

Introducción. Conceptos básicos

Virtualización vs. Contenedores	La virtualización emula hardware completo; los contenedores comparten el sistema anfitrión, aislando aplicaciones.
Docker	Sistema de contenedores en Linux que facilita el despliegue y ejecución de aplicaciones en entornos controlados y consistentes.
Cliente Docker	Interfaz que se comunica con el servidor.
Servidor Docker (Host)	Gestiona contenedores e imágenes.
Ventajas de Docker	Ahorra espacio y recursos. Compatible con entornos de desarrollo, pruebas y producción. Velocidad de ejecución cercana a la nativa.
Limitaciones	Menor rendimiento que en hardware dedicado. Persistencia de datos más compleja. Preferencia por línea de comandos (puede ser tedioso crear entornos gráficos).

Introducción. Conceptos básicos (cont)

Contenedor	Un contenedor es un entorno aislado que ejecuta aplicaciones con sus dependencias, compartiendo el kernel del sistema operativo anfitrión. Es más ligero y rápido que una máquina virtual.
Imagen	Plantilla inmutable que define qué contiene y cómo funciona un contenedor. Las imágenes incluyen el sistema de archivos y las aplicaciones necesarias y pueden ser almacenadas en repositorios como Docker Hub.
Volumen	Almacenamiento persistente que permite guardar datos de forma independiente al ciclo de vida del contenedor, facilitando el manejo de datos duraderos entre diferentes instancias.

Principales acciones

Crear un grupo docker	sudo groupadd docker
Añadir tu usuario a un grupo docker	sudo usermod -aG docker \$USER

Principales acciones (cont)

Crear contenedor apartir de una imagen, modo interactivo. Para poder interactuar con el contenedor creado, asignar terminal interactivo	docker run -it nombre-Imagen
Crear un contenedor con la imagen "ubuntu"	docker run -it --name=nombreRandom ubuntu /bin/bash
Mostrar informacion contenedores en ejecución	docker ps
Muestra información de todos los contenedores	docker ps -a
Arranca un contenedor concreto	docker start nombre-Contenedor
Ejecutar comandos dentro de un contenedor que esta en ejecución	docker exec
Crear y inicializar un contenedor	docker run
Detener contenedores activos con nombreContenedor	docker stop nombre-Contenedor
Conectarse a la consola de un contenedor en ejecución	docker attach IDCONTAINER
Muestra información del proceso de ejecución del contenedor	docker logs IDCONTAINER
Copiar fichero /tmp/nombreFichero del contenedor al directorio actual del anfitrión	docker cp IDCONTAINER:/tmp/nombreFichero ./



Principales acciones (cont)

Copiar un fichero del directorio actual anfitrión a la carpeta /tmp del contenedor	<code>docker cp ./nombreFichero IDCONTAINER:/tmp</code>
--	---

Volúmenes

Crear un volúmen	<code>docker volume create nombreVolumen</code>
Listar volúmenes	<code>docker volume ls</code>
Listar volúmenes activos	<code>docker volume ls -a</code>
Visualizar volúmenes	<code>docker volume inspect nombreVolumen</code>
Borrar volúmen	<code>docker volume rm nombreVolumen</code>
Crear contenedor y asignar volumen	<code>docker run -d -it -- nombreContenedor -v nombreVolumen</code>
Examinar contenido volumen "metadatos"	<code>docker volum inspect nombreVolumen</code>

Gestión de redes

Creamos una red	<code>docker network create nombre-red</code>
Listamos las redes existentes	<code>docker network ls</code>
Borramos una red	<code>docker network rm nombre-red</code>
Conectar el contenedor creado "ubuntu" a una red	<code>docker run -it -- network nombre-red ubuntu /bin/bash</code>
Conectar un contenedor a una red	<code>docker network connect IDRED IDCONTENEDOR</code>
Desconectar un contenedor de una red	<code>docker network disconnect IDRED IDCONTENEDOR</code>

Imágenes

Información imagenes disponibles	<code>docker images</code>
Buscar una imagen en concreto	<code>docker search nombreImagen</code>
Descargar localmente una imagen existente en remoto	<code>docker pull nombreImagen</code>
Eliminar localmente una imagen	<code>docker rmi nombreImagen</code>
Borrar un contenedor	<code>docker rm IDCONTENEDOR</code>
Mostrar historial de la creación de una imagen	<code>docker history nombreImagen</code>
Borra toda imagen local que no esté siendo usada por un contenedor	<code>docker rmi \$(docker images -q)</code>
Parar todos los contenedores	<code>docker stop \$(docker ps -a -q)</code>
Paramos todos los contenedores parados	<code>docker rm \$docker ps -a -q)</code>
Borrar imagenes y contenedores parados	<code>docker system prune -a</code>
Commit de un contenedor a una imagen local	<code>docker commit -m "comentario" IDCONTENEDOR usuario/imagen:version</code>
Guarda una copia de seguridad de una imagen en fichero ".tar".	<code>docker save -o copiaSeguridad.tar imagenA</code>
Restaura una copia de seguridad de una imagen en fichero ".tar".	<code>docker load -i copiaSeguridad.tar</code>

Docker Hub

Login a Docker Hub en remoto	<code>docker login</code>
Subir al repositorio una imagen	<code>docker push usuario/imagen:version</code>

Docker Compose

Inicia el sistema definido en "docker-compose.yml" en segundo plano. Arranca la aplicación con Compose	<code>docker-compose up -d</code>
Detiene servicios	<code>docker-compose stop</code>
Detiene y elimina los contenedores según la configuración de "docker-compose.yml"	<code>docker-compose down</code>
Detiene y elimina volúmenes y contenedores	<code>docker-compose down -v</code>
Muestra información de los servicios que se define en docker-compose.yml. Muestra información de los contenedores según la configuración de "docker-compose.yml"	<code>docker-compose ps</code>
Construye/descarga las imágenes de contenedores según la configuración de "docker-compose.yml"	<code>docker-compose build/pull</code>
Parámetro docker -link	Parámetro compose depends_on
Parámetro docker -mount	Parámetro compose volumes
Parámetro docker -e	Parámetro compose environment
Parámetro docker -p	Parámetro compose ports



Dockerfile

Crear imagen a partir de Dockerfile `docker build -t nombre-Imagen`

Imagen base de la que partimos `FROM`

Comandos que lanzaremos sobre la imagen base, para crear una nueva `RUN`

Comando asociado al lanzar un contenedor con la nueva imagen `CMD`

Para indicar los puertos por defecto expuestos que tendrá el contenedor `EXPOSE`

Descomprimir el contenido de un archivo en el directorio destino `ADD`

Copiar ficheros de la máquina anfitrión al nuevo contenedor `COPY`

Cambiar comandos que se lanzan por defecto al ejecutar un contenedor `ENTRYPOINT`

Cambiar el usuario por defecto (root) a la hora de lanzar comandos al crear la imagen `USER`

Cambiar el directorio de la imagen donde se ejecutaran los comandos `WORKDIR`

Definir variables de entorno en la imagen `ENV`

Enviar parámetros al propio Dockerfile `ARG`

Establecer volúmenes por defecto en la imagen `VOLUME`

Establecer metadatos dentro de la imagen mediante etiquetas `LABEL`

Dockerfile (cont)

Definir como se comprobará si ese contenedor está funcionando correctamente `HEALTH CHECK`



By **Jordi Allepuz**
cheatography.com/jordi-allepuz/

Not published yet.
 Last updated 28th October, 2024.
 Page 3 of 3.

Sponsored by **Readable.com**
 Measure your website readability!
<https://readable.com>