Lab 10. Working with Images and Containers

Objectives

- Download Hello-world Image from Registry
- Search for an Image in Registry
- Pull MySQL image and run mysql docker container
- Gain access to MySQL Server and container shell.

Prerequisites

- 64-bit Oracle Linux 7

Sequence 1. Working with Docker Images

Docker containers are run from Docker images. By default, it pulls these images from Docker Hub, a Docker registry managed by Docker, the company behind the Docker project. Anybody can build and host their Docker images on Docker Hub, so most applications and Linux distributions you'll need to run Docker containers have images that are hosted on Docker Hub.

1. To check whether you can access and download images from Docker Hub, type:

```
# docker run hello-world
```

2. The output, which should include the following:

```
[root@server ~]# docker run hello-world
Unable to find image 'hello-world:latest' locally
Trying to pull repository docker.io/library/hello-world ...
latest: Pulling from docker.io/library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:d58e752213a51785838f9eed2b7a498ffa1cb3aa7f946dda11af39286c3db9a9
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

3. You can search for images available on Docker Hub by using the docker command with the search subcommand. For example, to search for the MySQL image, type:

```
# docker search mysql
```

4. The script will crawl Docker Hub and return a listing of all images whose name match the search string. In this case, the output will be similar to this:

```
[root@localhost ~]# docker search mysql
NAME                              DESCRIPTION                                     STARS               OFFICIAL
mysql                             MySQL is a widely used, open-source relation…   9550                [OK]
mariadb                           MariaDB is a community-developed fork of MyS…   3469                [OK]
mysql/mysql-server                Optimized MySQL Server Docker images. Create…   700
centos/mysql-57-centos7           MySQL 5.7 SQL database server                   76
mysql/mysql-cluster               Experimental MySQL Cluster Docker images. Cr…   69
centurylink/mysql                 Image containing mysql. Optimized to be link…   61
deitch/mysql-backup               REPLACED! Please use http://hub.docker.com/r…   41
bitnami/mysql                     Bitnami MySQL Docker Image                      39
tutum/mysql                       Base docker image to run a MySQL database se…   35
schickling/mysql-backup-s3        Backup MySQL to S3 (supports periodic backup…   30
prom/mysqld-exporter                                                              28
linuxserver/mysql                 A Mysql container, brought to you by LinuxSe…   25
centos/mysql-56-centos7           MySQL 5.6 SQL database server                   19
circleci/mysql                    MySQL is a widely used, open-source relation…   19
databack/mysql-backup             Back up mysql databases to... anywhere!         18
mysql/mysql-router                MySQL Router provides transparent routing be…   15
arey/mysql-client                 Run a MySQL client from a docker container      14
openshift/mysql-55-centos7        DEPRECATED: A Centos7 based MySQL v5.5 image…   6
```

In the OFFICIAL column, OK indicates an image built and supported by the company behind the project.

5. Once you've identified the image that you would like to use, you can download it to your computer using the pull subcommand:

```
# docker pull mysql
```

```
[root@localhost ~]# docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
afb6ec6fdc1c: Downloading [==============================================>       ]  23.4
0bdc5971ba40: Download complete
97ae94a2c729: Download complete
f777521d340e: Download complete
1393ff7fc871: Download complete
a499b89994d9: Downloading [==============================================>       ]  11.6
7ebe8eefbafe: Download complete
597069368ef1: Download complete
ce39a5501878: Downloading [=>                                                    ]  3.21
7d545bca14bf: Waiting
```

6. To see the images that have been downloaded to your computer, type:

```
# docker images
```

7. The output should look similar to the following:

```
[root@localhost ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
mysql               latest              30f937e841c8        6 days ago          541MB
hello-world         latest              bf756fb1ae65        4 months ago        13.3kB
```

Sequence 2. Running a Docker Container

Containers can be interactive. After all, they are similar to virtual machines, only more resource-friendly. Let's run a container using MySQL image. The combination of the -i and -t switches gives you interactive shell access into the container:

Starting a MySQL Server Instance

1. To start a new Docker container for a MySQL Server, use:

```
# docker run --name=mysql1 -d mysql/mysql-server:8.0
```

```
[root@localhost ~]# docker run --name=mysql1 -d mysql/mysql-server:8.0
Unable to find image 'mysql/mysql-server:8.0' locally
8.0: Pulling from mysql/mysql-server
0e690826fc6e: Pull complete
0e6c49086d52: Pull complete
862ba7a26325: Pull complete
7731c802ed08: Pull complete
Digest: sha256:a82ff720911b2fd40a425fd7141f75d7c68fb9815ec3e5a5a881a8eb49677087
Status: Downloaded newer image for mysql/mysql-server:8.0
3a5f12589cacc581744ebfb81255a45ef71e52f6261cd895001f710493de0d18
```

2. Initialization for the container begins, and the container appears in the list of running containers when you run the **docker ps** command. For example:
docker ps

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE                  COMMAND                  CREATED             STATUS                        PORTS
         NAMES
3a5f12589cac        mysql/mysql-server:8.0   "/entrypoint.sh mysq…"   About a minute ago   Up About a minute (healthy)   3306/tc
60/tcp   mysql1
[root@localhost ~]#
```

3. The -d option used in the **docker run** command above makes the container run in the background. Use this command to monitor the output from the container:

```
# docker logs mysql1
```

```
2020-05-27T11:45:34.428930Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self sign
2020-05-27T11:45:34.502356Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connect
lib/mysql/mysql.sock'  port: 0  MySQL Community Server - GPL.
Warning: Unable to load '/usr/share/zoneinfo/iso3166.tab' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/leapseconds' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/tzdata.zi' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/zone.tab' as time zone. Skipping it.
Warning: Unable to load '/usr/share/zoneinfo/zone1970.tab' as time zone. Skipping it.
[Entrypoint] GENERATED ROOT PASSWORD: Cac;ecIHLIs80sKOn1Ow#ic4sEm

[Entrypoint] ignoring /docker-entrypoint-initdb.d/*

2020-05-27T11:45:41.244046Z 10 [System] [MY-013172] [Server] Received SHUTDOWN from user root.
2020-05-27T11:45:45.482335Z 0 [System] [MY-010910] [Server] /usr/sbin/mysqld: Shutdown complete
```

4. Check the password with:

```
# docker logs mysql1 2>&1 | grep 'GENERATED ROOT PASSWORD'
```

```
[root@localhost ~]# docker logs mysql1 2>&1 | grep 'GENERATED ROOT PASSWORD'
[Entrypoint] GENERATED ROOT PASSWORD: Cac;ecIHLIs80sKOn1Ow#ic4sEm
[root@localhost ~]#
```

5. Let us connect to MySQL Server from within the Container. Use the docker exec -it command to start a mysql client inside the Docker container you have started:

```
# docker exec -it mysql1 mysql -uroot -p
```

```
[root@localhost ~]# docker exec -it mysql1 mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 24
Server version: 8.0.20
```

```
mysql> alter user root@localhost identified by 'Sangwan#12';
Query OK, 0 rows affected (0.02 sec)
```

6. To have shell access to your MySQL Server container, use ***docker exec -it*** command to start a bash shell inside the container:

```
# docker exec -it mysql1 bash
bash-4.2#
```

Sequence 3. Stopping and Deleting a Container

1. To stop the MySQL Server container, use:

```
# docker stop mysql1
docker stop sends a SIGTERM signal to the mysqld process, so that
the server is shut down gracefully.
```

Also notice that when the main process of a container (mysqld in the case of a MySQL Server container) is stopped, the Docker container stops automatically.

2. To start the MySQL Server container again:

```
# docker start mysql1
```

3. To stop and start again the MySQL Server container with a single command:

```
# docker restart mysql1
```

4. To delete the MySQL container, stop it first, and then use the docker rm command:

```
# docker stop mysql1
# docker rm mysql1
```

Sequence 4. Listing Docker Containers and Cleanup

1. After using Docker for a while, you'll have many active (running) and inactive containers on your computer. To view the active ones, use:

```
# docker ps
```

2. To view all containers — active and inactive, pass it the -a switch:

```
# docker ps -a
```

3. To view the latest container you created, pass it the -l switch:

```
# docker ps -l
```

4. Stopping a running or active container is as simple as typing:

```
# docker stop container-id
```

5. Remove a Container

```
# docker rm <container id>
```

6. Remove Docker Images

```
# docker rmi <image ID>
```