DOCKER

COMANDOS	DESCRIPCIÓN
`docker version`	Ver versión de docker
`docker <comando> --help`	Ver ayuda para un comando
`docker run <imagen>`	Ejecuta un contenedor con la imagen proporcionada
`docker images`	Muestra todas las imágenes descargadas en el ordenador
`docker ps`	Muestra los contenedores activos
`docker ps -a`	Muestra TODOS los contenedores (activos y parados)
`docker pull <imagen>`	SOLO descarga una imagen y NO EJECUTA un contenedor
`docker ps -l`	Ver el ÚLTIMO contenedor arrancado/modificado
`docker ps -n 4`	Ver los 4 ÚLTIMOS contenedores arrancados/modificados
`docker ps -a -q`	Ver los ids de TODOS los contenedores
`docker ps -a -n 3 -s`	Calcula el tamaño total de los 3 últimos contenedores (-s)
`docker ps -a -f "name=<nombre>"`	Busca contenedores aplicando el filtro de nombre. (Ver filtros)
`docker images -q`	Ver los ids de las imágenes
`docker run --name <nombre> <imagen>`	Lanza un contenedor con nombre personalizado para esa imagen
`docker run --name <nombre> -it <imagen>`	Lanza un contenedor interactivo (-i) y nos proporciona una terminal (-t)
`docker run --name <nombre> -it <imagen> bash`	Hay ciertos contenedores que requieren la palabra bash para lanzar la terminal
`docker stop <contenedor>`	Para el contenedor especificado
`docker start <contenedor>`	Inicia el contenedor especificado

DOCKER

COMANDOS	DESCRIPCIÓN
docker start -i <contenedor>	Inicia un contenedor en modo interactivo
docker run -d <imagen>	Inicia un contenedor en segundo plano
docker run –name <nombre> -d -it <imagen>	Viene bien para mantener vivo un contenedor de sistema
docker rm <id contenedor | nombre >	Borra un contenedor
docker rmi <id imagen | nombre >	Borra una imagen
docker rmi -f <id imagen | nombre >	Borra una imagen aunque la esté usando un contenedor activo
docker exec <contenedor> <comando>	Lanza un comando contra un contenedor
docker attach <contenedor>	Ver la salida de un comando que se está ejecutando en un contenedor
docker logs <contenedor>	Ver lo que está mostrando por pantalla un contenedor
docker logs <contenedor> --tail 5	Muestra las últimas 5 líneas de la pantalla de un contenedor
docker logs <contenedor> -f	Ver lo que está mostrando por pantalla un contenedor sin parar
docker kill <contenedor>	Matar un contenedor
docker top <contenedor>	Mira el proceso que más recursos esté consumiendo de un contenedor
docker stats <contenedor>	Vemos información detallada de un contenedor (cpu, memoria, limite de memoria, etc…)
docker inspect <contenedor> > container.txt	Vuelca un JSON con las propiedades del contenedor
docker inspect <imagen> > image.txt	Vuelca un JSON con las propiedades de la imagen
docker system info	Muestra información del sistema relacionada con docker
docker system df	Nos muestra cantidad y tamaño de imágenes y contenedores

DOCKER

COMANDOS	DESCRIPCIÓN
docker system events	Muestra los eventos que pasan en docker (creación de contenedores, imágenes, redes, etc…)
docker system prune	Borra todas las cosas sin utilizar de docker. SUMO CUIDADO CON ESTE COMANDO
docker cp <fichero> <contenedor:ruta_fichero>	Copia un fichero a la carpeta /tmp dentro del contenedor
docker cp <contenedor:ruta_fichero> .	Copia un fichero del contenedor al sistema operativo real a la ubicación actual
docker run --name <nombre contenedor> -d -it -e V1=10 <imagen>	Crea un contenedor a partir de una imagen con una variable de entorno (-e V1 = 10)
docker run -it -d --rm --name <nombre contenedor> <imagen>	Crea un contenedor a partir de una imagen que se borra automáticamente al finalizar
docker run --name <nombre contendor> -d --env-file <archivo> <imagen>	Crea un contenedor con variables de entorno a partir de un fichero de propiedades
docker run --name <nombre contenedor> -d -p <puerto expuesto>:<puerto interno> <imagen>	Crea un contenedor en segundo plano. Lo que hace el argumento -p es decir que cuando un cliente acceda al puerto de la máquina real, se le redirigirá al puerto interno del contenedor.
docker network ls	Lista todas las redes de docker
docker network inspect <red>	Devuelve un JSON con las propiedades de la red. Podemos ver todos los contenedores asociados a esta misma
docker image inspect <imagen>	Devuelve un JSON con todas las propiedades de una imagen
docker network create --subnet=192.168.0.0/16 <nombre red>	Crea una red con las características proporcionadas
docker run -it --name <nombre contenedor> --network <nombre red> <nombre imagen>	Crea un contenedor interactivo y lo asocia a una red.
docker network connect <red> <contenedor>	Conecta un contenedor a la red
docker network create --subnet=192.168.0.0/16 --gateway=172.28.0.1 --ip-range <nombre red>	Crea una red con las características especificadas
docker rm `docker ps -aq`	Borra TODOS los contenedores

DOCKER

COMANDOS	DESCRIPCIÓN
`docker network rm <red>`	Borra una red
`/var/lib/docker/volumes`	Directorio de almacenamiento de docker
`docker run -it -v /datos –name <nombre contenedor> <imagen> bash`	Crea un contenedor con un volumen especificando el directorio dentro del contenedor
`docker volume ls`	Muestra los volúmenes disponibles
`docker volume inspect <id volumen>`	Vuelva un JSON con los datos del contenedor
`docker volume create <nombre volumen>`	Crea un volumen con un nombre personalizado
`docker run -it –name <nombre contenedor> -v <nombre volumen:/directorio de montaje> <nombre imagen> bash`	Crea un contenedor con un nombre volumen personalizado y un punto de montaje dentro del contenedor.
`docker run -it –name <nombre contenedor> -v <nombre volumen:/directorio>:ro <nombre imagen> bash`	Crea un contenedor y lo asocia a un volumen en modo de solo lectura.
`docker run -i –name <nombre contenedor_2> --volumes-from <nombre contenedor_1>`	Hace que el contenedor 2 utilice los volúmenes del contenedor 1.
`docker volume rm <nombre volumen>`	Borra un volumen
`docker volume prune`	Elimina volúmenes no utilizados
`docker run -it -v <directorio host>:<directorio contenedor> --name <nombre contenedor> <nombre imagen>`	Crea un contenedor asociando un directorio real con un directorio de dentro del contenedor
`docker diff <nombre contenedor>`	Enseña todos los cambios que ha tenido una imagen. La A significa append (añadido), la C changed (cambiado) y la D deleted (borrado).
`docker commit <contenedor inicial> <nombre de imagen nueva>`	Crea una imagen con nombre personalizado a partir de un contenedor.

CREACIÓN DE IMÁGENES

RUN:

El comando RUN ejecuta un comando. Por cada RUN se crea una capa de docker. Permite la concatenación de comandos.

No puede haber ningún comando interactivo. Por eso se pone el -y para Python.

```
FROM ubuntu
RUN apt-get update # Permite concatenación de comandos con &&
RUN apt-get install -y python
```

Para construir la imagen se utiliza el comando: `docker build`

CMD:

El comando CMD también ejecuta un comando. Es el comando por defecto del contenedor. SOLO PUEDE HABER UNO POR CONTENEDOR

```
FROM ubuntu
RUN apt-get update # Permite concatenación de comandos con &&
RUN apt-get install -y python
RUN echo 1.0 >> /etc/version && apt-get install -y git \
                  && apt-get install -y iputils-ping
CMD echo “Welcome to this container”
```

Esto ejecuta el mismo comando con exec. RECOMENDADO.

```
CMD [“echo”, “Welcome to this container”] # Esto ejecuta el mismo comando con exec. RECOMENDADO.
```

ENTRYPOINT:

El comando ENTRYPOINT ejecuta un comando al iniciar un contenedor. Es una buena práctica utilizar esta palabra reservada para ejecutar la aplicación ya que si hacemos algo como ENTRYPOINT [“node”, “arg1”, “arg2”] el contenedor sólo sería capaz de ejecutar comandos de node. SOLO SE PUEDE TENER UNO POR CONTENEDOR.

```
FROM ubuntu
RUN apt-get update # Permite concatenación de comandos con &&
RUN apt-get install -y python
RUN echo 1.0 >> /etc/version && apt-get install -y git \
                  && apt-get install -y iputils-ping

ENTRYPOINT [“df”, “-h”]
```

ENTRYPOINT Y CMD:

El comando ENTRYPOINT se puede mezclar con CMD. La siguiente imagen haría un ping. Pero el target se podría modificar. Para ejecutar este contenedor, se haría lo siguiente:

```
docker run <nombre imagen> <host/ip>
```

```
FROM ubuntu
RUN apt-get update # Permite concatenación de comandos con &&
RUN apt-get install -y python
RUN echo 1.0 >> /etc/version && apt-get install -y git \
                  && apt-get install -y iputils-ping
ENTRYPOINT [“ping”, “-c”, “3”]

CMD [“localhost”]
```

WORKDIR:

El comando WORKDIR establece el directorio en el que trabajan las otras directivas.

```
FROM ubuntu
RUN apt-get update # Permite concatenación de comandos con &&
RUN apt-get install -y python
RUN echo 1.0 >> /etc/version && apt-get install -y git \
                  && apt-get install -y iputils-ping
RUN mkdir /datos
WORKDIR /datos
RUN touch f1.txt
ENTRYPOINT [“bin/bash”]
```

COPY y ADD:

El comando COPY copia un archivo de la máquina real al contenedor.

```
FROM ubuntu
RUN apt-get update # Permite concatenación de comandos con &&
RUN apt-get install -y python
RUN echo 1.0 >> /etc/version && apt-get install -y git \
                  && apt-get install -y iputils-ping
RUN mkdir /datos
WORKDIR /datos
RUN touch f1.txt
COPY index.html . #El punto se refiere al directorio actual de la máquina real
COPY app.log /datos
#Añade el directorio docs de la maquina real al contenedor
ADD docs docs
#Copia todo lo que empieza por f (maquina real) al directorio datos del contendor
ADD f* /datos
# Copia el fichero tar al directorio WORKDIR del contenedor
ADD f.tar .
ENTRYPOINT [“bin/bash”]
```

El comando ADD cuando interactúa con un .tar lo que hace es al pegar ese archivo en el contenedor y lo desempaqueta. EL TAR NO COMPRIME.

ENV:

El comando COPY copia un archivo de la máquina real al contenedor.

```
FROM ubuntu
RUN apt-get update # Permite concatenación de comandos con &&
RUN apt-get install -y python
RUN echo 1.0 >> /etc/version && apt-get install -y git \
                  && apt-get install -y iputils-ping
RUN mkdir /datos
WORKDIR /datos
RUN touch f1.txt
COPY index.html . #El punto se refiere al directorio actual de la máquina real
COPY app.log /datos
#Añade el directorio docs de la maquina real al contenedor
ADD docs docs
#Copia todo lo que empieza por f (maquina real) al directorio datos del contendor
ADD f* /datos
# Copia el fichero tar al directorio WORKDIR del contenedor
ADD f.tar .
ENTRYPOINT [“bin/bash”]
```

El comando ADD cuando interactúa con un .tar lo que hace es al pegar ese archivo en el contenedor y lo desempaqueta. EL TAR NO COMPRIME.