

Tema 5: Introducción a Docker

Máster Universitario en Ingeniería Informática
Extracción y Explotación de la Información

Índice

- ¿Qué es Docker?
 - Docker vs. virtual machines
 - Tecnología docker
 - Ventajas y desventajas de docker
- Instalación y comandos básicos

¿Qué es Docker?



- Docker es el proyecto open-source que automatiza el despliegue de aplicaciones dentro de contenedores de software, lo que aporta una capa adicional de abstracción y automatización de la virtualización a nivel de sistema operativo en Linux.
- Docker representa una evolución de la tecnología patentada de la empresa dotCloud, Inc. (actualmente, Docker, Inc., con un valor sobre los \$1.000 M), aunque ha recibido contribuciones de Red Hat, IBM, Google, y otras.
- Con Docker se puede empaquetar una aplicación y sus dependencias en un contenedor virtual que se puede ejecutar en cualquier servidor Linux. Esto ayuda a alcanzar flexibilidad y portabilidad en la ejecución de aplicaciones, ya sea en instalaciones físicas, la nube pública, privada, etc.

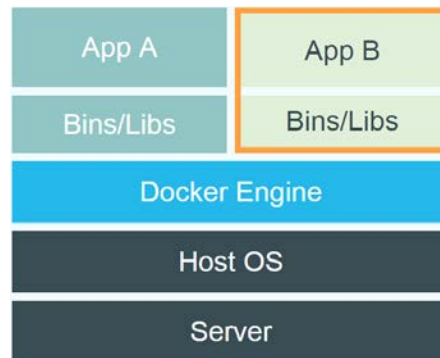
¿Qué es Docker?



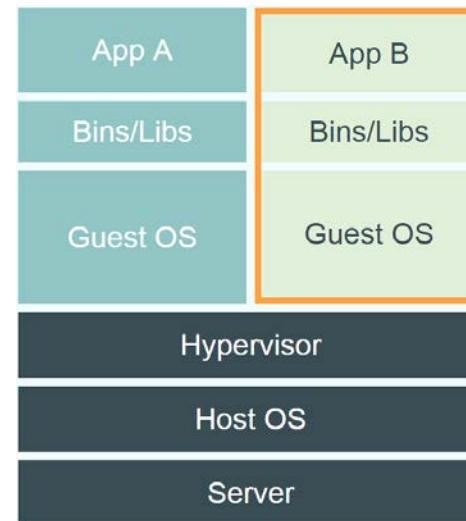
- Docker provee de un mecanismo sencillo integrado en un paquete de software para “construir, trasladar y ejecutar cualquier aplicación en cualquier lado” (any app anywhere).
- De forma similar al transporte de contenedores de mercancías (el contenedor es el mismo, lo que cambia es su contenido, pero independientemente de lo que haya dentro se puede llevar en camiones, barcos, ...)

Docker vs. Virtual Machines

- Docker utiliza características de aislamiento de recursos del kernel de Linux, lo que permite ejecutar "contenedores" independientes dentro de una sola instancia de Linux. Esto evita la sobrecarga de iniciar y mantener máquinas virtuales.



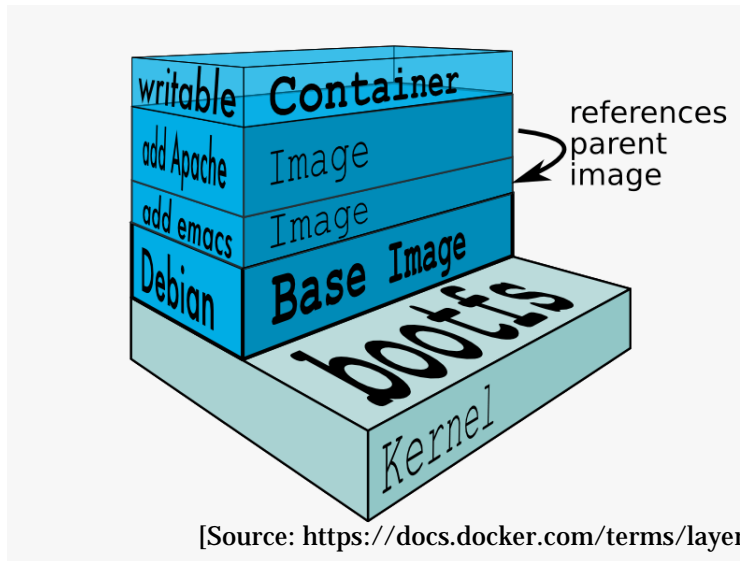
El contenedor docker comprende sólo la aplicación y sus dependencias. Se ejecuta como un proceso aislado en un espacio de usuario del sistema operativo anfitrión. Comparte el kernel con otros contenedores.



Cada aplicación virtualizada incluye, además de la aplicación (que pueden ser sólo unos pocos Mbs) y las librerías y binarios que requiere, también todo el sistema operativo invitado (que pueden ser de varios Gbs)

Tecnología Docker

- Docker utiliza la plataforma libvirt de virtualización
- Soporta LXC (Linux Containers): Múltiples sistema Linux independientes (contenedores) corriendo sobre un mismo host.
- Es un sistema de archivos en capas



Ventajas y desventajas Docker

- ¿Por qué correr 100 máquinas virtuales por separado cuando puedes tener 10 máquinas virtuales ejecutando 10 containers?
- Si muchas de las máquinas virtuales corren a un 1% de utilización, entonces Docker puede reducir la factura de forma significativa.
- Entre las limitaciones (por mencionar alguna), hay que decir que la API cambia de forma continua, al menos en las primeras versiones.
- No hay un almacenamiento persistente: si el contenedor cae, se pierden los archivos que se hayan creado dentro de él durante su ejecución (la recomendación es utilizar volúmenes de datos externos o hacer commit de los cambios cuando los haya).

Índice

- ✓ ¿Qué es Docker?
- Instalación y comandos básicos
 - Gestionar datos en contenedores
 - Contruir una imagen en docker
 - Ejemplo de Dockerfile
 - Comandos Dockerfile

Instalación y comandos básicos

- Instalar docker (<https://www.digitalocean.com/community/tutorials/como-instalar-y-usar-docker-en-ubuntu-16-04-es>)

```
bigdata@ubuntu:~$ sudo apt-get update
bigdata@ubuntu:~$ sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --
recv-keys 58118E89F3A912897C070ADBF76221572C52609D
bigdata@ubuntu:~$ sudo apt-add-repository 'deb https://apt.dockerproject.org/repo
ubuntu-xenial main'
bigdata@ubuntu:~$ sudo apt-get update
bigdata@ubuntu:~$ apt-cache policy docker-engine
bigdata@ubuntu:~$ sudo apt-get install -y docker-engine
```

- Comprobar que se está ejecutando

```
bigdata@ubuntu:~$ sudo systemctl status docker
```

- Opcional, si no queremos ejecutar con sudo:

```
bigdata@ubuntu:~$ sudo usermod -aG docker $(whoami)
```

Instalación y comandos básicos

- Los contenedores Docker se ejecutan desde imágenes de Docker. De forma predeterminada, estas imágenes se alojan en Docker Hub, un registro de Docker administrado por la compañía detrás del proyecto Docker.
- Se pueden construir y alojar imágenes Docker en Docker Hub. Estas imágenes en general son la base para ejecutar ciertas aplicaciones y distribuciones Linux que corren en los contenedores Docker.
- Para comprobar si puede acceder y descargar imágenes de Docker Hub:

```
bigdata@ubuntu:~$ docker run hello-world
bigdata@ubuntu:~$ docker search ubuntu
bigdata@ubuntu:~$ docker search cpgonzal
bigdata@ubuntu:~$ docker pull ubuntu
bigdata@ubuntu:~$ docker run ubuntu echo Hello World
```

Instalación y comandos básicos

- Para ver las imágenes descargadas:

```
bigdata@ubuntu:~$ docker images
```

- Para ejecutar un contenedor:

```
bigdata@ubuntu:~$ docker run -it ubuntu
```



```
root@d9b100f2f636:/#
```

container-id

- Para ver los contenedores:

```
bigdata@ubuntu:~$ docker ps -a
```

- Para detener un contenedor:

```
bigdata@ubuntu:~$ docker stop <container-id>
```

- Para eliminar contenedores:

```
bigdata@ubuntu:~$ docker rm $(docker ps -q -f status=exited)
```

Instalación y comandos básicos

- Un resumen de los comandos de docker:

attach	Adjunta a un contenedor corriendo
build	Construye un contenedor de un archivo Docker
commit	Crea una nueva imagen de los cambios del contenedor
cp	Copia archivos/carpetas de los contenedores del sistema de archivos a la ruta de host
diff	Inspecciona los cambios en el sistema de archivos de un contenedor
history	Muestra el historial de una imagen
info	Inserta un archivo en una imagen
kill	Mata a un contenedor en ejecución (corriendo)
load	Carga una imagen desde un archivo tar
ps	Lista los Contenedores

Gestionar datos en contenedores

- ¿Qué pasa si creamos un fichero en un contenedor?

```
bigdata@ubuntu:~$ docker run -it cpgonzal/ubuntu /bin/bash
root@d9b100f2f636:/# mkdir -p ~/docker/vol01
root@d9b100f2f636:/# echo "I'm not going anywhere" > ~/docker/vol01/test.txt
root@d9b100f2f636:/# exit
```

- Si volvemos a ejecutar de nuevo el contenedor:

```
bigdata@ubuntu:~$ docker run -it cpgonzal/ubuntu
root@d9b100f2f636:/# ls ~/docker/vol01
```

- Lo que se puede hacer es que, antes de salir del contenedor, se haga un commit de cambios:

```
bigdata@ubuntu:~$ docker commit -m "added sth" -a "ubuntu_new" 2e05ad1907c5 ubuntu
```

Gestionar datos en contenedores

- Una forma de persistir los datos utilizando docker consiste en utilizar volúmenes de datos, que son directorios especialmente diseñados para montar dentro de uno o más contenedores y que tiene algunas características de interés:
 - Volúmenes son inicializados cuando se crea un contenedor.
 - Los volúmenes pueden ser compartidos y reutilizados entre contenedores.
 - Los cambios a un volumen de datos son ejecutados directamente.
 - Los volúmenes de datos persisten aunque el contenedor se elimine.
- Los volúmenes de datos están diseñados para persistir los datos, con independencia del ciclo de vida del contenedor. Docker por tanto nunca elimina automáticamente los volúmenes cuando eliminas un contenedor.

Gestionar datos en contenedores

- Para crear un volumen de datos:

```
bigdata@ubuntu:~$ docker create -v /root/docker/vol01 --name datacontainer ubuntu
```

- Ejecutamos de nuevo el contenedor:

```
bigdata@ubuntu:~$ docker run -it --volumes-from datacontainer ubuntu /bin/bash
root@d9b100f2f636:/# mkdir -p ~/docker/vol01
root@d9b100f2f636:/# echo "Here I am!!!" > ~/docker/vol01/test.txt
root@d9b100f2f636:/# exit
```

- Si volvemos a ejecutar de nuevo el contenedor:

```
bigdata@ubuntu:~$ docker run -it --volumes-from datacontainer ubuntu /bin/bash
root@d9b100f2f636:/# ls ~/docker/vol01
```

Gestionar datos en contenedores

- También podemos obviar el uso de volúmenes de datos y compartir archivos entre el host y el contenedor

```
bigdata@ubuntu:~$ mkdir -p ~/docker/vol01  
bigdata@ubuntu:~$ echo "Other test" > ~/docker/vol01/other_test.txt  
bigdata@ubuntu:~$ docker run -v ~/docker/vol01:/root/docker/vol01 -it ubuntu
```


Contruir una imagen en docker

- Para construir una imagen en docker es necesario generar un archivo Dockerfile, que es un script con los comandos que serán ejecutados de forma secuencial por docker para crear automáticamente una nueva imagen de Docker.
- Recordemos que la imagen en Docker es la base sobre la que se arranca los contenedores, con lo cual generar una imagen con los elementos necesarios facilita mucho el despliegue de las aplicaciones.
- Los Dockerfile comienzan siempre con la definición de una imagen base utilizando el comando FROM. Desde ahí, el procesos de build continúa y cada acción se va ejecutando y finalizando con commits que guardan el estado de la imagen en el host

Ejemplo de Dockerfile

```
# Set the base image
FROM ubuntu


# Dockerfile author / maintainer
MAINTAINER Carlos P. <cpgonzal@gmail.com>

ARG GUID
ARG USER
ARG DATA_ROOT
ENV GUID ${GUID:-900}
ENV USER ${USER:-docker}
ENV DATA_ROOT ${DATA_ROOT:-/data}

## Set a default user. Available via runtime flag `--user docker`
## User should also have & own a home directory
RUN groupadd -r -g $GUID $USER \
    && useradd -g $USER -u $GUID $USER \
        && mkdir /home/$USER \
        && chown $USER:$USER /home/$USER

# create directory for persistence and give our user ownership
RUN mkdir -p $DATA_ROOT \
    && chown -R $USER:$USER $DATA_ROOT

#Assigns write permission to group
RUN echo "umask 0002" >> /etc/bash.bashrc
```



```
# persist data, set user
VOLUME $DATA_ROOT
USER $USER
```

```
CMD ["echo", "Data container for data and libraries"]
```

Comandos Dockerfile

- Un resumen de los comandos (<https://www.digitalocean.com/community/tutorials/como-instalar-y-usar-docker-en-ubuntu-16-04-es>) :

ADD	Copia un archivo desde el host en el contenedor
CMD	Configura comandos por defecto para ser ejecutado, o se pasa al punto de entrada ENTRYPOINT
ENTRYPOINT	Ajusta el punto de entrada por defecto de la aplicación desde el contenedor
ENV	Inicializa variables de entorno (por ejemplo, "variable=valor")
EXPOSE	Expone un puerto al exterior
FROM	Configura la imagen base para usar
MAINTAINER	Establece los datos de autor/propietario del archivo Dockerfile
RUN	Ejecuta un comando y cambia (commit) el resultado de la la imagen final (contenedor)

Comandos Dockerfile

- Un resumen de los comandos (<https://www.digitalocean.com/community/tutorials/como-instalar-y-usar-docker-en-ubuntu-16-04-es>) :

USER	Establece el usuario para ejecutar los contenedores de la imagen
VOLUMEN	Monta un directorio desde el host al contenedor
WORKDIR	Establece el directorio para las directivas de CMD que se ejecutarán

Actividad: crear una imagen con R

- Examinar el archivo dockerfileR:

```
bigdata@ubuntu:~$ vi DockerfileR
```

- Hacer un build de la imagen:

```
bigdata@ubuntu:~$ docker build --build-arg R_VERSION=3.3.2 -t docker-r -f DockerfileR .  
bigdata@ubuntu:~$ docker run --rm --name r-dock -it docker-r /bin/bash
```