SESIÓN 13 – CI/CD

Integración Continua y Despliegue Continuo con Jenkins, GitHub Actions y GitLab CI

INTRODUCCIÓN

Imaginad un proyecto con 5 personas programando.

- Cada uno sube código cuando quiere.
- Nadie prueba todo el proyecto entero.
- Un día se sube a producción... y falla.

Esto no es un caso teórico, es lo que pasaba (y pasa) sin CI/CD.

Problemas del desarrollo tradicional

- Las pruebas se hacen **tarde**
- Los errores se descubren **cuando ya es grave**
- Se depende de personas (“acuérdate de probar...")
- Los despliegues dan miedo
- Mucho “en mi ordenador funciona”

Necesitamos **automatizar** el proceso.

DEVOPS Y CI/CD:

¿Qué es DevOps?

DevOps **no es una herramienta**.

Es una **forma de trabajar** donde:

- Desarrollo (Dev)
- Operaciones (Ops)

Colaboran y **automatizan** todo el ciclo del software.

¿Dónde encaja CI/CD?

CI/CD es **el corazón técnico de DevOps**.

Piensa en CI/CD como: Un robot que revisa tu código cada vez que lo subes.

INTEGRACIÓN CONTINUA (CI)

Integración Continua (CI) es la práctica de **integrar y verificar automáticamente** los cambios de código cada vez que se suben a un repositorio.

¿Qué significa "integrar"?

No es solo subir código.

Integrar significa:

- Unir tu código con el del resto del equipo
- Comprobar que **todo sigue funcionando**

¿Qué hace un sistema de CI?

Cada vez que ocurre un evento (normalmente push o commit):

1. Descarga el código
2. Compila el proyecto (si procede)
3. Instala dependencias
4. Ejecuta tests
5. Analiza errores

Todo esto es automático

¿Por qué son tan importantes los tests?

Explícalo con ejemplo:

- Cambias una función
- Rompes otra sin darte cuenta

- El test falla
- El pipeline se detiene

El error no llega a producción

Resultado del CI

- Verde: el código es válido
- Rojo: hay un error que corregir

CI no evita errores, **los detecta pronto.**

DESPLIEGUE CONTINUO (CD)

Despliegue Continuo (CD) es la práctica de **publicar automáticamente** una versión del software cuando el CI ha sido exitoso.

No siempre significa producción.

CD puede desplegar en:

- Desarrollo
- Staging
- Preproducción
- Producción

Flujo completo CI/CD

Explícalo como una cadena lógica:

1. Programador hace push
2. CI verifica el código
3. Si todo está bien…
4. CD despliega automáticamente

Sin intervención humana

Diferencia CI vs CD

- CI: **verifica**
- CD: **publica**

¿QUÉ ES UN PIPELINE?

Definición clara

Un **pipeline** es una secuencia de pasos automáticos que se ejecutan en un orden definido.

Ejemplo sencillo de pipeline

1. Build
2. Test
3. Deploy

Cada paso:

- Depende del anterior
- Si uno falla: el resto no se ejecuta

Características clave de un pipeline

- Automático
- Repetible
- Trazable (logs)
- Seguro
- Versionado

Un pipeline es código que automatiza procesos

HERRAMIENTAS CI/CD

JENKINS

Qué es Jenkins

- Herramienta de CI/CD **open source**
- Muy usada históricamente
- Extremadamente configurable

Jenkinsfile

Los pipelines se definen en un archivo llamado: Jenkinsfile

Es código que describe:

- Qué hacer
- En qué orden
- Con qué condiciones

Ventajas

- Muy potente
- Miles de plugins
- Control total

Inconvenientes

- Instalación compleja
- Mantenimiento alto
- Más pensado para empresas grandes

GITHUB ACTIONS

Qué es

- Sistema de CI/CD integrado en GitHub
- No requiere instalación
- Se configura con archivos YAML

Dónde está el pipeline

.github/workflows/

Ejemplo:

ci.yml

Funcionamiento

- Escucha eventos (push, pull_request)
- Ejecuta acciones
- Muestra resultados en GitHub

Ventajas

- Fácil de usar
- Perfecto para proyectos pequeños y medianos

GITLAB CI

Qué es

- CI/CD integrado en GitLab
- Muy limpio conceptualmente
- Muy usado en entornos profesionales

Archivo principal

.gitlab-ci.yml

Estructura

- stages
- jobs
- scripts

Ventajas

- Muy claro
- Muy robusto
- Excelente para trabajo en equipo

CONCEPTOS CLAVE

Secrets

Nunca se escriben:

- contraseñas
- claves SSH
- tokens

Se usan:

- variables de entorno
- secretos del sistema

Rollback

- Volver a una versión anterior
- Fundamental en producción
- Se apoya en versionado y pipelines

Notificaciones

- Slack
- Email
- Teams
- El equipo sabe **al instante** si algo falla.

RESUMEN

- CI automatiza build y test
- CD automatiza despliegue
- Pipeline = pasos automáticos
- Jenkins: potente pero complejo
- GitHub Actions : simple e integrado
- GitLab CI : profesional y limpio
- CI/CD mejora calidad, velocidad y seguridad