Planificación del Pipeline de Integración y Entrega Continua (CI/CD)
Proyecto MVP – EduTech IA

1. Introducción
El objetivo del pipeline CI/CD es automatizar la integración, pruebas y despliegue del software, garantizando que cada cambio en el código se valide y despliegue de manera segura y rápida. Para este proyecto, se utilizará Jenkins (ejecutado en contenedores Docker) integrado con GitHub para orquestar todas las etapas del proceso, permitiendo iteraciones ágiles y un feedback inmediato.

2. Herramientas Empleadas
GitHub:
Repositorio centralizado donde se aloja el código fuente. Mediante webhooks, cada commit o pull request activa el pipeline en Jenkins.

Jenkins (en Docker):
Servidor de integración continua encargado de ejecutar el pipeline. La utilización de contenedores Docker para Jenkins asegura la portabilidad y consistencia entre entornos.

Docker:
Permite construir imágenes de la aplicación que se ejecutarán en entornos de pruebas y producción, garantizando que la aplicación se comporte de manera idéntica en todos los escenarios.

3. Flujo del Pipeline CI/CD

a) Commit y Notificación:
Los desarrolladores realizan commits en ramas del repositorio GitHub.
Un webhook notifica a Jenkins sobre los cambios en el código.

b) Etapa de Build:
Clonación: Jenkins clona el repositorio.
Compilación: Se compila el proyecto Java, generando artefactos (por ejemplo, JAR/WAR).

c) Etapa de Pruebas:
Pruebas Unitarias e Integración: Se ejecutan pruebas automatizadas para validar la funcionalidad del código.
Validación: Si alguna prueba falla, el pipeline se detiene y se envían notificaciones a los desarrolladores.

d) Packaging y Contenerización:
Los artefactos se empaquetan en una imagen Docker.
La imagen se construye y se almacena en un registro de contenedores (p.ej., Docker Hub o un repositorio privado).

e) Despliegue en Entorno de Pruebas (Staging):
La imagen Docker se despliega en un entorno de pruebas.
Se realizan pruebas adicionales (automatizadas y/o manuales) para validar la funcionalidad completa de la aplicación.

f) Despliegue a Producción:
Una vez aprobadas las pruebas en staging, se despliega la imagen en el entorno de producción.
Se pueden implementar estrategias de despliegue seguro (blue-green o rolling updates) para minimizar riesgos.

g) Notificaciones y Monitoreo:
Jenkins envía notificaciones del estado del pipeline (éxito o fallo) a través de correo electrónico o integraciones con herramientas de mensajería (por ejemplo, Slack).
Se configuran alertas y sistemas de monitoreo para supervisar el rendimiento y la estabilidad post-despliegue.

4. Beneficios del Pipeline

Automatización Completa: Minimiza errores humanos y acelera el ciclo de entrega.
Feedback Inmediato: Permite detectar y corregir problemas en etapas tempranas del desarrollo.
Consistencia en los entornos: Docker garantiza que la aplicación se ejecute de forma idéntica en desarrollo, pruebas y producción.
Colaboración Eficiente: La integración con GitHub facilita la revisión y fusión de cambios, mejorando la coordinación entre equipos.