

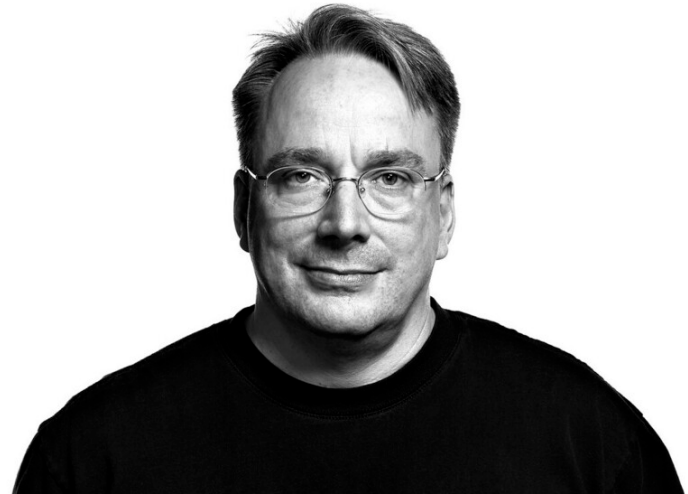
Bloque de GIT



David Bernal González

¿Qué es GIT?

- Sistema de control de versiones:
 - Un software que se encarga de controlar/administrar las distintas versiones de un programa.
- Es OpenSource (código abierto) y fue creado por Linus Tolward (el padre del kernel de Linux).



Beneficios de GIT

- Nos facilita que varias personas puedan trabajar con un mismo proyecto.
- Revertir los cambios realizados es un “snapshot” (captura del estado del proyecto actual). Ya que GIT es una especie de máquina del tiempo.
- Nos proporciona un registro mediante al que podemos ver quién y cuando ha realizado un cierto cambio.

GIT nos permite realizar “snapshots” (instantáneas) del estado de nuestro proyecto para posteriormente poder desplazarnos entre ellas como si de una máquina del tiempo se tratase.



Instalando GIT

- Para instalar GIT vamos a su [web oficial](#)
- En la que también tenéis acceso al libro “[Pro GIT](#)” de forma totalmente gratuita

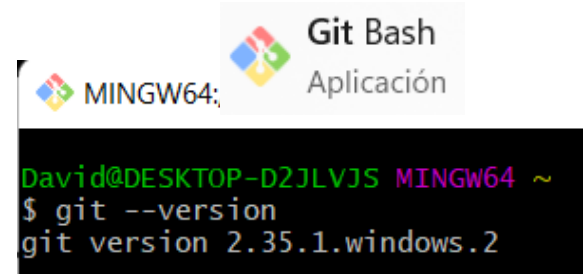


Pro Git by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on [Amazon.com](#).



GIT Bash

- Cuando instalamos GIT (en Windows), se nos instalará una especie de terminal Git Bash.
- Git Bash, es la forma más completa que tener para trabajar con GIT. Pese a ello, existen otras formas de trabajar con GIT mediante a una interfaz gráfica podéis ver más al respecto [en el siguiente enlace](#)
- Git Bash, al ser una emulación de la terminal Linux/UNIX, utiliza los comandos de Linux.



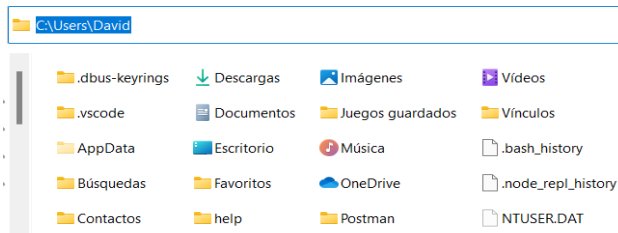
Desplazándonos con GitBash

- **pwd:** imprime la ruta del directorio actual
- **ls:** listar el contenido del directorio actual

MINGW64:/c/Users/David

```
David@DESKTOP-D2JLVJS MINGW64 ~  
$ pwd  
/c/Users/David
```

**La ruta por defecto (raíz) ~
es el directorio del usuario**



MINGW64:/c/Users/David

```
David@DESKTOP-D2JLVJS MINGW64 ~  
$ ls  
AppData/  
'Configuración local'@  
Contacts/  
Cookies@  
'Datos de programa'@  
Desktop/  
Documents/
```

Desplazándonos con GitBash

■ **CD:** cambiar de directorio

```
MINGW64:/c/Users/David/Desktop  
  
David@DESKTOP-D2JLVJS MINGW64 ~  
$ cd Desktop/  
  
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop  
$ |
```

Cuando cambiamos la ruta el comando CD podemos ver que a la ruta raíz se nos añade el directorio en el que nos hemos situado

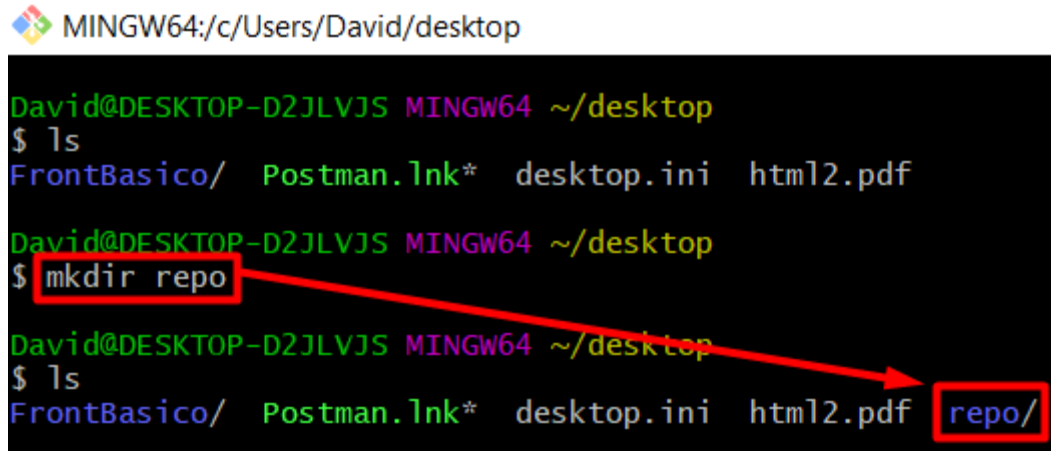
■ **CD ..:** retroceder

```
MINGW64:/c/Users/David  
  
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop  
$ pwd  
/c/Users/David/Desktop  
  
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop  
$ cd ..  
  
David@DESKTOP-D2JLVJS MINGW64 ~  
$ pwd  
/c/Users/David
```


Desplazándonos con GitBash

- **mkdir:** crea un directorio.

```
MINGW64:/c/Users/David/desktop  
  
David@DESKTOP-D2JLVJS MINGW64 ~/desktop  
$ ls  
FrontBasico/ Postman.lnk* desktop.ini html2.pdf  
  
David@DESKTOP-D2JLVJS MINGW64 ~/desktop  
$ mkdir repo  
  
David@DESKTOP-D2JLVJS MINGW64 ~/desktop  
$ ls  
FrontBasico/ Postman.lnk* desktop.ini html2.pdf repo/
```



Los directorios aparecen con una especie de color azulado

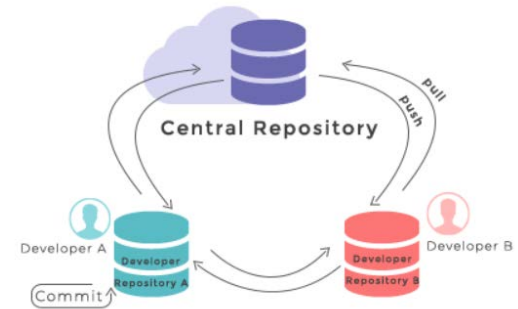
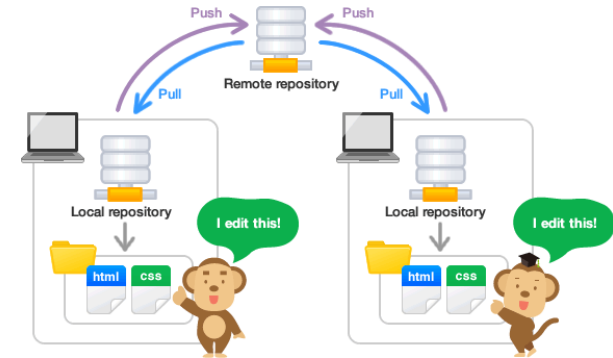
¿Qué es un repositorio?

- Un repositorio es la manera que tenemos de indicarle al sistema operativo y a GIT que un directorio trabajará con el sistema de control de versiones GIT.



Tipos de repositorio

- **Local repository:** es el repositorio que tenemos descargado físicamente en nuestro equipo (en local). Y sobre trabajamos solamente nosotros. Una vez realizados los cambios finalmente podemos (o no) subirlos al repositorio remoto.
- **Remote repository:** es el repositorio que tenemos en la nube, es decir, alojado sobre una de las plataformas de código como, por ejemplo: GitHub, BitBucket o GitLab. Y en el que se trabaja de forma colaborativa con el resto del equipo



¿Porqué tener dos tipos de repositorios?

- Ante el caso de perdida del repositorio local tenemos un Backup externo fuera de nuestro dispositivo (repository remote).
- Nos permite colaborar con otros usuarios de forma sencilla (repository remote).
- Nos ofrece la posibilidad de realizar desarrollos en local y decidir cuando estos serán compartidos con el resto del equipo (repository local).

Repositorios remotos

- Existen distintos repositorios remotos. Los principales son:

- ☐ [GitHub](#)
- ☐ [Bitbucket](#)
- ☐ [Gitlab](#)




- Nosotros trabajaremos sobre GitHub. Por lo que debemos crear una cuenta:



Configurando Git

- Para poder trabajar con los repositorios remotos, es necesario configurar el user.name y el user.email de la siguiente manera:

 MINGW64:/c/Users/David

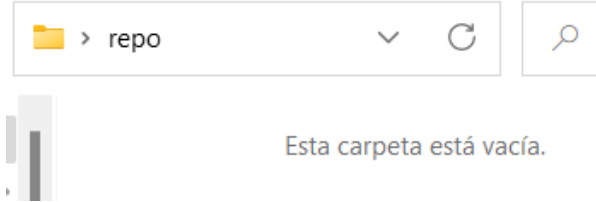
```
David@DESKTOP-D2JLVJS MINGW64 ~  
$ git config user.name  
  
David@DESKTOP-D2JLVJS MINGW64 ~  
$ git config --global user.name "David Bernal"  
  
David@DESKTOP-D2JLVJS MINGW64 ~  
$ git config user.name  
David Bernal
```

```
David@DESKTOP-D2JLVJS MINGW64 ~  
$ git config user.email  
  
David@DESKTOP-D2JLVJS MINGW64 ~  
$ git config --global user.email "dabernalgo@gmail.com"  
  
David@DESKTOP-D2JLVJS MINGW64 ~  
$ git config user.email  
dabernalgo@gmail.com
```

Creando un repositorio local

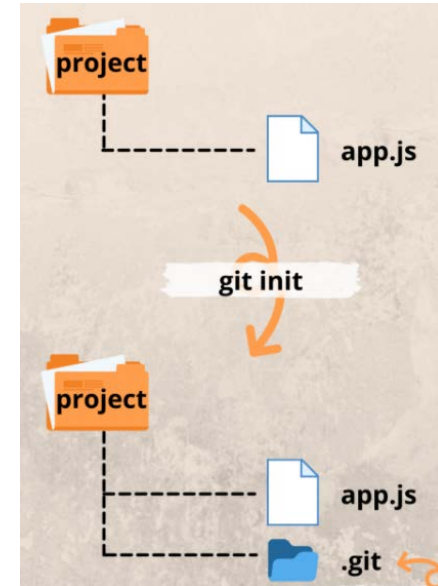
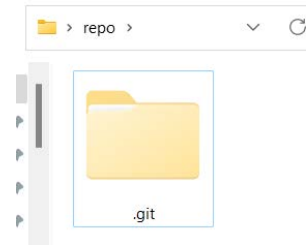
- Un directorio sin más no es considerado un repositorio de GIT

Directorio normal:



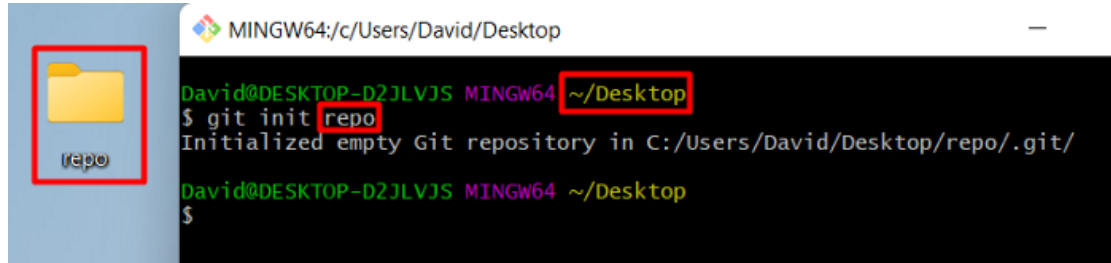
- Para que un directorio sea considerado un repositorio hay que indicárselo al sistema operativo y a Git mediante al comando Git init

Directorio considerado como un repositorio:



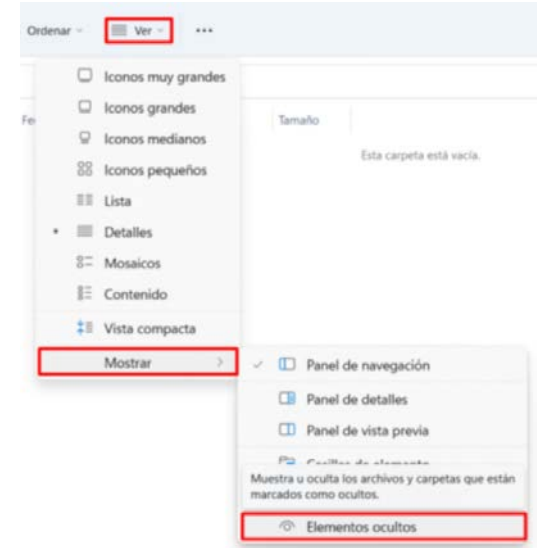
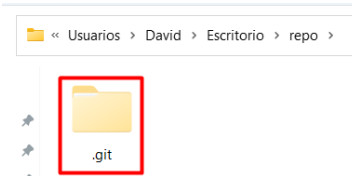
Creando un repositorio local

Si no vemos el directorio .git tendremos que habilitar la visualización de los elementos ocultos.



```
MINGW64/c/Users/David/Desktop  
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop  
$ git init repo  
Initialized empty Git repository in C:/Users/David/Desktop/repo/.git/  
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop  
$
```

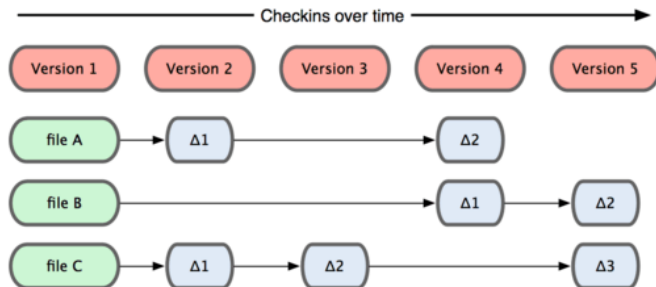
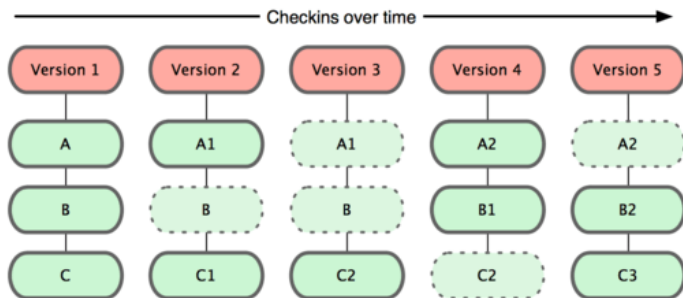
Si no utilizamos git init, la carpeta un directorio sin más. Pero cuando utilizamos git init, le estamos indicando al sistema operativo y al software de Git que vamos a crear un repositorio lo que nos creará un directorio en el interior del directorio .git (oculto) en el interior del directorio.



Dentro de este directorio no tendremos que tocar nada ya que contiene la BBDD mediante la cuál Git gestiona los “snapshots”

¿Qué es un snapshot?

- Un “snapshot” nos permite capturar del estado actual de los ficheros que se han añadido o modificado del proyecto.
- En Git somos nosotros los encargados de decidir en que momento queremos realizar un “snapshot” de los cambios.
- Git se basa en un sistema incremental en el que se realizan distintas “snapshots”. Vamos a ver un ejemplo:



Estados en Git

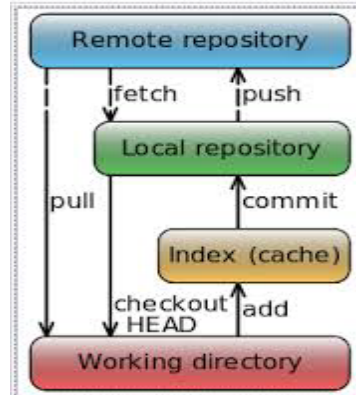
- **Working directory:** donde trabajamos con nuestros ficheros.
- **Staging area:** donde vas preparando los archivos que vas a subir al repositorio local
- **Local repository:** contiene los archivos que han sido guardados en nuestra máquina
- **Remote repository:** contiene los ficheros que son accesibles por el equipo al completo.



Moviendo los ficheros

Imaginaos que nuestro repositorio local es un avión.

- ☐ Inicialmente los pasajeros (ficheros) están en el working directory sin billetes.
- ☐ Git add permite decidir los pasajeros (ficheros) que pueden volar y por tanto hacer el embarque, para ello, son pasados al staging área
- ☐ Git commit introduce los ficheros “en el avión”, es decir, los pasa al repositorio local
- ☐ Git push envía “el avión”, es decir, permite que los cambios de los ficheros vayan hacia el repositorio remoto.



Git status

- Nos permite ver en que situación área de trabajo se encuentran nuestros ficheros:

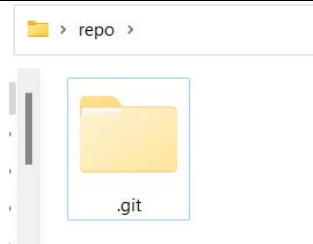
Ejemplo de fichero repositorio sin ficheros:

```
MINGW64:/c:/Users/David/Desktop/repo

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```



Ejemplo de fichero repositorio con ficheros:

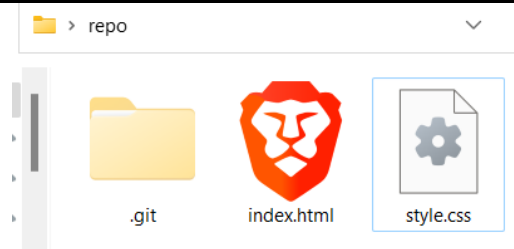
```
MINGW64:/c:/Users/David/Desktop/repo

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html
        style.css

nothing added to commit but untracked files present (use "git add" to track)
```



Añadiendo los ficheros al staging area

- Ahora que ya tenemos cambios en el working directory, vamos a añadirlos en nuestro staging área mediante al comando git add

Añadiendo todos los ficheros:

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html
        style.css

nothing added to commit but untracked files present (use "git add" to track)

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (master)
$ git add .

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html
        new file:   style.css
```

Añadiendo un solo fichero:

```
MINGW64:/c/Users/David/Desktop/repo

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (master)
$ git add index.html

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html
```



Retornando un ficheros al working directory

- Si finalmente no queremos subir un fichero y ya lo tenemos en el staging area, podemos echarlo hacía atrás de la siguiente manera:

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   index.html
    new file:   style.css

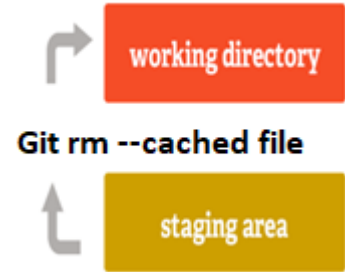
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (master)
$ git rm --cached style.css
rm 'style.css'

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (master)
$ git status
On branch master

No commits yet

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

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



Git commit

- Una vez tenemos los ficheros preparados en el staging area. Ya podemos realizar el envío al repositorio local:

```
MINGW64/c/Users/David/Desktop/repo

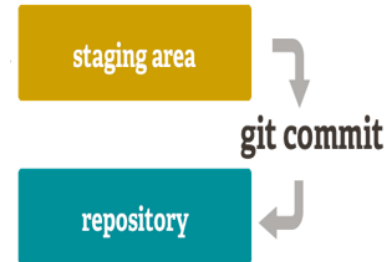
David@DESKTOP-D2JLV3S MINGW64 ~/Desktop/repo (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html
        new file:   style.css

David@DESKTOP-D2JLV3S MINGW64 ~/Desktop/repo (master)
$ git commit -m "Creo los ficheros index.html y style.css"
[master (root-commit) 4a88fdd] Creo los ficheros index.html y style.css
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 index.html
create mode 100644 style.css

David@DESKTOP-D2JLV3S MINGW64 ~/Desktop/repo (master)
$ git status
On branch master
nothing to commit, working tree clean
```



- Se utiliza para pasar los ficheros del staging área al repositorio local

Git log

- Nos permite visualizar un historial con los distintos commits que hemos realizado.
- En cada commit, podemos ver:
 1. El identificador del commit
 2. El autor del commit
 3. La fecha del commit
 4. El mensaje del commit

MINGW64:/c/Users/David/Desktop/repo

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (master)
$ git log
commit 4a88fdd26ac09e02f0cc1820435cf26036d8e62d (HEAD -> master)
Author: David Bernal <dabernalgo@gmail.com>
Date: Tue Mar 1 15:43:00 2022 +0100


    Creo los ficheros index.html y style.css
```

1
3

2
4

Git push

- Nos permite subir los cambios a un repositorio remoto.
- Antes de intentar realizar el push, debemos asociar un repositorio con el proyecto.
- Ya que sino... Vemos que nos dice que no está configurado a donde debe subirlos:

 MINGW64:/c/Users/David/Desktop/repo

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (master)
$ git push
fatal: No configured push destination.
Either specify the URL from the command-line or configure a remote repository using

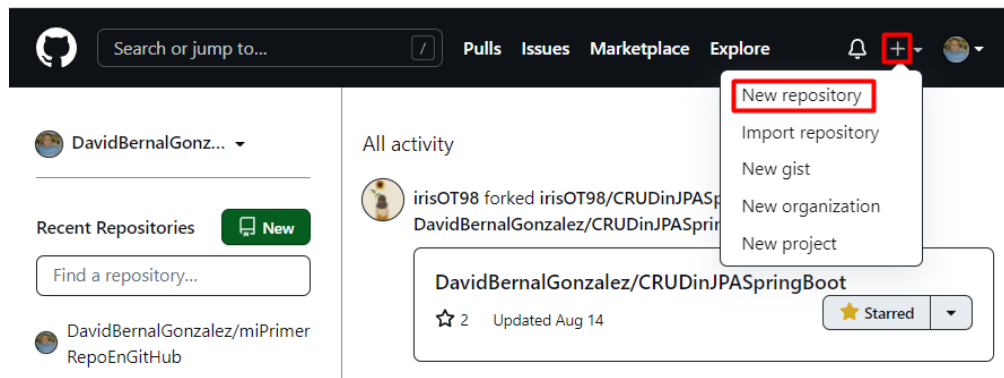
    git remote add <name> <url>

and then push using the remote name

    git push <name>
```

Creando el repositorio remoto desde Github

- Para crear un repositorio remoto vamos a GitHub:



Owner * / Repository name *

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

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:
Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

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

Create repository

Asociando el repositorio remoto con el repositorio local

- Si nos fijamos en el repositorio que se acaba de crear tenemos los pasos para asociar un repositorio ya existente:
- Si hacemos un remote para ver los repositorios remotos asociados, podemos ver que no tenemos nada asociado. Por ello, no podemos realizar el envío de información a nuestro repositorio remoto

...or push an existing repository from the command line

```
git remote add origin https://github.com/DavidBernalGonzalez/repo.git  
git branch -M main  
git push -u origin main
```

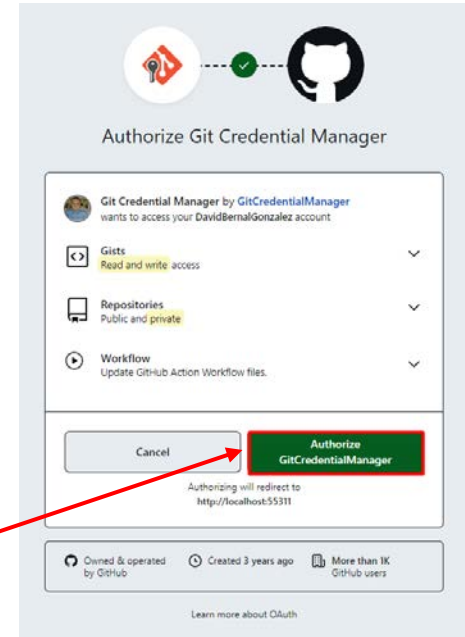
MINGW64:/c/Users/David/Desktop/repo

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)  
$ git remote -v
```

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)  
$
```

Asociando el repositorio remoto con el repositorio local

- Si pegamos el código que nos proporciona GitHub, vamos a asociar el repositorio remoto con nuestro repositorio local, crear una rama main y finalmente subir los cambios a dicha rama. Vamos a verlo:



Asociando el repositorio remoto con el repositorio local

- Una vez hagamos nos autenticamos en GitHub, podremos ver que:
- Tenemos el push realizado, por lo que los cambios ya deben aparecer en el repositorio remoto.
- Y además tenemos el repositorio remoto configurado.

```
MINGW64:/c:/Users/David/Desktop/repo

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ git remote -v

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ git remote add origin https://github.com/DavidBernalGonzalez/repo.git
git branch -M main
git push -u origin main

Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 248 bytes | 248.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/DavidBernalGonzalez/repo.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ git remote -v
origin https://github.com/DavidBernalGonzalez/repo.git (fetch)
origin https://github.com/DavidBernalGonzalez/repo.git (push)

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ |
```

Asociando el repositorio remoto con el repositorio local

- Si vamos a Github podemos ver que tenemos un snapshot (commit) realizado en el repositorio remoto. Concretamente en la rama Main:

The screenshot displays the GitHub interface for a public repository named 'repo' by user 'DavidBernalGonzalez'. The repository has 1 branch (main) and 0 tags. A red box highlights the '1 commit' indicator in the top right of the commit list. Below this, a table lists the files in the commit: 'index.html' and 'style.css', both committed 43 minutes ago. A red arrow points from the '1 commit' box to a detailed view of the commit at the bottom of the page. This view shows the commit message 'Creo los ficheros index.html y style.css' by 'DavidBernalGonzalez' committed 1 hour ago, with the commit hash '4a88fdd' and a code icon.

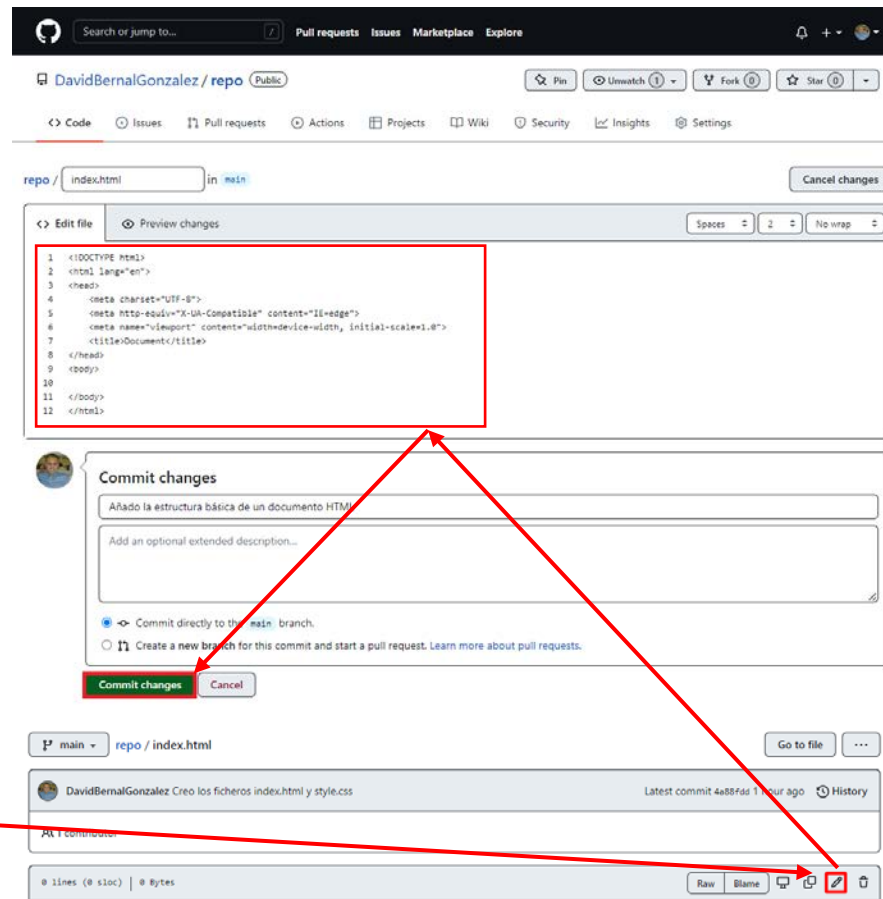
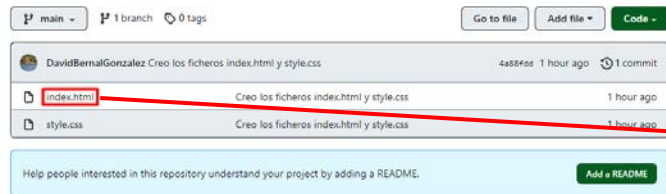
File	Commit Message	Time
index.html	Creo los ficheros index.html y style.css	43 minutes ago
style.css	Creo los ficheros index.html y style.css	43 minutes ago

Commits on Mar 1, 2022

Commit Message	Commit Hash	Time
Creo los ficheros index.html y style.css	4a88fdd	1 hour ago

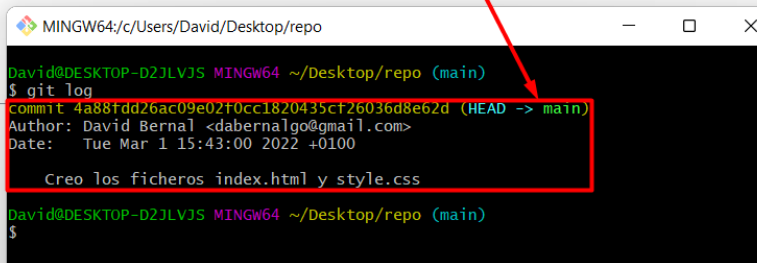
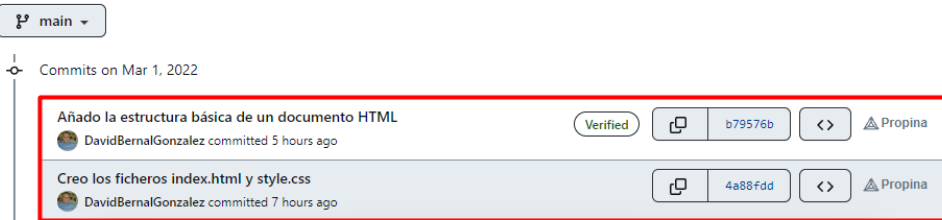
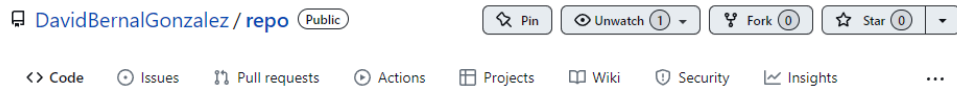
Git pull

- Ahora que el repositorio está en la nube, (en el caso de trabajar con varias personas) es posible que algún compañero que tenga permisos realice una modificación en el código del proyecto y realice una subida. Lo que puede suponer que nuestro repositorio no este actualizado. Nosotros lo vamos a hacer de una forma poco elegante directamente desde Git para ver lo que hace git pull:



Git pull

- Si nos fijamos, actualmente tenemos dos commits en el repositorio remoto y nosotros solamente 1 en nuestro repositorio. Por lo que tenemos que actualizar el repositorio utilizando git pull:



MINGW64:/c/Users/David/Desktop/repo

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ git pull origin main
From https://github.com/DavidBernalGonzalez/repo
 * branch      main       -> FETCH_HEAD
Updating 4a88fdd..b79576b
Fast-forward
 index.html | 12 ++++++++
 1 file changed, 12 insertions(+)

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ git log
commit b79576bea6c99e034b4c837f0bb05b7698386eb6 (HEAD -> main, origin/main)
Author: DavidBernalGonzalez <32896437+DavidBernalGonzalez@users.noreply.github.com>
Date:   Tue Mar 1 17:46:33 2022 +0100
    Añado la estructura básica de un documento HTML

commit 4a88fdd26ac09e02f0cc1820435cf26036d8e62d
Author: David Bernal <dabernalgo@gmail.com>
Date:   Tue Mar 1 15:43:00 2022 +0100
    Creo los ficheros index.html y style.css
```


Git pull

- Si abrimos el fichero ahora, podemos ver que tiene la estructura básica de un HTML:

```
index.html X
index.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Document</title>
8  </head>
9  <body>
10
11 </body>
12 </html>
```

Git clone

- Otra manera de trabajar con un proyecto, es directamente clonando el repositorio. Inclusive muchos desarrolladores cuando empiezan un proyecto desde 0 suelen crear el repositorio en GitHub y posteriormente clonarlo ya que estará listo para trabajar con él.

The image shows a GitHub repository page for 'DavidBernalGonzalez/repo' and a terminal window. Red arrows indicate the workflow: from the 'Code' button on GitHub to the terminal command, and from the repository URL in the terminal to the 'Code' button's dropdown menu.

GitHub Repository Page:

- Repository: DavidBernalGonzalez/repo (Public)
- Buttons: Code, Issues, Pull requests, Actions, Projects, Wiki
- Clone dropdown menu shows the URL: `https://github.com/DavidBernalGonzalez/repo.git`

Terminal Window (MINGW64):

```
David@DESKTOP-D2JLVJS MINGW64 ~/desktop
$ git clone https://github.com/DavidBernalGonzalez/repo.git
Cloning into 'repo'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 6 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.

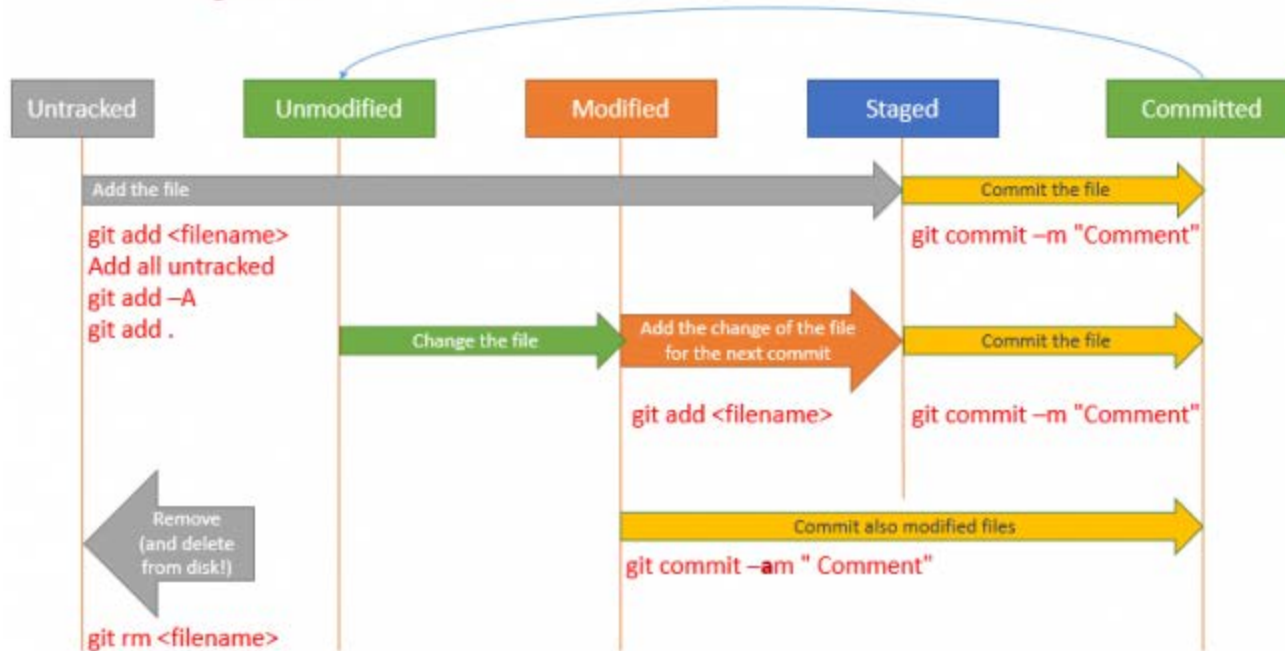
David@DESKTOP-D2JLVJS MINGW64 ~/desktop
$
```

Terminal Window (MINGW64):

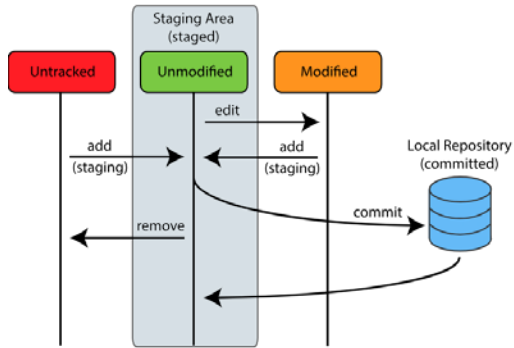
```
David@DESKTOP-D2JLVJS MINGW64 ~/desktop/repo (main)
$ git remote -v
origin https://github.com/DavidBernalGonzalez/repo.git (fetch)
origin https://github.com/DavidBernalGonzalez/repo.git (push)
```

GIT "File status Lifecycle"

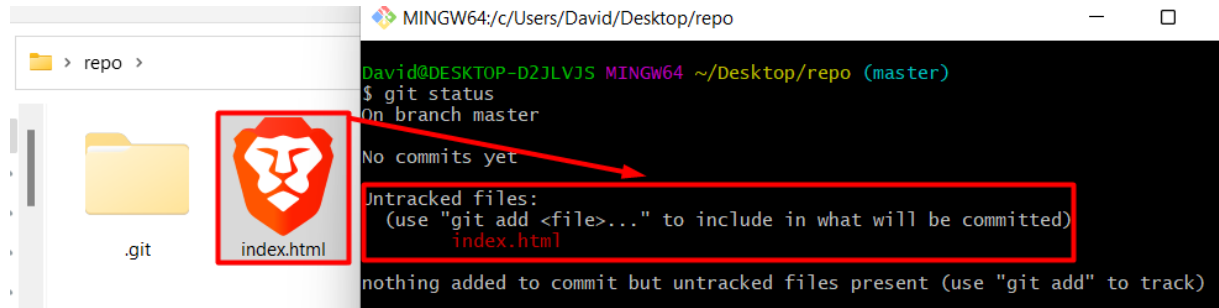
Check status with „git status“



GIT "File status Lifecycle"



Untracked (no registrado):



Unmodified (no modificado):

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (master)
$ git add .

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   index.html
```

Modificado (modificado):

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (master)
$ git status
On branch master

No commits yet

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

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   index.html
```

Ejercicios

- Ejercicios de repaso de GIT: <https://progate.com/git/>

The screenshot shows the Progate web application interface for a Git lesson. At the top, the breadcrumb navigation reads "Git Study I > The Git Flow > 1. Initializing Git". The user's profile "David BG(Lv.12)" is in the top right. On the left, the "Instructions" panel contains the text: "Welcome to the Git lesson! Ken the Ninja and Master Wooly want to use Git to develop a website called mysite together. Let's start preparing!". Below this, a "Terminal" tab is active, showing a terminal window with the prompt "mysite \$". The bottom of the interface features a "Reset" button, a "See Answer" button, and a green "Submit" button.

Ejercicios

- **EJERCICIO 1:**

- ☐ 1.1 - Crea un directorio llamado repo01 (desde tu máquina) e inicia el repositorio en local
- ☐ 1.2 - Añade un documento llamado **readme.md** (md de Markdown) y documenta en su interior todos los pasos que vas realizando para crear un repositorio
- ☐ 1.3 - Añade el fichero al staging area y haz un snapshot hacía el repositorio en local
- ☐ 1.4 - Crea un repositorio remoto llamado repo01, asócialo a tu repositorio local y sube los cambios al repositorio remoto

Ejercicios

■ EJERCICIO 2:

- 2.1 - Crea un repositorio llamado repo02 desde GitHub.
- 2.2 – Posteriormente, clónalo (con git clone), añade un fichero readme.md y haz un commit.
- 2.3 – Entra en este [manual de Markdown](#) y haz un resumen de los principales comandos de Git con los que hemos trabajando. Puedes utilizar tablas, imágenes, títulos, enlaces, etc. IMPORTANTE no subas todo el código de golpe, ya que es mejor que practiques los conceptos y por tanto, ves subiendo los cambios en distintas subidas a tu repositorio de GitHub y comprobando que los cambios se visualicen correctamente.

In case of fire



1. git commit

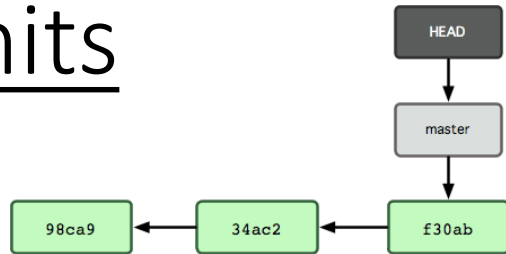


2. git push



3. leave building

Desplazadonos entre commits temporalmente



- Si hacemos un git log vemos que dos logs y mediante a HEAD podemos saber sobre que commit estamos trabajando:

MINGW64:/c/Users/David/desktop/repo

```
David@DESKTOP-D2JLVJS MINGW64 ~/desktop/repo ((b79576b...))
$ git log --graph --oneline --decorate
* b79576b (HEAD, origin/main, origin/HEAD, main) Añado la estructura básica de un documento HTML
* 4a88fdd Creo los ficheros index.html y style.css
```

- El Head, por tanto, es el puntero mediante al cual le indicamos a GIT sobre que commit estamos situados actualmente

MINGW64:/c/Users/David/desktop/repo

```
David@DESKTOP-D2JLVJS MINGW64 ~/desktop/repo ((b79576b...))
$ git log --graph --oneline --decorate
* b79576b (HEAD, origin/main, origin/HEAD, main) Añado la estructura básica de un documento HTML
* 4a88fdd Creo los ficheros index.html y style.css
```

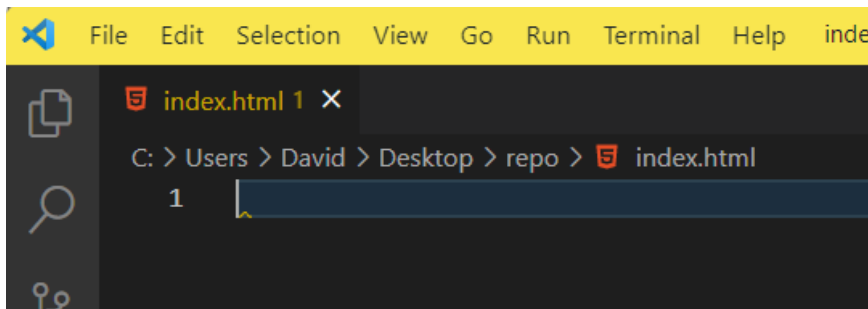
- ¿Como nos situamos en el commit 1? Con git checkout commit hash

```
David@DESKTOP-D2JLVJS MINGW64 ~/desktop/repo ((b79576b...))
$ git checkout 4a88fdd
Previous HEAD position was b79576b Añado la estructura básica de un documento HTML
HEAD is now at 4a88fdd Creo los ficheros index.html y style.css
```

```
David@DESKTOP-D2JLVJS MINGW64 ~/desktop/repo ((4a88fdd...))
$ git log --graph --oneline --decorate
* 4a88fdd (HEAD) Creo los ficheros index.html y style.css
```

Desplanzadonos entre commits temporalmente

- Si ahora miramos el fichero HTML, podemos ver que no tenemos el index.html relleno:



- Ya que en ese commit solamente hemos creado el fichero pero no hemos escrito nada en él

Desplanzadonos entre commits temp.

- Si hacemos un git log podemos ver que solamente nos muestra el commit actual:
- Entonces ¿Cómo conseguimos ver el hash del segundo commit? Utilizando el flag `--reflog`
- Finalmente, solamente tenemos que hacer un git checkout al hash del commit sobre el que nos queremos situar.
- Si nos fijamos siempre tenemos el main situado sobre el último commit por ello decimos que el situarnos en el commit es algo temporal solo movemos el HEAD
- También podemos ver los commits desde GitHub en el caso de tenerlos subidos al repositorio remoto:

MINGW64:/c/Users/David/desktop/repo

```
David@DESKTOP-D2JLVJS MINGW64 ~/desktop/repo ((4a88fdd...))
$ git log --graph --oneline --decorate
* 4a88fdd (HEAD) Creo los ficheros index.html y style.css
```

MINGW64:/c/Users/David/desktop/repo

```
David@DESKTOP-D2JLVJS MINGW64 ~/desktop/repo ((4a88fdd...))
$ git log --graph --oneline --decorate --reflog
* b79576b (origin/main, origin/HEAD, main) Añado la estructura básica de un documento HTML
* 4a88fdd (HEAD) Creo los ficheros index.html y style.css
```

```
David@DESKTOP-D2JLVJS MINGW64 ~/desktop/repo ((4a88fdd...))
$ git checkout b79576b
Previous HEAD position was 4a88fdd Creo los ficheros index.html y style.css
HEAD is now at b79576b Añado la estructura básica de un documento HTML
```

```
David@DESKTOP-D2JLVJS MINGW64 ~/desktop/repo ((b79576b...))
$ git log --graph --oneline --decorate
* b79576b (HEAD, origin/main, origin/HEAD, main) Añado la estructura básica de un documento HTML
* 4a88fdd Creo los ficheros index.html y style.css
```

main

Commits on Mar 1, 2022

Añado la estructura básica de un documento HTML

DavidBernalGonzalez committed 7 hours ago

Verified

b79576b

<>

Propina

Creo los ficheros index.html y style.css

DavidBernalGonzalez committed 9 hours ago

4a88fdd

<>

Propina

Desplanzadonos entre commits temporalmente

- Finalmente, para volver al presente, al momento en el que nos encontrábamos en la rama main debemos de realizarlo lo siguiente:

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo ((b79576b...))
$ git log --graph --oneline --decorate
* b79576b (HEAD, origin/main, origin/HEAD, main) Añado la estructura básica de un documento HTML
* 4a88fdd Creo los ficheros index.html y style.css

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo ((b79576b...))
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$
```

Visualizando los cambios realizados desde Git

- Para ver los cambios realizados en un commit podemos utilizar git show

```
MINGW64:/c/Users/David/Desktop/repo

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ git log --pretty=format:"%h - %an : %s"
b79576b - DavidBernalGonzalez : Añado la estructura básica de un documento HTML
4a88fdd - David Bernal : Creo los ficheros index.html y style.css

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ git show b79576b
commit b79576bea6c99e034b4c837f0bb05b7698386eb6 (HEAD -> main, origin/main, origin/HEAD)
Author: DavidBernalGonzalez <32896437+DavidBernalGonzalez@users.noreply.github.com>
Date: Tue Mar 1 17:46:33 2022 +0100

    Añado la estructura básica de un documento HTML

diff --git a/index.html b/index.html
index e69de29..100b87b 100644
--- a/index.html
+++ b/index.html
@@ -0,0 +1,12 @@
+<!DOCTYPE html>AM
+<html lang="en">AM
+<head>AM
+  <meta charset="UTF-8">AM
+  <meta http-equiv="X-UA-Compatible" content="IE=edge">AM
+  <meta name="viewport" content="width=device-width, initial-scale=1.0">AM
+  <title>Document</title>AM
+</head>AM
+<body>AM
+  AM
+</body>AM
+</html>AM

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$
```

Git branch – Creando una rama

- Hasta ahora todo nuestro trabajo se ha realizado en la rama master pero esto nos puede suponer que si no tenemos la rama actualizada y otro compañero sube los cambios podamos tener conflictos que son difíciles de resolver. Por ello, se aconseja trabajar en nuestro código desde una rama independiente. Vamos a ver como trabajar con ramas:
- Si visualizamos las ramas actuales vemos que solamente tenemos main:

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ git branch
* main
```
- Para trabajar con ramas lo primero que tenemos que hacer es crear una rama utilizamos git Branch nombreRama

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ git branch testBranch

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ |
```
- Con esto creamos la rama pero no nos situamos en ella. Si visualizamos las ramas podemos ver que

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ git branch
* main
  testBranch
```

Git branch – Situandonos en la rama

- El * nos indica sobre que rama estamos situados.

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ git branch
* main
  testBranch
```

- Para modificar la rama hacemos git branch y seleccionamos la rama a la que queremos ir:

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ git checkout testBranch
Switched to branch 'testBranch'

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (testBranch)
$ git branch
  main
* testBranch
```

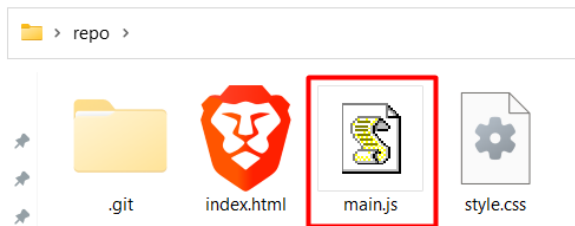
Git branch – Preparando los archivos

- Una vez situados sobre la rama, podemos ver que HEAD apunta hacia testBranch.

```
MINGW64/c/Users/David/Desktop/repo

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (testBranch)
$ git log --graph --all --decorate
* commit b79576bea6c99e034b4c837f0bb05b7698386eb6 (HEAD -> testBranch, origin/main, origin/HEAD, main)
| Author: DavidBernalGonzalez <32896437+DavidBernalGonzalez@users.noreply.github.com>
| Date: Tue Mar 1 17:46:33 2022 +0100
|
| Añado la estructura básica de un documento HTML
|
* commit 4a88fdd26ac09e02f0cc1820435cf26036d8e62d
| Author: David Bernal <dabernalgo@gmail.com>
| Date: Tue Mar 1 15:43:00 2022 +0100
|
| Creo los ficheros index.html y style.css
```

- Si añadimos un archivo nuevo al proyecto y lo pasamos al staging area:



```
MINGW64/c/Users/David/Desktop/repo

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (testBranch)
$ git status
On branch testBranch
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    main.js


nothing added to commit but untracked files present (use "git add" to track)

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (testBranch)
$ git add .

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (testBranch)
$ git status
On branch testBranch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   main.js
```


Git branch – Commiteando los archivos

- Una vez situados sobre la rama, podemos ver que HEAD apunta hacía testBranch.

 MINGW64:/c/Users/David/Desktop/repo

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (testBranch)
$ git commit -m "Añado el fichero main.js"
[testBranch b7f704a] Añado el fichero main.js
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 main.js
```

- Y finalmente subimos los ficheros:

 MINGW64:/c/Users/David/Desktop/repo

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (testBranch)
$ git log
commit b7f704a9548985ba01b8257c550e7ce66510ab8b (HEAD -> testBranch)
Author: David Bernal <dabernalgo@gmail.com>
Date: Wed Mar 2 02:08:48 2022 +0100

    Añado el fichero main.js

commit b79576bea6c99e034b4c837f0bb05b7698386eb6 (origin/main, origin/HEAD, main)
Author: David BernalGonzalez <32896437+DavidBernalGonzalez@users.noreply.github.com>
Date: Tue Mar 1 17:46:33 2022 +0100

    Añado la estructura básica de un documento HTML

commit 4a88fdd26ac09e02f0cc1820435cf26036d8e62d
Author: David Bernal <dabernalgo@gmail.com>
Date: Tue Mar 1 15:43:00 2022 +0100

    Creo los ficheros index.html y style.css

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (testBranch)
$ git push
fatal: The current branch testBranch has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin testBranch

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (testBranch)
$ git push --set-upstream origin testBranch
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 295 bytes | 295.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'testBranch' on GitHub by visiting:
remote:   https://github.com/DavidBernalGonzalez/repo/pull/new/testBranch
remote:
To https://github.com/DavidBernalGonzalez/repo.git
 * [new branch]      testBranch -> testBranch
branch 'testBranch' set up to track 'origin/testBranch'.
```

Git branch – Observando los cambios en GitHub

- Tras subir los cambios a GitHub vemos que tenemos dos ramas:

The screenshot shows the GitHub web interface for a repository. At the top, there's a navigation bar with 'main' selected, indicating 2 branches and 0 tags. Below this, a 'Switch branches/tags' dropdown is open, showing a search bar and a list of branches: 'main' (checked) and 'testBranch'. A 'View all branches' link is at the bottom of the dropdown. The main content area has tabs for 'Overview', 'Yours', 'Active', 'Stale', and 'All branches'. The 'Overview' tab is active, showing the 'Default branch' as 'main' (updated 8 hours ago by DavidBernalGonzalez) and 'Your branches' as 'testBranch' (updated 2 minutes ago by DavidBernalGonzalez). A 'New pull request' button is visible next to 'testBranch'.

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (testBranch)
$ git log --all --oneline --decorate --graph
* b7f704a (HEAD -> testBranch, origin/testBranch) Añado el fichero main.js
* b79576b (origin/main, origin/HEAD, main) Añado la estructura básica de un documento HTML
* 4a88fdd Creo los ficheros index.html y style.css
```

Git branch – Juntando la rama con master

- Para juntar una rama con master nos situamos en la rama main y hacemos un git merge de la rama que queremos mergear (fusionar)
- En este caso, vemos que el tipo de merge es fast-forward. En la siguiente diapositiva explicaremos la diferencia entre fast –forward y no fast-forward.

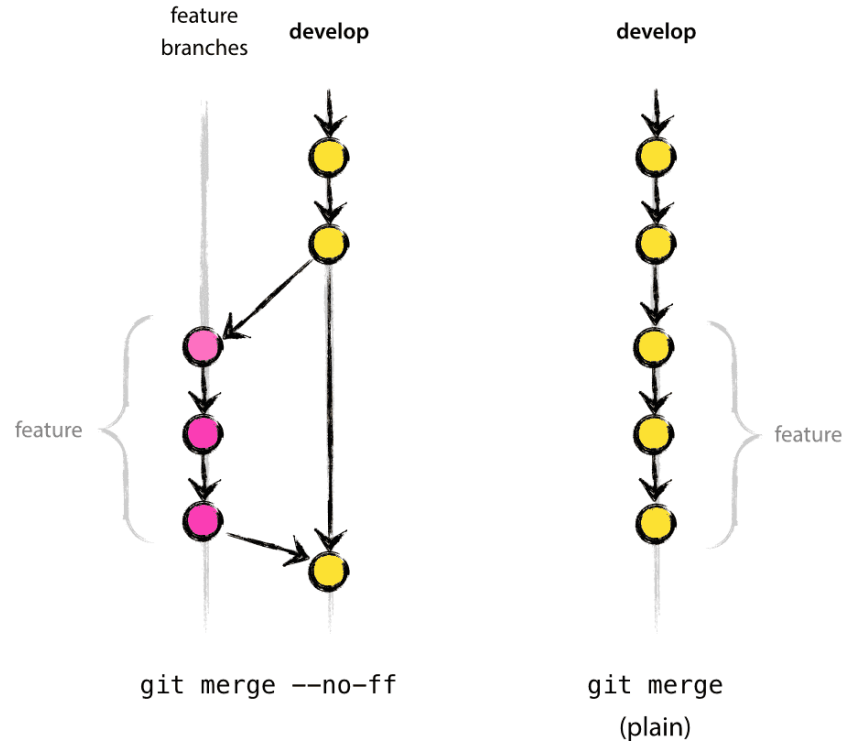
```
MINGW64:/c/Users/David/Desktop/repo

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (testBranch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

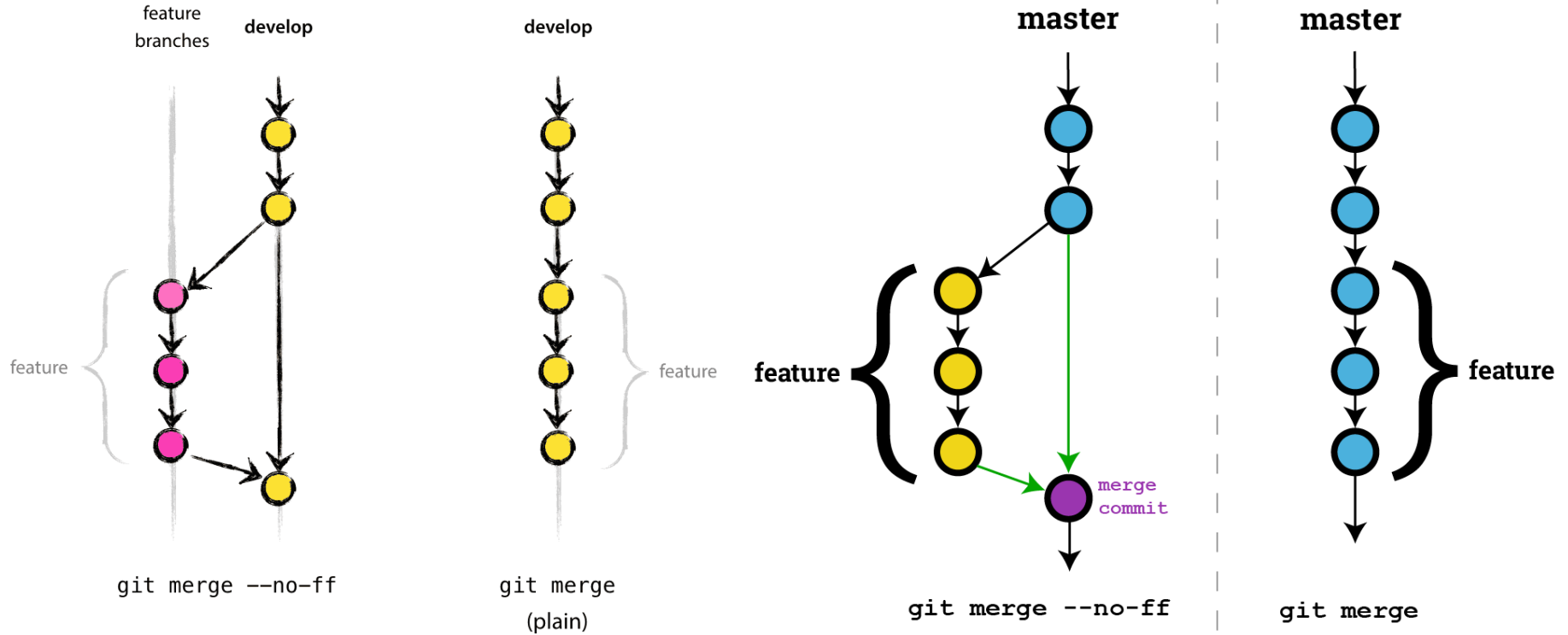
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ git merge testBranch
Updating b79576b..b7f704a
Fast-forward
 main.js | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 main.js
```

Tipos de merge

- “Fast forward” (FF) significa avance rápido y nos permite fusionar la rama que acabamos de crear directamente con la rama main sin realizar un commit de confirmación.
 - ☐ `git merge -ff`
 - ☐ `git merge --ff` = `git merge` Ya que por defecto se utiliza el `--ff` por defecto (default)
- En cambio, “No Fast Forward” (NFF), crea una nueva confirmación con varios padres. Lo que proporciona un mejor seguimiento del historial.
 - ☐ `git merge --no-ff newFeature -m 'add new feature'`



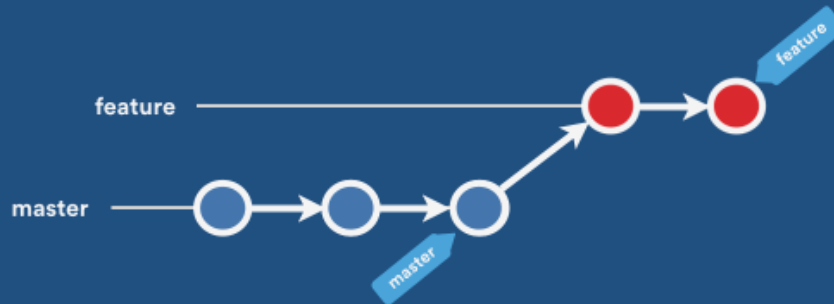
Tipos de merge



Tipos de merge

What is a fast-forward merge?

It will just shift the
master HEAD



Git branch – Ejemplo No-Fast-Forward

MINGW64:/c/Users/David/Desktop/repo

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ git branch
* main
  testBranch
```

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ git checkout -b noFastForwardExample
Switched to a new branch 'noFastForwardExample'
```

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (noFastForwardExample)
$ git branch
  main
* noFastForwardExample
  testBranch
```

MINGW64:/c/Users/David/Desktop/repo

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (noFastForwardExample)
$ git status
On branch noFastForwardExample
nothing to commit, working tree clean
```

MINGW64:/c/Users/David/Desktop/repo

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (noFastForwardExample)
$ git status
On branch noFastForwardExample
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  readme.md

nothing added to commit but untracked files present (use "git add" to track)

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (noFastForwardExample)
$ git add .

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (noFastForwardExample)
$ git status
On branch noFastForwardExample
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
  new file:   readme.md
```

Git branch – Ejemplo No-Fast-Forward

MINGW64:c:/Users/David/Desktop/repo

```
David@DESKTOP-D2JLV3S MINGW64 ~/Desktop/repo (noFastForwardExample)
$ git commit -m "Añado el fichero readme.md"
[noFastForwardExample b32b2b8] Añado el fichero readme.md
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 readme.md

David@DESKTOP-D2JLV3S MINGW64 ~/Desktop/repo (noFastForwardExample)
$ git log
commit b32b2b8718629436d12a22dece99090baa45bf41 (HEAD -> noFastForwardExample)
Author: David Bernal <dabernalgo@gmail.com>
Date:   Wed Mar 2 02:56:59 2022 +0100

    Añado el fichero readme.md

commit b7f704a9548985ba01b8257c550e7ce66510ab8b (origin/testBranch, testBranch, main)
Author: David Bernal <dabernalgo@gmail.com>
Date:   Wed Mar 2 02:08:48 2022 +0100

    Añado el fichero main.js

commit b79576bea6c99e034b4c837f0bb05b7698386eb6 (origin/main, origin/HEAD)
Author: DavidBernalGonzalez <32896437+DavidBernalGonzalez@users.noreply.github.com>
Date:   Tue Mar 1 17:46:33 2022 +0100

    Añado la estructura básica de un documento HTML

commit 4a88fdd26ac09e02f0cc1820435cf26036d8e62d
Author: David Bernal <dabernalgo@gmail.com>
Date:   Tue Mar 1 15:43:00 2022 +0100

    Creo los ficheros index.html y style.css
```


Git branch – Ejemplo No-Fast-Forward

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (noFastForwardExample)
```

```
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)
```

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
```

```
$ git branch
* main
  noFastForwardExample
  testBranch
```

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
```

```
$ git merge --no-ff noFastForwardExample -m "Ejemplo de noFastForward Merge"
Merge made by the 'ort' strategy.
 readme.md | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 readme.md
```

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
```

```
$
```

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
```

```
$ git log --all --oneline --decorate --graph
* fd49ba7 (HEAD -> main) Ejemplo de noFastForward Merge
|
| * b32b2b8 (noFastForwardExample) Añado el fichero readme.md
|/
* b7f704a (origin/testBranch, testBranch) Añado el fichero main.js
* b79576b (origin/main, origin/HEAD) Añado la estructura básica de un documento HTML
* 4a88fdd Creo los ficheros index.html y style.css
```

Eliminando ramas en local

- Para eliminar una rama en local realizamos lo siguiente:

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ git branch
* main
  noFastForwardExample
  testBranch

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ git branch -d testBranch
Deleted branch testBranch (was b7f704a).

David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ git branch
* main
  noFastForwardExample

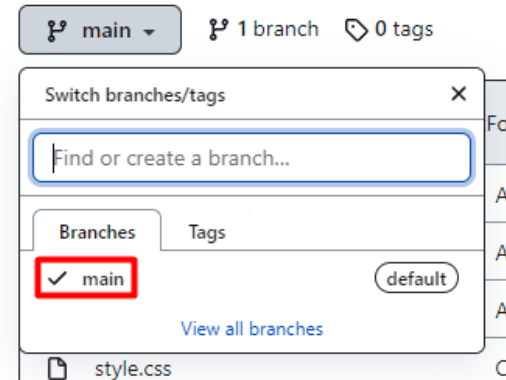
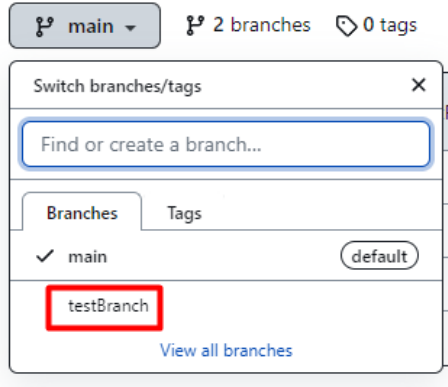
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$
```

Eliminando ramas en remoto

- Para eliminar una rama en remoto realizamos lo siguiente:

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ git push origin --delete testBranch
To https://github.com/DavidBernalGonzalez/repo.git
- [deleted]          testBranch
```

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repo (main)
$ git branch
* main
  noFastForwardExample
```



Ejercicio

■ EJERCICIO 3: FAST FORWARD

- 3.1 - Crea un directorio llamado repo03 (tu decides como lo haces) con un fichero readme.md vacío, haz un commit y súbelo a un repositorio remoto que tendrá el mismo nombre repo03.
- 3.2 - Crea una rama con tu nombre y la fecha (por ejemplo david02032022) **desde la que editaremos el fichero**
- 3.3 – Desde tu rama edita el fichero readme.md para que quede de la siguiente manera:

Repositorio 03

Mi primer ejercicio con ramas
- 3.4 – Haz un commit en tu rama
- 3-5 – Fusiona tu rama con master
- 3.6 – Haz un push hacía la nube
- 3-7 – Elimina solamente la rama en local david02032022 ya que si eliminásemos la remota no veríamos la rama en remoto

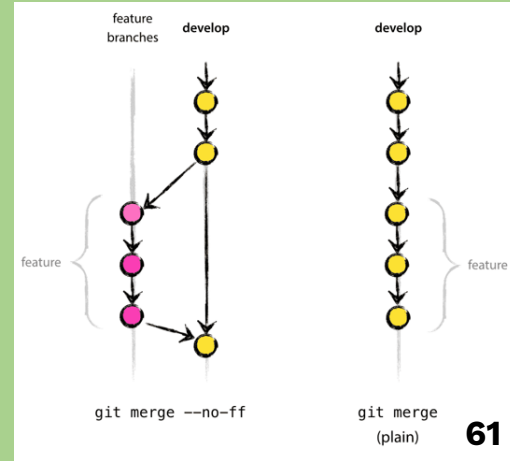
Ejercicio

■ EJERCICIO 3: FAST FORWARD

- 3.1 - Crea un directorio llamado repo03 (tu decides como lo haces) con un fichero readme.md vacío, haz un commit y súbelo a un repositorio remoto que tendrá el mismo nombre repo03.
- 3.2 - Crea una rama con tu nombre y la fecha (por ejemplo david02032022) **desde la que editaremos el fichero**
- 3.3 – Desde tu rama edita el fichero readme.md para que quede de la siguiente manera:
- 3.4 – Haz un commit en tu rama
- 3.5 – Haz un push hacía la nube
- 3-6 – Fusiona tu rama con master
- 3-7 – Elimina solamente la rama en local david02032022 ya que no queremos eliminar la rama main

Repositorio 03

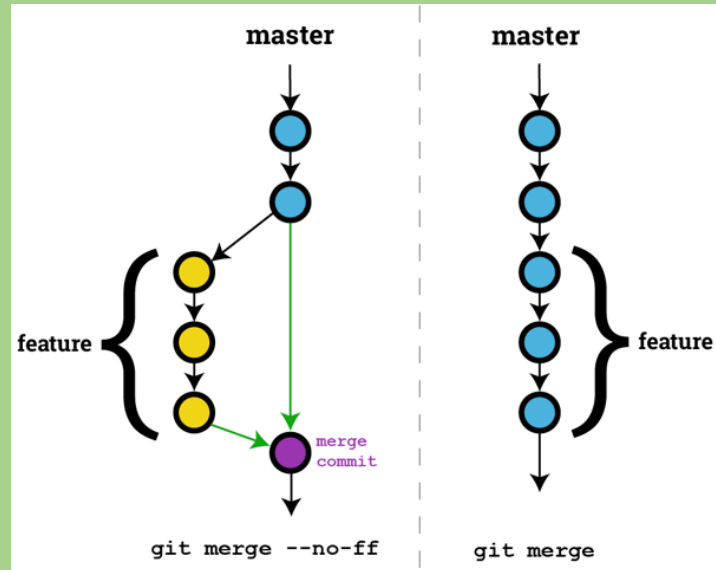
Mi primer ejercicio con ramas



Ejercicio

■ EJERCICIO 4: NO FAST FORWARD

- 4.1 - Crea un repositorio llamado repo04 y basando en el ejemplo que hemos visto realiza un commit no fast-forward. En este caso, antes de hacer el merge sube ambas ramas y luego haz el merge por si te equivocas poder volver a clonar el repositorio.



Ejercicio de alias

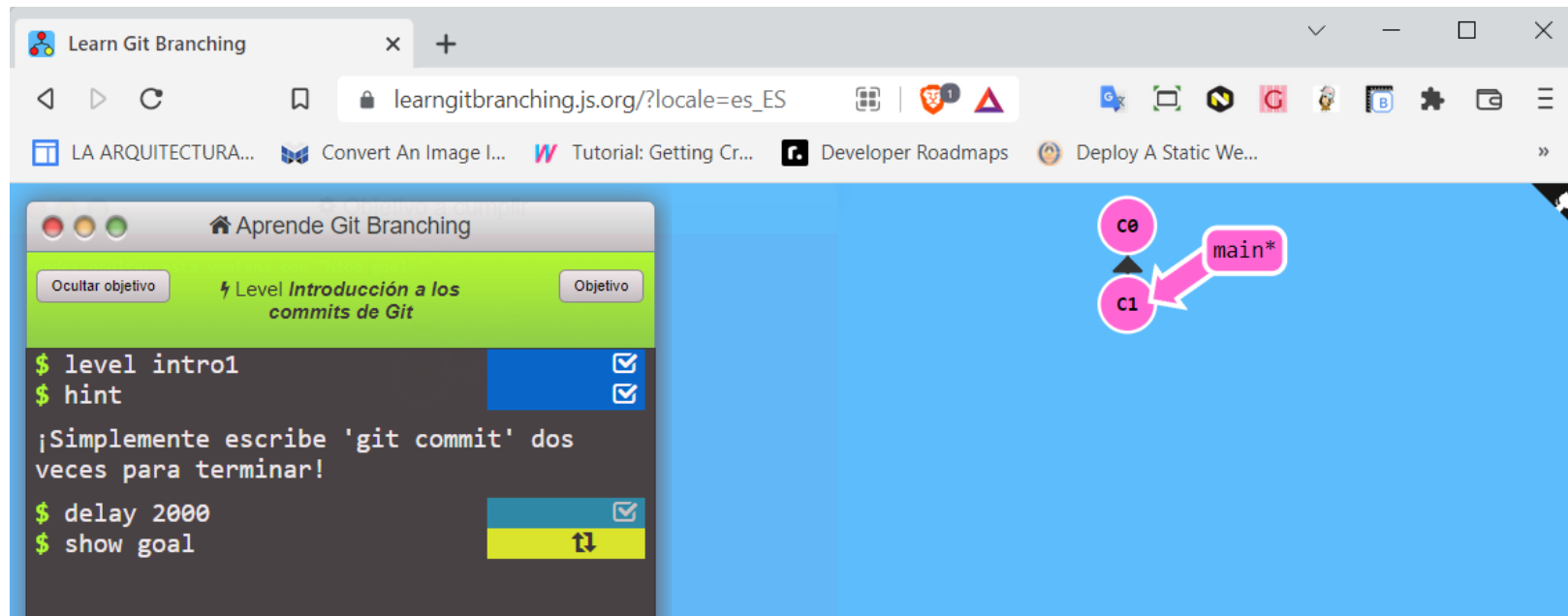
- Haz un alias con la finalidad de que cuando escribamos gitlog adog nos ejecute lo siguiente:



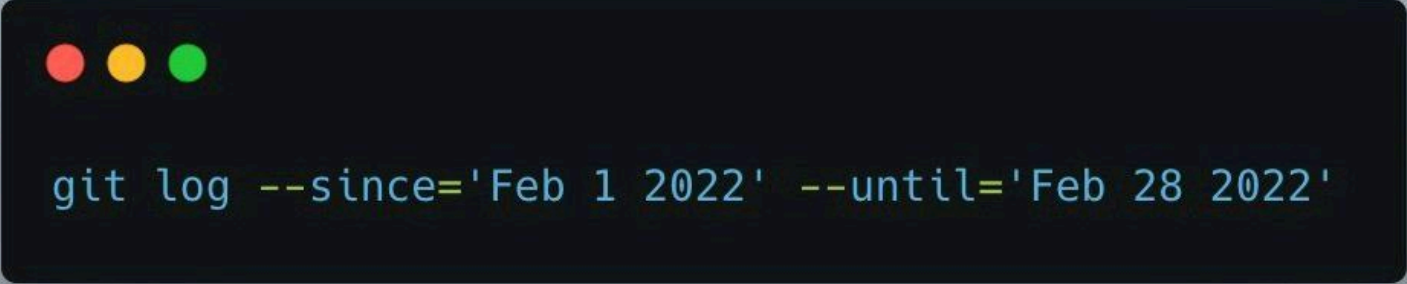
```
MINGW64~/c/Users/David/Desktop/repositorio-test
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/repositorio-test (main)
$ git log --all --decorate --oneline --graph
* 2759298 (HEAD -> main) Mi primer commit desde una rama
* 24e6851 (origin/main) Añado ficheros
```

Enlaces de interés

- [LearnGitBranching](https://learngitbranching.js.org/?locale=es_ES): nos permite aprender sobre varios aspectos de Git (como ramas, commits, etc) de forma muy visual.



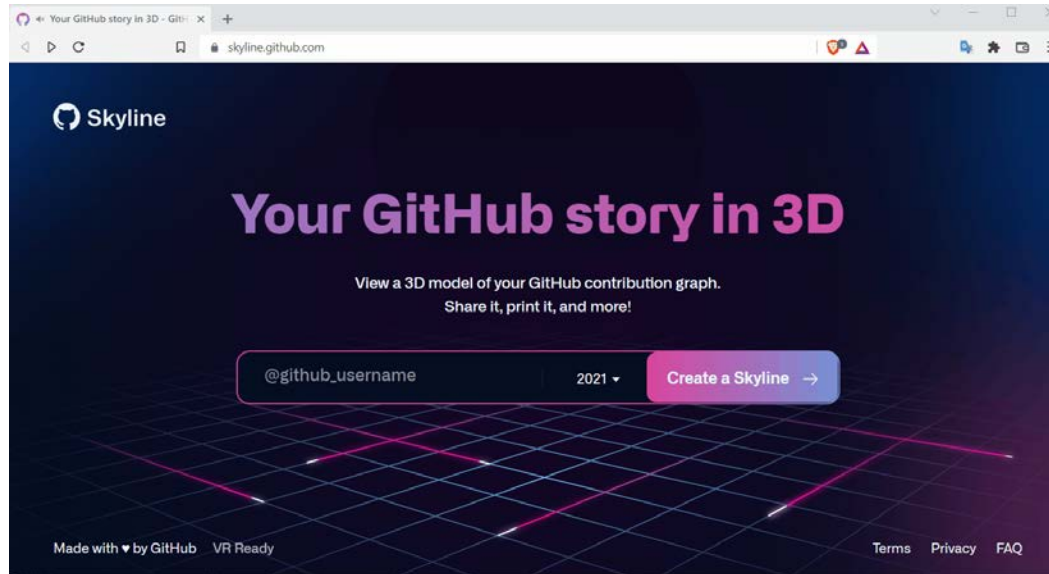
Truquito para filtrar fechas en un repo



```
git log --since='Feb 1 2022' --until='Feb 28 2022'
```

Truquito

- La siguiente web: <https://skyline.github.com/> nos permite realizar un skyline (paisaje) en función de los commits que hemos realizado.



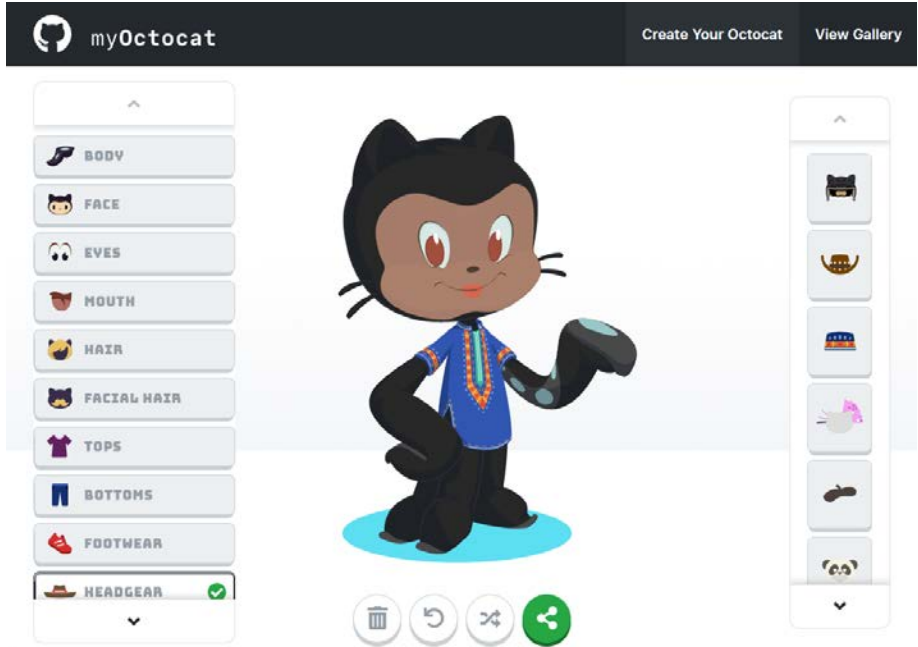
Finally... The GitHub bathroom



ALT

Octocat personalizado

- Octocat es la mascota de Git.
- La siguiente web:
<https://myoctocat.com/build-your-octocat/> nos permite crear/personalizar un octocat.



Enlaces de interés

- <https://gitexplorer.com/>

Normal type speed ☒ Fast type speed

Light Mode ☒ Dark Mode

Git Command Explorer

Find the right commands you need without digging through the web.

I want to:

clone

existing repo into a new directory

Usage

```
git clone <repo-url> <directory>
```

Note

The repo is cloned into the specified directory

Replace "directory" with the directory you want

Enlaces de interés

- <https://ohshitgit.com/es>

Oh Shit, Git!?!

Git es difícil: estropearlo es fácil y darse cuenta de cómo corregir tus errores es jodidamente imposible. La documentación de Git sufre del problema del huevo y la gallina, donde no puedes buscar cómo salir del lio, *a menos de que ya sepas el nombre de lo que tienes que saber* para poder arreglarlo.

Ejercicios

- Realiza las instrucciones definidas en el siguiente [readme.md del siguiente repositorio](#):

Práctica

Ejercicio de Git bash

1. Clona el repositorio <https://github.com/DavidBernalGonzalez/practicaGit.git>
2. Visualiza las distintas ramas que hay en el repositorio.
3. Dentro del repositorio hay una carpeta llamada practica01, la encontrarás en la rama "practica01". Sitúate en rama "practica01" para ver la carpeta practica01.
4. Sitúate en el directorio practica1 y edita los archivos a tu gusto. Una vez editados los ficheros, haz un git status para ver que ha pasado con los ficheros.
5. Crea una rama nueva, llámala practica1_APELLIDOS_NOMBRE
6. Muévete a la rama que has creado (practica1_APELLIDOS_NOMBRE)
7. En el directorio raíz, crea el fichero practica1_APELLIDOS_NOMBRE.txt y cualquier cosa en su interior. Y haz un commit de los cambios.
8. ¡Nos hemos equivocado! Queremos que en el interior del fichero practica1_APELLIDOS_NOMBRE.txt tendremos que poner nuestro nombre y apellidos. Vuelve a hacer otro commit revirtiendo los cambios (haciendo un nuevo commit) y sube los cambios al repositorio remoto:
9. Vuelve a la rama "practica1".
10. Haz un merge de tu rama con practica1
11. Añade dentro de la rama de tu carpeta todas los comandos que has utilizado. Puedes crear un documento por ejemplo de Word.

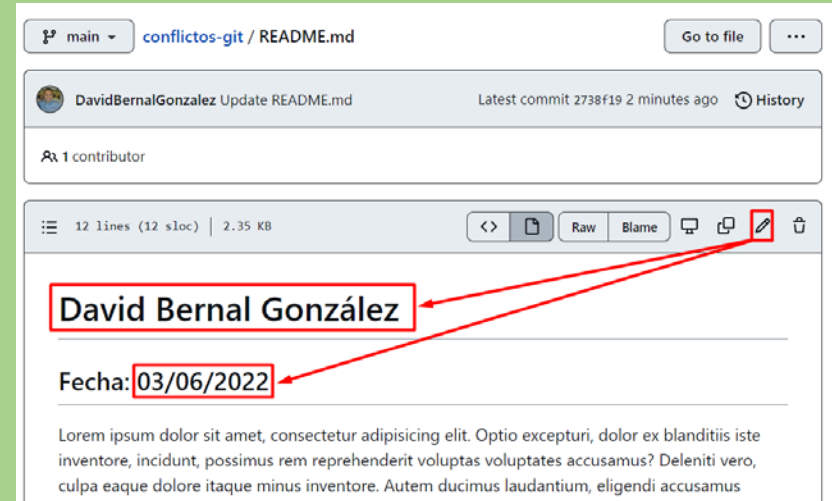
Ejercicio de como trabajar con Git desde un IDE

Probablemente en tu proyecto te den libertad para usar el cliente git y el IDE que prefieras, por lo que está muy bien que te vayas acostumbrando a trabajar con alguno

1. Visual Studio Code: <https://code.visualstudio.com/>
2. Vamos a utilizar VSC para clonar el repositorio <https://github.com/DavidBernalGonzalez/practicaGit.git>
3. Abre el proyecto desde VSV e investiga sobre como realizar un commit, push, crear rama, "mergear", borrar rama y resolver conflictos con los clientes.

Ejercicios de resolución de conflictos

- 1. Crea un repositorio llamado conflictos-git y añade un readme.md (puedes marcar la pestaña add readme.md cuando lo creas desde GitHub)
- 2. Añade en el repositorio [el siguiente contenido](#)
- 3. Clona el repositorio remoto conflictos-git a tu dispositivo (git clone).
- 4. Modifica desde GitHub, el repositorio poniendo tu nombre y la fecha de hoy en lugar del actual.
- ⚠ IMPORTANTE ⚠ NO HAGAS GIT PULL EN EL REPOSITORIO LOCAL**



Nombre modificado desde Git Hub

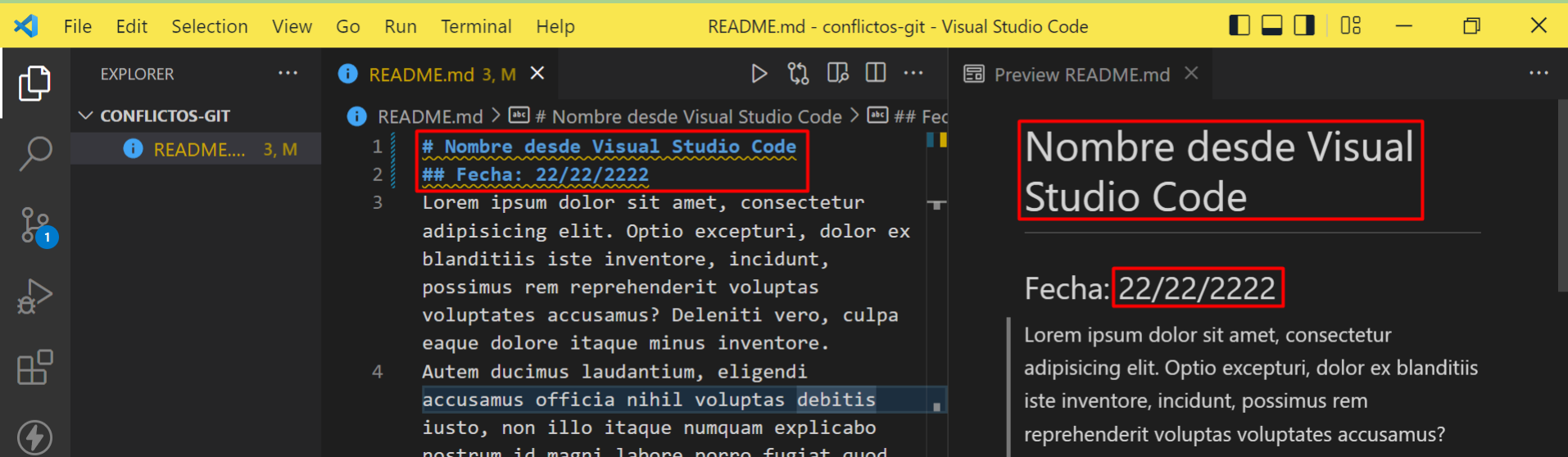
Fecha: **11/11/1111**

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Optio excepturi, dolor ex blanditiis iste inventore, incidunt, possimus rem reprehenderit voluptas voluptates accusamus? Deleniti vero, culpa eaque dolore itaque minus inventore. Autem ducimus laudantium, eligendi accusamus

Ejercicios de resolución de conflictos

- 5. Ves al repositorio local y modifica el nombre por Jaimito Fernández y pon como fecha 22/22/2222

⚠ IMPORTANTE ⚠ NO HAGAS GIT PULL EN EL REPOSITORIO LOCAL



Visual Studio Code interface showing a README.md file with a conflict resolution exercise. The file content includes a comment to change the name to 'Jaimito Fernández' and the date to '22/22/2222'. The Explorer sidebar shows the file is named 'README... 3, M'. A preview window on the right shows the rendered HTML of the README.

```
1 # Nombre desde Visual Studio Code
2 ## Fecha: 22/22/2222
3 Lorem ipsum dolor sit amet, consectetur
  adipiscing elit. Optio excepturi, dolor ex
  blanditiis iste inventore, incididunt,
  possimus rem reprehenderit voluptas
  voluptates accusamus? Deleniti vero, culpa
  eaque dolore itaque minus inventore.
4 Autem ducimus laudantium, eligendi
  accusamus officia nihil voluptas debitis
  iusto, non illo itaque numquam explicabo
  nostrum id magni labore porro fugiat quod
```

Nombre desde Visual Studio Code

Fecha: 22/22/2222

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Optio excepturi, dolor ex blanditiis iste inventore, incididunt, possimus rem reprehenderit voluptas voluptates accusamus?

Ejercicios de resolución de conflictos

- 6. Intenta hacer un push al repositorio remoto

 **IMPORTANTE**  **NO HAGAS GIT PULL EN EL REPOSITORIO LOCAL**

```
$ git status
On branch main
Your branch is up to date with 'origin/main'.
```

```
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
       modified:   README.md
```

no changes added to commit (use "git add" and/or "git commit -a")

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/conflictos-git (main)
$ git add .
```

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/conflictos-git (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.
```

```
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
       modified:   README.md
```

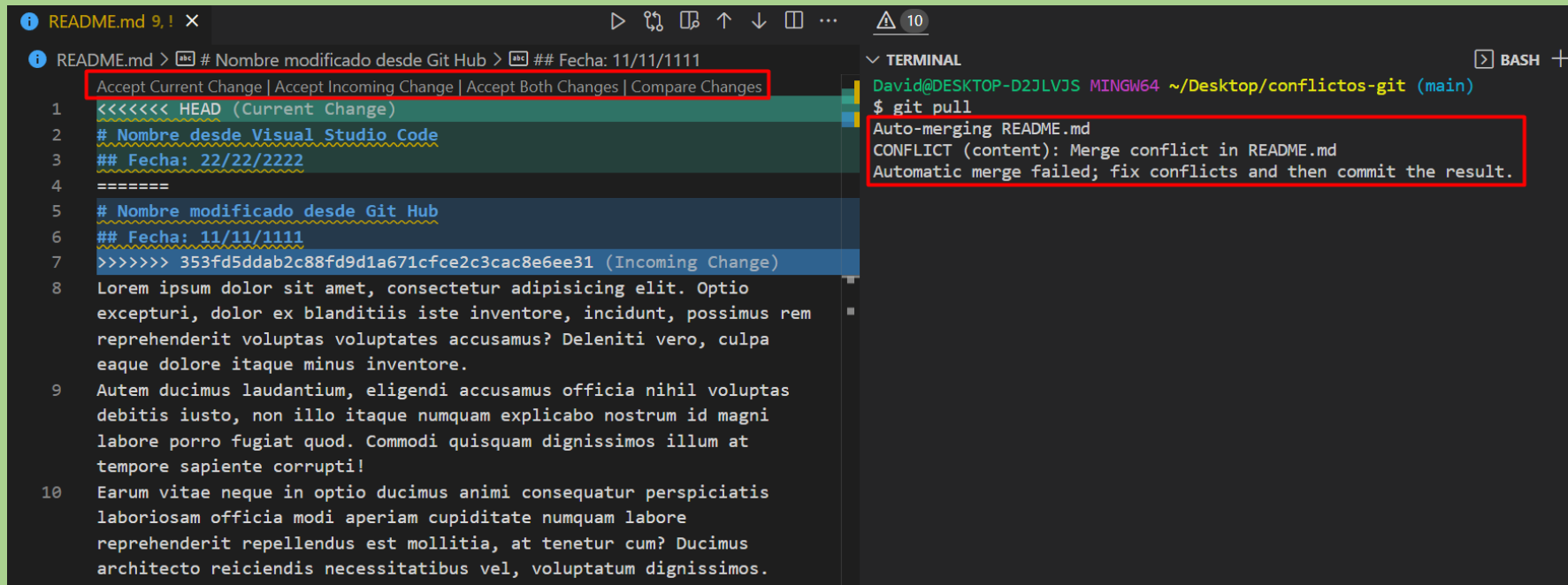
```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/conflictos-git (main)
$ git commit -m "Modifico readme.md"
[main 4dc12b6] Modifico readme.md
 1 file changed, 2 insertions(+), 2 deletions(-)
```

```
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/conflictos-git (main)
$ git status
On branch main
Your branch and 'origin/main' have diverged,
and have 1 and 2 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)

nothing to commit, working tree clean
```

Ejercicios de resolución de conflictos

- 7. Haz un git full y ves que Git no puede auto-resolver el conflicto. Por lo que nos deja la responsabilidad a nosotros como programadores:



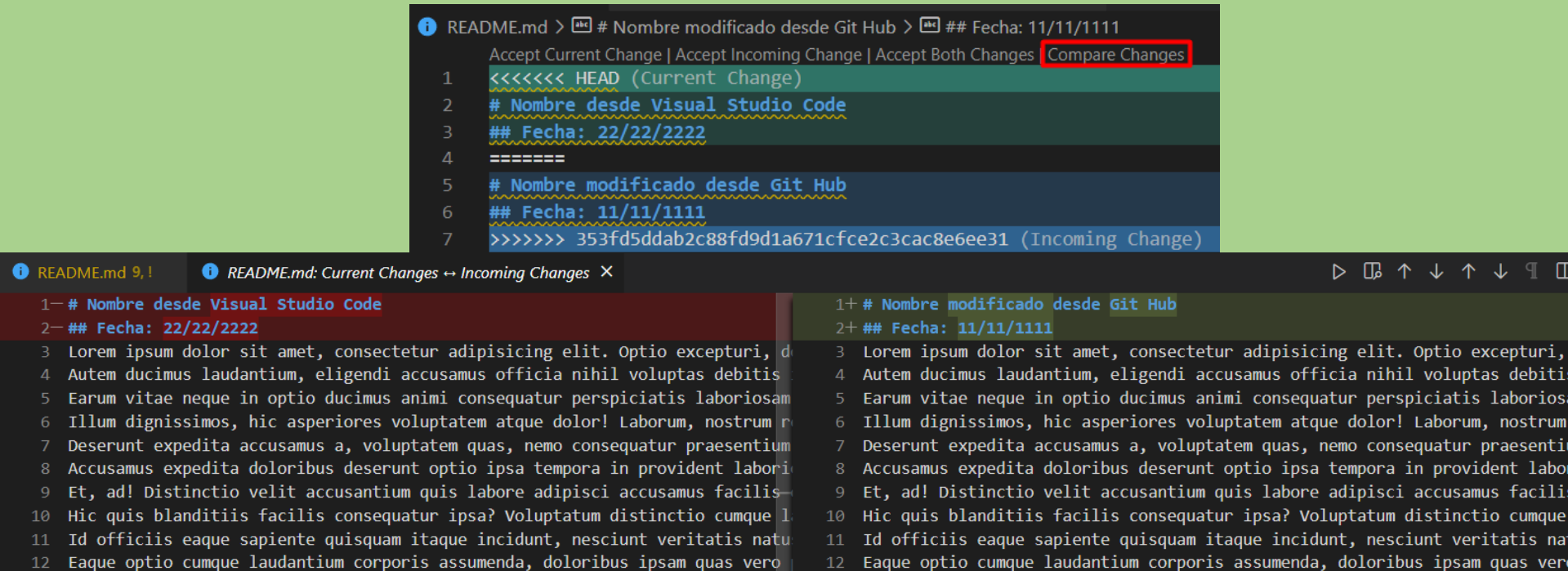
The image shows a VS Code editor window with a file named README.md. The editor displays a merge conflict between a local change and an incoming change from GitHub. The local change (HEAD) has a comment '# Nombre desde Visual Studio Code' and a date '## Fecha: 22/22/2222'. The incoming change (353fd5ddab2c88fd9d1a671cfce2c3cac8e6ee31) has a comment '# Nombre modificado desde Git Hub' and a date '## Fecha: 11/11/1111'. The conflict is resolved by accepting the incoming change, as indicated by the 'Accept Incoming Change' button being highlighted in the conflict resolution toolbar. The terminal window on the right shows the command '\$ git pull' being executed, resulting in an 'Auto-merging README.md' and a 'CONFLICT (content): Merge conflict in README.md' message. The terminal also displays the instruction 'Automatic merge failed; fix conflicts and then commit the result.'

```
README.md 9,1 X
README.md > ## # Nombre modificado desde Git Hub > ## Fecha: 11/11/1111
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
1 <<<<<< HEAD (Current Change)
2 # Nombre desde Visual Studio Code
3 ## Fecha: 22/22/2222
4 =====
5 # Nombre modificado desde Git Hub
6 ## Fecha: 11/11/1111
7 >>>>>> 353fd5ddab2c88fd9d1a671cfce2c3cac8e6ee31 (Incoming Change)
8 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Optio
excepturi, dolor ex blanditiis iste inventore, incidunt, possimus rem
reprehenderit voluptas voluptates accusamus? Deleniti vero, culpa
eaque dolore itaque minus inventore.
9 Autem ducimus laudantium, eligendi accusamus officia nihil voluptas
debitis iusto, non illo itaque numquam explicabo nostrum id magni
labore porro fugiat quod. Comodi quisquam dignissimos illum at
tempore sapiente corrupti!
10 Earum vitae neque in optio ducimus animi consequatur perspiciatis
laboriosam officia modi aperiam cupiditate numquam labore
reprehenderit repellendus est mollitia, at tenetur cum? Ducimus
architecto reiciendis necessitatibus vel, voluptatum dignissimos.

TERMINAL
David@DESKTOP-D2JLVJS MINGW64 ~/Desktop/conflictos-git (main)
$ git pull
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

Ejercicios de resolución de conflictos

- 7. Haz un git pull y ves que Git no puede auto-resolver el conflicto. Por lo que nos deja la responsabilidad a nosotros como programadores:



```

i README.md > # Nombre modificado desde Git Hub > ## Fecha: 11/11/1111
Accept Current Change | Accept Incoming Change | Accept Both Changes Compare Changes
1 <<<<<< HEAD (Current Change)
2 # Nombre desde Visual Studio Code
3 ## Fecha: 22/22/2222
4 =====
5 # Nombre modificado desde Git Hub
6 ## Fecha: 11/11/1111
7 >>>>>> 353fd5ddab2c88fd9d1a671cfce2c3cac8e6ee31 (Incoming Change)

i README.md 9: i README.md: Current Changes ↔ Incoming Changes X
1- # Nombre desde Visual Studio Code
2- ## Fecha: 22/22/2222
3 Lorem ipsum dolor sit amet, consectetur adipisicing elit. Optio excepturi, d
4 Autem ducimus laudantium, eligendi accusamus officia nihil voluptas debitis
5 Earum vitae neque in optio ducimus animi consequatur perspiciatis laboriosam
6 Illum dignissimos, hic asperiores voluptatem atque dolor! Laborum, nostrum r
7 Deserunt expedita accusamus a, voluptatem quas, nemo consequatur praesentium
8 Accusamus expedita doloribus deserunt optio ipsa tempora in provident labori
9 Et, ad! Distinctio velit accusantium quis labore adipisci accusamus facilis
10 Hic quis blanditiis facilis consequatur ipsa? Voluptatum distinctio cumque l
11 Id officiis eaque sapiente quisquam itaque incidunt, nesciunt veritatis natu
12 Eaque optio cumque laudantium corporis assumenda, doloribus ipsam quas vero

1+ # Nombre modificado desde Git Hub
2+ ## Fecha: 11/11/1111
3 Lorem ipsum dolor sit amet, consectetur adipisicing elit. Optio excepturi,
4 Autem ducimus laudantium, eligendi accusamus officia nihil voluptas debitis
5 Earum vitae neque in optio ducimus animi consequatur perspiciatis laborios
6 Illum dignissimos, hic asperiores voluptatem atque dolor! Laborum, nostrum
7 Deserunt expedita accusamus a, voluptatem quas, nemo consequatur praesenti
8 Accusamus expedita doloribus deserunt optio ipsa tempora in provident labor
9 Et, ad! Distinctio velit accusantium quis labore adipisci accusamus facilis
10 Hic quis blanditiis facilis consequatur ipsa? Voluptatum distinctio cumque
11 Id officiis eaque sapiente quisquam itaque incidunt, nesciunt veritatis nat
12 Eaque optio cumque laudantium corporis assumenda, doloribus ipsam quas vero
```

Conflictos de Git

- Existe la posibilidad de ver los cambios con git diff de la siguiente manera:

```
David@DESKTOP-D2JLV7S MINGW64 ~/Desktop/conflictos-git (main|MERGING)
```

```
$ git diff README.md
```

```
diff --cc README.md
```

```
index 20cefcd,240f8d4..0000000
```

```
--- a/README.md
```

```
+++ b/README.md
```

```
@@@ -1,5 -1,5 +1,10 @@@
```

```
++<<<<<< HEAD
```

```
+ # Nombre desde Visual Studio Code
```

```
+ ## Fecha: 22/22/2222
```

```
++=====
```

```
+ # Nombre modificado desde Git Hub
```

```
+ ## Fecha: 11/11/1111
```

```
++>>>>>> 353fd5ddab2c88fd9d1a671cfce2c3cac8e6ee31
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Optio excepturi, dolor ex blanditiis iste inventore, incididunt, possimus rem reprehenderit voluptas voluptates accusamus? Deleniti vero, culpa eaque dolore itaque minus inventore.

Autem ducimus laudantium, eligendi accusamus officia nihil voluptas debitis iusto, non illo itaque numquam explicabo nostrum id magni labore porro fugiat quod. Commodi quisquam dignissimos illum at tempore sapiente corrupti!

Earum vitae neque in optio ducimus animi consequatur perspiciatis laboriosam officia modi aperiam cupiditate numquam labore reprehenderit repellendus est mollitia, at tenet cum? Ducimus architecto reiciendis necessitatibus vel, voluptatum dignissimos.



Ejercicios de resolución de conflictos

- 8. Piensa en que versión quieres, es muy importante estar seguro de lo que queremos hacer para no tener dolores de cabeza y no darle a cualquier opción sin pensarlo. ¿Qué versión quiero? ¿Con que cambios me quiero quedar? Y finalmente, soluciona el conflicto

```
1 <<<<<< HEAD (Current Change)
2 # Nombre desde Visual Studio Code
3 ## Fecha: 22/22/2222
4 =====
5 # Nombre modificado desde Git Hub
6 ## Fecha: 11/11/1111
7 >>>>>> 353fd5ddab2c88fd9d1a671cfce2c3cac8e6ee31 (Incoming Change)
```

```
1- # Nombre desde Visual Studio Code
2- ## Fecha: 22/22/2222
3 Lorem ipsum dolor sit amet, consectetur adipisicing elit. Optio excepturi, d
4 Autem ducimus laudantium, eligendi accusamus officia nihil voluptas debitis
5 Earum vitae neque in optio ducimus animi consequatur perspiciatis laboriosam
6 Illum dignissimos, hic asperiores voluptatem atque dolor! Laborum, nostrum r
7 Deserunt expedita accusamus a, voluptatem quas, nemo consequatur praesentium
8 Accusamus expedita doloribus deserunt optio ipsa tempora in provident labori
9 Et, ad! Distinctio velit accusantium quis labore adipisci accusamus facilis
10 Hic quis blanditiis facilis consequatur ipsa? Voluptatum distinctio cumque l
11 Id officiis eaque sapiente quisquam itaque incidunt, nesciunt veritatis natu
12 Eaque optio cumque laudantium corporis assumenda, doloribus ipsam quas vero
```

```
1+ # Nombre modificado desde Git Hub
2+ ## Fecha: 11/11/1111
3 Lorem ipsum dolor sit amet, consectetur adipisicing elit. Optio excepturi,
4 Autem ducimus laudantium, eligendi accusamus officia nihil voluptas debitis
5 Earum vitae neque in optio ducimus animi consequatur perspiciatis laboriosa
6 Illum dignissimos, hic asperiores voluptatem atque dolor! Laborum, nostrum
7 Deserunt expedita accusamus a, voluptatem quas, nemo consequatur praesenti
8 Accusamus expedita doloribus deserunt optio ipsa tempora in provident labor
9 Et, ad! Distinctio velit accusantium quis labore adipisci accusamus facilis
10 Hic quis blanditiis facilis consequatur ipsa? Voluptatum distinctio cumque
11 Id officiis eaque sapiente quisquam itaque incidunt, nesciunt veritatis nat
12 Eaque optio cumque laudantium corporis assumenda, doloribus ipsam quas vero
```

Ejercicios de resolución de conflictos

- Basándote en el ejercicio anterior crea otro fichero (puedes utilizar el mismo) y tal y como hemos en el ejemplo anterior vamos a tener conflictos pero esta vez muchos más.
- Por ejemplo:
 - ☐ En el repositorio remoto, puedes cambiar todas las letras a por una @.
 - ☐ En el repositorio local, puedes cambiar todas las letras e por €.
- Haz que se generen conflictos compáralos analiza que es lo que quieres. En muchas ocasiones necesitaras incluso sentarte con otro compañero para ver conflicto a conflicto que dejáis, etc.
- Finalmente, resuélvelos en algunas ocasiones cogiendo los cambios de tu repositorio local y en otras ocasiones los cambios de tu repositorio remoto.