Docker Compose Example

This is the step by step document to understand uses of Docker compose. In this , I will create two Docker containers using Docker compose. One docker container will have MySQL database instance and another Docker container have Apache web server with our dummy application file.

Let’s follow step by step tutorial and watch the things happening there.

Step 1 – Create Directory Structure

First of all, create a directory structure. Here webapp is our web application directory. Also, create a index.html in webapp directory for testing.

```
$ mkdir dockercompose && cd dockercompose
$ mkdir webapp
$ echo "<h2>It Works</h2>" > webapp/index.html
```

Step 2 – Create Dockerfile for Webapp

Now create a Dockerfile in webapp directory to create a customized image for your application including Apache web server.

```
$ vim  webapp/Dockerfile
```

add following content

```
FROM tecadmin/ubuntu-ssh:16.04
```

```
RUN apt-get update \

   && apt-get install -y apache2



COPY index.html /var/www/html/

WORKDIR /var/www/html

CMD ["apachectl", "-D", "FOREGROUND"]

EXPOSE 80
```

Step 3 – Create Docker Compose File

Finally create a docker compose configuration file (docker-compose.yml) file in current directory. This will define all the containers will be used in your current setup.

```
$ vim  docker-compose.yml
```

add following content.

```
version: '3'
services:
  db:
    image: mysql
    container_name: mysql_db
    restart: always
    environment:
      - MYSQL_ROOT_PASSWORD="secret"
  web:
    image: apache
    build: ./webapp
    depends_on:
      - db
    container_name: apache_web
    restart: always
    ports:
      - "8080:80"
```

Above docker compose file has settings for two containers. The first container is for mysql database server and the second is for web server. The web container will run our application on Apache server. As this is customized we have defined build directory to webapp.

Step 4 – Build Webapp Image

Now, build an image using the following command. This will create an image named apache using Dockerfile and contents from webapp directory.

```
$ docker-compose build
```

read the below output of above command. I have skipped some part of output which is not required. The first line of below output shows that it skipped building for db container due to no build defined. For web container it uses **webapp/Dockerfile** to build an image.

```
db uses an image, skipping

Building web

Step 1/6 : FROM tecadmin/ubuntu-ssh:16.04

16.04: Pulling from tecadmin/ubuntu-ssh

b3e1c725a85f: Pull complete

4daad8bdde31: Pull complete

63fe8c0068a8: Pull complete

4a70713c436f: Pull complete

bd842a2105a8: Pull complete

c41407f48fa7: Pull complete

1fcfeb9b5ef4: Pull complete
```

```
13195a7d2240: Pull complete

b86be64bbda8: Pull complete

8c951fe917dc: Pull complete

f74bc80103b6: Pull complete

Digest: sha256:523d6fbc97954e9f77231bf54bfcfbbdd4805349887477fbac4a63dc735d777d

Status: Downloaded newer image for tecadmin/ubuntu-ssh:16.04

 ---> bb63b492da01

Step 2/6 : RUN apt-get update    && apt-get install -y apache2

 ---> Running in 00be0dd717ce

[[[Removed long output from here]]]

 ---> 41c731590234

Removing intermediate container 00be0dd717ce

Step 3/6 : COPY index.html /var/www/html/

 ---> 42f84d4c2243

Removing intermediate container 945aaee6cbde

Step 4/6 : WORKDIR /var/www/html

 ---> 40bebd21e352

Removing intermediate container e13f5f412906

Step 5/6 : CMD apachectl -D FOREGROUND

 ---> Running in ab0db1ef1c6e

 ---> 587bf2323289

Removing intermediate container ab0db1ef1c6e

Step 6/6 : EXPOSE 80

 ---> Running in 7bcbef52d585

 ---> 8f03d4135394

Removing intermediate container 7bcbef52d585
```

```
Successfully built 8f03d4135394

Successfully tagged apache:latest
```

Step 5 – Launch Docker Containers

Finally launch your containers using docker-compose up command. Use **-d** switch to run them in daemon mode.

```
$ docker-compose up -d
```

You can access your web application running on the apache_web container by accessing your docker host on port 8080. For example, http://dockerhost:8080/ where dockerhost is IP or hostname of your Docker host machine.

Step 6 – Update Content in Web Application

Let’s make a change in your web application. I have added some more content to webapp/index.html file as following.

```
$ echo "Welcome to Docker Compose Tutorial" >> webapp/index.html
```

Now use the following commands to rebuild webapp container and relaunch using docker-compose.

```
$ docker-compose build

$ docker-compose up -d
```