docker, comandos

Comandos para Docker

Comandos de información

```
docker info
```

```
docker version
```

Gestión de imágenes

Buscar una imagen

Para buscar una imagen usamos **docker search <imagen_a_buscar>**, por ejemplo

```
docker search centos
```

Listar las imágenes que tenemos descargadas

```
sudo docker images ls
```

Obtener información sobre una imagen concreta

```
sudo docker history <imagen>
```

Descargar una imagen

```
sudo docker pull <nombreimagen
```

Por ejemplo para descargar la imagen de kali linux

```
sudo docker pull kalilinux/kali-linux-docker
```

Obtener detealles sobre una imagen

```
docker inspect <imagen>
```

Borrar una imagen

```
docker image rm
```

Gestión de Contenedores

Crear un contenedor

```
docker create --name <nombre> <imagen>
```

Por ejemplo

```
create --name mihola holamundo
```

. Crea un contenedor llamado mihola, usando una imagen que se llama holamundo.

Lo crea pero no lo arranca

Arrancar un Contendor

```
sudo docker run -opciones nombre_imagen o codigo_imagen
```

```
sudo docker run -t -i kalilinux/kali-linux-docker /bin/bash
```

la opción -i es modo interactivo .

Por defecto docker usa nombre de contenedores aleatorios, si queremos que nuestro contenedor tenga un nombre determinado haremos

```
docker run --name <nombreimagen>
```

Arrancar un contenedor mapeando puertos

```
docker run -p <puerto host>:<puerto contenedor> <imagen>
```

Por ejemplo para exponer los puerto de un contenedor con ngnix

```
docker run -p 80:80 -p 443:443 nginx:latest
```

Si estamos usando una imagen creada con dockerfile en la que hemos definido un puerto . Entonces usaremos la opción P mayúscula para que el mismo nos mapee un puerto aleatorio del host al que hemos definido en el contenedor.

```
docker run -P --name micontenedor miimagen:latest
```

Ver los contenedores disponibles

Para que nos muestre los contenedores en ejecución

```
docker ps
```

Para que nos muestr todos los contenedores, activos o inactivos

```
docker ps -a
```

Los campos que muestra son:

- CONTAINER ID = Identificador único del contenedor
- IMAGE = La imagen utilizado para la creación del contenedor
- COMMAND = Comando ejecutado en el momento de crear el contenedor
- CREATED = Muestra el tiempo de vida que tiene el contenedor
- STATUS = Muestra el estado actual del contenedor
- PORTS = Muestra el puerto que la aplicación dentro del contenedor utiliza para recibir conexiones
- NAMES = Nombre del contenedor

Acceder a un contenedor

Para acceder al contenedor, además de crearlo, se puede hacer de dos maneras. Una es haciendo referencia al IMAGE ID y otra al repositorio (REPOSITORY) y la etiqueta (TAG).

```
docker run -i -t b72879fa579a /bin/bash
```

O también:

```
docker run -i -t ubuntu:14.04 /bin/bash
```

Para salir de una imagen, debes presionar CTRL+D.

Borrar contenedores sin uso

con el comando

```
docker system prune
```

con la opción -a elimina_

- los contenedores que no se usan
- los volúmenes que no se usan
- las imágenes que no se están usando
- las redes que no se están usando

Mucho ojo al ejecutar este comando en sistemas en producción

Etiquetar

También podemos poner una etiqueta a nuestros contenedores, y llamarlo por dicha etiqueta, lo cual nos permitirá organizar mejor todos nuestros contenedores. Para poner una etiqueta

```
docker tag id_imagen repositorio:etiqueta
```

Para llamar a dicho contenedor por la etiqueta, hacemos lo mismo que cuando lo llamamos por el id pero poniendo ahora la etiqueta

```
docker tun -i -t repositorio:etiqueta /bin/bash
```

Es habitual por ejemplo a una imagen que tengamos creada ponerle una etiqueta de latest para indicar que es la última versión disponible

```
docker tag miimagen:version miimagen:latest
```

De esta forma si hacemos un **docker images** veremos que la misma imagen aparece dos veces pero si nos fijamos en campo **image id** es la misma imagen. Es decir es como si hubieramos creado un enlace para poder llamar a la misma imagen.

Iniciar contenedor

```
docker start imagenid
```

o bien con

```
docker start nombre
```

Con estos comandos arrancamos el contenedor pero no nos conectamos al mismo. Si queremos acceder ejecutamos

```
docker attach id
```

Parar contenedor

Para parar un contenedor

```
docker stop imagenid_o nombre
```

Para parar todos los contenedores

```
docker stop $(docker ps -a -q)
```

Salir

Escribiendo **exit** en nuestro contenedor, o Pulsando CTRL+D salimos del mismo pero **parando la ejecución del mismo**. Si queremos salir del contenedor pero que se siga ejecutando debemos presionar CTRL, después P y luego Q

Guardar Contenedor

Las imágenes son plantillas de sólo lectura, que usamos de base para lanzar contenedores. Por tanto todo lo que hagamos en el contenedor sólo persiste en ese contenedor y **NO** se guardan en la imagen.

Si queremos que dichos cambios sean permanentes, debemos crear una nueva imagen con el contenedor personalizado.

```
docker commit -m "comentario" -a "autor" <identificadordelcontenedor>
nuevonombreimagen
```

Por ejemplo

```
docker commit -m "Imagen actualizada centos" -a "LC" c605d57c9aa1
centosactualizado:v1
```

Con commit creamos una nueva imagen en nuestro repositorio local.

Borrar Contenedor

```
docker rm <contenedor>
```

Para borrar todos los contenedores

```
docker rm $(docker ps -a -q)
```

Probar

```
docker container rm $(docker container ls -a -q)
```

Copiar desde un contenedor

Para copiar un fichero desde un contenedor a nuestra máquina hacemos

```
docker cp <nombre_contenedor o id>:<ruta_al_fichero>
<directorio_local_a_donde_copiar>
```

También podemos hacerlo a la inversa. Desde la máquina local al contenedor

Ejecutar comando

Podemos ejecutar un comando dentro de un contenedor con

```
docker exec <nombre o id contenedor> <comando>
```

Por ejemplo para iniciar un shell intereactivo

```
docker exec -it  micontenedor sh
```

Para sar un listado de los procesos que corren en el contenedor

```
docker exec micontenedor ps
```

BACKGROUND

Con la opción **-d** Nos permite ejecutar un contenedor en segundo plano y poder correr comandos sobre el mismo en cualquier momento mientras esté en ejecución. Se dice que es un contenedor demonizado y se ejecutará indefinidamente

por ejemplo

```
docker run -d --name tomcat:v8 miimagentomcat
```

logs

Para ver los logs que está generando un contenedor, ejecutaríamos el comando

```
docker logs <nombre contenedor o id>
```

Estadísticas de uso

con el comando stats obtenemos estadísticas de uso y consumo de nuestro contenedor

```
docker stats <nombre contenedor o id>
```

Gestión de volúmenes

Ver los volúmenes

Lista los volúmenes creados en Docker.

```
docker volume ls
```

Un volume nos permite guardar información de forma persistente. Permite que podamos destruir un contenedor sin perder los datos.

borrar todos los volúmenes

```
docker volume rm $(docker volume ls -q)
```