Tarea 1

GitHub, Pytest y Flake 8

Samuel Castro Villalobos | 2018089991

Leisel Villalobos Quesada | 2018135440

Profesor: Rodolfo Piedra Camacho

MT-7003 Microprocesadores y microcontroladores

Preguntas Teóricas:

1) ¿Diferencie la herramienta Git de Github?

Por un lado, la herramienta “Git” consiste en un sistema de control de versiones, por ende, es empleado para gestionar los archivos de un proyecto y además ayuda a mantener el registro de la historia de cada uno de los archivos que están siendo trabajados. Esta herramienta debe ser instalada localmente. [1]

Por otro lado, “Github” es una plataforma en la web para el control de versiones que emplea la herramienta Git. Esta plataforma está basada en la nube y permite que los desarrolladores carguen archivos y encuentren códigos de otros desarrolladores. Esta también es útil para proyectos grupales en los que la plataforma permite un medio de comunicación rápido y eficaz. [1]

2) ¿Qué es un branch?

Un “Git Branch” se define como una división en dos del estado del código con la que se crea un nuevo camino para la evolución de este. Son importante debido a que permiten crear nuevas funciones dentro de un código con el fin de optimizarlo sin obstruir o cambiar el funcionamiento básico de la rama principal y se pueden crear diferentes ramas de código para llegar al mismo funcionamiento, mediante diferentes formas. Conocer su funcionamiento es esencial para utilizar la herramienta Git de forma organizada. [2]

3) ¿Qué es un commit?

Un “commit” corresponde a la acción de subir o guardar archivos tanto a la plataforma en la nube como en la aplicación local, esto corresponde a que herramienta se esté usando para el manejo del proyecto. Esta acción es necesaria para actualizar los datos del proyecto y que estos sean compartidos en la plataforma digital. [3]

4) ¿Qué es la operación cherry-pick?

Un “cherry-pick” corresponde a un comando el cual permite elegir la confirmación de una rama y aplicarla a otra. Su mayor utilidad se basa en que permite deshacer cambios generados en el código. Además, es útil a la hora de trabajar un código en forma grupal, ya que si un desarrollador crea un código que ayuda a optimizar una parte del código, otros desarrolladores pueden utilizar esta misma optimización en otros aspectos del código de forma más efectiva. [4]

5) ¿Qué hace el comando git stash?

El comando “git stash” permite almacenar temporalmente los cambios que se realizan en un código, por lo que se puede trabajar en otros aspectos de este y luego regresar para aplicar los cambios. Estos son esenciales cuando el cambio o la optimización del código no ha sido terminada y no esta lista para ser aplicada ero se debe trabajar en otras áreas del programa. [5]

6) ¿Compare las operaciones git fetch y git pull

El comando “git fetch” descarga todos los datos de un repositorio remoto hacia el repositorio local de la computadora. Su principal característica es que los cambios no se integran al trabajo que se tiene en el repositorio local, sino que los aísla y por lo tanto no manipula, destruye o afecta los archivos locales. De esta forma es posible analizar los cambios que se realizaron en el repositorio remoto respecto al local y ver si se pueden o no integrar. [6]

El comando “git pull’ es una acción más comprometedora dado que a diferencia del “git fetch” lo que hace es descargar el contenido del repositorio remoto e inmediatamente actualizar repositorio local (el HEAD actual) para que estos dos coincidan. El “git pull” es la unión de los dos comandos “git fetch” y “git merge” en uno solo, ejecutando uno después del otro respectivamente. [7]

7) Asumiendo que usted está en un Branch llamado “secundario” y su Branch principal se llama “master” ¿Qué resultado espera de hacer git rebase master? ¿Qué resultado espera de hacer git rebase origin/master?

La idea con los comandos tanto de “merge” como de “rebase” es de tomar commits de un branch y colocarlos en otro branch (generalmente el master). En el caso de “rebase” que es el que se trata en este momento, se puede apreciar de

mejor forma a partir de la figura 1 donde cada circulo representa diferentes commits. [8]

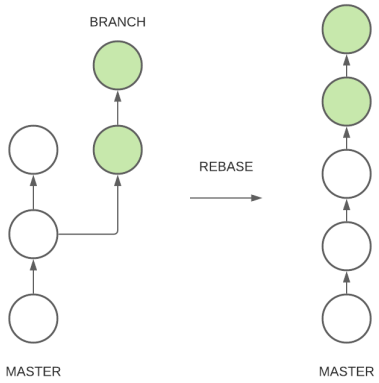

Figura 1. Comando rebase en un repositorio de forma gráfica.

De forma general, el rebase logra lo que se aprecia en la figura anterior. El branch en color verde significa el branch al que se le va a hacer el rebase y una vez hecho se puede apreciar en el mismo color como se incorpora al master. Sin embargo, para llegar aquí se debe hacer lo encontrado en la figura 2 (analizado de forma gráfica). [9]

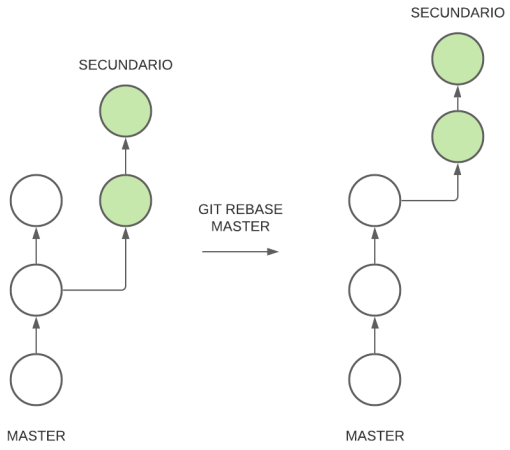

Figura 2. Comando “git rebase master” de forma gráfica.

Si se aplica el comando “git rebase master” estando en el branch secundario, lo que sucede es que los cambios/commits se “reanclan” al flujo principal (master) y los últimos commits que en este se hicieron. Algo que se aprecia fácilmente es que el branch de secundario sigue siendo como su nombre lo dice; un branch y no forma parte del flujo del master como tal, por lo que para hacer esto y que funcione como un solo flujo se debe utilizar el comando git rebase origin/master. De forma gráfica lo que hace este command se aprecia en la figura 3. [9]

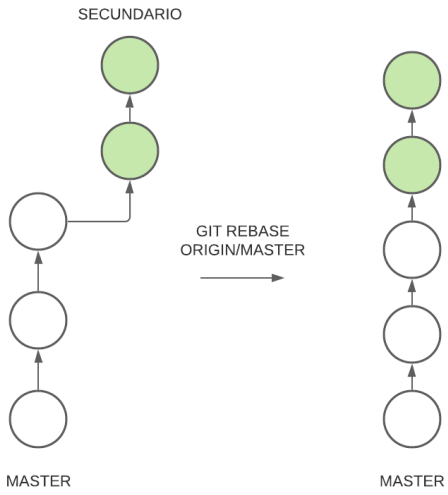

Figura 3. Comando “git rebase origin/master” de forma gráfica.

De esta forma quedan todos los cambios y commits realizados en un branch distinto integrados en el mismo flujo del master. De esta forma se facilita la comprensión del trabajo, además de que se puede modificar la cronología de los commits para algún beneficio específico. [10]

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

Lo que se pretende con las pruebas unitarias o bien el unit-testing es comprobar los componentes individuales de los programas informáticos. Aisla una parte del código y comprueba que funciona a la perfección. Se puede examinar que el funcionamiento sea el correcto para cada elemento antes de que se trabaje en el concepto más general de un programa.

Para una mejor conceptualización de lo que es el unit-testing, se puede comparar con lo que es integration-testing. Mientras que el integration-testing vela porque los distintos módulos trabajen de forma correcta cuando son combinados o utilizados en grupo, el unit-testing prueba los módulos de una aplicación de forma aislada (sin dependencias). Una característica adicional es que ayuda a comprobar de forma relativamente rápida y fácil si el componente funciona según lo previsto por el desarrollador, además es altamente automatizable. las pruebas unitarias tienen como objetivo la comprobación frecuente de diversos componentes, es por esto por lo que se realizan de forma automática. Así, con solo presionar un botón, los respectivos programas realizan varias pruebas unitarias al azar. [11]

Entre sus ventajas se encuentra:

- Permite detectar los errores a tiempo, de forma que se pueda reescribir el código o corregir errores sin necesidad de tener que volver al principio y rehacer el trabajo.
- Al realizar pruebas continuamente y detectar los errores, cuando el código está terminado, es un código limpio y de calidad. [12]
- Nos permiten detectar los errores rápidamente. Analizamos el código por partes, haciendo pequeñas pruebas y de manera periódica. [13]
- Permiten modificar partes del código sin afectar al conjunto para poder solucionar bugs.
- Gracias al continuo flujo de información y la superación de errores, se puede recopilar gran cantidad de información para evitar bugs venideros.

La gran mayoría de los lenguajes de programación cuenta con un software de pruebas unitarias apropiado, el cual se encargará de leer el código, fuente y comprobar si hay errores. [13]

9) Bajo el contexto de pytest. ¿Qué es un "assert"?

Bajo el contexto de pytest, el comando 'assert' permite el funcionamiento tal como el assert estándar de Python introduciendo la habilidad de dar más contexto sobre el porqué del fallo. Dicho comando logra corroborar si una condición en el código se cumple. Se llama a la función o método desde el código fuente y se corrobora que el resultado o respuesta que arroje sea el esperado. La salida de Pytest muestra exactamente dónde falló la prueba y cómo difieren los dos datos que se comparan. Esto es útil para resolver rápidamente una prueba fallida [14]

10) ¿Qué es Flake 8?

Flake 8 es una herramienta de linting para Python. [15] Una herramienta de lint (por su nombre en inglés) desarrolla análisis estático sobre el código fuente para revisar los posibles errores de sintaxis y proveer instrucciones acerca de cómo limpiar el código. [16] Flake8 es una herramienta muy completa debido a que abarca/verifica lo que es código según PEP8, pyflakes y circular complexity (por su nombre en inglés), que son herramientas utilizadas para verificar la eficiencia del código según ciertos criterios específicos. Mediante esta herramienta es posible mejorar la eficiencia de un código y el procesamiento de este en el computador. [17]

Bibliografía

[1] Lidgi González, «¿Qué es Git y GitHub?,» 20 Noviembre 2020. [En línea]. Available: https://aprendeia.com/que-es-git-y-github. [Último acceso: 27 Febrero 2021].

[2] A. Deymar, «Cómo usar una Git Branch,» 3 Febrero 2020. [En línea]. Available: www.hostinger.es/tutoriales/como-usar-git-branch. [Último acceso: 27 Febrero 2021].

[3] D. Pan, «Commits - Administrar Tu Repositorio,» 19 Octubre 2015. [En línea]. Available: https://codigofacilito.com/articulos/commits-administrar-tu-repositorio. [Último acceso: 27 Febrero 2021].

[4] Atalssian, «Git Cherry Pick,» [En línea]. Available: https://www.atlassian.com/es/git/tutorials/cherry-pick. [Último acceso: 27 Febrero 2021].

[5] Atlassian, «Git Stash,» [En línea]. Available: atlassian.com/es/git/tutorials/saving-changes/git-stash. [Último acceso: 27 Febrero 2021].

[6] Atlassian, «Git Fetch,» [En línea]. Available: atlassian.com/git/tutorials/syncing/git-fetch. [Último acceso: 27 Febrero 2021].

[7] Tower, «What's the difference between git fetch and git pull?,» [En línea]. Available: git-tower.com/learn/git/faq/difference-between-git-fetch-git-pull/. [Último acceso: 27 Febrero 2021].

[8] J. Mod, «A Better Git Workflow with Rebase,» 24 Noviembre 2017. [En línea]. Available: youtube.com/watch?v=f1wnYdLEpgI&t=191s. [Último acceso: 28 Febrero 2021].

[9] S. Kumar, «Git Rebase vs. Git Merge,» 28 Diciembre 2020. [En línea]. Available: dev.to/shubhamkumar10/git-rebase-vs-git-merge-4ikf. [Último acceso: 28 Febrero 2021].

[10] Atlassian, «Merging vs. Rebasing,» [En línea]. Available: atlassian.com/git/tutorials/merging-vs-rebasing. [Último acceso: 28 Febrero 2021].

[11] D. G. Rivera y A. C. Salas, «Unit testing y su necesaria aplicabilidad para validar la calidad del software,» 4 Diciembre 2020. [En línea]. Available: delfino.cr/2020/12/unit-testing-y-la-calidad-del-software. [Último acceso: 28 Febrero 2021].

[12] Yeeply, «¿Qué son las pruebas unitarias y cómo llevar una a cabo?,» [En línea]. Available: yeeply.com/blog/que-son-pruebas-unitarias/. [Último acceso: 28 Febrero 2021].

[13] Software Testing Help, «The Differences Between Unit Testing, Integration Testing And Functional Testing,» 18 Febrero 2021. [En línea]. Available: softwaretestinghelp.com/the-difference-between-unit-integration-and-functional-testing/. [Último acceso: 1 Marzo 2021].

[14] H. Krekel, Pytest Documentation, Merlinux, 2021, p. 25.

[15] D. Null, «What is Flake8 and why we should use it?,» 30 Enero 2017. [En línea]. Available: medium.com/python-pandemonium/what-is-flake8-and-why-we-should-use-it-b89bd78073f2. [Último acceso: 28 Febrero 2021].

[16] Sider Documentation, «Flake8,» [En línea]. Available: help.sider.review/tools/python/flake8. [Último acceso: 1 Marzo 2021].

[17] V. Freitas, «How to Use Flake8,» 5 Agosto 2016. [En línea]. Available: simpleisbetterthancomplex.com/packages/2016/08/05/flake8. [Último acceso: 1 Marzo 2021].