Professorship for Computer Science
Communication Services, Telecommunication Systems and Computer Networks

Container Virtualization

Docker by Example

August 21, 2018

Marcel Großmann, Stefan Kolb, Dr. Andreas Schönberger, Gabriel Nikol

Content

Docker Concepts

What does Docker do:

- "Docker allows you to package an application with all of its dependencies into a standardized unit for software development."
- "Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries - anything you can install on a server. This guarantess that it will always run the same, regardless of the environment it is running in."

These artifacts are packed and can be exchanged.

- Build, ship and run everywhere!

Background: Evolution of Virtualization

Figure 1: Uni-Logo

- Lightweight containers enable easy horizontal scalability.

Virtual Machines vs. Containers

Figure 2: Uni-Logo

Container vs. Images

- An image is blueprint for a container
- An image contains of a read-only set of filesystem layers
- An instance of an image is called container
- You can have many running containers of the same image
- Major difference: container adds a writeable filesystem layer on top of the image

Figure 3: images and containers

Docker CLI

Docker CLI

- Docker CLI (Command Line Interface) is the docker client

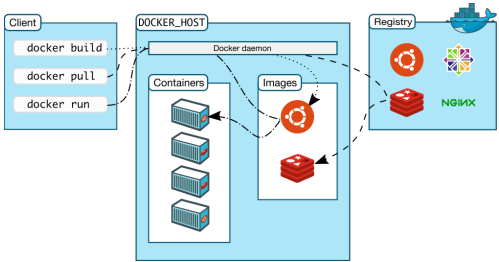

Figure 4: Docker overview

- **Daemon**: daemon of the server process to manage containers
- **Client**: user client to (remotely) controll the daemon
- **Registry**: platform for sharing and managing images

Docker List Commands

Shows **state** of containers, volumes and images.
Show **running** container only:

- *docker container ls*

Show **all** containers (both commands do the same):

- *docker container ls -a*
- *docker container ls –all*

The same for volumes, networks and images:

- docker volume ls –all
- docker image ls –all
- docker network ls

Figure 5: example of *docker container ls*

Docker Pull Command

The **docker pull IMAGE** command downloads an image from docker registry.
In our case (default) from the **public** registry - docker hub
Pull the **latest** image of busybox (default):

- docker pull busybox

Pull busybox with tag **1.29.2-glibc**:

- docker pull busybox:1.29.2-glibc

Docker Run Command

The **docker run [OPTIONS] IMAGE[:TAG]** command starts container from images.

- docker run busybox

Start container for interactive processes (*-i*) and allocate a tty (*-t*). Together written as *-it*.

Open busybox with shell:

- docker run -it busybox

Figure 6: *docker run -it busybox* opens shell in a busybox

Docker Inspect Command

You can also inspect container, volumes, networks and images giving you detailed information:

- docker container inspect ID|NAME
- docker volume inspect VOLUMENAME
- docker image inspect IMAGE
- docker network inspect ID|NAME

Figure 7: *docker image inspect ubuntu*

Docker Run Command contd.

Other very important docker run **[OPTIONS]**: Give container names (unique):

- docker run -it –name busy busybox

delete container foobar after running:

- docker run –rm busybox

Figure 8: *docker image inspect ubuntu*

Exercise 1

1. Open a first (power)shell.
2. download the image: busybox and ubuntu.
   What do you see? What is different between the two images?
3. start a **ubuntu** container with the name **foo**.
   List all directories (*ls*), create a new directory (*mkdir DIRNAME*), List all directories again.
4. Open a second (power)shell.
5. List all containers. What is the Id and name of your container?
6. Go back to the first (power)shell and exit the container.
   Grateful with *exit*, abort with *STRG+C*.
7. List all containers again. What changes?
8. delete the container by using: *docker rm ID|NAME*
9. Optional:
   inspect the ubuntu container and image
   delete the container with *docker container rm ID|NAME* and list all containers again

Docker handling detached containers

Normally container run in the background without interaction.
Run container without active tty with *–detach* or *-d* option:

- docker run –name influx –rm influxdb

Inspect stdout with *logs*

- docker logs influx

Stop running container with

- docker stop influx

Figure 9: *docker image inspect ubuntu*

environment variables

You can define environment variables passed to a container.
Syntax: *-e=VARIABLE* or *–env=VARIABLE*
Example: print out value of environment variable foo on console

- docker run -it -e foo=bar ubuntu
- / # echo $foo

Figure 10: *docker run -it -e foo=bar ubuntu*

publish ports

By default no ports are accessible outside of the container or docker network.

Syntax: **-p EXTERN:INTERN** or - **-publish EXTERN:VARIABLE**

Example: start php:apache container with port 80 published (not working)

- docker run -p 80:80 php:apache

Docker volume option

Mapping of local file system into container file system.
Syntax:

- **-v LOCALFS:CONTAINERFS**
- - **-volume LOCALFS:CONTAINERFS**

Local file system path: C:/User/Name/html Example: start php container with C:/User/Name/html mapped into /var/www/html

- docker run -v C:/User/Name/html:/var/www/html php:apache

Exercise 2

1. GoTo: `https://hub.docker.com/_/mysql/` and find the setable environment variables.
2. Run all container in detached mode
3. Run a mysql container named database with user **foo** and password **bar**.
4. Check the logs of the mysql
5. Run a php:apache container named webserver, publish port 80 and map directory Exercise2/html into /var/www/html
6. Ensure that the webservice's html page is available
7. list all running container and compare the output with Figure 11. It should be equal.

Figure 11: *Exercise2: docker container ls*

Docker volumes

Volumes without local file system. E.g. network/cloud file system.
Create a volume named database

- docker create database

Remember: list volumes

- docker volume ls

Map a volume like a file system volume by its name.

- docker run -v database:/var/www/html php:apache

Docker network

Linking Intern DNS without publishing internal ports globally to the outside

Docker network

Linking Intern DNS without publishing internal ports globally to the outside

References I

Questions ?

Gabriel Nikol
GabrielNikol@web.de