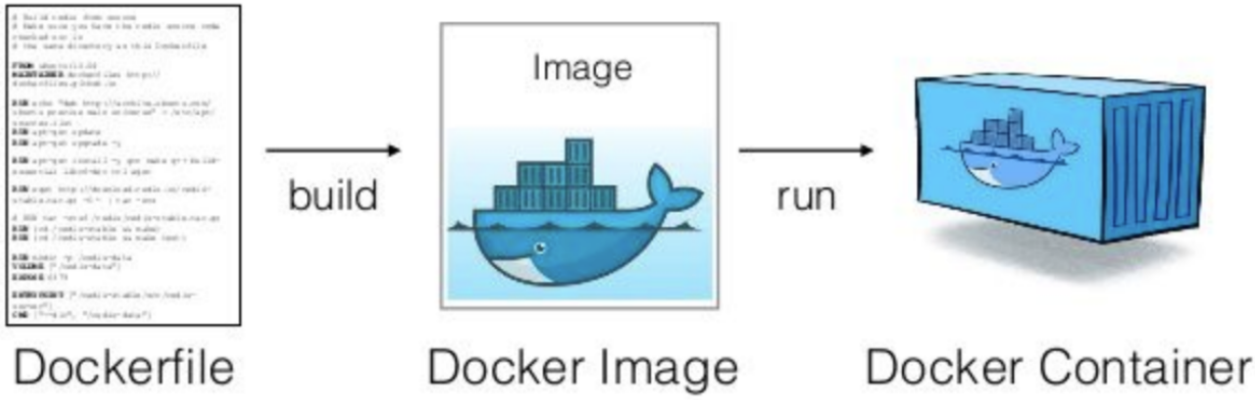
DOCKERFILE



DOCKERFILE

DockerFile es un archivo de texto en el cual tenemos disponible un conjunto de comandos e instrucciones que se ejecutarán de forma secuencial en una imagen base con el objetivo de crear una nueva imagen de una sola aplicación con un propósito definido.

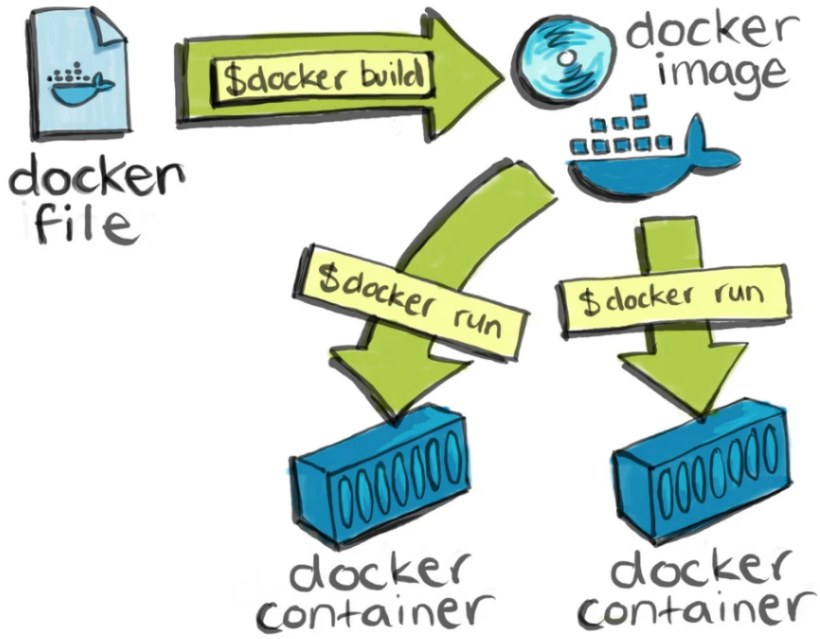


Ilustración 1. Flujo Docker file

1. ESTRUCTURA DOCKERFILE

Para poder desarrollar un *Dockerfile*, debemos seguir una estructura definida.

- El fichero debe llamarse Dockerfile, respetando mayúsculas y minúsculas
- Las líneas de comentario van precedidas del símbolo #
- El flujo típico de un fichero Dockerfile es
    - Selección de imagen base
    - Descarga e instalación de dependencias
    - Comandos a ejecutar al arrancar el contenedor
- Instrucciones:
    - FROM <imagen> [AS <fase>]
      Requerido→ especifica la imagen a utilizar como base. Se puede especificar un nombre de fase para los casos en los que se utilizan construcciones multi-fase.
    - RUN <comando>
      Ejecuta el comando en el momento de la creación de la imagen
    - CMD <comando>
      Requerido→ ejecuta el comando en el momento de ejecución del contenedor. Parámetros:
        - <comando>
          Es el comando a ejecutar en el momento de iniciar el contenedor. Es un objeto de tipo *array*, que contiene un primer elemento que se identificará como el comando a ejecutar, y todos los sucesivos elementos serán los parámetros a pasar a dicho comando. Por ejemplo, si en línea de comando quisiésemos ejecutar npm start para arrancar Node, <comando> tendría el valor ["npm","start"].
    - WORKDIR <ruta>
      Especifica la carpeta de trabajo dentro del contenedor. A partir del momento en que se ejecuta esta instrucción, todos los comandos siguientes se ejecutarán de forma relativa a <ruta>. Si <ruta> no existe, se creará automáticamente. Esta es una buena práctica cara a evitar conflictos de ficheros con la imagen base utilizada.
    - COPY [--from=<fase>] <origen> <destino>
      Copia los ficheros de <origen> en <destino>. Los parámetros son relativos al contexto de construcción (la carpeta donde se encuentra Dockerfile):
        - <origen> se encuentra en el sistema de archivos fuera del contenedor
        - <destino> se encuentra en el sistema de archivos dentro del contenedor
        - --from=<fase> cuando se especifica el modificador from, COPY busca los ficheros en la imagen referenciada por <fase> en vez del sistema de archivos del host.

- EXPOSE <puerto>
  Especifica el puerto (o puertos, se pueden indicar varios valores separados por un espacio) que el contenedor habilitará para conexión
- VOLUME <carpeta>
  Indica una carpeta cuyos datos persistirán en el sistema de archivos del host
- HEALTHCHECK [parametros] <comando>
  Indica el comando a utilizar para el healthcheck del contenedor (estado de salud). Donde:
  - <comando> es el comando a ejecutar
  - parametros puede contener los siguientes valores:
    - --interval <duration> intervalo entre comprobaciones (por defecto 30 segundos)
    - --timeout <duration> intervalo para considerar la llamada como fallida (por defecto 30 segundos)
    - --start-period <duration> intervalo para esperar a la primera comprobación (por defecto 0 segundos)
    - --retries <N> número de reintentos en caso de error (por defecto 3)

En el siguiente ejemplo vamos a desarrollar un dockerfile el cual parte de una base de la última versión de Debian, actualizamos el sistema, los paquetes y git. Seteamos el nombre *Florida* como variable *minombre* y mostramos el mensaje "Hola Florida" (sustituyendo el valor de la variable) al ejecutar *docker run*.

```
FROM debian:latest
RUN apt-get update && apt-get upgrade -y && apt-get install -y git
CMD ping google.es > archivo.txt
ENV minombre Florida
ENTRYPOINT echo "Hola $minombre"
COPY archivo.txt /home/$USER/archiv.txt
```

Ilustración 2. Ejemplo Dockerfile

Sin mucha importancia		
MAINTAINER	Podemos añadir nuestro nombre y correo. Los datos de quien ha creado el Dockerfile y la imagen.	Ejemplo: **MAINTAINER** redxlus luis@luisiblogdeinformatica.com
Obligatorio		
FROM	Es **obligatorio**. Al principio de cada Dockerfile. Indica cual es la imagen base.	Ejemplo: **FROM** debian:latest
De ejecución		
RUN	Ejecuta comandos en una nueva capa y crea un nuevo contenedor intermedio. Normalmente es usado para instalar nuevos paquetes.	Ejemplo: **RUN** apt-get update
CMD	El comando que se especifique a continuación, se ejecutara cuando haga `docker run` si no se especificar otro comando.	Ejemplo: **CMD** apt-get update
ENTRYPOINT	Igual que **CMD** pero no es ignorado si indicamos otro comando.	Ejemplo: **ENTRYPOINT** apt-get update
Datos		
COPY	Copia archivos o directorios desde la máquina local (donde esta instalado docker) al contenedor Docker.	Ejemplo: **COPY** texto.txt /directorio/ejemplo
ADD	Similar a **COPY**. Añade más funcionalidades. Permite 2 fuentes además de la posibilidad de que sean URLs. Se pueden usar escapes. Y si la usas con un archivo comprimido(gzip, bzip2, xz) te lo descomprime.	Ejemplo: **ADD** https://a.luisiblogdeinformatica.com/nc1 /
Otros		
EXPOSE	Indica (no expone) el puerto del contenedor. Útil para que los puertos no se expongan aleatoriamente. Es complementario a `docker run -p`. Si es `-p` (minúscula) tienes que indicar puerto entrada:puerto salida. Si es `-P` (mayúscula) el de salida es fijo y el de entrada aleatorio. También, si solo usas **EXPOSE** y al hacer `run` no especificas ni `-p` ni `-P` lo que va a pasar es que esos puertos solo sean accesibles desde otros contenedores, no desde el exterior.	Ejemplo: **EXPOSE** 8080

	Se puede especificar también si es TCP o UDP. Por defecto es TCP.	
VOLUME	Indica el destino de un volumen dentro del contenedor. Puede ser uno o varios. Igual que si especificáramos `docker run -v`.	Ejemplo: **VOLUME** /unadireccion
WORKDIR	Especifica el directorio desde el que se van a ejecutar las ordenes de: • **RUN** • **CMD** • **ENTRYPOINT** • **COPY** • **ADD** Sustituye al comando `cd`. Y si no existe el directorio, lo crea.	Ejemplo: **WORKDIR** /midirectorio
ENV	Variables del sistema o entorno personales. Podemos llamarlas después dentro del contenedor.	Ejemplo: **ENV** nombre luis Luego dentro del contenedor: `echo $nombre`
USER	Indica el nombre o UID que va a ejecutar las acciones del Dockerfile.	Ejemplo: **USER** luisdieguez