Contenedores y orquestación de contenedores

**Notas previas**: Este ejercicio se ha llevado a cabo sobre una distribución Ubuntu en una Raspberry Pi 4, pero en principio debería de ser compatible con cualquier otra distribución de Linux en la que se lleve a cabo.

Se ha optado por la utilización de Podman como solución de contenedores, por la ventaja que supone prescindir de un daemon centralizado y de los derechos de root para su funcionamiento, y por su casi total compatibilidad con Docker.

Debemos tener por lo tanto instalados podman y podman-compose en nuestro sistema:

sudo apt-get install podman

sudo apt-get install podman-compose

# Parte 1 - Identificación de componentes de la aplicación

La idea para este ejercicio es la de lograr la “contenerización” del software entregado durante la realización de las últimas prácticas correspondientes al módulo de BACKEND de este curso (principalmente lo contenido en las microactividades 9 y 10 del módulo 3)

La solución a implementar va a consistir en:

- Un contenedor que contendrá una base de datos relacional (mariadb)

- Un contenedor que contendrá una base de datos documental (mongodb)

- Un contenedor que contendrá un servidor de aplicaciones y una aplicación REST (tomcat + aplicación java/spring-boot)

# Parte 2 – Configuración de imágenes y generación de imagen propia

Para las dos bases de datos emplearemos las imágenes disponibles en Docker Hub, concretamente:

- MariaDb 10.11

- MongoDb 4.4.18

La imagen para la aplicación web será una imagen de generación propia, generada con el fichero Dockerfile incluido en la entrega, tomando como base de partida la imagen de Tomcat 10.1 disponible en Docker Hub.

FROM docker.io/tomcat:10.1

RUN rm -rf /usr/local/tomcat/webapps

RUN mv /usr/local/tomcat/webapps.dist /usr/local/tomcat/webapps

COPY repositorios-app.war /usr/local/tomcat/webapps/repositorios-app.war

COPY tomcat-users.xml /usr/local/tomcat/conf/tomcat-users.xml

COPY context.xml /usr/local/tomcat/webapps/manager/META-INF/context.xml

EXPOSE 8080

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

Las instrucciones RUN y COPY tienen como objetivo configurar en el directorio correcto (webapps) todos los proyectos por defecto que trae la imagen de tomcat (incluyendo el de la consola de administración) así como habilitar el uso de la consola de administración y un usuario para hacer uso de ella (admin / admin). También copiamos el WAR correspondiente a la aplicación REST, para que se despliegue y arranque al iniciar Tomcat.

La instrucción EXPOSE indica a docker (podman) que el contenedor debe “escuchar” en el puerto 8080.

Y con la instrucción ENTRYPOINT estamos configurando el ejecutable que se debe lanzar una vez se haya inicializado el contendor. En este caso, el script de arranque de Tomcat.

Ejecutando el comando podman build -t repositorios-app -f Dockerfile generamos la imagen a la que hemos dado el nombre de “repositorios-app” y como no hemos indicado versión, por defecto será “latest”.

# Parte 3 – Orquestación de imágenes, Gestión de la persistencia y de las redes

Llevamos a cabo todo esto por medio del siguiente fichero docker-compose.yml:

version: "3.0"

services:

repositorios:

image: repositorios-app

hostname: repositorios-app

ports:

- "8081:8080"

depends\_on:

- mariadb

- mongo

networks:

- repositorios\_network

mariadb:

image: "docker.io/mariadb:10.11"

hostname: mariadb

ports:

- 3307:3306

volumes:

- ./init\_db.sql:/docker-entrypoint-initdb.d/init\_db.sql

- mariadbdata:/var/lib/mysql

environment:

- MARIADB\_DATABASE=test

- MARIADB\_USER=user

- MARIADB\_PASSWORD=password

- MARIADB\_ROOT\_PASSWORD=root

networks:

- repositorios\_network

mongo:

image: "docker.io/mongo:4.4.18"

hostname: mongo

ports:

- 27017:27017

networks:

- repositorios\_network

volumes:

- mongodbdata:/data/db

networks:

repositorios\_network:

volumes:

mongodbdata:

mariadbdata:

Estamos configurando tres servicios base, con sus imágenes. Destacar que al referir a las imágenes de Docker Hub, como estamos utilizando Podman debemos añadir “docker.io/” antes del nombre de la imagen para que Podman sea capaz de indetificarla y descargarla correctamente.

Estamos configurando la dependencia entre nuestros servicios con “depends-on”. En concreto, la imagen con el servidor de aplicaciones y la aplicación REST depende de ambas bases de datos, porque es de donde consultan la información. Por lo tanto, los dos servicios mariadb y mongo son necesarios para un correcto arranque y funcionamiento del servicio repositorios.

Estamos configurando la persistencia de los dos servicios con las bases de datos. Aunque la aplicación REST actualmente solo realiza consultas contra las bases de datos, haciendo esto conseguimos que se conserve la información de las dos bases de datos en sucesivas ejecuciones, lo cual sería necesario en caso de que implementásemos algún servicio REST para insertar o editar sus datos, por ejemplo. Adicionalmente, en el caso concreto de MariaDB, estamos proporcionando un script sql de provisionamiento de la base de datos, que se encarga de crear y poblar las tablas necesarias la primera vez que se arranque esta imagen. También estamos configurando las propiedades de entorno de MariaDB que configuran la base de datos en particular que vamos a usar (test) así como los usuarios / contraseñas de la misma.

Por último, estamos configurando una red “repositorios-network” y asignándola a los servicios, de manera que todos los servicios pertenecen a la misma red, y exponiendo para cada servicio un puerto TCP para poder ser accedido desde fuera de la red de docker (podman). Con la propiedad “hostname” asignamos un nombre en particular para cada uno de los host, que nos sirva para identificarlos unívocamente. Esto por ejemplo es necesario para que, desde el código de la aplicación REST, podamos referenciar correctamente a los servicios de ambas bases de datos y se pueda conectar a ellas sin problemas.

Con la instrucción podman-compose up –d conseguimos que se descarguen e instalen las imágenes configuradas, si no están ya en el sistema, y se arranquen los tres contenedores, con la configuración de volúmenes, redes, etc indicada en el archivo docker-compose.yml

Una vez hecho, si todo ha ido bien, deberían de estar disponibles y acessibles desde la red de docker los servicios rest y las bases de datos, pero también desde fuera de la red en los puertos expuestos según el docker-compose. Por ejemplo en:

http://<host-docker>:8081/repositorios-app/relational/getText/1

http://<host-docker>:8081/repositorios-app/relational/printString/1

http://<host-docker>:8081/repositorios-app/documents/getText/1

http://<host-docker>:8081/repositorios-app/documents/printString/1

http://<host-docker>:8081/repositorios-app/actuator

http:// <host-docker>:8081/repositorios-app/actuator/prometheus

Y también se debería poder establecer una conexión con MariaDb y MongoDb a través de los puertos 3017 y 27017 respectivamente, utilizando para ello cualquier software de gestión de base de datos compatible con ellas.

NOTA: <host-docker> sería la IP o nombre del servidor donde corren nuestros contenedores.