



Sumario

Uso básico git.....	4
1. Creación de usuario y instalación.....	4
Instalación de git en CLI.....	4
2. Creación de nuestro primera aplicación usando git.....	4
Creación de Hola Mundo con php y repositorio.....	4
Aregar la aplicación y mirar su estado.....	5
3. Diferenciando workdir y staging.....	6
4. Archivos para ignorar.....	7
Creando archivo .gitignore.....	7
Creación de archivo gitignore global.....	7
5. Trabajando con el historial.....	8
6. Recuperando versiones Antiguas.....	9
7. Utilizando Tags.....	10
Crear tags.....	10
Historial de tags.....	10
Borrar tags.....	10
Visualizar cambios.....	10
Uso avanzado git.....	12
1. Deshacer cambios.....	12
Deshacer antes del staging.....	12
Deshacer cambios antes del commit.....	12
Quitar commits no deseados.....	13
Borrar los commits de una raíz.....	14
Modificar un commit.....	14
2. Movimiento y borrar archivos.....	15
Mover un archivo a otro directorio con git.....	15
Ramas.....	16
1. Que es una rama en git.....	16
Introducción.....	16
Usando raíces en git.....	16
2. Fusión de ramas y resolución de conflictos.....	17
Resolver conflictos.....	18
Rebasing vs Merging.....	19
3. Mezclar en la rama master.....	20
GITHUB.....	21
Creación de key publica/privada.....	21
Creación de un repositorio.....	22
Clonar un repositorio.....	24
Ramas remotas.....	25
Enviando Actualizaciones.....	25
Recibiendo actualizaciones.....	26
Problemas de sincronización.....	28
CITAR PROYECTOS EN GITHUB.....	30
Entrar en zenodo.....	30
Pagina de Zenodo y Github.....	30
Publicación por Github.....	31
Etiqueta DOI.....	32
FLUJO DE TRABAJO EN GITHUB.....	33

Incidencias.....	33
Crear una Rama.....	33
Crear commits.....	35
Lanzar Aplicación.....	35
Sincronizar.....	36

Uso básico git

1. Creación de usuario y instalación.

Primero vamos a crear una cuenta(en mi caso lo cree en github.com). Cuando lo tengamos vamos a instalar git mediante CLI.

Para ello usamos los siguientes comandos:

```
git config -g user.name "el nombre que quieras".
```

```
git config -g user.email "el email que quieras".
```

Instalación de git en CLI.

Primero miramos si lo tenemos instalado con el siguiente comando:

```
git -v
```

Primero hacemos un update y un upgrade:

```
sudo apt update y sudo apt upgrade.
```

Despues hacemos sudo apt-get install git y procedera a instalar todo lo necesario para que funcione git.

Para comprobar que esta instalado utilizamos git -v.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git -v
git version 2.34.1
```

2. Creación de nuestro primera aplicación usando git

Primero vamos a crear un directorio donde vamos a crear nuestro primer proyecto usando git.

Para ello nos ubicamos donde queramos(En mi caso en el Escritorio y creamos una carpeta)

```
mkdir el_nombre_de_la_carpeta.
```

Nos ponemos dentro de la carpeta

```
cd el_nombre_de_la_carpeta.
```

Creación de Hola Mundo con php y repositorio

Vamos a crear nuestro primer archivo usando nano hola.php y dentro de este archivo tiene que estar lo siguiente.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ cat hola.php
<?php
echo "Hola Mundo\n";
?>
```

Ahora vamos a iniciar el repositorio

```
david@david-VirtualBox:~/Escritorio/curso_de_git$ 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, llama 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/david/Escritorio/curso_de_git/.git/
david@david-VirtualBox:~/Escritorio/curso_de_git$
```

Agregar la aplicación y mirar su estado

Para guardar los cambios que hemos hecho en este repositorio primero usamos el comando git add.

Git add nombre_de_la_aplicación(en mi caso hola.php).

Después vamos hacer un commit que lo que indica es los últimos cambios realizados y cuando se han hecho.

Git commit -m “el_mensaje_que_quieras”.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git commit -m "Creación del proyecto"
[master (commit-raíz) 6669ecb] Creación del proyecto
 1 file changed, 4 insertions(+)
 create mode 100644 hola.php
david@david-VirtualBox:~/Escritorio/curso-de-git$
```

vamos hace un status y como vemos no aparece nada.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git status
En la rama master
nada para hacer commit, el árbol de trabajo está limpio
```

Como hemos realizado un commit ve que no hay ningun cambio sin hacer el commit pertinente.

Ahora nuestro archivo vamos a modificarlo y vamos a poner lo siguiente y hacer un git status.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ cat hola.php
<?php
@print "Hola {$argv[1]}\n";
?>
david@david-VirtualBox:~/Escritorio/curso-de-git$ 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:      hola.php

sin cambios agregados al commit (usa "git add" y/o "git commit -a")
david@david-VirtualBox:~/Escritorio/curso-de-git$
```

Como vemos al hacer el git status sale como que se ha modificado el archivo y ahora si que tendriamos que utilizar git add hola.php y hacer un git commit para que guarde los cambios pertinentes.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git add nota.php
david@david-VirtualBox:~/Escritorio/curso-de-git$ git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    modificados:  hola.php

david@david-VirtualBox:~/Escritorio/curso-de-git$
```

Al aparecer en verde esta esperando que hagamos un commit para que guarde la ultima version que nosotros queramos para ello utilizamos git commit -m “y lo que quieras poner”.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git commit -m "Parametrización del programa"
[master 3a73f30] Parametrización del programa
 1 file changed, 1 insertion(+), 1 deletion(-)
david@david-VirtualBox:~/Escritorio/curso-de-git$ git status
En la rama master
nada para hacer commit, el árbol de trabajo está limpio
david@david-VirtualBox:~/Escritorio/curso-de-git$
```

Como vemos en la imagen al hacer el commit se guarda la versión y al volver hacer status aparece que no han habido cambios.

3. Diferenciando workdir y staging.

Vamos a modificar el archivo nuestro y tiene que aparecer lo siguiente.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ sudo nano hola.php
david@david-VirtualBox:~/Escritorio/curso-de-git$ cat hola.php
<?php
$nombre = isset($argv[1]) ? $argv[1]: "Mundo";
@print "Hola, {$nombre}\n";
?>
```

Ahora vamos a guardar los cambios pero no vamos ha realizar el commit utilizando git add.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git add hola.php
david@david-VirtualBox:~/Escritorio/curso-de-git$
```

Volvemos a modificar el archivo nuestro poniendo un comentario y hacemos un git status para ver el estado.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    modificados:  hola.php

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:  hola.php

david@david-VirtualBox:~/Escritorio/curso-de-git$
```

Como vemos uno sale en verde y otro en rojo es porque al hacer git add previamente al 1 sale que esta preparado para hacer commit(Staging) mientras que el ultimo al no hacer git add sale que esta en rojo y no podemos hacerle commit(workdir).

Ahora vamos ha realizar un commit para guardar los cambios.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git commit -m "Se añade un parámetro por defecto"
[master 48f9969] Se añade un parámetro por defecto
 1 file changed, 2 insertions(+), 1 deletion(-)
david@david-VirtualBox:~/Escritorio/curso-de-git$ 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:      hola.php

sin cambios agregados al commit (usa "git add" y/o "git commit -a")
```

Al realizar un commit el que estaba en verde(Staging) desaparece del status y queda al que todavía no hemos realizado un git add.

Ahora hacemos git add. (esto significa que todo los archivos que están en espera de hacer un commit se van a guardar).

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git add .
david@david-VirtualBox:~/Escritorio/curso-de-git$ git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    modificados:      hola.php

david@david-VirtualBox:~/Escritorio/curso-de-git$ git commit -m "Se añade un comentario al cambio del valor por defecto"
[master 931090d] Se añade un comentario al cambio del valor por defecto
 1 file changed, 1 insertion(+)
david@david-VirtualBox:~/Escritorio/curso-de-git$ git status
En la rama master
nada para hacer commit, el árbol de trabajo está limpio
david@david-VirtualBox:~/Escritorio/curso-de-git$
```

Al hacer git status espera que hagamos al commit y al realizar el commit si volvemos hacer status ve que la rama esta vacía a la hora de hacer un commit.

4. Archivos para ignorar

Creando archivo .gitignore

Es un archivo de configuración utilizado en sistemas de control de versiones como Git. Su propósito es especificar qué archivos y directorios deben ser ignorados por Git y no deben ser rastreados ni incluidos en el repositorio. Esto es útil para evitar que archivos temporales, compilaciones, archivos de configuración local y otros elementos no deseados se incluyan accidentalmente en el historial de cambios.

Creamos nuestro archivo gitignore con los siguiente dentro.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ sudo nano .gitignore
david@david-VirtualBox:~/Escritorio/curso-de-git$ cat .gitignore
# .gitignore
dir1/
# ignora todo lo que contenga el directorio dir1
!dir1/info.txt
# El operador ! excluye del ignore a dir1/info.txt (si se guardaría)
dir2/*.txt
# ignora todos los archivos txt que hay en el directorio dir2
dir3/**/*txt
# ignora todos los archivos txt que hay en el dir3 ysus subdirectorios
*.o
# ignora todos los archivos con extensión .o en todos los directorios
```

Creación de archivo gitignore global

Este archivo te permite especificar patrones de ignorar que son aplicables a todos tus repositorios Git, lo que puede ser útil para evitar la inclusión de archivos o directorios comunes que no deben ser rastreados en ningún proyecto

```
Terminal - david@david-VirtualBox:~/Escritorio/curso-de-git
Archivo Editar Ver Terminal Pestañas Ayuda
david@david-VirtualBox:~/Escritorio/curso-de-git$ cat .gitignore_global
# Compiled source #
#####
*.com
*.class
*.dll
*.exe
*.o
*.so
# Packages #
#####
# it's better to unpack these files and commit the raw source
# git has its own built in compression methods
.7z
*.dmg
*.gz
*.iso
*.jar
*.rar
*.tar
*.zip
# Logs and databases #
#####
*.log
*.sql
*.sqlite
# OS generated files #
#####
.DS_Store
.DS_Store?
.*_
.Spotlight-V100
.Trashes
ehthumbs.db
Thumbs.db
*~
*.swp
# IDEs #
#####
.idea
.settings/
.classpath
.project
david@david-VirtualBox:~/Escritorio/curso-de-git$
```

5. Trabajando con el historial

Utilizamos el comando git log para ver todos los commit que hemos realizado durante todo el curso del proyecto.

```
commit 931090dfee689be3fd98517b1f291519905653ed (HEAD -> master)
Author: David09x <dark_mkkey@gmail.com>
Date:   Fri Jan 19 16:54:35 2024 +0100

    Se añade un comentario al cambio del valor por defecto

commit 48f9969357c9c7428a5acfccf806d6bd73ad560c7
Author: David09x <dark_mkkey@gmail.com>
Date:   Fri Jan 19 16:48:43 2024 +0100

    Se añade un parámetro por defecto

commit 3a73f30ebaac3e21aff941414385a2d1611655fe
Author: David09x <dark_mkkey@gmail.com>
Date:   Fri Jan 19 16:30:49 2024 +0100

    Parametrización del programa

commit 4be339eadcfb4ea221103194e814f73483b9227b
Author: David09x <dark_mkkey@gmail.com>
Date:   Fri Jan 19 16:25:33 2024 +0100

    Creación del proyecto
-
-
-
-
-
(FIN.)
```

También hay otras formas de que nos salga el log de los commit utilizando –online y utilizando un personalizado.

```
[2]+ Detenido          git log
david@david-VirtualBox:~/Escritorio/curso-de-git$ git log --oneline
931090d (HEAD -> master) Se añade un comentario al cambio del valor por defecto
48f9969 Se añade un parámetro por defecto
3a73f30 Parametrización del programa
4be339e Creación del proyecto
david@david-VirtualBox:~/Escritorio/curso-de-git$ git log --pretty=format:'%h %ad | %s%d [%an]' --graph --date=short
* 931090d 2024-01-19 | Se añade un comentario al cambio del valor por defecto (HEAD -> master) [David09x]
* 48f9969 2024-01-19 | Se añade un parámetro por defecto [David09x]
* 3a73f30 2024-01-19 | Parametrización del programa [David09x]
* 4be339e 2024-01-19 | Creación del proyecto [David09x]
david@david-VirtualBox:~/Escritorio/curso-de-git$
```

Como vemos utilizar git log –online te saca los commits de una manera mas corta y utilizando –pretty lo que hacemos es darle un formato que sale la fecha de cuando se ha realizado los commits.

6. Recuperando versiones Antiguas.

Vamos a recuperar la primera versión que hemos creado para ellos si vemos en el git log –oneline nuestra versión seria la 4be339e, utilizaremos git checkout para ir a la versión que queramos.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git checkout 4be339e
Nota: cambiando a '4be339e'.

Te encuentras en estado 'detached HEAD'. Puedes revisar por aquí, hacer cambios experimentales y hacer commits, y puedes descartar cualquier commit que hayas hecho en este estado sin impactar a tu rama realizando otro checkout.

Si quieras crear una nueva rama para mantener los commits que has creado, puedes hacerlo (ahora o después) usando -c con el comando checkout. Ejemplo:

git switch -c <nombre-de-nueva-rama>

D deshacer la operación con:

git switch -

Desactiva este aviso poniendo la variable de config advice.detachedHead en false

HEAD está ahora en 4be339e Creación del proyecto
david@david-VirtualBox:~/Escritorio/curso-de-git$ cat hola.php
<?php
echo "Hola Mundo\n";
?>
```

Como vemos, hemos vuelto al primer repositorio que hicimos.

Ahora hacemos lo mismo pero le indicamos la raiz que queremos estar.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git checkout master
La posición previa de HEAD era 4be339e Creación del proyecto
Cambiado a rama 'master'
david@david-VirtualBox:~/Escritorio/curso-de-git$ cat hola.php
<?php
// El nombre por defecto es Mundo
$nombre = isset($argv[1]) ? $argv[1]: "Mundo";
@print "Hola, {$nombre}\n";
?>
david@david-VirtualBox:~/Escritorio/curso-de-git$
```

Al utilizar checkout master hemos vuelto a la ultima versión de nuestro repositorio.

7. Utilizando Tags.

Crear tags

Los tags son utilizados para recuperar las versiones específicas que queramos. Para ello vamos a darle a nuestro repositorio varias versiones.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git tag v1
```

Ahora vamos a otra versión a darle otro tag.

Utilizamos git checkout v1^ o v1~1 y le damos un tag utilizando git tag v1-beta. Si queremos saber todos los tags que hemos utilizado tan solo hay que utilizar el comando git tag y se mostrarán todas las versiones que hemos realizado.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git checkout v1^
Nota: cambiando a 'v1^'.

Te encuentras en estado 'detached HEAD'. Puedes revisar por aquí, hacer cambios experimentales y hacer commits, y puedes descartar cualquier commit que hayas hecho en este estado sin impactar a tu rama realizando otro checkout.

Si quieras crear una nueva rama para mantener los commits que has creado, puedes hacerlo (ahora o después) usando -c con el comando checkout. Ejemplo:

git switch -c <nombre-de-nueva-rama>

O deshacer la operación con:

git switch -

Desactiva este aviso poniendo la variable de config advice.detachedHead en false

HEAD está ahora en 48f9969 Se añade un parámetro por defecto
david@david-VirtualBox:~/Escritorio/curso-de-git$ git tag v1-beta
david@david-VirtualBox:~/Escritorio/curso-de-git$ git tag
v1
v1-beta
```

Historial de tags

Si queremos ver todo el historial utilizamos el comando git hist master --all

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git hist master --all
* 931090d 2024-01-19 | Se añade un comentario al cambio del valor por defecto (tag: v1, master) [%]
| an]
* 48f9969 2024-01-19 | Se añade un parámetro por defecto (HEAD, tag: v1-beta) [%]
| an]
* 3a73f30 2024-01-19 | Parametrización del programa [%]
| an]
* 4be339e 2024-01-19 | Creación del proyecto [%]
| an]
david@david-VirtualBox:~/Escritorio/curso-de-git$
```

Borrar tags

Git tag -d nombre_tag.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git tag -d v1
```

Visualizar cambios

Con git diff se miraran todos los cambios que hemos realizado. Si utilizamos el comando sin especificar nada lo que hará es mostrar los cambios que no se han añadido todavía (los que no se han

hecho antes de usar git add). Tambien podemos utilizarlo para comparar ambas versiones git diff tag1 tag2.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git diff v1-beta v1
diff --git a/hola.php b/hola.php
index 449a414..b5d6f43 100644
--- a/hola.php
+++ b/hola.php
@@ -1,4 +1,5 @@
<?php
+// El nombre por defecto es Mundo
$nombre = isset($argv[1]) ? $argv[1]: "Mundo";
@print "Hola, {$nombre}\n";
?>
```

Uso avanzado git

1. Deshacer cambios

Deshacer antes del staging

Vamos a la rama ultima que tengamos con el comando git checkout master, despues modificamos el archivo hola y tiene que contener lo siguiente.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git checkout master
La posición previa de HEAD era 48f9969 Se añade un parámetro por defecto
cambiado a rama 'master'
david@david-VirtualBox:~/Escritorio/curso-de-git$ sudo nano hola.php
[sudo] contraseña para david:
david@david-VirtualBox:~/Escritorio/curso-de-git$ cat hola.php
<?php
// Este comentario está mal y hay que borrarlo
$nombre = isset($argv[1]) ? $argv[1]: "Mundo";
@print "Hola, {$nombre}\n";
?>
david@david-VirtualBox:~/Escritorio/curso-de-git$
```

Ahora hacemos un status y vemos que al hacer cambios estara esperando el git add y el commit.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ 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:      hola.php
```

Ahora utilizamos git checkout hola.php lo que hará es que se vuelva a la anterior versión que teníamos.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git checkout hola.php
Actualizada 1 ruta desde el índice
david@david-VirtualBox:~/Escritorio/curso-de-git$ cat hola.php
<?php
// El nombre por defecto es Mundo
$nombre = isset($argv[1]) ? $argv[1]: "Mundo";
@print "Hola, {$nombre}\n";
?>
david@david-VirtualBox:~/Escritorio/curso-de-git$ git status
En la rama master
nada para hacer commit, el árbol de trabajo está limpio
```

Deshacer cambios antes del commit.

Haremos lo mismo pero vamos hacer los cambios al staging sin utilizar commit para ello modificamos de nuevo el archivo hola.php y que quede así.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ cat hola.php
<?php
// Este comentario está mal y hay que borrarlo
$nombre = isset($argv[1]) ? $argv[1]: "Mundo";
@print "Hola, {$nombre}\n";
?>
```

Ahora vamos agregarlo para hacerle un commit para ello utilizamos git add hola.php y git status para ver que esta preparado para hacerle commit.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git add hola.php
david@david-VirtualBox:~/Escritorio/curso-de-git$ git status
En la rama master
Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    modificados:      hola.php
```

Ahora vamos a volver a quitarle el staging o ponerlo preparado para hacerle un commit para ello utilizamos el comando git reset HEAD hola.php y comprobamos con git status que su estado no esté previo para hacerle el git add.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git reset HEAD hola.php
Cambios fuera del área de stage tras el reset:
M      hola.php
david@david-VirtualBox:~/Escritorio/curso-de-git$ 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:      hola.php
```

Al estar en rojo con ello confirmamos de que le ha quitado la fase previa de hacerle un commit.

Vamos a ponerlo antes de hacer todos los cambios para ello hacemos git checkout hola.php.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git checkout hola.php
Actualizada 1 ruta desde el índice
david@david-VirtualBox:~/Escritorio/curso-de-git$ cat hola
cat: hola: No existe el archivo o el directorio
david@david-VirtualBox:~/Escritorio/curso-de-git$ cat hola.php
<?php
// El nombre por defecto es Mundo
$nombre = isset($argv[1]) ? $argv[1]: "Mundo";
@print "Hola, {$nombre}\n";
?>
```

Quitar commits no deseados

Vamos a quitar un commit que hemos realizado y no lo queríamos hacer, vamos a revestirlo.

Primero modificamos el archivo.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ cat hola.php
<?php
// Este comentario está mal y hay que borrarlo
$nombre = isset($argv[1]) ? $argv[1]: "Mundo";
@print "Hola, {$nombre}\n";
?>
```

Ahora vamos a hacer un commit del archivo.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ sudo nano hola.php
david@david-VirtualBox:~/Escritorio/curso-de-git$ git add hola.php
david@david-VirtualBox:~/Escritorio/curso-de-git$ git commit -m "Ups... este commit está mal."
[master c6658f3] Ups... este commit está mal.
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Ahora vamos a deshacer el cambio que hemos realizado y desharemos el cambio con el comando git revert.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git revert HEAD --no-edit
[master b179192] Revert "Ups... este commit está mal."
  Date: Sun Jan 21 16:44:34 2024 +0100
  1 file changed, 1 insertion(+), 1 deletion(-)
david@david-VirtualBox:~/Escritorio/curso-de-git$ git hist
* b179192 2024-01-21 | Revert "Ups... este commit está mal." (HEAD -> master) [%]
| an]
* c6658f3 2024-01-21 | Ups... este commit está mal. [%]
| an]
* 958d9c7 2024-01-19 | los ignore [%]
| an]
* 931090d 2024-01-19 | Se añade un comentario al cambio del valor por defecto (tag: v1) [%]
| an]
* 48f9969 2024-01-19 | Se añade un parámetro por defecto (tag: v1-beta) [%]
| an]
* 3a73f30 2024-01-19 | Parametrización del programa [%]
| an]
* 4be339e 2024-01-19 | Creación del proyecto [%]
| an]
```

Para ver que se ha realizado podemos utilizar git hist que lista todos los commits realizados y se nos indica que acción se han realizado sobre el, en este caso como vemos b179192 vemos Revert que indica que la acción realizada sobre el archivo se ha deshecho.

Borrar los commits de una raíz

En el apartado anterior vemos que aun queda registros de los cambios realizados, si queremos que queden hay que utilizar el comando git reset.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git reset --hard v1
HEAD está ahora en 931090d Se añade un comentario al cambio del valor por defecto
david@david-VirtualBox:~/Escritorio/curso-de-git$ git hist
* 931090d 2024-01-19 | Se añade un comentario al cambio del valor por defecto (HEAD -> master, tag: v1) [%]
| an]
* 48f9969 2024-01-19 | Se añade un parámetro por defecto (tag: v1-beta) [%]
| an]
* 3a73f30 2024-01-19 | Parametrización del programa [%]
| an]
* 4be339e 2024-01-19 | Creación del proyecto [%]
| an]
```

Al realizar el git reset usando el comando git hist vemos como en el historial ya no aparece el commit con Revert.

Modificar un commit.

Modificamos nuestro archivo y tiene que quedar así.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ cat hola.php
<?php
//Autor: David Amorós Sendra
// El nombre por defecto es Mundo
$nombre = isset($argv[1]) ? $argv[1]: "Mundo";
@print "Hola, {$nombre}\n";
?>
```

Ahora vamos hacer un commit.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git commit -a -m "Añadido el autor del programa"
[master be55a77] Añadido el autor del programa
 1 file changed, 1 insertion(+)
```

Nos hemos dado cuenta que se nos faltaba el correo electrónico pues de nuevo modificamos el archivo hola.php y le ponemos nuestro correo.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ sudo nano hola.php
david@david-VirtualBox:~/Escritorio/curso-de-git$ cat hola.php
<?php
//Autor: David Amorós Sendra <damoros2003@hotmail.com>
// El nombre por defecto es Mundo
$nombre = isset($argv[1]) ? $argv[1]: "Mundo";
@print "Hola, {$nombre}\n";
?>
```

Procedemos a realizar el siguiente commit.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git add hola.php
david@david-VirtualBox:~/Escritorio/curso-de-git$ git commit --amend -m "Añadido el autor del programa y su email"
[master d039f19] Añadido el autor del programa y su email
Date: Sun Jan 21 16:59:48 2024 +0100
1 file changed, 1 insertion(+)
david@david-VirtualBox:~/Escritorio/curso-de-git$ git hist
* d039f19 2024-01-21 | Añadido el autor del programa y su email (HEAD -> master) [%]
| an]
* 931090d 2024-01-19 | Se añade un comentario al cambio del valor por defecto (tag: v1) [%]
| an]
* 48f9969 2024-01-19 | Se añade un parámetro por defecto (tag: v1-beta) [%]
| an]
* 3a73f30 2024-01-19 | Parametrización del programa [%]
| an]
* 4be339e 2024-01-19 | Creación del proyecto [%]
| an]
```

Al realizar git hist vemos en el registro como el anterior commit previo a poner le email no esta y si que esta el ultimo que hemos realizado.

2. Movimiento y borrar archivos.

Mover un archivo a otro directorio con git.

Primero creamos un directorio(en mi caso lo llamare lib) y realizamos los siguientes comandos.

Git mv nombre_archivo nombre_directorio lo que hacemos con este comando es mover un archivo de un directorio y a otro y con git status podemos ver que es lo que se ha realizado.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ mkdir lib
david@david-VirtualBox:~/Escritorio/curso-de-git$ git mv hola.php lib
david@david-VirtualBox:~/Escritorio/curso-de-git$ git status
En la rama master
Cambios a ser confirmados:
 (usa "git restore --staged <archivo>..." para sacar del área de stage)
 renombrados:   hola.php -> lib/hola.php
```

Ramas

1. Que es una rama en git

Introducción

En Git, una rama es una línea independiente de desarrollo que permite trabajar en características o correcciones sin afectar la rama principal. Permite la creación, modificación y fusión de código de manera aislada, facilitando el trabajo colaborativo y la gestión de versiones en proyectos de software.

Usando raíces en git

Vamos a crear una raíz para ello usamos el comando git branch hola y para irnos a la rama haremos git checkout hola.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git branch hola
david@david-VirtualBox:~/Escritorio/curso-de-git$ git checkout hola
D      hola.php
A      lib/hola.php
Cambiado a rama 'hola'
```

Estos 2 pasos se podrían ejecutar con un solo comando: git checkout -b hola.

Vamos a crear un archivo en lib llamado HolaMundo que tiene que ser así.

```
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ cat HolaMundo.php
<?php

class HolaMundo
{
    private $nombre;
    function __construct($nombre)
    {
        $this->nombre= $nombre;
    }
    function __toString()
    {
        return sprintf ("Hola, %s.\n", $this->nombre);
    }
}
?>
```

Ahora modificamos nuestro hola.php y hacemos commit a los 2 archivos.

```
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ cat hola.php
<?php
//Autor: David Amorós Sendra <damoros2003@hotmail.com>
// El nombre por defecto es Mundo
require ('HolaMundo.php');

$nombre = isset($argv[1]) ? $argv[1]: "Mundo";
@print new HolaMundo($nombre);
?>
```



```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git add lib/HolaMundo.php
david@david-VirtualBox:~/Escritorio/curso-de-git$ git commit -m "Añadida la clase HolaMundo"
[hola 5646bb5] Añadida la clase HolaMundo
 2 files changed, 15 insertions(+)
  create mode 100644 lib/HolaMundo.php
  rename hola.php => lib/hola.php (100%)
david@david-VirtualBox:~/Escritorio/curso-de-git$ git add lib/hola.php
david@david-VirtualBox:~/Escritorio/curso-de-git$ git commit -m "hola usa la clase HolaMundo"
[hola 9e7cba7] hola usa la clase HolaMundo
 1 file changed, 3 insertions(+), 1 deletion(-)
```

Tenemos una manera de movernos entre ramsa para ello utilizamos git checkout y la rama seleccionada.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git checkout master
Ya en 'master'
david@david-VirtualBox:~/Escritorio/curso-de-git$ git checkout hola
Cambiado a rama 'hola'
```

Vamos a realizar cambios en la rama master para ellos vamos a la rama master usando git checkout master y creamos un archivo README con el siguiente contenido.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ cat README.md
# Curso de GIT

Este proyecto contiene el curso de introducción a GIT.
```

Despues hacemos un commit al archivo y vemos como ha quedado toda la rama con git hist.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git add README.md
david@david-VirtualBox:~/Escritorio/curso-de-git$ git commit -m "Añadido README.md"
[master 02cc695] Añadido README.md
 1 file changed, 3 insertions(+)
 create mode 100644 README.md
david@david-VirtualBox:~/Escritorio/curso-de-git$ git hist --all
* 02cc695 2024-01-22 | Añadido README.md (HEAD -> master) [%]
 | an]
 | * 9e7cba7 2024-01-21 | hola usa la clase HolaMundo (hola) [%]
 | | an]
 | * 5646bb5 2024-01-21 | Añadida la clase HolaMundo [%]
 | | an]
* d039f19 2024-01-21 | Añadido el autor del programa y su email [%]
 | an]
* 931090d 2024-01-19 | Se añade un comentario al cambio del valor por defecto (tag: v1) [%]
 | an]
* 48f9969 2024-01-19 | Se añade un parámetro por defecto (tag: v1-beta) [%]
 | an]
* 3a73f30 2024-01-19 | Parametrización del programa [%]
 | an]
* 4be339e 2024-01-19 | Creación del proyecto [%]
 | an]
```

2. Fusión de ramas y resolución de conflictos

Vamos a irnos a la rama hola usando git checkout hola y hacemos una fusión utilizando merge que es con el comando git merge master con ello lo que decimos es que la rama master se fusiona a la hola y tengan los mismos cambios. Listamos como quedaría la rama con git hist.

```

david@david-VirtualBox:~/Escritorio/curso-de-git$ git checkout hola
Cambiado a rama 'hola'
david@david-VirtualBox:~/Escritorio/curso-de-git$ git merge master
Merge made by the 'ort' strategy.
 README.md | 3 +++
 1 file changed, 3 insertions(+)
 create mode 100644 README.md
david@david-VirtualBox:~/Escritorio/curso-de-git$ git hist --all
* 82ad9a7 2024-01-22 | Merge branch 'master' into hola (HEAD -> hola) [%]
 \ an]
 * 02cc695 2024-01-22 | Añadido README.md (master) [%]
 | an]
 * 9e7cba7 2024-01-21 | hola usa la clase HolaMundo [%]
 | an]
 * 5646bb5 2024-01-21 | Añadida la clase HolaMundo [%]
 / an]
* d039f19 2024-01-21 | Añadido el autor del programa y su email [%]
 | an]
* 931090d 2024-01-19 | Se añade un comentario al cambio del valor por defecto (tag: v1) [%]
 | an]
* 48f9969 2024-01-19 | Se añade un parámetro por defecto (tag: v1-beta) [%]
 | an]
* 3a73f30 2024-01-19 | Parametrización del programa [%]
 | an]
* 4be339e 2024-01-19 | Creación del proyecto [%]
 an]
david@david-VirtualBox:~/Escritorio/curso-de-git$
```

Resolver conflictos

Vamos a la rama master dentro de ella modificamos el archivo hola.php y hacemos commit.

```

david@david-VirtualBox:~/Escritorio/curso-de-git$ git add hola.php
david@david-VirtualBox:~/Escritorio/curso-de-git$ git commit -m "Programa interactivo"
[master c0e566a] Programa interactivo
 1 file changed, 2 insertions(+), 2 deletions(-)
david@david-VirtualBox:~/Escritorio/curso-de-git$ git hist --all
* c0e566a 2024-01-22 | Programa interactivo (HEAD -> master) [%]
 | an]
 | * 82ad9a7 2024-01-22 | Merge branch 'master' into hola (hola) [%]
 | | an]
 | |
 | / 
* 02cc695 2024-01-22 | Añadido README.md [%]
 | an]
 * 9e7cba7 2024-01-21 | hola usa la clase HolaMundo [%]
 | an]
 * 5646bb5 2024-01-21 | Añadida la clase HolaMundo [%]
 / an]
* d039f19 2024-01-21 | Añadido el autor del programa y su email [%]
 | an]
* 931090d 2024-01-19 | Se añade un comentario al cambio del valor por defecto (tag: v1) [%]
 | an]
* 48f9969 2024-01-19 | Se añade un parámetro por defecto (tag: v1-beta) [%]
 | an]
* 3a73f30 2024-01-19 | Parametrización del programa [%]
 | an]
* 4be339e 2024-01-19 | Creación del proyecto [%]
 an]
```

Ahora vamos a la rama hola y hacemos un merge con la master.

```

david@david-VirtualBox:~/Escritorio/curso-de-git$ git checkout hola
Cambiado a rama 'hola'
david@david-VirtualBox:~/Escritorio/curso-de-git$ ls
lib README.md
david@david-VirtualBox:~/Escritorio/curso-de-git$ git merge master
Auto-fusionando lib/hola.php
CONFLICTO (contenido): Conflicto de fusión en lib/hola.php
Fusión automática falló; arregle los conflictos y luego realice un commit con el resultado.
```

El archivo hola.php tendría que parecerse a esto.

```

<?php
//Autor: David Amorós Sendra <damoros2003@hotmail.com>
<<<<< HEAD
// El nombre por defecto es Mundo
require ('HolaMundo.php');

$nombre = isset($argv[1]) ? $argv[1]: "Mundo";
@print new HolaMundo($nombre);
=====
print "Introduce tu nombre:";
$nombre = trim(fgets(STDIN));
@print "Hola, {$nombre}\n";
>>>>> master:hola.php
?>

```

Vamos a resolver el conflicto modificando el código para que quede así y luego haciendo git add.

```

david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ cat hola.php
<?php
//Autor: David Amorós Sendra <damoros2003@hotmail.com>
require ('HolaMundo.php');

print "Introduce tu nombre:";
$nombre = trim(fgets(STDIN));
print new HolaMundo($nombre);
?>

```

```

david@david-VirtualBox:~/Escritorio/curso-de-git$ git add lib/hola.php
david@david-VirtualBox:~/Escritorio/curso-de-git$ git commit -m "Solucionado el conflicto al fusionar con la rama master"
[hola fc3812d] Solucionado el conflicto al fusionar con la rama master

```

Rebasing vs Merging

Rebasing en Git es un proceso que permite reorganizar o combinar cambios en una rama, generalmente para mantener un historial de commits más limpio y fácil de entender.

Hacemos Git checkout hola para situarnos en la rama hola vamos hacer un git reset hard a commit de hola usa la clase holaMundo.

```

david@david-VirtualBox:~/Escritorio/curso-de-git$ git reset --hard aa4c7fb
HEAD está ahora en aa4c7fb hola usa la clase HolaMundo
david@david-VirtualBox:~/Escritorio/curso-de-git$ 

```

Con esto hemos dejado nuestra rama sin ningún merge de los anteriores.

Ahora vamos a hacer un rebase (seguimos en la rama hola).

```

david@david-VirtualBox:~/Escritorio/curso-de-git$ git rebase master
Auto-fusionando lib/hola.php
CONFLICTO (contenido): Conflicto de fusión en lib/hola.php
error: no se pudo aplicar aa4c7fb... hola usa la clase HolaMundo
ayuda: Resolve all conflicts manually, mark them as resolved with
ayuda: "git add/rm <conflicted files>", then run "git rebase --continue".
ayuda: You can instead skip this commit: run "git rebase --skip".
ayuda: To abort and get back to the state before "git rebase", run "git rebase --abort".
No se pudo aplicar aa4c7fb... hola usa la clase HolaMundo

```

Como vemos nos da un conflicto en el archivo hola.php, tenemos que hacer los cambios que hicimos antes y dejar el archivo así.

```

david@david-VirtualBox:~/Escritorio/curso-de-git$ sudo nano lib/hola.php
david@david-VirtualBox:~/Escritorio/curso-de-git$ cat lib/hola.php
<?php
// autor: David Amorós Sendra <damoros2003@hotmail.com>
require('HolaMundo.php');

print "Introduce tu nombre:";
$nombre = trim(fgets(STDIN));
print new HolaMundo($nombre);
?>

```

Ahora agregar los cambios en staging y sin hacer un commit porque vamos a continuar con el rebase.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git add lib/hola.php
david@david-VirtualBox:~/Escritorio/curso-de-git$ git status
rebase interactivo en progreso; sobre cad3052
Los últimos comandos realizados (2 comandos realizados):
  pick d9d17ad Añadida la clase HolaMundo
  pick aa4c7fb hola usa la clase HolaMundo
No quedan más comandos.
Estás aplicando un rebase de la rama 'hola' sobre 'cad3052'.
  (todos los conflictos corregidos: ejecuta "git rebase --continue")

Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    modificados:   lib/hola.php

david@david-VirtualBox:~/Escritorio/curso-de-git$ git rebase --continue
[HEAD desacoplado 1f1d087] hola usa la clase HolaMundo
  1 file changed, 3 insertions(+), 1 deletion(-)
Rebase aplicado satisfactoriamente y actualizado refs/heads/hola.
```

Ahora si hacemos git hist –all veremos un arbol completamente diferente.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git hist --all
* 1f1d087 2024-01-22 | hola usa la clase HolaMundo (HEAD -> hola) [%]
| an]
* ddc5791 2024-01-22 | Añadida la clase HolaMundo [%]
| an]
* cad3052 2024-01-22 | Programa interactivo (master) [%]
| an]
* 177b9bf 2024-01-22 | Añadido README.md [%]
| an]
* d2e9df2 2024-01-22 | Movido hola.php a lib [%]
| an]
* 5777d66 2024-01-22 | Añadido el autor del programa y su email [%]
| an]
* 9589168 2024-01-22 | añadiendo git ignore (tag: v1) [%]
| an]
* 1873739 2024-01-22 | Se añade un comentario al cambio del valor por defecto (tag: v1-beta) [%]
| an]
* 6b585b9 2024-01-22 | Se añade un parametro por defecto [%]
| an]
* ed6cf34 2024-01-22 | Parametrización del programa [%]
| an]
* 8f40c7d 2024-01-22 | Creación del proyecto [%]
| an]
```

3. Mezclar en la rama master

Ya hemos realizado todos los cambios en la rama hola y ahora queremos llevar todos los cambios que hemos realizado a la principal(master) vamos a utilizar git checkout master para situarnos en la rama master y luego haremos un git merge hola para fusionarlo.

```

david@david-VirtualBox:~/Escritorio/curso-de-git$ git checkout master
Cambiado a rama 'master'
david@david-VirtualBox:~/Escritorio/curso-de-git$ git merge hola
Actualizando cad3052..1f1d087
Fast-forward
 lib/HolaMundo.php | 17 ++++++-----+
 lib/hola.php       |   4 +-+-
 2 files changed, 20 insertions(+), 1 deletion(-)
  create mode 100644 lib/HolaMundo.php
david@david-VirtualBox:~/Escritorio/curso-de-git$ git hist --all
* 1f1d087 2024-01-22 | hola usa la clase HolaMundo (HEAD -> master, hola) [%]
| an]
* ddc5791 2024-01-22 | Añadida la clase HolaMundo [%]
| an]
* cad3052 2024-01-22 | Programa interactivo [%]
| an]
* 177b9bf 2024-01-22 | Añadido README.md [%]
| an]
* d2e9df2 2024-01-22 | Movido hola.php a lib [%]
| an]
* 5777d66 2024-01-22 | Añadido el autor del programa y su email [%]
| an]
* 9589168 2024-01-22 | añadiendo git ignore (tag: v1) [%]
| an]
* 1873739 2024-01-22 | Se añade un comentario al cambio del valor por defecto (tag: v1-beta) [%]
| an]
* 6b585b9 2024-01-22 | Se añade un parametro por defecto [%]
| an]
* ed6cf34 2024-01-22 | Parametrización del programa [%]
| an]
* 8f40c7d 2024-01-22 | Creación del proyecto [%]
| an]

```

Si nos fijamos cuando hacemos merge master pone que la fusión se ha realizado a sido fast-forward. Esta fusión tiene el problema de que no deja rastro de fusión y se recomienda utilizar otro comando (--no -ff) para que se vea que se ha fusionado una rama con una otra.

Vamos hacer un reset al commit Programa interactivo y haremos el merge usando el anterior comando.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git reset --hard cad3052
HEAD está ahora en cad3052 Programa interactivo
```

```

david@david-VirtualBox:~/Escritorio/curso-de-git$ git merge -m "Aplicando los cambios de la rama hola" --no-ff hola
Merge made by the 'ort' strategy.
 lib/HolaMundo.php | 17 ++++++-----+
 lib/hola.php       |   4 +-+-
 2 files changed, 20 insertions(+), 1 deletion(-)
  create mode 100644 lib/HolaMundo.php
david@david-VirtualBox:~/Escritorio/curso-de-git$ git hist --all
* 82286d0 2024-01-22 | Aplicando los cambios de la rama hola (HEAD -> master) [%]
| an]
* 1f1d087 2024-01-22 | hola usa la clase HolaMundo (hola) [%]
| an]
* ddc5791 2024-01-22 | Añadida la clase HolaMundo [%]
| an]
* cad3052 2024-01-22 | Programa interactivo [%]
| an]
* 177b9bf 2024-01-22 | Añadido README.md [%]
| an]
* d2e9df2 2024-01-22 | Movido hola.php a lib [%]
| an]
* 5777d66 2024-01-22 | Añadido el autor del programa y su email [%]
| an]
* 9589168 2024-01-22 | añadiendo git ignore (tag: v1) [%]
| an]
* 1873739 2024-01-22 | Se añade un comentario al cambio del valor por defecto (tag: v1-beta) [%]
| an]
* 6b585b9 2024-01-22 | Se añade un parametro por defecto [%]
| an]
* ed6cf34 2024-01-22 | Parametrización del programa [%]
| an]
* 8f40c7d 2024-01-22 | Creación del proyecto [%]
| an]

```

GITHUB

Creación de key publica/privada

Si no tienes una cuenta en github tienes que entras en <https://github.com/> y crearte una cuenta.

Los servidores de git utilizan una autentificación mediante claves publicas llamadas SSH, cada usuario tiene uno que se genera desde la cuenta.

Vamos a crear una nueva cuenta para ello utilizamos el comando ssh-key -t rsa -C "lo que queramos".

```
amj
david@david-VirtualBox:~/Escritorio/curso-de-git$ ssh-keygen -t rsa -C "Cuenta Thinstation"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/david/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/david/.ssh/id_rsa
Your public key has been saved in /home/david/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:jEeFmqlpm9k6sFeXdc6Tfju9f4hL3Laq3LW5dbIdwJM Cuenta Thinstation
The key's randomart image is:
+---[RSA 3072]---+
|   . . . |
| + .. . |
| o o . |
| + . . o . |
| . =. So + E |
| . + 0.0 .=.o |
| 0 0 . .0o=+o |
| . + . 0oo+0= |
| . . . o++B=+ |
+---[SHA256]---+
david@david-VirtualBox:~/Escritorio/curso-de-git$
```

Ir a la ruta /.ssh y buscar el archivo id_rsa.pub y copiar lo que hay dentro.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ cd
david@david-VirtualBox:~$ cd .ssh/
david@david-VirtualBox:~/ssh$ ls
id_rsa id_rsa.pub known_hosts known_hosts.old
david@david-VirtualBox:~/ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAQABAAAQgQCT7UM70906AhxurUytP58zw10pWsfG8KbLgNODIkaFo/X9BEijmQeI/nxc/kesCckArgKsua2KhhRw6/yLzOfU2/K4YzLzq68LNihn0z4k68HP4aj
FdHHT37tA6Q5v25aoqmUxIEgy40HQ7WY17Y5TvQ4/VdGkg/s4VHmxamGwSRaxdJ8Meq7se+qAokiYkAwqKL8/R1jyW0SqHpi/p53xIV7G64BZPSTJ6dcZxKYLqF04ZCu5wRzNYGpoL+IuigrVyxEP
R4bImTL0iyd751BY5AkR8VbsEb1nej4gWUpbcse/SbUYMT8M6BZNSS5rf1M61RZ2I3Bl1ds0eL7vD1arH8gohACFex5ASxje/DGNT5Fb74Zl3NKvPo1Yc4JPz4N4UJ5GAt0QToaS2Kp8iyYgf00kR
DITpdwdNnRm4puh+amZ5JLDS413pdJ500FTfJFrv9cyuFpLn++zcsFtqCtkOYf3kU74+kW9YVA1Jgy5u4DSv3axyfPo0Rk= Cuenta Thinstation
david@david-VirtualBox:~/ssh$
```

Ahora vamos a Github y accedemos con nuestra cuenta, vamos a settings y dentro de settings vamos a SSH and GPG keys y creamos una nueva SSH key. En el titulo ponemos lo que queramos y en la descripción lo que has copiado previamente en el archivo id_rsa.pub.

The screenshot shows the GitHub 'SSH keys' section. At the top right is a green button labeled 'New SSH key'. Below it, a message says 'This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.' Under the heading 'Authentication keys', there is a card for a key named 'David@gitEjercicio'. The card includes the key fingerprint 'SHA256:jEeFmqlpm9k6sFeXdc6Tfju9f4hL3Laq3LW5dbIdwJM', the date 'Added on Jan 24, 2024', and the status 'Never used — Read/write'. To the right of the card is a red 'Delete' button. At the bottom of the page, there is a note: 'Check out our guide to [connecting to GitHub using SSH keys](#) or troubleshoot [common SSH problems](#)'.

Creación de un repositorio

Vamos a repositorios y le damos a crear nuevo repositorio(aparece arriba a la derecha en verde) y a este repositorio lo llamaremos taller-de-git y su estructura sera tal que así.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * Repository name *

 David09x /
 taller-de-git is available.

Great repository names are short and memorable. Need inspiration? How about [congenial-robot](#) ?

Description (optional)

 Public

Anyone on the internet can see this repository. You choose who can commit.

 Private

You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

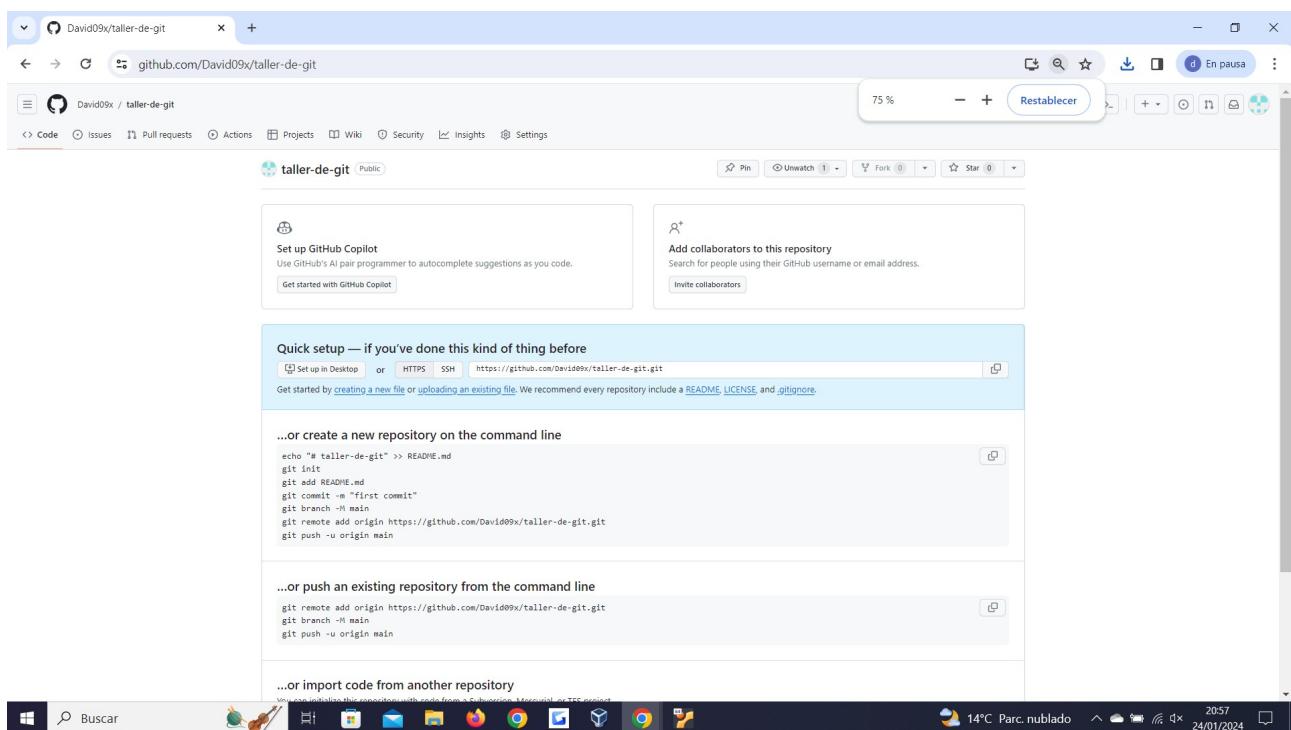
Choose a license

License: None ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)



Le damos abajo a continuar y nos aparece esto.



The screenshot shows the GitHub repository creation interface. At the top, it says "David09x/taller-de-git". Below that, there are sections for "Copilot", "Collaborators", and "Quick setup". The "Quick setup" section provides command-line instructions for initializing a repository:

```
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/David09x/taller-de-git.git
git push -u origin main
```

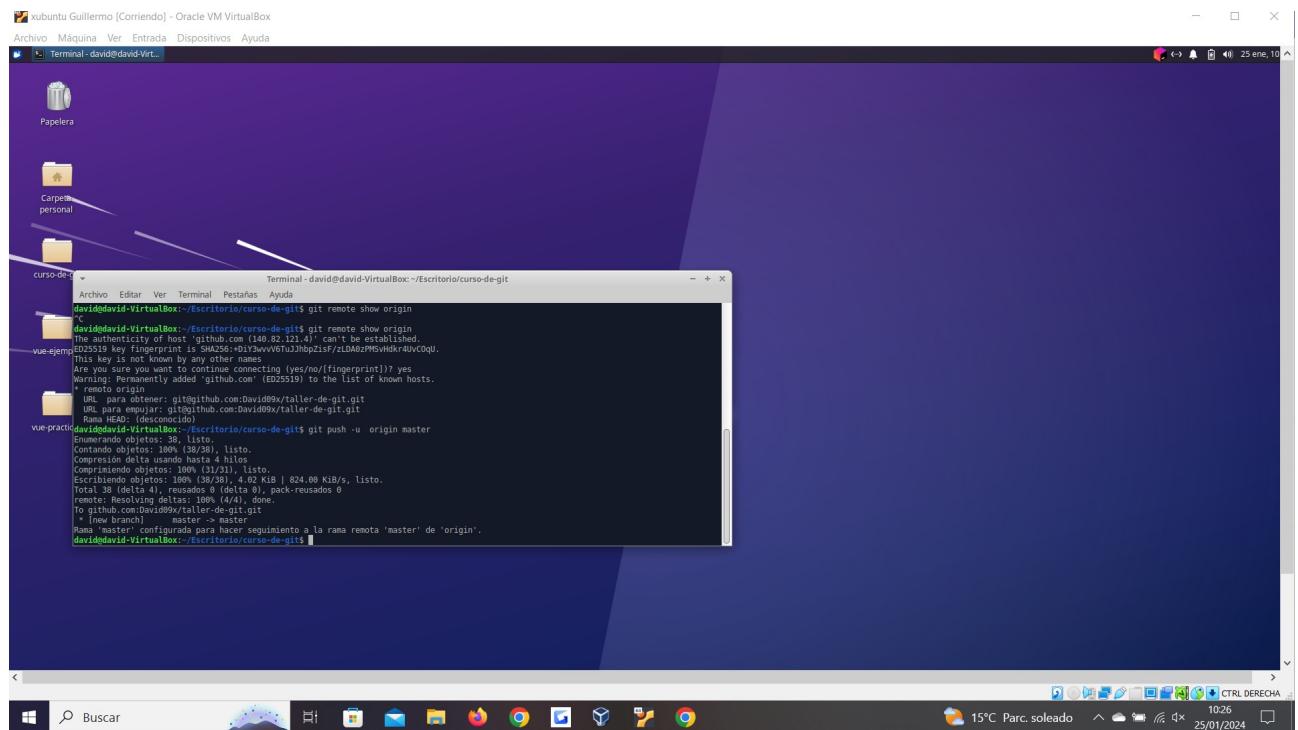
Below this, there are sections for pushing an existing repository and importing code from another repository. The bottom of the screen shows the Windows taskbar with various icons and the date/time.

Le damos para que nos de el path de como sería en SSH para poder subir nuestra carpeta previamente al repositorio.

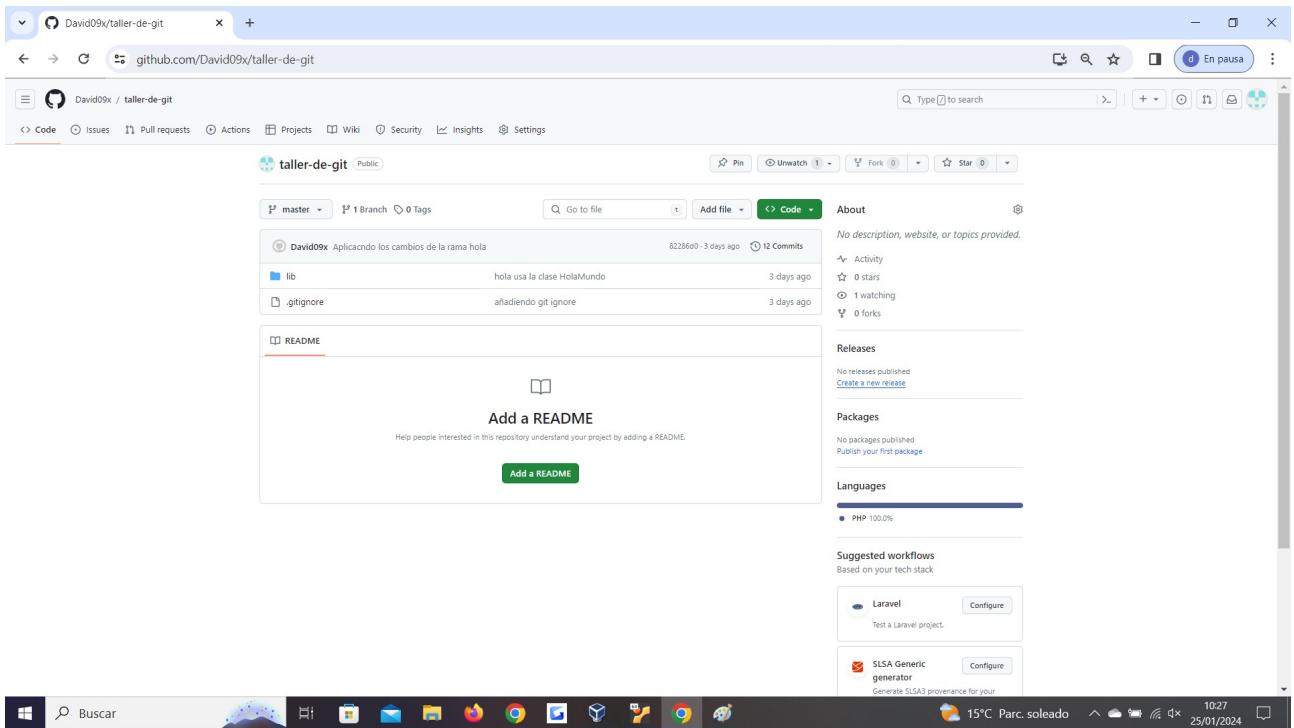
Ahora vamos a prepararnos para subir la carpeta nuestra al repositorio que nos hemos creado.

Para ellos vamos a usar el comando git remote add origin path_ssh y luego haremos un git push -u origin master que lo subira a la rama master.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git remote show origin
* remoto origin
  URL para obtener: git@github.com:David09x/taller-de-git.git
  URL para empujar: git@github.com:David09x/taller-de-git.git
  Rama HEAD: master
  Rama remota:
    master rastreada
  Rama local configurada para 'git pull':
    master fusiona con remoto master
  Referencia local configurada para 'git push':
    master publica a master (actualizado)
```



Ahora si vamos a github a al repositorio que hemos creado previamente veremos que están los archivos de la carpeta con los commits previos realizados.



Clonar un repositorio.

Si queremos clonar cualquier repositorio de github el comando sería git clone path_del_github

Ejemplo:

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git clone https://github.com/David09x/taller-de-git.git
```

Ramas remotas

El estado de nuestro proyecto a día de hoy debería de ser algo así.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git hist --all
* 82286d0 2024-01-22 | Aplicando los cambios de la rama hola (HEAD -> master, origin/master) [%]
|\ an]
| * 1f1d087 2024-01-22 | hola usa la clase HolaMundo (hola) [%]
| |\ an]
| | * ddc5791 2024-01-22 | Añadida la clase HolaMundo [%]
| |\ an]
| * cad3052 2024-01-22 | Programa interactivo [%]
| |\ an]
| * 177b9bf 2024-01-22 | Añadido README.md [%]
| |\ an]
| * d2e9df2 2024-01-22 | Movido hola.php a lib [%]
| |\ an]
| * 5777d66 2024-01-22 | Añadido el autor del programa y su email [%]
| |\ an]
| * 9589168 2024-01-22 | añadiendo git ignore (tag: v1) [%]
| |\ an]
| * 1873739 2024-01-22 | Se añade un comentario al cambio del valor por defecto (tag: v1-beta) [%]
| |\ an]
| * 6b585b9 2024-01-22 | Se añade un parametro por defecto [%]
| |\ an]
| * ed6cf34 2024-01-22 | Parametrización del programa [%]
| |\ an]
| * 8f40c7d 2024-01-22 | Creación del proyecto [%]
| |\ skinning
```

Podemos ver tambien la configuración de este repositorio de manera remota con el comando git remote show origin.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git remote show origin
* remoto origin
  URL para obtener: git@github.com:David09x/taller-de-git.git
  URL para empujar: git@github.com:David09x/taller-de-git.git
  Rama HEAD: master
  Rama remota:
    master rastreada
  Rama local configurada para 'git pull':
    master fusiona con remoto master
  Referencia local configurada para 'git push':
    master publica a master (actualizado)
```

Enviando Actualizaciones

Vamos a crear una fichero llamado LICENSE CON este formato.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ cat LICENSE
MIT License

Copyright (c) [year] [fullname]

Permission is hereby granted, free of charge, to any person obtaining a
copy
of this software and associated documentation files (the "Software"),
to deal
in the Software without restriction, including without limitation the
rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or
sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included
in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
IN THE
SOFTWARE.
```

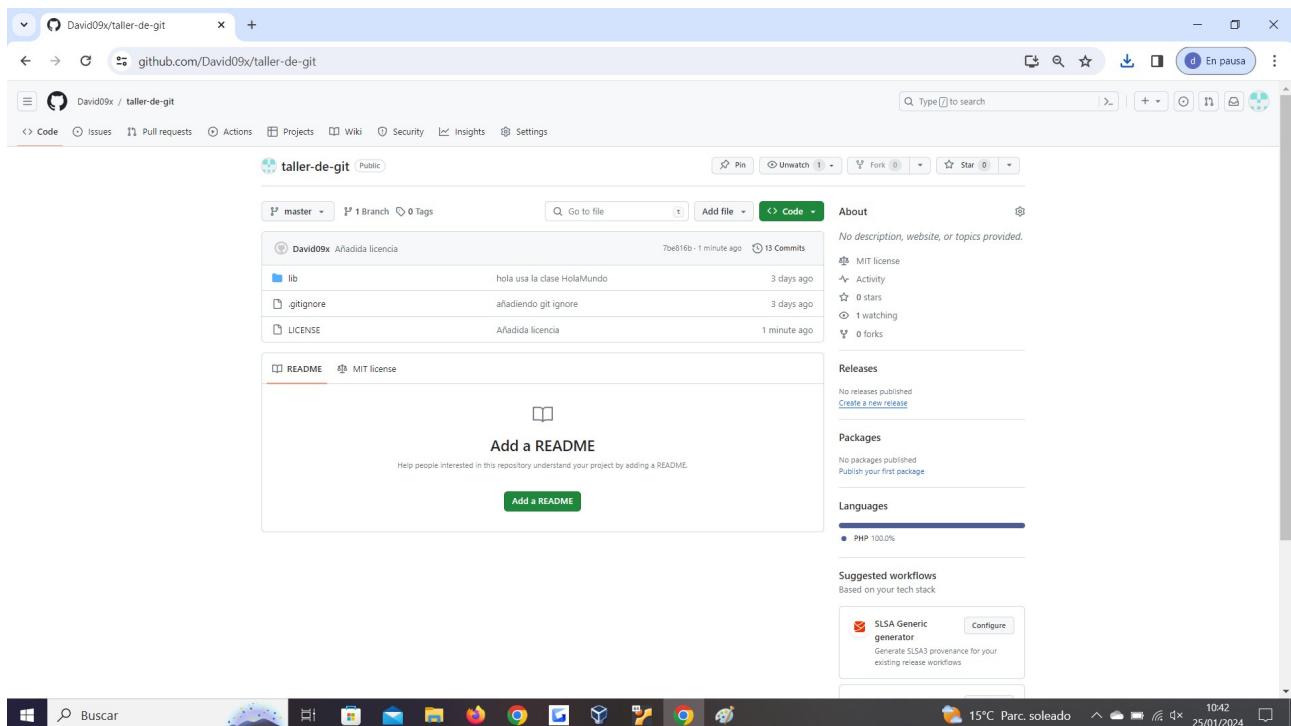
Ahora vamos a hacerle un commit y vemos como estarían la rama con los cambios realizados.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git add LICENSE
david@david-VirtualBox:~/Escritorio/curso-de-git$ git commit -m "Añadida licencia"
[master 7be816b] Añadida licencia
 1 file changed, 30 insertions(+)
 create mode 100644 LICENSE
```

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git hist --all
* 7be816b 2024-01-25 | Añadida licencia (HEAD -> master) [%]
| an]
* 82286d0 2024-01-22 | Aplicando los cambios de la rama hola (origin/master) [%]
| \
| * 1f1d087 2024-01-22 | hola usa la clase HolaMundo (hola) [%]
| | an]
| * ddc5791 2024-01-22 | Añadida la clase HolaMundo [%]
| / an]
* cad3052 2024-01-22 | Programa interactivo [%]
| an]
* 177b9bf 2024-01-22 | Añadido README.md [%]
| an]
* d2e9df2 2024-01-22 | Movido hola.php a lib [%]
| an]
* 5777d66 2024-01-22 | Añadido el autor del programa y su email [%]
| an]
* 9589168 2024-01-22 | añadiendo git ignore (tag: v1) [%]
| an]
* 1873739 2024-01-22 | Se añade un comentario al cambio del valor por defecto (tag: v1-beta) [%]
| an]
* 6b585b9 2024-01-22 | Se añade un parametro por defecto [%]
| an]
* ed6cf34 2024-01-22 | Parametrización del programa [%]
| an]
* 8f40c7d 2024-01-22 | Creación del proyecto [%]
| an]
```

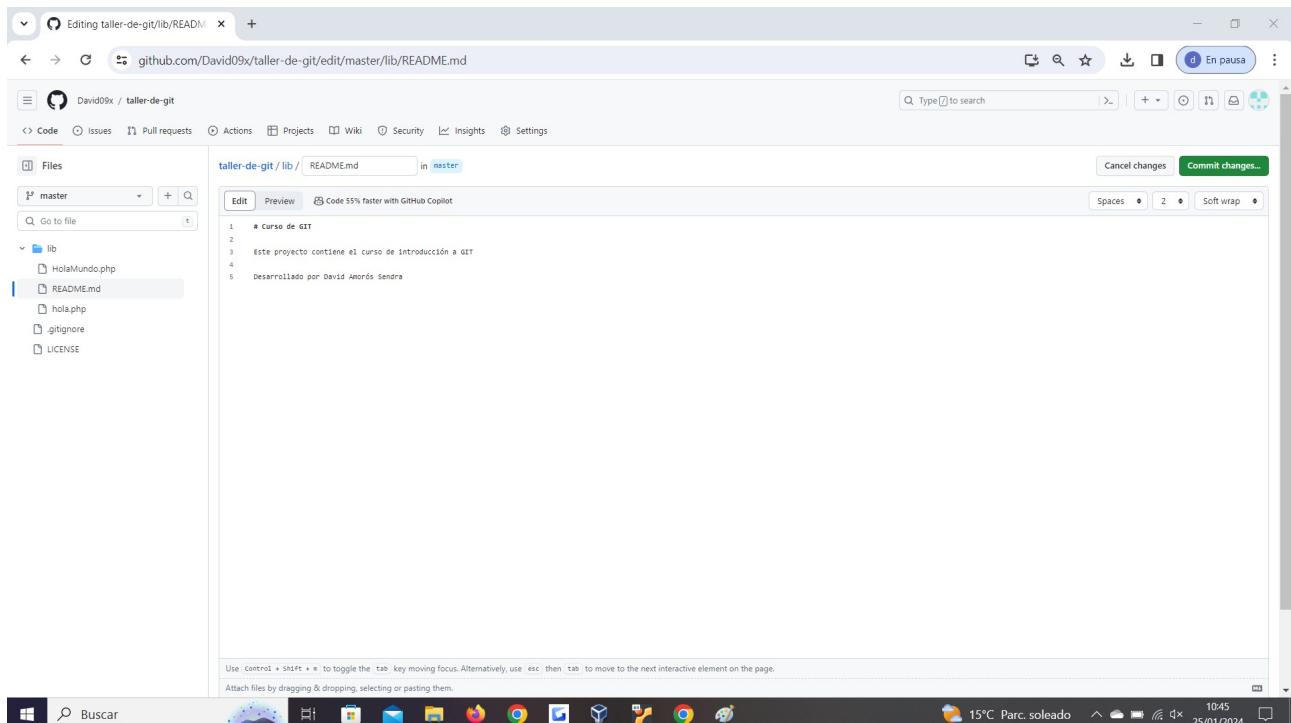
Si vamos a nuestro repositorio de github veremos que el archivo LICENSE no esta todavía subido, para ello vamos hacer un git push para que suba los ultimos cambios realizados.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git push
Enumerando objetos: 4, listo.
Contando objetos: 100% (4/4), listo.
Comprimiendo objetos: 100% (3/3), listo.
Escribiendo objetos: 100% (3/3), 944 bytes | 944.00 KiB/s, listo.
Total 3 (delta 0), reusados 0 (delta 0), pack-reusados 0
To github.com:David09x/taller-de-git.git
  82286d0..7be816b  master -> master
```

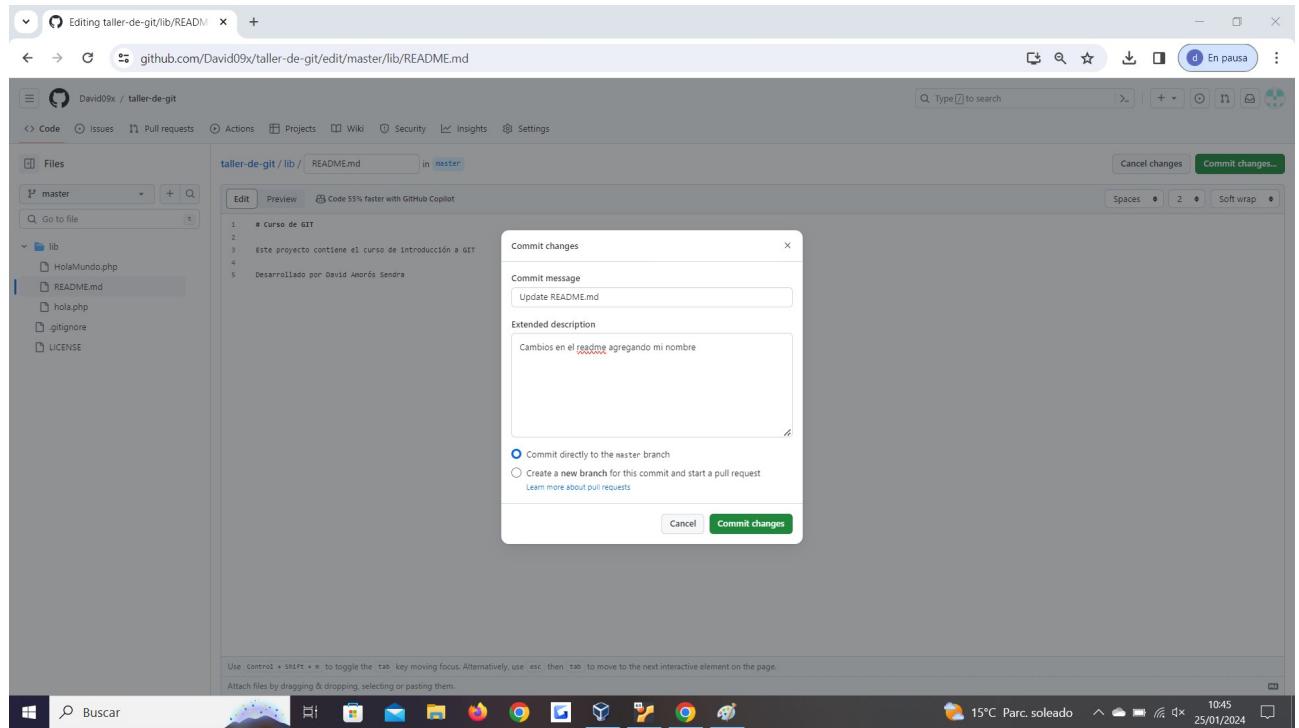


Recibiendo actualizaciones

Vamos a github y vamos a modificar el archivo README(todo desde la pagina web) y ponemos al final nuestro nombre.



Le damos arriba a commit change y ponemos update README.md y una pequeña descripción de lo que hemos realizado y le volvemos a dar a commit changes.



Queremos que los cambios realizados pues vamos hacer git fetch para obtener la ultima versión del repositorio.

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git fetch
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
Desempaquetando objetos: 100% (4/4), 1.11 KiB | 25.00 KiB/s, listo.
Desde github.com:David09x/taller-de-git
  7be816b..900bf53  master      -> origin/master
```

```
david@david-VirtualBox:~/Escritorio/curso-de-git$ git hist --all
* 900bf53 2024-01-25 | Update README.md (origin/master) [%]
| an]
* 7be816b 2024-01-25 | Añadida licencia (HEAD -> master) [%]
| an]
* 82286d0 2024-01-22 | Aplicacndo los cambios de la rama hola [%]
| an]
* 1f1d087 2024-01-22 | hola usa la clase HolaMundo (hola) [%]
| an]
* ddc5791 2024-01-22 | Añadida la calse HolaMundo [%]
| an]
* cad3052 2024-01-22 | Programa interactivo [%]
| an]
* 177b9bf 2024-01-22 | Añadido README.md [%]
| an]
* d2e9df2 2024-01-22 | Movido hola.php a lib [%]
| an]
* 5777d66 2024-01-22 | Añadido el autor del programa y su email [%]
| an]
* 9589168 2024-01-22 | añadiendo git ignore (tag: v1) [%]
| an]
* 1873739 2024-01-22 | Se añade un comentario al cambio del valor por defecto (tag: v1-beta) [%]
| an]
* 6b585b9 2024-01-22 | Se añade un parametro por defecto [%]
| an]
* ed6cf34 2024-01-22 | Parametrización del programa [%]
| an]
* 8f40c7d 2024-01-22 | Creación del proyecto [%]
| an]
```

Ahora vemos el caos contrario que el origin/master esta por delante de la rama local para ello debemos usar git merge para fusionar las 2 ramas y que este lo ultimo y solucionar conflictos.

Como al utilizar los comandos git fetch para hacer cambios y git rebase para mezclar ramas estan muy asociados, git ofrece una posibilidad para ahorrar pasos que se utiliza el comando git pull para ello vamos a probar modificando el README.

```
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ sudo nano README.md
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git add README.md
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git commit -m "Indicado que se realiza en el ASL"
[master e9fa719] Indicado que se realiza en el ASL
 1 file changed, 1 insertion(+), 1 deletion(-)
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ cat README.md
# Curso de GIT

Este proyecto contiene el curso de introducción a GIT del Aula de Software libre

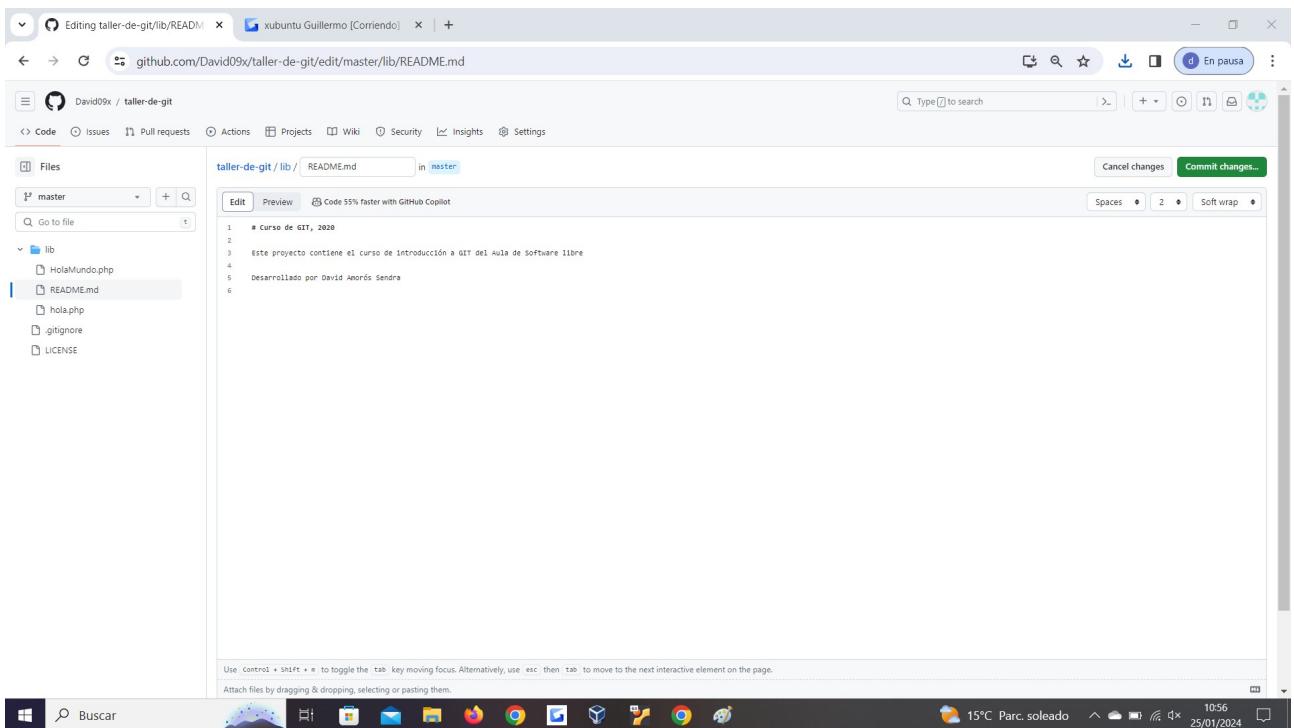
Desarrollado por David Amorós Sendra
```

```
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git hist --all
* e9fa719 2024-01-25 | Indicado que se realiza en el ASL (HEAD -> master) [%]
| an]
* 900bf53 2024-01-25 | Update README.md (origin/master) [%]
| an]
* 7be816b 2024-01-25 | Añadida licencia [%]
| an]
* 82286d0 2024-01-22 | Aplicacndo los cambios de la rama hola [%]
|\ an]
| * 1fld087 2024-01-22 | hola usa la clase HolaMundo (hola) [%]
| | an]
| * ddc5791 2024-01-22 | Añadida la calse HolaMundo [%]
|| an]
* cad3052 2024-01-22 | Programa interactivo [%]
| an]
* 177b9bf 2024-01-22 | Añadido README.md [%]
| an]
* d2e9df2 2024-01-22 | Movido hola.php a lib [%]
| an]
* 5777d66 2024-01-22 | Añadido el autor del programa y su email [%]
| an]
* 9589168 2024-01-22 | añadiendo git ignore (tag: v1) [%]
| an]
* 1873739 2024-01-22 | Se añade un comentario al cambio del valor por defecto (tag: v1-beta) [%]
| an]
* 6b585b9 2024-01-22 | Se añade un parametro por defecto [%]
| an]
* ed6cf34 2024-01-22 | Parametrización del programa [%]
| an]
* 8f40c7d 2024-01-22 | Creación del proyecto [%]
| an]
```

Problemas de sincronización.

Si intentamos hacer un git push lo lógico es que no nos deje porque el repositorio remoto se ha actualizado y nosotros todavía no hemos recibido los cambios. Vamos ha provocar una situación donde podamos ver eso, para ello modificamos el archivo README tanto a nivel local como a nivel web.

En github pondremos Curso de GIT, 2020.



y en local pondremos Curso de GIT, febrero y hacemos commits en los 2 lados.

```
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ cat README.md
# Curso de GIT, febrero

Este proyecto contiene el curso de introducción a GIT del Aula de Software libre

Desarrollado por David Amorós Sendra
```

Ahora vamos a hacer un pull con rebase y tendría que salir un conflicto ya que el problema es tener 2 versiones de diferentes de README, una solución sería fusionar los 2 y entrar al archivo README y elegir cual de los 2 quieras y hacer un push de nuevo.

```
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git pull --rebase
Auto-fusionando lib/README.md
CONFLICTO (contenido): Conflicto de fusión en lib/README.md
error: no se pudo aplicar 0cc9128... Añadido el mes al README
ayuda: Resuelve todos los conflictos manualmente, marcalos como resueltos con
ayuda: "git add/rm <archivos conflictados>", luego ejecuta "git rebase --continue".
ayuda: Puedes saltarte este commit: ejecuta "git rebase --skip".
ayuda: Para abortar y volver al estado anterior a "git rebase", ejecuta "git rebase --abort".
No se pudo aplicar 0cc9128... Añadido el mes al README
```

```
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git merge master
error: No es posible hacer merge porque tienes archivos sin fusionar.
ayuda: Corrige los conflictos en el árbol de trabajo y luego usa 'git add/rm <archivo>',
ayuda: como sea apropiado, para marcar la resolución y realizar un commit.
fatal: Saliendo porque existe un conflicto sin resolver.
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git push
fatal: Actualmente no estás en una rama.
Para empujar la historia que lleva al estado actual
(HEAD desacoplado), usa

    git push origin HEAD:<nombre-de-rama-remota>

david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git push origin HEAD:master
Everything up-to-date
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ cat README.md
<<<<< HEAD
# Curso de GIT, 2020
=====
# Curso de GIT, febrero
>>>>> 0cc9128 (Añadido el mes al README)

Este proyecto contiene el curso de introducción a GIT del Aula de Software libre
```

```
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ sudo nano README.md
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git commit -m "conflicto resultado"
U lib/README.md
error: No es posible realizar un commit porque tienes archivos sin fusionar.
ayuda: Corrigelos en el árbol de trabajo y entonces usa 'git add/rm <archivo>'.
ayuda: como sea apropiado, para marcar la resolución y realizar un commit.
fatal: Saliendo porque existe un conflicto sin resolver.
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git add README.md
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git commit -m "conflicto resultado"
[HEAD desacoplado 8571df7] conflicto resultado
 1 file changed, 1 insertion(+)
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git push
fatal: Actualmente no estás en una rama.
Para empujar la historia que lleva al estado actual
(HEAD desacoplado), usa

    git push origin HEAD:<nombre-de-rama-remota>

david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ cat README.md

# Curso de GIT, 2020

Este proyecto contiene el curso de introducción a GIT del Aula de Software libre

Desarrollado por David Amorós Sendra
```

```
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git rebase --continue
Rebase aplicado satisfactoriamente y actualizado refs/heads/master.
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git status
En la rama master
Tu rama está adelantada a 'origin/master' por 1 commit.
  (usa "git push" para publicar tus commits locales)

nada para hacer commit, el árbol de trabajo está limpio
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git push
Enumerando objetos: 7, listo.
Contando objetos: 100% (7/7), listo.
Compresión delta usando hasta 4 hilos
Comprimiendo objetos: 100% (4/4), listo.
Escribiendo objetos: 100% (4/4), 398 bytes | 199.00 KiB/s, listo.
Total 4 (delta 2), reusados 0 (delta 0), pack-reusados 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:David09x/taller-de-git.git
  c7a7446..8571df7  master -> master
```

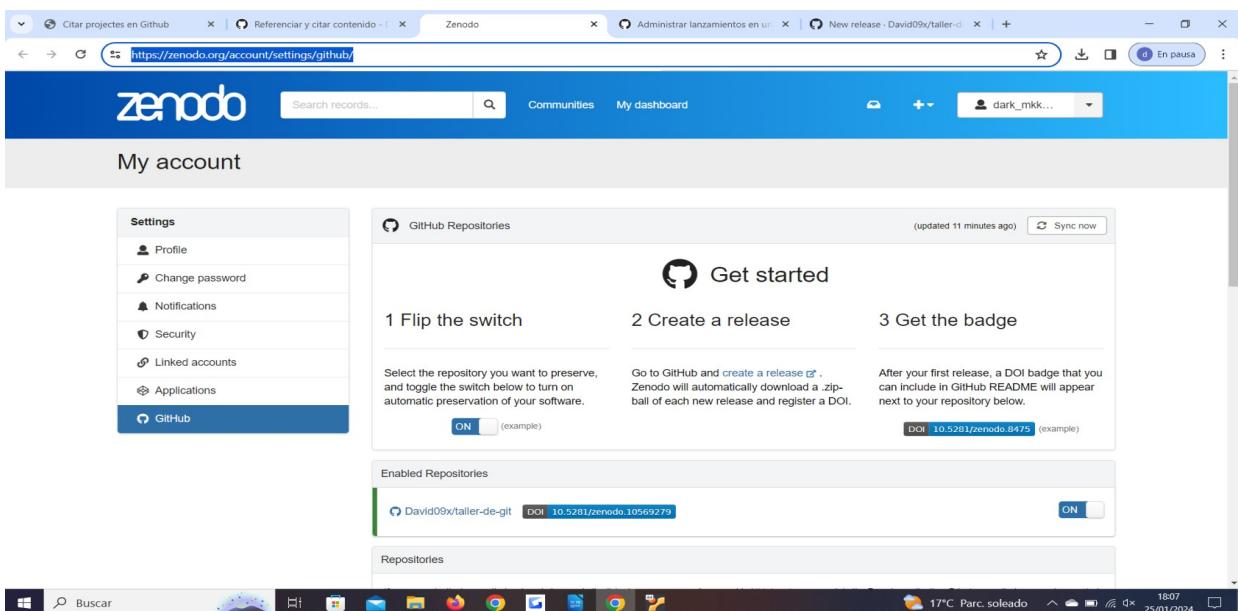
CITAR PROYECTOS EN GITHUB

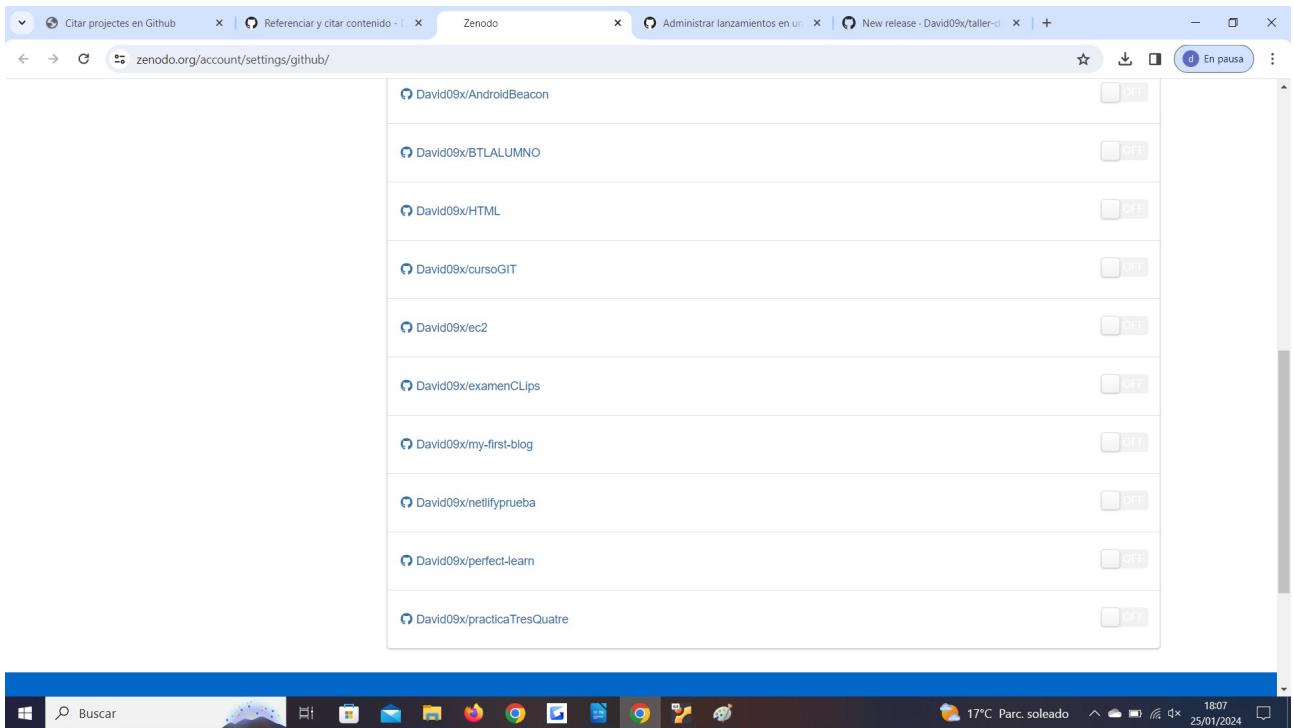
Entrar en zenodo.

Primero vamos a zenodo.org y nos pedirá el inicio de sesión y iniciamos la sesión con GITHUB-
Nos aparece los términos y la información de acceso y damos click a autorizar.

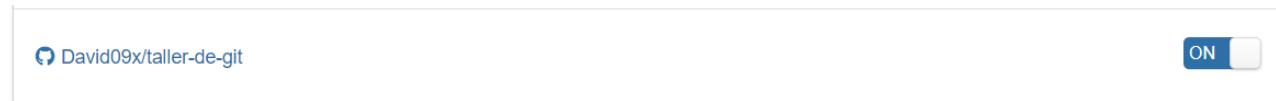
Página de Zenodo y Github

Después vamos a la página de zenodo.github y tendrían que aparecer todo los repositorios





Vemos que cada repositorio tiene un botón a la derecha pues activamos el botón del repositorio que queramos.



Publicación por Github

Vamos ahora a Github al repositorio en cuestión y a la derecha veremos que aparece Release pues clickeamos en él.

The screenshot shows a GitHub repository page for 'David09x/conflicto resultado'. The repository has 1 branch and 0 tags. The 'Code' tab is selected. On the right, there is an 'About' section with a note: 'No description, website, or topics provided.' It also shows '17 Commits' and a file list: 'lib' (conflicto resultado, 6 hours ago), '.gitignore' (añadiendo git ignore, 3 days ago), and 'LICENSE' (Añadida licencia, 7 hours ago). Below the repository details, there is a 'README' section with a 'Add a README' button and a note: 'Help people interested in this repository understand your project by adding a README.' To the right, there is a 'Releases' section with a note: 'No releases published' and a link 'Create a new release'. Below that is a 'Packages' section with a note: 'No packages published' and a link 'Publish your first package'. At the bottom is a 'Languages' section showing 'PHP 100.0%'.

Dentro de él como es nuevo le damos hacer una nueva publicación y lo rellenamos, debería de ser algo así.

The screenshot shows the GitHub 'Create a new release' interface. At the top, there are buttons for 'Choose a tag' (dropdown), 'Target: master' (dropdown), and 'Generate release notes'. Below these are fields for 'Title' (set to 'prueba git') and 'Description' (containing the text 'Esto es una prueba de release del ejercicio de GITHUB de Guillermo'). On the right, there are sections for 'Tagging suggestions' (with a note about prefixing version names with 'v'), 'Semantic versioning' (with a link to learn more), and a note about automatically labeling the release as the latest. At the bottom, there are checkboxes for 'Set as a pre-release' (unchecked) and 'Set as the latest release' (checked), followed by 'Publish release' and 'Save draft' buttons.

Etiqueta DOI

Si todo esta correcto le damos a publish release y volviendo a la pagina de Zenoro tendria que aparecer en ese repositorio una etiqueta DOI.

The screenshot shows the Zenodo 'Enabled Repositories' interface. It lists a single repository: 'David09x/taller-de-git' with a DOI of '10.5281/zenodo.10569279'. To the right of the DOI is a toggle switch labeled 'ON'.

FLUJO DE TRABAJO EN GITHUB

Incidencias.

Las incidencias en GitHub son registros que se utilizan para rastrear y gestionar problemas, sugerencias, tareas o cualquier otro tipo de colaboración en un proyecto.

The screenshot shows the GitHub interface for creating a new issue. At the top, there's a navigation bar with links for Code, Issues (which is highlighted), Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation bar, there's a section to 'Add a title' with a text input field labeled 'Title'. To the right of this, there are settings for 'Assignees' (set to 'No one—assign yourself') and 'Labels' (set to 'None yet'). Further down, there's a rich text editor for 'Add a description' with a toolbar for Write, Preview, H, B, I, etc., and a text area containing placeholder text 'Add your description here...'. To the right of the editor, there are sections for 'Projects' (set to 'None yet'), 'Milestone' (set to 'No milestone'), and 'Development' (described as showing branches and pull requests linked to this issue). Below the editor, there are buttons for 'Markdown is supported' and 'Paste, drop, or click to add files'. A large green 'Submit new issue' button is at the bottom right. At the very bottom of the page, there's a footer with the GitHub logo and copyright information: '© 2024 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information'.

Crear una Rama

Vamos a crear una nueva rama y la llamaremos tu_nombre/feature-1/create-changelog y vamos a crear un Archivo un archivo denominado AUTORES.md.

```
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git checkout -b david/feature-1/create-changelog
Cambiado a nueva rama 'david/feature-1/create-changelog'
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ sudo nano AUTORES.md
[sudo] contraseña para david:
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git add AUTORES.md
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ cat AUTORES.md
#Autores

David Amorós <damoros2003@hotmail.com>
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git commit -m "Añadido fichero de autores"
[david/feature-1/create-changelog 86145ce] Añadido fichero de autores
 1 file changed, 3 insertions(+)
 create mode 100644 lib/AUTORES.md
```

Al crearlo y darle el formato le hacemos un commit.

Vamos a intentar hacer un push y nos dara un fallo.

```
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git push
fatal: La rama actual david/feature-1/create-changelog no tiene una rama upstream.
Para realizar un push de la rama actual y configurar el remoto como upstream, usa
git push --set-upstream origin david/feature-1/create-changelog
```

Esto es porque al ser una rama nueva no sabe donde hacer push, por ello tenemos que hacerle un push utilizando origin.

Vamos a probar hacer un push pero utilizando el comando git push -u origin tu_nombre/feature-1/create-changelog.

```
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git push -u origin david09/feature-1/create-changelog
Enumerando objetos: 6, listo.
Contando objetos: 100% (6/6), listo.
Comprimiendo objetos: 100% (4/4), listo.
Escribiendo objetos: 100% (4/4), 470 bytes | 235.00 KiB/s, listo.
Total 4 (delta 1), reusados 0 (delta 0), pack-reusados 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'david09/feature-1/create-changelog' on GitHub by visiting:
remote:     https://github.com/David09x/taller-de-git/pull/new/david09/feature-1/create-changelog
remote:
To github.com:David09x/taller-de-git.git
 * [new branch]      david09/feature-1/create-changelog -> david09/feature-1/create-changelog
Rama 'david09/feature-1/create-changelog' configurada para hacer seguimiento a la rama remota 'david09/feature-1/create-changelog' de 'origin'.
```

Ahora la rama si se ha subido y podemos hacer un PR(pull request) para ello tenemos que ir a la pagina que nos da en remote(4 remote).

The screenshot shows the GitHub 'Open a pull request' interface. At the top, it says 'base: master' and 'compare: david09/feature-1/create-changelog'. A green checkmark indicates 'Able to merge'. Below this, there's a title field with 'Añadido fichero de autores' and a description field with 'Add your description here...'. To the right, there are sections for 'Reviewers' (No reviews), 'Assignees' (No one—assign yourself), 'Labels' (None yet), 'Projects' (None yet), 'Milestone' (No milestone), and 'Development' (Use Closing keywords in the description to automatically close issues). At the bottom is a large green 'Create pull request' button.

de titulo le ponemos Añadido archivo de AUTORES y en la description ponemos Closes #1 y damos a create pull y se puede visualizar lo que hemos creado.

The screenshot shows a GitHub pull request interface. At the top, it says "David09x wants to merge 1 commit into master from david09/feature-1/create-changelog". Below this, there are tabs for Conversation (0), Commits (1), Checks (0), and Files changed (1). A comment from David09x is shown, stating "Closes #1" and "Añadido fichero de autores". The commit hash is 88f5168. On the right side, there are review status indicators: Revi, No r, Still, Assi, and No r.

Crear commits

Vamos a modificar el archivo AUTORES.md y tiene que quedar tal que así.

```
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ cat AUTORES.md
#AUTORES
David Amorós Sendra <damoros2003@hotmail.com>
John Doe
```

Hacemos un git commit y luego un push para subirlo.

```
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git commit -am "Actualizado AUTORES.md"
[david09/feature-1/create-changelog dfaaa9e0] Actualizado AUTORES.md
 1 file changed, 1 insertion(+)
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git push
Enumerando objetos: 7, listo.
Contando objetos: 100% (7/7), listo.
Compresión delta usando hasta 4 hilos
Comprimiendo objetos: 100% (4/4), listo.
Escribiendo objetos: 100% (4/4), 477 bytes | 477.00 KiB/s, listo.
Total 4 (delta 1), reusados 0 (delta 0), pack-reusados 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:David09x/taller-de-git.git
 88f5168..dfaaa9e0  david09/feature-1/create-changelog -> david09/feature-1/create-changelog
```

Si vamos a Github dentro del repositorio y accedemos a commits nos saldra el commit realizado.

The screenshot shows a GitHub commit history. It lists a single commit: "Actualizado AUTORES.md" by David09x committed 1 minute ago. The commit hash is dfaaa9e0. There are options to copy the link or open the diff.

Lanzar Aplicación

Cada vez que hemos terminado de usar la rama ya podemos fusionarla con el master. Esta manera de hacerlo se puede hacer de 3 maneras

Mediante un merge commit(El que yo realice).

Esta manera de hacerlo es estar en la rama master(si no estas usando el comando git checkout master) y haciendo un merge con la rama

git merge --no-ff el_nombre_de_la_rama

Terminamos haciendo un git push para que lo suba.

```
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git checkout master
Cambiado a rama 'master'.
Tu rama está actualizada con 'origin/master'.
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git merge --no-ff david09/feature-1/create-changelog
Merge made by the 'ort' strategy.
 lib/AUTORES.md | 4 +++
 1 file changed, 4 insertions(+)
 create mode 100644 lib/AUTORES.md
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git push
Enumerando objetos: 1, listo.
Contando objetos: 100% (1/1), listo.
Escribiendo objetos: 100% (1/1), 242 bytes | 242.00 KiB/s, listo.
Total 1 (delta 0), reusados 0 (delta 0), pack-reusados 0
To github.com:David09x/taller-de-git.git
 8571df7..d2a5fad master -> master
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$
```

Mediante un rebase y merge

Usando git rebase master y luego nos situamos dentro de la rama master con el comando checkout master.

Despues igual que el anterior git merge –no-ff el_nombre_de_la_rama y hacemos push para que lo suba.

Mediante un squash commit y un merge

Nos situamos en la rama master con el comando git checkout master y hacemos un merge pero utilizando squash.

Git merge –squash tu_nombre_rama.

Y despues hacemos un git push para subirlo.

Sincronizar

Hemos cambiado de repositorio pero nuestra rama master no tiene los cambios del origin. Para ello lo vamos a sincronizar y como la rama ya no la necesitaremos la borraremos.

Nos situamos en la rama master utilizando git checkout master y hacemos un git pull –rebase –autostash, con esto sincronizamos la rama con la master y esta al dia.

Como ya tiene lo mismo la rama master que la rama actuar en la que estabamos trabajando al no necesitarla la podemos borrar utilizando git branch -D tu_nombre_rama.

```
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git checkout master
Ya en 'master'.
Tu rama está actualizada con 'origin/master'.
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git pull --rebase --autostash
Desde github.com:David09x/taller-de-git
 * [nuevo tag]      v1.1      -> v1.1
Ya está actualizado.
david@david-VirtualBox:~/Escritorio/curso-de-git/lib$ git branch -D david09/feature-1/create-changelog
Eliminada la rama david09/feature-1/create-changelog (era dfaa9e0).
```