Docker

Álvaro González Sotillo

8 de septiembre de 2024

Índice

1. Qué es una máquina virtual

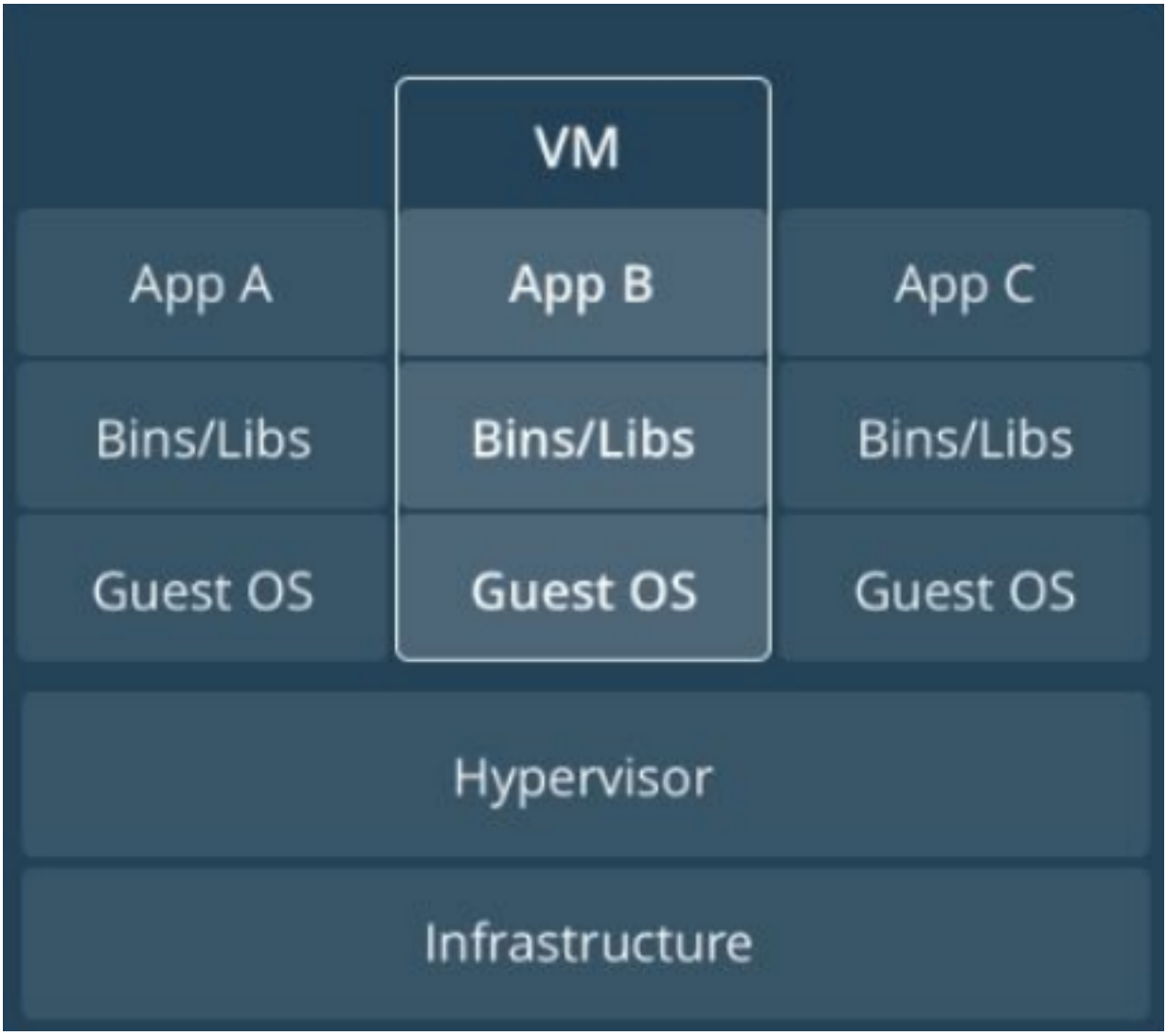
- Un proceso implementa todo el *hardware* de un PC
  - Representa los discos como ficheros
  - La tarjeta de vídeo utiliza una ventana en vez de un monitor
  - La tarjeta de red real se comparte, utilizando redes virtuales
- La BIOS/UEFI consigue arrancar normalmente, y ejecuta un sistema operativo
- Conclusión:
  - Un proceso en una máquina virtual no sabe que es *virtual*
  - Los procesos de la máquina virtual están **aislados** de la máquina real

2. Qué es Docker

- Docker integra diversas capacidades de aislamiento entre procesos que tiene Linux
  - chroot
  - cgroups
  - namespaces
- Usa además
  - mount points, para la comunicación entre *containers*
  - overlayfs, para construir containers acumulando *capas*
  - iptables/netfilter, para simular tarjetas de red
- Conclusión
  - Se pueden crear containers sin usar Docker
  - Docker también aisla procesos entre sí

3. Máquina virtual *vs* Docker

3.1. Comparación gráfica



3.2. Tabla comparativa

	Docker	Máquina Virtual
SO	SO compartido entre contenedores	Nuevo SO para cada VM
Seguridad	Menos seguro, se comparte el kernel	Más seguro, no se comparte nada
Rendimiento	Rendimiento rápido	La virtualización es más lenta
Tiempo de arranque	Rápido (segundos)	Lento (minutos)
Necesidades de memoria	Ligera	Requiere mucha memoria
Necesidades de almacenamiento	Normalmente megabytes	Normalmente gigabytes
Portabilidad	En cualquier linux moderno	El hardware emulado puede diferir entre diferentes s

4. Componentes Docker

- **Cliente**: Usa el *socket* de Docker para controlar el servidor
- **Servidor** (demonio): Gestiona las imágenes contenedores
- **Imagen**:
  - Personalizadas: con un *dockerfile*
  - Del registro: Imágenes prediseñadas (equivalentes a un fichero **OVA** para máquinas virtuales)
- **Contenedor** Una imagen en ejecución
- **Registro**: Almacén público de imágenes

5. Portabilidad de Docker

- ¿Hay Docker para Windows?
  - Respuesta corta: No
  - Respuesta larga:
    - Hay un *cliente* Docker para Windows
    - Se instala una máquina virtual con Linux
    - El cliente Docker se conecta al servidor Docker de la máquina virtual
    - También hay *containers* para Windows, solo instalables en Windows, que el cliente Docker sabe manejar

6. Instalación de Docker

- Página oficial
- Para Debian/Ubuntu:
  - Desinstalar las versiones instaladas con `apt-get`

```
for pkg in docker.io docker-doc docker-compose podman-docker containerd runc
do
    sudo apt-get remove $pkg
done
```

  - Usar el *script* (no recomendado en producción)

```
curl https://get.docker.com/ | sh
```

6.1. Prueba de instalación

- Solo `root` y el grupo `docker` pueden usar el servidor Docker
  - Tienen acceso a `/var/run/docker.sock`

```
docker run hello-world
```

7. Imágenes y *containers*

7.1. Conceptos básicos

- Una imagen se ejecuta en un *container*. La misma imagen se puede ejecutar en varios *containers*
- Una imagen provee un programa a ejecutar cuando se ejecuta con `run`
  - Cuando ese programa termina, el *container* se destruye
  - Se puede especificar otro programa a ejecutar
  - Se puede conectar la consola a dicho programa

```
docker run debian # crea un container y termina inmediatamente, bash no tiene entrada
docker run -it debian # crea un container y conecta la consola al programa por defecto
docker run debian /bin/bash -c "echo Un mensaje y termino"
```

7.2. Lista de containers e imágenes

```
docker images
docker ps # containers en ejecucion
docker ps -a # todos los containers
```

- Conclusión: `run` no ejecuta *containers*, sino que crea y ejecuta *containers*
  - `run` = `create` + `start`
- Los "parámetros"de un *container* se especifican en su creación. No se pueden cambiar al ejecutarlos.
- Los *container* tienen un nombre. Si no se especifica, Docker inventa uno.

8. Volúmenes

- Los *containers* son inmutables.
  - Los datos modificados en el *container* son una capa adicional (overlayfs)
  - Desaparecen al apagar el *container*
- La persistencia se puede conseguir con
  - volúmenes
  - *mounts*

8.1. *mounts*

- Equivalente a un `mount --bind`

```
docker run -it --mount type=bind,src="$(pwd)",target=/src debian bash
```

8.2. Volúmenes

- Puntos de montaje definidos al crear la imagen
  - Suelen ser directorios importantes para la aplicación *containerizada*
- Pueden ser
  - directorios creados y manejados internamente por Docker
  - directorios del *file system*

```
docker run -dit --name my-apache-app -p 8080:80 -v $HOME:/usr/local/apache2/htdocs/ httpd:2.4
```

- `-dit` : **i** nteractive, **t** ty, **d** etached
- `-p 8080:80` : El puerto `8080` de la dirección `0.0.0.0` se conecta a la *ip* del *container*, puerto `80`
- `-v` : El volumen `/usr/local/apache2/htdocs` será el directorio *home*
- Resultado: Apache corriendo en el puerto 8080 sirviendo los ficheros del directorio *home*

9. .Entrar.en un *container*

- Se pueden ejecutar comandos en un *container* en ejecución

```
docker exec -it my-apache-app bash
```

  - **-i**: Ejecutar de forma interactiva (redirigir entrada y salida estándar)
  - **-t**: Ejecutar en un TTY (una consola)

10. *Logs* de un *container*

- La salida del punto de entrada se considera el *log* del container

```
docker logs --follow nombre-container
```

11. Usuarios

- Los usuarios en Linux se identifican por un número
  - Docker comparte el *kernel* con el sistema operativo
  - Por tanto, los usuarios de los *containers* son los del Linux real
- Conclusiones
  - Se comparten los identificadores, no los nombres
  - El usuario *root* es el mismo en todos los *containers* y en el Linux real
  - Cuidado con la seguridad!

12. *Docker registry*

- Las imágenes se almacenan en registros
  - El registro por defecto es `https://registry.hub.docker.com`
  - Los siguientes comandos son equivalentes

```
docker run hello-world
docker run registry.hub.docker.com/hello-world
```

- Oracle tiene un registro para sus imágenes
  - `https://container-registry.oracle.com/`
  - Se necesita registro (gratis) en Oracle
  - La contraseña del registro no es la de la cuenta de Oracle, sino el *Auth token*

```
docker login container-registry.oracle.com/
```

12.1. *Cache* en el instituto

`10.1.33.201:8090`

- Es un *registry* inseguro (sin https).
- Para poder usarse hay que modificar/crear el fichero `/etc/docker/daemon.json`

```
{
    "insecure-registries":["10.1.33.201:8090"]
}
```

13. Práctica: Instalar Webmin

- Es un sistema para manejar servidores Linux mediante con una interfaz web
- Comprueba qué se ha instalado en el *container* y lanza una *shell* desde la web

```
sudo docker run -p 10000:10000 10.1.33.201:8090/johanp/webmin
```

14. Práctica: Instalar Portainer

- Es un sistema para manejar los contenedores Docker, instalado dentro de Docker (¿?)
  - Lo consigue dejando que el container acceda a `/var/run/docker.sock`
- https://docs.portainer.io/start/install-ce/server/docker/linux

```
docker volume create portainer_data
docker run -d -p 8000:8000 -p 9443:9443 \
  --name portainer \
  --restart=always \
  -v /var/run/docker.sock:/var/run/docker.sock \
  -v portainer_data:/data portainer/portainer-ce:latest
```

15. Práctica: instalar Oracle

- Usaremos la imagen `container-registry.oracle.com/database/enterprise:21.3.0.0`
- Se recomienda usar el *registry* del instituto, que no necesita contraseña
  - `sudo docker pull 10.1.33.201:8090/database/enterprise:21.3.0.0`

15.1. Instrucciones

- Instrucciones: https://container-registry.oracle.com/ords/ocr/ba/database/enterprise

```
docker run -d --name <container_name> \
 -p <host_port>:1521 -p <host_port>:5500 \
 -e ORACLE_SID=<your_SID> \
 -e ORACLE_PDB=<your_PDBname> \
 -e ORACLE_PWD=<your_database_password> \
 -e INIT_SGA_SIZE=<your_database_SGA_memory_MB> \
 -e INIT_PGA_SIZE=<your_database_PGA_memory_MB> \
 -e ORACLE_EDITION=<your_database_edition> \
 -e ORACLE_CHARACTERSET=<your_character_set> \
 -e ENABLE_ARCHIVELOG=true \
 -v [<host_mount_point>:]/opt/oracle/oradata \
container-registry.oracle.com/database/enterprise:21.3.0.0
```

```
Parameters:
 --name:                 The name of the container (default: auto generated
 -p:                     The port mapping of the host port to the container port.
                         Two ports are exposed: 1521 (Oracle Listener), 5500 (OEM Express)
 -e ORACLE_SID:          The Oracle Database SID that should be used (default:ORCLCDB)
 -e ORACLE_PDB:          The Oracle Database PDB name that should be used (default: ORCLPDB1)
 -e ORACLE_PWD:          The Oracle Database SYS, SYSTEM and PDBADMIN password (default: auto
 -e INIT_SGA_SIZE:       The total memory in MB that should be used for all SGA components (d
 -e INIT_PGA_SIZE:       The target aggregate PGA memory in MB that should be used for all se
 -e ORACLE_EDITION:      The Oracle Database Edition (enterprise/standard, default: enterpris
 -e ORACLE_CHARACTERSET: The character set to use when creating the database (default: AL32UT
 -e ENABLE_ARCHIVELOG:   To enable archive log mode when creating the database (default: fals
 -v /opt/oracle/oradata
                         The data volume to use for the database. Has to be writable by the U
                         If omitted the database will not be persisted over container recreat
 -v /opt/oracle/scripts/startup | /docker-entrypoint-initdb.d/startup
                         Optional: A volume with custom scripts to be run after database star
                         For further details see the "Running scripts after setup and on
                         startup" section below.
 -v /opt/oracle/scripts/setup | /docker-entrypoint-initdb.d/setup
                         Optional: A volume with custom scripts to be run after database setu
                         For further details see the "Running scripts after setup and on star
```

15.2. En nuestro caso

- Puerto `1521` del container al `1521` de la máquina *host*
- Ficheros de datos en `/opt/oracle/oradata` del la máquina *host*
- Nombre del container `oracle-enterprise-21.3.0.0`
- SID: `ORADOCKER`
- PDB: `PDBORADOCKER`
- Haz un *script* en el *host* para ejecutar `sqlplus` como `SYSDBA`
  - `/opt/oracle/sqplus-as-sysdba.sh`
  - con `docker exec`
  - puede ayudarte `bash -c`
- Haz un *script* en el *host* para ejecutar `sqlplus` como `SYSDBA` en la PDB
  - `/opt/oracle/sqplus-as-sysdba-pdb.sh`

16. Referencias

- Formatos:
  - Transparencias
  - PDF
  - Página web
  - EPUB
- Creado con:
  - Emacs
  - org-re-reveal

    - Latex
- Alojado en Github