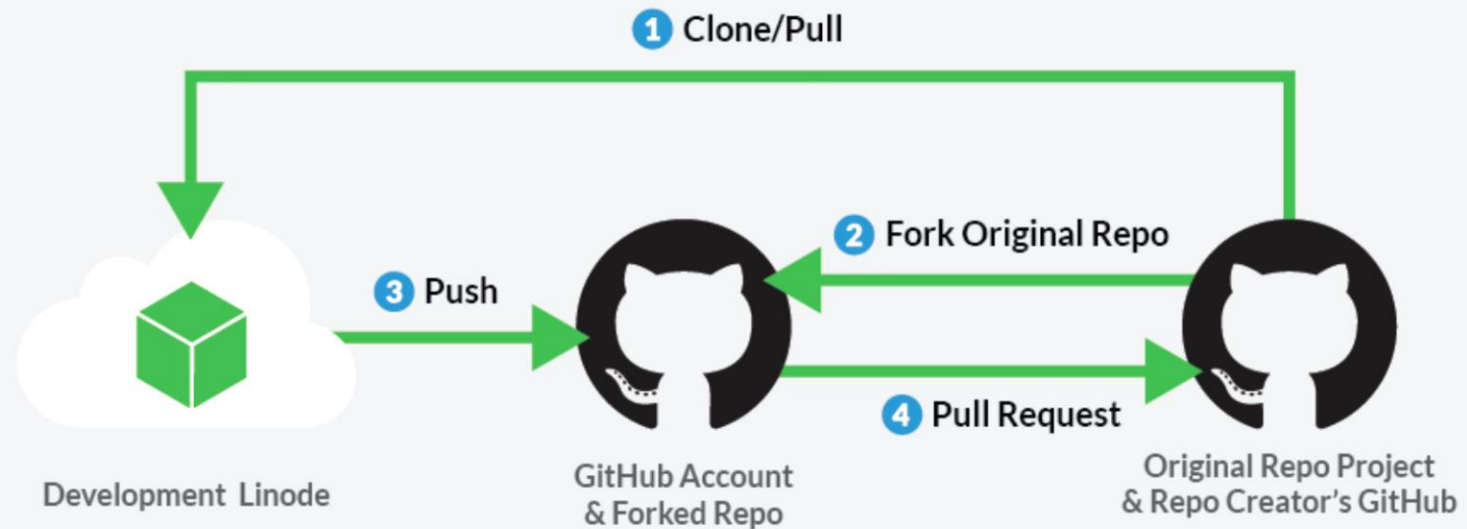
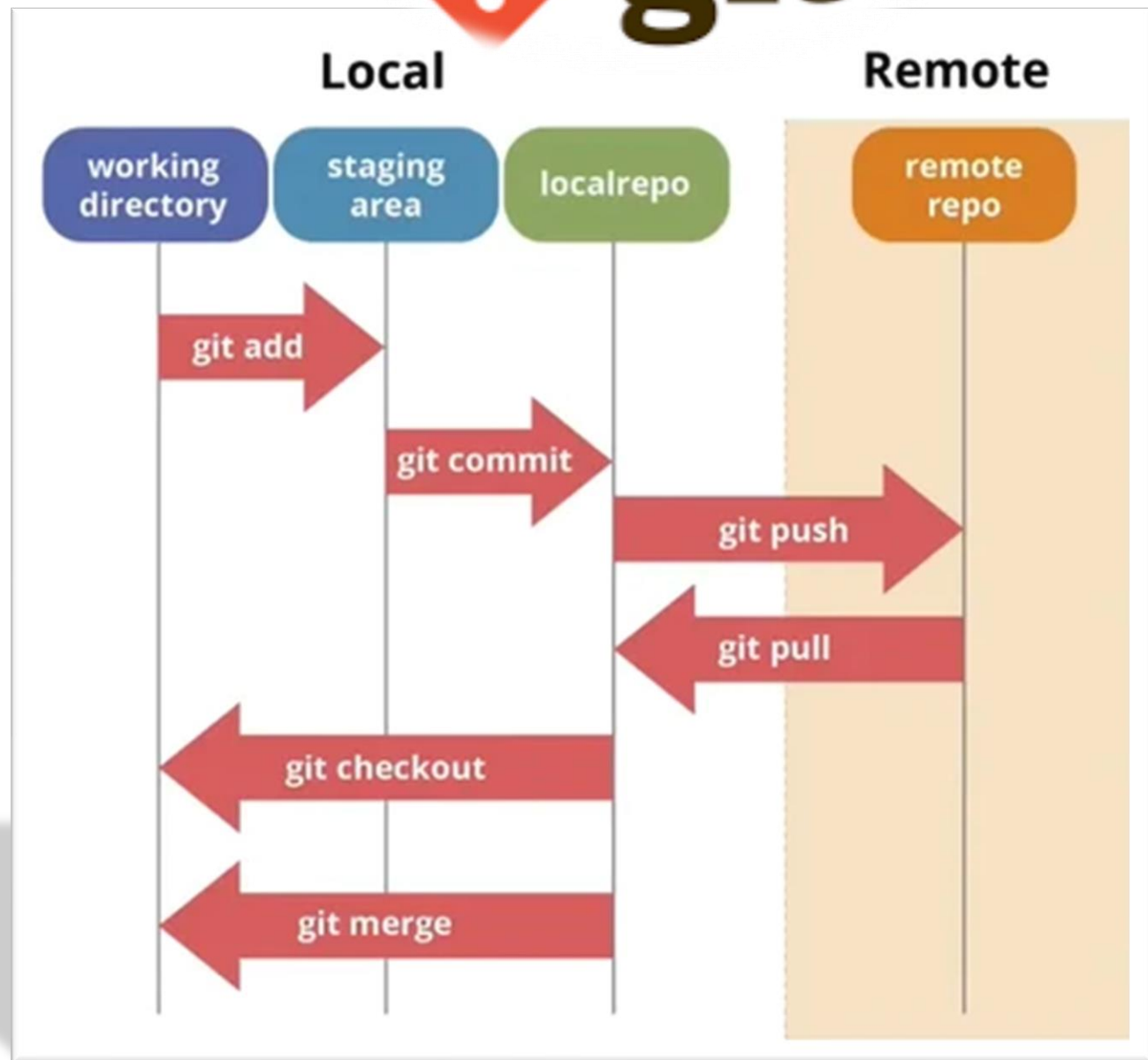




Git and GitHub








Abrir el desarrollador de códigos Atom



- Crear en el escritorio una carpeta de nombre **proyecto2**
 - Arrastrar la carpeta **proyecto2** al desarrollador de código Atom
 - Dentro de la carpeta copiar o crear archivos de cualquier extensión. (app.js , index.html,etc)
 - Abrir el **Git Bash**
- 
- The Git Bash logo, featuring a colorful diamond shape with a white branching diagram inside, and the text "Git Bash" written in white below it.
- Para que nos muestre dónde, en qué parte de nuestra pc estamos ubicados escribir **pwd**.
 - Para ubicarnos en el escritorio escribir **cd Desktop**
 - Para ubicarnos en la carpeta **proyecto2** , escribir **cd proyecto2**
 - Para ver cuántos archivos hay en esa carpeta escribir **ls**



AHORA VAMOS A EMPEZAR A UTILIZAR GIT PARA CONTROLAR CADA VERSIÓN MODIFICADA/ACTUALIZADA

LOCAL:

- **Working directory:**
 - `git init` : iniciamos, se crea una carpeta **.git** se va a encargar de administrar todos los códigos (carpeta oculta, no se debe tocar)
 - `git status`: permite ver los archivos que estamos trabajando
 - `git add` : para agregar los archivos y dale seguimiento o añadir en el archivo de trabajo, en este caso agregamos con **git add** .
 - `git status`: veremos nuevamente el estado de los archivos, ahora aparecerá en verde señalando que ya estamos dando seguimiento, ya se añadió al área de trabajo



LOCAL:

- **Staging área:**

- **Hacer modificaciones** Dentro de los archivos creados [escribir **console.log("hello word");** Texto html,etc)]
- git status
- git add .

- **Repository:**

- git commit: presionar la letra i y escribir "Mi primer commit" , luego ESC y copiar : **wq** o simplificar lo anterior mencionado con **git commit -m "Mi primer commit"**
- git log: muestra los Head que es una manera de diferenciar los commit



LOCAL:

- **Staging área:**

- git status , mostrará que ya los cambios están hechos y ya no hay nada para commitear
- **Hacer nuevas modificaciones** Dentro de los archivos creados o crea nuevos archivos
- git status
- git add .

- **Repository:**

- **git commit -m "Mi segundo commit"**
- git log
- git status



LOCAL:

Staging área:

Ahora supongamos que queremos descartar modificaciones hechas (git checkout) :

- Entonces modificar el archivo **index.html** (hacer HTML, enter y salen estructuras de códigos, dentro de ellas, en tittle escribimos **Mi primera página**).
- git status, observamos que muestra que index.html ha sido modificado
- git checkout -- **index.html**, observamos que se deshacen/revierten los cambios hechos recientemente
- git status, nos dice nada qué commitear,

pero no podemos volver a ese cambio realizado o reversión ya que no se han guardado los cambios, así que ahora procedamos a hacer los cambios y a guardar

- Una vez más modificar el archivo **index.html**, en tittle escribimos **Mi primera página**.



LOCAL:

- `git status`, nos muestra los cambios hechos

Ahora, queremos ver las diferencias en los cambios hechos:

- `git diff index.html`, nos muestra que antes solo había **Texto html** y que ahora están las estructuras de ese html más el tittle que agregamos (mi primera página)

Ahora hay que agregar los cambios para luego guardarlos

- `git add .`
- `git status`, vemos que ya se agregaron

• Repository:

Ahora a guardar los cambios

- `git commit -m "Mi tercer commit"`
- `git log`, nos muestra todos los commits con códigos guardados



LOCAL:

- `git status`, nos dice nada qué commitear

- **Staging área:**

Ahora, qué pasa si queremos agregar archivos o carpetas o textos que queremos ignorar (que solo tú lo veas, nadie más de los colaboradores o espectadores):

Entonces creamos una carpeta **test** y dentro de ella un archivo cualquiera **texto.js**

- `git status`, nos va a mostrar que hay un archivo que agregar, pero no queremos que se agregue

Entonces en el proyecto2 creamos un archivo **.gitignore** (como una especie de archivo donde le vamos a pedir que se guarden todos los archivos que se quieren ignorar o no agregar) dentro de este archivo escribimos el documento o archivo o carpeta que queremos ignorar **test.**, guardamos y luego

- `git status`, observamos que ya no aparece la carpeta **test** pero **.gitignore** si aparece como archivo que se debe agregar, este si lo podemos agregar porque contiene el listado de archivos que quiero ignorar.



LOCAL:

- git add .
- git status, nos muestra que se agregó el **.gitignore** mas no nos muestra **test**.

Podemos hacer modificaciones dentro de **.gitignore** y no nos mostrará esos archivos, pero si nos mostrará que se hicieron cambios en **.gitignore**

Entonces creamos una nueva carpeta **prueba2**

En **.gitignore** escribimos el nombre de esa carpeta **prueba2**, guardamos

- git status, nos muestra los cambios hechos en **.gitignore** mas no nos muestra la carpeta que pedimos sea ignorada

Así podemos crear incluso archivos, colocar textos dentro de los archivos y colocarlos en **.gitignore** para que no se muestren o agreguen

- git add . , para agregar **.gitignore**



LOCAL:

- **Repository:**

- `git commit -m "he agregado un .gitignore"`

Hasta allí ya tenemos una versión creada, un proyecto con todos los cambios de códigos, archivos, etc, entonces si escribimos:

- `git branch`, nos muestra una carpeta **master** (el master, el proyecto2)

AHORA SUPONGAMOS QUE QUEREMOS CREAR UNA MINIVERSIÓN, UNA VERSIÓN ALTERNATIVA DE LO QUE YA TENEMOS, UNA ESPECIE DE LOGIN

- `git branch login` (login es el nombre que le estamos poniendo a la versión alternativa que estamos creando)
- `git branch`, nos muestra que tenemos el **master** y **login**

Ahora supongamos que queremos movernos en la versión alternativa **login**

- `git checkout login`, me dice que he cambiado a la rama **login**



LOCAL:

- `git Branch`, indica que estamos dentro de la rama **login**

Importante: Otra alternativa para movernos a otra rama sin usar ‘`git branch login`’ es: **`git switch -c login`** (creará, retendrá las confirmaciones que se crearon y nos moverá directamente a la rama **login**)

Para deshacer el cambio (regresar al master) bastará con escribir: **`git switch -`**

- **Staging área:**

Entonces ahora podemos hacer cambios como agregar más archivos, por ejemplo carpeta **login**, carpeta **authentication** (dentro de esta agregar un archivo **index.js**), un archivo **texto.txt**, etc

- `git status`, nos muestra todos los cambios que se esta empezando a hacer y que nos dice que de debemos agregar
- `git add .`
- `git status`, observamos que se han guardado los archivos a excepción de la carpeta vacía (login) que git las ignora

- **Repository:**

- `git commit -m "versión alternativa con un login"`



LOCAL:

- git log, tenemos ahora las fotos múltiples, desde las antiguas hasta las recientes, pero esto en la rama **login**

Ahora cambiemos a la rama **master**

- git branch
- git checkout master, vemos que no se observan los archivos últimos agregados
- git log, vemos que no tenemos es último commit porque no le pertenece sino a la versión **login**

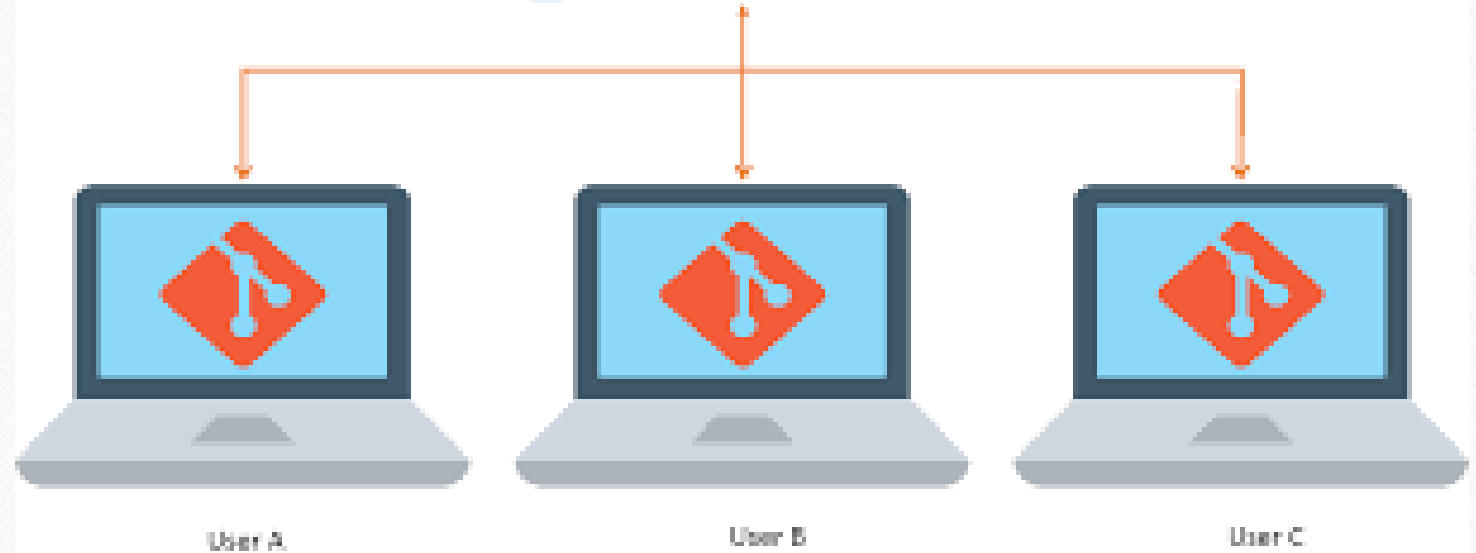
HASTA AQUÍ TENEMOS NUESTROS ARCHIVOS CON LAS MODIFICACIONES EN UN REPOSITORIO LOCAL, AHORA PASEMOS A UN REPOSITORIO REMOTO, DONDE PODAMOS SUBIR NUESTROS CODIGOS, COMPARTIR PARA QUE OTROS DESARROLLADORES COLABOREN, PARA ELLO VAMOS A **GITHUB** QUE TIENE MUCHAS HERRAMIENTAS PARA DESARROLLADORES, UNO DE ELLOS EL REPOSITORIO DE CÓDIGO, UNA PLATAFORMA WEB, DONDE SUBIREMOS NUESTROS CODIGOS WEB OPEN SOURCE, DE FORMA GRATUITA.



GitHub



GitHub





REMOTE:

Repository:

Vamos a crear nuestro repositorio en nuestra cuenta de GITHUB:

The screenshot shows the GitHub web interface for a user named DeniseRosalyn. The browser's address bar shows the URL `github.com/DeniseRosalyn?tab=repositories`, which is circled in red. Below the browser, the GitHub navigation bar includes a search bar and links for Pull requests, Issues, Marketplace, and Explore. The user's profile section on the left features a teal and white pixelated avatar. The main content area has tabs for Overview, Repositories (which is selected and circled in red), Projects, and Packages. A 'ProTip!' banner is visible. At the bottom, there is a search bar for repositories, filters for Type and Language, and a green 'New' button with a repository icon, which is also circled in red.



REMOTE:

Repository:

Le ponemos un nombre, GITHUB verifica si el nombre no se repite con un check:

Create a new repository

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

Owner *



DeniseRosalyn ▼

Repository name *

Mi proyecto ✓

Great repository names are short and simple. Your new repository will be created as Mi-proyecto. But s





REMOTE:

Repository:

En descripción le ponemos **Mi primer proyecto en GIT**

Description (optional)

Mi primer proyecto en GIT

- ☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**
You choose who can see and commit to this repository.

Dejamos marcado en **Public** (gratuita)

Clic en :

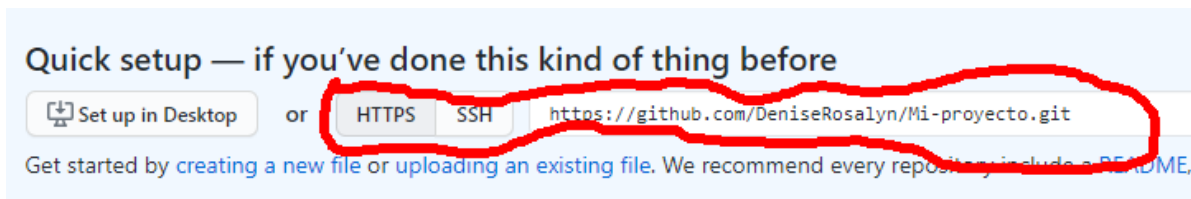
Create repository



REMOTE:

Repository:

Ahora ya tenemos nuestro código de internet:



Además nos entrega un conjunto de códigos, pero los importantes son los 2 últimos, los que vamos a copiar simplemente en GIT

...or create a new repository on the command line

```
echo "# Mi-proyecto" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/DeniseRosalyn/Mi-proyecto.git
git push -u origin main
```



LOCAL:

GIT

Con esos códigos le vamos a decir al proyecto donde vamos a almacenar el código (en el repositorio web GITHUB)

- git remote add origin <https://github.com/DeniseRosalyn/Mi-proyecto.git>

```
Denise@Denise MINGW64 ~/Desktop/prueba (master)
$ git remote add origin https://github.com/DeniseRosalyn/Mi-proyecto.git
```

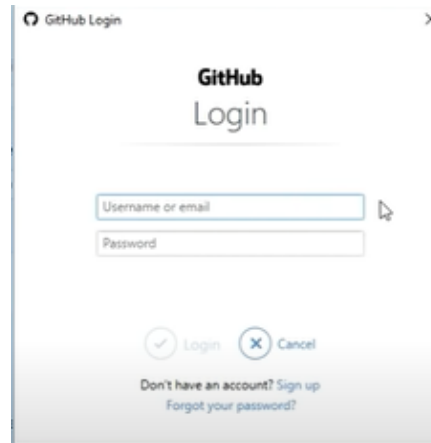
- git push -u origin master

```
Denise@Denise MINGW64 ~/Desktop/prueba (master)
$ git push -u origin master
```



REMOTE:

Lo que va a suceder es que se va a abrir una Ventana de GitHub pidiendo nuestra cuenta/usuario y contraseña (LOGIN)



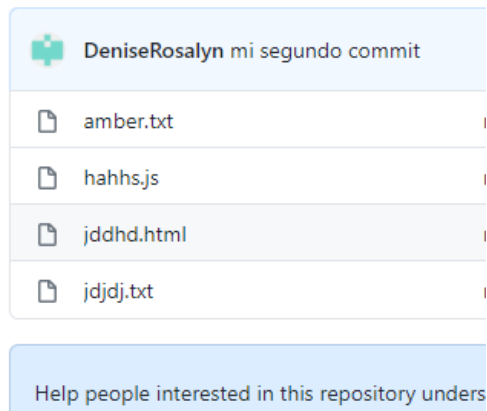
Ingresamos y entonces observaremos en el **GITbash** que empieza a subir nuestros códigos,

```
Denise@Denise MINGW64 ~/Desktop/prueba (master)
$ git push -u origin master
Logon failed, use ctrl+c to cancel basic credential prompt.
error: unable to read askpass response from 'C:/Program Files/Git/mingw64/libexe
c/git-core/git-gui--askpass'
Username for 'https://github.com':
remote: No anonymous write access.
fatal: Authentication failed for 'https://github.com/DeniseRosalyn/Mi-proyecto.g
it/'
```



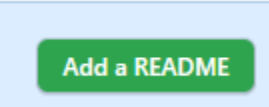
REMOTE:

Actualizamos nuestro repositorio de Código GitHub y veremos que se han subido nuestros códigos en archivos por separado.



Podemos agregar un archivo README (una especie de archivo readme que permite describir nuestro proyecto)

Damos clic en

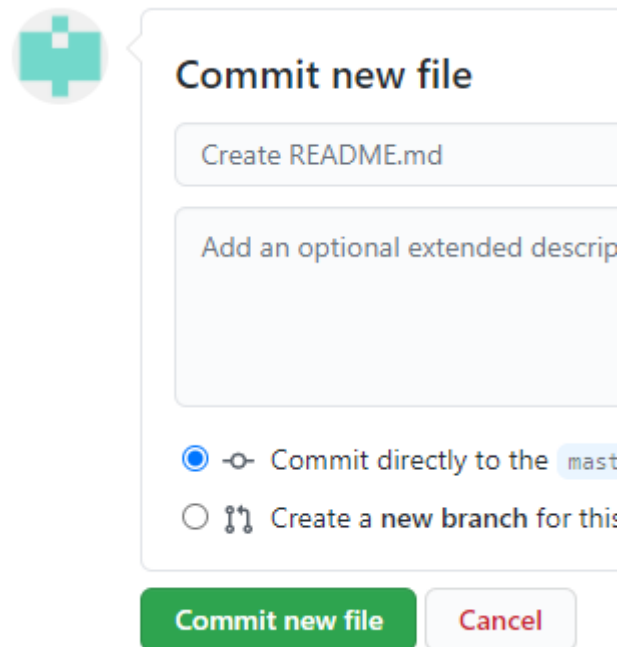




REMOTE:

Y podemos escribir allí **Este es un proyecto de prueba con GIT.**

Luego podemos escribir o crear un commit pero esta vez daremos clic en **Commit new file:**

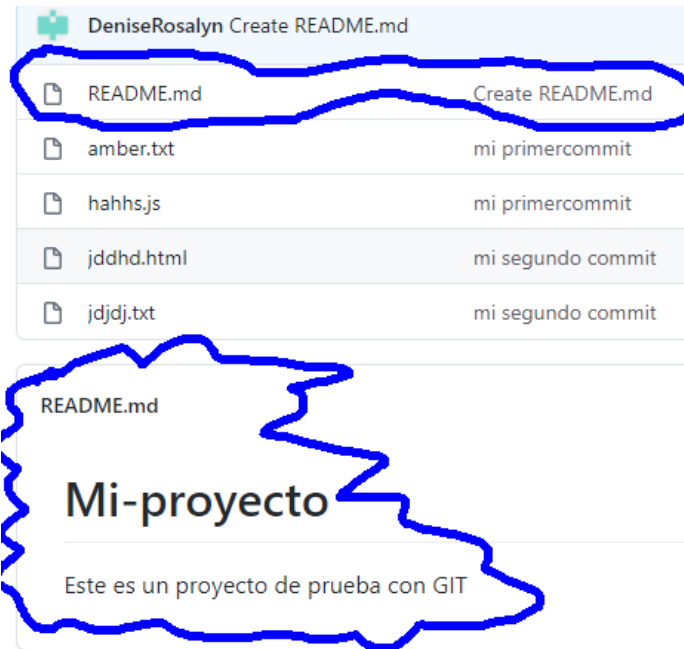


The image shows a GitHub 'Commit new file' dialog box. It features a teal plus icon in a circle on the left. The dialog has a title 'Commit new file'. Below the title, there is a button labeled 'Create README.md'. Underneath that is a text area with the placeholder text 'Add an optional extended description'. At the bottom, there are two radio button options: the first is selected and labeled 'Commit directly to the mast' (with 'mast' in a light blue box), and the second is labeled 'Create a new branch for this'. At the very bottom, there are two buttons: a green 'Commit new file' button and a white 'Cancel' button with a red border.



REMOTE:

Visualizamos ahora que se ha agregado un archivo README.md que no afecta en nada al proyecto, pero ahora se observa que se tiene una descripción del proyecto:



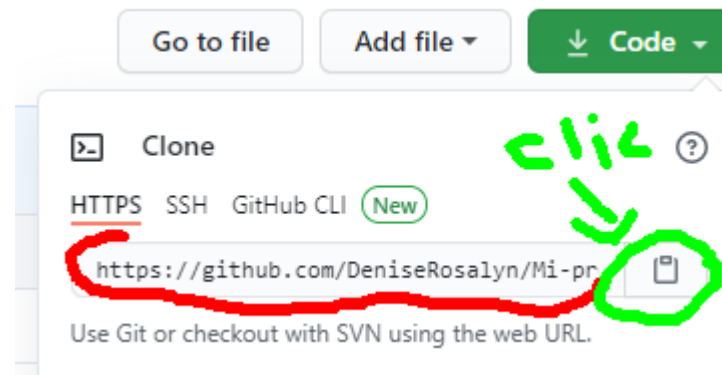


REMOTE:

También podemos clonar o descargar un archivo del repositorio de la siguiente manera:

(supongamos que se eliminó nuestro proyecto de nuestro escritorio o nuestro repositorio local, pero tenemos la ventaja de que está en el repositorio web, así que descarguémoslo desde allí)

Lo podemos hacer descargando o copiando el enlace:



Abrimos el **GitBash**

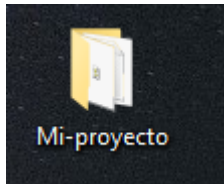
- cd Desktop
- git clone https://github.com/DeniseRosalyn/Mi-proyecto.git

```
Denise@Denise MINGW64 ~/Desktop (master)
$ git clone https://github.com/DeniseRosalyn/Mi-proyecto.git
Cloning into 'Mi-proyecto'...
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 12 (delta 2), reused 8 (delta 1), pack-reused 0
Unpacking objects: 100% (12/12), 1.33 KiB | 7.00 KiB/s, done.
```




GIT LOCAL:

Vemos que se copian nuestros archivos del proyecto en el escritorio de nuevo (vemos la carpeta del proyecto)



Podemos entrar de nuevo al proyecto para revisar

- `cd Mi-proyecto`
- `git log`, observamos que allí están todos los commits

Y así con todo lo realizado en las diapositivas es como puedes empezar a crear nuevos proyectos, ahora está en ti buscar más información sobre más comandos!!

BYE