

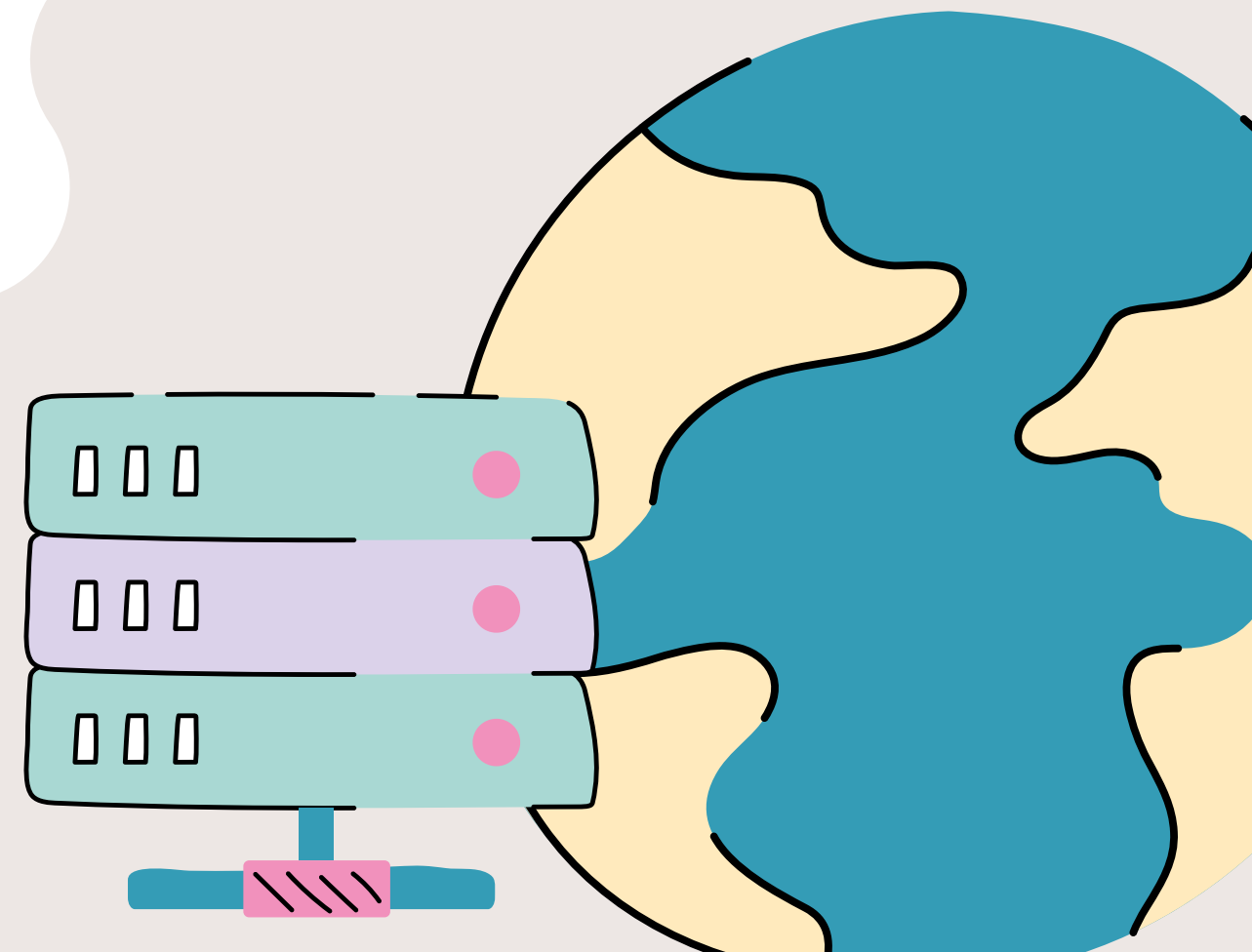


# Kubernetes

Automatiza, escala y gestiona tus aplicaciones en  
contenedores como un pro



Por Juan Duran



# ¿Qué es Kubernetes y por qué importa?

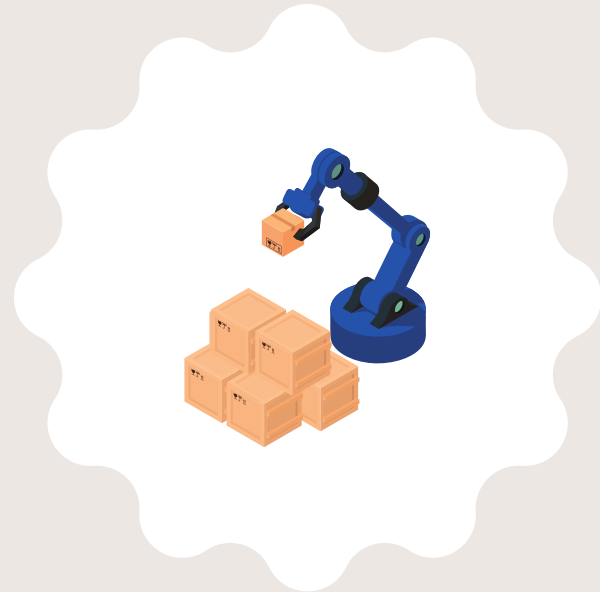
**Kubernetes** (K8s) es una **plataforma** de **orquestación** de contenedores de código abierto creada por **Google** y ahora mantenida por la CNCF (Cloud Native Computing Foundation).

Su objetivo es **automatizar** tareas como el **despliegue**, la **escalabilidad** y la **gestión** de aplicaciones en contenedores.

En la actualidad, con arquitecturas basadas en microservicios y despliegues constantes, Kubernetes se convierte en una herramienta imprescindible para el mundo **DevOps** y la **nube**. Su adopción está creciendo rápidamente en entornos empresariales y proyectos de todo tipo.



# Puntos clave



## Automatización

Kubernetes gestiona **automáticamente** los ciclos de vida de tus aplicaciones: **despliegue, escalado, reinicio** en caso de fallos y actualizaciones sin downtime.



## Contenedores

Permite ejecutar desde unas pocas hasta miles de **instancias** de **contenedores** distribuidos en múltiples servidores.








## Infraestructura

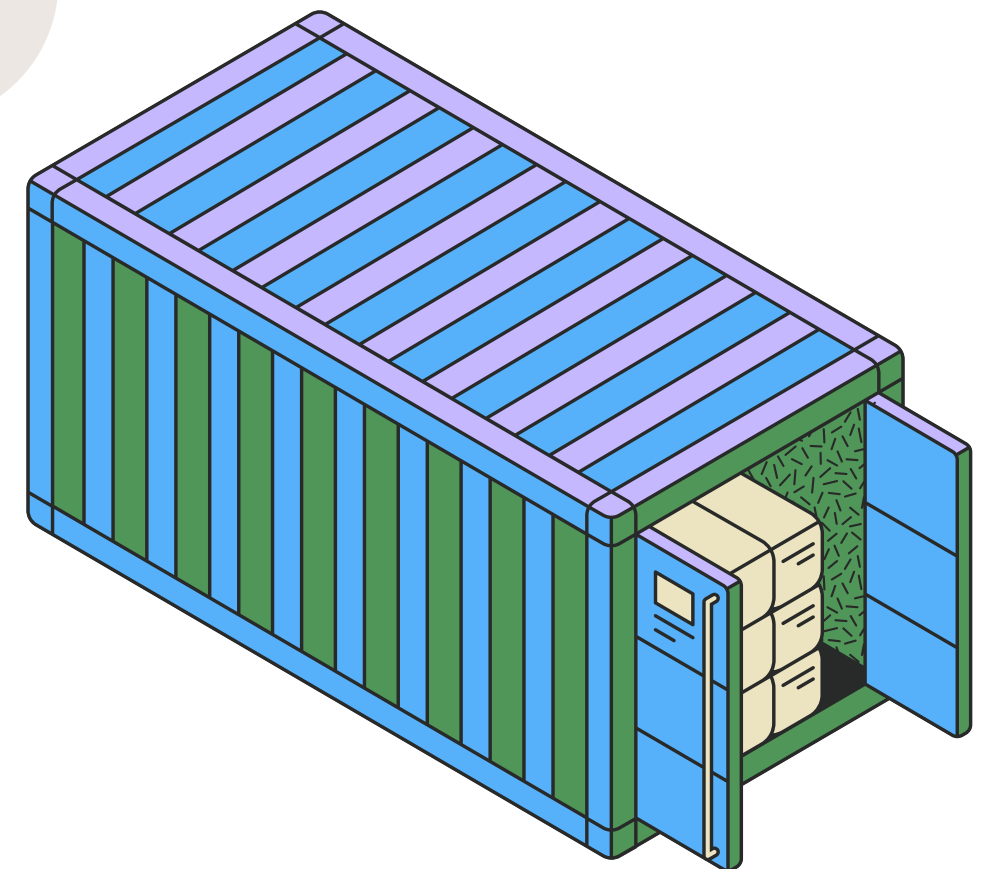
Kubernetes no está ligado a ningún proveedor. Puedes usarlo en **local**, en la **nube** (AWS, GCP, Azure) o en entornos híbridos.

# ¿Por qué usar contenedores?

Los **contenedores**, como los creados con **Docker**, permiten empaquetar una aplicación junto a todas sus dependencias, asegurando que se ejecute igual en cualquier entorno.

Ventajas principales:

-  **Aislamiento** entre servicios
-  **Portabilidad** (desarrollo, test y producción iguales)
-  **Arranque** rápido
-  **Menor consumo** de recursos frente a máquinas virtuales
-  Perfecto para **arquitecturas modernas** como microservicios



# Problemas sin Kubernetes

Sin Kubernetes, gestionar aplicaciones basadas en contenedores puede ser un caos. Algunos problemas típicos:

- **✗ Dificultad** para **escalar** manualmente
- **✗** Falta de **tolerancia** a fallos automática
- **✗** Mala **distribución** de recursos
- **✗ Dificultad** para realizar **despliegues** sin interrumpir el servicio
- **✗ Mayor complejidad** en la gestión operativa

Kubernetes resuelve todos estos desafíos con su enfoque declarativo y automatizado.



# Pros

✓ **Ventajas** de usar Kubernetes:

- **Escalado** automático de aplicaciones
- Alta **disponibilidad** y recuperación ante fallos
- **Despliegues** sin interrupciones
- **Compatible** con múltiples plataformas
- Gran **comunidad** y **ecosistema**

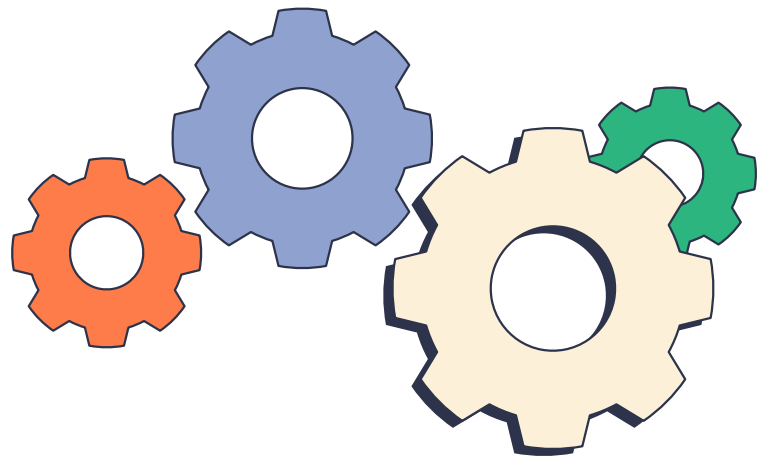


# Contras

✗ **Desventajas** o retos:

- Curva de **aprendizaje** inicial
- **Configuración** compleja (YAMLs, recursos, etc.)
- Necesita **conocimientos** de redes, almacenamiento y seguridad
- **Sobrecoste** para proyectos muy pequeños

# Componentes principales de Kubernetes



- **Pod:** Unidad básica que contiene uno o más contenedores que comparten red y almacenamiento.
- **Node:** Máquina física o virtual donde se ejecutan los Pods.
- **Cluster:** Conjunto de nodos gestionado como una sola unidad.
- **Control Plane:** Coordina el estado del clúster y toma decisiones.
- **Kubelet:** Agente que corre en cada nodo para asegurar que los contenedores estén ejecutándose correctamente.

# Cómo funciona Kubernetes



Kubernetes utiliza un **enfoque declarativo**. Tú defines el estado deseado de tu aplicación (por ejemplo, 3 instancias de una API), y Kubernetes se encarga de alcanzarlo y mantenerlo.

📌 Si un contenedor falla, lo reinicia.

📌 Si aumenta la demanda, puede escalar automáticamente.

📌 Si actualizas tu app, puedes hacer un “rolling update” sin interrupciones.



# Despliegues y operación

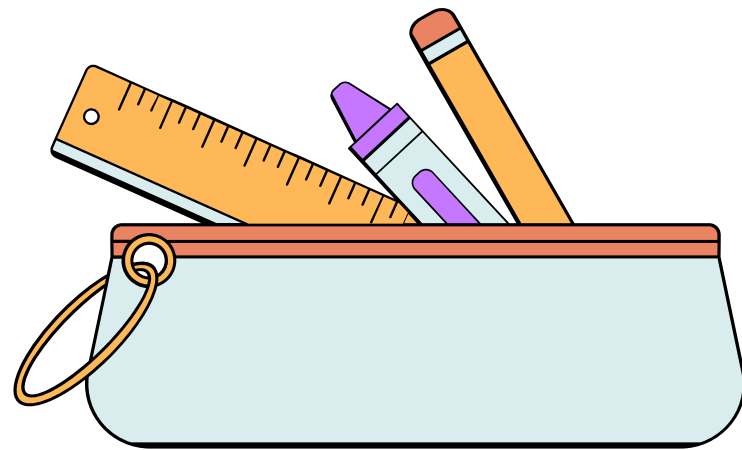


Kubernetes permite definir toda tu **infraestructura** como código usando archivos **YAML**. Esto da lugar a **despliegues reproducibles** y **auditable**s.

Algunas funcionalidades clave:

- **Services:** Exponen tus Pods internamente o al exterior.
- **Ingress:** Gestiona rutas y tráfico entrante con reglas HTTP.
- **Volumes:** Manejo avanzado de almacenamiento persistente.
- **Autoscalado:** Escala tus apps automáticamente según CPU o uso de memoria.

# Casos de uso reales



Kubernetes es usado en producción por grandes empresas y proyectos:

- 🌐 **Spotify**: Para escalar microservicios musicales.
- ✈️ **Airbnb**: Gestión de servicios globales con alta disponibilidad.
- 🧪 **CERN**: Para gestionar cargas de trabajo científicas.
- 🏛️ **BBVA, ING, Shopify**: Migraciones de arquitecturas monolíticas a microservicios.

Se integra fácilmente con herramientas del ecosistema como:

- **Helm** (gestión de paquetes)
- **Prometheus** (monitorización)
- **Istio** (servicio mesh)
- **ArgoCD** (despliegue continuo)



# Conclusiones



## **Automatización**

Kubernetes automatiza tareas críticas de operación de contenedores.

## **Escala**

Permite escalar aplicaciones y mantenerlas disponibles.

## **Entornos**

Funciona en múltiples entornos: local, nubes públicas o privadas.

## **DevOps**

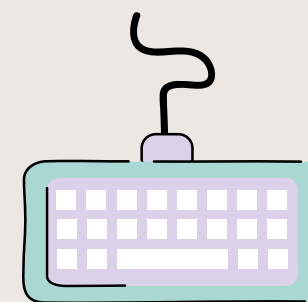
Es ideal para arquitecturas modernas y equipos DevOps.

## **Aprendizaje**

Requiere inversión inicial de aprendizaje, pero vale la pena.

## **Salidas laborales**

Aprender Kubernetes abre puertas en cloud, desarrollo y operaciones.



# Gracias



Por Juan Duran

**“Coding, Gaming and Leveling Up”**