

Git

Git: Version control system(VCS), sistemas de control de versiones.

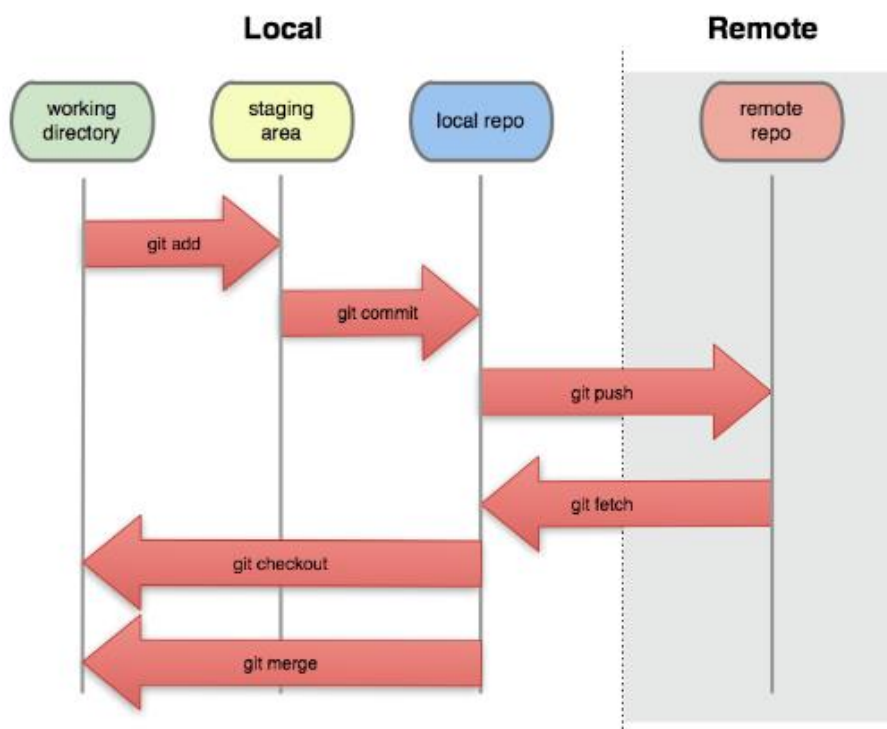
Es un software de control de versiones, repositorio oculto.

Características:

- Coordina trabajo entre múltiples desarrolladores
- Definir quién y cuándo se pueden hacer cambios
- Se puede revertir los cambios en el tiempo, otras versiones
- Repositorio local y remoto

Bluuweb-Git: <https://bluuweb.github.io/tutorial-github/guia/>

Flujo de trabajo de GIT

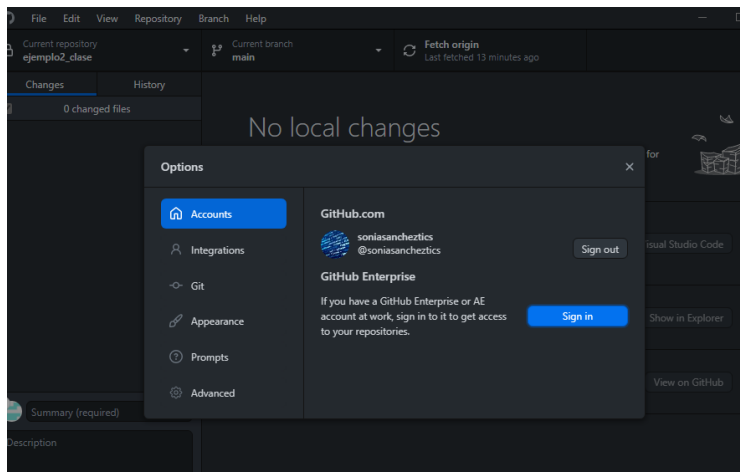


GitHub Desktop

Tener una cuenta en github.com

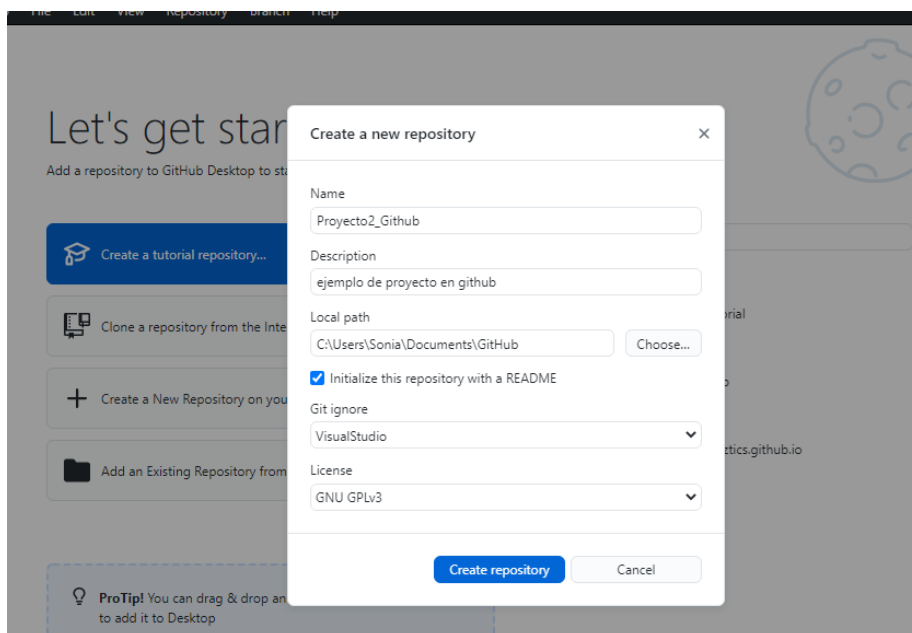
Instalar la aplicación: <https://desktop.github.com/>

Enlazar en file/options/sign in con nuestro github.com.

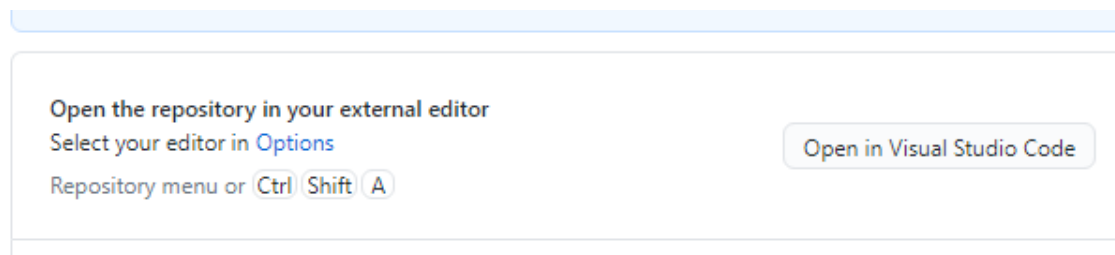


Veamos las diferentes opciones posibles desde el desktop:

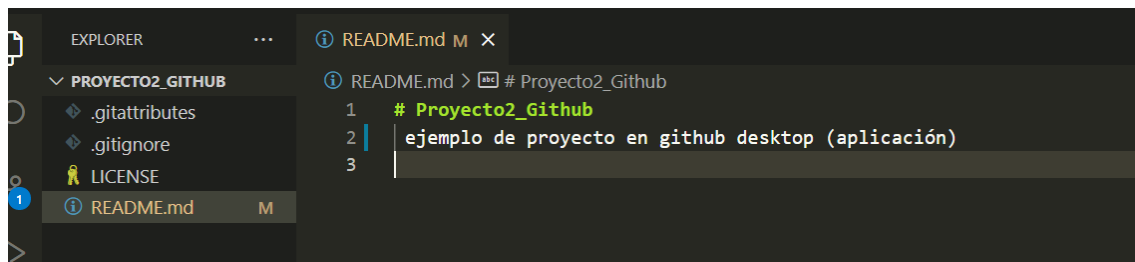
1- Crear nuevo repositorio



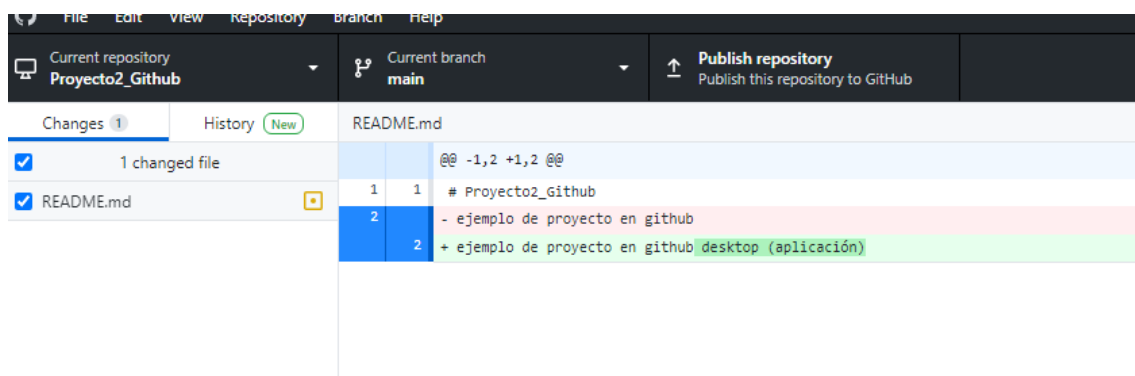
Se puede escoger editor (Options):



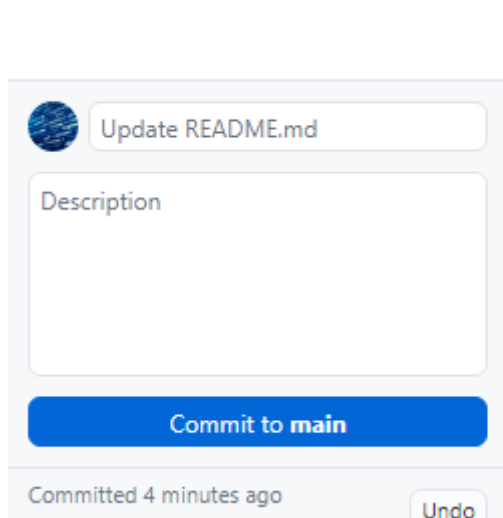
Hacemos cambios en el readme (archivo que describe lo que estamos realizando en el repositorio), por ejemplo :



Aparecerán los cambios:



Podemos hacer commit añadiendo comentarios en:



O sin comentarios, si no los necesita.

Y publicamos:

Publish your repository to GitHub

This repository is currently only available on your local machine. By publishing it on GitHub you can share it, and collaborate with others.

Always available in the toolbar for local repositories or **Ctrl** **P**

Publish repository

Open the repository in your external editor

Select your editor in [Options](#)

Repository menu or **Ctrl** **Shift** **A**

Open in Visual Studio Code

View the files of your repository in Explorer

Repository menu or **Ctrl** **Shift** **F**

Show in Explorer

Podemos poner otro nombre al repositorio ya en el github.com y hacerlo o no público (desactivar keep this code private):

Publish repository

GitHub.com | GitHub Enterprise

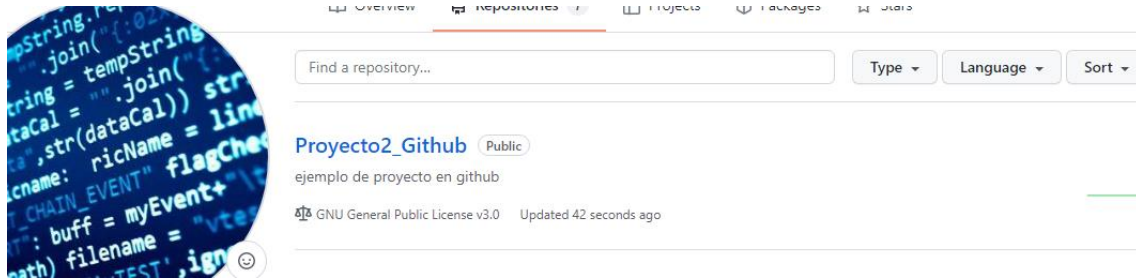
Name
Proyecto2_Github

Description
ejemplo de proyecto en github

☐ Keep this code private

Publish repository | Cancel

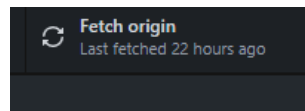
Se puede ver en github.com como ya se ha publicado:



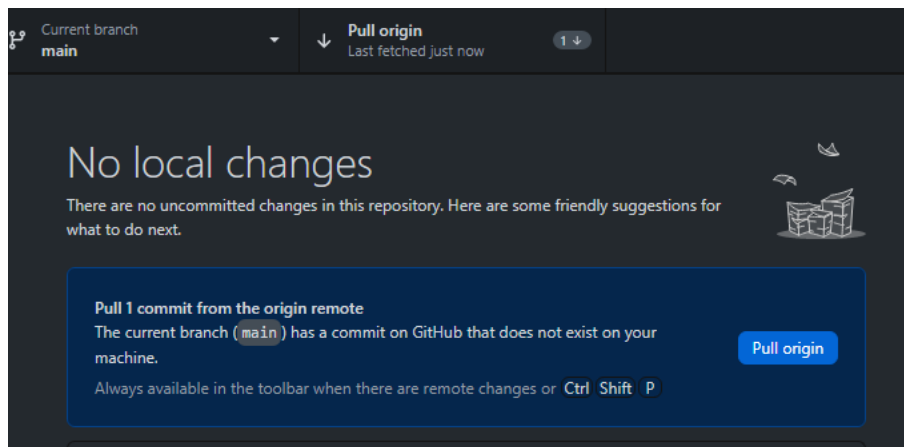
También desde el desktop desde Repository/ View on Github se puede ver en github.com.

Se verá el repositorio subido desde el desktop.

Si se realizan cambios desde github.com en el readme por ejemplo, deberemos después desde el desktop comprobar si han ha habido cambios antes de seguir trabajando con el proyecto, se ha de hacer un fetch :



Si se han realizado cambios desde el último fetch nos pedirá hacer un pull :



Con pull origin se actualizarán nuestros archivos en local.

2- Clonar el repositorio:

Primero partimos de un nuevo repositorio que creamos desde github.com: 'Proyecto3_Github' por ejemplo.

Owner *

soniasancheztics ▾

/


Repository name *


Proyecto3_Github ✓

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

Description (optional)

Proyecto 3 desde github.com

☒  **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: VisualStudio ▾

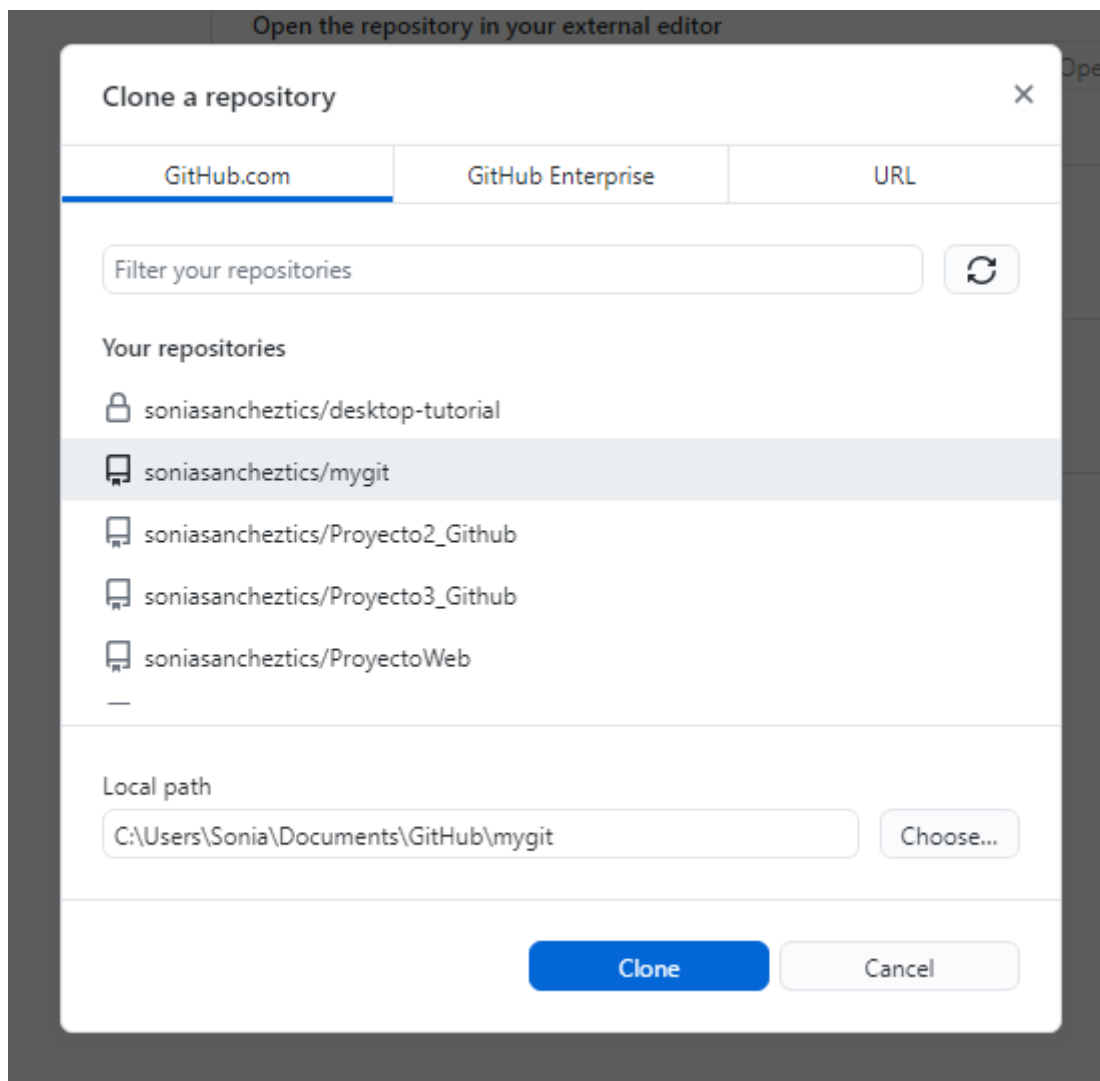
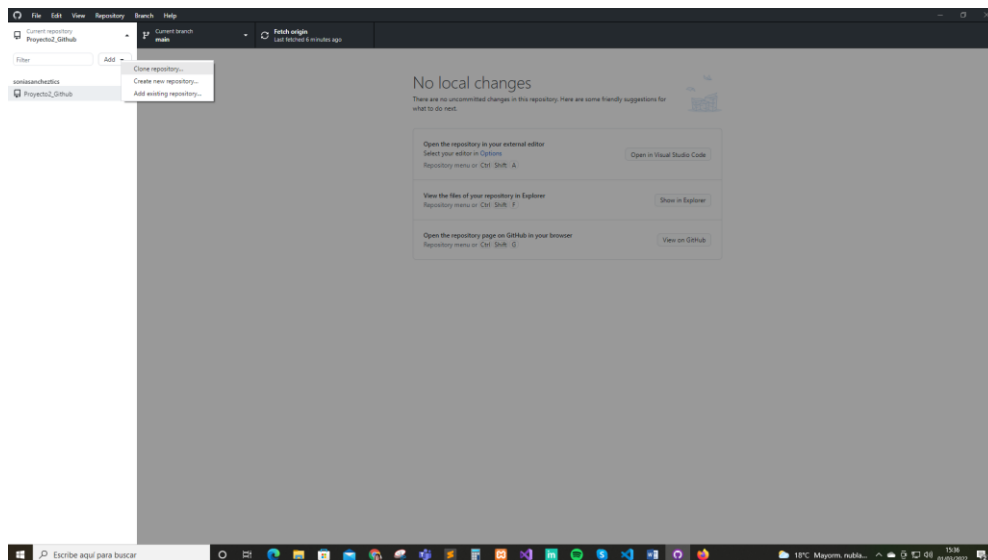
☒ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

License: GNU General Public ... ▾

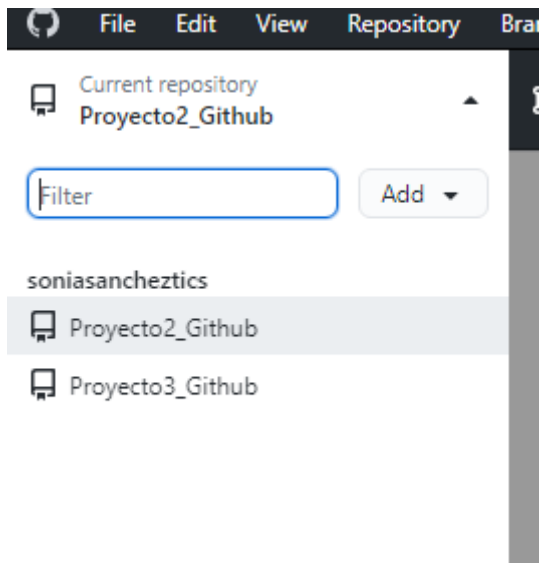
This will set  **main** as the default branch. Change the default name in your [settings](#).

Create repository

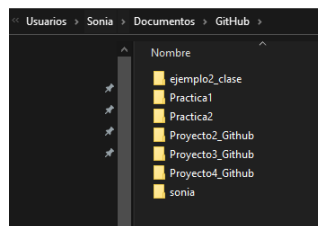
Y ya desde el desktop:



Aparecerán:



En local se pueden ver los repositorios:



Abrimos con visual studio code desde el github de desktop, modificamos archivos y hacemos **commit** y **push** para publicar.

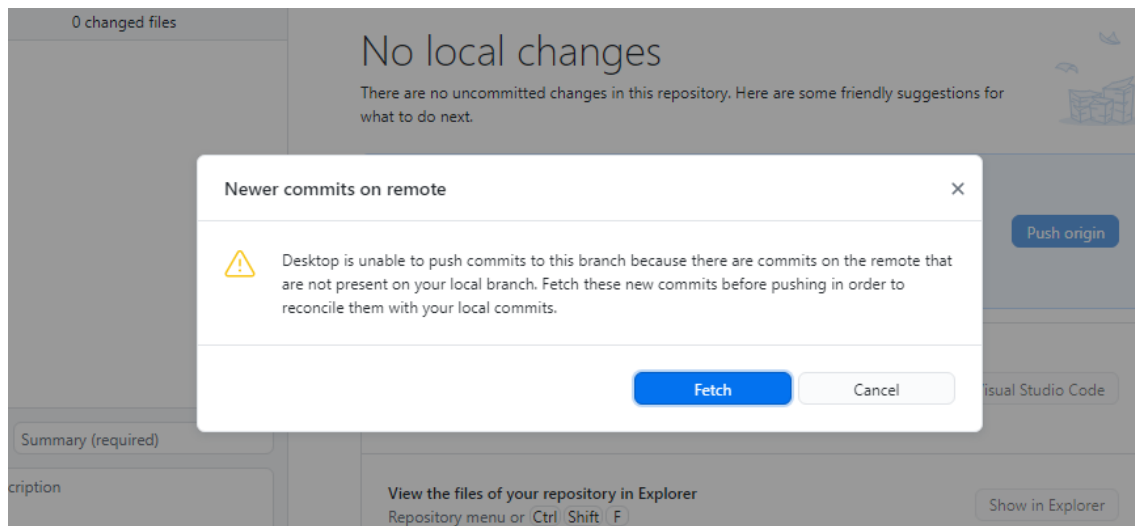
Aparecerán ya los cambios en github.com.

Lo mismo al revés, hacemos cambios en github.com (commit) y desde el desktop hacemos **fetch** y **pull** para sincronizar los cambios, desde nuestro vsc ya aparecerán los cambios.

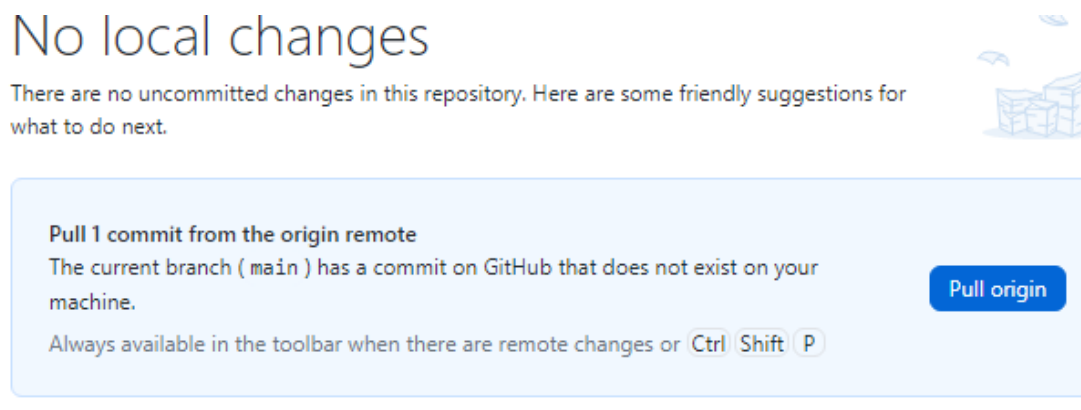
Vamos a ver qué sucede cuando se están realizando cambios tanto en local como en el servidor al mismo tiempo.

Hacemos cambios en el index.html desde github.com, guardamos y después hacemos cambios desde github-desktop.

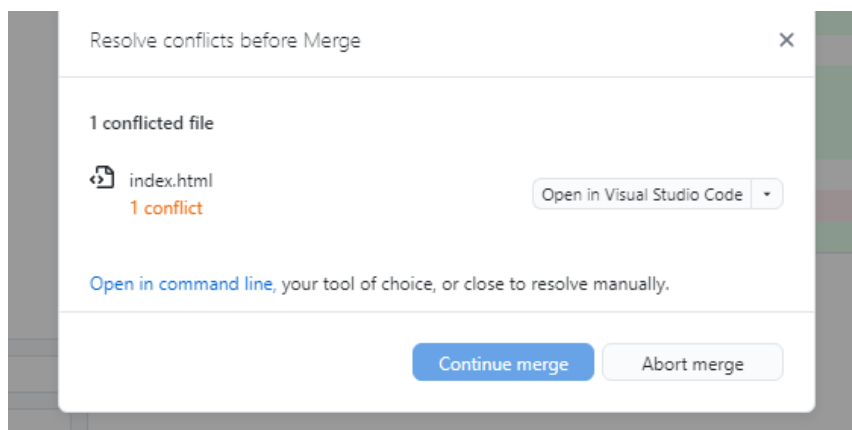
Al hacer commit y después push desde el desktop nos aparecerá:



Hacemos fetch y aparecerá:



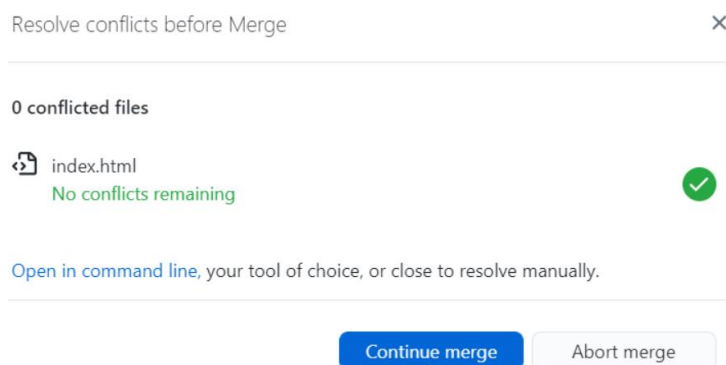
Hacemos pull :



Abrimos con Visual Studio:

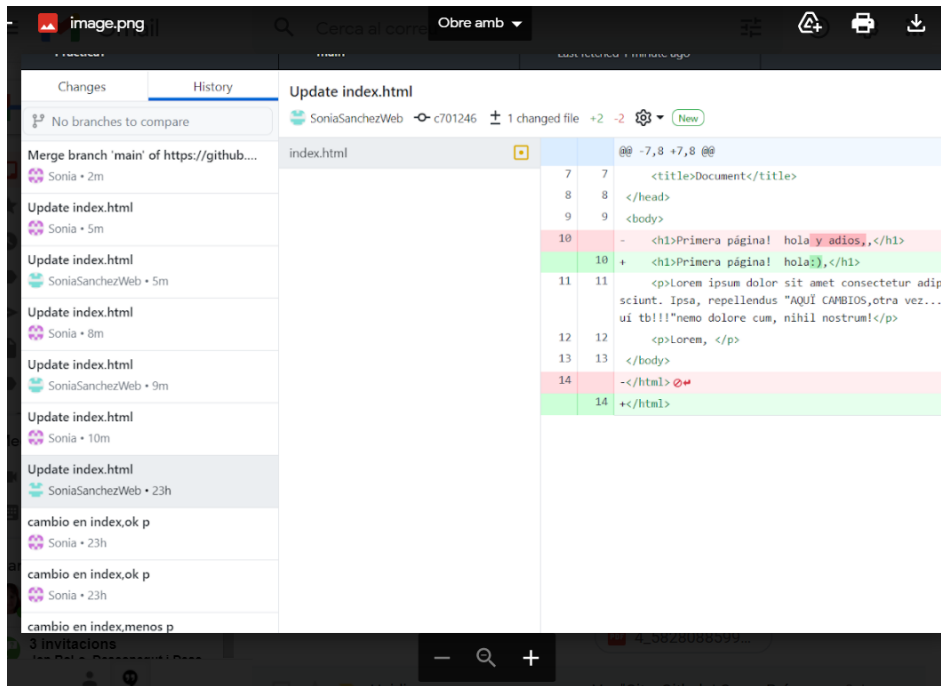
```
index.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta http-equiv="X-UA-Compatible" content="IE=edge">
6    <meta name="viewport" content="width=device-width, initial-scale=1.0">
7    <title>Document</title>
8  </head>
9  <body>
10  Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
11  <<<<<< HEAD (Current Change)
12  <h1>hola adios..</h1>
13  =====
14  <h1>hola holita.</h1>
15  >>>>>> 673d8fdcf319e2829c11e8c911454152e710f80 (Incoming Change)
16  </body>
17  </html>
```

Escogemos la opción deseada:



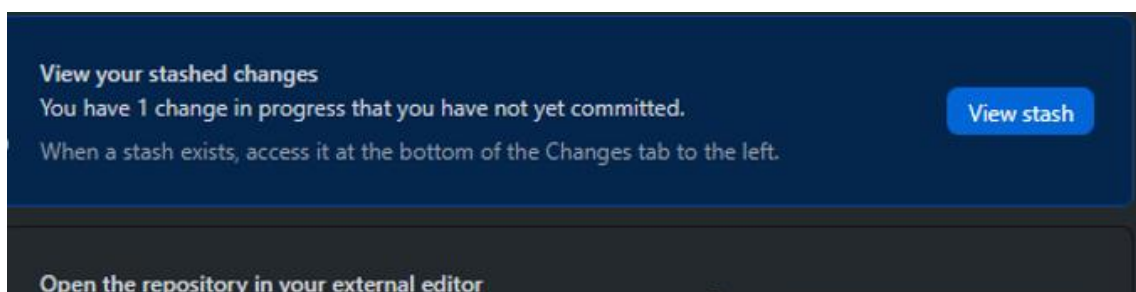
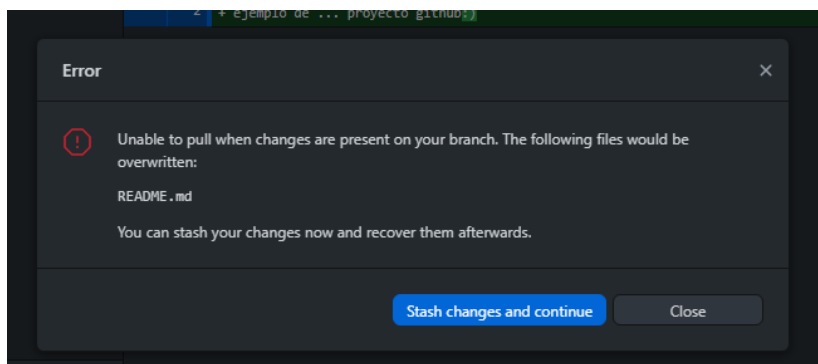
hacemos push y ya aparecerá en github.com correctamente.

Ya en historia se verán las modificaciones y cómo se han integrado en la rama principal.

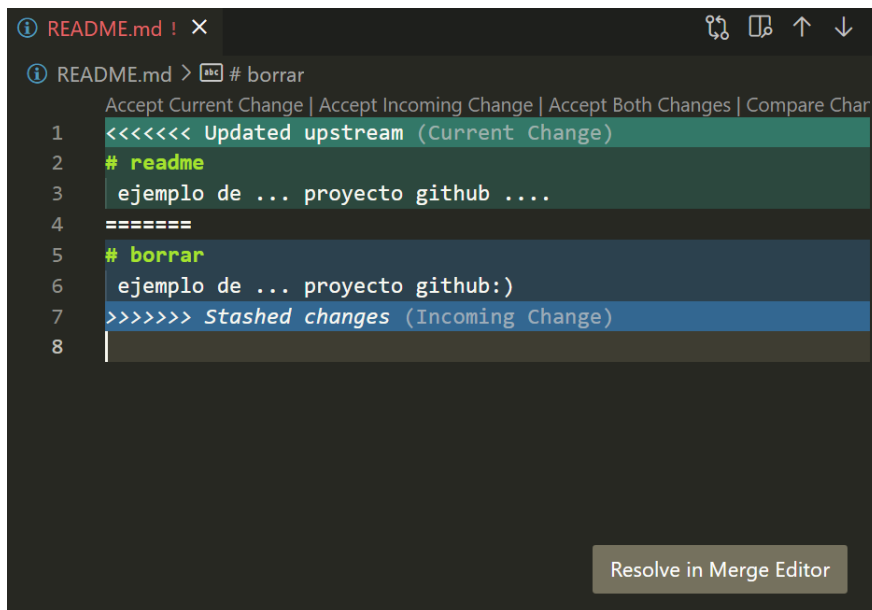


Primero siempre hacer fetch antes de empezar a trabajar desde el desktop.

Si hubiera sido al revés al intentar subir los archivos desde el desktop habiendo hecho cambios en local i en .com sucedería lo mismo al intentar hacer fetch y pull:



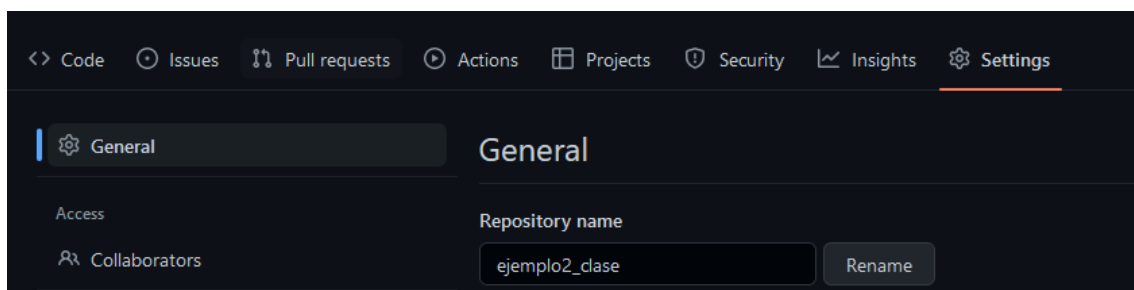
Si abrimos desde el VSC veremos las diferentes opciones para resolver el conflicto :



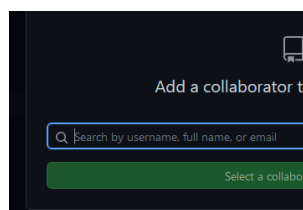
Trabajar en equipo y con ramas:

Se puede añadir colaboradores a un repositorio.

Desde el propio repositorio en settings:

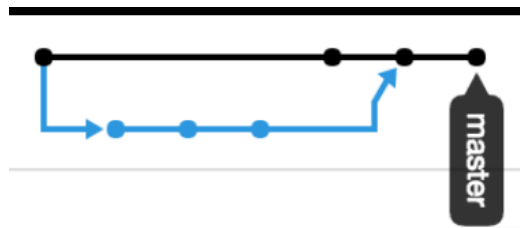


A partir del usuario de github.com se añaden como colaborador:



Se les envía una invitación para que puedan compartir el repositorio.

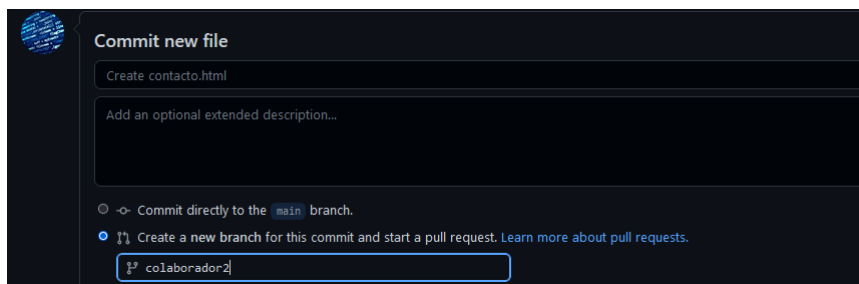
Una rama se representaría de la manera siguiente:



Un colaborador puede ser el encargado de estar haciendo una parte del proyecto, por ejemplo, es el encargado de hacer la página de contacto.

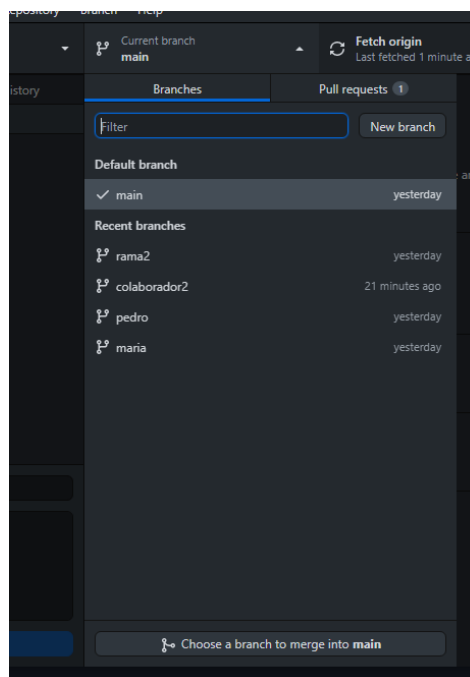
Una posible manera de crear una rama sería:

Al crear el archivo contacto.html:

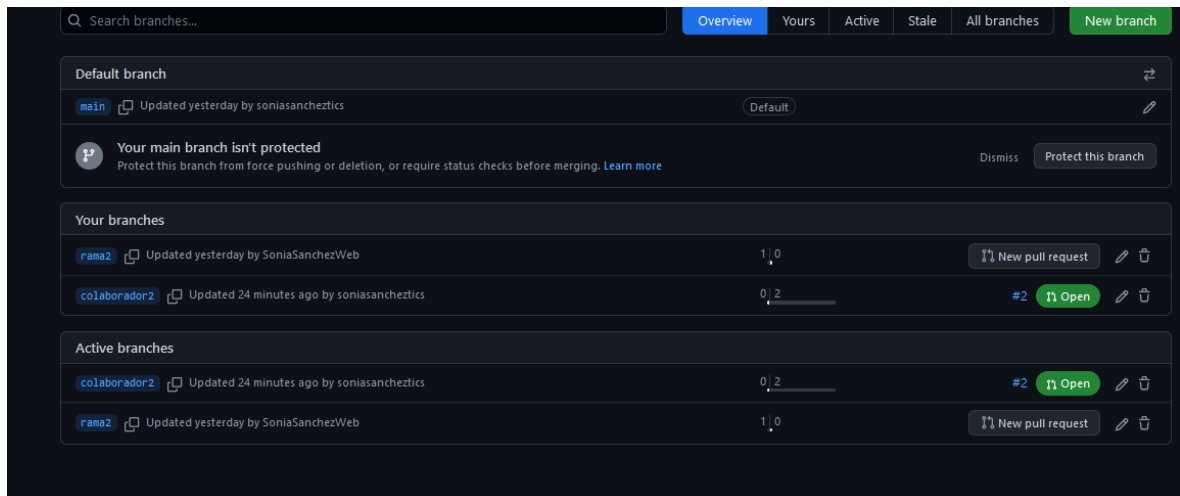


Se puede crear una nueva rama, desde esta se puede ir trabajando una parte del proyecto de manera paralela.

Cuando se quieran unir las ramas, desde el desktop debemos hacer merge hacia la principal:

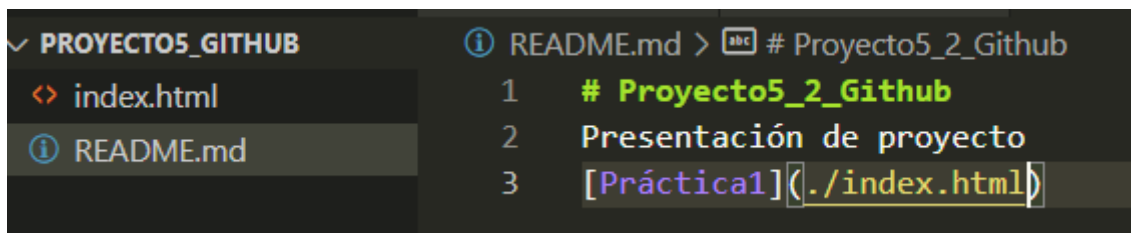


Y después se puede borrar esa rama desde el apartado de ramas del repositorio:

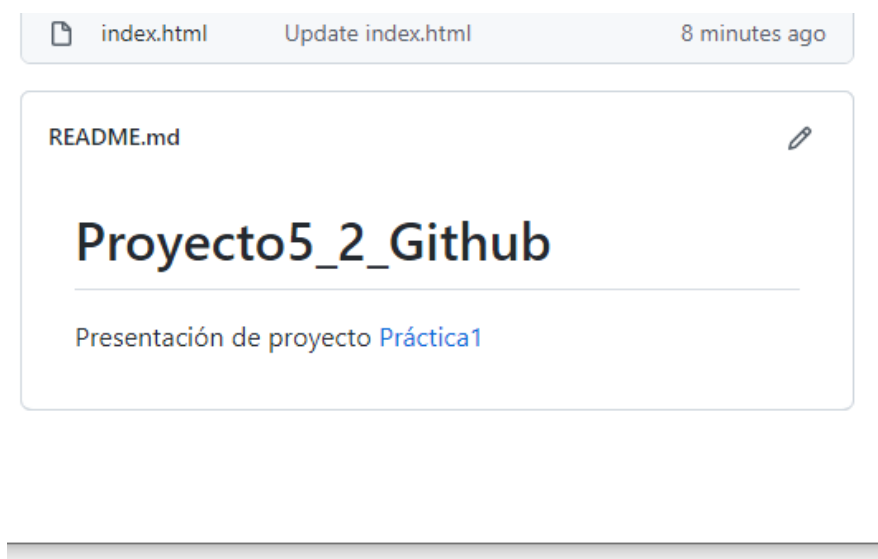


Vamos a **editar el archivo README** :

Vamos a crear enlaces para navegar por la carpeta, por ejemplo desde el vsc en local:



Desde github desktop commit y push y en github.com aparecerá:



El enlace nos abrirá:

1 contributor

12 lines (12 sloc) | 286 Bytes

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Document</title>
8  </head>
9  <body>
10     <h3>hola..adios..</h3>
11 </body>
12 </html>
```

El archivo readme es un archivo markdown, el cual convierte el texto a HTML con caracteres de formato.