DOCKERIZED WEB DEVELOPMENT

- LOCAL STACKS ARE SO 80S -

Holger Lösken // Sächsische Dampfschiffahrt

WHY THE F*** SHOULD I NEED THAT?

Many different projects with:

- different software
- different software versions
- different development environments (OSes)

Goals:

- Use your fav tools across all different projs and envs
- Exchange parts more easily

... and all with DOCKER

- Linux: native

```
$ dnf install docker // Red Hat based
$ apt-get install docker // Debian based
```

- Mac: Alpine* Linux container stable since July 2016

  DMG installer:
  `https://docs.docker.com/docker-for-mac/`

- Windows: Alpine* Linux container stable since July 2016

  MSI installer:
  `https://docs.docker.com/docker-for-windows/`

* Alpine Linux is a security-oriented, lightweight Linux distribution based on musl libc and

IMAGES AND CONTAINERS

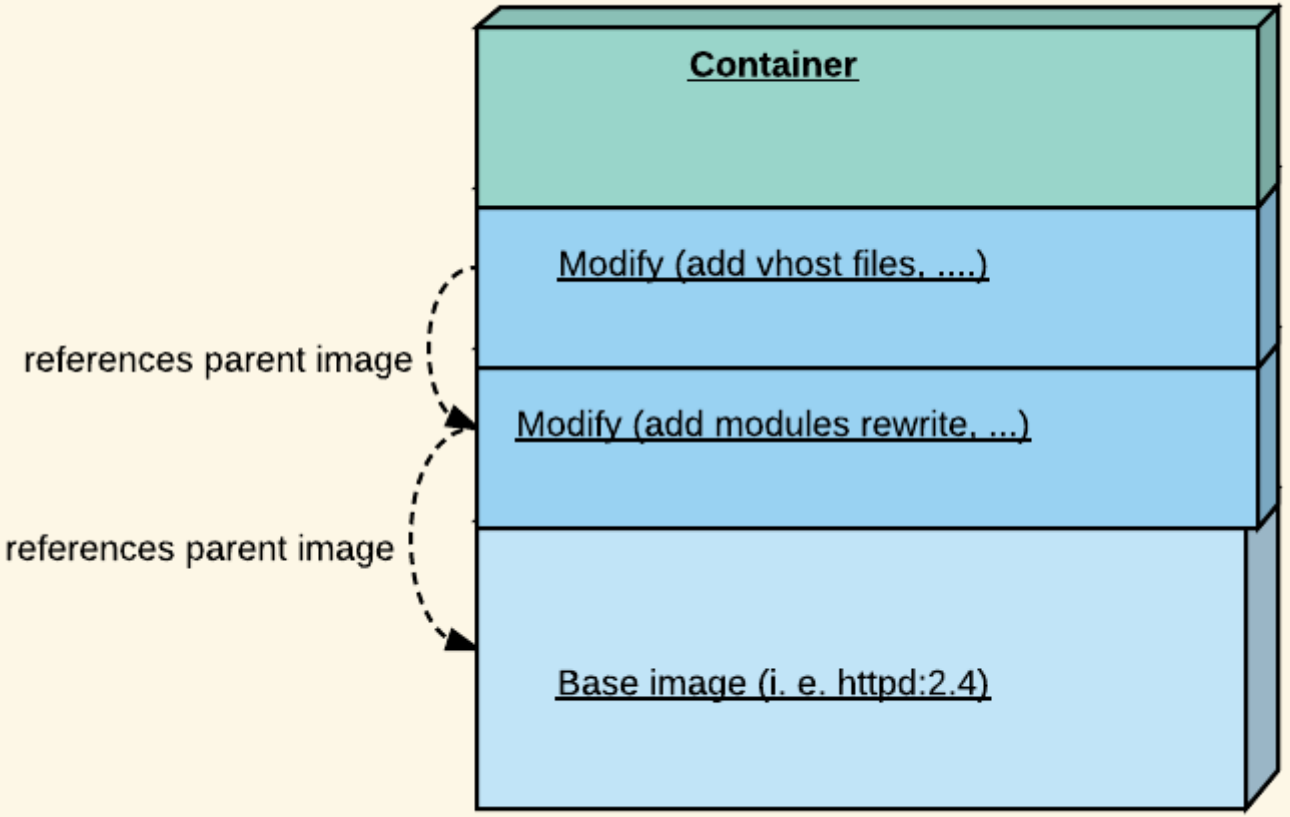
Images

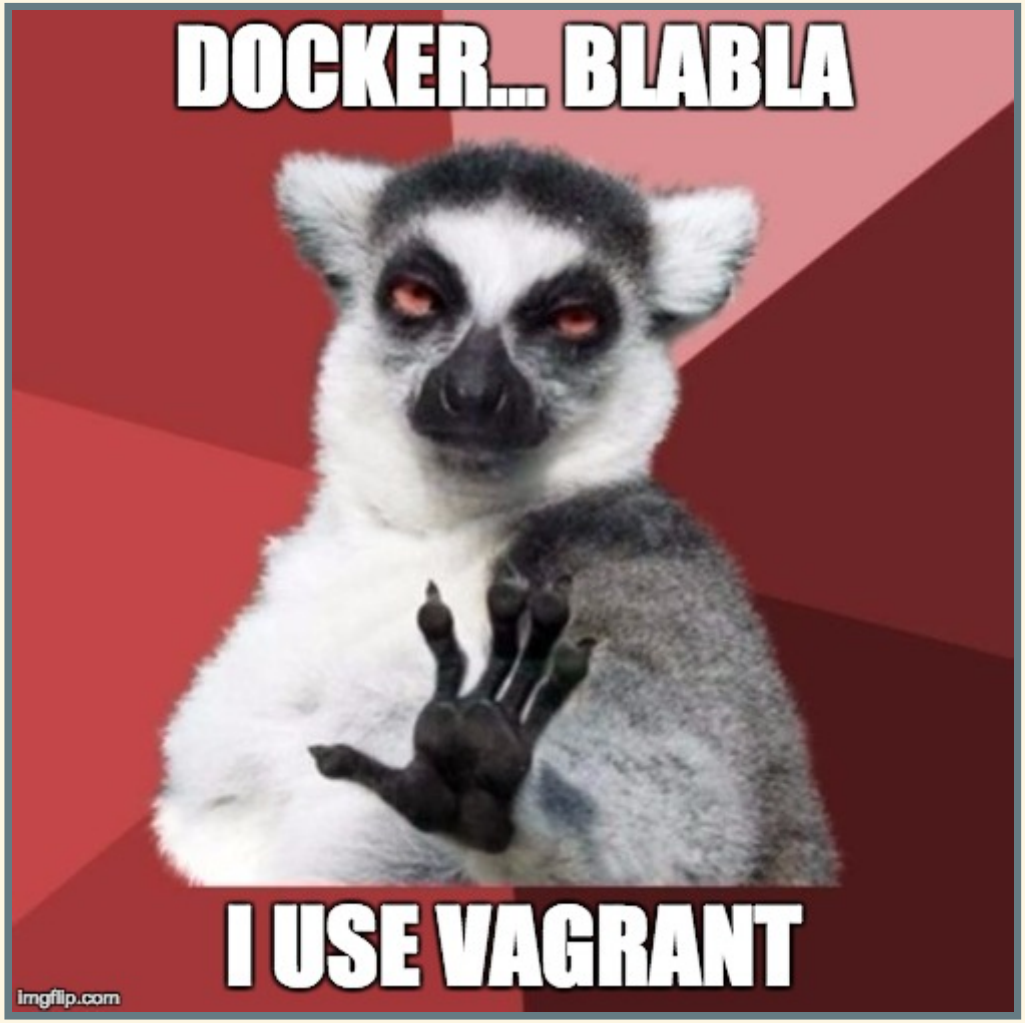
- Inert, immutable file; snapshot of a container
- Produce a container when started with `run`
- images are designed to be composed of layers of other images -> miminal amount of data

Containers

- Programming metaphor: if an image is a class, then a container is an instance of a class - a runtime object
- lightweight, portable encapsulations of an environment in which to run applications

- Base image, Dockerhub
- Each change can be a new image, local or on Dockerhub



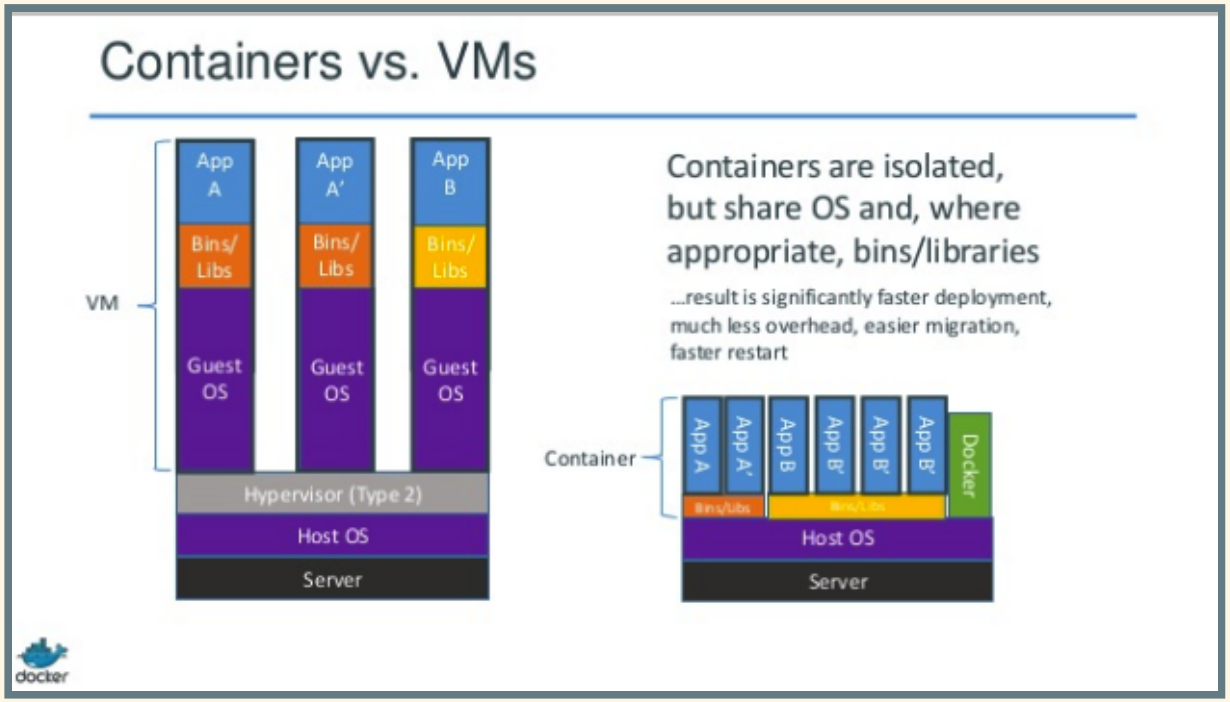

DOCKER... BLABLA
I USE VAGRANT
imgflip.com

DOCKER > VAGRANT > LOCAL STACK

WHAT MAKES THE DIFFERENCE...?

Virtualization

- Docker: on **service** level
- Vagrant: on **operating system** level
- local: on ... **no virtualization** ¯\(ツ)/¯



Basic differences

	Docker	Vagrant
OS	Mac,Win,Linux	Mac,Win,Linux
Starting time	Seconds	Minutes
Building image time	Short	Long
Size	100M+	1G+
Config file	Dockerfile (+ orchestr.)	Vagrantfile
Public images	Docker Hub	Vagrant Cloud

Docker: start virtualized Apache webserver

- Download image `httpd` (177 MB)
- Run (in seconds)

```
$ docker run -dit --name my-apache-app httpd:2.4
```

Vagrant: start virtualized Apache webserver

- Download image `ubuntu/xenial64` (278 MB) + Apache
- Run (in minutes)

```
$ vagrant init ubuntu/xenial64
```

See something?

- Docker: Apache (177 MB)
- Vagrant: Apache (289 MB +++)
- Docker: Run in minutes
- Vagrant: Run in seconds

???

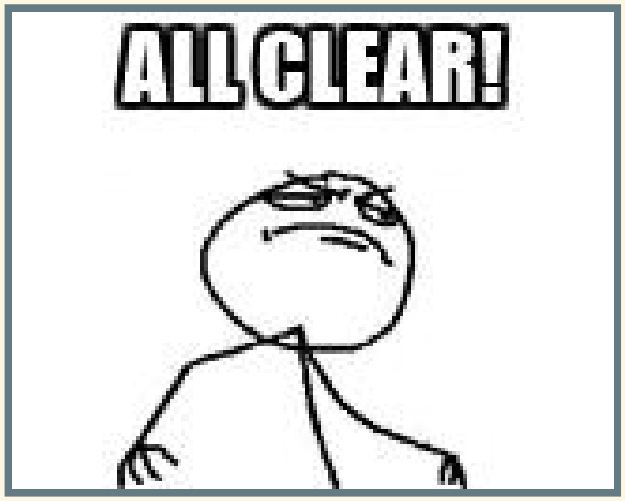
RECAP #1

WHAT YOU ALREADY LEARNED IN THE LAST COUPLE MINUTES ;-)

... and enough for the Vagrant bashing

What we learned until know:

- Why I might need a fully virtualizes setup with Docker
- What is Docker and (most important) differences to Vagrant
- What are Docker images and containers and where to get them



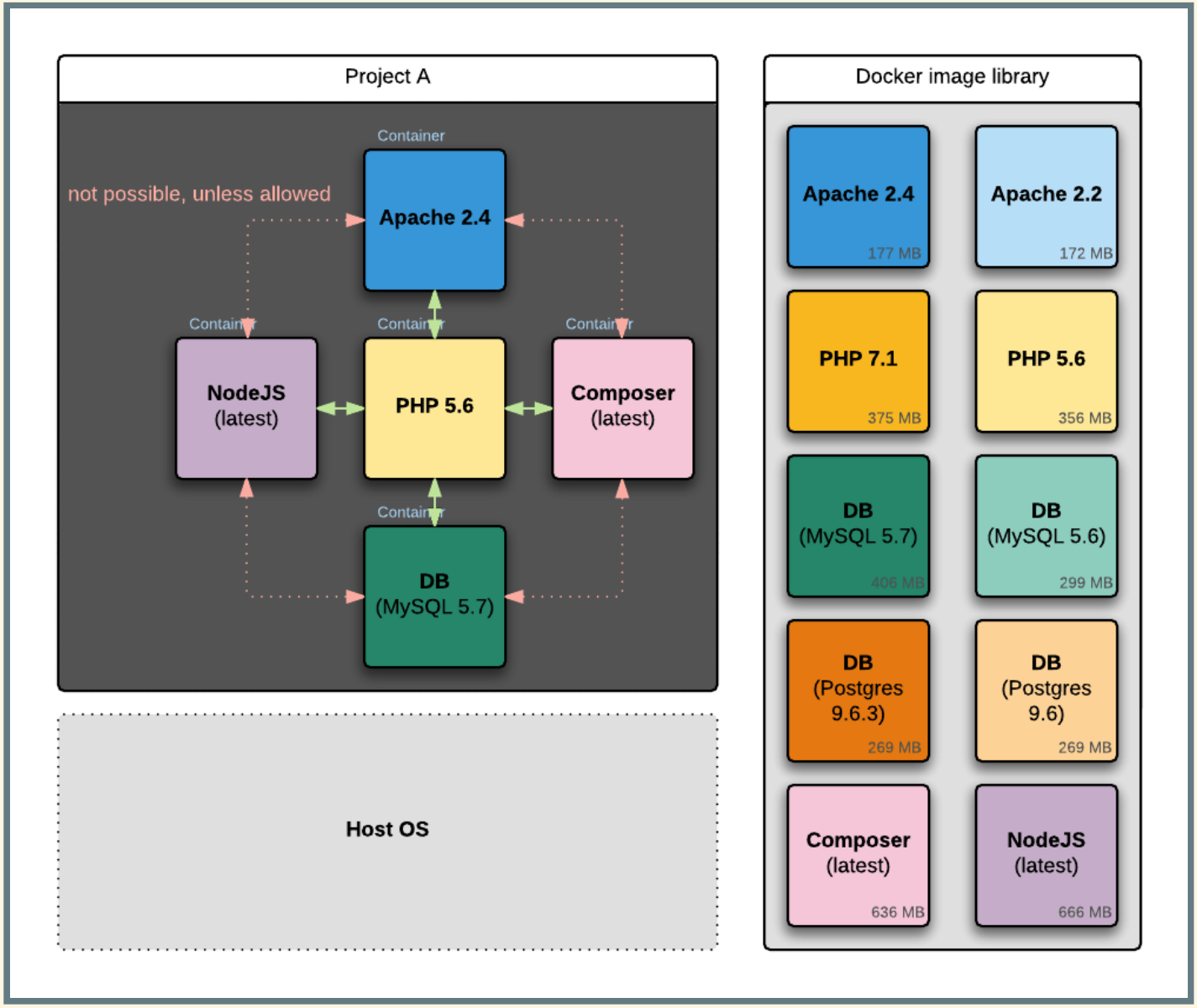
DOCKER IN WEB DEV

DOCKERIZE YOUR DEVELOPMENT ENVIRONMENT

Typical web dev project:

- Webserver, i. e. Apache
- Some language for the backend, i. e. PHP
- A database, i. e. MySQL
- Another language for the frontend, i. e. (Node)JS
- Handy tooling, i. e. Composer

... should be exchangeable in software and version used


Project A
Container
not possible, unless allowed
Apache 2.4
NodeJS
(latest)
PHP 5.6
Composer
(latest)
DB
(MySQL 5.7)
Host OS
Docker image library
Apache 2.4
177 MB
Apache 2.2
172 MB
PHP 7.1
375 MB
PHP 5.6
356 MB
DB
(MySQL 5.7)
406 MB
DB
(MySQL 5.6)
299 MB
DB
(Postgres
9.6.3)
269 MB
DB
(Postgres
9.6)
269 MB
Composer
(latest)
636 MB
NodeJS
(latest)
666 MB

Apache - `apache.dockerfile`

```
# Pull the exact version
FROM httpd:2.4
```

PHP `php.dockerfile`

```
FROM php:5.6
```

DB `mysql.dockerfile`

```
FROM mysql:5.7
```

Composer `composer.dockerfile`

```
# Pull the latest version available
FROM composer
```

NodeJS `nodejs.dockerfile`

```
FROM node
```

Tip: Use separate "Dockerfiles"!

Apache - `apache.dockerfile`

- Add `rewrite` and `fcgi` modules
- Set env `APACHE_LOG_DIR`
- Add your own vhost configuration

```
FROM httpd:2.4

RUN sed -i 's|#LoadModule rewrite_module|LoadModule rewrite_modul
    sed -i 's|#LoadModule proxy_module|LoadModule proxy_module|'
    sed -i 's|#LoadModule proxy_fcgi_module|LoadModule proxy_fcgi
    sed -i 's|#Include conf/extra/httpd-vhosts.conf|Include conf/

# Replace default with my vhost, don't forget edit hosts file!!
RUN echo '' > /usr/local/apache2/conf/extra/httpd-vhosts.conf
COPY my-vhost.conf:/usr/local/apache2/conf/extra/httpd-vhosts.con

ENV APACHE_LOG_DIR=/usr/local/apache2/logs

RUN apachectl restart
```

Tip: Insert `ServerName` from vhost into local `hosts` file

PHP - `php.dockerfile`

- Update sources + install software
- Install + enable PHP modules
- Create directory for source code

```
FROM php:5.6-fpm

RUN apt-get -y update
RUN apt-get -y install --no-install-recommends \
    libcurl4-gnutls-dev \
    libpng-dev \
    libjpeg-dev \
    libmcrypt-dev \
    libicu-dev

RUN docker-php-ext-install \
    curl \
    json \
    pdo \
    pdo_mysql \
    mbstring \
```

Make 'em fly...

Build the image

```
$ docker build -f php.dockerfile -t php56 .
$ docker build -f apache.dockerfile -t apache24 .
```

Run the container

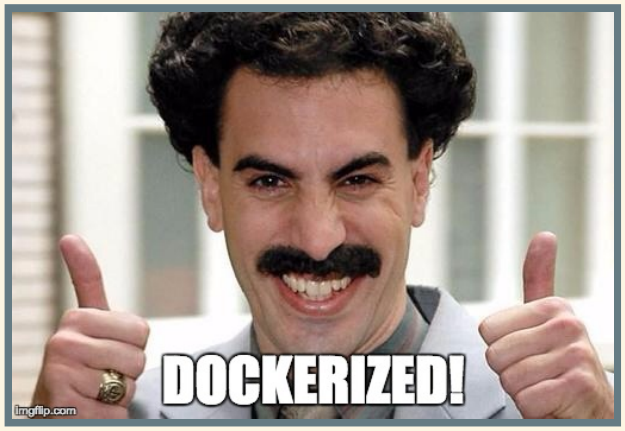
```
$ docker run -dit --name=project_a_php56 php56

$ docker run -dit -p 80:80 -p 443:443 \
      -v ~/test/src:/usr/local/apache2/htdocs \
      --link=project_a_php56 \
      --name=project_a_server apache24
```

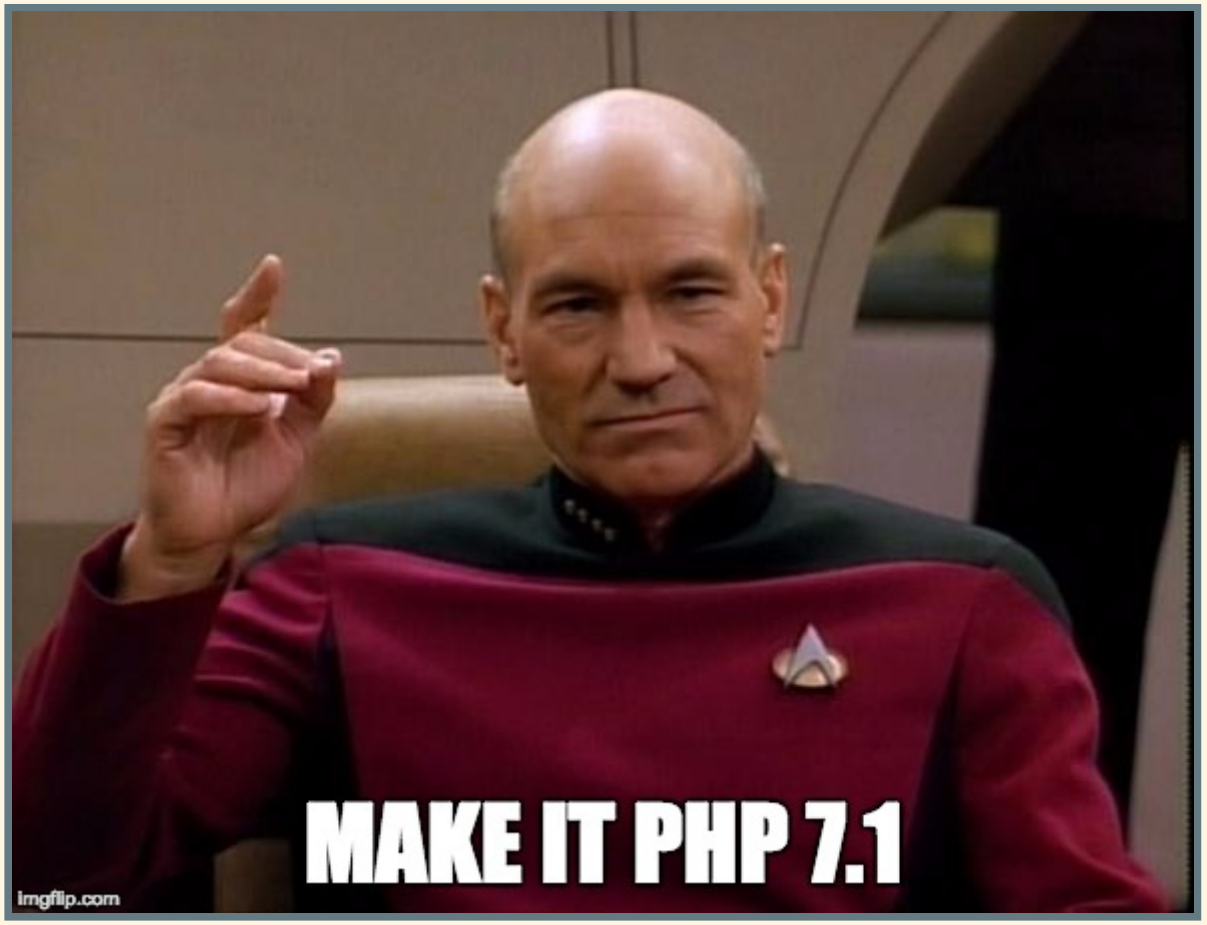
- Map ports VM:HOST
- Mount code directory `src` into the container
- Link Apache to PHP container

SUCCESS!

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND
a11089d64193        apache24            "httpd-foreground"
bffc05af1e5a        php56               "docker-php-entryp..."
```

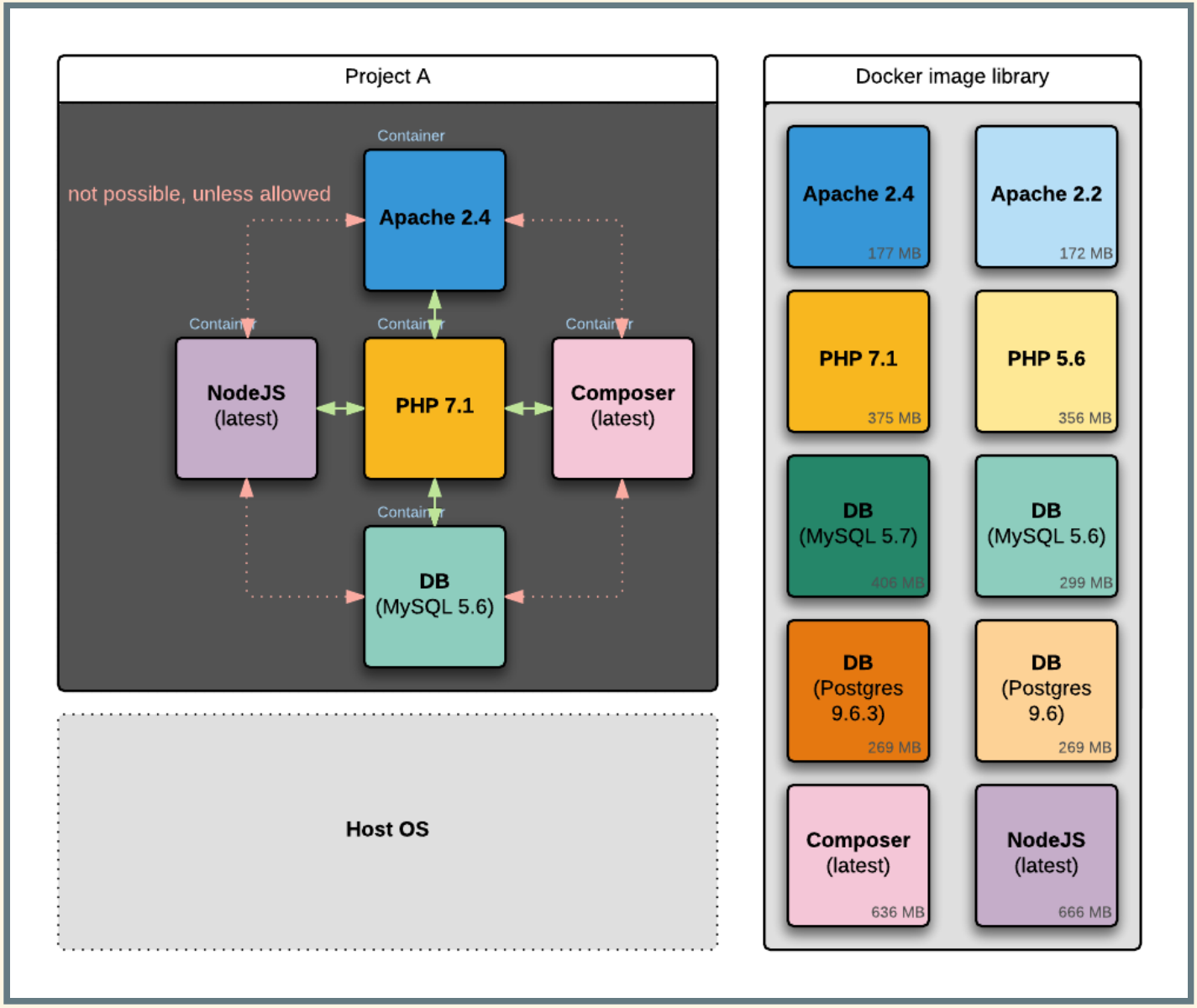


GUY WITH THE BIGGER CHAIR SAYS:



BEFORE DOCKER...

BUT NOW, AFTER A COUPLE OF MINUTES...


Project A
Container
not possible, unless allowed
Apache 2.4
NodeJS
(latest)
PHP 7.1
Composer
(latest)
DB
(MySQL 5.6)
Host OS
Docker image library
Apache 2.4
177 MB
Apache 2.2
172 MB
PHP 7.1
375 MB
PHP 5.6
356 MB
DB
(MySQL 5.7)
406 MB
DB
(MySQL 5.6)
299 MB
DB
(Postgres
9.6.3)
269 MB
DB
(Postgres
9.6)
269 MB
Composer
(latest)
636 MB
NodeJS
(latest)
666 MB

Apache `Dockerfile`

... stays the same

PHP `Dockerfile`

```
FROM php:7.1
```

DB `Dockerfile`

```
FROM mysql:5.6
```

Composer `Dockerfile`

... stays the same

NodeJS `Dockerfile`

... stays the same

Make 'em fly... second time

Build the image & run the container

```
$ docker build -f php.dockerfile -t php71 .

$ docker run -dit --name=project_a_php71 php71
```

Run the apache container

```
$ docker stop project_a_server && docker rm -f project_a_server
$ docker run -dit -p 80:80 -p 443:443 \
      -v ~/test/src:/usr/local/apache2/htdocs \
      --link=project_a_php71 \
      --name=project_a_server apache24
```

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  
e42fa8540220        apache24            "httpd-foreground"       
ee89b5b15d20        php71               "docker-php-entryp..."   
6523b463ee5b        php56               "docker-php-entryp..."   
```


DATABASE?... PERSIST?
WHERE?
imgflip.com

PERSISTENCE

Data volumes

- Marked directory inside of a container, exists to hold persistent or commonly shared data
- Connected to multiple containers
- Saved on host (`/var/lib/docker/volumes/...`)
- Persist when container destroyed

DATA VOLUME ISSUES

- Security
  No extra layer, just plain ro/rw privileges
- Data integrity
  Data containers provides no data integrity protection
- External storage
  Docker volume spanning from one host to another, not possible across multiple hosts - not verfied (sorry ;-)

SHORT RECAP #2

DOCKERFILE(S) + DOCKER COMMANDS...

RECAP #2

- Dockerfile or [service].dockerfile for your own images
- Persistence
- Commands

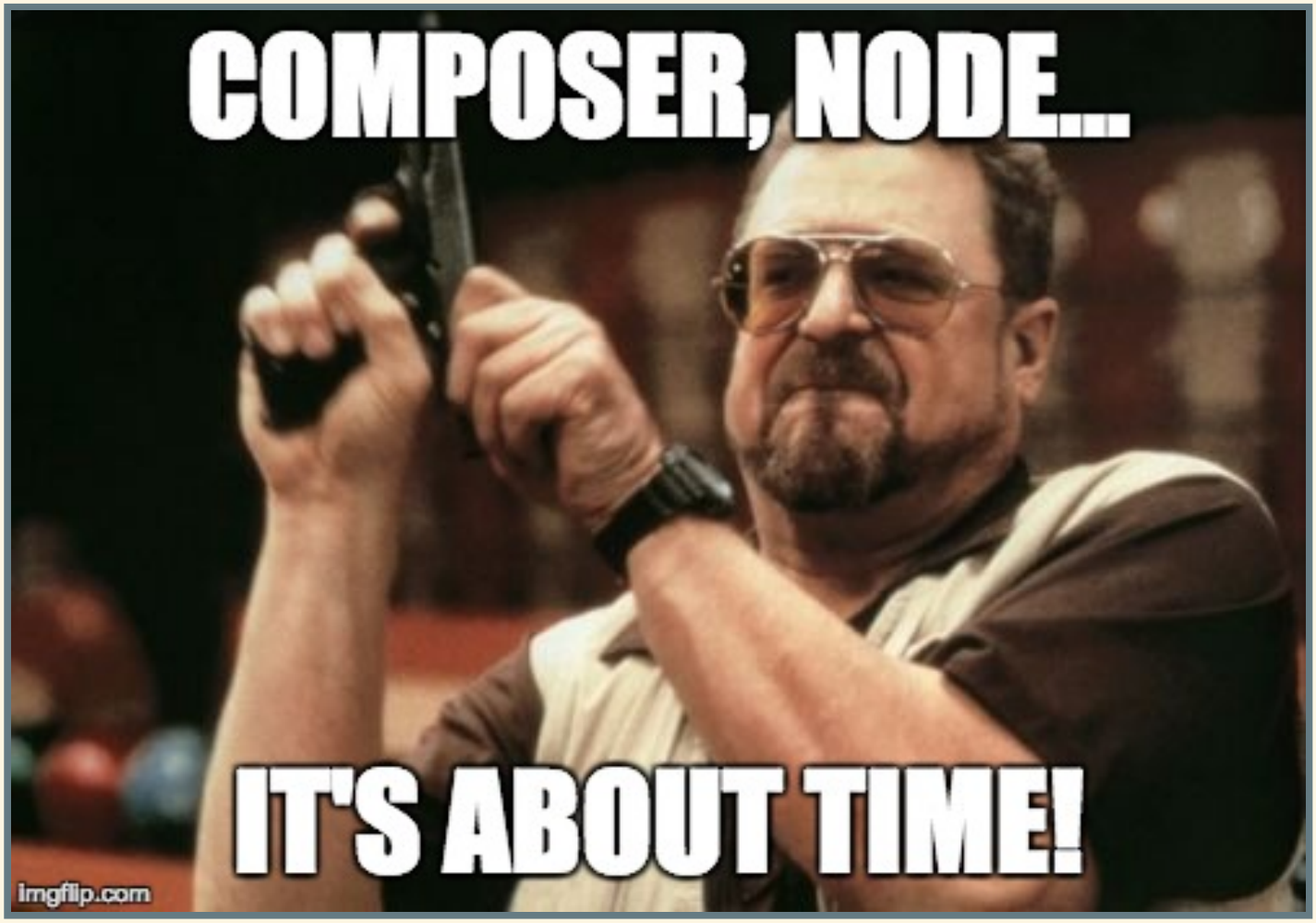
```
# Build image from Dockerfile
$ docker build [-f DOCKERFILE_PATH] -t IMAGE_TAG .
```

```
# Run a container based on an image
# [d]etached, [i]nteractive, allocate pseudo-[t]ty
$ docker run -dit [--volume HOST_PATH:CONTAINER_PATH] [--port HOS
```

```
# Show running containers and some info
$ docker ps
```

```
# Detailed info on container, i. e. links
$ docker inspect CONTAINERNAME
```

(LAMP) + COMPOSER AND NODE


COMPOSER, NODE...
IT'S ABOUT TIME!
imgflip.com

- Composer and Node not needed as foreground process (in PHP devel)
- Instant creation and execution when needed
- No coupling to running containers

=> Shell is our friend...

NODE IN DOCKER CONTAINER

```
npm () {
    tty=
    tty -s && tty=--tty
    docker run \
           $tty \
           --interactive \
           --rm \
           --user $(id -u):$(id -g) \
           --volume /etc/passwd:/etc/passwd:ro \
           --volume /etc/group:/etc/group:ro \
           --volume $HOME:/home/$USER \
           --volume $(pwd):/usr/src/ \
           node npm --prefix=/usr/src "$@"
}
```

COMPOSER IN DOCKER CONTAINER

```
composer () {
      tty=
      tty -s && tty=--tty
      docker run \
             $tty \
             --interactive \
             --rm \
             --user $(id -u):$(id -g) \
             --volume  ~/.composer:/composer \
             --volume /etc/passwd:/etc/passwd:ro \
             --volume /etc/group:/etc/group:ro \
             --volume $(pwd):/app \
             composer "$@"
}
```

NICE AND SHINY

ORCHESTRATION/COMPOSITION AND LESS COMMAND LINE...

ORCHESTRATION TOOLS OF DOCKER CONTAINER LIFECYCLES

- **Docker compose** (maintained by Docker core team)
- Dusty (uses Docker compose under the hood)
- Gockerize (Go-inspired)
- Crowdr (not developed anymore?)

DOCKER COMPOSE

- YAML file, `docker-compose.yml`
- All features that docker command line provides + extras
- One command to spin up X containers ready to use

```
// docker-compose.yml
version: '3'
services:
  server:
    ...
  php:
    ...
  database:
    ...
volumes:
  data:
```

READ FROM `.ENV FILE

```
// .env
DB_NAME=DUCKTALES
DB_USERNAME=donald
DB_PASSWORD=!123_AbCxYz
```

```
// docker-compose.yaml
version: '3'
services:
  ...
  database:
    image: mysql:5.7
    volumes:
      - ./src/:/var/www/html
      - data:/var/lib/mysql
    ports:
      - "3306:3306"
    environment:
      - "MYSQL_DATABASE=${DB_NAME}"
      - "MYSQL_USER=${DB_USERNAME}"
      - "MYSQL_PASSWORD=${DB_PASSWORD}"
      - "MYSQL_ROOT_PASSWORD=${DB_PASSWORD}"
```

SHORT SUMMARY

- One file, `YAML`
- Seamsless Docker integration, supports everything and slightly more
- No spacy cli paramters -> easier to read/understand

CODING TIME ...

TUTORIAL TO BUILD YOUR PROJECT W/ DOCKER

Questions so far...? Recap #3!!

THANKS!

I hope you'd fun! Questions...??

MAIL: post@codedge.de

TWITTER: cod2edge