La Terminal, GIT & GitHub

1

Terminal - Consola - CMD

¿Qué rayos es eso?

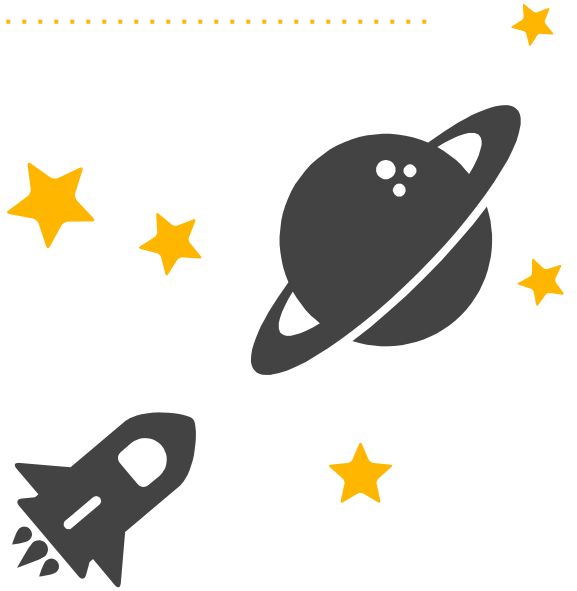
El terminal es un programa que está presente en todos los sistemas operativos y por medio del cual se pueden dar órdenes al sistema a través de líneas de comando.

¿Por qué usar la terminal?

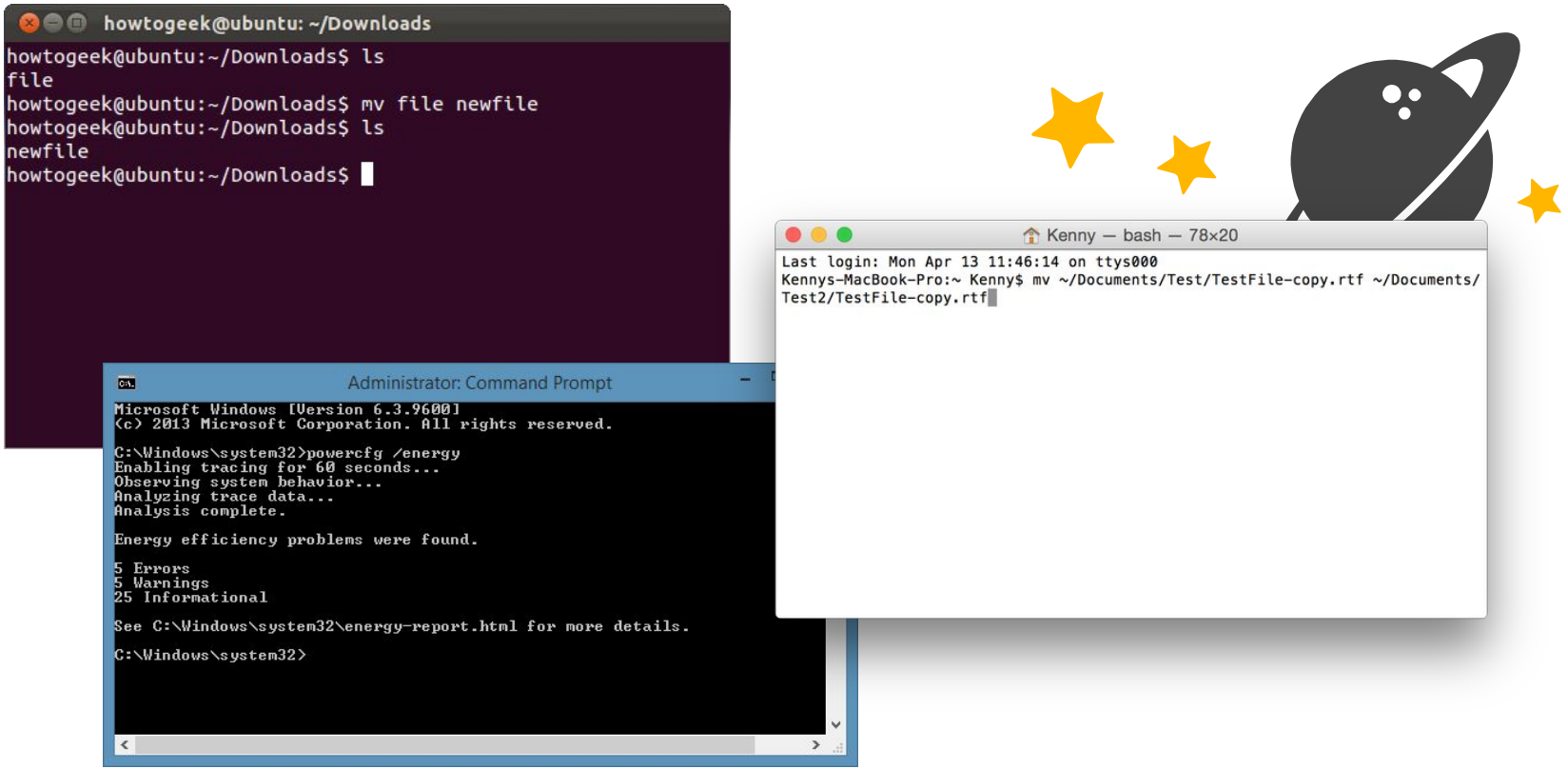
- Para tener mayor control sobre el SO.
- Porque es muy común en los entornos de desarrollo.
- Porque algunos lenguajes de programación lo *"requieren"*.

Si sabemos usar la terminal y nos acostumbramos a la misma, podremos optimizar mucho nuestro trabajo de programar.

¿Donde está la terminal?



Sea cual sea el SO que estemos 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 debemos saber

ls (en Mac y Linux muestra los archivos de la carpeta en la que estamos ubicados, en Windows también si usamos el **PowerShell**)

dir (en Windows muestra los archivos de la carpeta en la que estamos ubicados)

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

cd *nombre-carpeta* (nos permite acceder a la carpeta descrita)

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

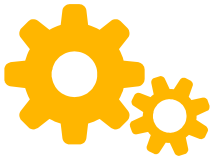
(limpia todo lo que hayamos escrito en la consola / Mac y Linux. En Windows en el **PowerShell**)

cls

(limpia todo lo que hayamos escrito en la consola / Windows)

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

Vamos a realizar una pequeña práctica que nos permita familiarizarnos más con la consola

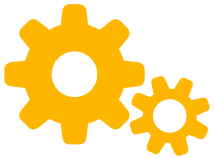


1. Con la consola, llegar hasta la carpeta ***Escritorio ó Desktop*** de nuestra máquina.

2. Una vez allí, crear una carpeta llamada "**test-consola**".

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.

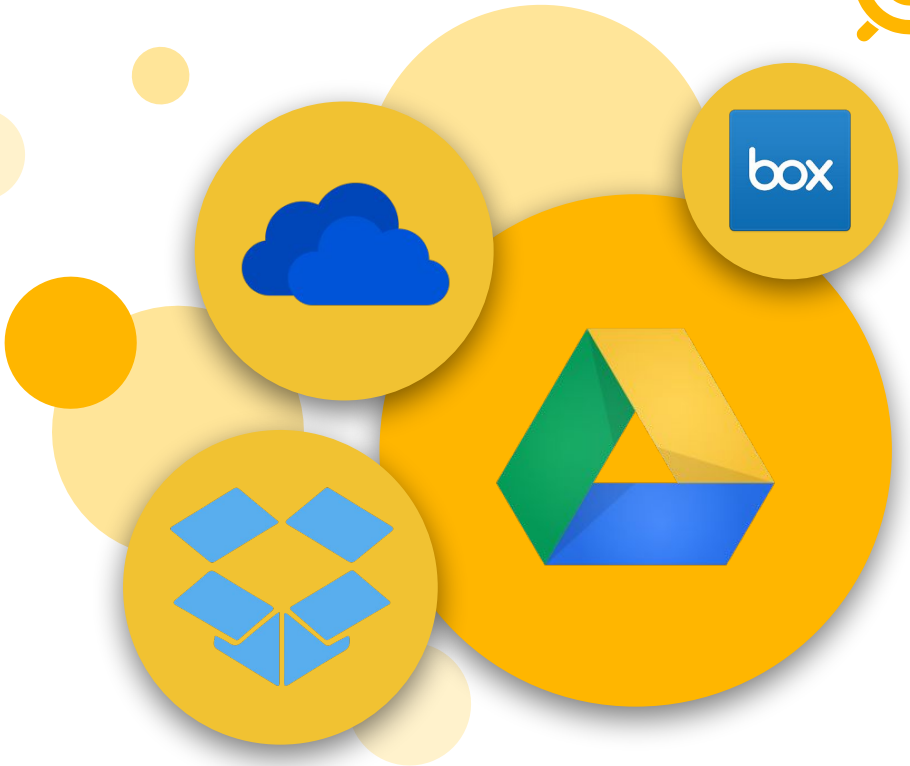
6. ¡Opps! Cometimos un error y creamos un archivo que no era, ahora tenemos que borrar por consola el archivo "**prueba.html**".

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

8. Finalmente, probar el siguiente comando "**atom .**" ¿Qué sucedió?, *atentís*, puede no hacer nada en windows.


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

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



¿PODEMOS COMPARTIR ASÍ NUESTROS PROYECTOS?

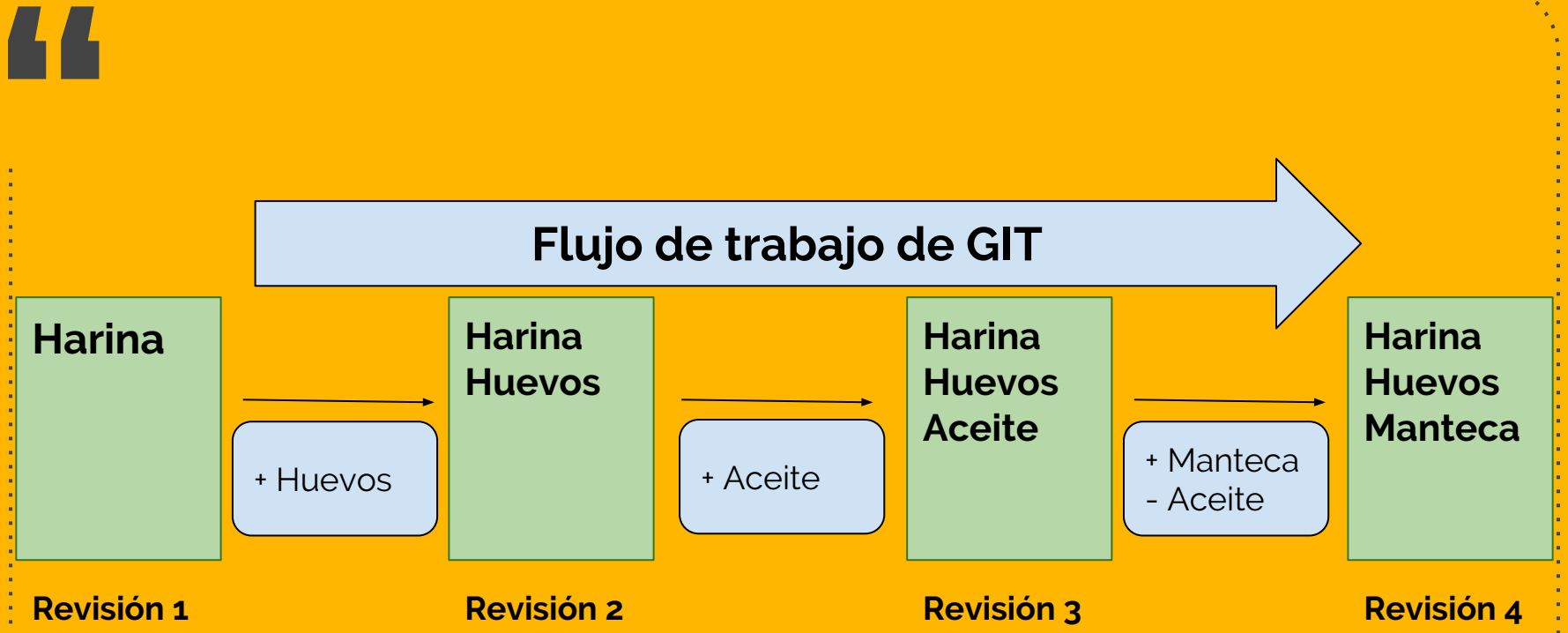
Necesitamos un software

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

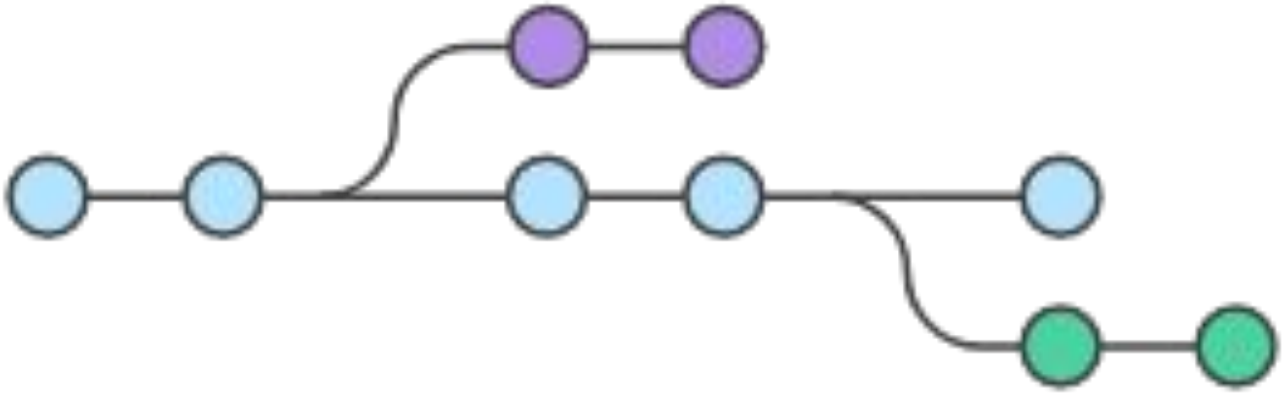
git

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 trabajamos con Git, hablamos de trabajar con un ***REPOSITORIO.*** *El cual es un lugar en donde se almacenan nuestros 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****.*

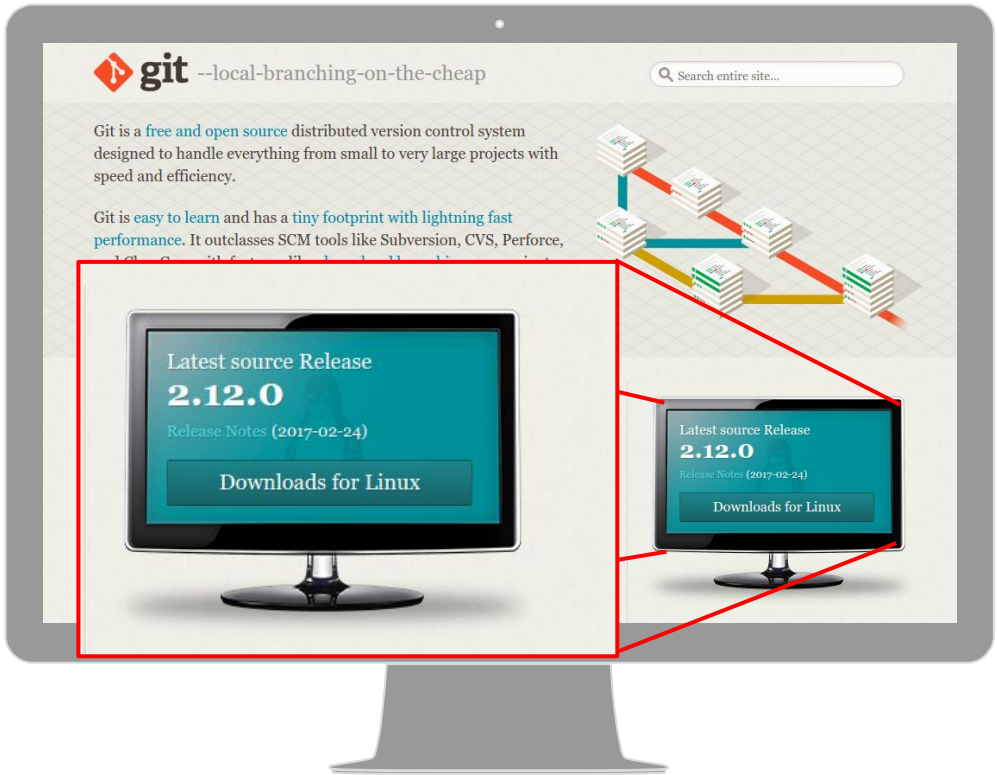
Ramas / Branches

git

Repo Local

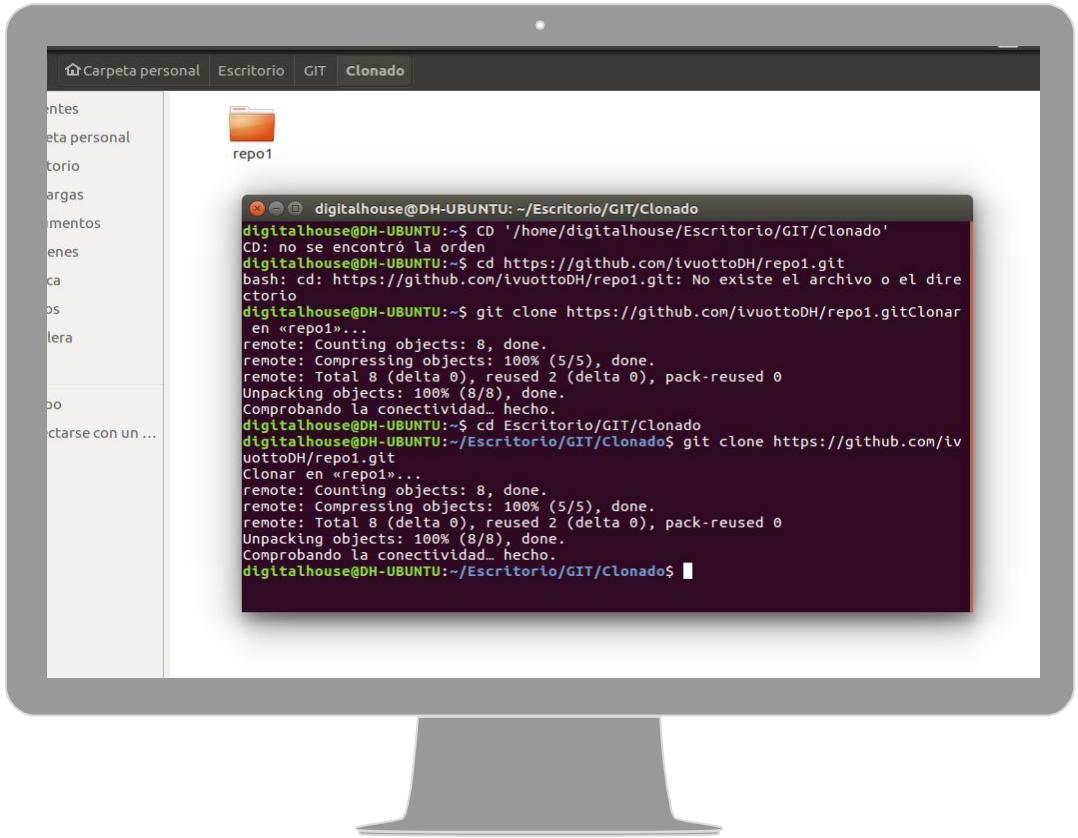
GitHub

Repo Remoto



Instalando GIT en nuestras máquinas

https://git-scm.com/



Creando el repositorio local

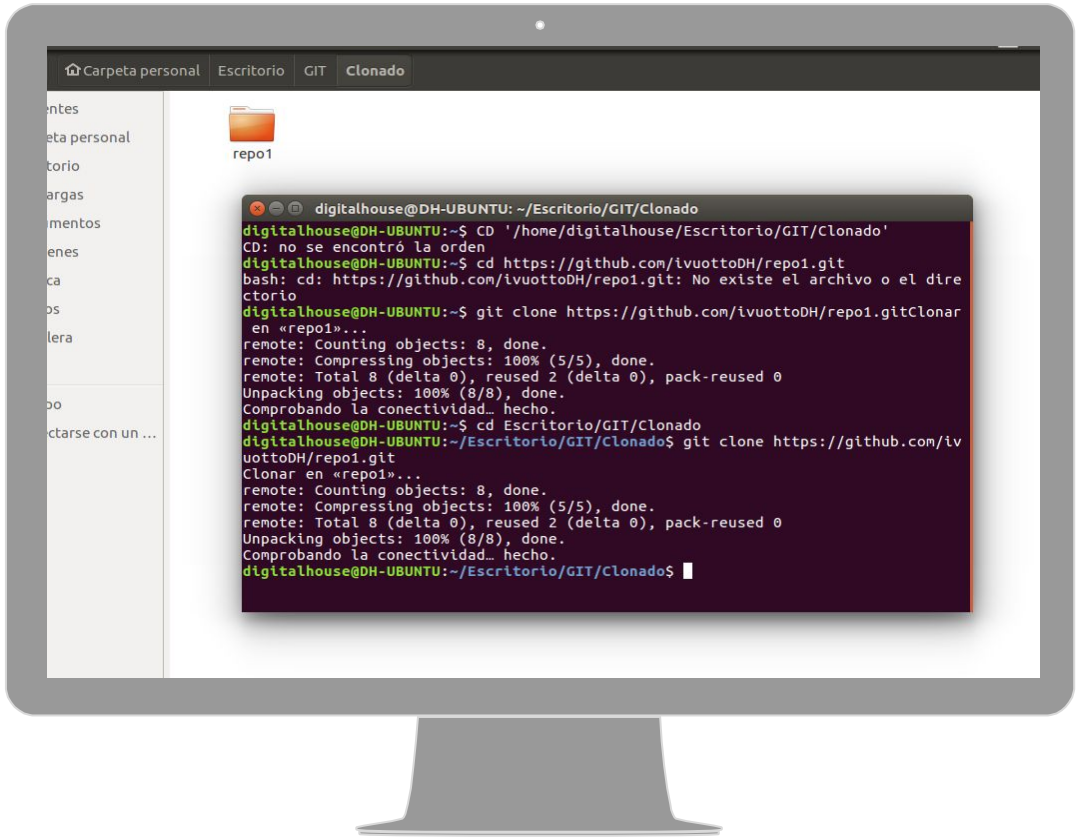
Lo primero será ubicarnos en donde queremos crear el repositorio y posteriormente escribir el siguiente comando:

git init

git init

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

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



Agregando nuestra identidad

Para que todo lo que hagamos quede "firmado" por nosotros, necesitamos decirle al repositorio quien somos, así:

git config user.name "Jhon_Doe"

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

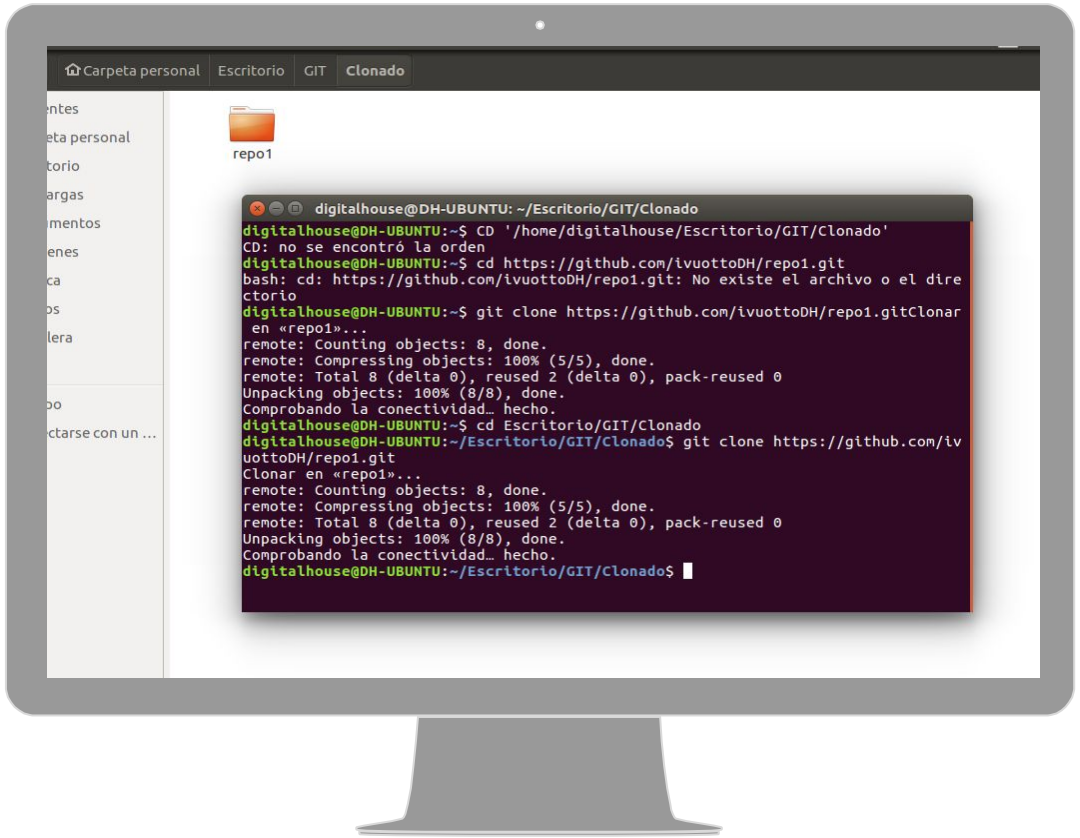
“

git config user.name " "

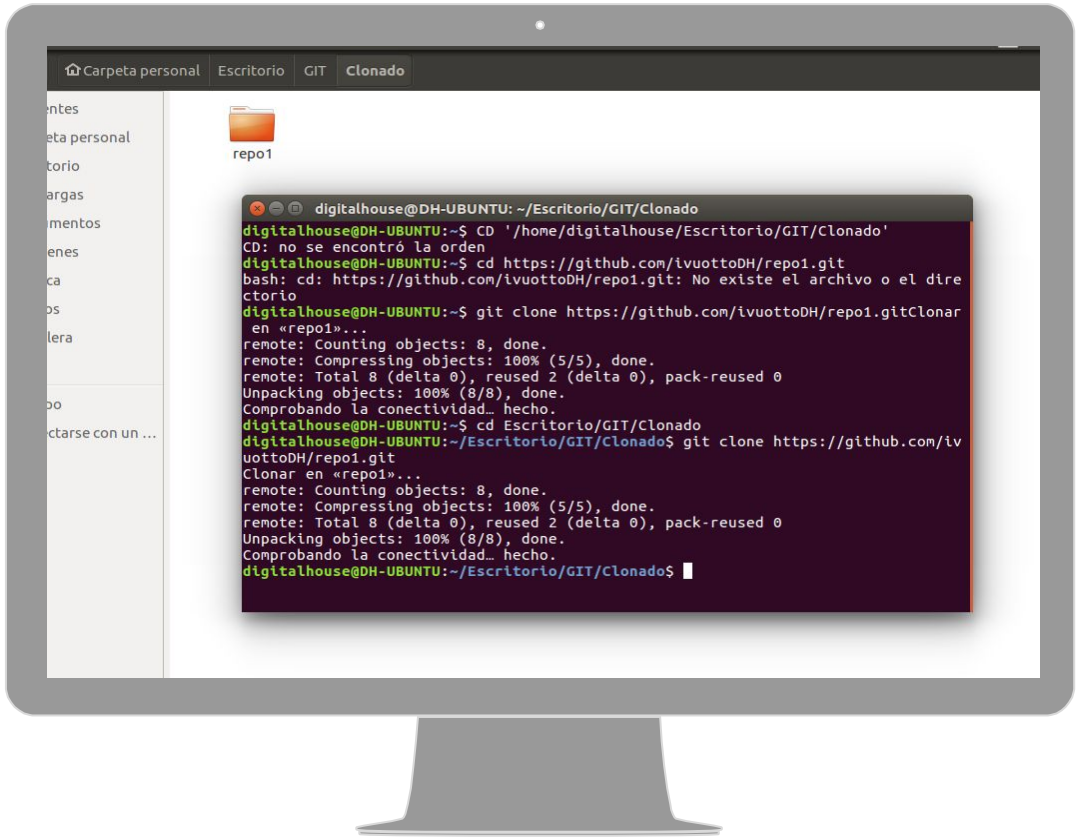
Dentro de las comillas pondremos nuestro usuario de Github.com

git config user.email " "

Dentro de las comillas pondremos el email con el que nos registramos en Github.com



Asignando nuestro repositorio remoto

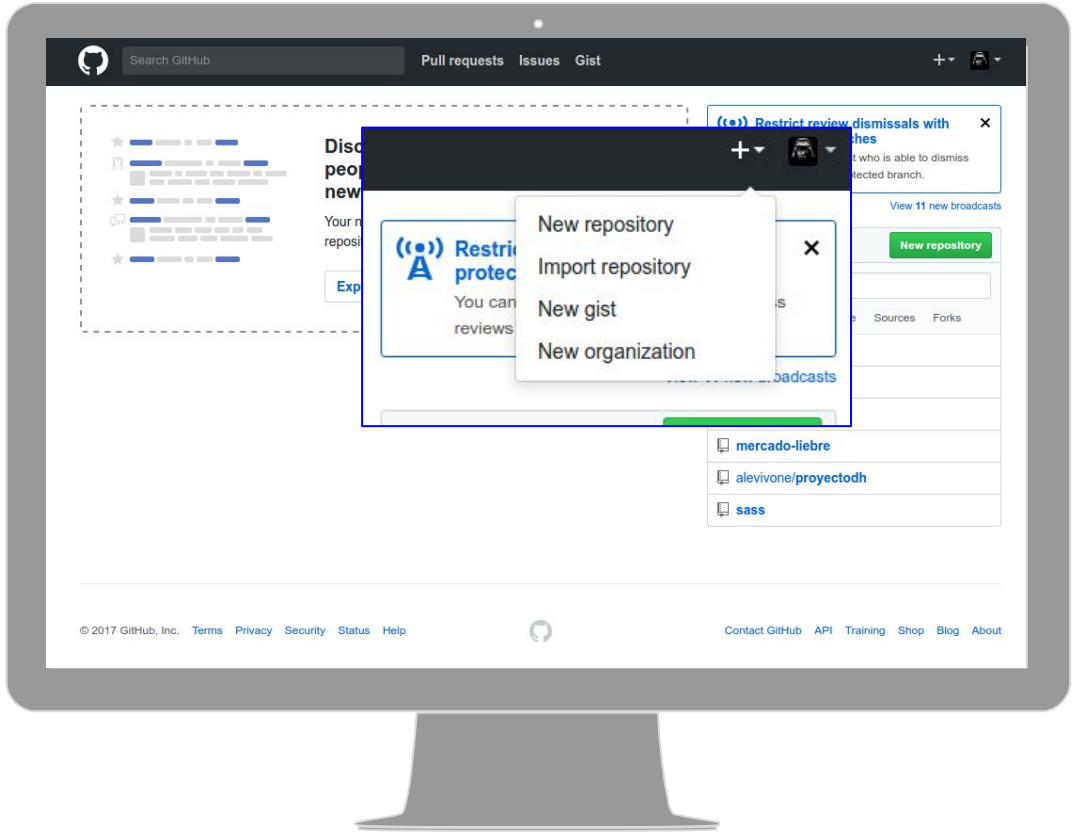


1ero

Vamos a crear nuestro repo en Github

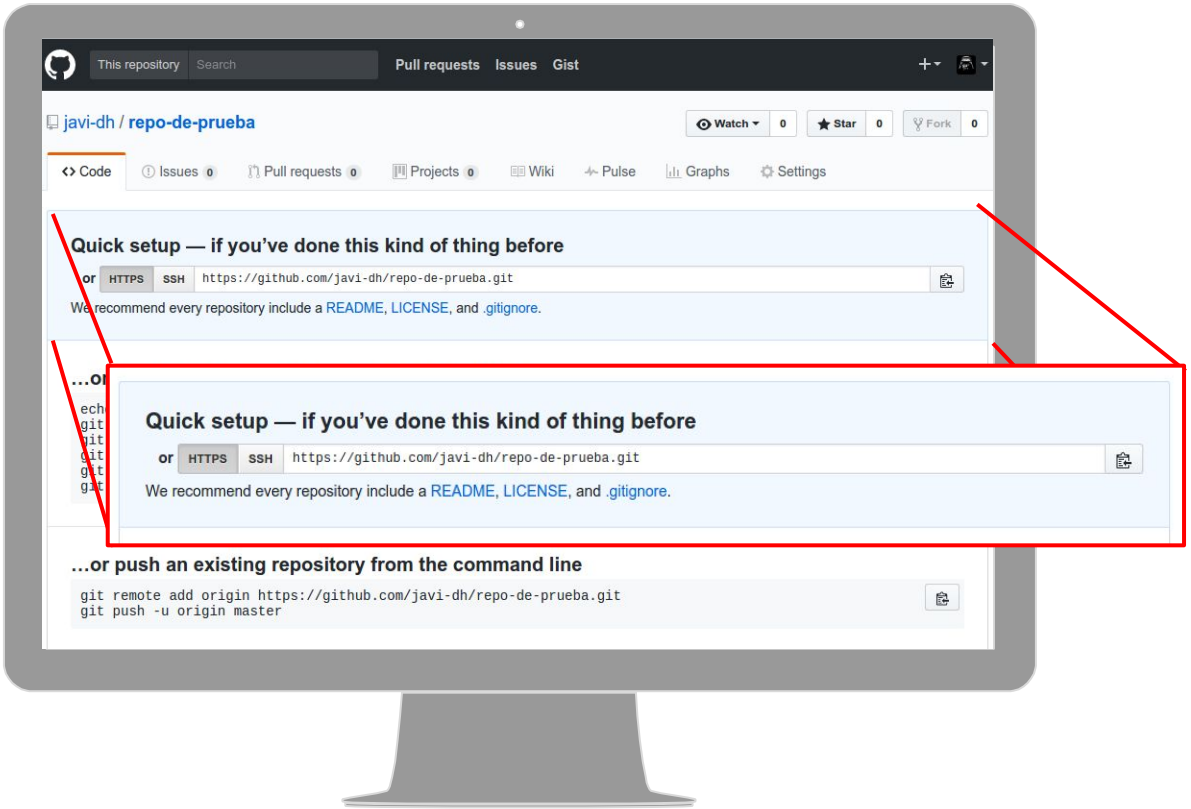
¿Cómo creamos el repositorio remoto?

Para ello vamos a utilizar nuestra cuenta de GitHub que creamos previamente.



Logueados en nuestra cuenta de GitHub vamos al ícono + y ahí elegimos la opción **New Repository**.

El nombre que elijamos puede ser cualquier, uno que no hayamos usado para otro repositorio.
De resto **NO TOCAR** nada más, solo el botón **CREATE**.



Luego veremos esta pantalla y ésta URL es la que necesitamos tener a mano en el paso de: **Asignando nuestro repositorio remoto.**

“

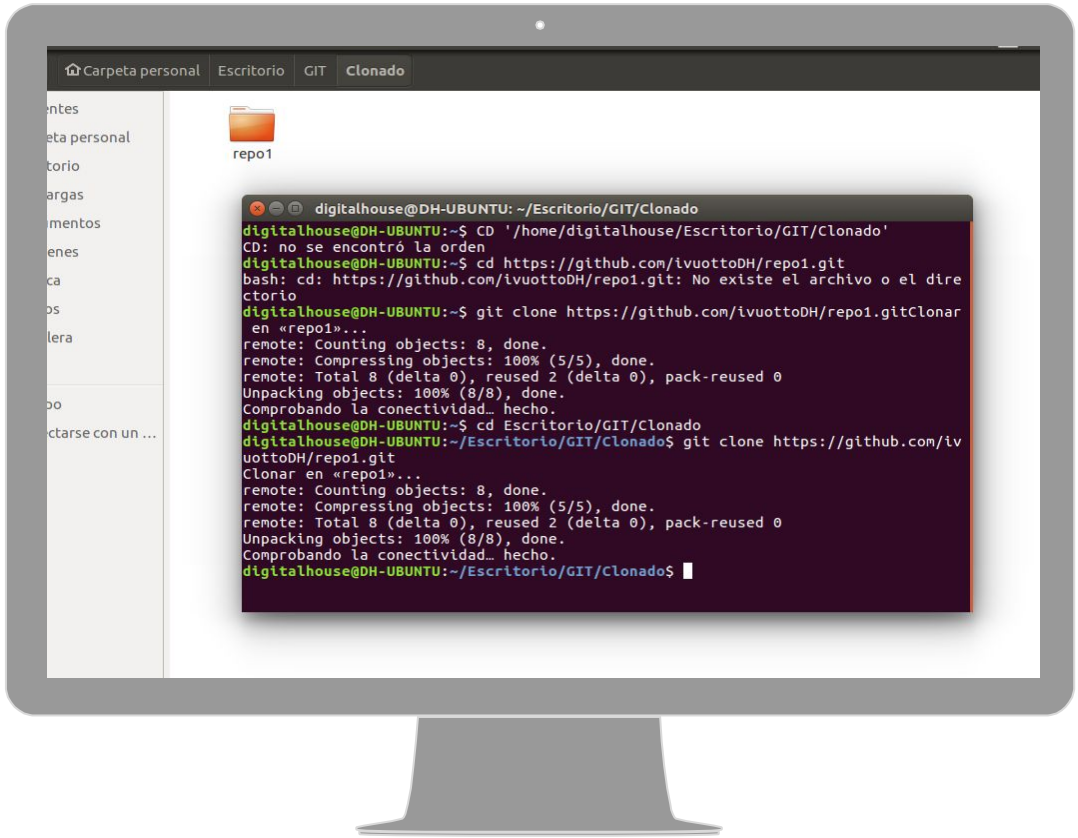
*Habiendo creado el **Repositorio Remoto** y para que nuestro **Repositorio Local** sepa a donde queremos subir nuestros archivos tenemos que especificarlo así:*

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

git remote add origin https://...

Con este comando, le estamos indicando a nuestro repositorio local, a donde queremos llevar (repositorio remoto) nuestros archivos.

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



Adicionando nuestros archivos al repositorio local

*“Hasta el momento, nuestros archivos no han sido agregados **temporalmente** al repositorio (**stage**) para ello tendremos que escribir el siguiente comando:*

git add .

ó

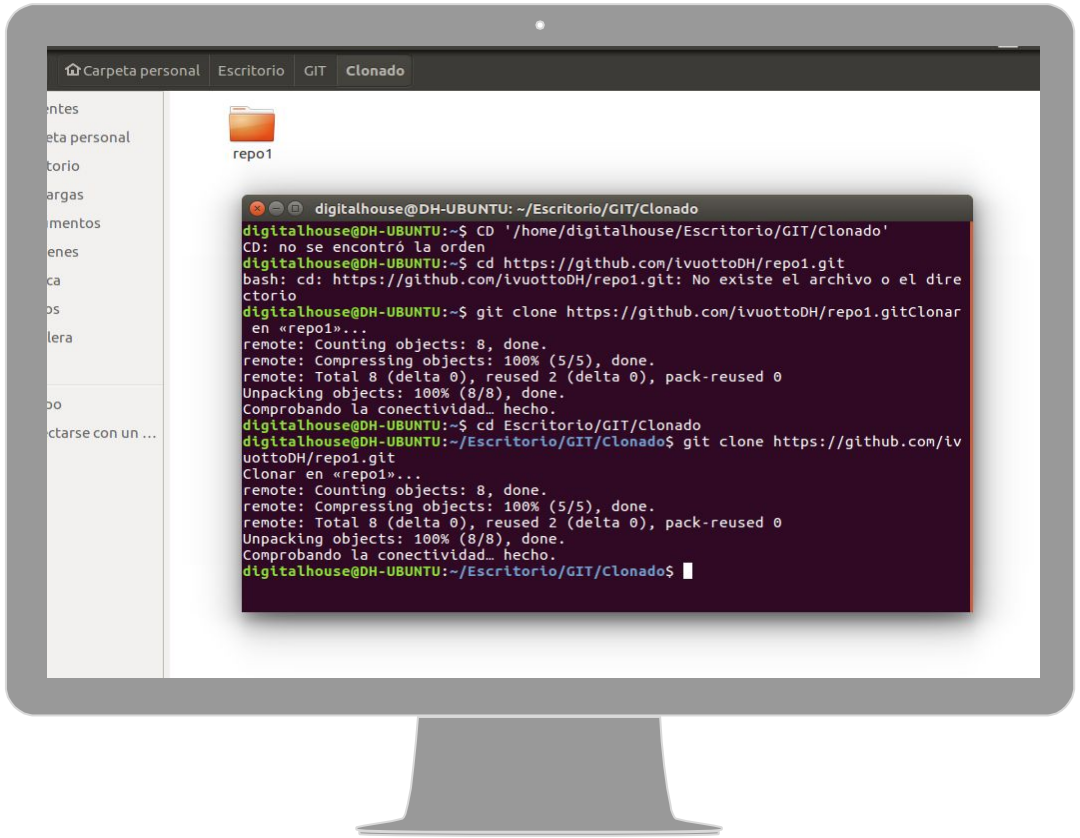
git add archivo.txt

git add .

*Agrega al **stage** (de manera temporal) todos los archivos que hayamos creado en nuestro proyecto.*

git add archivo.txt

*Agrega al **stage** (de manera temporal) solamente el archivo referenciado.*



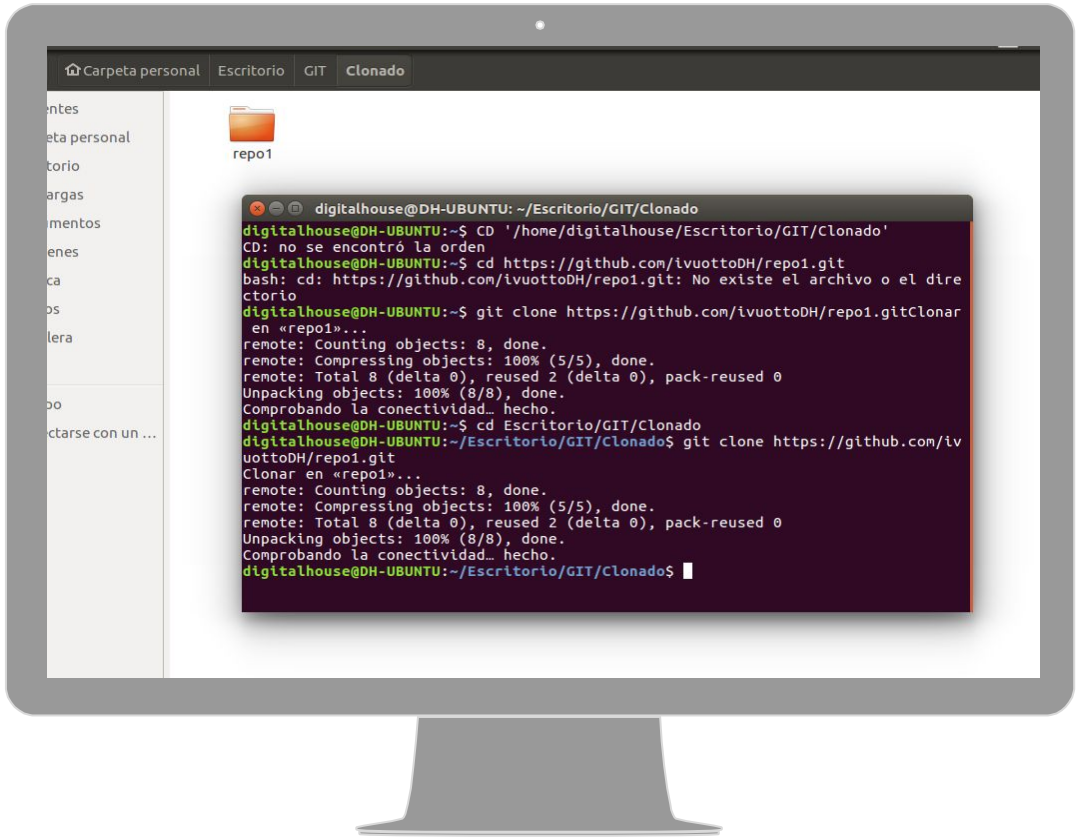
Testeando el status de nuestro repositorio

Cada vez que deseemos comprobar o verificar el estado de nuestro repositorio podremos escribir la siguiente línea de comando:

git status

git status

*Analiza el estado del repositorio, nos dirá si hay archivos que no se han agregado temporalmente al **stage** así como también si hay archivos agregados al stage pero no de forma (**commit**).*



Agregando oficialmente los archivos al stage

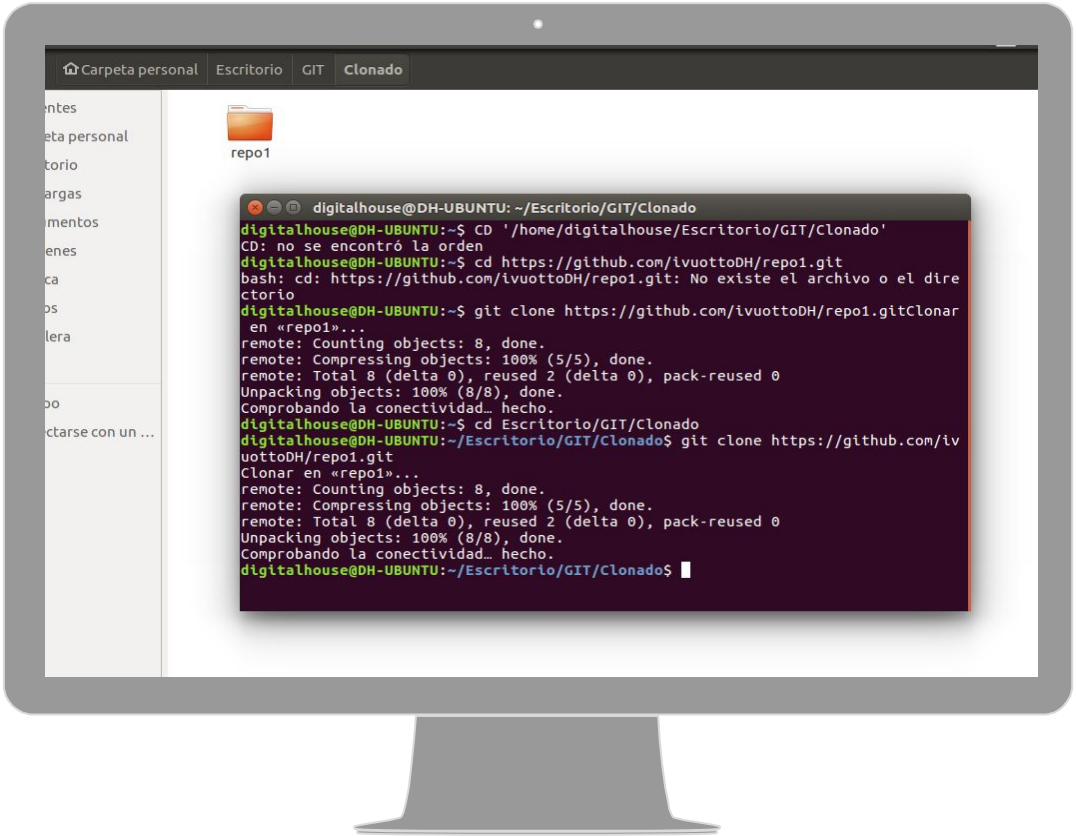
*Para finalmente confirmar que los archivos agregados al **stage** los queremos de **manera definitiva** escribiremos:*

git commit -m "un mensaje cualquier"

git commit -m "un mensaje cualquier"

*La directriz **commit**, le indica al repositorio que los archivos los queremos agregar de manera oficial. La **-m** indica que a continuación agregaremos un mensaje que especifique qué trabajo hicimos.*

*Los **commits** sirven como pequeños **backups** a los cuales podremos volver fácilmente si así lo necesitáramos.*



Enviando nuestros archivos del repositorio local al repositorio remoto

Para enviar los archivos que tenemos en nuestro repositorio local al repositorio remoto, escribiremos la siguiente línea:

git push origin master

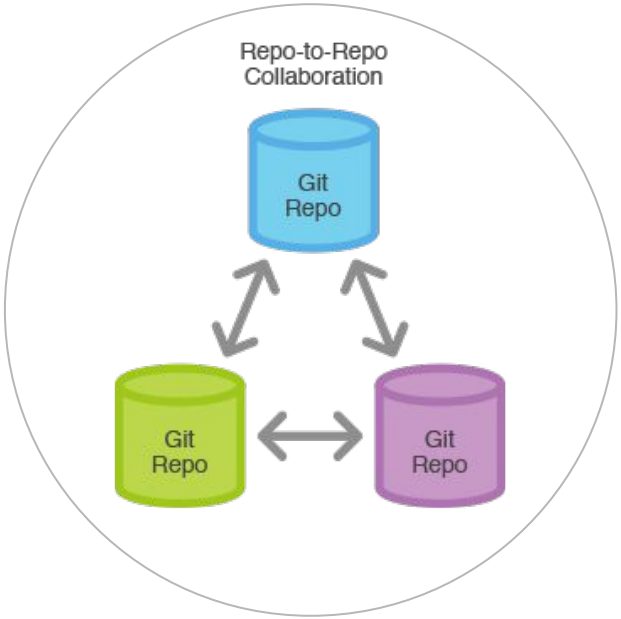
git push origin master

El push, permite enviar los archivos de nuestra máquina (repositorio local) al repositorio remoto.

*Al especificar **master**, estamos diciendo a qué rama del repositorio queremos enviar nuestros archivos.*

Bajando los archivos de repo remoto a nuestro repo local

A veces queremos bajar nuestro trabajo a la computadora de casa u otra, para ello **necesitaremos clonar el repo remoto** en nuestra máquina.



“

*Para **descargar por 1era vez** un repositorio remoto a nuestra máquina. Tendremos que clonar el mismo en el lugar que deseemos. El comando que necesitamos será:*

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

git clone https://....

git clone**, permite crear una copia idéntica del repositorio remoto en nuestra máquina. Para que podamos trabajar con los mismo archivos que tengamos hasta ese momento. Después de trabajarlos deberemos como siempre **pushearlos.

Ahora, la pregunta es ¿cómo hago para actualizar los archivos que hice en la máquina original con estos nuevos archivos?

Si lo que deseamos es actualizar los archivos en nuestro repositorio local con lo existente en el repositorio remoto, deberemos:

git pull origin master

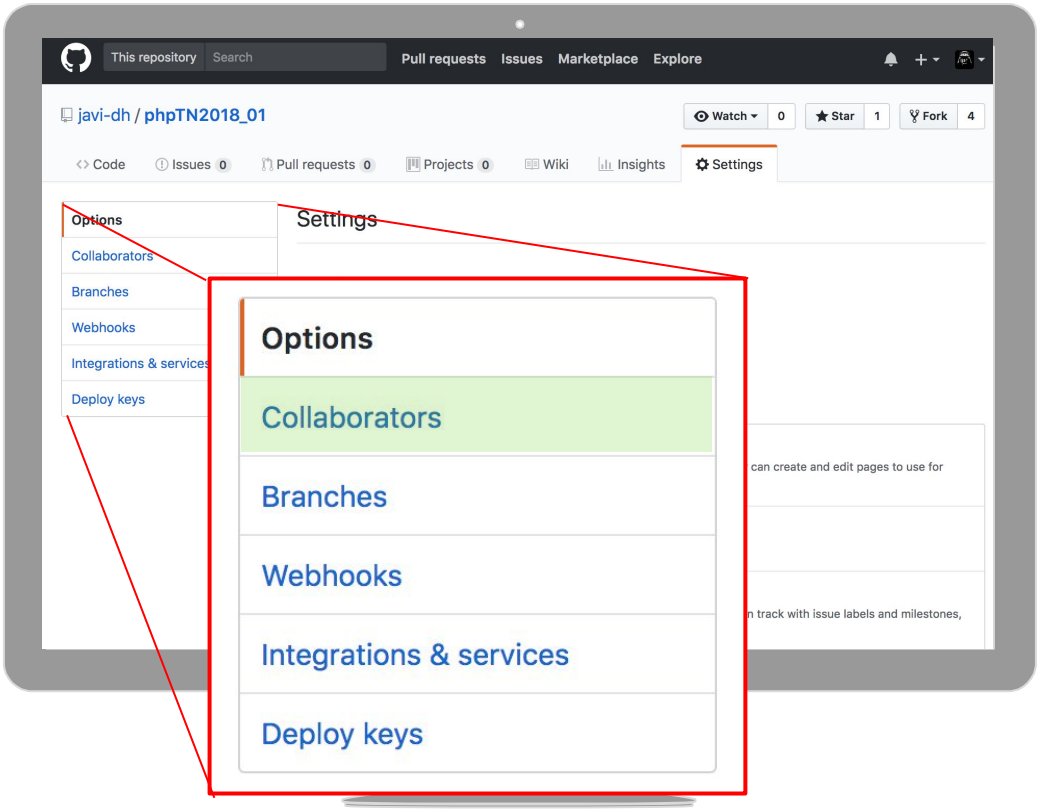
git pull origin master

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

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

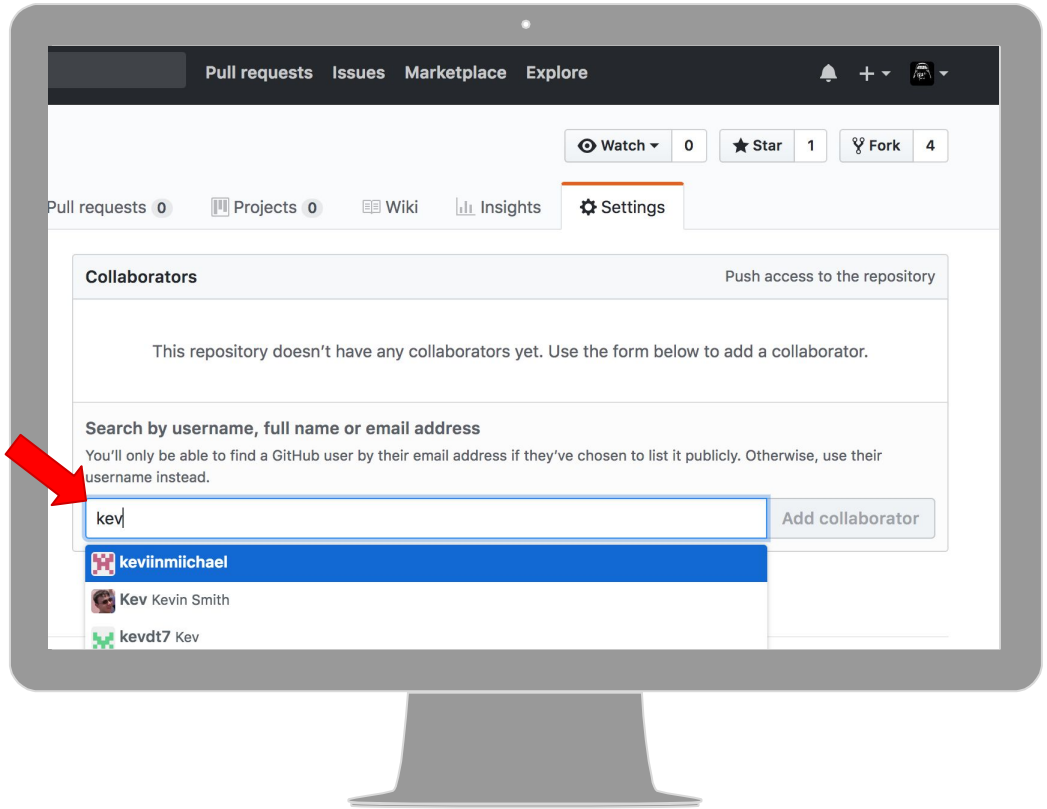
¿Cómo agregamos colaboradores al repositorio remoto?

Es muy común que trabajemos en equipo, y que queramos agregar a nuestro repositorio a nuevos miembros para que participen del mismo.



Una vez aquí iremos a la opción:

Collaborators



Aquí escribiremos el nombre de usuario de nuestro colega y después pulsaremos el botón:

Add collaborator

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

De esta manera, agregamos colaboradores a nuestro repositorio remoto.

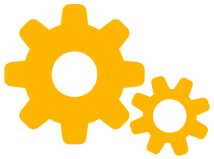
*Ahora, ellos también tienen el poder de **pushear** su trabajo a nuestro repositorio.*

*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 los **paso a paso** que necesitamos para trabajar con nuestro repositorio.

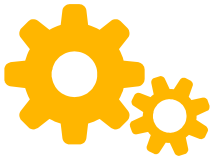


1. **git init** *//crea el repositorio*

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

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

4. **git remote add origin https://github.com/....**
   //apunta al 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 de 5 al 7, se repiten tantas veces como funcionalidades vayamos sumando al proyecto.

Y recuerden:
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 podemos visitar: