

**Guía: Preparación de ambiente GitHub Desktop, conectando con la cuenta de GitHub.**

Contenido

[II. Introducción. 3](#_Toc92763063)

[A. Arquitectura de conexión aplicativos. 3](#_Toc92763064)

[B. Arquitectura de funcionamiento Desarrollador utilizando Git Bash 3](#_Toc92763065)

[C. Arquitectura de funcionamiento Desarrollador utilizando GitHub Desktop 3](#_Toc92763066)

[III. Configuración GitHub Desktop con GitHub. 4](#_Toc92763067)

[A. Descarga e instalación de GitHub Desktop 4](#_Toc92763068)

[B. Inicio de Sesión 6](#_Toc92763069)

[C. Uso GitHub Desktop 8](#_Toc92763070)

[D. Actualizaciones con Visual Studio 10](#_Toc92763071)

[E. Trabajar con Ramas. 14](#_Toc92763072)

[F. Unir ramas. 18](#_Toc92763073)

# Introducción.

En la presente guía, revisaremos los procesos para conectar las herramientas GitHub Desktop, con la nube de GitHub. En la guía pasada llamada “3.1.2 Guía Preparación de ambiente Git Bash, conectando con la cuenta de GitHub” ya revisamos la conexión con Git Bash con GitHub, por cual, en esta guía, revisaremos la otra sesión de conexión.

## Arquitectura de conexión aplicativos.

Diagrama

Descripción generada automáticamente con confianza media

## Arquitectura de funcionamiento Desarrollador utilizando Git Bash

Logotipo, nombre de la empresa

Descripción generada automáticamente

## Arquitectura de funcionamiento Desarrollador utilizando GitHub Desktop

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

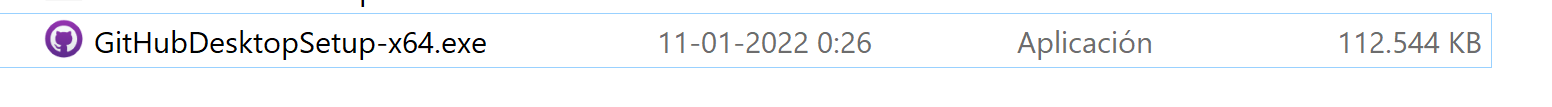
Descripción generada automáticamente

**Aclaración**: en desarrollador puedes trabajar con Git Bash y GitHub sin utilizar GitHub Desktop, o viceversa, esto es debido a que Git Bash y GitHub Desktop realizan el mismo proceso, la gran diferencia entre ambas, es que GitHub Desktop realiza los procesos por medio de una interfaz, y Git por medio de líneas de comandos

# Configuración GitHub Desktop con GitHub.

## Descarga e instalación de GitHub Desktop

1. Descargar el producto desde plataforma AVA, llamado 3.1.6 GitHub Desktop.7z, o desde internet, ingresando al siguiente link: <https://desktop.github.com/>
2. Descomprimir e iniciar el proceso de instalación.



1. Dar click en “**next**” y dejar preseleccionado todas las opciones que GitHub nos marca, estas opciones preseleccionadas, son las más adecuadas para un entorno de aprendizaje y primeros pasos.
2. Una vez finalizado el proceso, debería habilitarse la siguiente pantalla



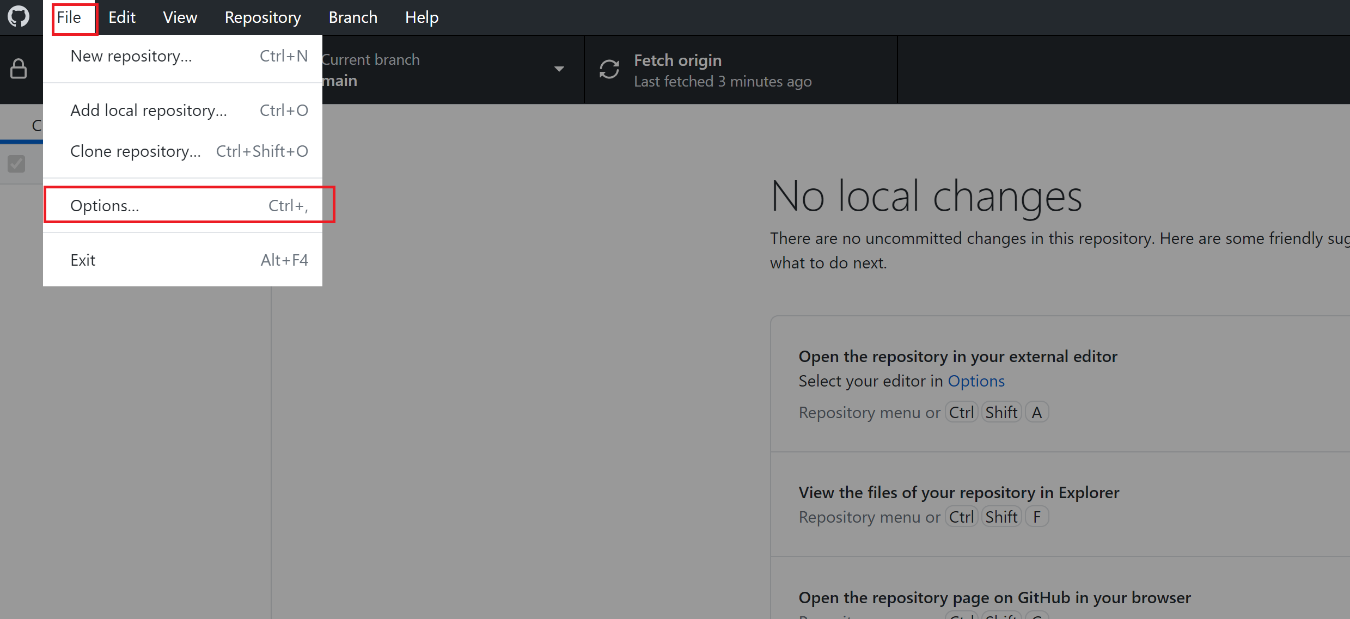
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

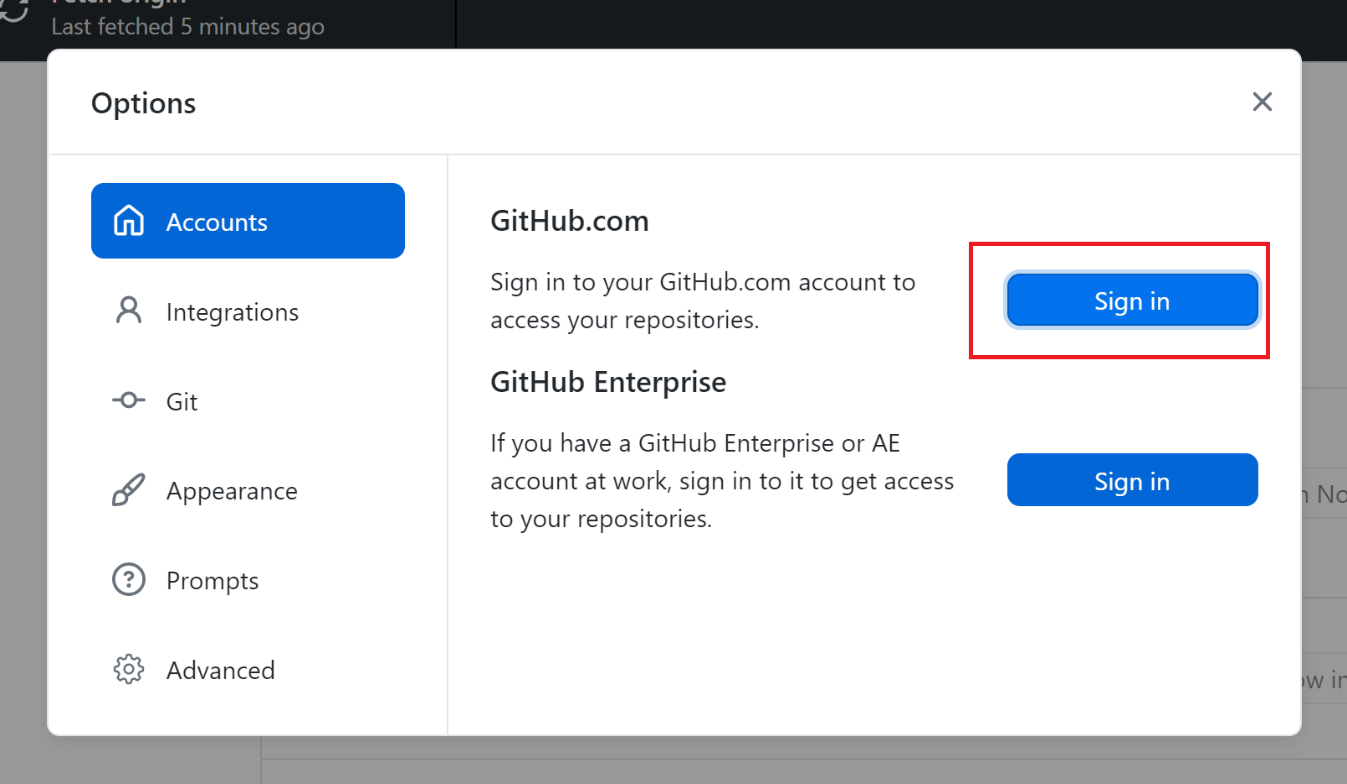
## Inicio de Sesión

En GitHub Desktop no deberemos incorporar una llave SSH, solo bastará con incorporar el inicio de sesión de GitHub.com en el aplicativo, en ese momento tendremos conectada nuestra cuenta.

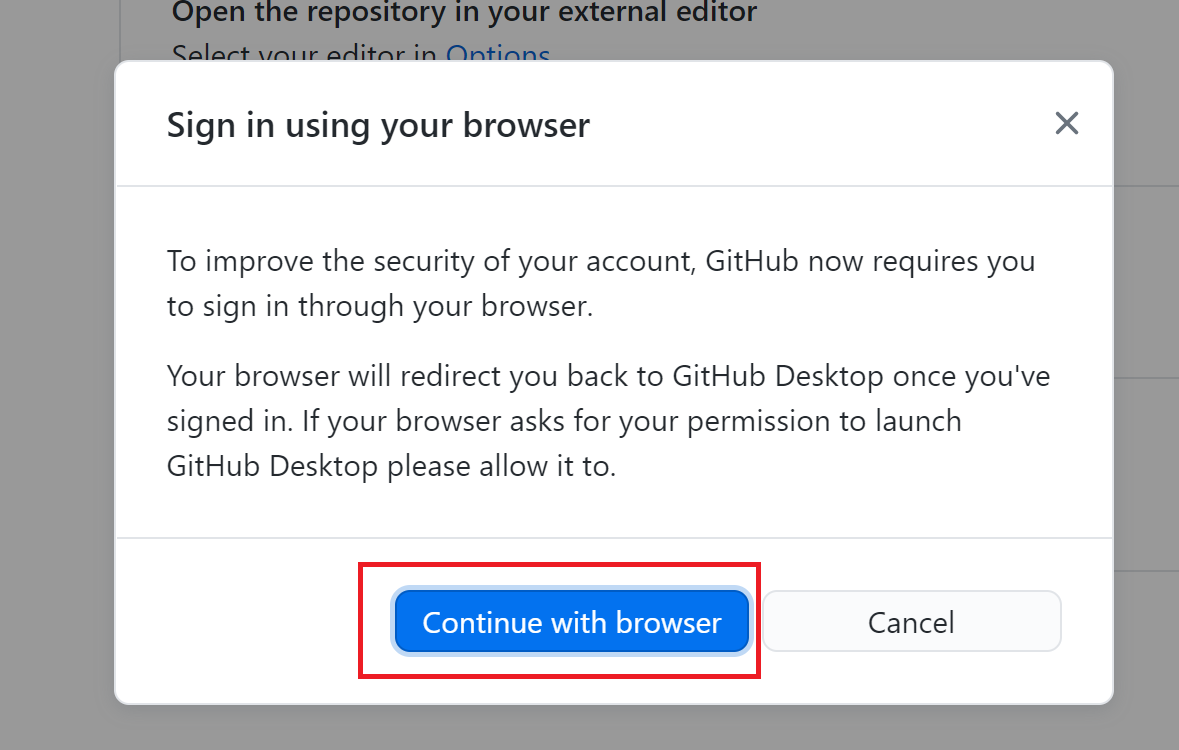
1. Dentro de GitHub Desktop, dirigirse a “**File**”, luego a “**Options**”



1. Presionar el botón “**Sign in**”



1. Solicitará GitHub Desktop acceso al browser para iniciar sesión en GitHub.

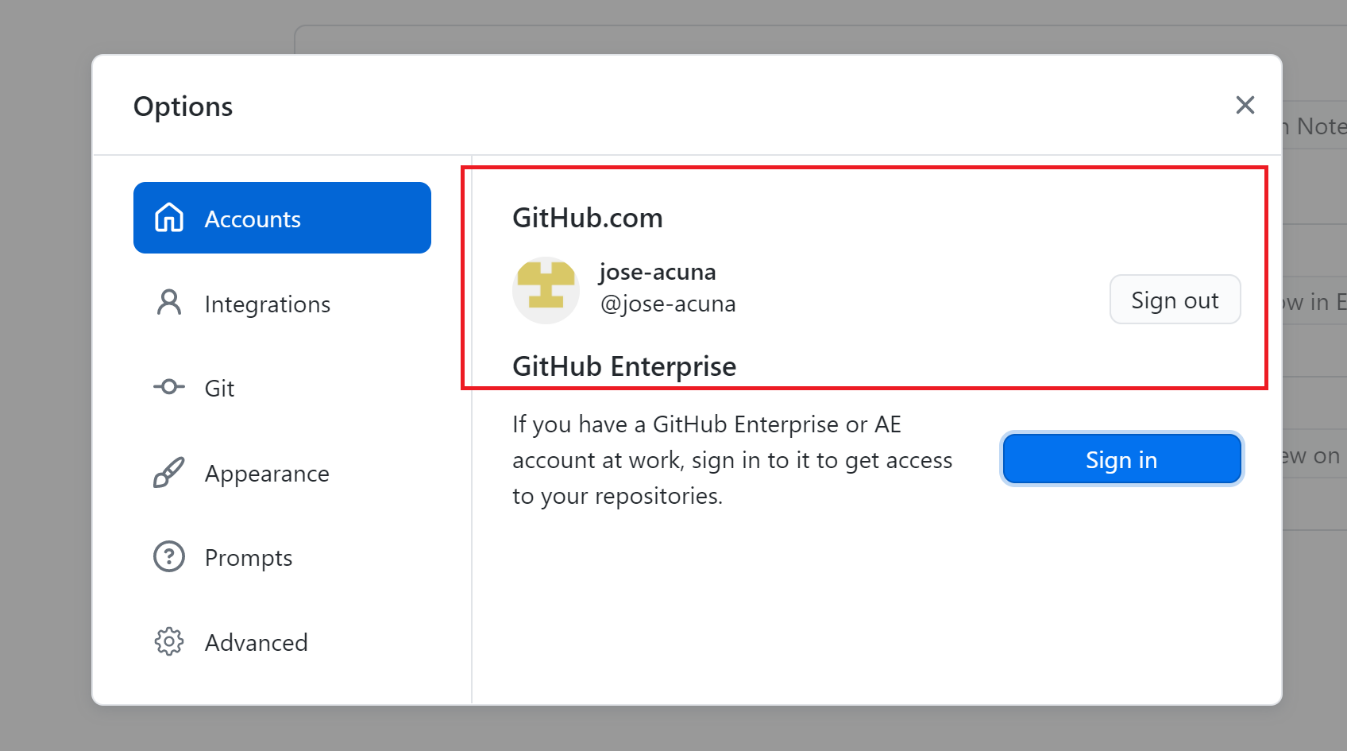


1. Iniciaremos sesión con nuestra cuenta GitHub creada al inicio del semestre.

Interfaz de usuario gráfica, Aplicación

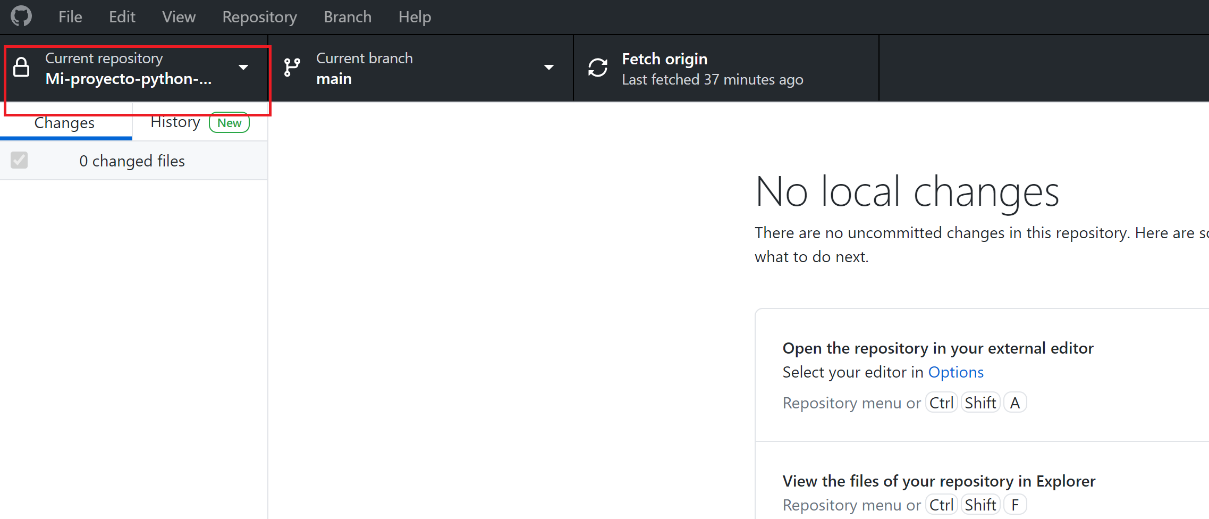
Descripción generada automáticamente

1. Para corroborar que el proceso se encuentra finalizado exitosamente, volveremos a “**Options**” y veremos que nuestra cuenta ya se encuentra asociada a GitHub Desktop.

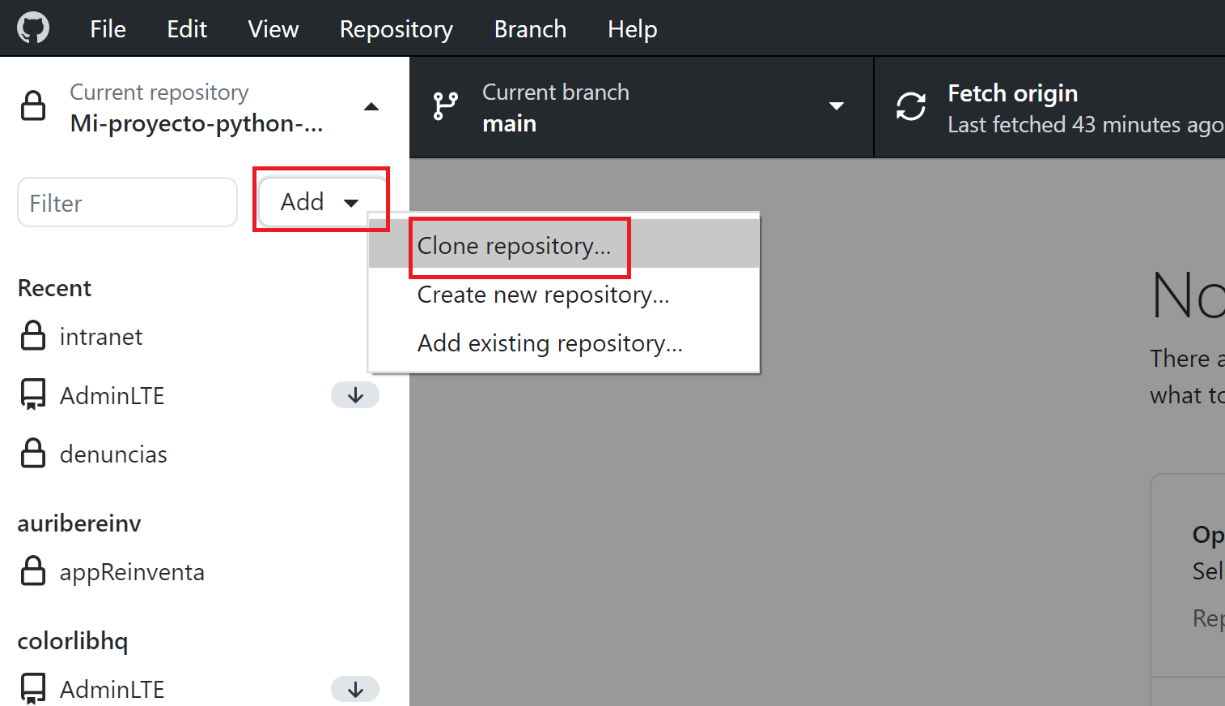


## Uso GitHub Desktop

1. Para descargar nuestros proyectos de GitHub.com, realizaremos el proceso de Clonar (descargar) los repositorios de la nube, a lo cual, presionaremos en el botón “**Current Repository**”

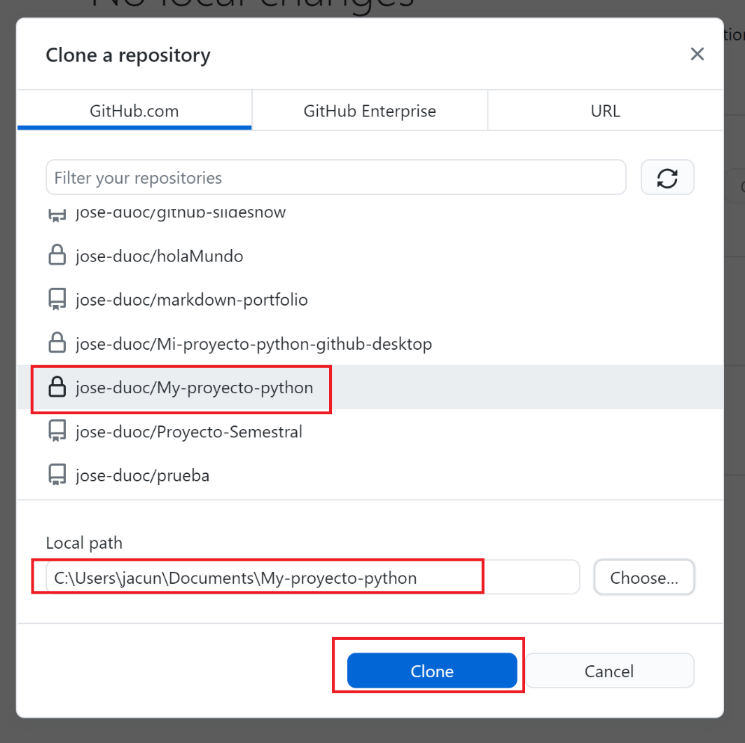


1. Luego presionaremos el botón “**Add**” y luego “**Clone repository**”



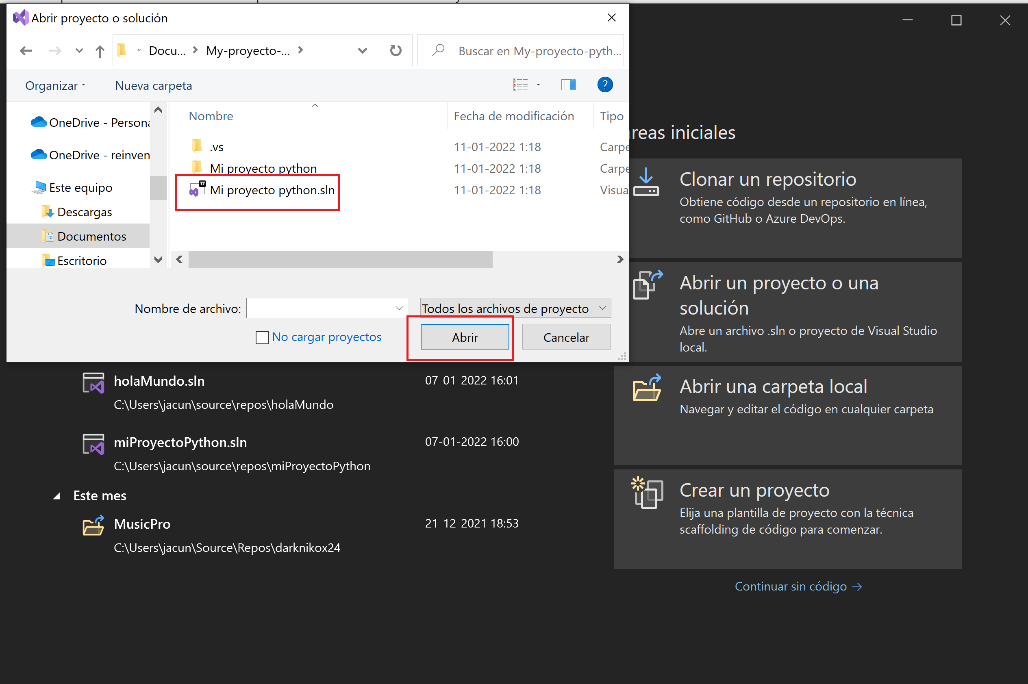
1. Dentro esta opción, seleccionaremos el proyecto que habíamos trabajado en clases anteriores, llamado “**My-proyecto-python**”.

**NOTA:** Es importante que el proyecto a descargar (clonar) se encuentra a una ubicación distinta al primer proyecto trabajado, con el fin de no provocar conflictos de archivos.

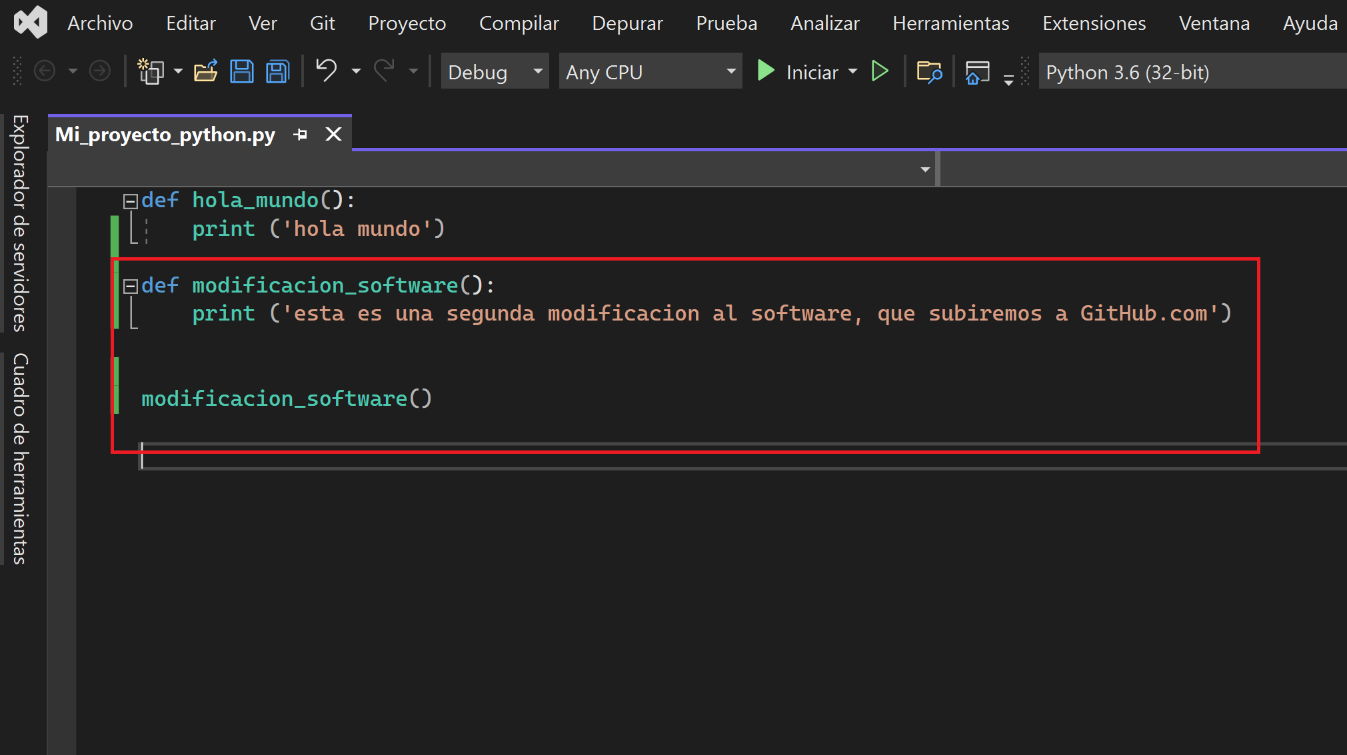


## Actualizaciones con Visual Studio

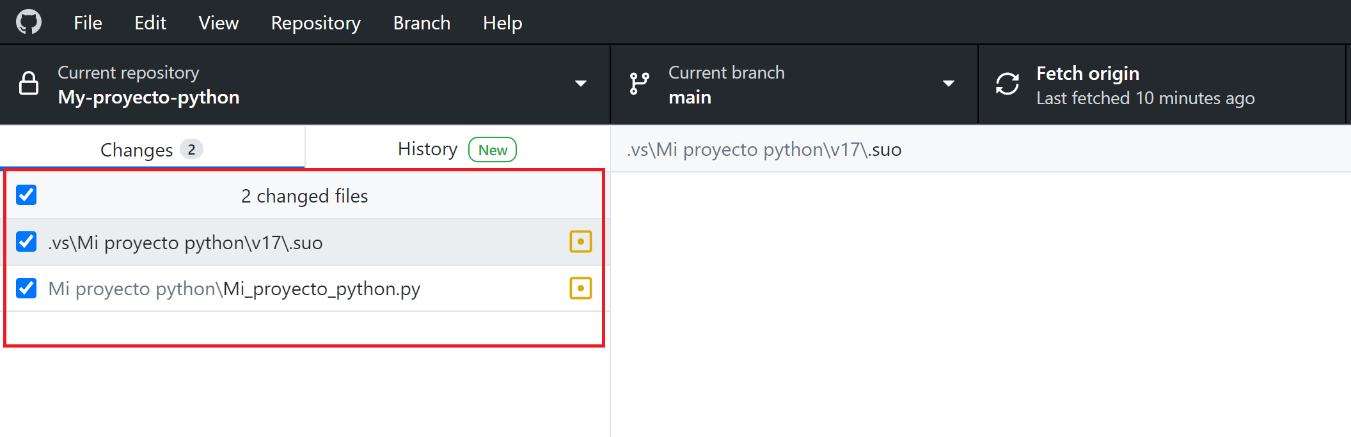
1. Ahora, importaremos el proyecto descargado desde GitHub.



1. Incorporamos un segundo método.

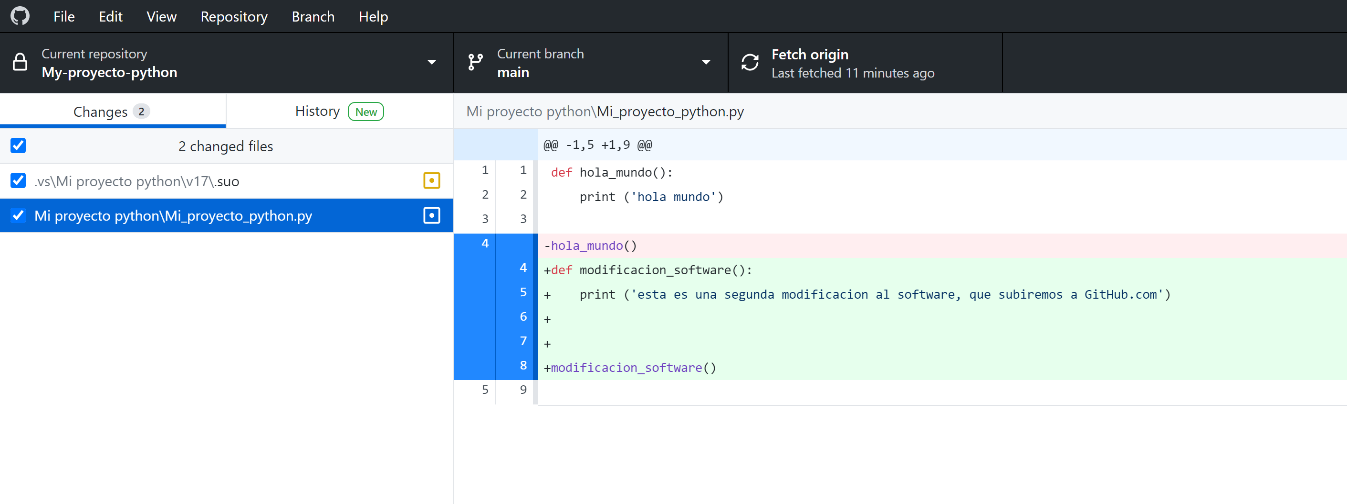


1. Una vez compilado el proyecto, cuando volvamos a GitHub Desktop, veremos que GitHub Desktop ha identificado cambios en dos archivos.

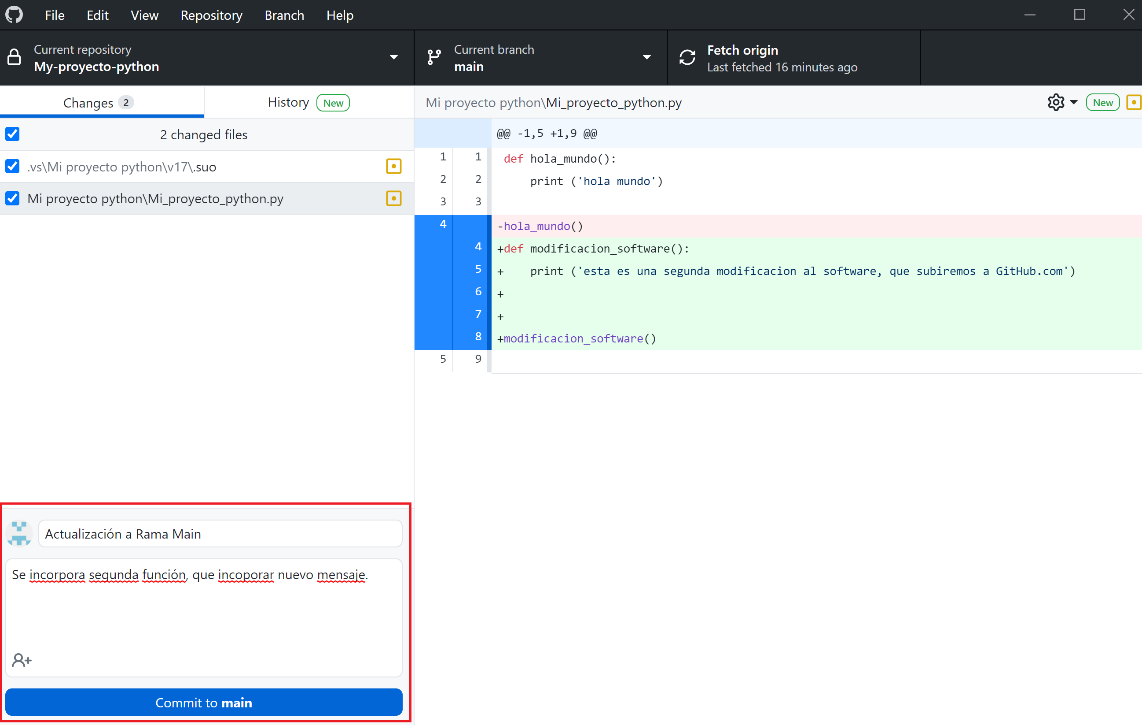


1. Al momento de pinchar en el archivo principal, GitHub Desktop nos indica lo siguiente

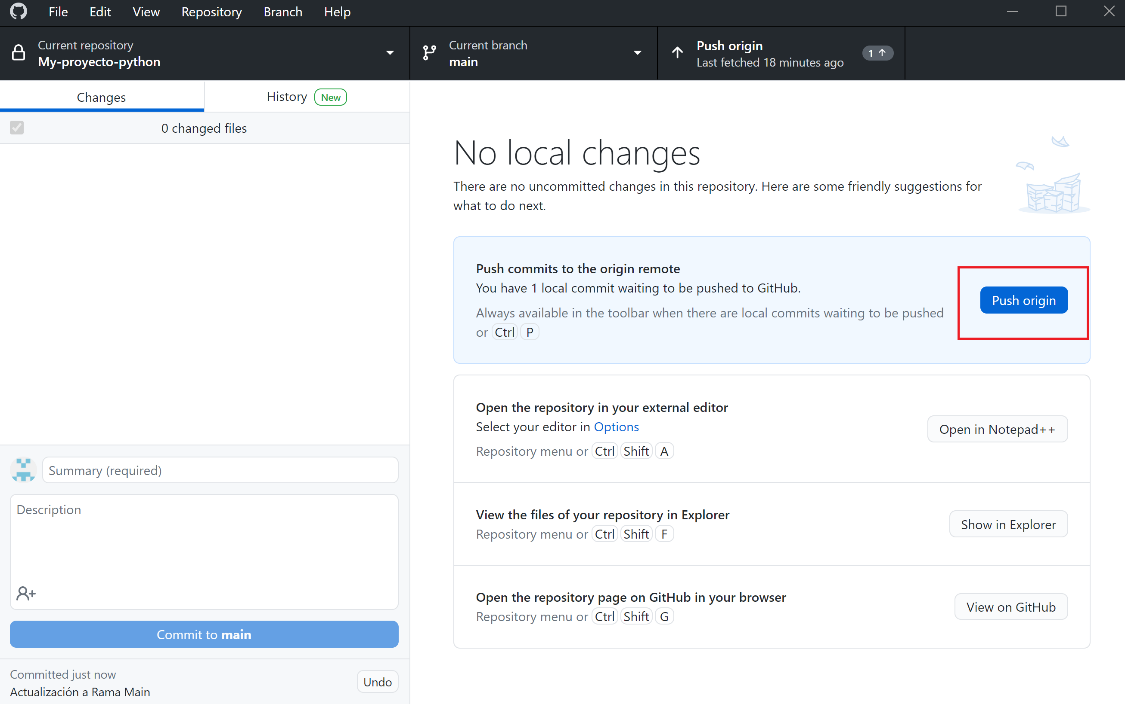
* De la línea 1 a la 3, no hay modificaciones, y por lo tanto se encuentra en blanco
* La antigua línea de código 4 la marca en fondo rojo junto con un símbolo “-”, eso significa que esa línea de código ha sido eliminada.
* De la nueva línea de código 4, a la línea 8, marca en fondo verde junto con un símbolo “+”, significa que a este archivo se está agregando nuevas sentencias.



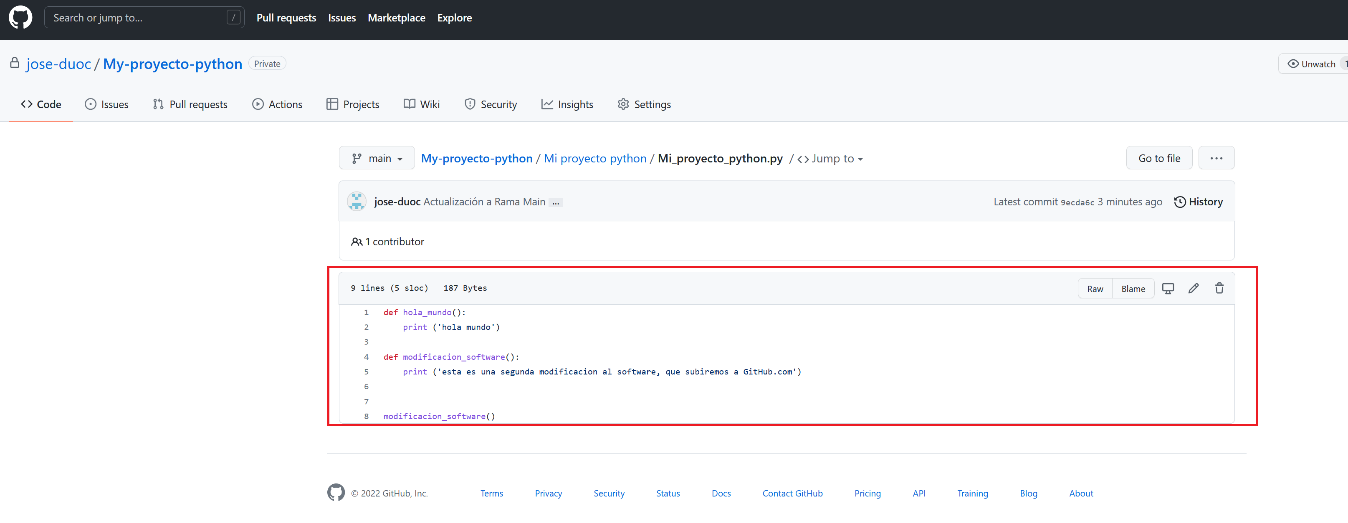
1. Incorporamos estas modificaciones a la rama principal “**Main**”, ingresando un nuevo comentario, una descripción, y luego presionando el botón “**Commit to main**”



1. Finalmente, para subir los cambios a GitHub.com, presionaremos en el botón “**Push Origin**”

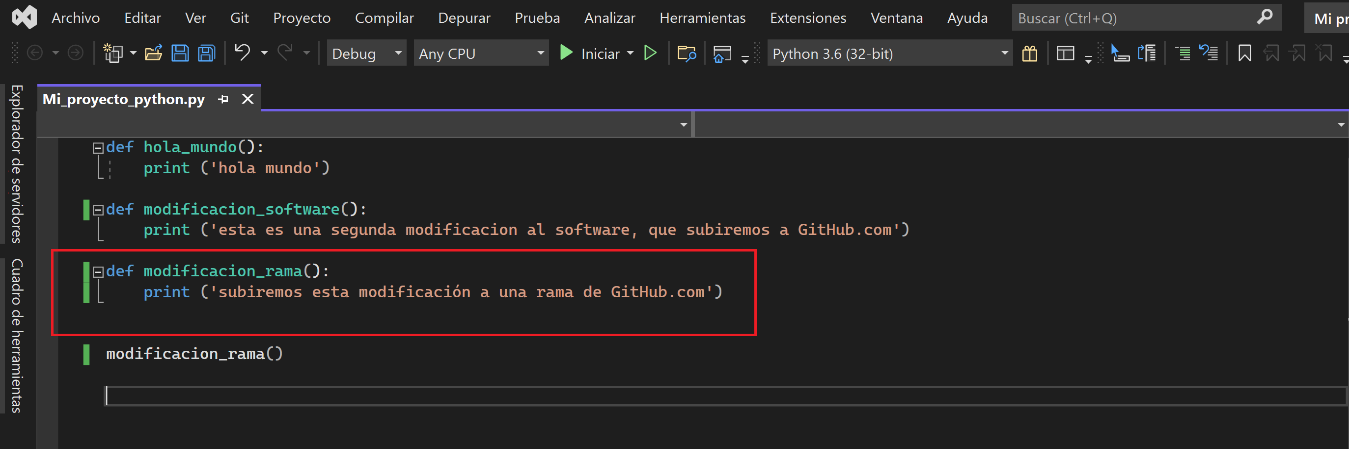


1. Si revisamos en GitHub.com, veremos como las actualizaciones realizadas están subidas.

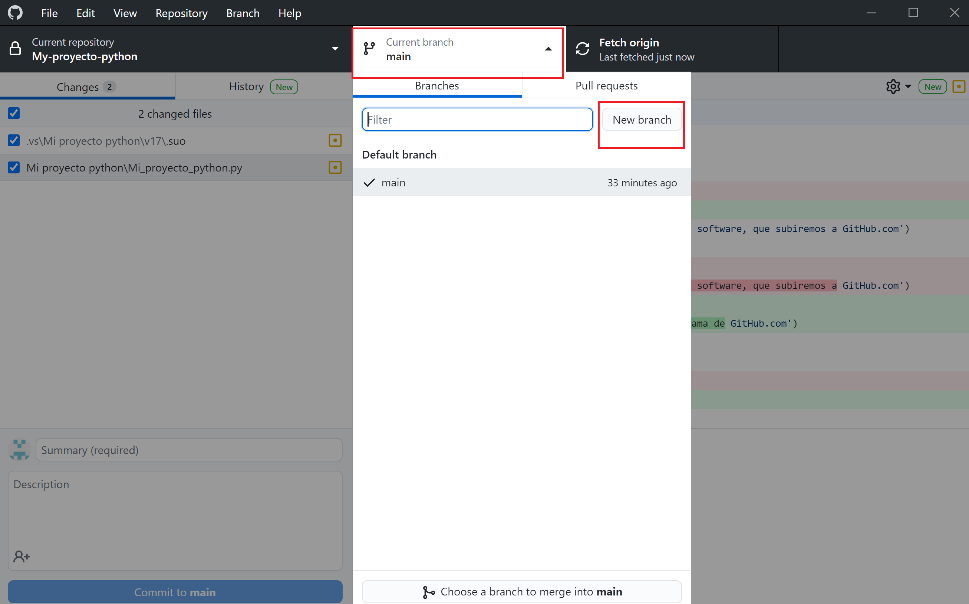


## Trabajar con Ramas.

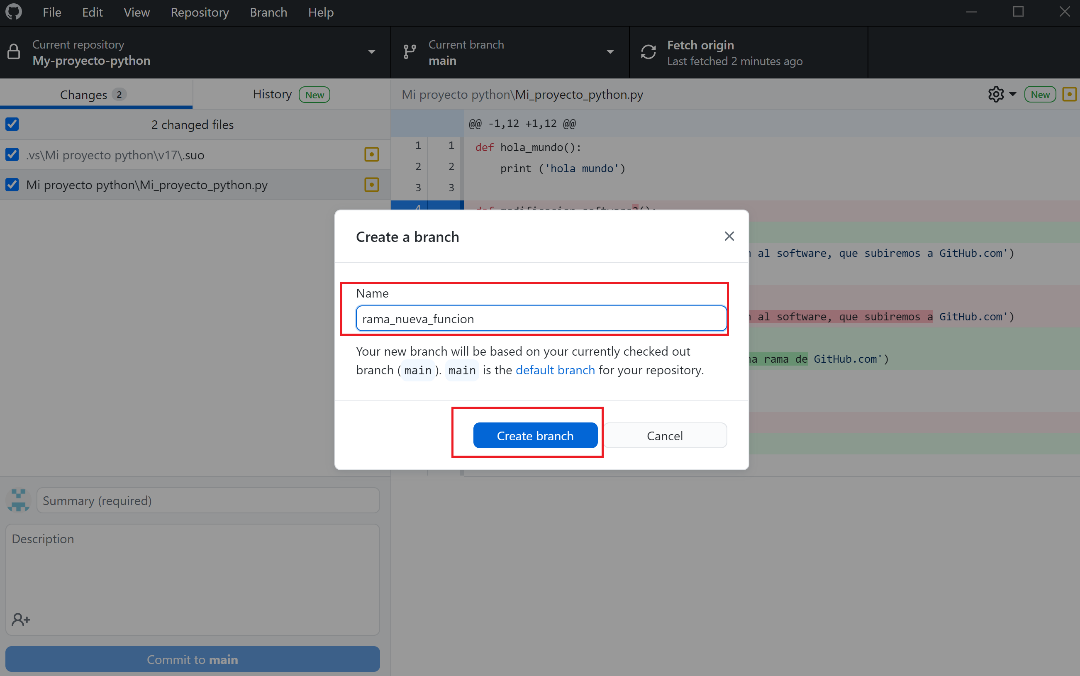
1. Para subir una modificación a una rama, deberemos realizar los siguiente procesos. Empezaremos por incorporar unas líneas a nuestro código.



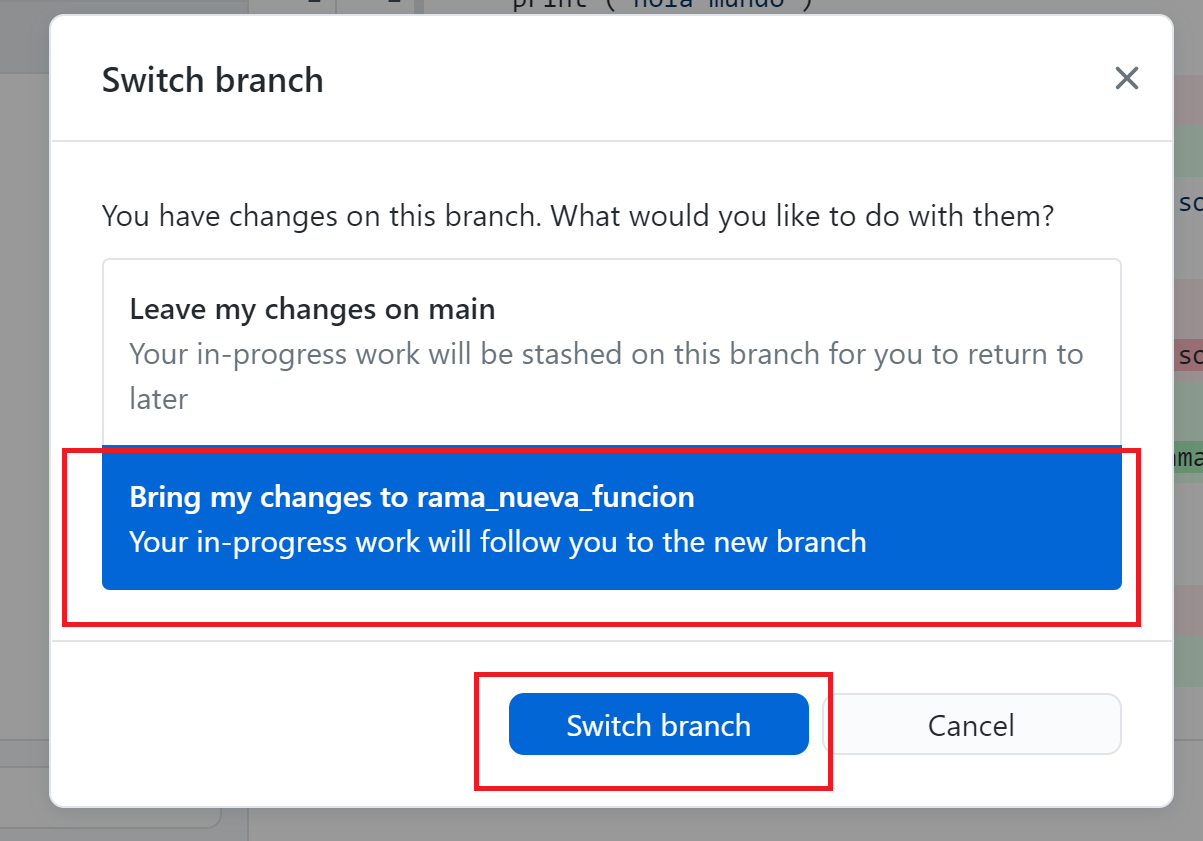
1. En GitHub Desktop, nos iremos a la opción “**Current branch**”, luego presionaremos “**New Branch**”



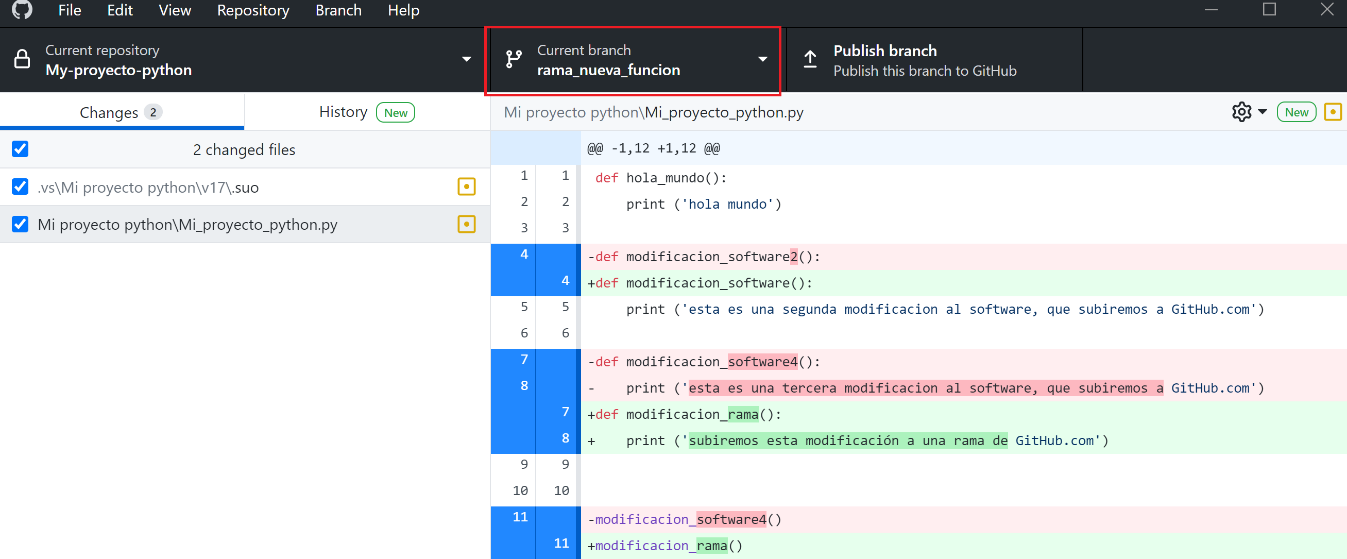
1. Le daremos un nombre y presionaremos “**Create Branch**”



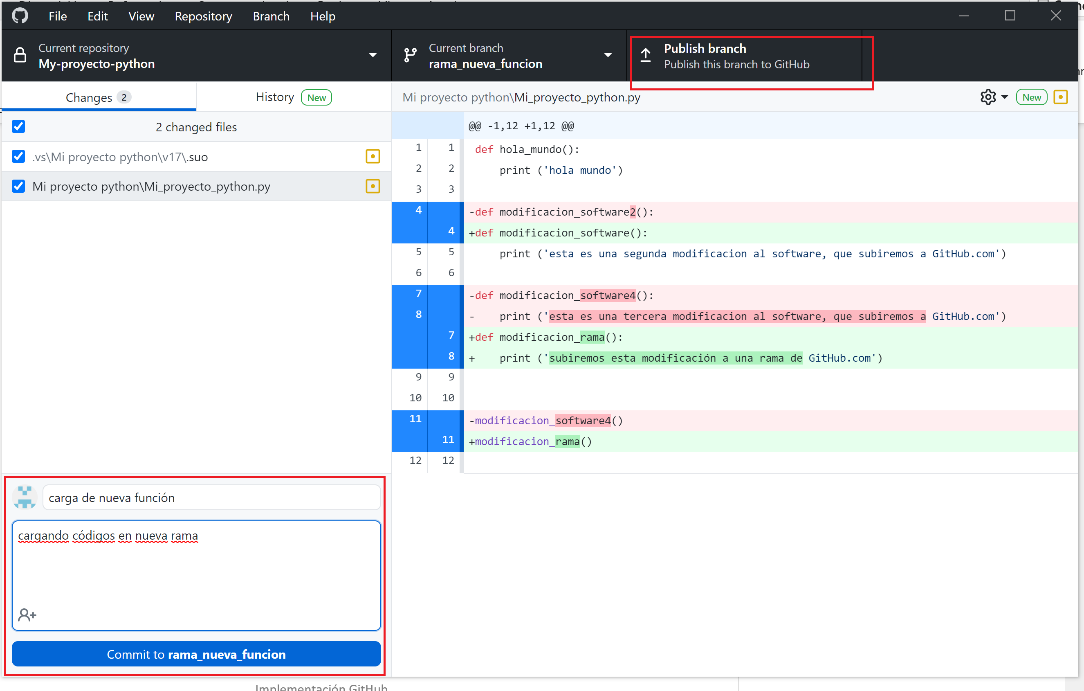
1. GitHub Desktop nos indicará que podemos cargar los cambios en la rama principal, o en la rama recién creada, seleccionaremos la segunda opción.



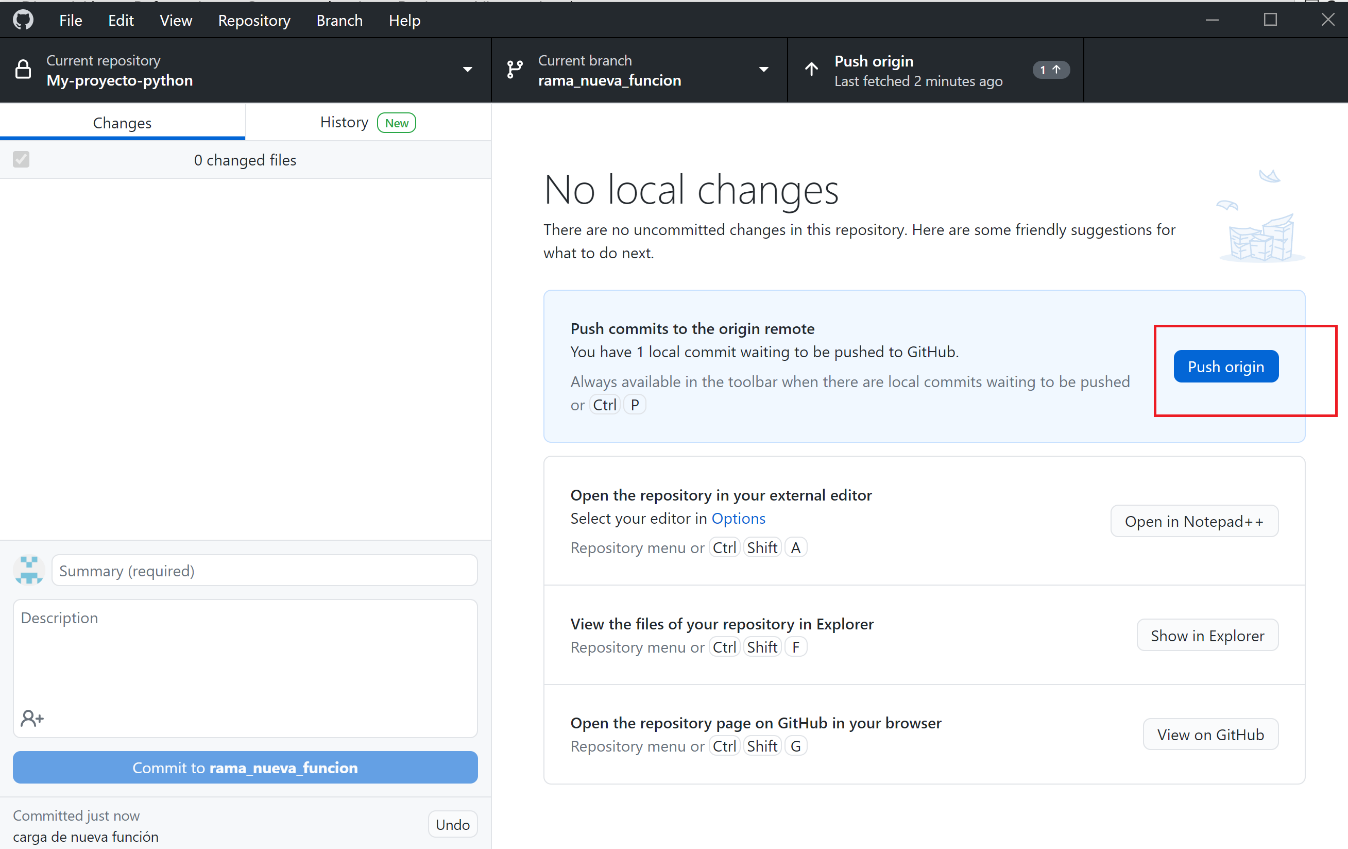
1. Como vemos en la imagen de abajo, aparecerá preseleccionada nuestra nueva rama donde realizaremos las actualizaciones.



1. Presionaremos en el botón “**Publish branch**” para crear nuestra rama en el GitHub.com, luego ingresaremos los datos para el “**commit**” en la nueva rama.

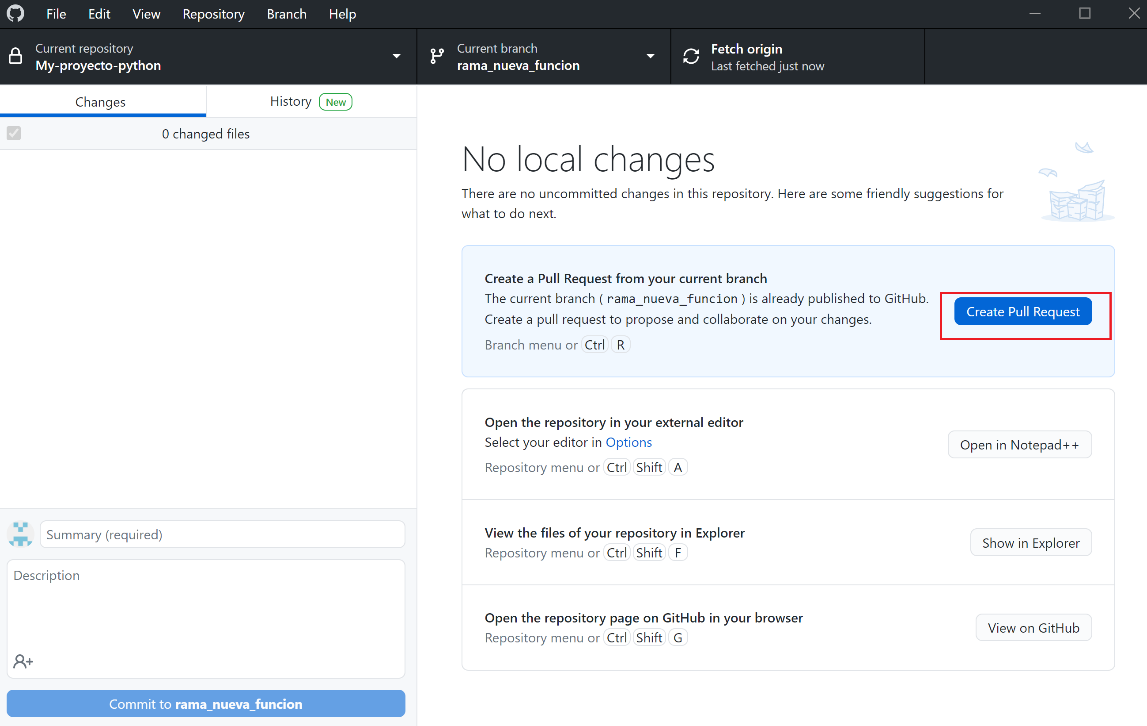


1. Por último, presionaremos en “**Push orign**” para cargar los datos.

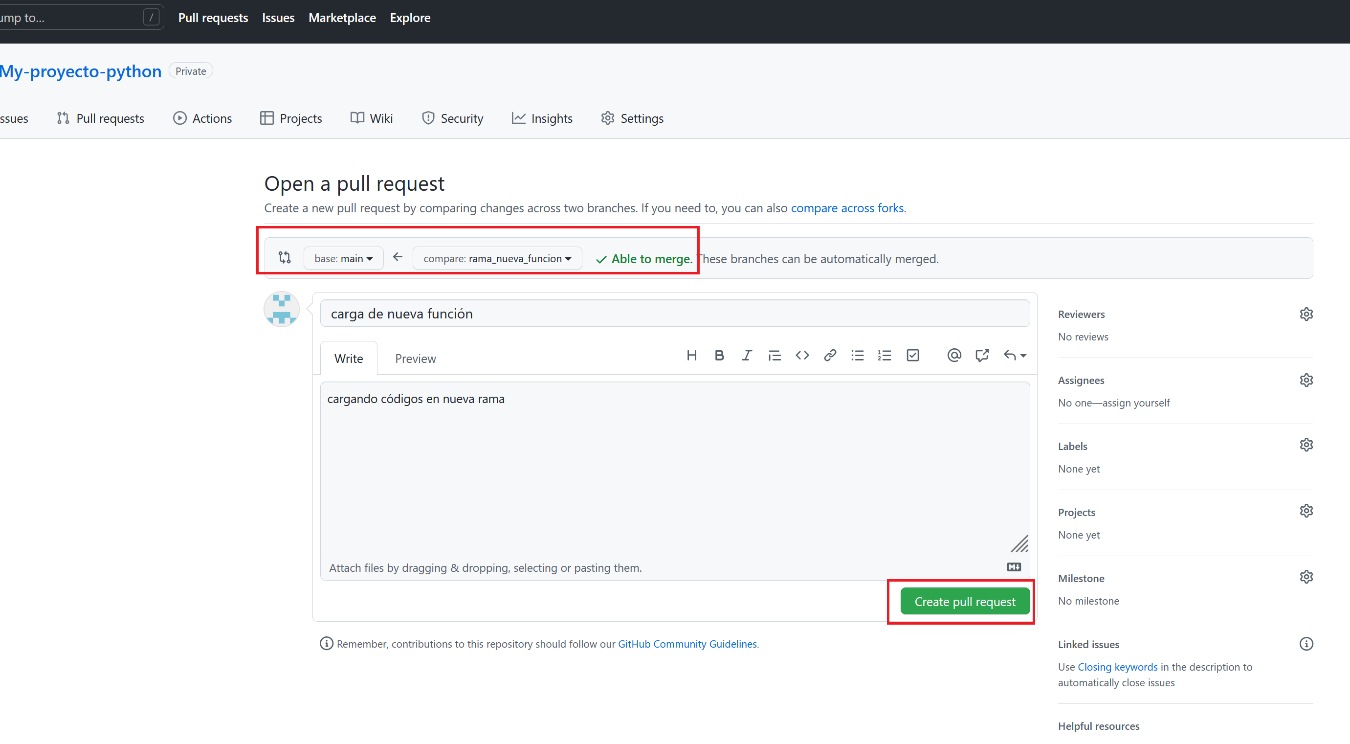


## Unir ramas.

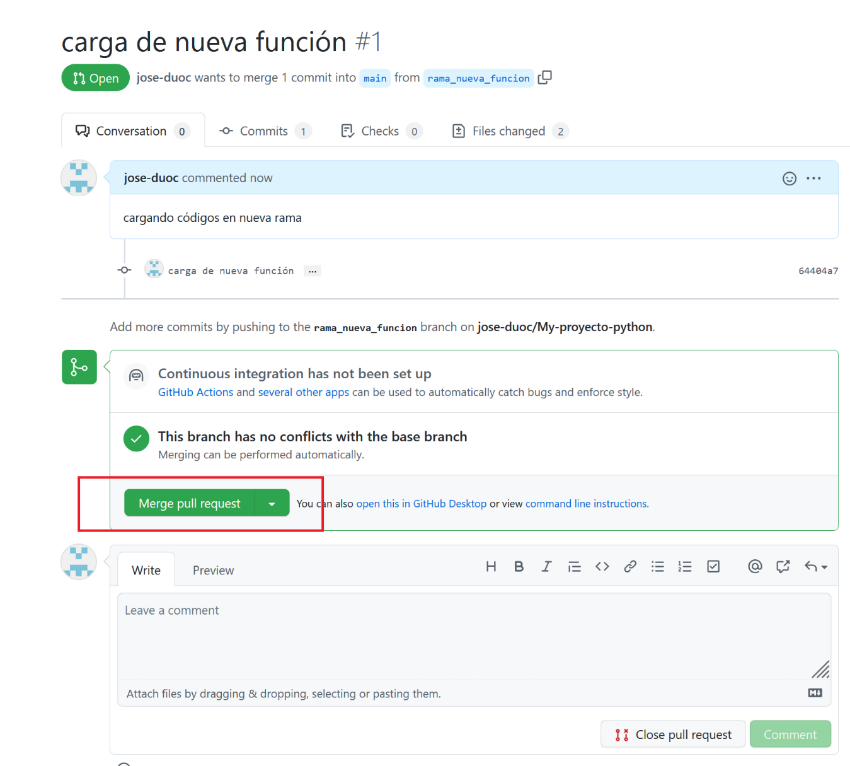
1. Cuando necesitemos unir ramas, realizaremos el proceso de “**Pull request**”.



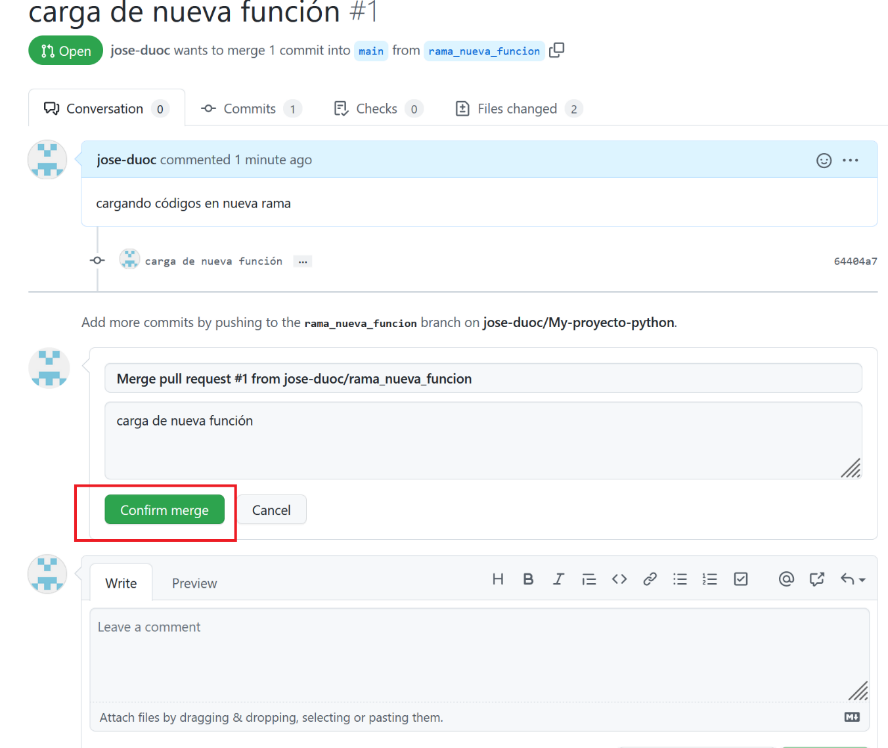
1. GitHub Desktop nos llevará a GitHub.com para realizar el proceso de unión de ramas. A continuación, veremos como GitHub nos indica que la rama llamada “**rama\_nueva\_funcion**” se unirá la rama “**Main**”. Presionaremos el botón “**Create pull request**”



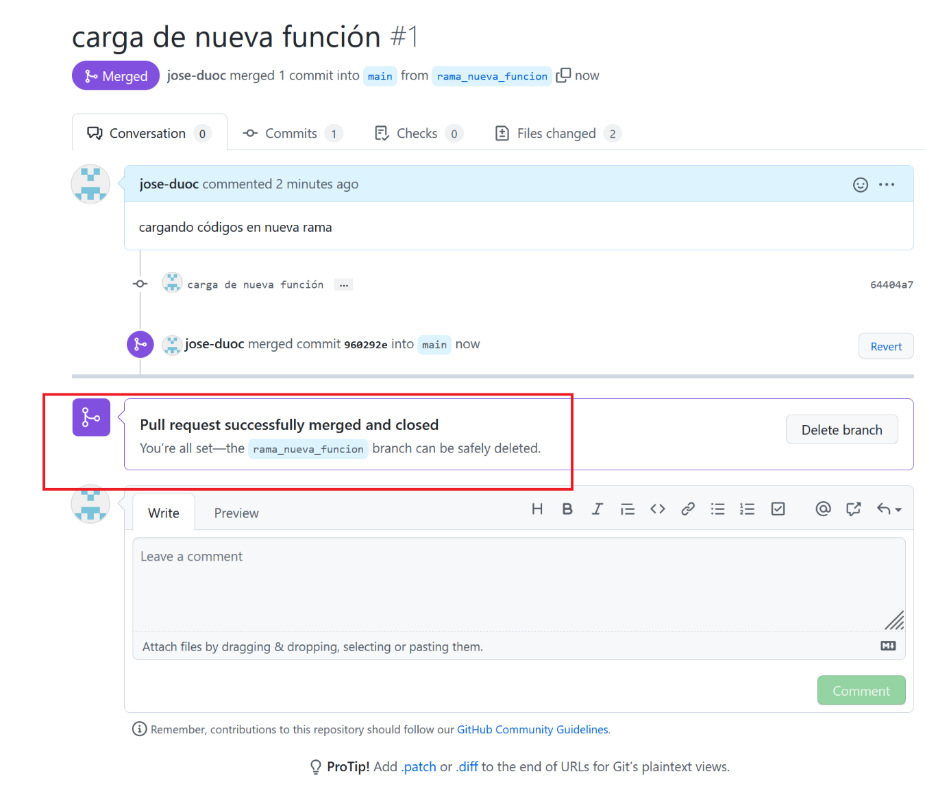
1. GitHub nos indica que no hay conflictos y que podemos presionar el botón “**Merge pull request**”



1. Le damos en “**Confirm Merge**”



1. GitHub nos indicará que ambas ramas fueron unidas.



**NOTA:** para evitar posibles conflictos de archivos, siempre se debe respetar tres reglas principales.

1.- Se recomienda que cada integrante de cada proyecto, solo este enfocado en la modificación de un área del proyecto, no es recomendable que dos programadores trabajen en un mismo archivo al mismo tiempo, si no se tiene la suficiente práctica, puede provocar conflictos de subida de archivos.

2.- Es recomendable que antes de trabajar en tu proyecto o repositorio local, siempre se verifique que no existan actualizaciones desde el servidor, en caso de haber, descargar (realizar Pull) antes de trabajar localmente.

3.- Es ideal que un proyecto de 3 integrantes sea trabajado a lo mínimo con 3 ramas, siendo cada rama utilizada por cada integrante. Así evitamos conflictos en archivos que puedan ser manipulados por más de una persona.

