



Sistema de control de versiones

Los sistemas de versionado registran cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo. Esto permite trabajar al mismo tiempo sobre la misma o diferentes versiones del código, pudiendo después combinarlas y mantener una progresión más organizada del código de nuestro sistema.

Los cambios realizados sobre los archivos son accesibles a lo largo del tiempo, dando la posibilidad de obtener fecha y autor de los cambios. También nos brinda la posibilidad de volver a diferentes estados del código o utilizar un estado en concreto del mismo.

Existen varios sistemas de control (SVN, Mercurial, etc.), el que se utilizará en esta cátedra es **Git**.

Git

Es un sistema de control de versiones descentralizado, en la actualidad es uno de los más populares, otra de sus características es que es gratuito y de código abierto.

- **Primeros pasos**

Para su instalación hay que acceder a la [web del proyecto](#) y descargar el instalador. El proceso es el mismo que se seguiría con otro software al momento de instalarlo.

Una vez instalado ya está listo para comenzar a utilizarse.

Servidor

Para poder tener nuestro código de forma remota, ya sea por temas de seguridad, espacio o con el fin de compartirlo, es necesario contar con un servidor remoto. En la actualidad hay varias empresas que brindan gratuitamente espacios de almacenamiento para repositorios de código.

Entre los más populares podemos nombrar, por ejemplo, GitHub, GitLab y BitBucket.

En esta cátedra (y este documento) se utilizará GitHub pero el fin y las formas de trabajo son similares. (Se dejará también un documento con BitBucket pero no se utilizara).



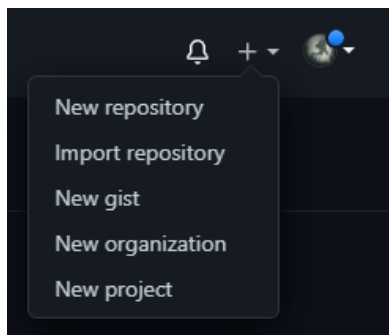
GitHub

Una vez realizada la instalación, para poder crear un repositorio (Almacén de archivos dentro del servidor). Debemos crear una cuenta (proceso es gratuito).

En GitHub la cantidad de repositorios que podemos crear es ilimitada.

- **Creación de un repositorio**

- Para crear uno abriremos el menú superior de la web y elegiremos la opción ***New repository (Nuevo repositorio)***.

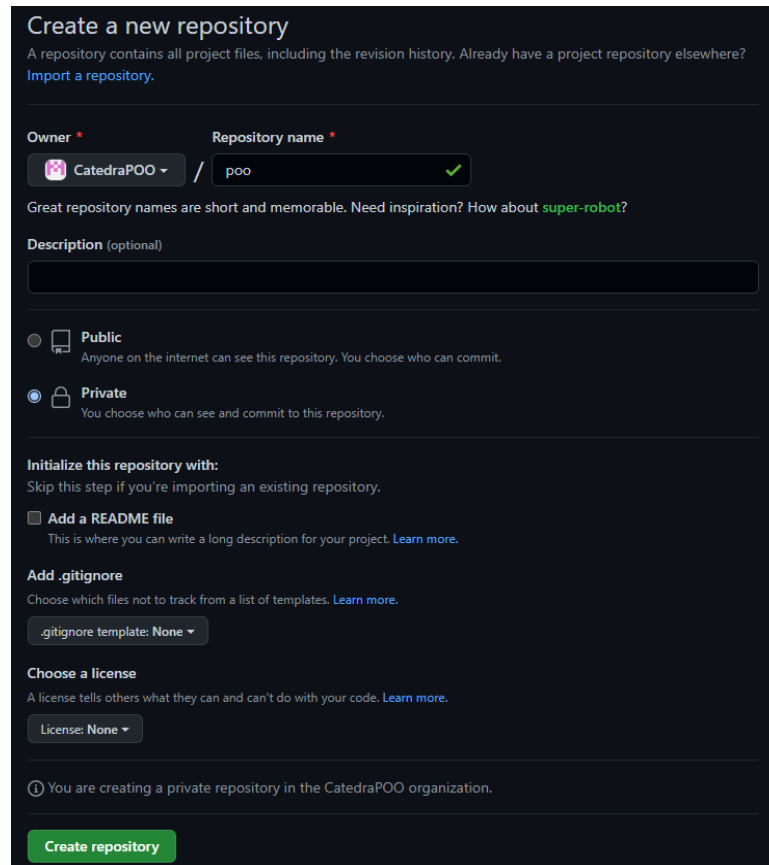


- Esta opción la encontraremos en la pantalla principal o en el listado de repositorios. Se ve actualmente como en la siguiente imagen.





- Luego, la pantalla para crear un repositorio se ve como a continuación



The screenshot shows the GitHub 'Create a new repository' interface. At the top, it says 'Create a new repository' and provides a brief explanation of what a repository is. Below this, there are two main sections: 'Owner' and 'Repository name'. The 'Owner' is set to 'CatedraPOO' and the 'Repository name' is 'poo'. There is a green checkmark next to the repository name. Below these fields, there is a 'Description (optional)' text area. Further down, there are two radio buttons for 'Public' and 'Private'. The 'Private' option is selected. Below this, there is a section 'Initialize this repository with:' which includes options to 'Add a README file' and 'Add .gitignore'. The 'Add .gitignore' section has a dropdown menu set to 'None'. At the bottom, there is a 'Choose a license' section with a dropdown menu set to 'None'. A green 'Create repository' button is at the bottom right.

En Repository name (Nombre del repositorio) debemos ingresar un nombre único para nuestra cuenta. Es decir, una cuenta puede tener múltiples repositorios pero no puede haber dos con el mismo nombre.

La siguiente opción más importante es la que nos permite elegir entre Public y Private. Si un repositorio es público el repositorio será visible para otros. En el caso de elegir privado, estará oculto pero podremos dar permiso a otros usuarios para que puedan acceder.

Al crear nuestro repositorio obtendremos una url única para poder clonarlo y trabajar con él de forma remota.



The screenshot shows the 'Quick setup' section of the GitHub interface. It has a dark blue background with white text. The text reads 'Quick setup — if you've done this kind of thing before'. Below this, there are three buttons: 'Set up in Desktop', 'HTTPS', and 'SSH'. To the right of these buttons is the URL 'https://github.com/CatedraPOO/prueba.git'.



Trabajando con Git

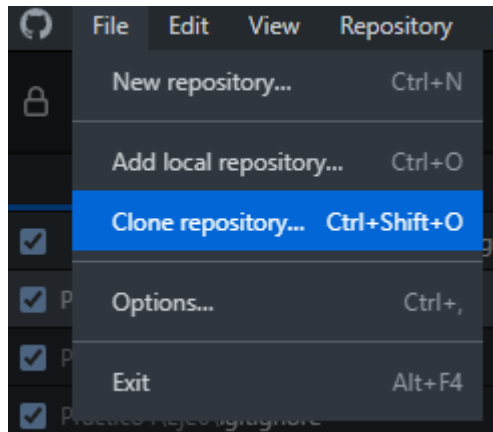
Se puede utilizar de dos formas, una por línea de comandos, sin embargo, al comenzar con esta tecnología puede ser mucho más cómodo utilizar una interfaz gráfica que haga el trabajo por nosotros.

En el caso de GitHub, nos ofrecen su propio sistema para esto: GitHub Desktop. El mismo puede descargarse desde <https://desktop.github.com/>

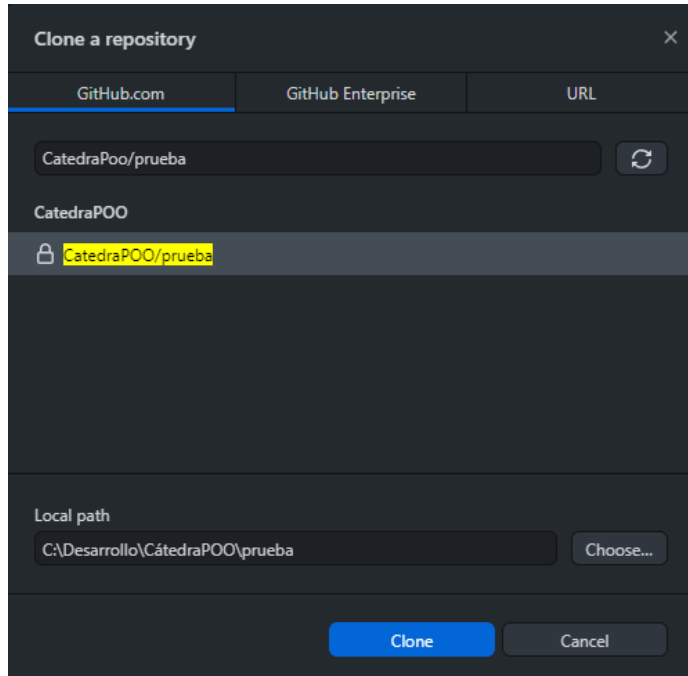
Una vez instalado debemos asociar nuestra cuenta para poder acceder a los repositorios remotos.

Clonar un repositorio existente

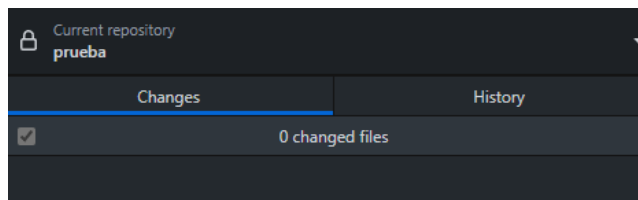
Para traer a nuestra computadora un repositorio ya existente en un servidor remoto es necesario clonarlo. Para eso, podemos hacer uso de la siguiente opción de Github Desktop



Esto abrirá una nueva pantalla donde podemos seleccionar el repositorio deseado y la ruta local donde lo clonaremos.

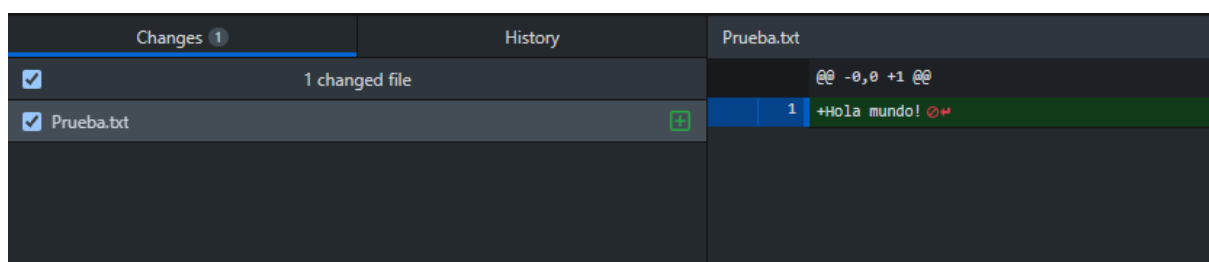


Por defecto veremos que no hay cambios en nuestro repositorio.



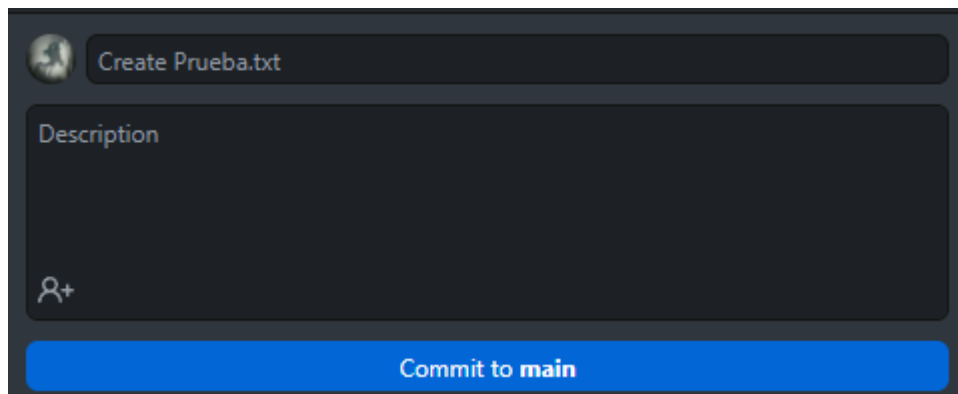
Commits

Si vamos a la carpeta donde está ubicado nuestro repositorio y, por ejemplo, añadimos un nuevo archivo GitHub Desktop mostrará automáticamente nuestros cambios. En la siguiente imagen puede verse a la izquierda un nuevo archivo llamado Prueba.txt y a la derecha los cambios del mismo(al ser un archivo nuevo, todos los cambios son líneas nuevas)





Sin embargo, estos cambios aún no son efectivos en el historial. Para eso, debemos hacer un **commit** de los mismos. En la parte inferior izquierda de esta pantalla veremos un formulario que permite poner un mensaje para el **commit**.



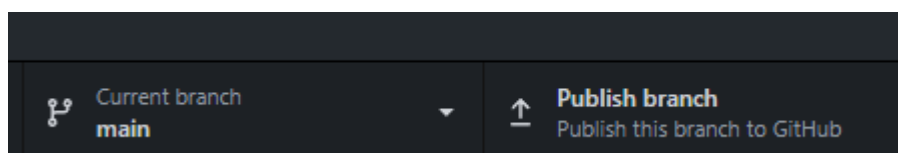
The screenshot shows the GitHub commit interface. At the top, there's a text input field with the placeholder "Create Prueba.txt". Below it is a large text area labeled "Description". At the bottom, there's a blue button labeled "Commit to main".

Al presionar **Commit to main**(ya veremos el significado de **main**) esta versión del archivo será almacenada en el historial de cambios de Git.

Si luego de realizar el commit volvemos a hacer cambios sobre el archivo estos se volverán a detectar y podremos ver qué cambió exactamente en nuestros archivos desde el último commit.

Changes 1		History	Prueba.txt
1 changed file			@@ -1 +1 @@
✓ Prueba.txt	+	1	-Hola mundo! ↗
		1	+Hola mundo modificado! ↗

En este punto, la última versión del código puede estar commiteada pero solo a nivel local. **El servidor remoto no tiene aún nuestros cambios.** Para eso es necesario que hagamos un **push** de los cambios locales al servidor remoto. Para eso presionamos el botón **Publish branch** en la parte superior de la interfaz



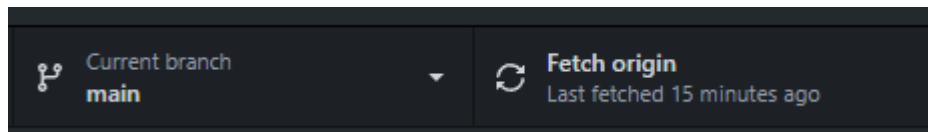
The screenshot shows the GitHub interface at the bottom. On the left, it says "Current branch main". On the right, there is a button labeled "Publish branch" with a subtext "Publish this branch to GitHub".

Al finalizar, podemos acceder por web a nuestro repositorio y validar que efectivamente los cambios ya están publicados.



Actualizar el repositorio local con cambios remotos

Si nuestra versión local estuviera desactualizada respecto a lo que se encuentra en el servidor remoto, ya sea porque hicimos cambios desde otra computadora o fueron realizados por otra persona, podemos hacer una actualización. Para eso podemos presionar el botón “**Fetch origin**” de la parte superior



Branches o Ramas

Podremos ver que GitHub nos menciona una branch de nombre “**main**” que fue creada automáticamente con nuestro repositorio.

Una branch es una de múltiples posibles ramas que puede tener su propia versión del código. Una de las ramas suele ser de la principal mientras las otras se crean a partir de ella para recibir modificaciones.

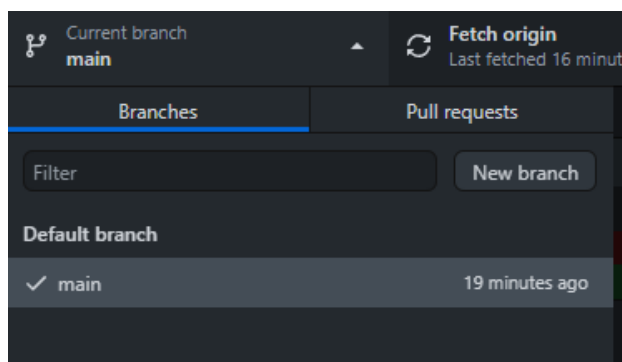
Esto permite que por ejemplo dos personas trabajen en espacios diferentes sin perjudicarse. Al finalizar con una feature, esa rama puede unificarse con la principal llevando hacia ella los cambios que se hayan realizado. A la acción de llevar commits de una rama a otra se le llama mergear (Merge).

El nombre y cuál es la rama principal de nuestro repositorio es algo que podemos definir nosotros mismos a conveniencia. Por lo general esta branch principal suele llamarse **main** o **master**.

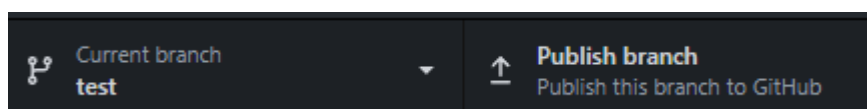


Crear una nueva branch

Presionando en el botón **Current Branch** se abrirá un menú con las branches que poseemos y nos permitirá cambiar o crear una nueva.



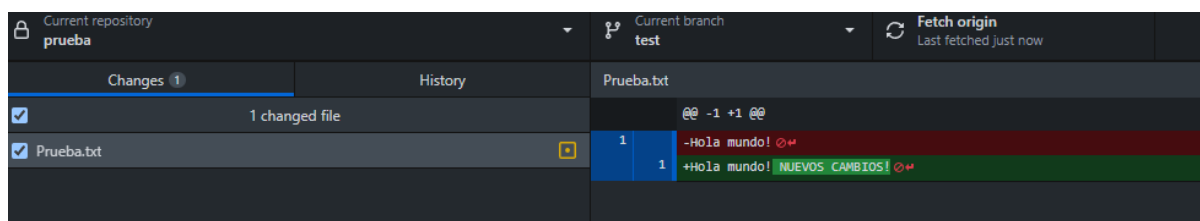
Es importante destacar que si creamos una nueva branch está solo estará en nuestro entorno local hasta que decidamos llevarla al servidor remoto. Para esto último solo tenemos que dar click en **Publish branch**



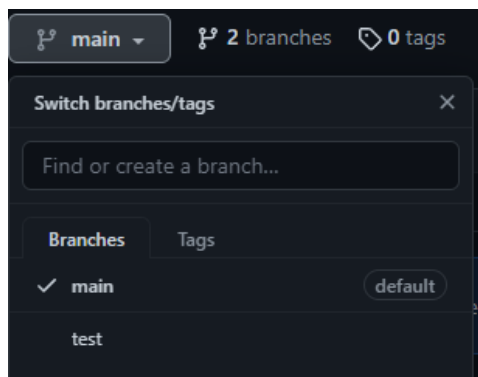
Pull request y merge

En caso de que estemos trabajando en una nueva branch es probable que en algún momento querramos que esos cambios se lleven a otra, ya sea o no la principal.

En la siguiente imagen se ven cambios sobre la branch **test** que luego serán commiteados y llevados al repositorio remoto.



Cuando todos los cambios están commiteados y la rama se encuentre en el repositorio remoto podremos seleccionarla desde la interfaz web



Al hacerlo veremos algo como lo siguiente

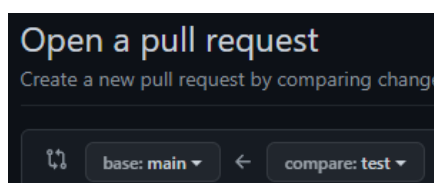


En la parte inferior izquierda de la imagen puede verse un texto que indica que la branch **test** tiene un commit de diferencia con la rama principal (en este caso **main**).

Además, en la parte superior derecha vemos un botón con el texto “**Compare & pull request**”. Si presionamos en el mismo veremos una nueva pantalla que nos permitirá iniciar el proceso de llevar los cambios de **test** a **main**.

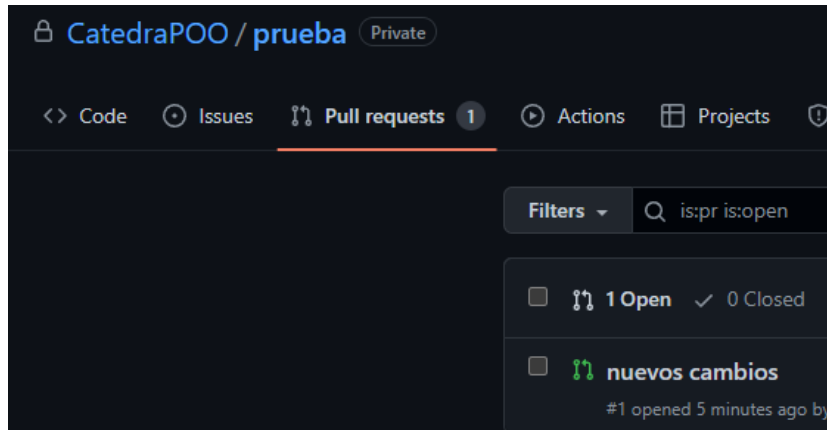
La acción de abrir un **Pull request** puede hacerse también desde la sección **Pull requests** del repositorio.

Ya en la pantalla de creación del pull request, al principio podemos seleccionar las branches. Y, en la parte inferior podemos ver los commits que se llevarán y los archivos modificados en esos commits. En este caso, como se ve en la imagen, la de destino es **main** y la de origen es **test**.

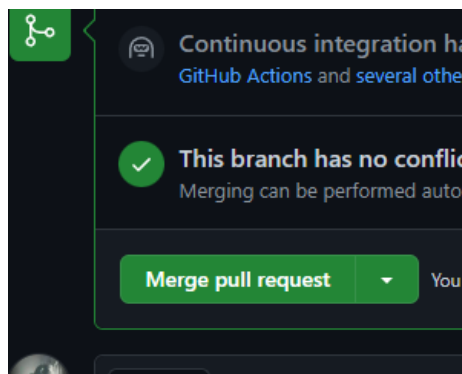




Una vez abierto el pull request, este quedará listado y podrá verse en la sección de pull requests



Al ingresar al pull request, se podrán validar los commits y archivos modificados. En caso de que se considere correcto realizar el merge, se hará uso del botón Merge pull request



Con esto, los commits de la branch de origen serán llevados a la branch de destino.

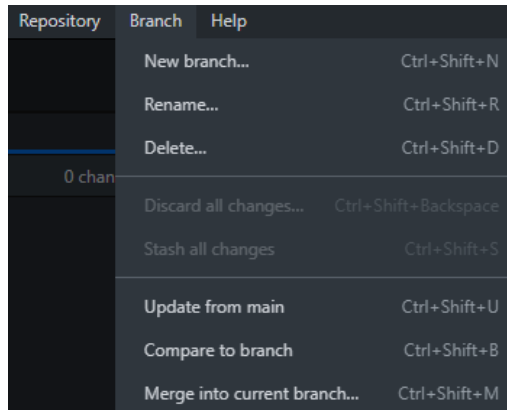
Merge local

En algunas ocasiones puede ser necesario llevar cambios de una rama a otra y no es necesario que se realice en el servidor remoto ni que se abra un pull request y alguien lo apruebe.

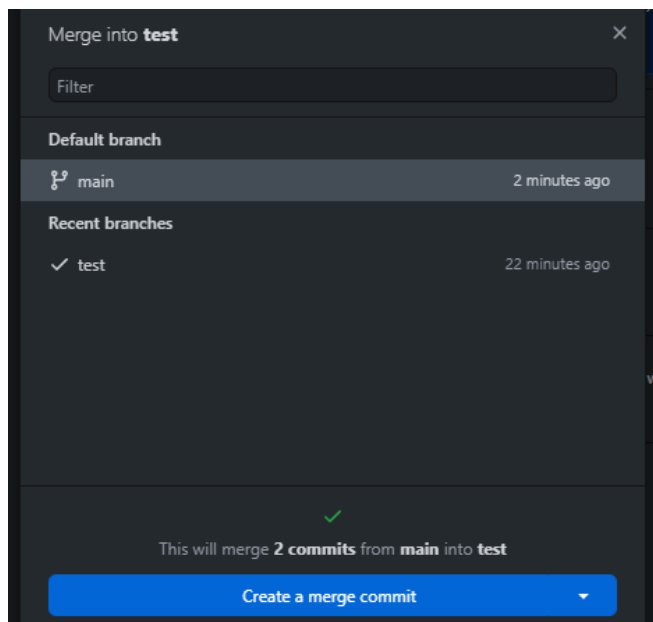
Supongamos que hay cambios en la rama main que queremos traer a nuestra rama test. En ese caso, teniendo seleccionada la rama test procedemos a hacer lo siguiente



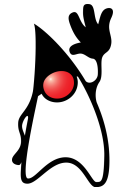
En el menú superior, accedemos a **Branch** y luego seleccionamos **Merge into current branch...**



Eso abrirá una pantalla donde podremos seleccionar la branch que queremos mergear en la que estamos utilizando.



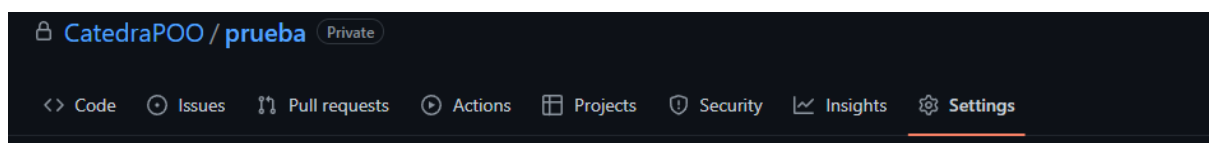
Bastará con presionar el botón **Create a merge commit** para fusionar las ramas en la nuestra. Eso traerá todos los commits que se encuentran en **main** y no existan en **test**.



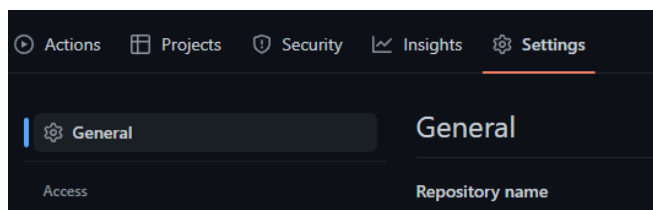
Visibilidad de un repositorio

En el caso de GitHub, tenemos la posibilidad de definir nuestros repositorios como privados o públicos. Si bien esto se define en la creación, es posible realizar modificaciones luego.

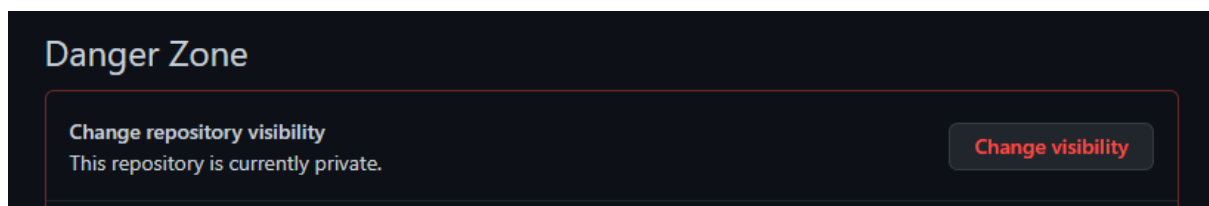
Para eso, desde la web de GitHub, estando dentro del repositorio tenemos que dirigirnos a la sección Settings (A la derecha dentro de la siguiente imagen)



Luego, dentro del menú lateral de esta sección elegimos General.



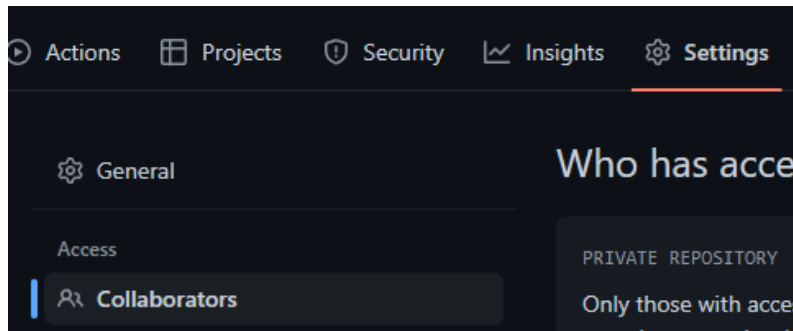
Nos dirigimos a las últimas opciones de esta sub sección y veremos la siguiente



Desde aquí podremos cambiar de público a privado y viceversa.

Colaboradores

Si queremos permitir que otros miembros de GitHub sean colaboradores de nuestro proyecto, tenemos que dirigirnos a la sub sección Collaborators dentro de Settings.



Aquí encontraremos una sección que permite invitar colaboradores al repositorio. La invitación debe ser aceptada para finalizar el proceso.

