

# Pipeline

implementación de un producto digital

# Integración Continua



#### Flujo normal Lineal:



## Integración Continua







2. Construcción y Jpdates automáticos

Jenkins



Revisión Código estático



5b. Update binario



4. Publicar



3a. Calidad



Desarrollo

5a. Despliegue en Test

Repositorio Binario

**Vulnerabilidades** 

Entorno de testing

6. Testing funcional



en Producción



7. Despliegue

#### 1. Push



1. Push



Repositorio



Desarrollo

Los desarrolladores (puestos locales) utilizarán herramientas de desarrollo como pueden ser:

Intellij: Para Java

Una vez las modificaciones hayan sido realizadas se realizará un Push, enviandose a al repositorio GitLab.

#### 2. Build y testing automáticos



Una vez en el repositorio, Gitlab informa a Jenkins a través de webhooks de que se ha realizado un push de nuevo código. Entonces Jenkins se descarga de Gitlab el nuevo código para compilar y, usando herramientas como Junit, TestNG o JMeter, lanza los **test automáticos**:



2. Construcción y Updates automáticos



- Tests unitarios, que prueban pequeñas partes aisladas del código.
- Tests de integración, que prueban cómo varias piezas trabajan juntas.
- Tests de regresión, que comprueban la compatibilidad con versiones anteriores.
- Tests de rendimiento, que verifican que el rendimiento está dentro de lo aceptable.

Si los test son correctos, pasaríamos a la siguiente fase de verificación estática del código.

### 3. Calidad y seguridad



Cuando ya se han realizado todos los test unitarios (automáticos) y estos han salido favorablemente, se procede a realizar la revisión estática del código y comprobación de vulnerabilidades. Para ello utilizamos

- Sonarqube: Revisa el código y genera un informe que ofrezca el resultado de dichas revisiones.
- OWASP: Comprueba si el código o las librerías están afectados por problemas de seguridad conocidos.



Vulnerabilidades

# 4. Publicación del binario



Una vez construido el binario, pasados los tests, y revisada la calidad del código, el binario generado está listo para ser publicado. Para ello utilizamos

 Nexus: Es un repositorio de archivos binarios, utilizado para almacenar el software generado junto con sus versiones. Las versiones finalistas (releases) guardadas en Nexus no pueden ser modificadas a posteriori, garantizando que no haya dos binarios diferentes con el mismo número de versión.







4. Publicar

Repositorio Binario

# 5. Despliegue



El binario almacenado en Nexus es el que se despliega, es decir, que se lleva a ejecutar a los entornos de testing y producción.

Si el binario es además usado como dependencia por otras aplicaciones, como por ejemplo una librería, también es actualizado localmente por el desarrollador



Desarrollo

Entorno de testing / producción

5b. Update binario



5a. Despliegue

Repositorio Binario

En Integración Continua, éste despliegue implica intervención humana, bien para hacer el despliegue, o para aprobarlo. Si el despliegue también fuera automático, hablaríamos de Entrega Continua.



# 6,7<sub>y</sub>8. Testing funcional y Despliegue en producción

Entorno de testing

7. Testing funcional



8. Despliegue en Producción



Entorno de producción

6. Despliegue en Testing



6. Una vez tenemos la aplicación versionada en Nexus, Jenkins se encarga de desplegarla en un entorno de testing.

7. Es el momento de que las herramientas de testing funcional simulen cómo un usuario interacciona con la aplicación, para comprobar que ésta hace lo que se espera.

8. Si los tests funcionales son satisfactorios, Jenkins desplegará la aplicación en el entorno de producción, que es al que acceden los usuarios finales.







La aplicación desplegada en producción tiene que ser monitorizada para verificar su correcto funcionamiento.

Nagios nos ayuda en la monitorización de la red, los servidores y la aplicación, alertando ante cualquier incidencia.

ELK nos recopila los logs de los servidores y aplicación, para su análisis posterior.