# CURSO PROFESIONAL GIT

¿Qué es GIT?

Es un sistema de control de versiones de lo que vamos trabajando a lo largo de un proyecto, así como una línea de tiempo, tener conocimiento de cómo estaban las cosas para que funcionen bien en las versiones futuras.

¿Dónde puedo usar control de versiones?

Sirve bien a la hora de escribir código sobre todo cuando se trabaja en equipo, aunque también se puede emplear en muchas áreas, funcionando muy bien a la hora de hacer trabajos colaborativos permitiendo tener actualizaciones de forma ordenada y no perder algo valioso durante el desarrollo del proyecto.

GIT en el mundo

Git es usado por las grandes empresas, básicamente, es la herramienta que permite que muchas de las aplicaciones que usas hoy en día como lo son Facebook, Twitter, Netflix, Google, Likedln. También tiene presencia en los equipos de trabajo que desarrollan sistemas operativos para los dispositivos, tales como lo que son: Microsoft, Android, LINUX

GITHUB/GITLAB

Son sitios que permiten tener las virtudes de GIT como sistema de control de versiones, pero en el internet como una red social de programadores para compartir trabajo y proyectos de esta forma trabajar colaborativamente y ver el trabajo de otros siempre y cuando recibas los permisos necesarios.

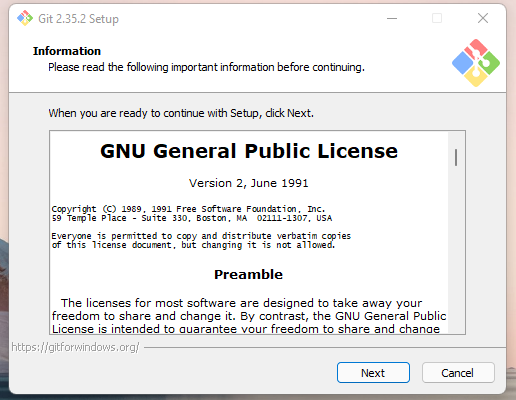
Para ello es necesario instalar GIT de manera local y hacer conexión con los servidores (Se verá en siguientes pasos).

# INSTALACION DE GIT

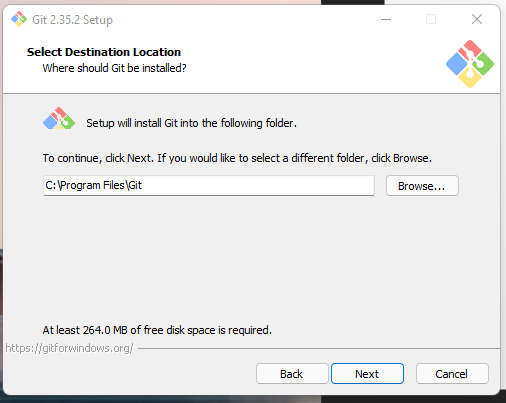
1. En cualquier navegador ingresamos la dirección (<http://git-scm.com/>) donde realizaremos la descarga y seleccionaos la versión de acuerdo a nuestro sistema operativo ya sea de 32 o 64 bits.



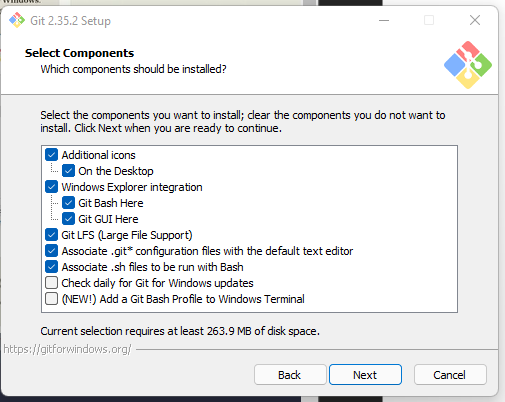
1. Una vez descargado lo que haremos será abrir el instalador, saldrá un cuadro de información sobre la licencia y le damos “Next”.



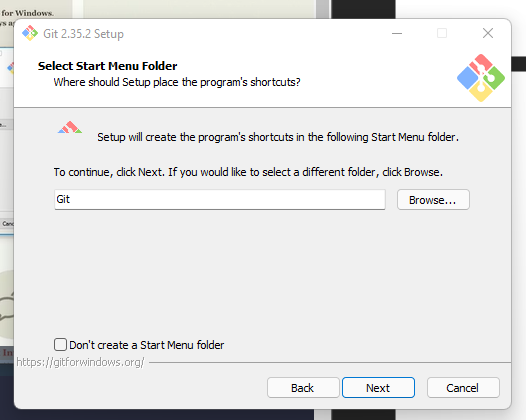
1. La siguiente parte llamada “Select Destination Location” es donde seleccionaremos en donde instalaremos el software, por ahora le dejaremos la ubicación por defecto, una vez seleccionado le daremos “Next” para continuar.



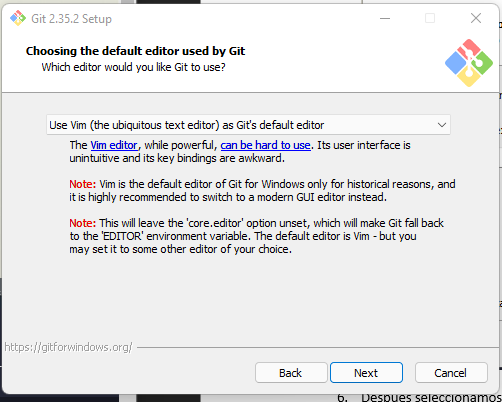
1. Luego en “Select Components” seleccionaremos los componentes le dejamos así tal cual esta, pero si queremos agregar el icono al escritorio solo marcamos la palomita en la parte de “Additional icons” y le damos “Next” para continuar.



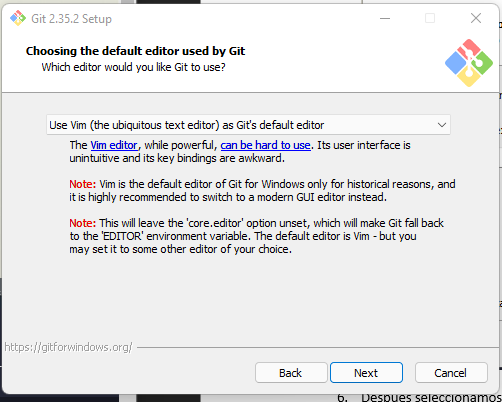
1. Luego “Next”.



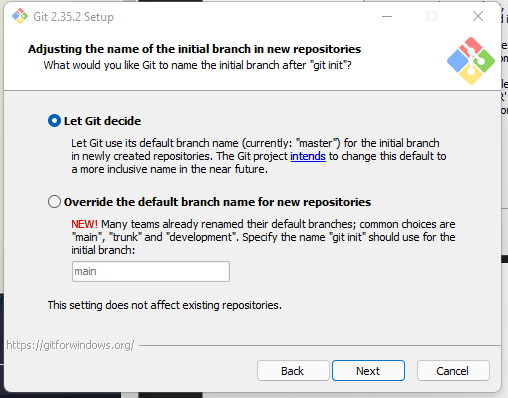
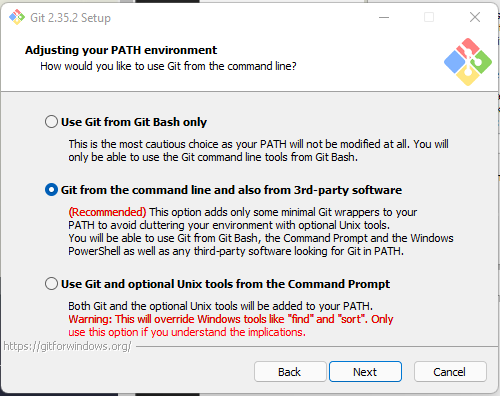
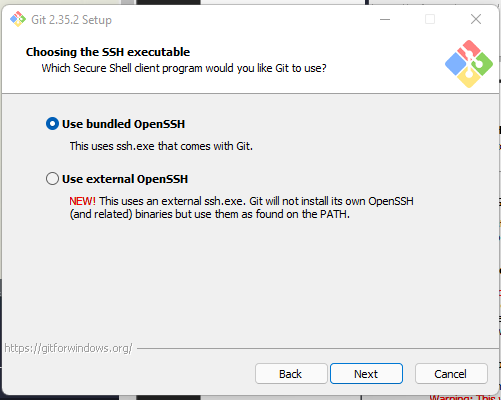
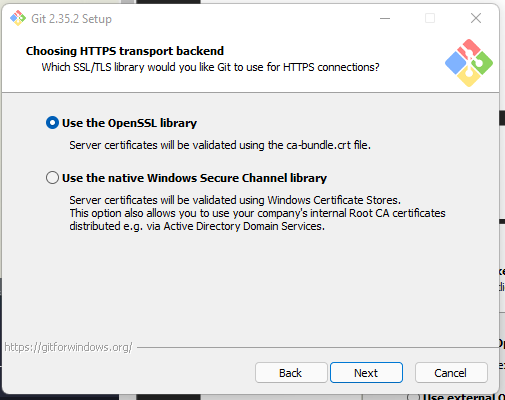
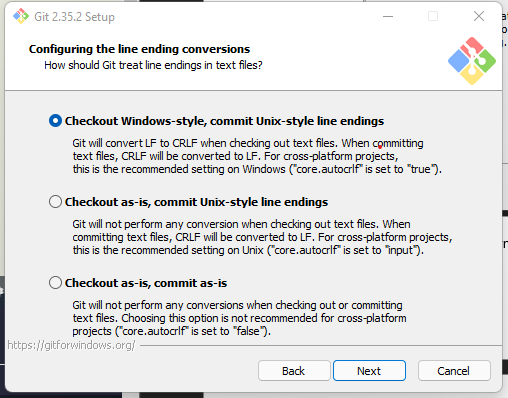
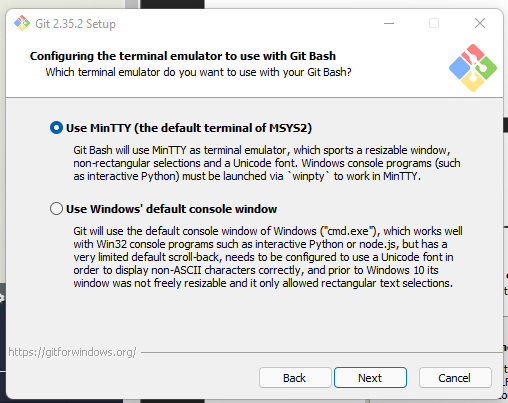
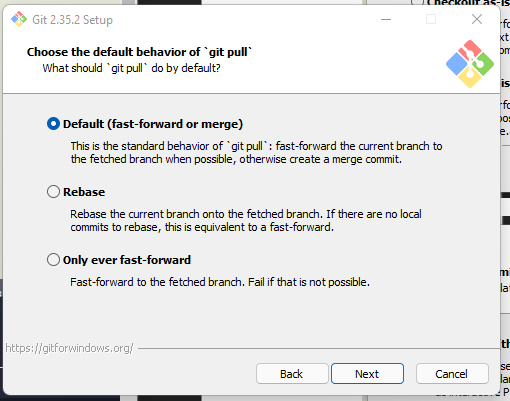
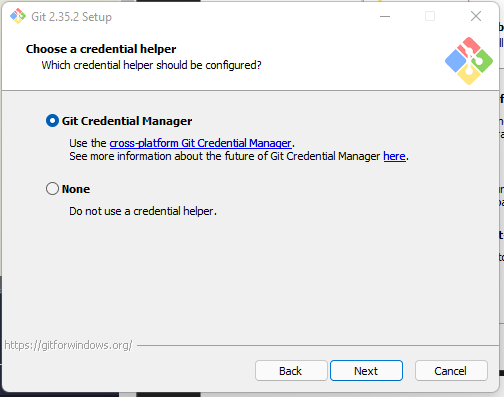
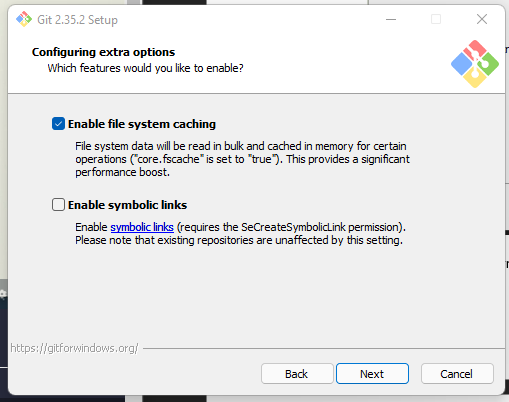
1. Después seleccionamos el editor Vim y le damos “Next” para continuar.



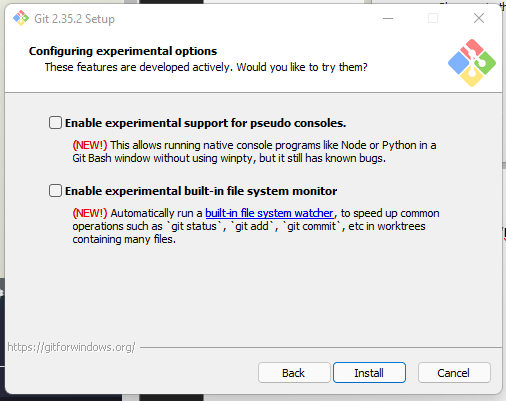
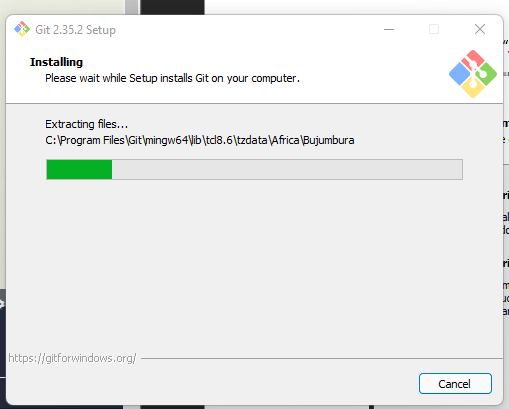
1. Luego dejamos seleccionado con el Prompt y le damos “Next” para continuar.



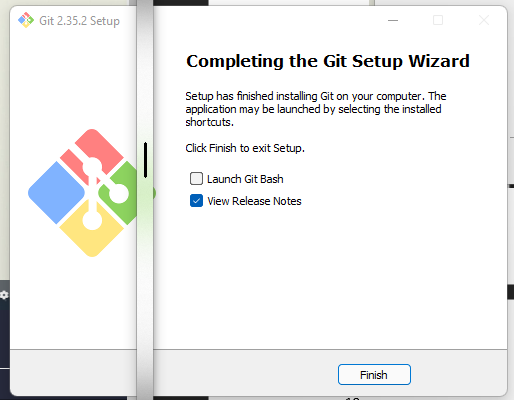
1. Luego “Next”.

1. Por último, le damos a “Install” y esperamos a que se termine de instalar el programa.

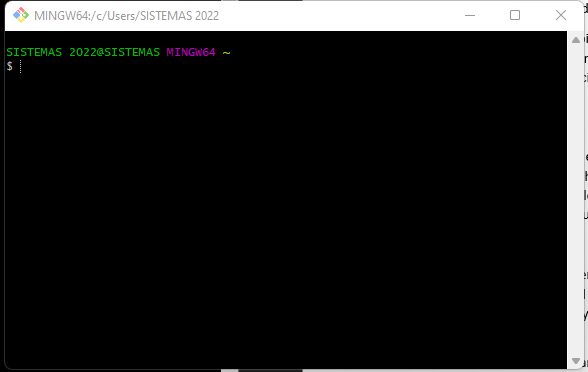
 

1. Listo con eso habremos instalado el GIT.



# NUEVO PROYECTO – IDENTIFICARME

Para iniciar un nuevo proyecto vamos al menú de inicio y buscamos “GIT BASH”, luego se nos abrirá esta ventana.



Lo primero será decirle a GIT como nos llamamos y para ello usaremos el sig. comando:

git config --global user.name "nombre" (Para muchos comandos usaremos git)

Para saber cuál es el nombre de la persona que está trabajando usaremos el mismo comando solo quitándole el nombre.

git config --global user.name

Lo siguiente será colocar nuestro email y para ello usaremos el sig. comando:

git config --global user.mail "email"

Para saber cuál es el email con el que está trabajando usaremos el mismo comando solo quitándole el nombre.

git config --global user.mail

# TRES AREAS EN GIT

## DIRECTORIO DEL TRABAJO

Es la carpeta donde están alojados los archivos del proyecto que procederemos a trabajar los cuales moldearemos para hacer un repositorio.

Para movernos usaremos el sig. comando:

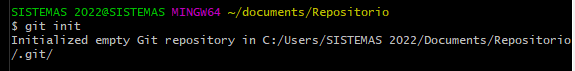
cd “nombre de la carpeta” (igual que los comandos de linux)

Para retroceder usaremos el sig. comando:

cd --

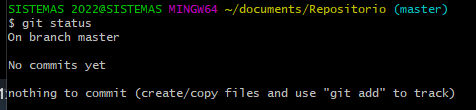
Para crear un repositorio local usaremos el sig. comando (saldrá el siguiente mensaje):

git init



Para saber el estado de nuestro repositorio usaremos el comando:

git status



Entre las cosas que nos puede decir son:

* “On branch master” quiere decir que nos encontramos en una rama maestra.
* “No commits yet” quiere decir que no hemos hecho commits (puntos de guardado).
* “Untracked files” quiere que tenemos archivos que no han sido trackeados, es decir, archivos que no han sido subidos al index área.

## INDEX AREA

Es una zona intermedia, donde podremos agregar archivos, para que GIT pueda registrar cambios.

Para añadir cambios al index usaremos el sig. comando:

git add “nombre y extensión del archivo”

**TIPS:**

* Utilice la finalización de tabulación. Escriba la primera parte del archivo y presione Tab. Si ha escrito lo suficiente para que sea único, se completará. …
* Rodee el nombre entre comillas: mv “Archivo con espacios” “Otro lugar”
* Utilice barras invertidas para escapar de los caracteres especiales: archivo mv con espacios en otro lugar.

Para añadir todos los archivos usaremos el sig. comando:

git add -A

Para quitar un archivo subido usaremos el sig. comando:

git rm --cached “nombre del archivo y su extension”

## GIT AREA

Es donde veremos el estatus actual del proyecto.

# COMMITS

Es un punto de salvado, es decir, al hacer un commit creamos un punto en nuestra línea de tiempo sobre nuestro trabajo y no perder progreso.

Para ello usamos el siguiente comando:

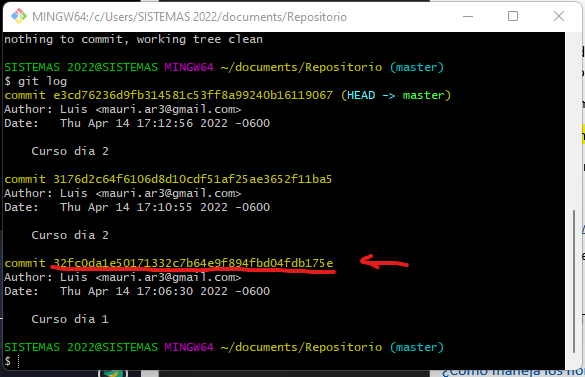
git commit -m “mensaje que describa el commit”

Para consultar lista de commits que se han hecho en el repositorio usaremos el comando

git log

## USAR COMMITS PARA VIAJAR EN EL TIEMPO

Para hacer un viaje, seleccionamos la clave del commit en cuestión.

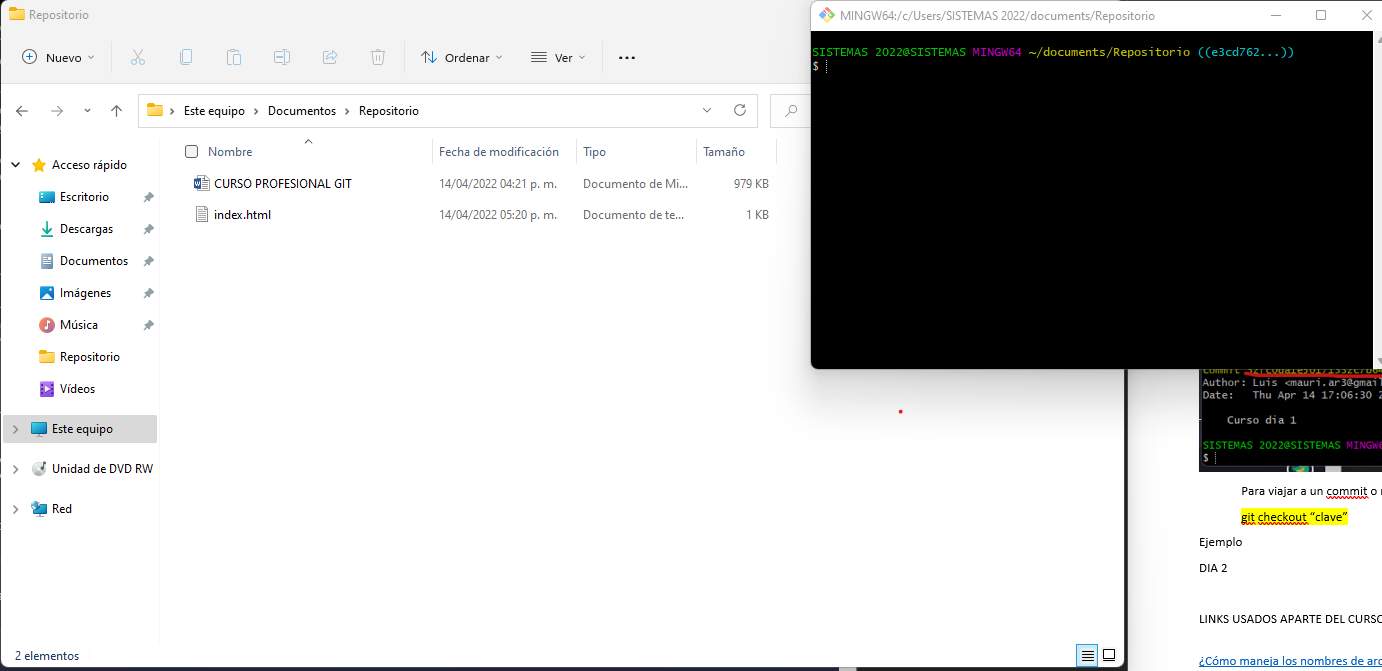


Para viajar a un commit o rama en el repositorio usaremos el sig. comando:

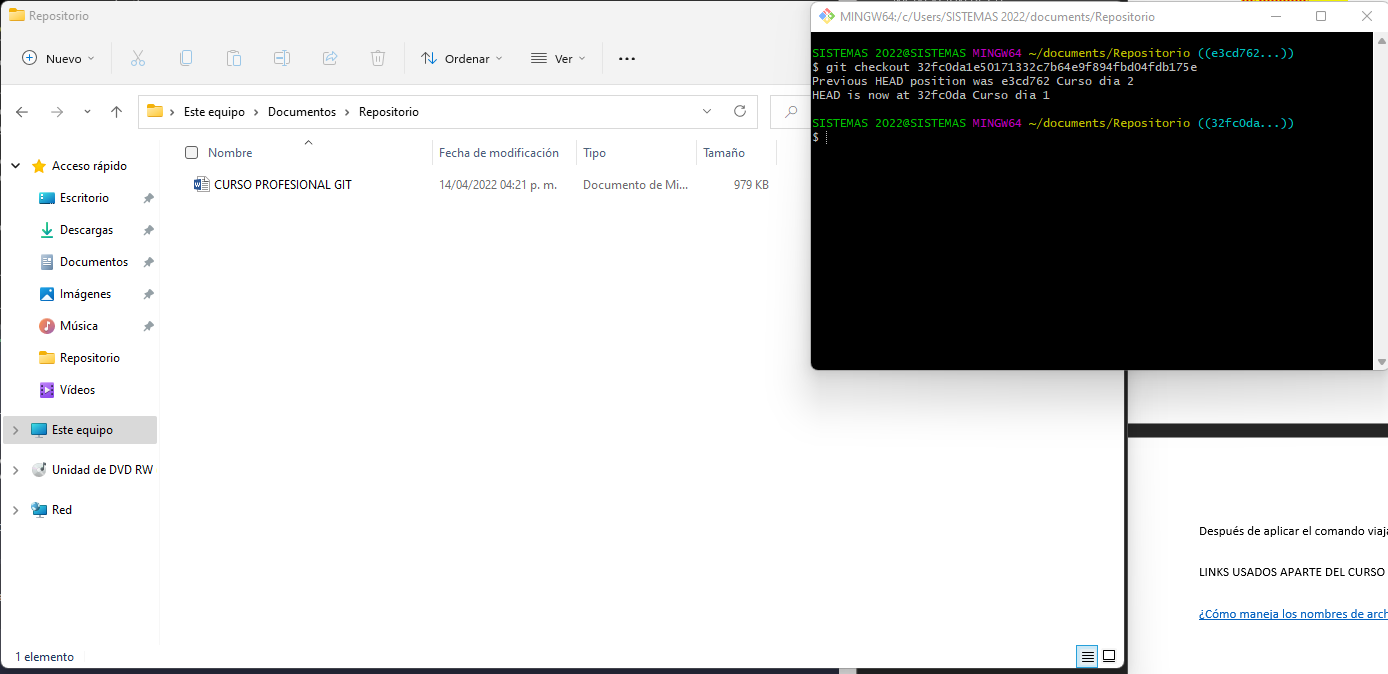
git checkout “clave”

Ejemplo

DIA 2



Después de aplicar el comando viajaremos al DIA 1



LINKS USADOS APARTE DEL CURSO

[¿Cómo maneja los nombres de archivo con espacios en Linux? - CompuHoy.com](https://www.compuhoy.com/como-maneja-los-nombres-de-archivo-con-espacios-en-linux/#:~:text=%C2%BFC%C3%B3mo%20maneja%20los%20nombres%20de%20archivo%20con%20espacios,el%20espacio%20como%20parte%20del%20nombre%20del%20archivo.)

# RESET

Para esto hay que tener dos cosas muy claras

* GIT: estatus actual del proyecto, es decir, ultimo salto o check out.
* MASTER: es el último commit elaborado.

Los reset son saltos como los check out pero que borrar a su paso y hay tres tipos:

* Reset soft: lleva a un commit borrando los commits posteriores, pero mantiene intacto el directorio, es decir no toca el código ni el trabajo, para ello usamos el sig. comando:

git reset --soft “clave”

* Reset mixed: lleva a un commit borrando los commits posteriores y al index área, sin embargo, no las borrara del directorio de trabajo de forma física.
* Reset Hard: borra todo lo que haya sucedido después del commit al que se dirige y lo convierte en el master, para ello usamos el sig. comando:

git reset --hard “clave”

# RAMAS

GIT puede manejar realidades paralelas (si tal cual como se oye), es decir, ramas de trabajo. Se puede definir como espacios paralelos de trabajo dentro de un mismo repositorio, es útil dado que un proyecto puede tomar diferentes caminos y una rama puede ser usada por diferentes personas que trabajan en el mismo proyecto

Para ver las ramas de un proyecto usamos el sig. comando:

git branch

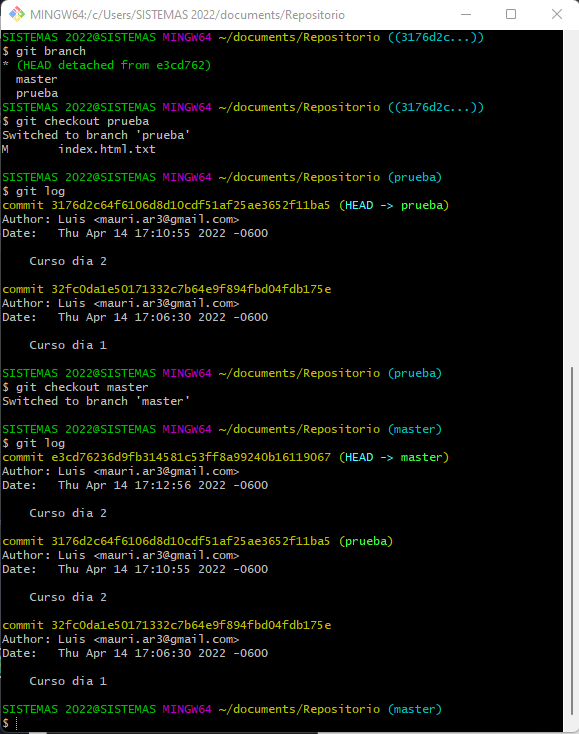
Para crear una rama usamos el sig. comando:

git branch “nombre”

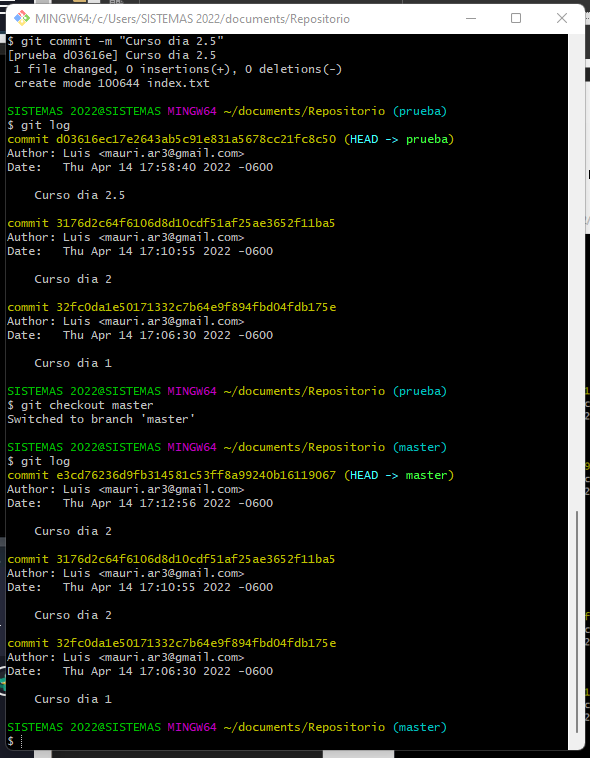
Para cambiar de rama usamos el sig. comando:

git ckeckout “nombre”

En la siguiente imagen se puede ver las ramas:



Si cambiamos las ramas y realizamos cambios, van cambiando los commits como se puede observar en la siguiente imagen:



**Nota:** al cambiar de ramas se cambia el progreso del trabajo por lo que hemos explicado previamente, es muy útil debido a que podemos hacer pruebas.

Para fusionar las ramas usamos el sig. comando:

git merge “nombre de la rama que queremos fusionar”

**Nota:** cuando se hace esto se abrirá un cuadro editor de texto, para ello es importante que recuerdes los comandos de edito vi

* Esc -- Cambia al modo de comando.
* (: w) -- Guarde y continúe editando.
* (: wq o ZZ) -- Guardar y salir / salir vi.
* (: q!) -- Salga de vi y no guarde los cambios.

Para borrar una rama usaremos el sig. comando:

git branch “nombre de la rama que queremos fusionar” -D

# TRABAJAR CON GITHUB

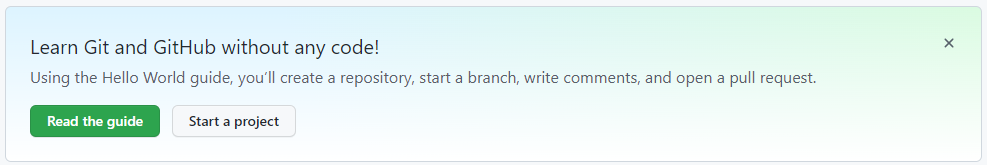
GITHUB

Es una herramienta con la cual distintas personas pueden colaborar o compartir un proyecto, trabajo o conocimiento, así como una red social cooperativa de una manera muy sencilla.

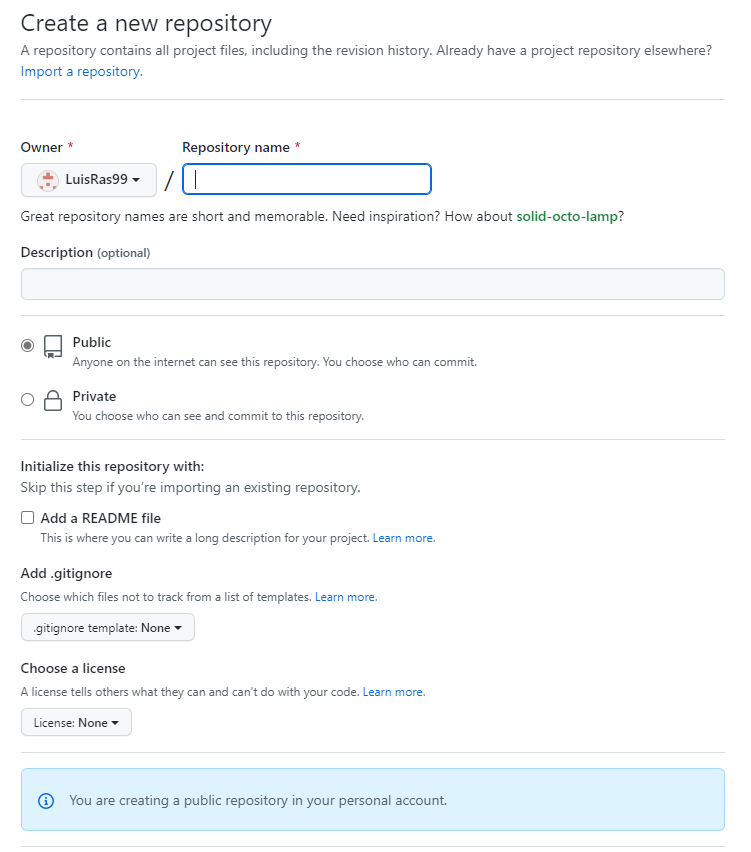
Como dato extra se pueden clonar repositorios de otras personas (copiar repositorio).

Lo primero que haremos será registrarnos en la página de GITHUB ([GitHub](https://github.com/)).

Para iniciar un nuevo proyecto le damos en “star project”.



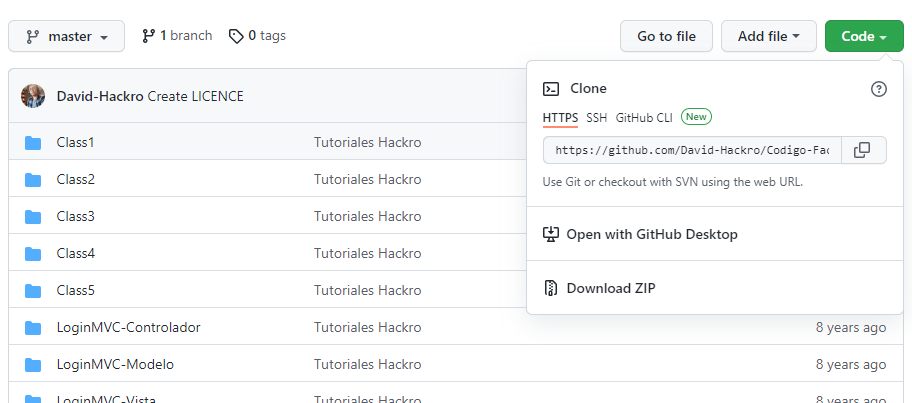
Después nos saldrá el siguiente menú:



Aquí hay varios puntos que comentar:

* Se puede poner un nombre al repositorio.
* Se le agrega una descripción.
* Se puede elegir si será público o privado, pero si se selecciona privado habrá que **pagar**.
* Ya es opcional si se le agrega el archivo Readme y las llaves.

En GITHUB podemos ver el trabajo de otras personas, donde podemos descargar o clonar el trabajo de otros, tenemos dos formas.



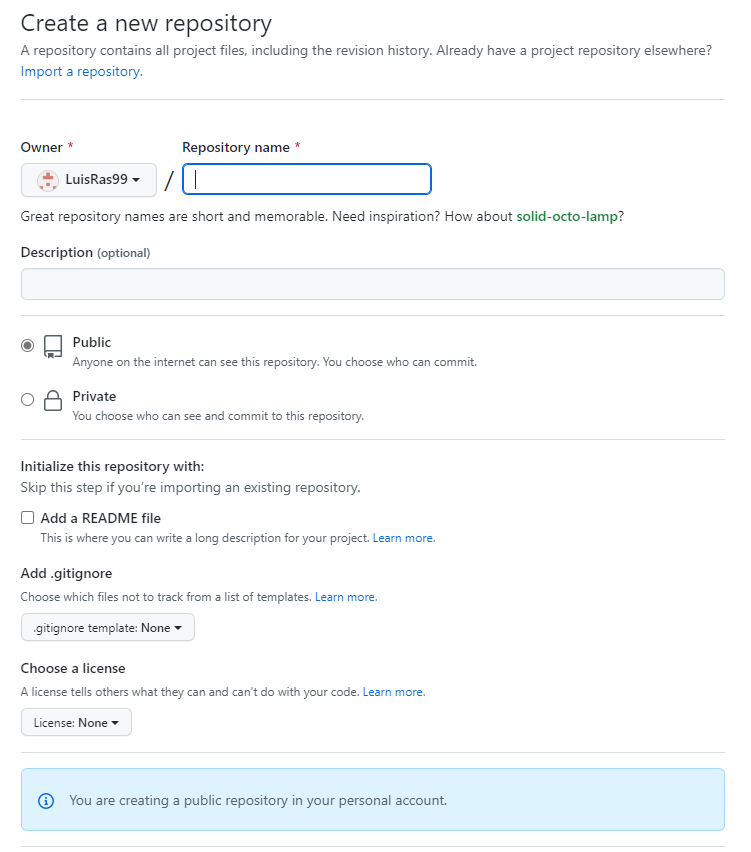
* Se puede descargar el ZIP con los archivos o desde la terminal usando el link que hay aparece.

Para descargar mediante link se usa el sig. comando:

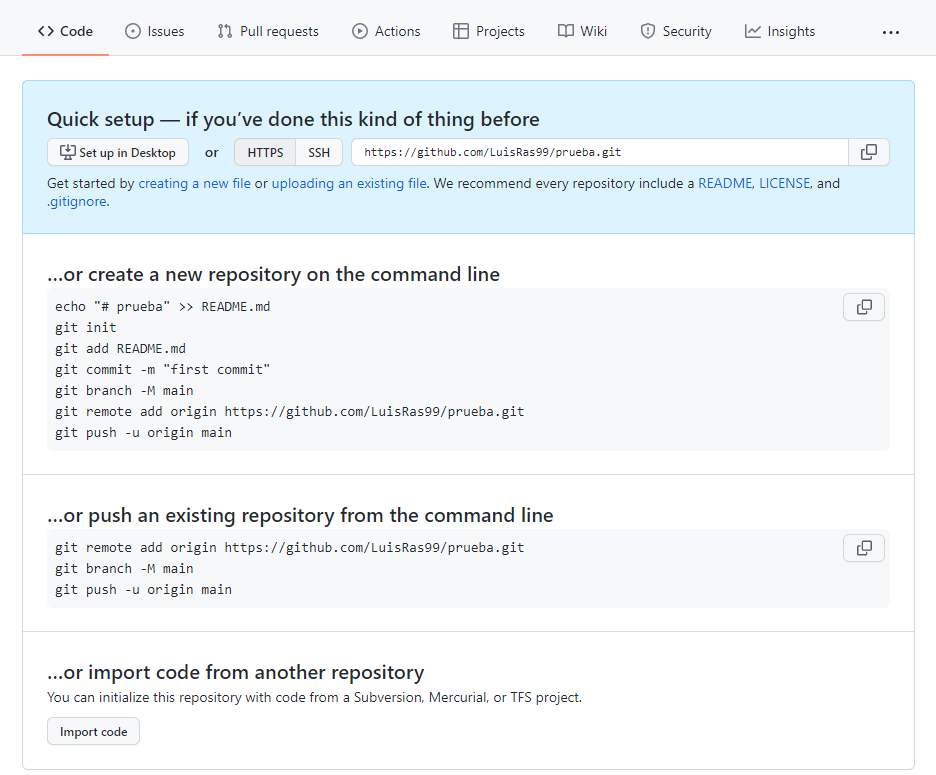
git clone “link”

# SUBIR UN PROYECTO LOCAL A GITHUB

Lo primer será crear un nuevo repositorio



Después nos saldrá el siguiente menú:



Nos dará un link con el podremos conectarnos a GITHUB a través de GIT, para hacerlo usaremos el sig. comando.

git remote add origin “link de GITHUB”

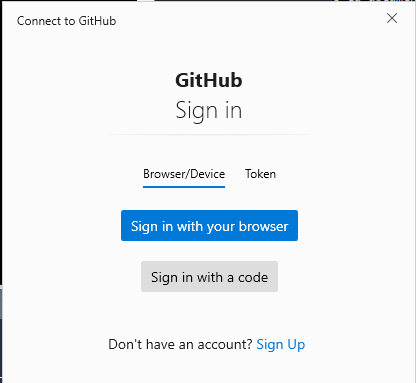
Para verificar que nos hemos conectado empleamos el sig. comando:

git remote add -v

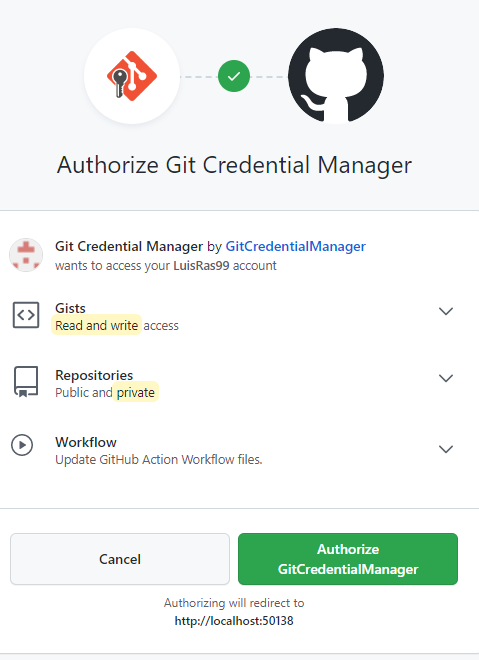
Para subir nuestros commits y nuestras ramas usaremos el sig. comando:

git push origin master

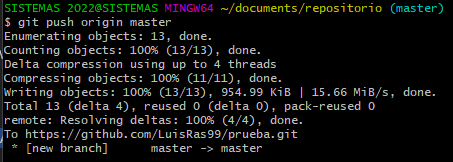
Al hacer esto sadra el sig. cuadro en el cual tendremos que iniciar sesión.

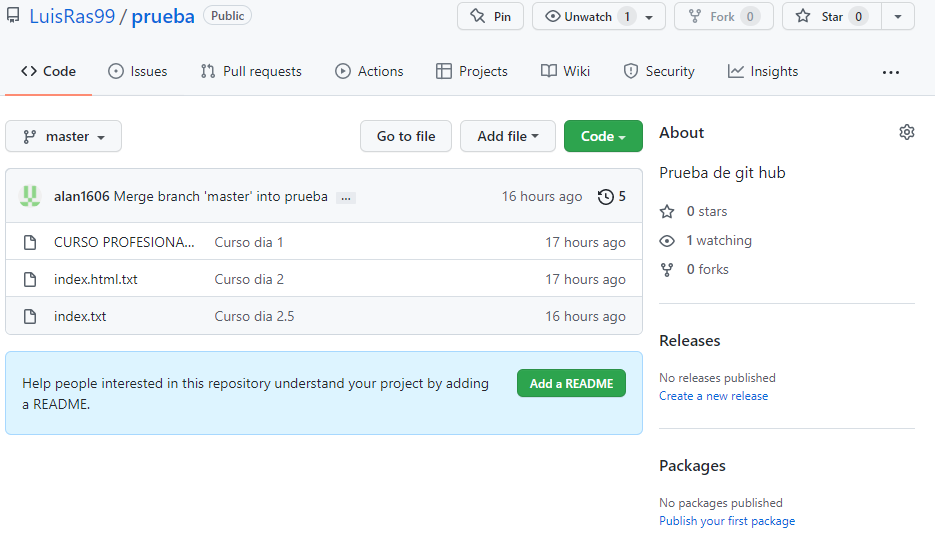


Después saldrá el siguiente donde otorgaremos los permisos.



Posterior a eso saldrá las sig. líneas de comando donde nos indicara que la conexión se ha establecido correctamente y en la página de GITHUB también se verá reflejado.

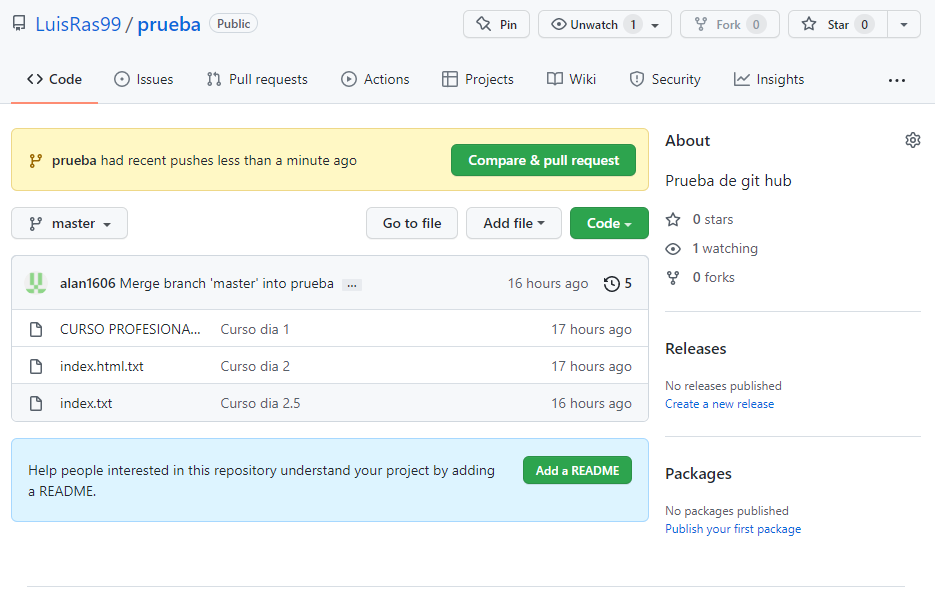




Para subir una rama usaremos el sig. comando:

git push origin “nombre de la nueva rama”

Nos saldrá un mensaje de confirmación.



En esta parte podremos intercambiar entre ramas.

