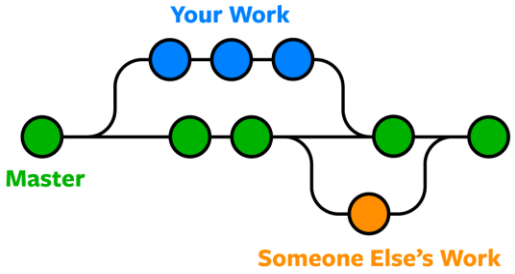
1) Diferencie la herramienta Git de Github

Git es un DVCS (sistema de control de versiones distribuido). Este es instalado y utilizado de manera local en una computadora; permite guardar, administrar y comparar diferentes versiones de un archivo.

Por otra parte, GitHub es un servicio basado en la nube en donde los usuarios pueden alojar repositorios de GitHub, permitiendo la colaboración en línea entre personas.

2) ¿Qué es un branch?

Un branch es una opción de edición de código muy funcional que posee GIT, la cual permite realizar cambios a una parte específica de un proyecto sin afectar el resto. Es decir, se encarga de aislar parte del proyecto y de esta forma no se necesita realizar una copia para editar el código. En el momento que se crea el proyecto este cuenta con un branch por defecto la cual es el Master branch o main, la cual se utiliza para escribir el código principal y es de donde saldrán los nuevos branches que se creen cuando se inicia el proyecto. Se puede apreciar mejor mediante la siguiente representación visual.

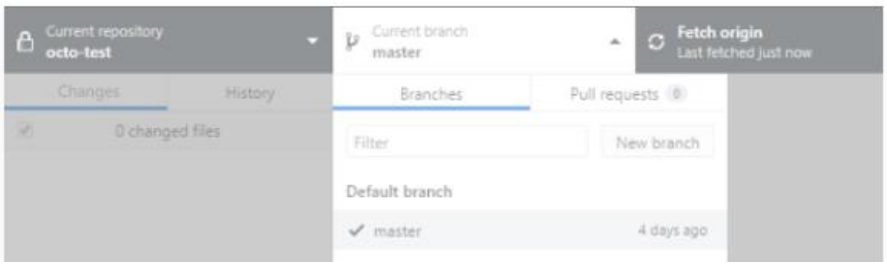


Como se ve en la imagen, los distintos branches que pueden salir de un Master Branch (Proyecto principal), estos branches pueden ser creadas por otros colaboradores como se puede apreciar. También es posible crear branches en otros branches, para no tener que afectar el progreso al realizar cambios en un Branch específico.

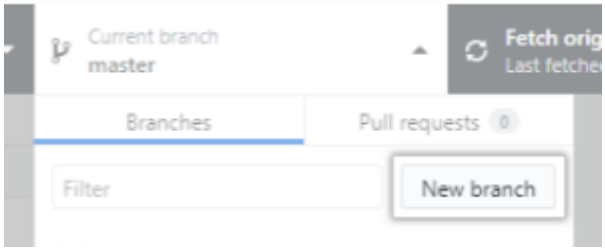
3) ¿Cómo se crea un nuevo Branch?

Para poder crear un nuevo Branch en GIT es necesario seguir la siguiente lista de pasos:

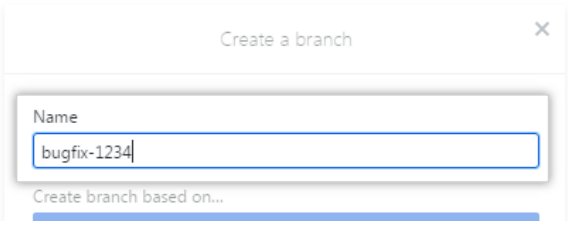
a) Primero se debe seleccionar en la parte superior de la app donde dice "Current Branch" ahí se selecciona el Branch que se desea que sea la base de su nuevo Branch, cuando se crea el proyecto solo estará la Master Branch entonces se selecciona esa al no tener otras.



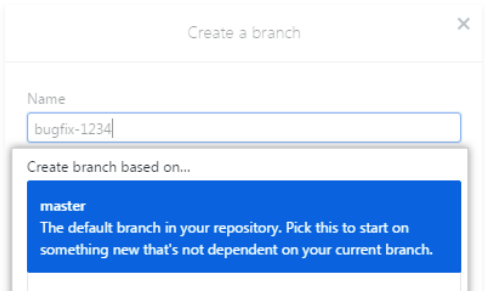
b) Luego se selecciona la opción “New Branch”.



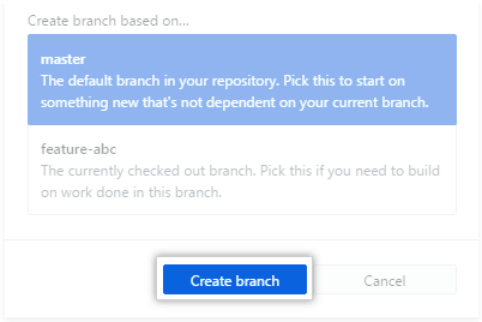
c) Se le asigna el nombre deseado a la nueva Branch.



d) Lugo en el menú desplegable se selecciona nuevamente la Branch que se desea que sea la base de la que se está creando.



e) Por último, se da click en la opción “Create Branch”.



4) ¿Qué es un commit?

Commit es un comando que captura una instantánea de los cambios preparados en ese momento del proyecto. Las instantáneas confirmadas se pueden considerar como versiones "seguras" de un proyecto. Git no las cambiará nunca a no ser que se le pida directamente. Antes de ejecutar un git commit, se utiliza el comando git add para pasar o "preparar" los cambios en el proyecto que se almacenarán en una confirmación.

5) ¿Qué es la operación “git cherry-pick”?

Esta operación permite seleccionar los cambios de uno o varios commits específicos de un branch y aplicarlos a otro branch. Puede ser utilizada para deshacer cambios; por ejemplo, un commit realizado de manera errónea en un branch, puede ser seleccionado mediante este comando y aplicarse al branch correcto.

6) Explique que es un “merge conflict” o “rebase conflict” en el contexto de tratar de hacer merge a un Pull Request o de completar una operación git rebase.

Un “merge conflict” sucede cuando los cambios realizados por un usuario entran directamente en conflicto con los cambios realizados por otro usuario, por lo que GIT no puede resolver las diferencias de manera automática generando un error, lo que si pudiese hacer en caso de que los cambios no chocaran entre sí. En el contexto de la pregunta esto sucede cuando al realizar un Pull Request la cual me permite editar el proyecto de otro usuario, los cambios que realice sean en la misma línea y en el mismo Branch en el que esté trabajando otro usuario con acceso al proyecto, por lo que ocasionará que GIT no pueda conciliar las diferencias, entonces dará la notificación de “merge conflict” para que los usuarios decidan cuál versión de los cambios debe imperar sobre la otra.

7) ¿Qué es una Prueba Unitaria o Unittest en el contexto de desarrollo de software?

En el contexto de desarrollo de software, una prueba unitaria o unittest es un método mediante el cual unidades individuales de código son sometidas a distintas pruebas con el fin de determinar si funcionan adecuadamente de forma individual. Esto asegura que pruebas futuras como de integración o de extremo a extremo no fallarán debido a que los bloques independientes de código funcionan correctamente.

8) Bajo el contexto de pytest. ¿Qué es un “assert”?

Un *assert* es la principal herramienta para realizar comprobaciones. Esto quiere decir que, si una expresión contenida dentro de un assert es falsa, se va a lanzar una excepción o un error; más concretamente un *AssertionError.* En otras palabras, es una función que va a retornar True o False, dependiendo de lo que se esté comprobando. Si se da un error o un False, el programa se va a detener ahí.

En el contexto de Pytest, esta función es muy útil para realizar test unitarios. Con el uso de assert() se pueden hacer comprobaciones de forma automática a nuestros códigos.

9) ¿Qué es Flake 8?

Flake 8 es una librería de Python que contiene PyFlakes, pycodestyle y el script McCabe de Ned Batchelder. Es un gran conjunto de herramientas que sirven para verificar un código fuente contra el PEP8, errores de programación y también para verificar la complejidad ciclomática. Se puede usar Flake 8 como una forma de limpiar el código.

10) Explique la funcionalidad de parametrización de pytest.

En Pytest se puede realizar una parametrización mediante dos métodos distintos. Uno de estos métodos es usando pytest.fixture(), con esto las funciones de decoración de accesorios se pueden pasar como parámetros a otras funciones. Por su parte, el más común es donde se utiliza pytest.mark.parametrize(), en donde dentro del paréntesis se pone el nombre del parámetro y la lista de datos; de la siguiente manera: @ pytest.mark.parametrize (“nombre del parámetro”, lista de datos). Con esto se permite probar múltiples escenarios usando una función; por eso, se puede escribir una función de prueba parametrizada para así asegurarse de que la validación de entrada funcione. Esta prueba intenta agregar una acción, obtiene la excepción y luego verifica que sea la excepción correcta; si no se obtiene una excepción, falla la prueba.