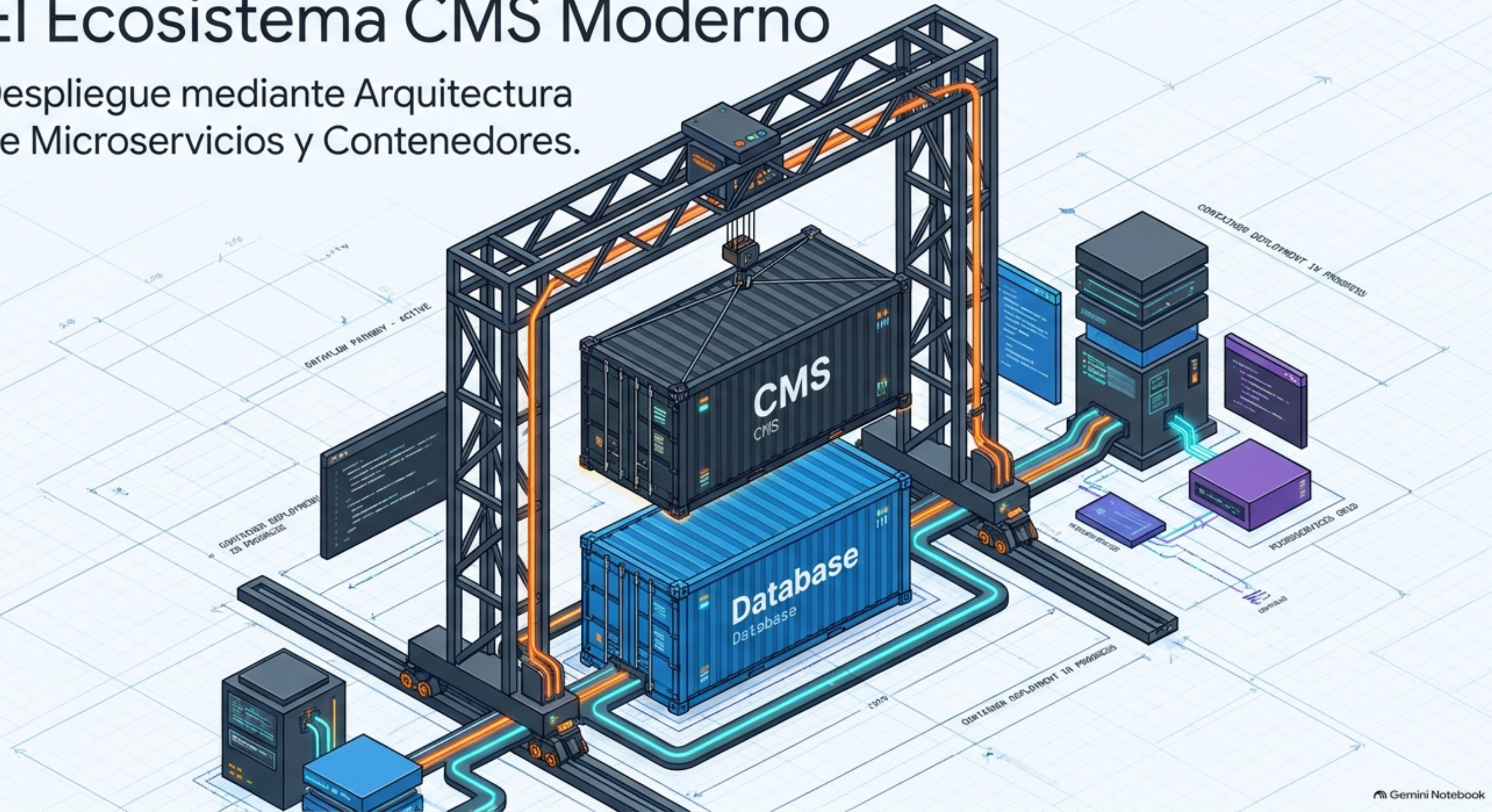


El Ecosistema CMS Moderno

Despliegue mediante Arquitectura de Microservicios y Contenedores.



Anatomía Básica de un Gestor de Contenidos

Definición Core: Un CMS es una interfaz accesible vía navegador web que separa el diseño de la información, eliminando la necesidad de maquetar código manualmente.

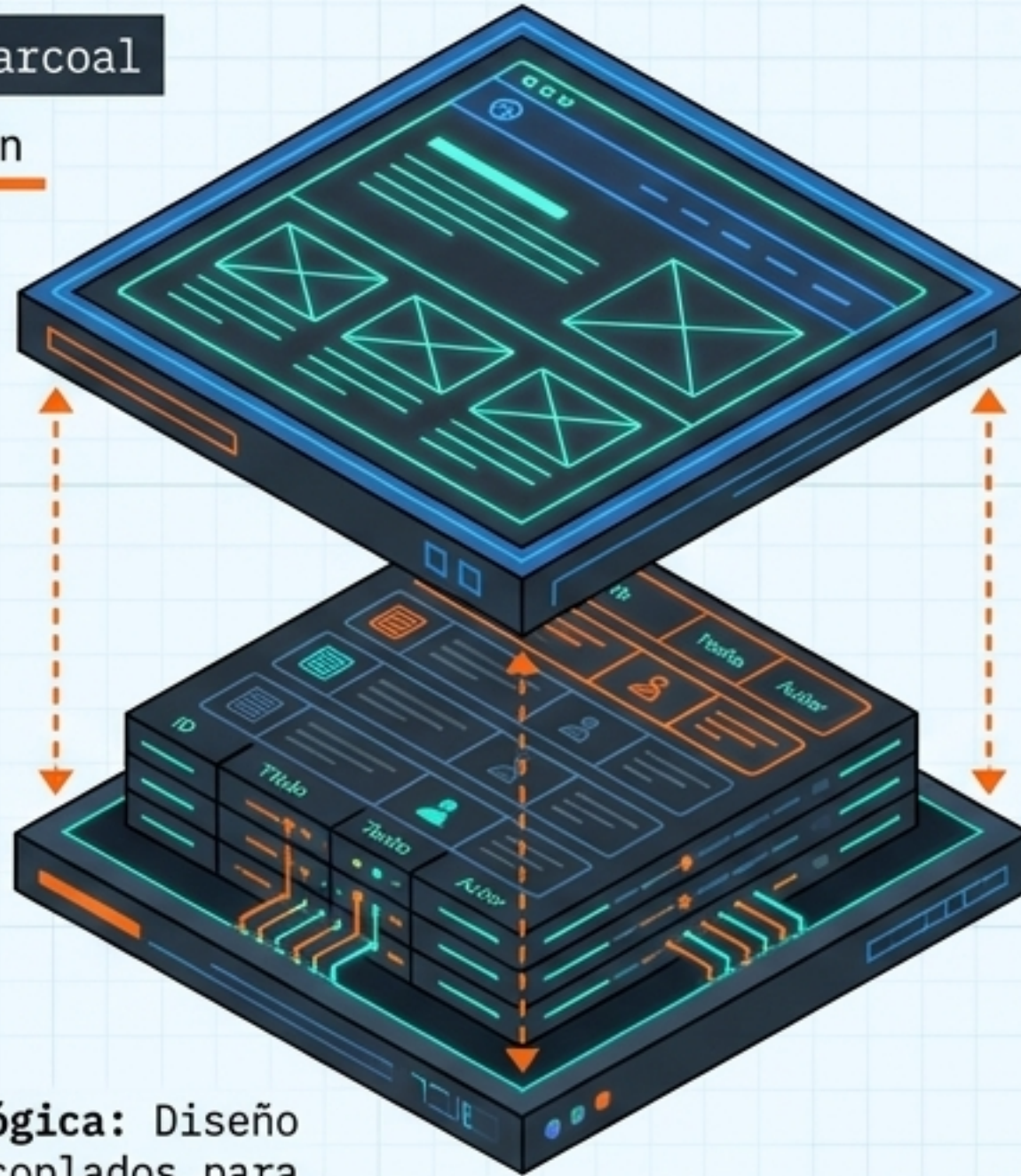
Terminal Charcoal

Presentación

Contenido

Contenido

Separación Lógica: Diseño y Datos Desacoplados para Flexibilidad y Gestión Eficiente.

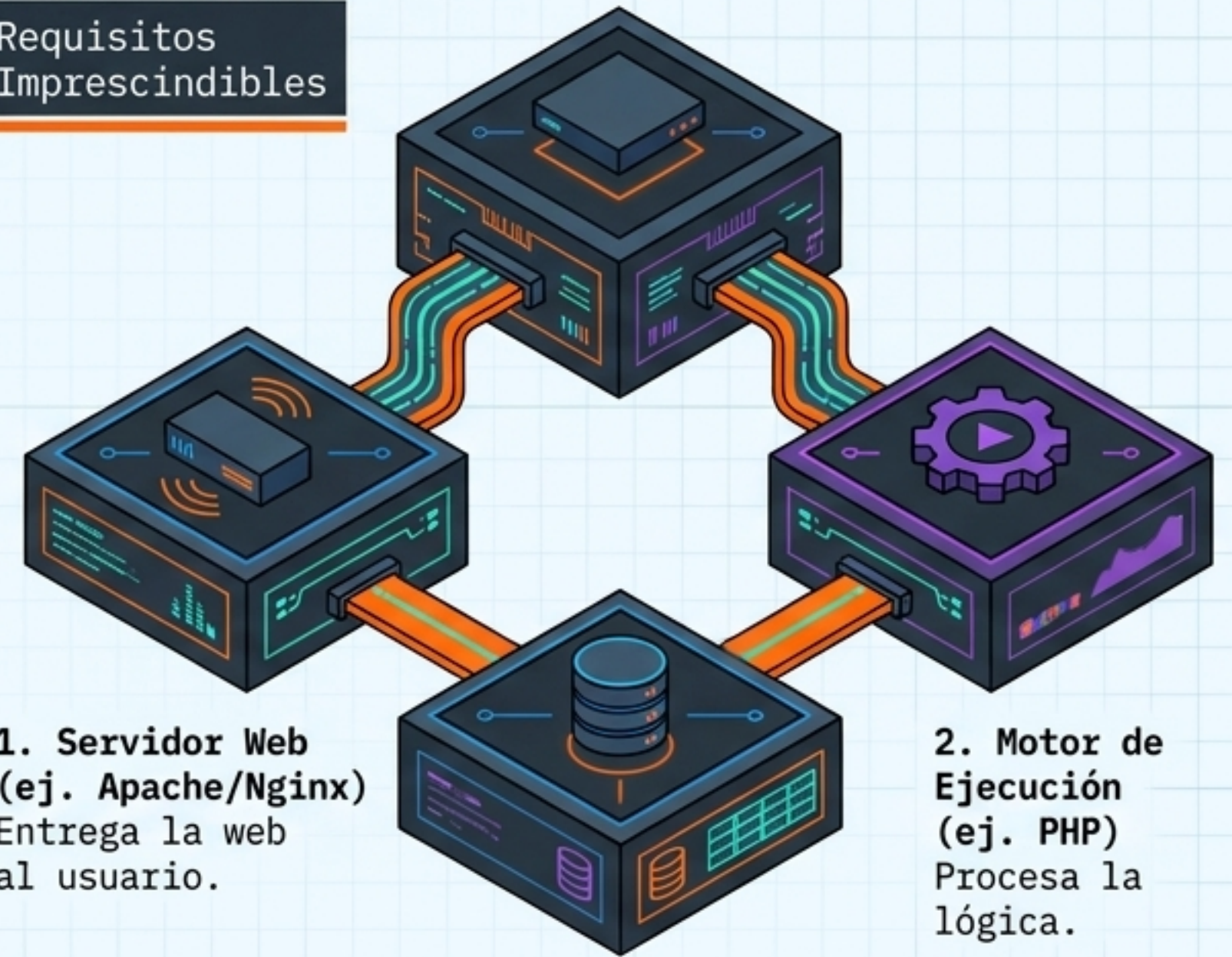


Requisitos
Imprescindibles

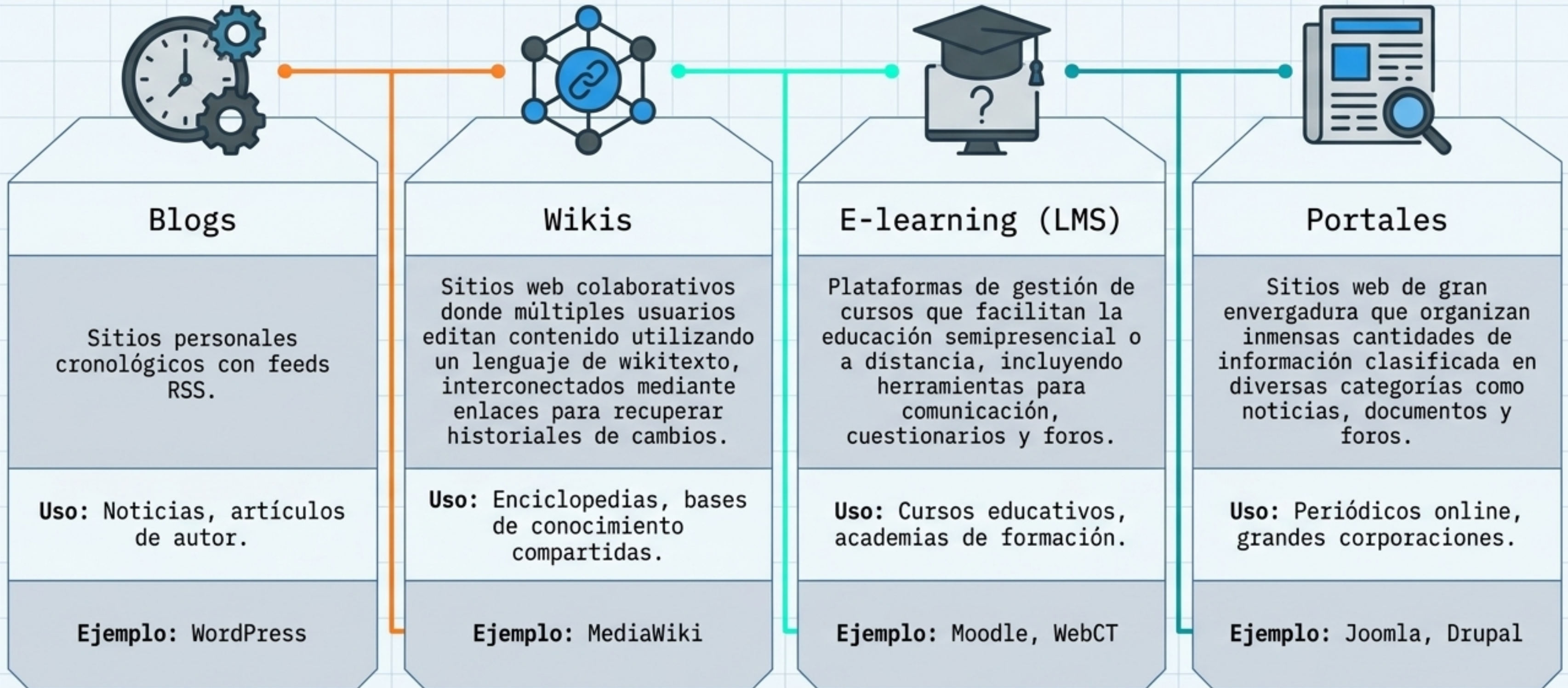
1. Servidor Web
(ej. Apache/Nginx)
Entrega la web
al usuario.

3. Base de Datos (SGBD)
(ej. MariaDB)
Almacena el contenido.

2. Motor de
Ejecución
(ej. PHP)
Procesa la
lógica.



Matriz de Tipología CMS



Las Reglas de Distribución: Licencias

The Baseline



Copyright (©)
Todos los derechos reservados.



Copyleft (C)
Distribución y modificación libre,
siempre que las obras derivadas
mantengan la misma licencia.



Completa



Parcial

Creative Commons Formulas



= Reconocimiento (BY):
Citar al autor.



= Sin Obra Derivada (ND):
Distribución exacta.



= No Comercial (NC):
Sin fines de lucro.



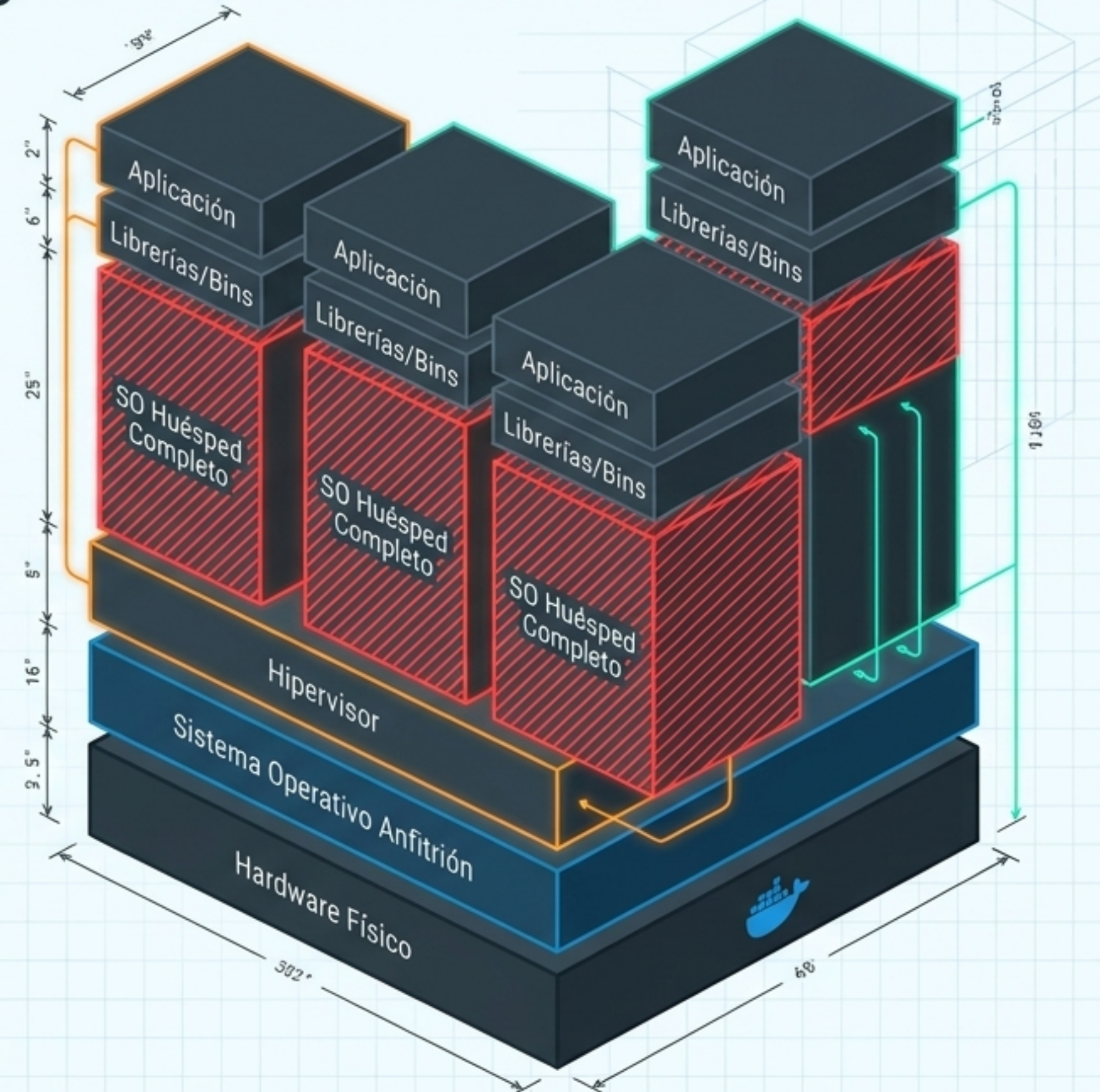
= Compartir Igual (SA):
Licencia idéntica para
derivados.

La Fricción del Despliegue Tradicional

Terminal Charcoal

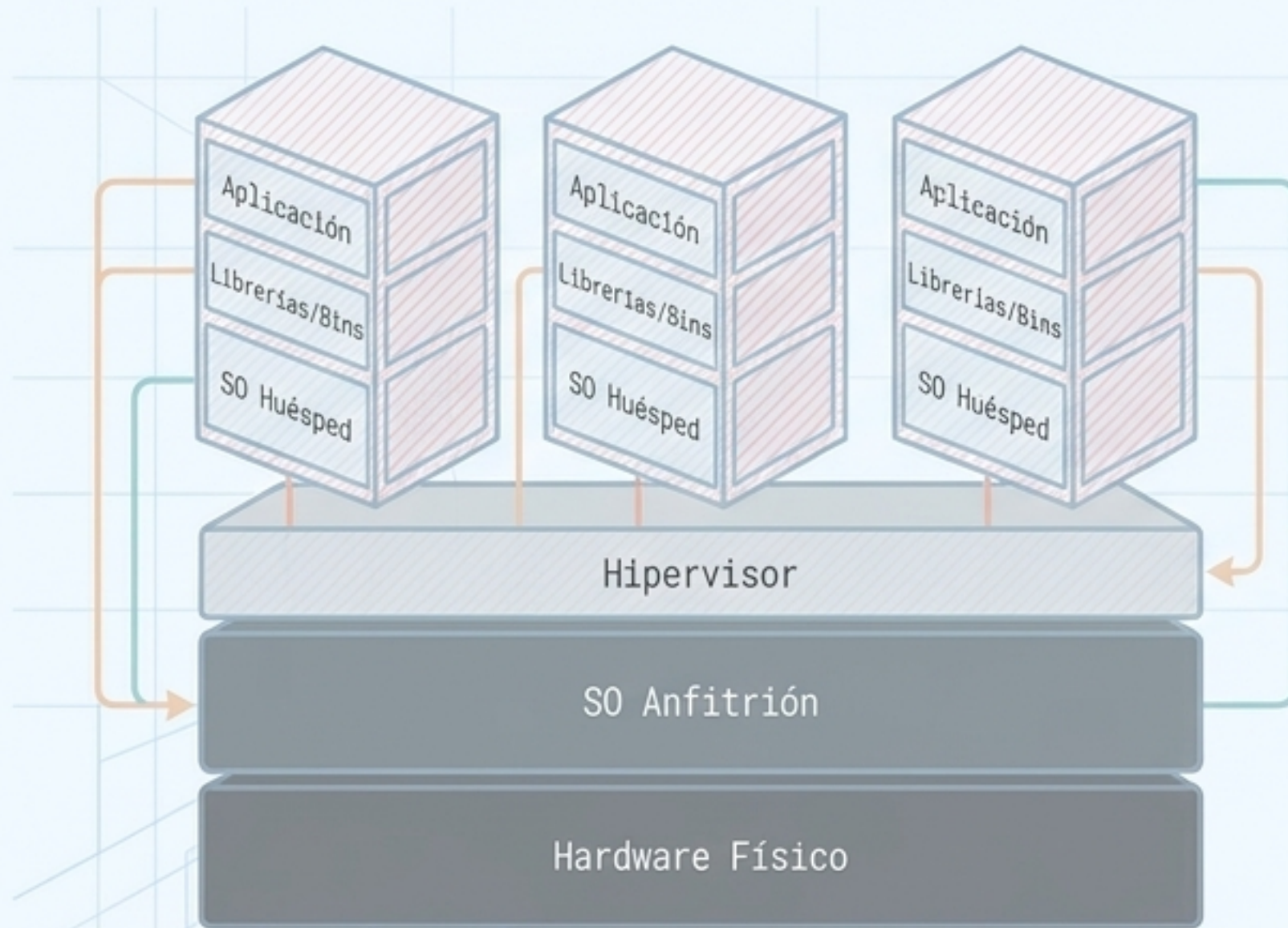
Arquitectura Monolítica (Máquinas Virtuales)

- **Duplicación:** Cada VM requiere una copia completa y pesada de un Sistema Operativo.
- **Lentitud:** El arranque está condicionado al inicio completo del SO huésped.
- **Desperdicio:** Recursos (RAM, CPU) bloqueados aunque la aplicación no los use.

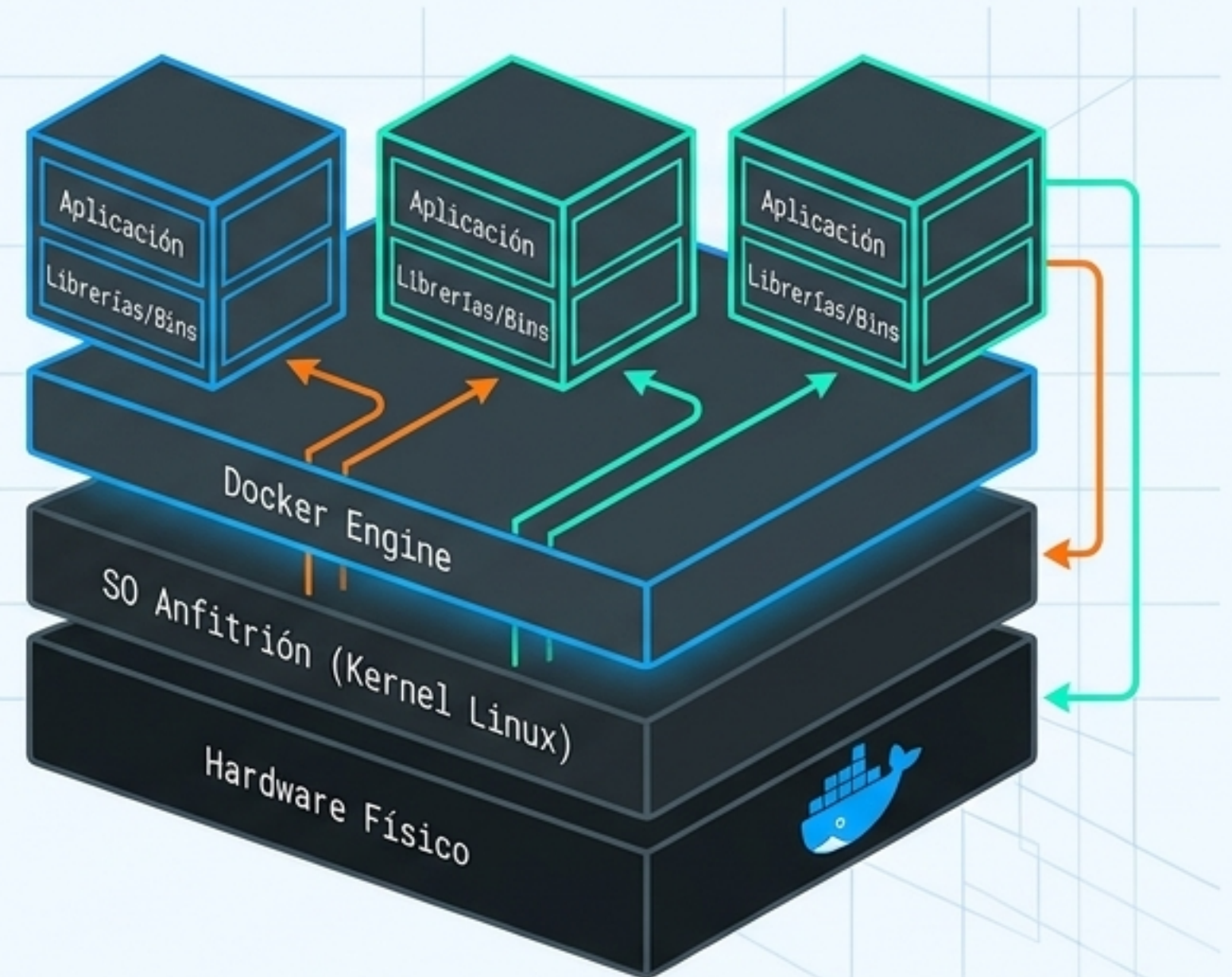


La Revolución del Contenedor Ligero

Arquitectura Monolítica (Máquinas Virtuales)



Arquitectura de Contenedores (Docker)



El Estándar Docker:

Encapsula la aplicación y sus dependencias en un entorno aislado, abstrayéndose del hardware.

Compartición de Kernel:

Utiliza directamente el núcleo del SO anfitrión (Linux). Resultado: Velocidad nativa y arranque en segundos.

Resiliencia:

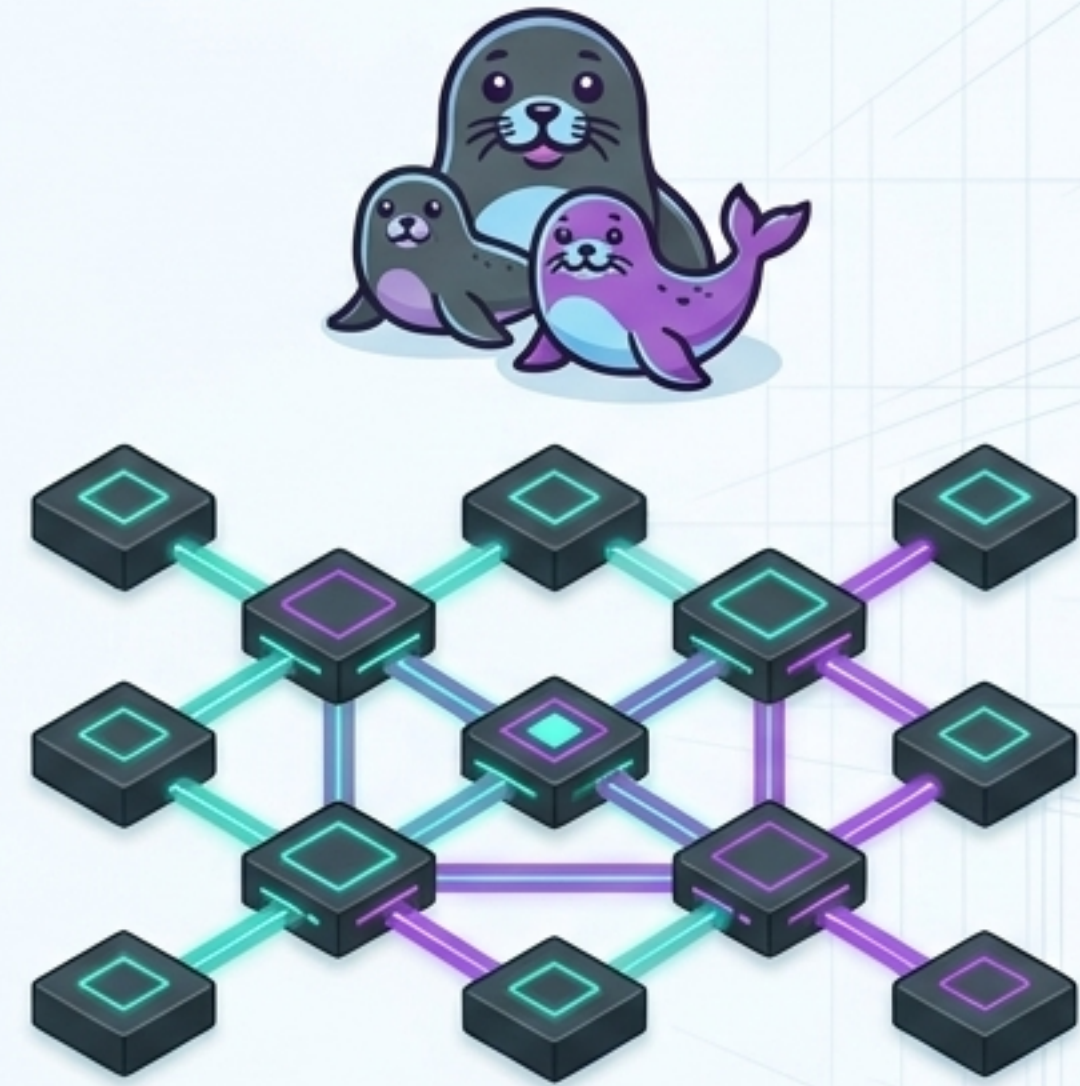
Si un contenedor cae por fallo, se reinicia instantáneamente sin afectar al resto.

Motores de Contenedores: Docker y Podman



Docker Inc.

El estándar de facto. Utiliza un demonio central en segundo plano para gestionar y orquestar todos los contenedores.



Podman (Red Hat)

100% compatible con comandos Docker, pero funciona sin demonio (daemonless). Mejora radicalmente la seguridad y la gestión de recursos al funcionar de forma independiente.

Nota: Ambos motores extraen imágenes estándar de repositorios públicos como Docker Hub o Red Hat Quay

El CMS como Arquitectura de Microservicios

El despliegue monolítico desaparece. Dividimos los componentes en servicios aislados que se comunican a través de una red privada virtual.



Base de Datos (MariaDB)
Aísla los datos. No expone puertos al exterior (3306 interno).

Aplicación (WordPress + PHP)
Contiene la lógica y el servidor web. Expone el puerto 8080 al host.

Explorador (FileBrowser)
Gestión web de ficheros. Expone el puerto 8082.

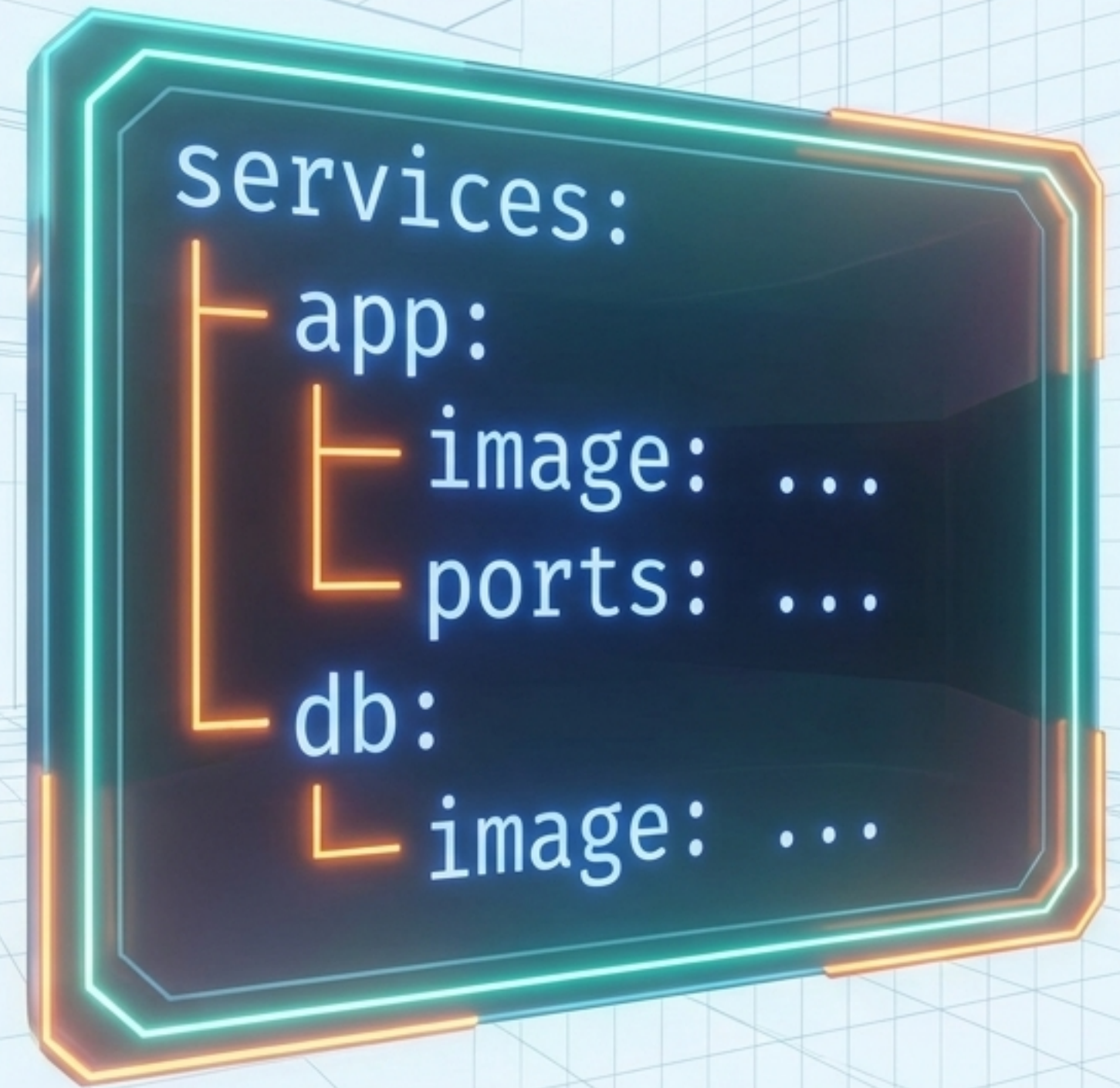
Orquestación: El Manifiesto YAML

Docker Compose: Herramienta para definir y ejecutar aplicaciones Docker de múltiples contenedores.

El formato YAML: Un archivo de texto (docker-compose.yml) que actúa como el "plano arquitectónico" de nuestra infraestructura logística.

Ventaja: Levantamos toda la infraestructura (App + BD + Redes + Volúmenes) con un solo comando:

```
podman-compose up -d
```



Anatomía de un Despliegue: La Base de Datos

```
services:
  db:
    image: mariadb:10.6.4-focal
    volumes:
      - ./db_data:/var/lib/mysql
    environment:
      - MYSQL_ROOT_PASSWORD=secreta
      - MYSQL_DATABASE=wordpress
      - MYSQL_USER=wordpress
```

1

La Imagen: El molde exacto del software descargado del repositorio.

2

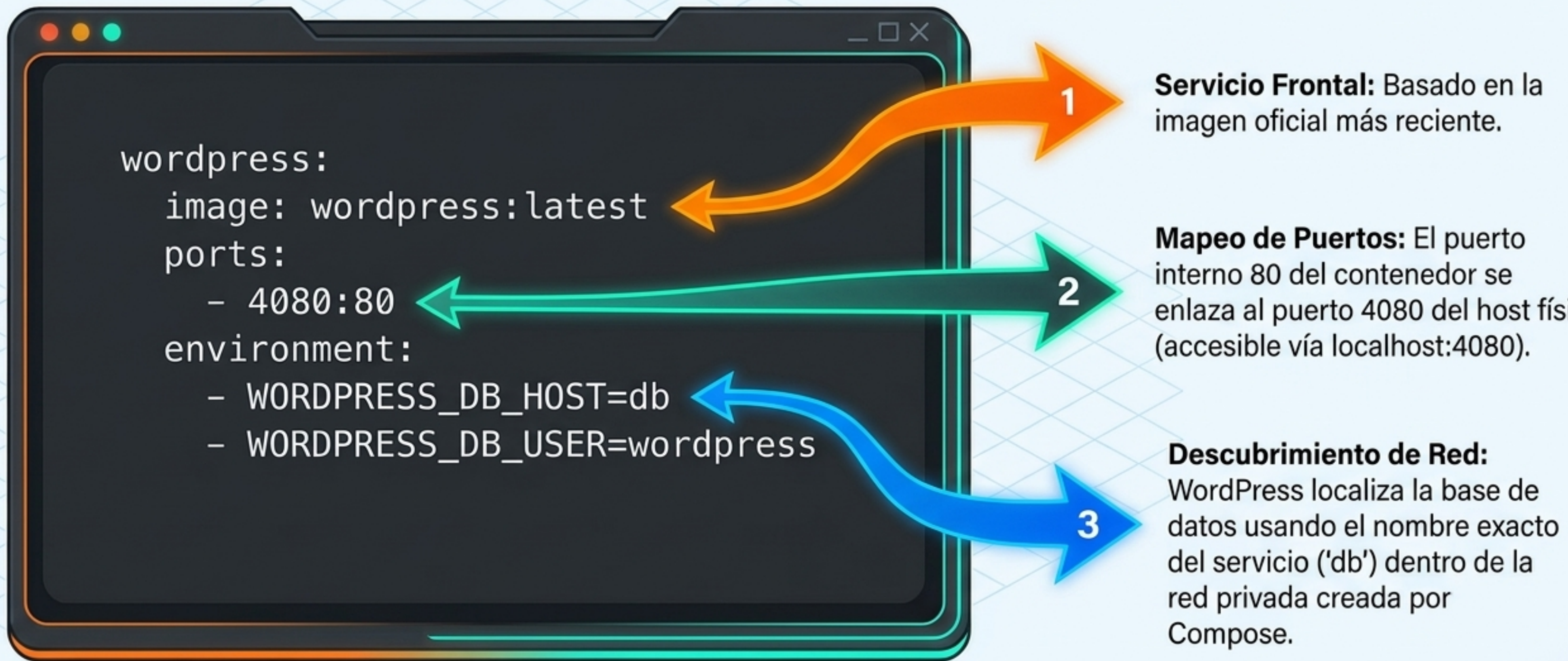
Persistencia: Mapeo de datos locales para que no se borren al destruir el contenedor.

3

Variables de Entorno: Inyección de contraseñas y configuración inicial sin tocar la consola interactiva.

Anatomía de un Despliegue: La Aplicación

```
wordpress:  
  image: wordpress:latest  
  ports:  
    - 4080:80  
  environment:  
    - WORDPRESS_DB_HOST=db  
    - WORDPRESS_DB_USER=wordpress
```



Servicio Frontal: Basado en la imagen oficial más reciente.

Mapeo de Puertos: El puerto interno 80 del contenedor se enlaza al puerto 4080 del host físico (accesible vía localhost:4080).

Descubrimiento de Red: WordPress localiza la base de datos usando el nombre exacto del servicio ('db') dentro de la red privada creada por Compose.

La Ley de la Persistencia: Volúmenes

El Problema:

Los contenedores son efímeros por diseño. Si se destruye un contenedor, todos los datos internos desaparecen en el vacío.

La Solución:

Volúmenes de Docker. Un espacio persistente en el disco duro del host físico que se "monta" como un túnel dentro del contenedor.

Resultado:

El contenedor puede actualizarse, destruirse o reiniciarse miles de veces; la base de datos permanecerá intacta y segura.



Herramientas de Gestión Visual

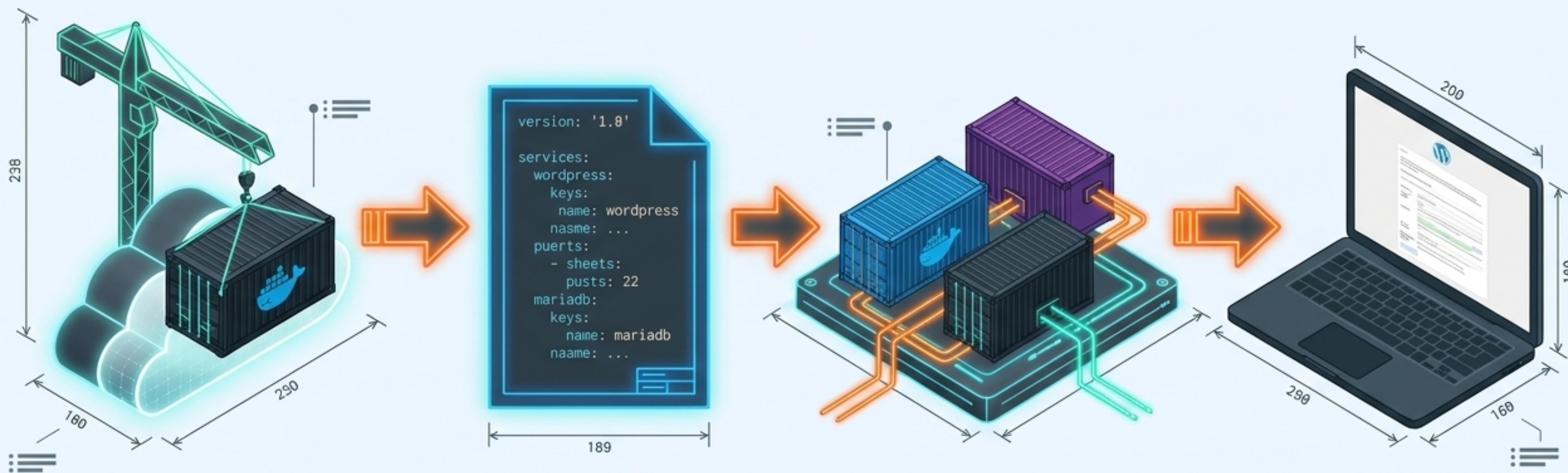
Administrar microservicios no requiere depender 100% de la terminal. Existen herramientas gráficas que se despliegan como contenedores.



Portainer: Interfaz gráfica (GUI) total para administrar Docker/Podman. Permite ver logs, reiniciar servicios y gestionar redes visualmente.

FileBrowser: Explorador web de archivos mapeado a la raíz del CMS. Elimina la necesidad de conectarse por SSH o FTP para editar configuraciones.

El Ciclo de Vida del Despliegue



1. PULL

Descarga de imágenes base (WordPress, MariaDB) desde repositorios (Docker Hub).

2. COMPOSE

El manifiesto YAML define las reglas, puertos, redes aisladas y variables.

3. UP

El motor despliega la pila completa de contenedores en milisegundos.

4. INSTALL

Acceso final vía localhost para completar la configuración visual del CMS.