**Tutorial configuración Pipeline Gitlab + Jenkins + Sonarqube para app de Java.**

Autor: Vansi Adonay Olivares

En este tutoral se muestra como realizar las siguientes tareas:

* Como usar Jenkins para hacer compilaciones sencillas
* Configurar Gitlab para comunicarse con Jenkins
* Integrar a Sonarqube con Jenkins
* Como crear un Pipeline en Jenkins
* Los pasos a seguir para armar un Pipeline de Java + Maven

Todo esto será montado usando contenedores de Docker. Si no sabe que es Docker, digamos que son maquinas virtuales mucho más ligeras que las máquinas virtuales tradicionales ya que solo son un proceso levantado. Usan un principio muy parecido al al usar el comando *chroot* donde se tiene un S.O dentro de un S.O.

Si desea aprender un poco más sobre docker, por favor visite el siguiente enlace:

https://en.wikipedia.org/wiki/Docker\_(software)

**Requisitos:**

Para ejecutar todo esto, en esencia necesita:

* Tener instalado docker y docker-compose (consulte la documentación de su sistema operativo para saber como instalarlo)

* Un sistema operativo Linux
* Hardware un tanto potente, aunque se puede hacer con cualquier cosa (todo esto fue hecho en un Core 2 Duo T5850 con mucha paciencia)
* Deseos de aprender sobre DevOps en forma práctica
* Mucha paciencia

Antes de comenzar, es bueno que el lector se familiarice con los términos mencionados allá arriba.

* **Jenkins:** Es una aplicación hecha en Java que busca automatizar todo el proceso de compilación, testeo y deployment de una aplicación.
* **Gitlab:** Es un manejador de repositorios de git con otras características integradas. Git es un sistema de control de versiones que guarda tu código y sus versiones y Gitlab es donde se guarda de forma centralizada el código
* **Sonarqube:** Es un “repositorio” de analisis de código estático. Guarda las estadísticas de como va tu código, permite crear perfiles de aceptación de calidad y otras cosas. Existen para casi cualquier lenguaje que sueñes.

**1.- ¿Qué se hará aquí?**

Lo que se quiere hacer es emular las condiciones de una empresa con una aplicación de Java absurda mente sencilla en la que hacer todo esto no vale realmente la pena, pero la experiencia de aprender a montarlo definitivamente lo vale.

Cuando las empresas son grandes y dependen mucho del software, no es posible darse el lujo de perder el tiempo excesivamente en compilando código, configurando pruebas, ejecutándolas una a una, etc si puedo hacer que una herramienta lo haga por mí. Esa es la idea de DevOps: hacer que desde la codificación hasta el deployment de una aplicación sea casi por completo automatizada.

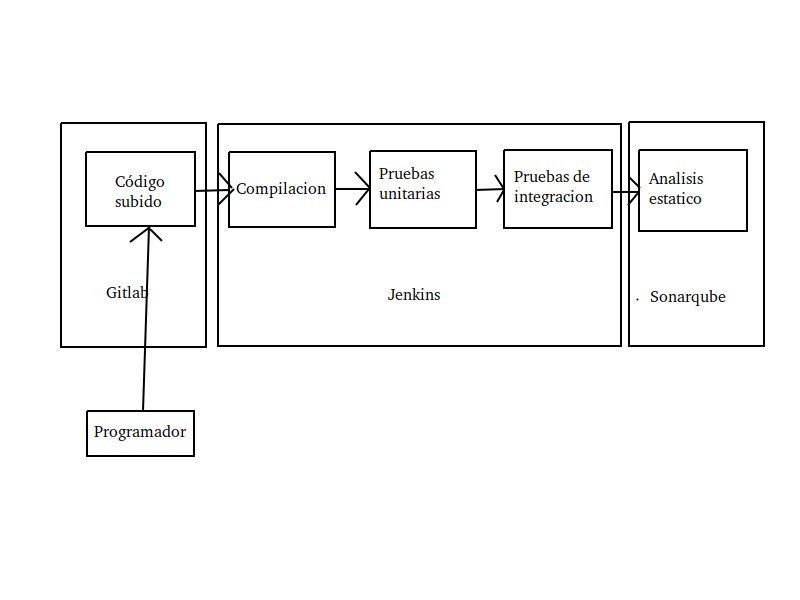
Primero se comienza teniendo el código en un lugar centralizado, donde se guardan los cambios realizados más recientes y cualquiera dentro de un equipo que necesite el código, trabajar en él, guardarlo y compartirlo lo hagan con un par de comandos. Es entonces cuando las herramientas como los sistemas de control de versiones - como git - hacen su magia. Git se encarga de manejar como tal el manejo del control de versiones: que agregar, cuando agregarlo, como obtener el código más reciente, etc y se complementan con un servidor central que guarda el código - un repositorio – como lo es Gitlab.

Ya teniendo el código guardado, hay que compilarlo y probarlo y aquí es donde se necesita algo que lo haga por mí. Una de las herramientas centrales en este proceso es un servidor de builds, que es donde se lleva el registro de todos los builds hechos y sus resultados, además de la automatización de obtener el código, compilarlo y testearlo. Allí es donde Jenkins entra en acción al solventar estas problemáticas.

Una de las pruebas que se pueden realizar sobre el código es el análisis estático, que busca analizar la calidad del código sin ejecutarlo, detectando vulnerabilidades, posibles errores y otras posibles fallas. Una de las herramientas populares que hace todo esto es Sonarqube que al igual que Jenkins guarda el historial de todo lo realizado (en este caso, pruebas).

La siguiente image describe el escenario que se busca recrear:

*Imagen 1: Pasos a configurar en el build.*

****

**2.- Levantando los servicios**

Para comenzar, estando en el directorio raíz del repositorio ejecute el siguiente comando:

bash deploy.sh

Espere a que se compile la imagen de Jenkins y a que se levanten todas la imágenes. Es posible que tome una buena cantidad de tiempo dependiendo del hardware que posea. La imagen de Gitlab toma notablemente mucho tiempo la primera vez que se levanta, por lo que se le pide sea muy paciente. Todas las imágenes han cargado si aparece en la consola algo así:

Referencias:

* <https://jenkins.io/doc/>
* <https://en.wikipedia.org/wiki/GitLab>