Deepak Sureshkumar + Follow
Technical Architect - Architecture Chapter - Emirates NBD
4 articles

Environment

Published on September 4, 2017

Hello all. Welcome back to my series of Docker stuffs. In the previous articles, we saw how to get a basic hands-on with Docker, and deploying a static website in AWS using Docker. Today, I'm going to take you a little further with building a docker image using the Docker file and to run linked container environments, by linking 2 different containers (one Standalone Java Application Container and one MySQL Container).

Prerequisites

- 64bit Linux machine on AWS
- Security Group to allow traffic from port 22 to telnet that EC2 instance
- MySQL Connector Java jar

MySQL Container

Here I'm going to follow the same steps given in one of my previous post on "Building MySQL on Docker".

- Run the MySQL container with the below command

```
docker run -d --name db \
 -e MYSQL_ROOT_PASSWORD=simple -e MYSQL_DATABASE=firstdb \
 -e MYSQL_USER=user -e MYSQL_PASSWORD=pass123 \
  myql
```

- Get into the bash shell of the container

```
docker exec -it db bash
```

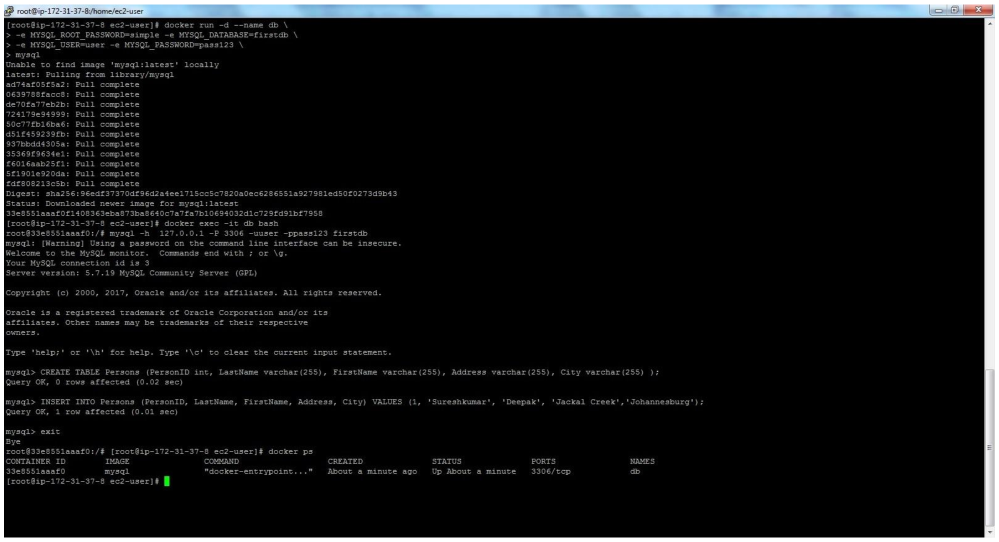
- Connect to MySQL

- Create table with some records

```
CREATE TABLE Persons (PersonID int, LastName varchar(255), FirstName varchar(25

INSERT INTO Persons (PersonID, LastName, FirstName, Address, City) VALUES (1, 'S
```

- Exit MySQL
- Instead of typing exit from shell, type CTRL+P followed by CTRL+Q. This will bring you out of the shell without killing the process.



root@ip-172-31-37-8:/home/ec2-user

```
[root@ip-172-31-37-8 ec2-user]# docker run -d --name db \
> -e MYSQL_ROOT_PASSWORD=simple -e MYSQL_DATABASE=firstdb \
> -e MYSQL_USER=user -e MYSQL_PASSWORD=pass123 \
> mysql
Unable to find image 'mysql:latest' locally
latest: Pulling from library/mysql
ad74af05f5a2: Pull complete
0639788facc8: Pull complete
de70fa77eb2b: Pull complete
724179e94999: Pull complete
50c77fb16ba6: Pull complete
d51f459239fb: Pull complete
937bbdd4305a: Pull complete
35369f9634e1: Pull complete
f6016aab25f1: Pull complete
5f1901e920da: Pull complete
fdf808213c5b: Pull complete
Digest: sha256:96edf37370df96d2a4ee1715cc5c7820a0ec6286551a927981ed50f0273d9b43
Status: Downloaded newer image for mysql:latest
33e8551aaaf0f1408363eba873ba8640c7a7fa7b10694032d1c729fd91bf7958
[root@ip-172-31-37-8 ec2-user]# docker exec -it db bash
root@33e8551aaaf0:/# mysql -h  127.0.0.1 -P 3306 -uuser -ppass123 firstdb
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.19 MySQL Community Server (GPL)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE TABLE Persons (PersonID int, LastName varchar(255), FirstName varchar(255), Address varchar(255), City varchar(255) );
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO Persons (PersonID, LastName, FirstName, Address, City) VALUES (1, 'Sureshkumar', 'Deepak', 'Jackal Creek','Johannesburg');
Query OK, 1 row affected (0.01 sec)

mysql> exit
Bye
root@33e8551aaaf0:/# [root@ip-172-31-37-8 ec2-user]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED              STATUS              PORTS               NAMES
33e8551aaaf0        mysql               "docker-entrypoint..."   About a minute ago   Up About a minute   3306/tcp            db
[root@ip-172-31-37-8 ec2-user]#
```

Introduction to Dockerfiles

Docker builds images by reading instructions from a Dockerfile. A Dockerfile is a simple text file that contains instructions that can be executed on the command line. Using docker build, you can start a build that executes all of the command-line instructions contained in the Dockerfile.

Common Dockerfile instructions start with RUN, ENV, FROM, MAINTAINER, ADD, and CMD, among others.

- **FROM -** Specifies the base image that the Dockerfile will use to build a new image.
- **MAINTAINER -** Specifies the Dockerfile Author Name and his/her email.
- **WORKDIR** - The WORKDIR is used to set where the command defined with CMD is to be executed.

- **ENV** - Sets the environment variables.
- **CMD -** Provides the facility to run commands at the start of container. overridden upon executing the docker run command.
- **ADD -** This copies the new files, directories into the Docker container specified destination.
- **EXPOSE -** This exposes specified port to the host machine.

Starting Java Application Container

So far, in all previous articles I was using docker images by pulling them from the Docker Hub. And now we are going to see how to use the Dockerfile to build our images in the local machine.

- Do an SFTP of the MySQL Connector Java jar file and Simple Java JDBC Connection program to read and print data from MySQL table, to the Linux VM
- Create a Dockerfile in same folder, file name as Dockerfile without any extension
- Dockerfile

```
FROM java:8
COPY . /
WORKDIR /
RUN javac DockerConnectMySQL.java
CMD ["java", "-classpath", "mysql-connector-java-5.1.6.jar:.","DockerConnectMySQL"]
```

FROM java:8 #(Get the latest Java 8 Image)

COPY . / #(Copy current directory files to "/" folder of container)

WORKDIR / #(Make "/" as working directory of container)

RUN javac DockerConnectMySQL.java #(Copy command already copied our Java program into the container and now we are just compiling our program)

CMD ["java", "-classpath", "mysql-connector-java-.1.6.jar:.","DockerConnectMySQL"] #(This will run while starting our container)

- Java File (DockerConnectMySQL.java)

```
import java.sql.*;
public class DockerConnectMySQL {
   static final String USER = "user";
   static final String PASS = "pass123";

   public static void main(String[] args) {
   Connection conn = null;
   Statement stmt = null;
   try{
      Class.forName("com.mysql.jdbc.Driver");

      System.out.println("Connecting to database...");
      conn = DriverManager.getConnection(DB_URL,USER,PASS);

      stmt = conn.createStatement();
      String sql;
      sql = "SELECT PersonID, FirstName, LastName, Address, City FROM Persons";
      ResultSet rs = stmt.executeQuery(sql);

      while(rs.next()){
         int id  = rs.getInt("PersonID");
         String first = rs.getString("FirstName");
         String last = rs.getString("LastName");
                 String address = rs.getString("Address");
                 String city = rs.getString("City");

         System.out.println("ID: " + id);
         System.out.println(", First: " + first);
         System.out.println(", Last: " + last);
                 System.out.println(", Address: " + address);
                 System.out.println(", City: " + city);
      }
      rs.close();
      stmt.close();
      conn.close();
   }catch(SQLException se){
      se.printStackTrace();
   }catch(Exception e){
      e.printStackTrace();
   }finally{
      try{
         if(stmt!=null)
            stmt.close();
      }catch(SQLException se2){
      }
      try{
         if(conn!=null)
            conn.close();
      }catch(SQLException se){
         se.printStackTrace();
      }
   }
 }
}
```

- **static final String DB_URL = "jdbc:mysql://db:3306/firstdb"; -** I've given the hostname as db, which is none other than our MySQL container name.
- Let's Build our first Dockerfile with image name as javadbapp

```
docker build -t javadbapp .
```

- Our first Dockerfile build is success, so what's next? Yes, running this Java application container
- Let's run our application

Free Upgrade to Premium

- Oops our application was not able to find the db host. This is because v different containers running in same machine, without having any link between them. This is similar, when you want network to connect an ap is running on one physical machine and DB server is on another one.

Create Link Between Containers

- Docker is having a provision to make connection between containers using Legacy container link or Docker Network.
- For simple understating, I'm going to use Legacy container link in this article to make the connection between containers. (*Note:- Legacy container link is a deprecated feature of Docker. Docker Network is the latest feature to make a bridge between containers. We will see in our next article using Docker Network feature*).
- Now running containers in linked mode

```
docker run --name linkcontainers --link db javadbapp
```

- And now our Java application is able to find the db host and executes successfully with printing the results on console

That's it. Same scenario can be handled in a better way with Spinning DB containers using scripts of DB dumps, instead of getting into DB container shell for creating tables and inserting records. And deploying any Spring / Spring Boot application instead of just a standalone Java application.

But I sense those kind of heavy configuration will need lots of time to understand the configurations and lots of efforts for debugging, if something goes wrong. Hence made this simple scenario for everyone to understand and have their own successful hands-on experience. Meet you next with a simple hands-on for Docker Orchestration using Swarm.

Report this

14 Likes

+4

Free Upgrade to Premium

Show previous comments

Uma Tiwari • 3rd
Cloud Quality Coach | Software Associate | DevOps Engineer | ITIL certified | Red Hat Certified

Hi deepak its really helpful but i am getting "java.lang.ClassNotFoundException: com.mysql.jdbc.Driver" exception can you help

Like Reply | 2 Likes · 1 Reply

Deepak Sureshkumar • 3rd
Technical Architect - Architecture Chapter - Emirates NBD

Did you add mysql-connector-java jar in the classpath. CMD ["java", "-classpath",
connector-java-.1.6.jar:.","DockerConnectMySQL"] (edited)

Like Reply

Uma Tiwari • 3rd
Cloud Quality Coach | Software Associate | DevOps Engineer | ITIL certified | Red Hat Certified

yes I did
CMD ["java","-classpath", "mysql-connector-java-5.1.6.jar:.", "DockerConnectMySQL"]

Like Reply | 2 Likes

8mo ···

Add a comment...

Deepak Sureshkumar
Technical Architect - Architecture Chapter - Emirates NBD

+ Follow

More from Deepak Sureshkumar

Container Orchestration - Simple Hands-on with Docker Swarm (Just ...
Deepak Sureshkumar on LinkedIn

Deploying Static Website with Docker in AWS
Deepak Sureshkumar on LinkedIn

Building MySQL on Docker
Deepak Sureshkumar on LinkedIn