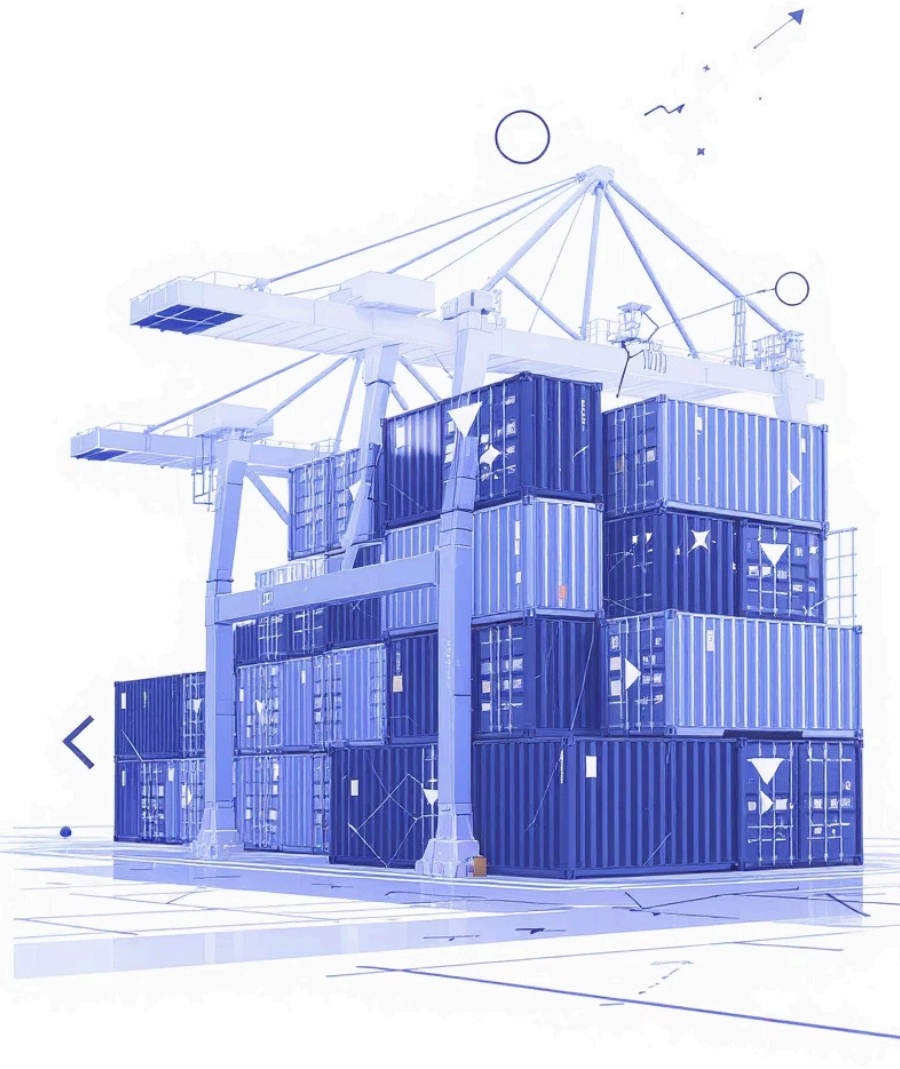
Dateneo®

Docker para Ingeniería de Datos

Contenedores, reproducibilidad y orquestación

Una introducción técnica para ingenieros de datos que buscan dominar contenedores y optimizar sus pipelines. Exploraremos desde los fundamentos hasta las mejores prácticas en entornos de producción.



Contexto histórico: la evolución hacia los contenedores

Antes: Máquinas Virtuales

Cada VM incluía su propio sistema operativo completo:

- Pesadas (GBs de almacenamiento)
- Arranque lento (minutos)
- Alto consumo de recursos
- Aislamiento completo pero ineficiente

Como tener un edificio completo para cada aplicación.

Ahora: Contenedores

Comparten el kernel del sistema operativo host:

- Ligeros (MBs en lugar de GBs)
- Arranque instantáneo (segundos)
- Eficientes en recursos
- Aislamiento suficiente y práctico

Como tener apartamentos en un mismo edificio.

La revolución de Docker

Docker popularizó los contenedores y transformó el desarrollo y despliegue de software desde 2013.

Portabilidad	Aislamiento	Reproducibilidad
Mismo comportamiento garantizado en cualquier entorno: desarrollo local, servidores on-premise o infraestructura cloud.	Cada aplicación opera en su propio entorno sin interferencias, evitando conflictos de dependencias o configuraciones.	Resultados consistentes y predecibles al eliminar las variaciones de entorno y el clásico "en mi máquina funciona".

> "Docker es el invento más importante del siglo XXI... la nube básicamente son contenedores corriendo."

La analogía de la cocina

Misma receta, mismos resultados

Una imagen Docker es como una receta detallada con ingredientes exactos y pasos precisos. Garantiza que cualquier cocinero (o máquina) obtenga el mismo resultado final.

Cocina limpia cada vez

Cada contenedor arranca desde cero, sin residuos de ejecuciones anteriores. Como empezar en una cocina recién limpia y desinfectada para cada servicio.

Probar nuevas recetas sin miedo

Experimentar con nuevas versiones o configuraciones sin afectar tu entorno principal. Puedes probar PostgreSQL 15 sin desinstalar tu versión 13 actual.

Los contenedores permiten "cocinar" aplicaciones en entornos aislados pero eficientes, con ingredientes (dependencias) perfectamente controlados.

Conceptos fundamentales de Docker

Imagen

Técnicamente: Plantilla inmutable con código, dependencias y configuraciones.

En la cocina: La receta con lista de ingredientes y pasos detallados.

Contenedor

Técnicamente: Instancia en ejecución de una imagen con su propio espacio de procesos.

En la cocina: El plato preparado y servido según la receta.

Capas (layers)

Técnicamente: Cada instrucción del Dockerfile crea una capa cacheable para optimizar builds.

En la cocina: Como preparar una lasaña por capas, cada una bien definida.

Volúmenes

Técnicamente: Almacenamiento persistente independiente del ciclo de vida del contenedor.

En la cocina: La helader donde guardas ingredientes aunque apagues la cocina.

Redes

Técnicamente: Interfaces virtuales para comunicación entre contenedores.

En la cocina: Walkie-talkies entre cocineros de diferentes estaciones.

Registros de Imágenes: dónde residen tus 'recetas'

Las imágenes Docker se almacenan y distribuyen a través de 'registros' (registries), que son repositorios centralizados. Es como una gran biblioteca de "recetas" de software.

Registros públicos: Docker Hub

El registro por defecto y más grande, donde encontrarás imágenes oficiales de software popular (Python, Node.js, PostgreSQL) y millones de imágenes creadas por la comunidad. Es ideal para acceder a componentes estándar y compartir proyectos de código abierto.

Registros de nube: AWS ECR, GCP GCR, Azure ACR

Los proveedores de la nube ofrecen sus propios servicios de registro (Amazon ECR, Google Container Registry, Azure Container Registry). Estos se integran directamente con sus ecosistemas, facilitando la implementación y gestión de aplicaciones en sus plataformas.

Registros privados

Puedes configurar tu propio registro privado, ya sea on-premise o en la nube, para almacenar imágenes propietarias de tu organización. Esto te da control total sobre la seguridad, el acceso y las políticas de distribución de tus imágenes.

Aplicaciones en ingeniería de datos

Empaquetado y reproducibilidad

- Encapsular scripts de ingesta, transformación y carga en imágenes versionadas
- Garantizar entornos idénticos en desarrollo, pruebas y producción
- Eliminar la famosa frase "en mi máquina funciona" de los pipelines

Infraestructura como código

- Levantar servicios complejos localmente (Kafka, PostgreSQL, Spark)
- Crear entornos de prueba completos sin infraestructura costosa
- Integración con orquestadores (Airflow, Prefect, Dagster) que usan contenedores como unidades de ejecución

Docker transforma los pipelines de datos al estandarizar cada componente como un contenedor independiente pero interconectable, facilitando tanto el desarrollo como el despliegue.

Buenas prácticas en Docker para ingeniería de datos

Usar imágenes base livianas

Preferir variantes `slim` o `alpine`. Por ejemplo, `python:3.9-slim` en lugar de `python:3.9` para reducir el tamaño final hasta en un 70%.

Versionar imágenes con tags específicos

Nunca usar solo `latest`. Siempre especificar versiones inmutables como `miapp:1.0.0`. El tag `latest` es solo un alias que no garantiza consistencia entre despliegues.

Secretos y configuración en runtime

No hardcodear credenciales ni configuraciones específicas del entorno en la imagen. Inyectarlas mediante variables de entorno o archivos montados al ejecutar.

Logs a stdout/stderr

Dirigir todos los logs a la salida estándar para que sean capturados por el orquestador, facilitando el monitoreo centralizado.

Separar datos de la aplicación

Usar volúmenes para persistir datos. Los contenedores deben ser efímeros; cualquier dato importante debe sobrevivir a su ciclo de vida.

Ejemplo práctico: un pipeline de datos en Docker

Dockerfile para un script de procesamiento

```
FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install -r requirements.txt

COPY scripts/ .
ENV LOG_LEVEL=INFO

CMD ["python", "process_data.py"]
```

Este Dockerfile crea una imagen ligera con Python y nuestro script, configurando variables de entorno para controlar su comportamiento.

Construir y ejecutar el contenedor

Construir la imagen:

```
docker build -t mi-pipeline:1.0.0 .
```

Ejecutar el contenedor:

```
docker run -v ./data:/app/data \
        -e DATABASE_URL=postgres://user:pass@db:5432/data \
        mi-pipeline:1.0.0
```

El volumen `./data` permite que el contenedor acceda a archivos locales, mientras que las variables de entorno configuran la conexión.

Docker: la base de los pipelines de datos modernos

Beneficios clave para ingeniería de datos

1	2	3
Reproducibilidad	**Portabilidad**	**Eficiencia**
Ejecuciones consistentes independientemente del entorno, eliminando errores por diferencias de configuración.	Desarrollo local, pruebas y producción con exactamente el mismo entorno y dependencias.	Menor consumo de recursos, despliegue más rápido y mejor aprovechamiento de infraestructura.