**GitHub y GitLab | La historia e importancia de los sistemas de control de versiones**

**¿Qué son?**

GitLab y GitHub son sistemas de control de versiones, son herramientas que hacen que los programadores puedan trabajar en equipo sin querer matarse entre ellos. Vamos a hablar de sus diferencias, de sus ventajas y desventajas.

**Historia**

En el mundo del pasado la gente manejaban las versiones con nombres como “Versión Final”, “Versión Final 2.0”, “Versión final final final este sí”. Curiosamente, hoy en día todavía hay personas que desgraciadamente viven en ese mundo del pasado y no conocen un mundo mejor. Esto no ocurre solamente en el mundo de la programación, sino también en el mundo del diseño, producción audiovisual y más.

Uno de los más famosos del pasado es Concurrent Version System o CVS. CVS era muy bueno, popular y también muy viejo, pero abrió al mundo algo muy lindo que es el sistema de control de versiones de una manera distribuida y que realmente funcionara en múltiples sistemas. Pero era difícil de entender.

Para hacerlo más fácil, para tener diferentes usuarios, que fuese Open Source y demás nace Subversión o SVN. Uno de los features (características) más interesantes es la **Culpa** o el comando **blame** para saber qué personas escribieron tales líneas de código y así poder decirles: “tú fuiste la persona que rompió el proyecto”. Subversion no era suficiente, en especial en el mundo Open Source. El proyecto más importante del mundo de código abierto es Linux. Creado por Linus Torvalds.

Así que decidió crear desde cero su propio sistema de control de versiones con ramas y lo llamó Git. Git es un sistema distinto a los anteriores porque las personas pueden hacer uso de ramas. Puedes tener tres ramas diferentes, con nombres diferentes, donde puedes estar haciendo cambios diferentes a un mismo archivo y donde diferentes personas pueden estar trabajando en cosas diferentes. Luego puedes fusionar todos esos cambios y mandarlos a producción en una sola rama que puede ser la rama master.

Así funciona el Kernel de Linux. Muchas veces múltiples Kernel de Linux son mantenidos al mismo tiempo, pero ¿qué pasa cuando se encuentra el mismo bug en los diferentes kernels? Se puede sacar una rama del bug, cambiar el bug y luego fusionar ese código en la versión final. Esa es la magia y eso permite que múltiples personas, incluso si no trabajan en el mismo lugar o si no se conocen, sino simplemente manejan los mismos estándares, puedan trabajar sobre el mismo archivo o sobre el mismo tipo de proyecto.

Hay un comic muy chistoso de XKCD qué explica cómo funciona Git:



Y tiene razón, Git es increíble, pero requiere que tengas conocimientos de terminal y línea de comandos. Y sí, todas las personas que programan deberían conocer sobre esto, pero puede resultar difícil en especial si estás empezando. Hay interfaces gráficas para Windows y Mac para hacerlo un poco más amigable, pero nadie había hecho algo centralizado hasta el 2012 cuando nació GitHub.

**GitHub**

GitHub es una de las más importantes herramientas de la historia de la programación. Es, en resumen, una red social de código. Es una serie de proyectos conectados con programadores, conectados con compañeros que le pueden poner una estrella o hacer un fork a tu proyecto.

Fueron tan populares en su momento que se dieron el lujo de levantar dos rondas de inversión. Una serie A de $100M, una serie B de $250M y fueron adquiridos por Microsoft por $7.5 billions.

**GitLab**

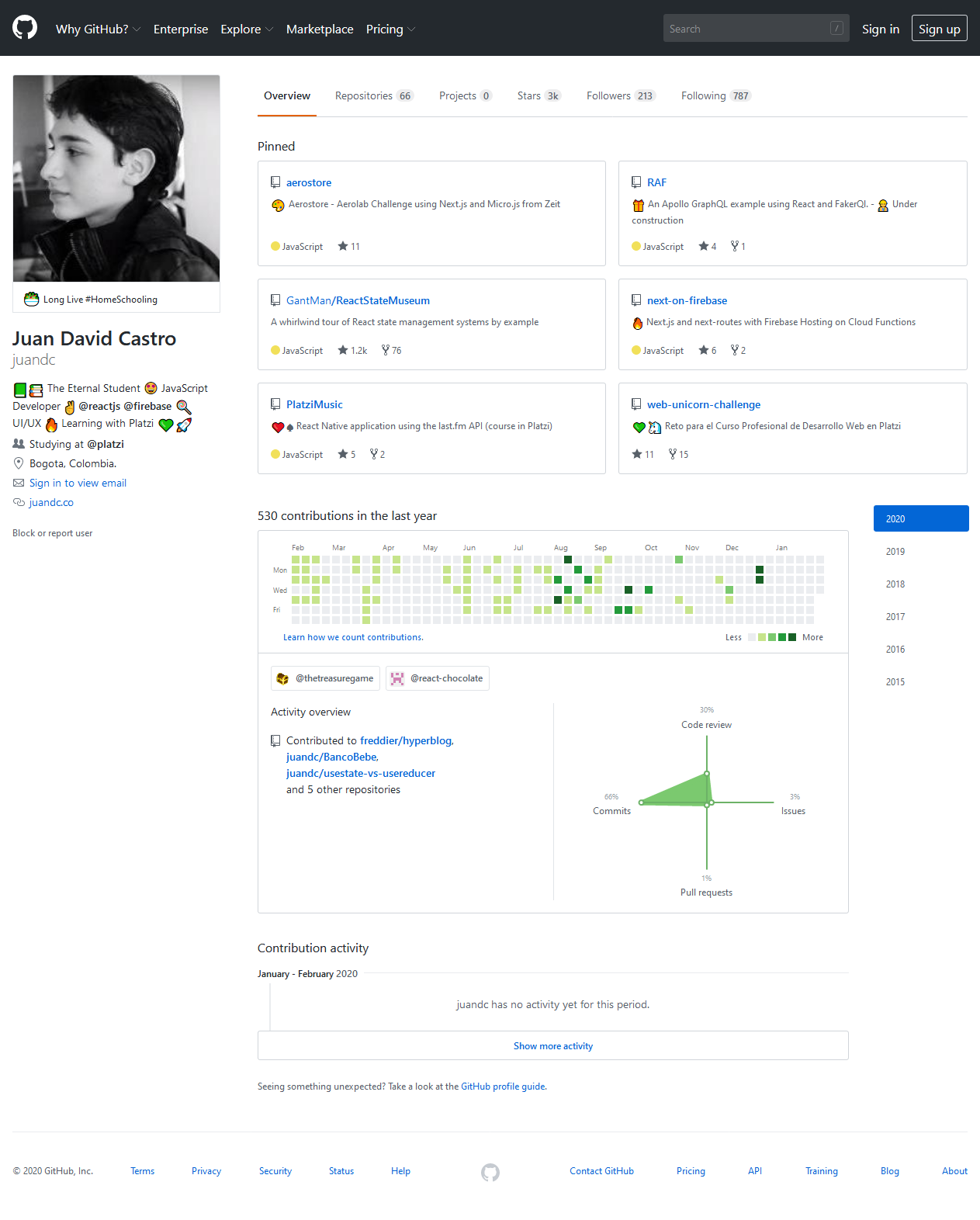
Años después, nace GitLab… Básicamente es GitHub, pero open source y a partir de ahí se empiezan a separar mucho más. La idea principal era tener algo igual a GitHub, pero de código abierto.

Es una empresa famosa porque pasó por Y Combinator, es YC W15 al igual que Platzi. Ellos han levantado cuatro rondas de inversión y han logrado obtener $165.5M y tienen una valuación de $1.1 billions.

**Aportes**

El aporte más grande de GitHub al mundo no es solo el hecho de colocarle una interfaz web al sistema de control de versiones, son los Pull Requests. ¿Qué es esto? Es tener un código y que otra persona pueda mejorarlo agregándole tal cantidad de líneas; el dueño de ese código tiene la opción de aceptar esa propuesta o no.

Otro de los aportes más increíbles de GitHub a la humanidad es el perfil de usuario de programador donde básicamente ustedes pueden entender qué es lo que hace cada uno de los Developers o programadores que le aporta a un proyecto.



**GitHub vs GitLab | PlatziLive**

[](https://www.youtube.com/watch?v=EscDe0jG6XM)

Esta es como la magia que tiene GitHub y por eso por mucho tiempo se veía como una empresa imposible de tocar.

Hasta que en los últimos años cambió la teoría en la que se desarrolla el software y nació el DevOps. La fusión de Development, el proceso de Quality Assurance y IT Operations o Operaciones de tecnología de la información. GitHub como tiene una API abierta logró que muchas empresas empezaran a aportarle a todo este ciclo de operaciones y nacieron una gran cantidad de startups o herramientas.

En el proceso varios programadores y equipos de desarrollo a medida que el código se volvía cada vez más complejo, se dieron cuenta que teníamos que cambiar nuestra metodología de desarrollo y empezaron a automatizar los procesos para que con un solo botón se enviara el código a producción, con otro se hicieran las pruebas ya programadas y si todo sale bien esperar a que otro programador lo revise con un Pull Requests y Code Review. Antes todo esto era manual.

Y eso crea básicamente tres grandes teorías en el mundo el desarrollo:

* **Continuous Integration**: Es un proceso en donde cada cambio que realizamos en nuestro código debe ser testeado y verificado por las pruebas que hemos escrito con anterioridad, haces el deploy o despliegue de tu código a staging.
* **Continuous Delivery**: Es el mismo proceso anterior, solo que de una manera más automatizada y además nuestros tests de aceptación o acceptance tests deben ser de una gran calidad. Porque Continuous Delivery se asegura de que cada cambio realizado esté listo para ser lanzado a producción.
* **Continuous Deployment**: Es el sueño, llevar código a producción sin que haya humanos en el proceso.

Ahora el proceso de DevOps se ha vuelto más complejo



Alguien debe encargarse de manejar todo el Ciclo de DevOps, cuáles son las herramientas, cómo se integran entre sí, porque hay muchas herramientas desplegadas y estas agregan capas y capas que hacen probable un ataque. Uno de los problemas más comunes últimamente en entornos de desarrollo es que los hackers obtienen acceso a uno de los procesos de Continous Deployment, Integration o Delivery y justo en ese proceso inyectan código que puede dañar nuestra aplicación, crear backdoors, añadir virus o ataques similares. También alguien debe monitorear y verificar que todo ese proceso sea defensible.

Y esa es la gracia de GitLab, ellos se dieron cuenta que como open source podían crear un entorno donde ocurriera todo, donde en vez de tener que usar quince herramientas distintas, ellos podían usar todo al mismo tiempo y fueran la herramienta única y definitiva para todo el ciclo de vida de DevOps.

Sin embargo, Microsoft ya había creado algo similar antes de comprar a Github llamado Azure DevOps. Integra todo el ciclo a través de Microsoft haciendo también Continuous Integration, Delivery colaboración, código, se puede integrar con Kubernetes (aunque es de google), con Docker y con lo que sea.

Conclusión

Son dos empresas que siguen creciendo mucho y cada vez están a la par con sus ofertas. Recuerda no casarte con una sola empresa, no digas que como ya aprendiste GitHub no deberías aprender nada más, porque no sabes qué puede pasar en el futuro.

Yo, por mucho tiempo, porque fui estúpido. Me resistí a aprender Git y GitHub porque creía que Subversion era genial y no eran tan significativamente superiores a Subversion para mí, pero ahora son mucho mejor. Y gracias a eso, creo yo mejoré como programador en la época. Por eso es importante nunca parar de aprender.