Práctica 5. Escalabilidad, tolerancia a fallos y desarrollo nativo en Kubernetes

Enunciado

Esta práctica tiene dos objetivos:

- Verificar cómo las diferentes técnicas y estrategias que proporciona Kubernetes permiten que una aplicación sea tolerante a fallos
- Hacer uso de alguna de las herramientas de desarrollo nativo Kubernetes

Tolerancia a fallos

Se usará la aplicación webapp-stateless que permite desplegar varias réplicas y la información compartida se almacena en una base de datos MySQL.

Se ejecutarán varias pruebas de Caos con diferentes técnicas aplicadas para que se pueda comprobar la tolerancia a fallos:

- Para simular el uso de la aplicación y comprobar su correcto funcionamiento, se usarán herramientas que típicamente se usan para hacer pruebas de carga (Artillery, JMeter u otra herramienta).
- Se usarán esas herramientas para ejecutar peticiones http sobre la página principal de la aplicación.
- En cada escenario, se anotará el % de peticiones correctas (200 OK) y el % de peticiones con algún tipo de error.
- Se ejecutarán las pruebas con los siguientes escenarios:
    - **Escenario 1:** Un único pod en el deployment. Sin matar pods (Sin caos). La herramienta de testing deberá mostrar un 100% de peticiones correctas.
    - **Escenario 2:** Un único pod en el deployment. Matando pods con chaos-pod-monkey. No se deberá matar el pod de la base de datos.
    - **Escenario 3:** Dos pods en el deployment. Matando pods con chaos-pod-monkey. No se deberá matar el pod de la base de datos. Se debería obtener un % de peticiones erróneas menor que en el escenario 2.
    - **Escenario 4:** Dos pods en el deployment. Uso de Gateway de Istio para aceptar peticiones. Matando pods con chaos-pod-monkey. No se deberá matar el pod de la base de datos. Se debería obtener un % de peticiones erróneas bastante reducido.

Se deberá entregar un vídeo mostrando la ejecución de los diferentes escenarios y las tasas de fallo obtenidos en cada uno de ellos.

Herramienta de desarrollo nativo Kubernetes

Se deberá usar alguna de las herramientas vistas en clase para desarrollo nativo Kubernetes con la aplicación EoloPlanner. El alumno podrá elegir qué herramienta concreta usar (okteto, VSCode, Quarkus dev tools, etc.) y el servicio concreto que quiere usar con la herramienta.

Se deberá crear un vídeo en el que se muestre:

- El uso de un punto de ruptura en el IDE en modo depuración cuando la aplicación se está ejecutando en Kubernetes.
- Cómo un cambio en el código se despliega de forma automática en el clúster (sin tener que reconstruir el contenedor y actualizar el Deployment).

Formato de entrega

La práctica se entregará teniendo en cuenta los siguientes aspectos:

- La práctica se entregará por el aula virtual con la fecha indicada.
- La práctica se entregará como un fichero .zip del proyecto Maven.
- El proyecto se puede crear con cualquier editor o IDE.
- Se deberá incluir en el fichero README una descripción básica del proyecto entregado.
- En caso de contener un vídeo que sea muy pesado y sea rechazado por la plataforma, se podrá subir el vídeo al espacio OneDrive del alumne e incluir en el README la URL de dicho vídeo.
- Las prácticas se podrán realizar de forma individual o por parejas. En caso de que la práctica se haga por parejas:
    - Sólo será entregada por uno de los alumnos
    - En la raíz del proyecto se incluirá un fichero `authors.txt` que contendrá el nombre y correo URJC de cada uno de los alumnos.