Proyecto De Laboratorio: Push-Pull

Para completar el curso, se propone el diseño e implementación de un servicio de filtrado de mensajes persistentes, desplegable sobre el Kumori PaaS, según se especifica a continuación.

Entorno de ejecución

Para llevar a cabo el proyecto se dispondrá de un cluster donde probar/ejecutar los resultados en la url: https://admission-preview.vera.kumori.cloud

Cada alumno dispone de un nombre de usuario en dicha plataforma consistente en su alias de email de registro, precedido por "cc_". El password os será suministrado por el profesor.

Condiciones de elaboración

El proyecto deberá ser elaborado de forma preferente por equipos de dos personas, aunque también es posible elaborarlo a título individual en casos excepcionales.

En caso de formar un equipo la entrega se realizará tan sólo sobre un depósito git en gitlab. Dicho depósito/proyecto deberá tener el nombre **CCPUSHPULL,** y deberá dársele permiso al usuario ***mbajosep@gmail.com*** como miembro del proyecto.

En el depósito git deberá verse claramente las contribuciones de cada miembro del equipo (en su historia de commits)

Parte de la entrega deberá ser un documento describiendo la solución adoptada, su arquitectura, representación en el modelo de KPaaS, y su funcionamiento.

Descripción proyecto PULL-PUSH

El proyecto consistirá en la definición y despliegue de un servicio sobre el PaaS de Kumori Systems con la siguiente funcionalidad:

1. Un microservicio **FRONTEND**, exponiendo un API REST mediante el cual un cliente podrá
    a. Insertar objetos JSON en el servicio (PUSH)
    b. Realizar querys sencillas para obtener objetos JSON que cumplan ciertas condiciones (PULL)
2. El FRONTEND insertará los objetos que le son enviados en otro microservicio que implementa una cola persistente de mensajes.

3. El FRONTEND, cuando recibe una query, contacta con el microservicio de base de datos para obtener los resultados y enviárselos al cliente.
4. Uno o varios microservicios **WORKER**, que se suscribirán a la cola persistente, para recibir un cierto tipo de mensajes.
5. El WORKER, cuando recibe un mensaje, lo acondiciona para guardarlo en el microservicio de Base de Datos.

En total se tienen los siguientes microservicios:

1. FRONTEND
2. $WORKER_i$
3. COLA
4. BBDD

Dependiendo de cuantos tipos de WORKER se tengan, habrá al menos cuatro microservicios diferentes.

De estos microservicios, FRONTEND y todos los $WORKER_i$ deberán ser implementados por vosotros usando bien nodejs, bien golang.

El microservicio COLA deberá ser una de las colas de mensaje que se proponen, y deberá ser integrada en la aplicación de servicio resultante.

El microservicio BBDD deberá ser una de la Bases de Datos que se proponen y deberá ser integrada en la aplicación de servicio resultante.

En todos los casos, será necesario crear el modelo de *componente* Kumori para cada microservicio.

Finalmente, será necesario producir el modelo de *aplicación de servicio* Kumori que integra a todos los roles (microservicios) basados en los componentes modelados (FRONTEND, WORKER, COLA, BBDD), y que deberá ser desplegada sobre la plataforma.

Opciones para el componente COLA

- ARTEMIS
- NATS
- KAFKA
- KubeMQ

NOTA: El modelado deberá incluir las opciones para replicación que ofrezca la BBDD seleccionada.

Opciones para el componente BBDD

- COCKROACHDB
- MARIADB
- CASSANDRA
- REDIS

NOTA: El modelado deberá incluir las opciones para replicación que ofrezca la BBDD seleccionada.

Consideraciones adicionales

Deberá desarrollarse también un test de integración mínimo que muestre el funcionamiento del servicio desplegado.

NOTA: Dicho test puede realizarse como un script bash con la ayuda de software como curl.