Guided Exercise

Deploying the Web Application and MySQL Containers

In this lab, you will create a script that runs and networks a Node.js application container and the MySQL container.

Outcomes

You should be able to network containers to create a multi-tiered application.

Before You Begin

You must have the To Do List application source code and lab files on **workstation**. To set up the environment for the exercise, run the following command:

```
[student@workstation ~]$ lab multicontainer-design start
```

1. Build the MySQL image.

   A custom MySQL 5.7 image is used for this exercise. It is configured to automatically run any scripts in the **/var/lib/mysql/init** directory. The scripts load the schema and some sample data into the database for the To Do List application when a container starts.

   1.1. Review the Dockerfile.

   Using your preferred editor, open and examine the completed Dockerfile located at **/home/student/DO180/labs/multicontainer-design/images/mysql/Dockerfile**.

   1.2. Build the MySQL database image.

   To build the base image, run the following **podman build** script.

```
[student@workstation ~]$ cd ~/DO180/labs/multicontainer-design/images/mysql
[student@workstation mysql]$ sudo podman build -t do180/mysql-57-rhel7 \
> --layers=false .
STEP 1: FROM rhscl/mysql-57-rhel7
Getting image source signatures
...output omitted...
Storing signatures
STEP 2: ADD root /
ERRO[0017] HOSTNAME is not supported for OCI image format, hostname d4f8f8506f2b
 will be ignored. Must use `docker` format
--> 41a6...cc7e
STEP 3: COMMIT do180/mysql-57-rhel7
...output omitted...
Writing manifest to image destination
Storing signatures
--> 8dc1...6933
```

The error shown is benign, and can be ignored. It will disappear in later versions of Podman.

1.3. This command builds a dedicated MySQL container image based on the **/home/student/DO180/labs/multicontainer-design/images/mysql** context folder and the **Dockerfile** file in it. Wait for the build to complete, and then run the following command to verify that the image is built successfully:

```
[student@workstation mysql]$ sudo podman images
REPOSITORY                           TAG     IMAGE ID      CREATED         SIZE
localhost/do180/mysql-57-rhel7       latest  8dc111531fce  21 seconds ago  444MB
registry.[...]/rhscl/mysql-57-rhel7  latest  c07bf25398f4  4 weeks ago     444MB
```

2. Build the Node.js parent image using the provided Dockerfile.

2.1. Review the Dockerfile.

Using your preferred editor, open and examine the completed Dockerfile located at **/home/student/DO180/labs/multicontainer-design/images/nodejs/Dockerfile**.

Notice the following instructions defined in the Dockerfile:

- Two environment variables, **NODEJS_VERSION** and **HOME**, are defined using the **ENV** command.
- Packages necessary for Node.js are installed with **yum** using the **RUN** command.
- A new user and group are created to run the Node.js application along with the **app-root** directory using the **RUN** command.
- The **enable-rh-nodejs8.sh** script, to run automatically at login, is added to **/etc/profile.d/** with the **ADD** command.
- The **USER** command is used to switch to the newly created **appuser** account.
- The **WORKDIR** command is used to switch to the **$HOME** directory for application execution.

2.2. Build the parent image.

To build the base image, run the **podman build** command.

```
[student@workstation ~]$ cd ~/DO180/labs/multicontainer-design/images/nodejs
[student@workstation nodejs]$ sudo podman build -t do180/nodejs \
> --layers=false .
STEP 1: FROM ubi7/ubi:7.7
Getting image source signatures
...output omitted...
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                    Arch    Version       Repository                Size
================================================================================
Installing:
```

```
  rh-nodejs8                x86_64 3.0-5.el7     ubi-server-rhscl-7-rpms   7.3 k
...output omitted...
Writing manifest to image destination
Storing signatures
--> 5e61...30de
```

2.3. Wait for the build to complete, and then run the following command to verify that the image has been built successfully. Several columns of the **`podman images`** column are not relevant, so you can use the **`--format`** option to format the output.

```
[student@workstation nodejs]$ sudo podman images \
> --format "table {{.ID}} {{.Repository}} {{.Tag}}"
IMAGE ID       REPOSITORY                                    TAG
78c2e6304f23   localhost/do180/mysql-57-rhel7                latest
5e610ea8d94a   localhost/do180/nodejs                        latest
c07bf25398f4   registry.access.redhat.com/rhscl/mysql-57-rhel7   latest
22ba71124135   registry.access.redhat.com/ubi7/ubi           7.7
```

3. Build the To Do List application child image using the provided Dockerfile.

3.1. Review the Dockerfile.

Using your preferred editor, open and examine the completed Dockerfile located at **`/home/student/DO180/labs/multicontainer-design/deploy/nodejs/Dockerfile`**.

3.2. Explore the Environment Variables.

Inspect the environment variables that allow the Node.js REST API container to communicate with the MySQL container.

1. View the file **`/home/student/DO180/labs/multicontainer-design/deploy/nodejs/nodejs-source/models/db.js`**, containing the database configuration provided below:

```
module.exports.params = {
  dbname: process.env.MYSQL_DATABASE,
  username: process.env.MYSQL_USER,
  password: process.env.MYSQL_PASSWORD,
  params: {
    host: "10.88.100.101",
    port: "3306",
    dialect: 'mysql'
  }
};
```

2. Notice the environment variables used by the REST API. These variables are exposed to the container using **`-e`** options with the **`podman run`** command in this guided exercise. Those environment variables are described below.

MYSQL_DATABASE
: The name of the MySQL database in the **`mysql`** container.

MYSQL_USER
: The name of the database user used by the **`todoapi`** container to run MySQL commands.

MYSQL_PASSWORD
: The password of the database user that the **todoapi** container uses to authenticate to the **mysql** container.

Note

The host and port details of the MySQL container are embedded with the REST API application. The host, as shown above in the **db.js** file, is the IP address of the **mysql** container.

3.3. Build the child image.

Examine the **/home/student/DO180/labs/multicontainer-design/deploy/nodejs/build.sh** script to see how the image is built. Run the following commands to build the child image.

```
[student@workstation nodejs]$ cd ~/DO180/labs/multicontainer-design/deploy/nodejs
[student@workstation nodejs]$ ./build.sh
STEP 1: FROM do180/nodejs
STEP 2: MAINTAINER username <username@example.com>
STEP 3: COPY run.sh build ${HOME}/
...output omitted...
Writing manifest to image destination
Storing signatures
--> 2b12...6463
```

Note

The **build.sh** script lowers restrictions for write access to the build directory, allowing non-root users to install dependencies.

3.4. Wait for the build to complete and then run the following command to verify that the image has been built successfully:

```
[student@workstation nodejs]$ sudo podman images \
> --format "table {{.ID}} {{.Repository}} {{.Tag}}"
IMAGE ID       REPOSITORY                                      TAG
2b127523c28e   localhost/do180/todonodejs                      latest
6bf46c84a3c5   localhost/do180/nodejs                          latest
fa0084527d05   localhost/do180/mysql-57-rhel7                  latest
c07bf25398f4   registry.access.redhat.com/rhscl/mysql-57-rhel7 latest
22ba71124135   registry.access.redhat.com/ubi7/ubi             7.7
```

4. Modify the existing script to create containers with the appropriate IP, as defined in the previous step. In this script, the order of commands is given such that it starts the **mysql** container and then starts the **todoapi** container before connecting it to the **mysql** container. After invoking every container, there is a wait time of 9 seconds, so each container has time to start.

4.1. Edit the **run.sh** file located at **/home/student/DO180/labs/multicontainer-design/deploy/nodejs/networked** to insert the **podman run** command at the appropriate line for invoking **mysql** container. The following screen shows the exact **podman** command to insert into the file.

```
sudo podman run -d --name mysql -e MYSQL_DATABASE=items -e MYSQL_USER=user1 \
-e MYSQL_PASSWORD=mypa55 -e MYSQL_ROOT_PASSWORD=r00tpa55 \
-v $PWD/work/data:/var/lib/mysql/data \
-v $PWD/work/init:/var/lib/mysql/init -p 30306:3306 \
--ip 10.88.100.101 do180/mysql-57-rhel7
```

In the previous command, the **MYSQL_DATABASE**, **MYSQL_USER**, and **MYSQL_PASSWORD** are populated with the credentials to access the MySQL database. These environment variables are required for the **mysql** container to run. Also, the **$PWD/work/data** and **$PWD/work/init** local folders as mounted as volumes into the container's file system.

Note the IP assigned to the container. This IP should be the same as the one provided in the **/home/student/DO180/labs/multicontainer-design/deploy/nodejs/nodejs-source/models/db.js** file.

4.2. In the same **run.sh** file, insert another **podman run** command at the appropriate line to run the **todoapi** container. The following screen shows the **docker** command to insert into the file.

```
sudo podman run -d --name todoapi -e MYSQL_DATABASE=items -e MYSQL_USER=user1 \
-e MYSQL_PASSWORD=mypa55 -p 30080:30080 \
do180/todonodejs
```

Note

After each **podman run** command inserted into the **run.sh** script, ensure that there is also a **sleep 9** command. If you need to repeat this step, the **work** directory and its contents must be deleted before re-running the **run.sh** script.

4.3. Verify that your **run.sh** script matches the solution script located at **/home/student/DO180/solutions/multicontainer-design/deploy/nodejs/networked/run.sh**.

4.4. Save the file and exit the editor.

5. Run the containers.

5.1. Use the following command to execute the script that you updated to run the **mysql** and **todoapi** containers.

```
[student@workstation nodejs]$ cd \
/home/student/DO180/labs/multicontainer-design/deploy/nodejs/networked
[student@workstation networked]$ ./run.sh
```

Note

It is possible that the IP selected for the container (10.88.100.101) is already reserved for another container, even if that container was already deleted. In this case, delete the **todonodejs** image and container before creating an updated one with the **sudo podman rmi -f todonodejs** command. Then delete the MySQL container with the **sudo podman rm -f mysql** command. Afterwards, return to step 3, and update both **db.js** and **run.sh** with another available IP.

5.2. Verify that the containers started successfully.

```
[student@workstation networked]$ sudo podman ps \
> --format="table {{.ID}} {{.Names}} {{.Image}} {{.Status}}"
CONTAINER ID   NAMES     IMAGE                                  STATUS
c74b4709e3ae   todoapi   localhost/do180/todonodejs:latest      Up 3 minutes ago
3bc19f74254c   mysql     localhost/do180/mysql-57-rhel7:latest  Up 3 minutes ago
```

6. Examine the environment variables of the API container.

Run the following command to explore the environment variables exposed in the API container.

```
[student@workstation networked]$ sudo podman exec -it todoapi env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=2c61a089105d
TERM=xterm
MYSQL_DATABASE=items
MYSQL_USER=user1
MYSQL_PASSWORD=mypa55
container=oci
NODEJS_VERSION=8.0
HOME=/opt/app-root/src
```

7. Test the application.

7.1. Run a **curl** command to test the REST API for the To Do List application.

```
[student@workstation networked]$ curl -w "\n" \
> http://127.0.0.1:30080/todo/api/items/1
{"description": "Pick up newspaper", "done": false, "id":1}
```

The **-w "\n"** option with **curl** command lets the shell prompt appear at the next line rather than merging with the output in the same line.

7.2. Open Firefox on **workstation** and point your browser to `http://127.0.0.1:30080/todo/`. You should see the To Do List application.

Note

Make sure to append the trailing slash (/).

Finish

On **workstation**, run the **lab multicontainer-design finish** script to complete this exercise.

```
[student@workstation ~]$ lab multicontainer-design finish
```

This concludes the guided exercise.