

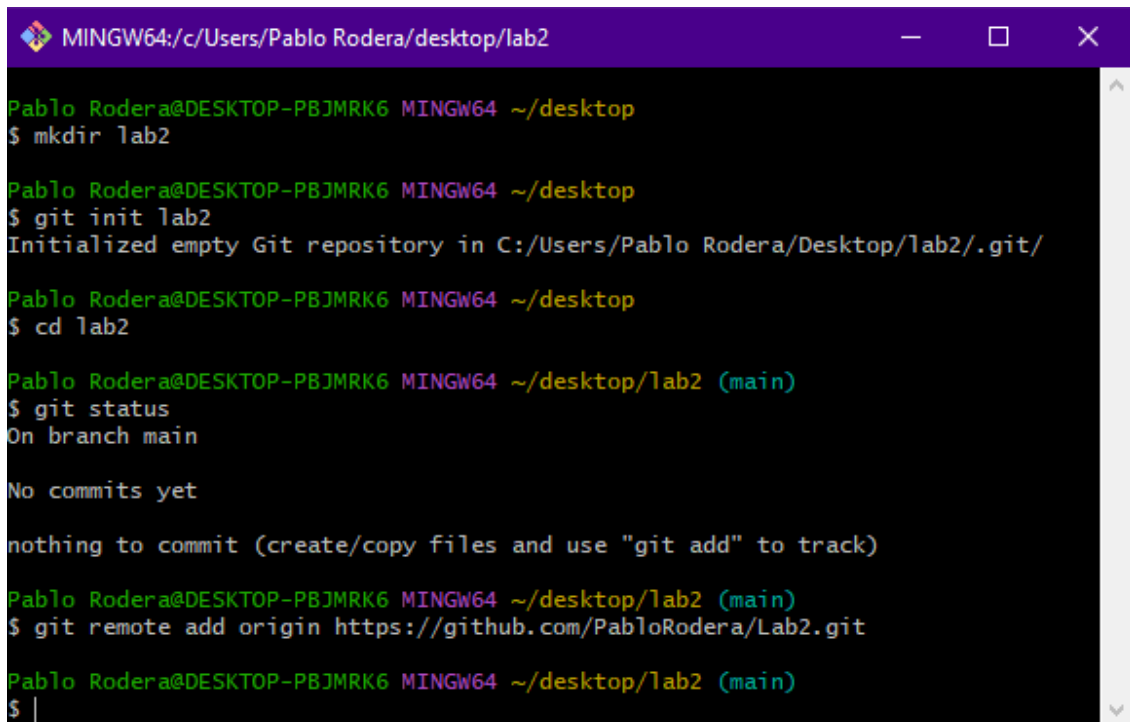
Sprint 1 – Lab 2 Gestión de repositorios Git

[LINK AL REPOSITORIO DE LA PRÁCTICA](#)

[LINK A MI REPOSITORIO DEL BOOTCAMP DONDE REALIZO TODO LO RELACIONADO CON ESTE](#)

(Por si queréis echar un vistazo, ya que lo uso casi diariamente para realizar el bootcamp)

Después de crearnos la cuenta en GitHub y crear un repositorio específico para esta práctica, creamos una carpeta local en la cual iniciamos nuestro repositorio local con el comando **"git init (nombre de la carpeta)"**, después con **"git status"** mostramos el estado actual de la rama y, por último, para enlazar nuestro repositorio local con el remoto, usamos el comando **"git remote add origin (url del repositorio remoto)"**.

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/Pablo Rodera/desktop/lab2'. The terminal shows the following commands and output: 1. 'mkdir lab2' is executed. 2. 'git init lab2' is executed, resulting in 'Initialized empty Git repository in C:/Users/Pablo Rodera/Desktop/lab2/.git/'. 3. 'cd lab2' is executed. 4. 'git status' is executed, showing 'On branch main' and 'No commits yet'. 5. 'git remote add origin https://github.com/PabloRodera/Lab2.git' is executed. The prompt '\$ |' is visible at the bottom.

```
MINGW64:/c/Users/Pablo Rodera/desktop/lab2
Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop
$ mkdir lab2

Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop
$ git init lab2
Initialized empty Git repository in C:/Users/Pablo Rodera/Desktop/lab2/.git/

Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop
$ cd lab2

Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (main)
$ git status
On branch main

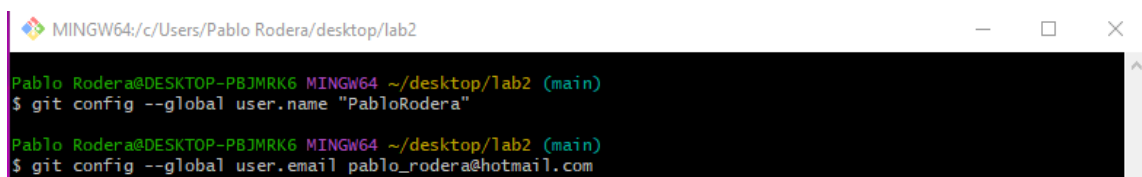
No commits yet

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

Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (main)
$ git remote add origin https://github.com/PabloRodera/Lab2.git

Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (main)
$ |
```

Configuramos nuestro mail de nuestra cuenta de GitHub y nuestro nombre de usuario a nivel global en nuestro sistema para que GitHub nos reconozca y cualquier trabajo que hagamos con Git en nuestro sistema sea bajo nuestro usuario, dando lugar al correcto funcionamiento de Git.

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/Pablo Rodera/desktop/lab2'. The terminal shows the following commands and output: 1. 'git config --global user.name "PabloRodera"' is executed. 2. 'git config --global user.email pablo_rodера@hotmail.com' is executed.

```
MINGW64:/c/Users/Pablo Rodera/desktop/lab2
Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (main)
$ git config --global user.name "PabloRodera"

Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (main)
$ git config --global user.email pablo_rodера@hotmail.com
```

Creamos la nueva rama como se nos pide en el laboratorio:

```
MINGW64:/c/Users/Pablo Rodera/desktop/lab2
Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (main)
$ git checkout -b lab2
Switched to a new branch 'lab2'

Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (lab2)
$ git branch
* lab2
  main

Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (lab2)
$ |
```

Creamos un fichero .txt de prueba en esta nueva rama y realizamos un commit:

```
Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (lab2)
$ touch prueba1ab2.txt

Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (lab2)
$ git status
On branch lab2
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  prueba1ab2.txt

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

Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (lab2)
$ git add .

Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (lab2)
$ git commit -m "Creamos fichero para hacer primer commit en rama lab2"
[lab2 56783ab] Creamos fichero para hacer primer commit en rama lab2
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 prueba1ab2.txt

Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (lab2)
$ |
```

El laboratorio nos pide que eliminemos el commit realizado anteriormente, lo eliminamos y volvemos al estado anterior de la siguiente forma:

```
Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (lab2)
$ git log --oneline
56783ab (HEAD -> lab2) Creamos fichero para hacer primer commit en rama lab2
d9cb8b7 (origin/main, main) prueba
3abd499 Initial commit

Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (lab2)
$ git reset --hard d9cb8b7
HEAD is now at d9cb8b7 prueba

Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (lab2)
$ git log --oneline
d9cb8b7 (HEAD -> lab2, origin/main, main) prueba
3abd499 Initial commit

Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (lab2)
$ |
```

Lo que realizamos arriba básicamente es mostrar el historial de commits (he realizado uno anteriormente para comprobar que todo funcionaba correctamente), y con el hash del commit al que queremos volver, usamos el comando **"git reset --hard (hash del commit AL QUE QUEREMOS VOLVER, NO DEL QUE QUEREMOS ELIMINAR)"** (está erróneo en los apuntes), con el parámetro **--hard** también eliminaremos los cambios a nivel local, si realizamos un ls, veremos que el archivo **prueba1ab2.txt** no está:

```
Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (lab2)
$ ls
LICENSE README.md hola/ prueba

Pablo Rodera@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (lab2)
$ |
```

El laboratorio nos pide que eliminemos la rama que hemos creado tanto a nivel local como remota, para ello, ya que la rama actualmente solo está en local, vamos a empujarla a remoto para poder realizar correctamente el ejercicio, esto lo hacemos con el siguiente comando:

```
Pablo Roder@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (lab2)
$ git push -u origin lab2
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'lab2' on GitHub by visiting:
remote:   https://github.com/PabloRoder@Lab2/pull/new/lab2
remote:
To https://github.com/PabloRoder@Lab2.git
 * [new branch]      lab2 -> lab2
branch 'lab2' set up to track 'origin/lab2'.

Pablo Roder@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (lab2)
$ git branch -a
* lab2
  main
remotes/origin/lab2
remotes/origin/main

Pablo Roder@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (lab2)
$ |
```

Como vemos, con el comando “**git branch -a**” nos muestra tanto las ramas locales como las remotas, indicándonos que se ha empujado al repositorio remoto correctamente.

Ahora procedemos a eliminar esta rama tanto local como remota, cambiaremos a la rama “main”, ya que no podemos eliminar la rama “lab2” si estamos actualmente en ella, primero la eliminaremos de manera local, para ello, usamos el siguiente comando:

```
Pablo Roder@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (lab2)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Pablo Roder@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (main)
$ git branch -d lab2
Deleted branch lab2 (was d9cb8b7).

Pablo Roder@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (main)
$ git branch -a
* main
  remotes/origin/lab2
  remotes/origin/main

Pablo Roder@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (main)
$ |
```

Como podemos observar, se ha eliminado del local, pero no del remoto, para eliminarla del remoto usaremos el siguiente comando:

```
Pablo Roder@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (main)
$ git push origin -d lab2
To https://github.com/PabloRoder@Lab2.git
- [deleted]      lab2

Pablo Roder@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (main)
$ git branch -a
* main
  remotes/origin/main

Pablo Roder@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (main)
$ |
```

Ahora la rama ya está eliminada totalmente.

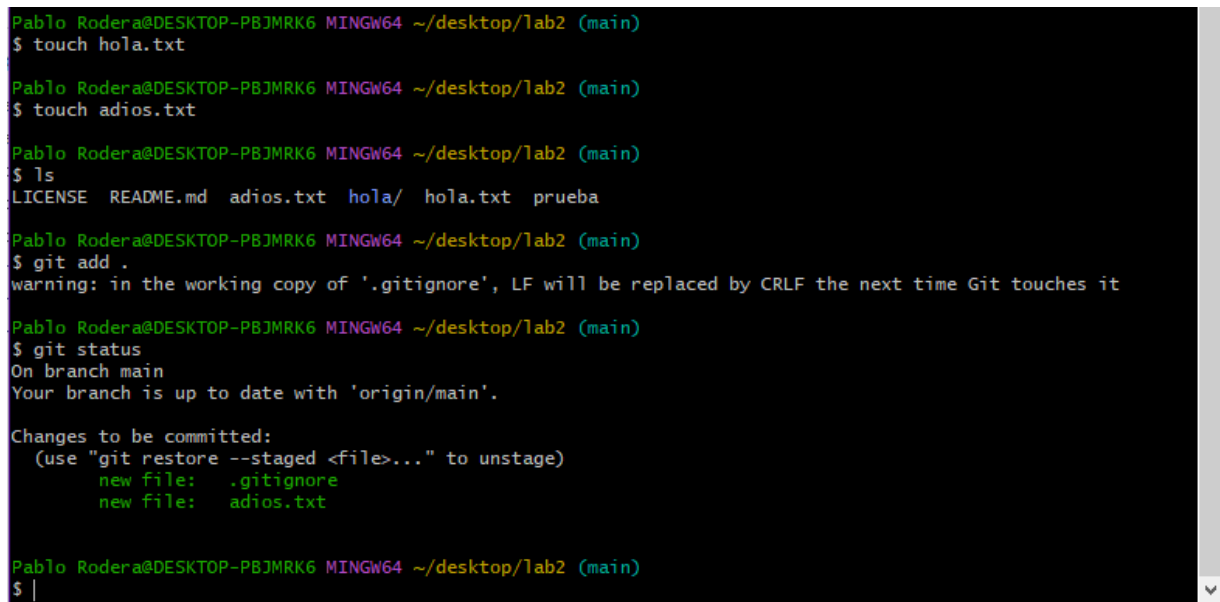
Lo siguiente que nos piden en el laboratorio es configurar nuestro Git para ignorar archivos, esto lo realizaremos mediante el archivo **.gitignore**, un archivo de texto que le dice a Git qué archivos o carpetas ignorar en un proyecto. Generalmente se coloca en el directorio raíz de un proyecto. También puedes crear un archivo global **.gitignore**, y cualquier entrada en ese archivo se ignorará en todos tus repositorios de Git.

Vamos a realizar una prueba sencilla en la que nuestro Git ignore el archivo **“hola.txt”**, para ello editamos nuestro archivo **.gitignore** y ponemos lo siguiente:



```
MINGW64:/c/Users/Pablo Roderer/desktop/lab2
GNU nano 7.2 .gitignore
# ignoramos el archivo hola.txt
hola.txt
```

Ahora cuando hagamos un **“git add .”** se ignorará el archivo **hola.txt**, vamos a realizar una prueba creando este archivo y otro diferente:



```
Pablo Roderer@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (main)
$ touch hola.txt

Pablo Roderer@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (main)
$ touch adios.txt

Pablo Roderer@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (main)
$ ls
LICENSE  README.md  adios.txt  hola/  hola.txt  prueba

Pablo Roderer@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (main)
$ git add .
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it

Pablo Roderer@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (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:   .gitignore
        new file:   adios.txt

Pablo Roderer@DESKTOP-PBJMRK6 MINGW64 ~/desktop/lab2 (main)
$ |
```

Como podemos ver, en el **“git status”** no aparece como cambio pendiente el archivo **“hola.txt”** ya que está siendo ignorado correctamente.

Por último, nos registramos en www.gitkraken.com y lo asociamos al repositorio creado:

