Practica 3 - Dockerfile

Markel Orallo

2021

# Multi-Stage Builds

Creamos la imagen a partir del Dockerfile:

FROM alpine:3.5 AS build RUN apk update && \  
apk add --update alpine-sdk RUN mkdir /app

WORKDIR /app  
ADD hello.c /app  
RUN mkdir bin  
RUN gcc -Wall hello.c -o bin/hello  
# Lightweight image returned as final product

FROM alpine:3.5

COPY --from=build /app/bin/hello /app/hello

CMD /app/hello

**Comprobar la imagen final mediante docker image ls (comprobar su tamaño) y describir el proceso. ¿Cuantos Stages tiene?**

# Escribir y generar el Dockerfile

**Crea una carpeta “myImage” y crea un fichero con el nombre Dockerfile dentro de esta carpeta.**

**Contenido del Dockerfile:**

*FROM centos:7*

*RUN yum update –y*

*RUN yum install -y wget*

**Esto sirve como receta para una imagen basada en centos:7, que tiene todos sus paquetes por defecto actualizados y wget instalado**

**Genera la imagen con el nombre myimage**

**Compruebe que su nueva imagen existe (docker image ls)**

**Ejecutar la imagen en un contenedor y haga wget en ese contenedor.**

# Build Caché

**Abra su Dockerfile y agregue otro paso RUN al final para instalar vim.**

**RUN yum install vim. Construya la imagen de nuevo como en el ejemplo anterior; ¿para qué pasos se utiliza la caché?**

**Construya la imagen de nuevo; ¿qué pasos utiliza esta vez la caché?**

**Cambia el orden de los dos comandos RUN para instalar wget y vim en el Dockerfile, y construye una vez más la imagen. ¿Qué pasos se almacenan en la memoria caché esta vez?**

# History Command

**Ejecuta docker image history a la imagen creada en el paso anterior y observa la información devuelta**

**Cambia las instrucciones de instalación de wget y vim para que se instalen con un único comando: RUN yum install -y wget vim**

**Genera la imagen de nuevo y llama al comando history. ¿Qué ha cambiado?**

# Configurar comandos por defecto

**Añade la siguiente línea al Dockerfile de la práctica anterior: CMD ["ping", "127.0.0.1", "-c", "5"]. Genera de nuevo la imagen y pon en marcha el contenedor. ¿Que obtienes?**

**Ahora ejecuta el siguiente comando al poner en marcha el contenedor: $docker container run myimage echo "hello world“ .¿Qué ha pasado esta vez?**

**Cambia la última línea del Dockerfile por: ENTRYPOINT ["ping"].Genera y pon en marcha la nueva imagen en un nuevo contenedor. ¿Qué pasa?**

**Prueba a poner en marcha mendiante este comando:**

*$docker container run myimage 127.0.0.1*

**¿Puedes describir las diferencias entre ENTRYPOINT y CMD? Cambia la última línea del Dockerfile por:**

**ENTRYPOINT ["ping", "-c", "3"] CMD ["127.0.0.1"]**

**Genera la imagen y pon en marcha el contenedor (sin pasarle comandos) ¿Te ha dado error? ¿Cómo interpreta el valor de CMD?**

**Pon en marcha el contenedor:**

**$docker container run myimage 8.8.8.8 ¿Qué ha pasado esta vez?**