# Práctico 2: Git y GitHub

## Objetivo:

El estudiante desarrollará competencias para trabajar con Git y GitHub,

Aplicando conceptos fundamentales de control de versiones, colaboración

En proyectos y resolución de conflictos, en un entorno simulado y guiado

Alumno: Matias Manuel Carro

### Actividades

**1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada**

**(Desarrollar las respuestas):**

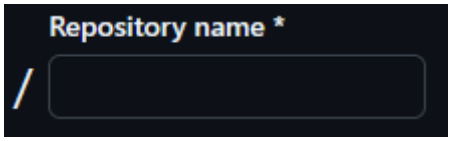
**• ¿Qué es GitHub?**

GitHub es una plataforma de desarrollo colaborativo, que se utiliza para alojar proyectos usando el sistema de control de versiones Git. En esta plataforma se alojan los proyectos creados por el usuario, además de las diferentes versiones que se crearon, las diferentes ramas que el usuario decide crear y un historial de cada una de las versiones subidas.

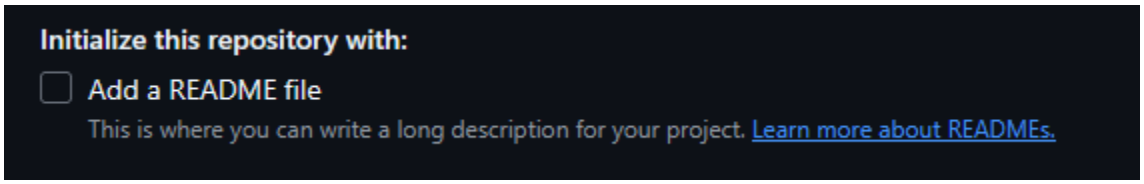
**• ¿Cómo crear un repositorio en GitHub?**

Para crear un repositorio en GitHub primero debemos tener una cuenta creada en GitHub, luego desde la página principal de nuestro perfil utilizamos el botón “New”

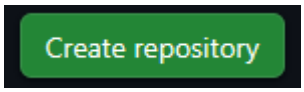

Como siguiente paso se le asigna un nombre al repositorio, en la opción “repository name”




De manera opcional se el puede agregar una descripción. También tenemos la opción de iniciar el repositorio con un README, donde podremos escribir de que se trata el proyecto / repositorio e información sobre el creador o el conjunto de creadores




Finalmente, lo creamos con el botón “Create Repository”




**• ¿Cómo crear una rama en Git?**

Para crear una rama en Git, desde la consola de Git Bash, utilizamos el comando:

*git branch* ***nombre-rama***

Donde “nombre-rama” será el nombre que deseamos nombrar la nueva rama

**• ¿Cómo cambiar a una rama en Git?**

Para cambiar de rama en git utilizamos el comando:

*git checkout* ***rama***

Donde “rama” es el nombre de la rama a la cual queremos cambiar

**• ¿Cómo fusionar ramas en Git?**

Se fusionan las ramas con el comando:

*git merge* ***rama***

Este comando fusiona la rama que pusimos en el código, con la rama actual en la cual estamos trabajando.

Por ejemplo, si estamos trabajando en la rama **“main”** y hacemos un merge con la rama **“sub”**, el contenido de la rama **“sub”** pasaría a estar dentro del contenido de la rama **“main”**

**• ¿Cómo crear un commit en Git?**

Se crea un commit con el comando:

*git commit -m* ***“mensaje”***

Esto confirma los cambios hechos en el proyecto o repositorio, en “mensaje” va el comentario o titulo que se quiera poner al cambio realizado.

**• ¿Cómo enviar un commit a GitHub?**

*git push*

Tener en cuenta que la primera vez que se va a hacer este "push" se debe utilizar el comando: *git push -u origin main* el comando *-u* crea la referencia a cual branch (en este ejemplo siendo la rama main) queremos realizar el push, una vez utilizado, se puede trabajar solamente con *git push* sin necesidad de especificar nuevamente todo

**• ¿Qué es un repositorio remoto?**

Un repositorio remoto son las versiones que subimos a GitHub, con el comando push. Hasta que no suban los cambios, los commit que realizamos se guardan dentro de la carpeta .git, de manera local en la pc que estemos trabajando.

**• ¿Cómo agregar un repositorio remoto a Git?**

Para agregar un repositorio remoto, se utiliza el comando

*git remote add origin **url***

donde dice URL, se copia y pega la dirección https de tu repositorio (que tiene que terminar en .git, la misma se consigue desde el botón de "<> code)

**• ¿Cómo empujar cambios a un repositorio remoto?**

Una vez que tengamos configurado el repositorio remoto, y tengamos agregados los archivos al stage (con el comando *git add .)* y hayamos confirmado los cambios (con *git commit)* para empujar los cambios utilizamos el comando *git push* (si es que ya configuramos previamente el git push con *git push -u origin main)*

**• ¿Cómo tirar de cambios de un repositorio remoto?**

Para obtener los cambios desde un repositorio remoto, usamos el comando *git pull*

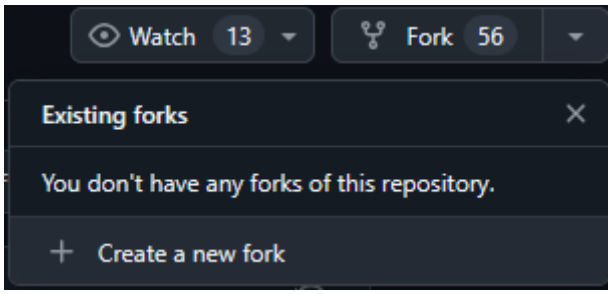
Siempre y cuando ya hayamos utilizado *-u origin main* durante el push, si no se debe utilizar el comando *git pull origin main*

**• ¿Qué es un fork de repositorio?**

Se trata de una copia que realizamos de otro repositorio que podemos modificar y hacer públicos tus cambios al proyecto original. Creando una versión distinta propia del proyecto.

**• ¿Cómo crear un fork de un repositorio?**

Para crear una bifurcación de un proyecto debemos ir al proyecto que queremos utilizar y arriba a la derecha se encuentra la opción de "fork" para que podamos crear nuestra propia bifurcación. Se hace desde la opción " + Create new fork"




**• ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?**

Una vez que tengamos el fork creado del repositorio,  realizamos una nueva rama del mismo. Con el código:

*git checkout -b "nueva-rama"* siendo nueva rama el nombre que queremos ponerle

una vez realizados los cambios, continuamos con los códigos:

*git add .*

*git commit -m "Nombre del cambio"*

Una vez terminado el commit, realizamos el nuevo push

*push push origin nueva-rama*

Al tener el nuevo update subido a github, vamos a tener el aviso de que la nueva rama tuvo un push nuevo,  con un botón que dice "compare & pull request"


Una vez dentro de este menú, tenemos que solicitar el branch al cual le queremos agregar los cambios realizados en la nueva rama.
debajo ponemos el titulo del update que realizamos y una descripción de los cambios.

Al tener todo listo, podemos crear el pull request


Una vez aceptado, se puede hacer un merge de la rama nueva hacia la rama main.

**• ¿Cómo aceptar una solicitud de extracción?**

Para realizarlo tenemos que buscar los pull request que fueron solicitados en el repositorio,

Ahí vamos a acceder a las solicitudes para poder revisar los cambios que se enviaron

Una vez dentro podemos comparar los cambios propuestos con el código del main branch

Escribimos un comentario sobre los cambios propuestos

Y seleccionamos aprobar la combinación de cambios propuestos, una vez listo, podemos realizar el merge con el botón “Merge pull request”

Para finalizar y completar el merge, seleccionamos “Confirm Merge”

- **¿Qué es un etiqueta en Git?**

La etiquetas se utilizan para marcar una versión especifica de nuestro proyecto, por ejemplo. Nombrar la versión 1.0 para poder después encontrarla fácilmente

- **¿Cómo crear una etiqueta en Git?**

Se crea con el comando

*git tag nombre*

también, se puede crear un tag con un mensaje

*git tag -a nombre -m mensaje*

- **¿Cómo enviar una etiqueta a GitHub?**

Una etiqueta se envia junto el push, con el comando

*git push –tags*

- **¿Qué es un historial de Git?**

Es el historial de los commit realizado del proyecto, estos incluyen el autor, la fecha y horario y el mensaje.

- **¿Cómo ver el historial de Git?**

Lo podemos ver desde la consola con *git log*

- **¿Cómo buscar en el historial de Git?**

Se busca con el comando: *git log --grep=”búsqueda”*

En búsqueda ponemos el mensaje que queremos buscar.

- **¿Cómo borrar el historial de Git?**

Para eliminar el historial de git, se utiliza el comando *git reset --hard HEAD~n*

*Donde n es el numero de commits que deseas eliminar, partiendo desde el ultimo hacia atrás.*

**• ¿Qué es un repositorio privado en GitHub?**

Es un repositorio que solo es visible para los usuarios autorizado

**• ¿Cómo crear un repositorio privado en GitHub?**

Al crear un repositorio, se elije la opción de repositorio privado

**• ¿Cómo invitar a alguien a un repositorio privado en GitHub?**

Dentro de “settings” del repositorio, en la opción “Collaborators” se encuentra la ventana de “Manage Access” donde se pueden agregar colaboradores al repositorio

**• ¿Qué es un repositorio público en GitHub?**

Es un repositorio al que todos los que visitan la pagina tienen acceso. Para podes utilizar el código, modificarlo o compartirlo.

**• ¿Cómo crear un repositorio público en GitHub?**

Es la opción por defecto al crear un repositorio, se debe seleccionar repositorio público

**• ¿Cómo compartir un repositorio público en GitHub?**

Para que puedan realizar modificaciones se realiza de la misma manera que uno privado, si es para que lo visualicen, se puede pasar directamente la URL de la pagina del repositorio

**2)**

**El ejercicio de la actividad 2 se encuentra realizado en el siguiente repositorio:**

**https://github.com/MatiasManuelCarro/UTN-TUPaD-Repositorio-TP2**

**3)**

**El ejercicio de la actividad 3 se encuentra realizado en el siguiente repositorio:**

**https://github.com/MatiasManuelCarro/conflict-exercise**