



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.

¿Dónde 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.

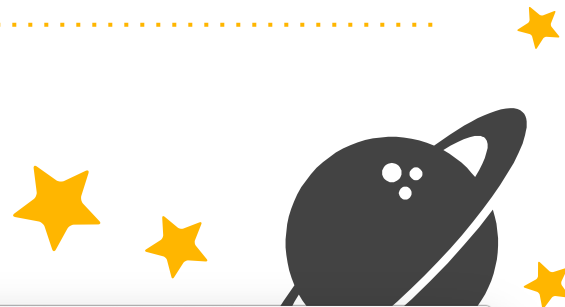
Energy efficiency problems were found.

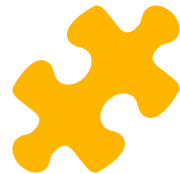
5 Errors
5 Warnings
25 Informational

See C:\Windows\system32\energy-report.html for more details.

C:\Windows\system32>
```

```
Kenny — bash — 78x20
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)

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 un 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)

cls

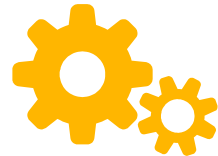
(limpia todo lo que hayamos escrito en la consola / Windows)



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

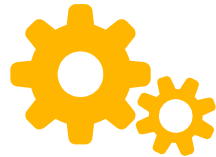
Una vez allí, crear una carpeta llamada "**test-consola**".

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

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

7

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

8

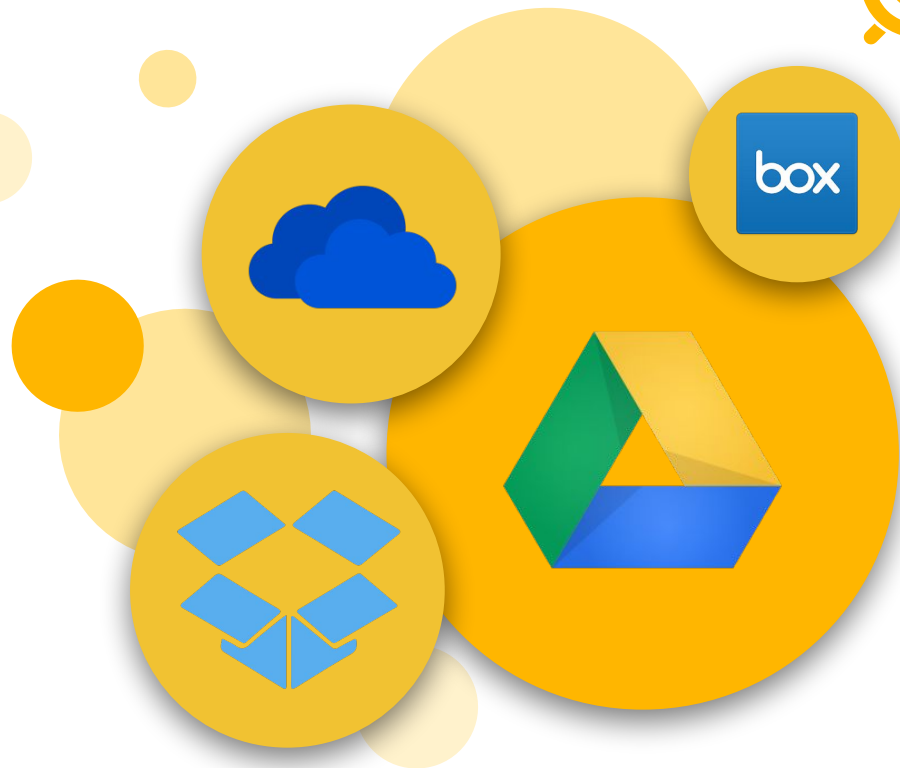
Finalmente, probar el siguiente comando **"atom ."** ¿Qué sucedió?, *atentís*, puede no hacer nada en windows.



GIT & GitHub

**BACK
TO THE FUTURE**

¿Cómo compartimos archivos en la actualidad?

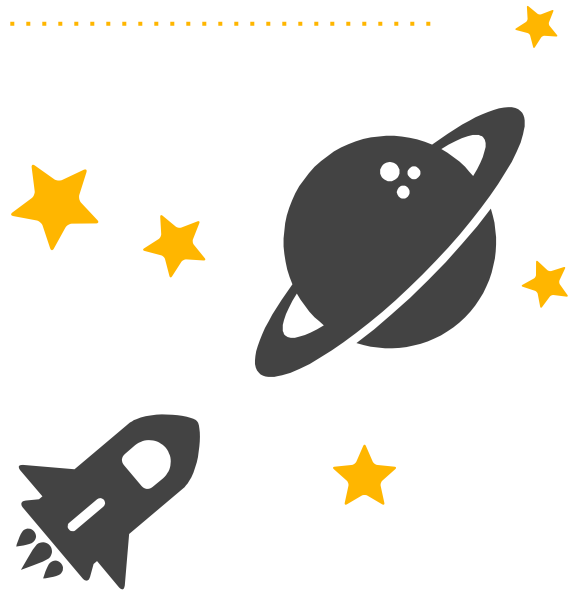


¿PODEMOS
COMPARTIR ASÍ
NUESTROS
PROYECTOS?



Necesitamos un software

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





git





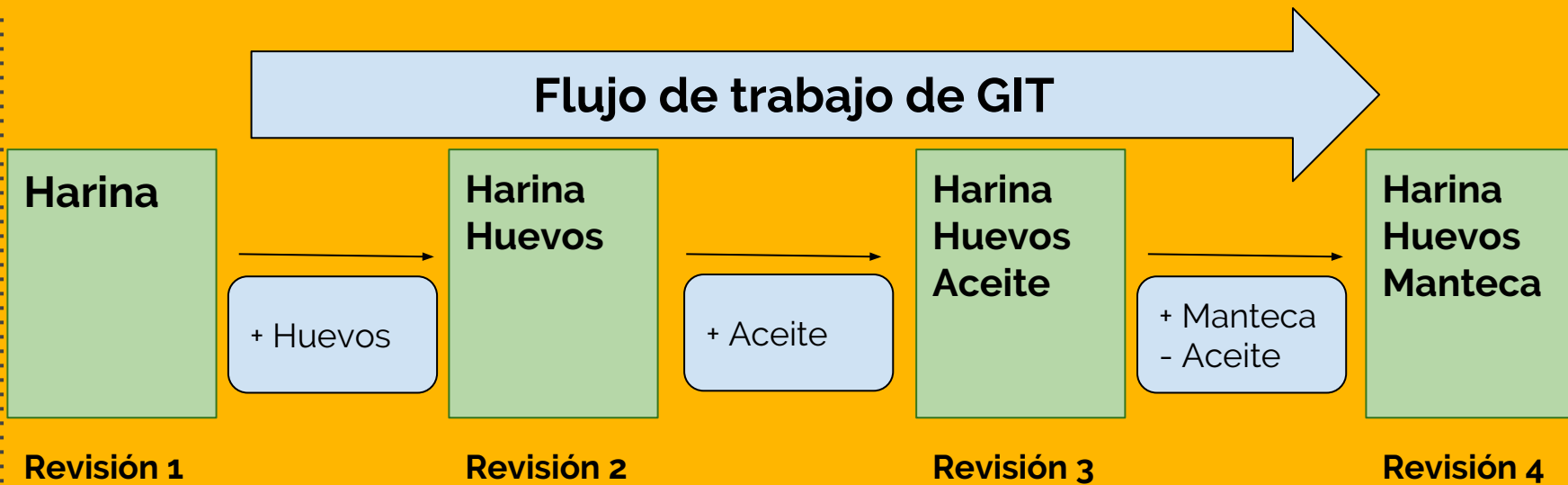
Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente.



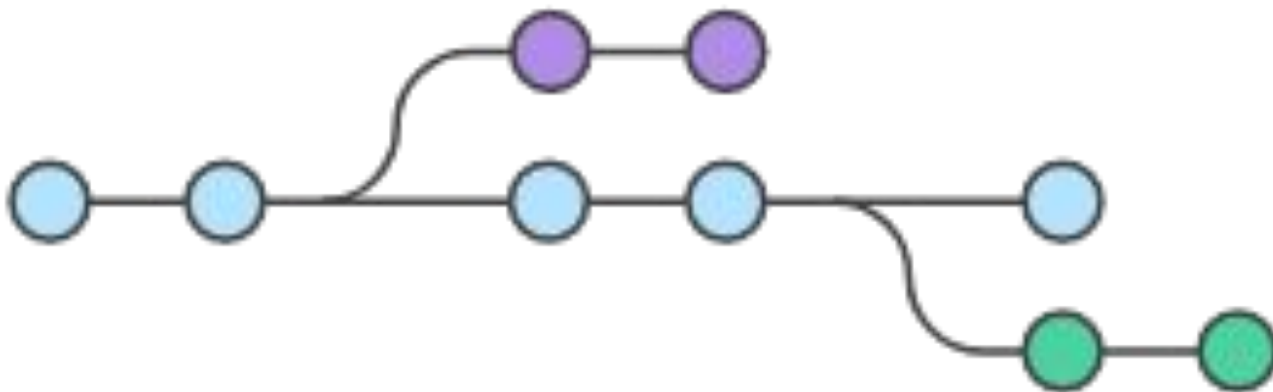
*Cuando trabajamos con Git, hablamos de trabajar con un **REPOSITORIO**. El cual es un lugar en donde se almacenan nuestros archivos. Hay dos tipos de **REPOSITORIOS** el **local** y el **remoto**.*



Flujo de trabajo de GIT



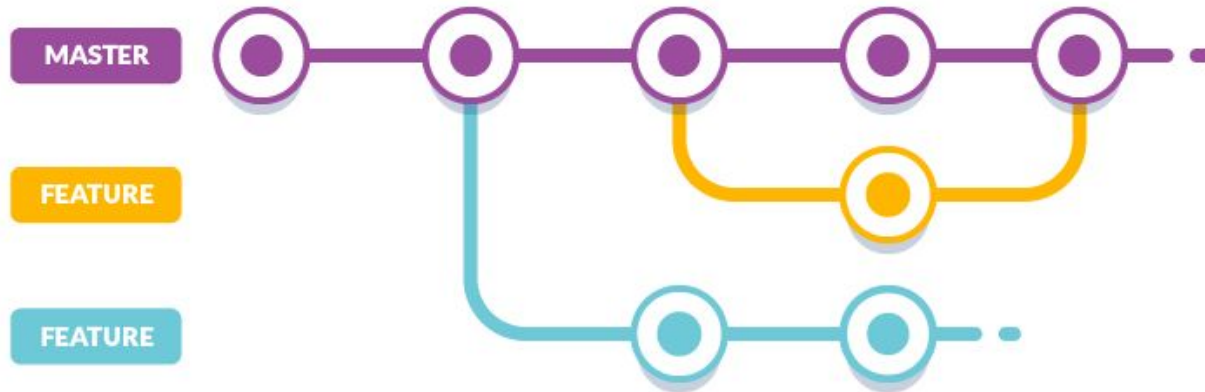
“ Ramas / Branches





*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





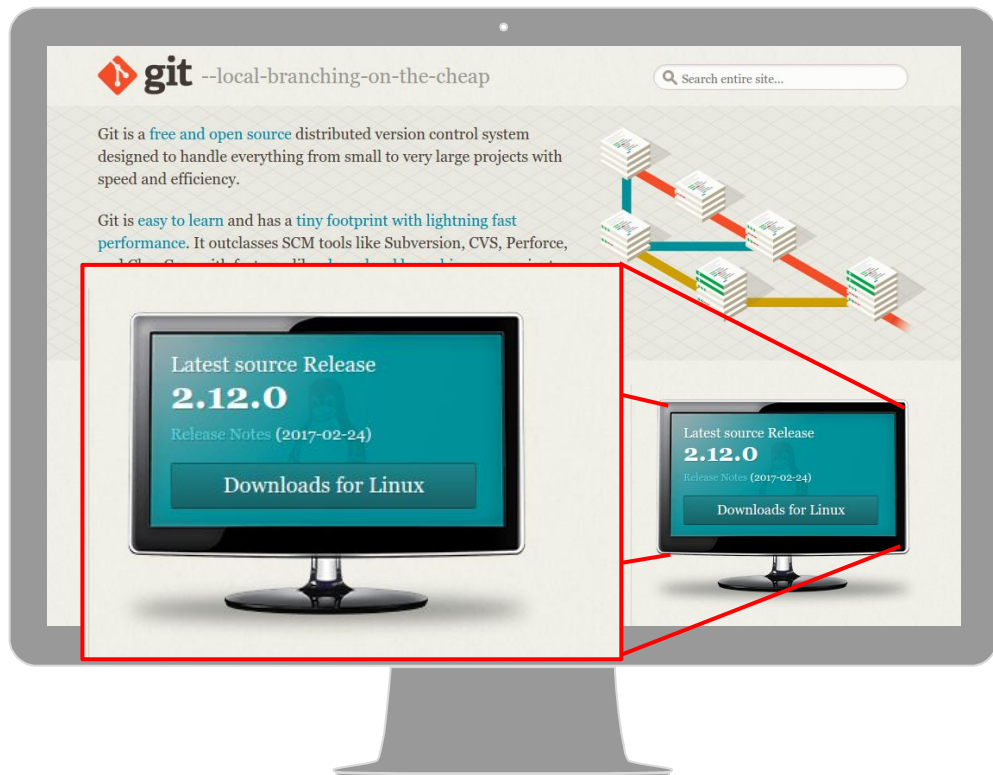
git

Repo Local



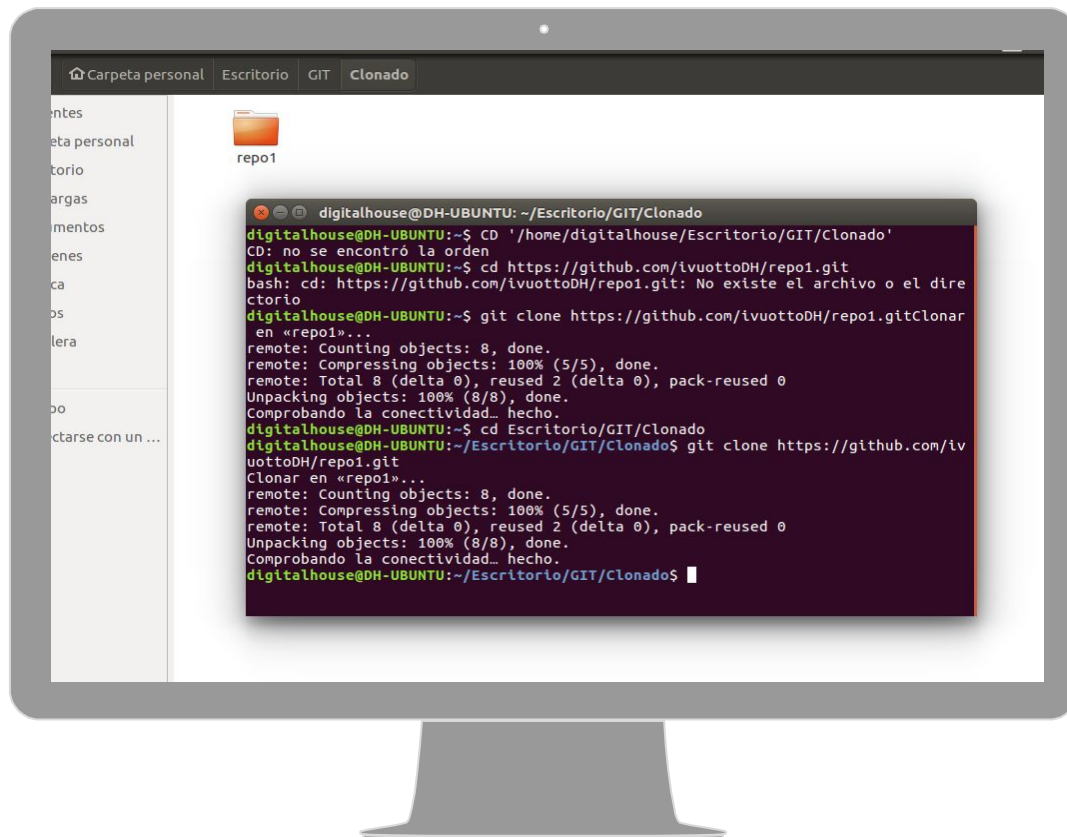
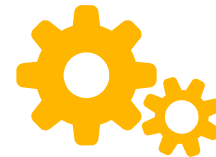
GitHub

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:*

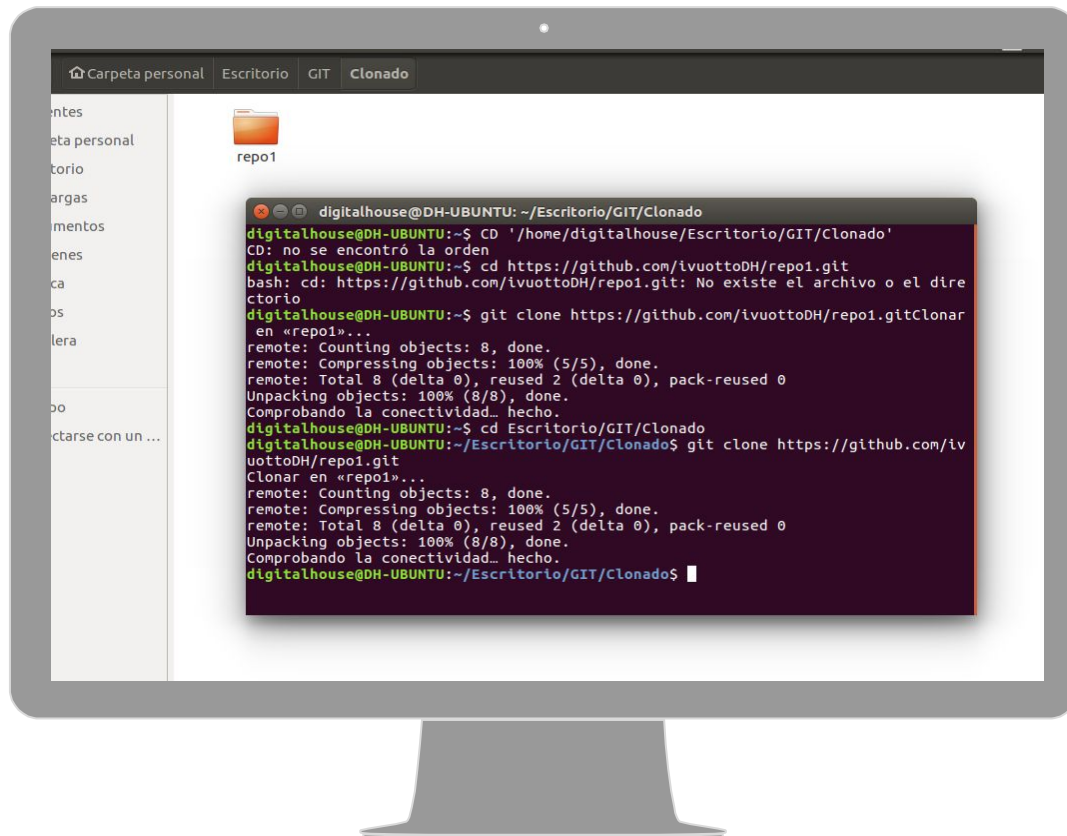
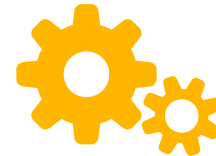
git init



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"
```

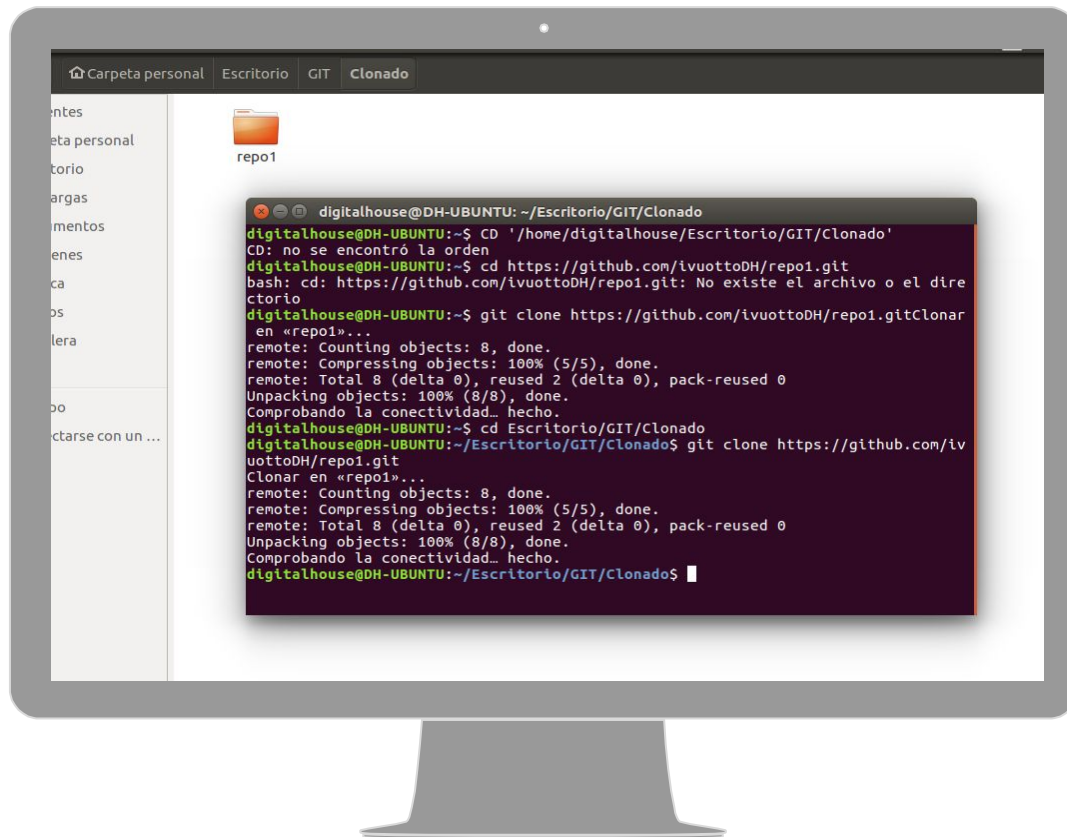
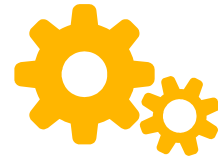


git config user.name " "

*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

“

*Para que nuestro repositorio local sepa a donde queremos subir nuestros archivos (**repositorio remoto**) tenemos que especificarlo así:*

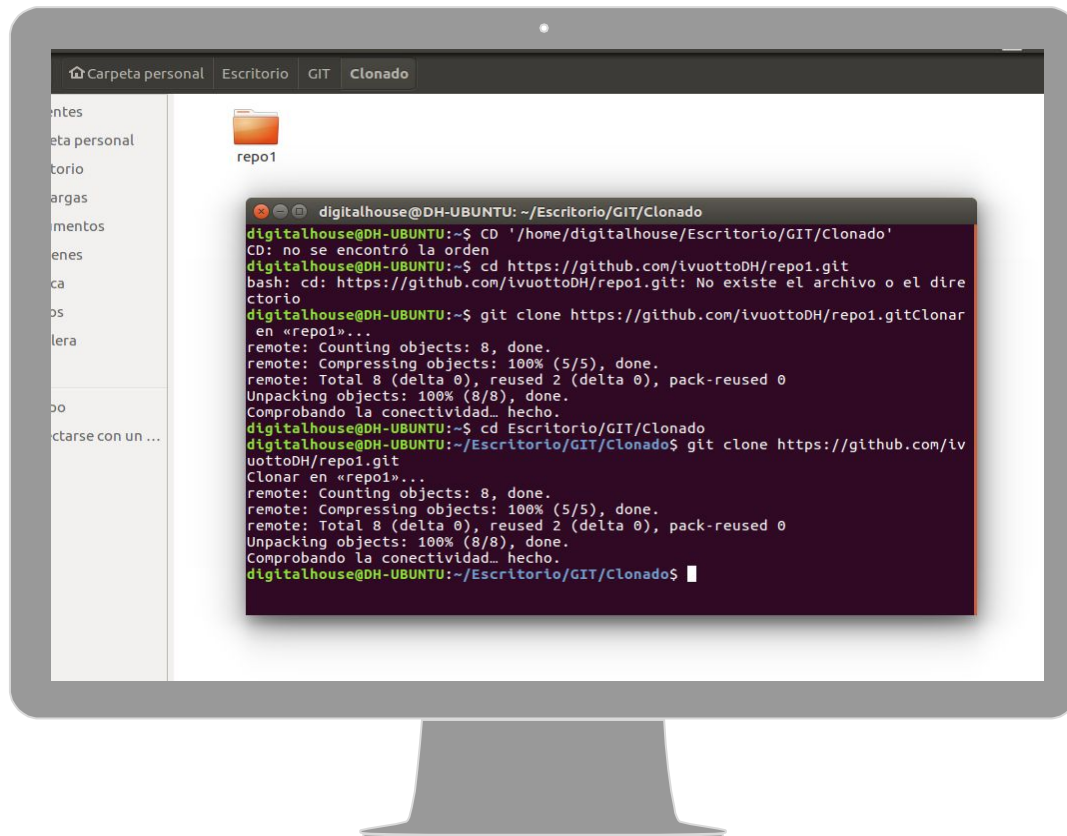
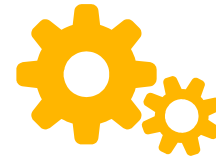
git remote add origin https://github.com/user/repo.git



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

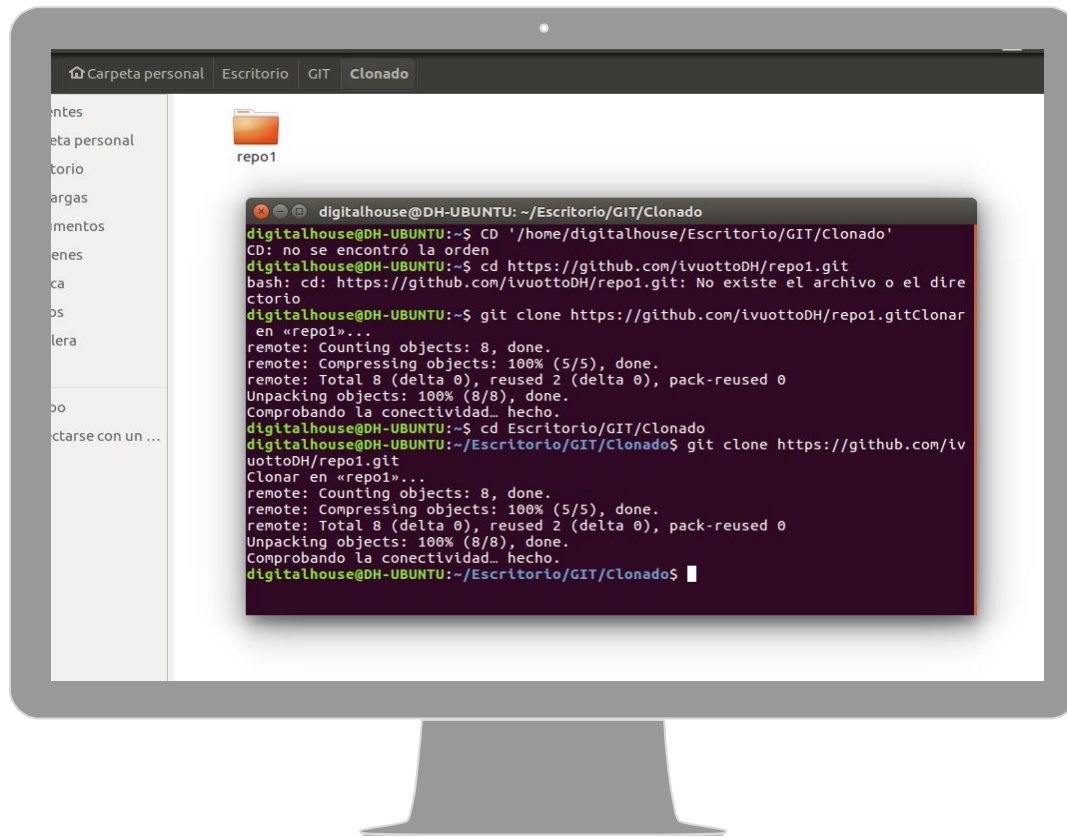
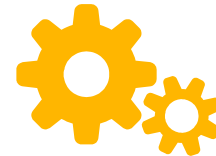


git add .

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



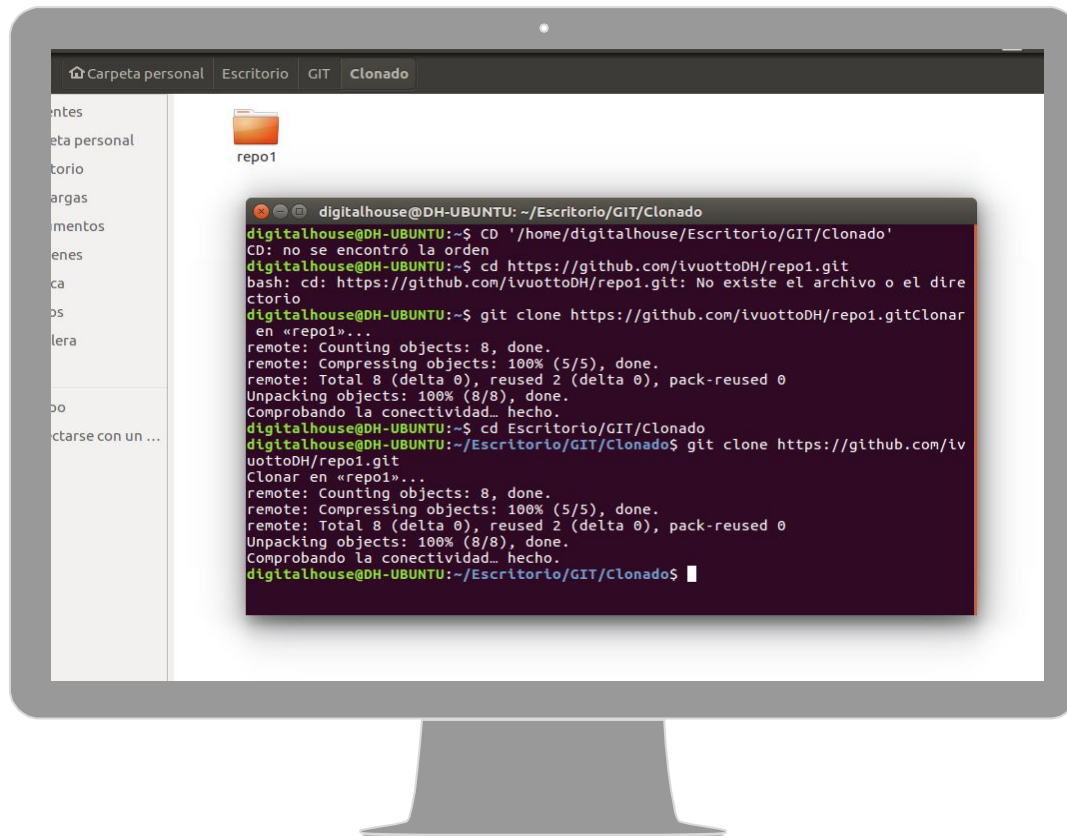
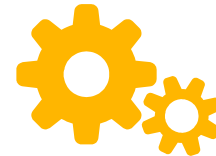
Cada vez que deseemos comprobar o verificar el estado de nuestro repositorio podremos escribir la siguiente línea de comando:

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:*

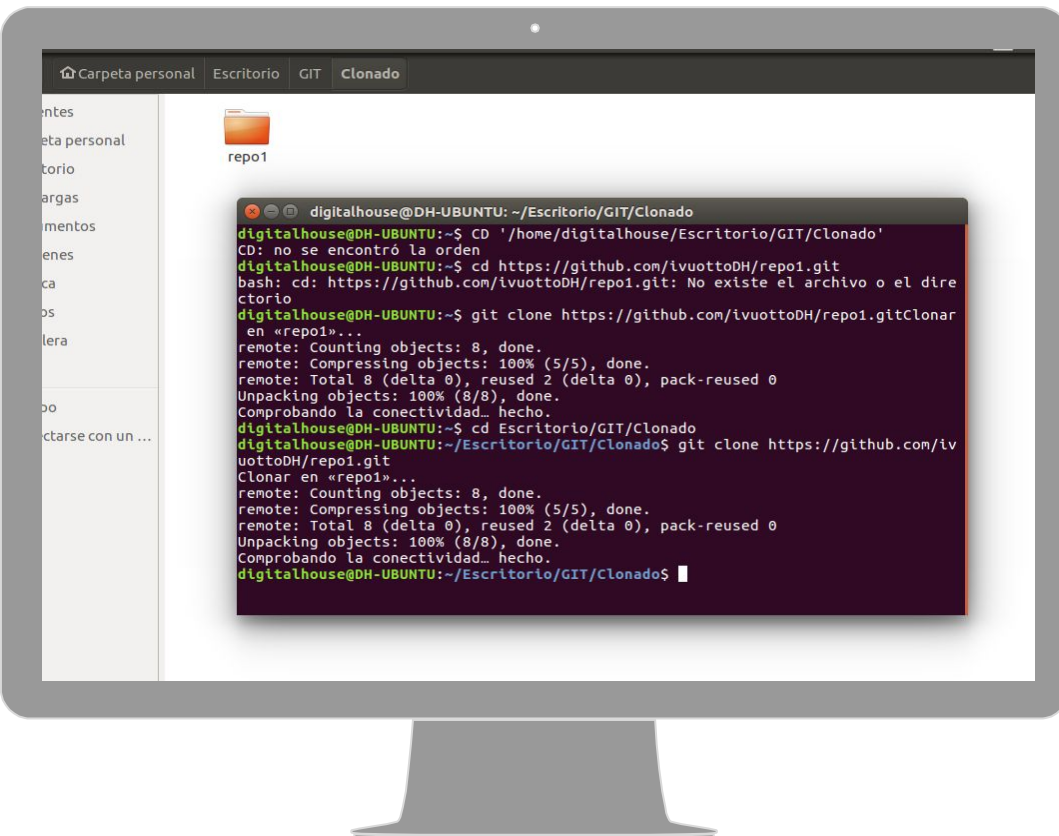
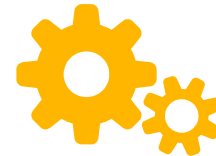
git commit -m "un mensaje cualquier"



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.

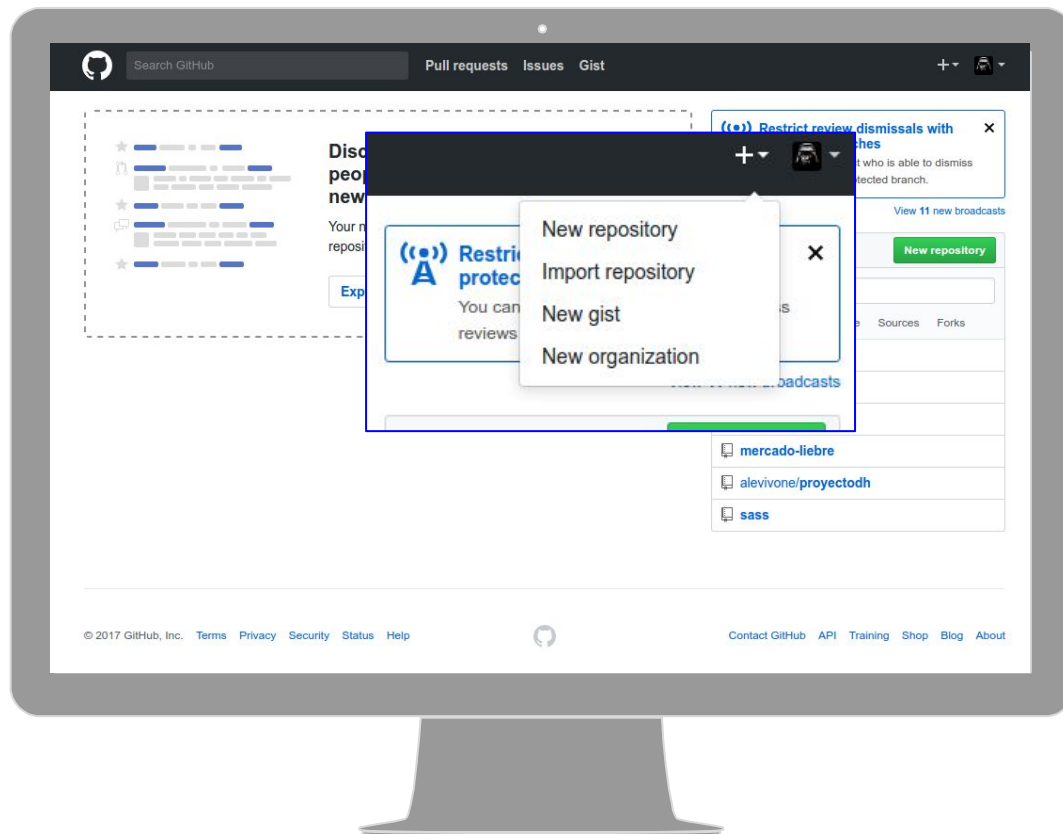
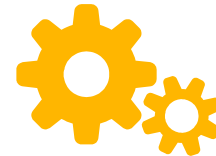
*Al especificar **master**, estamos diciendo a qué rama del repositorio queremos enviar nuestros archivos.*

Ahora

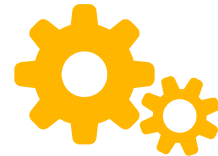
¿Cómo creamos el repositorio remoto?

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**.



The screenshot shows the GitHub 'Create a new repository' page. The form includes fields for 'Owner' (javi-dh), 'Repository name' (repo-de-prueba), 'Description' (optional), and visibility options (Public, Private). There are also checkboxes for 'Initialize this repository with a README' and dropdowns for 'Add .gitignore' and 'Add a license'. A green 'Create repository' button is at the bottom. Annotations include an orange arrow pointing to the repository name field with the text 'Darle un nombre', a red text warning '¡NO TOCAR NADA MÁS!' next to the visibility options, and another orange arrow pointing to the 'Create repository' button with the text 'CREAR REPOSITORIO'.

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner Repository name

javi-dh / repo-de-prueba ✓

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

Description (optional)

☐ Public
Anyone can see this repository. You choose who can commit.

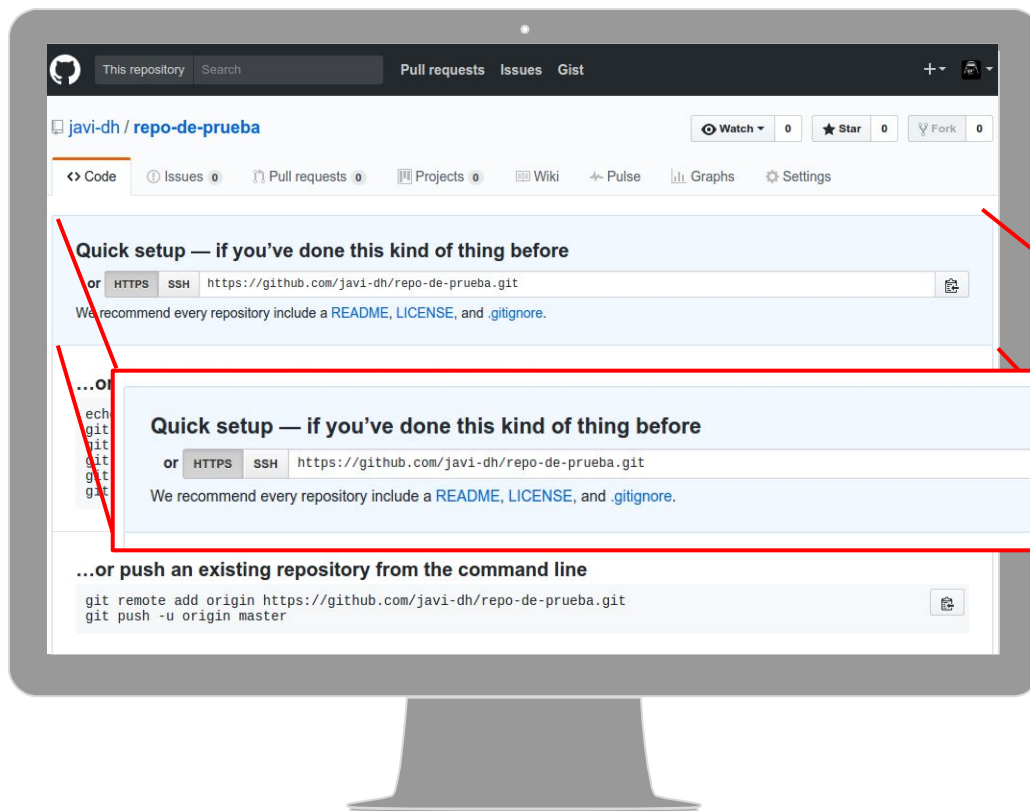
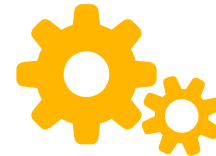
☐ Private
You choose who can see and commit to this repository.

☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None Add a license: None ⓘ

Create 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.**



Webs de consulta

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

