Proyecto final: Gestión de Eventos & Participantes

Objetivo:

Al finalizar este proyecto, los participantes serán capaces de desarrollar, contenerizar y desplegar una solución basada en microservicios para la gestión de eventos y sus participantes, utilizando Docker y Kubernetes. Esto incluirá la creación de imágenes Docker optimizadas, la configuración de un archivo Docker Compose para el desarrollo local, y el despliegue en un clúster de Kubernetes con configuración de servicios, replicasets, y acceso mediante Ingress, aplicando las mejores prácticas de escalabilidad, gestión de redes y almacenamiento en contenedores.

Planteamiento:

Una plataforma eficiente para gestionar eventos y sus participantes es crucial en diversos entornos, desde conferencias y talleres hasta eventos sociales y deportivos. Este proyecto propone el desarrollo de una solución compuesta por dos microservicios: uno encargado de administrar la información de los eventos (como fechas, ubicaciones y descripciones) y otro para gestionar las inscripciones de los participantes.

El desafío consiste en contenerizar ambos microservicios con Docker y desplegarlos en un clúster de Kubernetes. El sistema deberá implementar comunicación unidireccional entre los microservicios, configurarse para ser escalable y accesible mediante Kubernetes Ingress, y utilizar buenas prácticas en la gestión de redes, almacenamiento y despliegues. Este proyecto permitirá a los participantes integrar los conocimientos adquiridos en el curso y resolver un caso práctico basado en necesidades reales del mundo laboral.

Requerimiento:

1. Funcionales

1. Gestión de Eventos (Microservicio 1: ms-eventos):
    - Permitir registrar, consultar, actualizar y eliminar eventos.
    - Proveer una API REST para consultar la información de los eventos disponibles, incluyendo:
        - Nombre del evento.
        - Descripción.
        - Fecha y hora.
        - Ubicación.
        - Capacidad máxima.

2. Gestión de Inscripciones (Microservicio 2: ms-participantes):
    - Permitir a los participantes inscribirse en eventos existentes.
    - Validar que el evento al que se intenta inscribir el participante existe (consultando el Microservicio 1).
    - Listar todas las inscripciones realizadas por un participante.
    - Registrar las inscripciones en una base de datos y almacenar únicamente el ID del evento (no duplicar información).
    - Manejar restricciones como:
        - Capacidad máxima del evento.
        - Evitar duplicados en las inscripciones para un mismo participante y evento.

3. Comunicación entre Microservicios:
    - El Microservicio 2 debe consumir los datos del Microservicio 1 para validar la existencia de eventos.
    - Utilizar una API REST o mensajería asincrónica para la comunicación entre ambos microservicios.

2. No Funcionales

1. Contenerización:
    - Crear imágenes Docker optimizadas para ambos microservicios.
    - Publicar las imágenes en un registro Docker (Docker Hub).
    - Configurar los actuadores de Spring Boot y su exposición adecuada para que sean compatibles con Docker y Kubernetes.
    -

2. Despliegue en Kubernetes:
    - Implementar un clúster Kubernetes con:
        - Pods separados para cada microservicio.
        - Configuración de servicios para exponer ambos microservicios dentro del clúster.
        - Escalabilidad mediante réplicas (ReplicaSet o Deployment).
    - *Configurar Ingress para permitir acceso externo mediante rutas basadas en el contexto (por ejemplo, /eventos y /inscripciones).*

3. Persistencia:
    - Utilizar bases de datos separadas para cada microservicio.
    - *Configurar volúmenes en Kubernetes para almacenar los datos.*

4. Configuración de Red:
    - Configurar una red interna en Docker Compose y Kubernetes para permitir la comunicación entre los microservicios.
    - Garantizar que los microservicios solo se comuniquen mediante APIs definidas.

5. Probes y Políticas de Reinicio:
    - *Configurar liveness y readiness probes en Kubernetes para garantizar la disponibilidad de los servicios*.

- *Implementar políticas de reinicio adecuadas para los pods.*

6. Monitoreo y Disponibilidad
   - Configurar **Spring Boot Actuator** en cada microservicio para habilitar endpoints de monitoreo como /actuator/health, integrados con Kubernetes para probes y livenes y readiness.

7. Gestión de Tráfico:
   - Utilizar **Spring Cloud Gateway** como API Gateway para:
     - Gestionar las rutas hacia los microservicios (por ejemplo: /api1/eventos & /api2/inscriciones)
     - Implementar funcionalidades adicionales como balanceo de carga.

8. Documentación y Buenas Prácticas:
   - Seguir buenas prácticas en la creación del Dockerfile (minimizar capas, reducir el tamaño de la imagen, etc.).
   - *Configurar variables de entorno para credenciales y configuraciones, utilizando Kubernetes Secrets y ConfigMaps*

Recursos:

1. Tecnologías y Herramientas

- Lenguaje de programación:
  - Java 21
- Frameworks y librerías:
  - Spring Boot 3.3.x
  - Spring Cloud Gateway
  - Spring Boot Actuator
  - Spring Data JPA
  - Spring Web
- Base de datos:

    - Oracle DB 21c
- Contenerización:
    - Docker
    - Docker Compose
- Orquestación de contenedores:
    - Kubernetes

2. Herramientas de Desarrollo

- Entorno de desarrollo integrado (IDE):
    - Spring Tool Suite (STS)
- Gestión de dependencias:
    - Maven (pom.xml configurado)
- Control de versiones:
    - Git
    - GitHub (repositorio remoto)

3. Recursos de Configuración

- Archivos de configuración:
    - application.properties o application.yml para cada microservicio.
    - Archivos YAML de Kubernetes para:
        - Deployments.
        - Services.
        - Ingress.
        - ConfigMaps y Secrets.
    - Dockerfiles optimizados para ambos microservicios.
    - docker-compose.yml para desarrollo local.
- Endpoints de APIs:
    - Documentación de APIs con Swagger/OpenAPI (Opcional)

6. Material de Apoyo

- Guías:
    - Documentación oficial de Spring Boot y Spring Cloud.

  - Manuales de Docker y Kubernetes.
  - Material y prácticas del curso Docker y Kubernetes Intermedio
- Scripts o comandos predefinidos:
  - Scripts para inicializar bases de datos y cargar datos iniciales.
  - Comandos para construir imágenes y gestionar contenedores.
  - Comandos para desplegar en Kubernetes.

Entregable:

Entregables del Proyecto Final: Gestión de Eventos y Participantes

Los entregables se organizan en categorías que abarcan desde el código fuente hasta la documentación y los artefactos de despliegue

1. Código Fuente

1. Repositorios de Microservicios:
   - Código fuente de los dos microservicios (gestión de eventos y gestión de inscripciones).
   - Estructura del proyecto conforme a las mejores prácticas (separación en capas: controlador, servicio, repositorio, entidades, etc.).
   - Uso de control de versiones (Git) con un historial claro de commits.

2. Artefactos Contenerizados

1. Imágenes Docker:
   - Imágenes Docker de ambos microservicios, construidas y publicadas en un repositorio como Docker Hub.
   - Imágenes optimizadas según las mejores prácticas (Dockerfile eficiente).

2. Docker Compose:
   - Archivo docker-compose.yml para levantar ambos microservicios junto con sus bases de datos en un entorno de desarrollo local.

3. Configuración para Kubernetes

1. **Archivos YAML:**
   - **Deployments:** Configuración de pods y réplicas para ambos microservicios.
   - **Services:** Definición de servicios ClusterIP o NodePort para los microservicios.
   - **Ingress:** Configuración de rutas basadas en contexto para exponer las APIs externamente (Opcional, este tema se cubrió en enl nivel Básico)
   - **ConfigMaps y Secrets:** Manejo de variables de entorno y datos sensibles.
   - **Probes:** Liveness y readiness probes configuradas.
2. **Manifiestos completos:**
   - Conjunto de manifiestos YAML organizados y documentados en carpetas para cada microservicio.

4. Bases de Datos

1. **Scripts de Inicialización:**
   - Scripts SQL para inicializar las bases de datos con tablas necesarias y datos de prueba.
2. **Volúmenes:**
   - Configuración de volúmenes en Kubernetes para garantizar la persistencia de los datos.

5. Documentación

1. **Manual de Despliegue:**
   - Pasos detallados para:
     - Contenerizar y construir las imágenes.
     - Desplegar en Kubernetes utilizando los manifiestos YAML.
     - Configurar Docker Compose para desarrollo local.
2. **Diagrama de Arquitectura:**
   - Representación gráfica de la solución, incluyendo:
     - Estructura de microservicios.
     - Comunicación entre ellos.
     - Despliegue en Kubernetes.

6. Resultados

1. **Validación del Despliegue:**
    - Evidencia del despliegue exitoso en Kubernetes (por ejemplo, capturas de pantalla o logs que muestren los pods corriendo).
    - Pruebas realizadas en los servicios mediante Postmano o CURL.
2. **Registro de Imágenes:**
    - URL de las imágenes Docker publicadas (Docker Hub).