Proyecto final

Materia:	Sistemas Distribuidos
Término:	2019 I
Profesor:	Cristina Abad
Número de integrantes:	1 a 3 (restricción; estudiantes que repiten materia deben hacer proyecto de manera independiente)
Tema:	Aplicación web implementada con una arquitectura de micro-servicios
Fecha entrega proyecto:	Jueves 22 de agosto, 8h00 (en mi oficina y vía SidWeb); a partir de las 8am puedo llamar a cualquier grupo a que presente su proyecto. Si un estudiante no está presente, tendrá automáticamente 0 en el proyecto. No atenderé a grupos que lleguen tarde.
Sistema operativo:	Linux

Objetivos

- Aprender sobre cómo estructurar una aplicación web usando una arquitecturas de microservicios.
- Entender cómo se usan containers (Docker) y orquestadores de containers (Kubernetes) para implementar sistemas distribuidos escalables y tolerantes a fallos.
- Realizar una evaluación de rendimiento a un sistema distribuido real.
- Aprender a usar algunos de los servicios de un proveedor de cloud computing, como AWS.

Detalles

En este proyecto ustedes trabajaran con una aplicación web implementada usando una arquitectura de microservicios. Existen dos posibles aplicaciones de entre las cuales cada grupo debe seleccionar una; si trabajan con las dos, pueden tener hasta 10 puntos extra. Las dos aplicaciones son:

1. SockShop: https://microservices-demo.github.io/ , https://microservices-demo.github.io/docs/quickstart.html , https://github.com/microservices-demo/microservices-demo/blob/master/internal-docs/design.md y https://github.com/microservices-demo/microservices-demo
2. TeaStore: https://github.com/DescartesResearch/TeaStore, https://www.youtube.com/watch?v=6OcSNrErzGE y http://www.mscs.mu.edu/~mascots/Papers/MASCOTS2018_TeaStore.pdf

Una vez seleccionada la aplicación, cada grupo deberá hacer lo siguiente:

- Leer la documentación y aprender a levantar los servicios.
- Hacer un deployment en máquinas virtuales/containers en la nube (ej.: usando AWS EC2).
- El deployment debe usar un cluster de Kubernetes, el cual ustedes deberán configurar para conseguir tolerancia a fallos y/o elasticidad de los microservicios. Deben usar un cluster real; no el *minikube*.
- Realizar experimentos que permitan evaluar el rendimiento (latencia y throughput) de la aplicación web. Los resultados deben ser documentados adecuadamente, usando principios estadísticos y de visualización de datos, en el reporte final del proyecto.
- Opcional (puntos extra):
    - Si optan por SockShop, pueden reemplazar la cola de envíos de pedidos, la cual actualmente usa RabbitMQ, por una cola de ActiveMQ; adicionalmente, deben realizar una evaluación que permita comparar ambas implementaciones.
    - Si optan por TeaStore, pueden reemplazar el *recommender system* por uno que implementen ustedes y realizar una evaluación que permita comparar el servicio original con el que implementen ustedes.

Otros (requerimientos de la materia)

- Deben utilizar Github; empiecen por clonar el proyecto con el que vayan a trabajar (SockShop o TeaStore) y en el repo de ustedes, realizar cualquier desarrollo adicional que hagan, incluyendo scripts para ejecutar pruebas automatizadas para la evaluación de rendimiento. Enviarme el enlace al repositorio vía SidWeb, máximo hasta el jueves de la próxima semana. **TODOS los miembros del grupo deben tener más de un commit en el repositorio; el primer commit de cada miembro debe tener una fecha máxima de Junio 30 de 2019. Necesito poder saber en qué trabajó cada uno. Si lo que hizo algún miembro no lleva a commits, entonces esa persona deberá presentar, junto con el reporte final, un detalle de las actividades realizadas, con screenshots que documenten el haber trabajado en las mismas.**
- Deberán entregar un reporte final (2 a 6 páginas) en el que documenten el proyecto, resultados de las evaluaciones (latencia y throughput, y elasticidad o tolerancia a fallos), y detalle de actividades (con fechas y screenshots) para cada miembro del proyecto. La penalidad por no incluir el detalle de actividades será una reducción en la nota del proyecto del 50%.
    1. Para las pruebas de rendimiento, deben generar pedidos/requerimientos con alguna herramienta; pueden usar los scripts de evaluación (load tests) que vienen con SockShop/TeaStore, o usar alguna otra herramienta adecuada para este propósito; esto puede ser alguna herramienta que descarguen de Internet, o ustedes mismos programar algo en Python o Bash.
    2. La idea es que realicen pruebas que permitan observar cómo se cumple con los requerimientos no funcionales relacionados a rendimiento (latencia y throughput), elasticidad y tolerancia a fallos.

3. Para cada uno de los experimentos, deben mostrar los resultados usando box plots o CDFs, usando una escala adecuada para los ejes de las x y de las y, y etiquetas descriptivas para los ejes así como para la leyenda de cada gráfico. Pueden buscar ejemplos en algunos de los papers asociados a los vídeos que han visto en el semestre.

- Deployment en la nube: Puede ser en Azure, Google Cloud Platform o AWS. AWS otorga créditos gratuitos para estudiantes. Alternativamente, búsquenme en mi oficina y les puedo dar credenciales de acceso a AWS, usando unos créditos que tengo yo. **IMPORTANTE:** ESPOL tiene bloqueado el acceso al puerto 22, necesario para hacer SSH a instancias EC2. Esta parte la deben hacer desde su casa. El demo lo podrán mostrar sin problema, ya que ustedes se conectarán al servicio vía HTTP, lo cual no está bloqueado.