Linking containers

Sunday, 4 August 2019 9:48 PM

In a multi-tier application, both the application server container and database server container may need to share variables such as database login credentials. Of course, we can pass all database connectivity settings to the application container using envi ronment variables. It is very easy to make a mistake while passing multiple -e options to the docker run command, and it is very time-consuming, not to mention that it is very ineffective. Another option is to use container IP addresses to establish connectio ns. We can gather IP address information using docker inspect but it will be difficult to track this information in a multi-container environment.
This means that using environment variables is just not enough to build multi -tier applications where containers depend on each other.

Docker has a featured called *linked containers* to solve this problem. It automatically copies all environment variables from one container to another. Additionally, by linking containers, we can define environment variables based on the other container's IP address and exposed ports

Using linked containers is done by simply adding the --link container:alias option to the docker run command. For example, the following command links to a container named MariaDB using the DB alias:

$ docker run --link mariadb:db --name my_application httpd

```
njain50@ubuntu: ~/dockerdemo
njain50@ubuntu:~/dockerdemo$ docker run --link mariadb:db --name my_application  httpd
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.3. Set the 'ServerName' directive globally to suppress this messa
e
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.3. Set the 'ServerName' directive globally to suppress this messa
e
[Sun Aug 04 13:54:12.526568 2019] [mpm_event:notice] [pid 1:tid 140399824798848] AH00489: Apache/2.4.39 (Unix) configured -- resuming normal operations
[Sun Aug 04 13:54:12.526858 2019] [core:notice] [pid 1:tid 140399824798848] AH00094: Command line: 'httpd -D FOREGROUND'
```

The new my_application container will then get all variables defined from the linked container mariadb. Those variable names are prefixed by DB_ENV_ so as not to conflict with the new container's own environment variables.
Please be aware that the aliases are all uppercase.

Variables providing information about container IP addresses and ports are named according to the following scheme:

- {ALIAS}_PORT_{exposed-port}_TCP_ADDR
- {ALIAS}_PORT_{exposed-port}_TCP_PORT

Continuing with the MariaDB image example, the application container would get the following variables:

- DB_PORT_3306_TCP_ADDR
- DB_PORT_3306_TCP_PORT

If the linked container exposes multiple ports, each of them generates a set of environment variables.

Example:

We will be creating a WordPress container which needs access to a database server. This integration will require shared datab ase access credentials. The first step in creating this application is to create a database server:

```
njain50@ubuntu:~/dockerdemo$ docker run -d --name mariadb -e MYSQL_ROOT_PASSWORD=wordpress -e MYSQL_DATABASE=wordpress -e MYSQL_USER=wordpress -e MYSQL_PASSWORD=passwor
d mariadb
cb784b018972a3e1c719274f9a530245fd996ee00e1b1a791e933423303243b5
```

The next step is to run a WordPress container. In that command, we will link the wordpress container with the mariadb container:

```
njain50@ubuntu:~/dockerdemo$ docker run -d --name wordpress --link mariadb:mysql -p 8080:80 wordpress
94a13c65f6f6051d2870010a39a9c36e6d29ae9fa7e5d1011c413f38ee08e86f
njain50@ubuntu:~/dockerdemo$
```

Let's check container environments with the docker exec command:

```
njain50@ubuntu:~/dockerdemo$ docker exec -it wordpress env|grep -i mysql
MYSQL_PORT=tcp://172.17.0.2:3306
MYSQL_PORT_3306_TCP=tcp://172.17.0.2:3306
MYSQL_PORT_3306_TCP_ADDR=172.17.0.2
MYSQL_PORT_3306_TCP_PORT=3306
MYSQL_PORT_3306_TCP_PROTO=tcp
MYSQL_NAME=/wordpress/mysql
MYSQL_ENV_MYSQL_PASSWORD=password
MYSQL_ENV_MYSQL_ROOT_PASSWORD=wordpress
MYSQL_ENV_MYSQL_DATABASE=wordpress
MYSQL_ENV_MYSQL_USER=wordpress
MYSQL_ENV_GOSU_VERSION=1.10
MYSQL_ENV_GPG_KEYS=177F4010FE56CA3336300305F1656F24C74CD1D8
MYSQL_ENV_MARIADB_MAJOR=10.4
MYSQL_ENV_MARIADB_VERSION=1:10.4.7+maria~bionic
njain50@ubuntu:~/dockerdemo$
```

Screen clipping taken: 4/8/2019 10:08 PM

We can see here that the link set a number of *MYSQL_ENV* and *MYSQL_PORT* variables, which are used by the WordPress startup script.