Museo de Dinosaurios

FastAPI

CI/CD & DevOps

Pipeline de Integración Continua

Fecha: 09 de February de 2026
Versión: 1.0

Tabla de Contenidos

1. Introducción a DevOps

2. Arquitectura CI/CD

3. Docker - Containerización

4. GitHub Actions - Pipeline

5. Flujo de Trabajo

6. Despliegue Automático

7. Monitoreo y Mantenimiento

8. Troubleshooting

1. Introducción a DevOps

DevOps es una metodología que combina desarrollo (Dev) y operaciones (Ops). Su objetivo es automatizar y optimizar el ciclo de vida del software, desde el desarrollo hasta la producción. En esta aplicación, hemos implementado un pipeline CI/CD completo que garantiza calidad, seguridad y velocidad de entrega.

Beneficios de CI/CD:

■ **Automatización:** Eliminación de tareas manuales

■ **Calidad:** Tests automáticos en cada cambio

■ **Velocidad:** Despliegues más rápidos

■ **Confiabilidad:** Menos errores en producción

■ **Trazabilidad:** Registro completo de cambios

■ **Seguridad:** Análisis automático de vulnerabilidades

2. Arquitectura CI/CD

La arquitectura implementada se compone de varios componentes que trabajan en conjunto:

Componentes Principales:

Componente	Función	Tecnología
Control de Versiones	Gestionar código fuente	Git / GitHub
CI/CD	Automatizar pipeline	GitHub Actions
Containerización	Empaquetar aplicación	Docker
Registro	Almacenar imágenes	Docker Hub
Deployment	Desplegar aplicación	Docker / Cloud

3. Docker - Containerización

Docker es una plataforma que permite empaquetar la aplicación y todas sus dependencias en un contenedor, garantizando que funcione igual en desarrollo, testing y producción.

Dockerfile Multi-stage:

Utilizamos un Dockerfile con dos etapas para optimizar el tamaño final de la imagen:

Stage 1: Builder

• Imagen base: python:3.11-slim

• Instala dependencias necesarias

• Compila las librerías Python

• Limpia archivos innecesarios

Stage 2: Runtime

• Copia solo lo necesario del builder

• Crea usuario no-root para mayor seguridad

• Variables de entorno optimizadas

• Health check configurado

Ventajas:

■ Menor tamaño final (eliminamos herramientas de compilación)

■ Mayor seguridad (usuario no-root)

■ Mejor rendimiento

■ Cacheo eficiente de capas

4. GitHub Actions - Pipeline

GitHub Actions es un servicio de integración continua integrado en GitHub. Ejecuta automáticamente tareas cuando ocurren eventos (push, pull request, etc.).

Jobs del Pipeline:

Job 1: Test

1. Checkout del código

2. Setup Python 3.11

3. Instala dependencias

4. Ejecuta linting (Flake8)

Trigger: En cada push o pull request

Obligatorio: Sí (debe pasar para continuar)

Job 2: Docker Build & Push

1. Setup Docker Buildx (herramienta de build mejorada)

2. Login a Docker Hub

3. Build imagen Docker

4. Push a Docker Hub

Trigger: Solo después de pasar tests

Solo en: Push a main/develop (no en PRs)

Resultado: Imagen disponible en Docker Hub

Job 3: Notifications

1. Verifica estado de todos los jobs anteriores

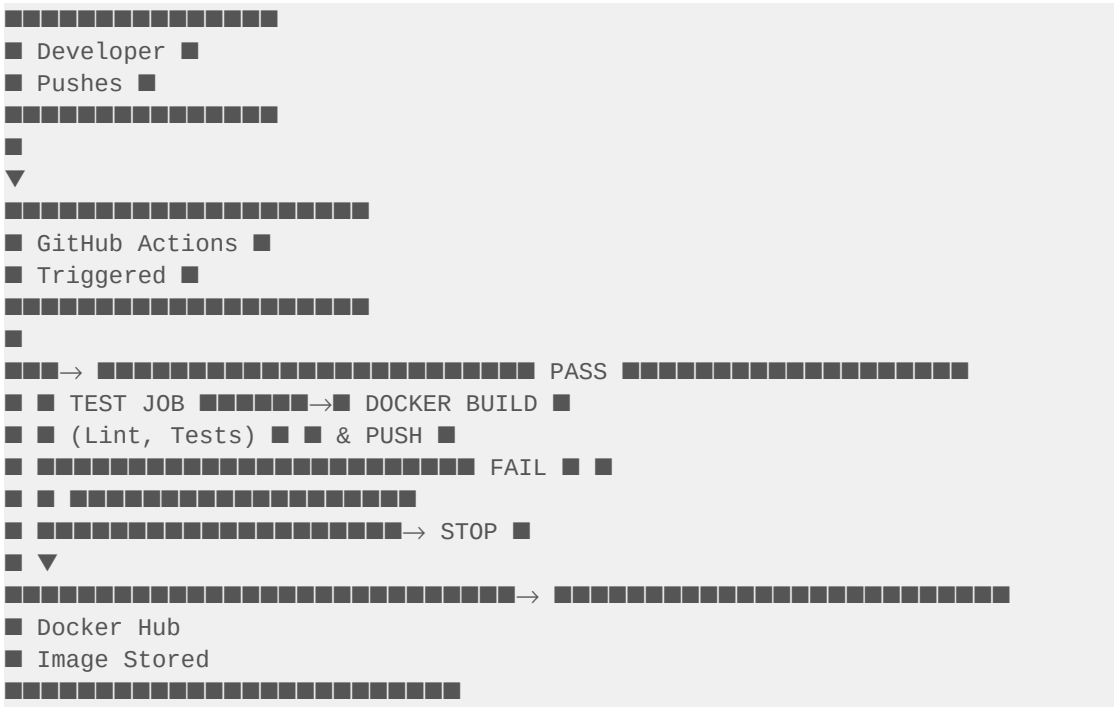
2. Notifica resultado final del pipeline

Útil para: Alertas por email, Slack, etc.

5. Flujo de Trabajo Completo

1. **Desarrollador** realiza cambios en el código
2. **Git push** a GitHub (a rama main o develop)
3. **GitHub detecta** el event (push) y dispara el workflow
4. **Job Test** se ejecuta (validación de código)
5. Si test **FALLA**: Pipeline se detiene, notifica error
6. Si test **PASA**: Continúa al siguiente job
7. **Job Build & Push** construye imagen Docker
8. **Sube a Docker Hub** la imagen con tags automáticos
9. **Job Notifications** notifica resultado final
10. **Imagen disponible** para despliegue en producción

Diagrama de Flujo:



6. Despliegue Automático

Tagging Strategy:

Las imágenes Docker se tagean automáticamente según la rama:

Tag	Cuándo	Ejemplo
latest	Rama main	usuario/museo:latest
develop	Rama develop	usuario/museo:develop
rama-nombre	Cualquier rama	usuario/museo:feature-x
main-sha	Commit hash	usuario/museo:main-a1b2c3d

Opciones de Deploy:

Opción 1: Docker Hub (Recomendado)

• Imagen se pushea automáticamente

• Cualquiera puede descargarla

• Bajo costo (gratuito para públicas)

• Comando: docker pull usuario/museo:latest

Opción 2: Heroku

• Deploy automático desde GitHub

• Gratuito (con limitaciones)

• Ideal para prototipo

Opción 3: AWS/Google Cloud

• Máxima escalabilidad

• Requiere configuración adicional

• Costo variable según uso

7. Monitoreo y Mantenimiento

Verificar ejecución del Pipeline:

1. Ir a GitHub → Tu repositorio → Actions

2. Seleccionar el workflow 'FastAPI CI/CD Pipeline'

3. Ver logs detallados de cada job

4. Descargar artefactos si es necesario

Métricas Importantes:

■ Tasa de éxito de tests (ideal: 100%)

■ Tamaño de imagen Docker (menor es mejor)

■■ Tiempo de ejecución del pipeline (< 10 min ideal)

■ Auditoría de cambios (quién, qué, cuándo)

■ Frecuencia de despliegues

8. Troubleshooting

Problema: El workflow no se ejecuta

• Revisar sintaxis del archivo YAML (validar en yamllint.com)

• Verificar que el archivo esté en .github/workflows/

• Confirmar que el trigger (push/PR) sea correcto

Problema: Docker build falla

• Verificar que Dockerfile sea válido

• Probar build localmente: docker build -t test .

• Revisar logs en GitHub Actions

Problema: Login a Docker Hub falla

• Verificar secrets configurados en GitHub Settings

• Revisar que token no esté expirado

• Regenerar token en Docker Hub si es necesario

Problema: Imagen muy grande

• Usar multi-stage Dockerfile (ya implementado)

• Limpiar archivos innecesarios

• Usar imagen base más pequeña (python:3.11-slim)

Conclusión

El pipeline CI/CD implementado proporciona una base sólida para el desarrollo, testing y despliegue automático de la aplicación. Permite a los equipos ser más productivos, garantiza la calidad del código y reduce significativamente el tiempo entre desarrollo e implementación en producción.

Checklist de Implementación:

■ Dockerfile multi-stage creado

■ docker-compose.yml configurado

■ .gitignore y .dockerignore creados

■ GitHub Actions workflow configurado

■ Documentación DevOps completada

■ Secrets configurados en GitHub (DOCKERHUB_USERNAME, DOCKERHUB_TOKEN)

■ Primer push a GitHub realizado

■ Workflow ejecutado exitosamente

■ Imagen subida a Docker Hub

Documento generado: 09/02/2026 a las 10:07:41