JÓVENES BICENTENARIO 3.0 – CERTUS

Docente: EDWIN MARAVI PÉREZ

POR: CARRION HUACANI GEAN CARLO

ACTIVIDAD ASINCRÓNICA N° 04

DOCKERFILE

1. Mejores prácticas archivo Dockerfile.

Validar la sintaxis: Un paso muy importante cuando escribimos líneas de código es validar la sintaxis. Entre los desarrolladores es muy común el uso de Linters, que permiten validar la sintaxis del código previo a la construcción y Docker no es la excepción. Encontrarás varias opciones de código abierto para ejecutar en la línea de comandos o plugins para hacer estas validaciones desde tu editor de texto favorito.

Optimizar el uso de las instrucciones: Con el comando "docker run" construimos una imagen desde un Dockerfile. El objetivo es que estas imágenes sean lo más livianas posible. Además de FROM, las instrucciones ADD, COPY Y RUN agregan una nueva capa a la imagen resultante. Lo ideal es optimizar el uso de estas instrucciones para mantener la imagen lo más ligera posible. En la imagen 2 te muestro unos ejemplos óptimos y no óptimos de su implementación.

Construir las imágenes en múltiples etapas: Docker tiene una funcionalidad llamada multi-stage, lo cual también es útil para reducir el tamaño de las imágenes, ya que permite utilizar imágenes diferentes en cada etapa. Te recomiendo entonces tener una etapa de construcción con una imagen que contenga el SDK completo del lenguaje que estás utilizando para compilar tu aplicación, y en la etapa final utilizar una imagen que solo contenga el runtime para ejecutar la aplicación. Esto definitivamente hará que tu aplicación sea más ligera y fácil de portar.

Excluir archivos irrelevantes: Para excluir archivos que no sean relevantes para la compilación o ejecución de tu aplicación puedes agregar un archivo adicional: .dockerignore. En este archivo puedes incluir patrones de exclusión o rutas específicas de archivos, similar al archivo .gitignore.

Esta práctica te permitirá excluir archivos cuando ejecutas la instrucción COPY y ADD.

Mantener las cosas simples: Para reducir la complejidad, dependencias y tiempos de compilación evita instalar paquetes adicionales o innecesarios. Incluso si solo utilizas un paquete para realizar cierta tarea en la etapa de construcción de imagen y luego ya no lo necesitarás, te recomiendo que lo desinstales.

2. Detallar instrucciones que pueden ser utilizadas en un archivo Dockerfile.

Instrucciones que podemos usar para crear un Dockerfile:

- FROM: Especifica la base para la imagen. Un archivo Dockerfile solo puede empezar por dos tipos de instrucción, una es ARG y la otra es FROM. Todas las imágenes Docker necesitan de otra imagen que les sirva de base. FROM sirve para indicar que imagen queremos usar.
- ENV: Se usa para declarar variables de entorno que van a ser visibles para las siguientes instrucciones y para el contenedor resultante.
- ARG: Permite definir una variable usable en el resto del Dockerfile con la sintaxis ${NOMBRE_DEL_ARG}
- RUN: Ejecuta el comando especificado. Se usa para instalar paquetes en el contenedor.
- COPY: indica a Docker que coja los archivos de nuestro equipo y los añada a la imagen con la que estamos trabajando. COPY crea un directorio en caso de que no exista
- ADD: hace exactamente lo mismo que COPY, pero además añade dos funcionalidades. La primera es que permite indicar contenidos vía URL, y la segunda es la extracción de archivos TAR.
- WORKDIR: Para dar claridad a tus archivos Dockerfile, es preferible usar WORKDIR a otra instrucción para cambiar el directorio actual de la imagen. Si el directorio especificado no existe, se creará automáticamente. Además, podemos usar este comando varias veces en el mismo archivo, si tenemos que trabajar con varios directorios a la vez.
- ENTRYPOINT: Docker tiene un Entrypoint por defecto, /bin/sh -c, que se ejecuta cuando el contenedor está listo. Este comando permite sobreescribirlo.
- CMD: Especifica el comando y argumentos que se van a pasar al contenedor. Se ejecutarán junto con lo indicado en el Entrypoint.
- EXPOSE: La instrucción EXPOSE muestra que puerto queremos publicar para acceder al contenedor y opcionalmente el protocolo. Es muy importante tener claro que esta instrucción NO publica directamente el puerto. Su función es comunicar a la persona que va a usar la imagen qué puerto se deben usar.
- LABEL: Nos permite aportar meta-datos a la imagen.

- VOLUME: Crea un punto en el sistema de archivos de la imagen que se va a usar para montar un volumen externo. Eso significa que la información que guardemos en el directorio indicado va a perdurar en caso de que el contenedor deje de ejecutarse
- USER: Establece el usuario que se va a usar cuando se ejecute cualquier operación posterior con RUN, CMD y ENTRYPOINT.
- SHELL: Permite especificar el uso de otra terminal, como zsh, csh, tcsh, powershell, u otras.
- STOPSIGNAL: Indica una señal que va a finalizar el contenedor.
- HEALTHCHECK: Indica a Docker una manera de testear el contenedor para verificar que sigue funcionando correctamente.
- ONBUILD: Cuando la imagen donde se encuentra se use como base de otra imagen, va a actuar de trigger y va a ejecutar el comando que le indiquemos.