

# INTRODUCCIÓN A GITHUB

## ¿Qué es GitHub? (<https://kinsta.com/es/base-de-conocimiento/que-es-github/>)

GitHub es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador, y que fue comprada por Microsoft en junio del 2018. La plataforma está creada para que los desarrolladores suban el código de sus aplicaciones y herramientas, y que como usuario no sólo puedas descargarte la aplicación, sino también entrar a su perfil para leer sobre ella o colaborar con su desarrollo.

## NO Confundir con Git

Git es un sistema de control específico de versión de fuente abierta creada por Linus Torvalds en el 2005.

Específicamente, Git es un sistema de control de versión distribuida, lo que quiere decir que la base del código entero y su historial se encuentran disponibles en la computadora de todo desarrollador, lo cual permite un fácil acceso a las bifurcaciones y fusiones.

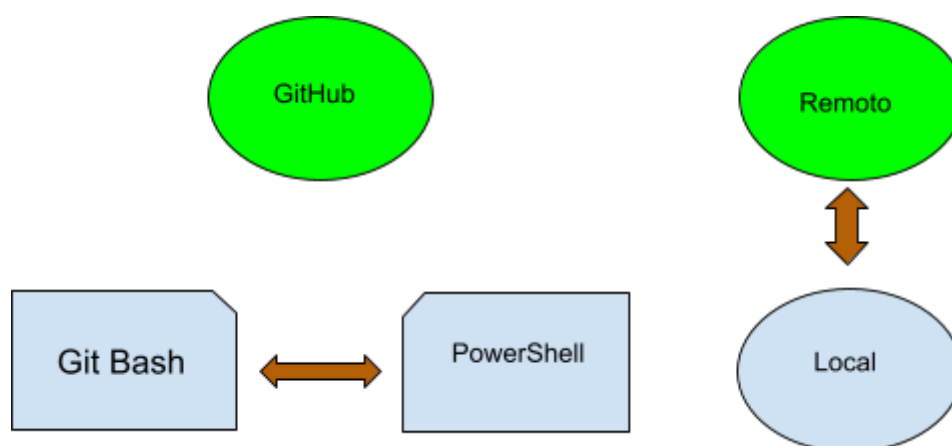
GitHub utiliza el sistema de control de versiones Git. Además de permitirte mirar el código y descargarte las diferentes versiones de una aplicación, la plataforma también hace las veces de red social conectando desarrolladores con usuarios para que estos puedan colaborar mejorando la aplicación.

# Cómo se trabaja con GitHub

GitHub se maneja desde tres posibles ámbitos:

La plataforma de GitHub directamente:

- Aquí encontrarás no sólo los repositorios residentes, sino también diversas herramientas para su control
- Git Bash, la terminal específica de Git:
- Desde tu propio terminal, a través de de la powershell de VSC. En estos dos casos, los comandos vienen a ser los mismos y en función de lo que fueras a realizar te viene mejor una terminal u otra.



## Como crear un repositorio en GitHub

(<https://keepcoding.io/blog/como-funciona-github-guia-para-novatos/>)

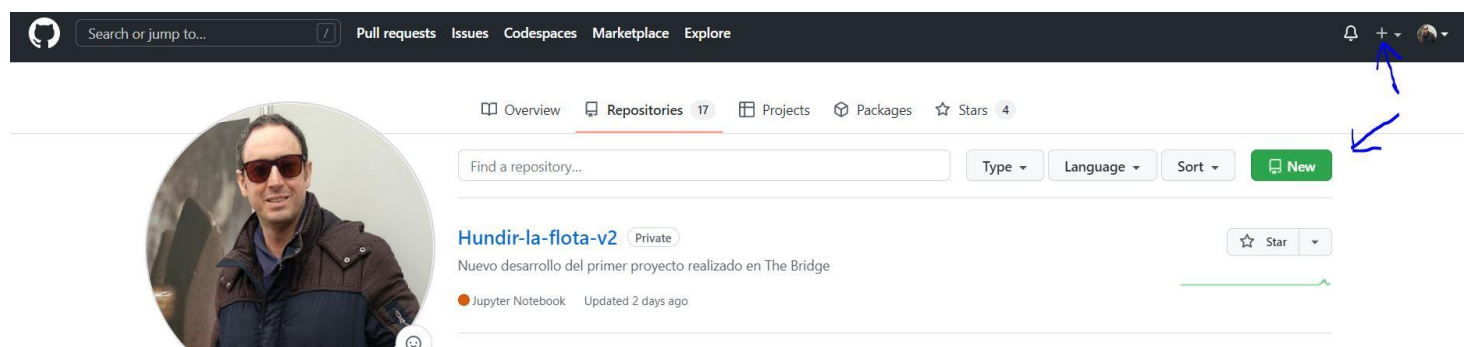
Un repositorio suele ser utilizado para organizar un solo proyecto.

Los datos que pueden estar almacenados en los repositorios son archivos, carpetas, imágenes, videos y conjuntos de datos, cualquier cosa que sea necesaria para tu proyecto. Frecuentemente, los repositorios incluyen un archivo **README**, este archivo posee información sobre tu proyecto.

GitHub facilita todo agregando uno al mismo tiempo que crea su nuevo repositorio.

Estos son los pasos para crear un repositorio:

# Creación del repositorio Remoto en GitHub:

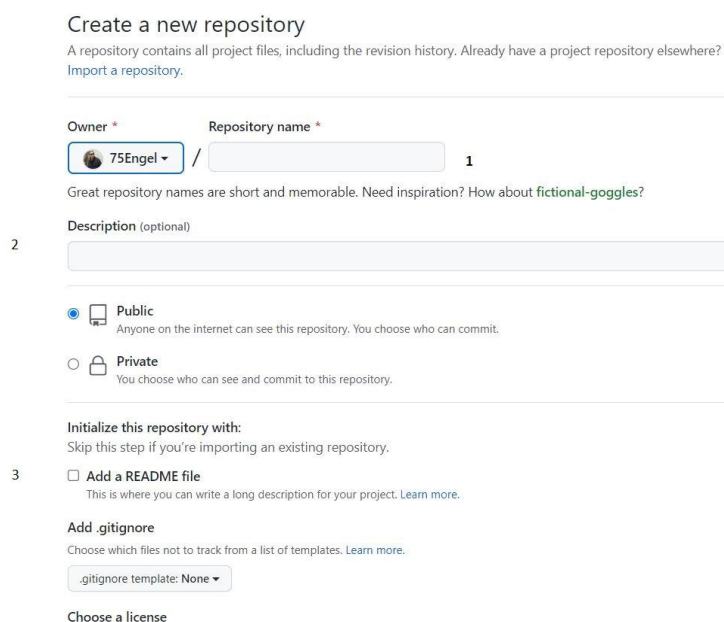


Clicando en cualquiera de estos dos puntos se inicia un nuevo repositorio:

1. Agregas el nombre del repositorio, algo específico

2. Opcional, pero recomendable, haz una pequeña descripción del proyecto.

**No olvides de dejar el repositorio como público**



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \* Repository name \*

75Engel / 1

Great repository names are short and memorable. Need inspiration? How about [fictional-goggles?](#)

Description (optional)

2

☒ Public  
Anyone on the internet can see this repository. You choose who can commit.

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

Initialize this repository with:  
Skip this step if you're importing an existing repository.

3 ☒ Add a README file  
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore  
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None

Choose a license

3. Agrega un Readme.md. Veremos algunos ejemplos de cómo sí o cómo no se debe de hacer un Readme, este fichero se escribe en Markdown.

Es la presentación del repositorio, describe el objetivo, la metodología aplicada, las fuentes, .... Veremos algunos ejemplos de cómo sí o cómo no se debe de hacer un Readme.

Haz click en “Crear repositorio”.

Ya tienes creado un repositorio nuevo en la web de GITHUB!!!!

## Clonamos el repositorio remoto para hacerlo local

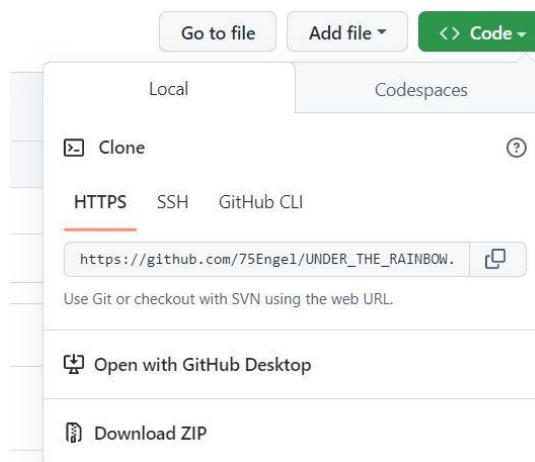
Vamos al repositorio que queremos clonar.

### Windows

Se copia la dirección que aparece en HTTPS

### Mac

Se copia la dirección que aparece en SSH, para lo cual se ha de estar logueado



### Git Bash

Con Explorador de Windows, vamos a la dirección de nuestro ordenador donde queremos dejar el repositorio clonado y abrimos Git Bash Here, donde aparecerá la terminal de Git Bash con la ruta de nuestra posición y realizamos un git clone [dirección copiada]

### Con VisualStudioCode

Abrimos el programa y nos dirigiremos a través de PowerShell a la dirección donde queremos clonar el repositorio. Una vez allí, haremos un git clone.

Ya tenemos vinculado el repositorio de nuestro terminal con el repositorio en GitHub

## CÓMO CAMBIAR EL NOMBRE DEL DIRECTORIO MATRIZ (DE MASTER A MAIN)

Los repositorios recién creados pueden usar el nombre master como la rama principal.

Para asegurar compatibilidad futura, se recomienda que actualices el nombre de esta rama a main.

Para esto es muy útil utilizar la terminal de Git Bash, ya que te permite visualmente saber en qué rama de tu repositorio te vas a encontrar.

1. Revisa el nombre de tu rama corriendo el siguiente comando en el computador de tu hogar.
  - o Si hay varias ramas, la rama activa mostrará un \* a la izquierda. Si ves que la rama principal es master, corre los siguientes comandos para actualizarlo a main.

```
[server]$ git branch -a
```

2. Asegúrate de que la rama activa sea master.

```
[server]$ git checkout master
```

3. Renómbralo usando la opción -m

```
[server]$ git branch -m main
```

Ejemplos de Readme:

Veamos varios ejemplos para ver cómo se hace un Readme:

EDA\_FIFA  
RAINBOW  
CHRISTIAN  
EDA\_MORTANDAD  
EDA\_OLYMPICS  
COVER\_PREDICTION

## Códigos básicos a la hora de trabajar en tu propio Repositorio:

Ya hemos visto cómo actualizar un repositorio desde GitHub hasta tu terminal con

### **git pull**

Ahora veremos cómo actualizar tu repositorio desde la terminal a tu repositorio en GitHub:

1. Hacemos unos cambios en tu repositorio local, modifica el Readme agregándole una línea.
2. Salva el fichero Readme.
3. Desde tu terminal de Powershell/Git Bash, sitúate en el path completo de tu repositorio.
4. Haz un **git add** (hay varias alternativas: `git add .` / `git add -A` / `git add path`)

¿Ha dado un error? No te preocupes, es normal.

Como verás te está diciendo que tienes que añadir el path de tu repositorio local

```
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub_prueba> git add -A
fatal: detected dubious ownership in repository at 'D:/Bootcamp_22/Javier/Repositorios/Taller_Git_hub_prueba'
'D:/Bootcamp_22/Javier/Repositorios/Taller_Git_hub_prueba' is on a file system that doesnot record ownership
To add an exception for this directory, call:
```

```
git config --global --add safe.directory D:/Bootcamp_22/Javier/Repositorios/Taller_Git_hub_prueba
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub_prueba> █
```

---

5. Haz un **git commit**: esto es, crea un comentario para saber qué has hecho en esta versión (eliminar, crear, modificar, ...)
6. Haz **git push** (con esto subes los cambios realizados a tu repositorio remoto).

Ahora vete a él, verás cómo se han realizado estos cambios en tu repositorio



## BUENAS PRÁCTICAS PARA ESCRIBIR COMENTARIOS EN GITHUB

A la hora de trabajar con versiones, es importante tener un cierto orden a la hora de escribir comments, dado que deben de describir las acciones que se han realizado en esta nueva versión y si además trabajas en un grupo de n personas, estandarizar los comments se volverá importante.

[https://www.linkedin.com/posts/midudev\\_cada-d%C3%ADa-escribimos-decenas-de-commits-para-activity-6994668226097860608-k0MF/](https://www.linkedin.com/posts/midudev_cada-d%C3%ADa-escribimos-decenas-de-commits-para-activity-6994668226097860608-k0MF/)

Hasta aquí, hemos visto cómo trabajar en tus propios proyectos y únicamente utilizando el main de tu propio repositorio remoto. Pero, ¿cómo trabajar cuando son varias personas que desarrollan diversos puntos del proyecto? Esto lo vamos a ver a partir de aquí.



# INTRODUCCIÓN AL TRABAJO EN EQUIPO

Cuando hacemos un proyecto, trabajaremos a varios niveles.

En un primer nivel se encuentra y te encuentras en la rama “main”.

La rama main será el “escaparate de una tienda”.

Será lo que queremos que vea nuestro cliente final.

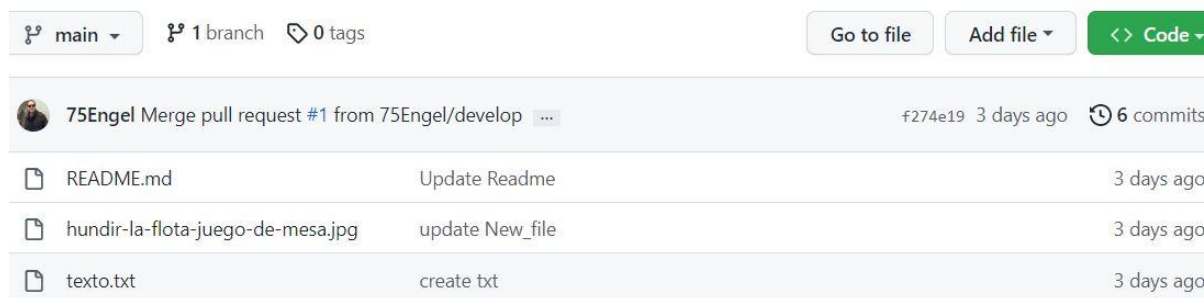
Trabajaremos en niveles inferiores a esta rama para subir todo a la rama main cuando tengamos finalizado el proyecto.

Cuánto más complejidad o número de personas involucradas en el proyecto haya, más profundidad y horizontalidad habrá en las ramas.

Para generar profundidad podemos realizarlo o bien a través de tu repositorio remoto o bien por tu repositorio local.



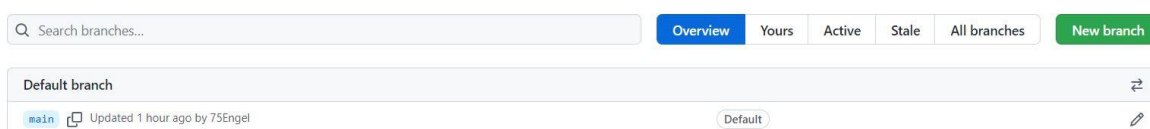
En tu repositorio remoto:



The screenshot shows the top of a GitHub repository page. At the top, there's a navigation bar with 'main' selected, '1 branch', and '0 tags'. To the right are buttons for 'Go to file', 'Add file', and 'Code'. Below this, a merge pull request is shown: '75Engel Merge pull request #1 from 75Engel/develop' with commit hash 'f274e19' and '3 days ago', and '6 commits'. Below the pull request, a list of recent commits is shown:

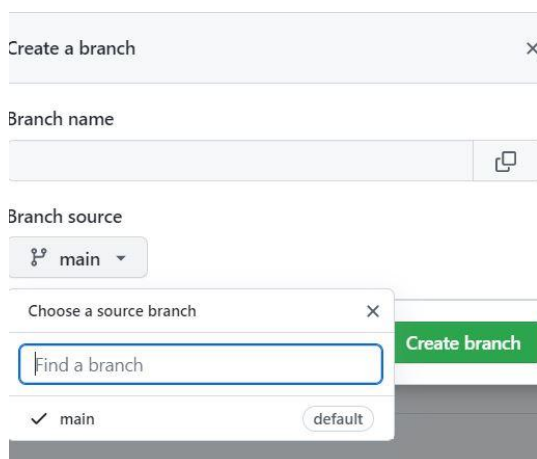
File	Commit Message	Time
README.md	Update Readme	3 days ago
hundir-la-flota-juego-de-mesa.jpg	update New_file	3 days ago
texto.txt	create txt	3 days ago

- Pinchas en branch, aquí aparece el número de ramas existentes en el proyecto



The screenshot shows the 'Branches' page in GitHub. It has a search bar 'Search branches...' and tabs for 'Overview', 'Yours', 'Active', 'Stale', 'All branches', and a 'New branch' button. Below the tabs, the 'Default branch' is shown as 'main', updated 1 hour ago by 75Engel. There is a 'Default' button and an edit icon.

- Pincha en New branch para generar una nueva rama.



The screenshot shows the 'Create a branch' dialog. It has a 'Branch name' input field and a 'Branch source' dropdown menu set to 'main'. Below the dropdown, there's a 'Choose a source branch' dialog with a search bar 'Find a branch' and a list of branches: 'main' (checked) and 'default'. A 'Create branch' button is visible on the right.

- Crea la rama develop, cuyo origen será la rama 'main'. Si existen ya ramas creadas, podrás definir cuál es la rama desde la que se va a generar a través de Branch source

En tu repositorio local:

Aquí trabajaremos principalmente en el repositorio local para subir los cambios al repositorio remoto.

Para crear ramas en niveles inferiores usaremos el comando **git branch nombre\_rama**.

Con el comando **git branch** a secas, comprobamos las ramas que tenemos en nuestro repositorio local. Cuando creamos una rama, al crearse a un nivel inferior, la nueva rama tendrá los archivos desde la que se crea (lo veremos más adelante).

Por ejemplo, si estamos en la rama main y creamos la develop, la rama develop tendrá los archivos que estaban en main, pero al revés no.

Aquí sí es importante estar colocado en la rama desde la que se va a generar el siguiente nivel inferior.

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> git branch
develop
* main
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> █
```

Sale que tenemos en nuestro repositorio local la rama main (con asterisco y en color verde porque es en la que estamos actualmente) y la rama develop. Para ver las ramas que tenemos tanto en nuestro repositorio local como el remoto, hacemos **git branch -a**.

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> git branch
develop
* main
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> git branch -a
develop
* main
remotes/origin/HEAD -> origin/main
remotes/origin/main
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> █
```



En rojo muestra nuestras ramas en remoto. Como se puede observar, la rama develop que hemos creado solo está en local, no en el remoto.

Para subir las ramas a nuestro repositorio remoto, cambiaremos a la rama que queremos subir a nuestro repositorio remoto (**git checkout** o **git switch** nombre\_rama) y hacemos **git push origin** nombre\_rama.

Ya tenemos hecha la rama en la que vamos a trabajar.

La rama develop será donde crearemos nuestra beta del proyecto, por lo que el paso final será fusionar la rama develop con main.

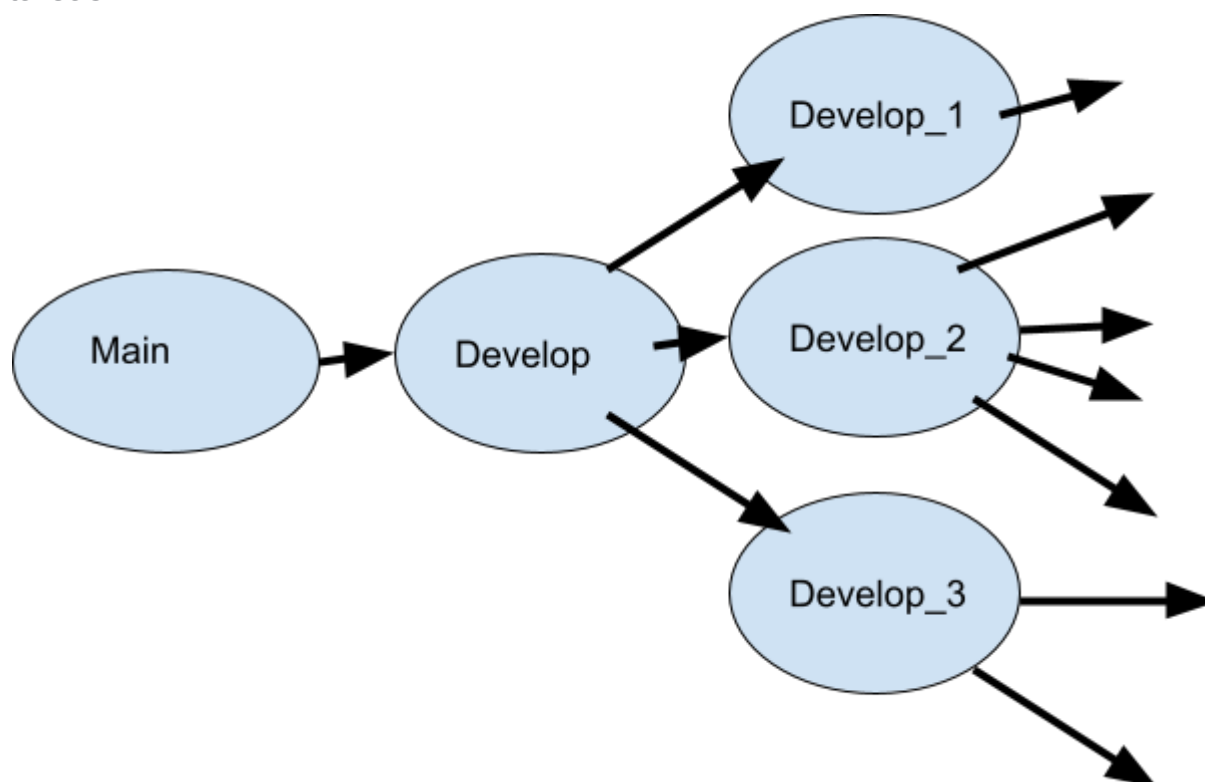
## Ejemplo de trabajo

Un proyecto, normalmente, se divide en varias tareas, que realizan diversos grupos.

Por ejemplo, un Hundir la Flota podría tener 3 grupos:

- un grupo que construya el tablero y lo defina,
- un grupo que genere los barcos, manual o aleatoriamente,
- un último grupo que se ocupe de los disparos y la contabilización de tocados y hundidos.

Sabiendo esto, tendríamos que crear una rama para cada una de esas tareas.



A diferencia de la creación de ramas en GitHub, aquí tenemos que cambiar a la rama desde la que deseamos crear las nuevas ramas para las tareas, y cada grupo haría un git branch nombre\_rama donde realizar sus partes de proyectos.

Por cierto, ¿está imagen es correcta?

## Trabajo en el repositorio local:

Hemos creado las ramas develop\_x (SOLO UNA POR PERSONA, salvo que esté en varios proyectos a desarrollar), vamos a crear en una de ellas una profundidad más en donde vamos a trabajar. Nombrarla “ticket\_[nombre]\_1”.

```
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> git switch develop
Switched to branch 'develop'
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> git branch develop_1
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> git switch develop_1
Switched to branch 'develop_1'
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> git branch ticket_javier_1
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> git switch ticket_javier_1
Switched to branch 'ticket_javier_1'
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> █
```

Vamos a agregar un fichero, tomad la imagen del entregable HLF y pegarla en la rama que habéis creado.

Para verificar la situación de esta rama, haced un **git status**.

```
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> git status
On branch ticket_javier_1
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  hundir-la-flota-juego-de-mesa.jpg

nothing added to commit but untracked files present (use "git add" to track)
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> █
```

← Branch en el que te encuentras

← Ficheros existentes y su situación

Realizamos un **git add .** / **git commit -m 'create image'**

Si volvemos a cambiar a “develop\_n” (que sería el nivel justo superior), nos encontraremos con que desaparece el archivo.

Esto es debido a que los cambios se han realizado en vuestro ticket, no en la rama en la que estamos ahora. Para que aparezca también aquí, tendremos que hacer un **git merge** ticket (se hace en la rama a la que queremos llevar los cambios y se pone el nombre de la rama en la que se han hecho esos cambios).

```
create mode 100644 hundir-la-flota-juego-de-mesa.jpg
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> git switch develop_1
Switched to branch 'develop_1'
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> git merge ticket_javier_1
Updating c3ee941..08af1dd
Fast-forward
 hundir-la-flota-juego-de-mesa.jpg | Bin 0 -> 71794 bytes
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 hundir-la-flota-juego-de-mesa.jpg
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> █
```

---

Como veis, ahora aparece el archivo en la “develop\_x”. Si ahora mismo, volviera a crear otra rama (por ejemplo, “ticket\_[nombre]\_2”), esta nueva rama contendría el archivo de la imagen.

Si tuviera que hacer más subtarear, procedemos del mismo modo. Entramos en la rama de la subtarea, hacemos los cambios pertinentes, volvemos a la “develop\_x” y hacemos el git merge.

Vamos a crear un fichero .txt en esta rama y escribimos lo siguiente:

“Con varios niveles de profundidad para aprender a manejarnos trabajando estamos muy importante este taller de guardar debes...”

Cierralo y haz un git status.

```
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> git status
On branch ticket_javier_2
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    texto

nothing added to commit but untracked files present (use "git add" to track)
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> git add .
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> git commit -m 'create txt'
[ticket_javier_2 e3deb2e] create txt
 1 file changed, 2 insertions(+)
 create mode 100644 texto
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> █
```

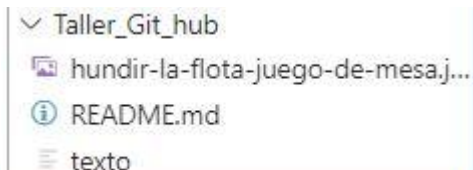
---

Hacemos un git add . , git commit -m ‘create txt’



A continuación, subimos un nivel con `git switch develop_x` y hacemos `git merge ticket_[nombre]_2`

```
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> git switch develop_1
Switched to branch 'develop_1'
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub> git merge ticket_javier_2
Updating 08af1dd..e3deb2e
Fast-forward
 texto | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 texto
PS D:\Bootcamp_22\Javier\Repositorios\Taller_Git_hub>
```



Una vez hemos hecho el merge de todos los archivos, hacemos un `git status` para comprobar que no hay nada de commitar, ya que lo has realizado en el nivel inferior.

Cuando tengamos esto, haremos un `git push origin develop_x` para subir la tarea al repositorio remoto.

Hasta aquí, todo el trabajo a realizar en el repositorio local vuestro.

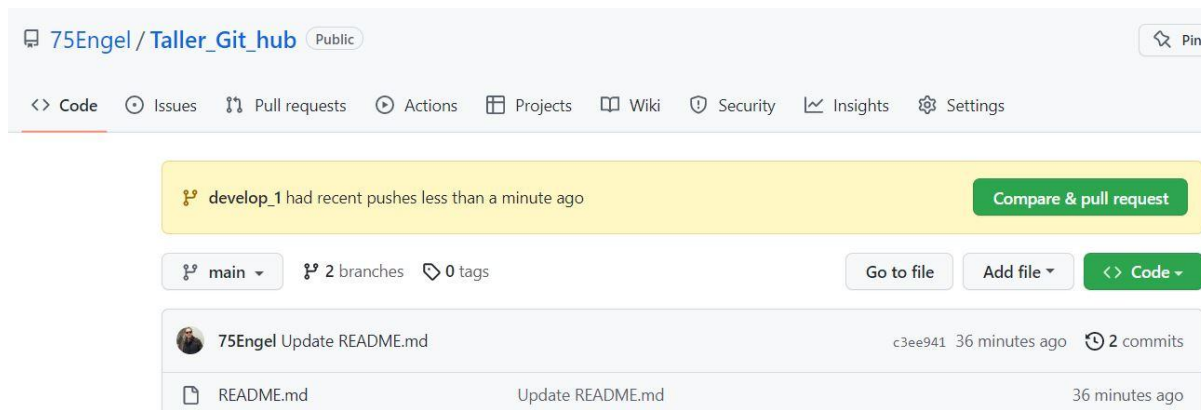
- Hemos creado varios niveles de profundidad para trabajar en el proyecto.
- Hemos creado varios tickets donde trabajar por partes en nuestra parte del proyecto.
- Hemos realizado fusiones de ramas para juntar diversas partes de vuestro proyecto.

¿Qué viene ahora?

- Confirmación de la subida de los tickets al repositorio remoto.
- Revisión y chequeo de los trabajos realizados por cada desarrollador.
- Fusión con la rama general de trabajo.

Trabajo en el repositorio remoto:

Si vamos a GitHub, nos damos cuenta de que ha salido una notificación:



The screenshot shows the GitHub interface for the repository '75Engel / Taller\_Git\_hub'. At the top, there's a navigation bar with links to Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below this, a yellow notification bar states 'develop\_1 had recent pushes less than a minute ago' with a 'Compare & pull request' button. Underneath, a summary of the repository shows 'main' as the selected branch, '2 branches', and '0 tags'. A recent commit by '75Engel' is listed, titled 'Update README.md', with commit hash 'c3ee941' and a timestamp of '36 minutes ago'. Below the commit summary, a file named 'README.md' is shown with the action 'Update README.md' and a timestamp of '36 minutes ago'.

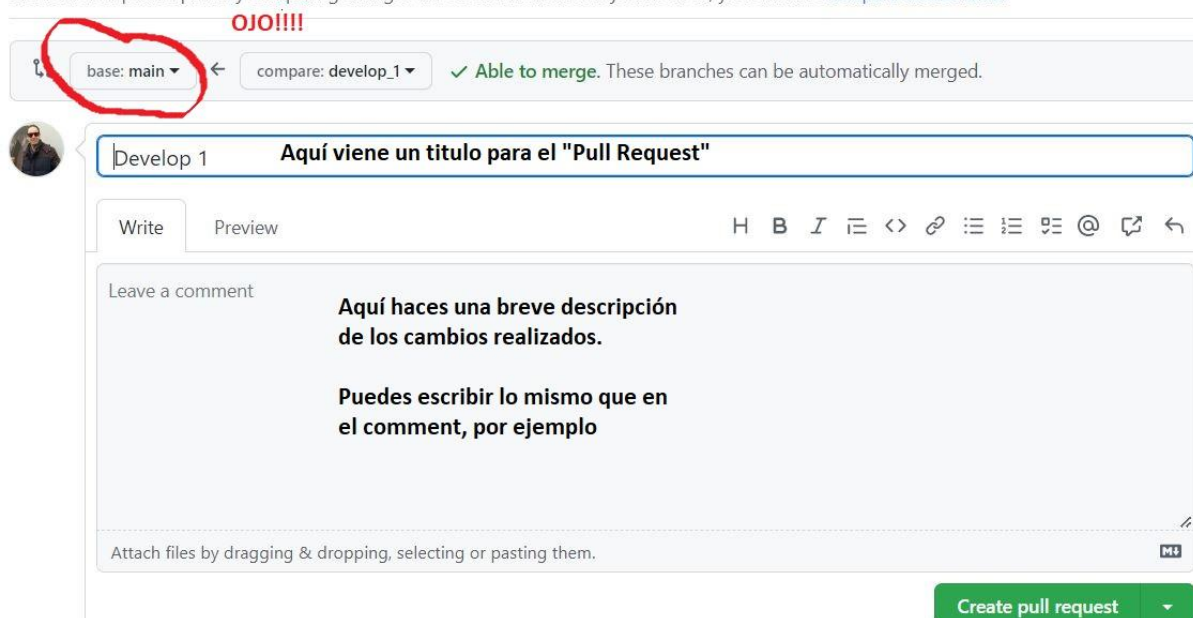
De este modo, hemos creado nuestra primera **PULL REQUEST**.

Esta Pull Request nos indica que alguien está intentando subir un archivo al repositorio remoto.

Si hacemos clic en el botón, nos llevará a la siguiente pantalla:

### Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

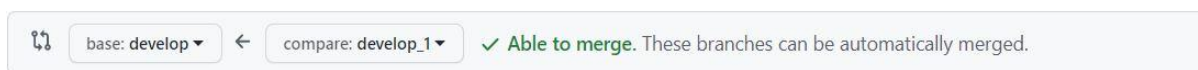


The screenshot shows the 'Open a pull request' form. At the top, there's a red warning 'OJO!!!!'. Below it, a section for branch selection shows 'base: main' and 'compare: develop\_1', with a green checkmark and the text 'Able to merge. These branches can be automatically merged.' Below this, there's a profile picture of a user and a text input field with the placeholder 'Aquí viene un título para el "Pull Request"'. Underneath, there are tabs for 'Write' and 'Preview'. The 'Write' tab is active, showing a text area with the placeholder 'Leave a comment' and instructions: 'Aquí haces una breve descripción de los cambios realizados.' and 'Puedes escribir lo mismo que en el comment, por ejemplo'. At the bottom, there's a text area for attaching files and a green 'Create pull request' button.

Aquí le pondremos una descripción de los cambios que se hayan hecho, para que todo el que quiera ver los cambios realizados sepa qué se ha hecho (es buena praxis).

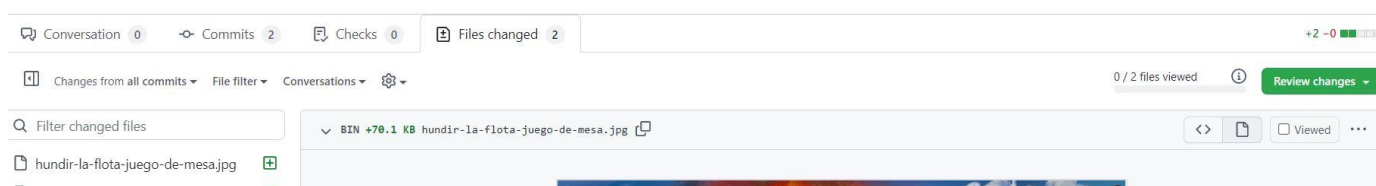


Como hemos dicho que trabajamos en develop y no en main, habrá que hacer el siguiente cambio:

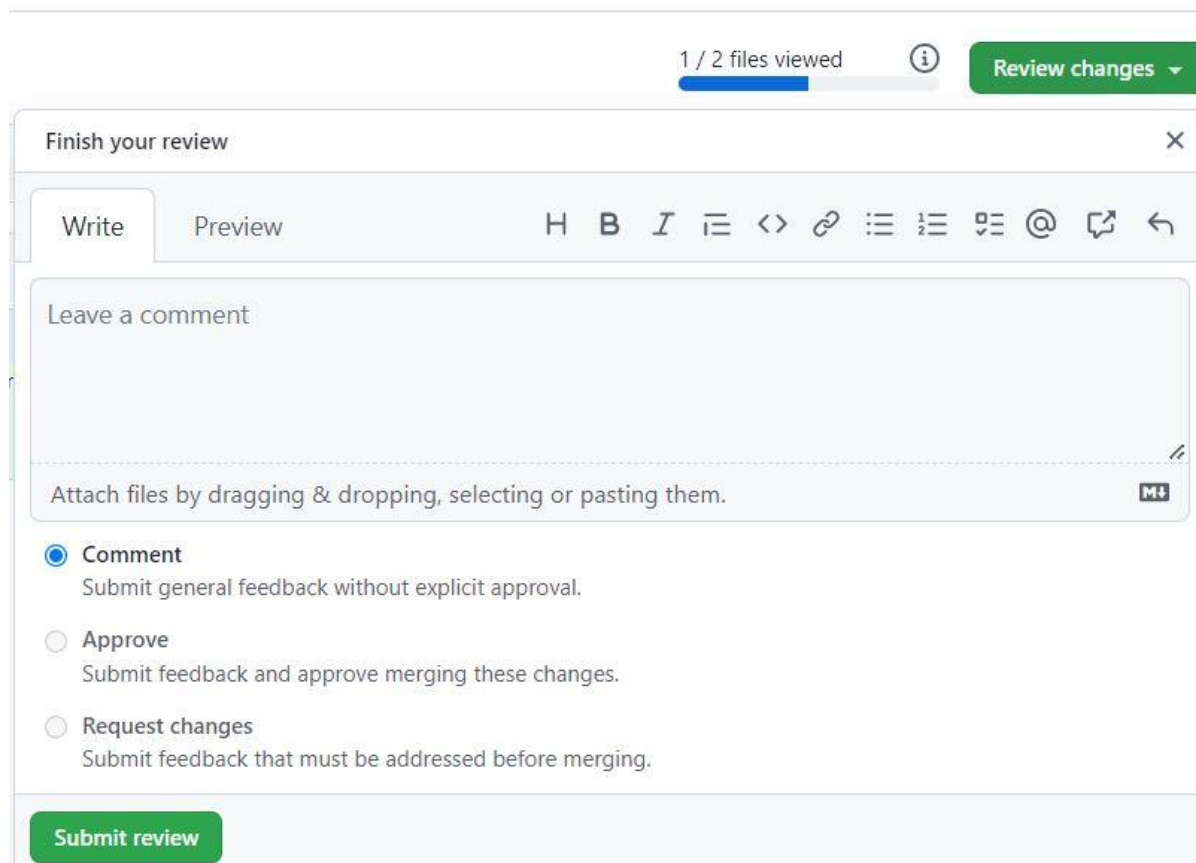


Ahora damos en “Create pull request”.

De esas 4 pestañas, en la que pone “Files changed” podemos ver los cambios que se han realizado.




Una vez comprobado todo, hacemos clic en “**Review changes**” y ahí nos dará opción de poner un comentario, aprobar o solicitar cambios (como nos lo estamos haciendo a nosotros mismos la pull request, no nos deja cambiar de opción).





Ponemos un comentario (esta vez sí es obligatorio) y damos a “Submit review”.


Aparece lo siguiente. Hacemos el “Merge pull request” y lo confirmamos

Add more commits by pushing to the **develop** branch on **75Engel/Taller\_Git\_hub\_prueba**.




 **Continuous integration has not been set up**  
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

 **This branch has no conflicts with the base branch**  
Merging can be performed automatically.

**Merge pull request**  You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

---

Add more commits by pushing to the **develop\_1** branch on **75Engel/Taller\_Git\_hub**.







Merge pull request #1 from 75Engel/develop\_1



Primer request


This commit will be authored by 106105557+75Engel@users.noreply.github.com



**Confirm merge** **Cancel**


Y ya tendremos nuestra rama develop actualizada en el repositorio remoto.




 **develop**   **3 branches**  **0 tags**

[Go to file](#) [Add file](#)  **Code** 

This branch is **3 commits ahead** of main. [Contribute](#) 

 **75Engel** Merge pull request #1 from 75Engel/develop\_1 

4965c3c 1 minute ago  **5 commits**

 README.md	Update README.md	1 hour ago
 hundir-la-flota-juego-de-mesa.jpg	create image	32 minutes ago
 texto	create txt	27 minutes ago



Si volvemos a la página principal del repositorio (haciendo clic en el nombre del repo), volverá a aparecer una Pull Request, ¿por qué?

Esto es debido a que los cambios no han llegado al main, ya que anteriormente lo hicimos desde develop\_x a la rama develop, y te da la opción de hacerlo desde develop a main.

En principio no le haremos caso hasta no terminar el proyecto.

Una vez hayamos terminado, sí que habrá que hacerlo.  
Será del mismo modo que el anterior pero de develop a main.

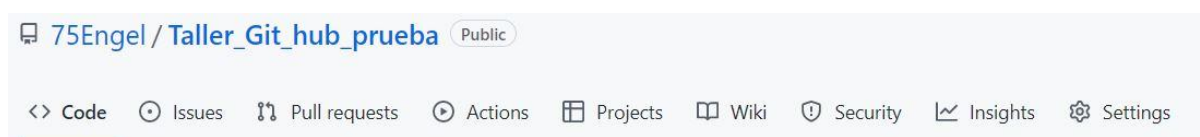
Pero... ¿solo trabajamos en nuestra cuenta?

Cómo hemos dicho antes GitHub permite que varias personas puedan colaborar para desarrollar un programa, una app, un proyecto en general.

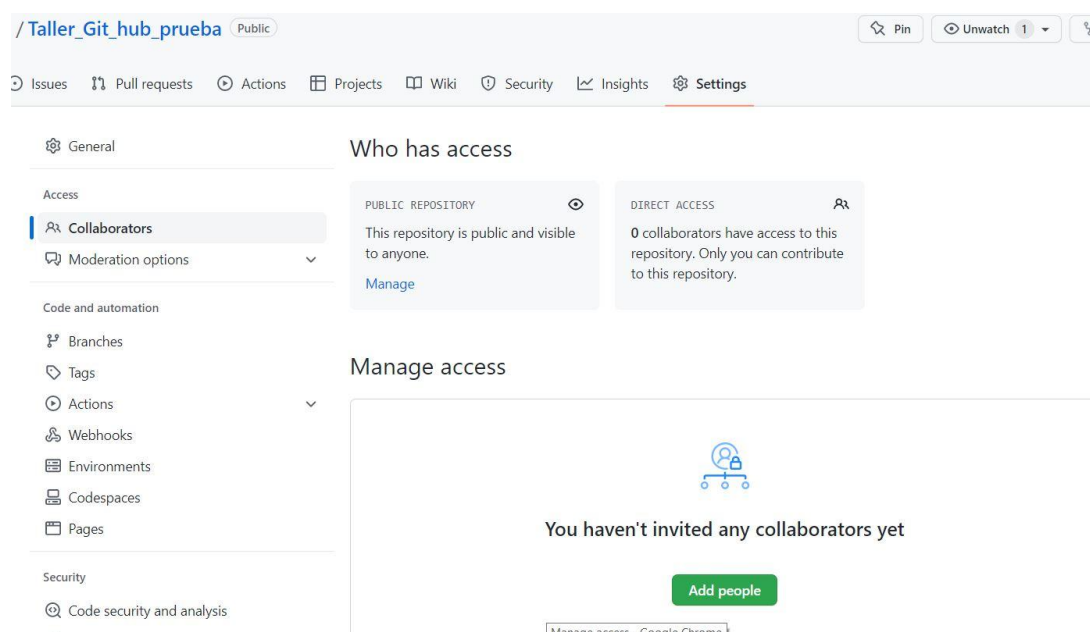
De esta manera, lo explicado anteriormente se pueden repartir tareas a partir de un único Repositorio, donde se dará permisos a otros usuarios a trabajar en él y, en su caso, permitir que se pueda tener acceso a este Repositorio desde su propia cuenta de GitHub.

¿Cómo se puede agregar colaboradores a un repositorio?

Estamos en nuestro repositorio, somos el administrador del proyecto:



Os dirigís a Settings y de ahí a Collaborators y clickais en Add people

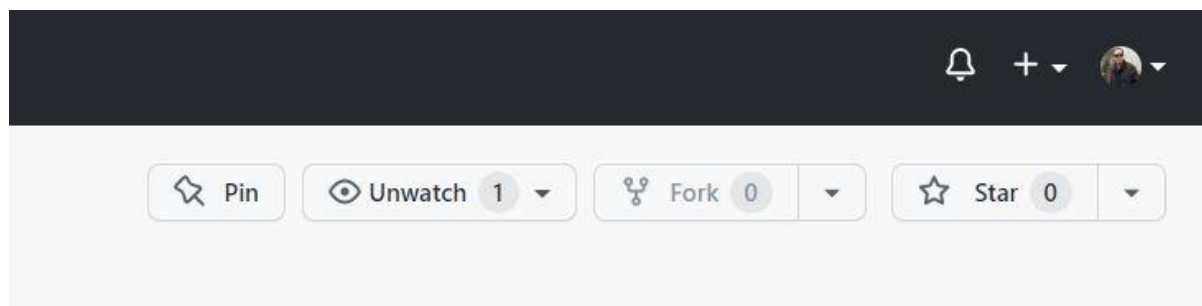


Os pedirá que confirméis que sois la persona responsable de este GitHub (importante acordarse de la contraseña)

A partir de aquí, solo tenéis que escribir el nick de vuestro@s compañer@s de proyecto y de este modo les llegará una invitación a unirse al mismo.

y ¿cómo agrego un repositorio en el que he colaborado a mi GitHub?

En nuestro GitHub disponemos de un botón para que nuestros colaboradores puedan realizar las copias de nuestros repositorios.



Nos situamos en el repositorio en el que hemos colaborado y en la zona derecha de la página, clicamos en Fork y Create a new Fork.

Aparecerá la siguiente pestaña

### Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Owner \*

 75Engel ▾

Repository name \*


 

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

☒ Copy the `main` branch only

Contribute back to Masara00/me\_gusta\_el\_furgol\_3\_0 by adding your own branch. [Learn more.](#)

 You are creating a fork in your personal account.

Create fork

Volvéis a clicar en Create fork.

Y ya está!!

## CHEATSHEETS DE GITHUB

Aqui dejamos un par de Cheatsheets para que tengáis acceso a diversos comandos que se utilizan en repositorios de GitHub

<https://education.github.com/git-cheat-sheet-education.pdf>

<https://github.com/MikeRodeman/Git-Cheat-Sheet>