

DOCKER

Docker ejecuta los procesos de forma aislada, pero en tu propia maquina, de forma "mágica" con los Cgroups y namespaces

Dockerfiles y docker-compose



FROM	Crear la imagen desde otra imagen oficial	
RUN	Va seguido de los comandos a ejecutar dentro del contenedor	
CMD	Comando que se ejecutará al hacer docker run	Si no se indica, se ejecuta el de la imagen del FROM. Se ejecuta como argumento de lo que se indica en el ENTRYPOINT
LABEL	Etiquetar la imagen	
EXPOSE	Exponer puertos	
ADD	Copia en el contenedor, permite poner URL's	
COPY	Copia en el contenedor, solo rutas locales	
ENV	Variables de entorno a incluir en el contenedor	
ENTRYPOINT	Se ejecuta siempre que se arranca un contenedor, lo que se indique aquí	default: "/bin/sh", "c" Si no se indica, se ejecuta el de la imagen del FROM
VOLUME	Ruta en la que se guardan los datos	

Construir imagen (Repositorio de imágenes <u>hub.docker.com</u>)



Docker cache

La 1º vez que se construye la imagen tarda un poco en ejecutar todo lo necesario.

Si construimos de la imagen por 2º vez solo se ejecutarán las sentencias que no estén cacheadas, es decir, que hayan sido cambiadas.

Si hacemos justo después del FROM un COPY de nuestro código fuente, cada vez que este cambie se ejecutará el COPY y todas las sentencias que se encuentren por debajo de nuestro COPY.

Por esto el orden en que escribimos nuestro dockerfile es muy importante. Deben ir arriba las sentencias que menos cambian.

Ya que cada sentencia FROM, RUN, COPY ... crea una nueva capa que se cachea, desde que una de esta capas se modifique/cambie, se tienen que volver a crear la capa y cada una de las que están por debajo.

1 sentencia - 1 capa

Esto solo crea una capa de cache y hace que ocupe menos espacio. Si tenemos dos RUN como arriba nos crearía 2 capas.

Imágenes pequeñas

Las imágenes deben ocupar lo menos posible.

Una imagen pesara más según las sentencias que tenga que ejecutar. Por ejemplo es mejor concatenar las sentencias del RUM con &&

RUN apt-get update
RUN apt-get install -y \
bzr \
cvs \
git \
mercurial \
subversion

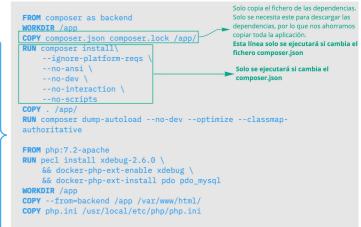
RUN apt-get update && apt-get install -y
bzr \
cvs \
git \
mercurial \
cubversion

Multistage build

Es un *dockerfile* con varias sentencias **FROM**. Nos permiten separar la imagen de *build*, donde generamos nuestra aplicación, de la imagen que finalmente ejecutaremos, haciendo que ocupe menos y sea más segura.



Good practice



docker-compose

Es un fichero de configuración .yml en el que se especifica como se levantan los contenedores evitando hacerlo de forma manual (docker run ...).

	build	Construye las imágenes definidas en el fichero docker-compose
	up	Levanta (y construye, si no lo está) los contenedores especificados en el fichero
	up -d	Igual modo detach
	upbuild	Fuerza a construir la imagen
docker-compose	ps	Lista los contenedores levantados
docker-compose	logs -f nameService	Muestra los logs del servicio indicando el nombre del servicio
	top	Muestra los procesos en ejecución
	stop	Para los contenedores especificados en el fichero
	down	Elimina los contenedores parados

Volumen

Sirve para persistir los datos de por ejemplo una base de datos, de forma que si el contenedor se para o se elimina no perdemos la información ya que la tendremos en local.

También vale para publicar aplicaciones de lenguajes interpretados, ya que simplemente tenemos que tener en el volumen los ficheros generados para desplegar la aplicación.

docker-compose.yml

