USING DOCKER
AS PHP DEVELOPER

by Robert Koch

at #ViennaPHP

21.07.2016

AGENDA

INTRODUCTION

ABOUT ME

- Coding PHP for Fun since 2004
- Studied Software Engineering
- Coding PHP for Fun & Profit,
  mostly for www.e-2.at
  - Wordpress-SEO-centric web agency
  - Dealing with sports & betting data

INTRODUCTION

DOCKER

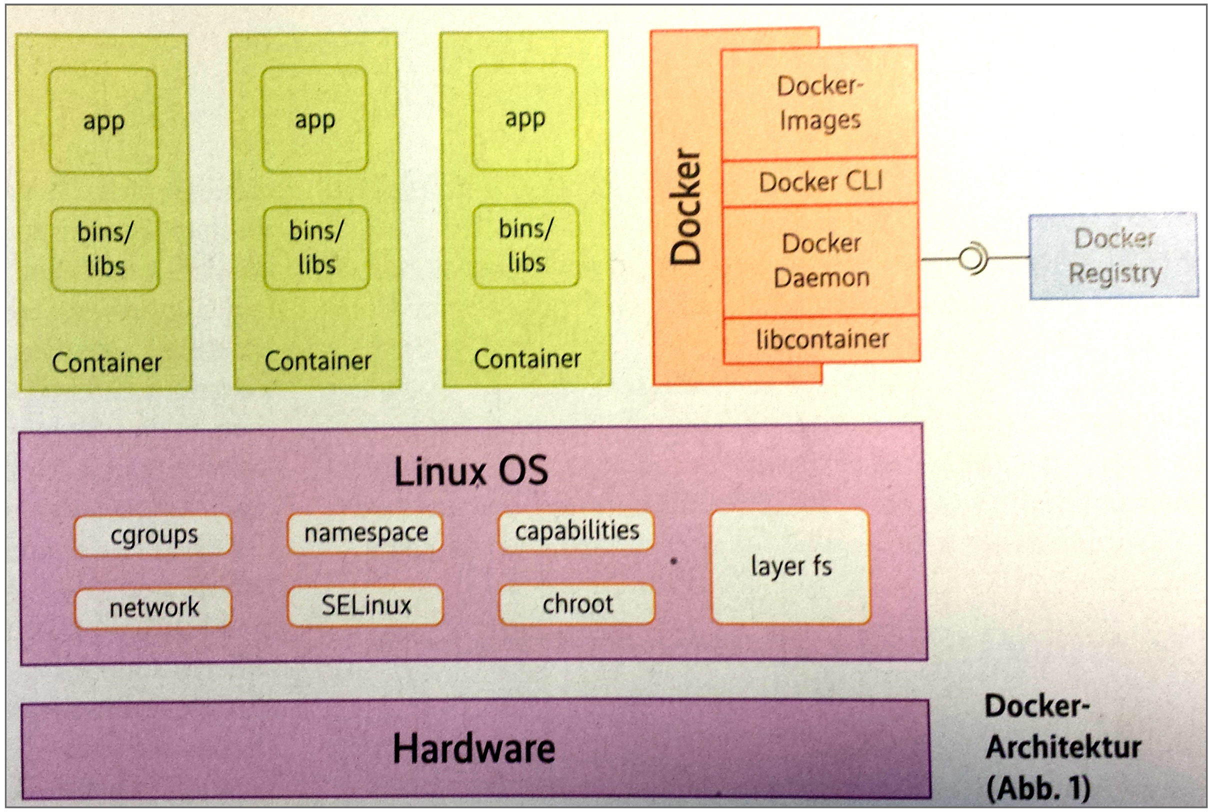
“Docker is the better virtualization”

- No virtual machine
- Uses system kernel
- Boots in seconds

This talk covers Dockers v1.11

INTRODUCTION

OVERALL ARCHITECTURE



Docker-Architektur (Abb. 1)

THE HARD PART

“Get docker running”

POSSIBLE GOTCHAS

- 32-bit system
- Old version in OS' repositories
- (non-linux system)
- Version Upgrade
- Docker's underlying file system

INSTALL DOCKER

- Use the OS' repository
- Docker Toolbox on Windows < 10 or older MacOS < 10.10
    - Is a VirtualBox
- Install from binaries ▾

INSTALL FROM BINARIES

1. CHECK REQUIREMENTS

```
#!/bin/bash

wget "https://raw.githubusercontent.com/docker/docker/master/contrib/check-config.sh"

sudo bash check-config.sh
```

INSTALL FROM BINARIES

2. INSTALL TOOLS

```
#!/bin/bash

# retrieve latest docker
wget "https://get.docker.com/builds/Linux/x86_64/docker-latest.tgz"
tar -xvzf docker-latest.tgz

# retrieve latest docker-compose
url=$(curl -s https://api.github.com/repos/docker/compose/releases/latest \
    | grep '"browser_download_url"' \
    | grep "$(uname -s)-$(uname -m)" \
    | head -n 1 \
    | cut -d '"' -f 4)
wget --output-document docker-compose "${url}"

# install
chmod +x docker/* docker-compose
sudo mv docker/* docker-compose --target-directory /usr/local/bin/

# user management
sudo groupadd docker && sudo usermod -aG docker $USER
```

INSTALL FROM BINARIES

3. START THE DOCKER DAEMON

```
bash$ sudo docker daemon -s overlay
```

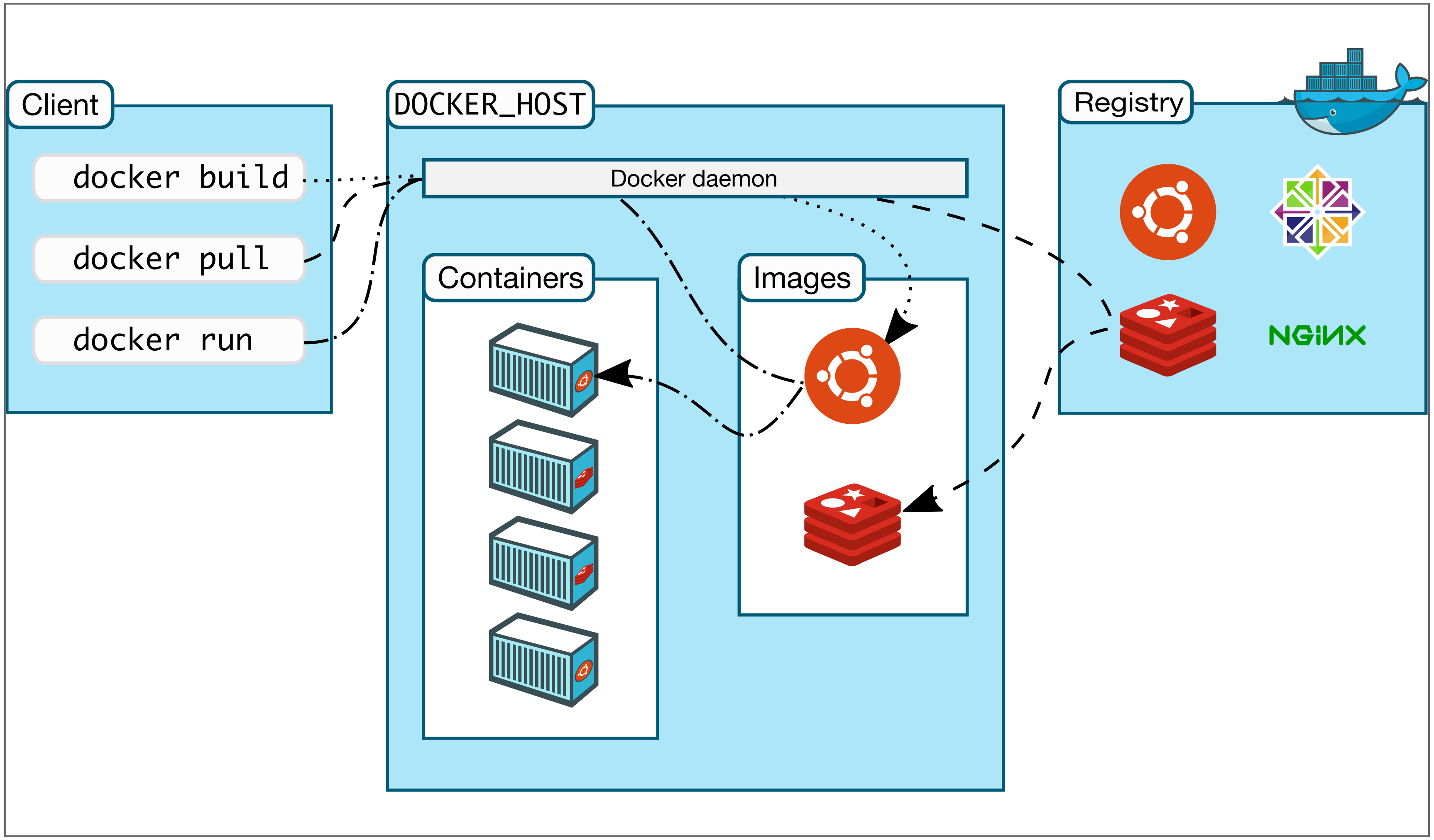
THE EASY PART

“Using docker”

BASIC CONCEPT

Dockerfile	*build plan*
↓	
Docker Image	*class definition*
↓	
Docker Containers	*instances*

USING DOCKER: ARCHITECTURE



USE DOCKER: DOCKERFILE

“build plan for image”

```
FROM debian
MAINTAINER Robert Koch <robert@e-2.at>

RUN apt-get update \
&& apt-get install -y apache2 \
&& apt-get clean;

EXPOSE 443
EXPOSE 80

#COPY httpd.conf /etc/apache/conf/

VOLUME /var/www/htdocs

COPY start.sh /start.sh
CMD ["/bin/bash", "/start.sh"]
```

start.sh

```bash
#!/bin/bash

http_ip=$(hostname -i)

echo " -----------------------------------------------------------"
echo "| Use the following URLs to access the dockerized webserver:"
echo "|  * http://${http_ip}/"
echo " -----------------------------------------------------------"

function terminate_apache() {
    echo "Terminate Apache WebServer:"
    kill -TERM `cat /usr/local/apache2/logs/httpd.pid`
}

# calling command with exec allows Ctrl+C to terminate Apache
exec apachectl -DFOREGROUND
```

USE DOCKER: DOCKERFILE

- Features inheritance
- Commands (some)
  - *FROM* - specify parent image
  - *MAINTAINER*
  - *RUN* - commands
  - *COPY* - files into container
  - *ADD* - more advanced copy (auto-extract archives & remote URL)
  - *ENV* - set environment variables
  - *EXPOSE* - promote ports
  - *USER*
  - *VOLUME* - creates a mount point
  - *ENTRYPOINT* - default: `/bin/sh -c`
  - *CMD* - default: *none*

USE DOCKER: BUILD IMAGES

```
bash$ docker build .
```

```
Sending build context to Docker daemon 3.072 kB
Step 1 : FROM debian
 ---> 1b088884749b
Step 2 : MAINTAINER Robert Koch <robert@e-2.at>
 ---> Using cache
 ---> ce07960154bb
Step 3 : RUN apt-get update && apt-get install -y apache2 && apt-get cle
 ---> Using cache
 ---> eb5a4e9618bc
Step 4 : EXPOSE 443
 ---> Using cache
 ---> 9b60b2a095c9
Step 5 : EXPOSE 80
 ---> Using cache
 ---> 507a95ea3af4
Step 6 : VOLUME /var/www/htdocs
 ---> Using cache
 ---> 29594bda03a7
Step 7 : COPY start.sh /start.sh
 ---> 3c576daa3ab4
Removing intermediate container c504a5789acf
Step 8 : CMD /bin/bash /start.sh
 ---> Running in 3ffe2ccb2724
 ---> 889c66728551
```

USE DOCKER: FETCH IMAGE

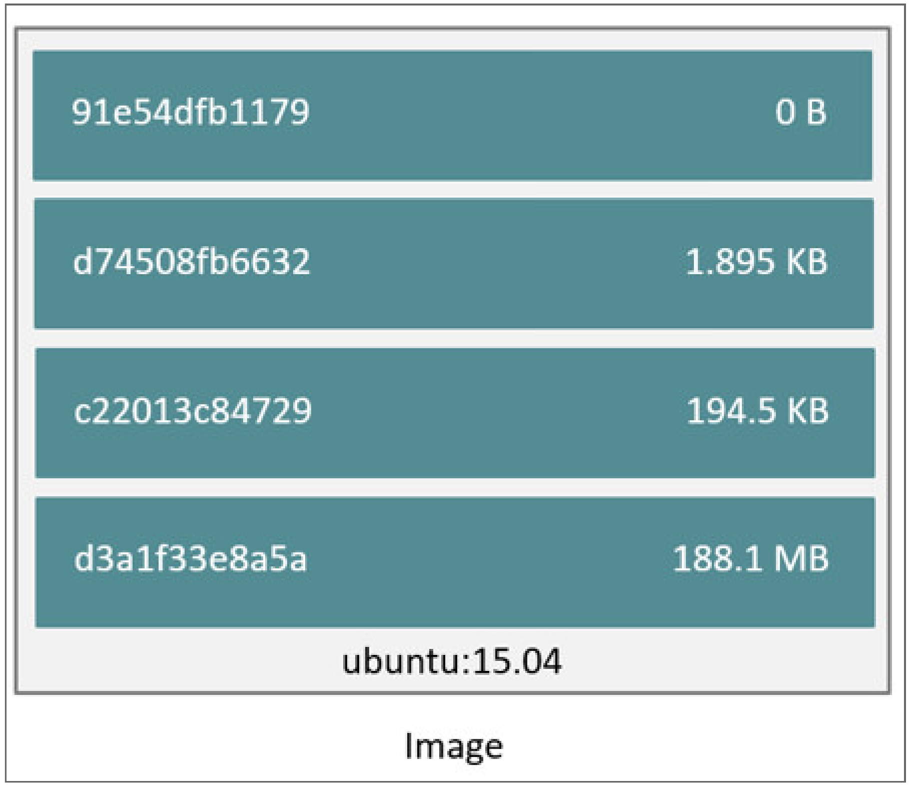
```
bash$ docker pull mariadb:latest
```

```
latest: Pulling from library/mariadb

5c90d4a2d1a8: Already exists
89e2627e9201: Already exists
76f6983b0fe0: Already exists
653577605512: Already exists
23e145a85462: Pull complete
ad4f74b37b82: Pull complete
ed3e1c3a2596: Pull complete
34c67c4a9ce5: Pull complete
a87a3e369167: Downloading [=========>            ] 30.767 MB/79.95 MB
692642483619: Download complete
1b4e355d86bb: Download complete
cd32d1285550: Verifying Checksum
```

```
Digest: sha256:358f6b50afd9c25707e97869f0c57de802c53973a90a2ff49e283501fccce1b2
Status: Downloaded newer image for mariadb:latest
```

USE DOCKER: IMAGE LAYERS



USE DOCKER: LIST IMAGES

```
bash$ docker images
```

```
REPOSITORY    TAG      IMAGE ID       CREATED          SIZE
mariadb       latest   854370756011   48 minutes ago   619.9 MB
<none>        <none>   3ffe2ccb2724   54 minutes ago   447.9 MB
```

```
bash$ docker build --tag=apache .
```

```
bash$ docker tag 3ffe2ccb2724 apache
```

```
REPOSITORY    TAG      IMAGE ID       CREATED          SIZE
apache        latest   3ffe2ccb2724   54 minutes ago   447.9 MB
```

USE DOCKER

CREATE CONTAINER

```
bash$ docker run \
  --name my-db \
  -e MYSQL_ROOT_PASSWORD=schneckerl \
  mariadb:latest
```

```
...
2016-07-18 21:59:31 139736693999552 [Note] mysqld: ready for connections.
```

LIST RUNNING CONTAINERS

```
bash$ docker ps
```

```
CONTAINER ID  IMAGE           CREATED             STATUS             PORTS     NAMES
102733de1735  mariadb:latest  About a minute ago  Up About a minute  3306/tcp  my-db
```

LIST ALL CONTAINERS

```
bash$ docker ps -a
```

USE DOCKER

STOP CONTAINER

```
bash$ docker stop my-db
```

RE-RUN CONTAINER

```
bash$ docker start my-db
```

ATTACH TO CONTAINER

```
bash$ docker exec -it my-db /bin/bash
[root@102733de1735 /]#
```

USE DOCKER

CONNECT TO CONTAINER

```
bash$ docker inspect \
    --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' \
    my-db
172.17.0.2

# in former versions:
# docker inspect --format '{{ .NetworkSettings.IPAddress }}' my-db

bash$ mysql -h 172.17.0.2 -u root -p
```

USE DOCKER

IMPROVE BY USING PORT MAPPINGS

```
bash$ docker run --name my-db \
    -p 3309:3306 \
    -e MYSQL_ROOT_PASSWORD=schneckerl \
    my-db
```

```
bash$ docker ps
...    PORTS                      ....
...    0.0.0.0:3309->3306/tcp     ....
```

```
bash$ mysql -h 127.0.0.1 -u root -p --port=3309
```

USE DOCKER

IMPROVE BY USING DATA VOLUMES

i.e. mapping local folders into containers

```
bash$ docker run --name my-db \
    -v /home/robert/mysql-data:/var/lib/mysql \
    -p 3309:3306 \
    -e MYSQL_ROOT_PASSWORD=schneckerl \
    mariadb
```

```
Dockerfile:

...
VOLUME /var/lib/mysql
...
```

DOCKER PHILOSOPHY

- Repeatability & Portability
- One service per container
- Never build panacea-like (“eierlegenden Wollmilchsäue”) containers
- Unix philosophy: do one thing and do it well
  - Docker promotes a micro-service architecture
  - Docker promotes re-use

THE AWESOME PART - DOCKER COMPOSE

The glue that eases micro-service architectures

Docker-Compose is an orchestration tool

php7/docker-compose.yml:

```
fpm:
    build: ./docker/phpfpm
    expose:
        - 9000
    ports:
        - 9000:9000
    links:
        - redis
    volumes:
        - /var/www:/var/www
    environment:
        - HOSTNAME=$ORIGIN
    restart: always

redis:
    image: redis:3.0
    restart: always
```

THE AWESOME PART - DOCKER COMPOSE

```
bash php7/$ docker-compose build
bash php7/$ docker-compose up
bash php7/$ docker-compose ps
bash php7/$ docker-compose down
bash php7/$ docker-compose rm
```

- **Container creation**
  - Naming by default *<folder>_<name>_1*, e.g.
    - php7_fpm_1
    - php7_redis_1
- **Linking**
- **Storage**

DATA STORAGE POSSIBILITIES

- **Data Volume**

  ```
  bash$ ... -v HOST_DIR:CONTAINER_DIR
  ```

  - Use for local data - development
- **Data Container**
  - Exits immediately
  - Defined within `docker-compose.yml`
  - Use for shared data from host
- **Named Volume**
  - New concept - Docker reserves space somewhere
  - Persistent storage - not deleted on `docker-compose rm`
  - Use for SQL data / Composer cache directory (no host directory mount)

DOCKER-REGISTRY

- A repository for images

DOCKER HUB

https://hub.docker.com

- Publicly available
- Explore
  - Lots of official images
  - Dockerfile
  - Starting instructions

PRIVATE REGISTRY

https://hub.docker.com/_/registry/

```
bash$ docker run -p 5000:5000 -v /home/robert/registry:/tmp/registry-dev registry
```

LIST IMAGES

```
bash$ curl -X GET your-repo-host:5000/v2/_catalog
```

```
{
    "repositories":[
        "e2/base",
        "e2/database",
        "e2/phpfpm54",
        "e2/phpfpm70"
    ]
}
```

FETCH IMAGE

```
bash$ docker pull e2/database
```

PUSH IMAGE

```
bash$ docker push e2/database
```

DEMO TIME

- Startup Apache container via:

  ```
  bash$ docker run apache
  ```

  - Show container-name & -id with `docker-ps`, restart with --name (rm ctr)
  - Show IP and Web
  - Connect via IP:

    ```
    bash$ docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' apac
    ```

- Run `docker-compose up` on Slim/Laravel projects
  - Show `docker ps`
  - Explain linking
  - Show `docker-compose.yml`

DOCKER ECOSYSTEM IN USE

- **PHP7**
  - Custom PHP7-FPM (`php:7.0.8-fpm`)
  - Redis (`redis:3.0`)
  - Some Apache virtual hosts proxies their files to `127.0.0.1:9000`
- **Continuous Integration**
  - Custom Jenkins (`jenkins`)
  - Custom Jenkins Slave based on PHP 7.0 FPM
  - Custom Jenkins Slave based on PHP 5.4 FPM
  - Custom MariaDB (`centos:7.2.1511`)
  - Selenium (`selenium/standalone-chrome-debug`)
  - MongoDB (`mongo`)
- **Special Service**
  - Custom Java (`java:8-jre`)
  - MongoDB (`mongo`)

LESSONS LEARNED

- NAT (port mapping) is provided by non-persisted iptables-rules
- Hostname is different, *127.0.0.1* is different
- Use private registries
- User within container has UID=1000 per default
- Use *.user.ini* for custom php settings
    - `php_value` within *.htaccess* does not work
    - `ini_set` may not work as well
- Docker images never shrink, after they grew
    - Keep number of layers low by combining commands
- Docker-ecosystem changes continuously
    - Compose, Engine, Machine, Toolbox, Swarm, Cloud, ...)
    - Syntax changes

OFFICIAL BEST PRACTICES

https://docs.docker.com/v1.11/engine/userguide/eng-image/dockerfile_best-practices/

WHY DOCKER FOR DEVELOPMENT

- Automatic setup - no manual steps
- Write once, run everywhere
- Never miss required tooling
  - lockrun
  - extra php modules
  - image crusher tools
  - ant
  - locales
  - ...

NO MORE

“does not work on my machine”

THE END

```
#!/bin/bash

# delete all containers
docker rm $(docker ps -a -q)

# delete all images or
docker rmi $(docker images -q)
s
# just delete dangling images
docker images --quiet --filter=dangling=true | xargs --no-run-if-empty docker rmi
```