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

Para la organización correcta de un proyecto, es necesario establecer un Pipeline con metas específicas y tangibles, y asegurarse que cada fase se ha completado correctamente antes de pasar a la siguiente, evitando en la medida de lo posible tener que volver hacia atrás, pero en ocasiones no se puede evitar, por lo que hay te tener claros en que etapa se encuentran esos errores. En este Pipeline contaremos con ocho etapas: Control de versiones, construcción y dependencias, análisis del código estático, pruebas automáticas, despliegue del entorno de pruebas, aprobación natural y despliegue a producción y finalmente monitoreo y alertas.

1. Control de versiones

El objetivo de esta fase es asegurar la estabilidad del código, además de crear backups en caso de cualquier error a la hora de cambiar el código. Para ello, el equipo de desarrolladores estará trabajando en un mismo repositorio en github, creando diversas ramas para una mayor organización.

2. Construcción y dependencias.

En esta etapa compilaremos el código y gestionaremos todas las dependencias necesarias, GitHub Actions/Jenkins ejecuta la compilación del código, por ejemplo, con Maven/Gradle en Java. Después se verificará la instalación de las dependencias, usando npm, pip o Gradle y finalmente se generará un artefacto empaquetado en .jar.

3. Análisis del código estático

Este paso asegurará la calidad del código, comprobando que no se encuentren fallos de seguridad o malas prácticas. Para la comprobación usaremos SonarQube.

4. Pruebas automáticas

El objetivo será certificar la estabilidad y funcionalidad del sistema, usando diferentes métodos para las comprobaciones, como pruebas unitarias con Junit, pruebas de integración simulando conexiones con bases de datos y APIs y pruebas funcionales con Selenium para validar la interfaz.

5. Despliegue en un entorno de pruebas

Para este paso usaremos Docker, lo que nos permitirá crear un contenedor aislado, que nos permite seguir trabajando con el código en un entorno controlado, por lo que se podrá detectar errores sin que afecten en gran medida.

6. Aprobación manual y despliegue a producción

Aquí llegamos si todas las anteriores etapas han sido completadas con éxito, si no encontramos más fallos en el código y todo funciona correctamente desplegaremos la MVP.

7. Monitoreo y alertas

Una vez el software ya es accesible al público solo falta su mantenimiento, tendremos que monitorear su rendimiento y si ha cumplido las expectativas presentadas al comienzo del proyecto. Integraremos herramientas de monitoreo como Prometheus y configuraremos alertas con Sentry en caso de errores.