

Actividad 4: Planificación del Pipeline de Integración y Entrega Continua (CI/CD)

1. Herramientas seleccionadas

- Repositorio de código: GitHub
- Gestión de versiones: Git
- Automatización de CI/CD: GitLab CI/CD
- Contenedorización: Docker
- Orquestación: Kubernetes (opcional para escalabilidad)

2. Etapas del Pipeline CI/CD

- Commit y Control de Versiones:
Acción: Desarrolladores suben código a un repositorio Git con ramas de desarrollo (develop) y producción (main).
- Integración Continua (CI):
Activación: Un nuevo commit en develop activa el pipeline en GitLab CI/CD.

Pasos:

- Instalación de dependencias (Node.js)
- Análisis estático del código (SonarQube, Junit)
- Ejecución de pruebas de integración
 - Si las pruebas fallan, el pipeline se detiene y notifica a los desarrolladores.

3. Construcción de la Aplicación

- Generación de contenedores Docker
Se genera una imagen Docker con la versión del código y se almacena en un Docker Registry (DockerHub/GitLab)

4. Despliegue en Entorno de Staging

- Automatización con Docker Compose/Kubernetes
 - a) Se despliega la aplicación en un entorno de staging para pruebas de aceptación.
 - b) Pruebas end-to-end (Cypress, Selenium).
 - c) Pruebas de carga y seguridad.

5. Aprobación Manual para Producción

- Si las pruebas en staging son exitosas, un revisor aprueba el despliegue a producción.

6. Despliegue en Producción (CD)

- Se usa Rolling Update o Blue-Green Deployment con Kubernetes.
 - a. Actualización de la infraestructura en AWS/GCP.
 - b. Limpieza de imágenes y contenedores antiguos.

7. Monitoreo y observabilidad

- Se implementan Prometheus y Grafana para monitorear métricas del sistema.
- Logs centralizados con ELK Stack (Elasticsearch, Logstash, Kibana).
- Alertas automatizadas en caso de errores críticos.