Sistemas Distribuidos Tarea 2

Mauricio Solar msolar@inf.utfsm.cl
Ayudante: Humberto Farias Aroca humberto.farias@usm.cl

1. Objetivos

Familiarizarse con uso de arquitecturas distribuidas usando microservicios. Nos enfocaremos en trabajar en tipos de tecnologías:

- Remote Procedures (ejemplo: REST o RPC).
- Asynchronous Messaging (ejemplo: RabbitMQ o Apache Kafka).

Definición: "*Los microservicios son un enfoque para desarrollar una única aplicación como un conjunto de pequeños servicios, cada uno ejecutándose en su propio proceso y comunicándose con mecanismos ligeros, a menudo una API a través de HTTP." (James Lewis y Martin Fowler)*"

2. Generales

- El lenguaje a utilizar es Python.
- Para cada actividad se deberá trabajar usando virtualización en base a containers, donde cada parte de la arquitectura, por ejemplo el servidor, deberá ser un container de Docker.
- En todas las actividades se deberá configurar un deployment automático de la arquitectura usando los archivos *docker-compose.yml* y *dockerfile*.

3. Actividades

Actividad 1: Chat RPC

En esta actividad se debe crear un chat donde será posible que hayan n-clientes y un servidor central de coordinación de mensajes. Para esta actividad se debe usar el framework gRPC[1] para implementar el sistema de mensajería usando el protocolo RPC(Remote Procedure Call).

Servidor:

- Se debe crear el gRPC server.
- Se deberá disponer de un servicio de envío y recepción de mensajes.
- Cada vez que llega un mensaje debe enviarse al destinatario manteniendo un *log.txt*. con los mensajes enviados.
- Se deberá disponer de un servicio para obtener el listado completo de los clientes.
- Se deberá disponer de un servicio para obtener todos los mensajes enviados por el cliente que lo solicite.

Clientes:

- Cada cliente deberá tener un ID único.
- Cada cliente debe crear un *channel* con el servidor.
- Cada cliente podrá enviar y recibir mensajes a través del servidos a otros clientes de forma asincrona. Es decir el cliente envía el mensaje y puede seguir enviando otros al mismo destinatario a otros sin esperar que el primero llegue.
- El cliente enviará los mensajes por linea de comando.

[1] https://grpc.io/

- Los mensajes que le lleguen al cliente también deberán ser desplegados en la linea de comando.

Mensajes
- Cada mensaje deberá tener un ID único.
- Se debe incluir el texto del mensaje y un timestamp.

Generales:
- Tanto el servidor como todos los clientes deben ser containers de Docker.
- Se debe partir con 1 servidor y 2 clientes, esto debe ser desplegado de forma automática con Docker compose.
- Se debe poder registrar a nuevos clientes. Se deben especificar en las indicaciones de cómo realizar esto en su arquitectura.

Actividad 2: RabbitMQ

En esta segunda parte se deberá implementar la misma arquitectura de la parte 1. Pero en este caso no usando llamadas remotas a procesos si no mensajes asíncronos usando *RabbitMQ*[2] como bróker de mensajería o gestor de colas.

Actividad 3: informe comparativo

Se debe preparar un informe con las principales diferencias técnicas entre las dos maneras de implementar este sistema mensajería distribuido basado en microservicios. El informe debe terminar con una recomendación técnica de cuál es según su criterio la mejor tecnología para este problema.

4. Consideraciones generales
- Consultas sobre la tarea se deben realizar en moodle o enviar un correo al ayudante (humberto.farias@usm.cl).

5. Reglas de entrega
- La tarea se realiza en **grupos de 2 personas**.
- La fecha de entrega es el día **30 de Octubre** donde se debe subir el link a un repositorio de GitHub con los archivos de las actividades. Por favor dejar los repositorios públicos.
- La corrección se realizará sobre los archivos presentes un repositorio de GitHub creado para este fin por cada estudiante.
- Se ejecutará el comando *docker-compose build* y *docker compose up*. Y la arquitectura cliente-servidor deberá desplegarse de forma autónoma.
- Cada día o fracción de atraso, se penalizará con un descuento de **10 puntos**.
- Copias se calificarán con **nota 0**.

[2] https://www.rabbitmq.com/