Docker tutorial

Ivan Murashko

Contents

Introduction

There are several examples of docker usage. They are collected in one place mainly for future references.

The source code for examples can be found in the article git repository [2] in the folder **dockertutorial/src**.

1 Base commands

1.1 Simple program run

You can run a command (**uname -a**) with

```
$ docker run ubuntu uname -a
```

The container **ubuntu:latest** will be used in the case.

The command execution status can be viewed with

```
$ docker ps -a
CONTAINER ID   IMAGE   COMMAND     CREATED         ...
b5e8d82cce29   ubuntu  "uname -a"  5 seconds ago   ...
```

If you run the docker with –**rm** flag then the status info will not be stored.

1.2 Interactive session

You can run an interactive shell with

```
$ docker run -it ubuntu /bin/bash
```

where **-it** means –**interactive** –**tty**, **ubuntu** the latest ubuntu image and **/bin/bash** - the command to be start

1.3 Docker as a daemon

First of all run the docker in interactive mode and as daemon

```
$ docker run -itd ubuntu
```

possible output

```
649dae02de59ea3eb065a40b1248b2d322986e563ab12af3126fa4bb4710008a
```

Check the docker run in daemon mode with

```
$ docker ps
```

output:

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND     ...
649dae02de59        ubuntu              "/bin/bash" ...
```

Execute **ls /var** command in the run docker

```
$ docker exec -it 649dae02de59 ls /var
backups  cache  lib  local  lock  log  mail  opt  run  spool  tmp
```

Stop it with

```
$ docker stop 649dae02de59
649dae02de59
```

Check the result

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND   ...
```

1.4 Network daemon

You can run a nginx web server with the following command

```
$ docker run -p 8080:80 -d nginx
```

This will map docker container port 80 to the host port 8080 or in other words make the nginx server available on the host machine via port 8080:

```
$ telnet localhost 8080
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Server: nginx/1.17.4
Date: Sat, 28 Sep 2019 18:44:45 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 24 Sep 2019 14:49:10 GMT
Connection: close
ETag: "5d8a2ce6-264"
Accept-Ranges: bytes

Connection closed by foreign host.
```

1.5 Stop all

You can stop all containers with

```
$ docker container stop $(docker container ls -aq)
```

1.6 Cleanup

The following command will remove everything

```
$ docker system prune -a
WARNING! This will remove:
        - all stopped containers
        - all networks not used by at least one container
        - all images without at least one container associated to them
        - all build cache
Are you sure you want to continue? [y/N] y
Deleted Containers:
b5e8d82cce2942a24c709b630ff4e0dd705b89d78f2777065446ce97cf152cab
...
Total reclaimed space: 6.113GB
```

2 Creating docker images

2.1 Build

In the example we will create a docker image that will help us to compile and run code that uses thrift protocol [1]. The necessary libs for us are C++ and PHP.

The Dockerfile can be found in the article git repository [2] in the folder **dockertutorial/src/thrift**.

```
FROM ubuntu:18.04

ENV DEBIAN_FRONTEND=noninteractive
ENV COMPOSER_ALLOW_SUPERUSER 1
ENV THRIFT_VERSION v0.12.0
ENV THRIFT_SRC https://github.com/apache/thrift/archive/${THRIFT_VERSION}.tar.gz

RUN buildDeps=" \
                automake \
                bison \
                curl \
                flex \
                g++ \
                php7.2-dev \
                php7.2-xml \
                git \
                composer \
                libboost-dev \
                libboost-filesystem-dev \
                libboost-program-options-dev \
                libboost-system-dev \
                libboost-test-dev \
                libevent-dev \
                libssl-dev \
                libtool \
                make \
                pkg-config \
        "; \
        apt-get update && \
    apt-get install -y --no-install-recommends $buildDeps && \
    rm -rf /var/lib/apt/lists/* \
        && curl -k -sSL "${THRIFT_SRC}" -o thrift.tar.gz \
        && mkdir -p /usr/src/thrift \
        && tar zxf thrift.tar.gz -C /usr/src/thrift --strip-components=1 \
        && rm thrift.tar.gz \
        && cd /usr/src/thrift \
        && ./bootstrap.sh \
        && ./configure \
        && make \
        && make install \
        && cd / \
        && rm -rf /usr/src/thrift \
        && rm -rf /var/cache/apt/* \
        && rm -rf /var/lib/apt/lists/* \
        && rm -rf /tmp/* \
        && rm -rf /var/tmp/*


CMD [ "thrift" ]
```

You can compile it with

```
$ cd src/thrift/
$ docker build -t thriftbuilder .
...
Successfully tagged thriftbuilder:latest
$
```

You can look at the image with

```
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
thriftbuilder       latest              be1ccc48fb53        About a
minute ago
```

You can test the result with

```
$ docker run --rm thriftbuilder  thrift --version
Thrift version 0.12.0
```

The –**rm** option is used to be sure that the stopped container was removed.

2.2 Cleanup

To remove the image you can use

```
$ docker image rm thriftbuilder
```

If you got an error during the removal:

```
Error response from daemon: conflict: unable to remove repository
reference "thriftbuilder" (must force) - container dc0836d6f65f is
using its referenced image be1ccc48fb53
```

then try to remove the stopped container before

```
$ docker container rm dc0836d6f65f
```

TBD

3 Apps in docker

I am going to create a small application that consists of 2 parts. The first one is front-end written in php that communicates with a back-end server written in C++. The communication is done via thrift protocol [1]. The front-end server as well as back-end server are run in different container and all communication is done via the network.

3.1 Build and run server (C++) application

We are going to create a separate docker image for C++ application. The image Dockerfile can be found in the article git repository [2] in the folder **dockertutorial/src/cpp**.

```
FROM thriftbuilder
RUN mkdir -p /usr/src/cpp
COPY ./cpp /usr/src/cpp/
COPY ./thrift/proto.thrift /usr/src/cpp/proto.thrift
RUN cd /usr/src/cpp \
   && make all
CMD /usr/src/cpp/server
```

As you can see it is based on the thriftbuilder image created before.
The daemon can be built with the following command run from **./src** folder:

```
$ docker build -t cppapp -f cpp/Dockerfile .
```

It also builds the application inside the docker. It can be run as follows

```
$ docker run --rm -p 9090:9090 -d cppapp
```

3.2 Build and run client (php) application

TBD

References

[1] Facebook. Apache thrift. — https://thrift.apache.org/.

[2] Murashko, I. Articles git repository. — 2019. — https://github.com/ivanmurashko/articles.