparación (staging área) no se vacía.

Esta área siempre contendrá el fuente (los ficheros) de la instantánea a incorporar en el repositorio.

Esta área es como una especie de “área de preproducción” y cuando se realiza un commit sería como el paso a producción de ese fuente.

GIT Workflow

- Si después del commit inicial se realiza un cambio en un fichero, éste no se refleja automáticamente en el área de preparación (staging area).



GIT Workflow

- Para trasladar el cambio al área de preparación tienes que volver a emitir el comando add

```
git add file
```



Tras realizar de nuevo `git add file1` el index/staging area se tiene el mismo fuente actualizado que en el proyecto.

GIT Workflow

- Tras añadir los cambios al index ahora podrías tomar una instantánea del fuente en el área de preparación con el comando commit:

```
git commit -m "Se arregla bug tal y tal..."
```



Tras el commit el repositorio contiene una segunda instantánea con los cambios introducidos.

Es importante introducir un mensaje descriptivo de los cambios que se han implementado.

GIT Workflow

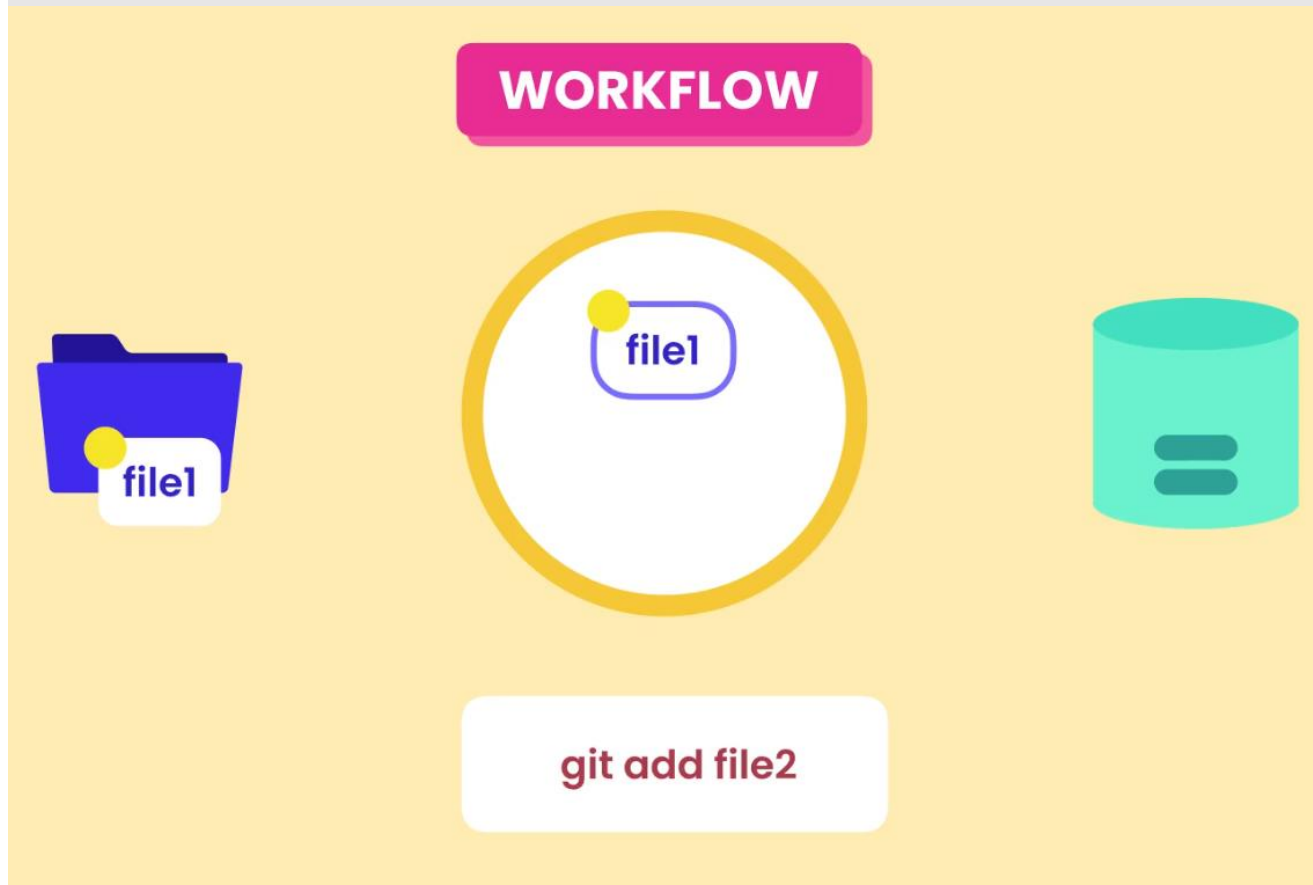
- De igual forma, si se elimina un fichero del directorio del proyecto o trabajo, este se mantendrá en el área de preparación.



GIT Workflow

Si ejecutas de nuevo el comando add para el fichero eliminado entonces se eliminará del área de preparación.

```
git add file2
```



Aunque el comando parezca que está añadiendo el fichero, si actualmente se borró la acción que realiza es la de eliminarlo del área de preparación.

GIT Workflow

Si ahora realizas un commit, tendrás el tercer *snapshot* en el repositorio con la imagen de lo que estaba en el área de preparación en ese instante.



GIT Workflow

- Ahora se tienen 3 *snapshots/commits* en el repositorio.
- La información que se guarda de cada *commit* es la siguiente:
 - ID, generado automáticamente por GIT (se trata de un código Hash del snap).
 - Mensaje
 - Autor
 - Instantánea (snapshot) del fuente.
 - A diferencia de otros VCS GIT no almacena DELTAS o lo que cambia del fuente
 - GIT almacena todo el fuente en *stale* en la *snapshot* de forma eficiente:
 - Comprime todo el contenido (ficheros).
 - No guarda duplicados de contenido.

Staging/Preparando ficheros

```
user@ubudesk:~/proyecto0
user@ubudesk ~/proyecto0 master echo hola, mundo > file1.txt
user@ubudesk ~/proyecto0 master cat file1.txt
hola, mundo
user@ubudesk ~/proyecto0 master echo hola, mundo2 > file2.txt
user@ubudesk ~/proyecto0 master cat file2.txt
hola, mundo2
user@ubudesk ~/proyecto0 master git status
En la rama master

No hay commits todavía

Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
    file1.txt
    file2.txt

no hay nada agregado al commit pero hay archivos sin seguimiento presentes (usa
"git add" para hacerles seguimiento)
user@ubudesk ~/proyecto0 master
```

Creamos 2 ficheros de texto plano con contenido y verificamos que ambos se encuentran sin seguimiento, es decir, que no han pasado al área de preparación.

Staging/Preparando ficheros

#Para añadir ficheros se pueden usar varias opciones, como añadirlos #de forma individual:

```
git add file1.txt
```

```
git add file2.txt
```

#En una línea

```
git add file1.txt file2.txt
```

#Utilizar globbing de línea de comandos

```
git add *.txt
```

#O añadir al stage todos los ficheros y directorios del proyecto de forma recursiva

```
git add .
```

```
user@ubudesk:~/proyecto0
user@ubudesk:~/proyecto0$ zsh
[oh-my-zsh] plugin 'zsh-syntax-highlighting' not found
user@ubudesk:~/proyecto0 } master git status
En la rama master

No hay commits todavía

Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
    file1.txt
    file2.txt

no hay nada agregado al commit pero hay archivos sin seguimiento presentes (usa "git add" para h
acerles seguimiento)
user@ubudesk:~/proyecto0 } master ls
file1.txt file2.txt
user@ubudesk:~/proyecto0 } master git add .
user@ubudesk:~/proyecto0 } master + git status
En la rama master

No hay commits todavía

Cambios a ser confirmados:
  (usa "git rm --cached <archivo>..." para sacar del área de stage)
    nuevos archivos: file1.txt
    nuevos archivos: file2.txt

user@ubudesk:~/proyecto0 } master +
```

Staging/Preparando ficheros

```
#Ahora vamos a introducir una modificación  
#sobre uno de los ficheros ya en  
#seguimiento en el área de preparación  
(stage)
```

```
echo adios >> file1.txt
```

```
#Revisamos el estado mediante el #comando:
```

```
git status
```

file1.txt está en el stage con la versión anterior del fichero, la nueva versión se destaca con el mensaje de modificado.

```
user@ubudesk ~/proyecto0 master +- echo adios >> file1.txt  
user@ubudesk ~/proyecto0 master +- cat file1.txt  
hola, mundo  
adios  
user@ubudesk ~/proyecto0 master +- git status  
En la rama master  
  
No hay commits todavía  
  
Cambios a ser confirmados:  
  (usa "git rm --cached <archivo>..." para sacar del área de stage)  
    nuevos archivos: file1.txt  
    nuevos archivos: file2.txt  
  
Cambios no rastreados para el commit:  
  (usa "git add <archivo>..." para actualizar lo que será confirmado)  
  (usa "git restore <archivo>..." para descartar los cambios en el directorio de trabajo)  
    modificados:    file1.txt
```


Staging/Preparando ficheros



Staging/Preparando ficheros

```
#Si añadimos al stage el nuevo fichero, ya tendremos la segunda
#versión en él
git add file1.txt
git status
```

```
user@ubudesk ~/proyecto0 } master ±+ git add file1.txt
user@ubudesk ~/proyecto0 } master + git status
En la rama master

No hay commits todavía

Cambios a ser confirmados:
  (usa "git rm --cached <archivo>..." para sacar del área de stage)
    nuevos archivos: file1.txt
    nuevos archivos: file2.txt
```

- Una opción para inspeccionar los el *stage* puede ser con el modo verbose de status:

git status -v

- Si quieres ver las diferencias:

git diff

```

user@ubudesk ~/proyecto0 } master + echo adios >> file1.txt
user@ubudesk ~/proyecto0 } master ±+ git status -v
En la rama master

No hay commits todavía

Cambios a ser confirmados:
  (usa "git rm --cached <archivo>..." para sacar del área de stage)
nuevos archivos: file1.txt
nuevos archivos: file2.txt

Cambios no rastreados para el commit:
  (usa "git add <archivo>..." para actualizar lo que será confirmado)
  (usa "git restore <archivo>..." para descartar los cambios en el directorio de trabajo)
modificados:      file1.txt

diff --git a/file1.txt b/file1.txt
new file mode 100644
index 0000000..c562e5a
--- /dev/null
+++ b/file1.txt
@@ -0,0 +1,2 @@
+hola, mundo
+adios
diff --git a/file2.txt b/file2.txt
new file mode 100644
index 0000000..e3ea11f
--- /dev/null
+++ b/file2.txt
@@ -0,0 +1 @@
+hola, mundo2
user@ubudesk ~/proyecto0 } master ±+ cat file1.txt
hola, mundo
adios
adios
user@ubudesk ~/proyecto0 } master ±+

```

Commit

- Para crear un *snapshot* de lo que tenemos en el *stage* utilizamos:

```
git commit -m "Mensaje corto describiendo la instantánea.."
```

- Si quieres introducir un mensaje largo:

```
git commit
#Y se abrirá el editor
por defecto, VSCode en
nuestro caso.
```

```
user@ubudesk ~/proyecto0 } master ±+ git status
En la rama master

No hay commits todavía

Cambios a ser confirmados:
(usa "git rm --cached <archivo>..." para sacar del área de stage)
nuevos archivos: file1.txt
nuevos archivos: file2.txt

Cambios no rastreados para el commit:
(usa "git add <archivo>..." para actualizar lo que será confirmado)
(usa "git restore <archivo>..." para descartar los cambios en el directorio de trabajo)
modificados: file1.txt

user@ubudesk ~/proyecto0 } master ±+ git add .
user@ubudesk ~/proyecto0 } master + git commit
[master (commit-raíz) 8c5cf68] Cabecera del mensaje
2 files changed, 4 insertions(+)
create mode 100644 file1.txt
create mode 100644 file2.txt
user@ubudesk ~/proyecto0 } master
```

```
user@ubudesk > ~/proyecto0 > } master ±+ git status
En la rama master

No hay commits todavía

Cambios a ser confirmados:
(usa "git rm --cached <archivo>..." para sacar del área de stage)
nuevos archivos: file1.txt
nuevos archivos: file2.txt

Cambios no rastreados para el commit:
(usa "git add <archivo>..." para actualizar lo que será confirmado)
(usa "git restore <archivo>..." para descartar los cambios en el directorio de trabajo)
modificados: file1.txt

user@ubudesk > ~/proyecto0 > } master ±+ git add .
user@ubudesk > ~/proyecto0 > } master + git commit
[master (commit-raíz) 8c5cf68] Cabecera del mensaje
2 files changed, 4 insertions(+)
create mode 100644 file1.txt
create mode 100644 file2.txt
user@ubudesk > ~/proyecto0 > } master
```

El plugin de git para zsh indica información en la barra del prompt:



Cambios *unstage*, existen cambios que no se han añadido al stage



Cambios en *stage* pendientes de añadir al repositorio mediante *commit*



Directorio y rama *master* actual sin cambios pendientes en directorio ni *stage*.

Commit

- Recomendaciones para realizar:
 - Realiza varios commits a lo largo del día a medida que avanzas en el desarrollo.
 - No esperes a llevar una semana de desarrollo para realizar commit.
 - Cada commit debe representar un conjunto de cambios lógicos relacionados con una determinada función o fase de una función.
 - No mezcles diferentes cambios lógicos del código en un commit, por ejemplo:
 - Corrección de un bug + nueva función.
 - Introduce mensajes descriptivos referentes al commit.

Commit

- Commit directo sin pasar por el *stage*.
 - *No recomendable.*

#La opción -a es por todos los ficheros modificados en el directorio. -m para introducir el mensaje en línea.

```
git commit -am "Mensaje de commit directo a todos los ficheros"
```

```
o0 Público snap Videos zsh-syntax-highlighting
user@ubudesk ~ ➔ cd proyecto0
user@ubudesk ~/proyecto0 ➔ master ➔ ls
file1.txt file2.txt
user@ubudesk ~/proyecto0 ➔ master ➔ git status
En la rama master
nada para hacer commit, el árbol de trabajo está limpio
user@ubudesk ~/proyecto0 ➔ master ➔ echo "Commit directo" >> file1.txt
user@ubudesk ~/proyecto0 ➔ master ± ➔ git commit -am "Esto es un commit directo, nada
recomendable.. mejor siempre pasar por el stage."
[master de3f4c3] Esto es un commit directo, nada recomendable.. mejor siempre pasar por el
stage.
1 file changed, 1 insertion(+)
user@ubudesk ~/proyecto0 ➔ master ➔ git status
En la rama master
nada para hacer commit, el árbol de trabajo está limpio
user@ubudesk ~/proyecto0 ➔ master ➔
```

Eliminando ficheros

- Desde el directorio de trabajo eliminamos un fichero con:

```
#Eliminamos el fichero  
rm file2.txt  
git status
```

```
#El fichero sigue en el stage  
git ls-files
```

```
#Para eliminarlo del stage tienes que hacer el add del fichero  
git add file2.txt
```


Eliminando ficheros

```
user@ubudesk ~/proyecto0 master ls
file1.txt file2.txt
user@ubudesk ~/proyecto0 master rm file2.txt
user@ubudesk ~/proyecto0 master ± git status
En la rama master
Cambios no rastreados para el commit:
  (usa "git add/rm <archivo>..." para actualizar a lo que se le va a hacer commit)
  (usa "git restore <archivo>..." para descartar los cambios en el directorio de trabajo)
    borrados:      file2.txt

sin cambios agregados al commit (usa "git add" y/o "git commit -a")
user@ubudesk ~/proyecto0 master ± git ls-files
file1.txt
file2.txt
user@ubudesk ~/proyecto0 master ± git add file2.txt
user@ubudesk ~/proyecto0 master + git ls-files
file1.txt
user@ubudesk ~/proyecto0 master + git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    borrados:      file2.txt
```

Eliminando ficheros

- Puedes realizar la operación de eliminar del directorio de trabajo a la vez que del *stage* mediante un solo comando:

```
git rm file2.txt
```

#La línea anterior equivale a las 2 líneas siguientes:

```
rm file2.txt
```

```
git add file2.txt
```

#Igual que con add puedes utilizar listas de ficheros así como globbing (*, ?,)

```
git rm fich1 fich2 *.cpp
```

#Nota si sólo quieres eliminar un fichero del directorio de trabajo y mantenerlo
#en el stage ejecuta solo:

```
rm fich
```

Renombrando ficheros

- Si renombas un fichero en el directorio de trabajo del proyecto que esté en seguimiento, tendrás que realizar los siguientes pasos:

```
#Para renombrar el fichero en el Shell de linux  
mv file1.txt renamedFile1.js
```

```
#Seguidamente compruebas el estado del repo  
git status
```

```
#Controlas el fichero que se ha eliminado file1.txt  
git add file1.txt
```

```
#Controlas el nuevo fichero renombrado  
git add renamedFile1.js
```

Renombrando ficheros

```
user@ubudesk:~$ cd proyecto0
user@ubudesk:~/proyecto0$ ls
file1.txt
user@ubudesk:~/proyecto0$ mv file1.txt renamedFile1.js
user@ubudesk:~/proyecto0$ ls
renamedFile1.js
user@ubudesk:~/proyecto0$ git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    borrados:      file2.txt

Cambios no rastreados para el commit:
  (usa "git add/rm <archivo>..." para actualizar a lo que se le va a hacer commit)
  (usa "git restore <archivo>..." para descartar los cambios en el directorio de trabajo)
    borrados:      file1.txt

Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
    renamedFile1.js

user@ubudesk:~/proyecto0$
```

Renombrando ficheros

```
user@ubudesk:~/proyecto0$ git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    borrados:      file2.txt

Cambios no rastreados para el commit:
  (usa "git add/rm <archivo>..." para actualizar a lo que se le va a hacer commit)
  (usa "git restore <archivo>..." para descartar los cambios en el directorio de trabajo)
    borrados:      file1.txt

Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
    renamedFile1.js

user@ubudesk:~/proyecto0$ git add file1.txt
user@ubudesk:~/proyecto0$ git add file renamedFile1.js
fatal: ruta especificada 'file' no concordó con ningún archivo
user@ubudesk:~/proyecto0$ git add renamedFile1.js
user@ubudesk:~/proyecto0$ git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    borrados:      file2.txt
    renombrados:   file1.txt -> renamedFile1.js
```

Renombrando ficheros

- Renombrando ficheros con una sola línea:

```
git mv renamedFile1.js file1.js
```

```
user@ubudesk:~/proyecto0$ git add file1.txt
user@ubudesk:~/proyecto0$ git add file renamedFile1.js
fatal: ruta especificada 'file' no concordó con ningún archivo
user@ubudesk:~/proyecto0$ git add renamedFile1.js
user@ubudesk:~/proyecto0$ git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    borrados:      file2.txt
    renombrados:    file1.txt -> renamedFile1.js

user@ubudesk:~/proyecto0$ git mv renamedFile1.js file1.js
user@ubudesk:~/proyecto0$ git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    renombrados:    file1.txt -> file1.js
    borrados:      file2.txt
```

Ignorando ficheros

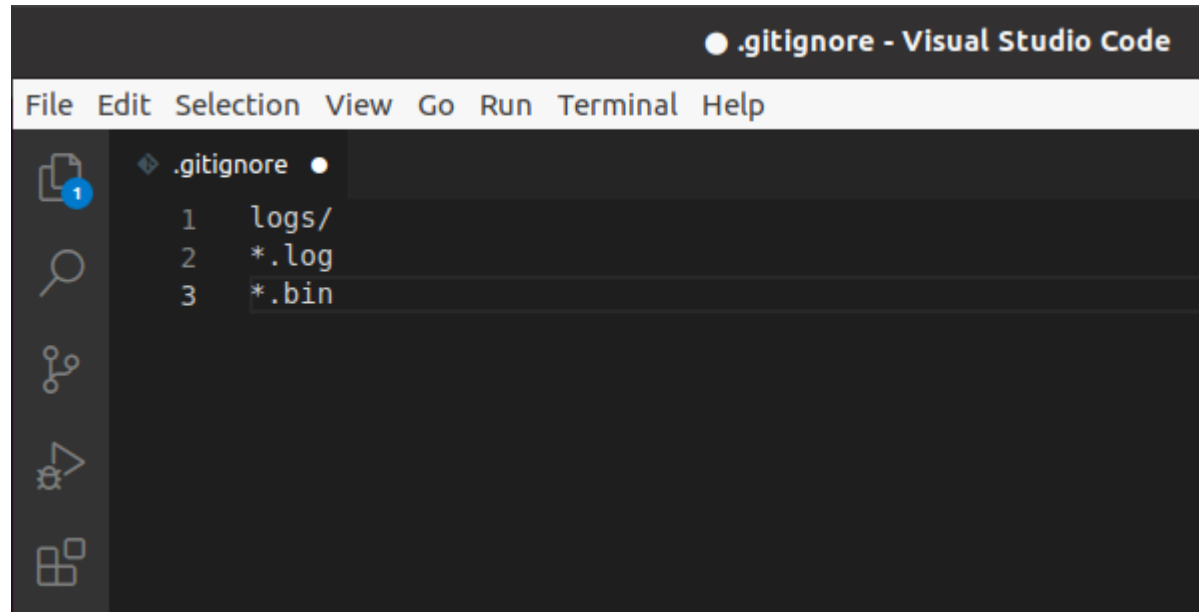
- Ficheros o directorios del proyecto que no queremos bajo seguimiento del repositorio.
- Ficheros binarios, logs, sistema,...
 - Plantillas de ficheros .gitignore en <https://github.com/github/gitignore>
 - Ejemplo queremos ignorar todo el contenido de un directorio de logs. Para ello, creamos en el raíz del proyecto un fichero *.gitignore* con code.

```
user@ubudesk:~/proyecto0$ ls
file1.js  logs
user@ubudesk:~/proyecto0$ git status
En la rama master
Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
    logs/

no hay nada agregado al commit pero hay archivos sin seguimiento presentes (usa "git add" para hacerlos seguimiento)
user@ubudesk:~/proyecto0$ code .gitignore
user@ubudesk:~/proyecto0$
```

Ignorando ficheros

- En el editor introducimos por línea los directorios o ficheros que queremos impedir que se sigan.



The screenshot shows the Visual Studio Code editor interface. The title bar at the top reads ".gitignore - Visual Studio Code". Below the title bar is a menu bar with the following items: File, Edit, Selection, View, Go, Run, Terminal, and Help. On the left side, there is a sidebar with icons for Explorer, Search, Source Control, Run and Debug, and Extensions. The Explorer icon is highlighted with a blue circle and the number 1. The main editor area displays the content of the .gitignore file, which consists of three lines:

```
1 logs/  
2 *.log  
3 *.bin
```


Ignorando ficheros

- Basta con añadir el fichero *.gitignore* con los directorios y ficheros que queremos ignorar para que desaparezcan del seguimiento propuesto por el repositorio.

```
user@ubudesk:~/proyecto0$ echo hola > logs/hola.log
user@ubudesk:~/proyecto0$ git status
En la rama master
Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
    logs/

no hay nada agregado al commit pero hay archivos sin seguimiento presentes (usa "git add" para hacerlos seguimiento)
user@ubudesk:~/proyecto0$ git add .gitignore
user@ubudesk:~/proyecto0$ git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    nuevos archivos: .gitignore
```

Fíjate después del add de .gitignore ya no se sigue al directorio logs/

Ignorando ficheros

```
user@ubudesk:~/proyecto0$ echo hola > logs/hola.log
user@ubudesk:~/proyecto0$ git status
En la rama master
Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
    logs/

no hay nada agregado al commit pero hay archivos sin seguimiento presentes (usa "git add" para hacerl
es seguimiento)
user@ubudesk:~/proyecto0$ git add .gitignore
user@ubudesk:~/proyecto0$ git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    nuevos archivos: .gitignore

user@ubudesk:~/proyecto0$ git commit -m "Añado .gitignore al proyecto"
[master a61582d] Añado .gitignore al proyecto
 1 file changed, 3 insertions(+)
 create mode 100644 .gitignore
user@ubudesk:~/proyecto0$ git status
En la rama master
nada para hacer commit, el árbol de trabajo está limpio
user@ubudesk:~/proyecto0$
```

NOTA IMPORTANTE:

Si un fichero o directorio ya se encuentra en seguimiento por git, y, posteriormente, se añade o actualiza .gitignore para evitar su seguimiento. Esta acción no tendrá efecto. Seguirá el seguimiento. Tendrá que eliminarse del repositorio para que sea efectivo el cambio en .gitignore.

Short status (status -s)

- Versión corta del comando status

```
git status -s
```

```
user@ubudesk:~/proyecto0$ echo hola >> file1.js
user@ubudesk:~/proyecto0$ echo hola > file2.js
user@ubudesk:~/proyecto0$ git status -s
 M file1.js
?? file2.js
user@ubudesk:~/proyecto0$
```

M de color rojo a la derecha (2ª posición de línea) significa fichero bajo seguimiento modificado en el directorio de trabajo.

```
user@ubudesk:~/proyecto0$ git status -s
 M file1.js
?? file2.js
user@ubudesk:~/proyecto0$ git add .
user@ubudesk:~/proyecto0$ git status -s
 M file1.js
A file2.js
user@ubudesk:~/proyecto0$
```

M de color verde a la izquierda (1ª posición de línea) significa fichero bajo seguimiento modificado en el directorio de trabajo.

A de color verde a la izquierda (1ª posición de línea) significa fichero nuevo añadido al *stage* desde el directorio de trabajo.

Short status (status -s)

- Después de hacer commit no hay nada pendiente en el directorio de trabajo.

```
user@ubudesk:~/proyecto0$ git status -s
M file1.js
?? file2.js
user@ubudesk:~/proyecto0$ git add .
user@ubudesk:~/proyecto0$ git status -s
M file1.js
A file2.js
user@ubudesk:~/proyecto0$ git commit -m "Modifico file1.js y añado file2.js"
[master 5c9304d] Modifico file1.js y añado file2.js
 2 files changed, 2 insertions(+)
 create mode 100644 file2.js
user@ubudesk:~/proyecto0$ git status -s
user@ubudesk:~/proyecto0$
```

Cambios staged/unstaged (diff)

- Diff de cambios en *stage* a último cambio *commit* en el repositorio local mediante el comando:

```
git diff --staged
```

- Diff de cambios en *directorio de trabajo* a *stage* mediante el comando:

```
git diff
```

```
user@ubudesk: ~/proyecto0
user@ubudesk:~/proyecto0$ git status -s
user@ubudesk:~/proyecto0$ echo linuxcando >> file1.js
user@ubudesk:~/proyecto0$ echo "viva el shell" > file3.js
user@ubudesk:~/proyecto0$ git status -s
  M file1.js
  ?? file3.js
user@ubudesk:~/proyecto0$ git add .
user@ubudesk:~/proyecto0$ git status -s
M   file1.js
A   file3.js
user@ubudesk:~/proyecto0$ git diff --staged
diff --git a/file1.js b/file1.js
index ce5ceb2..d48e451 100644
--- a/file1.js
+++ b/file1.js
@@ -3,3 +3,4 @@ adios
 adioss
 Commit directo
 hola
+linuxcando
diff --git a/file3.js b/file3.js
new file mode 100644
index 0000000..7b88364
--- /dev/null
+++ b/file3.js
@@ -0,0 +1 @@
+viva el shell
```

Vemos diff de cambios del *stage* a último *commit* en el repositorio local.

ANÁLISIS SALIDA *git diff*:

--- a/file1.js -> fichero en commit; cambios de línea en él se presentan con un signo -
+++ b/file1.js -> fichero en el stage; cambios de línea en él se presentan con un signo +

@@ -3,3 +3,4 @@

-3,3 empezando en la línea 3 (-3) se muestran 3 líneas (,3) del fichero ---, es decir, del commit.

+3,4 empezando en la línea 3 (+3) se muestran 4 líneas (,4) del fichero +++, es decir, el del stage.

```

user@ubudesk:~/proyecto0$ cat file1.js
hola, mundo
adios
adioss
Commit directo
hola
linuxeando
user@ubudesk:~/proyecto0$ code file1.js
user@ubudesk:~/proyecto0$ 

```

file1.js - Visual Studio Code

File Edit Selection View Go Run Terminal Help

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. [Manage](#) [Le](#)

JS file1.js 9+ X

home > user > proyecto0 > JS file1.js

```

1 hola, mundo
2 adios
3 ESTA LÍNEA ES EN MEDIO PARA MOSTRAR EL DIFF...
4 adioss
5 Commit directo
6 hola
7 linuxeando
8 

```

Fíjate que se ha tenido que añadir al stage para comparar con el último commit y nos dice en file1.js:

@@ -1,5 +1,7 @@ Fichero en el commit mostrado a partir de la línea 1 un número de 5 líneas y fichero en el stage a partir de línea 1 mostrado un número de 7 líneas, es decir, 2 nuevas con

+

```

user@ubudesk:~/proyecto0$ git diff --staged
diff --git a/file1.js b/file1.js
index ce5ceb2..d48e451 100644
--- a/file1.js
+++ b/file1.js
@@ -3,3 +3,4 @@ adios
 adioss
 Commit directo
 hola
+linuxeando
diff --git a/file3.js b/file3.js
new file mode 100644
index 0000000..7b88364
--- /dev/null
+++ b/file3.js
@@ -0,0 +1 @@
+viva el shell
user@ubudesk:~/proyecto0$ git add .
user@ubudesk:~/proyecto0$ git diff --staged
diff --git a/file1.js b/file1.js
index ce5ceb2..87871f1 100644
--- a/file1.js
+++ b/file1.js
@@ -1,5 +1,7 @@
 hola, mundo
 adios
+ESTA LÍNEA ES EN MEDIO PARA MOSTRAR EL DIFF...
 adioss
 Commit directo
 hola
+linuxeando
diff --git a/file3.js b/file3.js
new file mode 100644
index 0000000..7b88364
--- /dev/null
+++ b/file3.js
@@ -0,0 +1 @@
+viva el shell
user@ubudesk:~/proyecto0$ 

```

```
file1.js - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage

JS file1.js 9+ x
home > user > proyecto0 > JS file1.js
1 hola, mundo
2 adios PRUEBA EDICIÓN SOBRE LÍNEA...
3 ESTA LÍNEA ES EN MEDIO PARA MOSTRAR EL DIFF...
4 adioss
5 Commit directo
6 hola
7 linuxcando
8
```

```
user@ubudesk:~/proyecto0$ git add .
user@ubudesk:~/proyecto0$ git diff --staged
diff --git a/file1.js b/file1.js
index ce5ceb2..3f9d8a1 100644
--- a/file1.js
+++ b/file1.js
@@ -1,5 +1,7 @@
 hola, mundo
-adios
+adios PRUEBA EDICIÓN SOBRE LÍNEA...
+ESTA LÍNEA ES EN MEDIO PARA MOSTRAR EL DIFF...
 adioss
 Commit directo
 hola
+linuxcando
diff --git a/file3.js b/file3.js
new file mode 100644
index 0000000..7b88364
--- /dev/null
+++ b/file3.js
@@ -0,0 +1 @@
+viva el shell
user@ubudesk:~/proyecto0$
```

IMPORTANTE - Análisis de cabecera diff: @@ -1,5 +1,7 @@

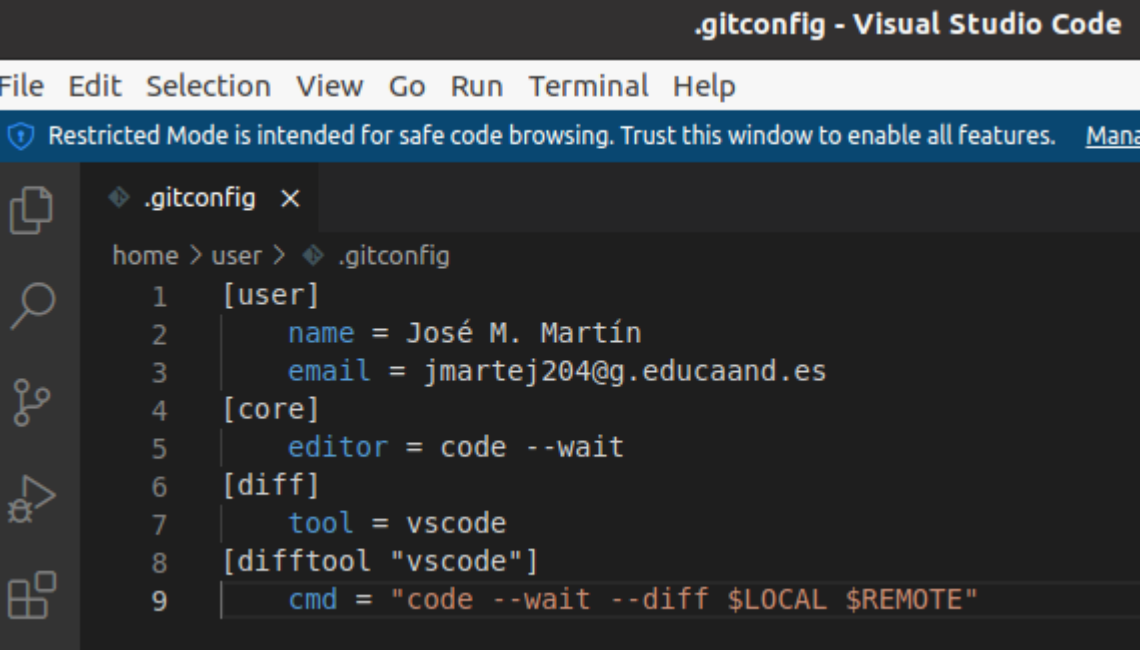
-1,5 se muestra bloque de últ. Fichero en commit, a partir de línea 1 en longitud de 5 líneas. Las líneas que pertenecen a este fichero van con signo - y en rojo de eliminación.

+1,7 se muestra bloque de fichero en stage, a partir de línea 1 en longitud de 7 líneas. Las líneas que pertenecen a este fichero van con signo + y en verde de añadido.

Herramientas para comparar ficheros

- Las más populares Visual Diff's son:
 - KDiff3 <http://kdiff3.sourceforge.net/>
 - P4Merge <https://www.perforce.com/products/helix-core-apps/merge-diff-tool-p4merge>
 - WinMerge <https://winmerge.org/>
 - VSCode, nos centraremos en éste.
 - Para utilizarlo configuramos git

```
user@ubudesk:~/proyecto0$ git config --global -e  
ayuda: Esperando que tu editor cierre el archivo ...
```



The screenshot shows the Visual Studio Code editor with the title bar ".gitconfig - Visual Studio Code". The menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. A status bar at the top indicates "Restricted Mode is intended for safe code browsing. Trust this window to enable all features." with a "Manage" link. The editor displays the contents of the ".gitconfig" file, which is located at "home > user > .gitconfig". The file content is as follows:

```
1 [user]
2     name = José M. Martín
3     email = jmartej204@g.educaand.es
4 [core]
5     editor = code --wait
6 [diff]
7     tool = vscode
8 [difftool "vscode"]
9     cmd = "code --wait --diff $LOCAL $REMOTE"
```

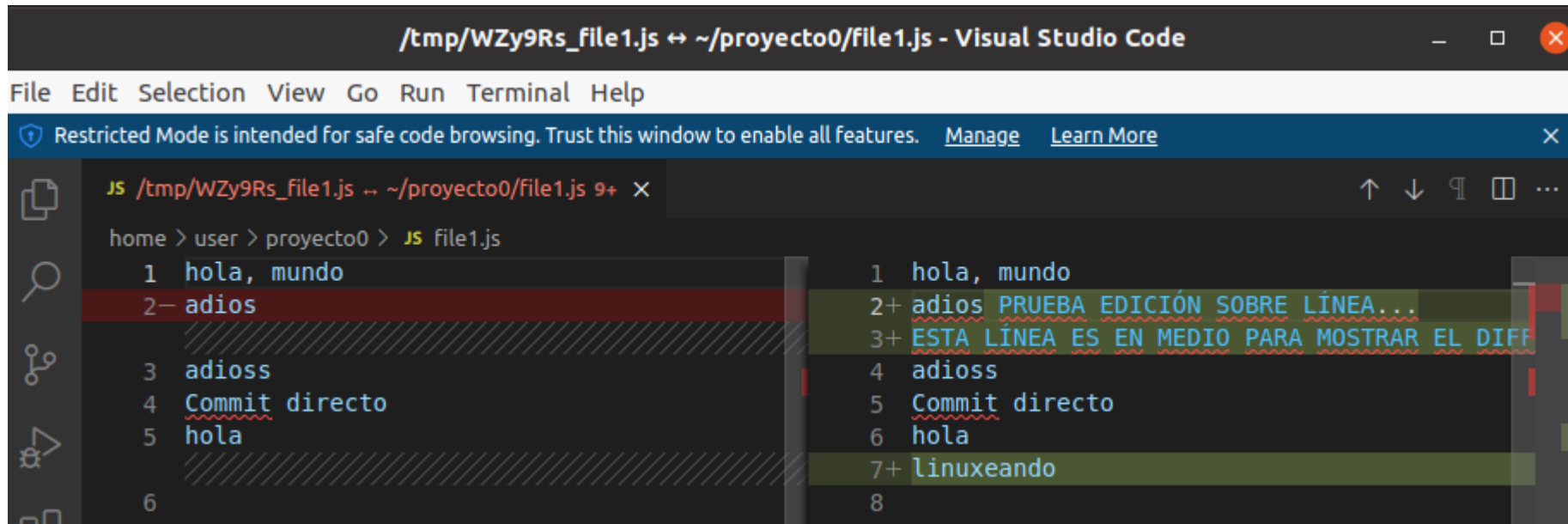
Herramientas para comparar ficheros

- Para invocar difftool como vscode configurado

```
user@ubudesk:~/proyecto0$ git difftool --staged
```

```
Viewing (1/2): 'file1.js'
```

```
Launch 'vscode' [Y/n]? Y
```



Fichero a la derecha es el fichero en stage.

Fichero a la izquierda es el fichero en último commit.

Se muestra la diferencia..

Histórico de commits

- Comandos para ver el historial de commits realizados:

```
git log
```

```
#Versión del histórico de commits con resumen por commit en una sola línea
```

```
git log --oneline
```

```
#Versión del histórico de commits con resumen por commit en una sola línea en
```

```
#orden reverso
```

```
git log --oneline --reverse
```

```
user@ubudesk:~/proyecto0$ git log
commit 5c9304d4d12cab37925f8b3acdca0de9a479e8f1 (HEAD -> master)
Author: José M. Martín <jmartej204@g.educaand.es>
Date:   Wed Sep 29 21:49:58 2021 +0200
```

Modifico file1.js y añado file2.js

```
commit a61582de3e714fb566c7a44b61e10d62d52a5d65
Author: José M. Martín <jmartej204@g.educaand.es>
Date:   Wed Sep 29 16:24:30 2021 +0200
```

Añado .gitignore al proyecto

```
commit 7258bb67bfbd44be594fa728bd7df48cd5214d63
Author: José M. Martín <jmartej204@g.educaand.es>
Date:   Tue Sep 28 17:32:41 2021 +0200
```

Cambios

```
commit de3f4c3f00b0bd5008aef25c211fd61a9b814c05
Author: José M. Martín <jmartej204@g.educaand.es>
Date:   Mon Sep 27 06:34:39 2021 +0200
```

Esto es un commit directo, nada recomendable.. mejor siempre pasar por el stage.

```
commit 8c5cf6862fb3c096f74f32c9e04770395878cb4e
Author: José M. Martín <jmartej204@g.educaand.es>
Date:   Tue Sep 14 08:54:12 2021 +0200
```

Cabecera del mensaje

Cuerpo del mensaje donde se describen los cambios realizados...

ANÁLISIS: Versión de una sola línea de comando

master es la rama principal (también llamada *main*) donde se mezclarán las restantes ramas creadas más tarde.

HEAD (*cabeza o cabezal*) indica que actualmente nos encontramos haciendo commits sobre la rama master HEAD -> master. *HEAD* va a indicar la rama actual sobre la que se realizan los commits

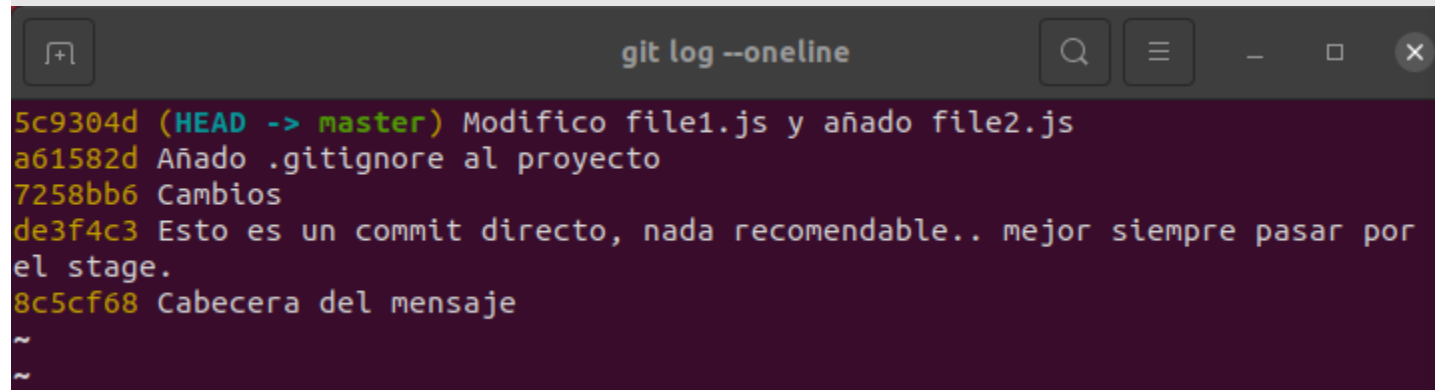
```
user@ubudesk:~/proyecto0$ git log --oneline
5c9304d (HEAD -> master) Modifico file1.js y añado file2.js
a61582d Añado .gitignore al proyecto
7258bb6 Cambios
de3f4c3 Esto es un commit directo, nada recomendable.. mejor siempre pasar por el stage.
8c5cf68 Cabecera del mensaje
```

```
user@ubudesk:~/proyecto0$ git log --oneline --reverse
8c5cf68 Cabecera del mensaje
de3f4c3 Esto es un commit directo, nada recomendable.. mejor siempre pasar por el stage.
7258bb6 Cambios
a61582d Añado .gitignore al proyecto
5c9304d (HEAD -> master) Modifico file1.js y añado file2.js
user@ubudesk:~/proyecto0$
```

Visualización de commits

- Para visualizar un commit se emplea el comando *show*:

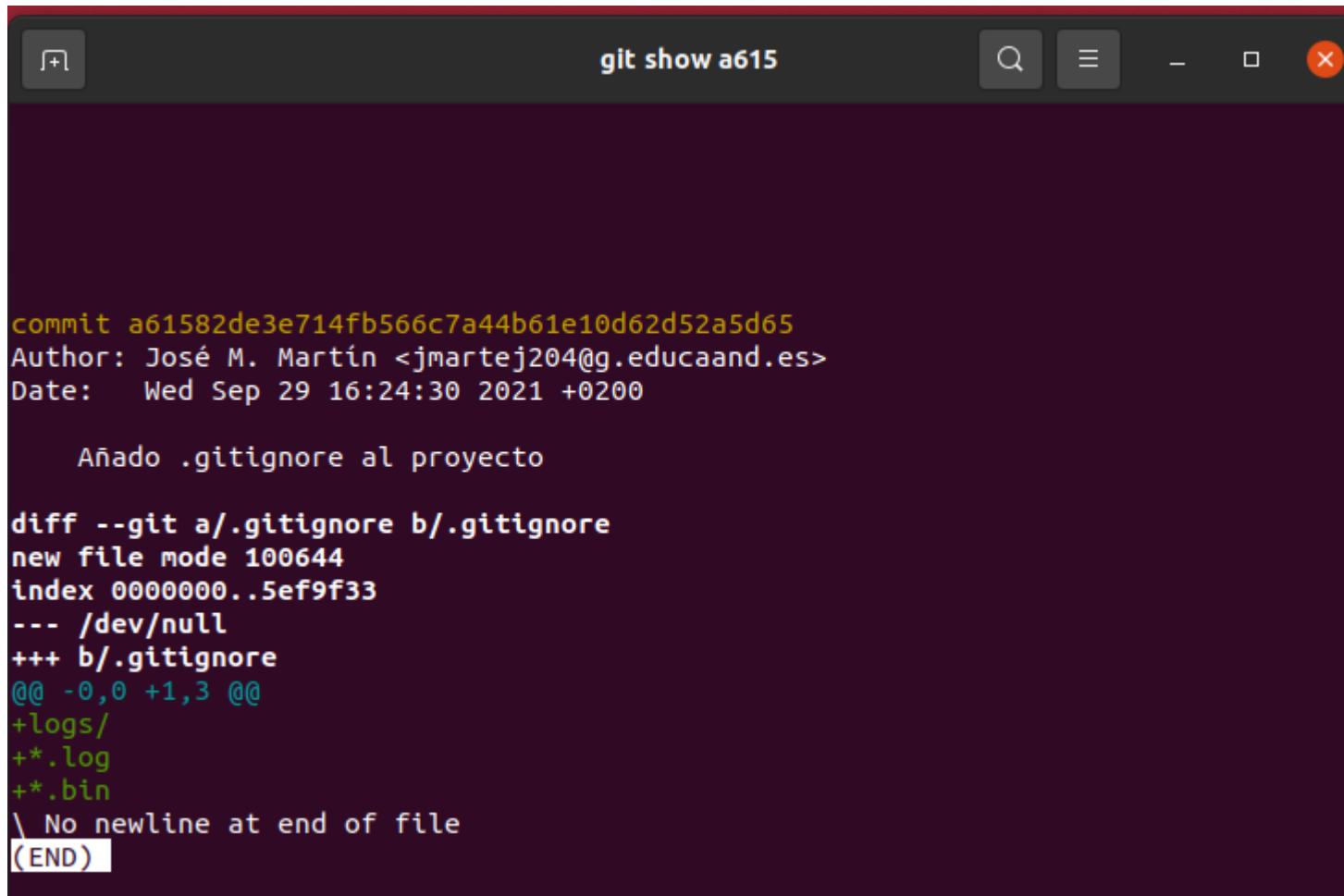
```
#Lanzamos primero el histórico de commits  
git log --oneline
```



```
5c9304d (HEAD -> master) Modifico file1.js y añado file2.js  
a61582d Añado .gitignore al proyecto  
7258bb6 Cambios  
de3f4c3 Esto es un commit directo, nada recomendable.. mejor siempre pasar por  
el stage.  
8c5cf68 Cabecera del mensaje  
~  
~
```

NOTA SALIR: Para salir de git log utiliza la opción *q* de quit.

```
#Seleccionando todo o parte del ID del commit (que lo identifique sin ambigüedad)  
que nos interesa con el comando show visualizamos este commit:  
git show a615
```

A screenshot of a terminal window with a dark background. The window title is 'git show a615'. The output shows the commit hash 'a61582de3e714fb566c7a44b61e10d62d52a5d65', the author 'José M. Martín', and the date 'Wed Sep 29 16:24:30 2021'. The commit message is 'Añado .gitignore al proyecto'. The diff shows a new file '.gitignore' with content: 'logs/', '*.log', and '*.bin'. The diff ends with '\ No newline at end of file' and '(END)' on a new line.

```
commit a61582de3e714fb566c7a44b61e10d62d52a5d65
Author: José M. Martín <jmartej204@g.educaand.es>
Date:   Wed Sep 29 16:24:30 2021 +0200

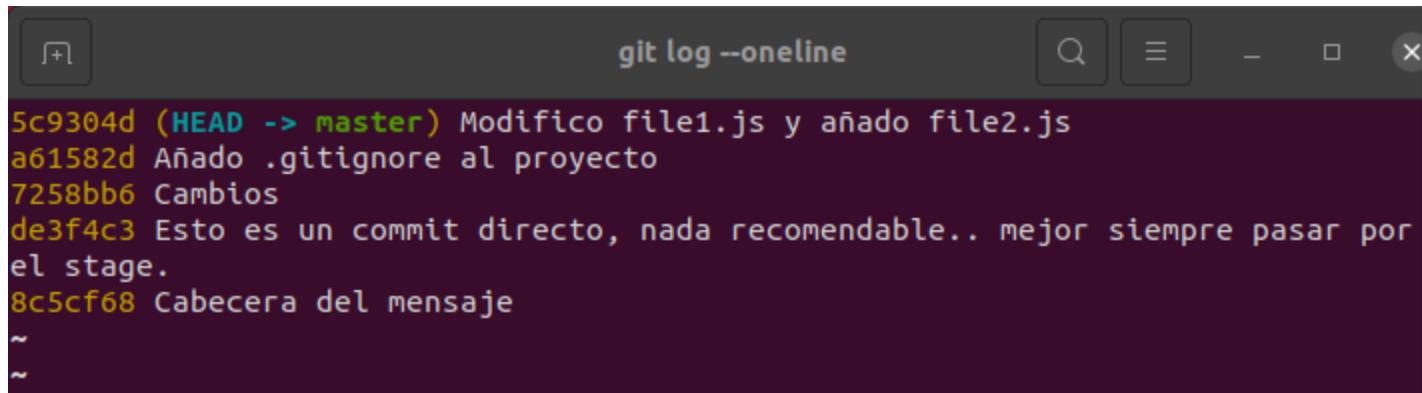
    Añado .gitignore al proyecto

diff --git a/.gitignore b/.gitignore
new file mode 100644
index 00000000..5ef9f33
--- /dev/null
+++ b/.gitignore
@@ -0,0 +1,3 @@
+logs/
+*.log
+*.bin
\ No newline at end of file
(END)
```

NOTA SALIR: Para salir de git show utiliza la opción *q* de quit.

Visualización de commits

- Otra opción es utilizar el *puntero* de HEAD:



```
git log --oneline
5c9304d (HEAD -> master) Modifico file1.js y añadido file2.js
a61582d Añado .gitignore al proyecto
7258bb6 Cambios
de3f4c3 Esto es un commit directo, nada recomendable.. mejor siempre pasar por
el stage.
8c5cf68 Cabecera del mensaje
~
~
```

#Indicando HEAD~número de pasos de commits hacia atrás para visualizar, por ejemplo
#si queremos visualizar el commit de *Cambios* utilizaría HEAD~2, es decir, de la
#posición HEAD, final de commits, retrocede 2 para mostrar:

```
git show HEAD~2
```

#También puedes centrarte en un fichero o ruta específica indicándoselo mediante los
#: en el ejemplo siguiente nos quedamos sólo con el fichero .gitignore

```
git show HEAD~1:.gitignore
```




```
git show HEAD~2

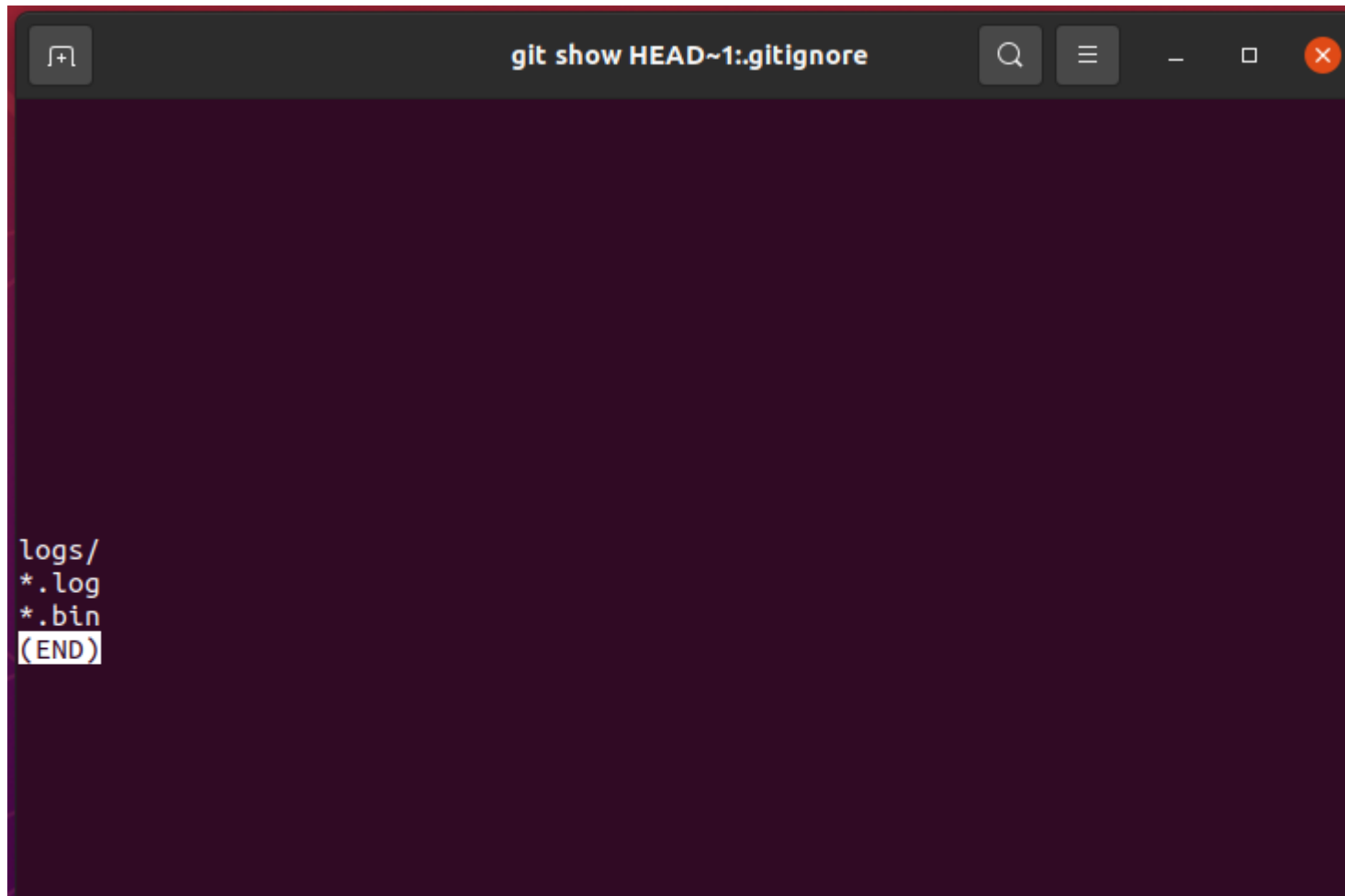
commit 7258bb67bfb44be594fa728bd7df48cd5214d63
Author: José M. Martín <jmartej204@g.educaand.es>
Date: Tue Sep 28 17:32:41 2021 +0200

    Cambios

diff --git a/file1.txt b/file1.js
similarity index 100%
rename from file1.txt
rename to file1.js
diff --git a/file2.txt b/file2.txt
deleted file mode 100644
index e3ea11f..0000000
--- a/file2.txt
+++ /dev/null
@@ -1,0,0 @@
-hola, mundo2
(END)
```

Si nos fijamos en la salida de este comando observamos que para el fichero file1 lo que registra el repo es un cambio de nombre.

Mientras que para file2.txt lo que se observa es la eliminación del fichero pasando de file2.txt a /dev/null el fichero vacío del sistema.



The image shows a terminal window with a dark background. The title bar at the top reads "git show HEAD~1:.gitignore". The terminal content displays the following text:

```
logs/  
*.log  
*.bin  
(END)
```

Salida exacta del commit HEAD~1
es decir, *cabeza menos 1* para el
fichero concreto *.gitignore*

```
git log --oneline
5c9304d (HEAD -> master) Modifico file1.js y añado file2.js
a61582d Añado .gitignore al proyecto
7258bb6 Cambios
de3f4c3 Esto es un commit directo, nada recomendable.. mejor siempre pasar por
el stage.
8c5cf68 Cabecera del mensaje
~
~
```

#En este ejemplo sobre el mismo histórico nos posicionamos en el primer commit

#realizado:

```
git show HEAD~4
```

#Equivalente a

```
git show 8c5c
```

```
git show HEAD~4

commit 8c5cf6862fb3c096f74f32c9e04770395878cb4e
Author: José M. Martín <jmartej204@g.educaand.es>
Date: Tue Sep 14 08:54:12 2021 +0200

    Cabecera del mensaje

    Cuerpo del mensaje donde se describen los cambios realizados...

diff --git a/file1.txt b/file1.txt
new file mode 100644
index 0000000..29e578a
--- /dev/null
+++ b/file1.txt
@@ -0,0 +1,3 @@
+hola, mundo
+adios
+adioss
diff --git a/file2.txt b/file2.txt
new file mode 100644
index 0000000..e3ea11f
--- /dev/null
+++ b/file2.txt
@@ -0,0 +1 @@
:

```

Destaca que puesto que es el primer commit nos muestra diferencias respecto del fichero vacío del sistema /dev/null

Visualización de commits

- Para visualizar el listado de los ficheros implicados en un commit tenemos la opción *git ls-tree*:

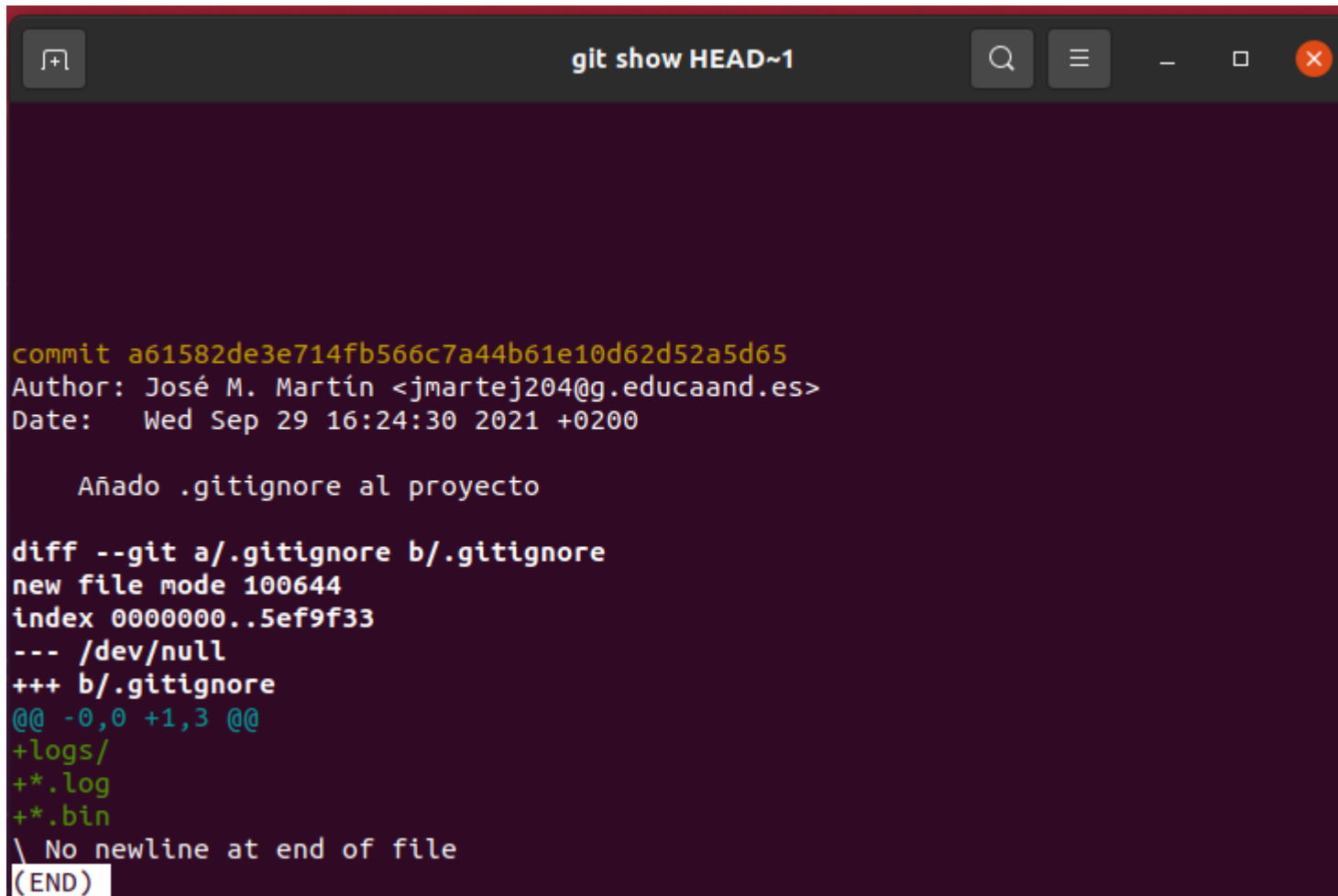
```
git log --oneline
5c9304d (HEAD -> master) Modifico file1.js y añado file2.js
a61582d Añado .gitignore al proyecto
7258bb6 Cambios
de3f4c3 Esto es un commit directo, nada recomendable.. mejor siempre pasar por el stage.
8c5cf68 Cabecera del mensaje
~
~
```

```
git ls-tree HEAD~1
```

```
user@ubudesk ~/proyecto0 master ➤ git ls-tree HEAD~1
100644 blob 5ef9f338ad71b56491cd5db7e6acbf246f772535 .gitignore
100644 blob d26c9e7e9c0ac6195694311fc7557b36c2c6eced file1.js
```

```
#Fíjate en el contraste con show
git show HEAD~1
```

Visualización de commits

A screenshot of a terminal window with a dark background. The title bar at the top reads 'git show HEAD~1'. The terminal output shows commit details for commit 'a61582de3e714fb566c7a44b61e10d62d52a5d65' by 'José M. Martín' on 'Wed Sep 29 16:24:30 2021 +0200'. The commit message is 'Añado .gitignore al proyecto'. The diff output shows a new file 'b/.gitignore' being added, containing patterns for logs, log files, and binaries. The output ends with '(END)'.

```
commit a61582de3e714fb566c7a44b61e10d62d52a5d65
Author: José M. Martín <jmartej204@g.educaand.es>
Date:   Wed Sep 29 16:24:30 2021 +0200

    Añado .gitignore al proyecto

diff --git a/.gitignore b/.gitignore
new file mode 100644
index 0000000..5ef9f33
--- /dev/null
+++ b/.gitignore
@@ -0,0 +1,3 @@
+logs/
+*.log
+*.bin
\ No newline at end of file
(END)
```

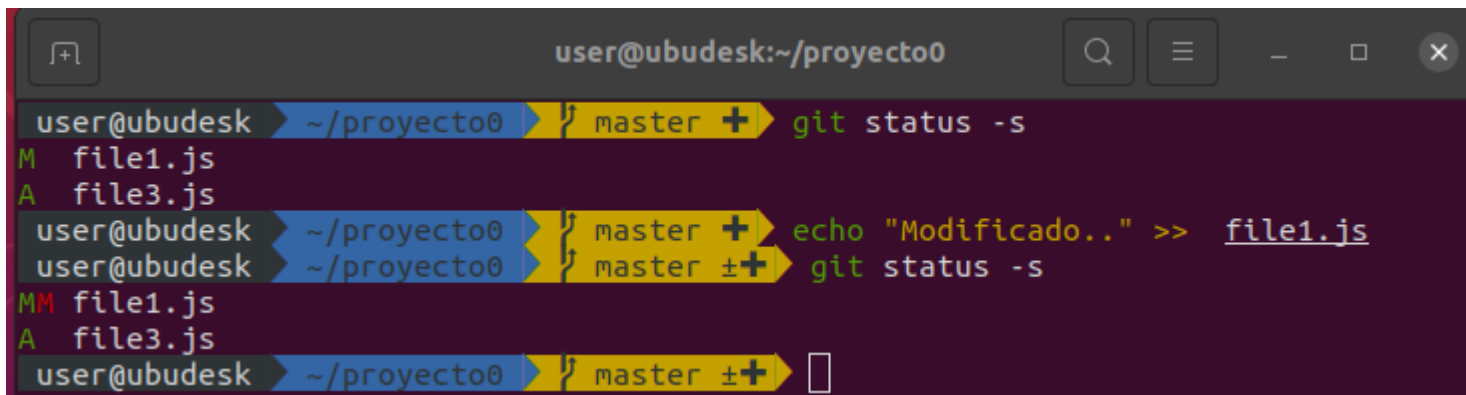
En este commit sólo nos está mostrando los cambios en un fichero *.gitignore* que se suma nuevo al repositorio.

Ls-tree te muestra listado el contenido o snapshot completo del repositorio en ese commit, no los cambios respecto del anterior.

file1.js que aparece en el listado no ha cambiado respecto del commit previo a HEAD~1

Unstaging

- *Unstaging* es la operación de sacar un fichero del *stage* o área de preparación previa al commit recomendado desde la versión git 2.23 frente al comando *reset* más confuso.
- *Básicamente, se trata de deshacer el comando git add.*
 - Para ello, en las últimas versiones de git se recomienda emplear el comando *restore*.
- Fijémonos en un escenario como el siguiente:



```
user@ubudesk:~/proyecto0
user@ubudesk > ~/proyecto0 > master + git status -s
M file1.js
A file3.js
user@ubudesk > ~/proyecto0 > master + echo "Modificado.." >> file1.js
user@ubudesk > ~/proyecto0 > master ++ git status -s
MM file1.js
A file3.js
user@ubudesk > ~/proyecto0 > master ++
```

```
user@ubudesk:~/proyecto0
user@ubudesk > ~/proyecto0 > master ++ git status -s
M file1.js
A file3.js
user@ubudesk > ~/proyecto0 > master ++ echo "Modificado.." >> file1.js
user@ubudesk > ~/proyecto0 > master ++ git status -s
MM file1.js
A file3.js
user@ubudesk > ~/proyecto0 > master ++ cat file1.js
hola, mundo
adios PRUEBA EDICIÓN SOBRE LÍNEA...
ESTA LÍNEA ES EN MEDIO PARA MOSTRAR EL DIFF...
adioss
Commit directo
hola
linuxeando
Modificado..
user@ubudesk > ~/proyecto0 > master ++ git restore --staged file1.js
user@ubudesk > ~/proyecto0 > master ++ git status -s
M file1.js
A file3.js
user@ubudesk > ~/proyecto0 > master ++
```

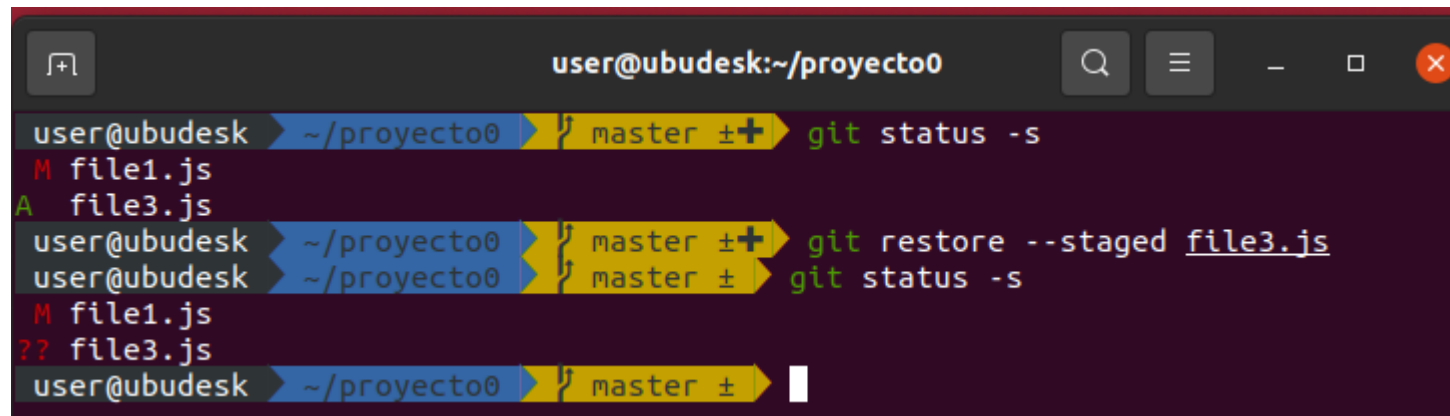
Después de aplicar *restore --staged* sobre el fichero file1.js desaparece la M verde del stage y sólo queda la M roja del directorio de trabajo.

```
user@ubudesk > ~/proyecto0 > master ++ git restore --staged file1.js
user@ubudesk > ~/proyecto0 > master ++ git status -s
M file1.js
A file3.js
user@ubudesk > ~/proyecto0 > master ++ cat file1.js
hola, mundo
adios PRUEBA EDICIÓN SOBRE LÍNEA...
ESTA LÍNEA ES EN MEDIO PARA MOSTRAR EL DIFF...
adioss
Commit directo
hola
linuxeando
Modificado..
user@ubudesk > ~/proyecto0 > master ++
```

Se puede entender que *restore --staged* lo que hace es restaurar la copia del repositorio al *stage* de modo que no hay cambios en el stage para el fichero.

Unstaging

- Si realizamos la misma operación sobre file3.js



```
user@ubudesk:~/proyecto0
user@ubudesk > ~/proyecto0 > master ±+ git status -s
M file1.js
A file3.js
user@ubudesk > ~/proyecto0 > master ±+ git restore --staged file3.js
user@ubudesk > ~/proyecto0 > master ± git status -s
M file1.js
?? file3.js
user@ubudesk > ~/proyecto0 > master ±
```

file3.js deja de estar bajo nuevo seguimiento del stage, es decir, hemos revertido *git add* de fichero nuevo, de ahí los signos de interrogación ?? de la versión *short* de status.

```
user@ubudesk:~/proyecto0
user@ubudesk > ~/proyecto0 > master ± git status
En la rama master
Cambios no rastreados para el commit:
  (usa "git add <archivo>..." para actualizar lo que será confirmado)
  (usa "git restore <archivo>..." para descartar los cambios en el directorio de
trabajo)
    modificados:    file1.js

Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
    file3.js

sin cambios agregados al commit (usa "git add" y/o "git commit -a")
user@ubudesk > ~/proyecto0 > master ±
```

Si vemos la versión normal de status nos refiere que el fichero file3.js ha pasado a no tener seguimiento.

Descartando cambios locales

- Para descartar cambios en el directorio de trabajo emplearemos el comando restore. En el siguiente escenario se muestra cómo trabaja.

```
user@ubudesk:~/proyecto0
user@ubudesk ~/proyecto0 master ± git status -s
M file1.js
?? file3.js
user@ubudesk ~/proyecto0 master ± cat file1.js
hola, mundo
adios PRUEBA EDICIÓN SOBRE LÍNEA...
ESTA LÍNEA ES EN MEDIO PARA MOSTRAR EL DIFF...
adioss
Commit directo
hola
linuxeando
Modificado..
user@ubudesk ~/proyecto0 master ± git restore .
user@ubudesk ~/proyecto0 master git status -s
?? file3.js
user@ubudesk ~/proyecto0 master cat file1.js
hola, mundo
adios
adioss
Commit directo
hola
user@ubudesk ~/proyecto0 master
```

file1.js está modificado en el directorio de trabajo, cuando se aplica el restore a todo el directorio (`git restore .`) este fichero se revierte a la copia más cercana que en este caso será la almacenada en el repositorio.

El siguiente status -s ya no muestra cambio en file1.js y el print por pantalla nos muestra que su contenido se ajusta ahora al contenido del HEAD para file1.js

```
git show HEAD:file1.js

hola, mundo
adios
adioss
Commit directo
hola
(END)
```

Contenido del HEAD:file1.js en correspondencia con el *restore* realizado al directorio de trabajo.

```
Modificado...
user@ubudesk ~/proyecto0 } master ± git restore .
user@ubudesk ~/proyecto0 } master   git status -s
?? file3.js
```

Nótese que el fichero file3.js que no estaba bajo seguimiento del repositorio, es decir, no estaba en el repositorio ni en el *stage*, y, por tanto, sigue en el mismo estado ??

```
user@ubudesk: ~/proyecto0
user@ubudesk: ~/proyecto0 } master git show HEAD:file1.js
user@ubudesk: ~/proyecto0 } master git clean
fatal: clean.requireForce default en true y ninguno de -i, -n, ni -f entregado;
rehusando el clean
X user@ubudesk: ~/proyecto0 } master git clean -h
uso: git clean [-d] [-f] [-i] [-n] [-q] [-e <patrón>] [-x | -X] [--] < rutas>...

    -q, --quiet           no imprimir nombres de archivos borrados
    -n, --dry-run         dry run (ejecución en seco)
    -f, --force           forzar
    -i, --interactive     limpieza interactiva
    -d                    borrar directorios completos
    -e, --exclude <patrón>
                           agregar <patrón> a las reglas de ignorancia
    -x                    borrar archivos ignorados, también
    -X                    borrar solo archivos ignorados

X user@ubudesk: ~/proyecto0 } master git clean -fd
Borrando file3.js
user@ubudesk: ~/proyecto0 } master
```

Para eliminar ficheros y directorios sin seguimiento por el repositorio podemos utilizar el comando *git clean -fd*

PELIGRO: git clean -fd borra los ficheros y directorios completos sin seguimiento por git (??) de forma que no se puede deshacer!!!