Crear un Dockerfile en Play with Docker

En este ejercicio, aprenderemos a crear y ejecutar un archivo **Dockerfile** básico en **Play with Docker** para construir un contenedor de Ubuntu con la herramienta **htop** instalada. **htop** es una herramienta interactiva que nos permite monitorizar el sistema en tiempo real.

Pasos a seguir:

1. **Accede a Play with Docker:**
    - Dirígete a Play with Docker y abre una nueva sesión.
2. **Crear el Dockerfile:**
    - En la terminal, utiliza un editor de texto como nano o vi para crear un archivo llamado Dockerfile.
    - Ingresa el siguiente código en el Dockerfile:

```
FROM ubuntu:latest

RUN apt-get -y update

RUN apt-get -y upgrade

RUN apt-get -y install htop

CMD ["bash"]
```

3. Este código:
    - Utiliza la última versión de Ubuntu como imagen base.
    - Actualiza y mejora los paquetes de Ubuntu.
    - Instala **htop**.
    - Mantiene el contenedor activo ejecutando bash.
4. **Construir la imagen:**
    - En la terminal, navega al directorio donde se encuentra el Dockerfile y ejecuta el siguiente comando para construir la imagen:

```
docker build -t ubuntu-htop .
```

    - Aquí, -t ubuntu-htop etiqueta la imagen como ubuntu-htop.
5. **Ejecutar el contenedor:**
    - Una vez creada la imagen, puedes ejecutar el contenedor con:

```
docker run -it ubuntu-htop
```

    - Este comando lanza el contenedor en modo interactivo.

6. **Iniciar htop:**
    - Dentro del contenedor, escribe htop para iniciar la herramienta de monitorización interactiva y visualizar los procesos del sistema.
7. **Cerrar el contenedor:**
    - Para salir de htop, presiona q.
    - Luego, puedes salir del contenedor escribiendo exit.

Crear un Entorno Web Completo en Play with Docker

Esta práctica consiste en crear tres contenedores personalizados utilizando Dockerfile para configurar servidores de Nginx, Apache Tomcat y MySQL.

Pasos Previos

1. **Accede a Play with Docker:**
    - Ve a Play with Docker e inicia sesión.
    - Abre una nueva sesión y prepara el entorno.
2. **Descarga de recursos:**
    - Para facilitar la configuración de estos contenedores, descarga el archivo de recursos desde GitHub:

wget https://github.com/shokone/Vagrant-Docker/raw/master/resources.tar.gz

    - Descomprime el archivo descargado:

tar -xzf resources.tar.gz

    - Ahora deberías tener los archivos necesarios para cada contenedor.

1. Configuración del Contenedor Nginx

1. **Crea el Dockerfile para Nginx:**
    - Usa un editor como nano para crear un Dockerfile en el directorio de Nginx:

nano Dockerfile

    - Ingresa el siguiente contenido:

```
FROM nginx:1.17

LABEL MAINTAINER "imartinez@example.com"

RUN apt-get update -y \
  && apt-get upgrade -y \
  && apt-get install curl -y \
  && rm /etc/nginx/conf.d/default.conf
```

COPY html /usr/share/nginx/html

COPY conf /etc/nginx/conf.d

ENV APP_PORT=80

EXPOSE $APP_PORT

HEALTHCHECK CMD curl -s --fail http://localhost:$APP_PORT || exit 1

2. **Construye la imagen de Nginx:**

docker build -t custom-nginx .

3. **Ejecuta el contenedor de Nginx:**

docker run -d -p 80:80 custom-nginx

2. Configuración del Contenedor Tomcat

1. **Crea el Dockerfile para Tomcat:**
    - Crea un Dockerfile en el directorio de Tomcat:

nano Dockerfile

- Ingresa el siguiente contenido:

FROM tomcat:8.5.50-jdk8

LABEL MAINTAINER "imartinez@example.com"

ENV JDBC_DDBB=example_db \

JDBC_URL=jdbc:mysql://10.100.1.200:3306/example_db?connectTimeout=0&socketTimeout=0&autoReconnect=true \

JDBC_USER=db_user \

JDBC_PASS=db_password \

APP_PORT=8080

COPY code/webapps /usr/local/tomcat/webapps

EXPOSE $APP_PORT

HEALTHCHECK CMD curl -s --fail http://localhost:$APP_PORT || exit 1

CMD ["catalina.sh", "run"]

2. **Construye la imagen de Tomcat:**

docker build -t custom-tomcat .

3. **Ejecuta el contenedor de Tomcat:**

docker run -d -p 8080:8080 custom-tomcat

3. Configuración del Contenedor MySQL

1. **Crea el Dockerfile para MySQL:**
    - Crea un Dockerfile en el directorio de MySQL:

nano Dockerfile

    - Ingresa el siguiente contenido:

```
FROM mysql:5.6

ENV MYSQL_ROOT_PASSWORD=mysqlpassword \

  MYSQL_DATABASE=example_db \

  MYSQL_USER=db_user \

  MYSQL_PASSWORD=db_password \

  APP_PORT=3306

COPY data/mysql-init.sql /docker-entrypoint-initdb.d/init.sql

EXPOSE $APP_PORT

CMD ["mysqld"]
```

2. **Construye la imagen de MySQL:**

```
docker build -t custom-mysql .
```

3. **Ejecuta el contenedor de MySQL:**

```
docker run -d -p 3306:3306 custom-mysql
```

Verificación del Entorno

1. **Nginx** estará disponible en el puerto 80.
2. **Tomcat** estará disponible en el puerto 8080.
3. **MySQL** estará disponible en el puerto 3306.

Para verificar que los contenedores están activos y funcionando correctamente, puedes utilizar el comando:

```
docker ps
```