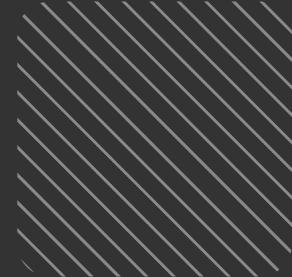


La Terminal, GIT & GitHub

IT BOARDING

BOOTCAMP





// Terminal - Consola - CMD



¿Qué rayos es eso?

IT BOARDING

BOOTCAMP



“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.”.

IT BOARDING

BOOTCAMP

// ¿Por qué usar la terminal?

- Para tener mayor control sobre el Sistema Operativo.
- 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.

IT BOARDING

BOOTCAMP



¿Dónde está la terminal?

Sea cual sea el Sistema Operativo que estemos usando, acceder a la misma es muy sencillo.

```
howto geek@ubuntu: ~/Downloads
howto geek@ubuntu:~/Downloads$ ls
file
howto geek@ubuntu:~/Downloads$ mv file newfile
howto geek@ubuntu:~/Downloads$ ls
newfile
howto geek@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.

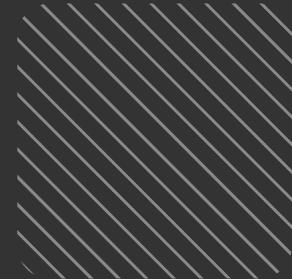
C:\Windows\system32>
```

```
Last login: Mon Apr 13 11:46:14 on ttys000
Kenny's-MacBook-Pro:~ Kenny$ mv ~/Documents/Test/TestFile-copy.rtf ~/Documents/Test2/TestFile-copy.rtf
```

Comandos **básicos** que **debemos** saber

IT BOARDING

BOOTCAMP





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

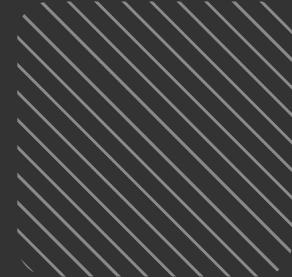
(limpia todo lo que hayamos escrito en la consola / Windows)



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

IT BOARDING

BOOTCAMP



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

IT BOARDING

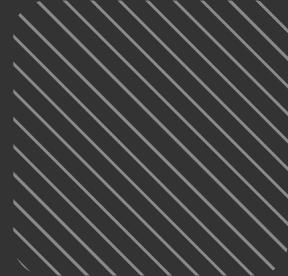
BOOTCAMP

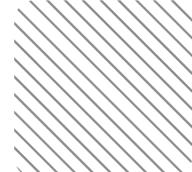


git

IT BOARDING

BOOTCAMP



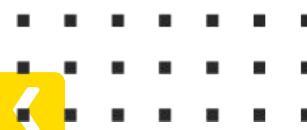


Cuando trabajamos con Git, hablamos de trabajar con un **REPOSITORIO**. El cual es un lugar en donde se almacenan nuestros archivos. Hay dos tipos de **REPOSITORIOS** el **local** y el **remoto**.

IT BOARDING

BOOTCAMP





Branch Principal



Harina

Harina
Huevos

Harina
Huevos
Aceite

Harina
Huevos
Manteca

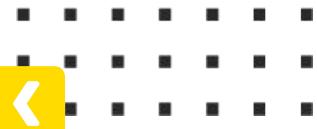
Revisión 1

Revisión 2

Revisión 3

Revisión 4





Branch Principal

Harina
Huevos
Aceite

Revisión 3

Pull

Harina
Huevos
Manteca

Revisión 4

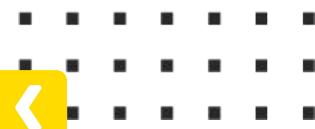
Rollback

Harina
Huevos
Manteca

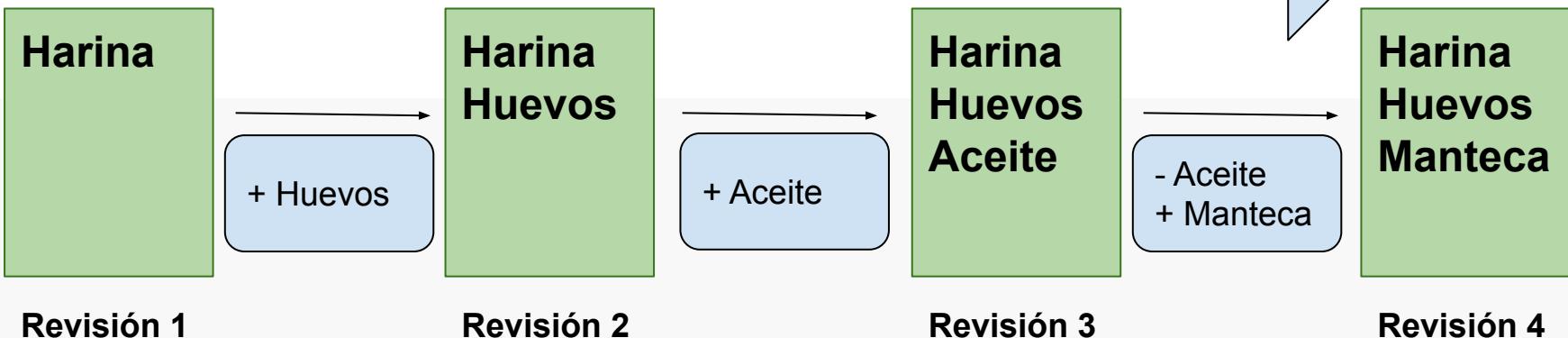
Copia Actual

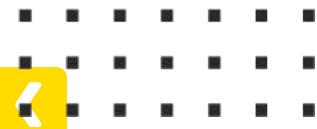
Push



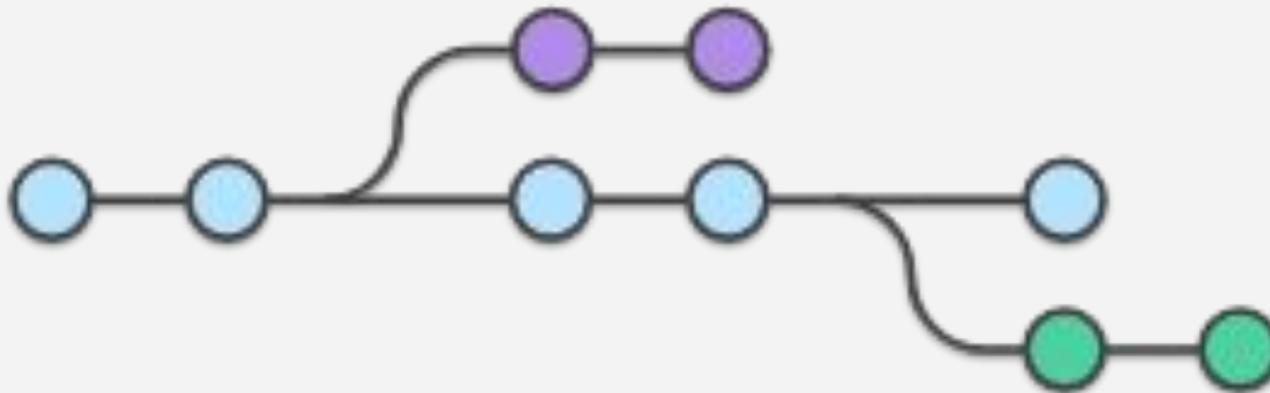


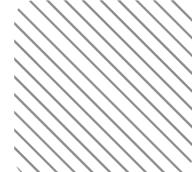
Branch Principal





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 **main.**

IT BOARDING

BOOTCAMP



Revisión 4

Harina
Huevos
Aceite

Revisión 6

Harina
Huevos

Nuevas Funcionalidades

Branch Principal

Harina
Huevos
Aceite

Revisión 3

Harina
Huevos
Manteca

Revisión 5

Revisión 4

Harina
Huevos
Aceite

Revisión 6

Harina
Huevos
Cafe

Nuevas Funcionalidades

Branch Principal

Harina
Huevos
Aceite

Revisión 3

Harina
Huevos
Manteca

Revisión 5

Revisión 4

Harina
Huevos
Aceite

Revisión 6

Harina
Huevos
Cafe

Nuevas Funcionalidades

?

Branch Principal

Harina
Huevos
Aceite

Revisión 3

...

Revisión xx

Harina
Huevos
Manteca

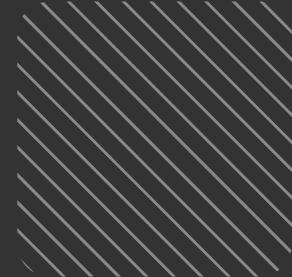
Revisión 5



CONFLICTOS

IT BOARDING

BOOTCAMP





Conflictos Comunes

Sea que usemos la consola de comandos o un GUI, el error más frecuente es cuando dos o más usuarios tocan el mismo archivo en la misma línea de código.

Para evitar esto, la recomendación más eficiente es dividir bien el laburo y no trabajar al mismo tiempo varias personas sobre el mismo archivo.



git

Repo Local



GitHub

Repo Remoto



IT BOARDING

BOOTCAMP

Creando el repositorio local



The screenshot shows a Linux desktop interface with a terminal window open. The terminal window has a dark background and displays the following command-line session:

```
digitalhouse@DH-UBUNTU:~$ cd '/home/digitalhouse/Escritorio/GIT/Clonado'
digitalhouse@DH-UBUNTU:~$ cd https://github.com/ivuottoDH/repo1.git
bash: cd: https://github.com/ivuottoDH/repo1.git: No existe el archivo o el directorio
digitalhouse@DH-UBUNTU:~$ git clone https://github.com/ivuottoDH/repo1.git Clonar en «repo1»...
remote: Counting objects: 8, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 2 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.
Comprobando la conectividad... hecho.
digitalhouse@DH-UBUNTU:~$ cd Escritorio/GIT/Clonado
digitalhouse@DH-UBUNTU:~/Escritorio/GIT/Clonado$ git clone https://github.com/ivuottoDH/repo1.git
Clonar en «repo1»...
remote: Counting objects: 8, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 2 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.
Comprobando la conectividad... hecho.
digitalhouse@DH-UBUNTU:~/Escritorio/GIT/Clonado$
```



“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



The image shows a laptop screen with a terminal window open. The terminal window has a dark background with white text. It displays two separate git clone commands being run. The first command is run from the directory '/home/digitalhouse/Escritorio/GIT/Clonado'. The second command is run from the directory '~/Escritorio/GIT/Clonado'. Both commands attempt to clone a repository from the URL 'https://github.com/ivuottoDH/repo1.git'. The output of the first command shows an error message indicating that the directory does not exist. The output of the second command shows the process of cloning the repository, including counting objects, compressing objects, and unpacking objects.

```
digitalhouse@DH-UBUNTU:~$ cd '/home/digitalhouse/Escritorio/GIT/Clonado'
digitalhouse@DH-UBUNTU:~$ cd https://github.com/ivuottoDH/repo1.git
bash: cd: https://github.com/ivuottoDH/repo1.git: No existe el archivo o el directorio
digitalhouse@DH-UBUNTU:~$ git clone https://github.com/ivuottoDH/repo1.git Clonar en «repo1»...
remote: Counting objects: 8, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 2 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.
Comprobando la conectividad... hecho.
digitalhouse@DH-UBUNTU:~$ cd Escritorio/GIT/Clonado
digitalhouse@DH-UBUNTU:~/Escritorio/GIT/Clonado$ git clone https://github.com/ivuottoDH/repo1.git
Clonar en «repo1»...
remote: Counting objects: 8, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 2 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.
Comprobando la conectividad... hecho.
digitalhouse@DH-UBUNTU:~/Escritorio/GIT/Clonado$
```



“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



The image shows a computer monitor with a terminal window open. The terminal window has a dark background and displays the following command and its execution:

```
digitalhouse@DH-UBUNTU:~$ cd '/home/digitalhouse/Escritorio/GIT/Clonado'
digitalhouse@DH-UBUNTU:~$ cd https://github.com/ivuottoDH/repo1.git
bash: cd: https://github.com/ivuottoDH/repo1.git: No existe el archivo o el directorio
digitalhouse@DH-UBUNTU:~$ git clone https://github.com/ivuottoDH/repo1.git Clonar en «repo1»...
remote: Counting objects: 8, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 2 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.
Comprobando la conectividad... hecho.
digitalhouse@DH-UBUNTU:~$ cd Escritorio/GIT/Clonado
digitalhouse@DH-UBUNTU:~/Escritorio/GIT/Clonado$ git clone https://github.com/ivuottoDH/repo1.git
Clonar en «repo1»...
remote: Counting objects: 8, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 2 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.
Comprobando la conectividad... hecho.
digitalhouse@DH-UBUNTU:~/Escritorio/GIT/Clonado$
```





Vamos a crear nuestro repo en Github



```
digitalhouse@DH-UBUNTU:~$ cd '/home/digitalhouse/Escritorio/GIT/Clonado'
digitalhouse@DH-UBUNTU:~$ cd https://github.com/ivuottoDH/repo1.git
bash: cd: https://github.com/ivuottoDH/repo1.git: No existe el archivo o el directorio
digitalhouse@DH-UBUNTU:~$ git clone https://github.com/ivuottoDH/repo1.git Clonar en «repo1»...
remote: Counting objects: 8, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 2 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.
Comprobando la conectividad... hecho.
digitalhouse@DH-UBUNTU:~$ cd Escritorio/GIT/Clonado
digitalhouse@DH-UBUNTU:~/Escritorio/GIT/Clonado$ git clone https://github.com/ivuottoDH/repo1.git
Clonar en «repo1»...
remote: Counting objects: 8, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 2 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.
Comprobando la conectividad... hecho.
digitalhouse@DH-UBUNTU:~/Escritorio/GIT/Clonado$
```



// ¿Cómo creamos el repositorio remoto?

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

IT BOARDING

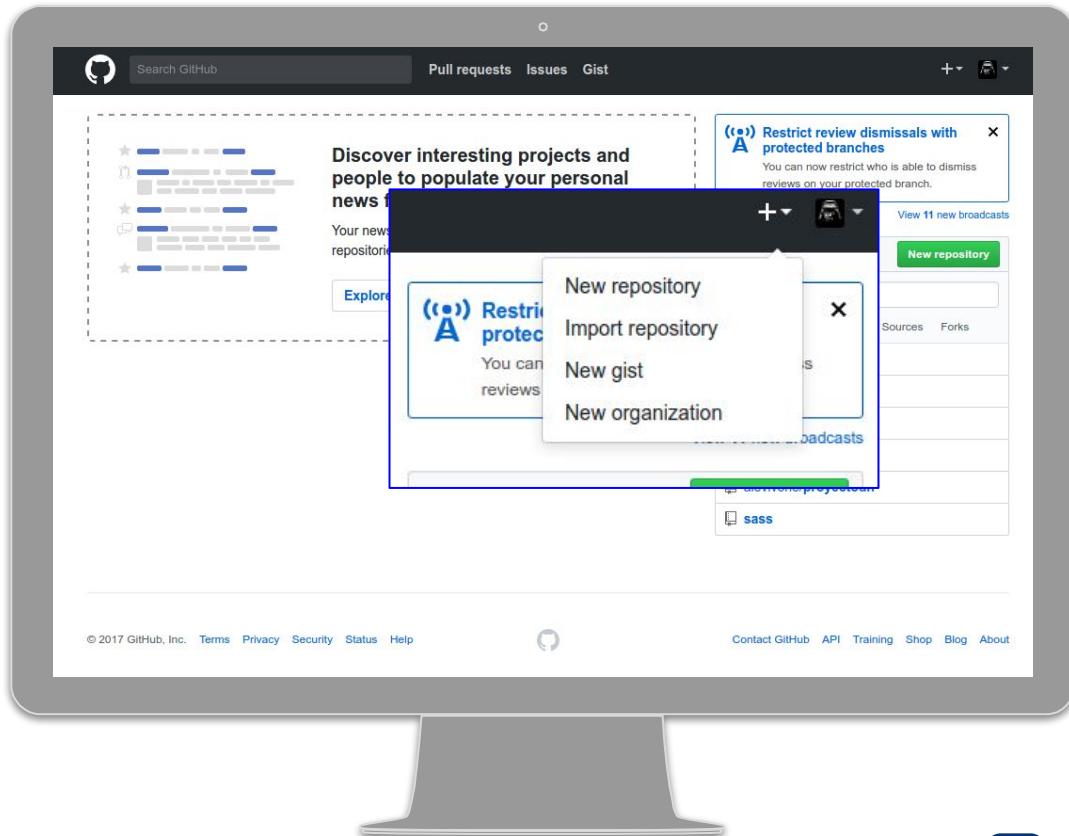
BOOTCAMP



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

IT BOARDING

BOOTCAMP





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

IT BOARDING

BOOTCAMP

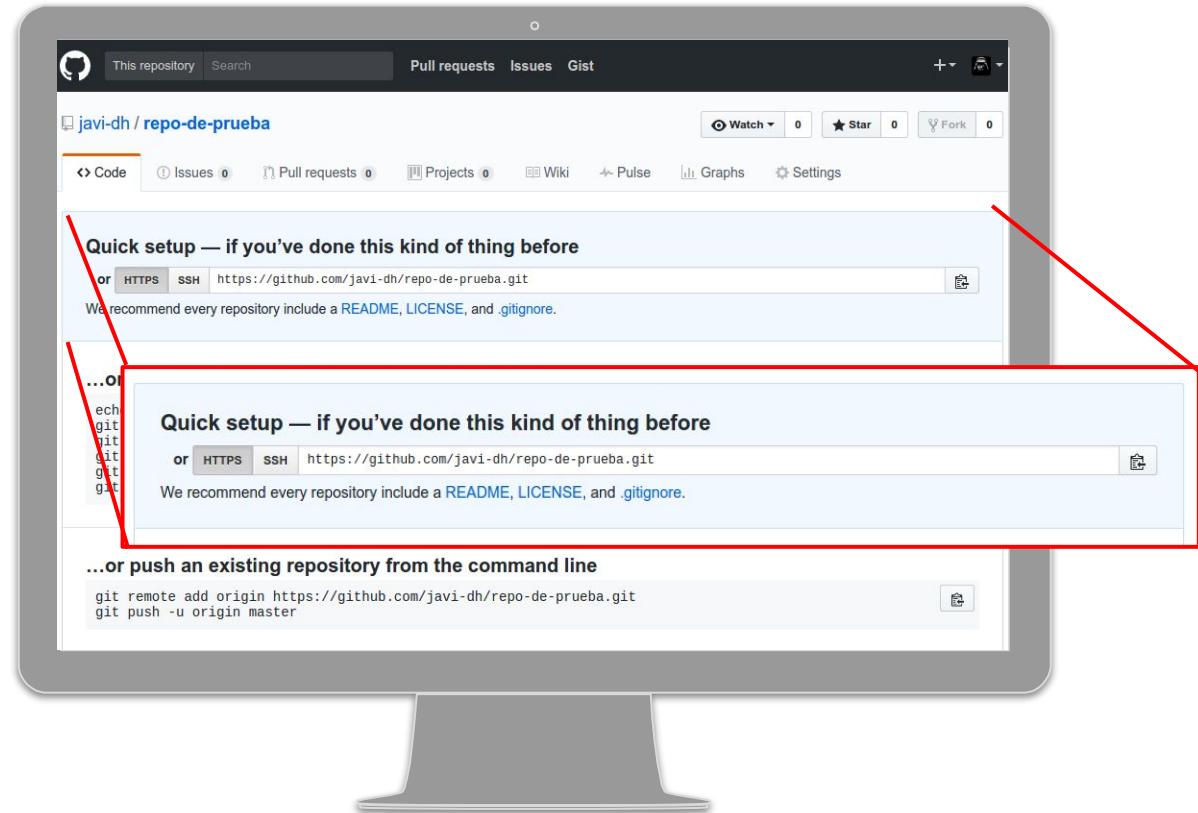




// Luego veremos esta pantalla y ésta URL es la que necesitamos tener a mano en el paso de:
Asignando nuestro repositorio remoto.

IT BOARDING

BOOTCAMP



// 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
```

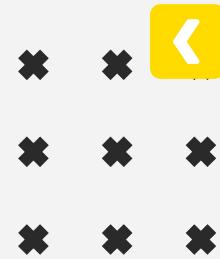
IT BOARDING

BOOTCAMP

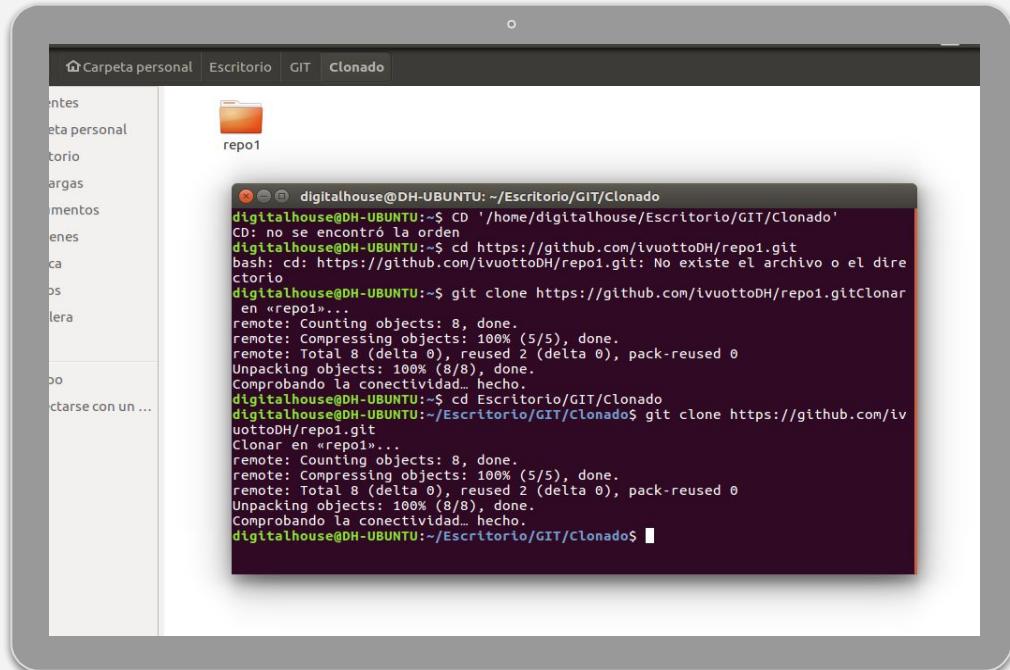
git remote add origin https://...

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**



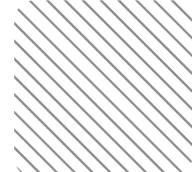
Adicionando nuestros archivos al repositorio local



The image shows a computer monitor with a terminal window open. The terminal window has a dark background and contains the following text:

```
digitalhouse@DH-UBUNTU:~$ cd '/home/digitalhouse/Escritorio/GIT/Clonado'
CD: no se encontró la orden
digitalhouse@DH-UBUNTU:~$ cd https://github.com/ivuottoDH/repo1.git
bash: cd: https://github.com/ivuottoDH/repo1.git: No existe el archivo o el directorio
digitalhouse@DH-UBUNTU:~$ git clone https://github.com/ivuottoDH/repo1.git Clonar en «repo1»...
remote: Counting objects: 8, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 2 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.
Comprobando la conectividad... hecho.
digitalhouse@DH-UBUNTU:~$ cd Escritorio/GIT/Clonado
digitalhouse@DH-UBUNTU:~/Escritorio/GIT/Clonado$ git clone https://github.com/ivuottoDH/repo1.git
Clonar en «repo1»...
remote: Counting objects: 8, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 2 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.
Comprobando la conectividad... hecho.
digitalhouse@DH-UBUNTU:~/Escritorio/GIT/Clonado$
```





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

IT BOARDING

BOOTCAMP



git add --all



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

git add archivo.txt

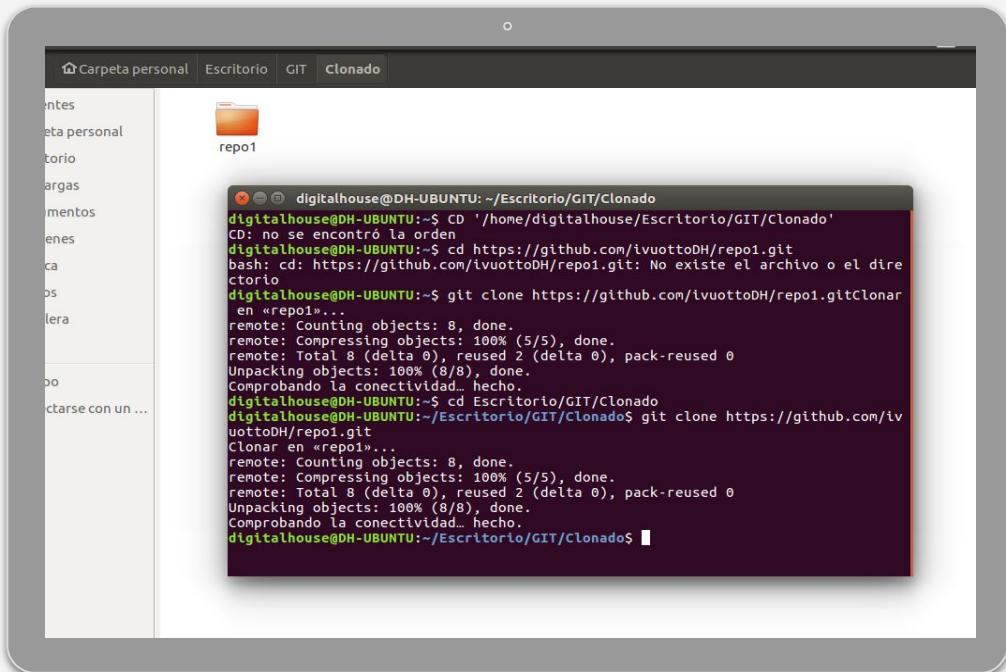
Agrega al stage (de manera temporal) solamente el archivo referenciado.

IT BOARDING

BOOTCAMP



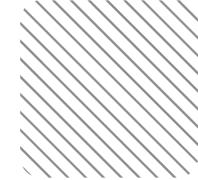
Testeando el status de nuestro repositorio



The image shows a computer monitor with a terminal window open. The terminal window has a dark background and displays the following command and its execution:

```
digitalhouse@DH-UBUNTU:~$ cd '/home/digitalhouse/Escritorio/GIT/Clonado'
digitalhouse@DH-UBUNTU:~$ cd https://github.com/ivuottoDH/repo1.git
bash: cd: https://github.com/ivuottoDH/repo1.git: No existe el archivo o el directorio
digitalhouse@DH-UBUNTU:~$ git clone https://github.com/ivuottoDH/repo1.git Clonar en «repo1»...
remote: Counting objects: 8, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 2 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.
Comprobando la conectividad... hecho.
digitalhouse@DH-UBUNTU:~$ cd Escritorio/GIT/Clonado
digitalhouse@DH-UBUNTU:~/Escritorio/GIT/Clonado$ git clone https://github.com/ivuottoDH/repo1.git
Clonar en «repo1»...
remote: Counting objects: 8, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 2 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.
Comprobando la conectividad... hecho.
digitalhouse@DH-UBUNTU:~/Escritorio/GIT/Clonado$
```





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

git status

IT BOARDING

BOOTCAMP





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**).



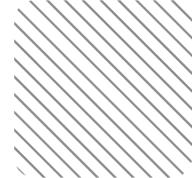
* * *

* * *

Agregando oficialmente los archivos al stage

```
digitalhouse@DH-UBUNTU:~$ cd '/home/digitalhouse/Escritorio/GIT/Clonado'
CD: no se encontró la orden
digitalhouse@DH-UBUNTU:~$ cd https://github.com/ivuottoDH/repo1.git
bash: cd: https://github.com/ivuottoDH/repo1.git: No existe el archivo o el directorio
digitalhouse@DH-UBUNTU:~$ git clone https://github.com/ivuottoDH/repo1.git Clonar en «repo1»...
remote: Counting objects: 8, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 2 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.
Comprobando la conectividad... hecho.
digitalhouse@DH-UBUNTU:~$ cd Escritorio/GIT/Clonado
digitalhouse@DH-UBUNTU:~/Escritorio/GIT/Clonado$ git clone https://github.com/ivuottoDH/repo1.git
Clonar en «repo1»...
remote: Counting objects: 8, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 2 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.
Comprobando la conectividad... hecho.
digitalhouse@DH-UBUNTU:~/Escritorio/GIT/Clonado$
```





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

git commit -m "un mensaje cualquier"

IT BOARDING

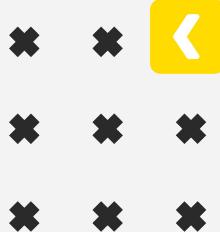
BOOTCAMP



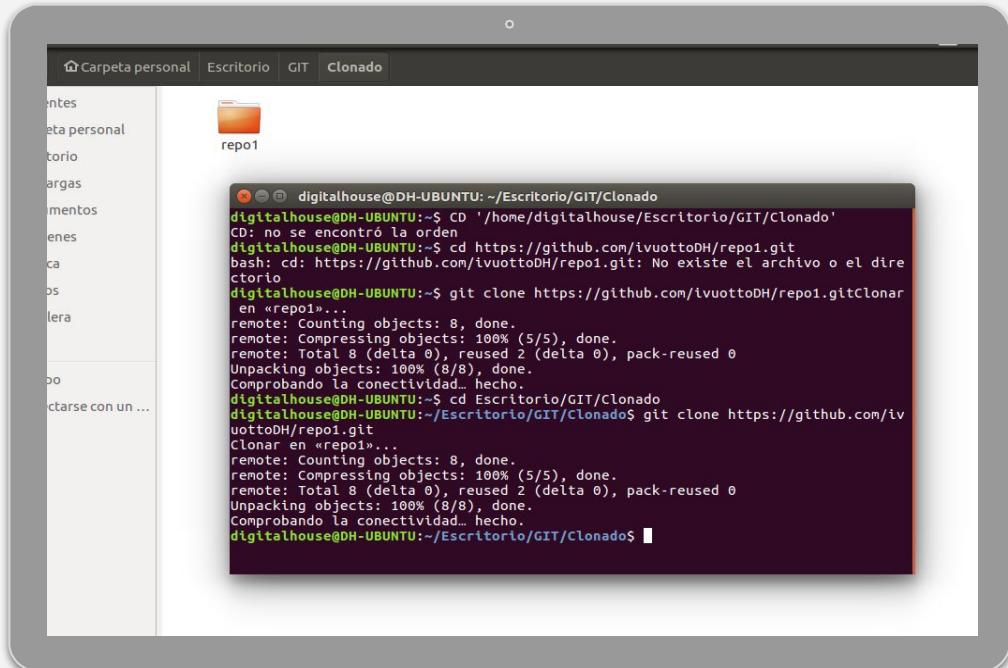
git commit -m "un mensaje cualquier"

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.



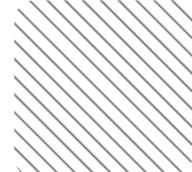
Enviando nuestros archivos del repositorio local al repositorio remoto



The image shows a computer monitor with a terminal window open. The terminal window has a dark background and contains the following text:

```
digitalhouse@DH-UBUNTU:~/Escritorio/GIT/Clonado
digitalhouse@DH-UBUNTU:~$ cd '/home/digitalhouse/Escritorio/GIT/Clonado'
CD: no se encontró la orden
digitalhouse@DH-UBUNTU:~$ cd https://github.com/ivuottoDH/repo1.git
bash: cd: https://github.com/ivuottoDH/repo1.git: No existe el archivo o el directorio
digitalhouse@DH-UBUNTU:~$ git clone https://github.com/ivuottoDH/repo1.git Clonar en «repo1»...
remote: Counting objects: 8, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 2 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.
Comprobando la conectividad... hecho.
digitalhouse@DH-UBUNTU:~$ cd Escritorio/GIT/Clonado
digitalhouse@DH-UBUNTU:~/Escritorio/GIT/Clonado$ git clone https://github.com/ivuottoDH/repo1.git
Clonar en «repo1»...
remote: Counting objects: 8, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 2 (delta 0), pack-reused 0
Unpacking objects: 100% (8/8), done.
Comprobando la conectividad... hecho.
digitalhouse@DH-UBUNTU:~/Escritorio/GIT/Clonado$
```





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

git push origin main

IT BOARDING

BOOTCAMP





git push origin main

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

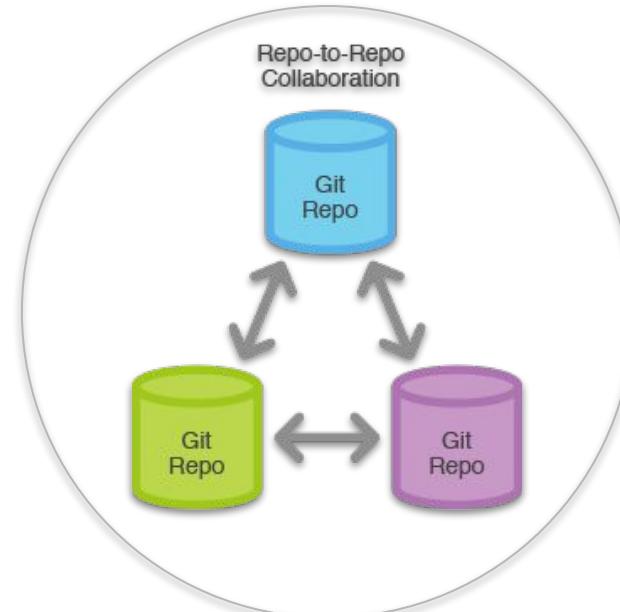
Al especificar main, estamos diciendo a qué rama del repositorio queremos enviar nuestros archivos.





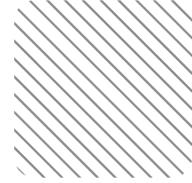
// Bajando los archivos de repo remoto a nuestro repo local

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



IT BOARDING

BOOTCAMP



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`

IT BOARDING

BOOTCAMP



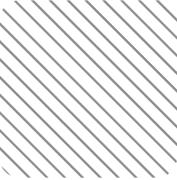


git clone https://....

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 main

IT BOARDING

BOOTCAMP





git pull origin main

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.



IT BOARDING

BOOTCAMP



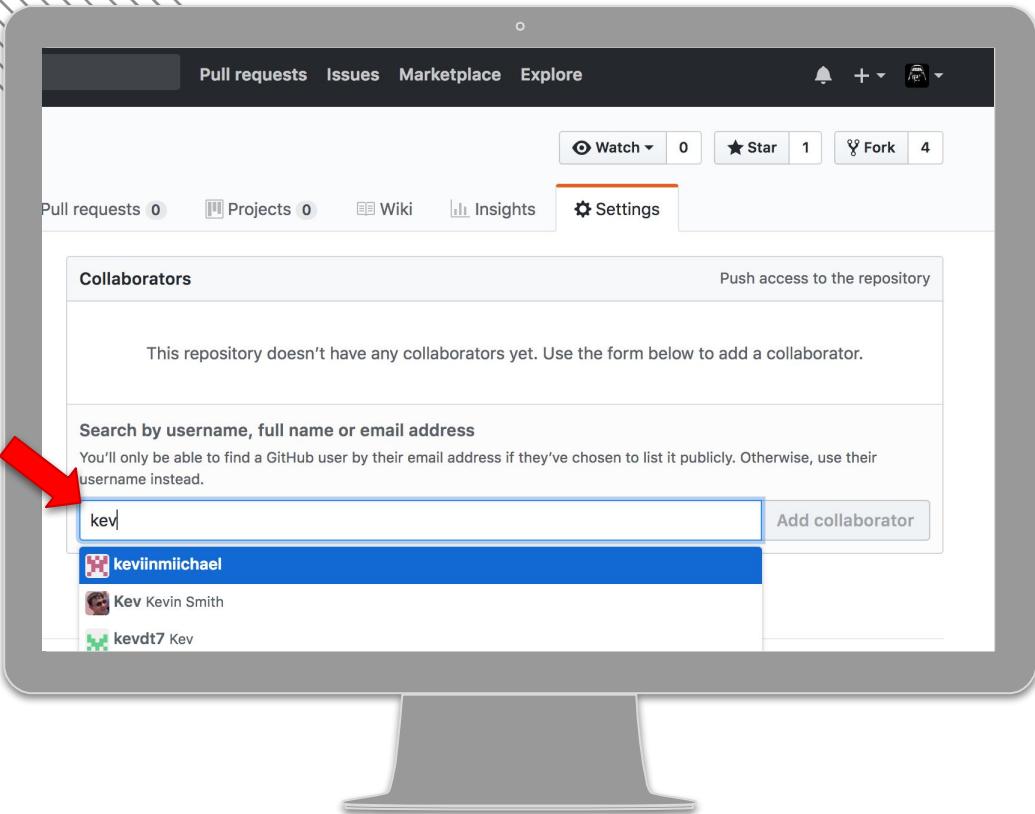
// Una vez aquí iremos a la opción:
Collaborators

IT BOARDING

BOOTCAMP

The screenshot shows a GitHub repository page for 'javi-dh / phpTN2018_01'. The top navigation bar includes links for This repository, Search, Pull requests, Issues, Marketplace, Explore, Watch (0), Star (1), Fork (4), and Settings. The Settings tab is active. On the left, there's a sidebar with Options, Collaborators, Branches, Webhooks, Integrations & services, and Deploy keys. A red box highlights the 'Collaborators' section, which contains sub-links for Options, Collaborators, Branches, Webhooks, Integrations & services, and Deploy keys.

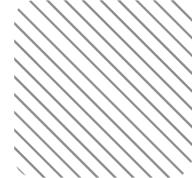




/// Aquí escribiremos el nombre de usuario de nuestro colega y después pulsaremos el botón:
Add collaborator

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 main

IT BOARDING

BOOTCAMP





Receta paso a paso

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



01 `git init` //crea el repositorio



02 `git config user.name "hanSolo"`
//agrega nuestra identidad - username



03 `git config user.email "hansolo@starwars.com"`
//agrega nuestra identidad - email



04 `git remote add origin https://github.com/....`
//apunta al repositorio remoto

IT BOARDING

BOOTCAMP



05

git add .

//agrega todos los cambios al repo local



06

git commit -m 'mensaje del commit'

//hito histórico - comitea los cambio hechos



07

git push origin main

//manda los cambios al repositorio remoto



...

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

Webs de consulta

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

<http://dev.to/git>

<http://ohshitgit.com>

<http://git-scm.com>



Gracias.

IT BOARDING

BOOTCAMP

