Aplicaciones de múltiples contenedores

Hasta este punto, has estado trabajando con aplicaciones de contenedor único. Pero ahora agregarás MySQL a la pila de aplicaciones. A menudo surge la siguiente pregunta: "¿Dónde se ejecutará MySQL? ¿Lo instalo en el mismo contenedor o ejecútelo por separado?" En general,cada contenedor debe hacer una cosa y hacerla bien.

Recuerda que los contenedores, de forma predeterminada, se ejecutan de forma aislada y no saben nada sobre otros procesos o contenedores en la misma máquina. Entonces, ¿cómo se permite que un contenedor hable con otro? La respuesta está en la creación de redes. Si colocas los dos contenedores en la misma red, pueden comunicarse entre sí.

En los siguientes pasos, primero creará la red y luego adjuntará el contenedor MySQL.

1. Crea la red.

   $ docker network create todo-app

2. Inicia un contenedor MySQL y conéctelo a la red. También definirás algunas variables deentorno que la base de datos utilizará para inicializar la base de datos.

```
$ docker run -d \
  --network todo-app --network-alias mysql \
  -v todo-mysql-data:/var/lib/mysql \
  -e MYSQL_ROOT_PASSWORD=secret \
  -e MYSQL_DATABASE=todos \
  mysql:8.0
```

Notarás un volumen nombrado todo-mysql-data en el comando anterior que está montado en /var/lib/mysql, que es donde MySQL almacena sus datos. Sin embargo, nunca ejecutaste un comando docker volume créate. Docker reconoce que deseas utilizar un volumen con nombre y crea uno automáticamente.

3. Para confirmar que tienes la base de datos en funcionamiento, conéctate a la base de datos.

   $ docker exec -it <mysql-container-id> mysql -u root -p

Cuando aparezca la solicitud de contraseña, escribe secret. En el shell MySQL, enumera las bases de datos y verifica que ves la base de datos.

```
mysql> SHOW DATABASES;
```

4. Salga del shell MySQL para regresar al shell de su máquina.

   mysql> exit

Ahora tienes una base de datos todos y está lista para usar.

Ahora que sabes que MySQL está funcionando, puedes usarlo. Pero, ¿cómo se usa? Si ejecutas otro contenedor en la misma red, ¿cómo encuentra el contenedor? Recuerda que cada contenedor tiene su propia dirección IP.

Para responder las preguntas anteriores y comprender mejor las redes de contenedores, utilizarás nicolaka/netshoot. Un contenedor, que viene con muchas herramientas que son útiles para solucionar problemas o depurar problemas de red.

1. Inicia un nuevo contenedor usando la imagen nicolaka/netshoot. Asegúrate de conectarlo a la misma red.

   $ docker run -it --network todo-app nicolaka/netshoot

2. Dentro del contenedor, usarás el comando dig, que es una herramienta DNS útil. Vas a buscar la dirección IP para el nombre de host mysql.

   $ dig mysql

En la "SECCIÓN DE RESPUESTA", verá un registro mysql que se resuelve en 172.23.0.2 (lo más probable es que su dirección IP tenga un valor diferente). Si bien mysql normalmente no es un nombre de host válido, Docker pudo resolverlo en la dirección IP del contenedor que tenía ese alias de red. Recuerda que usaste el --network-alias anterior.

Lo que esto significa es que tu aplicación simplemente necesita conectarse a un host llamado mysql y se comunicará con la base de datos.

La aplicación Todo admite la configuración de algunas variables de entorno para especificar la configuración de conexión MySQL. Ellos son:

- MYSQL_HOST- el nombre de host del servidor MySQL en ejecución
- MYSQL_USER- el nombre de usuario que se utilizará para la conexión
- MYSQL_PASSWORD- la contraseña a utilizar para la conexión
- MYSQL_DB- la base de datos a utilizar una vez conectado

Ahora puedes iniciar tu contenedor listo para desarrollo. Clona el repositorio de git: git clone https://github.com/docker/gettingstarted-app.git y accede a la carpeta getting-started-app generada

1. Especifica cada una de las variables de entorno anteriores, así como también conecta el contenedor a la red de la aplicación. Asegúrate de estar en el directorio getting-started-app cuando ejecute este comando.

```
$ docker run -dp 127.0.0.1:3000:3000 \
  -w /app -v "$(pwd):/app" \
  --network todo-app \
  -e MYSQL_HOST=mysql \
  -e MYSQL_USER=root \
  -e MYSQL_PASSWORD=secret \
  -e MYSQL_DB=todos \
  node:18-alpine \
  sh -c "yarn install && yarn run dev"
```

2. Si observa los registros del contenedor (docker logs -f <container-id>), debería ver un mensaje similar al siguiente, que indica que está usando la base de datos mysql.

   ```
   $ nodemon src/index.js
   [nodemon] 2.0.20
   [nodemon] to restart at any time, enter `rs`
   [nodemon] watching dir(s): *.*
   [nodemon] starting `node src/index.js`
   Connected to mysql db at host mysql
   Listening on port 3000
   ```

3. Abra la aplicación en su navegador y agregue algunos elementos a su lista de tareas pendientes. Estos se estarán agregando a la BBDD!!

Usar Docker Compose

La gran ventaja de usar Compose es que puedes definir tu pila de aplicaciones en un archivo, mantenerla en la raíz del repositorio de tu proyecto (ahora tiene control de versión) y permitir fácilmente que otra persona contribuya a tu proyecto. Alguien sólo necesitaría clonar tu repositorio e iniciar la aplicación usando Compose. De hecho, es posible que veas bastantes proyectos en GitHub/GitLab haciendo exactamente esto.

En el getting-started-app, crea un archivo llamado compose.yaml.

├── getting-started-app/

│ ├── compose.yaml

│ ├── node_modules/

│ ├── package.json

│ ├── spec/

│ ├── src/

│ └── yarn.lock

Pega el contenido del archivo compose.yaml que hay en el repositorio de GitHub. Ahora que tienes tu compose.yaml, puedes iniciar su solicitud.

1. Asegúrate de que no se estén ejecutando otras copias de los contenedores primero. Usa docker ps para enumerar los contenedores y docker rm -f <ids> paraeliminarlos.
2. Inicia la pila de aplicaciones usando docker compose up.

   $ docker compose up -d

Cuando ejecutas el comando anterior, deberías ver un resultado como el siguiente:

Creating network "app_default" with the default driver

Creating volume "app_todo-mysql-data" with default driver

Creating app_app_1 ... done

Creating app_mysql_1 ... done

Notarás que Docker Compose creó el volumen y también una red. De forma predeterminada, Docker Compose crea automáticamente una red específica para la pila de aplicaciones (razón por la cual no se definió una en el archivo Compose).

3. Mira los registros usando docker compose logs -f. Verás los registros de cada uno de losservicios entrelazados en una sola secuencia.

Si ya ejecutaste el comando, verás un resultado similar a este:

mysql_1 | 2019-10-03T03:07:16.083639Z 0 [Note] mysqld: ready for connections.

mysql_1 | Version: '8.0.31' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL Community Server (GPL)

app_1 | Connected to mysql db at host mysql

app_1 | Listening on port 3000

4. En este punto, deberías poder abrir tu aplicación en el el puerto 3000

¡Accede a la BBDD y comprueba como todo lo que agregas a tu aplicación aparece también allí reflejado!