CONCEPTOS FUNDAMENTALES DE DOCKER

HELDA YESENIA VALBUENA CASTRO

UNIVERSIDAD DE CUNDINAMARCA

FACULTAD DE INGENIERIA – NOCHE

LINEA DE PROFUNDIZACION III

GESTOR DE EL CONOCIMIENTO: Alexander Matallana

CHIA, CUNDINAMARCA

22 sep. 25

Conceptos fundamentales de Docker

¿Qué es Docker y en que de diferencia de una máquina virtual?

- **¿Qué es Docker?**

  Docker es una plataforma de código abierto que permite a los desarrolladores crear, implementar, ejecutar, actualizar y gestionar aplicaciones en contenedores.

  Los contenedores son componentes estandarizados y ejecutables que combinan el código fuente de la aplicación con las bibliotecas del sistema operativo (SO) y las dependencias necesarias para ejecutar ese código en cualquier entorno.

  En pocas palabras es un entorno en el cual se pueden ejecutar contenedores que puede ejecutarse tanto en la maquina como en la nube.

  Diferencia de una máquina virtual.

  Docker y las máquinas virtuales (VM) son dos tecnologías que se utilizan en el despliegue de aplicaciones. En el ciclo de vida del desarrollo de software, el despliegue prepara el código de la aplicación para que se ejecute para los usuarios finales. Docker es una plataforma de código abierto que los desarrolladores utilizan para empaquetar el software en unidades estandarizadas denominadas *contenedores*. El contenedor tiene tanto el código de la aplicación como su entorno, incluidas las bibliotecas, las herramientas del sistema y la versión ejecutable. Con Docker, puede implementar y escalar aplicaciones en cualquier máquina y garantizar que su código se ejecute de manera uniforme. Por el contrario, una máquina virtual es una copia digital de una máquina física. Es posible tener varias máquinas virtuales que tengan sistemas operativos propios e individuales que se ejecuten en el mismo sistema operativo host. Los desarrolladores configuran la máquina virtual para crear el

entorno de la aplicación. También es posible ejecutar contenedores de Docker en máquinas virtuales.

Se pueden evidenciar 4 parámetros marcados para las diferencias establecidas entre estos.

Arquitectura

- **Máquina virtual (VM):**
  - Requiere un **hipervisor** (como VirtualBox, VMware o Hyper-V).
  - Cada VM contiene un sistema operativo completo invitado (Guest OS), además de la aplicación y sus librerías.
  - Esto implica mayor consumo de recursos (RAM, CPU y disco).
- **Docker (contenedores):**
  - Los contenedores comparten el **kernel del sistema operativo anfitrión**.
  - Solo incluyen la aplicación y sus dependencias, no un SO completo.
  - Son más ligeros y rápidos de iniciar.

Rendimiento

- **VM:** Más pesadas, tardan más en arrancar porque cargan todo un sistema operativo.
- **Docker:** Arranca en segundos porque aprovecha el kernel del host y solo levanta lo necesario.

Portabilidad

- **VM:** Menos portátiles porque dependen del hipervisor y de la imagen de la máquina virtual.
- **Docker:** Muy portátil; puedes correr el mismo contenedor en Linux, Windows, Mac o la nube con solo instalar Docker.

Uso de recursos

- **VM:** Consume más memoria y almacenamiento (varios GB).
- **Docker:** Usa menos recursos, las imágenes suelen ser más pequeñas (MBs a pocos GB).

Según la explicación, “Docker permite virtualizar aplicaciones a través de contenedores ligeros que comparten el kernel del sistema operativo, mientras que una máquina virtual requiere virtualizar todo el hardware y un sistema operativo completo, lo que la hace más pesada en recursos” (ChatGPT, comunicación personal, 22 de septiembre de 2025).

¿Qué es una imagen en Docker y como se relaciona en un contenedor?

¿Qué es una Docker Imagen?

Las imágenes son plantillas de solo lectura que contienen instrucciones para crear un contenedor. Una imagen de Docker crea contenedores que se ejecutan en la plataforma Docker.

Como la imagen se compone de muchas capas estas pueden ser modificadas dentro de su entorno, estas poseen dependencias del SO para que pueda ser ejecutado por la aplicación.

Se relaciona con el contenedor ya que es una instancia de la ejecución de la imagen añadiendo capas de funcionalidad básicas que luego se crean para ejecutar en un contenedor. Con esto simplificar el desarrollo y las pruebas.

¿Qué es Docker hub y como se utiliza para descargar imágenes?

Iniciemos con que Docker hub es el repositorio oficial de Docker en la nube.

Docker Hub: Permite centralizar la gestión de contenedores y simplificar su distribución. Gracias a esta plataforma, los equipos DevOps pueden acceder fácilmente a las imágenes de contenedores, personalizarlas, actualizarlas y compartirlas para una mejor colaboración.

Para realizar la descarga de una imagen se realiza de la siguiente forma

Ingresar a la terminal

Usar comando docker pull _nombre de la imagen en la terminal.

Por **ejemplo,** docker pull nombre_imagen:tag

- docker pull ubuntu
- docker pull mysql:8.0

¿Que son los volúmenes en Docker?

Los volúmenes son un mecanismo para almacenar datos fuera de los contenedores. Todos los volúmenes son administrados por Docker y se almacenan en un directorio dedicado en su host.

Los volúmenes funcionan con contenedores de Linux y Windows. Hay varios controladores disponibles para almacenar datos de volúmenes en diferentes servicios. El almacenamiento local en el host Docker es la opción predeterminada, pero también existen volúmenes NFS, recursos compartidos CIFS/Samba y adaptadores de almacenamiento en bloque a nivel de dispositivo. Los complementos de terceros también pueden añadir opciones de almacenamiento adicionales.

¿Cuándo utilizar volúmenes Docker?

Los volúmenes están diseñados para facilitar la implementación de contenedores Docker con estado. Necesitará usar un volumen cuando un contenedor requiera almacenamiento persistente para guardar permanentemente archivos nuevos y modificados.

Almacenamiento de la base de datos

Datos de la aplicación

Cachés esenciales

Copias de seguridad de datos prácticas

Compartir datos entre contenedores

Escritura en sistemas de archivos remotos

¿Qué es una red en Docker?

La red Docker es el sistema que permite que los contenedores se comuniquen entre sí, con el host y con redes externas. Define cómo se mueven los datos entre contenedores y entre sistemas durante la ejecución de aplicaciones contenedorizadas.

Proporciona entornos de red aislados y flexibles mediante controladores integrados como bridge, host, overlay y none. Cada controlador admite diferentes casos de uso, como desarrollo local, orquestación basada en enjambre o integración con infraestructura heredada.

La configuración adecuada de la red es fundamental para el rendimiento, la seguridad y el descubrimiento de servicios.

Encontramos tipos de red:

Puente: La opción predeterminada para contenedores independientes

Host: Elimina el aislamiento de la red utilizando directamente la pila de red del host.

Superposición: Permite la creación de redes multihost mediante Docker Swarm.

macvlan: Asigna una dirección MAC a cada contenedor, haciéndolo aparecer como un dispositivo físico en la red.

Ipvlan: Utiliza un método diferente para gestionar el tráfico.

¿Qué es Docker compose y qué ventajas tiene frente a ejecutar un contenedor con Docker ramdom?

- **¿Qué es Docker Compose?**
  Docker Compose es una **herramienta** versátil que te permite **definir y gestionar** aplicaciones multi-contenedor de forma sencilla. Con Docker Compose, puedes describir la **configuración** de tu entorno de desarrollo en un archivo YAML, especificando los servicios, volúmenes y redes necesarios para tu aplicación.

    1. Ventajas
    2. Permite levantar **múltiples contenedores a la vez** con un solo comando.
    3. Centraliza toda la configuración en un archivo docker-compose.yml.
    4. Facilita la **repetición del entorno** en cualquier máquina sin errores manuales.
    5. Crea automáticamente una **red interna** para que los servicios se comuniquen por nombre.
    6. Soporta **variables de entorno** y configuraciones organizadas.

7. Permite **escalar servicios** (ej. varias instancias de un contenedor) fácilmente.
8. Mejora la **colaboración en equipo** al compartir solo el archivo de configuración.
9. Simplifica el **mantenimiento y actualización** de servicios.
10. Ahorra tiempo frente a ejecutar muchos docker run con parámetros largos.

Repaso BD: entidades, atributos, relaciones, cardinalidad y formas normales de una base de datos

¿Qué es una entidad en base de datos?

Con respecto a las bases de datos, una entidad es algo que existe y puede ser identificado.

Por ejemplo, una persona, un libro o un coche. Las entidades tienen atributos que las describen, como el nombre de la persona, el título del libro o el modelo del coche.

¿Qué son los atributos en Bd?

Los atributos son características o propiedades que describen a una entidad o relación en una base de datos. Los atributos son la información específica que se almacena en una base de datos sobre las entidades o relaciones y están directamente relacionados con ellas para describir y diferenciar cada una de manera precisa y completa

Por ejemplo, para una entidad “Persona”, los atributos pueden ser “nombre”, “apellido”, “edad”, “dirección”, entre otros. Para una relación

“Compra”, los atributos pueden ser “fecha de compra”, “cantidad comprada”, “precio unitario”, entre otros. Los atributos permiten describir y distinguir a las entidades o relaciones en la base de datos

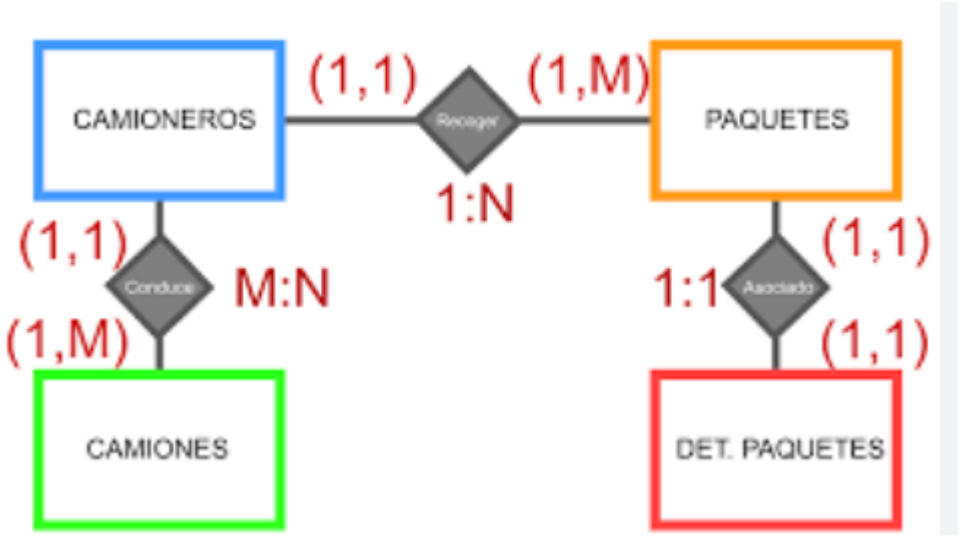
¿Qué es una relación?

Una relación es una conexión entre dos o más entidades en una base de datos.

Por ejemplo, una persona puede tener una relación de “compra” con un libro si esa persona ha comprado el libro. Las relaciones se utilizan para representar las interacciones o conexiones entre las entidades. Las relaciones también tienen atributos, como la fecha en que se realizó la compra.

Cardinalidad

Es una característica que establece la cantidad de instancias de una entidad que pueden estar relacionadas con otra entidad. Se representa en el diagrama mediante una etiqueta, que puede ser: 1:1 (uno a uno), 1:N (uno a muchos) y N:N (muchos a muchos).



Formas normales

1NF, 2NF, y 3NF son los tres primeros tipos de normalización de base de datos. Significan primera forma normal, segunda forma normal y tercera forma normal, respectivamente

La primera forma normal – 1NF

Para una tabla ser la primera forma normal, debe cumplir el siguiente criterio:

- una sola celda no debe contener más de un valor (atomicidad)
- debe haber una clave primaria para identificación
- no filas o columnas duplicadas
- cada columna debe tener solamente un valor por cada fila en la tabla

La segunda forma normal – 2NF

El 1NF solamente elimina los grupos repetitivos, no la redundancia. Por eso hay 2NF.

Una tabla se dice que está en 2NF si cumple el siguiente criterio:

- ya está en 1NF
- no tiene dependencia parcial. Es decir, todos los atributos no claves son totalmente dependientes de la clave primaria

La tercera forma normal – 3NF

Cuando una tabla está en 2NF, elimina los grupos repetitivos y la redundancia, pero no elimina la dependencia parcial transitiva.

Esto significa que un atributo no principal (un atributo que no forma parte de la clave del candidato) es dependiente de otro atributo no principal. Esto es lo que la tercera forma normal (3NF) elimina.

Así que, para que una tabla esté en 3NF, debe:

- estar en 2NF

- no tiene dependencia parcial transitiva

Bibliografía

https://www.ibm.com/es-es/think/topics/docker

https://chatgpt.com/c/68d1df68-e1bc-832f-a498-ab1615fb897e

https://aws.amazon.com/es/compare/the-difference-between-docker-vm/

https://datascientest.com/es/docker-hub-todo-sobre

https://spacelift.io/blog/docker-volumes

https://imaginaformacion.com/tutoriales/que-es-docker-compose

https://circleci.com/blog/docker-image-vs-container/

https://www.freecodecamp.org/espanol/news/normalizacion-de-base-de-datos-formas-normales-1nf-2nf-3nf-ejemplos-de-tablas/#1nf

https://www.google.com/search?q=que+es+docker&oq=Que+es+doker&gs_lcrp=EgZjaHJvbWUqCQgBEAAYChiABDIGCAAQRRg5MgkIARAAGAoYgAQyCQgCEAAYChiABDIJCAMQABgKGIAEMgkIBBAAGAoYgAQyCQgFEAAYChiABDIJCAYQABgKGIAEMgkIBxAAGAoYgAQyCQgIEAAYChiABDIJCAkQABgKGIAE0gEIOTIxMGowajeoAgCwAgA&sourceid=chrome&ie=UTF-8