Certified Tech Developer
The Ultimate Degree

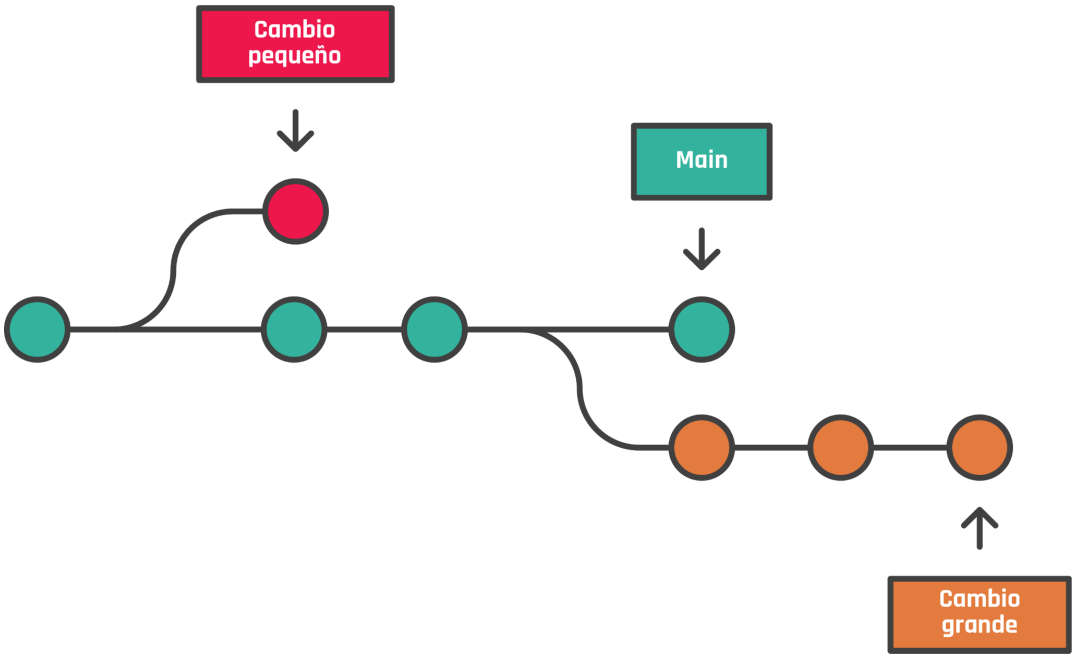
Ramas de Git

¿Qué es y cómo se utiliza una branch?

Una **rama o branch de Git** nos permitirá trabajar en versiones de nuestros archivos de manera mucho más ordenada. Hasta ahora vimos que el repositorio que creamos con Git o clonamos de Github tiene por defecto una rama raíz llamada **Main** o **Master** —dependiendo de la versión de Git— en la cual vamos guardando las versiones de nuestro trabajo cada vez que hacemos un commit.

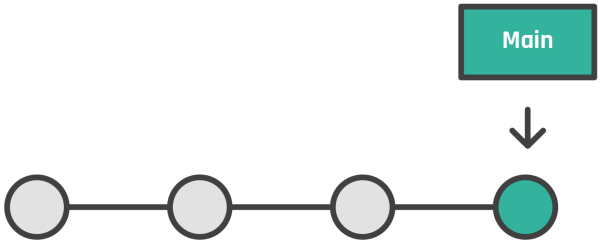
En los proyectos que realizaremos como desarrolladores, todos los cambios que hagamos en un repositorio deben ser socializados con un equipo. Ese equipo será el encargado de aprobar nuestras modificaciones. Entonces, cuando trabajemos en nuevas funcionalidades, debamos modificar errores o simplemente queramos ordenar de una mejor manera nuestros archivos, deberíamos hacerlo sobre un espacio separado. Hasta que nuestro equipo apruebe esos cambios. Para ello se utilizan las **ramas** o **branches.**

Una **branch** funcionará como un espacio de trabajo solo para nosotros, para que guardemos nuestros cambios y hagamos todas las pruebas necesarias hasta que estemos seguros de socializar esos cambios.



Creación de ramas

Es importante entender que una rama es solo una línea de desarrollo que tiene como base el repositorio, pero que no actúa directamente sobre él, o sea que no lo modifica. Si comenzamos con un repositorio que tiene este aspecto:



Al crear ramas lo que generamos son **versiones del repositorio** donde están los archivos del mismo **más los cambios** que le iremos sumando.

https://drive.google.com/file/d/1E3qsLjZ5WX8hfZV9-DPHL2kqCsFwyJC8/view?usp=sharing

En el primer diagrama podemos visualizar un repositorio con dos ramas, además de la Main, una para un cambio pequeño y otra para un cambio grande. Al utilizar ramas no solo es posible trabajar en un repositorio de forma paralela, sino que se evita que subamos archivos o código dudoso a la rama Main.

Git branch

Para crear una nueva rama utilizamos el comando **git branch**. Git branch nos permite crear, enumerar, cambiar el nombre y eliminar ramas. No permite cambiar entre ramas o volver a unir un historial bifurcado.

- **`git branch`**

  Enumera todas las ramas de tu repositorio, es similar a git branch --list.

- **`git branch <branch>`**

  Crea una nueva rama llamada <branch>.

- **`git branch -d <branch>`**

  Elimina la rama llamada <branch>. Git evita que eliminemos la rama si tiene cambios que aún no se han fusionado con la rama Main.

- **`git branch -D <branch>`**

  Fuerza la eliminación de la rama especificada, incluso si tiene cambios sin fusionar.

Git checkout

Para moverse de una rama a otra, se ejecuta el comando

- `git checkout nombre_rama`

Generalmente, Git solo permitirá que nos movamos a otra rama si no tenemos cambios. Si tenemos cambios, para cambiarnos de rama, debemos:

1. Eliminarlos (deshaciendo los cambios).
2. Confirmarlos (haciendo un git commit).

Guardar cambios y subirlos al repositorio remoto

Una vez que terminamos de realizar los cambios que queremos en nuestra branch, ejecutamos los mismos comandos que vimos hasta ahora: git add, git commit, git status y git log. Pero cuando queramos subir esos cambios, debemos utilizar git push con el nombre de la rama en que estamos posicionados:

- `git push origin <branch>`

Así también, para traer los cambios de esa rama utilizamos el git pull agregando desde donde queremos traer los cambios:

- `git pull origin <branch>`

¡Hasta la próxima!