

Cuestionario [Actividad 1]:

1. Github es un repositorio en la nube. En este repositorio se pueden subir los cambios realizados en un proyecto. Esto permite que otro integrante del equipo, o uno mismo, desde otro dispositivo tenga la posibilidad de trabajar, actualizar su código local o incluso enviar sus propios cambios.
2. Para crear un repositorio en GitHub inicialmente hay que tener una cuenta registrada. Una vez obtenida la cuenta hay un símbolo "+" el cual lleva a la pantalla de creación del repositorio. Ahí se le da un nombre, se eligen ciertas opciones como su privacidad, la licencia bajo la que está sujeto el código, entre otros.
3. Se puede crear una rama en git con el comando **git branch [NOMBRE_DE_LA_RAMA]**. Por lo general suelo utilizar el comando **git checkout -b [NOMBRE_DE_LA_RAMA]**. Esto me evita tener que usar el comando **git checkout [NOMBRE_DE_LA_RAMA]** para cambiar a la rama recién creada.
4. El principal comando para cambiar entre ramas es **git checkout [NOMBRE_DE_LA_RAMA]**. También existe el comando **git switch [NOMBRE_DE_LA_RAMA]**, pero checkout permite hacer varias cosas más que switch no permite.
5. Para fusionar ramas en git se utiliza el comando merge. Para usarlo correctamente hay que estar posicionado en la rama en la cual queremos volcar los cambios. Por ejemplo si tuviésemos 2 ramas llamadas **rama_alpha** y **rama_beta** y quisiéramos volcar todos los cambios de la rama **alpha** en la **beta** deberíamos cambiarnos a la rama **beta**, si no estuviéramos en ella, con el comando **git checkout rama_beta** y ahí utilizar el comando **git merge rama_alpha**.
6. Para crear un commit en git debemos utilizar el comando del mismo nombre. Antes de utilizarlo debemos asegurarnos que todos los archivos que fueron modificados se hayan "stageado". Si utilizamos el comando **git status** podremos asegurarnos que todos los archivos que queremos incluir en el commit estén preparados. De no ser así podemos usar el comando **git add** . para incluir todos los archivos faltantes. Luego usando el comando **git commit -m "[MENSAJE_DE_COMMIT]"** podemos realizarlo. Es recomendable utilizar un mensaje claro y descriptivo de los cambios que involucre el commit.
7. Para enviar un commit a GitHub se utiliza el comando push. Este comando envía los cambios realizados en la rama donde se corre al repositorio remoto. El comando es **git push origin [NOMBRE_DE_LA_RAMA]**.
8. Git es un software de control de versiones local. Eso implica que el seguimiento y los cambios son controlados en la computadora donde estoy trabajando. Un repositorio remoto es un servicio en la nube que

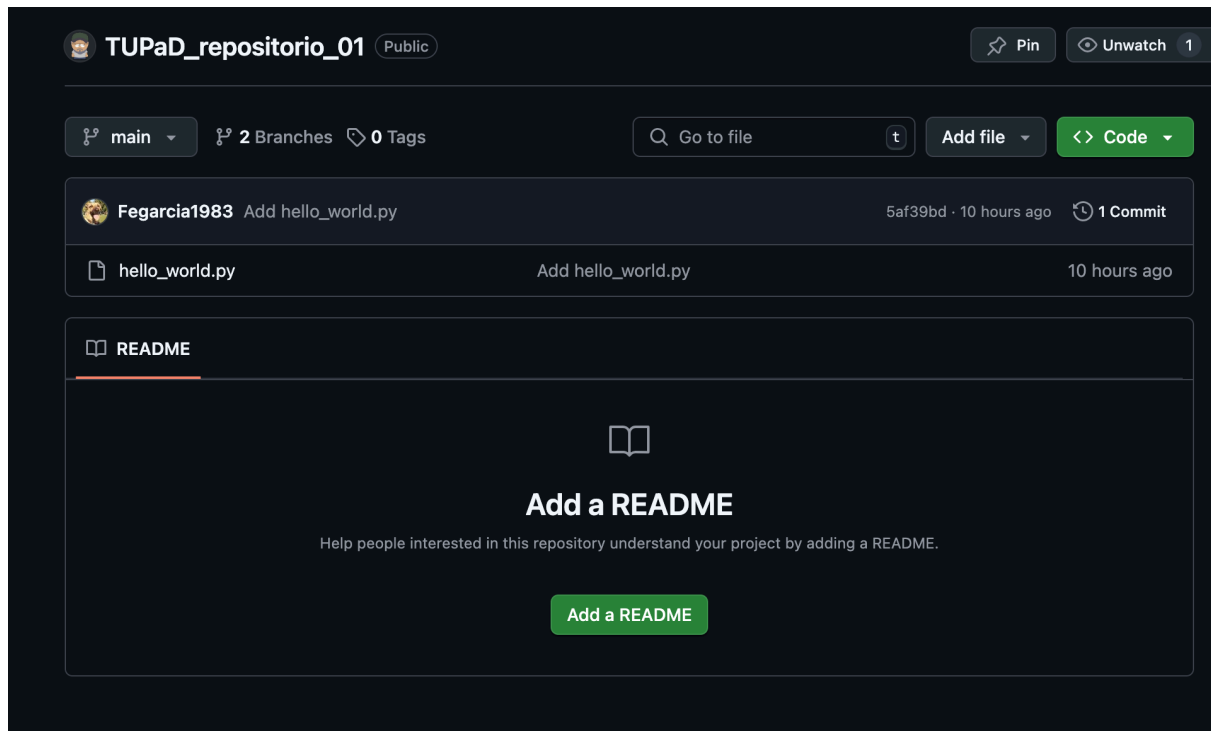
- permite llevar estos cambios locales a la nube con el fin de poder trabajar de manera colaborativa o en diferentes dispositivos.
9. Para agregar un repositorio remoto a Git se utiliza el comando `remote`. Por lo general es una tarea que se realiza una sola vez en cada repositorio. El comando es **`git remote add origin [URL_DEL_REPOSITORIO]`**.
 10. Para hacer push de los commits al repositorio remoto se debe usar el comando `push`. Se puede hacer **`git push origin [NOMBRE_DE_LA_RAMA]`** o simplemente **`git push`**, si anteriormente se utilizó el comando **`git push -u origin [NOMBRE_DE_LA_RAMA]`**.
 11. Para traer cambios de un repositorio remoto se utiliza el comando `pull`. El comando es **`git pull origin [NOMBRE_DE_LA_RAMA]`**, al igual que en el caso de push, si en la rama se utilizó el flag `-u` al momento de hacer un push, con hacer simplemente **`git pull`** alcanzaría.
 12. Al hacer un fork estamos realizando una copia de un repositorio de un tercero a nuestra cuenta. Esto nos permite poder modificar el código de terceros, para hacer adaptaciones según nuestras preferencias o también para aportar al proyecto del tercero en un futuro.
 13. Para crear un fork se puede ingresar al repositorio del cual queremos hacer el fork desde GitHub y arriba a la derecha siempre hay un botón para poder hacer el fork del mismo. Claramente debemos estar logueados inicialmente en GitHub.
 14. Normalmente en Git y GitHub se trabaja en ramas para poder hacerlo de manera colaborativa. Una vez que desde una rama se commitearon todos los cambios que queremos realizar y estos se pushearón, desde GitHub se puede crear lo que se llama un pull request. Este comando permite hacer un merge de las dos ramas en el repositorio remoto.
 15. Estando en GitHub aparece un aviso de que hay un pull request pendiente de ser mergeado. Si no hay conflictos este se podrá mergear normalmente. De haberlos es necesario resolverlos antes de que se mergee el pull request.
 16. Una etiqueta o tag es un puntero que señala a un commit en específico. Como no pueden ser modificados son utilizados principalmente para el versionado de software o para marcar hitos importantes en la vida del repositorio.
 17. Hay un comando `tag` que nos permite crear tags en Git. Este se puede utilizar sin flags o con flags del estilo `-a` y `-m` los cuales hacen referencia a un tag anotado y al mensaje del mismo (parecido a cuando se hace un commit). Ejemplos del comando **`git tag [NOMBRE_DEL_TAG]`**, **`git tag -a [NOMBRE_DEL_TAG] -m "[MENSAJE]"`**.
 18. Para enviar un tag a GitHub se puede utilizar el mismo comando push que se utiliza para enviar los commits. Se puede usar de la misma manera, pero en vez de poner el nombre de la rama se utiliza el nombre del tag. Por ejemplo: **`git push origin [NOMBRE_DEL_TAG]`**.

También se pueden enviar todos los tags juntos con el comando **git push origin -tags**.

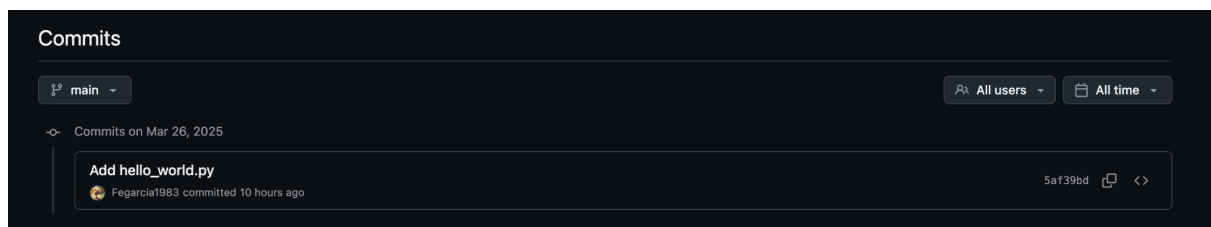
- 19.El historial de git es un listado de los commits realizados en el repositorio. Este historial se puede ver con el comando **git log**, este comando tira un listado de todos los commits con datos como el autor del mismo, el hash correspondiente, la fecha de creación y el mensaje de cada uno.
 - a. Para borrar el historial de git basta con borrar la carpeta .git que se genera al inicializar un repo.
- 20.Un repositorio privado en GitHub es un repositorio que no se encuentra visible o accesible para cualquier usuario dentro de GitHub. Solamente se puede acceder al mismo por medio de una invitación.
- 21.Para crear un repositorio privado en GitHub basta simplemente con tildar la opción de *repositorio privado* al momento de crear el mismo. Por default esta opción suele estar tildada en *repositorio público*.
- 22.Para invitar a alguien a un repositorio privado debemos ingresar al tab *settings* del repo en cuestión y en el navbar a la izquierda vamos a la sección *colaboradores*. Ahí se puede enviar una invitación por mail o nombre de usuario de GitHub.
- 23.Un repositorio público de GitHub es un repositorio que se encuentra visible a cualquier usuario que tenga una cuenta de GitHub. Esto les permite clonar o hacer un fork de este mismo sin necesidad de pedir autorización.
- 24.Para crear un repositorio público basta con crear un repositorio nuevo, por default la opción de repositorio público es la que está seleccionada.
- 25.Para compartir un repositorio público de GitHub es suficiente compartir la url del repositorio.

Trabajando con Git y GitHub [Actividad 2]:

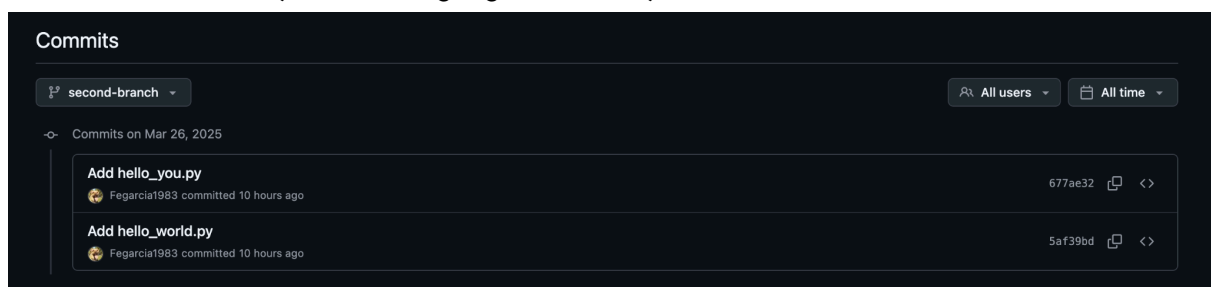
1- crea un repositorio



2- sube un archivo (archivo subido a rama main)

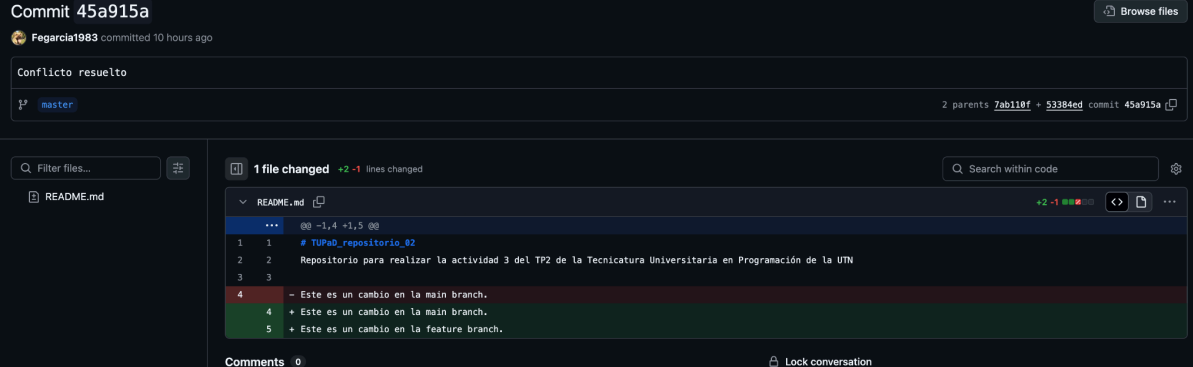


3- crea un branch, sube o agrega archivo, sube el branch



Resolviendo conflictos [Actividad 2]:

Pasos similares al anterior, pero en vez de agregar un archivo, modifiqué la línea 4 del archivo README.MD en ambas ramas antes de hacer un merge. Eso generó conflictos. Adjunto commit de resolución, donde me quedé con ambos cambios:



The screenshot shows a GitHub commit page for commit 45a915a by user Fegarcia1983, committed 10 hours ago. The commit message is "Conflicto resuelto". The commit details show 2 parents: 7ab110f and 53384ed. The file list shows 1 file changed (README.md) with +2 lines and -1 line changed. The diff view for README.md shows the following changes:

```
@@ -1,4 +1,5 @@
1 1 # TUPaD_repositorio_#2
2 2 Repositorio para realizar la actividad 3 del TP2 de la Tecnicatura Universitaria en Programación de la UTN
3 3
4 - Este es un cambio en la main branch.
4 + Este es un cambio en la main branch.
5 + Este es un cambio en la feature branch.
```

The diff view also shows a "Lock conversation" button at the bottom right.