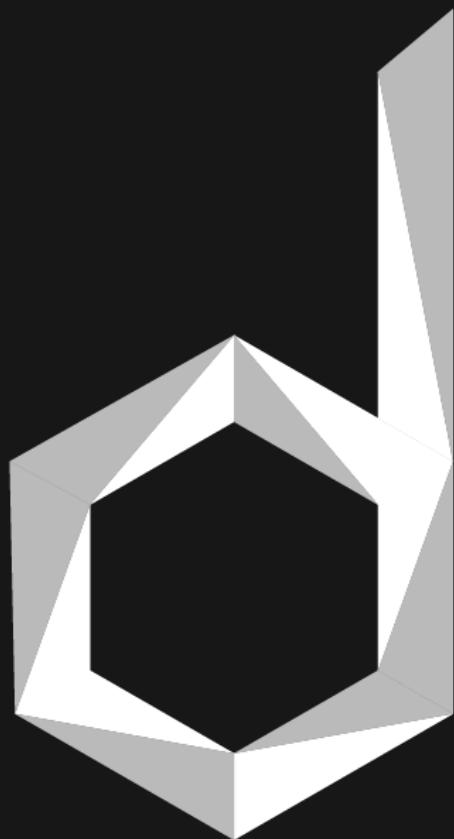


# INGENIERÍA MECATRÓNICA



## DI\_CERO

DIEGO CERVANTES RODRÍGUEZ

DEVOPS Y CONTROL DE VERSIONES DE ARCHIVOS DE TEXTO PLANO

GIT Y GITHUB

DevOps: Manual de Git y GitHub

# Manual de Git

## Contenido

¿Qué es Git? .....	4
Comandos de la Consola Windows: CMD (MS-DOS).....	5
○ pwd .....	5
○ cd nombre_carpeta.....	5
○ ls .....	5
○ clear.....	5
○ mkdir .....	5
○ history .....	5
○ touch nombre_archivo.extension .....	6
○ cat nombre_archivo.extension .....	6
○ vi nombre_archivo.extension .....	6
○ rm nombre_archivo.extension.....	6
○ nombre_comando --help .....	6
Conceptos Git .....	6
Etapas de Git: .....	6
Ramas de Git: .....	7
Comandos de Repositorios Git .....	8
Crear un Repositorio Local Nuevo:.....	8
1. pwd .....	8
2. Datos de usuario Git.....	8
a. git config --global user.name "nombre_de_usuario" .....	8
b. git config --global user.email "email_del_usuario" .....	8
i. git config --list .....	8
3. git init .....	9
Manejo de cambios en las etapas de Git:.....	9
4. git status.....	9
5. git add -A.....	10
c. git rm nombre_archivo.extension .....	10
i. git rm --cached nombre_archivo.extension .....	10
6. git commit -m "mensaje de esa versión" .....	11
a. git commit -am "mensaje de esa versión".....	11
b. git commit --ammend .....	11
Acceso a la línea de tiempo de Git/Comparación de cambios entre versiones: .....	13
7. git log.....	13
a. git log nombre_archivo.extension .....	14
b. git log --stat.....	14
8. git show.....	15
a. git show nombre_archivo.extension .....	15
9. git diff número_commit_1 número_commit_2 .....	16

10.	git checkout número_commit_versión_anterior nombre_archivo.extension .....	18
b.	git checkout nombre_rama nombre_archivo.extension .....	18
11.	git reset.....	19
a.	git reset número_commit_versión_anterior --hard .....	19
b.	git reset número_commit_versión_anterior --soft .....	19
c.	git reset HEAD.....	19
	Manejo de ramas en Git: .....	20
12.	git branch nombre_rama_nueva .....	21
a.	git branch -D nombre_rama_a_borrar .....	21
13.	git checkout nombre_rama .....	22
14.	git branch.....	25
15.	git merge nombre_rama_a_fusionar.....	26
16.	git log --all --graph .....	26
a.	git log --all --graph --decorate --oneline .....	27
i.	alias nombre_variable_consola = “comando_largo_a_guardar” .....	27
17.	git show-branch .....	28
a.	git show-branch --all .....	28
b.	gitk .....	28
	Manejo de conflictos al fusionar ramas: .....	29
	<b>Creación de un repositorio remoto en GitHub: .....</b>	<b>32</b>
	Conectar un repositorio local con uno remoto dentro de GitHub:.....	36
18.	git remote add nombre_repositorio url_https_extraida_de_github .....	37
a.	git remote remove origin .....	38
19.	git remote .....	38
a.	git remote -v .....	38
20.	git pull origin nombre_rama_a_mandar.....	40
b.	git pull origin nombre_rama_a_mandar –allow-unrelated-histories.....	40
21.	git push origin nombre_rama_a_mandar.....	41
	Cifrado adicional entre repositorios locales y remotos de Git Hub (llaves SSH): .....	46
22.	ssh-keygen -t rsa -b 4096 -C “email_del_usuario” .....	49
i.	ssh-keygen -p .....	50
2.	eval \$(ssh-agent -s).....	51
3.	ssh-add ~/ssh/id_rsa .....	51
23.	git remote set-url nombre_repositorio url_ssh_extraida_de_github .....	53
a.	git remote remove origin .....	54
	Releases de versiones en proyectos de GitHub (tags): .....	56
24.	git tag -a nombre_tag “nuevo_mensaje_del_commit” número_commit .....	56
25.	git tag .....	56
a.	git show-ref --tags.....	56
i.	git push origin --tags .....	56
b.	git tag -d nombre_tag .....	58
i.	git push origin :refs/tags/nombre_tag.....	58
	Manejo de ramas en un repositorio remoto/Pull Request: .....	59
	<b>Extraer contenido de un repositorio remoto a uno local y Colaborar en él: .....</b>	<b>71</b>
26.	git clone url_http_o_ssh_del_repositorio_remoto .....	74
27.	git pull .....	75
a.	git fetch.....	75
i.	git merge .....	75

Contribuir con proyectos de código abierto en GitHub (Forks): .....	75
Conectar un dominio de algún servidor con un proyecto GitHub: .....	81
<b>Manejo de errores/condiciones en Git.....</b>	<b>82</b>
Creación del archivo .gitignore: .....	82
Uso del hosting gratuito de GitHub (GitHub Pages): .....	85
Eliminación de una rama externa y fusión de su historial (rebase):.....	93
28.     git rebase nombre_rama .....	93
29.     git stash.....	97
a.     git stash list.....	97
i.     git stash drop .....	97
b.     git stash pop .....	97
c.     git stash branch nombre_nueva_rama.....	97
30.     git cherry-pick número_commit_rama_distinta.....	99
Resolución de errores fatales en la línea de tiempo del proyecto:.....	99
31.     git reflog.....	100
32.     git grep palabra_a_buscar .....	103
a.     git grep -n palabra_a_buscar .....	103
b.     git grep -c palabra_a_buscar .....	103
33.     git log -S "etiqueta_a_buscar" .....	103



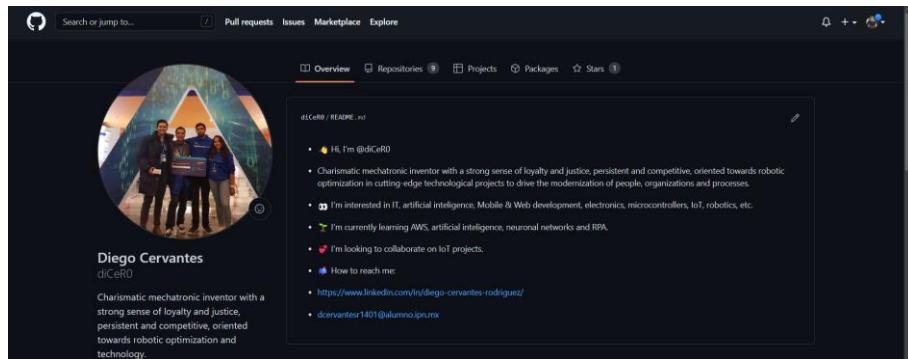
# ¿Qué es Git?

Git sirve para el **control de versiones**, esto se refiere a los diferentes cambios que puede tener principalmente un archivo de texto plano (con código de programación), de esta manera cuando cometa un error en una versión nueva, **puedo regresar en el tiempo a donde no existía el error presente en el proyecto, es posible que existan variantes distintas de un archivo al mismo tiempo y además varias personas pueden trabajar a la vez en un proyecto de una forma más sencilla.**

Los sistemas de control de versiones usualmente indican en cada versión:

- La fecha de cada cambio.
- La hora de cada cambio.
- El autor del cambio.
- Un mensaje del cambio.

GitHub fue hecho por el creador del sistema operativo Linux llamado Linus Torvalds.



La consola que se utiliza para mandar carpetas de mi ordenador a GitHub se llama **Git Bash**, con ella se crean **Repositorios locales** en mi computadora para luego asociarlos con los **Repositorios remotos** almacenados en la nube de GitHub, explicaremos el término repositorio después.

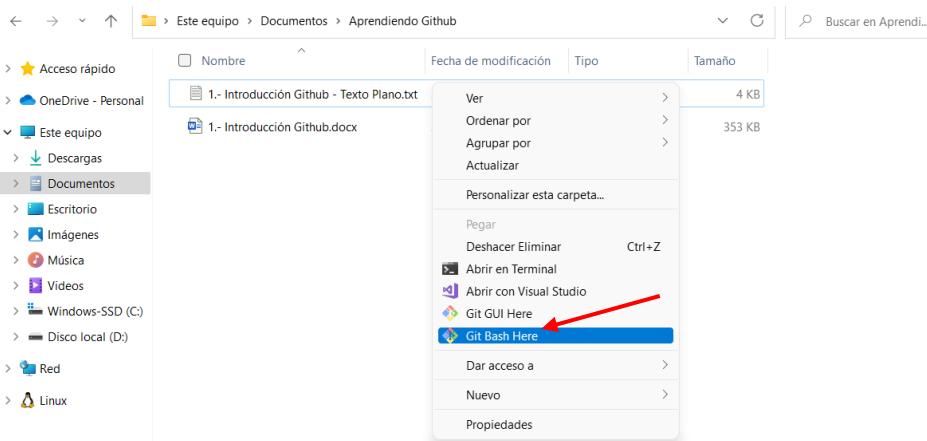
A screenshot of a terminal window titled 'MINGW64:/c/Users/diego/AppData/Roaming/SPB\_Data'. The window shows a command prompt with the text 'diego@LaPoderosa MINGW64 ~ \$ |'.

Git está hecho principalmente para guardar y editar versiones de archivos de texto plano como lo son: Archivos de código y de texto hechos con el bloc de notas, no reconocerá de forma tan precisa los cambios hechos en archivos binarios como lo son: Archivos de Word, imágenes, PDF, archivos CAD (SolidWorks), etc. pero si los puede subir a la nube.

Las mejores prácticas indican que los archivos binarios no se deberían agregar a repositorios, ya que los cambios realizados en ellos no serán reconocidos automáticamente en el sitio web de GitHub o no se actualizarán correctamente (a menos que use el modo incógnito de mi navegador o presione las teclas: **CTRL + SHIFT + R**), además de que mientras más archivos binarios agregue a GitHub, más pesado se vuelve el proyecto en la nube.

# Comandos de la Consola Windows: CMD (MS-DOS)

- **pwd**: Muestra el directorio (ruta de carpetas) en el que nos encontramos.
- **cd nombre\_carpeta**: El comando change directory sirve para introducirse en una carpeta en específico del directorio actual, para ello la consola se debe encontrar en una carpeta superior a la otra donde nos queremos introducir. *Si ponemos solo unas cuantas letras del nombre de la carpeta y presionamos la tecla TAB, el nombre se completa por sí solo.*
  - **cd ..**: Comando para retroceder en el directorio, osea salir de la carpeta donde nos encontramos.
  - **cd directorio**: Se utiliza para llevar la Git Bash a una dirección en específico, no importando dentro de qué carpeta me encuentre actualmente.
    - **Para que este comando funcione correctamente debemos cambiar los signos \ por / en el nombre del directorio, además no funciona para entrar de golpe a direcciones con nombres muy largos.**
      - *Cuando el nombre del directorio es muy largo a veces sirve poner un slash (/) en medio de palabras para que lo acepte Git Bash.*
  - Además de utilizar el comando **cd**, también simplemente se puede ir al explorador de archivos, dar clic derecho dentro de la carpeta donde quiero estar y seleccionar la opción de **Git Bash Here**.



- **ls**: Comando que muestra todos los **archivos y carpetas no ocultas** que se encuentran dentro de la cual estamos situados, **pero NO en forma de lista**.
  - **ls -al**: Sirve para mostrar un **listado de los archivos y carpetas ocultas y no ocultas** que se encuentren dentro de la carpeta actual.
  - **ls -l**: Sirve para mostrar un **listado de los archivos y carpetas no ocultas** que se encuentren dentro de la carpeta actual.
  - **ls -a**: Sirve para mostrar los **archivos y carpetas ocultas y no ocultas** que se encuentren dentro de la carpeta actual, **pero NO en forma de lista**.
- **clear**: Comando que limpia el historial de comandos que haya en la consola Git Bash, también se puede hacer con las teclas: **CTRL + L**.
- **mkdir**: Comando make directory que sirve para crear una nueva carpeta.
- **history**: Muestra por orden todos los comandos que se han ejecutado en el Git Bash.
  - **history -d**: Comando que borra todo el historial de comandos ejecutados.

- **i**<sub>num\_history</sub>: Ejecuta un comando del history indicado por su número.

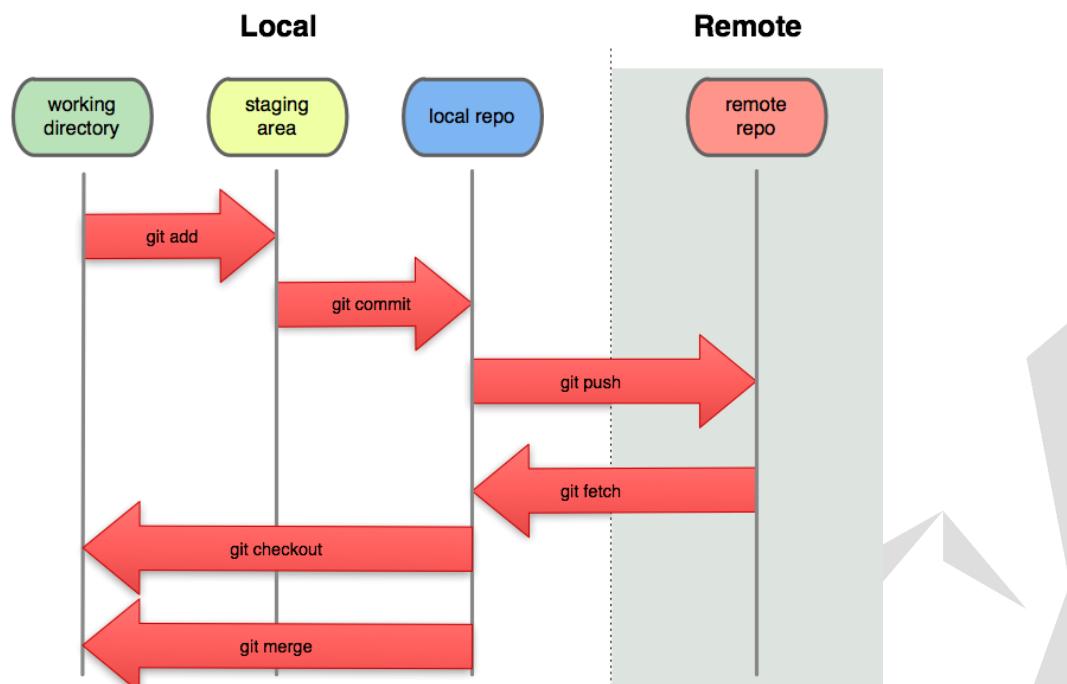
Los siguientes comandos que sirven para editar archivos dependen de que tan largo sea el nombre del archivo y/o directorio, además de que pueden ser bloqueados por el antivirus para realizar su función.

- **touch nombre\_archivo.extension**: Permite crear archivos vacíos, se le debe indicar la extensión y nombre del archivo.
- **cat nombre\_archivo.extension**: Comando que permite leer y mostrar en consola el contenido de un archivo de texto plano.
- **vi nombre\_archivo.extension**: Comando que permite editar un archivo.
- **rm nombre\_archivo.extension**: Comando que permite borrar un archivo.
- **code**: Abre el editor de texto Visual Studio Code en la carpeta donde está Git Bash.
  - **code nombre\_archivo.extension**: Comando que abre un archivo específico con el editor de texto Visual Studio Code desde la Git Bash.
- **nombre\_comando --help**: Sirve para que se muestre una descripción dada por la misma consola de Git Bash de cada uno de los comandos descritos anteriormente.

## Conceptos Git

### Etapas de Git:

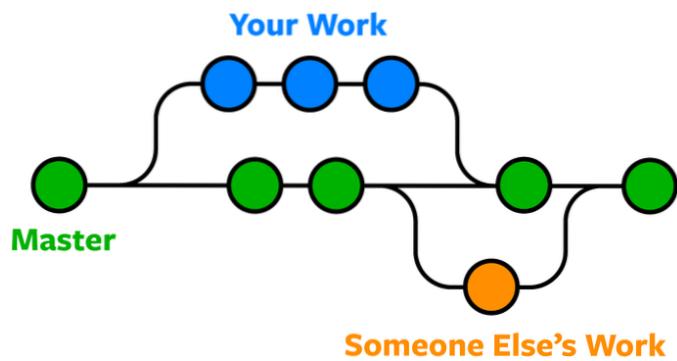
En Git existen 4 etapas para subir un código o texto plano a un repositorio, donde repositorio se refiere a una base de datos que almacenará las diferentes versiones de un proyecto, pero recordemos que existen dos tipos de repositorios en GitHub, el **local** y el **remoto**.



- **Working Directory:** Es la zona también llamada ***untracked*** donde se pueden hacer cambios en el código que todavía no están intencionados a subirse al repositorio.
- **Staging Area:** Es la zona también llamada ***tracked*** que se refiere a la parte media entre haber hecho cambios en el código y subirlos al repositorio local, en esta parte se pueden tener varios cambios considerados a subirse, sin haberse subido todavía al repositorio local. Se lleva a cabo con el comando **git add**.
- **Repositorio local:** En el repositorio local podemos acceder a las diferentes versiones de nuestro proyecto por medio de la consola Git Bash, pero dicha base de datos de las versiones del proyecto no está subida a la nube de GitHub, solamente existe en nuestro ordenador, específicamente en la carpeta .git que es invisible, por lo cual, si se llegara a descomponer el ordenador, se pierde el proyecto con todo y sus distintas versiones. Se lleva a cabo con el comando **git commit**.
- **Repositorio remoto:** Es una copia del repositorio local que está guardada en la nube utilizando la plataforma de GitHub, GitLab, BitBucket, un servidor remoto propio, etc. Se utiliza el comando **git push** para mandar los cambios a la nube o **git pull** para traer los cambios de un repositorio remoto subido a GitHub hacia un repositorio local.

## Ramas de Git:

Git no solo sirve para el manejo de versiones, sino también para **permitir que varias personas trabajen en un solo proyecto al mismo tiempo, para eso sirven las ramas o branches**, las cuales son herramientas útiles porque los proyectos muy complejos es mejor dividirlos en diferentes partes que se trabajen en paralelo sobre un mismo archivo, cada rama se verá como si fuera un archivo totalmente diferente, con sus propias versiones almacenadas en su línea de tiempo individual, para que cuando estas ya funcionen de forma separada, se puedan unir y crear un solo archivo que tenga todas las partes trabajadas, concluyendo así un proyecto grande o que se quiera trabajar de forma modular.



La branch principal del proyecto (rama de **producción**) siempre se llama **master** en el **Repositorio local de Git Bash** y **main** en el **Repositorio remoto de Git Hub**, pero se puede crear un número ilimitado de ramas adicionales al proyecto, asignándoles a cada una un nombre distinto y al final uniéndolas por medio del comando **git merge** a la rama principal.

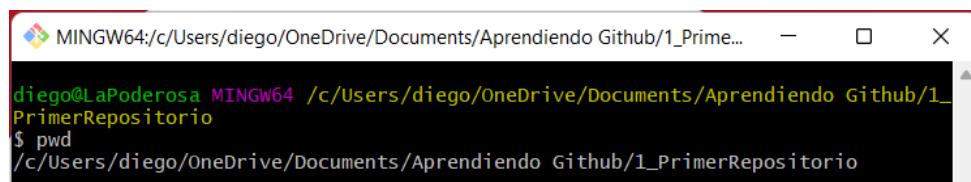
Es usual que cuando haya un bug (error) en el código, se cree una rama nueva llamada **hotfix** o **bugfixing** (rama de **debugging**) para resolverlo, ya que haya sido resuelto el problema, esta se une con la rama **master** o **main**.

Además, por buenas prácticas es habitual que se tenga una rama llamada **staging**, **development** o **staging develop** (rama de **desarrollo**) que es una copia de las ramas principales **master** o **main** y sirve para desarrollar el código que no estamos seguros si todavía funciona completamente, ya que estemos seguros de que su código funciona, se fusiona con la branch principal.

## Comandos de Repositorios Git

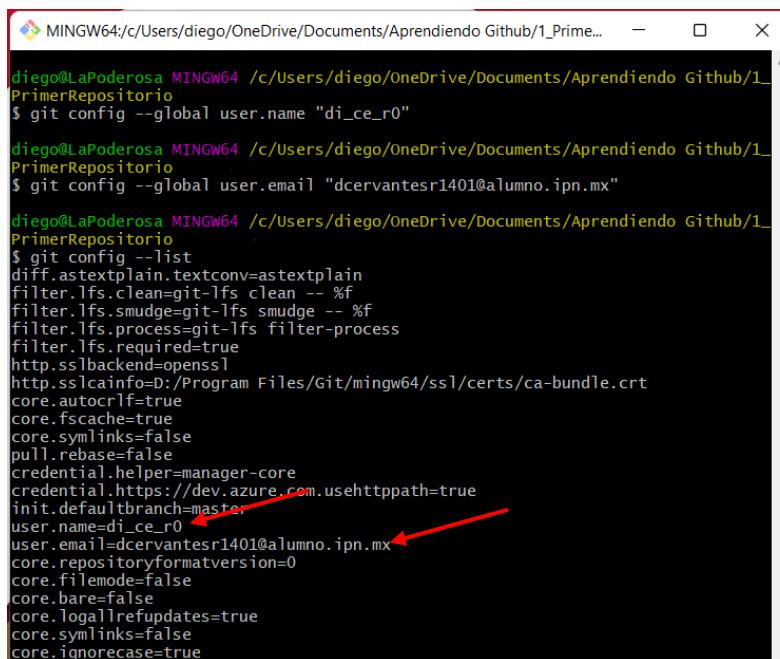
Crear un Repositorio Local Nuevo:

1. **pwd**: Primero que nada, por medio de los **comandos de consola CMD** dentro de la Git Bash **nos debemos situar en la carpeta donde se encuentren los archivos y carpetas que queremos subir a GitHub**.



```
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ pwd
/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
```

2. **Datos de usuario Git**: Como en cada versión del proyecto se indica la fecha, hora, autor y un mensaje del cambio, se debe declarar desde un inicio nuestro nombre y correo de usuario.
  - a. **git config --global user.name "nombre\_de\_usuario"**: Comando que declara el nombre de usuario.
  - b. **git config --global user.email "email\_del\_usuario"**: Comando que declara el email del usuario, *este debe ser el mismo correo asociado a la cuenta de Git Hub (donde queremos subir posteriormente el repositorio a la nube)*.
    - i. **git config --list**: Muestra un listado de las características del repositorio local, como lo es el nombre de usuario, correo, etc.

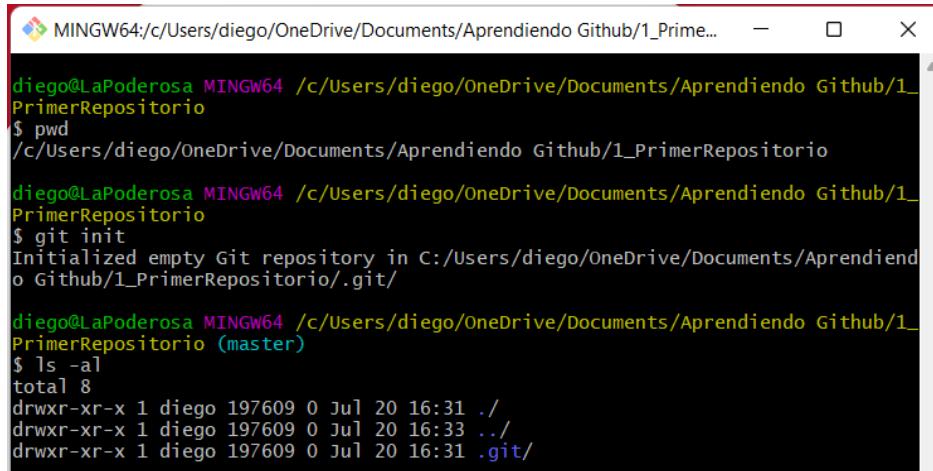


```
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git config --global user.name "di_ce_r0"
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git config --global user.email "dcervantesr1401@alumno.ipn.mx"

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=D:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.http://dev.azure.com/usehttppath=true
init.defaultbranch=master
user.name=di_ce_r0
user.email=dcervantesr1401@alumno.ipn.mx
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
```

3. **git init:** Comando que **crea un Repositorio local vacío** en cualquier carpeta de mi computadora, estableciendo así una base de datos que almacenará todas las posibles versiones de ese proyecto en específico.

*Cuando este comando se ejecuta bien, se crea una carpeta oculta (la cual es identifiable porque antes de su nombre lleva un punto) .git que solo puede ser vista con el comando ls -al, además de designar la rama a la que pertenece el repositorio creado, que en un inicio siempre será la rama master.*

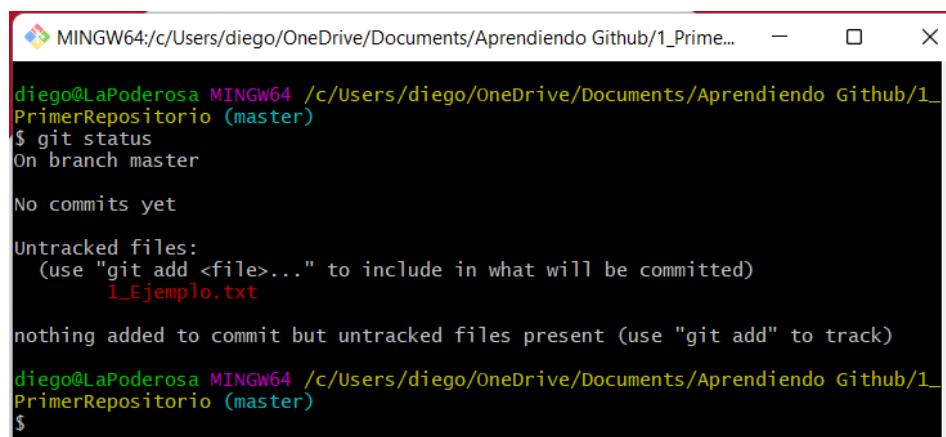


```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ pwd
/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git init
Initialized empty Git repository in C:/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio/.git/
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ ls -al
total 8
drwxr-xr-x 1 diego 197609 0 Jul 20 16:31 .
drwxr-xr-x 1 diego 197609 0 Jul 20 16:33 ../
drwxr-xr-x 1 diego 197609 0 Jul 20 16:31 .git/
```

Manejo de cambios en las etapas de Git:

4. **git status:** Comando para observar el estado en el que se encuentra el repositorio actual, indicando:

- La rama del repositorio.
- Si algún **cambio previamente detectado** se ha mandado al **Repositorio local** por medio de algún commit.
- Si los **nuevos cambios detectados** se encuentran en el **Working directory** o **Staging Area**.
  - Los cambios que se encuentren en la etapa de **Working directory** se muestran con **letras rojas**.
  - Los cambios que se encuentren en la etapa de **Staging Area** se muestran con **letras verdes**.



```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git status
On branch master
No commits yet
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    1_Ejemplo.txt
nothing added to commit but untracked files present (use "git add" to track)
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$
```

5. **git add -A**: Comando que asocia al **Staging Area** todos los archivos contenidos en la carpeta donde previamente se creó un repositorio local con el comando **git init**.
  - a. **git add .**: Comando que asocia al **Staging Area** todos los archivos que hayan tenido un cambio desde la versión anterior. *Este es el mejor comando que se puede utilizar para mandar los cambios.*
  - b. **git add nombre\_archivo.extension**: Comando que asocia al **Staging Area** un archivo en específico de la carpeta.

*Al utilizar el comando **git add** se agregan los cambios del archivo o archivos al **Staging Area** para que estén listos a que posteriormente se les dé un **commit** y así sean agregados al **Repositorio local**.*

```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Prime...
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    1_Ejemplo.txt

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

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git add -A

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   1_Ejemplo.txt
```

*Para borrar un archivo de la **Staging Area** si es que todavía no lo queremos mandar al repositorio local se deben usar dos comandos uno después de otro, primero el comando **git rm** y luego el comando **git rm --cached**.*

- c. **git rm nombre\_archivo.extension**: Comando que saca un archivo del **Staging Area** para regresarlo al **Working Directory**, por sí solo no funciona porque debe borrarse igual de la memoria caché del **Staging Area** para que en realidad se pueda regresar al **Working Directory**.
  - i. **git rm --cached nombre\_archivo.extension**: Comando que se debe usar después de haber ejecutado el comando **rm** para quitar un archivo del **Staging Area**.

```

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git rm 1_Ejemplo.txt
error: the following file has changes staged in the index:
  1_Ejemplo.txt
(use --cached to keep the file, or -f to force removal)

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:  1_Ejemplo.txt

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git rm --cached 1_Ejemplo.txt
rm '1_Ejemplo.txt'

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    1_Ejemplo.txt

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

```

6. **git commit -m “mensaje de esa versión”:** Comando con el que se suben los cambios al **Repositorio local**, osea a la base de datos que guarda las versiones del proyecto, pero que solamente existe dentro de mí misma computadora, al mandar el cambio ya no saldrá ningún mensaje acerca de este cuando se utilice el comando **git status**, ya que se habrá vaciado la **Staging Area**.
- git commit -am “mensaje de esa versión”:** Comando que es igual a utilizar el comando **git add . y git commit -m “mensaje”** al mismo tiempo.
  - git commit --ammend:** Comando que sirve para añadir lo que sea que esté en el **Staging Area** al último commit mandado al **Repositorio local**, es una forma de editar el último cambio mandado, hasta permitiéndome cambiar el mensaje del commit. Antes de ejecutar este comando se tuvo que haber usado **git add**.

```

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git commit -m "Primer commit del repositorio local"
[master (root-commit) 71944ea] Primer commit del repositorio local
 1 file changed, 1 insertion(+)
 create mode 100644 1_Ejemplo.txt

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git status
On branch master
nothing to commit, working tree clean

```

- c. Es de buenas prácticas siempre dejar un mensaje en los commit, pero si se me pasó ponerlo y puse el comando **git commit** solito, la consola de Git Bash me obligará a poner un mensaje y me meterá a la siguiente ventana, que es un editor de texto de terminal llamado **bim**, donde se deberá colocar un mensaje:
    - i. Si no me deja escribir el mensaje se deben presionar las siguientes teclas, donde al hacerlo aparecerá la palabra **INSERT** abajo: **ESC + I**.
    - ii. Para poder salir de esta ventana y volver a la consola normal se deben presionar las siguientes teclas que sirven para indicarle a **bim** que guarde el archivo, esto en consecuencia manda el commit con el mensaje que hayamos escrito: **ESC + SHIFT + Z + Z**.
    - iii. Ya que hayamos salido del editor de texto **bim**, se habrá mandado el commit con su mensaje.

```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Prime...
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (master)
$ got commit
bash: got: command not found

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (master)
$ git commit
[master 9850d5a] commit al que se me olvido ponerle mensaje, upsi
 1 file changed, 1 insertion(+)

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (master)
$
```

Cuando se utilice el comando **git status** después de haber realizado un cambio en el archivo de texto plano, este se verá reflejado en la consola de Git Bash. Para subir los nuevos cambios reconocidos al **Repositorio local** se deben repetir los comandos 5 y 6.

*Los cambios son reconocidos automáticamente tanto por **Visual Studio Code** como por **Git Bash**, en Visual Studio se pone una notificación en la rama de la extensión de Git y se coloca una línea verde en los elementos que se encuentren en el **Working Directory**, en la **Git Bash** los cambios serán reconocidos solamente después de ejecutar el comando **git status**.*

The screenshot shows the Visual Studio Code interface. On the left is the Explorer sidebar with a tree view of a repository named 'APRENDIENDO GITHUB'. Inside this folder are several files: '1\_Ejemplo.txt' (marked with a green 'M' icon), '1\_PrimerRepository' (marked with a grey dot icon), '1\_Ejemplo.txt' (marked with a green 'M' icon), '~-Introducción Github.docx', '1.-Introducción Github.docx', and '1.-Texto Plano - Introducción Github.txt'. A red arrow points from the first '1\_Ejemplo.txt' entry in the tree to its corresponding entry in the status bar at the bottom right of the Explorer. Another red arrow points from the status bar entry to the commit message in the terminal window below.

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1\_Prime...

```

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (master)
$ git commit -m "Primer commit del repositorio local"
[master (root-commit) 71944ea] Primer commit del repositorio local
 1 file changed, 1 insertion(+)
 create mode 100644 1_Ejemplo.txt

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (master)
$ git status
On branch master
nothing to commit, working tree clean

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (master)
$ git status
On branch master
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:   1_Ejemplo.txt

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

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (master)
$ |

```

Acceso a la línea de tiempo de Git/Comparación de cambios entre versiones:

*El comando **git log** sirve para saber el número de un **commit** en específico y así poder acceder a una versión anterior en la línea de tiempo del Repositorio local.*

7. **git log:** Comando para mostrar el historial de versiones de todas las ramas que conforman un proyecto con las características que las diferencian. Si el número de versiones es mayor a las que se me permite ver en la pantalla de la consola, aparecerán dos puntos (:), debemos ir hacia abajo o arriba con las flechas del teclado para verlas todas y presionar la siguiente tecla para poder salir de ahí: **Q**.

- a. **git log nombre\_archivo.extension**: Comando para mostrar las versiones (línea de tiempo) de todas las ramas de un archivo específico del proyecto.
  - i. Cada commit cuenta con su número, rama, el autor que lo realizó, correo, fecha, hora y mensaje incluido.
  - ii. Por medio del **número de commit** es que después se podrá regresar el proyecto a cierta versión. Al **número de commit** también se le llama hash.

```

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git log
commit 0c89ea3f6c2a39be88071b825e02d49d50adc426 (HEAD -> master)
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date: Thu Jul 21 18:29:15 2022 -0500

    Segundo commit del repositorio local

commit 71944eacf19dfa42d0b140ddbe226c967e517160
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date: Thu Jul 21 18:16:07 2022 -0500

    Primer commit del repositorio local

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ 

```

- b. **git log --stat**: Comando que muestra cada commit con detalles adicionales, como los archivos que tuvieron cambios, el número líneas de código (o filas de texto) que se **borraron** en cada archivo indicadas con un signo **-** y las que se **añadieron** en cada archivo indicadas con un signo **+**.

```

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git log --stat
commit 1a54d43f7fd8341d720b7cc046388002f0687948 (HEAD -> master)
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date: Mon Sep 5 01:45:18 2022 -0500

    Crear carpeta css y cambios en archivo txt y html

1_Ejemplo1.txt | 8 ++++++-
2_Ejemplo2.html | 3 ++
css/styles.css | 5 +++++
3 files changed, 14 insertions(+), 2 deletions(-)

commit 595cc841970dd6a4e5d2a38478573acca9dcfe4d
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date: Mon Sep 5 01:34:15 2022 -0500

    Renombrar ejemplo 1 y crear ejemplo 2 después del reset

1_Ejemplo.txt => 1_Ejemplo1.txt | 0
2_Ejemplo2.html | 11 ++++++++
2 files changed, 11 insertions(+)

commit 71944eacf19dfa42d0b140ddbe226c967e517160
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date: Thu Jul 21 18:16:07 2022 -0500

    Primer commit del repositorio local

1_Ejemplo.txt | 1 +
1 file changed, 1 insertion(+)

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ 

```

*El comando git show sirve para ayudarnos a saber por qué se ha roto un código en la última versión, ya que muestra los cambios específicos que se hicieron entre la última versión hecha y una anterior a ella.*

8. **git show:** Comando para mostrar todos los cambios realizados en los archivos del proyecto, indicando exactamente el cambio que se realizó en cada uno de ellos y quién lo hizo. *Si el número de versiones es mayor a las que se me que permite ver en la pantalla de la consola, aparecerán dos puntos (:), debemos ir hacia abajo o arriba con las flechas del teclado para verlas todas y presionar la siguiente tecla para poder salir de ese modo de visualización: Q.*
- git show nombre\_archivo.extension:** Comando para mostrar los cambios entre el último commit y la versión actual de un archivo específico del proyecto.
    - Con el comando **git show** se muestra:
      - El último commit realizado, mostrando su número, rama, autor, fecha y mensaje.**
      - Un objeto llamado **diff** que compara las dos últimas versiones de cada archivo del proyecto, una llamada **--a** (la más antigua) y la otra **+++b** (la más reciente).
      - En medio de 2 arrobas **@@** se muestra el número de bytes que cambiaron entre las dos versiones.
      - El contenido de la versión anterior donde hubo un cambio se muestra con letras rojas.**
      - El contenido de la nueva versión donde hubo un cambio se muestra con letras verdes.**
      - Las partes del archivo que no hayan cambiado se mostrarán con letras blancas.
      - La palabra “\ No new line at end of file” indica que no se agregó un Enter adicional al archivo.

The terminal window shows the following output:

```
MINGW64:/c/Users/Education-Platzi/proyecto1
commit ab1f35ead47ab653bf2b680333cffee4409d48bb
Author: Freddy Vega <team@platzi.com>
Date: Mon Apr 15 11:09:15 2019 -0700

Este es el primer commit de este archivo

Education-Platzi@LAPTOP-E80MR8RH MINGW64 ~/proyecto1 (master)
$ git show historia.txt
commit 9748f6531ec74adead2b2a4a03a55b7f6e379059 (HEAD -> master)
Author: Freddy Vega <team@platzi.com>
Date: Mon Apr 15 11:11:03 2019 -0700

    Cambios al archivo de historia para reflejar la edad correcta

diff --git a/historia.txt b/historia.txt
index e562d97..2b1ecf0 100644
--- a/historia.txt
+++ b/historia.txt
@@ -1,5 +1,6 @@
 Esta es la historia de Freddy Vega

-Freddy Vega tiene 32 años y nació en Colombia
+Freddy Vega tiene 33 años y nació en Colombia,
+viviendo en todo el mundo

Hoy hablaremos de su historia.
\ No newline at end of file

Education-Platzi@LAPTOP-E80MR8RH MINGW64 ~/proyecto1 (master)
$ |
```

The file explorer window shows a folder named "proyecto1" containing a file named "historia.txt". The text editor window shows the contents of "historia.txt":

```
Archivo Edición Formato Ver Ayuda
Esta es la historia de Freddy Vega

Freddy Vega tiene 33 años y nació en Colombia,
viviendo en todo el mundo.

Hoy hablaremos de su historia.
```

*El comando **git diff** es mejor que el comando **git show** al ayudarnos a saber por qué se ha roto un código entre dos versiones, ya que muestra los cambios que se hicieron entre dos versiones específicas cualquiera, no solo entre las últimas dos.*

9. **git diff número\_commit\_1 número\_commit\_2**: Comando que sirve para ver las diferencias entre dos versiones cualquiera del repositorio, para ello necesitamos los **números de dos commits que queramos comparar**, estos se obtienen al utilizar el comando **git log** que se utiliza para ver el historial de versiones.

- a. Para copiar y pegar el número de los commits no se puede usar comandos de teclado como CTRL + C, a fuerza se debe dar clic derecho sobre el número y seleccionar la opción de copiar, lo mismo se hace para pegar.

```

MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Prime... - git log
commit 14018a [diego@diegok 2022-09-05 17:41:17]
Author: di_ce_r0 <dcervantesr14018a@lumino.ipn.mx>
Date: Fri Sep 2 07:41:17 2022 -0500
    cambio 2
commit dade2e0092048d17e8447a3a9ba4433d081e
Author: di_ce_r0 <dcervantesr14018a@lumino.ipn.mx>
Date: Fri Sep 2 07:48:12 2022 -0500
    Segundo commit del repositorio local
commit 0c9bae3f6c2a39be88071b823e02dd9d50adc42
Author: di_ce_r0 <dcervantesr14018a@lumino.ipn.mx>
Date: Thu Jul 21 18:29:15 2022 -0500
    Segundo commit del repositorio local
commit 71944ec19df42d0b140db226c967e517160
Author: di_ce_r0 <dcervantesr14018a@lumino.ipn.mx>
Date: Thu Jul 21 18:16:07 2022 -0500
    Primer commit del repositorio local
diego@diegok MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Prime... - git log
commit 71944ec19df42d0b140db226c967e517160
Author: di_ce_r0 <dcervantesr14018a@lumino.ipn.mx>
Date: Thu Jul 21 18:29:15 2022 -0500
    Segundo commit del repositorio local
commit 0c9bae3f6c2a39be88071b823e02dd9d50adc42
Author: di_ce_r0 <dcervantesr14018a@lumino.ipn.mx>
Date: Thu Jul 21 18:29:15 2022 -0500
    Segundo commit del repositorio local
commit dade2e0092048d17e8447a3a9ba4433d081e
Author: di_ce_r0 <dcervantesr14018a@lumino.ipn.mx>
Date: Fri Sep 2 07:41:17 2022 -0500
    cambio 2
commit 14018a [diego@diegok 2022-09-05 17:41:17]
Author: di_ce_r0 <dcervantesr14018a@lumino.ipn.mx>
Date: Fri Sep 2 07:48:12 2022 -0500
    Segundo commit del repositorio local
diego@diegok MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Prime... - git diff

```

- b. Solo se debe dejar un espacio entre los dos números de commits que se quiere comparar y es mejor si el **commit\_1** corresponde a la **versión más antigua** y el **commit\_2** corresponde a la **versión más reciente**:
  - i. El contenido de la **versión del commit 1 (la más antigua)** donde hubo un cambio se muestra con **letras rojas**.
  - ii. El contenido de la **versión del commit 2 (la más reciente)** donde hubo un cambio se muestra con **letras verdes**.
  - iii. Las partes que no hayan cambiado se mostrarán con **letras blancas**.
    1. La palabra “\ No new line at end of file” indica que no se agregó un Enter adicional al archivo.

```

MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Prime... - git diff 14018a...71944ec
diff --git a/empleo.txt b/empleo.txt
index 9850d5a..7edee4951cb 100644
--- a/empleo.txt
+++ b/empleo.txt
@@ -1,6 @@ 
\ No newline at end of file
<ejemplo 1 de texto plano para crear un repositorio y subirlo a github.
<cambio MARK 1 o lweis.
<cambio MARK 2 o lweis o lweis.
<.
<cambio de la versión 2 del archivo.
\ No newline at end of file
diego@diegok MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Prime... - git diff 14018a...71944ec

```

- c. Al hacer la comparación entre dos versiones distintas, podremos notar cuando una alteración del archivo se haya realizado en medio de líneas de código que no cambiaron, cuando se hayan eliminado cosas, se hayan añadido nuevas, etc. teniendo en cuenta el código de colores mencionado anteriormente.

```

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Primer...
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (master)
$ git log
commit d7506ef0109b9822f4824b2b1ab0a2432361154b (HEAD -> master)
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date:   Fri Sep 2 09:58:15 2022 -0500

    cambios intermedios y borrar partes

commit 9850d5a160ea4a2ba66fec0729747edee4951cbb
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date:   Fri Sep 2 07:41:17 2022 -0500

    commit al que se me olvido ponerle mensaje, upsi

commit dad8e2e0092048d17e84a77a3a9ba4433d0b81ed
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date:   Fri Sep 2 07:38:12 2022 -0500

    cambio 2

commit 0c89ea3f6c2a39be88071b825e02d49d50adc426
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date:   Thu Jul 21 18:29:15 2022 -0500

    Segundo commit del repositorio local

commit 71944eacf19dfa42d0b140ddbe226c967e517160
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date:   Thu Jul 21 18:16:07 2022 -0500

    Primer commit del repositorio local

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (master)
$ git diff 9850d5a160ea4a2ba66fec0729747edee4951cbb d7506ef0109b9822f4824b2b1ab0a2432361154b
diff --git a/1_Ejemplo.txt b/1_Ejemplo.txt
index c496381..ea52ff9 100644
--- a/1_Ejemplo.txt
+++ b/1_Ejemplo.txt
@@ -1,6 +1,7 @@
-Ejemplo 1 de texto plano para crear un repositorio y subirlo a GitHub.
+Ejemplo 2 de texto plano para crear un repositorio y subirlo a GitHub.

Cambio MARK 1 oliwis.
+Cambio MARK 3 oliwis.
Cambio MARK 2 oliwis oliwis.

-Cambio de la versión 2 del archivo.
\ No newline at end of file
+Eliminé una parte anterior y puse esta juas juas
\ No newline at end of file

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (master)
$ ...

```

**10. git checkout número\_commit\_versión\_anterior nombre\_archivo.extension:** Comando que sirve para regresar un archivo del proyecto a una versión anterior, **regresando al pasado sin borrar las versiones posteriores del proyecto**. Para ello se debe indicar el número de commit y nombre del archivo (con todo y su extensión) de donde queremos visualizar su versión anterior, pudiendo identificar la versión a la que se quiere acceder al visualizar el autor que lo realizó, correo, fecha, hora o mensaje incluido. **Si no se pone un nombre de archivo en el comando, se regresa todos los archivos del proyecto a esa versión.**

- ¡¡Hay que tener cuidado al borrar o cambiar los nombres de los archivos!!** Esto será considerado por Git como un cambio y se podrán crear duplicados de los mismos archivos dentro del repositorio local cuando se quiera acceder a versiones anteriores de ciertos archivos dentro del proyecto antes o después de haberlos borrado o renombrado.

```

Archivos Editar Selección Ver ñ Opciones Terminal Ayuda
2.Ejemplo2.html M x styles.css
1.Ejemplo1.txt
2.Ejemplo2.html

MINDWBA4\Aprendiendo\OneDrive\Documents\Aprendiendo GitHub\1_Primeros pasos
diego@diegob: ~\GitHub\1_Primeros pasos /c/Users/diego/OneDrive/Documents/Aprendiendo GitHub\1_
Primeros pasos
$ git log --stat
commit 595c841970d8a6d2a38478573accabfe0
Author: d_lce_0 <dcervantes1401@lumo.ipn.mx>
Date: Mon Sep 5 01:34:15 2022 -0500
    Crear carpeta css y cambios en archivo txt y html
    L1.Ejemplo1.txt | 8 ++++++
    2.Ejemplo2.html | 3 ++
    2.Ejemplo2.html | 3 -
    3 files changed, 14 insertions(+), 2 deletions(-)
commit 595c841970d8a6d2a38478573accabfe0
Author: d_lce_0 <dcervantes1401@lumo.ipn.mx>
Date: Mon Sep 5 01:34:15 2022 -0500
    Renombrar ejemplo 1 y crear ejemplo 2 después del reset
    L1.Ejemplo1.txt => L1.Ejemplo1.txt | 11 ++++++++
    2.Ejemplo2.html | 11 ++++++++
    2 Files changed, 11 insertions(+)
commit 1794eac13df420801406db226967e517100
Author: d_lce_0 <dcervantes1401@lumo.ipn.mx>
Date: Mon Sep 5 01:34:15 2022 -0500
    Primer commit del repositorio local
    L1.Ejemplo1.txt | 1 file changed, 1 insertion(+)
diego@diegob: ~\GitHub\1_Primeros pasos /c/Users/diego/OneDrive/Documents/Aprendiendo GitHub\1_
Primeros pasos
$ git commit -m "Primer commit del repositorio local"
L1.Ejemplo1.txt
1 path added.
diego@diegob: ~\GitHub\1_Primeros pasos /c/Users/diego/OneDrive/Documents/Aprendiendo GitHub\1_
Primeros pasos
$ 

```

- git checkout nombre\_rama nombre\_archivo.extension:** Despues de haber visualizado una versión anterior, restauramos el archivo a la versión actual simplemente repitiendo el comando anterior pero en vez de poner el **número del commit** se coloca el **nombre de la rama** donde nos encontramos. **Si no se pone un nombre de archivo en el comando, se regresa todos los archivos del proyecto a la última versión de la rama.**
- Dentro de **Visual Studio Code** existe una pestaña del lado izquierdo llamada **LÍNEA DEL TIEMPO** en donde se pueden ver los nombres de los commit y las veces que se ha guardado el archivo en el **Working Directory**, al dar clic sobre ellas podemos ver el contenido de las versiones.

```

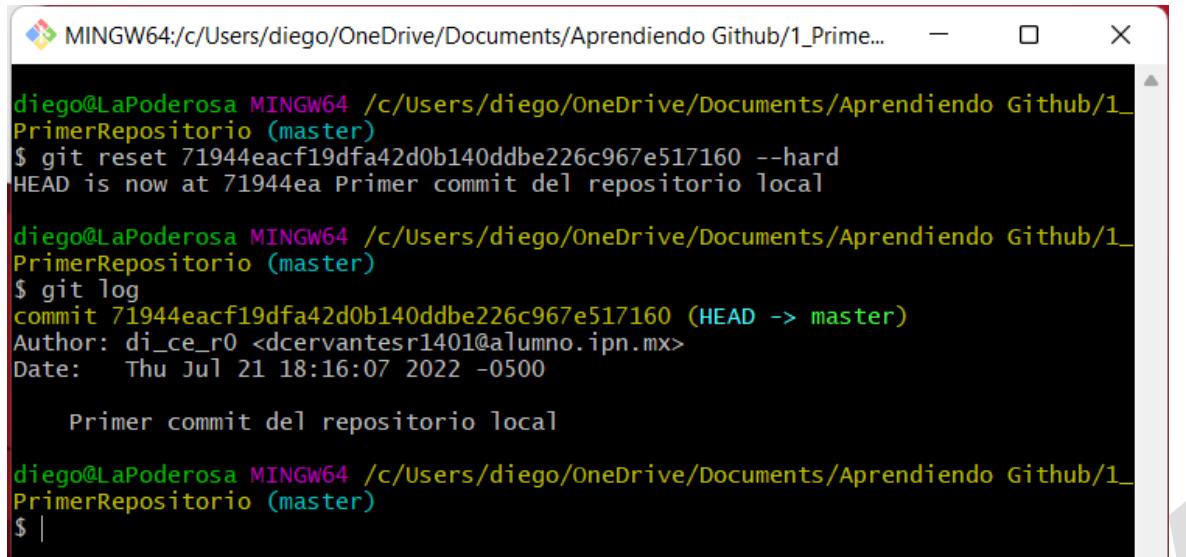
Archivos Editar Selección Ver ñ Opciones Terminal Ayuda
2.Ejemplo2.html x styles.css
1.Ejemplo1.txt
2.Ejemplo2.html

MINDWBA4\Aprendiendo\OneDrive\Documents\Aprendiendo GitHub\1_Primeros pasos
diego@diegob: ~\GitHub\1_Primeros pasos /c/Users/diego/OneDrive/Documents/Aprendiendo GitHub\1_
Primeros pasos
$ 
Create carpeta css y cambios en archivo txt y html
1.Ejemplo1.txt | 8 ++++++
2.Ejemplo2.html | 3 ++
2.Ejemplo2.html | 3 -
3 files changed, 14 insertions(+), 2 deletions(-)
commit 595c841970d8a6d2a38478573accabfe0
Author: d_lce_0 <dcervantes1401@lumo.ipn.mx>
Date: Mon Sep 5 01:34:15 2022 -0500
    Renombrar ejemplo 1 y crear ejemplo 2 después del reset
    L1.Ejemplo1.txt => L1.Ejemplo1.txt | 0
    2.Ejemplo2.html | 11 ++++++++
    2 Files changed, 11 insertions(+)
commit 1794eac13df420801406db226967e517100
Author: d_lce_0 <dcervantes1401@lumo.ipn.mx>
Date: Mon Sep 5 01:34:15 2022 -0500
    Primer commit del repositorio local
    L1.Ejemplo1.txt | 1 file changed, 1 insertion(+)
diego@diegob: ~\GitHub\1_Primeros pasos /c/Users/diego/OneDrive/Documents/Aprendiendo GitHub\1_
Primeros pasos
$ git commit -m "Primer commit del repositorio local"
L1.Ejemplo1.txt
1 path added.
diego@diegob: ~\GitHub\1_Primeros pasos /c/Users/diego/OneDrive/Documents/Aprendiendo GitHub\1_
Primeros pasos
$ git status
Changes to be committed:
  (use "git add <file>..." to stage files...)
    modified: 2.Ejemplo2.html
diego@diegob: ~\GitHub\1_Primeros pasos /c/Users/diego/OneDrive/Documents/Aprendiendo GitHub\1_
Primeros pasos
$ git add .
diego@diegob: ~\GitHub\1_Primeros pasos /c/Users/diego/OneDrive/Documents/Aprendiendo GitHub\1_
Primeros pasos
$ git status
no branch set
nothing to commit, working tree clean
diego@diegob: ~\GitHub\1_Primeros pasos /c/Users/diego/OneDrive/Documents/Aprendiendo GitHub\1_
Primeros pasos
$ 

```

11. **git reset**: Comando que sirve para regresar el proyecto a una versión anterior de una manera agresiva, **permitiéndonos volver al pasado sin la posibilidad de regresar al futuro**, ya que borra todas las versiones posteriores. Existen dos versiones de este comando, una **dura** y una **suave**. Es muy importante recordar que **¡¡¡usar este comando es MUY peligroso!!!** porque borrará todas las versiones futuras, a las que ya no podré acceder, esto es evidente debido a que no aparecerán los commits consecuentes al actual cuando ejecute el comando **git log**.

- git reset número\_commit\_versión\_anterior --hard**: Comando que sirve para regresar el proyecto a una versión anterior cualquiera del **Repositorio local**, a la cual podremos acceder al saber su número de commit, pudiendo identificar la versión a la que se quiere acceder al visualizar el autor que lo realizó, correo, fecha, hora o mensaje incluido. Esta es la versión **dura** del comando reset que borra los cambios que estén almacenados en el **Staging Area** y regresa el contenido de lo que haya en el **Working Directory** a la versión indicada **sin poder restaurar versiones futuras**.
- git reset número\_commit\_versión\_anterior --soft**: Comando que sirve para regresar el proyecto a una versión anterior cualquiera del **Repositorio local**, a la cual podemos acceder al saber su número de commit, pudiendo identificar la versión a la que se quiere acceder al visualizar el autor que lo realizó, correo, fecha, hora o mensaje incluido. Esta es la versión **suave** del comando reset que no borra lo que haya en el **Staging Area** y regresa el contenido del **Working Directory** a la versión indicada **sin poder restaurar versiones futuras, con esta opción podríamos llegar a salvar la última versión en la que estábamos antes de aplicar el reset pero no las demás**.
- git reset HEAD**: Comando que borra todos los archivos del **Staging Area**.



```

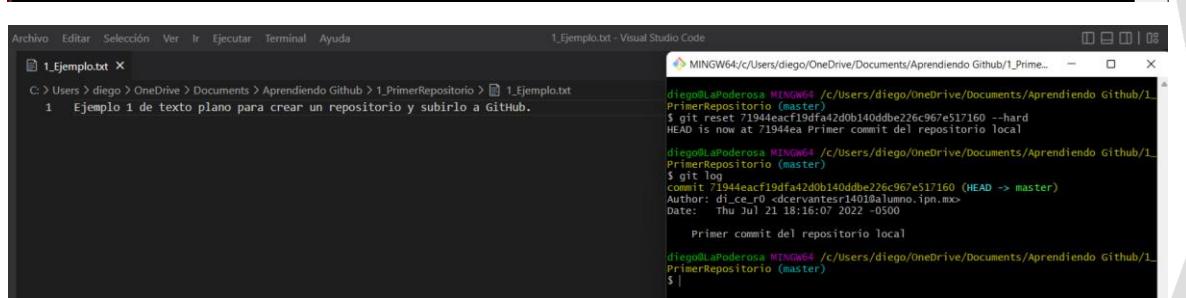
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Primer...
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git reset 71944eacf19dfa42d0b140ddbe226c967e517160 --hard
HEAD is now at 71944ea Primer commit del repositorio local

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git log
commit 71944eacf19dfa42d0b140ddbe226c967e517160 (HEAD -> master)
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date:   Thu Jul 21 18:16:07 2022 -0500

    Primer commit del repositorio local

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ |

```



```

Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda
1.Ejemplo.txt
C: > Users > diego > OneDrive > Documents > Aprendiendo Github > 1_PrimerRepositorio > 1_Ejemplo.txt
1 Ejemplo 1 de texto plano para crear un repositorio y subirlo a GitHub.

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Primer...
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git reset 71944eacf19dfa42d0b140ddbe226c967e517160 --hard
HEAD is now at 71944ea Primer commit del repositorio local

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git log
commit 71944eacf19dfa42d0b140ddbe226c967e517160 (HEAD -> master)
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date:   Thu Jul 21 18:16:07 2022 -0500

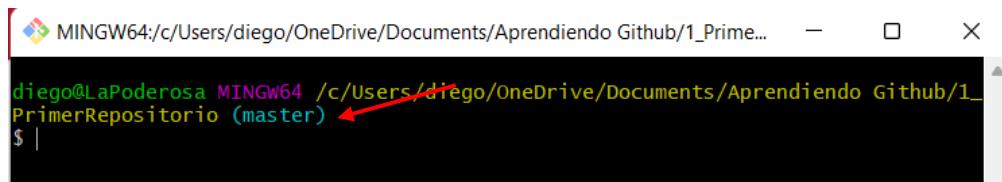
    Primer commit del repositorio local

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ |

```

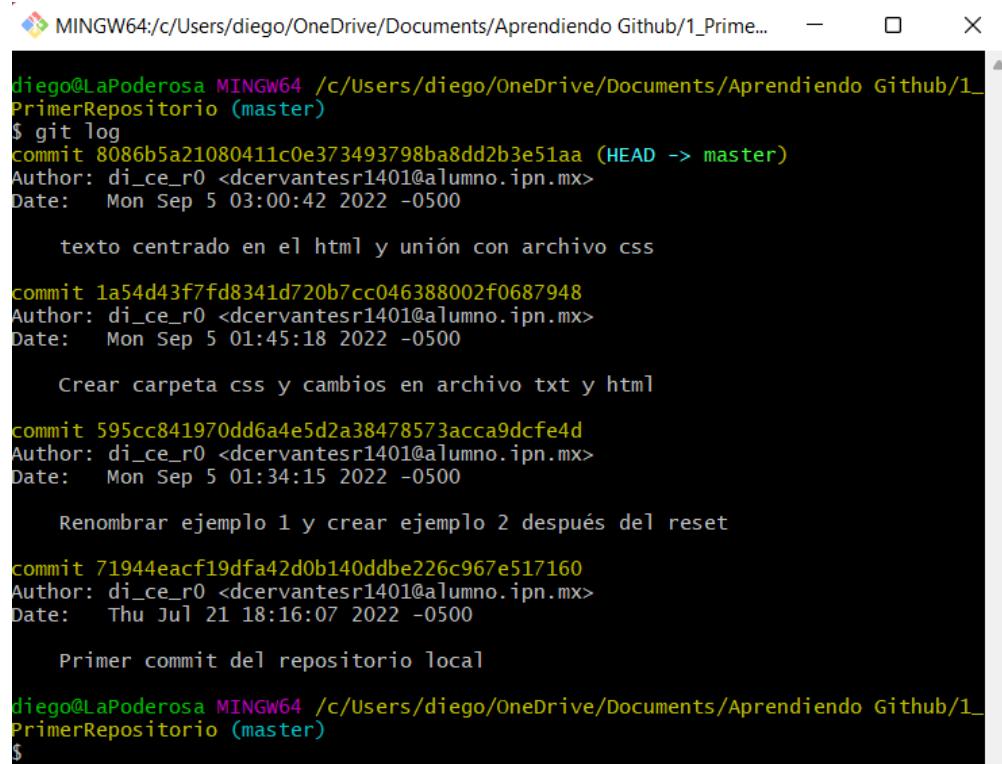
Manejo de ramas en Git:

*Las ramas sirven para crear cambios secundarios que no afecten al proyecto principal, recordemos que la rama original siempre se llama **master** en el **Repositorio local** y **main** en el **Repositorio remoto**, podemos ver la rama en la que nos encontramos siempre en la Git Bash porque sale alado del nombre del directorio entre paréntesis.*



A screenshot of a terminal window titled "MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1\_PrimerRepositorio". The command "git branch" is run, and the output shows "diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1\_PrimerRepositorio (master)". A red arrow points to the word "master" in the output.

*Es importante mencionar que el apuntador del commit más reciente de la rama en la que nos encontramos actualmente tiene un nombre especial para diferenciarse de los demás y ese nombre es **HEAD** o cabecera.*



A screenshot of a terminal window titled "MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1\_PrimerRepositorio". The command "git log" is run, showing the commit history. The output includes:

```
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git log
commit 8086b5a21080411c0e373493798ba8dd2b3e51aa (HEAD -> master)
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date:   Mon Sep 5 03:00:42 2022 -0500

    texto centrado en el html y unión con archivo css

commit 1a54d43f7fd8341d720b7cc046388002f0687948
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date:   Mon Sep 5 01:45:18 2022 -0500

    Crear carpeta css y cambios en archivo txt y html

commit 595cc841970dd6a4e5d2a38478573acca9dcfe4d
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date:   Mon Sep 5 01:34:15 2022 -0500

    Renombrar ejemplo 1 y crear ejemplo 2 después del reset

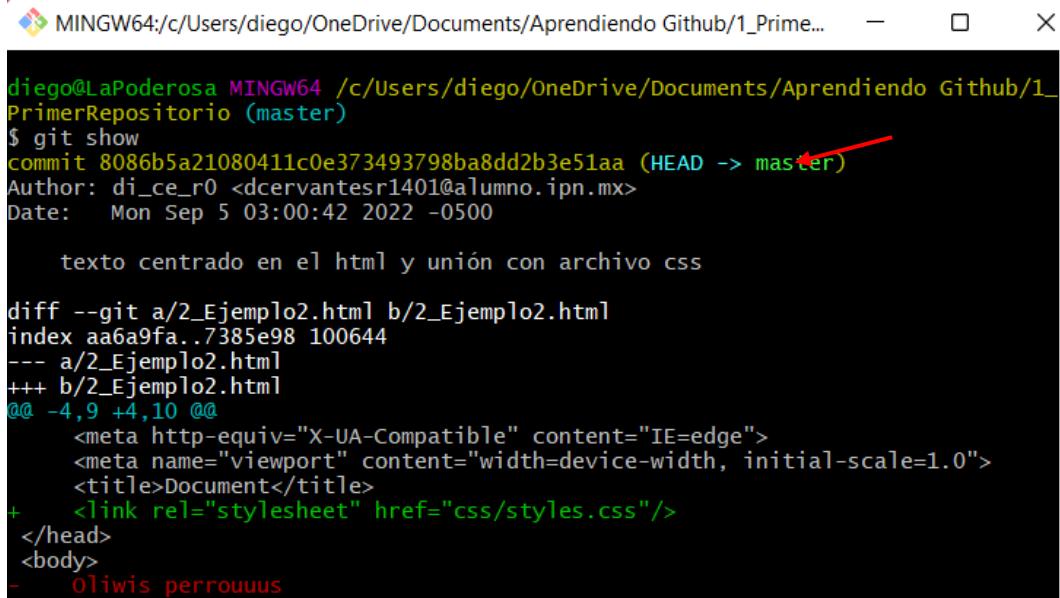
commit 71944eacf19dfa42d0b140ddbe226c967e517160
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date:   Thu Jul 21 18:16:07 2022 -0500

    Primer commit del repositorio local

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$
```

*Cuando lleguemos ver un error llamado **Detached HEAD** lo que está indicando es que no estamos situados en el commit más reciente de la branch y si enviamos un cambio desde un commit más viejo, el commit más reciente de la rama actual se perdería, por lo que debemos regresar al último commit y enviar los cambios desde ahí o aplicar un **git reset** para que se vacíen los commits de esa rama hasta el punto donde nos encontramos. Se regresa el **HEAD** al último commit con la instrucción: **git checkout nombre\_rama** como se vio en el comando 10.b.*

*Con el comando **git show** puedo ver en que rama se encuentra el **HEAD**, porque recordemos que la cabecera solo apunta al commit en el que nos encontramos actualmente, indicando de igual manera la rama en la que estamos situados.*



```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Prime...
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git show
commit 8086b5a21080411c0e373493798ba8dd2b3e51aa (HEAD -> master)
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date: Mon Sep 5 03:00:42 2022 -0500

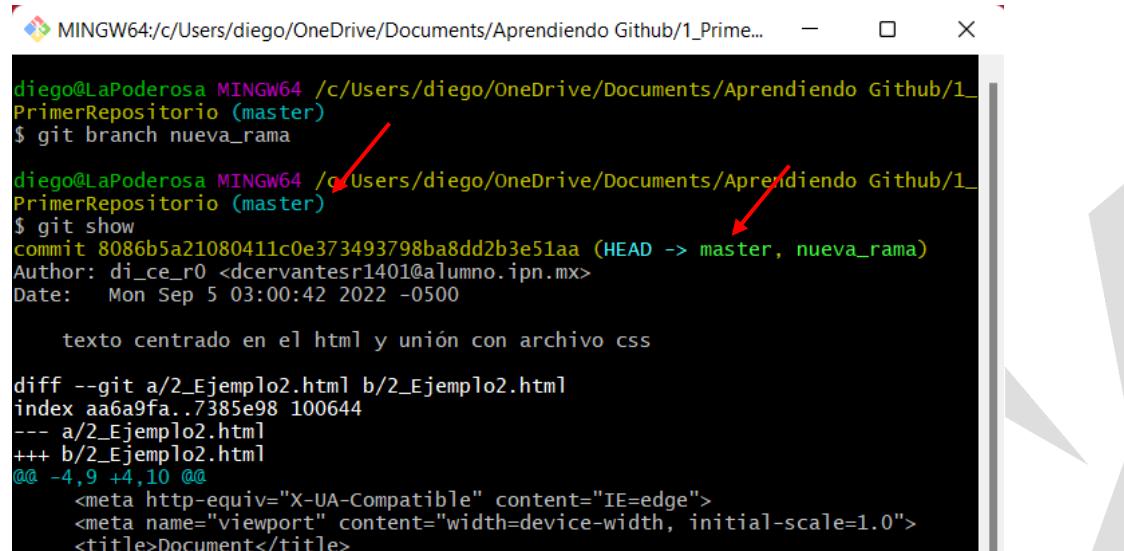
    texto centrado en el html y unión con archivo css

diff --git a/2_Ejemplo2.html b/2_Ejemplo2.html
index aa6a9fa..7385e98 100644
--- a/2_Ejemplo2.html
+++ b/2_Ejemplo2.html
@@ -4,9 +4,10 @@
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
+   <link rel="stylesheet" href="css/styles.css"/>
</head>
<body>
-   Oliwis perrouuu
```

Cuando alguien crea una **rama nueva**, literal está creando una **copia del último commit**, no de toda la línea del tiempo, sino solo del **HEAD**, pero en una línea del tiempo nueva, es por eso que en un inicio el **HEAD** apunta a dos ramas a la vez. Todos los cambios que se hagan en esta nueva branch no afectarán en nada a la rama principal (**master o main**) hasta que las fusionemos con el comando **git merge**, pero recordemos que al ejecutar el comando **git log**, aparecerán los commits de todas las ramas, pero eso no significa que cada rama incluya la línea de tiempo de todas las demás.

12. **git branch nombre\_rama\_nueva**: Comando que sirve para crear una rama nueva, moviendo así el **HEAD** hacia ella para que ahora se pueda crear una nueva línea de tiempo. Recordemos que con el comando **git show** podemos ver las ramas a las cuales está apuntando el **HEAD**. La primera rama que se indique dentro del paréntesis es en la que nos encontramos.

- a. **git branch -D nombre\_rama\_a\_borrar**: Comando para borrar una rama, cuando queramos deshacernos de una rama debemos estar situados sobre una distinta a la que queremos eliminar.



```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Prime...
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git branch nueva_rama

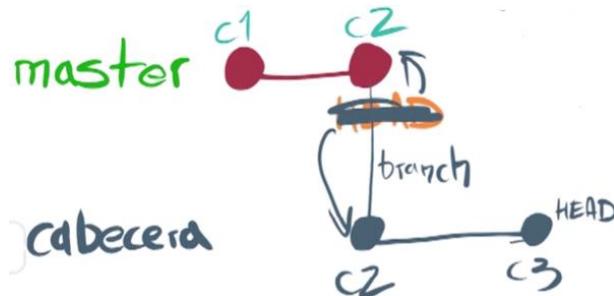
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git show
commit 8086b5a21080411c0e373493798ba8dd2b3e51aa (HEAD -> master, nueva_rama)
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date: Mon Sep 5 03:00:42 2022 -0500

    texto centrado en el html y unión con archivo css

diff --git a/2_Ejemplo2.html b/2_Ejemplo2.html
index aa6a9fa..7385e98 100644
--- a/2_Ejemplo2.html
+++ b/2_Ejemplo2.html
@@ -4,9 +4,10 @@
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
```

13. **git checkout nombre\_rama**: Comando que sirve para moverme de una rama a otra. Esto se debe hacer porque cuando se crea una nueva rama no nos movemos automáticamente a ella, solamente se crea una copia del commit actual, esa es la razón por la cual cuando se crea una rama nueva, el **HEAD** apunta a dos ramas a la vez, porque en ese nodo o commit conviven dos ramas. Una vez que nos hayamos movido de rama, para checar que en efecto nos hemos cambiado podemos usar el comando **git status** o **git branch** (que será explicado a continuación).

- a. El **HEAD** dejará de estar en dos ramas a la vez cuando realicemos nuestro primer **commit** en la rama nueva, ya que en este punto se creará una nueva línea de tiempo en la rama recién creada. (que puede tener cualquier nombre como “cabecera” o “nueva\_rama”).



```

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Prime...
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git status
On branch master
nothing to commit, working tree clean

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git checkout nueva_rama
Switched to branch 'nueva_rama'

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (nueva_rama)
$ git status
On branch nueva_rama
nothing to commit, working tree clean

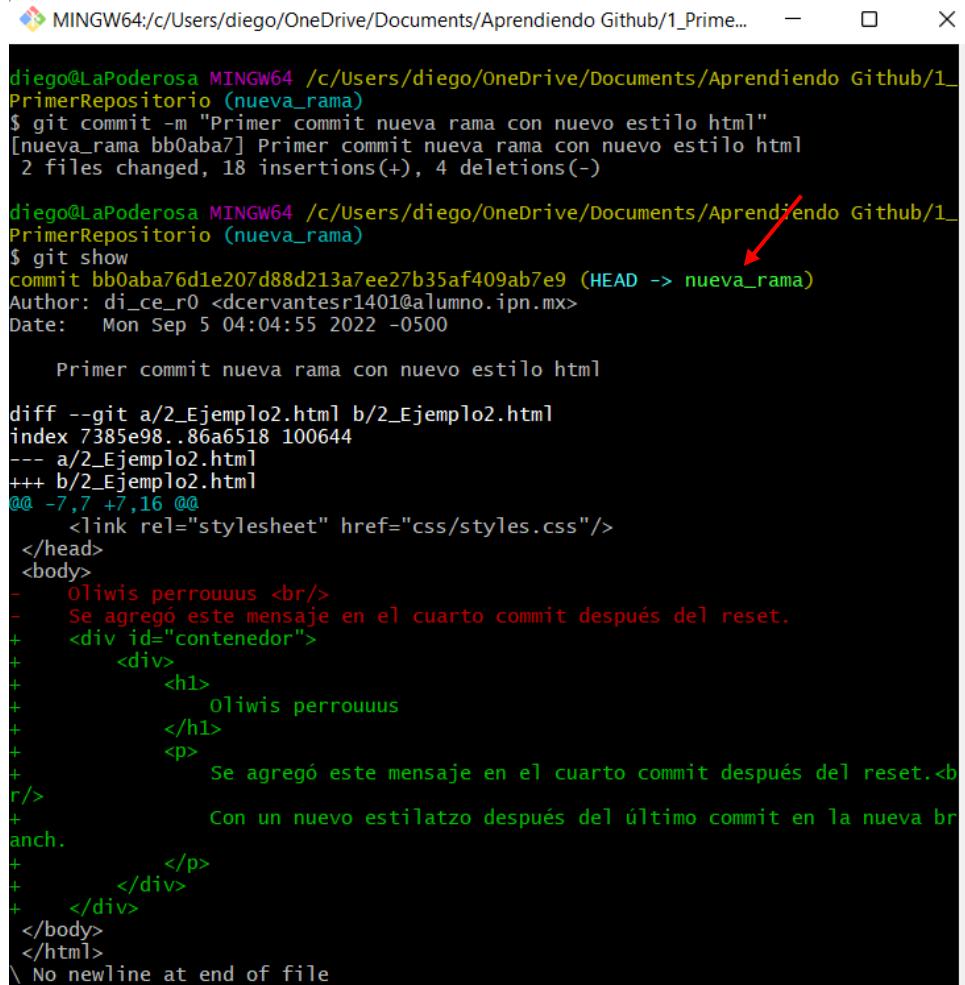
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (nueva_rama)
$ git show
commit 8086b5a21080411c0e373493798ba8dd2b3e51aa (HEAD -> nueva_rama, master)
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date:   Mon Sep 5 03:00:42 2022 -0500

    texto centrado en el html y unión con archivo css

diff --git a/2_Ejemplo2.html b/2_Ejemplo2.html
index aa6a9fa..7385e98 100644
--- a/2_Ejemplo2.html
+++ b/2_Ejemplo2.html
@@ -4,9 +4,10 @@
     <meta http-equiv="X-UA-Compatible" content="IE=edge">
     <meta name="viewport" content="width=device-width, initial-scale=1.0">
     <title>Document</title>
+    <link rel="stylesheet" href="css/styles.css"/>
</head>
```

Puedo realizar los cambios que sea en un archivo de la rama secundaria y se verán reflejados en el código dentro de **Visual Studio**, pero en cualquier momento me puedo regresar a la rama **master** o **main** donde se verá el código original sin ningún cambio.

Además, no importando en que punto de la línea del tiempo me encuentre dentro de la nueva rama, podré mover el puntero **HEAD** al último commit de las demás por medio del comando **git checkout nombre\_rama** en cualquier momento.



```

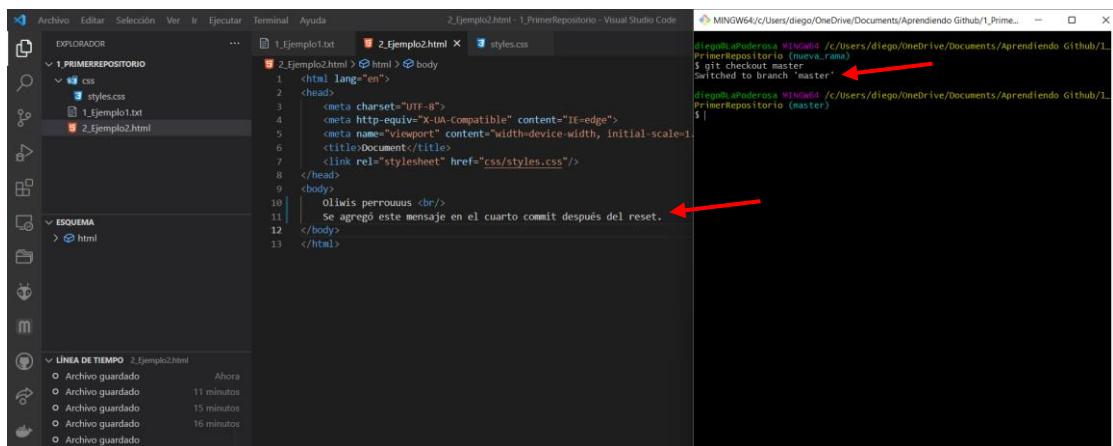
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Prime...
$ git commit -m "Primer commit nueva rama con nuevo estilo html"
[nueva_rama bb0aba7] Primer commit nueva rama con nuevo estilo html
 2 files changed, 18 insertions(+), 4 deletions(-)

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Prime...
$ git show
commit bb0aba76d1e207d88d213a7ee27b35af409ab7e9 (HEAD -> nueva_rama)
Author: di_ce_r0 <dcervantesri1401@alumno.ipn.mx>
Date:   Mon Sep 5 04:04:55 2022 -0500

    Primer commit nueva rama con nuevo estilo html

diff --git a/2_Ejemplo2.html b/2_Ejemplo2.html
index 7385e98..86a6518 100644
--- a/2_Ejemplo2.html
+++ b/2_Ejemplo2.html
@@ -7,7 +7,16 @@
     <link rel="stylesheet" href="css/styles.css"/>
</head>
<body>
-    Oliwis perrouuu <br/>
-    Se agregó este mensaje en el cuarto commit después del reset.
+    <div id="contenedor">
+        <h1>
+            Oliwis perrouuu
+        </h1>
+        <p>
+            Se agregó este mensaje en el cuarto commit después del reset.<b>
+        </p>
+        Con un nuevo estilatzo después del último commit en la nueva br
anch.
+    </p>
+    </div>
+</body>
</html>
\ No newline at end of file

```



Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda

Explorador

LÍNEA DE TIEMPO 2\_Ejemplo2.html

1\_PRIMERREPOSITORIO

- ✓ CSS
  - styles.css
- 2\_Ejemplo1.txt
- 2\_Ejemplo2.html

ESQUEMA

2\_Ejemplo2.html - 1\_PrimerRepositorio - Visual Studio Code

```

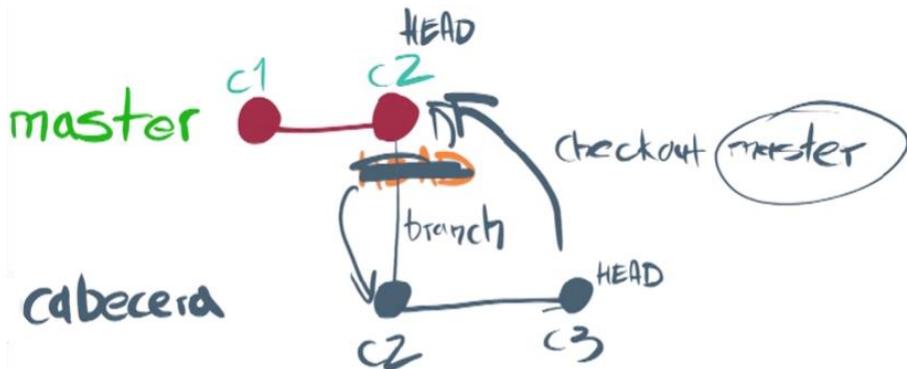
1_Ejemplo1.txt 2_Ejemplo2.html styles.css
2_Ejemplo2.html > html > body
1
2 <head>
3     <meta charset="UTF-8">
4     <meta http-equiv="X-UA-Compatible" content="IE=edge">
5     <meta name="viewport" content="width=device-width, initial-scale=1.
6     <title>Document</title>
7     <link rel="stylesheet" href="css/styles.css"/>
8 </head>
9 <body>
10    Oliwis perrouuu <br/>
11    Se agregó este mensaje en el cuarto commit después del reset.
12 </body>
13 </html>

```

MINGW64/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1\_Prime...
\$ git checkout master
Switched to branch 'master'

Al ejecutar el comando **git log** o **git log --stat**, se mostrará una serie de commits que pertenecen a la rama indicada en su último commit antes de crear una branch nueva o cambiar a otra, apareciendo también en la LÍNEA DE TIEMPO de Visual Studio Code.

*Lo que sucede cuando me muevo de una línea del tiempo a otra que se encuentra en una rama distinta por medio del comando `git checkout` es que el `HEAD` se está moviendo del último commit de una rama a otro.*

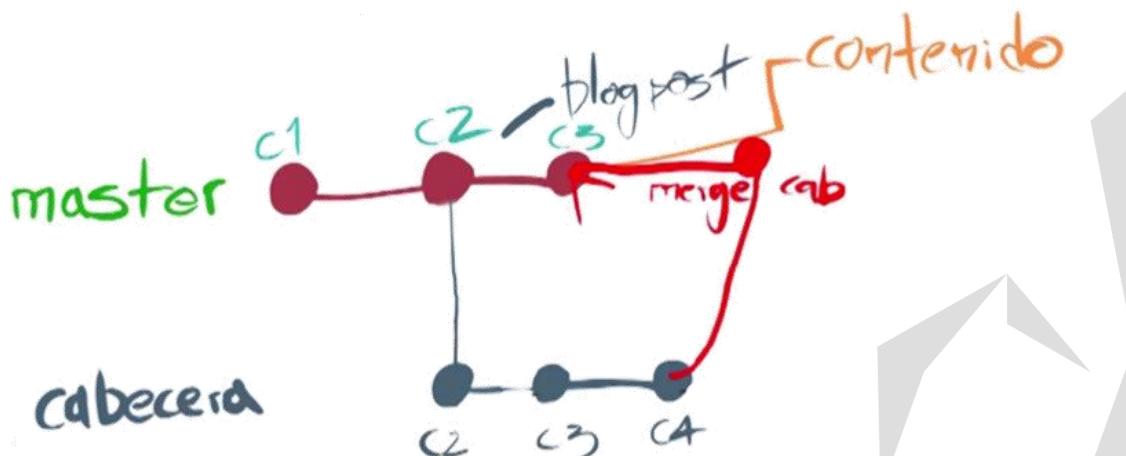


14. **git branch:** Comando que muestra un listado de todas las ramas existentes en el proyecto, resaltando con el color verde y un asterisco \* la rama en la cual nos encontramos actualmente.

```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRe...
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_
PrimerRepositorio (master)
$ git branch
* master
  nueva_rama

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_
_PrimerRepositorio (master)
$
```

*Cuando se fusionan dos archivos distintos en uno solo con el comando `git merge` no se crea tal cual una rama nueva que incluya la unión, sino que la rama adicional de desarrollo se incluye a la rama actual desde donde ejecuté el comando (que usualmente es la rama principal `master` o `main`), al hacer esto se creará un nuevo commit que incluirá el contenido de las dos ramas fusionadas y conservará la línea de tiempo de la rama desde donde se invocó el comando, aunque esto no significa que ya no podamos acceder a la rama anterior, esa línea de tiempo aún se conserva.*



15. **git merge nombre\_rama\_a\_fusionar**: Comando que **fusiona el contenido del último commit de una rama externa hacia el contenido de la rama en la que me encuentro actualmente**, logrando de esta forma unir dos módulos de un proyecto grande. El merge se considera como un commit, por lo que se añadirá al historial de versiones este movimiento, que se podrá ver con el comando **git log**.
16. **git log --all --graph**: Comando que sirve para mostrar de forma gráfica todos los commits que se le han hecho al proyecto con todo y las ramas que los conforman.

```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Prueba  
* commit ca01a0a6f88a8425053fdb16a5304afb2c784e07 (HEAD -> main, origin/main)  
| Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>  
| Date: Sun Sep 11 07:22:54 2022 -0500  
|  
|     commit hecho después de añadir la conexión cifrada SSH  
|  
* commit d64377d865619426e1a82b90fe3481386abe1139  
| Author: Diego Cervantes <dcervantesr1401@alumno.ipn.mx>  
| Date: Sun Sep 11 04:47:52 2022 -0500  
|  
|     cambio hecho desde Git Hub.  
|  
* commit 5a5945907ee9ed6bb7259a889d9a90bc17c6f02a  
| Merge: 70a411e 682551d  
| Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>  
| Date: Sun Sep 11 04:21:14 2022 -0500  
|  
|     Merge branch 'main' of https://github.com/diceR0/Primer_repositorio_prueba  
|  
* commit 682551da9c50873271448d3fa7a560d9cf82c75d  
| Author: Diego Cervantes <dcervantesr1401@alumno.ipn.mx>  
| Date: Sun Sep 11 04:15:46 2022 -0500  
|  
|     Initial commit  
|  
* commit 27b07131f5a081244ba95b643ac7474a26df5658 (master)  
| Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>  
| Date: Fri Sep 9 03:19:48 2022 -0500  
|  
|     agregado mensaje rama master papus  
|  
* commit 70a411ef4165b0aa5af73b21b56153697b905232  
| Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>  
| Date: Mon Sep 5 05:32:11 2022 -0500  
|  
|     mejoramiento de estilo despues del merge  
|  
* commit 3d2799e5b5aa75506b37065b38cb1a2a375cf64  
| Merge: 14ab156 cb9ca25  
| Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>  
| Date: Mon Sep 5 05:21:26 2022 -0500  
|  
|     merge de las dos ramas despues de resolver el conflicto  
|  
* commit cb9ca25ff51231061ef71d9b42911d6a033e7f8c (nueva_rama)  
| Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>  
| Date: Mon Sep 5 04:35:49 2022 -0500  
|  
|     Cambio nueva rama texto  
|  
* commit bb0aba76d1e207d88d213a7ee27b35af409ab7e9  
| Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>  
| Date: Mon Sep 5 04:04:55 2022 -0500  
|  
|     Primer commit nueva rama con nuevo estilo html  
|
```

- a. **git log --all --graph --decorate --oneline:** Comando muestra de forma gráfica todos los commits y ramas que conforman al proyecto de forma comprimida y concreta. **Este es el mejor comando de visualización.**

```

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git log --all --graph --decorate --oneline
* ca01a0a (HEAD -> main, origin/main) commit hecho después de añadir la conexión cifrada SSH
* d64377d cambio hecho desde Git Hub.
* 5a59459 Merge branch 'main' of https://github.com/diCeRO/Primer_repositorio_prueba
| \
| * 682551d Initial commit
| * 27b0713 (master) agregado mensaje rama master papus
|
| * 70a411e mejoramiento de estilo después del merge
| * 3d2799e merge de las dos ramas después de resolver el conflicto
|
| * cb9ca25 (nueva_rama) Cambio nueva rama texto
| * bb0aba7 Primer commit nueva rama con nuevo estilo html
| * 14ab156 ultimo cambio antes del merge en la rama master
| * f96b312 cambio de estilo papuus
| * 78e0eee Cabecera nuevecita papus
|
| * 8086b5a texto centrado en el html y unión con archivo css
| * 1a54d43 Crear carpeta css y cambios en archivo txt y html
| * 595cc84 Renombrar ejemplo 1 y crear ejemplo 2 después del reset
| * 71944ea Primer commit del repositorio local
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (main)
$ |

```

- i. **alias nombre\_variable\_consola = "comando\_largo\_a\_guardar":** Con este comando puedo guardar un comando muy largo como el anterior en una variable con un nombre que yo me acuerde para ejecutarlo de forma más sencilla

- Este nombre de variable solo se guarda de forma temporal, cuando reinicie el ordenador ya no estará.

```

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git log --all --graph --decorate --oneline
* ca01a0a (HEAD -> main, origin/main) commit hecho después de añadir la conexión cifrada SSH
* d64377d cambio hecho desde Git Hub.
* 5a59459 Merge branch 'main' of https://github.com/diCeRO/Primer_repositorio_prueba
| \
| * 682551d Initial commit
| * 27b0713 (master) agregado mensaje rama master papus
|
| * 70a411e mejoramiento de estilo después del merge
| * 3d2799e merge de las dos ramas después de resolver el conflicto
|
| * cb9ca25 (nueva_rama) Cambio nueva rama texto
| * bb0aba7 Primer commit nueva rama con nuevo estilo html
| * 14ab156 ultimo cambio antes del merge en la rama master
| * f96b312 cambio de estilo papuus
| * 78e0eee Cabecera nuevecita papus
|
| * 8086b5a texto centrado en el html y unión con archivo css
| * 1a54d43 Crear carpeta css y cambios en archivo txt y html
| * 595cc84 Renombrar ejemplo 1 y crear ejemplo 2 después del reset
| * 71944ea Primer commit del repositorio local
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (main)
$ alias arbolito='git log --all --graph --decorate --oneline'
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (main)
$ arbolito
* ca01a0a (HEAD -> main, origin/main) commit hecho después de añadir la conexión cifrada SSH
* d64377d cambio hecho desde Git Hub.
* 5a59459 Merge branch 'main' of https://github.com/diCeRO/Primer_repositorio_prueba
| \
| * 682551d Initial commit
| * 27b0713 (master) agregado mensaje rama master papus
|
| * 70a411e mejoramiento de estilo después del merge
| * 3d2799e merge de las dos ramas después de resolver el conflicto
|
| * cb9ca25 (nueva_rama) Cambio nueva rama texto
| * bb0aba7 Primer commit nueva rama con nuevo estilo html
| * 14ab156 ultimo cambio antes del merge en la rama master
| * f96b312 cambio de estilo papuus
| * 78e0eee Cabecera nuevecita papus
|
| * 8086b5a texto centrado en el html y unión con archivo css
| * 1a54d43 Crear carpeta css y cambios en archivo txt y html
| * 595cc84 Renombrar ejemplo 1 y crear ejemplo 2 después del reset
| * 71944ea Primer commit del repositorio local
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (main)
$ |

```

17. **git show-branch**: Comando que sirve para mostrar todas las ramas que conforman al proyecto en el **Repositorio local**, mostrando primero un listado de todas las branches, después de los 3 guiones medios (---) se muestra una línea de tiempo con los commits más recientes de cada una de las ramas existentes.

- a. **git show-branch --all**: Comando que realiza la misma función que el comando anterior, pero este muestra tanto las ramas del **Repositorio local** como las del **Repositorio remoto**.

```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Pr... ━ ━ X
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github
/1_PrimerRepositorio (main)
$ git show-branch
* [main] commit hecho después de añadir la conexión cifrada SSH
! [master] agregado mensaje rama master papus
! [nueva_rama] Cambio nueva rama texto
---
* [main] commit hecho después de añadir la conexión cifrada SSH
* [main^] cambio hecho desde Git Hub.
* [main~2^2] Initial commit
+ [master] agregado mensaje rama master papus
*+ [main~3] mejoramiento de estilo después del merge
-- [main~4] merge de las dos ramas después de resolver el conflicto
*++ [nueva_rama] Cambio nueva rama texto

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github
/1_PrimerRepositorio (main)
$ git show-branch --all
* [main] commit hecho después de añadir la conexión cifrada SSH
! [master] agregado mensaje rama master papus
! [nueva_rama] Cambio nueva rama texto
! [origin/main] commit hecho después de añadir la conexión cifrada SSH
---
* + [main] commit hecho después de añadir la conexión cifrada SSH
* + [main^] cambio hecho desde Git Hub.
- - [main~2] Merge branch 'main' of https://github.com/diCeRO/Primer_repositorio_prueba
* + [main~2^2] Initial commit
+ [master] agregado mensaje rama master papus
*+ + [main~3] mejoramiento de estilo después del merge
-- - [main~4] merge de las dos ramas después de resolver el conflicto
*++ [nueva_rama] Cambio nueva rama texto

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github
/1_PrimerRepositorio (main)
$ |
```

- b. **gitk**: Comando que abre un programa externo que muestra las ramas y commits existentes en el proyecto de la forma más visual posible.

1\_PrimerRepository: All files - gitk

**main** → **remotes/origin/main** commit hecho después de añadir la conexión cifrada SSH

- cambio hecho desde Git Hub
- Merge branch 'main' of https://github.com/diCeR0/Primer\_repositorio\_prueba
- Initial commit
- mejoramiento de estilo después del merge
- merge de las dos ramas después de resolver el conflicto
- nueva rama Cambio nueva rama texto
- Primer commit nueva rama con nuevo estilo html
- último cambio antes del merge en la rama master
- cambio de estilo papuus
- Cabecera nueveta papus
- texto centrado en el html y unión con archivo css
- Crear carpeta css y cambios en archivo txt y html
- Renombrar ejemplo 1 y crear ejemplo 2 después del reset
- Primer commit del repositorio local

SHA1 ID: ca01a0a6f808a8425053fdb16a5304af2b2c784e07

Find  commit containing:

Search  Patch  Tree

Diff  Old version  New version Lines of context: 3  Ignore space change Line diff

Author: di\_ce\_r0 <dcervantes1401@alumno.ipn.mx> 2022-09-11 07:22:54  
Committer: di\_ce\_r0 <dcervantes1401@alumno.ipn.mx> 2022-09-11 07:22:54  
Parent: d64377d8656194261a2b90fe34801386abe1139 (cambio hecho desde Git Hub.)  
Branches: main, remotes/origin/main

Follows:  
Precedes:

```

commit hecho después de añadir la conexión cifrada SSH
index a635b54..bb69056 100644
--- 2_Ejemplo2.html ---
index a325289..89a8f64 100644
--- css/styles.css ---

```

## Manejo de conflictos al fusionar ramas:

Cuando se ejecuta el comando **git merge** se le debe rezar a Iron Man porque esto puede ocasionar problemas; si existe un **conflicto**, la consola Git Bash me indicará que no se pueden fusionar las dos ramas.

El conflicto se crea cuando dos programadores cambiaron las mismas líneas de código y estas son distintas, esto se refiere a si el programador 1 metió una línea de código en la fila 10 del archivo html y el programador 2 metió una línea de código distinta en la misma fila 10 del mismo archivo html, ahí es cuando Git dice que hay un conflicto entre los dos códigos 😬 y marcará lo siguiente:

2\_Ejemplo2.html - 1\_PrimerRepository - Visual Studio Code

Explorador

ESQUEMA

LÍNEA DE TIEMPO

Terminal

```

2_Ejemplo2.html | 2_Ejemplo2.html | styles.css 9,1
diogolapoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRe...
$ git branch
* master
  nueva_rama
diogolapoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRe...
$ git status
nothing to commit, working tree clean
diogolapoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRe...
$ git merge nueva_rama
Automatic merge failed; fix conflicts and then commit the result.
diogolapoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRe...
$ 

```

*La sintaxis del conflicto indica su inicio con signos de menor que y la palabra HEAD y su fin con signos de mayor que y el nombre de la rama que quiero fusionar. En esta parte debemos decidir cual código se quiere conservar y borrar el otro junto con los indicadores de inicio y fin del conflicto: <<< HEAD y >>> nombre\_rama.*

```
<html>
  <head>
    <title>El titulo del post</title>
    <link rel="stylesheet" href="css/estilos.css" />
  </head>
  <body>
    <div id="container">
      <div id="cabecera">
        Hyperblog
    <<<<< HEAD
      <span id="tagline">Tu blog maestro</span>
-----
      <span id="tagline">Tu blog de cabecera</span>
>>>>> cabecera
      </div>
      <div id="post">
        <h1>Este es el título atractivo e interesante
        <p>Y este es el parrafo de inicio donde vamos
        <p>Los blogs son la mejor forma de compartir i
        <p>Suscríbete y dale like</p>
      </div>
      </div>
    </body>
  </html>
```

*Afortunadamente, Visual Studio Code nos muestra una forma de resolver estos conflictos de manera más sencilla, mostrándome las opciones de:*

- **Accept Current Change:** Esta opción hace que se descarten los cambios provenientes de la rama nueva, conservando el código tal como estaba originalmente en la rama actual.
- **Accept Incoming Change:** Esta opción hace lo opuesto a la anterior, descartando el código original y conservando solamente los cambios provenientes de la rama nueva.
- **Accept Both Changes:** Encuentra la forma de hacer que el código original y el código proveniente de la otra rama convivan sin que tengan errores, aunque usualmente debemos meterle mano para que quede bien el código.
- **Compare Changes:** Compara los cambios hechos entre los archivos provenientes de las dos ramas.

- **Start Live Session:** Esta opción no importa, ya que permite realizar una videollamada con la cuenta de la otra persona que comparte la carpeta del proyecto desde GitHub, pero no es muy usada.

```

Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes | Start Live Share Session
<<<<< HEAD (Current Change)
Oliwies perrouus <br/>
Se agregó este mensaje en el cuarto commit después del reset.
<div id="cabecera_master">
    Cabecera creada en el master papuuus :P
</div>
===== 
<div id="contenedor">
    <div>
        <h1>
            Oliwies perrouus
        </h1>
        <p>
            Se agregó este mensaje en el cuarto commit después del reset.<br/>
            Con un nuevo estilatzo después del último commit en la nueva branch.<br/>
            Nuevo cambio hecho en la rama nueva... nuevo uy si.
        </p>
    </div>
</div>
>>>>> nueva_rama (Incoming Change)
</div>
</body>
</html>

```

Después de haber resuelto el conflicto con alguna de las opciones dadas por **Visual Studio**, ya se nos permitirá realizar un **git commit -m “mensaje”** como se muestra en el [comando 6](#) para que se puedan unir ambas ramas y con ello se habrán fusionado los dos archivos.

```

Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda
EXPLORADOR 1_PRIMERREPOSITORIO ...
css styles.css 2_Ejemplo2.html styles.css
1_Ejemplo1.txt
1_Ejemplo2.html
ESQUEMA body #contenedor #cabecera.master
LÍNEA DE TIEMPO styles.css
merge de las dos ramas después ... Ahora
Archivo guardado 1 min
último cambio antes del merge ... 32 minutos
Deshacer / Rehacer 42 minutos
Archivo guardado
Archivo guardado
cambio de estilo papuuus di... 44 minutos
Archivo guardado 45 minutos
Cabecera nuevete papus di... 47 minutos
Archivo guardado 48 minutos
Archivo guardado
Archivo guardado
Archivo guardado
Archivo guardado
master 0 △ 0 Live Share

```

```

MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (Master)
$ git branch
* master
nueva_rama

MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (Master)
$ git status
On branch master
nothing to commit, working tree clean

MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (Master)
$ git merge nueva_rama
Automatic merge conflict in css/styles.css
CONFLICT (content): Merge conflict in css/styles.css
Auto-merging 2_Ejemplo2.html
CONFLICT (content): Merge conflict in 2_Ejemplo2.html
Automatic merge failed; fix conflicts and then commit the result.

MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (Master)
$ git status
On branch master
nothing to commit, working tree clean

MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (Master)
$ git merge --abort
Automatic merge failed; fix conflicts and then commit the result.

MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (Master)
$ git status
On branch master
nothing to commit, working tree clean

MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (Master)
$ git commit -am "merge de las dos ramas después de resolver el conflicto"
[master 3d2799e] merge de las dos ramas después de resolver el conflicto
MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (Master)
$ git status
On branch master
nothing to commit, working tree clean

MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (Master)
$ git log

```

Esto lo podemos confirmar con el comando **git show** que mostrará en verde los cambios que se hayan realizado en el nuevo commit, representando la unión de los dos códigos provenientes de ambas ramas y además se podrá ver reflejado en el código que los dos archivos fueron unidos.

Oliwis perrouus

Se agregó este mensaje en el cuarto commit después del reset.

Cabecera creada en el master papouus :P

```

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio [master]
diego@apoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio [master]
commit 3d279e9eb5aa5506b37065b38cb1ea2a375fc64 (HEAD -> master)
Merge: d14ab156 cb9ca25
Author: diego_cero <dcervantes1401@alumno.ipn.mx>
Date: Mon Sep 5 05:21:26 2022 -0500

    merge de las dos ramas después de resolver el conflicto

diff --cc 2_Ejemplo2.html
index 4536fb3..7800fde..0e5bc75
--- a/2_Ejemplo2.html
+++ b/2_Ejemplo2.html
@@@ -7,10 -7,17 +7,21 @@
<link rel="stylesheet" href="css/styles.css"/>
</head>
<body>
    Oliwis perrouus <br/>
    Se agregó este mensaje en el cuarto commit después del reset.
    Cabecera creada en el master papouus :P
<div id="contenedor">
    <h1> Oliwis perrouus
    </h1>
    <p> Se agregó este mensaje en el cuarto commit después del reset.<br/>
        Con un nuevo estilatzo después del último commit en la nueva branc...
    </p>
</div>
</body>
</html>
diff --cc css/styles.css
index e'627db,8824a10..560c9a6
--- a/css/styles.css
+++ b/css/styles.css
@@@ -1,12 -1,10 +1,19 @@
body{
    font-family: Arial, Helvetica, sans-serif;
    font-size: 100px;
    color: chartreuse;
    text-align: center;
}

#contenedor{
    margin-left: 10%;
    width: 80%;
    border: 2px solid black;
    text-align: center;
}

```

*Si lo queremos, podemos viajar a la rama anterior de igual manera por medio de la instrucción `git checkout nombre_rama` como se vio en el comando 13.*

Oliwis perrouus

Se agregó este mensaje en el cuarto commit después de

Con un nuevo estilatzo después del ultimo commit en la nu...

Nuevo cambio hecho en la rama nueva... nuevo uy si.

```

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio [master]
diego@apoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio [master]
git checkout nueva_rama
Switched to branch 'nueva_rama'

diego@apoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio [nueva_rama]
$ 

```

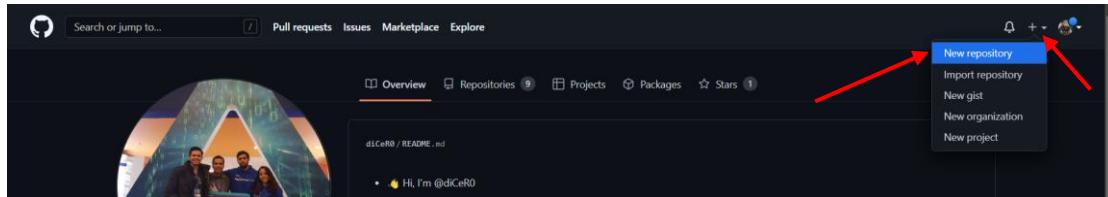
## Creación de un repositorio remoto en GitHub:

*La plataforma donde se puede tener una infinidad de repositorios remotos es GitHub, esta es una página web a la que nos registramos de forma gratuita y donde interactuamos con una infinidad de proyectos hechos por gente de todo el mundo, es una de las herramientas colaborativas más importantes en el mundo del desarrollo donde podemos publicar nuestros proyectos para que puedan ser vistos por todo*

*mundo o que estén restringidos para solo nuestro uso y visualización, además sirve como currículum para puestos de programación.*

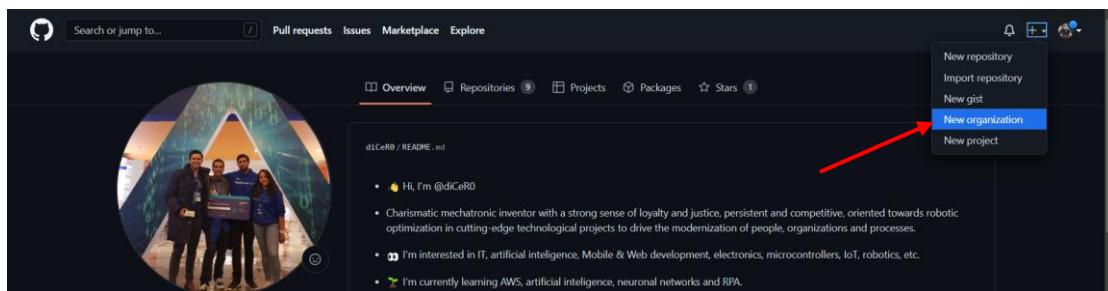
*En GitHub primero que nada debemos hacer nuestra cuenta para poder crear un **Repositorio remoto** al que se une uno de nuestros **Repositorio locales**, los pasos para crearlo son descritos a continuación:*

- a) **Repositorio nuevo:** Para conectar un **Repositorio local** de mi computadora con un **Repositorio remoto**, se debe crear desde cero un nuevo repositorio dentro de la página de GitHub, **no se debe seleccionar la opción de Import repository** porque esto lo que hará es crear una copia de un repositorio que ya está disponible de forma online (subido a la nube por GitHub u otro medio), *para crear el repositorio nuevo se selecciona la opción de: + → New repository.*

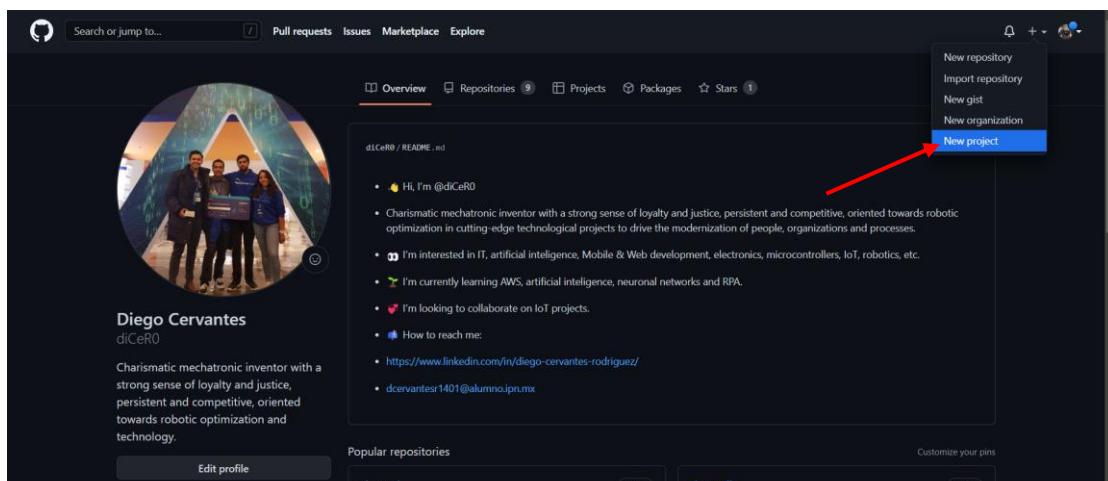


*Además de los repositorios, existen más cosas que se pueden crear dentro de GitHub como las descritas a continuación:*

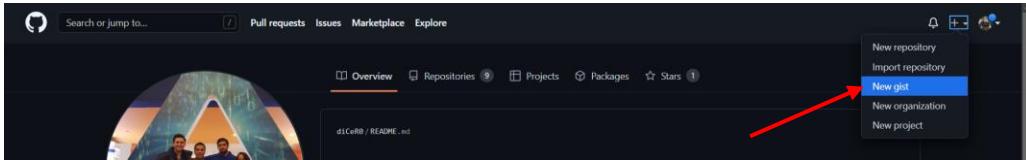
- **Organización:** Es como una empresa dentro de GitHub compuesta de varios proyectos, *esta se crea desde la opción de: + → New organization.*



- **Proyecto:** Es un conjunto de repositorios que conforman un programa o simplemente tienen algo que ver entre ellos, *se crea con la opción de: + → New project.*



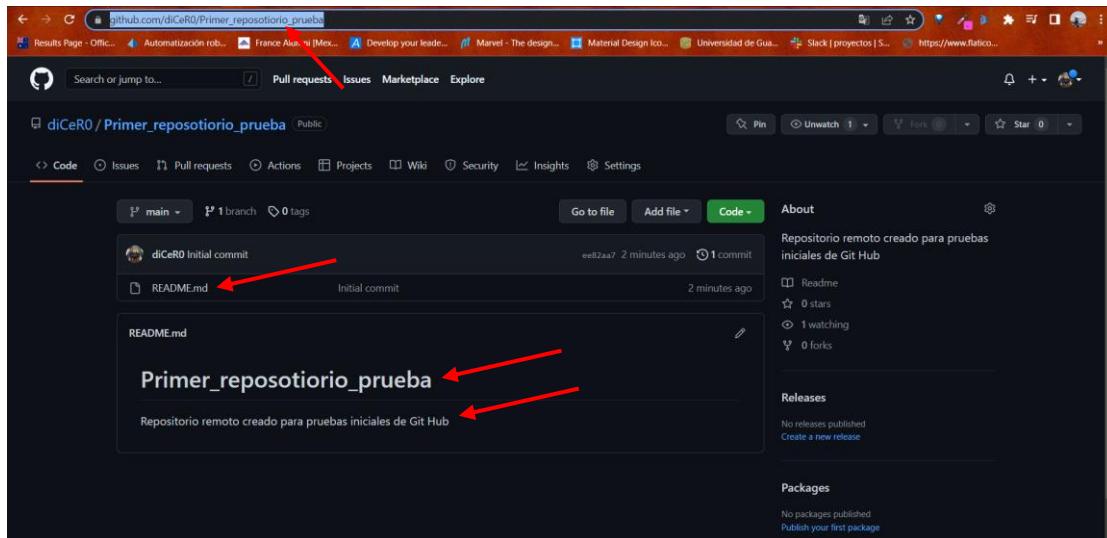
- **Gist:** Es un pedacito de código que se puede compartir, este se crea en la opción de: + → New gist.



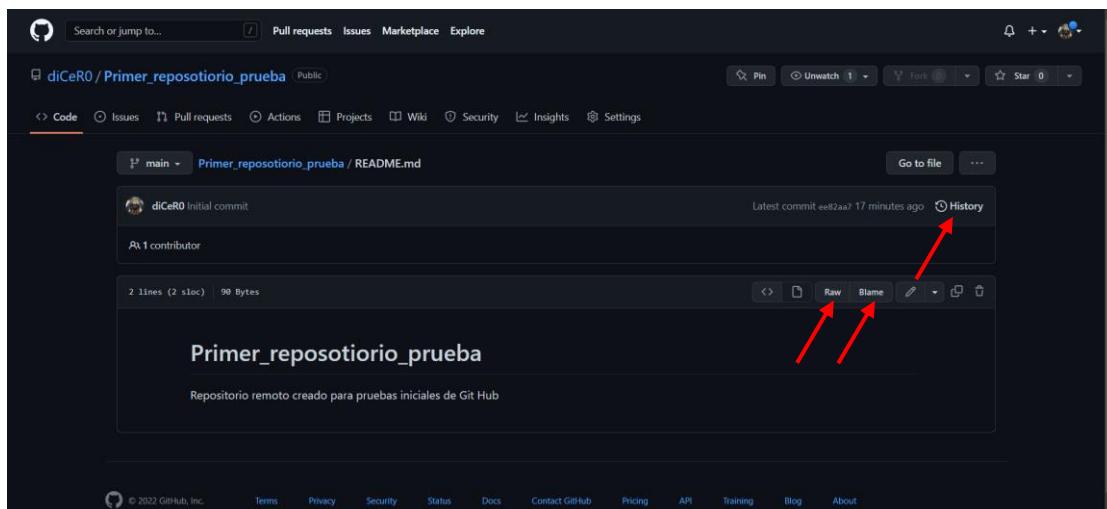
- b) Una vez que se haya seleccionado la opción de **New repository**, se deben indicar las siguientes características del repositorio, ya que las hayamos definido debemos dar clic en el botón de **Create repository** para finalizar:
- **Dueño y nombre del repositorio.**
  - **Descripción.**
  - **Modificadores de acceso:** Donde se declara si el repositorio será público o privado. Esto solo implica que pueda ser visto, no editado.
  - **Inicializar el repositorio con un archivo README:** El archivo README.md (mark down) se verá cuando alguna persona entre al repositorio por medio de Git Hub, es un archivo construido casi siempre hecho con HTML que funciona como un **manual de instrucciones del proyecto**, puede incluir links, artículos, etc. Se agrega a la carpeta del repositorio.
    - Inicializar el repositorio con un archivo Git Ignore:** No todos los archivos de un proyecto se deben agregar a un repositorio, como aquellos que contengan contraseñas, archivos PHP con accesos a bases de datos, etc. Para ello sirve git ignore, indicando cuales archivos serán ignorados al subir cambios al repositorio.
    - Agregar un archivo de licencia al repositorio:** Existen varias licencias incluidas en Git Hub a través de las cuales se puede publicar el código del repositorio, ya sea de código abierto, semiabierto, cerrado, etc.

The screenshot shows the 'Create a new repository' page. It has fields for 'Owner' (set to 'diCeRO'), 'Repository name' ('Primer\_repositorio\_prueba'), 'Description' ('Repositorio remoto creado para pruebas iniciales de Git Hub'), and 'Visibility' (set to 'Public'). Under 'Initialize this repository with:', the 'Add a README file' checkbox is checked. There are sections for '.gitignore' template and 'Choose a license' (set to 'None'). At the bottom, a note says 'This will set `main` as the default branch. Change the default name in your settings.' and a note about creating a public repository in a personal account. The 'Create repository' button is highlighted with a red arrow.

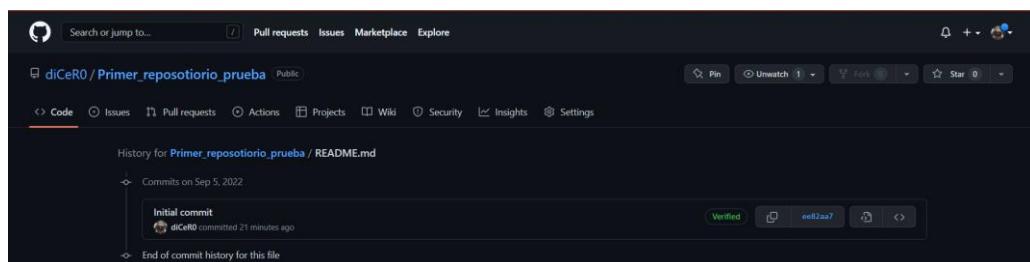
- c) Una vez que se haya creado el repositorio aparecerá una URL que tiene su nombre, además de indicar en la parte de abajo los archivos que lo conforman (que por el momento es solamente el de README.md), su nombre y descripción.



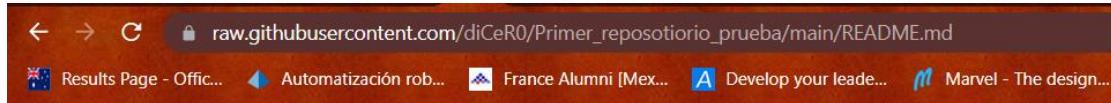
- d) Cuando se dé clic en alguno de los archivos que conforman al repositorio (que inicialmente solo será el archivo REAMDE.md) se mostrarán 3 opciones para visualizar el contenido del código:



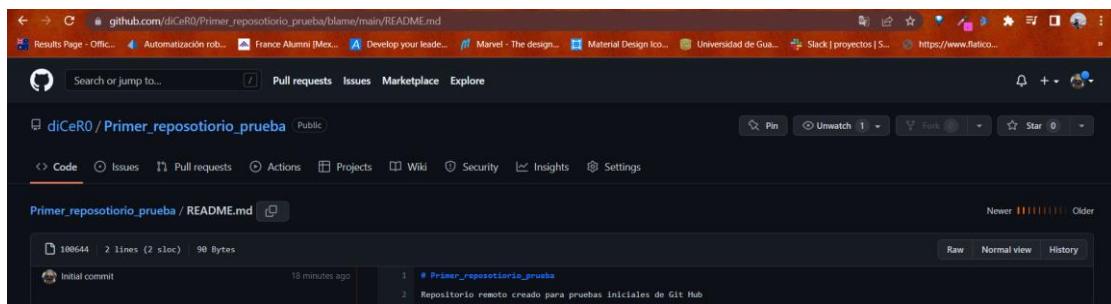
- **History:** Muestra la línea del tiempo de los commits que se le han hecho al proyecto, con todo y el autor que realizó cada acción.



- **Raw:** Muestra el código plano en el explorador, como si lo estuviera viendo desde un bloc de notas.

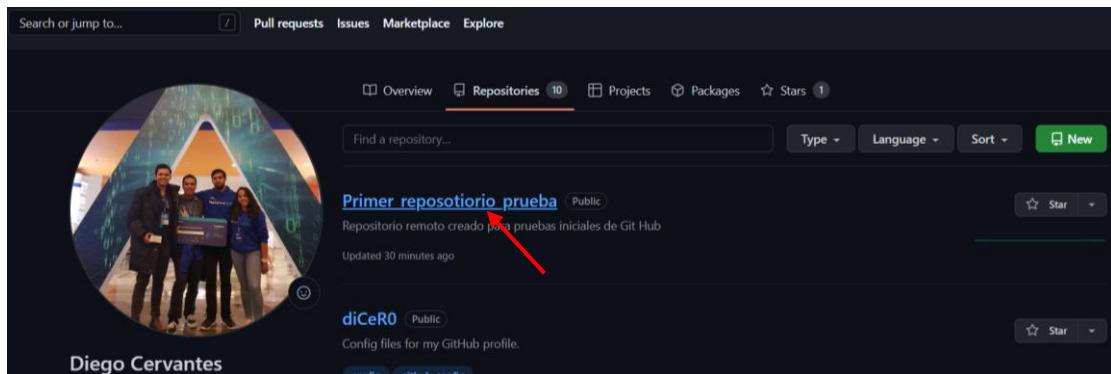
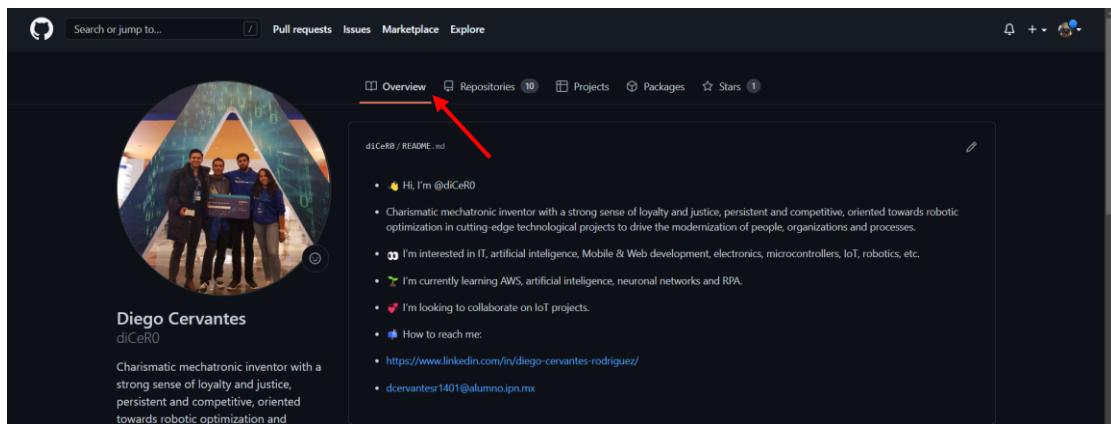


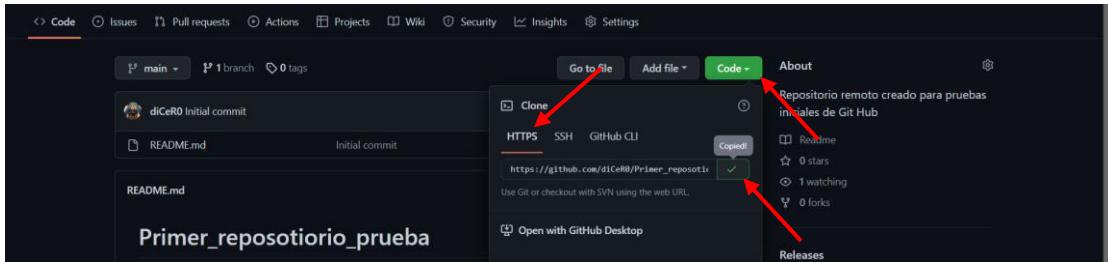
- **Blame:** Muestra cada parte del código con los autores que la realizaron.



Conectar un repositorio local con uno remoto dentro de GitHub:

*Una vez que se haya creado un nuevo repositorio en GitHub, para conectarlo debemos introducirnos a: Repositories → Seleccionar el nombre del **Repositorio remoto** al que queremos asociar nuestro **Repositorio local** → Dar clic en el botón de Code → HTTPS → Copiar la URL.*





Ya copiada la URL de Git Hub nos introduciremos a la carpeta del **Repositorio local** por medio de la consola Git Bash utilizando los **comandos de consola CM-DOS**, donde existirá la carpeta .git que almacena toda la base de datos de las versiones y ramas que conforman al proyecto. Adicionalmente nos debemos asegurar que nos encontremos en la rama **master** del **Repositorio local** y que no haya ningún cambio pendiente a mandar en la **Staging Area** antes de pretender conectarlo con la rama **main** del **Repositorio remoto** en GitHub.

```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ pwd
/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ ls -al
total 17
drwxr-xr-x 1 diego 197609 0 Sep  5 19:32 .
drwxr-xr-x 1 diego 197609 0 Sep  5 23:02 ..
drwxr-xr-x 1 diego 197609 0 Sep  5 19:32 .git/
-rw-r--r-- 1 diego 197609 133 Sep  5 03:06 1_Ejemplo1.txt
-rw-r--r-- 1 diego 197609 917 Sep  5 19:32 2_Ejemplo2.html
drwxr-xr-x 1 diego 197609 0 Sep  5 19:32 css/

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git status
On branch master
nothing to commit, working tree clean

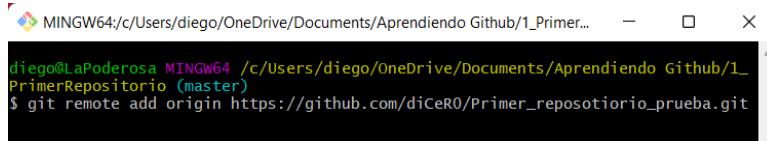
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git branch
* master
  nueva_rama

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ |
```

18. **git remote add nombre\_repositorio url\_https\_extraida\_de\_github:** Comando que sirve para conectar un **Repositorio local** con todas sus versiones y ramas a un nuevo **Repositorio remoto vacío** creado previamente en GitHub. Se debe haber creado el repositorio remoto antes de hacer esto, ya que de ahí es de donde sacamos la **URL** que utiliza este comando por medio del botón: *Code → HTTPS*.

El **nombre del repositorio** utilizado por buenas prácticas siempre es **origin**, aunque se le puede poner el nombre que sea realmente y además puedo conectar un mismo **Repositorio local** a varios **repositorios** a la vez, además de **origin**.

- a. **git remote remove origin**: Comando utilizado para eliminar la conexión actual entre un **Repositorio local** y un **Repositorio remoto**, sirve para cuando el **repositorio de GitHub** tuvo algún problema y deseamos borrarlo, pero queremos conservar el contenido del **Repositorio local** para conectarlo a un nuevo **Repositorio remoto** posteriormente.



```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Primer...
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git remote add origin https://github.com/diceR0/Primer_repositorio_prueba.git
```

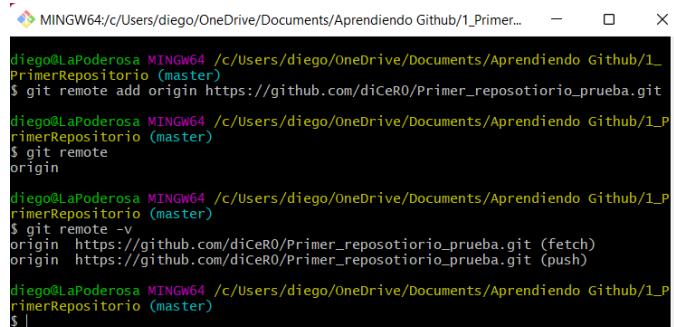
19. **git remote**: Comando que sirve para verificar el estado de la conexión realizada entre un **Repositorio local** y un **Repositorio remoto**.

- a. **git remote -v**: Muestra detalles adicionales de la conexión realizada entre un **Repositorio local** y un **Repositorio remoto**.
  - i. **Nada**: Cuando al ejecutar el comando **git remote** no se muestra nada, lo que significa es que el **Repositorio local** no está conectado a ningún **Repositorio remoto**.
  - ii. **nombre\_repositorio**: Se muestra esta respuesta en la consola de Git Bash al ejecutar el comando **git remote** cuando la conexión entre ambos repositorios fue realizada correctamente, el nombre del repositorio usualmente es **origin** y es el que es declarado desde el comando 18: **git remote add**.
  - iii. **nombre\_repositorio url\_https\_extraida\_de\_github (fetch)**: Indica que se pueden **recibir** las nuevas versiones en el repositorio local de mi computadora.
    - 1. **fetch**: Recibir contenido de la nube a mi ordenador: **Repositorio remoto** → **Repositorio local**.
  - iv. **nombre\_repositorio url\_https\_extraida\_de\_github (push)**: Indica que se pueden **mandar** las nuevas versiones al repositorio remoto de GitHub.
    - 1. **push**: Mandar contenido de mi ordenador a la nube: **Repositorio local** → **Repositorio remoto**.



```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Primer...
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git remote add origin https://github.com/diceR0/Primer_repositorio_prueba.git

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git remote
origin
```



```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Primer...
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git remote -v
origin https://github.com/diceR0/Primer_repositorio_prueba.git (fetch)
origin https://github.com/diceR0/Primer_repositorio_prueba.git (push)

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ |
```

*Es importante volver a mencionar que, aunque la rama principal de **Git Bash** se llama **master**, la rama principal de **GitHub** se llama **main**, por lo que, si hago el push con la rama **master**, se crearán dos ramas principales distintas, causando un problema, por lo cual se deben seguir los siguientes pasos antes de seguir adelante con la conexión entre ambos repositorios:*

- a) *Primero que nada, se debe crear una nueva rama llamada **main** con el comando **git branch main**.*
- b) *Nos debemos cambiar a esa rama **main** con el comando **git checkout main**.*
- c) *Debemos hacer una copia de la rama **master** con el comando **git merge master**, ya una vez situados en la nueva rama **main**.*
- d) *Y ahora ya podremos ejecutar los comandos **git pull** y **git push** para conectar ambos repositorios.*

```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Pri...
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/
1_PrimerRepositorio (master)
$ git branch
* master
  nueva_rama

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/
1_PrimerRepositorio (master)
$ git branch main

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/
1_PrimerRepositorio (master)
$ git checkout main
Switched to branch 'main'

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/
1_PrimerRepositorio (main)
$ git branch
* main
  master
  nueva_rama

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/
1_PrimerRepositorio (main)
$ git merge master
Already up to date.

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/
1_PrimerRepositorio (main)
$ git status
On branch main
nothing to commit, working tree clean

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/
1_PrimerRepositorio (main)
$ git show
commit 70a411ef4165b0aa5af73b21b56153697b905232 (HEAD -> main, master)
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date:   Mon Sep 5 05:32:11 2022 -0500

    mejoramiento de estilo despues del merge

diff --git a/2_Ejemplo2.html b/2_Ejemplo2.html
index 0e5bc75..87f413f 100644
--- a/2_Ejemplo2.html
+++ b/2_Ejemplo2.html
@@ -11,6 +11,7 @@
    Se agregó este mensaje en el cuarto commit después del reset.
    <div id="cabecera_master">
        Cabecera creada en el master papuuus :P
+
    </div>
    <div id="contenedor">
        <div>
            <h1>
```

20. **git pull origin nombre\_rama\_a\_mandar**: Comando que **jala** todas las versiones de una rama guardada en el **Repositorio remoto** que está almacenado en la plataforma de Github hacia el **Repositorio local** que había sido previamente asociado con el comando **git remote add origin url\_https\_extraida\_de\_github**. **Después de que se haya asociado ambos repositorios, este paso se debe ejecutar siempre antes de realizar un git push origin nombre\_rama\_a\_mandar.**

- a. Cada **rama** del proyecto debe ser manejada de forma individual en GitHub por medio de los comandos **git pull** y **git push**.

```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master) - X
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git pull origin main
From https://github.com/diCeR0/Primer_repositorio_prueba
 * branch            main      -> FETCH_HEAD
fatal: refusing to merge unrelated histories

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ |
```

- b. **git pull origin nombre\_rama\_a\_mandar --allow-unrelated-histories**: Comando que se ejecuta **solo la primera vez** que se va a conectar un **Repositorio remoto** con un **Repositorio local**, permitiendo fusionar el contenido de ambos repositorios.

Al ejecutar el comando aparecerá una ventana emergente del editor de texto de consola llamado **bim**, donde simplemente daré **ENTER** y con eso se habrán fusionado (hecho **merge**) ambas ramas: La rama **main** del **Git Bash** y la rama **main** de **GitHub**.

- Para poder salir de esta ventana y volver a la consola normal se debe presionar las siguientes teclas que sirven para indicarle a bim que guarde el archivo, esto en caso de que al dar ENTER no podamos salir: **ESC + SHIFT + Z + Z**.

```

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (master)
$ git pull origin main
From https://github.com/diCeRO/Primer_repositorio_prueba
 * branch           main      -> FETCH_HEAD
fatal: refusing to merge unrelated histories

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (master)
$ git pull origin main --allow-unrelated-histories
From https://github.com/diCeRO/Primer_repositorio_prueba
 * branch           main      -> FETCH_HEAD
Merge made by the 'recursive' strategy.
 README.md | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 README.md

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (master)
$ |

```

21. **git push origin nombre\_rama\_a\_mandar:** Comando que **envía** todas las versiones de una rama perteneciente al **Repositorio local** hacia el **Repositorio remoto** que había sido previamente asociado a GitHub con el comando **git remote add origin url\_https\_extraida\_de\_github**.  
**Siempre antes de haber ejecutado esta instrucción se debe haber ejecutado el comando git pull origin nombre\_rama\_a\_mandar.**

- Cada **rama** del proyecto debe ser manejada de forma individual en GitHub por medio de los comandos **git pull** y **git push**.

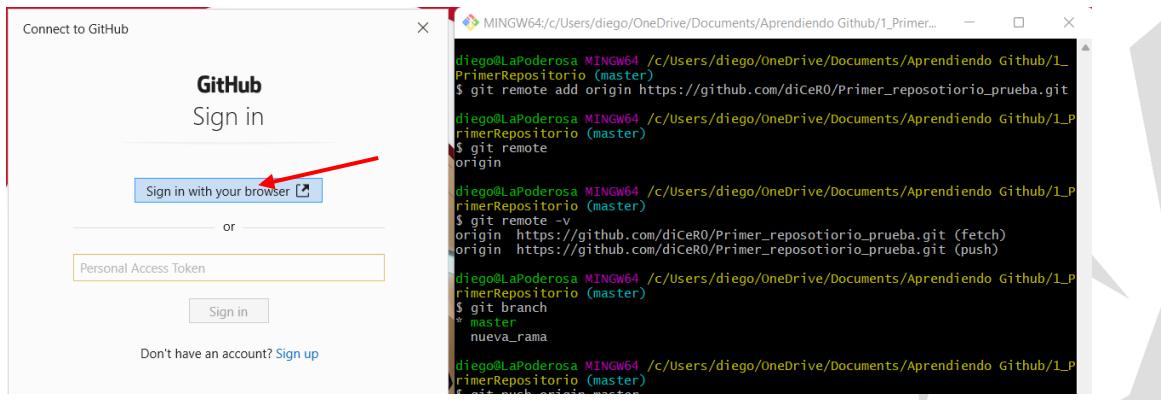
```

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ git push origin main
Enumerating objects: 51, done.
Counting objects: 100% (51/51), done.
Delta compression using up to 12 threads
Compressing objects: 100% (41/41), done.
Writing objects: 100% (50/50), 4.80 KiB | 702.00 KiB/s, done.
Total 50 (delta 18), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (18/18), done.
To https://github.com/diCeRO/Primer_repositorio_prueba.git
  682551d..5a59459 main -> main

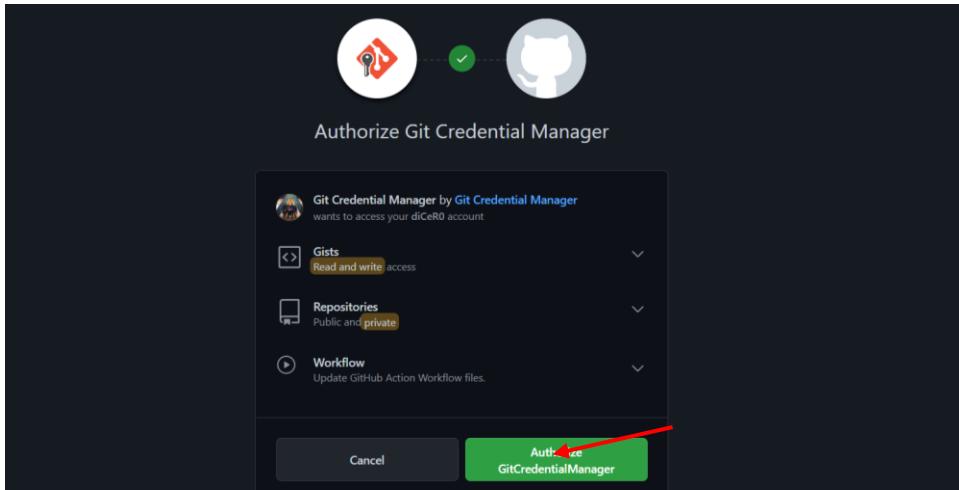
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ |

```

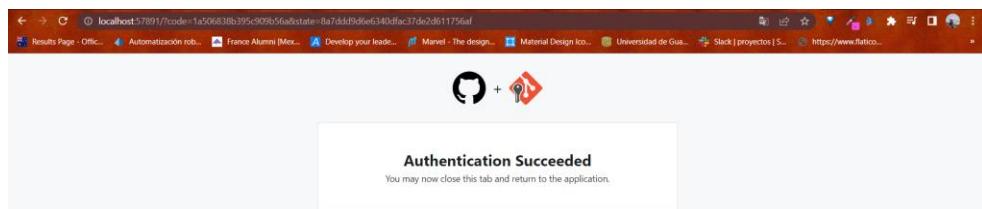
*Al querer conectar por primera vez nuestra cuenta de GitHub con la Git Bash por medio del comando git pull o git push se abrirá una ventana emergente, en la cual deberemos dar clic en el botón que dice Sign in with your browser.*



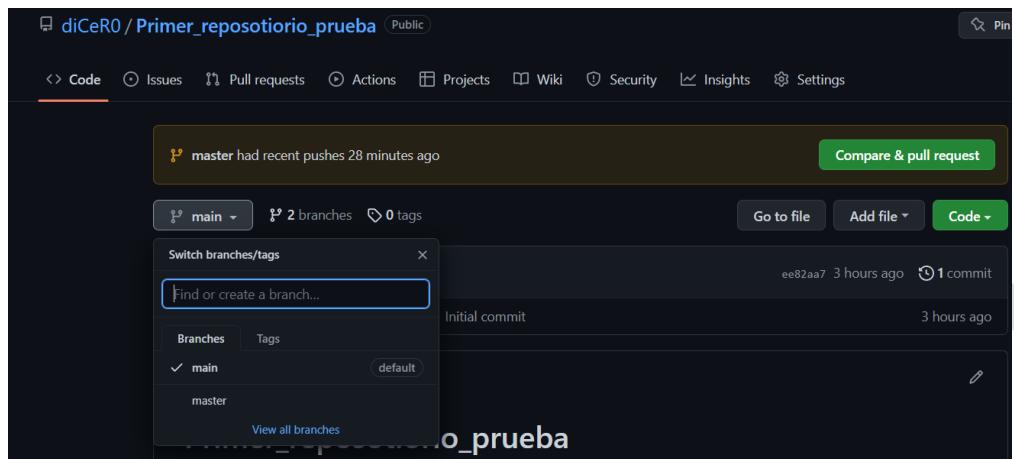
- b. Posteriormente se abrirá en el navegador una página en donde debemos indicar el nombre de usuario y la contraseña de GitHub si es que lo requiere, para finalmente dar clic en el botón de **Authorize GitCredential Manager**.



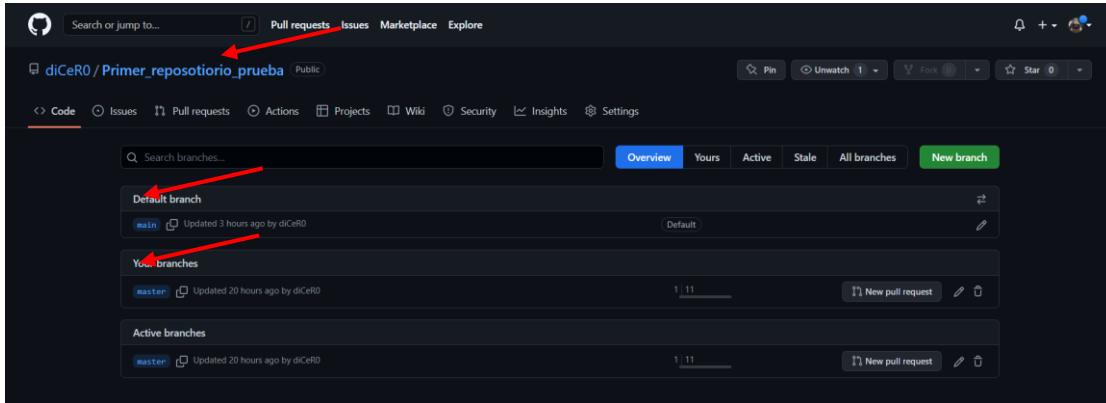
- c. Ya que haya introducido mi usuario y contraseña en la página, me aparecerá la siguiente ventana, indicando que se ha ejecutado correctamente el proceso.



- i. Recordemos que **la rama principal de Git Bash se llama master**, pero **la rama principal de GitHub se llama main**, esto en apoyo al movimiento de **#BlackLifesMatter**, por lo cual como se había mencionado previamente, debemos tomar en cuenta que el nombre de la branch al usar los comandos **git pull** y **git push** sea **main**, no **master**, sino se crearán dos ramas principales distintas en el proyecto, esto lo podré observar desde la página de Git Hub en donde se indican las ramas del repositorio.

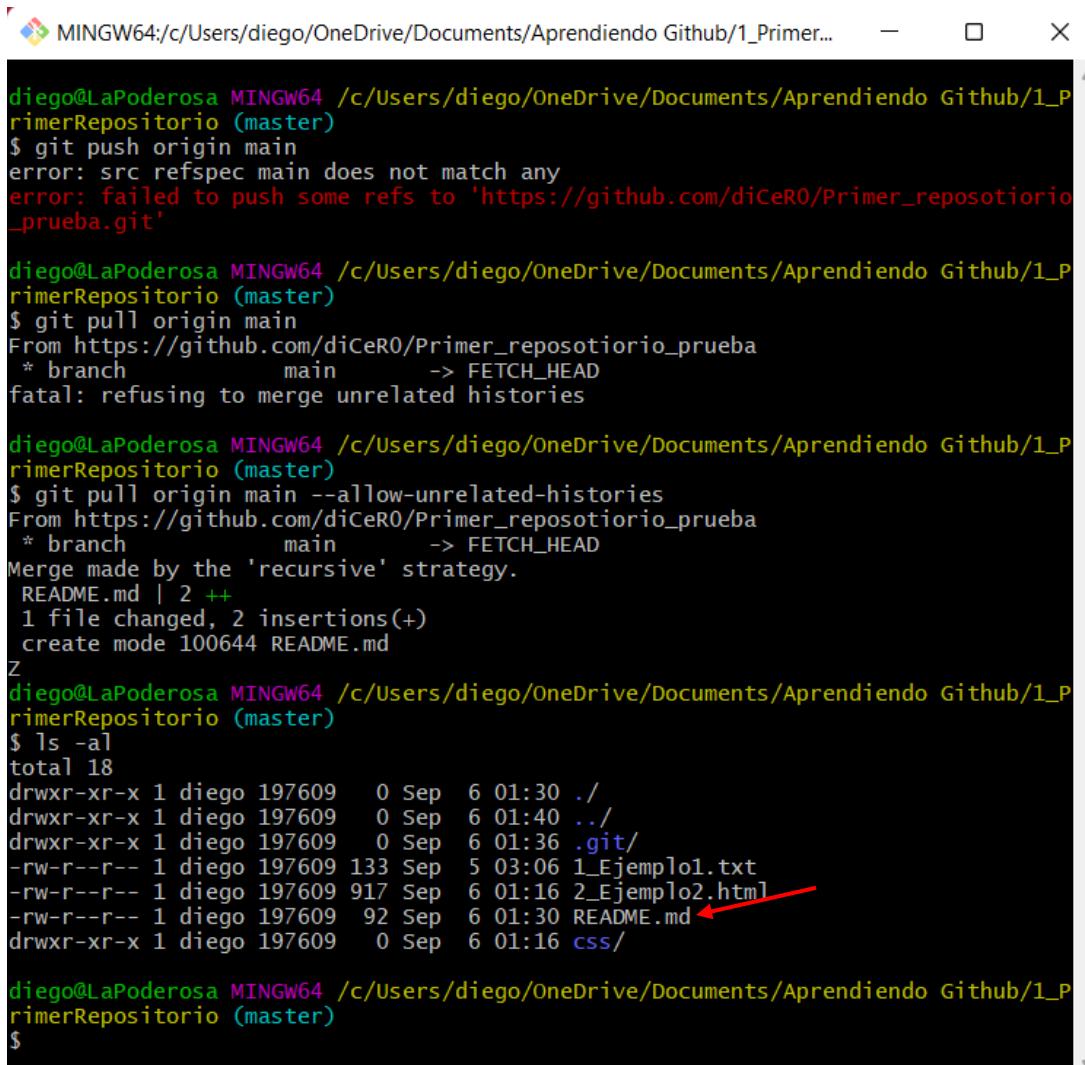


ii. En donde podremos notar que, en efecto, la rama principal o default del proyecto es la de **main**, no la de **master**.



The screenshot shows a GitHub repository named 'diCeR0 / Primer\_repositorio\_prueba'. The 'Code' tab is selected. In the center, there's a search bar for branches and a button to 'New branch'. Below it, a table lists branches under 'Default branch' (with 'main' listed) and 'Your branches' (with 'master' listed). There are also sections for 'Active branches'.

Una forma de confirmar que se hayan conectado correctamente ambos repositorios es que ahora podremos ver en los archivos del **Repository local** al archivo **README.md** cuando ejecutemos el comando de consola **ls -al**.



```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git push origin main
error: src refspec main does not match any
error: failed to push some refs to 'https://github.com/diCeR0/Primer_repositorio_prueba.git'

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git pull origin main
From https://github.com/diCeR0/Primer_repositorio_prueba
 * branch            main      -> FETCH_HEAD
fatal: refusing to merge unrelated histories

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git pull origin main --allow-unrelated-histories
From https://github.com/diCeR0/Primer_repositorio_prueba
 * branch            main      -> FETCH_HEAD
Merge made by the 'recursive' strategy.
 README.md | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 README.md
Z
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ ls -al
total 18
drwxr-xr-x 1 diego 197609  0 Sep  6 01:30 .
drwxr-xr-x 1 diego 197609  0 Sep  6 01:40 ..
drwxr-xr-x 1 diego 197609  0 Sep  6 01:36 .git/
-rw-r--r-- 1 diego 197609 133 Sep  5 03:06 1_Ejemplo1.txt
-rw-r--r-- 1 diego 197609 917 Sep  6 01:16 2_Ejemplo2.html
-rw-r--r-- 1 diego 197609  92 Sep  6 01:30 README.md
drwxr-xr-x 1 diego 197609  0 Sep  6 01:16 css/

```

Ya que se haya confirmado la conexión entre ambos repositorios podremos ejecutar el comando **git push origin main** desde la rama **main** para de que se vean las carpetas y archivos del proyecto en el **Repositorio remoto** mostrado en la página web de Git Hub, desde donde también se podrán hacer cambios a los archivos.

```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git status
On branch master
nothing to commit, working tree clean

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git branch
  main
* master
  nueva_rama

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (main)
$ git branch
* main
  master
  nueva_rama

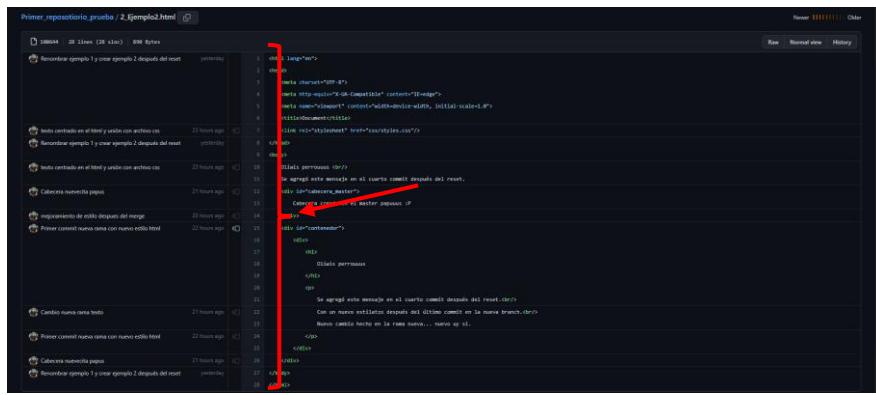
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (main)
$ git push origin main
Everything up-to-date

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (main)
$ |
```

The screenshot shows a GitHub repository page for 'diCeR0 / Primer\_reposotiorio\_prueba'. The repository has 2 branches and 0 tags. A red arrow points to a merge commit titled 'Merge branch 'main' of https://github.com/diCeR0/Primer\_repositorio...'. The commit message is 'mejoramiento de estilo despues del merge'. The commit was made by 'diCeR0' 19 minutes ago. The repository description is 'Repositorio remoto creado para pruebas iniciales de Git Hub'. The sidebar shows 0 stars, 1 watching, and 0 forks.

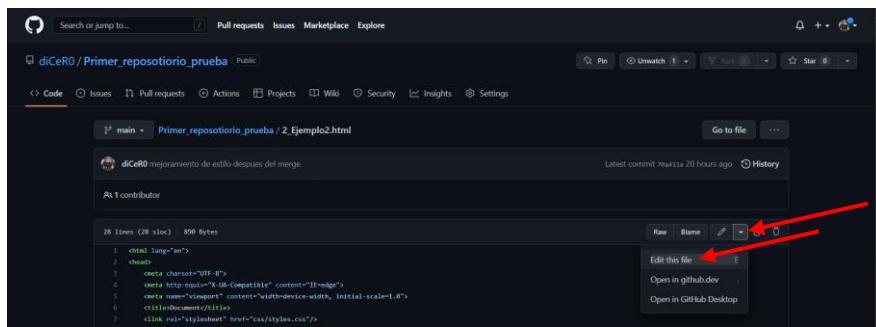
*La forma de acceder a cada archivo del proyecto es dar clic en él, esto para poder realizar cambios desde el sitio web de Git Hub u observar aspectos de los commits de mejor forma.*

*En la plataforma de GitHub usando el botón de Blame (ya habiendo dado clic en el archivo que queremos acceder), se puede ver de forma mucho más gráfica y visual quién fue el que hizo cada línea de código, con qué commit y que mensaje tiene cada uno de ellos.*

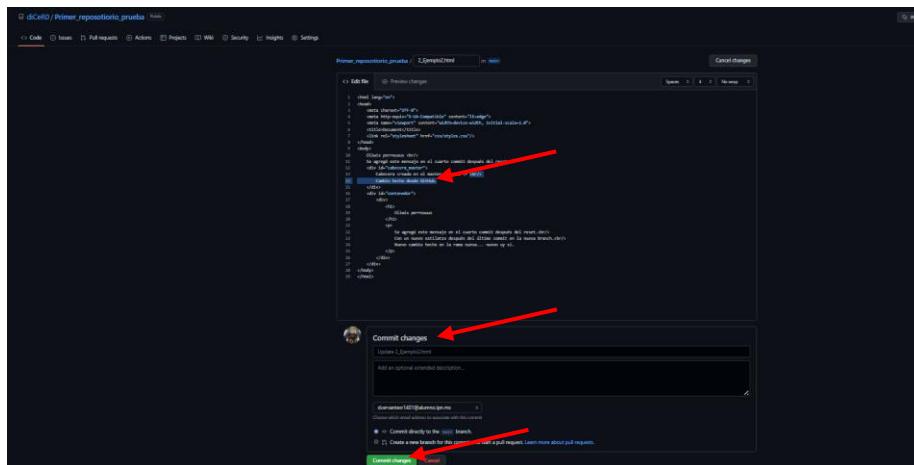


A screenshot of a GitHub repository page showing the blame view for a file named '2\_Ejemplo2.html'. The blame view displays the history of changes made to the file, with each line of code showing the author, date, and message of the commit that introduced that specific line. A red arrow points to the 'Blame' button in the top right corner of the code editor area.

*De igual manera se pueden editar los archivos desde GitHub seleccionando la opción de **Edit this file** cuando me introduzco en alguno de los archivos del repositorio remoto dentro del sitio web.*



*Al hacerlo me abrirá un editor de texto desde el cual puedo hacer cambios, añadir el mensaje del commit que estoy realizando y mandarlos desde el botón que dice **Commit changes**, pero para que esto se vea reflejado en el proyecto de mi computadora, debo ejecutar el comando **git pull origin nombre\_rama** desde la consola de Git Bash.*



Oliwis perrouus  
Se agregó este mensaje en el cuarto commit después del reset.

## Cabecera creada en el master papuuus :P Cambio hecho desde GitHub.

```
MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo_Github/1_Lección_1_repositorio [main]
git commit -am "Nuevo cambio hecho en la rama main"
[master (root-commit) 7b/c13c... 27760e3] main
  first commit
  1 file changed, 1 insertion(+), 2 deletions(-)
  create mode 100644 2.8temp02.html
git push -u origin main
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Delta compression using up to 64 kbytes/s.
remote: Compressing objects: 100% (3/3), done.
remote: Writing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0)
To https://github.com/cicero0Primer/repositorio_prueba
   7b/c13c... 27760e3 main        -> origin/main
```

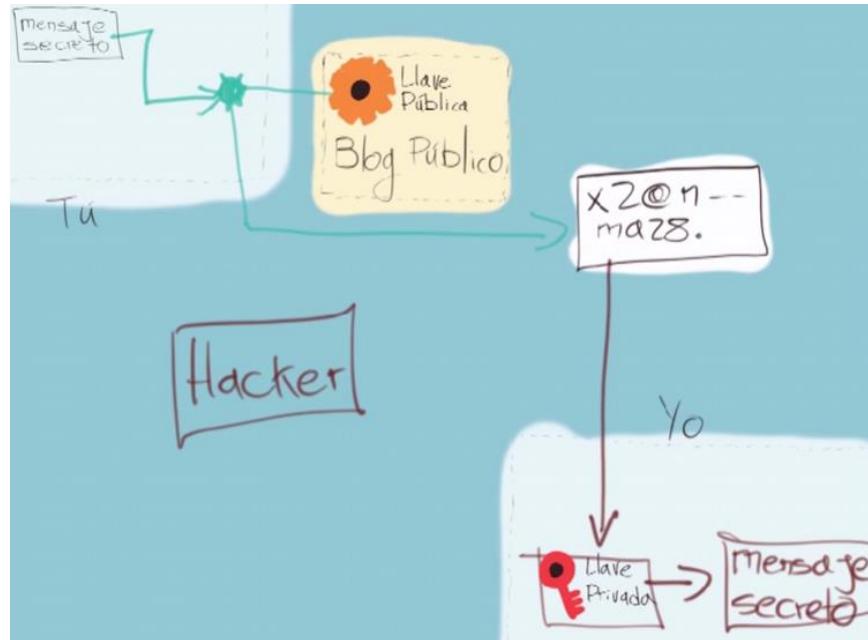
### Cifrado adicional entre repositorios locales y remotos de Git Hub (llaves SSH):

**Criptografía:** La criptografía es la ciencia que encripta mensajes secretos por medio de métodos matemáticos, un ejemplo de ello es el uso de números primos para crear claves y cifrar mensajes por medio de claves públicas y privadas.

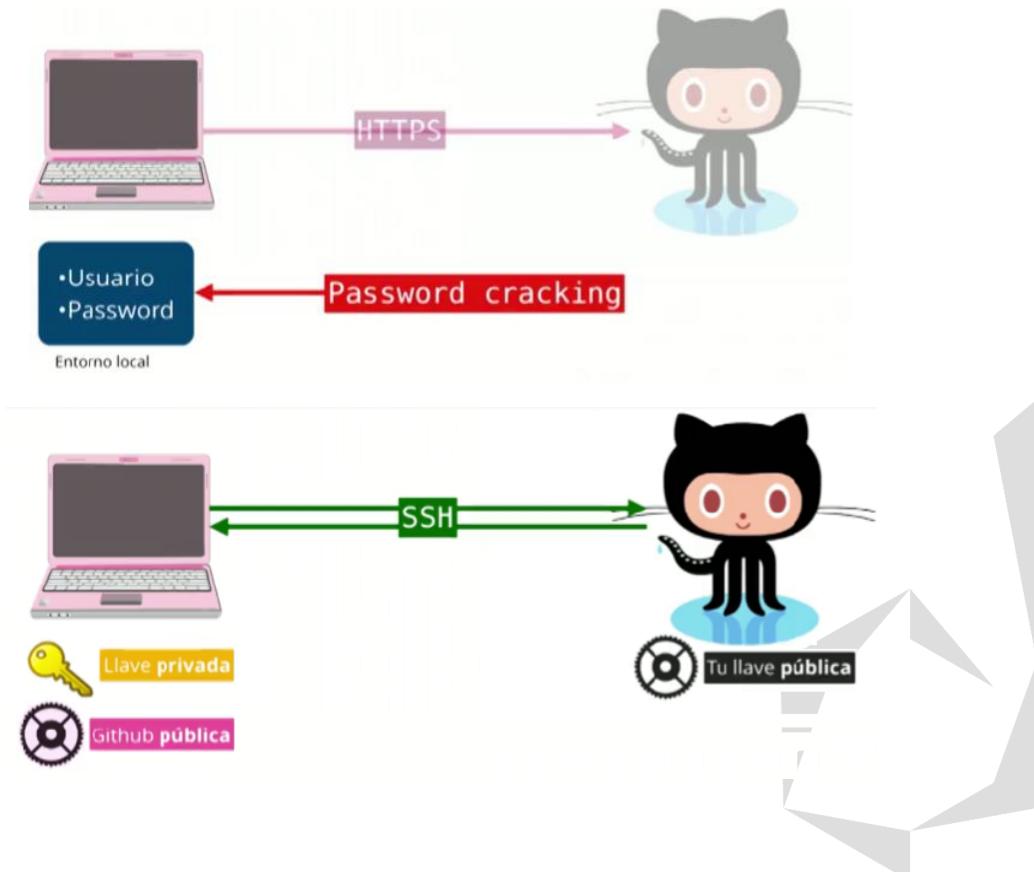
- **Clave pública (Public Key):** Se refiere a un número grande y público conformado de la multiplicación de otros dos números.
  - i. Utilizaré dos números primos para elaborar una clave pública, por ejemplo, el 3 y el 11, estos los multiplico y obtengo el número 33, ahora le digo a cualquier persona que quiera enviarme un mensaje secreto: “Quien quiera enviarme un mensaje secreto que utilice el número 33 como clave pública” esto lo puedo decir abiertamente, me da igual que cualquiera sepa que quien se quiera comunicar conmigo usará el número 33. Ahora, lo que debería hacer una persona que quisiera descifrar el mensaje encriptado, es hacer el proceso inverso, obteniendo los dos números primos que tuve que multiplicar para construir la clave pública. Esto cuando el número primo que conforma mi public key es muy grande se complica mucho.
- **Clave privada Private Key:** Se refiere a dos números primos que multiplicados crean un número grande que representa la clave pública.
  - i. Esos dos números primos que multiplicados dan como resultado mi clave pública son mi clave privada, en el ejemplo descrito anteriormente los números serían el 3 y el 11 que se usan para generar el número 33, que es mi clave pública.

Está fácil factorizar el número 33 dado en el ejemplo y obtener los números primos que lo conforman, pero con claves públicas enormes es cuando se ve la fuerza de este método, ya que para obtener esa clave se tuvieron que haber multiplicado dos números de clave privada muy grandes y obtener esos dos números primos que al multiplicarlos nos proporciona la clave pública es extremadamente difícil. Este método específico que usa números primos se llama **RSA**, pero no es el único método de cifrado existente.

Los métodos de cifrado no solamente impiden el acceso al mensaje secreto, sino que además al mismo mensaje lo vuelven un código incomprendible para que, aunque algún hacker intercepte el mensaje y sepa la clave pública, necesite utilizar la clave privada para poder decodificar el mensaje y que este sea entendible. Para que dos lados se puedan compartir un mensaje cifrado, ambos deben poseer una llave pública.



Esto se está mencionando porque ya nos conectamos al **Repositorio remoto** por medio de una URL de conexión **HTTPS**, pero hay un método de cifrado en GitHub mucho mejor, que sirve para que en el caso de que alguien me robe mi ordenador y se entere de mi usuario y contraseña de Git Bash, GitHub o ambas, exista una capa de seguridad extra que funcione a través de llaves públicas y privadas, asegurando así que el código fuente de mis proyectos esté resguardado, este método se llama **SSH o Secure SHell**, que es el mismo protocolo que se usa para conectarnos a servidores remotos por medio de consola.



*Las llaves SSH no se crean por repositorio o proyecto, sino por cada computadora que se quiera conectar, si un usuario tiene 3 computadoras deberá tener 3 claves SSH, por lo cual debemos primero que nada situarnos en la carpeta de home de nuestro ordenador, a la cual accedemos por medio del comando de consola cd, después de haber ejecutado este comando aparecerá el siguiente símbolo en el directorio de la Git Bash: ~, luego deberé ejecutar el comando git config -l para observar el correo que está asociado al **Repositorio local** de mi equipo, que debe ser el mismo que esté asociado a mi cuenta de GitHub, si no es así se debe ejecutar el comando 2.b para cambiarlo.*

```

MINGW64:/c/Users/diego/AppData/Roaming/SPB_Data
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/
1_PrimerRepositorio (main)
$ cd
diego@LaPoderosa MINGW64 ~
$ git config -l
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=D:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential_https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=di_ce_r0
user.email=dcerantesr1401@alumno.ipn.mx

diego@LaPoderosa MINGW64 ~
$
```

Overview Repositories 6 Projects Packages Stars 1

**d1CeR0 / README.md**

- Hi, I'm @diCeR0
- Charismatic mechatronic inventor with a strong sense of loyalty and justice, persistent and competitive, oriented towards robotic optimization in cutting-edge technological projects to drive the modernization of people, organizations and processes.
- I'm interested in IT, artificial intelligence, Mobile & Web development, electronics, microcontrollers, IoT, robotics, etc.
- I'm currently learning AWS, artificial intelligence, neuronal networks and RPA.
- I'm looking to collaborate on IoT projects.
- How to reach me:
- <https://www.linkedin.com/in/diego-cervantes-rodriguez/>
- dcerantesr1401@alumno.ipn.mx

**Popular repositories**

<b>ahuevotl</b> ejemplo de como usar git Public	<b>examenDevf</b> Aplicación para imprimir la latitud, longitud y User Agent del usuario. Public
<b>mexicod3project</b> Public	<b>PRACTICAS_MICROS_EQ4</b> Public

**Edit profile**

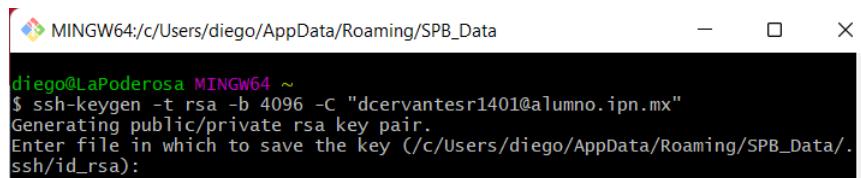
DUAL  
México  
[dcerantesr1401@alumno.ipn.mx](mailto:dcerantesr1401@alumno.ipn.mx)  
<https://sdual.com/>

*Ahora debemos crear nuestra clave SSH, para ello debemos escribir el siguiente código en la consola Git Bash, que representa la creación de un cifrado RSA que usa el ejemplo de números primos que explicamos anteriormente.*

22. **ssh-keygen -t rsa -b 4096 -C “email\_del\_usuario”**: Comando que crea un conjunto de llaves públicas y privadas por medio de un cifrado RSA que se basa en el uso de números primos para el encriptado de un mensaje secreto, que en este caso es la **conexión segura SSH (Secure SHell)** entre todos los repositorios **locales** y **remotos** que establezca con el correo asociado a mi Git Bash, que debe ser el mismo que está asociado a mi cuenta de Git Hub. **Esto se debe ejecutar desde la carpeta home de mi ordenador, a la cual accedo al ejecutar el comando de consola cd.**

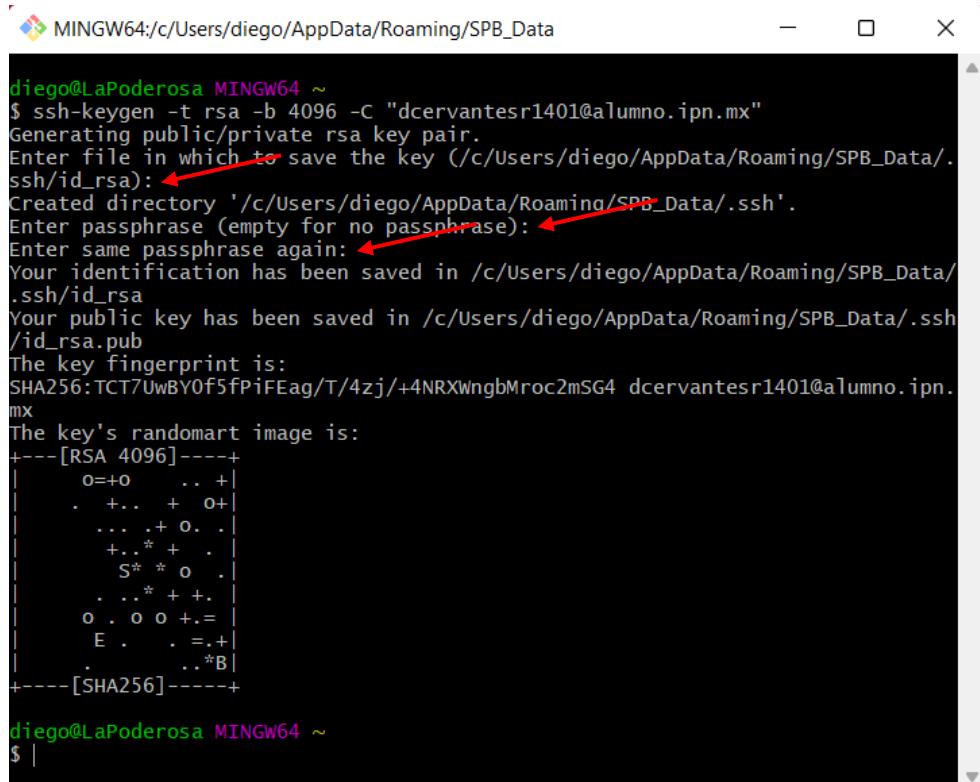
Para conectar la llave privada (**id\_rsa**) a mi computadora se siguen los siguientes pasos:

- Al ejecutar el comando se creará un archivo que guarde la **llave privada** en la carpeta indicada, debemos recordar cual es esta carpeta para poder ver la clave posteriormente. *La carpeta que guarda la clave estará oculta, esto lo sé porque antes de su nombre tiene un punto (.ssh), aquí solo se debe dar clic en la tecla ENTER.*



```
MINGW64:/c/Users/diego/AppData/Roaming/SPB_Data
diego@LaPoderosa MINGW64 ~
$ ssh-keygen -t rsa -b 4096 -C "dcervantesr1401@alumno.ipn.mx"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/diego/AppData/Roaming/SPB_Data/./
ssh/id_rsa):
```

- Después se puede indicar una contraseña adicional alfanumérica que se le va a asignar a mi llave pública y privada, luego se da clic dos veces en la tecla ENTER y aparecerá el siguiente mensaje en la consola. *Debemos de estar conscientes que deberemos introducir esta contraseña cada que ejecutemos un git pull o git push, para ver si la queremos declarar o no.*



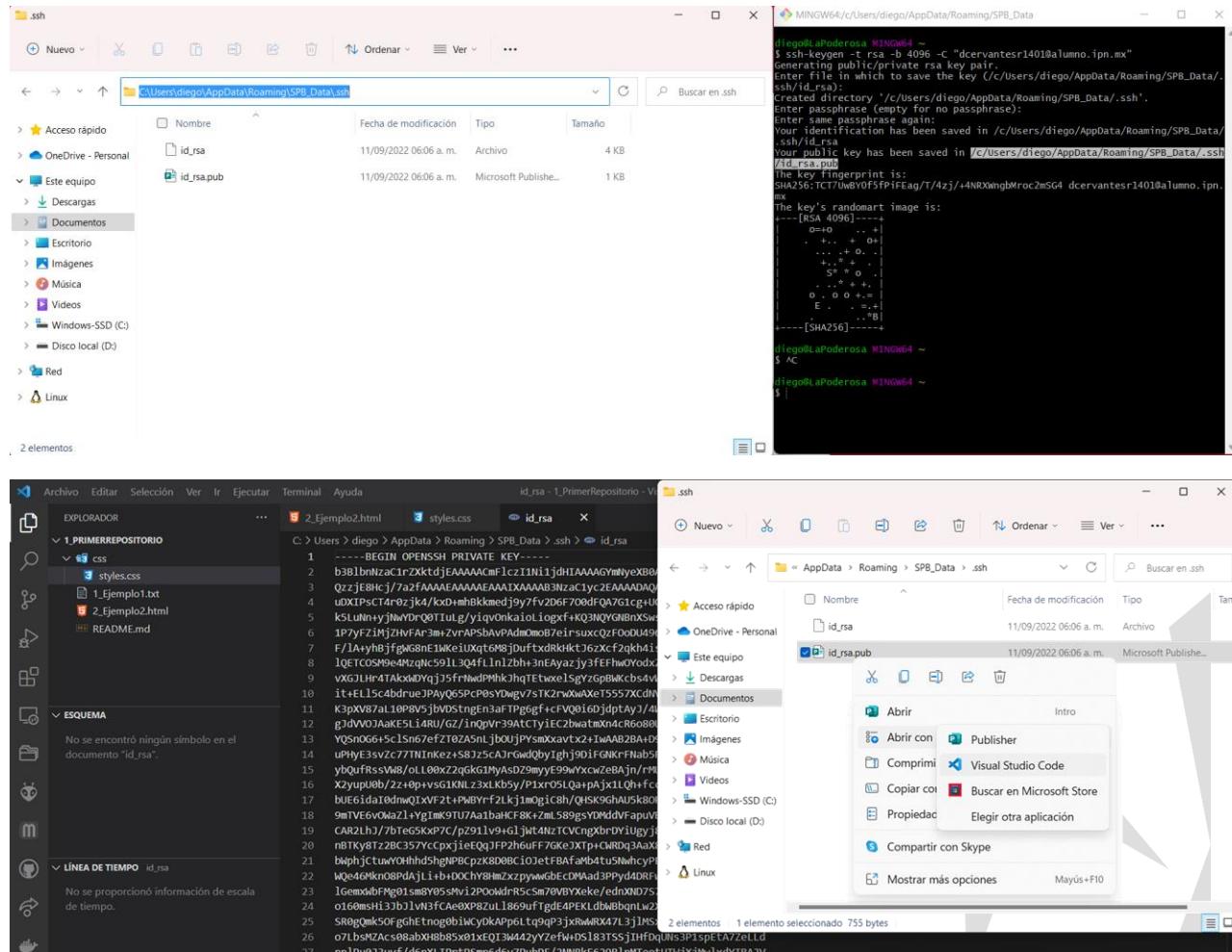
```
MINGW64:/c/Users/diego/AppData/Roaming/SPB_Data
diego@LaPoderosa MINGW64 ~
$ ssh-keygen -t rsa -b 4096 -C "dcervantesr1401@alumno.ipn.mx"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/diego/AppData/Roaming/SPB_Data/./
ssh/id_rsa): ←
Created directory '/c/Users/diego/AppData/Roaming/SPB_Data/.ssh'.
Enter passphrase (empty for no passphrase): ←
Enter same passphrase again: ←
Your identification has been saved in /c/Users/diego/AppData/Roaming/SPB_Data/
.ssh/id_rsa
Your public key has been saved in /c/Users/diego/AppData/Roaming/SPB_Data/.ssh/
id_rsa.pub
The key fingerprint is:
SHA256:TCT7UwBY0f5fPiFEag/T/4zj/+4NRXWngbMroc2mSG4 dcervantesr1401@alumno.ipn.
mx
The key's randomart image is:
+---[RSA 4096]---+
|   o=+o   . + |
|   . +... +  o+ |
|   ... +. o. . |
|   +...* +  . |
|       S* * o  . |
|   . . .* + +. |
|   o . o o +.= |
|       E . . =.+ |
|           . .*B |
+---[SHA256]---+
diego@LaPoderosa MINGW64 ~
$ |
```

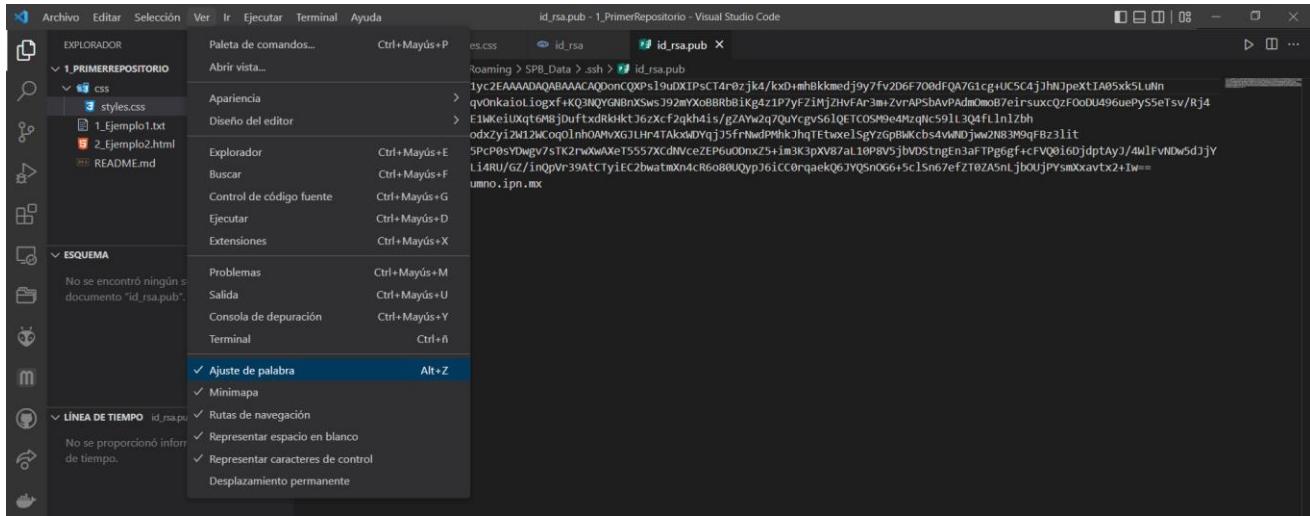
- i. **ssh-keygen -p:** Comando usado por si quiero cambiar la contraseña adicional o la ubicación de las claves SSH.

```
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ ssh-keygen -p
Enter file in which the key is (/c/Users/diego/AppData/Roaming/SPB_Data/.ssh/id_rsa):
Enter old passphrase:
Key has comment 'dcervantesr1401@alumno.ipn.mx'
Enter new passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved with the new passphrase.

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ |
```

- c. Para poder ver las llaves pública y privada debemos introducirnos al directorio indicado en la consola cuando ejecutamos el comando, donde:
- El archivo id\_rsa corresponde a la llave privada.
  - El archivo id\_rsa.pub corresponde a la llave pública.
1. Es recomendable abrir ambos archivos con el editor de código **Visual Studio Code**.





2. **eval \$(ssh-agent -s)**: Comando para confirmar que el servidor de SSH está corriendo, esto se confirma con el siguiente mensaje, que muestra un número de puerto distinto para cada ordenador: **Agent pid número\_puerto**.
3. **ssh-add ~/ssh/id\_rsa**: Comando que agrega la **llave privada** que acabamos de crear a nuestro sistema operativo, indicando la ruta donde está guardada, que hace referencia al home por medio del símbolo de ~, al terminar de hacerlo me mostrará el mensaje que dice: *Identidad añadida y la ruta donde la añadió.*

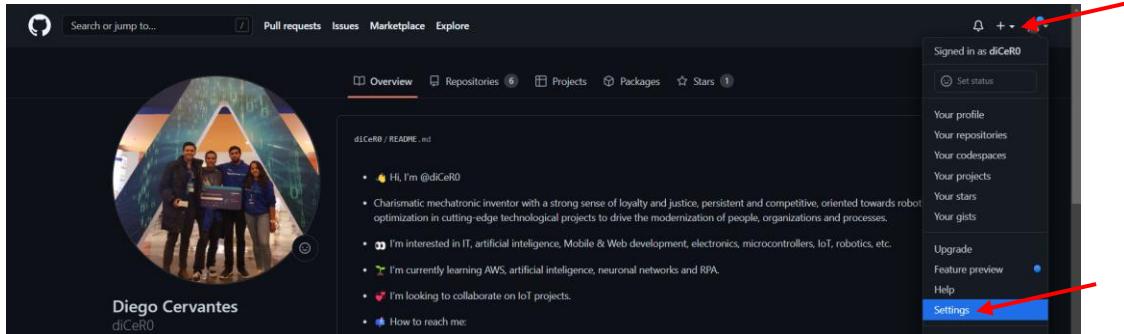
```
MINGW64:/c/Users/diego/AppData/Roaming/SPB_Data
diego@LaPoderosa MINGW64 ~
$ eval $(ssh-agent -s)
Agent pid 1201

diego@LaPoderosa MINGW64 ~
$ ~
bash: /c/Users/diego/AppData/Roaming/SPB_Data: Is a directory

diego@LaPoderosa MINGW64 ~
$ ssh-add ~/ssh/id_rsa
Enter passphrase for /c/Users/diego/AppData/Roaming/SPB_Data/.ssh/id_rsa:
Identity added: /c/Users/diego/AppData/Roaming/SPB_Data/.ssh/id_rsa (dcervante
sr1401@alumno.ipn.mx)

diego@LaPoderosa MINGW64 ~
$
```

Para conectar la **llave pública (id\_rsa.pub)** ahora debemos introducirnos a la siguiente opción de nuestro perfil de GitHub: *Configuración (Settings) → SSH and GPC keys → New SSH key → Title: Título de mi computadora → Key: Llave pública → Add SSH Key → Poner mi contraseña de GitHub → Aparecerá la nueva clave añadida en SSH Keys.*



Screenshot of GitHub user profile page for 'Diego Cervantes'. A red arrow points to the 'SSH and GPG keys' section in the left sidebar.

**Public profile**

- SSH and GPG keys **(highlighted)**

**Bio**

Charismatic mechatronic inventor with a strong sense of loyalty and justice, persistent and competitive, oriented towards robotic optimization and technology.

Screenshot of GitHub user settings page for 'Diego Cervantes'. A red arrow points to the 'New SSH key' button.

**SSH keys**

New SSH key **(highlighted)**

Screenshot of GitHub user settings page for 'Diego Cervantes' showing the 'Add new SSH key' dialog. A red arrow points to the 'Title' field with the value 'Laptop\_la\_poderosa'.

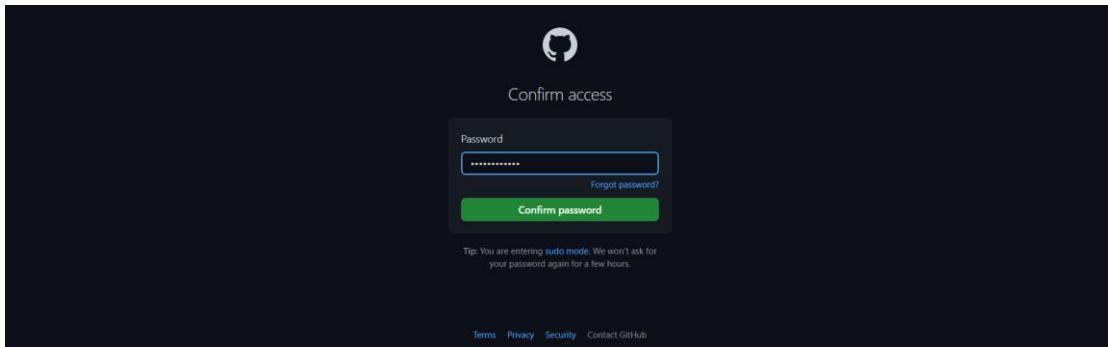
**SSH keys / Add new**

Title: Laptop\_la\_poderosa **(highlighted)**

Key type: Authentication Key

```
ssh-rsa
AAAAAB3NzaC1yc2EAAQABAAQADQ08AAACQD0nCQXPs...
... (long string of characters)
```

Add SSH key **(highlighted)**



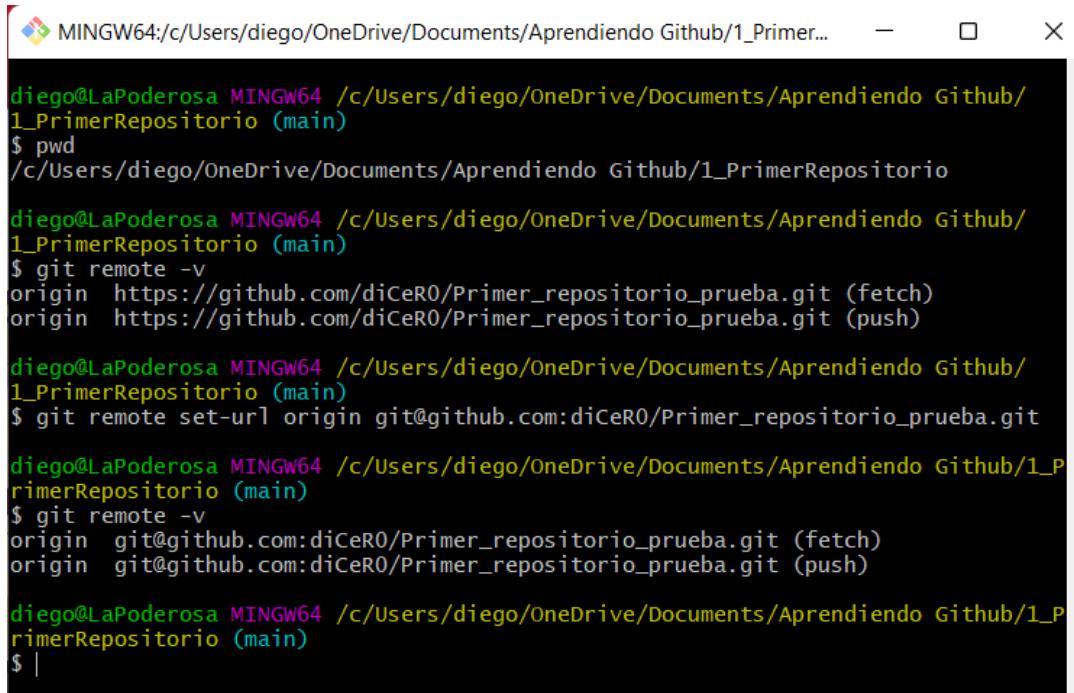
*Ahora debemos irnos a nuestro repositorio dentro de la página web de GitHub e introducirnos a la siguiente opción: Code → SSH → Copiar la URL.*

*Después de haber copiado esta URL nos introduciremos a la carpeta del proyecto en la Git Bash por medio de los **comandos de consola CMD**, donde luego de habernos conectado previamente a un **repositorio remoto** que normalmente se llama **origin** por medio del comando 18, ejecutaremos un comando que cambia la url **HTTPS** previamente asociada al repositorio por la nueva **url SSH**.*

23. **git remote set-url nombre\_repositorio url\_ssh\_extraida\_de\_github:** Comando que cambia la conexión HTTPS previamente hecha entre un **Repositorio local** y un **Repositorio remoto** por una conexión SSH cifrada. Se debe haber creado previamente la llave pública y privada por

medio del comando 22, ya que al hacerlo podremos obtener la **URL** que utiliza este comando por medio del botón: *Code → SSH*. El **nombre del repositorio** utilizado siempre es **origin**, aunque se le puede poner el nombre que sea realmente.

- git remote remove origin:** Comando utilizado para eliminar la conexión actual entre un **Repositorio local** y un **Repositorio remoto**, sirve para cuando el **repositorio de GitHub** tuvo algún problema y deseamos borrarlo, pero queremos conservar el contenido del **Repositorio local** para conectarlo a un nuevo **Repositorio remoto** posteriormente.



```

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Primer...
$ pwd
/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ git remote -v
origin  https://github.com/diCeRO/Primer_repositorio_prueba.git (fetch)
origin  https://github.com/diCeRO/Primer_repositorio_prueba.git (push)

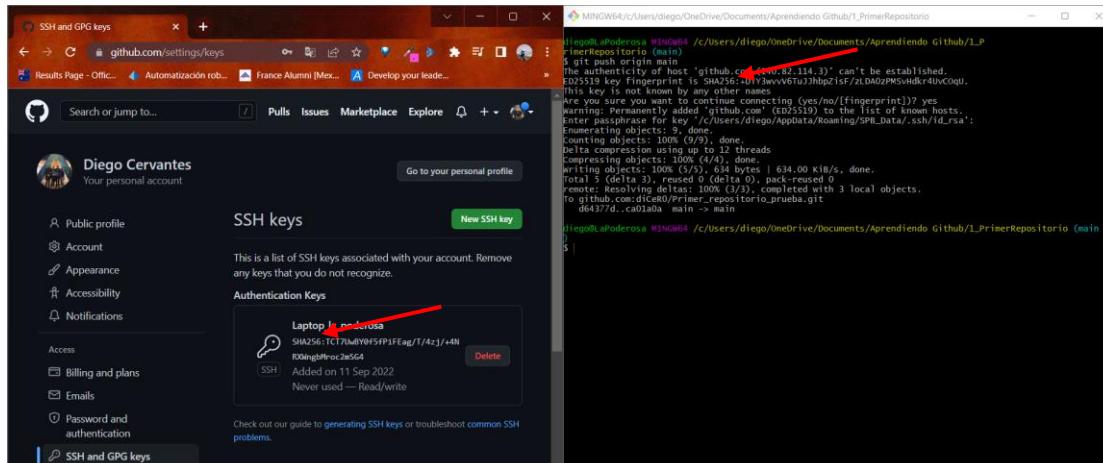
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ git remote set-url origin git@github.com:diCeRO/Primer_repositorio_prueba.git

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ git remote -v
origin  git@github.com:diCeRO/Primer_repositorio_prueba.git (fetch)
origin  git@github.com:diCeRO/Primer_repositorio_prueba.git (push)

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ |

```

La primera vez que realicemos un **git pull** o **git push** hacia el **Repositorio remoto** lo que pasará es que me pedirá que confirme que quiero realizar la conexión con el host y me mostrará el mismo número de id que tiene la key dentro de GitHub antes de los dos puntos (:), luego deberé introducir la contraseña alfanumérica si es que si la estoy usando y al hacerlo ya me permitirá empujar o jalar los cambios entre repositorios.

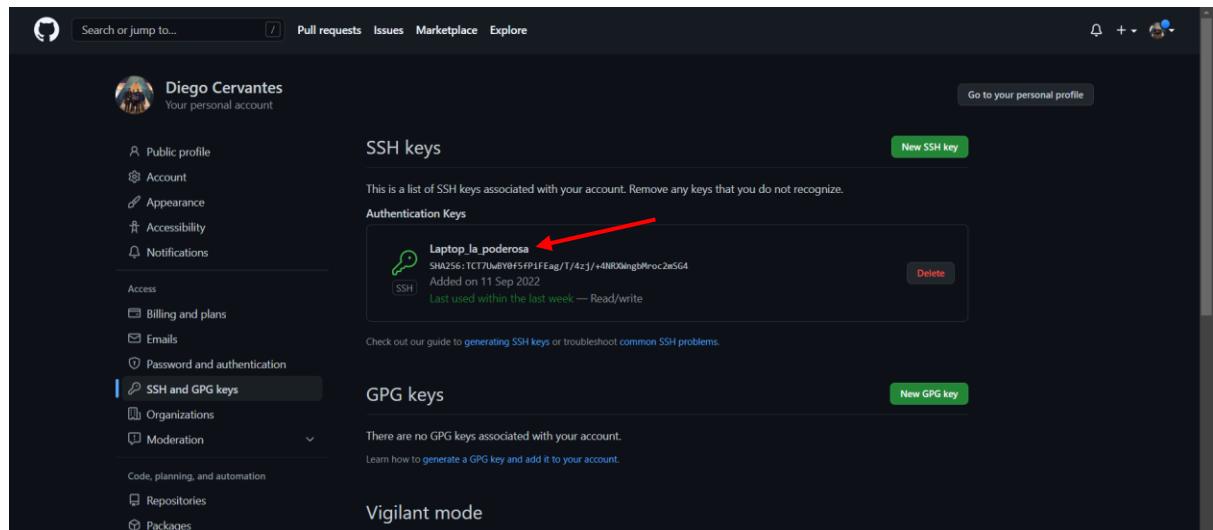


*El único cambio notable ahora es que cada que quiera hacer un **git pull** o **git push**, me pedirá introducir la contraseña que indiqué cuando cree la conexión SSH con la clave pública y privada SRA, esto lo puedo evitar si ejecuto el comando **22.b.i**, además de que ahora la llave cambia a color verde dentro de la página de GitHub porque ya está siendo usada.*

```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_Pri...
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/
rimerRepositorio (main)
$ git push origin main
The authenticity of host 'github.com (140.82.114.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMsvHdkr4UvC0qu.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Enter passphrase for key '/c/Users/diego/AppData/Roaming/SPB_Data/.ssh/id_rsa'
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 634 bytes | 634.00 KiB/s, done.
Total 5 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To github.com:diCeR0/Primer_repositorio_prueba.git
  d64377d..ca01a0a  main -> main

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/
)
$ git pull origin main
Enter passphrase for key '/c/Users/diego/AppData/Roaming/SPB_Data/.ssh/id_rsa'
:
From github.com:diCeR0/Primer_repositorio_prueba
 * branch            main      -> FETCH_HEAD
Already up to date.

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/
1_PrimerRepositorio (main)
$
```

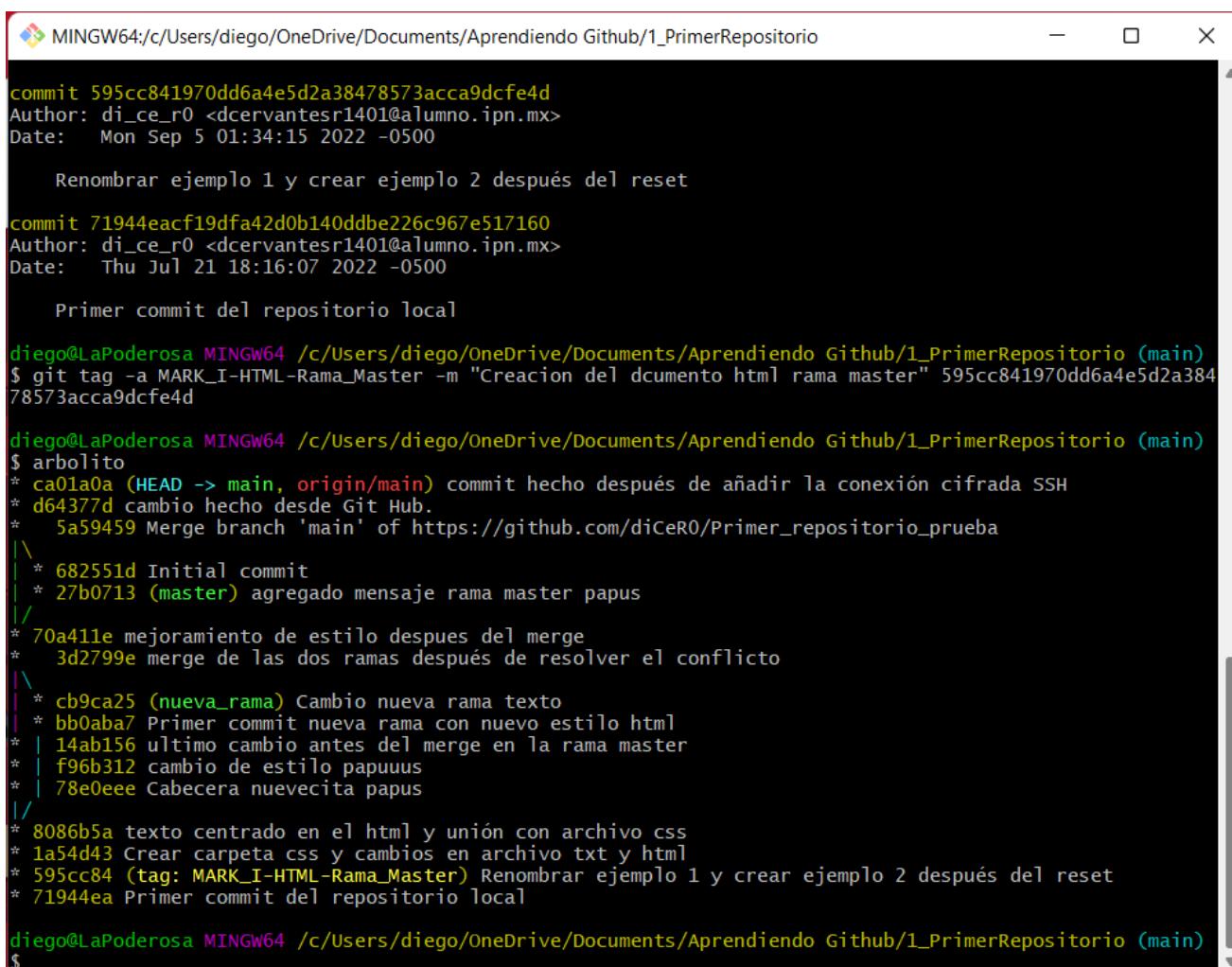


Releases de versiones en proyectos de GitHub (tags):

**Tag:** Un tag se refiere a un nombre especial con el que identifico un commit en específico para diferenciarse de los demás, parecido al comando alias, que se utilizó para guardar una instrucción muy larga en el comando 16.a.i.

24. **git tag -a nombre\_tag “nuevo\_mensaje\_del\_commit” número\_commit:** Comando que asocia cierto número de commit a un nombre en específico para que pueda ser mejor reconocido al observar todas las versiones del proyecto cuando se ejecute un comando en la consola de Git Bash como **git log –all –graph –decorate –oneline** o dentro de la plataforma de GitHub.

Al número del commit también se le llama hash.



The screenshot shows a terminal window titled "MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1\_PrimerRepositorio". The terminal displays the following command and its output:

```
commit 595cc841970dd6a4e5d2a38478573acca9dcfe4d
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date: Mon Sep 5 01:34:15 2022 -0500

Renombrar ejemplo 1 y crear ejemplo 2 después del reset

commit 71944eacf19dfa42d0b140ddbe226c967e517160
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date: Thu Jul 21 18:16:07 2022 -0500

Primer commit del repositorio local

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (main)
$ git tag -a MARK_I-HTML-Rama_Master -m "Creacion del documento html rama master" 595cc841970dd6a4e5d2a38478573acca9dcfe4d

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (main)
$ arbolito
* ca01a0a (HEAD -> main, origin/main) commit hecho después de añadir la conexión cifrada SSH
* d64377d cambio hecho desde Git Hub.
* 5a59459 Merge branch 'main' of https://github.com/diCeR0/Primer_repositorio_prueba
| \
| * 682551d Initial commit
| * 27b0713 (master) agregado mensaje rama master papus
| /
| * 70a411e mejoramiento de estilo despues del merge
| * 3d2799e merge de las dos ramas después de resolver el conflicto
| \
| * cb9ca25 (nueva_rama) Cambio nueva rama texto
| * bb0aba7 Primer commit nueva rama con nuevo estilo html
| * 14ab156 ultimo cambio antes del merge en la rama master
| * f96b312 cambio de estilo papuuus
| * 78e0eee Cabecera nuevecita papus
| /
* 8086b5a texto centrado en el html y unión con archivo css
* 1a54d43 Crear carpeta css y cambios en archivo txt y html
* 595cc84 (tag: MARK_I-HTML-Rama_Master) Renombrar ejemplo 1 y crear ejemplo 2 después del reset
* 71944ea Primer commit del repositorio local

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (main)
$
```

25. **git tag:** Comando que muestra un listado de todos los tags declarados en el proyecto, que básicamente son commits con nombres especiales para diferenciarlos de los demás y que a su vez aparecen dentro de la plataforma de GitHub por separado.

a. **git show-ref --tags:** Comando que muestra un listado de los números de commits junto con los tags que están asociados.

i. **git push origin --tags:** Comando que manda los tags creados en consola hacia GitHub.

- Lo importante no es que se vea el tag en la consola de Git Bash, sino en la plataforma de GitHub, por lo cual se debe primero ejecutar el comando `git pull origin main`, después `git push origin --tags` y aunque no parezca que pasa nada, ni que nunca hubo ningún cambio a mandarse en el **Staging Area**, ahora se podrá ver el tag en el sitio web de GitHub.

```

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ git tag
MARK_I-HTML-Rama_Master

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ arbolito
* ca01a0a (HEAD -> main, origin/main) commit hecho después de añadir la conexión cifrada SSH
* d64377d cambio hecho desde Git Hub.
* 5a59459 Merge branch 'main' of https://github.com/diCeRO/Primer_repository_prueba
| \
| * 682551d Initial commit
| * 27b0713 (master) agregado mensaje rama master papus
| /
| * 70a411e mejoramiento de estilo despues del merge
| * 3d2799e merge de las dos ramas después de resolver el conflicto
| \
| * cb9ca25 (nueva_rama) Cambio nueva rama texto
| * bb0aba7 Primer commit nueva rama con nuevo estilo html
| * 14ab156 ultimo cambio antes del merge en la rama master
| * f96b312 cambio de estilo papuuus
| * 78e0eee Cabecera nuevecita papus
| /
| * 8086b5a texto centrado en el html y unión con archivo css
| * 1a54d43 Crear carpeta css y cambios en archivo txt y html
| * 595cc84 (tag: MARK_I-HTML-Rama_Master) Renombrar ejemplo 1 y crear ejemplo 2 después del reset
| * 71944ea Primer commit del repositorio local

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ git show-ref --tags
1f0d665b20710af95969403753a3448b19f27fb refs/tags/MARK_I-HTML-Rama_Master

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ git status
On branch main
nothing to commit, working tree clean

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ git pull origin main
From github.com:diCeRO/Primer_repository_prueba
 * branch            main      -> FETCH_HEAD
Already up to date.

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ git push origin main
Everything up-to-date

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ git push origin --tags
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 203 bytes | 203.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:diCeRO/Primer_repository_prueba.git
 * [new tag]        MARK_I-HTML-Rama_Master -> MARK_I-HTML-Rama_Master

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$
```

*Ya después de haber ejecutado el comando `git push origin --tags`, lo que pasará es que en donde se muestran las ramas del proyecto dentro de la plataforma de GitHub, aparecerá una nueva opción*

en la pestaña que dice **Tags** y al dar clic en ella, podremos acceder a los archivos correspondientes al commit que declaramos como tag dentro de la consola Git Bash, esto se usa mucho cuando se ha alcanzado un progreso grande en un proyecto largo, para que así se guarde ese tipo de checkpoint.

The first screenshot shows the repository 'diCeR0 / Primer\_repositorio\_prueba' with a dropdown menu open over the 'main' branch. It lists several tags, including 'MARK\_I-HTML-Rama\_Master' and 'View all tags'. A red arrow points from the 'Tags' tab in the dropdown to the tag 'MARK\_I-HTML-Rama\_Master'. The second screenshot shows the same repository after a reset, with a new branch 'MARK\_I-HTML-R...' and a tag '1\_Ejemplo1.txt'. A red arrow points from the branch name to the tag. The third screenshot shows the repository after changes, with a commit message 'Renombrar ejemplo 1 y crear ejemplo 2 después del reset' and two files: '1\_Ejemplo1.txt' and '2\_Ejemplo2.html'. A red arrow points from the commit message to the file '2\_Ejemplo2.html'.

- b. **git tag -d nombre\_tag:** Comando para borrar un tag, se debe ejecutar junto con otro comando para que se borre realmente dentro de la plataforma GitHub.
  - i. **git push origin :refs/tags/nombre\_tag:** Comando que se ejecuta después del anterior descrito y se usa en conjunto con el comando **git push origin --tags** para borrar un tag de Git Hub.

```

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository
$ git tag
MARK_I-HTML-Rama_Master

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ git tag -d MARK_I-HTML-Rama_Master
Deleted tag 'MARK_I-HTML-Rama_Master' (was 1f0d665)

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ git tag

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ git pull origin main
From github.com:diCeR0/Primer_repositorio_prueba
 * branch            main       -> FETCH_HEAD
Already up to date.

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ git push origin --tags
Everything up-to-date

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ |

```

The screenshot shows the GitHub interface for a repository named 'Primer\_repository\_prueba'. The 'Code' tab is selected. A dropdown menu is open over the 'Tags' section, with the 'Tags' option highlighted by a red arrow. The menu also includes 'Branches' and 'View all tags'. Below the dropdown, there is a list of commits:

- la conexión cifrada SSH (ca01a0a 1 hour ago) 15 commits
- commit hecho después de añadir la conexión cifrada SSH (1 hour ago)
- Crear carpeta css y cambios en archivo txt y html (6 days ago)
- commit hecho después de añadir la conexión cifrada SSH (1 hour ago)
- Initial commit (5 hours ago)

On the right side of the screen, there is an 'About' section with the repository's description: 'Primer repositorio de prueba para refreshar conocimientos de Git'. It also shows statistics: 0 stars, 1 watching, and 0 forks.

```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ git tag
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ git push origin :refs/tags/MARK_I-HTML-Rama_Master
To github.com:diCeR0/Primer_repository_prueba.git
 - [deleted]      MARK_I-HTML-Rama_Master
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$
```

This screenshot is similar to the one above, but the 'Tags' section in the dropdown menu now displays 'Nothing to show'. The 'View all tags' link is highlighted by a red arrow. The commit list and repository details are identical to the first screenshot.

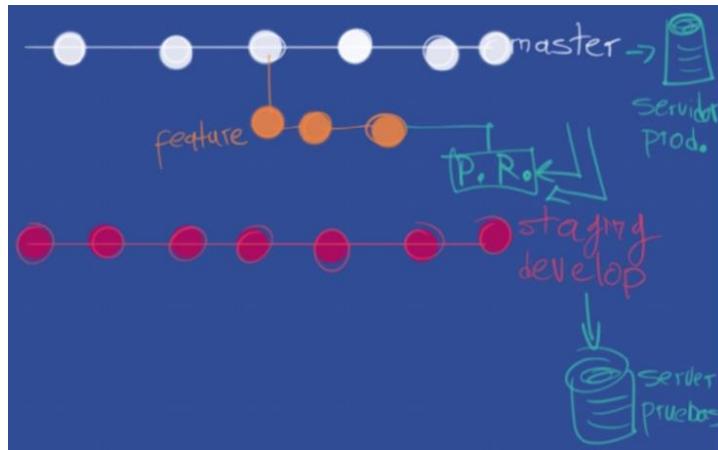
Manejo de ramas en un repositorio remoto/Pull Request:

*Por buenas prácticas se considera que hay una rama principal llamada **main** y una copia suya de pruebas llamada **staging develop** que siempre debe estar actualizada para que sean iguales. A la rama de desarrollo o pruebas **staging develop** es a donde se agregan los **features**, las cuales son ramas de módulos adicionales que están en desarrollo y la rama **hotfix** que es una o varias ramas para arreglar errores.*

*De igual manera existe el concepto de **Pull Request**, que se refiere a un estado intermedio antes de ejecutar un **merge** entre dos ramas situadas en un **Repositorio remoto** subido en el sitio web de GitHub, permitiéndonos así que otros miembros del repositorio puedan checar los cambios que se hicieron en la rama de **features** o **hotfix** antes de aprobar que se unan a la rama de **staging develop**, para que cuando ya todo el código de la rama de **desarrollo** funcione bien, se fusioné con la rama*

principal (*de producción*) llamada **main** y el proyecto se considere como concluido, o que mínimo ha llegado a una versión digna de asignarse a un Tag (*Release*). La herramienta de **Pull Request (P.R.)** no es parte de la consola *Git Bash*, sino de la plataforma *GitHub*.

En la vida real, la rama de producción **main** y su copia de pruebas **staging develop** cuentan con su propio servidor o dominio, donde además vale la pena mencionar que el puesto de trabajo que maneja todos estos pasos en una empresa se llama **DevOps**, el cual básicamente es un administrador del entorno de desarrollo de un proyecto, manejando la forma de trabajo de los programadores.



Dentro de *GitHub* cada rama se maneja de forma separada (como archivos completamente diferentes que muestran versiones distintas de un mismo proyecto) y a su vez no están relacionadas en un inicio con las ramas de *Git Bash* hasta que se vayan subiendo una a una de forma manual.

- Los comandos 12, 13, 14 y 15 (**git branch** y **git checkout**) se encargan de manejar las branches que existen actualmente en mi **Repositorio local** o de hacer ramas nuevas para posteriormente ejecutar el comando **git push origin nombre\_rama** y de esa manera crearlas en la plataforma de *GitHub*.

Terminal de Git Bash:

```
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo GitHub/1_PrimerRepositorio (main)
$ git branch
* master
  nueva_rama
  prueba_binaria
$
```

Interfaz web de GitHub (Captura de pantalla de una página de cambios en la rama master):

Oliwis perrouus  
Se agregó este mensaje en el cuarto commit después del reset.  
Cabeecera creada en el master papuuus :P  
Cambio hecho desde Git Hub.  
Cambio hecho después de añadir la conexión SSH.

Oliwis perrouus  
Se agregó este mensaje en el cuarto commit después del reset.

Rama master papuuus

Oliwis perrouuuus

Se agregó este mensaje en el cuarto commit después del reset.

# Cabecera creada en el master papuuus :P

## Oliwis perrouuuus

Se agregó este mensaje en el cuarto commit después del reset.  
Con un nuevo estilatzo después del último commit en la nueva branch.  
Nuevo cambio hecho en la rama nueva...  
nuevo uy si.

```

MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git branch
* master
  nueva_rama
  prueba_binaria

diego@Anderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (main)
$ git checkout master
Switched to branch 'master'
diego@Anderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git branch
* master
  nueva_rama
  prueba_binaria

diego@Anderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ 

```

Oliwis perrouuuus

Se agregó este mensaje en el cuarto commit después del reset.  
Con un nuevo estilatzo después del último commit en la nueva branch.  
Nuevo cambio hecho en la rama nueva... nuevo uy si.

```

MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git branch
* master
  nueva_rama
  prueba_binaria

diego@Anderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (main)
$ git checkout master
Switched to branch 'master'
diego@Anderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git branch
* master
  nueva_rama
  prueba_binaria

diego@Anderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git checkout nueva_rama
Switched to branch 'nueva_rama'
diego@Anderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (nueva_rama)
$ git branch
* nueva_rama
  master
  prueba_binaria

diego@Anderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (nueva_rama)
$ 

```

Oliwis perrouuuus

Se agregó este mensaje en el cuarto commit después del reset.

# Cabecera creada en el master papuuus :P

## Cambio hecho desde Git Hub.

[Cambio hecho después de añadir la conexión SSH.](#)  
[Prueba binaria de subir una imagen \(archivo binario\) a GitHub.](#)

## Oliwis perrouuuus

```

MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git branch
* master
  nueva_rama
  prueba_binaria

diego@Anderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (main)
$ git checkout master
Switched to branch 'master'
diego@Anderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git branch
* master
  nueva_rama
  prueba_binaria

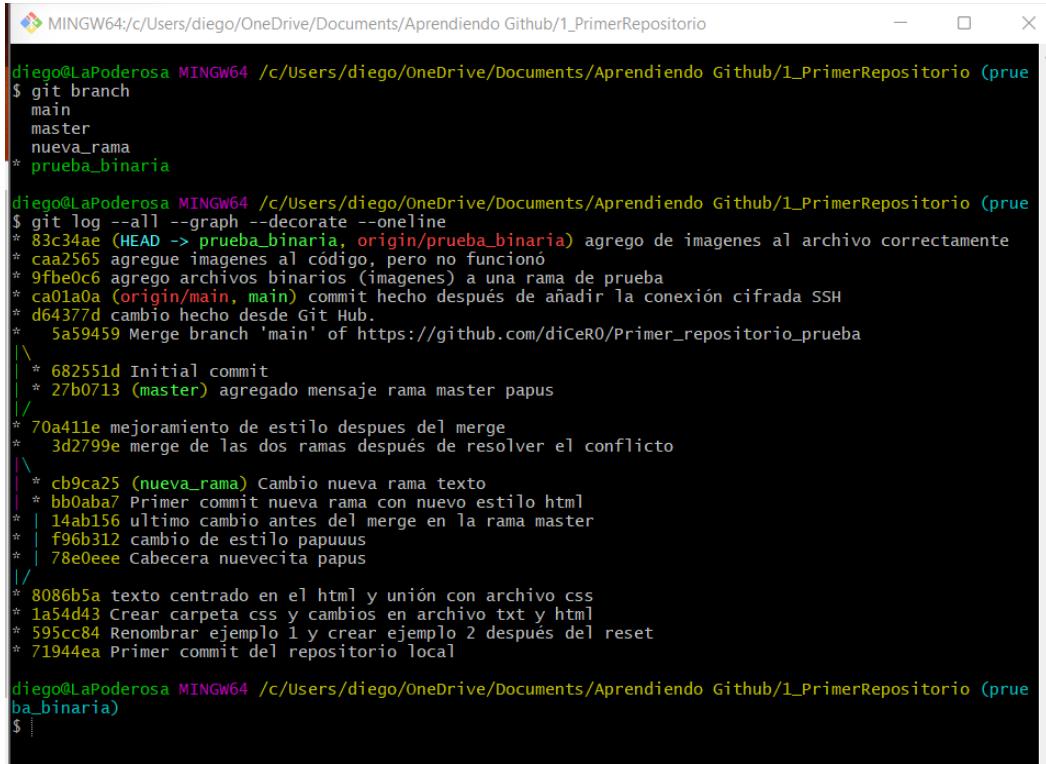
diego@Anderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (master)
$ git checkout nueva_rama
Switched to branch 'nueva_rama'
diego@Anderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (nueva_rama)
$ git branch
* nueva_rama
  master
  prueba_binaria

diego@Anderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (nueva_rama)
$ git checkout prueba_binaria
Switched to branch 'prueba_binaria'
diego@Anderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (prueba_binaria)
$ git branch
* prueba_binaria
  master
  nueva_rama
  prueba_binaria

diego@Anderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (prueba_binaria)
$ 

```

Cada rama tiene versiones distintas de mi proyecto como se acaba de mostrar y también puedo observar de una manera muy gráfica como conviven todas las ramas en la línea de tiempo de mi proyecto con el comando 16.a: **git log --all --graph --decorate --oneline**. Esta es la línea de tiempo que está conformada por todas las ramas presentadas anteriormente:



```

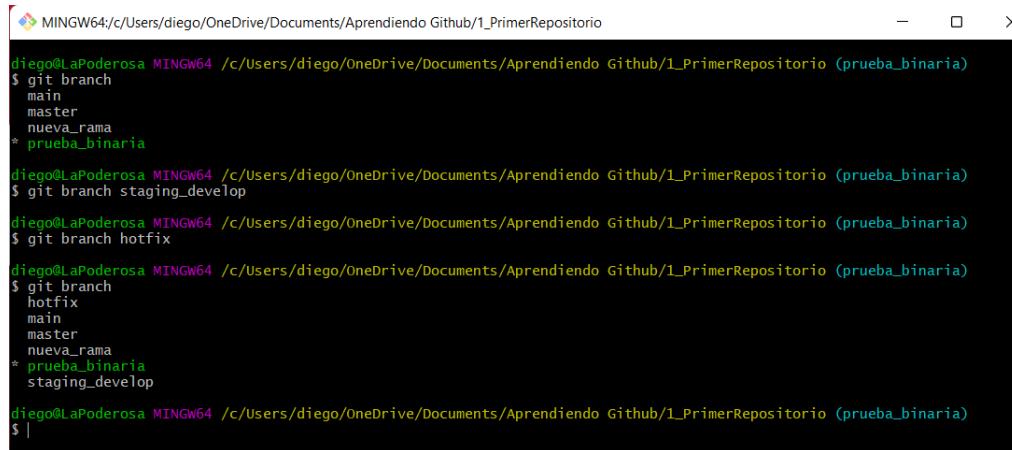
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (prueba_binaria)
$ git branch
  main
  master
  nueva_rama
* prueba_binaria

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (prueba_binaria)
$ git log --all --graph --decorate --oneline
* 83c34ae (HEAD -> prueba_binaria, origin/prueba_binaria) agrego de imagenes al archivo correctamente
* caa2565 agregue imagenes al código, pero no funcionó
* 9fbe0c6 agrego archivos binarios (imagenes) a una rama de prueba
* ca01a0a (origin/main, main) commit hecho después de añadir la conexión cifrada SSH
* d64377d cambio hecho desde Git Hub.
* 5a59459 Merge branch 'main' of https://github.com/diceR0/Primer_repository_prueba
|\ 
| * 682551d Initial commit
| * 27b0713 (master) agregado mensaje rama master papus
|/
* 70a411e mejoramiento de estilo despues del merge
* 3d2799e merge de las dos ramas despues de resolver el conflicto
|\ 
| * cb9ca25 (nueva_rama) Cambio nueva rama texto
| * bb0aba7 Primer commit nueva rama con nuevo estilo html
* 14ab156 ultimo cambio antes del merge en la rama master
* | f96b312 cambio de estilo papuus
* | 78e0eee Cabecera nuevecita papus
|/
* 8086b5a texto centrado en el html y unión con archivo css
* 1a54d43 Crear carpeta css y cambios en archivo txt y html
* 595cc84 Renombrar ejemplo 1 y crear ejemplo 2 después del reset
* 71944ea Primer commit del repositorio local

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (prueba_binaria)
$ ...

```

Las ramas que queramos subir al **Repositorio remoto** de GitHub primero se deben crear en mi **Repositorio local** con la consola Git Bash, por lo cual, vamos a hacer todas las ramas descritas anteriormente, que son la rama principal de **producción main**, la rama de **desarrollo staging develop**, la rama de **debug hotfix** y las ramas de **features**.



```

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (prueba_binaria)
$ git branch
  main
  master
  nueva_rama
* prueba_binaria

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (prueba_binaria)
$ git branch staging_develop
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (prueba_binaria)
$ git branch hotfix
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (prueba_binaria)
$ git branch feature
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (prueba_binaria)
$ |

```

- Los comandos 20 y 21 (git pull y push) se utilizan de la siguiente forma para copiar el contenido de una rama en la otra y así crear la rama de **desarrollo staging develop** o cualquier otra:

- *Primero que nada, debemos haber subido previamente la rama principal del proyecto (o cualquier otra) desde nuestro **Repositorio local** hacia el **Repositorio remoto** de GitHub por medio de los comandos:*
  - **git pull origin nombre\_rama\_subida.**
  - **git push origin nombre\_rama\_subida.**
- *Luego desde la consola Git Bash vamos a crear en nuestro **Repositorio local** una **rama nueva**, en la cual pretendemos crear una copia de la rama previamente subida al **Repositorio remoto** de GitHub, esto lo llevamos a cabo usando el comando:*
  - **git branch nombre\_rama\_nueva.**
- *Después nos situamos en la **rama nueva** donde queremos copiar el contenido de la otra, usando el comando:*
  - **git checkout nombre\_rama\_nueva.**
- *Posteriormente jalamos del **Repositorio remoto** el contenido de la rama que queremos copiar hacia nuestra rama actual (que es la **rama nueva**), esto se logra por medio del comando:*
  - **git pull origin nombre\_rama\_subida.**
- *Finalmente, lo que debemos hacer para que se cree esa **rama nueva** pero dentro del **Repositorio remoto** en la plataforma de GitHub es ejecutar el comando:*
  - **git push origin nombre\_rama\_nueva.**
- *Con eso ya será suficiente para que se haya creado una **rama nueva** en el repositorio dentro del sitio web de GitHub que tenga una copia del contenido de una **rama subida previamente**.*
  - *Podemos confirmar que en efecto existe una nueva rama dentro de GitHub en la pestaña que dice **Switch branches/tags**, donde dentro de la pestaña de **Branches** aparecerá una nueva rama con el mismo nombre que la **rama nueva** que pretendía añadir.*

The screenshot shows a dual-monitor setup. The left monitor displays a GitHub repository page for 'Primer\_repository\_prueba'. The repository has 0 stars and 0 forks. It lists branches: main, prueba\_binaria, and staging\_develop. The 'staging\_develop' branch is highlighted with a red arrow and labeled 'had recent pushes less than a minute ago'. The right monitor shows a terminal window titled 'MINGW64' with the following command history:

```

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository
$ git checkout staging_develop
switched to branch "staging_develop"
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (stag
$ git branch
* main
  master
  nueva_rama
  prueba_binaria
  staging_develop
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (stag
$ git pull origin main
From github.com:diCeR0/Primer_repository_prueba
 * branch      main            -> FETCH_HEAD
Already up to date.

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (stag
$ git push origin staging_develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Create a pull request for 'staging_develop' on GitHub by visiting:
remote:   https://github.com/diCeR0/Primer_repository_prueba/pull/new/staging_develop
remote:
To github.com:diCeR0/Primer_repository_prueba.git
 * [new branch]  staging_develop -> staging_develop
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (stag
$ 

```

- Además, dentro de la página del **Repository remoto** el número de branches aumenta, haciendo evidente que se ha añadido una nueva rama. Es importante mencionar que en los proyectos de GitHub no se suben ramas a lo loco, sino en realidad versiones que vale la pena conservar, mientras que dentro de mi **Repository local** si puedo tener las ramas que quiera creadas con Git Bash.

- Cuando realice un cambio se deben ejecutar los siguientes comandos en la consola de Git Bash:
  - git status:** Comando para ver si hay algún cambio que falta por mandarse en los archivos del **Repository local**.
  - git pull origin nombre\_rama\_nueva:** Comando que jala los cambios de la **rama nueva** que acabo de añadir al **Repository remoto** de GitHub hacia mi **Repository local**, hacer esto es de buenas prácticas por si algún otro colaborador del repositorio mandó un cambio nuevo antes de que yo pretendiera mandar uno.
  - git commit -am "mensaje":** Comando para mandar los cambios hacia el **Repository local**, de tal forma que puedan ser subidos al GitHub posteriormente.
  - git push origin nombre\_rama\_nueva:** Comando que envía los cambios de mi **Repository local** a la **rama nueva** que acabo de añadir al **Repository remoto** en la plataforma de GitHub.

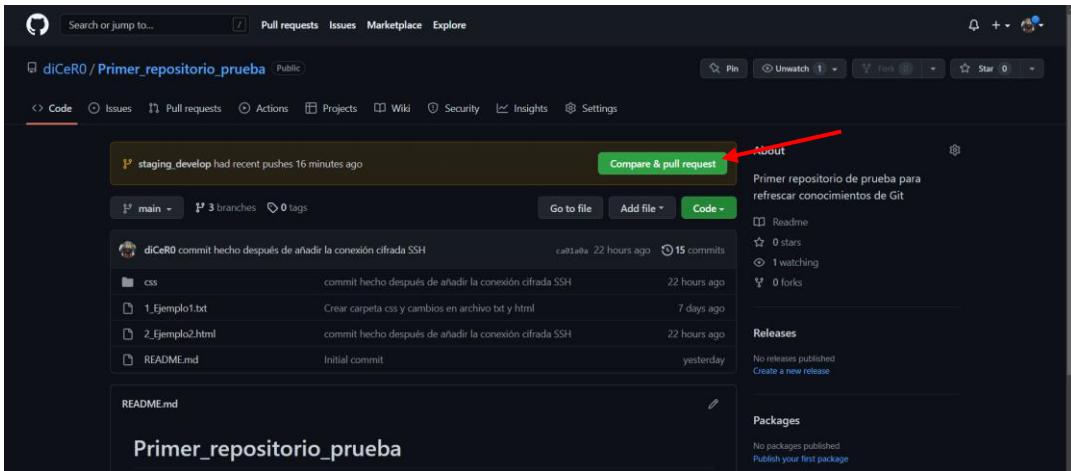
```

diego@diegopoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (staging_develop)
$ git status
On branch staging_develop
Change not staged for commit:
  (use "git add <file>" to update what will be committed)
    (use "git add --all <file>" to discard changes in working directory)
      modified: 2_Ejemplo2.html
      modified: css/styles.css
no changes added to commit (use "git add" and/or "git commit -a")
diego@diegopoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (staging_develop)
$ git pull origin staging_develop
From github.com:diCeR0/Primer_repositorio_prueba
 * [new branch] staging_develop -> staging_develop
Already up to date.

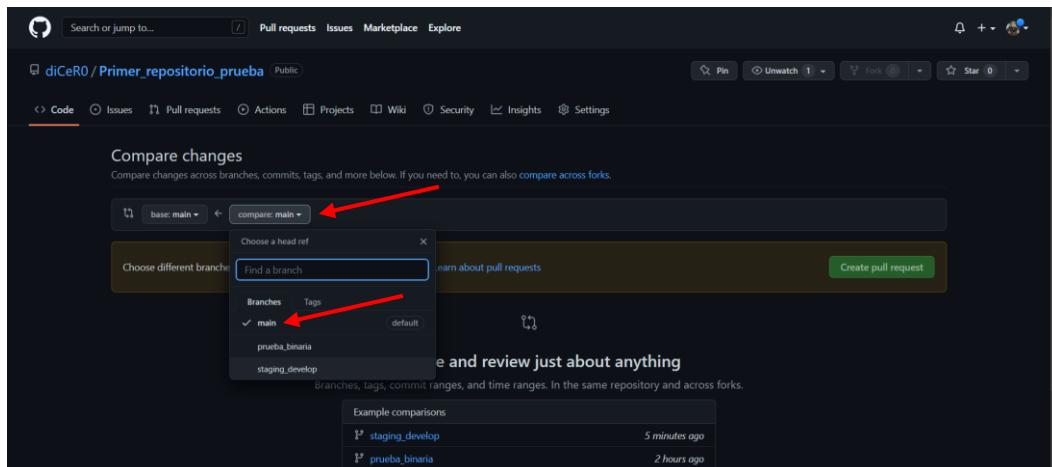
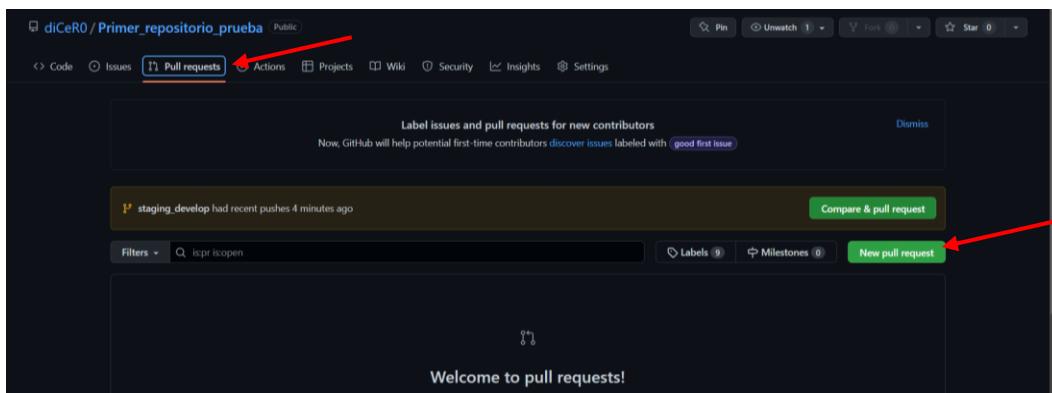
diego@diegopoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (staging_develop)
$ git commit -am "Cambios para activar la función de Pull request desde la rama staging develop"
[staging_develop 471cfca] Cambios para activar la función de Pull request desde la rama staging develop
 2 files changed, 2 insertions(+), 2 deletions(-)
diego@diegopoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (staging_develop)
$ git push origin staging_develop
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 12 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 566.00 KiB/s, done.
remote: Total 10 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 2 local objects.
To github.com:diCeR0/Primer_repositorio_prueba.git
 83c3ae..471cfca staging_develop -> staging_develop

```

- Al haber creado una **rama nueva** en el **Repositorio remoto de GitHub** es cuando aparece la opción de **Compare & Pull Request**, cuando le doy clic lo que hace es comparar la rama que indique en la pestaña de base: (rama principal del merge) con la rama que indique en la pestaña de compare: (rama que se va a fusionar con la de base), al final recordemos que el chiste de esto es fusionar ambas branches en una sola, cuya rama principal será la de base.



- Para llegar a la misma página de comparación también me puedo introducir a la opción de **Pull requests** → **New pull request** → **Seleccionar la rama base:** → **Seleccionar la rama compare:** → **Create pull request**.



The screenshot shows a GitHub repository page for 'diCeR0 / Primer\_repositorio\_prueba'. The 'Code' tab is selected. A comparison between the 'main' branch and the 'staging Develop' branch is shown. The message 'Able to merge' is displayed, indicating that the branches can be automatically merged. A red arrow points to the green 'Create pull request' button at the top right of the comparison interface.

- *En este punto puedo escribir:*
  - *El mensaje enviado a los demás colaboradores del Pull Request.*

The screenshot shows the 'Open a pull request' interface on GitHub. It displays a comparison between the 'main' branch and the 'staging Develop' branch. The message 'Able to merge' is shown. A red arrow points to the 'Leave a comment' text input field within the pull request form. To the right of the form, there are sections for 'Reviewers', 'Assignees', 'Labels', 'Projects', and 'Milestone'.

- *Más abajo puedo observar ver la línea de tiempo de los commits mandados hacia el repositorio.*

The screenshot shows the commit history for September 12, 2022, in the 'Primer\_repositorio\_prueba' repository. The history includes five commits from the 'staging' branch, with the most recent ones being from 15 minutes ago. The commits are: 'agrego archivos binarios (imagenes) a una rama de prueba', 'agregue imagenes al código, pero no funcionó', 'agrego de imagenes al archivo correctamente', 'Cambio para activar la función de Pull request desde la rama staging ...', and 'Cambios para activar la función de Pull request desde la rama staging...'. A red arrow points to the first commit message.

- Despues puedo ver exactamente los cambios hechos en cada uno de los archivos que conforman al proyecto, donde se mostrarán de **color rojo** las partes que sean diferentes entre las dos ramas y pertenezcan a la rama base: y de **color verde** las partes que sean distintas entre ambas ramas y pertenezcan a la rama compare:

④ 5 commits

④ 4 files changed

All 1 contributor

Commits on Sep 12, 2022

agrego archivos binarios (imagenes) a una rama de prueba  
sc4R0 committed 5 hours ago

agregue imagenes al código, pero no funcionó  
sc4R0 committed 2 hours ago

agrego de imágenes al archivo correctamente  
sc4R0 committed 2 hours ago

Cambio para activar la función de Pull request desde la rama staging ...  
sc4R0 committed 15 minutes ago

Cambios para activar la función de Pull request desde la rama staging...  
sc4R0 committed 12 minutes ago

Showing 4 changed files with 19 additions and 2 deletions.

Split Unified

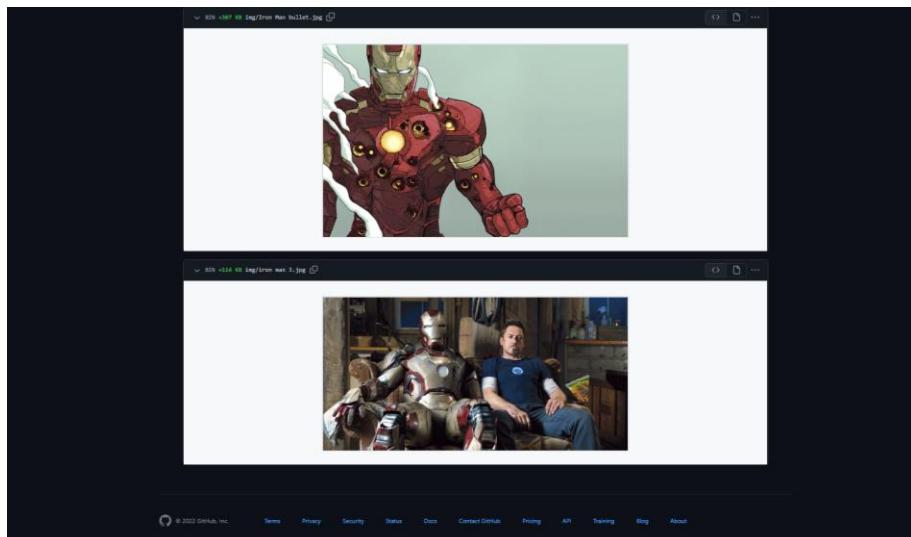
1\_fuepiro.html

T	U	O
13	13	cabecera creada en el master papaus -<h1>
14	14	cambio hecho desde GIT Hub.chr/
15	15	<span>
16	*	cambio hecho desde donde la creéste SSH.
17	*	Cambio hecho después de subir la conexión SSH.chr/
18	*	</span>
19	*	</div>
20	*	Get id="contenedores"
21	*	<div>
22	*	<div>
23	*	<div> personal
24	*	<div> activar un pull Request(chr/)
25	*	<div> personal
26	*	Cambios para activar un Pull Request(chr/)
27	*	</div>
28	*	</div>
29	*	</div>
30	*	</div>
31	*	Se agregó este mensaje en el cuarto commit después del reset(chr/)
32	*	
33	*	</div>
34	*	</html>

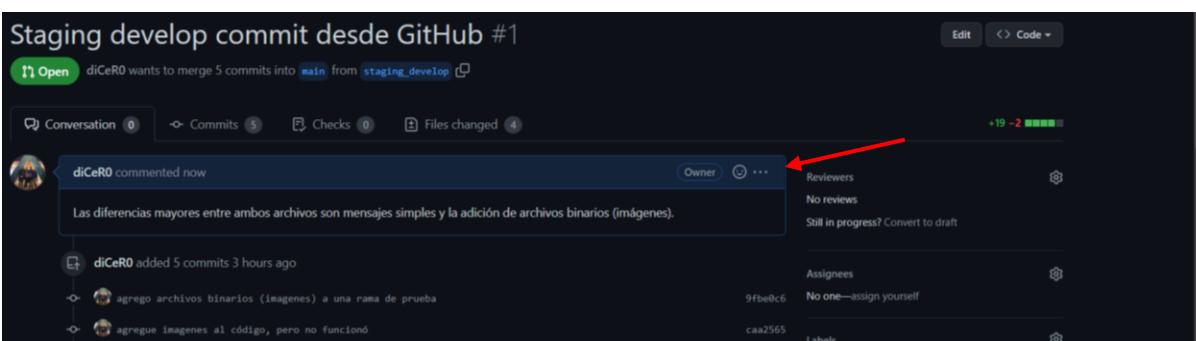
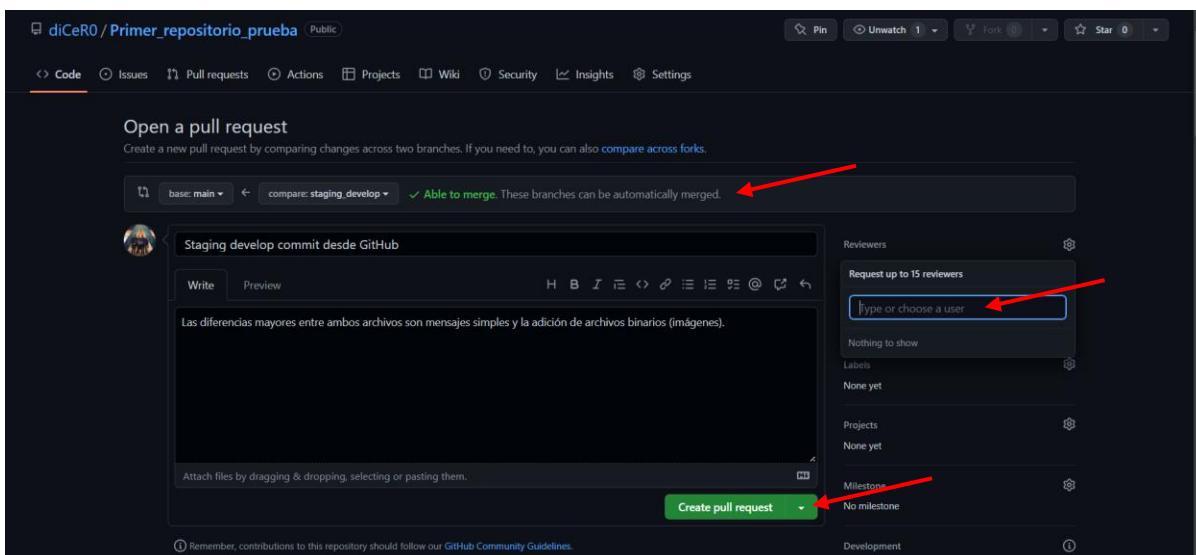
css/styles.css

T	U	O
10	10	text-decoration: underline;
11	11	color: tomato;
12	12	}
13	*	&#160;
14	*	background-color: rgba(0, 0, 0, 0.5);
15	*	color: white;
16	*	text-align: end;
17	*	padding-right: 45px;
18	*	font-size: 40px;
19	*	
20	*	&#160;
21	*	color: gray;
22	*	text-align: end;
23	*	
24	*	background-image: url(/img/Iron Man/bullet.jpg);
25	*	background-repeat: no-repeat;
26	*	

- *Por último, hasta abajo se mostrarán los archivos binarios que se añadieron al comparar ambas versiones.*

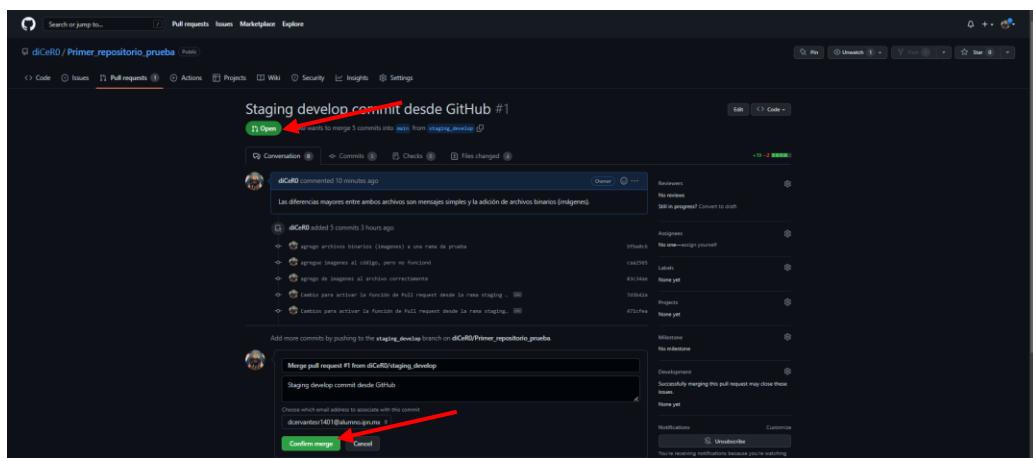
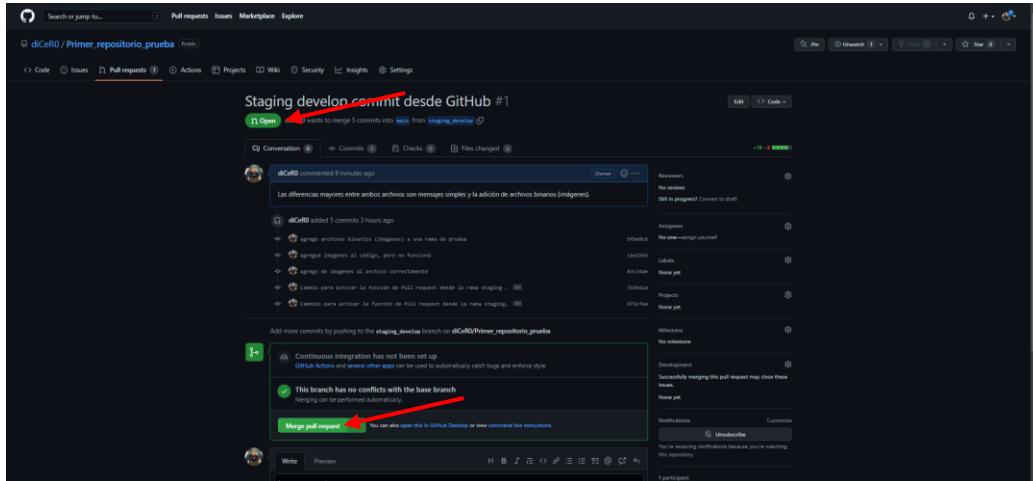


- Antes de mandar el **Pull request** se puede añadir **Reviewers**, pero esto solo se puede asignar a los usuarios que previamente se les había dado acceso al repositorio como se mostrará en una sección posterior.
  - Ahora se dará clic en el botón de **Create pull request** y lo que pasará es que se le estará enviando a todos los colaboradores del repositorio el mensaje que le haya añadido al **Pull Request** cuando se comparó las dos ramas, esto para que todos los colaboradores puedan denotar errores o aciertos entre ambas, añadiendo un mensaje cada que lo hagan y también les pedirá a los **reviewers** que aprueben o desaprueben que se realice el **merge** entre las dos ramas.
  - Para que este proceso pueda avanzar, todos los colaboradores deben haber indicado si la fusión de ramas se debe hacer o no y en caso de que no se haya aprobado el **merge**, se abrirá un **chat** para indicar los cambios pertinentes para que se apruebe.

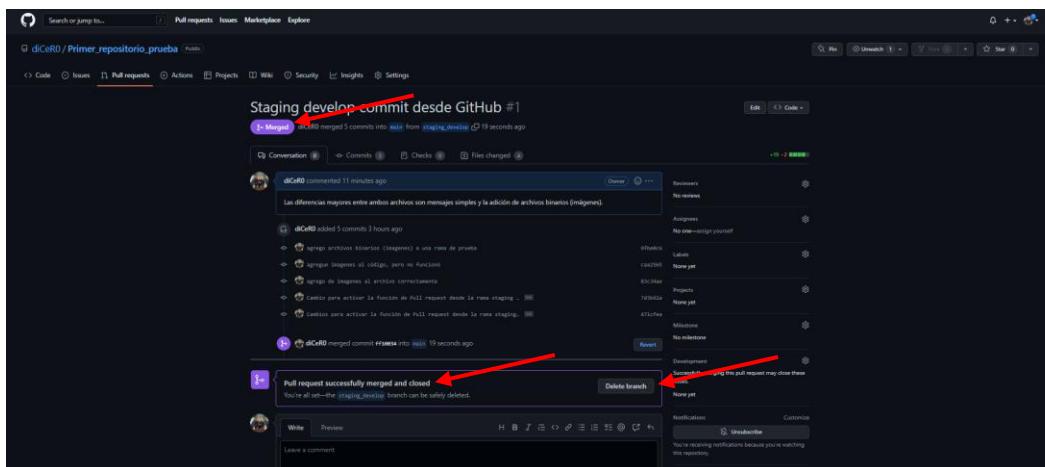


- Finalmente se dará clic en el botón de **Merge Pull Request**, a partir de este punto depende totalmente de los **Reviewers** que yo mismo haya declarado si se pueda ejecutar el **merge** entre las dos ramas o no.
  - En un inicio el **Pull Request** muestra un ícono de color verde que dice **Open**, pero cuando todos los colaboradores del repositorio den clic en el botón de **Confirm merge**, el ícono antes mencionado cambiará a ser de color púrpura y dirá **Merged**, indicando que el **Pull Request** se ha concluido.

- Ya habiendo realizado el merge puedo borrar la branch, ya que estas usualmente son para resolver errores.



- Ya que haya sido aprobado el merge, tendré la opción de conservar la rama asignada a la opción de compare: o de eliminarla, ya que varias veces las ramas a las que se les aplica un Pull Request son las de hotfix o bugfixing (de debuggig). Si quiero eliminar la rama simplemente debo dar clic en el botón que dice Delete branch.



- En este momento como borré la rama al terminar de ejecutar el **Pull request**, lo que pasará es que ya no podré acceder a ella desde GitHub ni Git Bash.

The screenshot shows a GitHub repository page for 'Primer\_repository\_prueba'. The 'Code' tab is selected. In the 'Default branch' section, there is one branch named 'main' with a green status bar indicating it was updated 6 minutes ago by 'diCeR0'. In the 'Your branches' section, there is one branch named 'prueba\_binaria' with a green status bar indicating it was updated 3 hours ago by 'diCeR0'. In the 'Active branches' section, there is one branch named 'prueba\_binaria' with a green status bar indicating it was updated 3 hours ago by 'diCeR0'. At the bottom of the page, there are links for Terms, Privacy, Security, Status, Docs, Contact GitHub, Pricing, API, Training, Blog, and About.

- Recordemos que no es lo mismo las ramas que existan en el **Repositorio local** de Git Bash a las que existan en el **Repositorio remoto** de GitHub, aunque siga apareciendo la rama en el **Repositorio local**, eso no indica que la rama no haya sido borrada del **Repositorio remoto**.

```

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ git branch
  hotfix
* main
  master
  nueva_rama
  prueba_binaria
  staging_develop

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ git pull origin staging_develop
fatal: couldn't find remote ref staging_develop

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ git branch
  hotfix
* main
  master
  nueva_rama
  prueba_binaria
  staging_develop

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ git checkout main
Already on 'main'

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ git pull origin main
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), 645 bytes | 80.00 KiB/s, done.
From github.com:diCeR0/Primer_repository_prueba
 * [branch]      main      -> FETCH_HEAD
   ca01a0a..ff10034 main      -> origin/main
Updating ca01a0a..ff10034
Fast-forward
 2_Ejemplo2.html      |    6 +-----
 css/styles.css       |   15 ++++++-----+
 img/Iron Man bullet.jpg | Bin 0 -> 314067 bytes
 img/iron man 3.jpg    | Bin 0 -> 116791 bytes
 4 files changed, 19 insertions(+), 2 deletions(-)
 create mode 100644 img/Iron Man bullet.jpg
 create mode 100644 img/iron man 3.jpg

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepository (main)
$ 
```

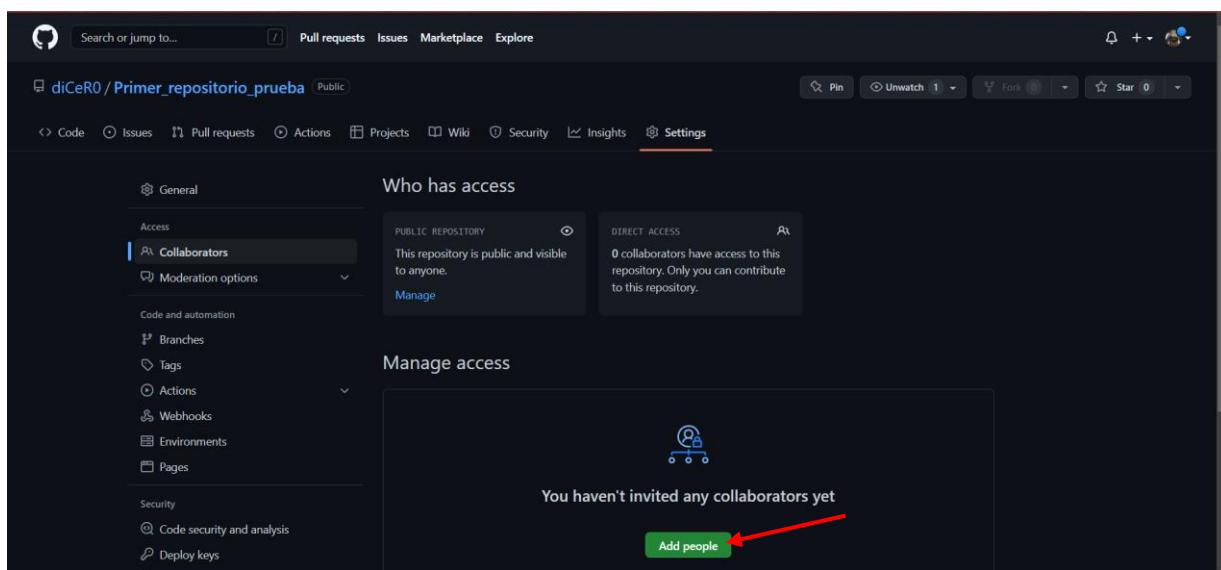
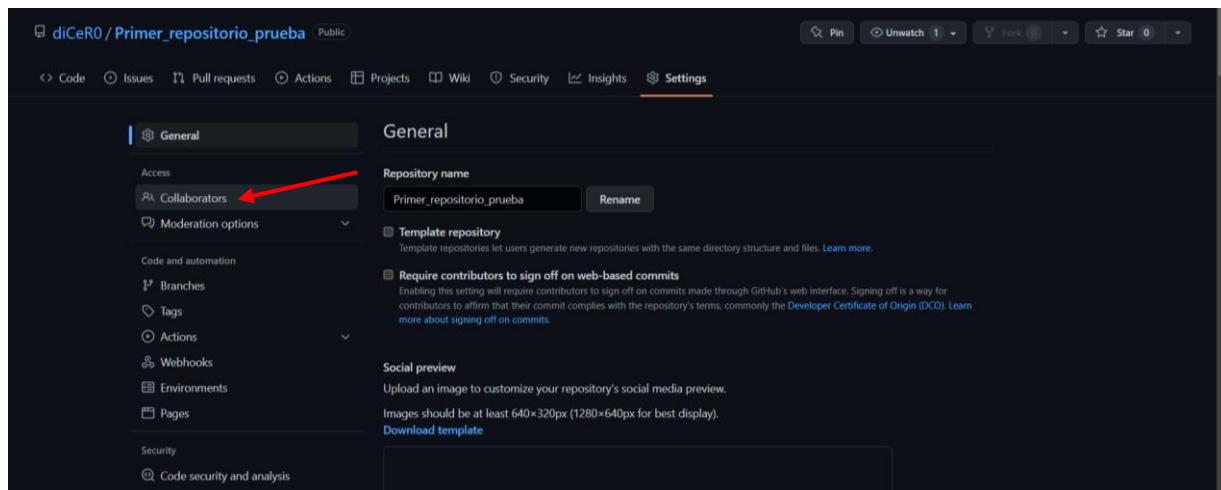
Extraer contenido de un repositorio remoto a uno local y **Colaborar** en él:

*Si es que un nuevo integrante del repositorio solo quiere traer el código de un **Repositorio remoto** hacia un **Repositorio local** de su ordenador y ya, solo debe seguir el comando 26 descrito a continuación, seguido del comando 20 que es el de **git pull**.*

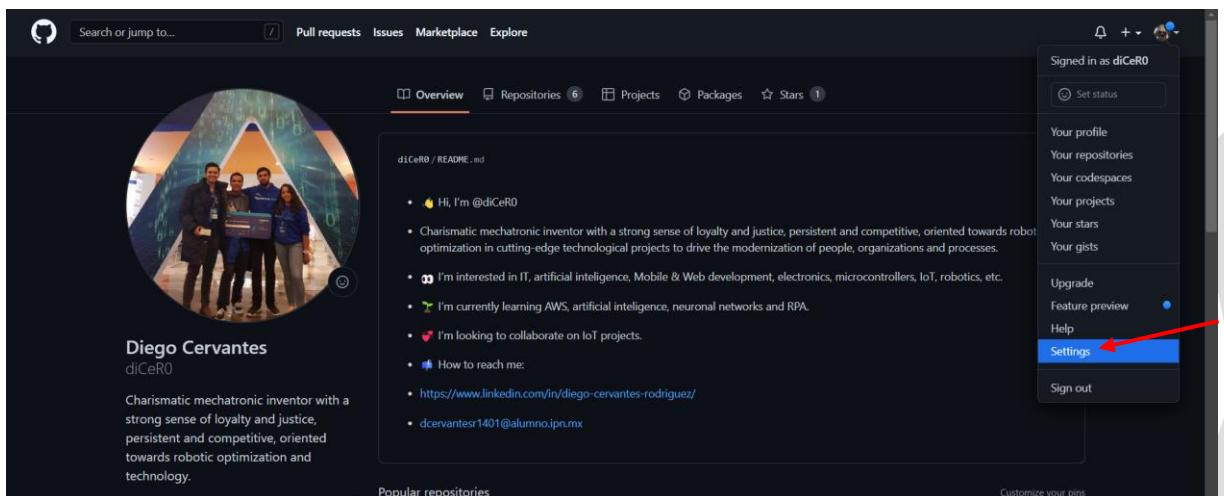
*Sin embargo, si el integrante quiere mandar cambios al **Repositorio remoto** de GitHub, primero que nada, el dueño del repositorio le debe dar acceso por medio de la siguiente opción: **Repositorios** → **Clic en el repositorio al que queremos dar acceso** → **Settings del repositorio** → **Collaborators** → **Add people** → **Añadir cuenta o correo de GitHub**.*

The figure consists of three vertically stacked screenshots of the GitHub web interface, each with a red arrow pointing to a specific element.

- Screenshot 1:** Shows the user profile of "diCeR0". The "Projects" tab is highlighted with a red arrow. The page displays a README.md file with a bio and interests.
- Screenshot 2:** Shows a search results page for repositories. A repository named "Primer\_repository\_prueba" by "diCeR0" is highlighted with a red arrow. It shows a thumbnail image of four people, a brief description, and commit history.
- Screenshot 3:** Shows the detailed view of the repository "Primer\_repository\_prueba". The "Settings" tab is highlighted with a red arrow. The page includes a code editor showing commits, a README.md file with content, and sections for About, Releases, and Packages.



Además, para que podamos añadir una persona a un repositorio su correo debe ser visible, para ello la otra persona debe elegir la siguiente opción: *Settings de GitHub → Emails → Checar que sea visible dando clic en el radio button que dice Recieve all emails...*



The screenshot shows the GitHub Public profile page for 'Diego Cervantes'. On the left, a sidebar menu lists various account management sections. The 'Emails' section is highlighted with a red arrow. The main content area displays the 'Public profile' settings, including fields for 'Name' (set to 'Diego Cervantes'), 'Public email' (set to 'dcervantesr1401@alumno.ipn.mx'), and a 'Bio' section describing 'Charismatic mechatronic inventor with a strong sense of loyalty and justice, persistent and competitive, oriented towards robotic optimization and technology.'

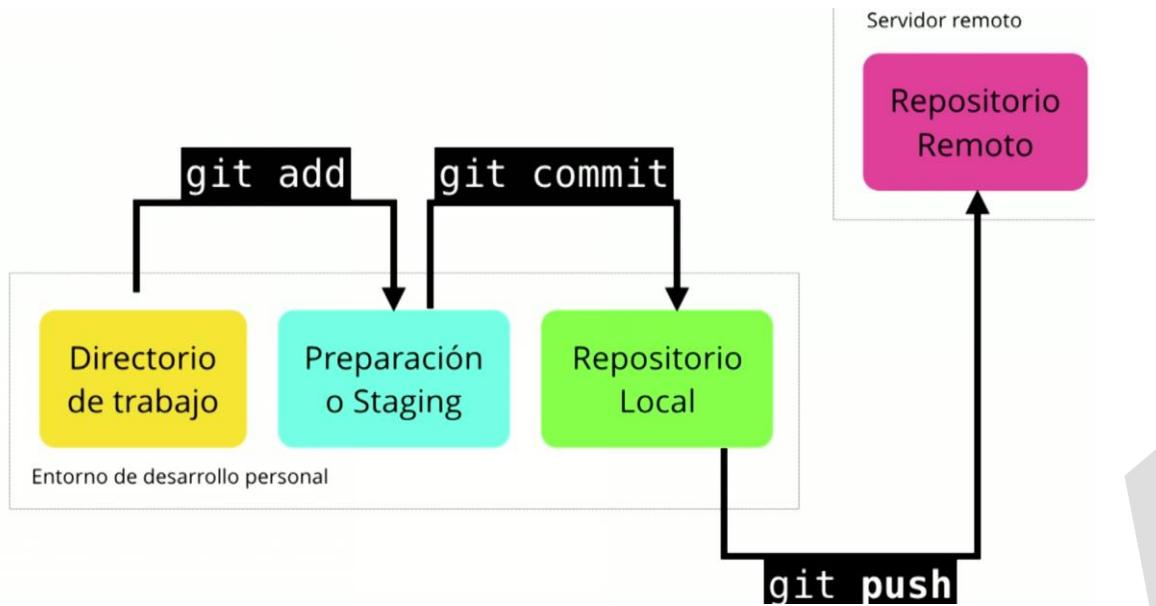
The screenshot shows the GitHub 'Emails' settings page for 'Diego Cervantes'. The sidebar menu highlights the 'Emails' section. The main content area shows two email addresses: 'dcervantesr1401@alumno.ipn.mx' (Primary) and 'diego-rko@live.com.mx' (Backup). Below this, there are sections for 'Primary email address' and 'Backup email address', both set to their respective addresses with a 'Save' button. At the bottom, there is a 'Keep my email addresses private' checkbox and a note about web-based Git operations. The 'Email preferences' section at the very bottom contains two radio button options: 'Receive all emails, except those I unsubscribe from.' (selected) and 'Only receive account related emails, and those I subscribe to.' A red arrow points to the first option.

Una vez que se haya añadido el usuario al repositorio, le llegará un correo con un link para que pueda clonar el repositorio, donde podrá editar completamente el **Repositorio remoto** de GitHub por medio de los comandos `git pull origin main` y `git push origin main`.

26. **git clone url\_http\_o\_ssh\_del\_repositorio\_remoto**: Comando inicial que reemplaza al comando `git init` para que en vez de que se cree un **Repositorio local vacío** en una carpeta .git invisible, se cree una copia de un **Repositorio remoto** de GitHub dentro de nuestra computadora, trayendo con ella una copia de toda la línea de tiempo (versiones) de su rama principal `main`.

- a. La **URL del repositorio remoto** puede ser alguna de las siguientes:
  - i. **url\_https\_extraída\_de\_github**: URL HTTPS de **seguridad moderada** extraída como se muestra en el [comando 18](#).
  - ii. **url\_ssh\_extraída\_de\_github**: URL SSH de **seguridad encriptada** extraída como se muestra en el [comando 23](#).
  - iii. La tercera opción es usar la URL obtenida de un correo recibido cuando el dueño del repositorio dentro de GitHub me invite a colaborar usando la opción de: Repositories → Seleccionar el nombre del **Repositorio remoto** del que queremos crear una copia → Settings del repositorio → Collaborators → Add people → Poner el nombre de usuario o correo de la cuenta que el dueño del repositorio quiere añadir como colaborador.

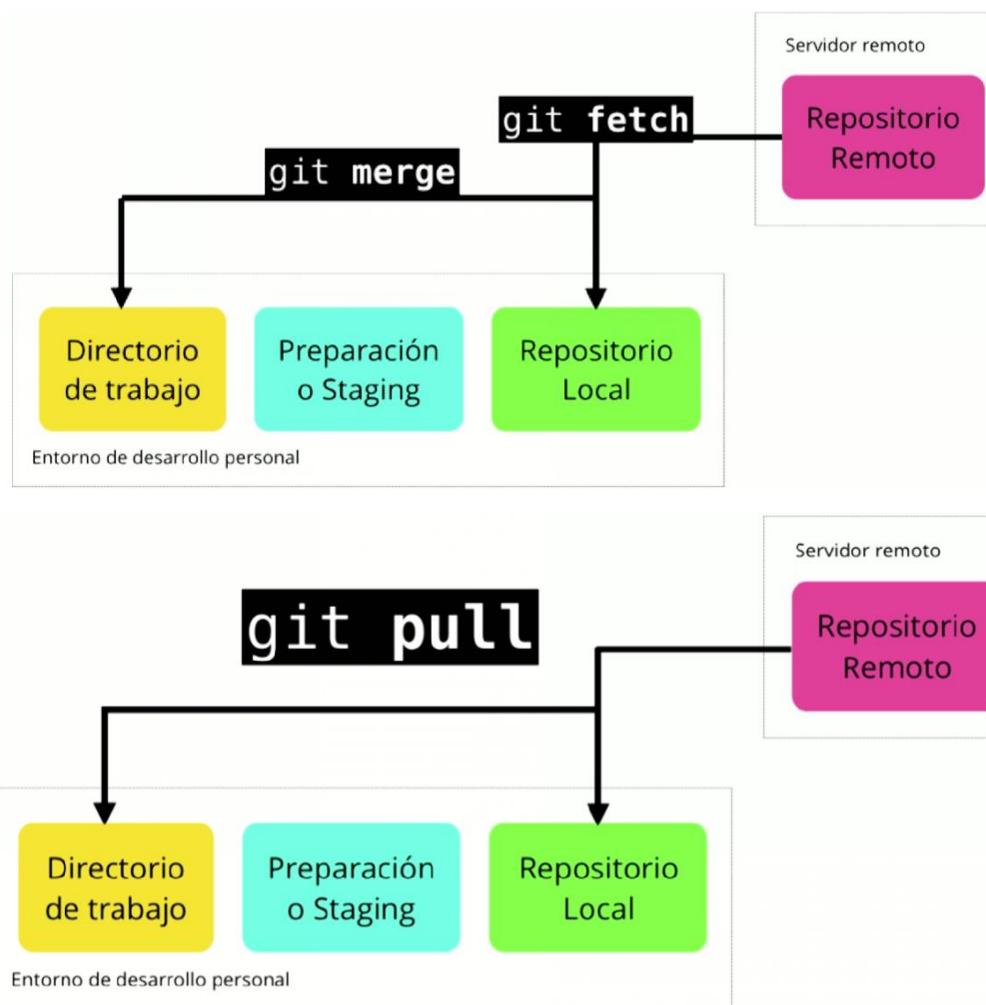
Ya que se haya traído el contenido del repositorio remoto, se deben repetir los [comandos 4, 5 y 6](#) para poder mandar los cambios al repositorio local, pero una vez que se hayan mandado los cambios al repositorio local, se debe usar el comando `git push origin main` para enviarlos al repositorio remoto.



Recordemos que GitHub además de realizar el manejo de versiones sirve para **trabajar colaborativamente de forma remota en un mismo proyecto**, por lo tanto, cuando se trabaja con repositorios remotos, se intuye que no somos los únicos trabajando en él, por lo que **SIEMPRE** antes de mandar un nuevo cambio al repositorio, debemos jalar los cambios que alguien más haya podido hacer antes, para ello es que sirve el comando `git pull`.

27. **git pull:** Comando que extrae el contenido del **Repositorio remoto** al **Working Directory** de mi **Repositorio local**. Esto se puede hacer de forma separada por medio de los comandos **git fetch** y **git merge**.

- a. **git fetch:** Comando que sirve para traer los cambios que alguien más haya realizado al repositorio remoto hacia el repositorio local, pero esto no significa que estos cambios ya se vean reflejados en los archivos de mi ordenador, osea que se vean dentro del **Working Directory**, por lo que se debe ejecutar un comando adicional llamado **git merge**.
  - i. **git merge:** Comando que sirve para unificar el repositorio local con el contenido del **Working Directory**.



Contribuir con proyectos de código abierto en GitHub (Forks):

*La diferencia más grande de cuando se quiere trabajar en un proyecto de código abierto es que como no estamos asignados como **colaboradores** dentro de la plataforma de GitHub, no podremos ejecutar el comando **git push** para subir cambios del **Repositorio local** al **Repositorio remoto**, crear nuevas ramas ni tags; lo que sí se puede hacer es clonar el proyecto con el comando 26 llamado **git clone url\_http\_o\_ssh\_del\_repositorio\_remoto** para que se tenga una copia exacta del proyecto en nuestro ordenador, subir esa copia a un **Repositorio remoto distinto** dentro de GitHub y después*

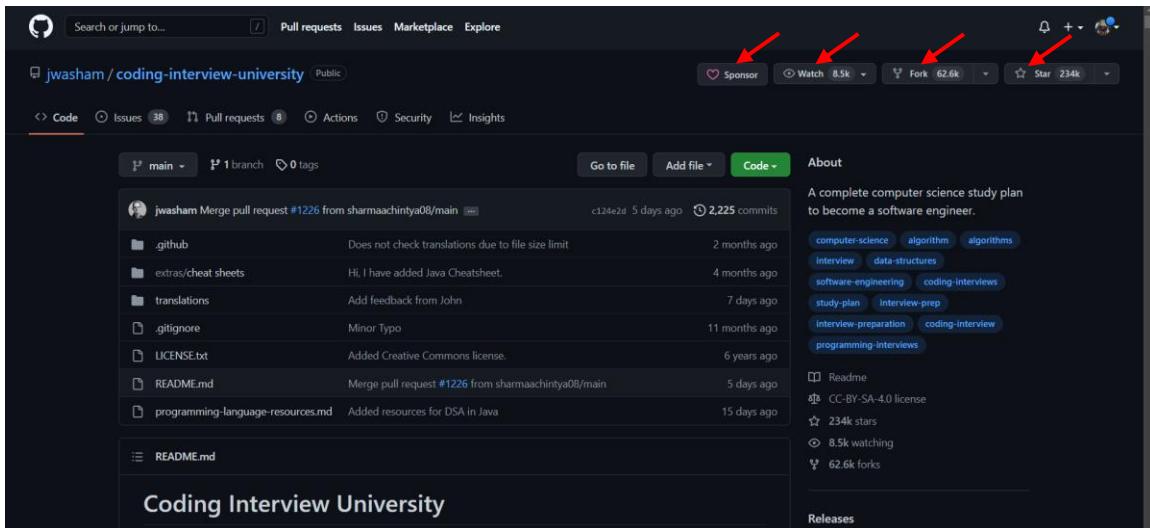
*publicar en la cuenta del dueño del repositorio original el cambio que pretendemos subir para que su propietario lo vea y considere añadirlo por medio de un **Pull Request**.*

*Para lograr esta colaboración en proyectos de código abierto se creó el concepto de **Fork o Bifurcación**.*

Antes que nada, recordemos que GitHub es una red social, por lo cual se puede patrocinar ciertos proyectos (Sponsor), seguirlos (Watch), crear bifurcaciones (Fork) y darles like (Star).

- **Sponsor:** Para que se le pueda depositar dinero al creador del repositorio y así apoyarlo a que siga con su investigación y proyecto open source.
- **Watch:** Cuando siga un proyecto me llegarán notificaciones en el momento en el que se añada un nuevo artículo, contenido u ocurra una conversación entre los contribuyentes del repositorio (otra gente que le dio watch al proyecto).
  - Cuando le dé Watch a un proyecto puedo decidir si quiero que se me notifique solo cuando se cree una publicación que mencione mi nombre de usuario, seguir toda la actividad del repositorio, ignorar los cambios del repositorio o que solamente me informe cuando sucedan ciertas cosas como: Errores, Pull requests, Releases, Discusiones en el chat o alertas de seguridad.
- **Fork o Bifurcación:** Es una característica única de GitHub en la que se crea una copia exacta del estado actual de un **Repositorio remoto público** de GitHub hacia nuestro ordenador, este nuevo **Repositorio local** podrá servir como otro origen, dándonos la posibilidad de con él crear un nuevo **Repositorio remoto** dentro de GitHub que tenga diferencias al código original, incluyendo de igual forma su propia línea de tiempo.
- **Star:** Es igual a darle like a un proyecto dentro de la plataforma de GitHub, esto hará que me lleguen avisos cuando haya cambios en el código o los archivos del proyecto.

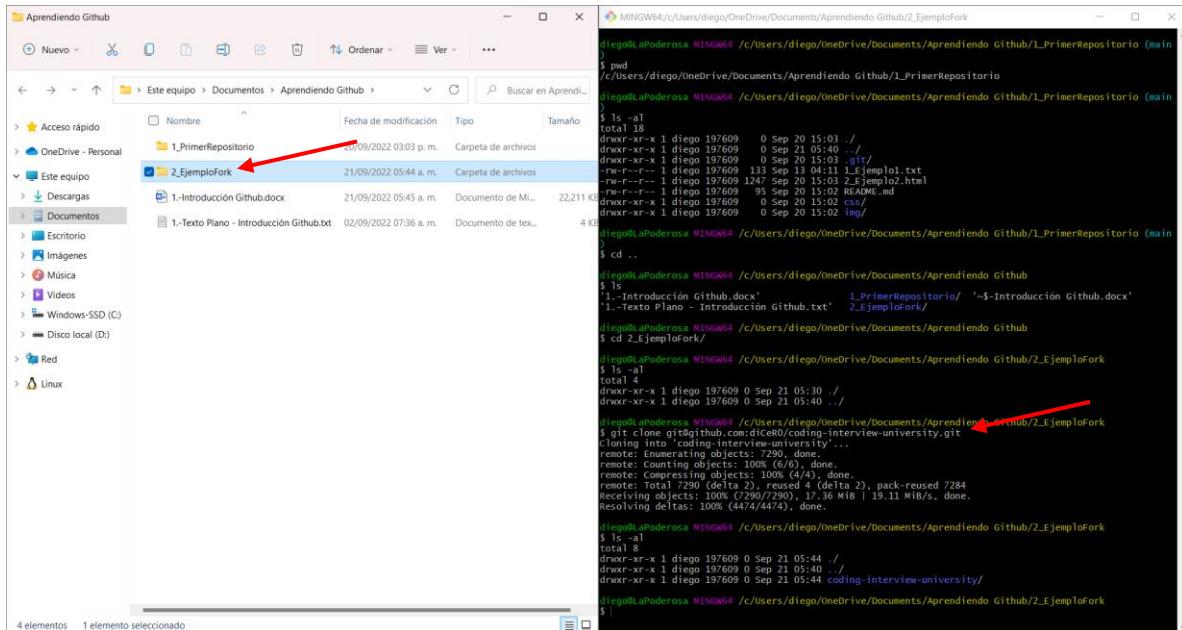
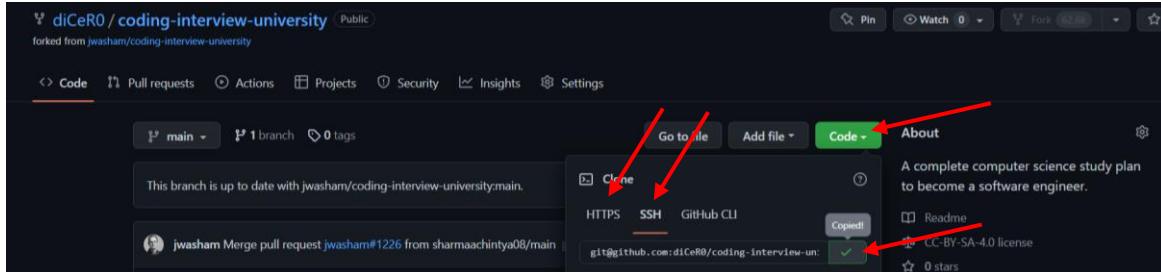
*El siguiente ejemplo de un repositorio llamado **coding-interview-university** sirve para prepararse para entrevistas de trabajo de código: <https://github.com/jwasham/coding-interview-university>*



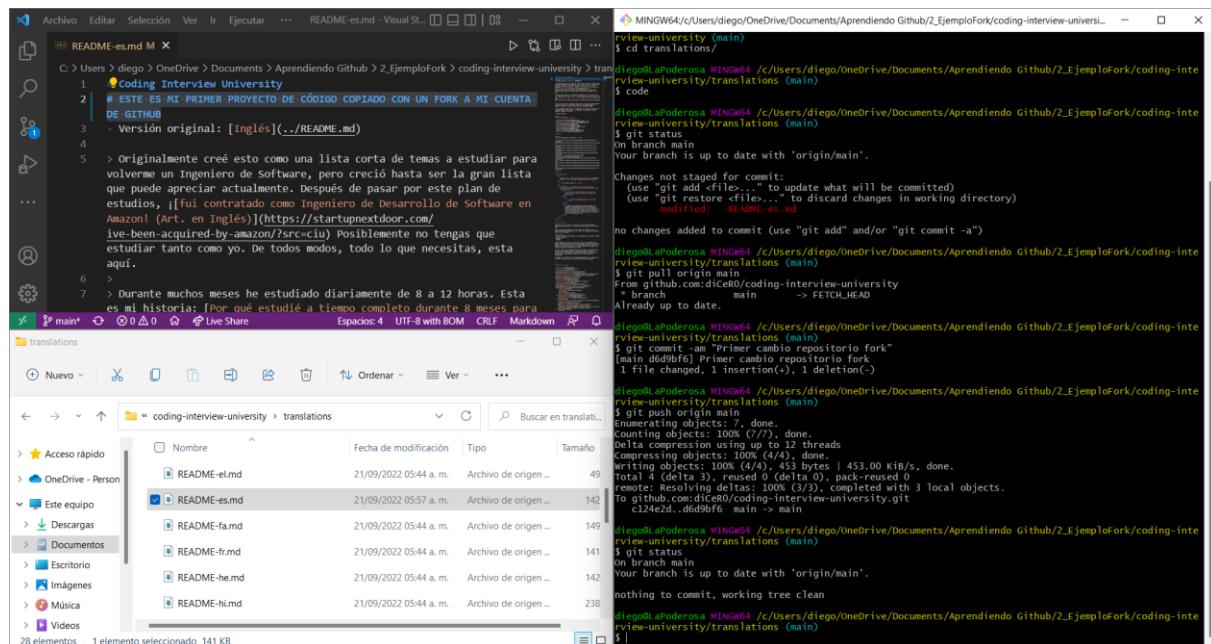
*Para crear una Bifurcación de un proyecto de código abierto se siguen los pasos descritos a continuación: Fork → Se asigna el dueño y título del Fork, su descripción, se indica si solo se va a copiar la rama main o todas → Se da clic en el botón de Create Fork.*

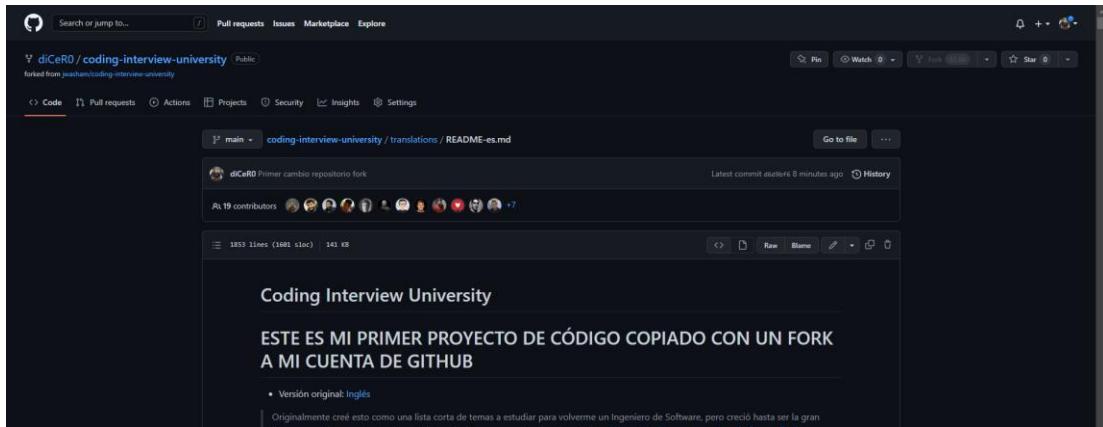
Al crear el Fork lo que logramos es que se cree una copia exactamente igual al estado actual del proyecto de open source pero dentro de nuestra cuenta de GitHub, con los mismos archivos, artículos, línea del tiempo (historial de commits de todos los colaboradores del proyecto), releases, etc. Todo esto perteneciente a la rama main, porque así fue como se lo indiqué cuando cree la Bifurcación.

Antes de jalar el contenido de una bifurcación debo preparar la carpeta donde quiero clonar su contenido dentro de mi ordenador, luego dependiendo de si ya he creado un cifrado SSH dentro de mi cuenta de GitHub, podré usar la URL SSH o si no he creado ese cifrado usaré la URL HTTPS con la opción de: **Repositories → Seleccionar el nombre del Repositorio remoto** que fue creado cuando realizamos el Fork → Dar clic en el botón de Code → **HTTPS o SSH** → **Copiar la URL** → Ejecutar el comando 26, git clone url.



*Antes de jalar el contenido de una bifurcación debo preparar la carpeta donde quiero clonar su contenido dentro de mi ordenador, con este nuevo **Repositorio local** podré mandar cambios hacia el **Repositorio remoto** que fue creado cuando realizamos el Fork.*

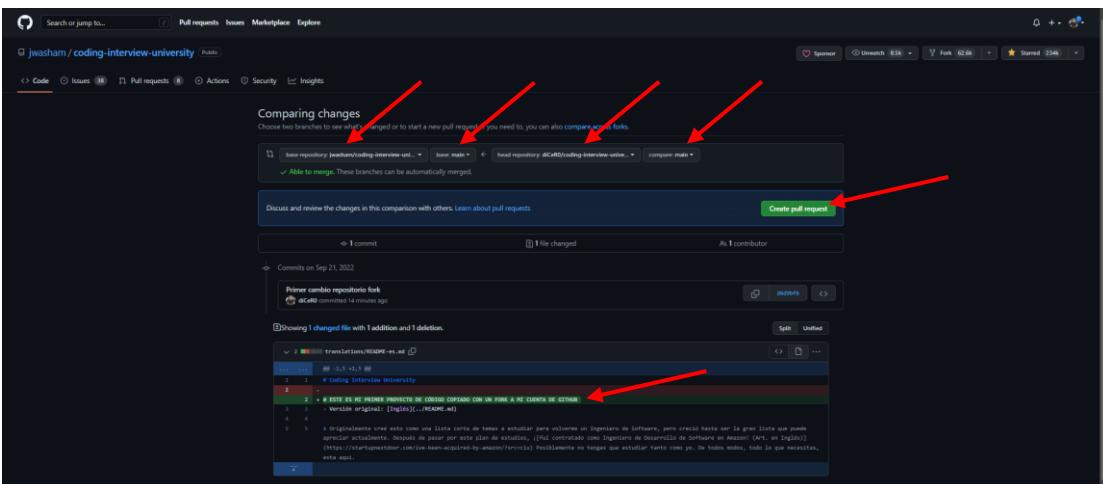
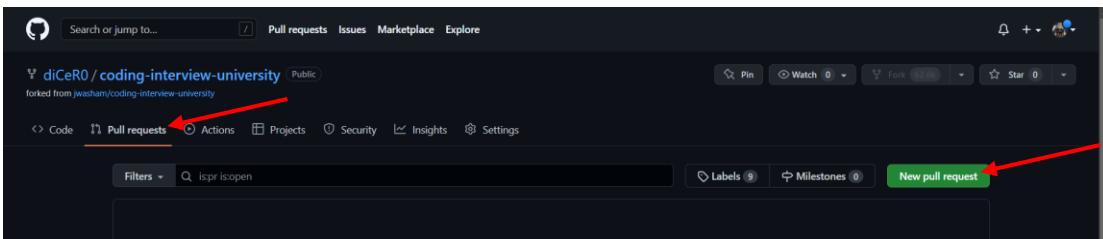




*En este momento es importante denotar la diferencia entonces entre Colaborador y Contribuyente.*

- **Colaborador:** Persona que puede realizar cambios en un repositorio ya que tiene permiso asignado a través de su cuenta de GitHub por el dueño del proyecto.
- **Contribuyente:** Persona que usualmente ayuda a desarrollar un proyecto de código abierto (open source) a través de Forks.

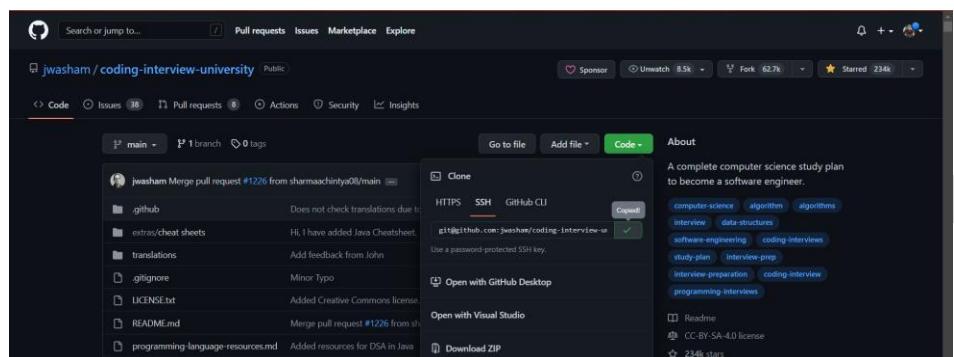
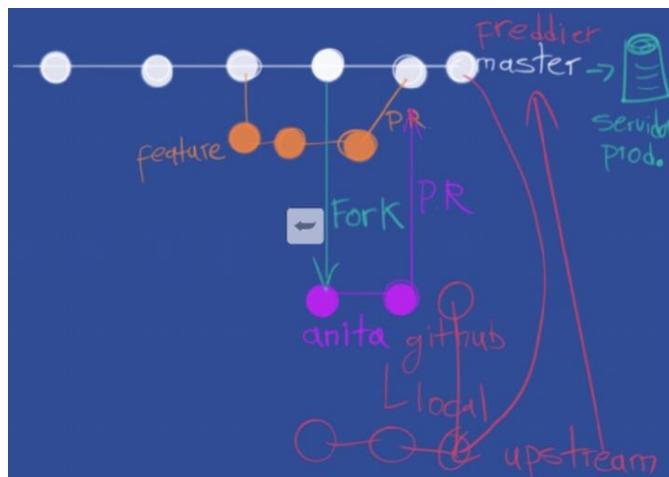
*En este punto si quiero ejecutar un Pull Request primero debo asegurarme que el cambio que quiero mandar es un buen cambio, ya que estos proyectos de código abierto son el trabajo de una comunidad entera de personas y debe tomarse seriamente, para ello debo introducirme a la opción de: Pull requests → New pull request → Seleccionar la rama base: (rama a donde se unirán los cambios de la otra rama a comparar) y la rama compare: (que se unirá a la rama base:), ambas indicando el usuario al que pertenecen → Realmente checar que los cambios son beneficiosos para el proyecto Open Source → Clic en el botón de Create pull request .*



Todas las interacciones hechas con un proyecto de open source como crear un Fork, darle like con Star, ejecutar un Pull Request con todo y los cambios que se quieren hacer, etc. serán visualizadas por el dueño del repositorio por medio de las notificaciones que le llegan a su cuenta de GitHub.

Aquí sucede algo interesante ya que si quiero conectar mi **Repositorio local** no solamente con la copia del **Repositorio remoto del Fork**, sino con el **Repositorio remoto original del proyecto open source** para que cuando el dueño del repositorio original cree cambios, estos cambios sean actualizados igual dentro de mi ordenador, debo establecer una conexión con un nuevo repositorio de GitHub, esto al ejecutar el comando 18: `git remote add nombre_repositorio url_https_o_ssh_extraida_de_github` para de esa forma crear una conexión con un nombre de repositorio distinto a `origin`, que normalmente se llama **upstream** (es importante mencionar que este nuevo repositorio llamado **upstream** debe ser conectado a una URL con cifrado HTTPS o SSH extraída de la cuenta del repositorio original en GitHub, no de la bifurcación creada en mi propia cuenta) y ahora se ejecutarán los comandos 20 y 21 de la siguiente manera para trabajar con los dos repositorios remotos:

- **Repositorio remoto original:** Con el nombre del nuevo repositorio se actualizan los cambios del **Repositorio remoto original** hacia el **Repositorio local** de mi ordenador con la instrucción `git pull upstream nombre_rama`.
- **Repositorio remoto del Fork:** Una vez extraídos los cambios del repositorio original **upstream** por medio del comando `git pull`, se deben mandar los cambios a mi **Repositorio local** con un commit y luego mandarlos al **Repositorio remoto del Fork** con el comando `git push origin nombre_rama`.



```

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/2_EjemploFork/coding-interview-university
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/2_EjemploFork/coding-interview-university (main)
$ git remote
origin

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/2_EjemploFork/coding-interview-university (main)
$ git remote -v
origin  git@github.com:diCeRO/coding-interview-university.git (fetch)
origin  git@github.com:diCeRO/coding-interview-university.git (push)

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/2_EjemploFork/coding-interview-university (main)
$ git remote add upstream git@github.com:jwasham/coding-interview-university.git

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/2_EjemploFork/coding-interview-university (main)
$ git remote
origin
origin
upstream

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/2_EjemploFork/coding-interview-university (main)
$ git remote -v
origin  git@github.com:diCeRO/coding-interview-university.git (fetch)
origin  git@github.com:diCeRO/coding-interview-university.git (push)
upstream    git@github.com:jwasham/coding-interview-university.git (fetch)
upstream    git@github.com:jwasham/coding-interview-university.git (push)

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/2_EjemploFork/coding-interview-university (main)
$ git pull upstream main
From github.com:jwasham/coding-interview-university
 * branch      main            -> FETCH_HEAD
 * [new branch] main            -> upstream/main
Already up to date.

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/2_EjemploFork/coding-interview-university (main)
$ git push origin main
Everything up-to-date

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/2_EjemploFork/coding-interview-university (main)
$ |

```

Conectar un dominio de algún servidor con un proyecto GitHub:

*Para poder conectar el contenido de un proyecto de GitHub como lo son imágenes, archivos, código, etc. con un dominio perteneciente a algún host de internet como lo es Go Daddy, Hostinger, Hostgator, etc. Debemos acceder a la carpeta de su proyecto por medio de la consola Git Bash y luego ejecutar el comando [26](#): `git clone url_http_o_ssh`, para con ello crear una nueva carpeta que tenga todo el contenido del **Repositorio remoto** guardado en GitHub.*

```

Terminal Shell Edit View Window Help
reddier -- cristalab@freddieserver1: /var/www/freddier.com/html -- ssh cristalab@freddier.com -- 115x30
cristalab@freddieserver1:~$ cd /var/www/freddier.com/html/
cristalab@freddieserver1:/var/www/freddier.com/html$ ls -al
total 16
drwxr-xr-x 2 cristalab cristalab 4096 Apr 18 18:48 .
drwxr-xr-x 3 root      root     4096 Apr 18 18:43 ..
-rw-r-xr-x 1 cristalab cristalab   43 Feb 18 06:10 index.html
-rw-r--r-- 1 root      root     14 Apr 18 18:39 test.html
cristalab@freddieserver1:/var/www/freddier.com/html$ cat index.html
<strong>Solo un test</strong> del sistema.
cristalab@freddieserver1:/var/www/freddier.com/html$ cat test.html
Es una prueba
cristalab@freddieserver1:/var/www/freddier.com/html$ git clone https://github.com/freddier/hyperblog.git
Cloning into 'hyperblog'...
remote: Enumerating objects: 121, done.
remote: Counting objects: 100% (121/121), done.
remote: Compressing objects: 100% (67/67), done.
remote: Total 121 (delta 47), reused 107 (delta 38), pack-reused 0
Receiving objects: 100% (121/121), 306.67 KiB | 18.04 MiB/s, done.
Resolving deltas: 100% (47/47), done.
cristalab@freddieserver1:/var/www/freddier.com/html$ ls -al
total 20
drwxr-xr-x 3 cristalab cristalab 4096 Apr 18 18:53 .
drwxr-xr-x 3 root      root     4096 Apr 18 18:43 ..
drwxrwxr-x 5 cristalab cristalab 4096 Apr 18 18:53 hyperblog
-rw-r-xr-x 1 cristalab cristalab   43 Feb 18 06:10 index.html
-rw-r--r-- 1 root      root     14 Apr 18 18:39 test.html
cristalab@freddieserver1:/var/www/freddier.com/html$ 

```

The top image shows a GitHub repository page for 'freddier/hyperblog'. It displays 34 commits, 4 branches, 1 release, and 3 contributors. The commit history includes several merges and initial commits. The README.md file contains the text 'hyperblog' and 'Un blog increíble para el curso de Git y Github de Platzi'. The bottom image shows a browser window displaying a blog post with the title 'Este es el título atractivo e interesante del post' and some introductory text.

*Cuando cree cambios en el repositorio, para que estos puedan ser reflejados en el dominio del sitio web a donde fueron subidos se deben ejecutar los comandos 20 y 21: **git pull nombre\_repositorio nombre\_rama**, a este proceso se le llama hacer un **deploy al servidor de producción**.*

## Manejo de errores/condiciones en Git

Creación del archivo .gitignore:

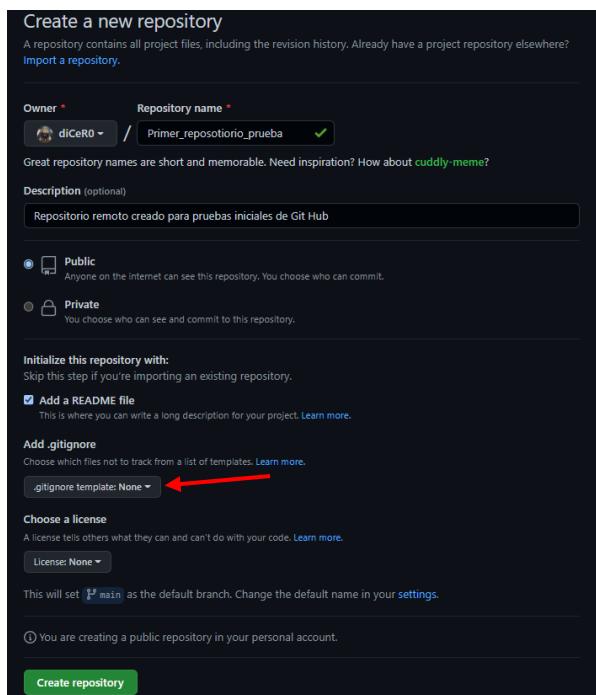
Recordemos que cuando se crea un nuevo **Repositorio remoto vacío** dentro de GitHub para que se vincule con un **Repositorio local** previamente creado se deben indicar los siguientes aspectos del proyecto:

- Una vez que se haya seleccionado la opción de **New repository**, se indican las siguientes características del repositorio, ya que las hayamos definido debemos dar clic en el botón de **Create repository** para finalizar y crear así el nuevo repositorio:
  - Dueño y nombre del repositorio.**
  - Descripción.**
  - Modificadores de acceso:** Donde se declara si el repositorio será público o privado. Esto solo implica que pueda ser visto, no editado.
  - Inicializar el repositorio con un archivo README:** El archivo README.md (mark down) se verá cuando alguna persona entre al repositorio por medio de Git Hub, es un archivo construido casi siempre hecho con HTML que funciona como un **manual de instrucciones del proyecto**, puede incluir links, artículos, etc. Se agrega a la carpeta del repositorio.

- i. **Iniciar el repositorio con un archivo Git Ignore:** No todos los archivos de un proyecto se deben agregar a un repositorio, como aquellos que contengan contraseñas, archivos PHP con accesos a bases de datos, imágenes muy pesadas, archivos binarios indeseables, etc. Para ello sirve el archivo `git ignore`, indicando cuáles archivos serán ignorados cuando se quiera subir cambios al repositorio.
- ii. **Agregar un archivo de licencia al repositorio:** Existen varias licencias incluidas en Git Hub a través de las cuales se puede publicar el código del repositorio, ya sea de código abierto, semiabierto, cerrado, etc.

La parte que nos interesa de esto es **Git Ignore**, el cual es un archivo llamado exclusivamente `.gitignore` que se crea en la misma carpeta del proyecto y literalmente sirve para describir una lista de los archivos que se deben ignorar cuando se suban los cambios al **Repositorio remoto** de GitHub, para ello se sigue la siguiente sintaxis:

- **`*.extensión_archivos`:** Cuando se pone un asterisco, un punto y el nombre de alguna extensión dentro de un archivo `.gitignore` lo que se está indicando es que todos los archivos que tengan la extensión indicada, deberán ser ignorados cuando se suban al repositorio, esto se verá reflejado cuando ejecutemos el comando `git status`.
- **`carpeta1/carpeta2/carpeta3`:** De igual manera se pueden indicar directorios específicos para que todos los archivos de ese directorio no sean subidos al **Repositorio remoto**, para ello debemos tomar en cuenta la carpeta en la que nos encontramos actualmente para indicar una dirección de carpeta correctamente.
- **`carpeta1/carpeta2/carpeta3/*.extensión_archivos`:** También se pueden mezclar las dos opciones anteriores para que solamente los archivos incluidos en un directorio que tengan una extensión específica no sean subidos al repositorio.
- **#Comentarios:** Es importante mencionar que los comentarios dentro de los archivos `.gitignore` se colocan poniendo un signo de número.



```

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git status
On branch 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: 2_Ejemplo2.html
      modified: css/styles.css

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    img/iron man.jpeg
    img/right flight iron man.jpg

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

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ 

```

```

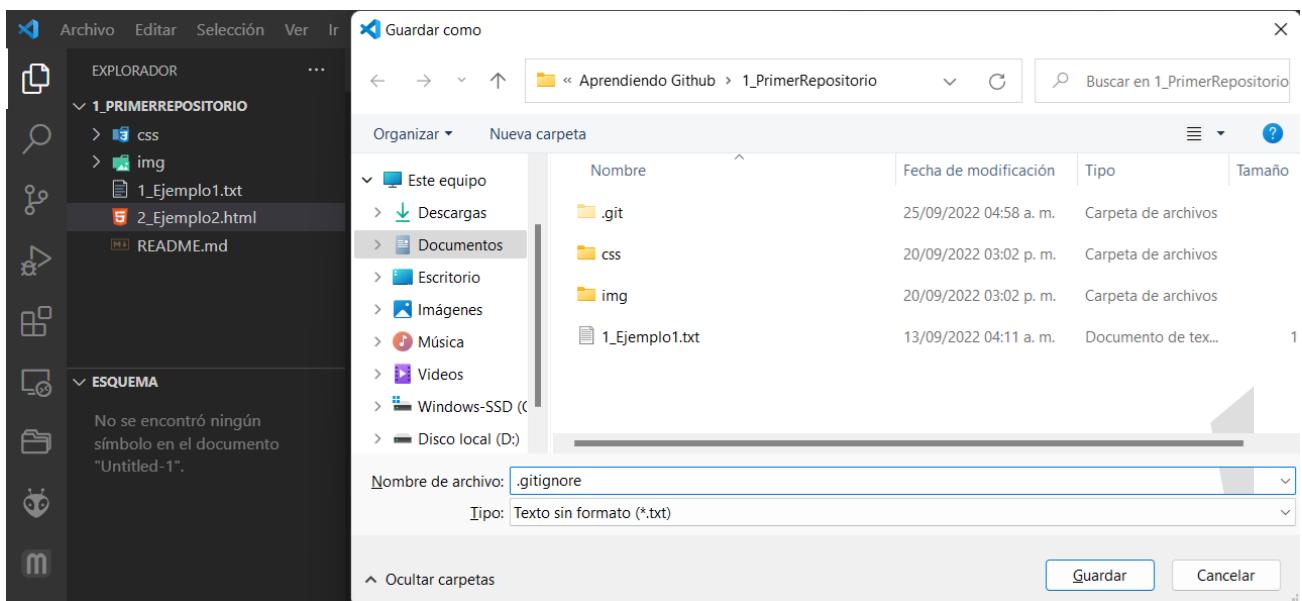
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified: 2_Ejemplo2.html
      modified: css/styles.css

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    img/iron man.jpeg
    img/right flight iron man.jpg

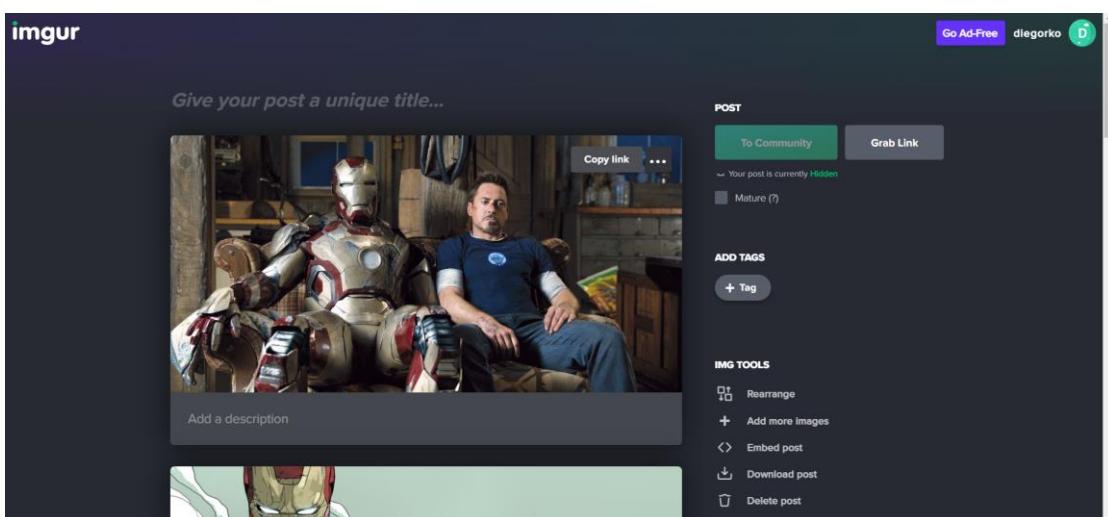
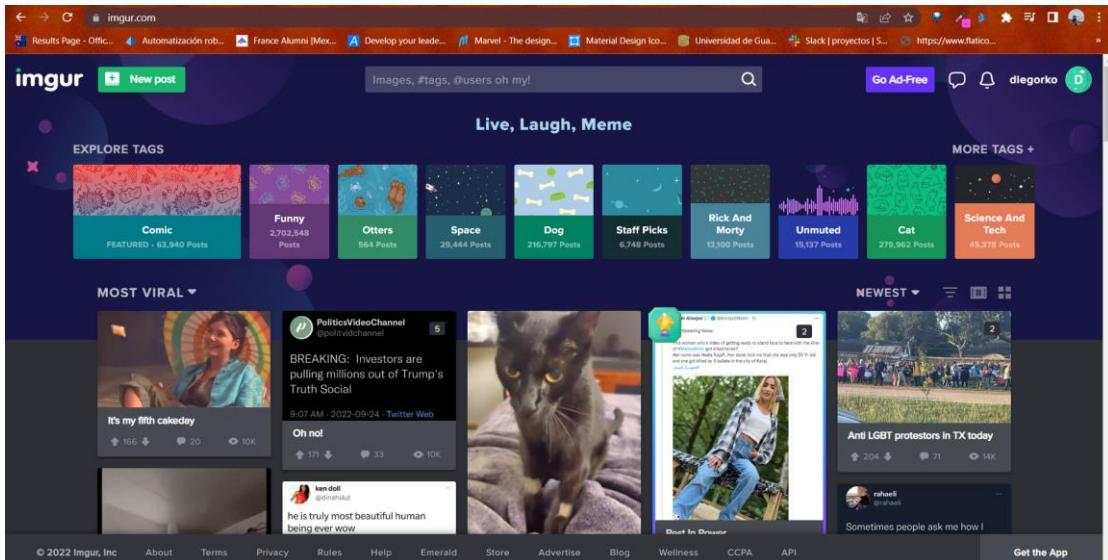
no changes added to commit (use "git add" and/or "git commit -a")

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ 

```

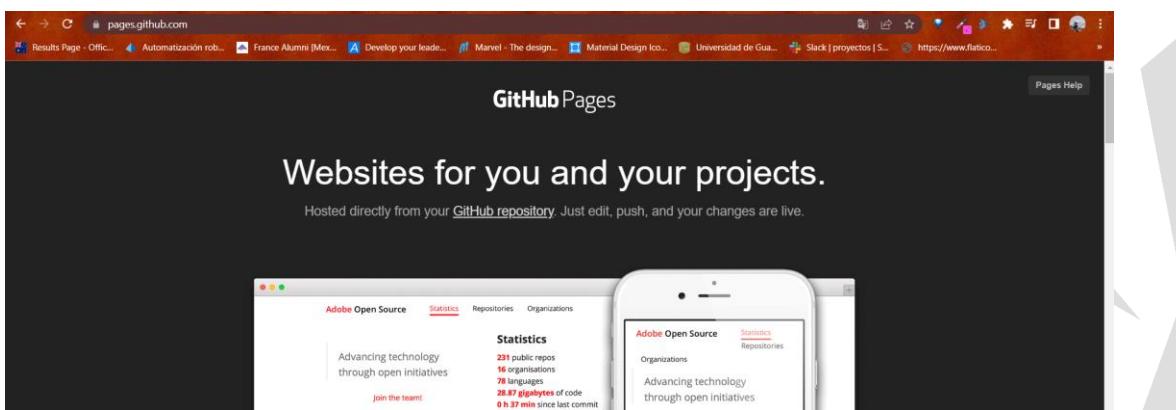


Para evitar subir imágenes a mi repositorio las puedo referenciar con una URL para cada una y que estén subidas en otro lado, la forma de hacer esto es por ejemplo usando la página imgur que sirve para subir imágenes y que así no estén subidos archivos binarios a mi repositorio u host: <https://imgur.com/>



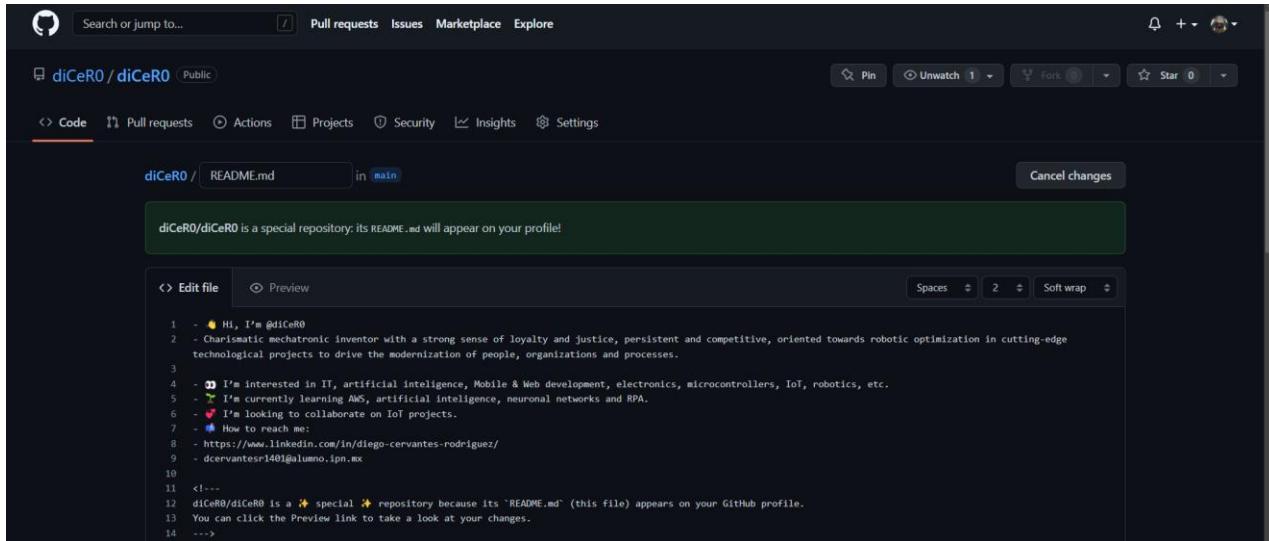
## Uso del hosting gratuito de GitHub (GitHub Pages):

*GitHub tiene un servicio de hosting gratis donde se puede publicar el contenido de nuestros repositorios para que por medio de un subdominio de GitHub se puedan ver de forma online, para ello se debe utilizar el servicio de GitHub Pages: <https://pages.github.com/>*

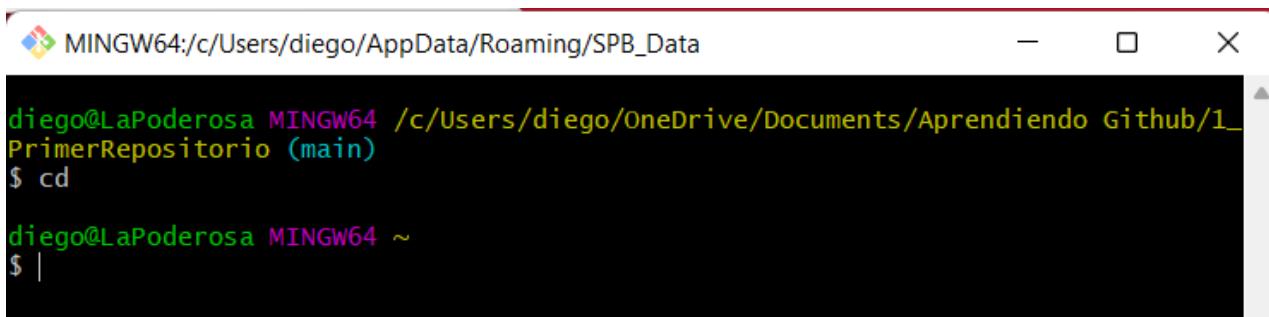


*Los pasos que se deben seguir para vincular el servicio de GitHub con GitHub Pages y posteriormente crear un dominio que muestre el contenido de cada uno de mis repositorios creados es el siguiente:*

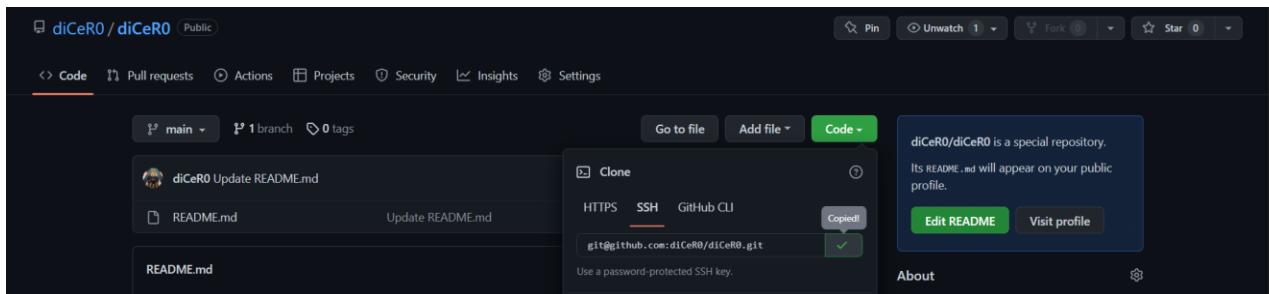
1. *Para poder conectar por primera vez el servicio de GitHub con GitHub Pages, se debe crear un nuevo **Repositorio remoto** vacío que tenga nuestro mismo nombre de usuario (que de hecho puede que ya esté creado porque GitHub lo pide de igual manera para que sea la introducción de mi perfil).*



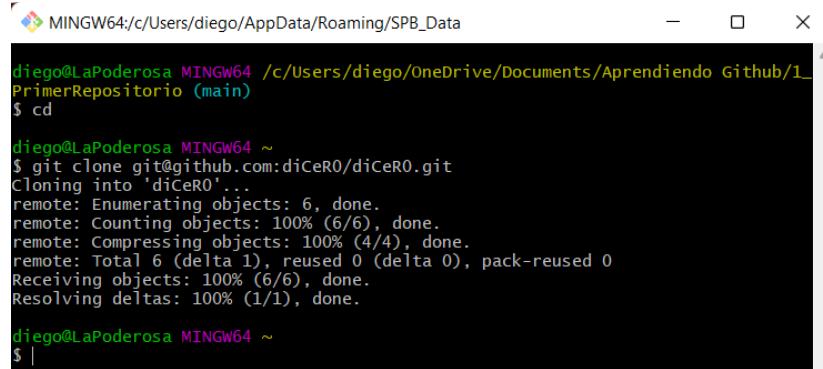
2. *Luego debo dirigirme al directorio home de mi ordenador, para ello debo postrarme en la Git Bash y colocar el comando cd, haciendo que aparezca el directorio ~.*



3. *Ahora debo obtener su URL por medio de la opción: Code → HTTPS o SSH → Copiar.*



4. Dentro del directorio home (~) es donde se debe ejecutar el comando 26: **git clone url\_http\_o\_ssh**, para con ello crear una nueva carpeta que tenga mi nombre de usuario de GitHub.

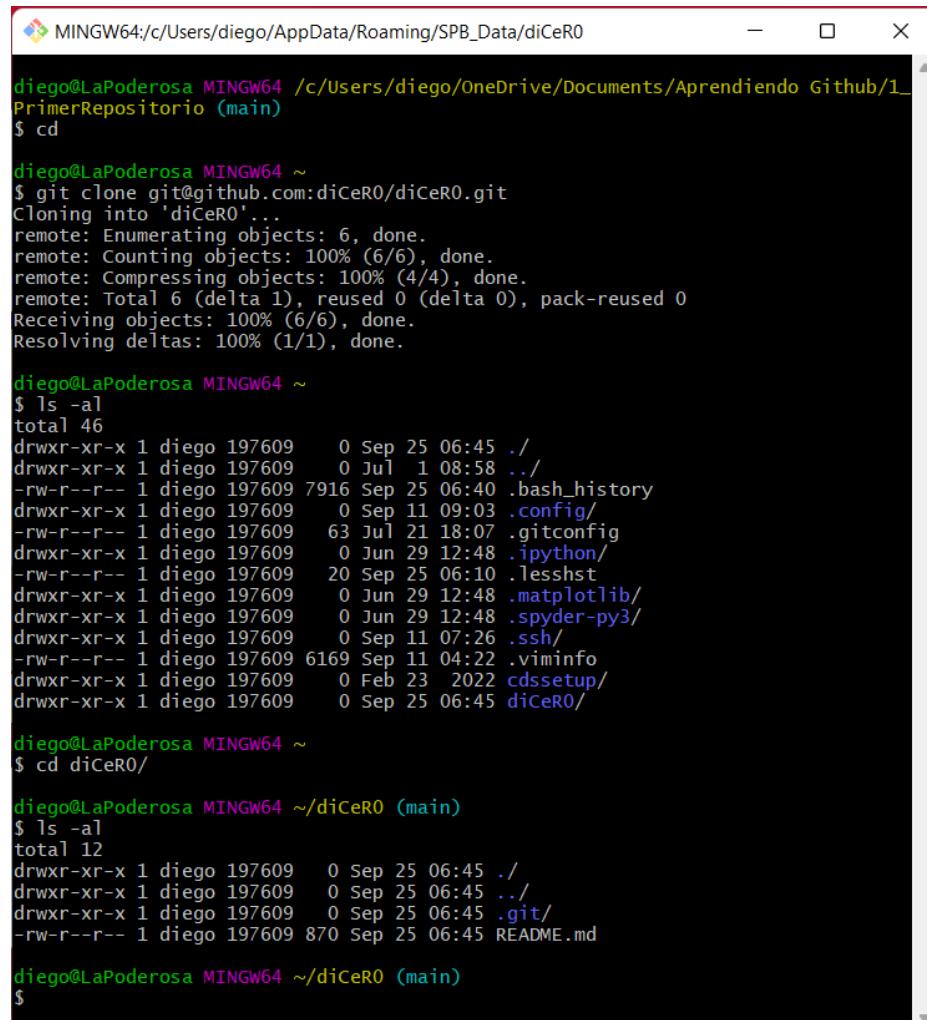


```
MINGW64:/c/Users/diego/AppData/Roaming/SPB_Data
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_
PrimerRepositorio (main)
$ cd

diego@LaPoderosa MINGW64 ~
$ git clone git@github.com:diCeRO/diCeRO.git
Cloning into 'diCeRO'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
Resolving deltas: 100% (1/1), done.

diego@LaPoderosa MINGW64 ~
$ |
```

5. Cuando entremos a la carpeta que tiene el mismo nombre de usuario de GitHub, podremos notar que aparece el nombre de la rama principal y la carpeta .git con el archivo README.md, en esta carpeta debemos crear un archivo llamado **index.html** para que esto sea lo que aparezca cuando ingresemos al dominio.



```
MINGW64:/c/Users/diego/AppData/Roaming/SPB_Data/diCeRO
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_
PrimerRepositorio (main)
$ cd

diego@LaPoderosa MINGW64 ~
$ git clone git@github.com:diCeRO/diCeRO.git
Cloning into 'diCeRO'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
Resolving deltas: 100% (1/1), done.

diego@LaPoderosa MINGW64 ~
$ ls -al
total 46
drwxr-xr-x 1 diego 197609 0 Sep 25 06:45 .
drwxr-xr-x 1 diego 197609 0 Jul 1 08:58 ../
-rw-r--r-- 1 diego 197609 7916 Sep 25 06:40 .bash_history
drwxr-xr-x 1 diego 197609 0 Sep 11 09:03 .config/
-rw-r--r-- 1 diego 197609 63 Jul 21 18:07 .gitconfig
drwxr-xr-x 1 diego 197609 0 Jun 29 12:48 .ipython/
-rw-r--r-- 1 diego 197609 20 Sep 25 06:10 .lesshst
drwxr-xr-x 1 diego 197609 0 Jun 29 12:48 .matplotlib/
drwxr-xr-x 1 diego 197609 0 Jun 29 12:48 .spyder-py3/
drwxr-xr-x 1 diego 197609 0 Sep 11 07:26 .ssh/
-rw-r--r-- 1 diego 197609 6169 Sep 11 04:22 .viminfo
drwxr-xr-x 1 diego 197609 0 Feb 23 2022 cdssetup/
drwxr-xr-x 1 diego 197609 0 Sep 25 06:45 diCeRO/

diego@LaPoderosa MINGW64 ~
$ cd diCeRO/

diego@LaPoderosa MINGW64 ~/diCeRO (main)
$ ls -al
total 12
drwxr-xr-x 1 diego 197609 0 Sep 25 06:45 .
drwxr-xr-x 1 diego 197609 0 Sep 25 06:45 ../
drwxr-xr-x 1 diego 197609 0 Sep 25 06:45 .git/
-rw-r--r-- 1 diego 197609 870 Sep 25 06:45 README.md

diego@LaPoderosa MINGW64 ~/diCeRO (main)
$
```

6. Ya creada la página index.html, se debe ejecutar un **git add .** y **git commit -m "mensaje"** para que se una al repositorio, posteriormente se utiliza el comando **git pull** y **git push**.

```
diego@LaPoderosa MINGW64 ~/diCeR0 (main)
$ ls -al
total 20
drwxr-xr-x 1 diego 197609 0 Sep 25 07:08 .
drwxr-xr-x 1 diego 197609 0 Sep 25 06:45 ..
drwxr-xr-x 1 diego 197609 0 Sep 25 07:09 .git/
-rw-r--r-- 1 diego 197609 870 Sep 25 06:45 README.md
-rw-r--r-- 1 diego 197609 754 Sep 25 07:06 index.html

diego@LaPoderosa MINGW64 ~/diCeR0 (main)
$ git add .

diego@LaPoderosa MINGW64 ~/diCeR0 (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)
    new file:   index.html

diego@LaPoderosa MINGW64 ~/diCeR0 (main)
$ git commit -m "Archivo index.html añadido al repositorio de Git Pages"
[main ba94e51] Archivo index.html añadido al repositorio de Git Pages
 1 file changed, 22 insertions(+)
 create mode 100644 index.html

diego@LaPoderosa MINGW64 ~/diCeR0 (main)
$ git pull origin main
From github.com:diCeR0/diCeR0
 * branch          main      -> FETCH_HEAD
Already up to date.

diego@LaPoderosa MINGW64 ~/diCeR0 (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 721 bytes | 721.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:diCeR0/diCeR0.git
  41bf689..ba94e51  main -> main

diego@LaPoderosa MINGW64 ~/diCeR0 (main)
$
```

7. Posteriormente deberá configurar el repositorio dentro de GitHub por medio de la siguiente opción: *Repositories → Repositorio con mi nombre de usuario → Settings del repositorio → Pages → Source: Deploy from a branch → Branch: Main → Save.*

Search or jump to... Pull requests Issues Marketplace Explore

Overview Repositories 8 Projects Packages Stars 4

Hi, I'm @diCeR0

- Charismatic mechatronic inventor with a strong sense of loyalty and justice, persistent and competitive, oriented towards robotic optimization in cutting-edge technological projects to drive the modernization of people, organizations and processes.
- I'm interested in IT, artificial intelligence, Mobile & Web development, electronics, microcontrollers, IoT, robotics, etc.

Search or jump to... Pull requests Issues Marketplace Explore

Overview Repositories 8 Projects Packages Stars 4

Find a repository... Type Language Sort New

**diCeR0** (Public) Config files for my GitHub profile.

`config` `github-config`

HTML Updated 9 minutes ago

**Primer\_repositorio\_prueba** (Public) Primer repositorio de prueba para refreshar conocimientos de Git

HTML Starred

**hyperblog\_platzi\_example** (Public) Forked from freddier/hyperblog

Un blog increíble para el curso de Git y Github de Platzi

Star

Search or jump to... Pull requests Issues Marketplace Explore

diCeR0 / diCeR0 (Public)

Code Pull requests Actions Projects Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

diCeR0 Archivo index.html añadido al repositorio de Git Pages 11 minutes ago 3 commits

README.md Update README.md 3 months ago

index.html Archivo index.html añadido al repositorio de Git Pages 11 minutes ago

README.md

diCeR0/diCeR0 is a special repository. Its README.md will appear on your public profile.

Edit README Visit profile

About Config files for my GitHub profile.

github.com/diCeR0

Search or jump to... Pull requests Issues Marketplace Explore

diCeR0 / diCeR0 (Public)

Code Pull requests Actions Projects Security Insights Settings

General

Access Collaborators Moderation options

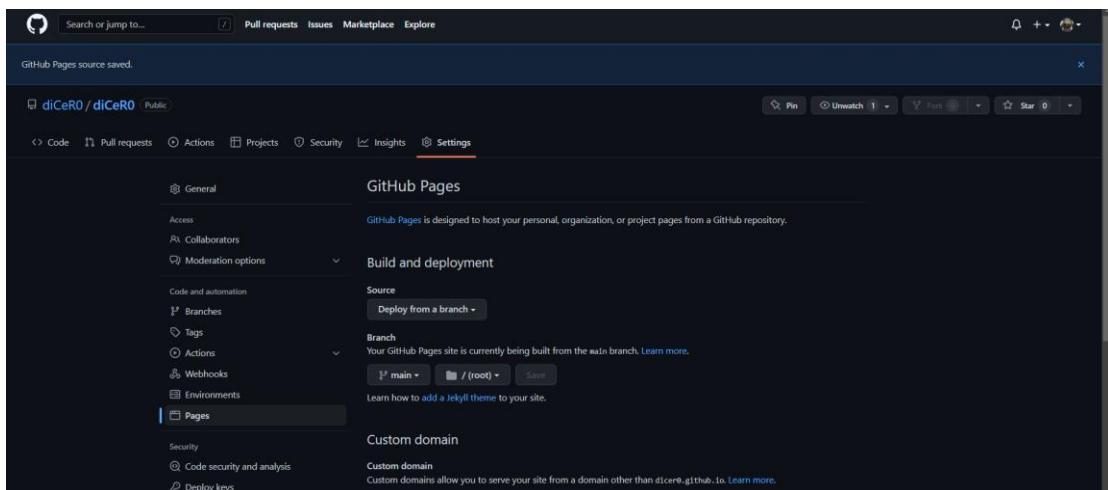
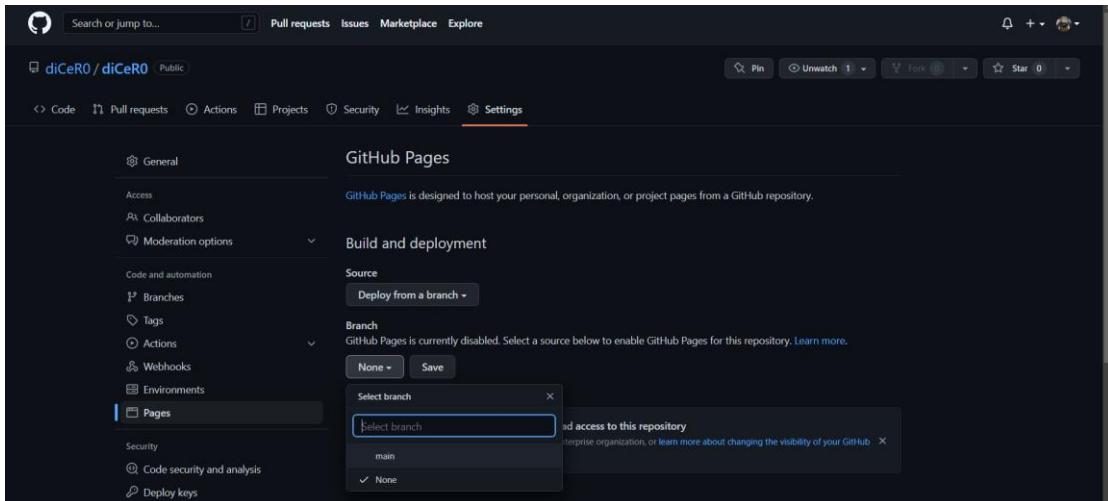
Code and automation Branches Tags Actions Webhooks Environments Pages

Repository name diCeR0 Rename

Template repository

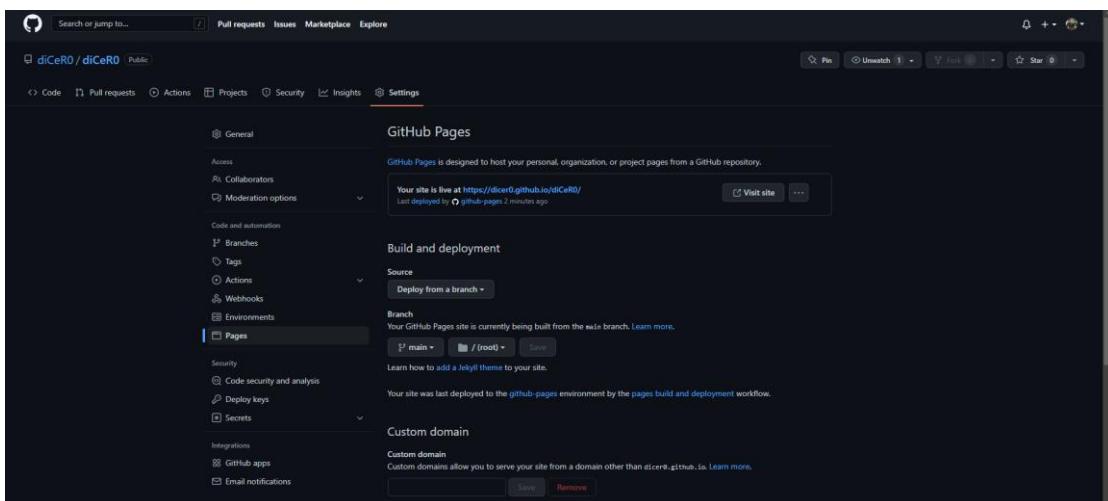
Require contributors to sign off on web-based commits

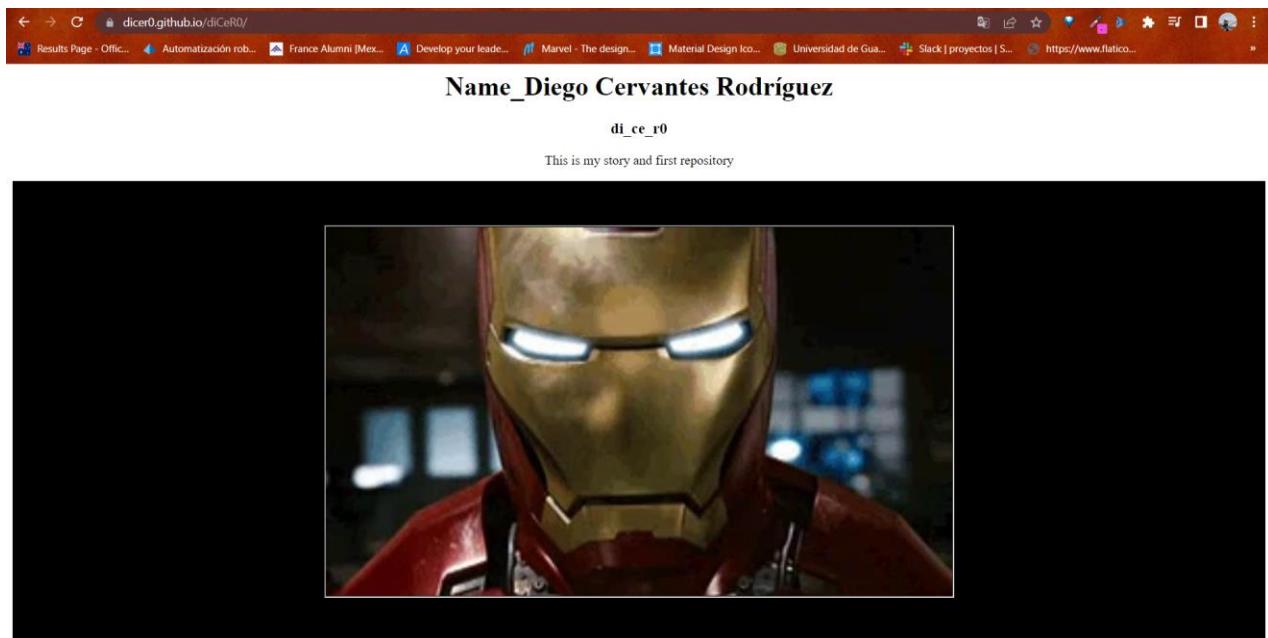
Social preview Upload an image to customize your repository's social media preview. Images should be at least 640x320px (1280x640px for best display). Download template



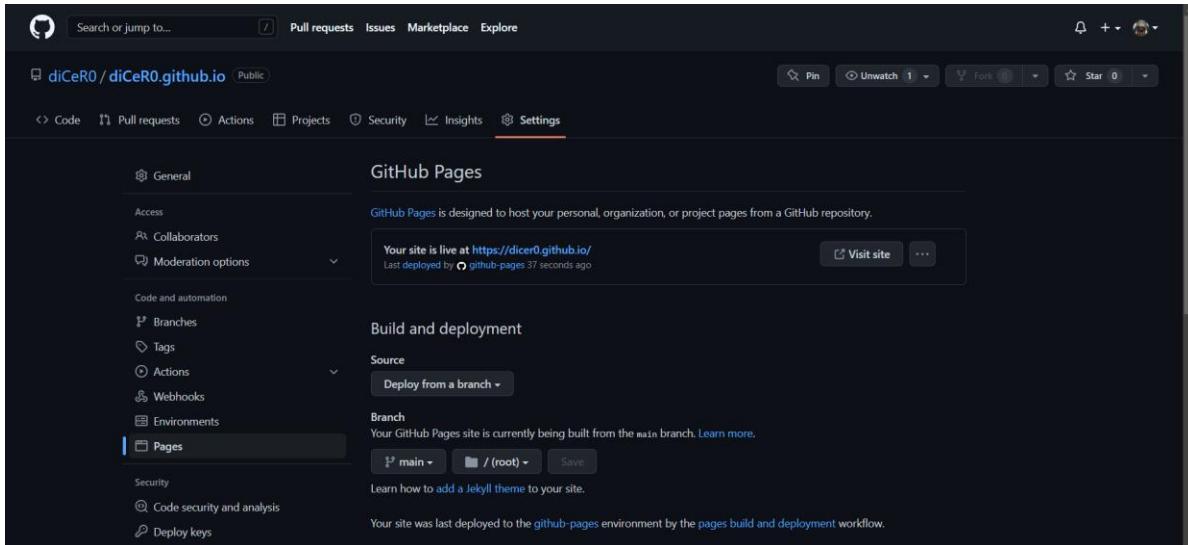
*Hasta podemos declarar un dominio personalizado en la parte que dice Custom Domain.*

8. *habiendo empujado el archivo index.html y configurado el repositorio, podré ingresar al siguiente dominio y ver su contenido: nombre\_usuario\_github.github.io/nombre\_repositorio, esto hasta se muestra dentro de la pestaña de Pages:*

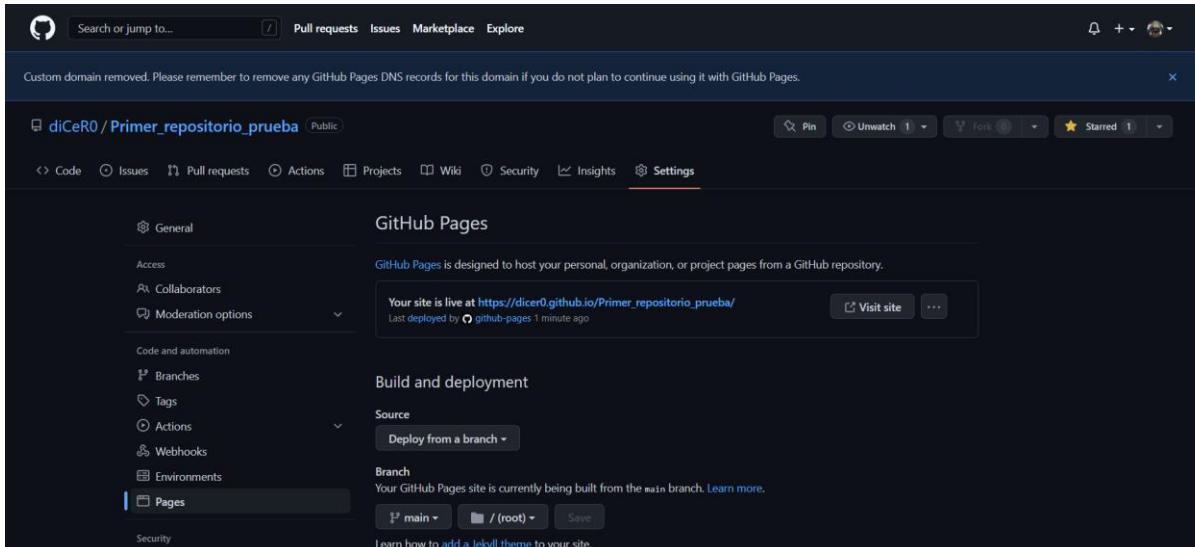




9. Si quiero que el dominio del proyecto sea `nombre_usuario.github.io` en vez de `nombre_usuario.github.io/nombre_repositorio`, lo que se hace es cambiar el nombre del repositorio a `nombre_usuario.github.io`. Este cambio se verá reflejado dentro de la pestaña de Pages.

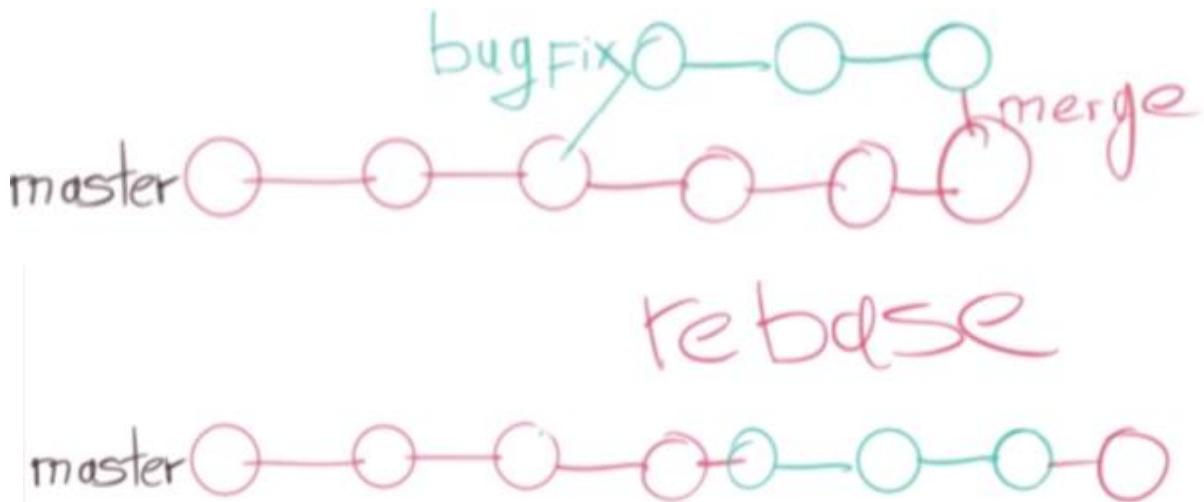


10. *Esto se puede aplicar a cualquier repositorio aplicando la siguiente opción: Repositories → Repositorio que quiero ver en un dominio online → Settings del repositorio → Pages → Source: Deploy from a branch → Branch: Main → Save. Debemos esperar un poco para que esto se vea reflejado, una vez completado el proceso el dominio aparecerá dentro de la página de GitHub en la pestaña de Pages.*



Eliminación de una rama externa y fusión de su historial (rebase):

Hay algunas ramas (más que nada la llamada **Bugfixing** que se utiliza para eliminar errores) que no nos sirve tenerlas en la línea de tiempo del proyecto, por lo cual algunas veces conviene eliminarlas al 100% sin necesariamente quitar su historial, para ello existe un comando llamado **rebase**, que funciona de manera muy distinta a realizar un **merge**. Esto solamente se realiza con las ramas del **Repositorio local**, es de muy mala práctica hacerlo en las ramas del **Repositorio remoto**, esto porque recordemos que las ramas y tags de un proyecto almacenadas en GitHub son versiones alternativas o checkpoints importantes, mientras que las ramas y tags en local son solo pruebas, de las cuales se eligen las mejores para mandarse a GitHub.



28. **git rebase nombre\_rama**: Es un comando que solo se debe aplicar a repositorios locales porque lo que hace es borrar una rama de la línea de tiempo de Git, uniendo su historial a la rama indicada en la instrucción, logrando así que parezca como si la rama eliminada nunca hubiera existido, por lo que es muy distinto a aplicar el comando **git merge**.
- a. Este comando siempre se debe ejecutar primero desde la rama que quiero eliminar.
  - b. Luego debemos cambiarnos a la rama que quiero que se conserve y aplicar el mismo comando, pero ahora desde esta rama.
    - i. Si no se siguen de forma correcta estos pasos, se creará un problema que solo se podrá resolver con el comando 11 git reset.
  - c. Una vez unidas las dos ramas con **git rebase**, se debe ejecutar el comando 12.a git branch -D nombre\_rama para eliminar completamente la rama.

En el siguiente ejemplo el contenido de las ramas **hotfix** y **prueba\_binaria** son iguales, por lo cual se quiere eliminar una de ellas:

- La rama que se eliminará es **hotfix**, por lo cual se deberá ejecutar el comando **git rebase nombre\_rama** primero en ella, mencionando el nombre de la rama **prueba\_binaria**.
- Luego se ejecuta el mismo comando **git rebase nombre\_rama** en la rama **prueba\_binaria** pero mencionando el nombre de la rama **hotfix**, para que al final la rama **prueba\_binaria** sea la que permanezca en la línea de tiempo.

```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositoryo
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositoryo
(hotfix)
$ alias arbolito="git log --all --graph --decorate --oneline"

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositoryo
(hotfix)
$ arbolito
* 997d689 (origin/main, main) Merge branch 'main' of github.com:diCeRO/Primer_repository_prueba
| \
| * bb72ec4 Delete CNAME
| * e052218 Create CNAME
* | fca57d3 Cambios para mostrar el repositorio online en un dominio de GitHub Pages
|/
* 8536483 Cambio del archivo read me
* ee21668 Cambio del archivo read me
* 6522af9 Cambio del archivo read me
* e8ed70b Cambio del archivo read me
* 386e7d0 Agregado del archivo .gitignore y añadido de imágenes de ironman al archivo de la rama main
* 3f3282a cambio puntito
* ff10034 Merge pull request #1 from diCeRO/staging_develop
| \
| * 471cfea (origin/staging_develop, staging_develop) Cambios para activar la función de Pull request desde la rama staging develop
| * 7d3b42a Cambio para activar la función de Pull request desde la rama staging develop
| * 83c34ae (HEAD -> hotfix, origin/prueba_binaria, prueba_binaria) agrego de imagenes al archivo correctamente
| * caa2565 agregue imagenes al código, pero no funcionó
| * 9fbe0c6 agrego archivos binarios (imagenes) a una rama de prueba
|/
* ca01a0a commit hecho después de añadir la conexión cifrada SSH
* d64377d cambio hecho desde Git Hub.
* 5a59459 Merge branch 'main' of https://github.com/diCeRO/Primer_repository_prueba
| \
| * 682551d Initial commit
| * 27b0713 (master) agregado mensaje rama master papus
|/
* 70a411e mejoramiento de estilo después del merge
* 3d2799e merge de las dos ramas después de resolver el conflicto
| \
| * cb9ca25 (nueva_rama) Cambio nueva rama texto
| * bb0aba7 Primer commit nueva rama con nuevo estilo html
* | 14ab156 ultimo cambio antes del merge en la rama master
* | f96b312 cambio de estilo papuuus
* | 78e0eee Cabecera nuevecita papus
|/
* 8086b5a texto centrado en el html y unión con archivo css
* 1a54d43 Crear carpeta css y cambios en archivo txt y html
* 595cc84 Renombrar ejemplo 1 y crear ejemplo 2 después del reset
* 71944ea Primer commit del repositorio local
```

```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositoryo
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositoryo
(hotfix)
$ git branch
* hotfix
  main
  master
  nueva_rama
  prueba_binaria
  staging_develop

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositoryo
(hotfix)
$ git rebase prueba_binaria
Current branch hotfix is up to date.

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositoryo
(hotfix)
$ git checkout prueba_binaria
Switched to branch 'prueba_binaria'

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositoryo
(prueba_binaria)
$ git rebase hotfix
Current branch prueba_binaria is up to date.
```

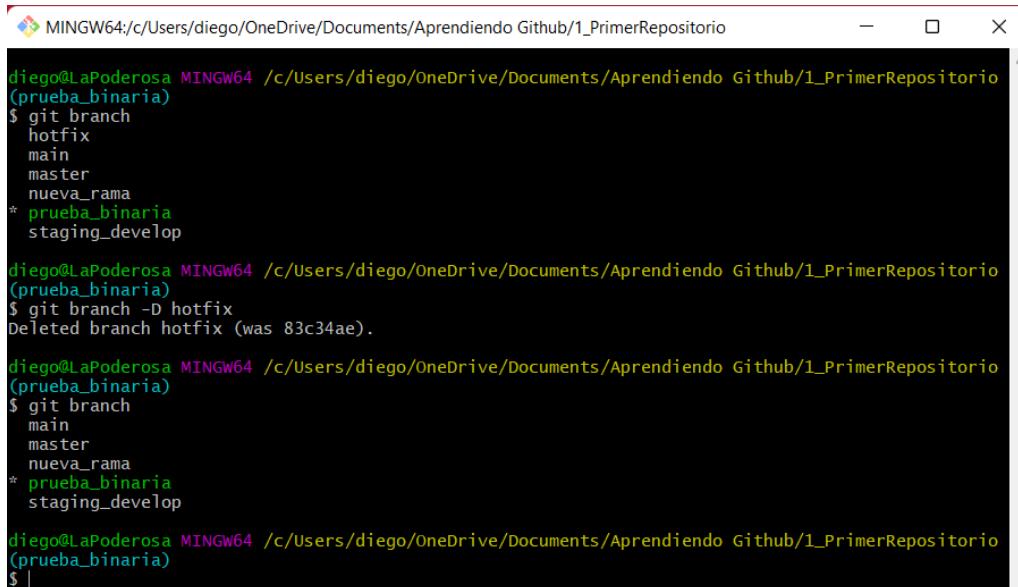
```

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
(prueba_binaria)
$ arbolito
* 997d689 (origin/main, main) Merge branch 'main' of github.com:diCeRO/Primer_repositorio_prueba
| \
| * bb72ec4 Delete CNAME
| * e052218 Create CNAME
* | fca57d3 Cambios para mostrar el repositorio online en un dominio de GitHub Pages
|/
* 8536483 Cambio del archivo read me
* ee21668 Cambio del archivo read me
* 6522af9 Cambio del archivo read me
* e8ed70b Cambio del archivo read me
* 386e7d0 Agregado del archivo .gitignore y añadido de imágenes de ironman al archivo de la rama main
* 3f3282a cambio puntito
* ff10034 Merge pull request #1 from diCeRO/staging_develop
| \
| * 471cfea (origin/staging_develop, staging_develop) Cambios para activar la función de Pull request desde la rama staging develop
| * 7d3b42a Cambio para activar la función de Pull request desde la rama staging develop
| * 83c34ae (HEAD -> prueba_binaria, origin/prueba_binaria, hotfix) agrego de imagenes al archivo correctamente
| * caa2565 agregue imagenes al código, pero no funcionó
| * 9fbe0c6 agrego archivos binarios (imagenes) a una rama de prueba
|/
* ca01a0a commit hecho después de añadir la conexión cifrada SSH
* d64377d cambio hecho desde Git Hub.
* 5a59459 Merge branch 'main' of https://github.com/diCeRO/Primer_repositorio_prueba
| \
| * 682551d Initial commit
| * 27b0713 (master) agregado mensaje rama master papus
|/
* 70a411e mejoramiento de estilo despues del merge
* 3d2799e merge de las dos ramas después de resolver el conflicto
| \
| * cb9ca25 (nueva_rama) Cambio nueva rama texto
| * bb0aba7 Primer commit nueva rama con nuevo estilo html
* | 14ab156 ultimo cambio antes del merge en la rama master
* | f96b312 cambio de estilo papuuus
* | 78e0eee Cabecera nuevecita papus
|/
* 8086b5a texto centrado en el html y unión con archivo css
* 1a54d43 Crear carpeta css y cambios en archivo txt y html
* 595cc84 Renombrar ejemplo 1 y crear ejemplo 2 después del reset
* 71944ea Primer commit del repositorio local

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
(prueba_binaria)
$ |

```

Después de haber aplicado el comando **git rebase nombre\_rama** podremos borrar completamente la rama de forma local con el comando 12.a git branch -D nombre\_rama, de igual manera podemos conservar la rama si no aplicamos el comando y no habrá ningún problema.



```

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
(prueba_binaria)
$ git branch
hotfix
main
master
nueva_rama
* prueba_binaria
staging_develop

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
(prueba_binaria)
$ git branch -D hotfix
Deleted branch hotfix (was 83c34ae).

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
(prueba_binaria)
$ git branch
main
master
nueva_rama
* prueba_binaria
staging_develop

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
(prueba_binaria)
$ |

```

```

MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositoryo
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositoryo (prueba_binaria)
$ arbolito
* feb7b26 (HEAD -> prueba_binaria, origin/prueba_binaria) git rebase ramas hotfix y prueba_unitaria hacia GitHub pt2
* 484fe98 git rebase ramas hotfix y prueba_unitaria hacia GitHub
* b4d2cb0 git rebase eliminando el historial de la rama hotfix y uniéndola a la rama prueba_unitaria
* 997d689 (origin/main, main) Merge branch 'main' of github.com:diCeR0/Primer_repositoryo_prueba
  \
  * bb72ec4 Delete CNAME
  * e052218 Create CNAME
  * fca57d3 Cambios para mostrar el repositorio online en un dominio de GitHub Pages
  /
* 8536483 Cambio del archivo read me
* ee21668 Cambio del archivo read me
* 6522af0 Cambio del archivo read me
* e8ed70b Cambio del archivo read me
* 386e7d0 Agregado del archivo .gitignore y añadido de imágenes de ironman al archivo de la rama main
* 3f3282a cambio puntito
* ff10034 Merge pull request #1 from diCeR0/staging_develop
  \
  * 471cfea (origin/staging_develop, staging_develop) Cambios para activar la función de Pull request desde la rama staging develop
  * 7dbb42a Cambio para activar la función de Pull request desde la rama staging develop
  /
* 83c34ae agrego de imagenes al archivo correctamente
* caa2565 agregue imagenes al código, pero no funcionó
* 9fbe0c6 agrego archivos binarios (imagenes) a una rama de prueba
  \
* ca01a0a commit hecho después de añadir la conexión cifrada SSH
* d64377d cambio hecho desde Git Hub.
* Sa59459 Merge branch 'main' of https://github.com/diCeR0/Primer_repositoryo_prueba
  \
  * 682551d Initial commit
  * 27b0713 (master) agregado mensaje rama master papus
  /
* 70a411e mejoramiento de estilo despues del merge
* 3d2799e merge de las dos ramas despues de resolver el conflicto
  \
  * cb9ca25 (nueva_rama) Cambio nueva rama texto
  * bb0aba7 Primer commit nueva rama con nuevo estilo html
  * 14ab156 ultimo cambio antes del merge en la rama master
  * f96b312 cambio de estilo papuus
  * 78e0eee Cabecera nuevecita papus
  /
* 8086b5a texto centrado en el html y unión con archivo css
* 1a54d43 Crear carpeta css y cambios en archivo txt y html
* 595cc84 Renombrar ejemplo 1 y crear ejemplo 2 después del reset
* 71944ea Primer commit del repositorio local

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositoryo (prueba_binaria)
$ |

```

De igual manera se puede visualizar esta fusión y eliminación de ramas con el comando 17.b llamado `gitk`, que abre un visualizador de branches del proyecto permitiendo así ver de mejor manera todas las ramas y líneas de tiempo.

Cuando nos movemos de una rama a otra dentro de GitHub, no debe existir ningún cambio pendiente en el **Staging Area**, ya que, sino la consola Git Bash me marcará un error y no me dejará moverme de rama, ¿pero que pasaría si estos cambios no los quiero mandar al **Repositorio local** todavía, pero si quiero moverme a la otra rama? Para eso existe el **stash**, que es una zona de memoria donde puedo guardar temporalmente los cambios que todavía no quiero mandar al repositorio, pero si quiero sacar del **Staging Area**, a su vez regresando el proyecto al estado del último commit realizado.

29. **git stash**: Comando que sirve para almacenar los nuevos cambios realizados en el proyecto en una memoria temporal llamada **stash**, vaciando de esta manera el contenido del **Staging Area**.

- git stash list**: Comando que muestra una lista de todos los cambios guardados en el **stash**, indicando la palabra **WIP (Work In Progress)**, el número del último commit y la rama a la que pertenece el cambio.
  - git stash drop**: Elimina todos los cambios guardados en el stash list.
- git stash pop**: Saca los cambios almacenados en el stash y los regresa al **Staging Area**.
- git stash branch nombre\_nueva\_rama**: Comando que crea una copia del contenido de mi rama actual en una **nueva rama** con todo y los nuevos cambios almacenados actualmente en el **stash** en el **Staging Area** de la dicha nueva branch, vaciando de esta manera el **stash** de la rama actual.

```

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git status
On branch master
nothing to commit, working tree clean

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git status
On branch master
Changes to be staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified:   2_Ejemplo2.html

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

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git branch
* master
  nueva_rama
  prueba_binaria
  staging_desarrollador

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git checkout prueba_binaria
error: Your local changes to the following files would be overwritten by checkout:
  2_Ejemplo2.html
Please, commit your changes or stash them before you switch branches.
Aborting

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ 
```

Cuando utilice el comando **git stash** lo que pasará es que los nuevos cambios que se encuentren en el **Staging Area** se dejarán de ver en el código y este se verá cómo estaba antes de esos cambios, aunque estos no serán borrados, sino que estarán guardados temporalmente en el **stash**.

```

Archivo Editar Selección Ver Ir ... 2_Ejemplo2.html x README.md gitignore st D ...
EXPLORADOR ... 2_Ejemplo2.html x README.md gitignore st D ...
    ✓ 1_PRIMERREPOSITORIO
        ✓ css styles.css
        > img
            ✓ gllignore
        2_Ejemplo1.txt
        README.md
    ✓ ESTRUCTURA
        ✓ html
            ✓ head
            ✓ meta
            ✓ meta
            ✓ meta
            ✓ title
            ✓ link
        ✓ hoja
            ✓ LINEA DE TIEMPO 2_Ejemplo2.html
                ○ Archivo guardado 1 min
                ○ Archivo guardado 22 minutos
                ○ Archivo guardado 25 minutos
                ○ Archivo guardado 33 minutos
                ○ Archivo guardado 12 h
                ○ Deshacer / Rehacer 14 h
                ○ Archivo guardado
                    ✓ Agregado del archivo .gitignore
                ○ Archivo guardado 15 h
                ○ Archivo guardado
                    ✓ Archivo renombrado
    ✓ L
    ✓ S
    ✓ C
    ✓ M
    ✓ A
    ✓ V
    ✓ P
    ✓ I
    ✓ F
    ✓ O
    ✓ U
    ✓ Live Share
Ún. 18, col. 1 Espacios: 4 UTF-8 CR LF HTML R ...
MINGW64/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git status
On branch main
nothing to commit, working tree clean
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git status
On branch 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: 2_Ejemplo2.html
no changes added to commit (use "git add" and/or "git commit -a")
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git branch
* main
  master
  nueva_rama
  prueba_binaria
  staging_develop
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git checkout prueba_binaria
error: Your local changes to the following files would be overwritten by checkout:
 2_Ejemplo2.html
Please commit your changes or stash them before you switch branches.
Aborting
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git stash
Saved working directory and index state WIP on main: 997d689 Merge branch 'main' of github.com:dicero/Primer_repositorio_prueba
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ 

```

Mientras la **Staging Area** se encuentre vacía y los nuevos cambios estén en el **stash** me podré mover entre las distintas ramas del proyecto, pero es importante mencionar que los cambios guardados en el **stash** no deberán ser liberados en una rama distinta a la que pertenecían, sino se podrán crear conflictos, para poder ejecutar el comando **git stash pop** que libera los últimos cambios debemos estar situados en la rama donde pertenecían.

```

Archivo Editar Selección Ver Ir ... 2_Ejemplo2.html x README.md styles.css ...
EXPLORADOR ... 2_Ejemplo2.html x README.md styles.css ...
    ✓ 1_PRIMERREPOSITORIO
        ✓ css styles.css
        > img
            ✓ gllignore
        2_Ejemplo1.txt
        2_Ejemplo2.html
        README.md
    ✓ ESTRUCTURA
        ✓ html
            ✓ head
            ✓ meta
            ✓ meta
            ✓ meta
            ✓ title
            ✓ link
        ✓ hoja
            ✓ LINEA DE TIEMPO 2_Ejemplo2.html
                ○ Archivo guardado 10 minutos
                ○ Archivo guardado 30 minutos
                ○ Archivo guardado 31 minutos
                ○ Archivo guardado 33 minutos
                ○ Archivo guardado 34 minutos
                ○ Archivo guardado 41 minutos
                ○ Archivo guardado 42 minutos
                ○ Archivo guardado 12 h
                ○ Deshacer / Rehacer 14 h
                ○ Archivo guardado 15 h
                ○ Archivo guardado
                    ✓ Archivo renombrado
    ✓ L
    ✓ S
    ✓ C
    ✓ M
    ✓ A
    ✓ V
    ✓ P
    ✓ O
    ✓ U
    ✓ Live Share
Ún. 34, col. 1 Espacios: 4 UTF-8 CR LF HTML R ...
MINGW64/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git status
On branch main
nothing to commit, working tree clean
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git branch
* main
  master
  nueva_rama
  prueba_binaria
  staging_develop
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git checkout prueba_binaria
error: Your local changes to the following files would be overwritten by checkout:
 2_Ejemplo2.html
Please commit your changes or stash them before you switch branches.
Aborting
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git stash
Saved working directory and index state WIP on main: 997d689 Merge branch 'main' of github.com:dicero/Primer_repositorio_prueba
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git stash list
stash@{0}: WIP on main: 997d689 Merge branch 'main' of github.com:dicero/Primer_repositorio_prueba
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git checkout prueba_binaria
Switched to branch 'prueba_binaria'
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
(prueba_binaria)
$ 

```

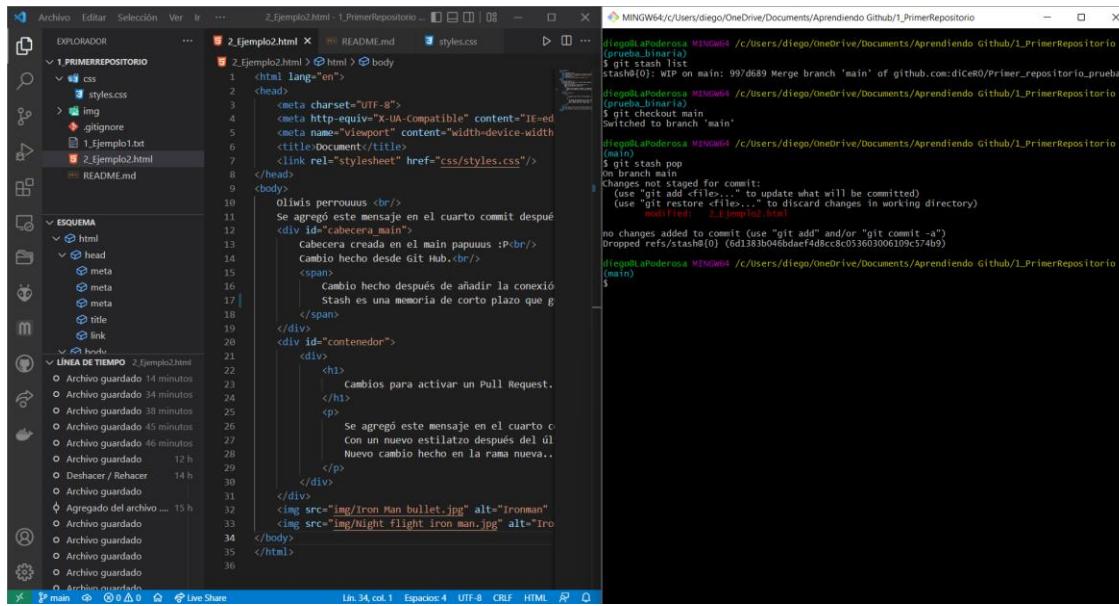
```

MINGW64/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
(prueba_binaria)
$ git stash list
stash@{0}: WIP on main: 997d689 Merge branch 'main' of github.com:dicero/Primer_repositorio_prueba
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
(prueba_binaria)
$ git checkout main
Switched to branch 'main'
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
(main)
$ 

```

Ya restaurados los cambios, puedo hacer con ellos lo que sea, ya sea borrarlos, presionar CTRL + Z para deshacerlos, mandarlos al **Repositorio local**, etc.

El comando **git stash** no solamente sirve para guardar cambios que si quiera mandar al **Repositorio local** después, sino para eliminar errores cometidos en varios archivos en el código, antes de que estos sean mandados al repositorio con un **commit**, esto porque **git stash** no solo guarda esos cambios de forma temporal, sino que me regresa al estado del último **commit** realizado, **es como si diera CTRL + Z en varios archivos a la vez**. Por eso mismo es que el **stash** se usa mucho para realizar experimentos que no merecen tener una rama propia, sino que solo se intentan y si no sale el resultado esperado se mandan esos cambios al **stash** y se borran con el comando **git stash drop**.



The screenshot shows a Windows terminal window titled 'MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1\_PrimerRepositorio'. It displays a series of git commands:

```
diego@diegobarrera MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git stash list
stash@{0}: WIP on main: 997d689 Merge branch 'main' of github.com:dice00/Primer_repositorio_prueba
(prueba_blanca)
diego@diegobarrera MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git stash pop
On branch main
Changes were staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      (use "git reset" to discard local changes)

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (6d138b046bdeef4dbccc053603906109c574b9)
diego@diegobarrera MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$
```

Below the terminal, a code editor shows the content of '2\_Ejemplo2.html'. The code includes several comments explaining the state of the code after different commits:

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link rel="stylesheet" href="css/styles.css"/>
</head>
<body>
  Olíwis perrouus <br/>
  Se agregó este mensaje en el cuarto commit después
  de agregar el archivo 'img/iron_man_bullet.jpg'.
  <div id="cabecera_main">
    Cabecera creada en el main papuuus :P<br/>
    Cambio hecho desde Git Hub.<br/>
    <span>
      Cambio hecho después de añadir la conexión
      Stash es una memoria de corto plazo que g
    </span>
  </div>
  <div id="contenedor">
    <div>
      Cambios para activar un Pull Request.
    </div>
    <p>
      Se agregó este mensaje en el cuarto commit
      Con un nuevo estilazo después del primero.
      Nuevo cambio hecho en la rama nueva...
    </p>
  </div>
  
  
</body>
```

Cuando queremos traer el contenido de cierto **commit** anterior al último **commit** perteneciente a una rama externa a la que me encuentro actualmente, puedo utilizar el comando **git cherry-pick número\_commit**.

30. **git cherry-pick número\_commit\_rama\_distinta**: Comando que jala el contenido de un **commit** viejo de en una rama diferente a la cual me encuentro ahorita, hacia mi rama actual. **Es malas prácticas usar este comando, hay que tener cuidado con él porque puede crear un desastre si no se usa bien, ya que en principio el código de las distintas ramas puede ser muy diferente entre sí.**

Resolución de errores fatales en la línea de tiempo del proyecto:

*Git no olvida nada, entonces aun cuando hayamos borrado archivos importantes del proyecto, hecho un merge accidental entre dos ramas, borrado una branch importante o demás escenarios catastróficos de pesadilla, hay un comando que nos puede servir para salvarlo todo llamado **git reflog**.*

Recordemos que el comando **git log** sirve para ver el historial de commits realizados en el repositorio, pero este puede ser afectado por otros comandos, ya sea porque haya sido borrada una rama completa

*o el log de la línea de tiempo hasta cierto punto si ejecuto por ejemplo el comando **11 git reset** por accidente, pero el comando **git reflog** no perdona nada, ya que muestra el historial de todos, absolutamente todos los comandos y acciones ejecutadas en todas las ramas del proyecto.*

**31. git reflog:** Comando que muestra el historial de todos los movimientos ejecutados por el **HEAD** en el proyecto, mostrando el número de commit (hash) de cada operación, la rama en la que se encontraba el **HEAD** y el mensaje del commit.

- Una vez encontrado el hash o número del **HEAD** que me retorna al estado del proyecto que quiero retornar, se utiliza el **comando 11 git reset número\_commit\_(hash)\_o\_HEAD\_versión\_anterior --soft o --hard** para retornar al estado que quiero antes del error.

```

MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git branch
  master
* nueva_rama
+ prueba_binaria
  staging/develop

MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git commit -am "Error fatal resuelto solo con git reflog"
[prueba_binaria 3fb7e1b] Error fatal resuelto solo con git reflog
MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git reflog
3fb7e1b (HEAD -> prueba_binaria) HEAD@{0}: commit (merge): Error fatal resuelto solo con git reflog
fcb7b26 (origin/prueba_binaria) HEAD@{1}: checkout: moving from master to prueba_binaria
27b0713 (master) HEAD@{2}: commit (merge): Error fatal resuelto solo con git reflog
b02daae (origin/main, main) HEAD@{3}: reset: moving to HEAD
b02daae (origin/main, main) HEAD@{4}: checkout: moving from prueba_binaria to main
997d689 HEAD@{5}: checkout: moving from prueba_binaria to main
fcb7b26 (origin/prueba_binaria) HEAD@{6}: checkout: moving from main to prueba_binaria
997d689 HEAD@{7}: reset: moving to HEAD
997d689 HEAD@{8}: checkout: moving from prueba_binaria to main
fca57d3 HEAD@{9}: commit: git rebase ramas hotfix y prueba_unitaria hacia GitHub pt2
484fe98 HEAD@{10}: commit: git rebase ramas hotfix y prueba_unitaria hacia GitHub
b4d2c0f HEAD@{11}: commit: checkout: moving from prueba_binaria eliminando el historial de la rama hotfix y uniéndola a la rama prueba_unitaria
83c34ac HEAD@{12}: checkout: moving from hotfix to prueba_binaria
83c34ac HEAD@{13}: checkout: moving from prueba_binaria to hotfix
83c34ac HEAD@{14}: checkout: moving from hotfix to prueba_binaria
83c34ac HEAD@{15}: checkout: moving from prueba_binaria to hotfix
83c34ac HEAD@{16}: checkout: moving from staging/develop to prueba_binaria
471cfea (origin/staging/develop, staging/develop) HEAD@{17}: checkout: moving from hotfix to staging/develop
83c34ac HEAD@{18}: checkout: moving from prueba_binaria to hotfix
83c34ac HEAD@{19}: checkout: moving from hotfix to prueba_binaria
83c34ac HEAD@{20}: checkout: moving from prueba_binaria to hotfix
83c34ac HEAD@{21}: checkout: moving from nueva_rama to prueba_binaria
109c425 HEAD@{22}: commit: Cambio previamente guardado en el stash
27b0713 (master) HEAD@{23}: checkout: moving from hotfix to master
83c34ac HEAD@{24}: checkout: moving from main to hotfix
997d689 HEAD@{25}: pull origin main: Merge made by the 'recursive' strategy.
fca57d3 HEAD@{26}: checkout: moving from main to staging/develop
27b0713 (master) HEAD@{27}: checkout: moving from main to master
fca57d3 HEAD@{28}: commit: Cambios para mostrar el repositorio online en un dominio de GitHub Page

```

```

MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
$ git reset --hard HEAD@{2}
HEAD is now at 27b0713 agregado mensaje rama master papus

MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
(prueba_binaria)
$ git branch
  master
* 27b0713 HEAD -> prueba_binaria, master) HEAD@{0}: reset: moving to HEAD@{2}
-3b7e1b1 HEAD@{1}: commit (merge): Error fatal resuelto solo con git reflog
fcb7b26 (origin/prueba_binaria) HEAD@{2}: checkout: moving from master to prueba_binaria
27b0713 (HEAD -> prueba_binaria, master) HEAD@{3}: commit: Cambio previamente guardado en el stash
b02daae (origin/main, main) HEAD@{4}: commit: Cambio previamente guardado en el stash
997d689 HEAD@{5}: checkout: moving from prueba_binaria to main
fcb7b26 (origin/prueba_binaria) HEAD@{6}: checkout: moving from prueba_binaria to main
997d689 HEAD@{7}: commit: git rebase ramas hotfix y prueba_unitaria hacia GitHub
b4d2c0f HEAD@{8}: commit: checkout: moving from prueba_binaria eliminando el historial de la rama hotfix y uniéndola a la rama prueba_unitaria
83c34ac HEAD@{13}: checkout: moving from hotfix to prueba_binaria
83c34ac HEAD@{14}: checkout: moving from prueba_binaria to hotfix
83c34ac HEAD@{15}: checkout: moving from hotfix to prueba_binaria
83c34ac HEAD@{16}: checkout: moving from prueba_binaria to hotfix
83c34ac HEAD@{17}: checkout: moving from staging/develop to prueba_binaria
471cfea (origin/staging/develop, staging/develop) HEAD@{18}: checkout: moving from hotfix to staging/develop
83c34ac HEAD@{19}: checkout: moving from prueba_binaria to hotfix
83c34ac HEAD@{20}: checkout: moving from hotfix to prueba_binaria
83c34ac HEAD@{21}: checkout: moving from prueba_binaria to hotfix
83c34ac HEAD@{22}: checkout: moving from nueva_rama to prueba_binaria
cb9c425 (nueva_rama) HEAD@{23}: checkout: moving from master to nueva_rama
27b0713 (HEAD -> prueba_binaria, master) HEAD@{24}: checkout: moving from hotfix to master
83c34ac HEAD@{25}: commit: Cambio del archivo .gitignore
997d689 HEAD@{26}: pull origin main: Merge made by the 'recursive' strategy.
fca57d3 HEAD@{27}: checkout: moving from master to main
27b0713 (HEAD -> prueba_binaria, master) HEAD@{28}: checkout: moving from main to master
fca57d3 HEAD@{29}: commit: Cambios para mostrar el repositorio online en un dominio de GitHub Page

```

*Hay que tener cuidado porque este comando si se utiliza mal puede llevar el contenido de una rama dentro de otra, en este caso como se usó mal el HEAD indicado en el comando git reset, se llevó el contenido de la rama master dentro de la rama prueba\_binaria.*



## Cabecera creada en el master papuuus :P

**Oliwis perrouuus**

Se agregó este mensaje en el cuarto commit después del reset.  
Con un nuevo estilatzo después del último commit en la nueva branch.  
Nuevo cambio hecho en la rama nueva... nuevo uy si.

```
MINGW64/c/Users/diego/OneDrive/Documents/Aprendiendo GitHub/1_PrimerRepository
diego@Alfonso: MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo GitHub/1_PrimerRepository
(prueba_binaria)
$ git reset HEAD@{1} --hard
27b0f13 (HEAD -> prueba_binaria, master) HEAD@{0}: checkout: moving from nueva_rama to prueba_binaria
c09ca25 (nueva_rama) HEAD@{1}: checkout: moving from master to nueva_rama
27b0f13 (HEAD -> prueba_binaria, master) HEAD@{2}: checkout: moving from main to master
3772030 HEAD@{3}: commit: Cambio de archivo read me
386e4d0 HEAD@{4}: checkout: moving from prueba_binaria to main
386e4d0 HEAD@{5}: commit: Cambio de archivo read me
386e4d0 HEAD@{6}: checkout: moving from prueba_binaria to main
386e4d0 HEAD@{7}: commit: Cambio de archivo read me
386e4d0 HEAD@{8}: checkout: moving from prueba_binaria to main
386e4d0 HEAD@{9}: commit: Cambio de archivo read me
386e4d0 HEAD@{10}: checkout: moving from prueba_binaria to main
386e4d0 HEAD@{11}: commit: Cambio de archivo read me
386e4d0 HEAD@{12}: reset: moving to HEAD
386e4d0 HEAD@{13}: checkout: moving from prueba_binaria to main
386e4d0 HEAD@{14}: commit: git rebase ramas hotfix y prueba_unitaria hacia GitHub
386e4d0 HEAD@{15}: commit: git rebase ramas hotfix y prueba_unitaria hacia GitHub
386e4d0 HEAD@{16}: checkout: moving from prueba_binaria to main
386e4d0 HEAD@{17}: commit: git rebase eliminando el historial de la rama hotfix y uniéndola a la rama prueba_binaria
386e4d0 HEAD@{18}: checkout: moving from prueba_binaria to main
386e4d0 HEAD@{19}: commit: Cambio previamente guardado en el stash
386e4d0 HEAD@{20}: checkout: moving from prueba_binaria to main
386e4d0 HEAD@{21}: commit: Cambio previamente guardado en el stash
386e4d0 HEAD@{22}: checkout: moving from prueba_binaria to main
386e4d0 HEAD@{23}: commit: Cambio previamente guardado en el stash
386e4d0 HEAD@{24}: checkout: moving from prueba_binaria to main
386e4d0 HEAD@{25}: commit: Cambio del archivo read me
386e4d0 HEAD@{26}: checkout: moving from prueba_binaria to main
386e4d0 HEAD@{27}: commit: Cambio del archivo read me
386e4d0 HEAD@{28}: checkout: moving from prueba_binaria to main
386e4d0 HEAD@{29}: commit: Cambio del archivo read me
386e4d0 HEAD@{30}: pull origin main. Merge made by the recursive strategy.
386e4d0 HEAD@{31}: checkout: moving from prueba_binaria to main
27b0f13 (HEAD -> prueba_binaria, master) HEAD@{12}: checkout: moving From main to master
fca57d3 HEAD@{33}: commit: Cambios para mostrar el repositorio online en un dominio de GitHub Page
...
8536483 HEAD@{34}: commit: Cambio del archivo read me
ee21668 HEAD@{35}: commit: Cambio del archivo read me
6322af6 HEAD@{36}: commit: Cambio del archivo read me
6322af6 HEAD@{37}: commit: Cambio del archivo read me
386e4d0 HEAD@{38}: checkout: moving From staging_develop to main
471cfea (origin/staging_develop, staging_develop) HEAD@{39}: checkout: moving from prueba_binaria to staging_develop
386e4d0 HEAD@{40}: checkout: moving From nueva_rama to prueba_binaria
c09ca25 (nueva_rama) HEAD@{41}: checkout: moving from master to nueva_rama
27b0f13 (HEAD -> prueba_binaria, master) HEAD@{2}: checkout: moving from main to master
386e4d0 HEAD@{43}: checkout: moving from prueba_binaria to main
386e4d0 HEAD@{44}: checkout: moving from main to hotfix
386e4d0 HEAD@{45}: commit: Agregado del archivo .gitignore y añadido de imágenes de ironman al archivo de la rama main
```

```
MINGW64/c/Users/diego/OneDrive/Documents/Aprendiendo GitHub/1_PrimerRepository
diego@Alfonso: MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo GitHub/1_PrimerRepository
(prueba_binaria)
$ git reset HEAD@{1} --hard
HEAD is now at 2eb7b26 git rebase ramas hotfix y prueba_unitaria hacia GitHub pt2
diego@Alfonso: MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo GitHub/1_PrimerRepository
(prueba_binaria)
nothing to commit, working tree clean
diego@Alfonso: MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo GitHub/1_PrimerRepository
(prueba_binaria)
$ |
```



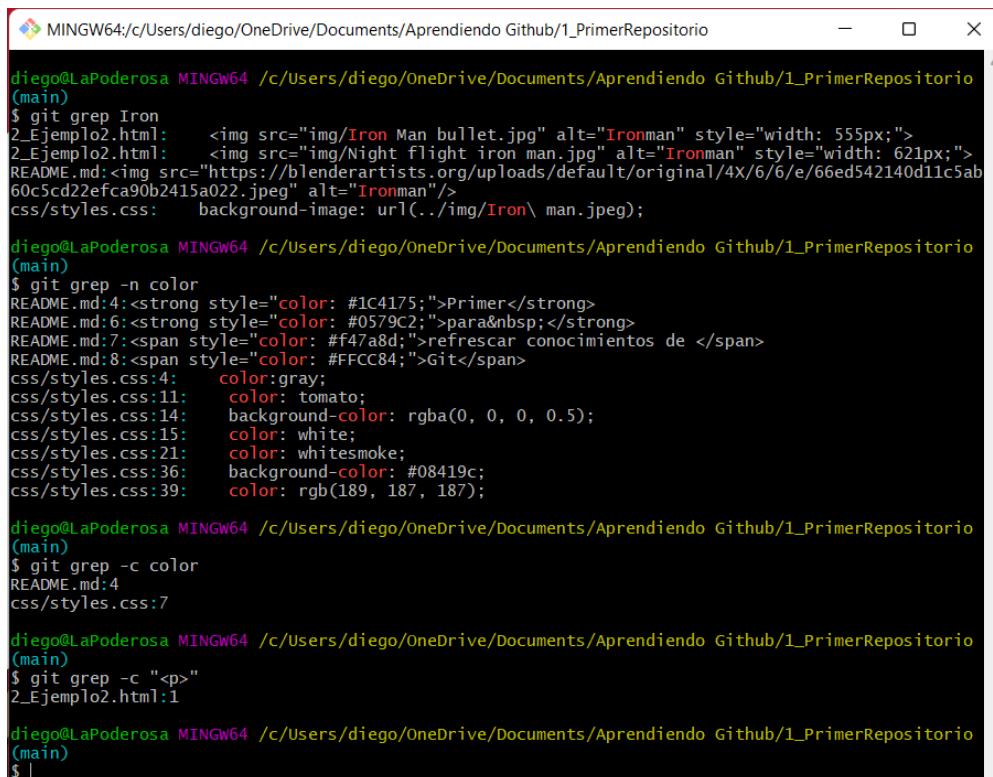
*De esta manera se habrá restaurado el proyecto después de un error fatal exitosamente, pero cabe mencionar que, aunque en el **git log** parece que el problema nunca existió, cuando ejecutamos **git reflog**, se puede ver el commit del error fatal, los cambios del HEAD entre ramas, etc. De ahí no se borra nada.*

```
MINGW64:/c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
(prueba_binaria)
$ alias arbolito="git log --all --graph --decorate --oneline"

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio
(prueba_binaria)
$ arbolito
* b02daae (origin/main, main) Cambio previamente guardado en el stash
* 997d689 Merge branch 'main' of github.com:diceR0/Primer_repositorio_prueba
| \
| * bb72ec4 Delete CNAME
| * e052218 Create CNAME
| * fca57d3 Cambios para mostrar el repositorio online en un dominio de GitHub Pages
| /
* 8536483 Cambio del archivo read me
* ee21668 Cambio del archivo read me
* 6522afd Cambio del archivo read me
* e8ed70b Cambio del archivo read me
* 386e7d0 Agregado del archivo .gitignore y añadido de imágenes de ironman al archivo de la rama main
* 3f3282a cambio puntito
* ff10034 Merge pull request #1 from diceR0/staging_develop
| \
| * 471cfea (origin/staging_develop, staging_develop) Cambios para activar la función de Pull request desde la rama staging develop
| * 7d3b42a Cambio para activar la función de Pull request desde la rama staging develop
| * feb7b26 (HEAD -> prueba_binaria, origin/prueba_binaria) git rebase ramas hotfix y prueba_unitaria hacia GitHub pt2
| * 484fe98 git rebase ramas hotfix y prueba_unitaria hacia GitHub
| * b4d2cb0 git rebase eliminando el historial de la rama hotfix y uniéndola a la rama prueba_unitaria
| /
| * 83c34ae agrego de imagenes al archivo correctamente
| * caa2565 agregue imagenes al código, pero no funcionó
| * 9fbe0c6 agrego archivos binarios (imagenes) a una rama de prueba
| /
* ca01a0a commit hecho después de añadir la conexión cifrada SSH
* d64377d cambio hecho desde Git Hub.
* 5a59459 Merge branch 'main' of https://github.com/diceR0/Primer_repositorio_prueba
| \
| * 682551d Initial commit
| * 27b0713 (master) agregado mensaje rama master papus
| /
* 70a411e mejoramiento de estilo despues del merge
* 3d2799e merge de las dos ramas después de resolver el conflicto
| \
| * cb9ca25 (nueva_rama) Cambio nueva rama texto
| * bb0aba7 Primer commit nueva rama con nuevo estilo html
* 14ab156 ultimo cambio antes del merge en la rama master
* f96b312 cambio de estilo papuuus
* 78e0eee Cabecera nuevecita papus
| /
* 8086b5a texto centrado en el html y unión con archivo css
* 1a54d43 Crear carpeta css y cambios en archivo txt y html
* 595cc84 Renombrar ejemplo 1 y crear ejemplo 2 después del reset
* 71944ea Primer commit del repositorio local
```

32. **git grep palabra\_a\_buscar**: Comando que sirve para buscar cierta palabra utilizada en alguno de los archivos pertenecientes a mi **Repositorio local**.

- git grep -n palabra\_a\_buscar**: Comando que indica la carpeta, el nombre del archivo y línea de código donde se encuentra la palabra que quiero buscar en mi **Repositorio local**.
- git grep -c palabra\_a\_buscar**: Comando que indica el número de veces que aparece la palabra indicada y dónde aparece dentro del **Repositorio local**.
  - Para poder buscar etiquetas con los comandos **git grep** se debe poner dobles comas alrededor de la etiqueta a buscar:
    - git grep "etiqueta\_a\_buscar"**



```
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (main)
$ git grep Iron
2_Ejemplo2.html:    
2_Ejemplo2.html:    
README.md:
css/styles.css:    background-image: url(..../img/Iron\ man.jpeg);

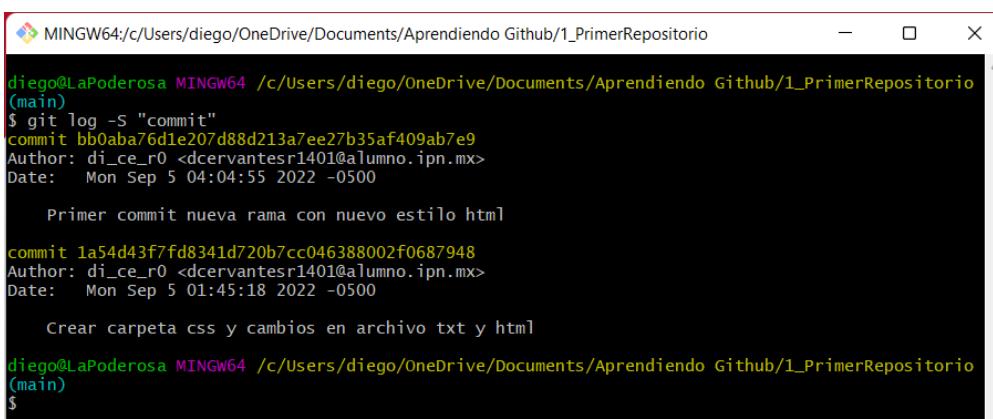
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (main)
$ git grep -n color
README.md:4:<strong style="color: #1c4175;">Primer</strong>
README.md:6:<strong style="color: #0579C2;">para&nbsp;</strong>
README.md:7:<span style="color: #f47a8d;">refrescar conocimientos de </span>
README.md:8:<span style="color: #FFCC84;">Git</span>
css/styles.css:4:    color:gray;
css/styles.css:11:    color: tomato;
css/styles.css:14:    background-color: rgba(0, 0, 0, 0.5);
css/styles.css:15:    color: white;
css/styles.css:21:    color: whitesmoke;
css/styles.css:36:    background-color: #08419c;
css/styles.css:39:    color: rgb(189, 187, 187);

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (main)
$ git grep -c color
README.md:4
css/styles.css:7

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (main)
$ git grep -c "<p>"
2_Ejemplo2.html:1

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (main)
$ |
```

33. **git log -S "etiqueta\_a\_buscar"**: Comando que sirve para buscar cierta palabra usada en alguno de los commits pertenecientes a la línea del tiempo de todas las ramas del proyecto.



```
diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (main)
$ git log -S "commit"
commit bb0aba7d1e207d88d213a7ee27b35af409ab7e9
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date:   Mon Sep 5 04:04:55 2022 -0500

    Primer commit nueva rama con nuevo estilo html

commit 1a54d43f7fd8341d720b7cc046388002f0687948
Author: di_ce_r0 <dcervantesr1401@alumno.ipn.mx>
Date:   Mon Sep 5 01:45:18 2022 -0500

    Crear carpeta css y cambios en archivo txt y html

diego@LaPoderosa MINGW64 /c/Users/diego/OneDrive/Documents/Aprendiendo Github/1_PrimerRepositorio (main)
$ |
```

34.

