

Nombre: Daniel Felipe Batón Hernández

Fecha:

Profesor:

Materia:

Institución:

Curso:

Nota:

Que es docker

- Es una plataforma de código abierto para crear y ejecutar aplicaciones en contenedores, que son estandarizados y ejecutables que incluyen el código de la aplicación, bibliotecas, dependencias y configuración. Permite a los desarrolladores empaquetar sus aplicaciones junto con todo lo necesario para ejecutarlas, asegurando que funcionen de manera consistente en cualquier entorno. Esta tecnología simplifica el proceso de desarrollo, prueba e implementación de software, y ofrece ventajas como la modularidad, el control de versiones y la eficiencia en el uso de recursos de computación con las máquinas virtuales tradicionales.

¿Como Funciona Docker?

- **Imagenes:** Es una plantilla de solo lectura que contiene las instrucciones y todos los componentes necesarios (código, librerías, dependencias y archivos de configuración), para crear un contenedor docker. Es un paquete ejecutable e independiente que permite ejecutar software en un entorno consistente y reproducible, y apartir de una única imagen se pueden crear múltiples instancias de contenedores idénticos.
- **Contenedor:** Es un paquete ejecutable, portable y autónomo que incluye todo lo necesario para ejecutar una aplicación; aislando la aplicación de su entorno. Permite que el software ejecutable, se ejecute de forma constante en cualquier lugar, ya sea el portatil de un desarrollador, un servidor local o una máquina en la nube.
- **Docker Engine:** Es la aplicación principal en el ecosistema docker, instalada en el sistema host, que actúa como el motor del cliente-servidor para crear, ejecutar y gestionar contenedores docker. Incluye un proceso demonio "Dockerd", que gestiona los contenedores, un servicio de clientes de líneas de comandos "CLI", para interactuar y una "API REST", para la comunicación entre el cliente y el demonio. Es la tecnología de código abierto que permite empaquetar aplicaciones y sus dependencias en contenedores estándar.
- **Docker Registry:** Es un repositorio para almacenar y distribuir imágenes del docker, que son el "código" para los contenedores. Docker Hub, es un servicio de registro más grande y popular, actuando como el registro predeterminado del docker, que alberga repositorios públicos de código abierto y permite la creación de repositorios privados para gestionar y compartir imágenes de forma segura.

- Docker Compose: Es una herramienta de Docker, para definir y ejecutar aplicaciones multi-contenedor utilizando un archivo YAML. Permite describir todos los servicios de la aplicación, sus dependencias y su configuración (redes y volúmenes), en un único archivo de texto. Con un solo comando, puedes iniciar, parar y gestionar todos los servicios de tu aplicación, lo facilita la orquestación de entornos de desarrollo, pruebas y despliegues contantes.

Archivo YAML = En el contexto Docker, generalmente llamado docker-compose.yml, es un archivo de configuración que utiliza el lenguaje de serialización de datos YAML, para definir y gestionar aplicaciones en contenedores que utilizan varios servicios y dependencias. En lugar de crear y ejecutar cada contenedor individualmente, este archivo permite declarar los servicios, sus redes, volúmenes, y otras configuraciones y luego usar el comando docker-compose up, para iniciar y gestionar todos los componentes de la aplicación con una sola instrucción.

- Docker Swarm: Es una herramienta nativa de Docker, para la orquestación de contenedores, que permite ejecutar agrupar múltiples nodos de Docker en un clúster que se comporta como un único sistema virtual. Se usa para escalar y gestionar aplicaciones en entornos distribuidos, ofreciendo balanceo de carga, alta disponibilidad y la capacidad de ejecutar servicios en un conjunto cooperativo de máquinas o nodos.

¿Ventajas de Docker?

Portabilidad: Los contenedores pueden ejecutarse de forma idéntica en cualquier lugar, ya sea en portatil de un desarrollador, en un servidor de pruebas o en un entorno de producción en la nube, garantizando entornos coherentes.

Eficiencia de Recursos: A diferencia de las máquinas virtuales tradicionales, los contenedores comparten el kernel del sistema operativo (redes, host, lo que hace mucho más ligeros, más rápidos de iniciar y consumen menos espacio en RAM, y espacio en disco).

Aislamiento: Cada contenedor se ejecuta en su propio entorno aislado, lo que evita que las dependencias y configuraciones de una aplicación interfieran con otras aplicaciones que se ejecuten en la misma máquina.

Desarrollo y despliegue más rápido: Docker agiliza el ciclo de vida de las aplicaciones. Permite desarrollar, probar y desplegar aplicaciones de manera más rápida, fiable y eficiente, simplificando el proceso desde la creación del código hasta

Escalabilidad: Facilita la creación de múltiples instancias de un contenedor para distribuir los cargos de trabajo, lo que permite escalar las aplicaciones de manera eficiente y adaptarlas a la demanda.

Entornos coherentes: Elimina el clásico problema "en mi máquina si funciona" ya que el contenedor garantiza que el entorno de la aplicación es siempre el mismo, independientemente de donde se use.

Simplificación de la Infraestructura: Al empaquetar aplicaciones y dependencias juntas, Docker simplifica la gestión de la infraestructura y reduce la complejidad en la configuración de servidores.

Ecosistema amplio: Existe un gran sistema de herramientas y aplicaciones listas para usar, así como servicios como Docker Hub, para gestionar y compartir imágenes, que avanza más el uso del Docker.

Dockerfile

Dockerfile:

- Es un documento de texto que incluye instrucciones para construir imágenes Docker. Al leer este archivo, Docker construirá automáticamente nuevas imágenes.
- El comando "docker build" creará una imagen apartir de un Dockerfile y un contexto. Este contexto es un conjunto de archivos que se encuentran en la ruta URL especificada.
- En primer lugar, introduce el siguiente comando:

```
docker build  
docker build -f /Path/to/a/Dockerfile  
docker build -t mytag/myapp
```

- Esto enviará todo el contexto al demonio Docker. Para apuntar a un archivo Docker en tu sistema de archivos. Si la construcción tiene éxito, puedes especificar un repositorio y una etiqueta de donde guardan la nueva imagen. Despues el demonio Docker realizará una validación de Dockerfile. Si hay algún problema de sintaxis, devolverá error.

Docker Desktop

- Para empezar a utilizar Docker en tu entorno Mac, Linux, Windows. Puedes instalar Docker Desktop. Ofrece una interfaz sencilla y fácil de usar para gestionar aplicaciones, contenedores e imágenes en tu ordenador personal.
- Con Docker Desktop, no tendrás que utilizar la línea de comandos para realizar tareas esenciales. Esto puede acelerar el flujo de trabajo de desarrollo. También con un mercado de extensiones incorporado que facilita el comienzo del uso de herramientas de desarrolladores de terceros. Estas incluyen software para depurar, probar y asegurar tus aplicaciones.

Ademas Docker Desktop es (personal) gratuito y de uso personal y de pequenas empresas. Sin embargo, ten en cuenta que las personas mas grandes tendran que adquirir una suscripcion apartir de 5 dolares el mes.

Docker Hub

- Docker Hub es una plataforma donde puedes encontrar y compartir imagenes de contenedores. Es el mayor repositorio de imagenes de contenedores del mundo. Con recursos de desarrolladores de la comunidad, proyectos de codigo abierto y proveedores de softwares independientes (ISV).
- Estas son algunas de las caracteristicas incluidas en el Docker Hub:
 - Repositorios para enviar y enviar imagenes de contenedores.
 - Crea equipos y organizaciones con acceso a repositorios privados
 - Imagenes oficiales de Docker.
 - Activa acciones con webhooks
 - Crea imagenes de contenedores desde Github o Bitbucket y envialas a Docker hub.
- Para empezar a utilizar Docker Hub. Tendras que crear tu primer repositorio. Simplemente tendras que darle un nombre y especificar su visibilidad.

Resumen:

- Si tienes que gestionar varias aplicaciones, Docker podra ser una herramienta eficaz para organizarlas de forma independiente. Como utiliza contenedores, Docker puede ayudarte a desarrollar cada aplicacion sin el riesgo de que haya lenguajes de programacion, bibliotecas o frameworks en conflicto.
- Aunque Docker suele ser mas eficiente que una Maquina Virtual (VM), trabajadora en un servidor fisico tendra ser mas rapido. Como tambien depende de una interfaz de linea de comandos, Docker puede ser una gran opcion para los principiantes absolutos.
- Como desarrollador de WordPress es probable que necesites crear rápidamente entornos de pruebas locales. Con DevKinsta, puedes crear sitios de WordPress, basados en Docker. Asi podras empezar a desarrollar temas y plugin sin conflictos.

Nombre: Daniel Felipe Bata Hernandez

Fecha:

Profesor:

Materia:

Institución:

Curso:

Nota:

- Comandos Principales:

- Verificar instalación → docker -version
- Descargar una imagen → docker pull nombre_imagen
- Listar imágenes disponibles en el sistema → docker images
- Ejecutar un contenedor docker run nombre_imagen
- Listar contenedores en ejecución → docker ps
- Listar contenedores (incluidos detenidos) → docker ps -a
- Detener contenedor → docker rm id_contenedor
- Congela los procesos activos de un contenedor → docker pause
- Reanuda los procesos activos de un contenedor → docker unpause
- Eliminar imágenes, Puede ser una sola o varias → docker rmi [ImageId]
o la vez, por medio del id de la imagen → docker rmi [ImageId]
- Renombrar un contenedor. Es útil cuando tienes varios contenedores y quieres diferenciarlos en función de su finalidad → docker rename [OldName] [NewName]
- Crear nuevas imágenes después de realizar cambios en los archivos de un contenedor → docker commit [ContainerId] [Image-Of-Image]
- Proporciona el historial de una imagen especificada, ayudandote a comprender como se creó y mostrando el tamaño de la imagen → docker history [Image]
- Recupera los registros de un contenedor, proporciona información sobre las operaciones de un contenedor → docker logs
- Eliminar una imagen → docker rmi id_imagen
- Construir una imagen a partir de Dockerfile → docker build más el nombre de imagen

Redes:

- Permiten la comunicación entre contenedores, y con el exterior, utilizando diferentes tipos de redes, (de) puente, superpuerta, o macvlan para aislar o conectar aplicaciones en distintas máquinas. Sirven para que las aplicaciones y microservicios puedan interactuar de forma segura y eficiente, facilitando la portabilidad y escalabilidad de las aplicaciones en cualquier entorno.

Redes puente: (Bridge Networks).

- Permiten que los contenedores dentro de un mismo host Docker, se comuniquen entre sí en una red privada. Para que sea accesible desde el exterior, se debe configurar la asignación de puertos del host o los puertos del contenedor.
- Crea un puente virtual en el host y conectan a todos los contenedores a esa red, permitiendo la comunicación entre ellos mediante sus direcciones IP.

Red de host (host)

- En este modo, un contenedor elimina el aislamiento de red entre el contenedor y el host del docker, compartiendo directamente la red del host.

Red overlay:

- Permite a los contenedores comunicarse en diferentes hosts, ideal para organizar aplicaciones distribuidas en un cluster.
- Permiten que los contenedores en diferentes hosts se comuniquen como si estuvieran en la misma red física, utilizando tecnologías como VXLAN. Son ideales para aplicaciones distribuidas en la nube.

Redes Macvlan:

- Asignan a cada contenedor una dirección MAC única, haciendo parecer un dispositivo único físico real en la red. Esto es útil para que las aplicaciones heredadas o que requieran acceso directo a la red, fizica.

¿Qué hacen las redes en Docker?

- Permiten que contenedores que necesiten interactuar, como una aplicación web y su base de datos, puedan comunicarse entre sí.
- Aislamiento de aplicaciones: se pueden crear redes separadas para diferentes aplicaciones, de manera que los contenedores de una red, no puedan acceder a los de otra, mejorando la seguridad.
- Gestión de tráfico: permiten controlar y filtrar tráfico que fluye entre los contenedores y el exterior, utilizando el sistema de cortafuegos del host.

Ejemplo de uso:

- Cuando se despliega una aplicación web con docker, es común crear una red docker para que la aplicación web y la base de datos puedan comunicarse de forma segura y sin exponerla directamente a internet. El comando docker network se utiliza para crear y gestionar estas redes en docker.

Buenas Prácticas y Seguridad.

- El uso de un contenedor lleva responsabilidades de seguridad.
 - Utilice imágenes oficiales o verificadas: Las imágenes oficiales se actualizan.
 - Fija versiones de las imágenes para evitar que un cambio inesperado rompa la aplicación.
 - Actualice docker y el sistema operativo para corregir vulnerabilidades del kernel y de la plataforma.
 - Actualice redes personalizadas y segmentación, declare redes internas con compose o Docker CLI para aislar servicios y minimice la exposición de puertos.
 - Gestione secretos: Evite almacenar contraseñas en el dockerfile utilizando variables de entorno gestores de secretos "swarm", Kubernetes secrets o montajes de archivos.
 - Minimize tamaño y capas: Elimine paquetes innecesarios y emplee multi-stage builds para reducir la superficie de ataque.