Proyecto Final Plataformas II

Objetivos de Aprendizaje

- Implementar una arquitectura completa de microservicios en Kubernetes
- Aplicar prácticas avanzadas de DevOps en entornos de contenedores
- Configurar sistemas robustos de observabilidad y monitoreo
- Implementar estrategias de despliegue modernas y seguras
- Demostrar dominio de los conceptos clave de Kubernetes

Repositorio Base

Trabajarán con los microservicios del código disponible en:

https://github.com/SelimHorri/ecommerce-microservice-backend-app/

El proyecto requiere la implementación de **todos los microservicios** disponibles en el repositorio, incluyendo:

- Service Discovery
- Cloud Config
- API Gateway
- Proxy Client
- User Service
- Product Service
- Favourite Service
- Order Service
- Shipping Service
- Payment Service

Requerimientos del Proyecto

1. Arquitectura e Infraestructura (15%)

- Diseño e implementación de la arquitectura completa de microservicios en Kubernetes
- Configuración de un clúster Kubernetes (Minikube, Kind o solución cloud)
- Implementación de todos los microservicios del repositorio base
- Documentación detallada de la arquitectura con diagramas
- Implementación de Namespaces para separar ambientes (dev, qa, prod)
- Configuración correcta de dependencias entre servicios respetando la arquitectura original

2. Configuración de Red y Seguridad (15%)

- Servicios Kubernetes configurados correctamente (ClusterIP, NodePort)
- Implementación de Ingress Controller para el acceso a la API Gateway
- NetworkPolicies configuradas para restringir comunicación entre servicios siguiendo el patrón de arquitectura
- Configuración de TLS/HTTPS para endpoints públicos
- Implementación de ServiceAccounts con permisos mínimos necesarios (RBAC)
- Escaneo de imágenes en busca de vulnerabilidades
- Implementación de seguridad a nivel de pod (Pod Security Standards)

3. Gestión de Configuración y Secretos (10%)

- Migración de todas las configuraciones de Spring Boot a ConfigMaps
- Implementación de Secrets para credenciales y datos sensibles
- Implementación de rotación de secretos
- Uso de variables de entorno y/o volúmenes para inyectar configuraciones
- Gestión centralizada de la configuración (aprovechando el servicio de Cloud Config)
- Implementación opcional de Sealed Secrets o HashiCorp Vault

4. Estrategias de Despliegue y CI/CD (15%)

- Pipeline de CI/CD completo con Jenkins, GitLab CI o GitHub Actions para todos los microservicios
- Implementación de Canary Deployments para los servicios orientados al cliente
- Implementación de Blue-Green Deployments para servicios críticos
- Configuración de pruebas automatizadas como gate para los despliegues
- Estrategia de rollback automatizado ante fallos
- Uso de Helm Charts para empaquetar los microservicios
- Implementación de dependencias correctas en el orden de despliegue

5. Almacenamiento y Persistencia (10%)

- Implementación de Persistent Volumes y Persistent Volume Claims para las bases de datos
- Configuración de StorageClass adecuada para el entorno
- Backup y restauración de datos persistentes
- Implementación de bases de datos con replicación (opcional)
- Estrategia para migraciones de esquemas de base de datos
- Gestión adecuada del estado para servicios stateful

6. Observabilidad y Monitoreo (15%)

- Implementación completa de Prometheus + Grafana
- Aprovechamiento de los Actuator endpoints de Spring Boot para monitoreo
- Configuración de alertas para situaciones críticas
- Sistema de logging centralizado (ELK, Loki, o similar)
- Tracing distribuido con Jaeger o similar (integrando con Zipkin que ya está en la arquitectura)
- Instrumentación de aplicaciones para exponer métricas
- Dashboards personalizados para diferentes stakeholders (técnicos y de negocio)
- Monitoreo de las comunicaciones entre servicios

7. Autoscaling y Pruebas de Rendimiento (10%)

- Implementación de Horizontal Pod Autoscaler para todos los microservicios
- Implementación de KEDA para escalado basado en eventos
- Métricas personalizadas para autoscaling (no solo CPU/memoria)
- Pruebas de estrés con JMeter, Locust o similar
- Análisis de resultados y optimización de recursos
- Implementación de Quality of Service (QoS) classes
- Pruebas de carga que simulen escenarios de uso real de la aplicación e-commerce

8. Documentación y Presentación (10%)

- Documentación técnica completa y detallada
- Repositorio Git organizado con README claro
- Manual de operaciones para el sistema
- Video demostrativo del funcionamiento
- Presentación del proyecto
- Documentación de la arquitectura original y los cambios realizados para Kubernetes

Bonificaciones

Integración Cloud

- Implementación en un proveedor cloud (AWS, GCP, Azure)
- Uso de servicios cloud nativos para complementar la solución
- Configuración de múltiples zonas de disponibilidad
- Implementación de cloud-native storage

GitOps

- Implementación de ArgoCD o Flux para sincronización GitOps
- Repositorio de configuración declarativa

- Rollbacks automatizados basados en GitOps
- Implementación de Progressive Delivery

Service Mesh

- Implementación de Istio, Linkerd o similar
- Configuración de mTLS entre todos los servicios
- Implementación de retries, circuit breakers y traffic shifting
- Visualización del tráfico entre servicios

Seguridad Avanzada

- Implementación de Open Policy Agent (OPA) o Kyverno
- Configuración de Pod Security Policies o Pod Security Standards
- Escaneo continuo de vulnerabilidades en el pipeline
- Implementación de recursos seguros por defecto
- Pentesting documentado de la aplicación

Chaos Engineering

- Implementación de Chaos Mesh o Chaos Toolkit
- Pruebas de resiliencia documentadas para todos los servicios
- Análisis de resultados y mejoras implementadas
- Game day documentado con escenarios de fallo