

APUNTES DE GITHUB

https://www.programacionfacil.org/cursos/git_github

<https://www.xataka.com/basics/que-github-que-le-ofrece-a-desarrolladores>

Github se trata de una de las principales plataformas para crear proyectos abiertos de herramientas y aplicaciones, fue comprada por Microsoft en junio del 2018. **Colaborativo.** Puede ser descargado y revisado por cualquier usuario lo que ayuda a mejorar el producto y crear ramificaciones a partir de él. **Y si prefieres que tu código no se vea, también pueden crearse proyectos privados.**

Github utiliza el sistema de control de versiones Git.

Git es uno de estos sistemas de control de versiones, que permite comparar el código de un archivo **para ver las diferencias entre las versiones**, restaurar versiones antiguas si algo sale mal, y fusionar los cambios de distintas versiones. También permite trabajar con distintas ramas de un proyecto, como la de desarrollo para meter nuevas funciones al programa o la de producción para depurar los bugs.

Las principales características de la plataforma es que ofrece las mejores características de este tipo de servicios sin perder la simplicidad, y es **una de las más utilizadas del mundo** por los desarrolladores. Es multiplataforma, y tiene multitud de interfaces de usuario.

Así pues, Github es **un portal para gestionar las aplicaciones que utilizan el sistema Git**

1. Crear una cuenta en github.

https://www.programacionfacil.org/cursos/git_github/capitulo_11_crear_cuenta_github.html

Como ya la tenéis de programación no hace falta que lo veamos de nuevo.

2. Crear un repositorio.

Una vez creada tu cuenta te logeas y vamos a crear un repositorio en GitHub




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 *

 BLANCAPROF ▾

 /

repositorio_pruebas ✓

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

Description (optional)

repositorio para pruebas

☐ 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.

☒ 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

A license tells others what they can and can't do with your code. [Learn more.](#)

License: None ▾

This will set  main as the default branch. Change the default name in your [settings](#).

 You are creating a private repository in your personal account.

Create repository

3. Subir un proyecto a nuestro repositorio GitHub

- `git remote add <alias_repo> <url>`: Nos permite añadir un repositorio remoto y asociarlo con un alias. `git clone` automáticamente crea el alias origin apuntando a la url desde la que clonamos.
- `git remote -v`: Lista todos los alias a repositorios remotos que tenemos dados de alta en nuestro repositorio local.
- `git remote rename <nombre_antiguo> <nombre_nuevo>`: Modifica el alias de un repositorio remoto.
- `git remote remove <remoto>`: Elimina el alias del remoto indicado.
- `git push <repo-remoto> <rama>`: Envía todos los commits de la rama indicada que no existan en el repositorio remoto.
- `git pull <repo>`: Es el equivalente a realizar un fetch + merge (lo veremos más adelante). Si no indicamos repositorio, por defecto usa el repositorio origin.

1º conectar el repositorio local con el repositorio de GitHub.

```
Usuario@LAPTOP-K7QKBJ9Q MINGW64 ~/Documents/proyectos_git/proyecto_1 (master)
$ git remote add origin https://github.com/BLANCAPROF/repositorio_pruebas
```

Si te equivocas puedes deshacerlo con `git remote rm origin` y para ver lo que tienes en el origin puedes hacerlo con `git remote -v`

```
Usuario@LAPTOP-K7QKBJ9Q MINGW64 ~/Documents/proyectos_git/proyecto_1 (master)
$ git remote rm origin

Usuario@LAPTOP-K7QKBJ9Q MINGW64 ~/Documents/proyectos_git/proyecto_1 (master)
$ git remote -v

Usuario@LAPTOP-K7QKBJ9Q MINGW64 ~/Documents/proyectos_git/proyecto_1 (master)
$ git remote add origin https://github.com/BLANCAPROF/repositorio_pruebas

Usuario@LAPTOP-K7QKBJ9Q MINGW64 ~/Documents/proyectos_git/proyecto_1 (master)
$ git remote -v
origin https://github.com/BLANCAPROF/repositorio_pruebas (fetch)
origin https://github.com/BLANCAPROF/repositorio_pruebas (push)
```

Ahora ya está listo para enviar a github. `git push origin master`

La primera vez, se abrirá una ventana y me pedirá el usuario y le tendremos que dar permiso, antes de esto **crear un token nuevo copiarlo en algún sitio que lo tengáis a mano siempre**

MINGW64:/c:/Users/Usuario/Documents/proyectos_git/proyecto_1

```
Usuario@LAPTOP-K7QKBJ9Q MINGW64
$ git remote rm origin

Usuario@LAPTOP-K7QKBJ9Q MINGW64
$ git remote -v

Usuario@LAPTOP-K7QKBJ9Q MINGW64
$ git remote add origin https://

Usuario@LAPTOP-K7QKBJ9Q MINGW64
$ git remote -v
origin https://github.com/BLANC
origin https://github.com/BLANC

Usuario@LAPTOP-K7QKBJ9Q MINGW64
$ git push origin master
```

GitHub Login

GitHub
Login

Username or email

Password



Login



Don't have an account? Sign up
Forgot your password?

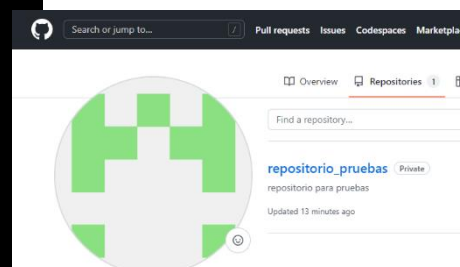
Desde el 13 de agosto de 2021, GitHub ya no acepta contraseñas de cuentas al autenticar las operaciones de Git. En lugar de la contraseña, debemos agregar una **PAT (Personal Access Token)**

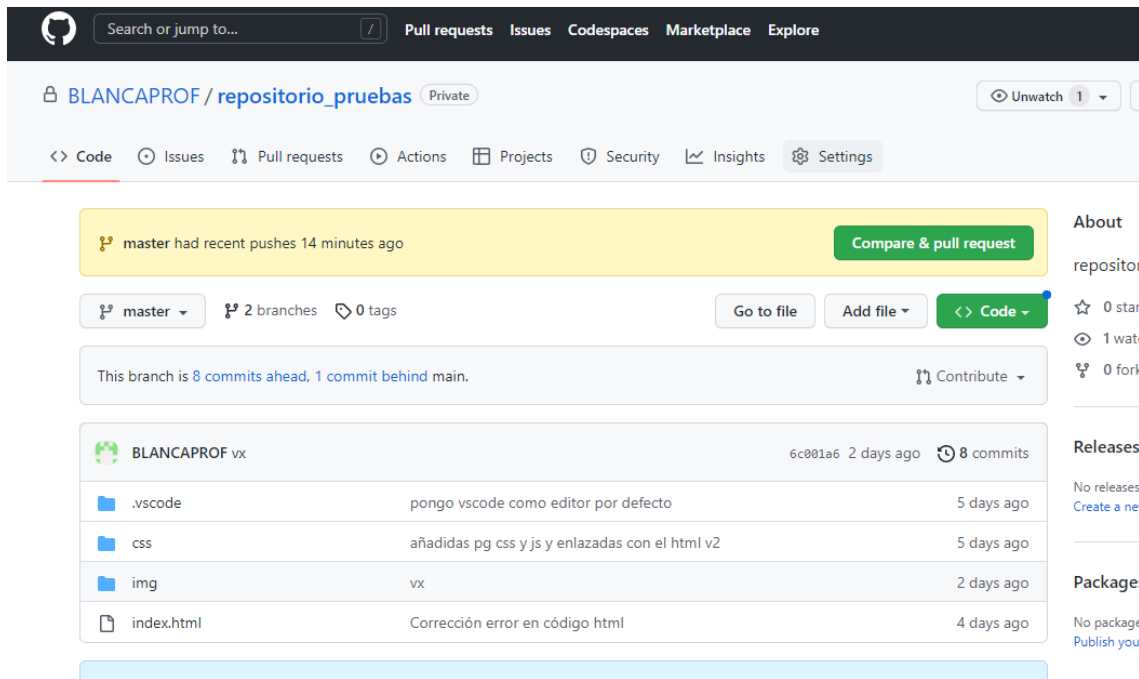
Crear Personal Access Token en GitHub

Desde tu cuenta de GitHub,

- I. Settings =>
- II. Developer Settings =>
- III. Personal Access Token =>
- IV. Generate New Token (Give your password) =>
- V. Fill-up the form => click Generate token
=> Copy the generated Token

```
Usuario@LAPTOP-K7QKBJ9Q MINGW64 ~/Documents/proyectos_git/proyecto_1 (master)
$ git push origin master
Ligon failed, use ctrl+c to cancel basic credential prompt.
Enumerating objects: 35, done.
Counting objects: 100% (35/35), done.
Delta compression using up to 4 threads
Compressing objects: 100% (28/28), done.
Writing objects: 100% (35/35), 40.69 KiB | 2.71 MiB/s, done.
Total 35 (delta 9), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (9/9), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/BLANCAPROF/repositorio_pruebas/pull/new/master
remote:
To https://github.com/BLANCAPROF/repositorio_pruebas
* [new branch]      master -> master
```

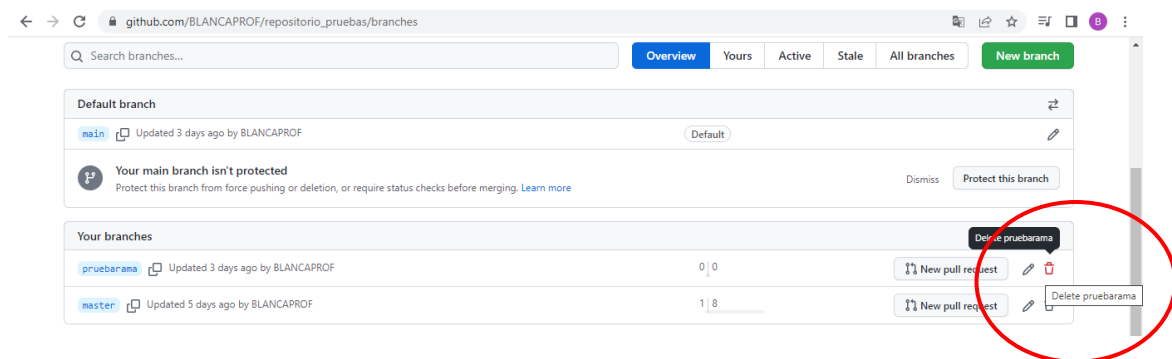




Se han subido todos los archivos menos los que hay en el archivo .gitignore

4. Eliminar una rama en Github

Para eliminar una rama de GitHub, entra en un repositorio y haz click en el botón "Branches". Haz click en el icono de la papelera roja de la rama que quieras eliminar.



¿Cómo se puede restaurar una rama eliminada de GitHub?

Al eliminar una rama en GitHub, el icono de papelera cambiará a un botón "Restore" que sirve para restaurar una rama eliminada.

¿Cómo se cambia la rama por defecto de Git?

Para cambiar la rama por defecto de Git, solo tienes que abrir la consola Git Bash y escribir el siguiente comando: **git config --global init.defaultBranch main**

Esto da el nombre de main como rama principal por defecto en todos los proyectos. Si le indicas otro nombre, será ese nombre en lugar de main.

Parece ser que en versiones anteriores a la 2.28 de Git, este comando no funciona. Utiliza el comando `git version` para comprobar tu versión de Git. Si es antigua, quizás sea el momento de actualizar.

Los repositorios ya creados con rama master seguirán con ella. Los nuevos, tendrán la rama main.

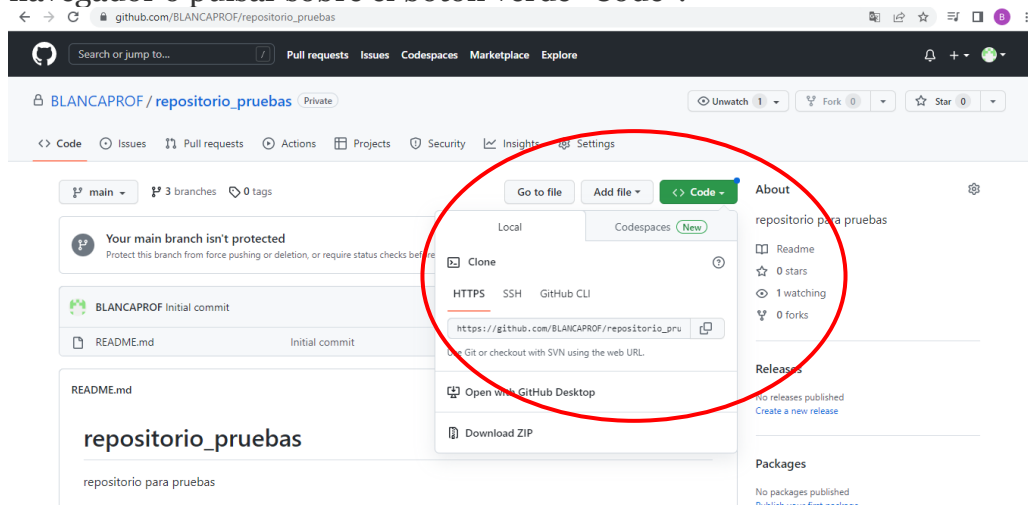
```
Usuario@LAPTOP-K7QKB39Q MINGW64 ~/Documents/proyectos_git/proyecto_3
$ git init
Initialized empty Git repository in C:/Users/Usuario/Documents/proyectos_git/proyecto_3/.git/
```

```
Usuario@LAPTOP-K7QKB39Q MINGW64 ~/Documents/proyectos_git/proyecto_3 (main)
$ ls
index2.html
```

5. Clonar un proyecto a nuestro repositorio

`git clone <repository> [<directory>]` # Clonar ruta de repositorio a directorio si se define, sino usa nombre del repositorio

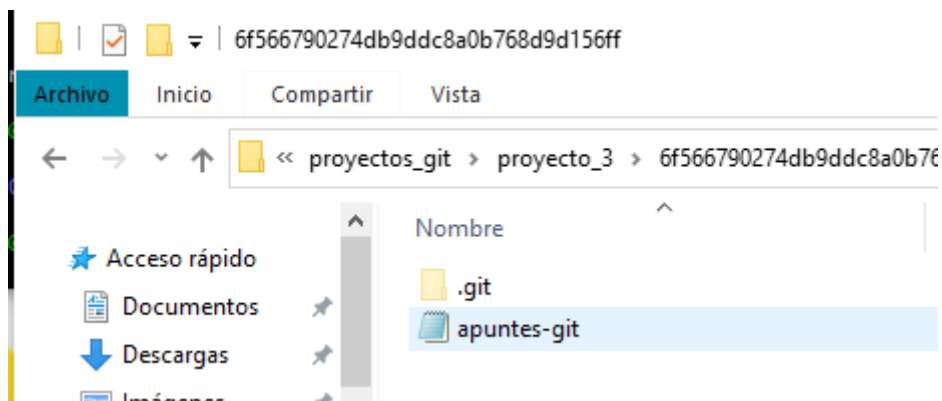
Para clonar (descargar) un repositorio de GitHub en tu equipo, necesitas copiar la URL del mismo. Lo puedes hacer desde la barra de URL del navegador o pulsar sobre el botón verde "Code":



Abre Git Bash en la ruta en la que quieras que se añada la carpeta del repositorio. Escribe el siguiente comando:

```
git clone https://gist.github.com/6f566790274db9ddc8a0b768d9d156ff.git
Usuario@LAPTOP-K7QKB39Q MINGW64 ~/Documents/proyectos_git/proyecto_3 (main)
$ git clone https://gist.github.com/joarias1/6f566790274db9ddc8a0b768d9d156ff
Cloning into '6f566790274db9ddc8a0b768d9d156ff'...
remote: Enumerating objects: 6, done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 6
Unpacking objects: 100% (6/6), 2.68 KiB | 20.00 KiB/s, done.

Usuario@LAPTOP-K7QKB39Q MINGW64 ~/Documents/proyectos_git/proyecto_3 (main)
$ ls
6f566790274db9ddc8a0b768d9d156ff/
```



¿Qué ocurre si hay más de una persona trabajando en un mismo proyecto de GitHub?

GitHub está más pensado para trabajar en equipo que en solitario. ¿Qué pasa si eliminamos un archivo en el que otra persona está trabajando?

Esto no es un problema para Git ya que para poder subir cualquier pequeño cambio al repositorio de GitHub, tendremos que realizar una sincronización del estado actual del repositorio.

Esto se hace con el comando **git pull**, una combinación del comando **git fetch** y **git merge**.

git fetch lo que hace es actualizar todas las referencias de una rama remota en el repositorio local.

git merge lo que hace es fusionar la referencia de rama remota correspondiente a la rama actual.

No hace falta de momento que entiendas estos dos comandos. Vamos a utilizar **git pull** para realizar estas dos acciones, lo que se resume en que se nos **sincronizarán los cambios entre el repositorio de GitHub y el local**

Probémoslo sin realizar ningún cambio desde la clonación, a ver que pasa. Tienes que estar con el repositorio local activo en Git Bash como en la imagen:

```
Usuario@LAPTOP-K7QKBJ9Q MINGW64 ~/Documents/proyectos_git/proyecto_3
$ cd 6f566790274db9ddc8a0b768d9d156ff

Usuario@LAPTOP-K7QKBJ9Q MINGW64 ~/Documents/proyectos_git/proyecto_3/6f566790274db9ddc8a0b768d9d156ff (master)
$ ls
apuntes-git.md

Usuario@LAPTOP-K7QKBJ9Q MINGW64 ~/Documents/proyectos_git/proyecto_3/6f566790274db9ddc8a0b768d9d156ff (master)
$ git pull
Already up to date.
```

El resultado es que me da un mensaje de "Already up to date.", que significa en español que está actualizado. Es decir, tenemos el repositorio igual que como está en GitHub.

Ahora, supongamos que alguien sube un archivo al repositorio y hace un commit en GitHub.

¿Cómo se pueden crear o importar archivos en GitHub?

Voy a hacerlo con otro repositorio mío para no tocar nada en el otro.

```
Usuario@LAPTOP-K7QKB39Q MINGW64 ~/Documents/proyectos_git/proyecto_2/proyecto2 (ramal)
$ git remote add origin https://github.com/BLANCAPROF/proyecto2.git

Usuario@LAPTOP-K7QKB39Q MINGW64 ~/Documents/proyectos_git/proyecto_2/proyecto2 (ramal)
$ git push origin ramal
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 846 bytes | 141.00 KiB/s, done.
Total 9 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/BLANCAPROF/proyecto2.git
 * [new branch]      ramal -> ramal

Usuario@LAPTOP-K7QKB39Q MINGW64 ~/Documents/proyectos_git/Copia_de_Proyecto_2
$ ls
proyecto2/

Usuario@LAPTOP-K7QKB39Q MINGW64 ~/Documents/proyectos_git/Copia_de_Proyecto_2
$ cd proyecto2

Usuario@LAPTOP-K7QKB39Q MINGW64 ~/Documents/proyectos_git/Copia_de_Proyecto_2/proyecto2 (ramal)
$ git pull
Already up to date.
```

Haz click en el botón "Add file" del repositorio.

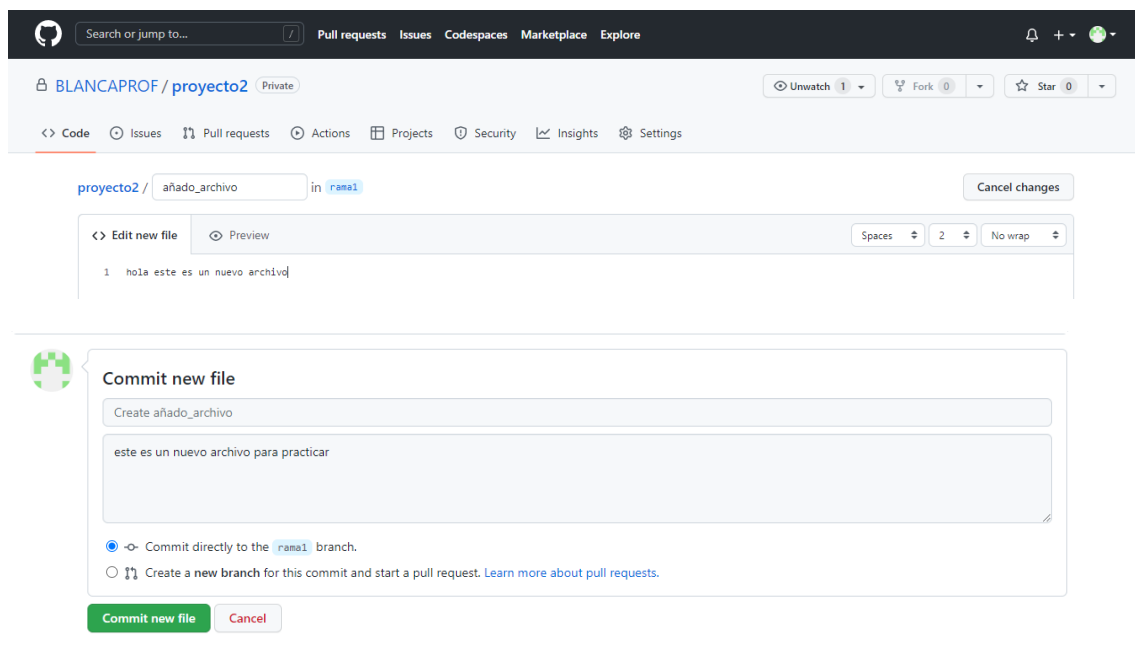
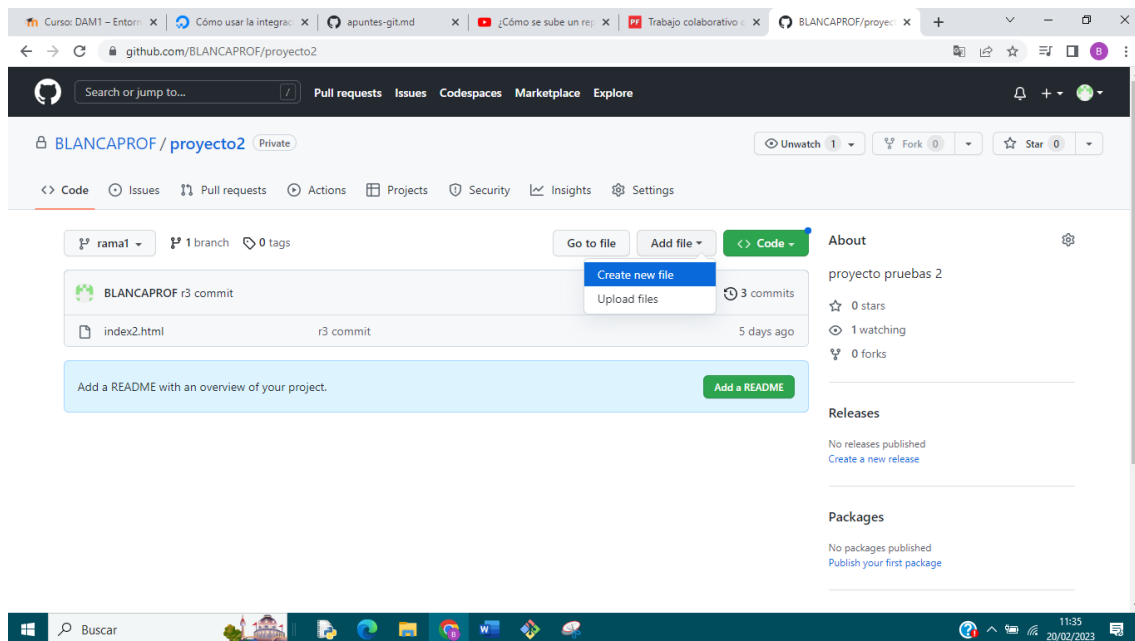
Si quieres crear un archivo haz click en "Create new file"../

En cambio, si lo que deseas es importar archivos ya creados, lo puedes hacer con el botón "Upload files".

Al crear un archivo, le daremos un nombre y escribimos lo que queramos.

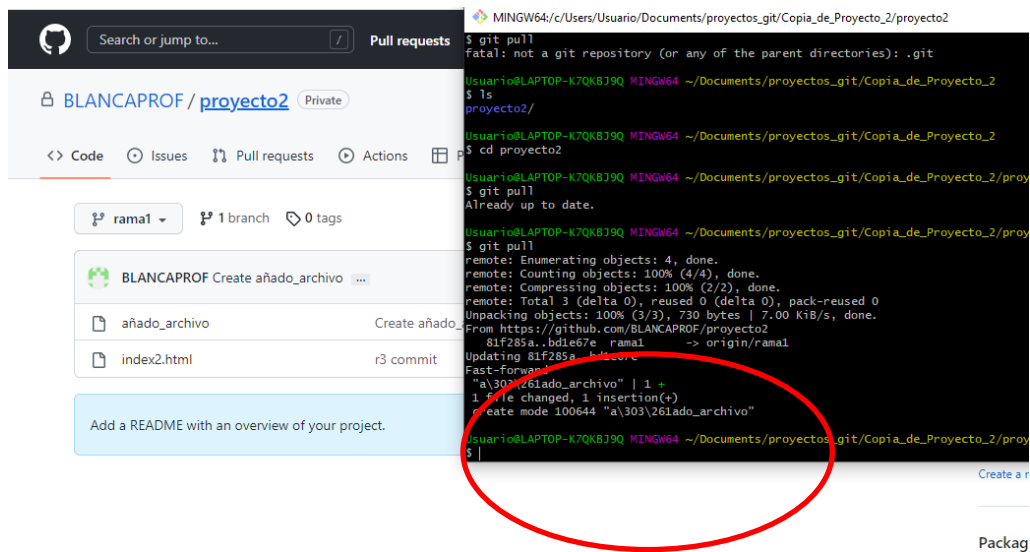
Para hacer el commit de este nuevo archivo, vamos a la parte inferior de la misma página y pulsamos el botón verde "Commit new file".

Aquí escribimos el mensaje corto del commit y opcionalmente, podemos extendernos más en el área de texto más grande.

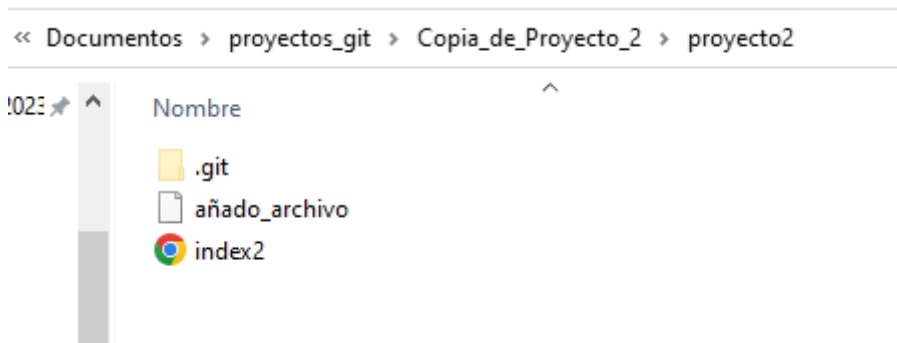


Estoy escribiendo este archivo en la rama main. Si deajo marcada la opción "Commit directly to the main branch", hará el commit en esta rama, en cambio, si queremos que se cree en otra rama, lo podemos indicar seleccionando la opción "Create a new branch for this commit and start a pull request."

Ahora, en tu repositorio local, prueba el git pull:

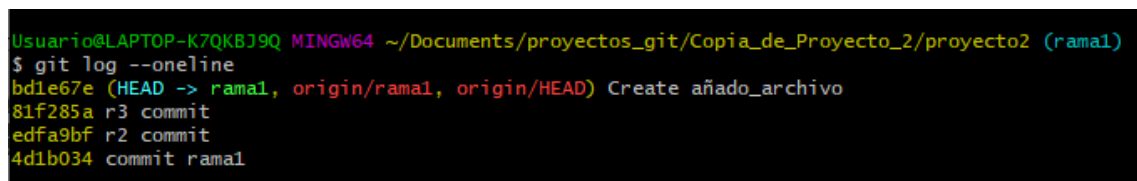


Se acaban de descargar todos los cambios. En la carpeta aparece el nuevo archivo:



No solo se ha descargado el archivo, si no que todo el contenido de la carpeta .git con sus commits. Lo podemos ver con un git log:

Aquí puedes ver localmente el commit que he hecho antes en GitHub.



Ahora, voy a crear un archivo en el repositorio local. Además, haré el commit en una nueva rama que no existe en el repositorio de GitHub.

Creo la rama nueva, me muevo a ella, y hago el commit.

```
git checkout -b rama_pruebas
nano prueba1
git add --all
git commit -m "Se ha añadido un archivo de pruebas"
```

```

Usuario@LAPTOP-K7QKB39Q MINGW64 ~/Documents/proyectos_git/Copia_de_Proyecto_2/proyecto2 (rama1)
$ git checkout -b rama_pruebas
Switched to a new branch 'rama_pruebas'

Usuario@LAPTOP-K7QKB39Q MINGW64 ~/Documents/proyectos_git/Copia_de_Proyecto_2/proyecto2 (rama_pruebas)
$ nano prueba1

Usuario@LAPTOP-K7QKB39Q MINGW64 ~/Documents/proyectos_git/Copia_de_Proyecto_2/proyecto2 (rama_pruebas)
$ git status
On branch rama_pruebas
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        prueba1

nothing added to commit but untracked files present (use "git add" to track)

Usuario@LAPTOP-K7QKB39Q MINGW64 ~/Documents/proyectos_git/Copia_de_Proyecto_2/proyecto2 (rama_pruebas)
$ git add --all
warning: LF will be replaced by CRLF in prueba1.
The file will have its original line endings in your working directory

Usuario@LAPTOP-K7QKB39Q MINGW64 ~/Documents/proyectos_git/Copia_de_Proyecto_2/proyecto2 (rama_pruebas)
$ git status
On branch rama_pruebas
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   prueba1

Usuario@LAPTOP-K7QKB39Q MINGW64 ~/Documents/proyectos_git/Copia_de_Proyecto_2/proyecto2 (rama_pruebas)
$ git commit -m "se ha a adido un archivo de pruebas"
[rama_pruebas 0b43dec] se ha a adido un archivo de pruebas
1 file changed, 1 insertion(+)
create mode 100644 prueba1

```

Ahora, para subir esta nueva rama con sus archivos, solo tienes que tener el repositorio actualizado como te acabo de ense ar e introducir el siguiente comando:

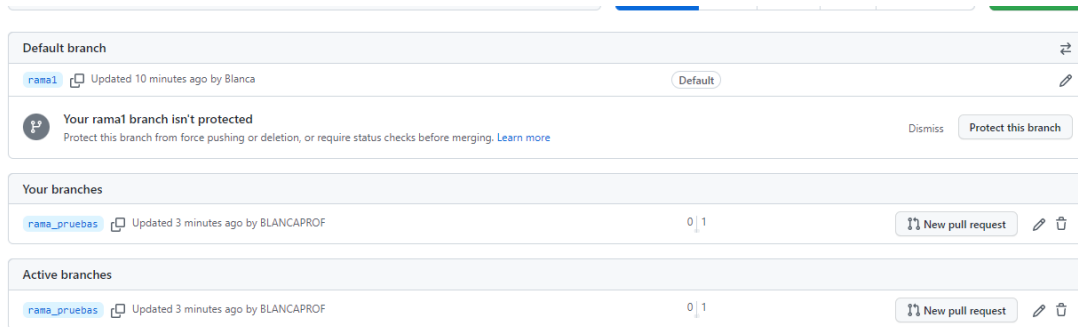
git push origin rama_pruebas

```

Usuario@LAPTOP-K7QKB39Q MINGW64 ~/Documents/proyectos_git/Copia_de_Proyecto_2/proyecto2 (rama_pruebas)
$ git branch
      rama1
* rama_pruebas

Usuario@LAPTOP-K7QKB39Q MINGW64 ~/Documents/proyectos_git/Copia_de_Proyecto_2/proyecto2 (rama_pruebas)
$ git push origin rama_pruebas
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 368 bytes | 368.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'rama_pruebas' on GitHub by visiting:
remote:   https://github.com/BLANCAPROF/proyecto2/pull/new/rama_pruebas
remote:
To https://github.com/BLANCAPROF/proyecto2.git
 * [new branch]      rama_pruebas -> rama_pruebas

```



Por cierto, si quieres subir todas las ramas, solo quita el nombre de la rama y utiliza la opción **--all**:

git push origin --all

```
Usuario@LAPTOP-K7QKBJ9Q MINGW64 ~/Documents/proyectos_git/Copia_de_Proyecto_2/proyecto2 (rama_pruebas)
$ git checkout rama1
Switched to branch 'rama1'
Your branch is up to date with 'origin/rama1'.

Usuario@LAPTOP-K7QKBJ9Q MINGW64 ~/Documents/proyectos_git/Copia_de_Proyecto_2/proyecto2 (rama1)
$ git merge rama_pruebas
Updating bd1e67e..0b43dec
Fast-forward
 prueba1 | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 prueba1

Usuario@LAPTOP-K7QKBJ9Q MINGW64 ~/Documents/proyectos_git/Copia_de_Proyecto_2/proyecto2 (rama1)
$ ls
añado_archivo  index2.html  prueba1

Usuario@LAPTOP-K7QKBJ9Q MINGW64 ~/Documents/proyectos_git/Copia_de_Proyecto_2/proyecto2 (rama1)
$ git pull
Already up to date.

Usuario@LAPTOP-K7QKBJ9Q MINGW64 ~/Documents/proyectos_git/Copia_de_Proyecto_2/proyecto2 (rama1)
$ git push origin --all
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/BLANCAPROF/proyecto2.git
 bd1e67e..0b43dec  rama1 -> rama1

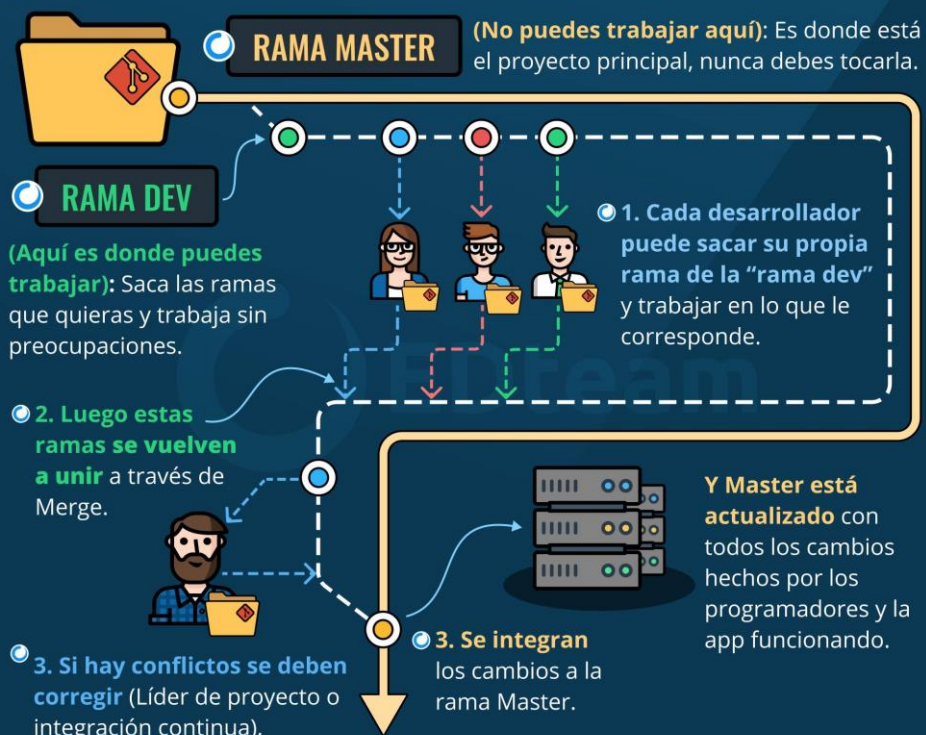
Usuario@LAPTOP-K7QKBJ9Q MINGW64 ~/Documents/proyectos_git/Copia_de_Proyecto_2/proyecto2 (rama1)
$ |
```

¿Cómo podemos editar archivos ya subidos en GitHub?

Esto como un pequeño extra. Para editar un archivo directamente desde GitHub, solo tienes que hacer un click sobre él y te abrirá el visor de código. Para acceder al editor, haz click en el icono del lápiz señalado en la imagen:

Realiza los cambios que quieras y haces el commit .

¿CÓMO TRABAJAR EN EQUIPO CON GIT?



Aprende cómo usan Git los equipos de desarrollo de software en:

`> ed.team/cursos/git-workflow`

