TP II: Contenedores y escalabilidad

Introducción

Nos dan una aplicación desarrollada en Spring Boot 2 que corre en un solo servidor y cuyo código está en el siguiente repositorio:

https://github.com/distribuida2/no-escalapp

Requerimientos funcionales

1. Migrar el ambiente para ser corrido en un contenedor. El contenedor debe correr con un máximo de 512mb de RAM y ½ core.
2. Correrl Exigir al software y generar pérdida de requests. Hay varias opciones para usar: JMeter, Gatling o alguna solución ad-hoc.
3. Escalar la solución horizontalmente y disponibilizar más de un nodo de esta aplicación.
4. Debemos minimizar los puntos únicos de falla.
5. Debe reducir en al menos la mitad la cantidad de 5XX devueltos por el servidor (que está en aproximadamente un 50%).
6. Migrar los servicios que no sean escalables y adoptar una solución adecuada y que no tenga puntos únicos de falla (en este punto se recomienda usar Redis + Sentinel, que es el motor de base de datos que usaremos como caso de estudio en la materia).
7. ¿La solución planteada sirve si los resources guardados son críticos y no puede haber diferencias e inconsistencias? En cualquier caso, explicar las razones y plantear al menos dos escenarios para guardar información de manera consistente de ser necesario y detallar las soluciones planteadas, beneficios y contras de cada una.

Requerimientos no funcionales

1. Hacer tests de unidad con JUnit. Todas las clases y métodos deben estar cubiertos por tests.
2. Integrar el trabajo práctico con Travis CI.
3. Debe haber una medida de cobertura de los tests.
4. El trabajo debe estar en un repositorio público del estilo github o gitlab.
5. Debe estar hecho con Maven.

Fecha de entrega

Martes 03/12/2019