Tutorial - Control de versiones

Elaborado por Carlos Rodríguez y Ricardo Bernal

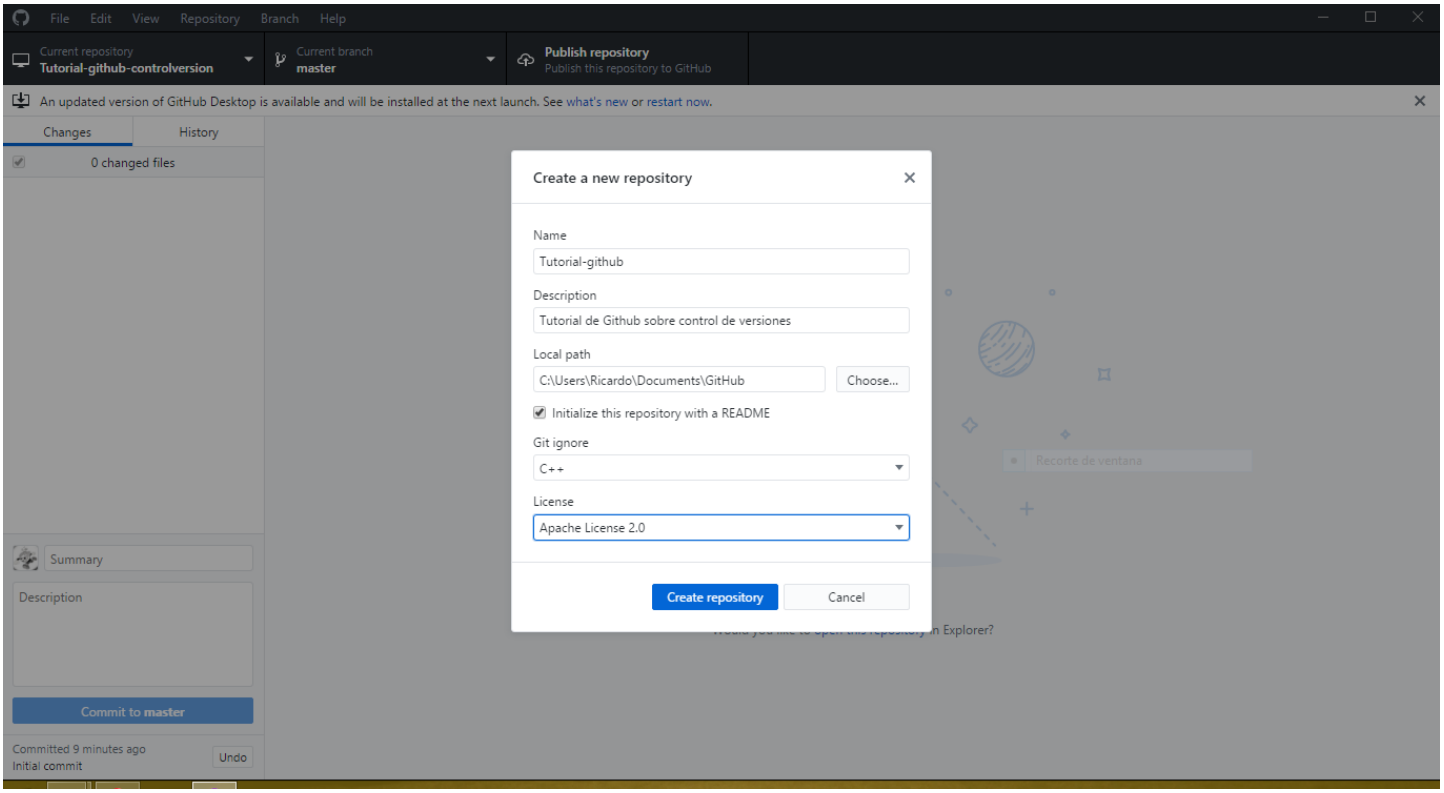
El objetivo de este tutorial es explicar el funcionamiento básico de GitHub (su versión Desktop) como sistema de control de versiones lo cual permita rastrear o realizar seguimiento al desarrollo y cambios de tus archivos.

Antes de empezar, se necesita:

- Instalar GitHub Desktop en tu computadora. Puedes hacerlo por medio del siguiente link.
- Crear una cuenta en GitHub donde se alojaran tus repositorios.

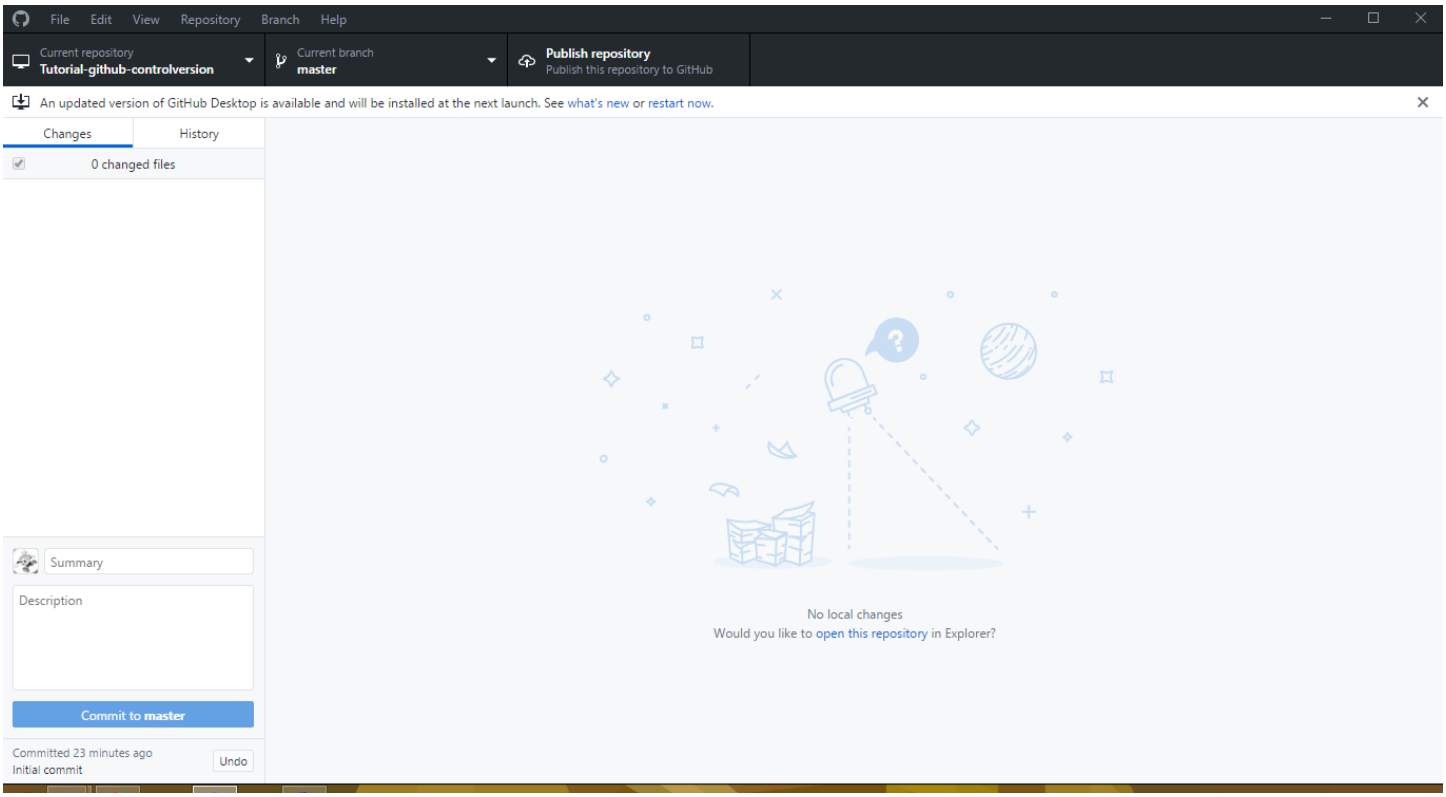
Una vez hecho, podemos comenzar creando un repositorio. Para ello, dirígete a la barra principal superior y haz click en **File > New Repository** (Ctrl+N). Se mostrará un recuadro donde nos piden que le otorguemos un nombre al repositorio, una descripción sobre este y el directorio local donde haremos la copia local del repositorio.

Para inicializar el repositorio, marcaremos la opción "**Initialize this repositorio with a README**". Este archivo permite ofrecer un resumen o guía rápida sobre nuestra aplicación o librería (visible desde la interfaz web de GitHub). Para terminar, elegiremos un lenguaje de desarrollo para crear el archivo *gitignore* el cual nos permite especificar que archivos y/o carpetas deben ignorarse al hacer *push* (subir) el repositorio. Una vez hecho esto, hacemos click en **Create Repository**

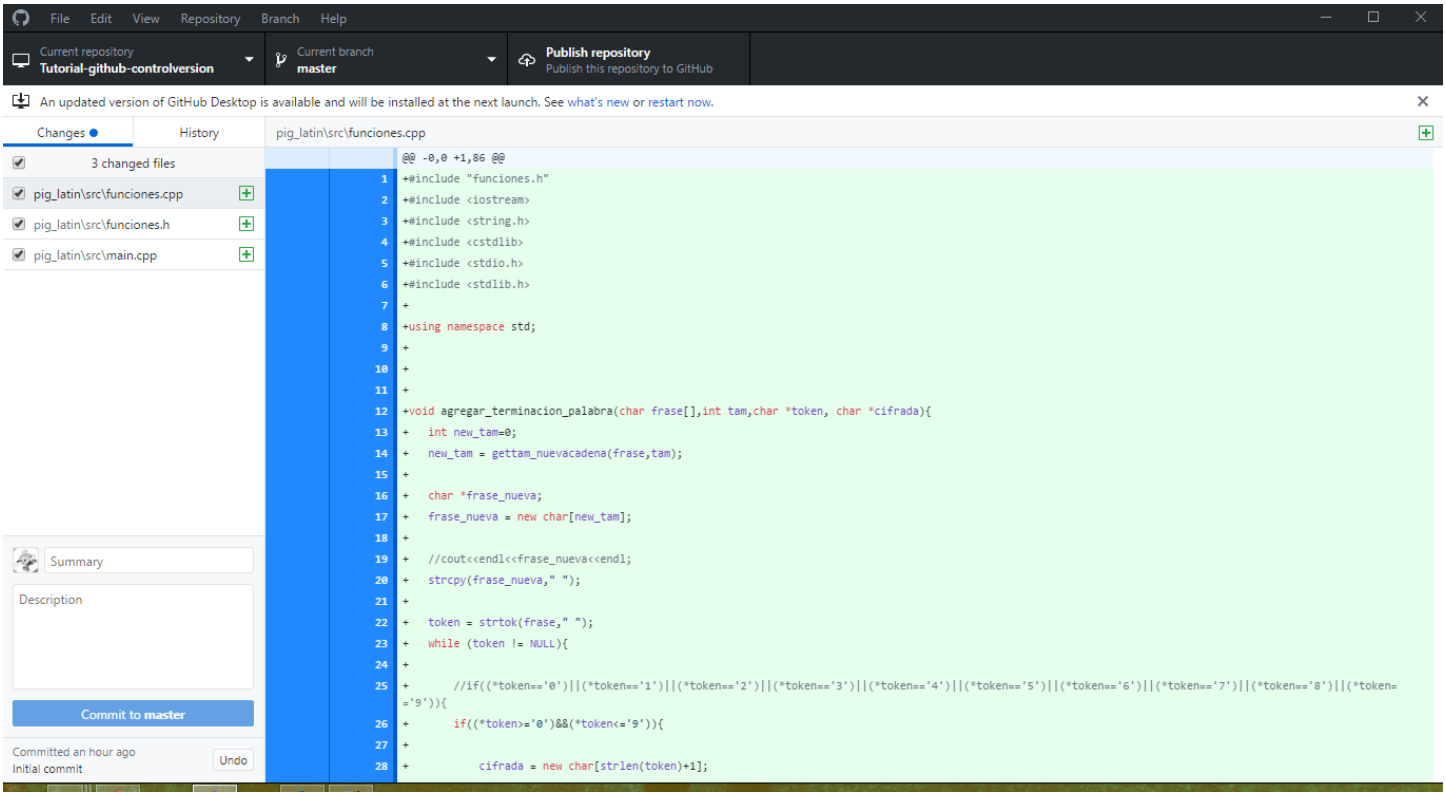


Ya tenemos creado nuestro repositorio pero solo de forma local, hace falta publicarlo en GitHub. Para ello nos ubicamos en la barra que está debajo de la barra principal. Esta barra tiene 3 botones respectivos que nos permite cambiar entre el repositorio actual y otros repositorios que poseamos, cambiar entre la rama (branch) actual y otras, y publicar o cargar los cambios realizados en el repositorio (depende de la situacion).

Para publicar nuestro repositorio, hacemos click en el último botón "**Publish repository**".

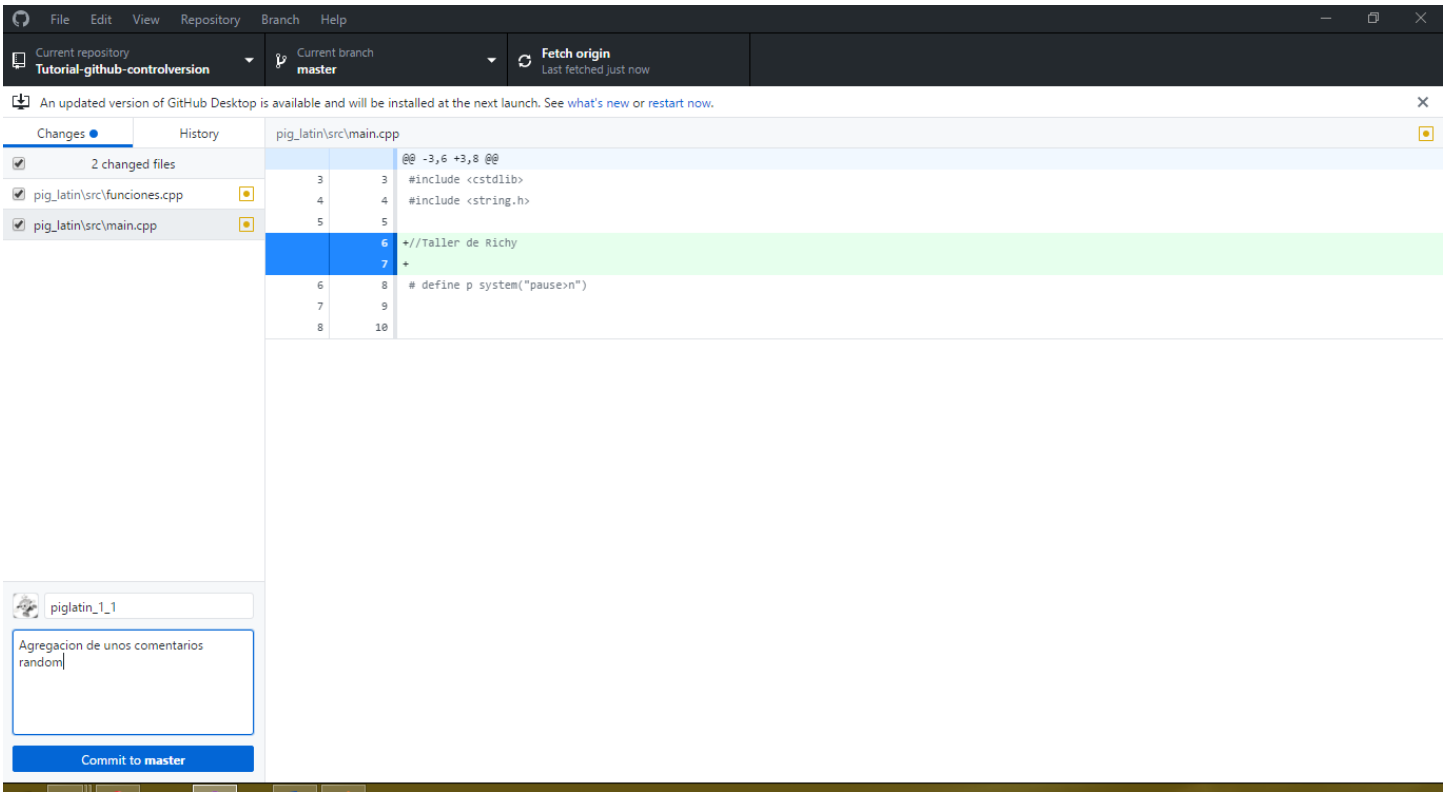


Una vez publicado, ya podemos comenzar a trabajar en nuestro repositorio. Para efectos practicos de este tutorial, copie en el directorio del repositorio (Ctrl+Shift+F) un taller sobre piglatin en C++.

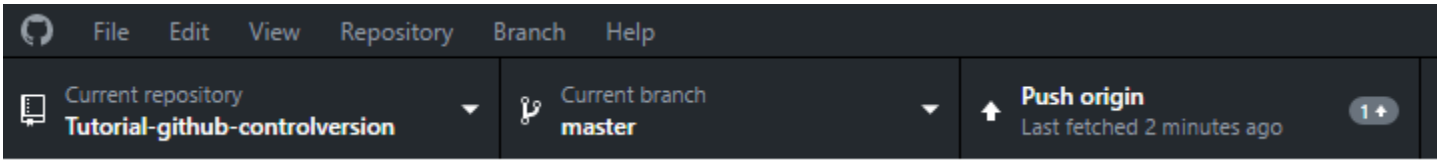


Para anotar los cambios (**commit**) en el repositorio en GitHub, debes dar un resumen/titulo de los cambios y, de manera opcional, incluir un mensaje opcional que describa de forma breve en que consiste este “commit”.

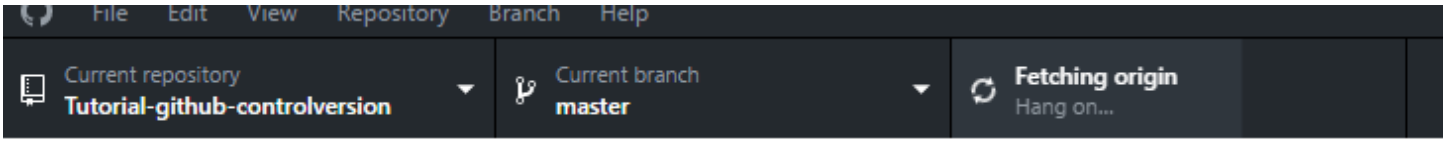
Cuando anotes el cambio verás que aparece el mensaje "**Commit to master**". Esto quiere decir que te refieres a la rama 'master'. En un repositorio Git es posible tener varias ramas. Para anotar los cambios, hacemos click en "**Commit to master**".



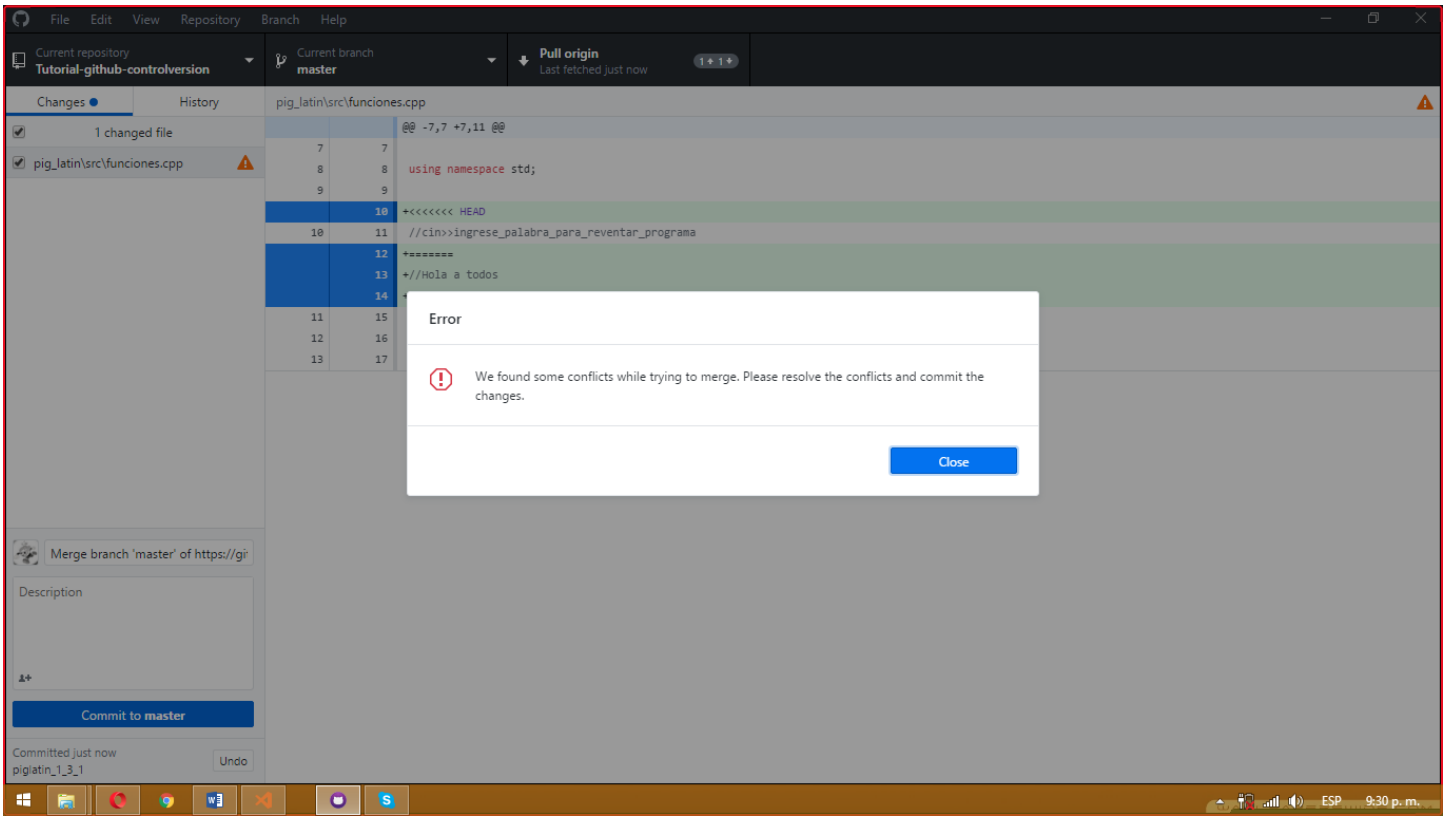
Una vez anotado los cambios, hay que cargarlos al origen el cual es el repositorio en GitHub. Esto se hace con tal solo hacer click en "**Push origin**".



Veremos cómo va cargando y aplicando los cambios anotados anteriormente.

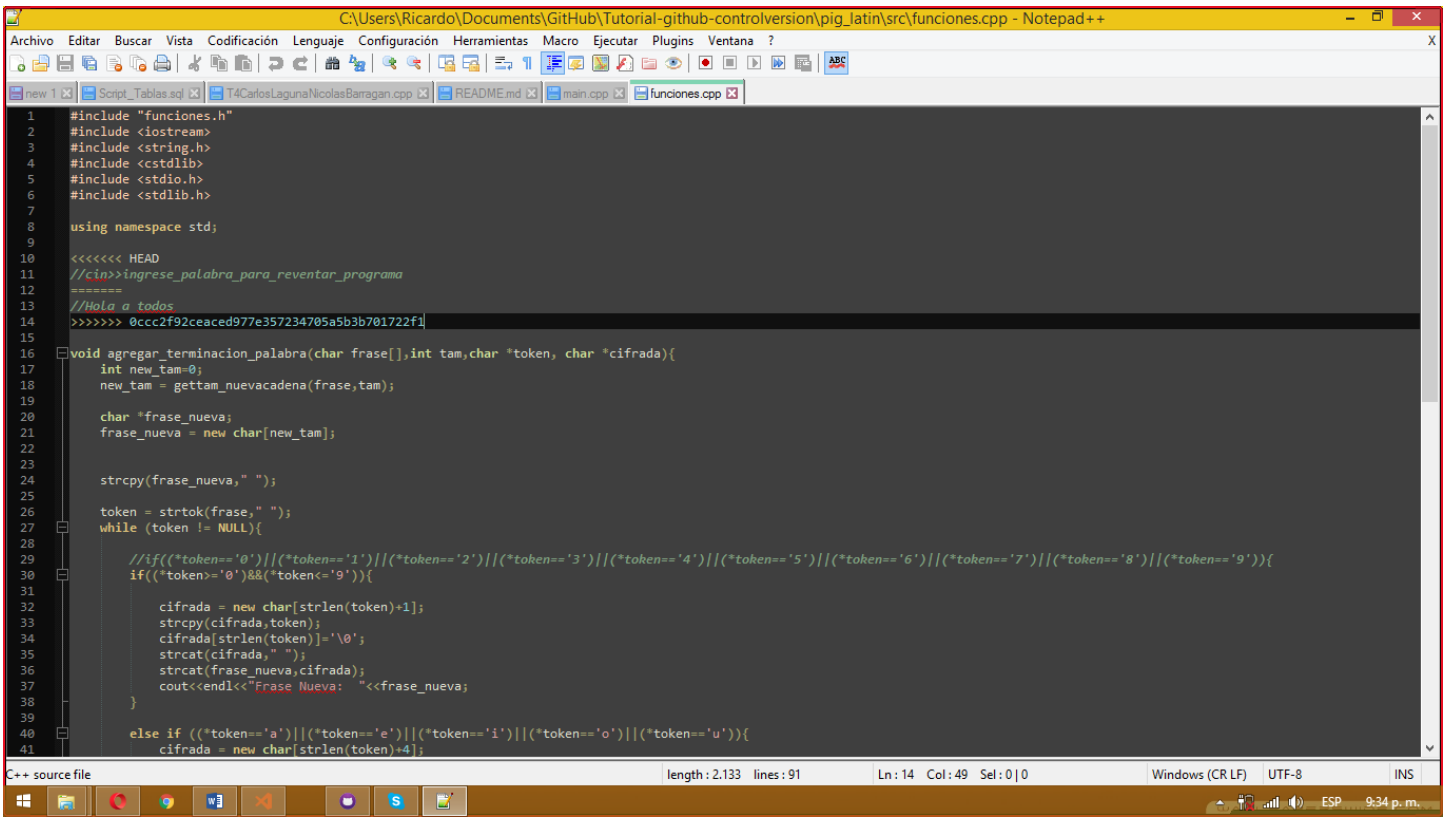


Uno de los errores más comunes es que los archivos existentes en el repositorio de GitHub no concuerden con el contenido de los archivos en tu repositorio local. Nos aparecerá una ventana indicando el conflicto al intentar mezclar (*merge*) los archivos.



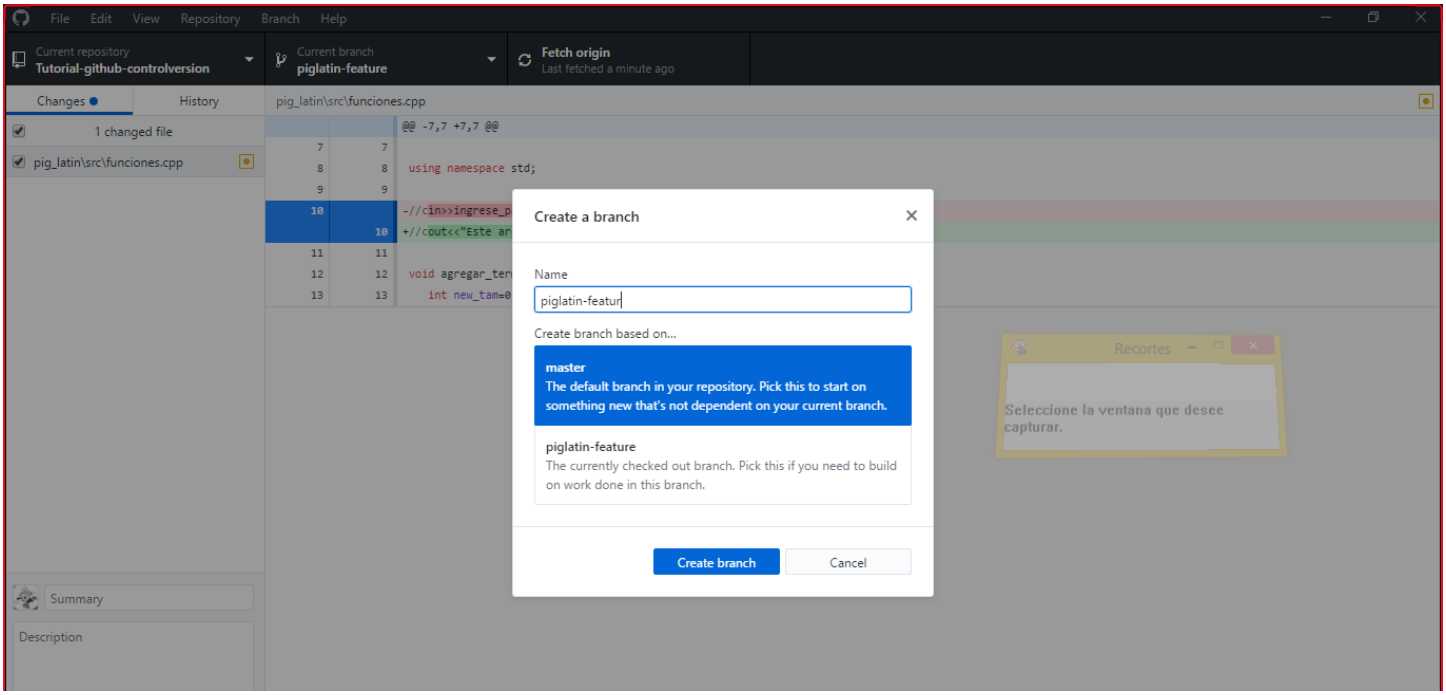
GitHub para facilitarnos el trabajo al solucionar este conflicto, colocara banderas/indicadores en el documento en las líneas que no coincidan entre la versión local y la versión remota del repositorio. La bandera HEAD indicará el cambio actual mientras que el cambio entrante estará marcado por un numero de versión generado por GitHub.

Para solucionar esto, es suficiente con decidir con que cambios vamos a quedarnos y eliminar las banderas. Después, volvemos a anotar los cambios y hacemos *push*.

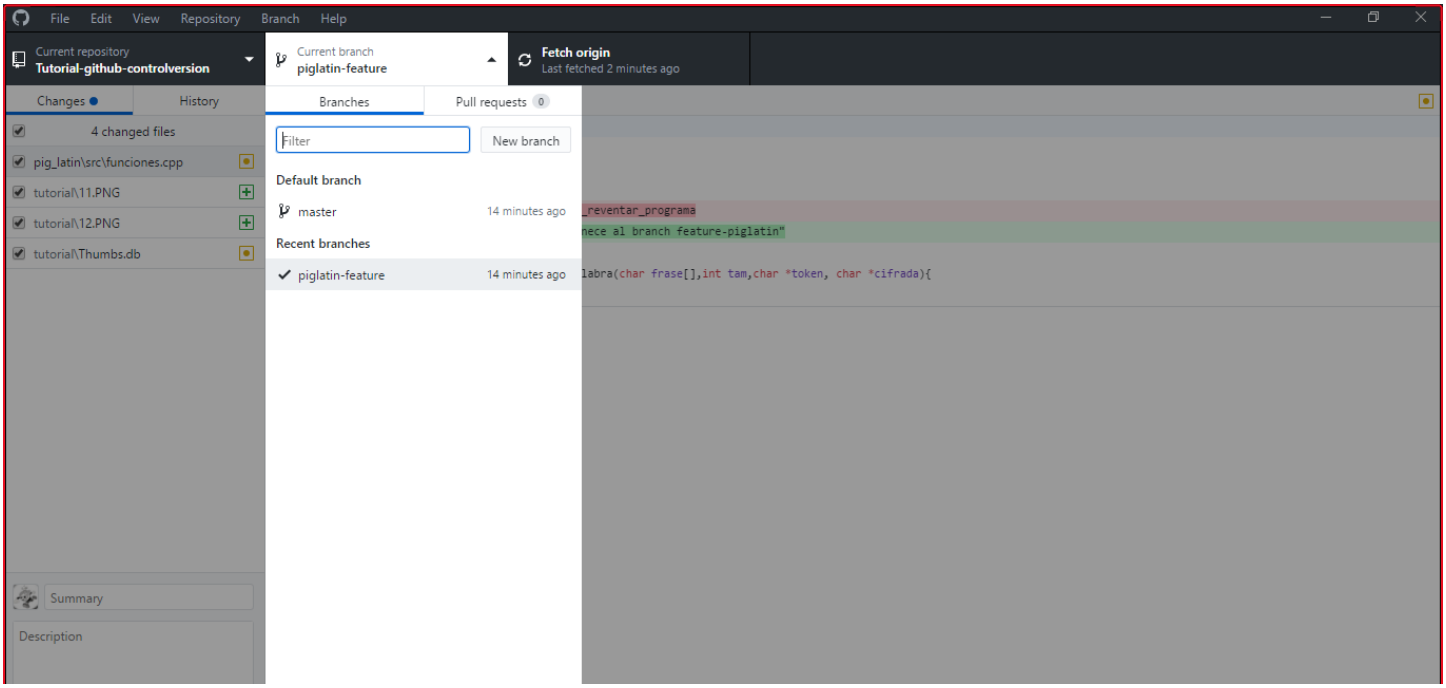


Como se explicó anteriormente, las ramas (*branch*) son, en esencia, lugares distintos en los que puedes trabajar el mismo repositorio. Se crean con el fin de no interferir con el esfuerzo de otro colaborador que trabaja sobre los mismos archivos que nosotros.

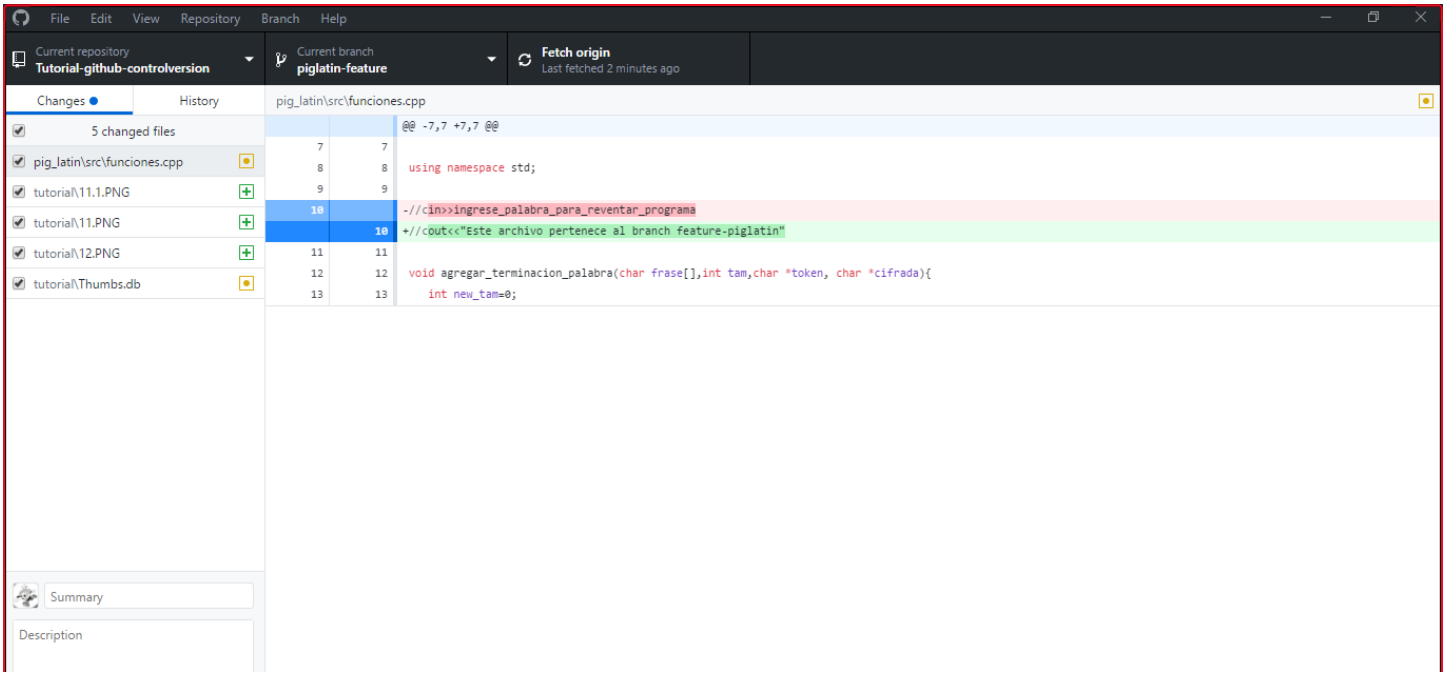
Para crea un nuevo branch, en la barra principal hacemos click en **Branch > New Branch.** Debemos colocar un nombre para identificar la nueva rama e indicar de cual branch se basara (hara una copia). Hacemos click en **Create branch.**



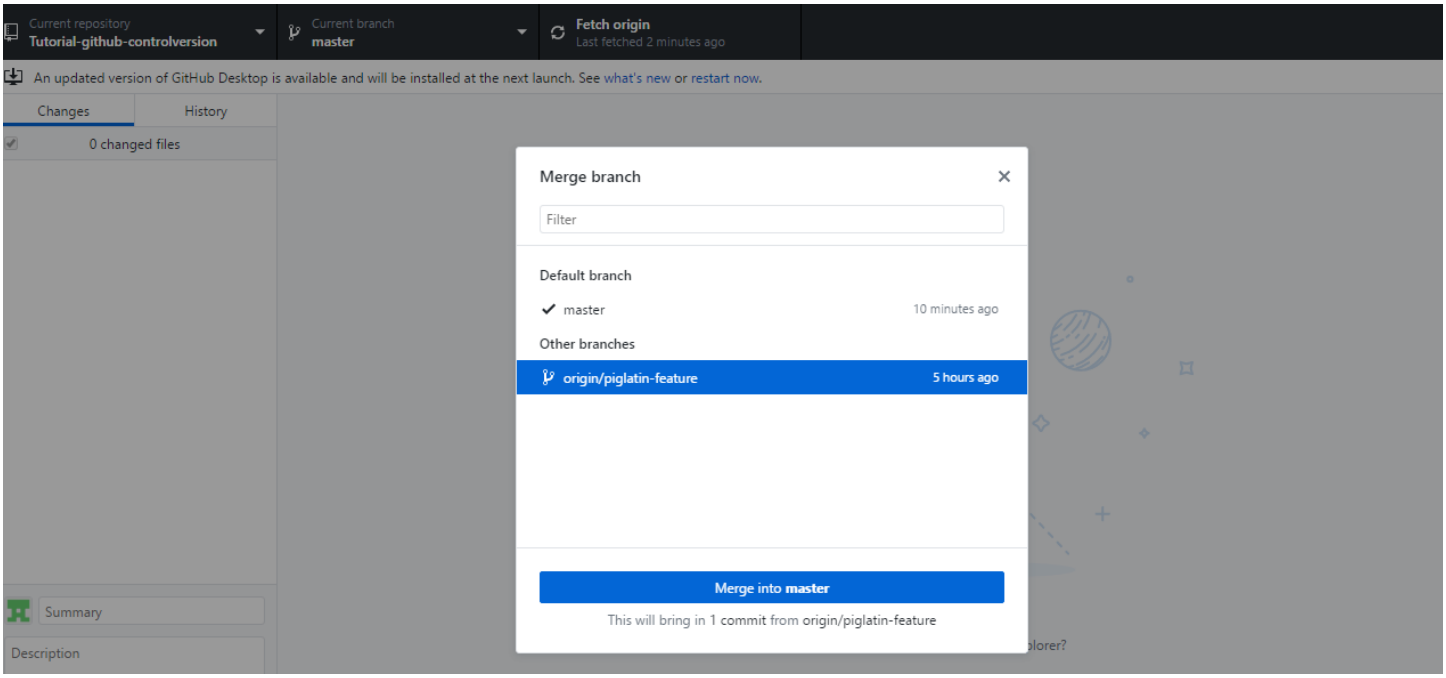
Ahora podemos visualizar los distintos branch y cambiar entre ellos.



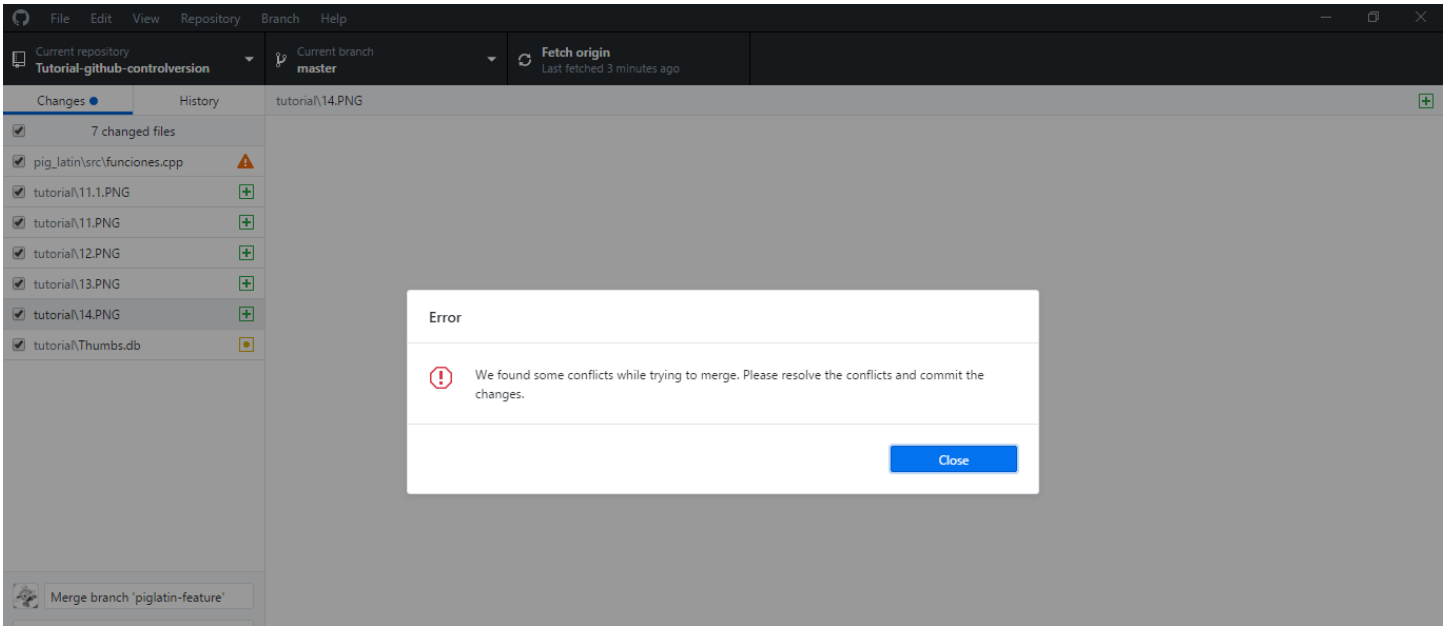
Para ilustrar como mezclar (*merge*) dos ramas y sus posibles conflictos, hemos modificado los archivos tanto en la rama maestra como en la rama extra *piglatin-feature* que hemos creado, pero solo hemos anotado y realizado push a los cambios de los archivos en la rama principal. Los cambios en la rama piglatin-feature se han anotado de forma local.



Para mezclar o hacer *merge* de dos ramas, nos dirigimos a la barra superior y hacemos click en **Branch > Merge into current branch.** Seleccionamos la rama que deseamos mezclar con nuestra rama principal, en este caso *master*. Al seleccionarla, en la parte inferior de la ventana se nos indicara el número de commits que se realizaran al realizar la mezcla. Hacemos click en **Merge into master**, todos los commits realizados en la otra rama pasaran a aplicarse a la rama maestra o *master*.



Puede suceder que distintos colaboradores intenten hacer merge de una a otra rama y eso desincronize los repositorios locales de los demás. Al ocasionar eso, no podrá aplicar los cambios locales hasta que acepte los nuevos cambios ya aplicados en el repositorio de GitHub.



GitHub al igual que con el problema anterior, marcara los cambios entrantes y locales en los archivos para que decidamos cuales aplicar y cuales eliminar.