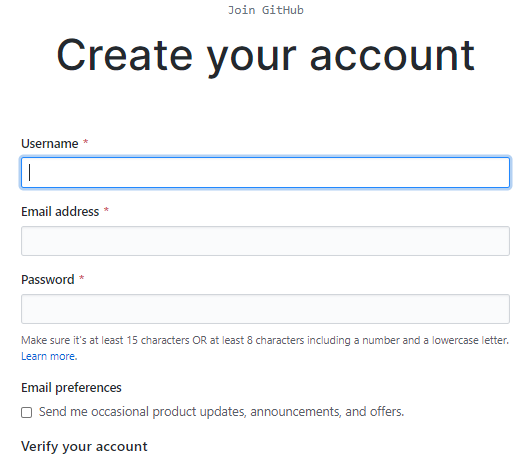
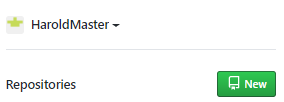
**TUTORIAL BÁSICO PARA GITHUB**

Antes de seguir este tutorial es necesario que conozcas la funcionalidad básica de GIT.

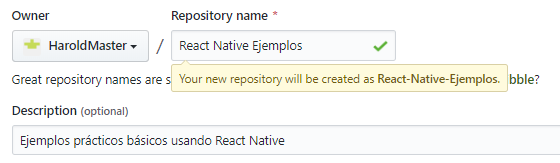
1. Primero debemos registrarnos en la página principal de Github.com



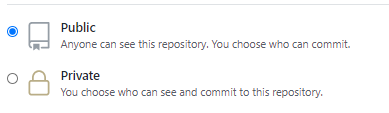
1. Ahora nos ubicamos en la parte izquierda superior de la pagina principal y señalamos New



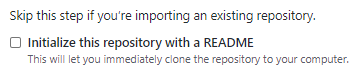
1. En la nueva página podemos llenar los campos de la siguiente manera:
   1. Comenzamos dándole un nombre único a nuestro proyecto. Podemos añadir también una descripción de lo que trata.



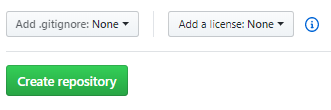
* 1. El repositorio tendrá por defecto marcado la casilla de público, es decir que cualquier miembro de Github podrá ver tu código. Si la casilla privada es marcada solo podrá ser visto por las personas que tu elijas.



* 1. En la parte inferior podemos elegir si queremos que el proyecto se cree con un README, esta opción esta desmarcada por defecto y así la dejaremos debido a que si la seleccionamos se hará un commit inicial automáticamente y podemos tener problemas en nuestro primer push. Es una buena práctica agregar un README para describir de mejor manera al proyecto, explicar las tecnologías contenidas, el uso, etc. No lo creamos desde el inicio pero podemos agregarlo después con la extensión de archivo .md.

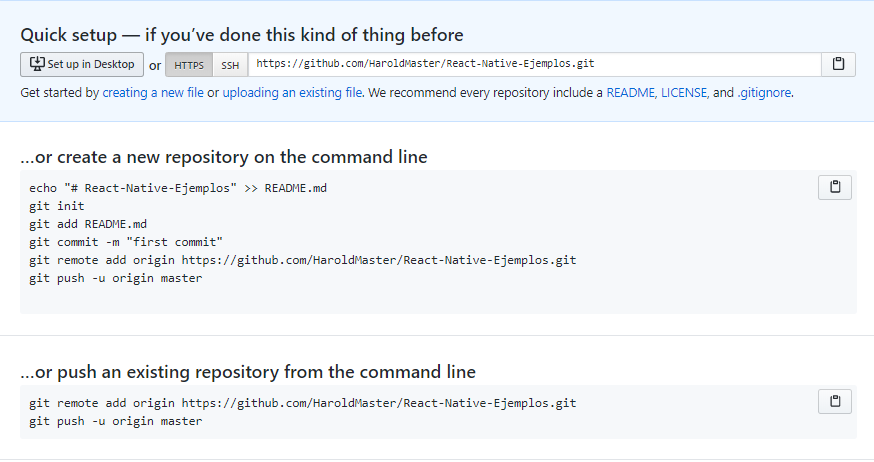


* 1. También puedes elegir si agregar un archivo .gitignore y una licencia para tu proyecto. Finalmente creamos el repositorio.

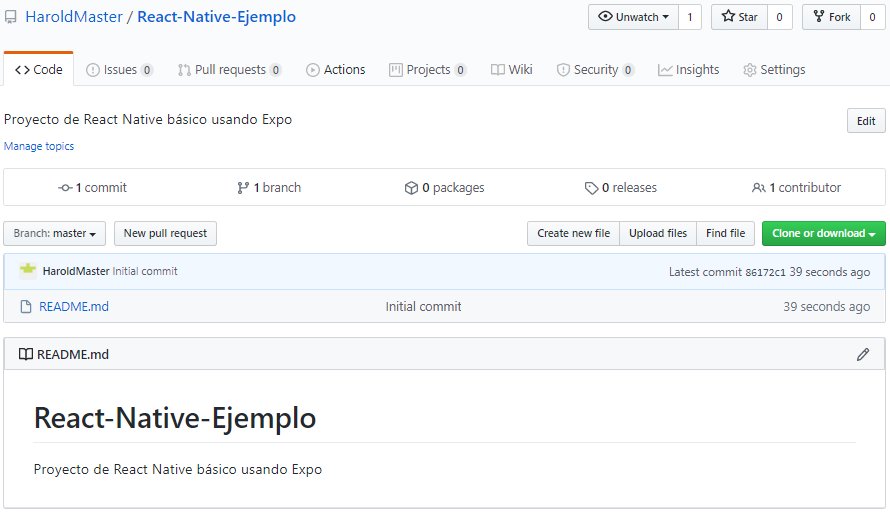


Para más información de las licencias puedes visitar el siguiente link: [https://help.Github.com/es/Github/creating-cloning-and-archiving-repositories/licensing-a-repository](https://help.github.com/es/github/creating-cloning-and-archiving-repositories/licensing-a-repository)

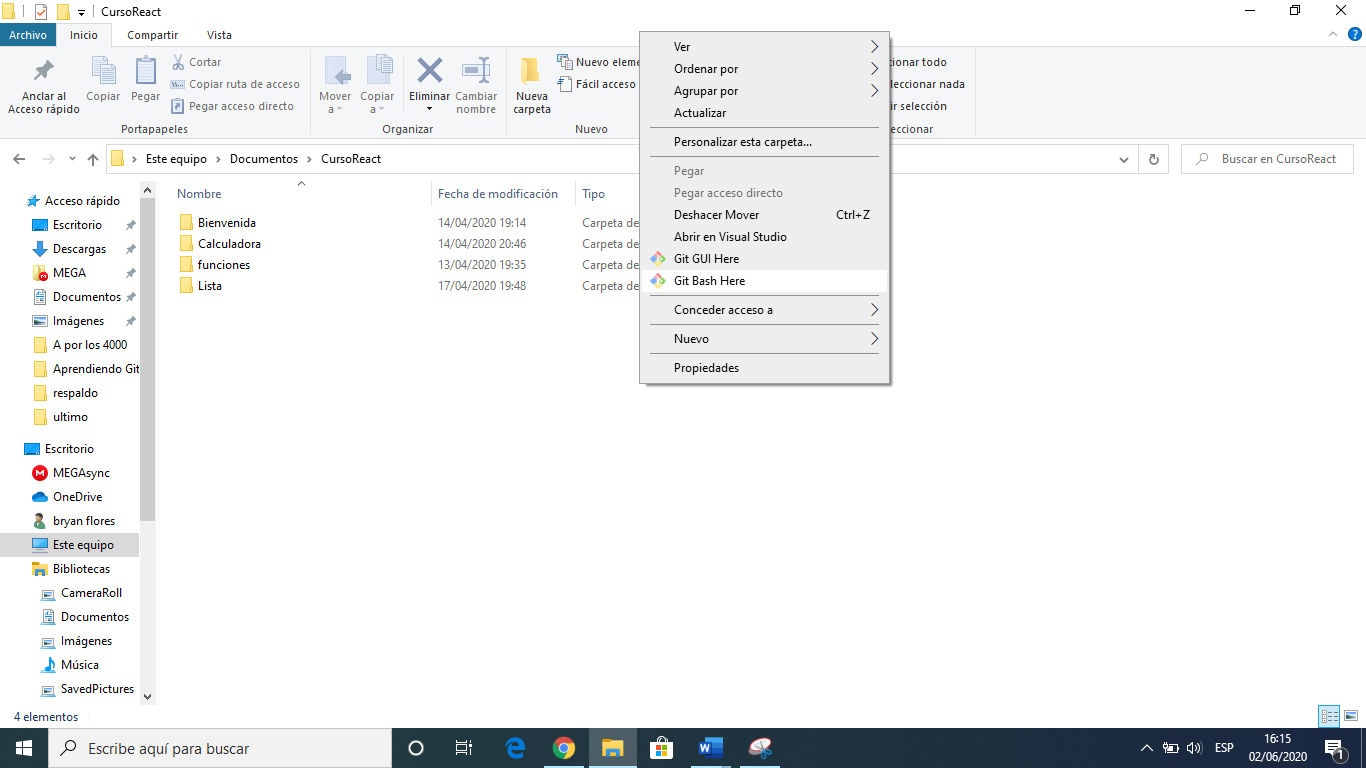
* 1. Después de crear el repositorio se nos mostrará una página con los pasos a seguir para subir el repositorio a Github.



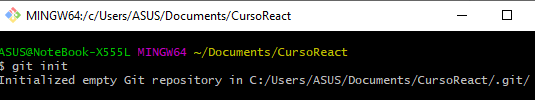
* 1. Si agregamos el archivo README en el momento de la creación del repositorio, automáticamente nos lo detectará como un commit en el proyecto y la página se verá de esta manera:



1. Ahora para llevar el proyecto desde el repositorio en git de mi computador, hacia el Github debo hacer los siguientes pasos:
   1. Abro el git bash en la carpeta del proyecto.



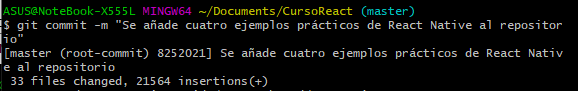
* 1. Creo el repositorio con el comando git init



* 1. Agrego los documentos al staging area mediante el comando git add .



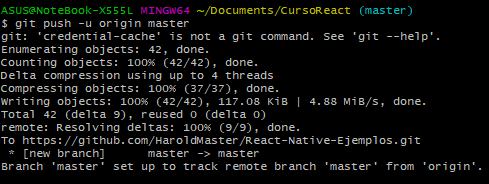
* 1. Realizo el primer git commit del proyecto



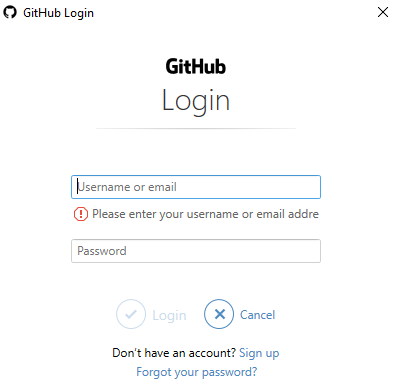
* 1. Añadimos el origen de nuestro proyecto mediante el comando git remote add origin [UrldelRepositorio.git]



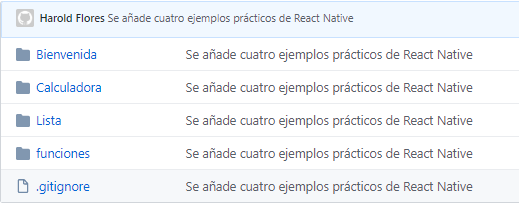
* 1. Por ultimo haremos un git push -u origin master



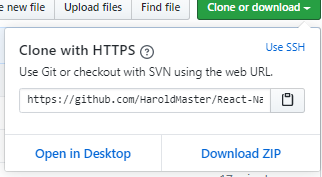
La primera vez nos pedirá que nos autentiquemos en el Github.



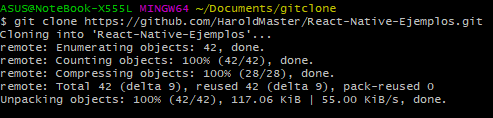
* + 1. Si creamos el README desde el principio no se hará el push normalmente debido a que en el repositorio de Github hay un archivo que en el repositorio local no. Para solucionar eso tenemos que hacer un git pull origin master --allow-unrelated-histories, esto traerá el README de Github al computador, después continuamos desde el paso 4.3.
  1. Finalmente ya podremos ver el repositorio de Github con los archivos agregados.

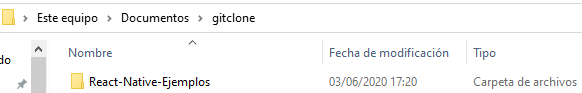


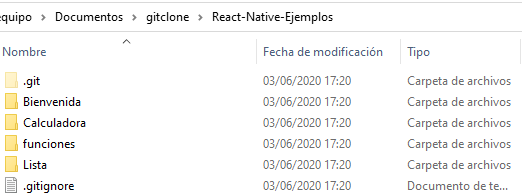
1. Si quiero tener los archivos de mi repositorio Github en otro computador, puedo clonar el repositorio. Se lo puede hacer de dos maneras:
   1. Descargándolo desde la página web de Github



* 1. Desde la terminal haciendo un git clone [url] ubicándonos dentro de la carpeta destino de los archivos



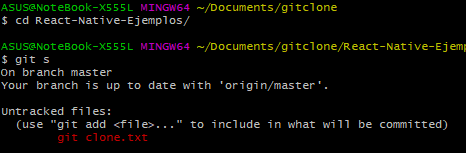




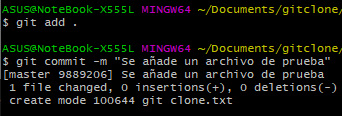
* 1. Puedo realizar cualquier cambio y subirlo al repositorio normalmente.
* A modo de ejemplo creamos el archivo git clone



* Ingresamos a la carpeta del repositorio, verificamos el estado del repositorio y observamos que el nuevo archivo está en rojo debido a que no está añadido al repositorio local.



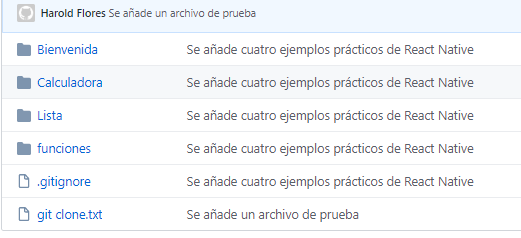
* Añadimos el nuevo archivo al repositorio



* Realizamos un push al repositorio del Github.



* Finalmente ya podemos observar los cambios en el repositorio del Github.

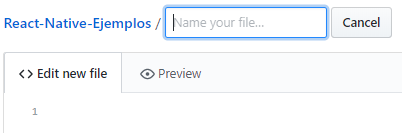


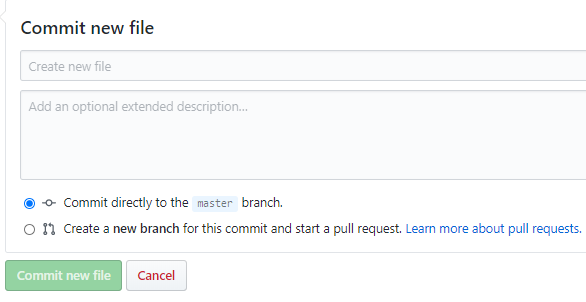
1. La interfaz de Github es muy grande y completa, pero los elementos básicos que debes conocer son los siguientes:

* En la parte derecha de la interfaz encontramos 4 botones.

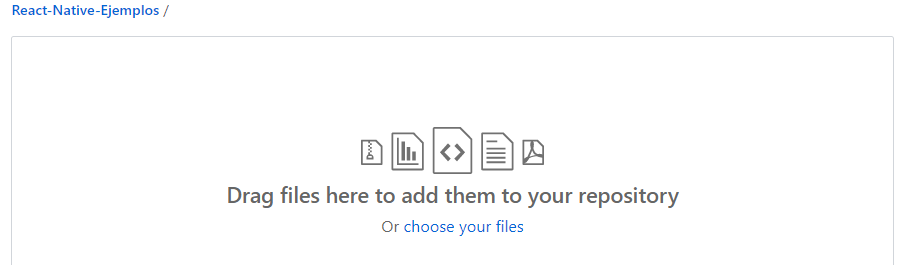


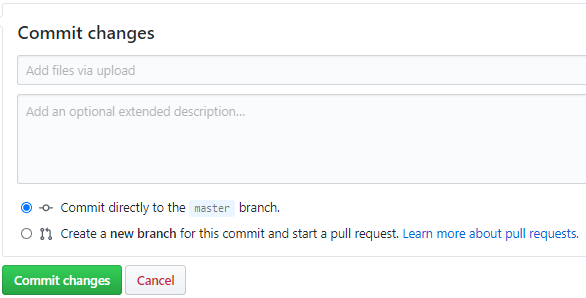
* + Crear un nuevo archivo, el cual no es muy usado debido a que los archivos deben crearse o modificarse desde el ordenador, pero si se hace uso de este botón se creará un archivo en blanco al cual podemos también añadirle contenido y también hacerle un commit debido a que es un cambio.



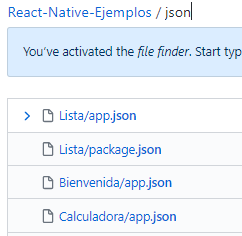
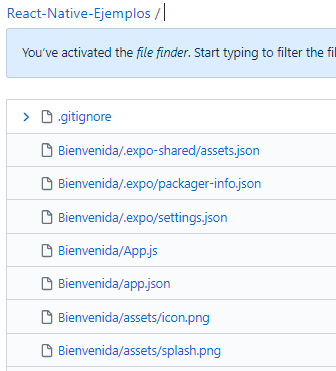


* + Cargar archivos te permite pasar archivos desde el pc al repositorio de Github con el respectivo commit.





* + El botón encontrar archivos te presenta el árbol de directorios de todos los archivos contenidos en el proyecto, así mismo implementa un buscador para encontrar archivos específicos.



* En la parte izquierda de la interfáz encontramos los botones para mostrar las ramas de nuestro proyecto y la solicitud de pull request. El primer boton es útil cuando manejamos 2 o más ramas de nuestro proyecto.

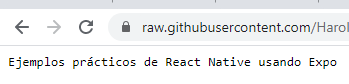
Cuando queremos modificar un proyecto que no es nuestro podemos hacer un fork para traer una copia de su repositorio al nuestro y una vez terminadas nuestras implemetaciones podemos solicitar al dueño del repositorio original que añada nuestros cambios a su repositorio, esta solicitud se la hace a través de un pull request el cual será revisado por el autor y aprobado o declinado dependiendo de su criterio.



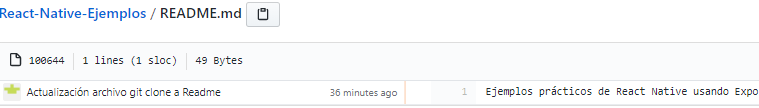
* Cuando seleccionamos un archivo dentro del Github, se nos presentan estas tres opciones:



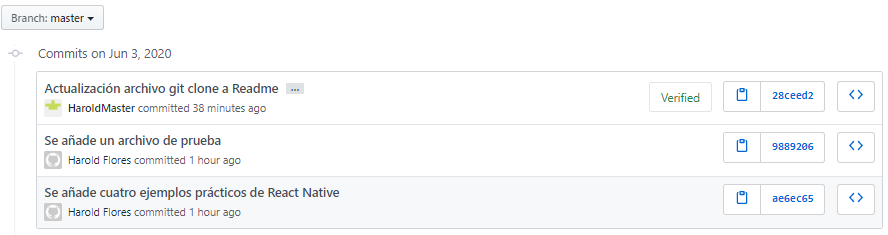
* + Raw abre el contenido del documento seleccionado.

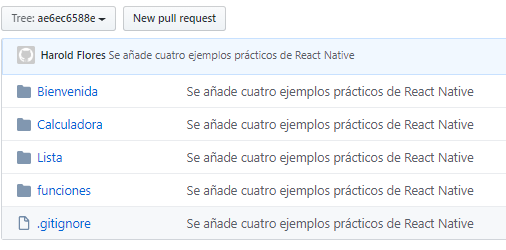


* + Blame muestra los cambios del archivo a través del tiempo con los respectivos commits.

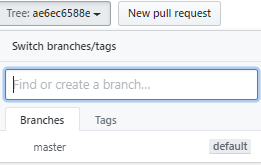


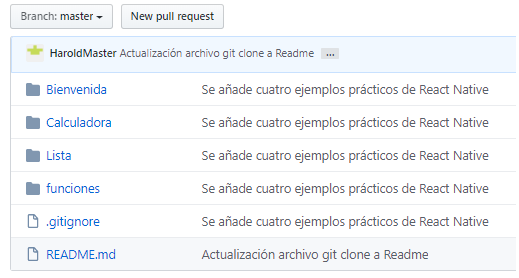
* + History muestra mas a detalle los commits realizados, por ejemplo si voy a mi primer commit, cuando no tenía agregado el archivo readme, este no me aparecerá en mi rama principal debido a que hasta ese entonces no existía.





Para salir de este commit basta con seleccionar master dentro del botón que muestra las ramas.

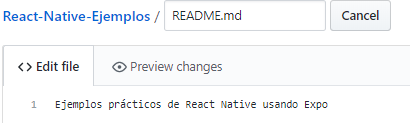




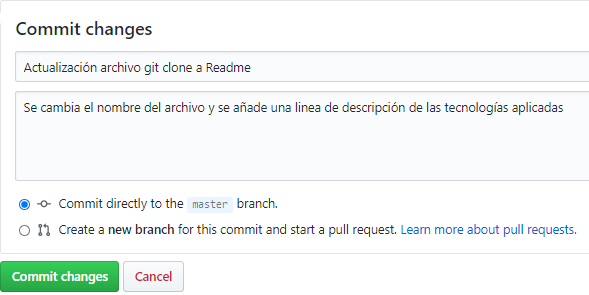
1. Si por cualquier motivo se realizó un cambio en el repositorio de Github y quieres que se vea reflejado en el repositorio local, es necesario hacer un git pull, este comando traerá los cambios de Github al computador.

* A modo de ejemplo cambiaremos el archivo gitclone.txt y lo cambiaremos por un archivo README.md desde el repositorio en Github.

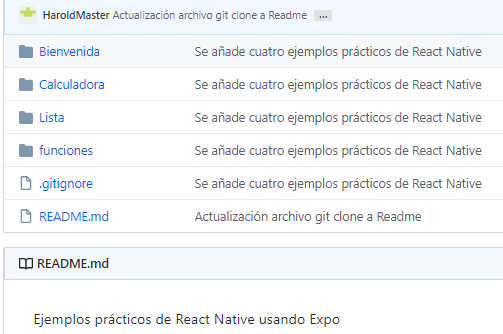




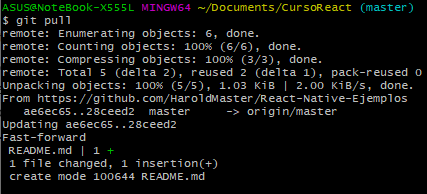
* Llenamos los datos del nuevo commit, ya que es un cambio que debe estar registrado.

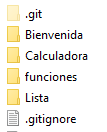
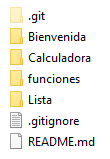


* Una vez realizados los cambios podemos ver ya los archivos actualizados en nuestra rama master.



* Estos cambios fueron realizados desde Github, pero el repositorio local de la computadora no refleja automáticamente estos cambios. Es aquí cuando el comando git pull nos ayuda a traer estos nuevos cambios desde Github hacia el repositorio local.



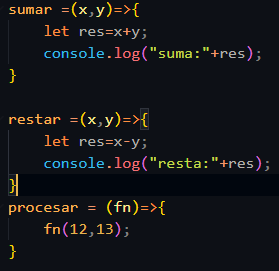
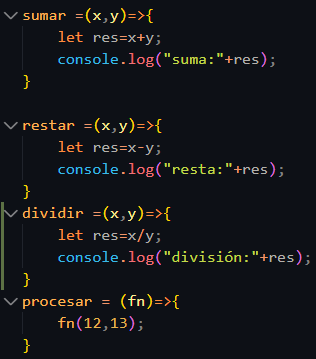


1. En el tutorial de git vimos como gestionar las ramas de manera local, pero ahora veremos como crearlas y eliminarlas desde nuestro Github. Podemos crear las ramas de dos maneras: La primera creando la rama desde el git local y luego hacer un push al Github y la segunda creando la rama desde el Github y haciendo un pull hacia el repositorio local.

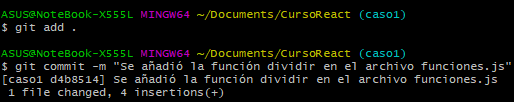
* En el primer ejemplo crearemos una nueva rama desde el git local y la trasladaremos al Github.
  + Añadimos la nueva rama.



* + Realizamos un cambio en la rama, en este caso modificaré el archivo funciones.js.



* + Guardamos y lo agregamos al repositorio local.

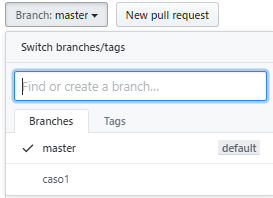


* + Hacemos un push pero en vez de hacerlo al origen master lo hacemos con el nombre de la rama. Esto creará automáticamente la rama dentro de Github con los cambios correspondientes.

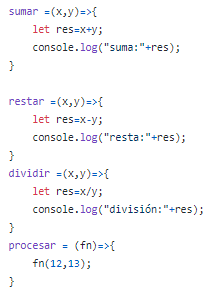


Gihub:

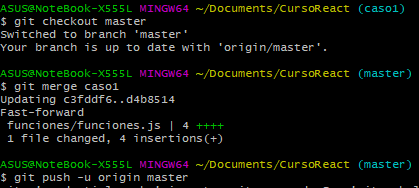






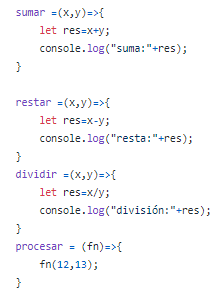


* + Ahora para unir las ramas, nos dirigimos al repositorio local, y haremos un git merge [Rama] desde master y realizamos un git push -u origin master.



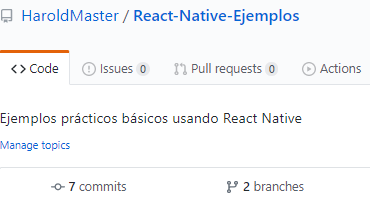
Github:

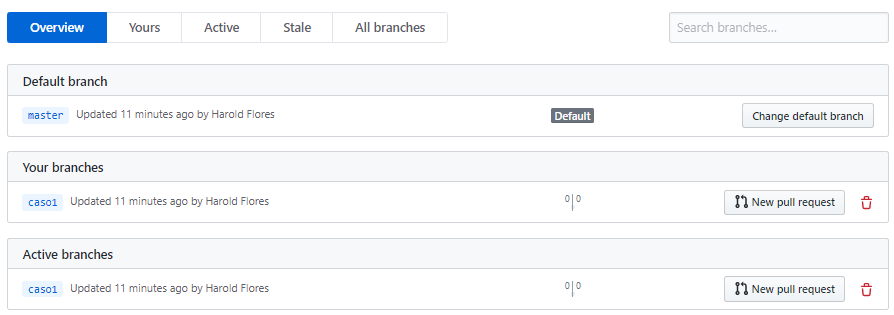




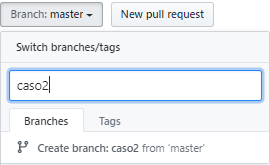
* + Finalmente si ya no necesitamos las ramas y queremos borrarlas, debemos hacerlo tanto del repositorio local como del Github, pero por separado ya que quitar la rama del git local no significa que la rama del Github haya sido eliminada.



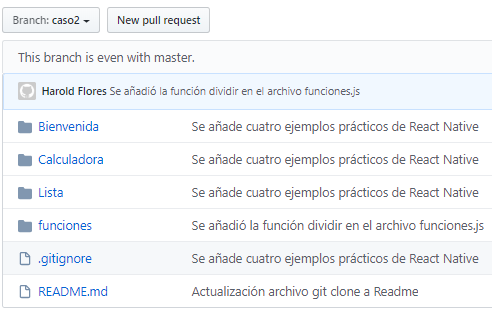




* En el segundo ejemplo crearemos la rama desde el Github y su contenido la trasladaremos al git local.
  + Nos ubicaremos en el Github en el botón de despliegue de la ramas y en el buscador digitamos el nombre de la nueva rama.



* + Automáticamente se creará la nueva rama con todo el contenido de la rama master.

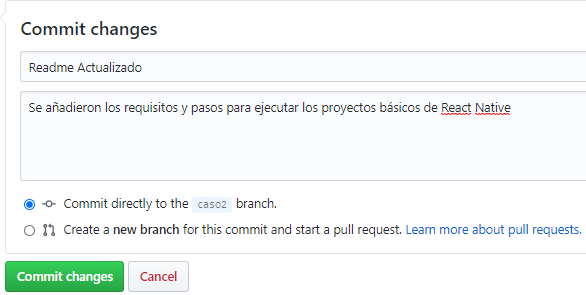


* + Para traer el contenido de esta rama al repositorio local, abrimos el git bash en la rama master y creamos una nueva rama con el mismo nombre. Es importante que tenga el mismo nombre debido a que si en un futuro hacemos push a la rama dará conflictos.

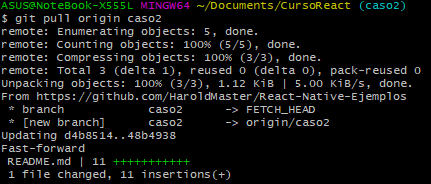


* + - Si por error creaste la rama con otro nombre, puedes modificarlo desde la misma rama mediante el comando git branch -m [NuevoNombre]
  + Si tenemos una modificación en el Github y queremos traer los cambios a la rama local, usamos el git pull origin [Rama]
    - Para ejemplificar este caso, llenaremos el archivo readme de la nueva rama con nueva información. Y hacemos el commit.

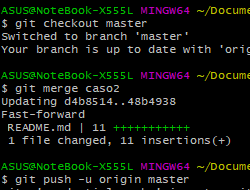




* + - Para traer estos cambios al repositorio local, podemos hacer un git pull origin [Rama]



* + - Ya que puedo visualizar los cambios desde mi repositorio local, lo puedo agregar a la rama master haciendo un git merge para unirlo en el repositorio local y un git push origin master para subirlo al Github.



* + - Ahora ya tengo el commit de la rama caso2 en mi rama master tanto del repositorio local como del Github.

