Resumen Docker

Que es?

Software de virtualizacion que facilita el desarollo y el despligue de aplicaciones

Como funciona?

Empaqueta la apliction junto con todo los que necesita para funcionar dentro de un “Contenedor”, estos contenedores son portables; faciles de distribuir y desplegar

Que facilita?

El trabajo: Un contenedor es independiente del SO y de los sevicios de la maquina donde se esta ejecutando, con esto se logra estadarizar el proceso de preparar los servicions necesarios de forma local, permitiendo centrarse en el desarollo del programa

Despligue: No se requieren dependencias o configurar especificas en el servidor por cada aplicacion

Docker VS. MV

Docker solo virtualiza unicamente la capa de aplicacion, que estan los servicios y las aplicaciones indicadas, todo ya instalado en el contenedor

MV virtualiza todo el SO, es decir virtualiza la capa de aplicacion y Kernel

Los dos utiliza una capa de “hypervisor” para poder comunicar con el SO hosts

Diferencia entre los dos:

- El tamano de los contenedores de docker es mucho menor, ya que el tamano un contenedor de docker solo en de los MB y los de MV llegan a ser GB
- Los contenedores del docker aranca mucho mas rapido
- Tecnicamente docker es solo es compatible con linux y los MV son compatibles con todo

El problema de la compatibilidad se produce por que los vcontenedores de docker se basan en linux pero con el capa de “Hypervisor” que permite ejecutar en windows o MAC (Dockje Desktop)

Imagenes VS Contenedores

Imagenes: es el paquete de la app al completo

Contenedores: es una instanacia que ejecuta una imagen de docker, basicamente es una instancia de una imagen (se puede ejecutar multiples instancias de la misma imagen)

Docker Registries

Es un almacen en cloud para distribucion de imagenes de docker (Docker Hub)

Registry VS Repository

Dentro de un registro de docker, cada aplicacion tendra su propio repositorio, y dentro de dicho repositorio se almacenaran diferentes versiones de imagenes de esa aplicacion

Port binding

Es el hecho para que el contenedor sea visble y accessible des del exterior

Commandos para Docker

- **pull:** Descargar una imagen de docker hub y almacena en local

  docker pull [Nombre_imagen]:[Etiqueta_version]

- **run:** Se crea y se arranca un contenedor

  docker run [Nombre_imagen]:[Etiqueta]

- **ps:** Sirve para ver el estado de los contenedores en marcha

  docker ps

- **run –d:** Crea y arranca un contenedor con la imagen en modo “detach”

  docker run -d [Nombre_imagen]:[Etiqueta]

- **logs:** Para ver los logs de un contenedor en modo detach

  docker logs [ID_Contenedor]

- stop: Para detener un contenedor en marcha

  docker stop [ID_Container]

- ps –a: sirve para ver el estado de los contenedores en marcha y detenida

  docker ps -a

- start: reutiliza un contenedor ya creado

  docker start [ID_Contenedor]

- run —name: hace posible derle un nombre a un contenedor

  docker run --name [Nombre_contenedor] [Nombre_Imagen]:[Etiqueta_ver]

- run -p: arrancar un contenedor que puede hacer el port binding

  docker run -p[Puerto_Host]:[Puerto_contenedor][Nombre_Imagen]:[Etiqueta_ver]

Despliegue de app en Docker

Desarrollar y empaquetar

- Primero, se crea la aplicación con todo lo necesario (código, librerías, dependencias, configuraciones).
- Luego, todo esto se empaqueta dentro de una imagen de Docker (un archivo que contiene la aplicación y su entorno).

Desplegar en un servidor

- Se debe definir cómo se ha de construir la imagen: para ello se genera un fichero “Dockerfile” que contiene las instrucciones de montaje de la imagen.
- Una vez creada la imagen, se sube al servidor y se ejecuta dentro de un contenedor.

Múltiples contenedores para distintos servicios

- Muchas aplicaciones necesitan otros servicios, como una base de datos (ejemplo: MySQL).
- En Docker, cada servicio se puede ejecutar en su propio contenedor.
- Así, una aplicación puede tener:
  - Un contenedor con el backend.
  - Otro contenedor con la base de datos MySQL.
  - Otro contenedor con un servidor web (como Nginx o Apache).

DockerFile

- From: Define la imagen base sobre la cual se construirá la nueva imagen. Puede ser una imagen oficial de Docker Hub o una personalizada.

  FROM php:8.1-apache

- Label: Permite agregar metadatos como el autor o información del mantenimiento.

  LABEL maintainer="Juan Luis <juan@example.com>"

- Copy: Copia archivos desde la máquina host al contenedor.

  COPY . /var/www/html

- Run: Ejecuta comandos en el proceso de construcción de la imagen (instalar paquetes, configurar permisos, etc.).

  RUN apt-get update && apt-get install -y libpng-dev

- Workdir: Cambia el directorio donde se ejecutarán los siguientes comandos.

  WORKDIR /var/www/html

- Expose: Indica en qué puerto se ejecutará la aplicación dentro del contenedor.

  EXPOSE 80

- Env: Define variables de entorno dentro del contenedor.

  ENV APP_ENV=production

- Cmd: Define el proceso principal que se ejecutará cuando el contenedor inicie

  CMD ["apache2-foreground"]

Construir el imagen

Una vez que el dockerfile ha sido creado con el commando build se crea la imagen

docker build [Ruta_de_Dockerfile]

El flag -t o --tag sirve para darle un nombre a la imagen y (opcionalmente) una etiqueta de versión, empleando el formato “nombre:tag”

docker build -t [Ruta_de_Dockerfile] : [Version_de_tu_app]

RUN de mi-app

Ahora que se ha generado la imagen, es posible crear el contenedor y ejecutar el programa mediante el comando RUN:

docker run -d -p 3000:3000 mi-app:1.0