

IT Bootcamp

La Terminal, GIT & GitHub

DigitalHouse>



SCHOOL OF
INNOVATION®



TECH
ACADEMY

Índice

1. [Terminal - Consola - CMD](#)
2. [Git](#)
3. [Repositorio Local](#)
4. [Repositorio Remoto](#)
5. [Resumen](#)

1 | Terminal - Consola - CMD

“

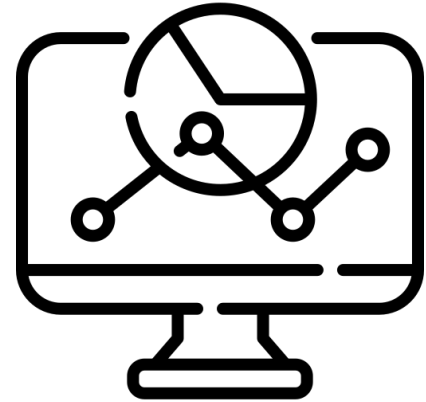
La terminal es un programa que está
presente en todos los sistemas
operativos y por medio del cual se
pueden dar órdenes al sistema **a través**
de líneas de comando

”



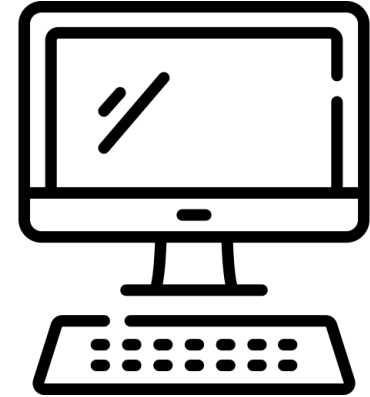
¿Por qué usar la terminal?

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



La terminal

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



¿Dónde está la terminal?

Sea cual sea el Sistema Operativo 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
Kenny-MacBook-Pro:~ Kenny$ mv ~/Documents/Test/TestFile-copy.rtf ~/Documents/
Test2/TestFile-copy.rtf
```

¿Dónde está la terminal?

En **Linux**: la forma más fácil de abrir una Terminal es usando Ctrl + Alt + T.

En **Windows**: Presionando WIN + R y aparecerá una ventana que pone ejecutar. En ella escribimos cmd.

En el **Mac**, podemos hacerlo desde:

- Hacer clic en el icono de Launchpad, en el Dock, escribir Terminal en el campo de búsqueda y, a continuación, hacer clic en Terminal.
- En el Finder, abrir la carpeta /Aplicaciones/Utilidades y, a continuación, hacer doble clic en Terminal.

Comandos básicos I



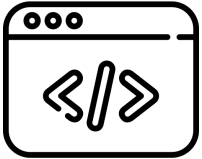
ls

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



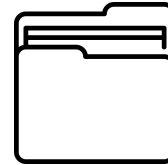
cd ..

Nos permite retroceder a una carpeta previa



dir

En Windows muestra los archivos de la carpeta en la que estamos ubicados



cd nombre_carpeta

Nos permite acceder a la carpeta descrita

Comandos básicos II (en Windows)



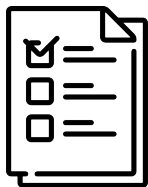
mkdir algo

crea una carpeta con el nombre "algo."



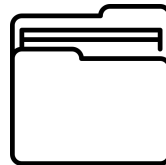
delete archivo.txt

Elimina un archivo con el nombre "archivo.txt"



type nul> archivo.txt

Crea un archivo de texto "archivo.txt"



rename archivo.txt otro.txt

Cambia el nombre "archivo.txt" a "otro.txt"

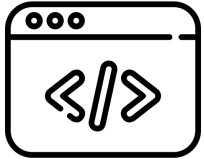
• **Tip:** Para las palabras delete y rename se pueden usar sus comandos abreviados "del" y "ren" respectivamente

Comandos básicos III



clear

Limpiamos todo lo que hayamos escrito en la consola / Mac y Linux. En Windows en el PowerShell



cls

Limpiamos todo lo que hayamos escrito en la consola / Windows

2 | Git

“

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**

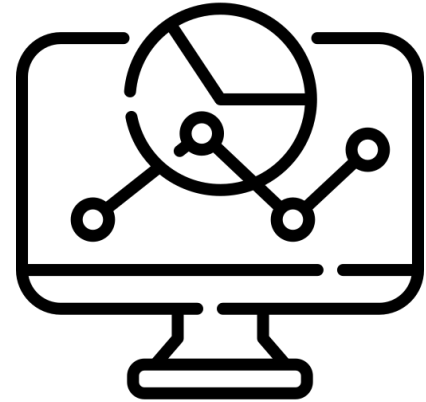


”

¿Qué son los repositorios?

Un repositorio es un lugar donde podemos guardar y administrar diferentes archivos. Nos sirven como “contenedores” de información.

Un repositorio puede ser local o remoto:



Repositorios



Repo Local

Se ejecutan de forma
"local" en cada
computadora

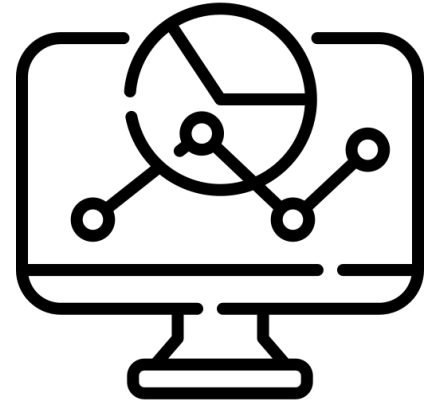


Repo Remoto

Están alojados en algún
servidor externo.
Existen servicios de
repositorios remotos,
por ejemplo: GitHub,
GitLab, entre otros.

¿Cómo compartimos archivos?

Utilizando un software que nos permita hacer un correcto seguimiento y control de las diferentes versiones.





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.



3 | Repositorio Local



El **repositorio local** es una carpeta que tendrá los archivos de nuestro proyecto en nuestra computadora, a diferencia del **repositorio remoto** que puede estar, por ejemplo en **GitHub**.



Creando el repositorio local

Lo primero será ubicarnos en la carpeta 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 quienes somos, para esto debemos tener un **usuario** y un **token de acceso personal**, que será nuestro password, para autenticar a nuestro usuario de git en el repositorio local.

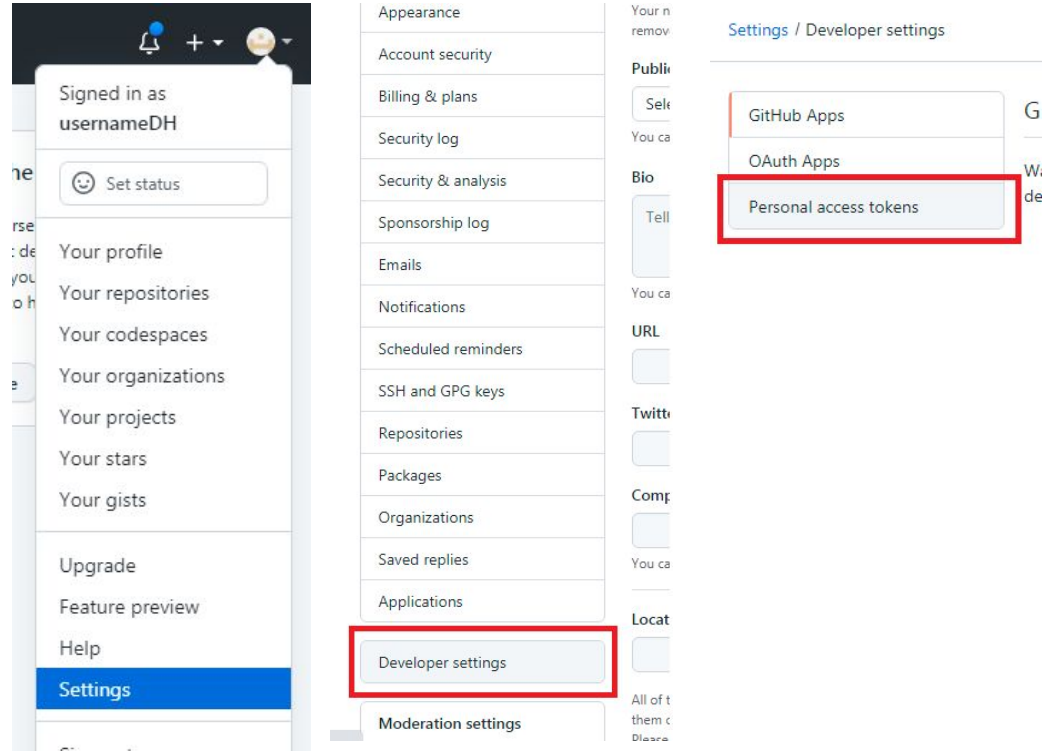
Token de acceso personal (pat)

Se utilizan como complemento o en lugar de una contraseña para la autenticación en **GitHub** y poder sincronizar el repositorio local con el repositorio remoto.

Como precaución de seguridad, **GitHub** elimina automáticamente los tokens de acceso personales que no se han usado durante un año.

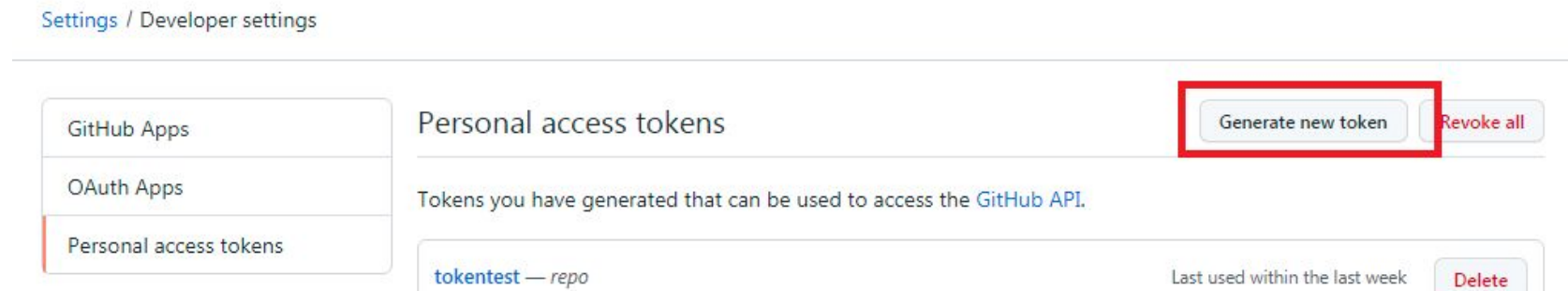
Generando el PAT

Logueados en nuestra cuenta de **GitHub** hacemos clic en el avatar y ahí elegimos la opción **Settings**, luego seleccionamos **Developer settings** y por último **Personal access tokens**.



Generando el PAT

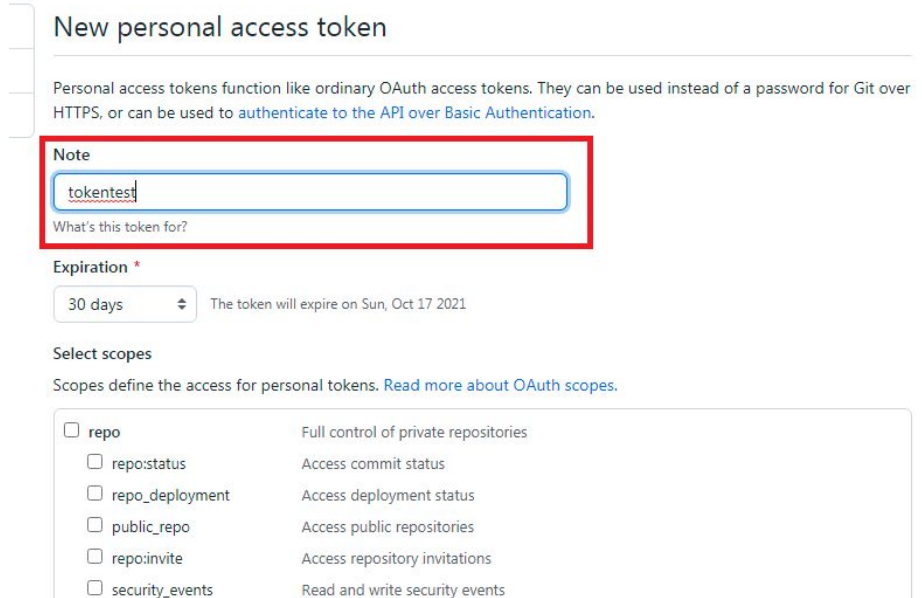
Hacemos clic en Generar un nuevo token, luego debemos ingresar un nombre descriptivo para el mismo.



Generando el PAT

Por último, seleccionamos los alcances o permisos que vamos a otorgarle al token.

Para usar el token para acceder a repositorios desde la línea de comando, seleccionamos **repo**.



New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

tokentest

What's this token for?

Expiration *

30 days The token will expire on Sun, Oct 17 2021

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo_deployment	Access deployment status
<input type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events

“

Debemos **copiar** y **resguardar** el **token**, ya que por razones de seguridad, una vez que salgamos de la página de github no podremos volver a verlo, solo nos mostrará el nombre que ingresamos al principio.



”

Agregando nuestra identidad

Una vez generado nuestro PAT, ya podemos indicarle al repositorio local nuestra configuración, esto es para que después podamos conectarnos al repositorio remoto, en este caso github, así:

```
>_ git config user.name "Jhon_Doe"  
    git config user.email "jhon@email.com"  
    git config user.password "jhon-PAT"
```

• **Tip:** puede usarse la flag `--global` seguido al "git config", para configurar por única vez, ya que Git usará esta información para todo lo que hagas en ese sistema.

Autenticación

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

git config user.password " "

Dentro de las comillas
pondremos nuestro
personal access token

4 | Repositorio Remoto

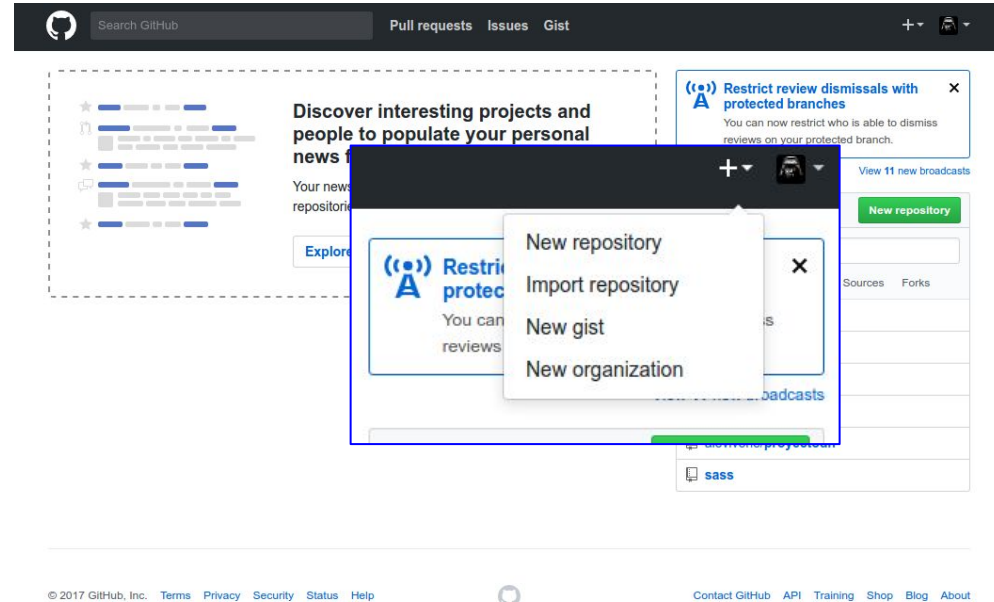
¿Qué es GitHub?

- Es uno de los proveedores de repositorios remotos más utilizado en el mundo y en el ambiente del desarrollo.
- GitHub se comunica con nuestro repositorio local de Git, permitiéndonos crear diferentes versiones y agregar colaboradores para compartir o colaborar con el trabajo que estemos haciendo.
- Para crear una cuenta en GitHub podemos acceder a su página web oficial: <https://github.com/>

¿Cómo creamos el repositorio remoto?

Primero vamos a crear nuestro repositorio en Github.

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



Creando el repositorio remoto

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**.

Create a new repository

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

Owner

javi-dh

Repository name

repo-de-prueba

Darle un nombre

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.

¡NO TOCAR NADA MÁS!

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

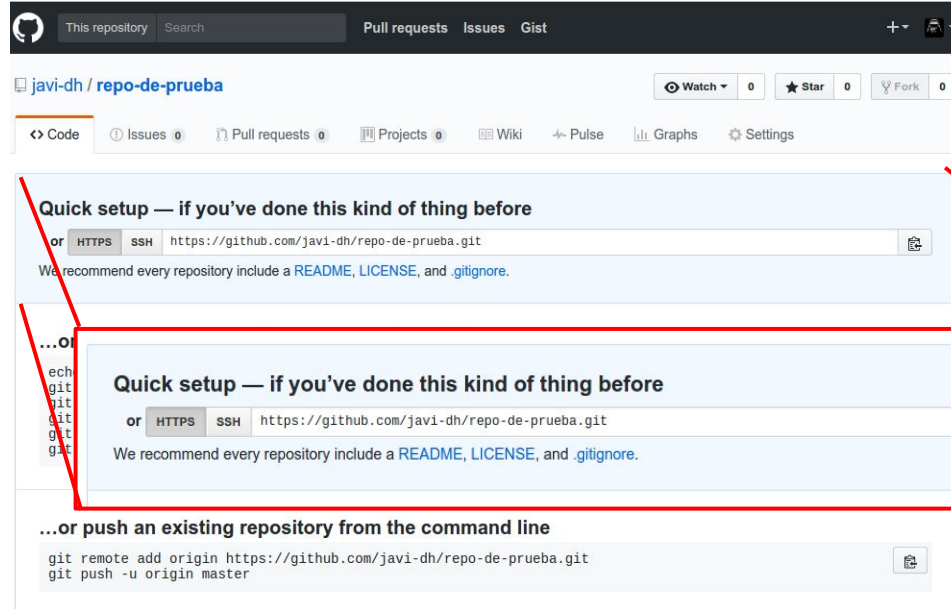


Create repository

CREAR REPOSITORIO

Creando el repositorio remoto

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



Asociando los repos

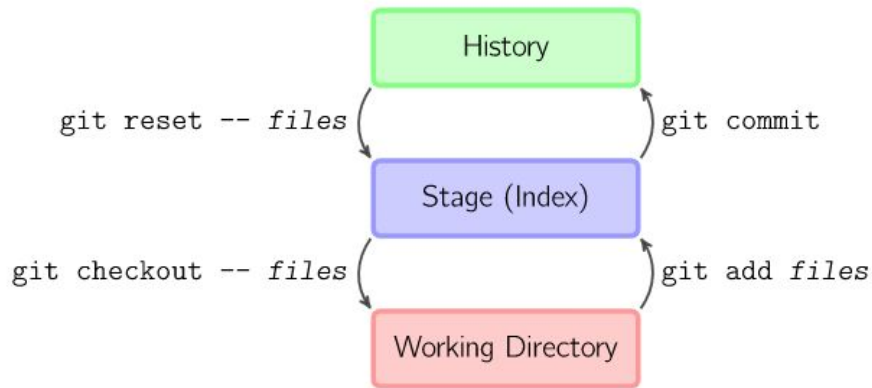
Habiendo creado el **Repositorio Remoto** y para que nuestro **Repositorio Local** sepa a donde queremos subir nuestros archivos tenemos que especificarlo. Con `git remote add`, le estamos indicando a nuestro repositorio local, a donde queremos llevar (repositorio remoto) nuestros archivos. La **URL** la obtendremos al crear un repositorio remoto en **Github.com**

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

¿Qué es el Stage?

Stage es un **estado intermedio** entre la versión local de un archivo con el que estemos trabajando y la versión definitiva que enviaremos a un repositorio remoto.

Es un punto como de “Stand By” donde los archivos se encuentran a la espera de ser seleccionados (o no) para finalmente ser versionados.



Subiendo al repositorio remoto

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

```
>_ git add .  
git add archivo.txt
```

Agregando al stage

`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.

Confirmando el stage

Para confirmar que los archivos agregados al stage los queremos de manera definitiva usaremos el comando `commit`, indicando 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.

```
>_ git commit -m "un mensaje cualquier"
```

Subiendo al repositorio remoto

Para enviar los archivos que tenemos en nuestro repositorio local al repositorio remoto, usamos **push**. 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.

```
>_ git push origin main
```

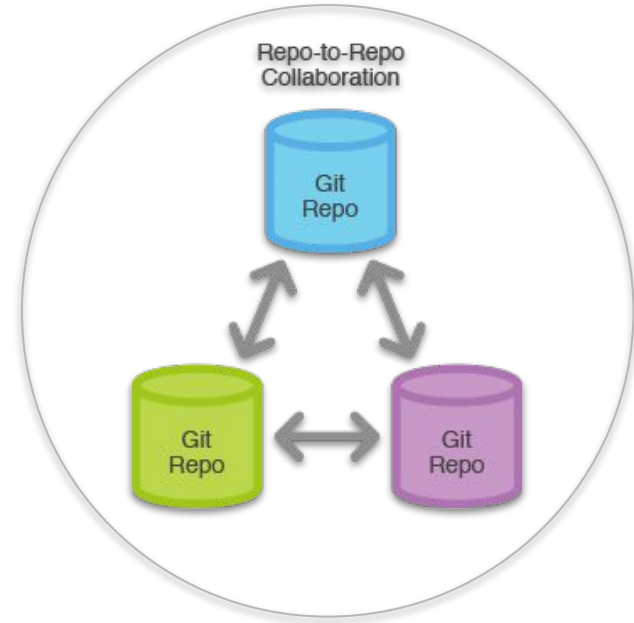

Estado del repositorio local

Podemos comprobar o verificar el **estado de nuestro repositorio**, analizando si hay archivos que no se han agregado temporalmente al **stage** así como también si hay archivos agregados al stage pero no de forma definitiva (**commit**).

```
>_ git status
```

Bajando al repositorio local

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



Clonando el repositorio remoto

Para **descargar por 1era vez** un repositorio remoto a nuestra máquina. Tendremos que **clonar** el mismo en la carpeta que deseemos. Con **git clone**, se crea 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**, subirlos al repo remoto.

```
>_ git clone https://github.com/user/repoName
```

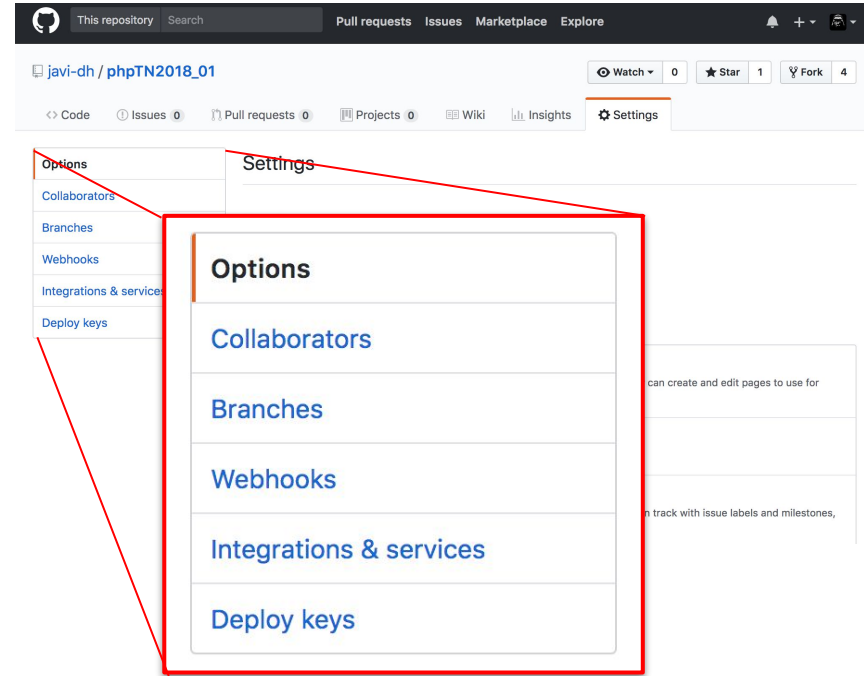
Actualizando el repositorio local

En caso que necesitemos actualizar el repositorio local con los cambios del repo remoto debemos usar **pull**, con este comando obtendremos, 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.

```
>_ git pull origin main
```

Agregando colaboradores

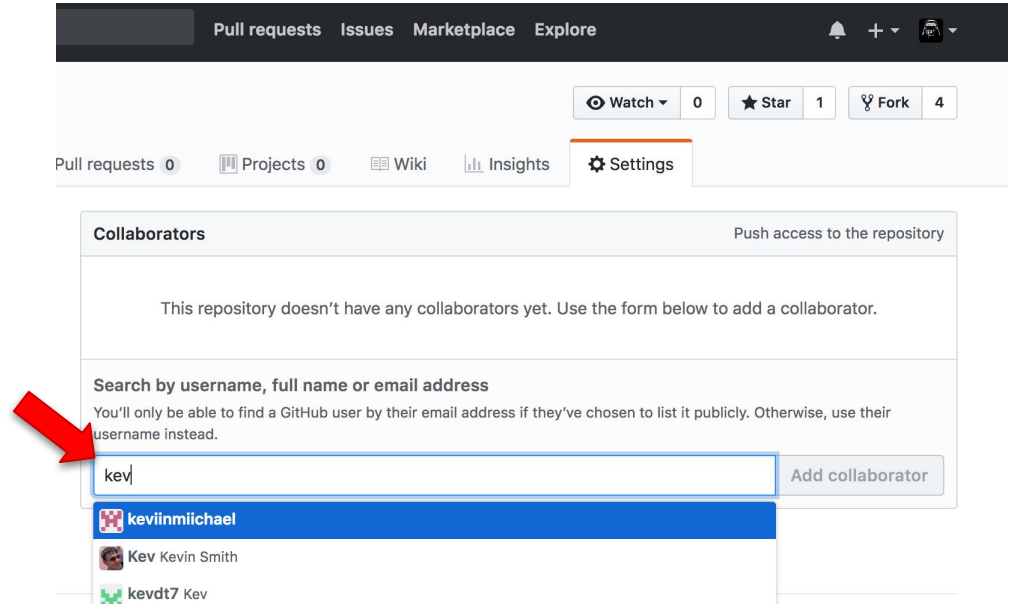
Es muy común que trabajemos en equipo, y que queramos agregar a nuestro repositorio a nuevos miembros para que participen del mismo. Esto lo lograremos desde **Collaborators**.






Nuevos colaboradores

Luego ingresamos el nombre de usuario de nuestro colega y pulsaremos el botón: **Add collaborator.**

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



The screenshot shows the GitHub 'Add collaborator' page. At the top, there's a navigation bar with links for Pull requests, Issues, Marketplace, and Explore. Below this, there are buttons for Watch (0), Star (1), and Fork (4). The 'Settings' tab is selected. The main section is titled 'Collaborators' and contains a message: 'This repository doesn't have any collaborators yet. Use the form below to add a collaborator.' Below this is a search bar with the placeholder text 'Search by username, full name or email address'. A red arrow points to the search input field, which contains the text 'kev'. Below the input field, there is a list of search results. The first result, 'kevinmichael', is highlighted in blue. The other two results are 'Kev Kevin Smith' and 'kevdt7 Kev'. An 'Add collaborator' button is located to the right of the search results.

Collaborators	Push access to the repository
This repository doesn't have any collaborators yet. Use the form below to add a collaborator.	
Search by username, full name or email address	
You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.	
kev	Add collaborator
 kevinmichael	
 Kev Kevin Smith	
 kevdt7 Kev	



Los colaboradores también tienen el poder de **push**ear su trabajo a nuestro repositorio. Por ello es importante, al momento de sentarnos antes de arrancar a trabajar hacer un `git pull origin main`



6 | Resumen

Paso a paso

Veamos el paso a paso que necesitamos para trabajar con git

1 | git init

Crea el repositorio

2 | PAT

En Github generar el PAT que usaremos como contraseña

3 | Configurar el nombre de usuario

`git config user.name "hanSolo"` agregamos nuestra identidad -
username = hansolo
por ejemplo

Paso a paso

4

Asociamos el email del usuario

git config user.email "hansolo@starwars.com", agrega a nuestra identidad en el repositorio local el email.

5

Agregamos el PAT

git config user.password hansolotk agregamos el password que generamos anteriormente en el paso 2

6

Sincronizamos los repositorios

git remote add origin <https://github.com/...>

Sincronizamos nuestro repo local para que se asocie al repo remoto.

Paso a paso

7

`git add .`

agrega todos los
cambios al repo local

8

`git commit -m
'mensaje'`

hito histórico -
omitea los cambio
hechos

9

`git push origin
main`

manda los cambios al
repositorio remoto

Material extra



Material extra

[PAT](#)

<http://dev.to/git>

<http://ohshitgit.com>

<http://git-scm.com>

<https://docs.github.com/>

DigitalHouse>