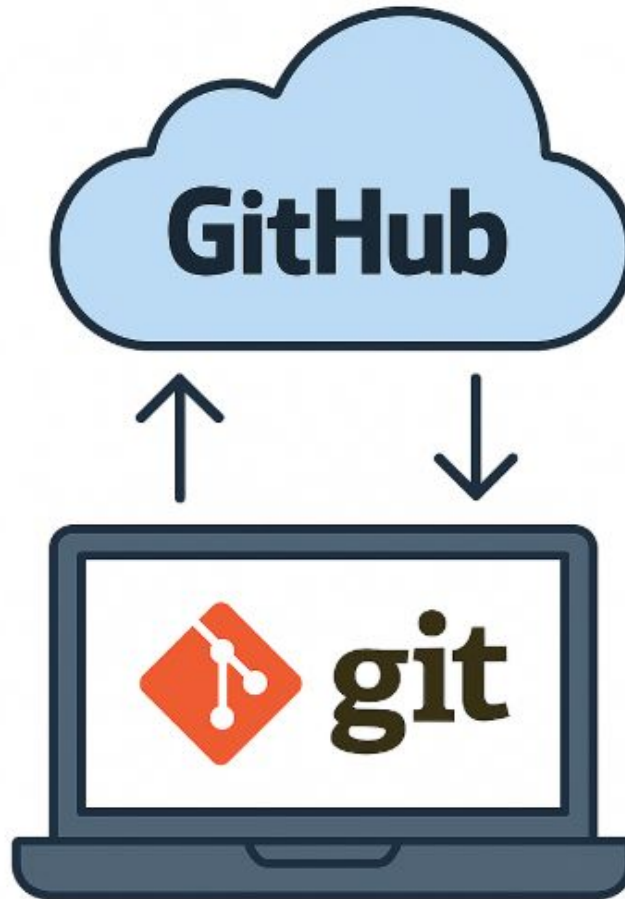


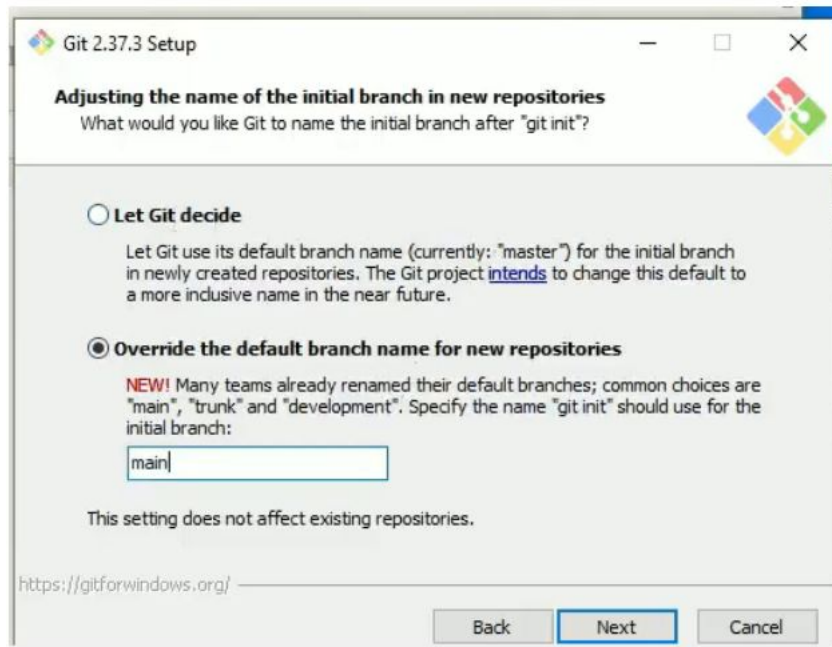
Guía práctica de Git y GitHub

- Instalación y configuración inicial del entorno Git Bash.



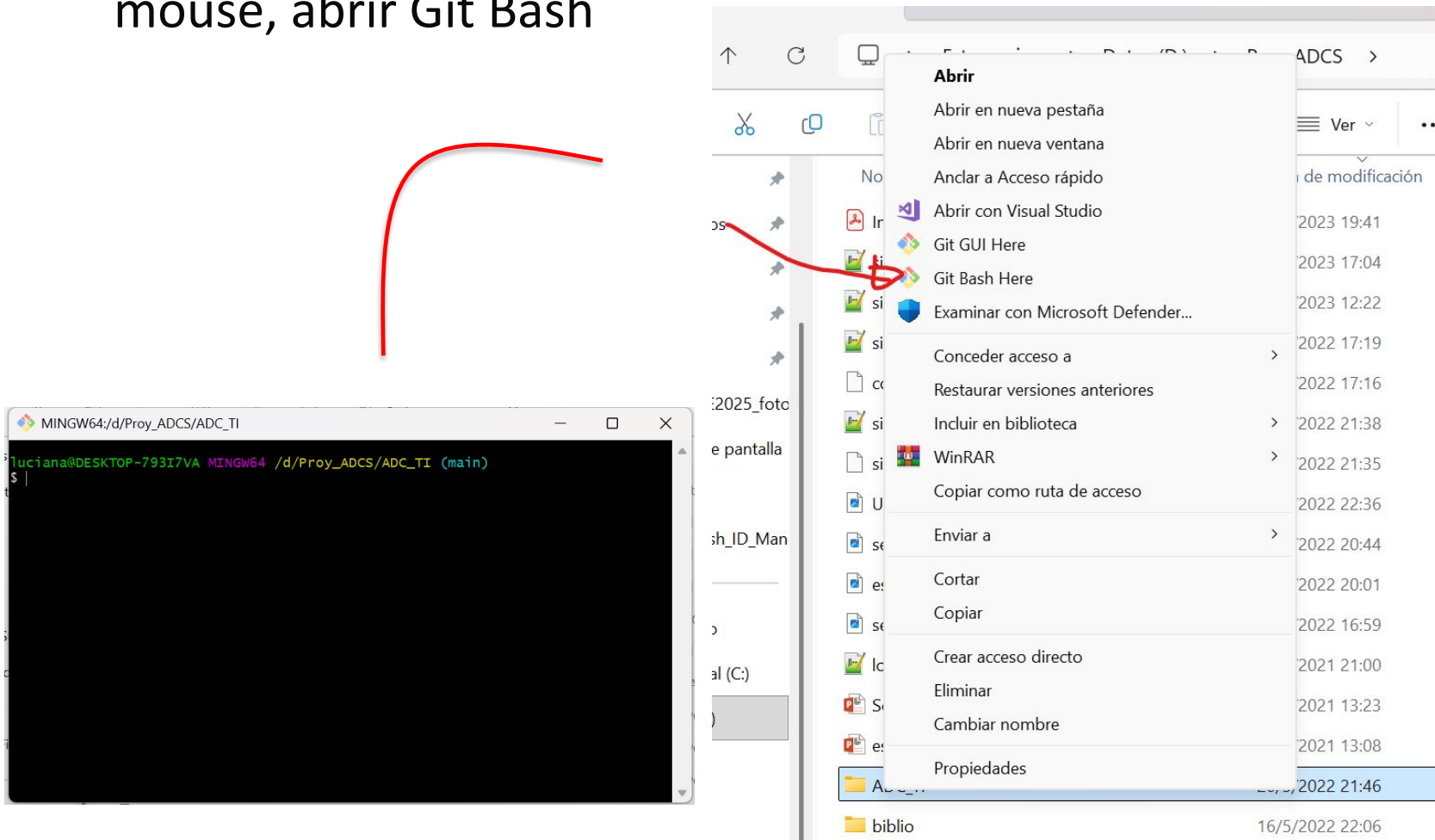
Instalación

- 1. Descargar Git desde git-scm.com/downloads
- 2. Instalar con las opciones por defecto. Excepto en “adjust the name of the initial branch in new repositories”.



Instalación

- 3. Para usarlo, una vez instalado, pararse en la carpeta donde está el proyecto con el que quiero trabajar. Botón derecho del mouse, abrir Git Bash



Configuración inicial

En Git Bash se configura el usuario, si quiere trabajar otra persona en la misma compu hay que eliminarlo y hacer esto de nuevo:

- Configurar usuario:
- `git config --global user.name "TuNombre"`
- `git config --global user.email "tucorreo@gmail.com"`
- Para eliminar credenciales previas:
- Panel de control > Cuentas > Administrador de credenciales.

=> credenciales de Windows

Ahí elimino la cuenta de github que haya y tengo que desde el gitbash asociar mi usuario y email.

Configuración inicial

En Git Bash se configura el usuario, si quiere trabajar otra persona en la misma compu hay que eliminarlo y hacer esto de nuevo:

- Configurar usuario:
- `git config --global user.name "TuNombre"`

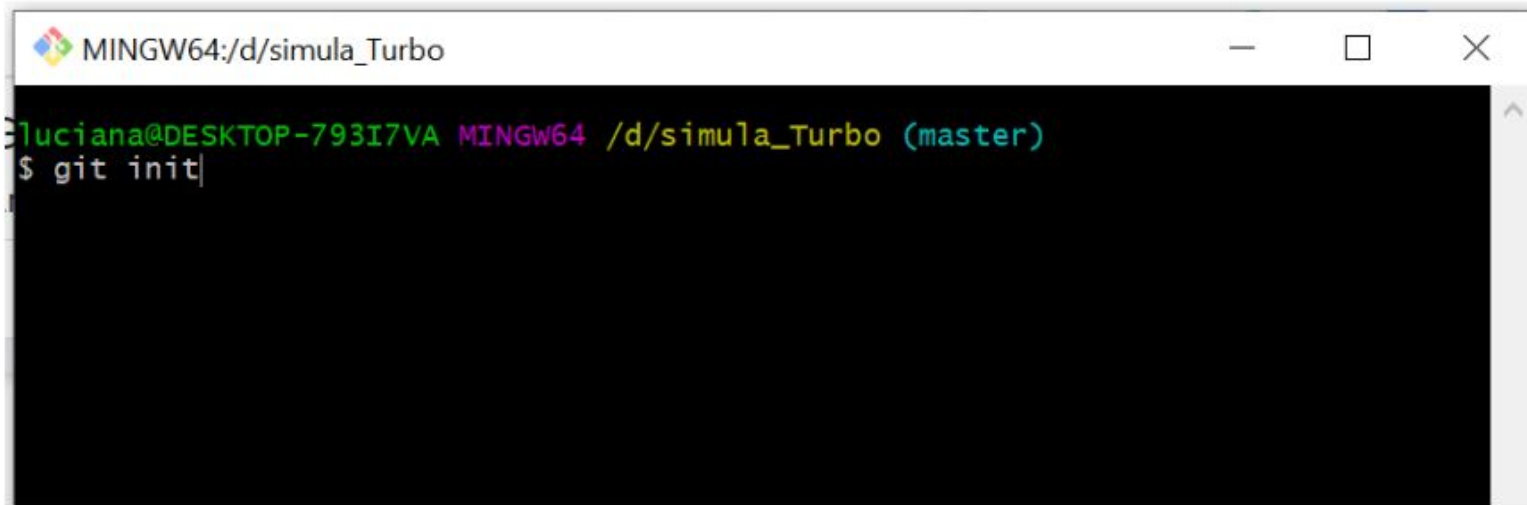
**Tip: Para pegar texto en Git Bash
SHIFT + INSERT (no Ctrl+V).**

=> credenciales de Windows

Ahí elimino la cuenta de github que haya y tengo que desde el gitbash asociar mi usuario y email.

Crear un nuevo proyecto

- 1. Abrir Git Bash en la carpeta del proyecto.
- 2. Inicializar repositorio local con: **git init**



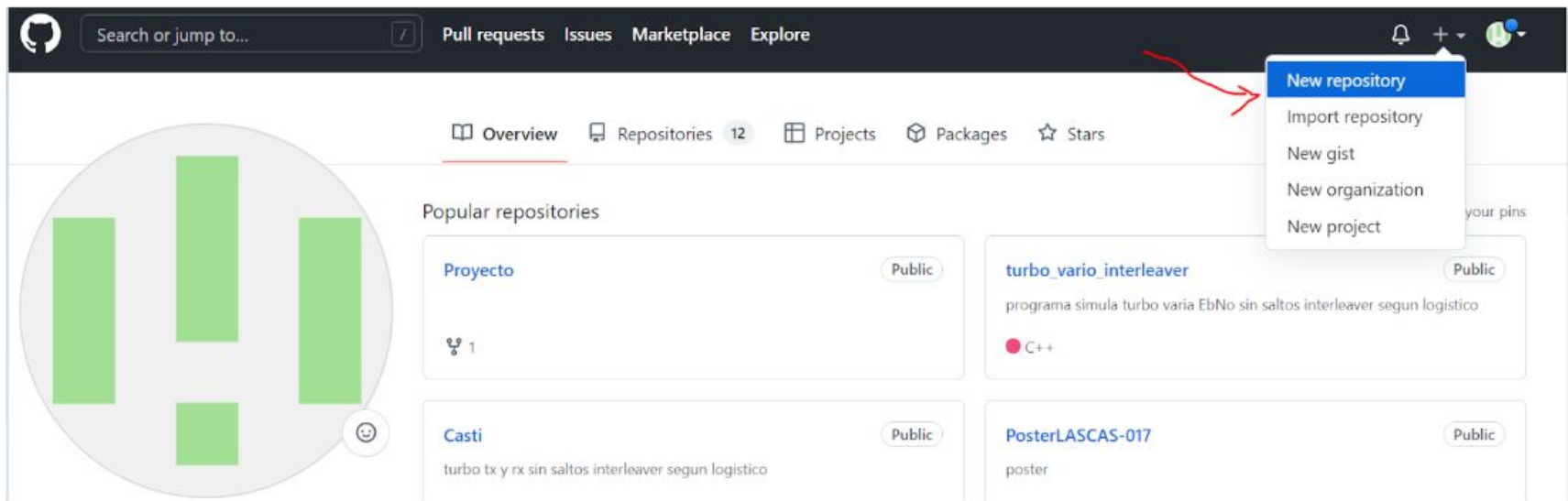
```
MINGW64:/d/simula_Turbo  
luciana@DESKTOP-793I7VA MINGW64 /d/simula_Turbo (master)  
$ git init
```

Esto crea una carpeta oculta .git y deja listo el proyecto para empezar a versionar.

Asociar el proyecto con un repositorio en GitHub

Lo subo a la nube y despues siempre que modifiko algo subo la actualización.

- Entro a Github (<https://github.com/login>) me logueo, saco usuario si no tengo, y voy a la cruz arriba a la derecha:



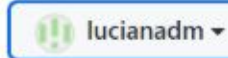
Asociar el proyecto con un repositorio en GitHub

Abre:

Create a new repository

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

Owner *



Repository name *



Great repository names are short and memorable. Need inspiration? How about [curly-octo-winner?](#)

Description (optional)



 **Public**

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



 **Private**

You choose who can see and commit to this repository.

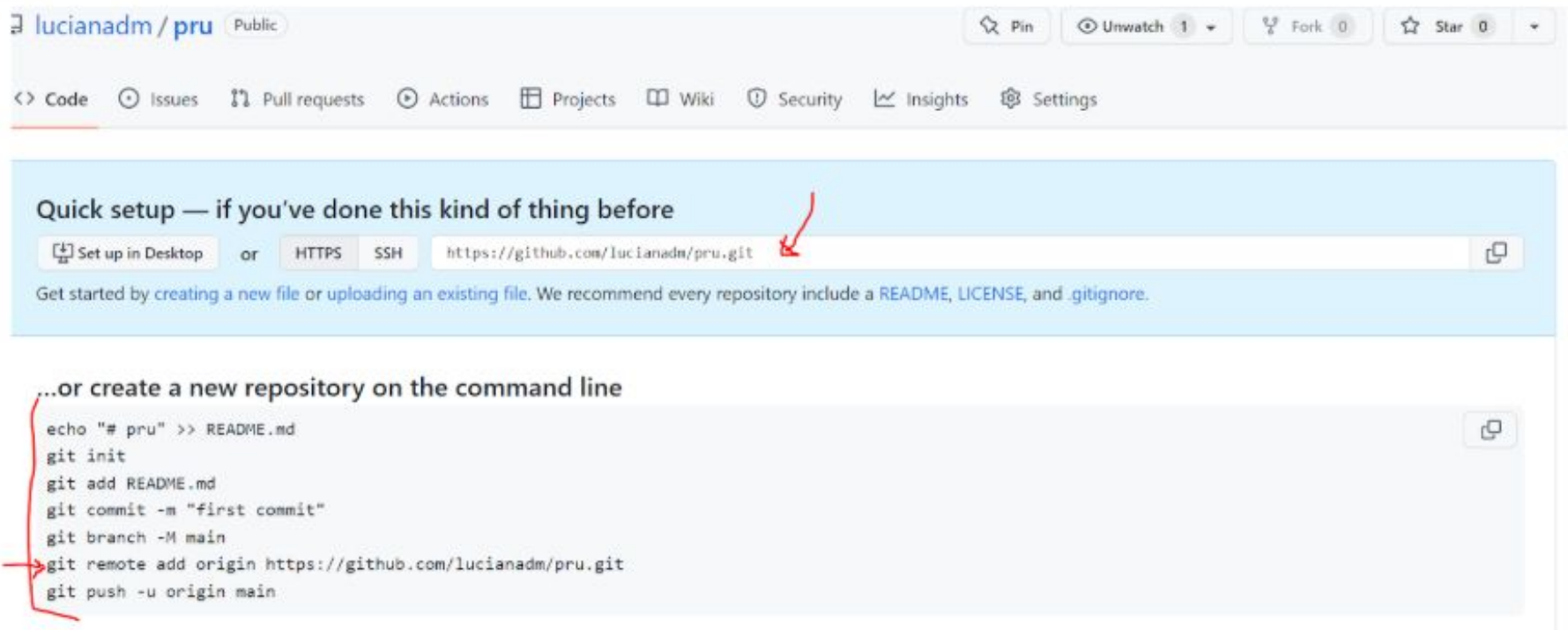
Initialize this repository with:

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

Crear un nuevo repositorio (botón “New Repository”).
Elijo el nombre, el resto se puede dejar como esta.

Asociar el proyecto con un repositorio en GitHub

- Copiar la direccion (flecha roja arriba) que vamos a usar en el GitBash para asociarlo con el proyecto que tenemos en la compu



The screenshot shows the GitHub interface for a repository named 'lucianadm/pru'. The repository is public. The 'Quick setup' section is highlighted in light blue. It contains a text input field with the URL 'https://github.com/lucianadm/pru.git', which is pointed to by a red arrow. Below this, there is a section titled '...or create a new repository on the command line' with a list of git commands. A red bracket and arrow point to the 'git remote add' command in the list.

lucianadm / pru Public

Pin Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH `https://github.com/lucianadm/pru.git`

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

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

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

Asociar el proyecto con un repositorio en GitHub

- Copiar la direccion (flecha roja arriba) que vamos a usar en el GitBash para asociarlo con el proyecto que tenemos en la compu

Ahora de nuevo en el Gitbash (la compu):

git remote add origin <https://github.com/lucianadm/pru.git> ⇒
esto se hace una unica vez y ya queda asociado el github con el proyecto.(origin es el repositorio remoto)

Si por alguna razon lo asocie con una direccion mal, o quiero cambiar el repositorio (del github) con el que esta asociado mi carpeta de la PC:

Código para cambiar de directorio desde GitBash:

git remote set-url origin NUEVA_URL_DEL_REPOSITORIO

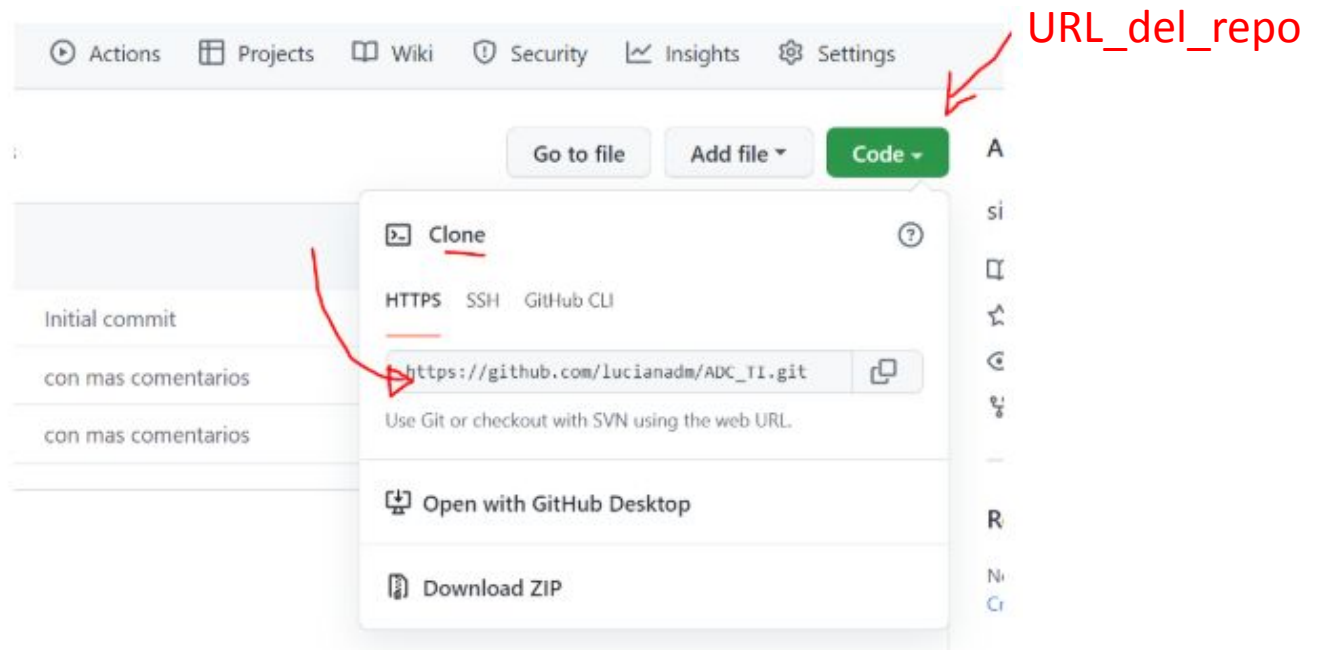
Subir cambios al repositorio

Hago las modificaciones en la carpeta en mi PC, creo archivos, modifico archivos, etc
Despues hago los siguientes comandos:

- `git add .`
- `git commit -m"mensaje"`
- `git push origin main`

Clonar un repositorio

Si quiero bajarme un proyecto, hacer modificaciones y subirlas, tengo que clonar el proyecto (bajarlo a mi pc y que quede asociado).



- `git clone URL_del_repo`

Clonar un repositorio

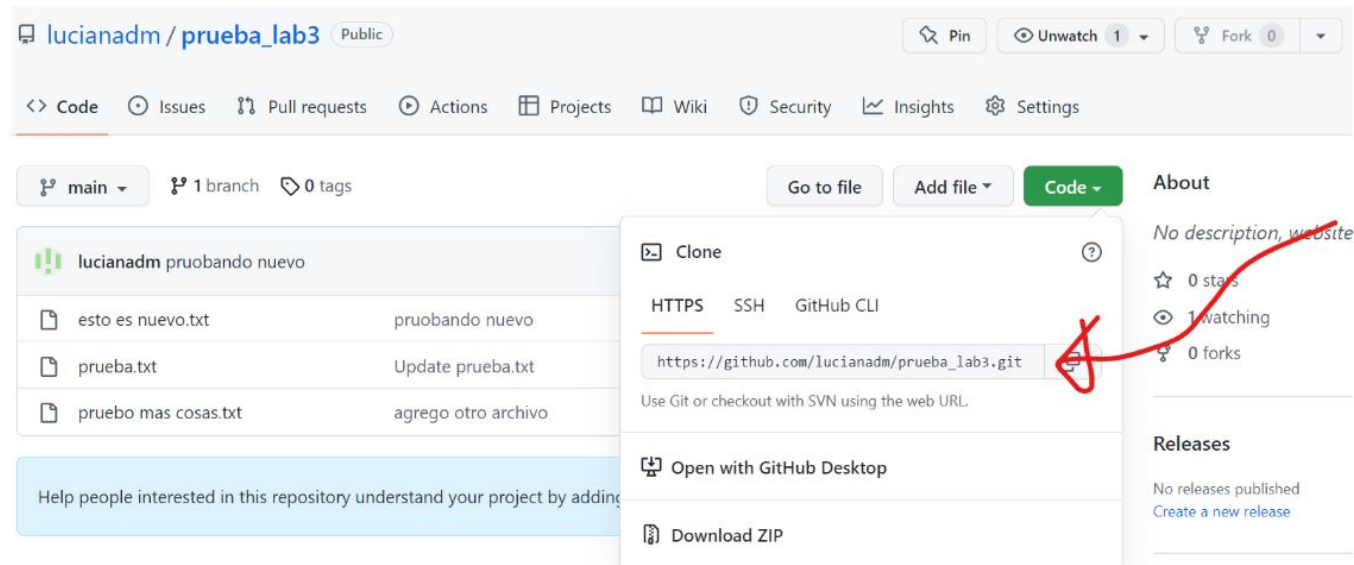
Ejemplo:

Estoy con el gitbash en el disco D:

```
luciana@DESKTOP-793I7VA MINGW64 /D
$
```

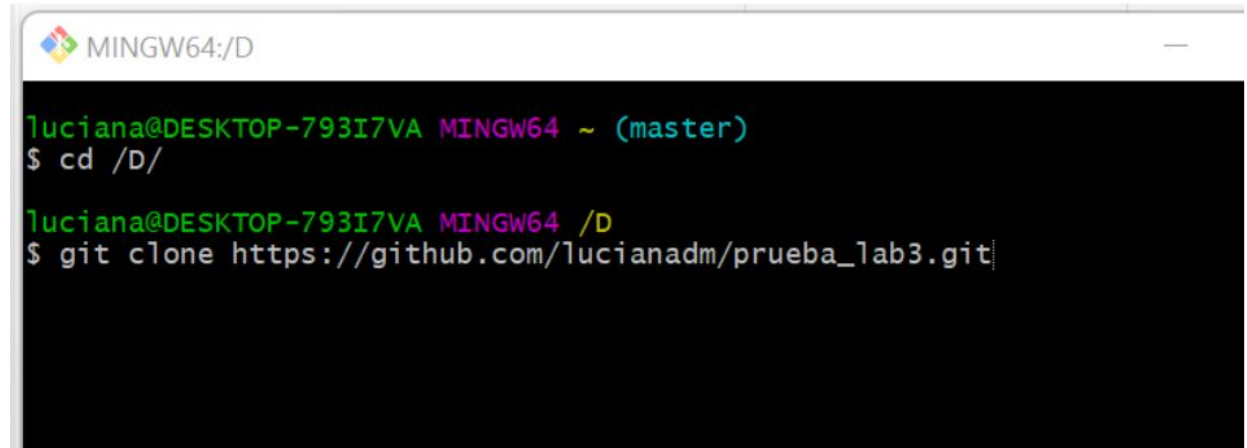
Clono desde github el repositorio Prueba_Lab3

Para ello copio la dirección del github de ese repositorio:



Clonar un repositorio

Ejemplo:
y desde el gitbah:

A screenshot of a Windows terminal window titled 'MINGW64:/D'. The prompt is 'luciana@DESKTOP-793I7VA MINGW64 ~ (master)'. The user enters '\$ cd /D/' and the prompt changes to 'luciana@DESKTOP-793I7VA MINGW64 /D'. Then the user enters '\$ git clone https://github.com/lucianadm/prueba_lab3.git'.

```
MINGW64:/D
luciana@DESKTOP-793I7VA MINGW64 ~ (master)
$ cd /D/
luciana@DESKTOP-793I7VA MINGW64 /D
$ git clone https://github.com/lucianadm/prueba_lab3.git
```

Me crea en el disco D una carpeta que se llama prueba_lab3 donde va a estar el repositorio.

Modifico lo que quiera de esa carpeta.

Ahora me tengo que parar con el gitbash en la carpeta prueba_lab3!!!

y recién ahí hago:

git add .

git commit -m"comento los cambios"

git push origin main

Clonar un repositorio

Ejemplo
y desde el

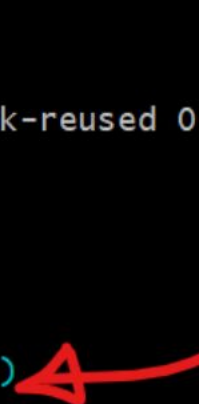
```
luciana@DESKTOP-793I7VA MINGW64 ~ (master)
$ cd /d/

luciana@DESKTOP-793I7VA MINGW64 /d
$ git clone https://github.com/lucianadm/prueba_lab3.git
Cloning into 'prueba_lab3'...
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 14 (delta 1), reused 10 (delta 0), pack-reused 0
Receiving objects: 100% (14/14), done.
Resolving deltas: 100% (1/1), done.

luciana@DESKTOP-793I7VA MINGW64 /d
$ cd /d/prueba_lab3

luciana@DESKTOP-793I7VA MINGW64 /d/prueba_lab3 (main)
$ git add .

luciana@DESKTOP-793I7VA MINGW64 /d/prueba_lab3 (main)
$
```



Me crea en el c
repositorio.

Modifico lo que quiera de esa carpeta.

Ahora me tengo que parar con el gitbash en la carpeta prueba_lab3!!!

y recién ahí hago:

git add .

git commit -m"comento los cambios"

git push origin main

Ignorar archivos

- Para evitar subir ciertos archivos:
- Crear un archivo `.gitignore` y listar ahí los nombres o carpetas a excluir.

Ejemplo:

`apunte.pdf`

`/Carpeta`

No me sube ni el archivo `apunte.pdf` ni la carpeta `Carpeta`, ambos están en la carpeta del repositorio local y no se suben al github.

Trabajo en grupo

- 1. El coordinador crea el repositorio y agrega colaboradores.
- 2. Cada integrante clona el proyecto.
- 3. Cada integrante crea su propia rama.
- 4. Cada uno trabaja en su propia rama:
 - `git branch mi_rama`
 - `git checkout mi_rama`

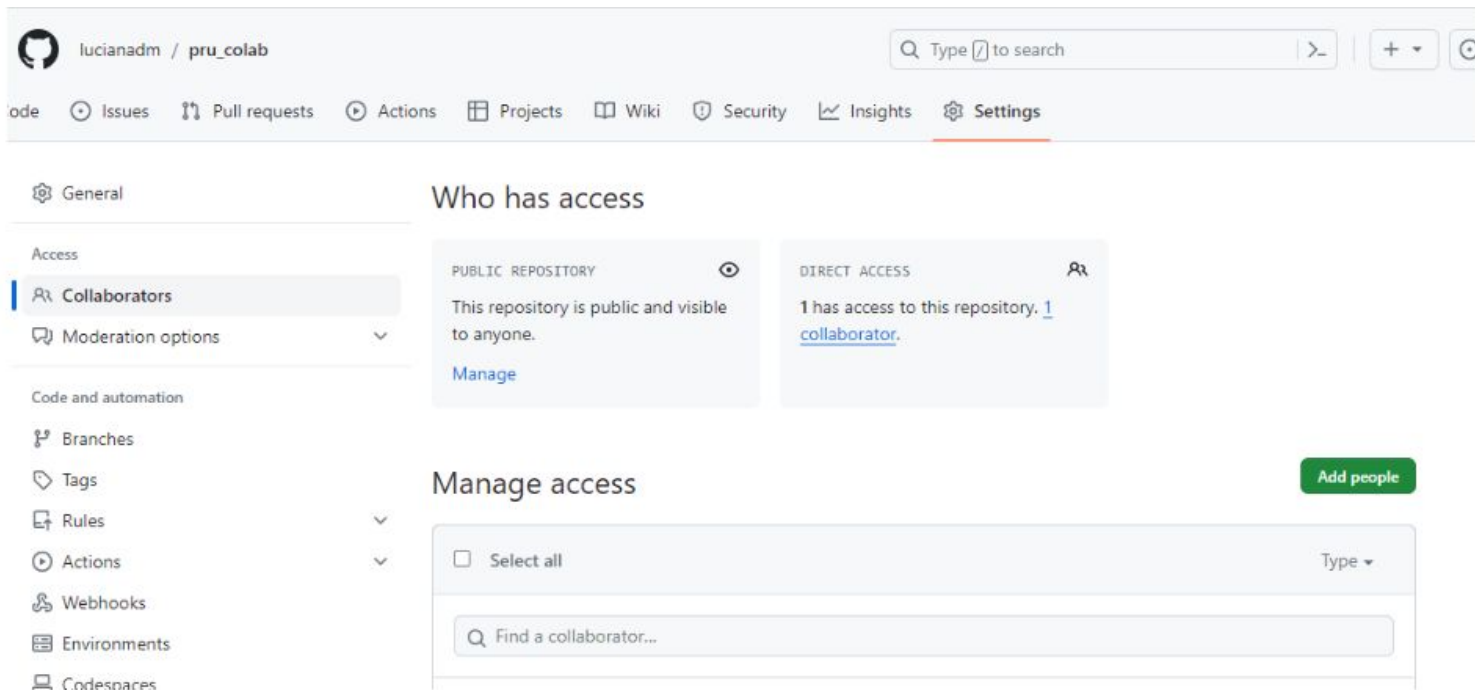
Trabajo en grupo

- 1. El coordinador crea el repositorio y agrega colaboradores.
- 2. Cada integrante clona el proyecto.
- 3. Cada integrante crea su propia rama.
- 4. Cada uno trabaja en su propia rama:
- `git branch mi_rama`
- `git checkout mi_rama`

Trabajo en grupo

- 1. El coordinador crea el repositorio y agrega colaboradores.

En GitHub, y agrega colaboradores en Settings → Collaborators → Add people.



Trabajo en grupo

- 1. El coordinador crea el repositorio y agrega colaboradores.
- 2. Cada integrante clona el proyecto.
- 3. Cada integrante crea su propia rama.
- 4. Cada uno trabaja en su propia rama:
 - git branch mi_rama
 - git checkout mi_rama

Trabajo en grupo

- 3. Cada integrante crea el repositorio y agrega colaboradores.

Genera una rama nueva a partir del proyecto, para no modificar el proyecto principal y desp se pide unir lo que hizo en su rama al proyecto principal. Ejecuta gitbash en la carpeta que descargo.

- **git branch mi_rama**⇒ creo la rama
- **git checkout mi_rama** => me paso a la rama
nombre_de_la_rama para q lo q haga se cargue ahí
- Trabajo en mi compu en el proyecto (que ahora es una rama)
- Cuando termino subo al Github los cambios (a mi rama)

Trabajo en grupo

- 3. Cada integrante crea el repositorio y agrega colaboradores.

Genera una rama nueva a partir del proyecto, para no modificar el proyecto principal y desp se pide unir lo que hizo en su rama al proyecto principal. Ejecuta gitbash en la carpeta que descargo.

- **git bran**

- **git chec**

nombre

- Trabajo

- Cuando

```
luciana@DESKTOP-793I7VA MINGW64 /d/pru_branch/TyDD (rama_lu)
$ git add .
luciana@DESKTOP-793I7VA MINGW64 /d/pru_branch/TyDD (rama_lu)
$ git commit -m"agrego a mi rama?"
[rama_lu 078622e] agrego a mi rama?
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 AGREGO_ARCH_LUCIANA.txt
luciana@DESKTOP-793I7VA MINGW64 /d/pru_branch/TyDD (rama_lu)
$ git push origin rama_lu
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 286 bytes | 286.00 KiB/s, done.
Total 3 (delta 1), reused 1 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
```

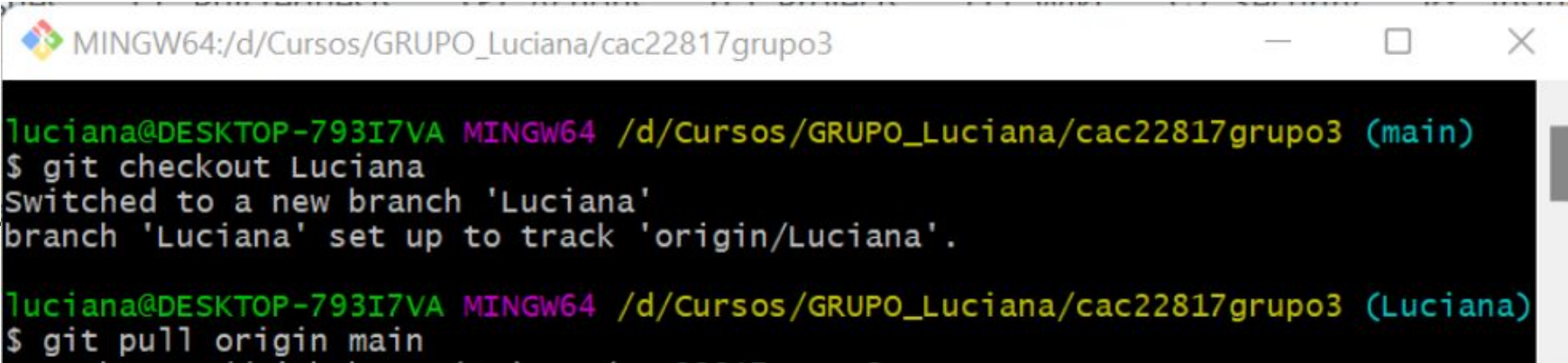
Trabajo en grupo

- 1. El coordinador crea el repositorio y agrega colaboradores.
- 2. Cada integrante clona el proyecto.
- 3. Cada integrante crea su propia rama.
- 4. Cada uno trabaja en su propia rama.

Trabajo en grupo

Cada vez que el colaborador va a trabajar:

1. Se para en su repositorio local (carpeta del proyecto).
2. Se asegura de estar en su rama:
git checkout mi_rama
3. Trae los últimos cambios del repositorio principal (main) y los incorpora en su rama:
git pull origin main

A screenshot of a Windows terminal window with a black background and white text. The title bar at the top shows the path 'MINGW64:/d/Cursos/GRUPO_Luciana/cac22817grupo3' and standard window controls. The terminal shows a user named 'luciana' at a desktop named 'DESKTOP-793I7VA' running 'git checkout Luciana'. The output indicates a new branch 'Luciana' was created and set to track 'origin/Luciana'. The user then runs 'git pull origin main' while on the 'Luciana' branch.

```
4 MINGW64:/d/Cursos/GRUPO_Luciana/cac22817grupo3 (main)
$ git checkout Luciana
Switched to a new branch 'Luciana'
branch 'Luciana' set up to track 'origin/Luciana'.

luciana@DESKTOP-793I7VA MINGW64 /d/Cursos/GRUPO_Luciana/cac22817grupo3 (Luciana)
$ git pull origin main
```


Trabajo en grupo

Cada vez que el colaborador va a trabajar:

1. Se para en su repositorio local (carpeta del proyecto).
2. Se asegura de estar en su rama:

git checkout mi_rama

3. Trae los últimos cambios del repositorio principal (main) y los incorpora en su rama:

git pull origin main

(esto actualiza la rama con los cambios más recientes que otros hayan subido al main)

4. Trabaja en su rama y, cuando termina, sube sus modificaciones:

git add . git commit -m "comentario" git push origin mi_rama

Cuando el colaborador quiere pasar a la rama ppal (Main) los cambios que hizo en su rama

- **git fetch origin** ==> Traé los últimos cambios del repositorio remoto
- **git rebase origin/main** ==> Esto mueve tus commits al “tope” de los commits más recientes de main, integrando los cambios más nuevos de otros desarrolladores antes de fusionar.
- Si hay conflictos, se resuelven con:
 - git status
 - # editás los archivos en conflicto
 - git add <archivo>
 - git rebase --continue
 -
- **git checkout main** ==> Cambiar a la rama principal
- **git pull origin main** ==> Actualizar tu main local con la remota
- **git merge mi_rama** ==> Fusionar tu rama en main
- **git push origin main** ==> Subir los cambios al repositorio remoto

Actualizar rama y fusionar cambios

- Traer cambios del main y fusionar:
- `git fetch origin`
- `git rebase origin/main`
- Resolver conflictos, luego:
- `git checkout main`
- `git merge mi_rama`
- `git push origin main`

Recursos adicionales

- Videos del Prof. Alejandro Zapata:
- <https://www.youtube.com/watch?v=ptXiQwE535s>
- Fundamentos de Git y GitHub:
- <https://bluuweb.github.io/tutorial-github/guia/fundamentos.html>