La Terminal, GIT & GitHub

1

Terminal - Consola - CMD

¿Qué rayos es eso?

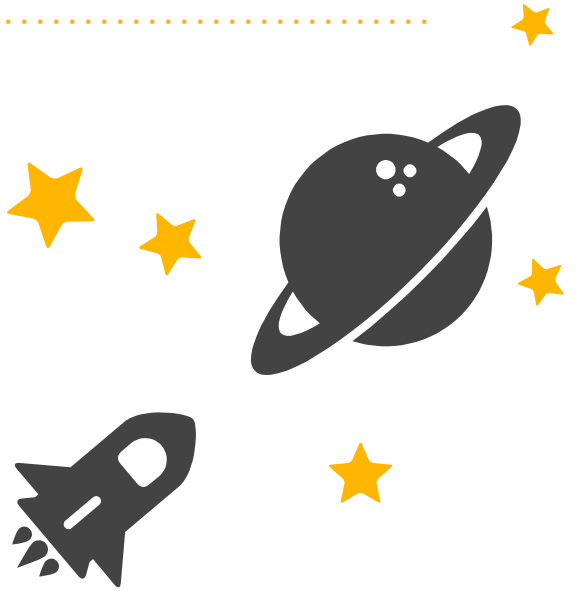
*El terminal es un **software** que está presente en todos los sistemas operativos y con el cual puedo **dar órdenes al sistema** a través de líneas de comando.*

¿Por qué usar la terminal?

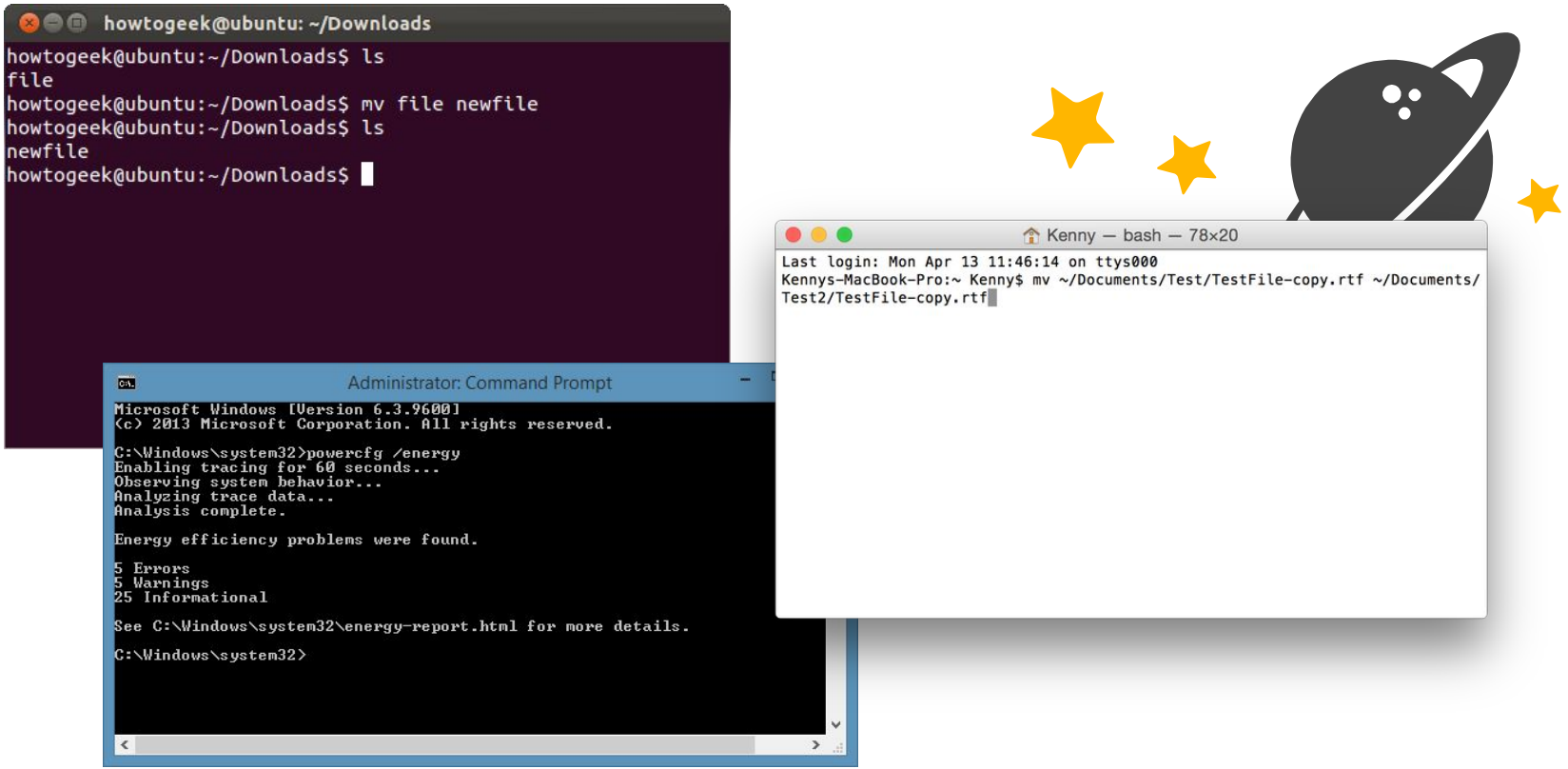
- Para tener mayor control sobre el SO.
- Porque es muy común en los entornos de desarrollo.
- Porque algunos lenguajes de programación lo *"requieren"*.

Si sé usar la terminal y me acostumbro a la misma, podré optimizar mucho mi trabajo al momento de programar.

¿Donde está la terminal?



Sea cual sea el SO que esté usando, acceder a la misma es muy sencillo.


howtogeek@ubuntu: ~/Downloads
howtogeek@ubuntu:~/Downloads$ ls
file
howtogeek@ubuntu:~/Downloads$ mv file newfile
howtogeek@ubuntu:~/Downloads$ ls
newfile
howtogeek@ubuntu:~/Downloads$
Administrator: Command Prompt
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
C:\Windows\system32>powercfg /energy
Enabling tracing for 60 seconds...
Observing system behavior...
Analyzing trace data...
Analysis complete.
Energy efficiency problems were found.
5 Errors
5 Warnings
25 Informational
See C:\Windows\system32\energy-report.html for more details.
C:\Windows\system32>
Kenny — bash — 78×20
Last login: Mon Apr 13 11:46:14 on ttys000
Kennys-MacBook-Pro:~ Kenny$ mv ~/Documents/Test/TestFile-copy.rtf ~/Documents/
Test2/TestFile-copy.rtf

Comandos básicos que debo aprender

ls (muestra los archivos de la carpeta en la que estoy ubicado, en Windows, solo si uso el **PowerShell**)

dir (solo Windows. Muestra los archivos de la carpeta en la que estoy ubicado)

cd .. (me permite retroceder a una carpeta previa)

cd *nombre-carpeta* (me permite acceder a la carpeta mencionada)

mkdir *algo*

(crea una carpeta con el nombre "**algo**")

touch *archivo.txt*

(crea una archivo de texto "**archivo.txt**")

rm *archivo.txt*

(elimina un archivo con el nombre "**archivo.txt**")

mv *nombre.txt otro.txt*

(cambia el nombre "**archivo.txt**" a "**otro.txt**")

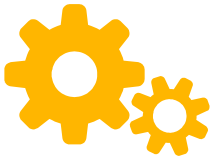
clear

(limpia todo lo que haya escrito en la consola. En Windows, solo en **PowerShell**)

cls (limpia todo lo que haya escrito en la consola, solo Windows)

Momento de **perder el miedo** a la Terminal

Pequeña práctica que me permita familiarizarme más con la consola

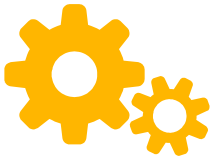


1. Con la consola, llegar hasta la carpeta ***Escritorio ó Desktop*** de mi máquina.

2. Una vez allí, crear una carpeta llamada "**estoy-programando**".

3. Ingresar a la carpeta recién creada.

4. Una vez dentro, crear dos archivos, uno llamado "**prueba.html**" y otro llamado "**home.html**".



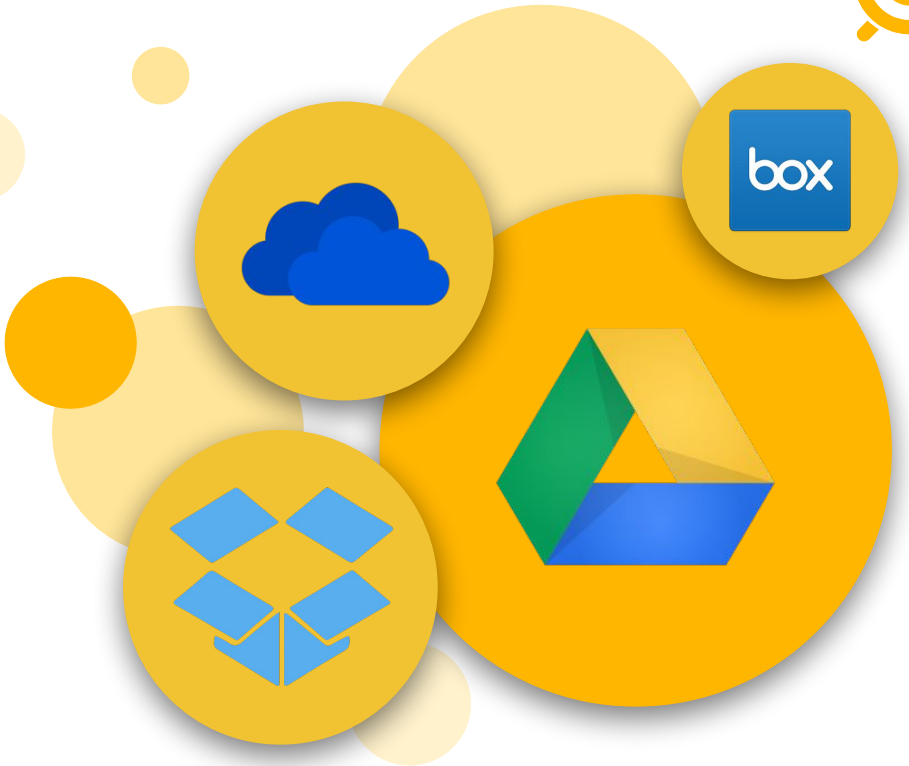
5 Verificar en consola que dichos archivos hayan sido creados correctamente.

6 ¡Opps! Cometí un error y creé un archivo que no era, ahora tendré que borrar por consola el archivo "**prueba.html**".

7 ¡Opps! Cometí otro error, y tendré que cambiar el nombre del archivo "**home.html**" a "**index.html**".


ROADS? WHERE WE'RE GOING
WE DON'T NEED ROADS!
GIT & GitHub
BACK TO THE FUTURE

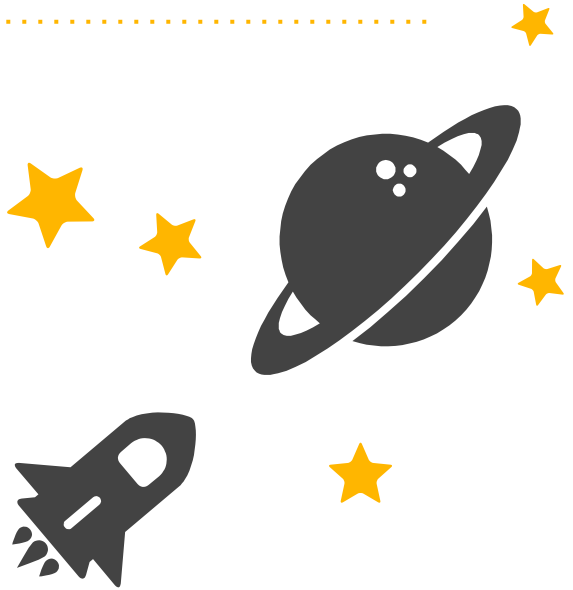
¿Cómo **compartimos** archivos en la **actualidad**?



¿PUEDO COMPARTIR ASÍ MIS PROYECTOS?

Necesito un software

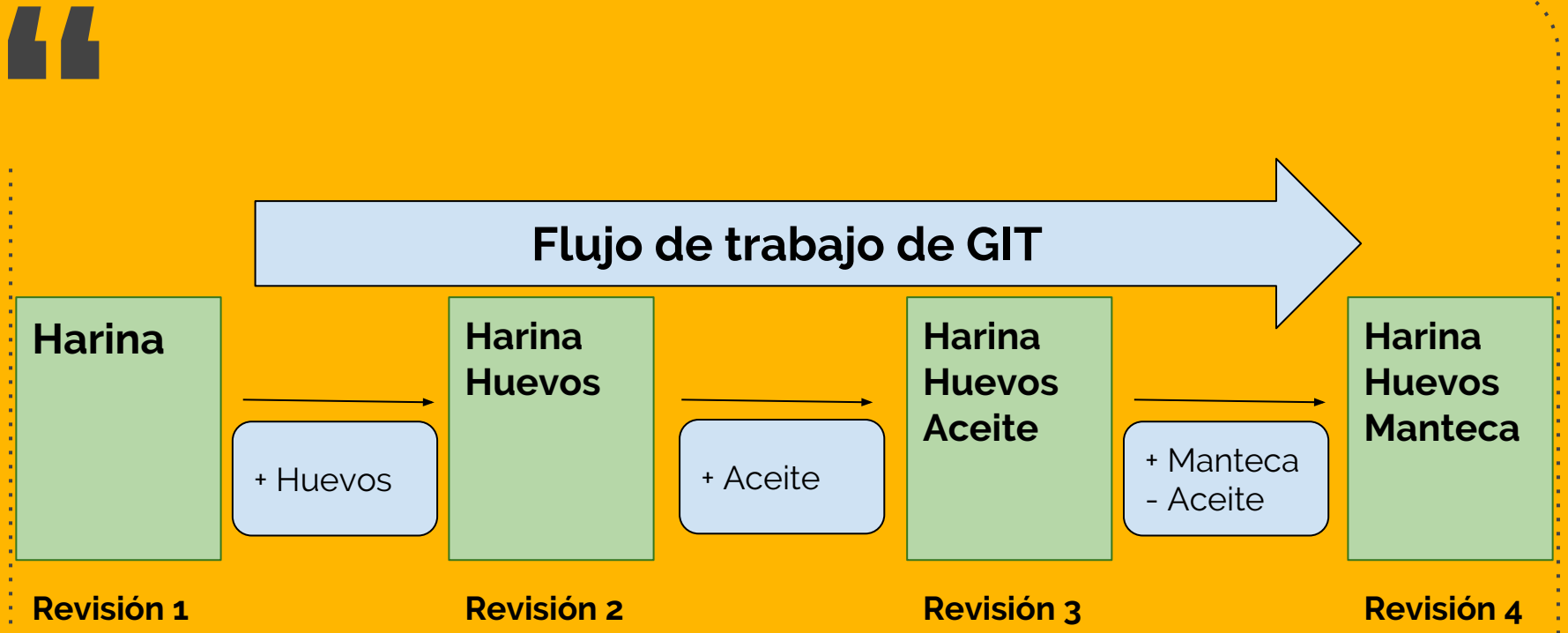
que me permita hacer un correcto seguimiento y control de versiones.



git

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente.

*Cuando trabajo con Git, hablo de trabajar con un **REPOSITORIO.** El cual es un lugar en donde se almacenan mis archivos. Hay dos tipos de **REPOSITORIOS** el **local** y el **remoto.***


Flujo de trabajo de GIT
Harina
+ Huevos
Harina
Huevos
+ Aceite
Harina
Huevos
Aceite
+ Manteca
- Aceite
Harina
Huevos
Manteca
Revisión 1
Revisión 2
Revisión 3
Revisión 4

Ramas / Branches

Una rama en Git es un "espacio" dentro del repositorio donde se almacenan nuestros archivos.
Por defecto en Git, la rama principal se llama ***master****.*

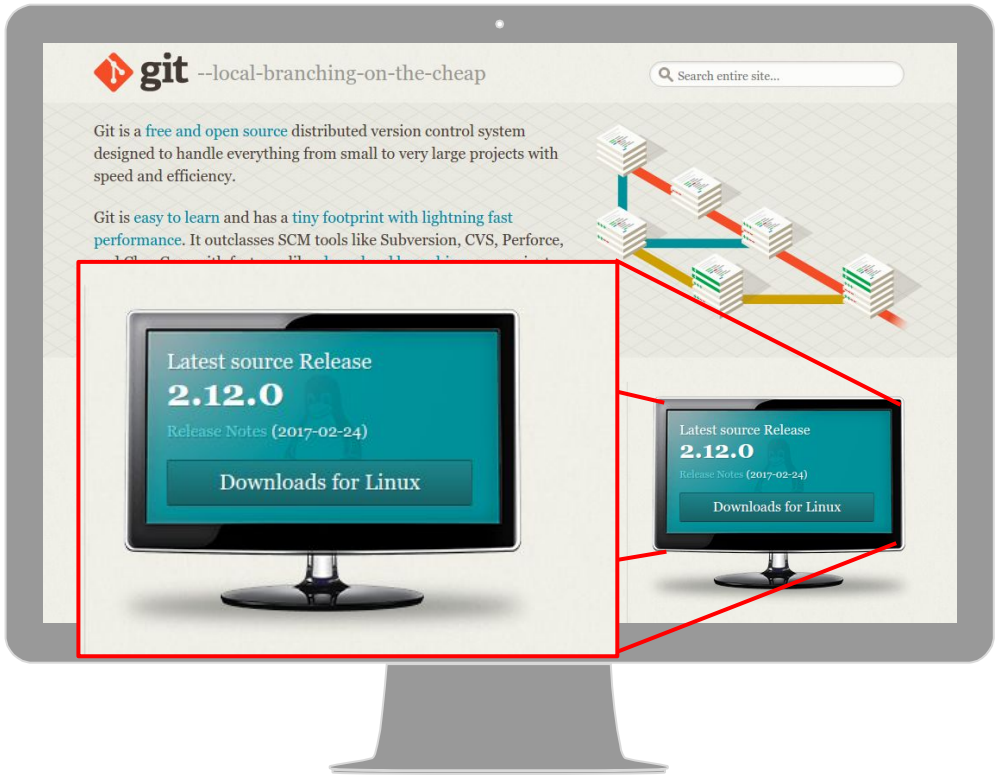
Puedo tener la cantidad de ***branches*** *que desee.*

git

Repo Local

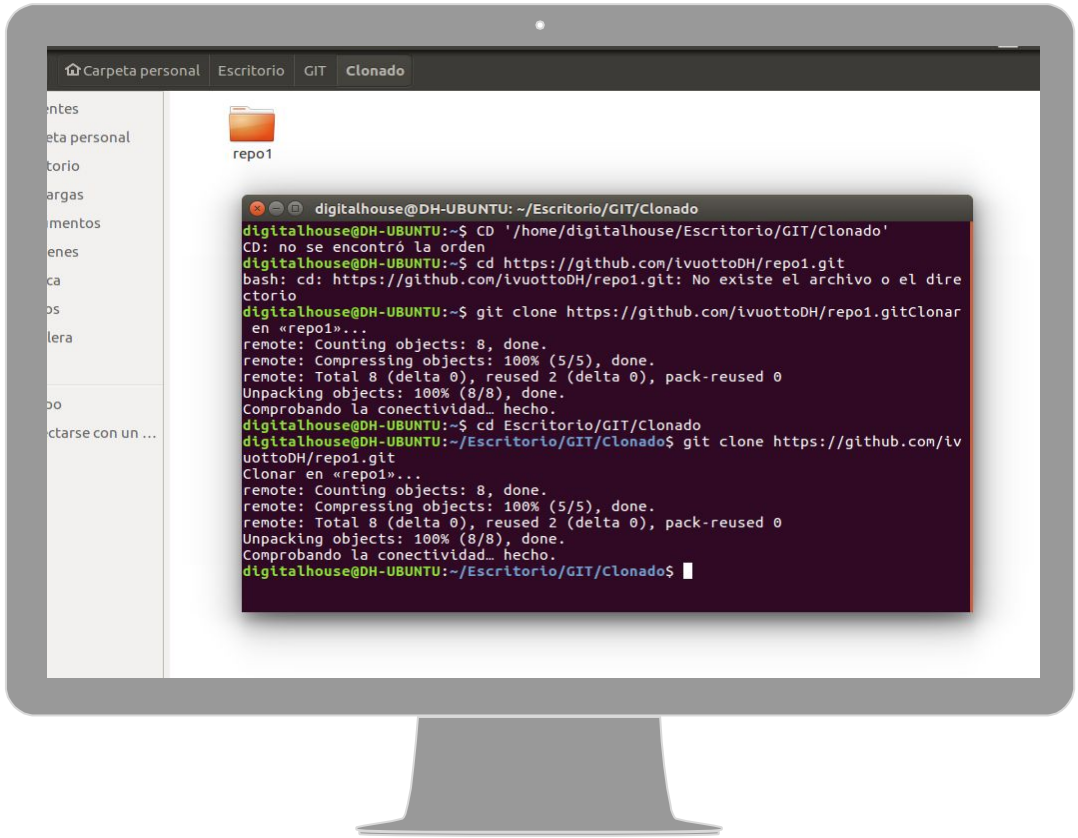
GitHub

Repo Remoto



Instalando GIT en mi máquina

https://git-scm.com/



Creando el repositorio local

Lo primero será, con la Terminal, llegar hasta la carpeta en donde quiero crear el repositorio y posteriormente escribir el siguiente comando:

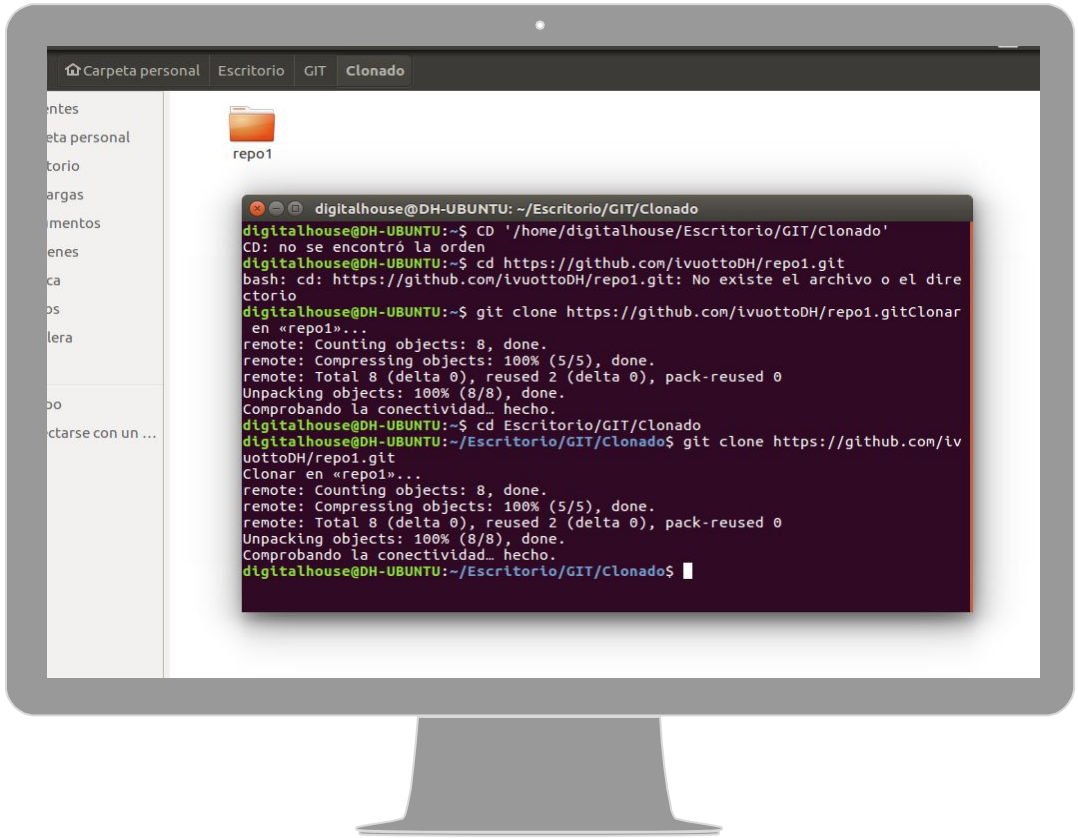
git init

git init

Crea un repositorio local (en mi máquina) y me permite comenzar a utilizar todas las funcionalidades de GIT.

Generalmente crea una carpeta oculta la cual contiene todo el repositorio y sus distintas ramificaciones.

Esta carpeta no la necesito tocar para nada.



Agregando mi identidad

*Para que todo lo que se haga dentro del repositorio, quede asignado a un "**autor**". Necesito decirle al repositorio quien soy, así:*

git config user.name "Jhon_Doe"

git config user.email "jhon@email.com"

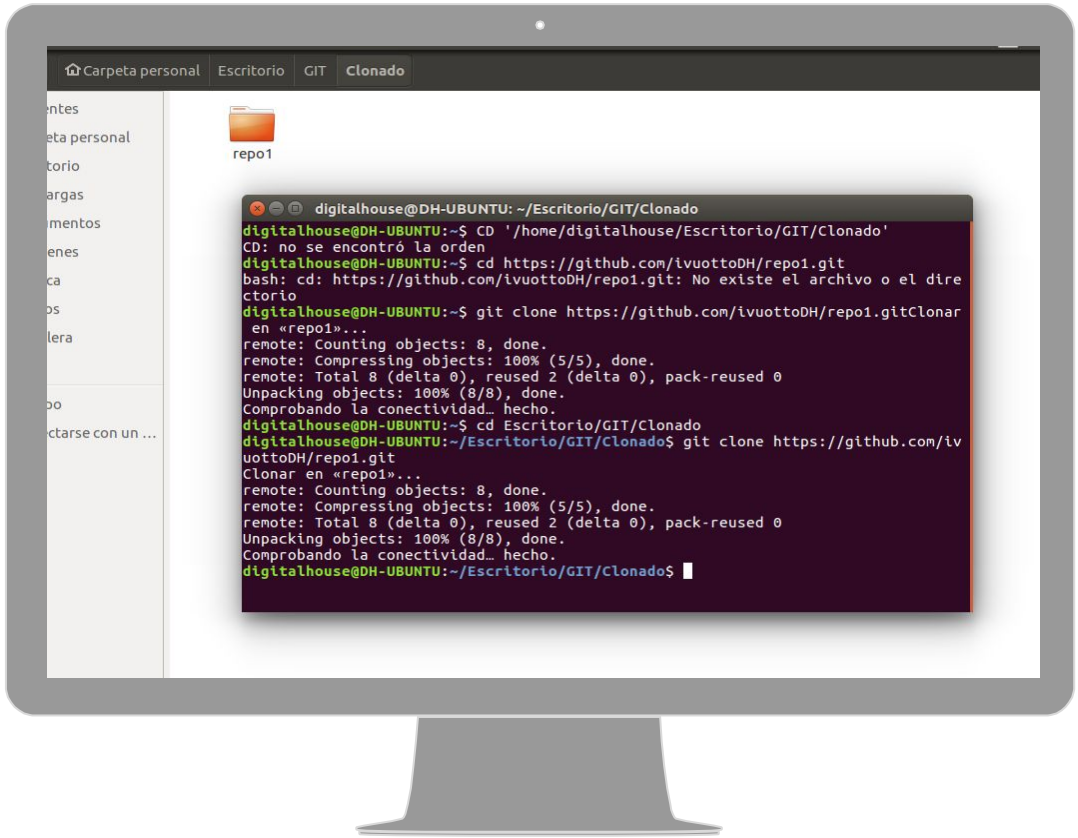
“

git config user.name " "

Dentro de las comillas pondré mi usuario de Github.com

git config user.email " "

Dentro de las comillas pondré el email con el que me registré en Github.com



Agregando archivos al repositorio local

Hasta el momento, mis archivos no han sido agregados al ***repositorio local,*** *para ello tendré que escribir el siguiente comando:*

git add .

ó

git add archivo.txt

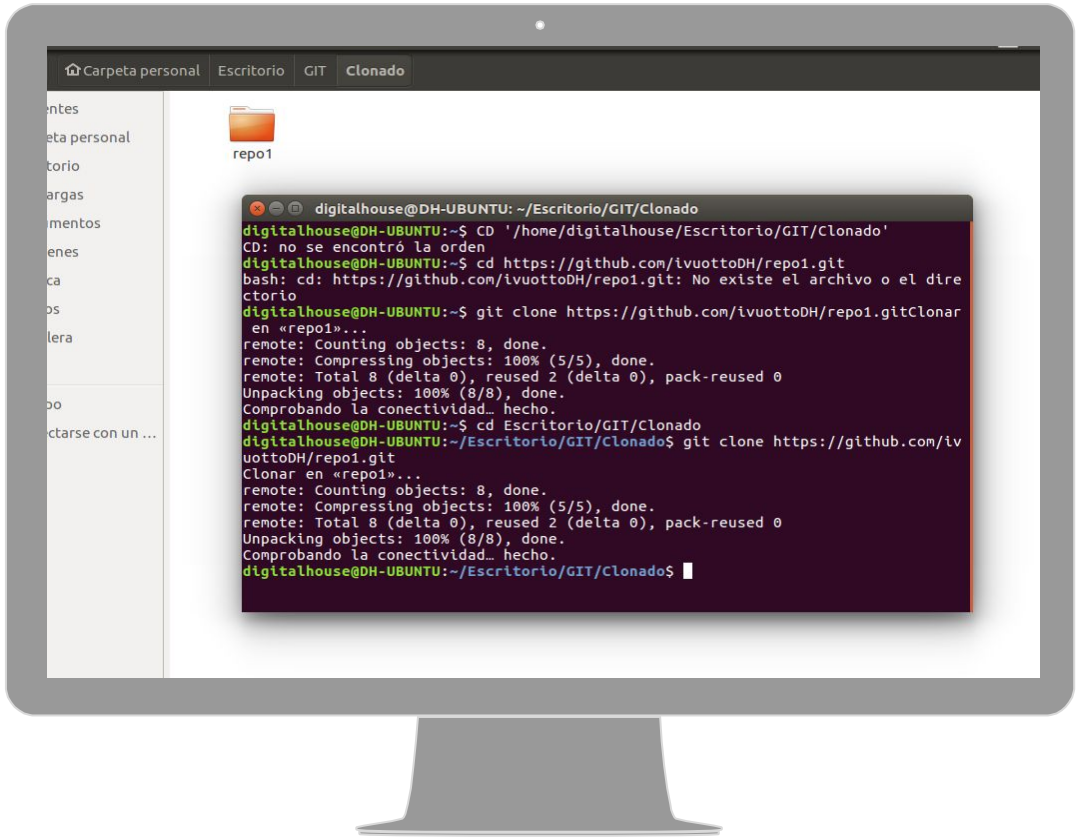
“

git add .

*Agrega al **stage** (paquete de archivos) todos los archivos que haya creado en mi proyecto.*

git add archivo.txt

*Agrega al **stage** (paquete de archivos) solamente el archivo referenciado.*



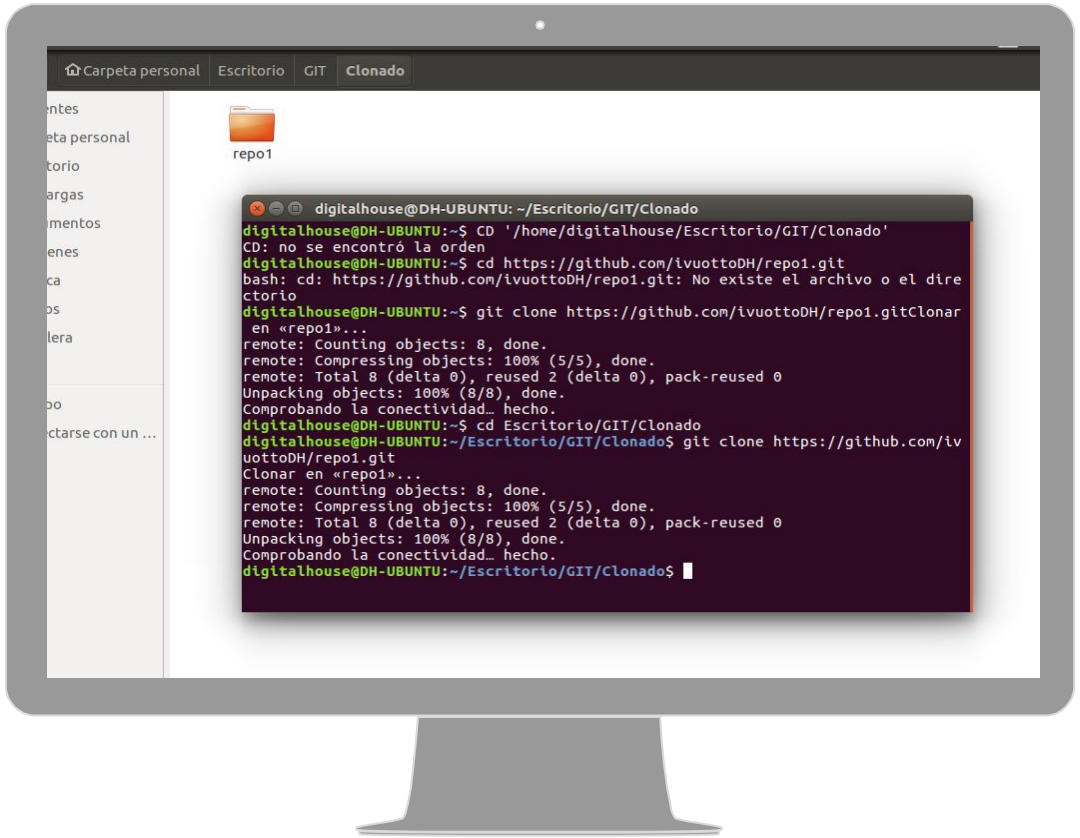
Revisando el status de mi repositorio

Cada vez que desee comprobar o verificar el estado de mi repositorio local deberé escribir la siguiente línea de comando:

git status

git status

*Analiza el estado del repositorio local, me dirá si hay archivos que no se han agregado al **stage** así como también si hay archivos agregados al stage, pero no de forma definitiva ("**commiteados"**).*



Agregando archivos al stage de forma definitiva

“

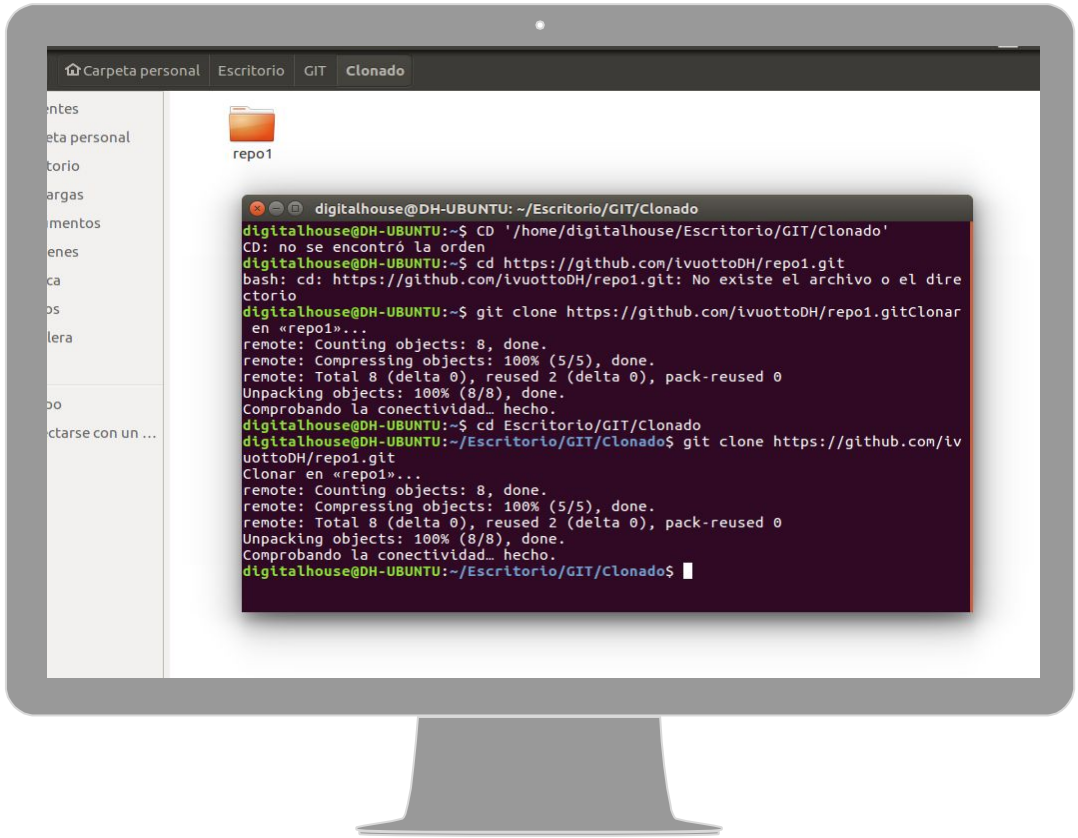
*Para finalmente confirmar que los archivos agregados al **stage** los quiero de **manera definitiva** escribiré:*

git commit -m "un mensaje cualquiera"

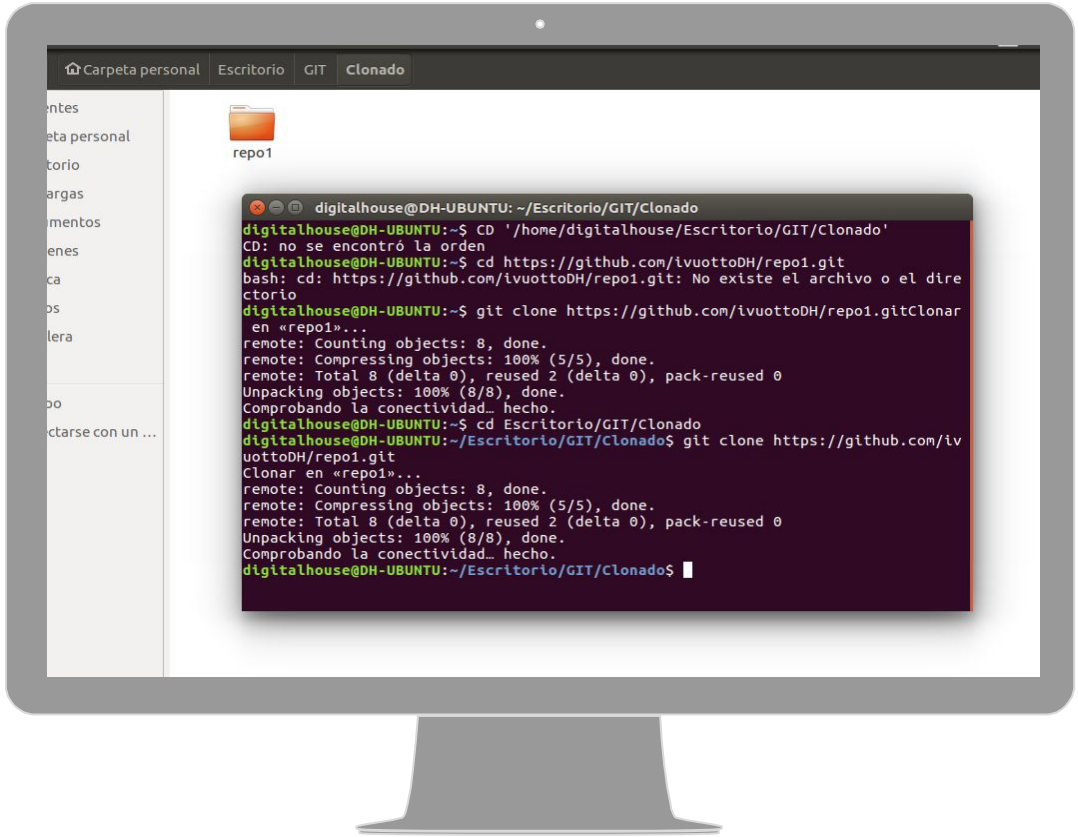
git commit -m "un mensaje cualquiera"

*La directriz **commit**, le indica al repositorio local que los archivos los quiero agregar de manera oficial. La **-m** indica que a continuación agregaré un mensaje que especifique qué trabajo hice.*

*Los **commits** sirven como pequeños **backups** a los cuales podré acceder fácilmente si así lo necesito.*



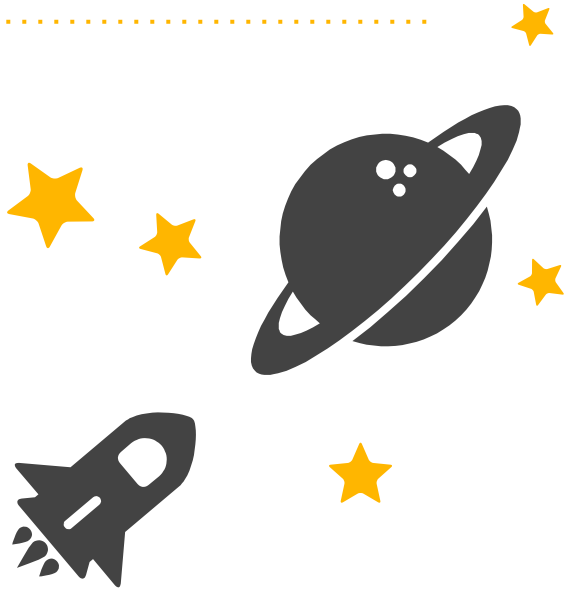
Conectando con el repositorio remoto



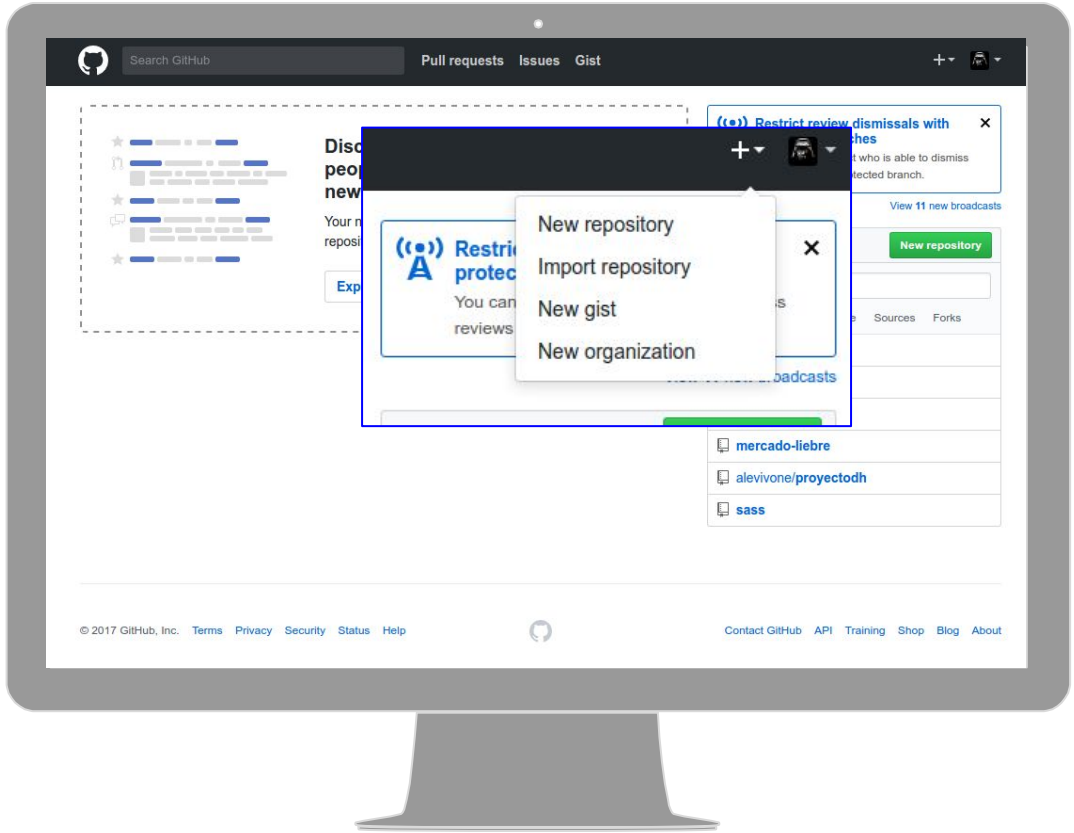
1ero

Debe existir un repositorio en Github

¿Cómo crear el repositorio remoto?

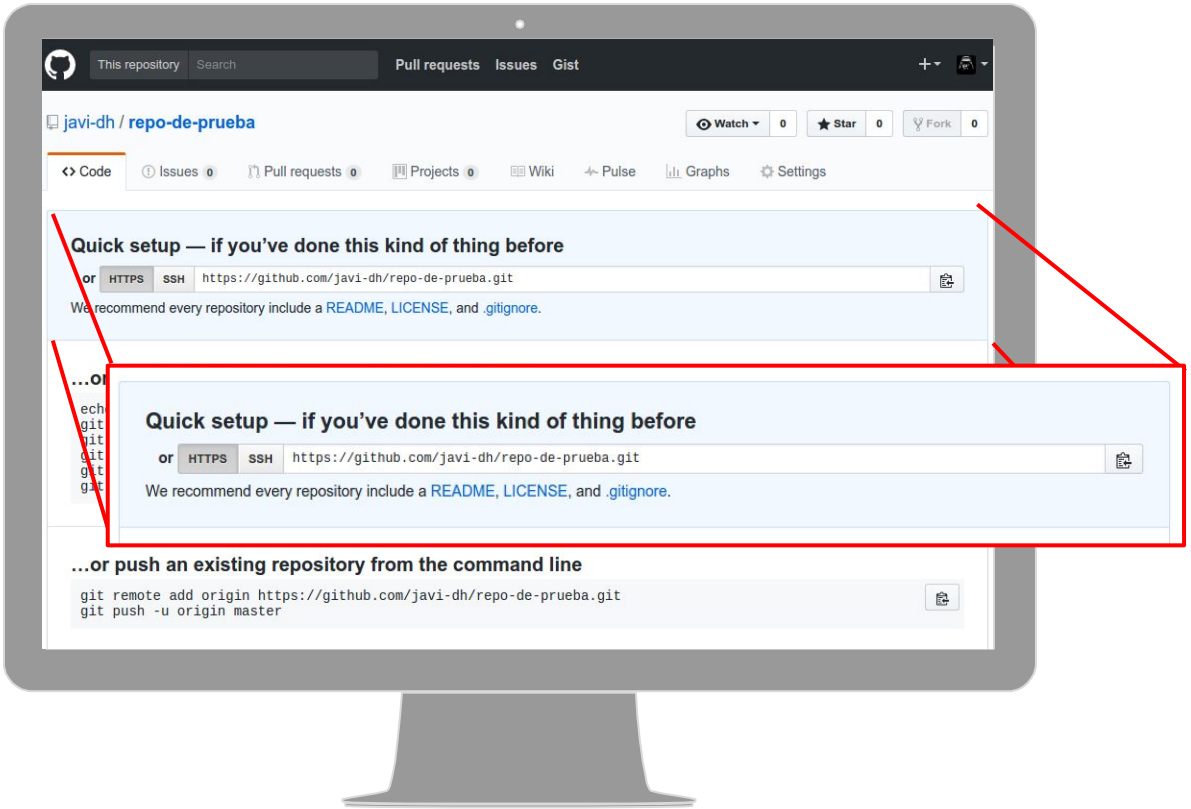


Para ello voy a utilizar mi cuenta de GitHub.



Logueado en mi cuenta de GitHub, voy al ícono + y ahí elijo la opción **New Repository**.

Ahora, debo asignarle un nombre al repositorio. Puede ser cualquiera. De resto **NO TOCARÉ** nada más, solo el botón **CREATE**.



Luego veré esta pantalla. La URL que aparece allí es la que necesito tener a mano en el paso de:

Conectando con el repositorio remoto.

Habiendo creado el ***Repositorio Remoto*** *es hora de conectarlo con mi* ***Repositorio Local,*** *para que pueda subir allí los archivos.*

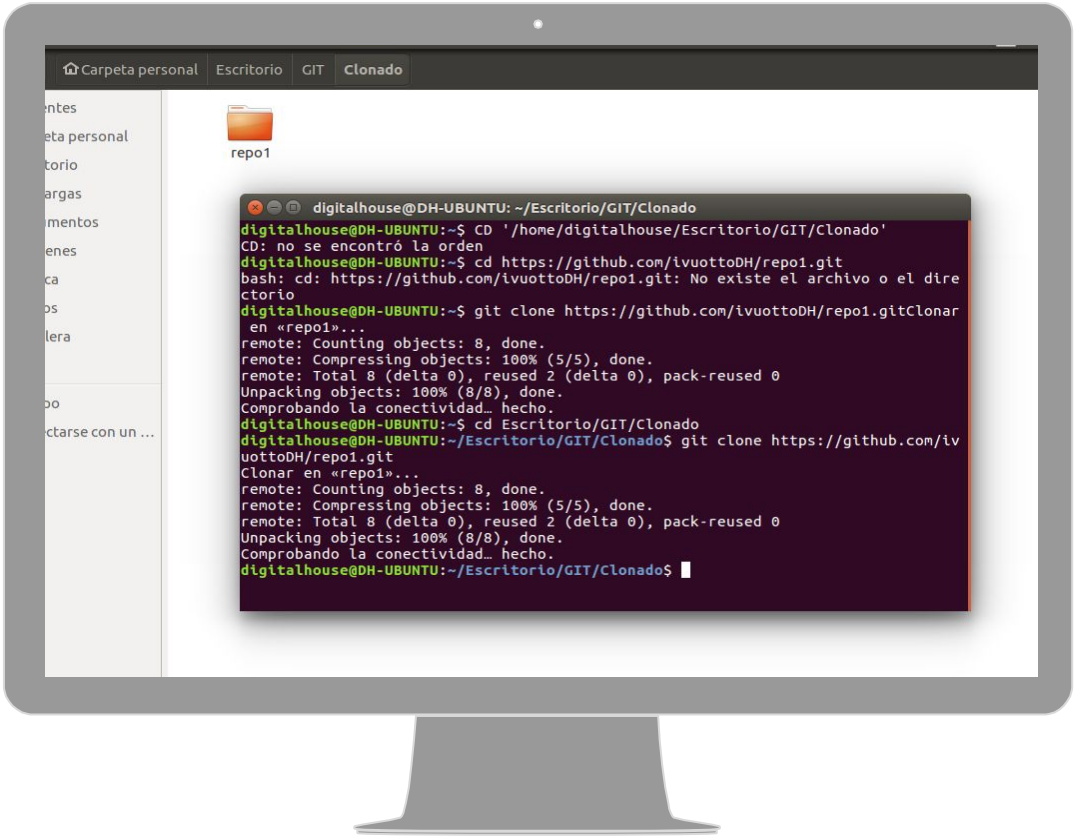
Ésto lo especifico así:

git remote add origin **https://github.com/*user*/*repo*.git**

git remote add origin https://...

Con este comando, le estoy indicando a mi repositorio local, a donde quiero subir los archivos.

La URL la obtendremos al crear un **repositorio en Github.com**



Enviando mis archivos, del repositorio local al repositorio remoto

Para enviar los archivos que tengo en mi repositorio local al repositorio remoto, escribiré la siguiente línea:

git push origin master

git push origin master

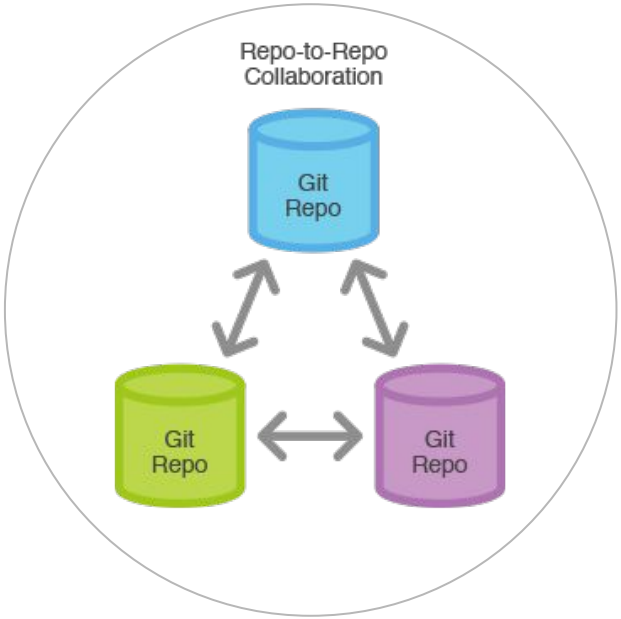
El push, permite enviar los archivos de mi repositorio local al repositorio remoto.

*Al especificar **master**, estoy diciendo que enviaré ese paquete de archivos a dicha rama.*

***Importante**: no se puede hacer un **push** sin antes haber hecho un **commit**.*

Bajando los archivos del repositorio remoto a mi repositorio local

A veces necesitaré bajar mi trabajo a otra computadora, para ello **deberé clonar el repositorio remoto** en dicha máquina.



“

*Para **descargar por 1era vez** un repositorio remoto a otra máquina. Tendré que clonar el mismo en el lugar que desee. El comando que necesito es:*

git clone https://github.com/user/repoName

git clone https://....

git clone**, permite crear una copia idéntica del repositorio remoto en cualquier otra máquina. Para que pueda trabajar con los mismos archivos que tengo hasta ese momento. Después de trabajarlos deberé como siempre **pushearlos.

Ahora, la pregunta es: ¿cómo hago para actualizar los archivos que hice en esta máquina con los archivos en mi máquina oficial?

Si lo que deseo es actualizar los archivos en mi repositorio local con lo existente en el repositorio remoto, deberé:

git pull origin master

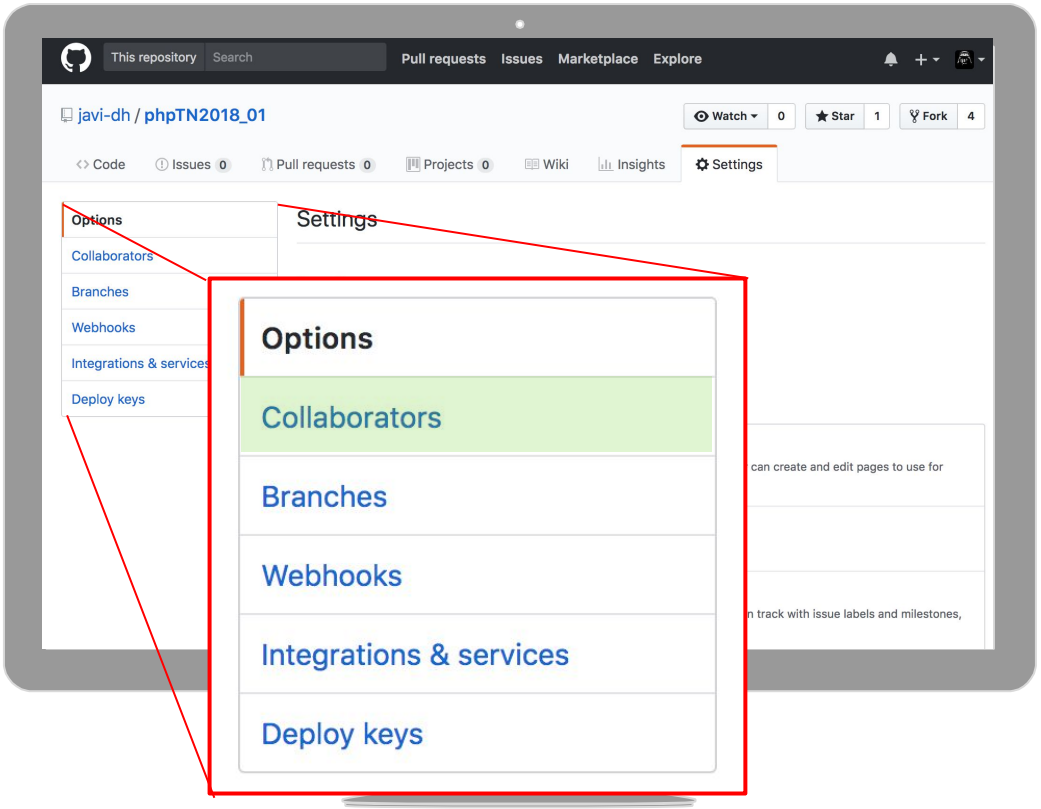
git pull origin master

***git pull**, baja a mi repositorio local, los cambios o archivos nuevos que se hayan **pusheado** al repositorio remoto desde otra máquina,*

Este comando es muy funcional si trabajo con más colaboradores en el mismo proyecto.

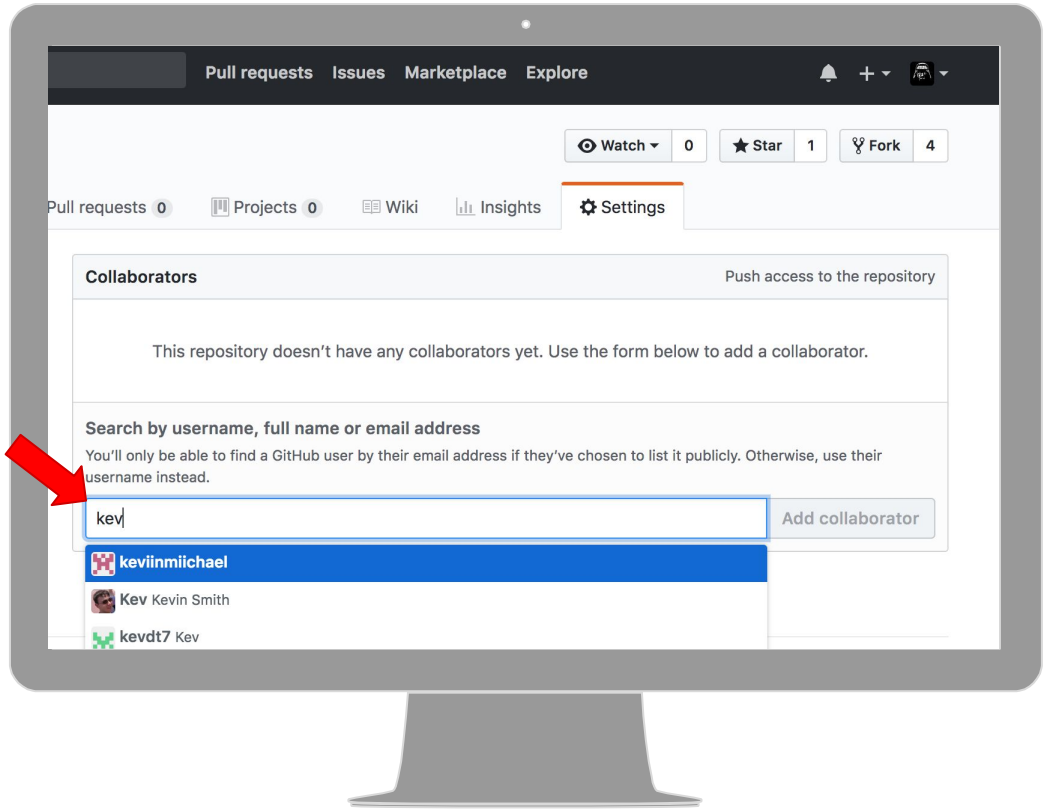
¿Cómo agrego colaboradores al repositorio remoto?

Es muy común que trabaje en equipo, y que quiera agregar a mi repositorio a nuevos miembros para que participen del mismo.



Una vez aquí iré a la opción:

Collaborators



Aquí escribiré el nombre de usuario de mi colega y después pulsaré el botón:
Add collaborator

La persona recibirá un email, donde deberá aceptar la invitación.

De esta manera, agrego colaboradores a mi repositorio remoto.

*Ahora, ellos también tienen el poder de **pushear** su trabajo a mi repositorio.*

*Por ello es importante, al momento de sentarnos a trabajar, **antes de arrancar** hacer un:*

git pull origin master

Receta paso a paso

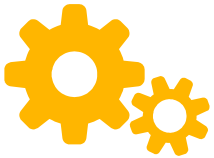
Los siguientes son los **paso a paso** que necesito aprender para trabajar con nuestro repositorio.

1. **git init** *//crea el repositorio*

2. **git config user.name "hanSolo"**
//agrega mi identidad - username

3. **git config user.email "hansolo@starwars.com"**
//agrega mi identidad - email

4. **git remote add origin https://github.com/....**
//conecta con el repositorio remoto



5 **git add .**
//agrega todos los cambios al repo local

6 **git commit -m 'mensaje del commit'**
//hito histórico - comitea los cambio hechos

7 **git push origin master**
//manda los cambios al repositorio remoto

... **Los pasos del 5 al 7,** se repiten tantas veces como funcionalidades vayamos sumando al proyecto.

Y recuerda:
In case of fire

1. git commit

2. git push

3. leave building

Webs de consulta

Siempre viene bien tener a la mano documentación, para ello puedo visitar: