Anibal Alvarado Andrade
Jhon Corredor

Docker

En el mundo actual, la eficiencia, la escalabilidad y la portabilidad de las aplicaciones son elementos fundamentales para el exito de cualquier proyecto de software. A lo largo de los años se han creado distintas soluciones para facilitar la ejecución de aplicaciones en diferentes entornos sin que surjan conflictos de dependencias, configuraciones o sistemas operativos. Una de las tecnologías más revolucionarias en este campo es docker.

Docker es una plataforma que permite, crear, desplegar y ejecutar aplicaciones dentro de contenedores.

Estos contenedores son entornos aislados que incluyen todo lo necesario para que una aplicación funcione correctamente: el codigo, las bibliotecas, las dependencias y la configuración. De esta forma, se asegura que la aplicación se ejecute de la misma manera, independientemente de dónde se instale.

2. Que es Docker?

Docker es un software de código abierto creado en 2013 por la empresa Docker Inc. que se utiliza para automatizar el despliegue de aplicaciones en contenedores de software.

En palabras simples Docker es una caja que contiene una aplicación junto con todo lo necesario para que funcione.

Gracias a esto un programa que corre en una computadora puede funcionar exactamente igual, sin necesidad de ajustes adicionales en otra.

Principales casactarísticas de Docker

1. Portabilidad: Una aplicación empacada en un contenedor puede ejecutarse en Linux, Windows o en la nube sin cambios

2. Eficiencia: Al compartir el mismo kernel del sistema operativo, los contenedores consumen menos recursos que las maquinas virtuales.

3. Rapidez: Iniciar un contenedor toma segundos, mientras que una maquina virtual puede tardar minutos.

4. Escalabilidad: permite desplegar múltiples instancias de una aplicación con facilidad.

3. Conceptos Fundamentales

3.1 Imagen

Una imagen es una plantilla de solo lectura, que contiene el sistema de archivos, bibliotecas, dependencias y el programa que se quiere ejecutar. Se puede ver como la "receta" de un contenedor.

Ejemplo: Una imagen de Nginx contendra el servidor web Nginx y lo necesario para ejecutarlo.

3.2 Contenedor

Un contenedor es una instancia de ejecución de una imagen, cuando se ejecuta una imagen, se crea un contenedor. Este contenedor puede modificarse, detenerse, reiniciarse o eliminarse.

Ejemplo: Si descargas la imagen de MySQL y la ejecutas, el contenedor será una base de datos MySQL en funcionamiento.

Comparacion entre Imagen y Contenedor

- Imagen -> es la receta
- Contenedor -> es el platillo preparado siguiendo la receta.

3.3 Dockerfile

El dockefile es un archivo de texto que contiene instrucciones para construir una imagen. Sirve como una lista de pasos que docker seguirá para armar el entorno.

Ejemplo:

```
From ubuntu:20.04
RUN apt-get update && apt-get install -y python3
COPY app.py /app/app.py
CMD ["python3", "/app/app.py"]
```

3.4 Docker Hub

Es un repositorio en linea similar a github, pero especializado en imágenes de docker. Allí se pueden encontrar imágenes oficiales de sistemas operativos, bases de datos, servidores web y otros programas listos para usar.

3.5 Volumenes

Los volúmenes son mecanismos de Docker para almacenar datos de manera persistente. Cuando un contenedor se elimina, normalmente se pierden los datos, pero los volúmenes se pueden guardar en el sistema anfitrión.

3.6 Redes en Docker

Los contenedores pueden comunicarse entre sí gracias a las redes que gestiona Docker. Esto permite, por ejemplo, que un contenedor de una aplicación se conecte con un contenedor de base de datos.

Docker maneja distintos tipos de redes, siendo las principales:

- Bridge: Es la red por defecto. Permite que contenedores en el mismo host se comuniquen entre sí usando su nombre como si fuera un dominio (Docker actúa como un DNS interno).
- Host: El contenedor comparte directamente la red del anfitrión, con mayor rendimiento pero menos aislamiento.
- None: Sin conexión de red, útil para procesos que deben ejecutarse completamente aislados.

Una de las ventajas es que los contenedores pueden descubrirse por nombre, lo que evita usar direcciones IP estáticas. Por ejemplo si se crea una red y se conectan varios contenedores:

```
docker network create mi_red
docker run -d --name db --network mi_red mysql:5.7
docker run -d --name app --network mi_red my-app-image
```

dentro del contenedor app, se puede acceder al contenedor db simplemente usando el nombre db como host de la cadena de conexión.

Además, se pueden exponer puertos al exterior para que servicios en un contenedor estén disponibles desde el host:

```
docker run -p 8080:80 nginx
```

Nombre: Anibal Alvarado Andrade
Profesor: Jhon Corredor
Institución: SENA
Fecha:
Materia:
Curso:
Nota:

4 Comparación entre docker y las VM

Antes de docker, la forma tradicional de ejecutar aplicaciones en entornos aislados era utilizando Maquinas Virtuales (VM). Una maquina virtual simula un sistema operativo completo sobre otro, gracias a un software llamado hipervisor (como VMware o Hyper-X o VirtualBox).

4.1 Diferencias claves

VM	Docker
Incluye un SO completo	Comparte el kernel del SO anfitrión
Pesado (Gb)	Liviano (MBs)
Tiempo de arranque en Minutos	arranque en segundos
Consume demasiados recursos del sistema	Comparte recursos eficientemente
No es tan portable ni flexible	Es muy portable y flexible

5. Conceptos avanzados

A medida que los proyectos crecen, el uso básico de Docker ya no es suficiente. Ejecutar un par de contenedores manualmente está bien para entornos pequeños, pero en la práctica se necesitan decenas o cientos de contenedores funcionando en conjunto, comunicándose y manteniéndose estables. Para estos escenarios avanzados, Docker ofrece o se integra con herramientas más potentes.

5.1 Orquestación

Cuando hay muchos contenedores, es necesario organizarlos, coordinarlos y asegurarse de que sigan funcionando correctamente. Este proceso se llama orquestación.

Las herramientas de orquestación se encargan de:

- Iniciar y detener contenedores automáticamente.
- Distribuir la carga entre varios contenedores (balanceo de carga).
- Reiniciar servicios si un contenedor falla.
- Escalar aplicaciones es decir, aumentar o reducir la cantidad de contenedores según la demanda.

Docker Compose

Es una herramienta que permite definir y ejecutar múltiples contenedores con un solo archivo de configuración llamado docker-Compose.yml.

Ejemplo:

```
Version: '3'
services:
  db:
    Image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: ejemplo
  app:
    build: .
    ports:
      - "8080:80"
    depends_on:
      - db
```

Con solo ejecutar docker-compose up, se levanta tanto la base de datos (db) como la aplicación (app) conectados en una red interna automática.

Kubernetes

Es un orquestador mucho más avanzado, utilizado en producción por empresas gigantes como Google, Netflix o Spotify.

Kubernetes permite:

- Ejecutar contenedores en varios servidores a la vez
- Escalar horizontalmente (más contenedores) o verticalmente (más recursos).
- Gestionar actualizaciones sin interrumpir el servicio.
- Definir reglas de seguridad y de red entre microservicios.

5.2 Multi-stage Builds

Uno de los problemas comunes al construir imágenes es que pueden volverse muy pesadas, ya que incluyen dependencias, compiladores o librerías que solo se necesitan durante el proceso de la construcción, pero no en la ejecución final.

Para resolver esto docker introdujo los Multi-stage builds. Esta técnica permite usar varias etapas en un dockerfile:

- Etapa inicial donde se compila o construye la aplicación.
- Una Etapa final donde solo se copian los archivos realmente necesarios para ejecutar la app.

Ejemplo

```
# Etapa 1
FROM golang:1.19 As build
WORKDIR/app
COPY . .
RUN go build -o my-app
```

```
#Etapa 2
FROM alpine:latest
WORKDIR /root/
COPY --from=build /app/my-app
CMD ["./my-app"]
```

6 Conclusion

Docker es una tecnología que ha transformado la manera en la que las aplicaciones son desarrolladas, distribuidas y ejecutadas. Gracias a su ligereza, rapidez y portabilidad, se ha convertido en la base de la mayoría de entornos modernos en la nube.

7. Comandos útiles

7.1 Comandos básicos de información

```
docker --version
docker info
docker help
```

7.2 Imagenes

```
docker imagges // list All
docker pull ubuntu // Dowload
docker rmi <imagen> // delete
docker build -t mi-app .
docker history <imagen>
// historial de capas de una imagen
```

7.3 Contenedores

```
docker ps // list containers
docker p -a
docker run Hello-world
docker run -it ubuntu
docker start <id>
docker stop <id>
docker restart <id>
docker rm <id>
docker logs <id>
docker exec -it <id> sh
```

7.4 Redes

```
Docker network ls
Docker network create my_red
Docker network inspect my_red (app)
docker network connect my_red app
docker network disconnect my_red db
```

7.5 docker Compose

```
docker-compose up
docker-compose up -d
docker-Compose down
docker-Compose ps
docker-compose logs
```