Docker - eine Einführung für Softwareentwickler

Rainer Stropek | time cockpit

Agenda

Docker war in den Medien in den letzten Monaten eines der heißesten Themen. Spätestens seit Microsoft Docker in Azure unterstützt und Docker auch für Windows angekündigt wurde, haben sich viele Entwickler auf dieser Plattform gefragt, ob die Containertechnologie für sie relevant ist.

Rainer Stropek (Azure MVP) möchte in dieser Session zeigen, warum das der Fall ist. Er erläutert kurz die Grundlagen von Docker und demonstriert danach die Funktionsweise an einem durchgehenden Beispiel. Wir arbeiten uns bis zum Betrieb von ASP.NET-vNext-Anwendungen in Docker-Containern vor. Anhand des Beispiel lernen Sie Docker kennen und können danach einschätzen, welche Auswirkungen Docker-Container auf ihre zukünftige Arbeit haben werden.

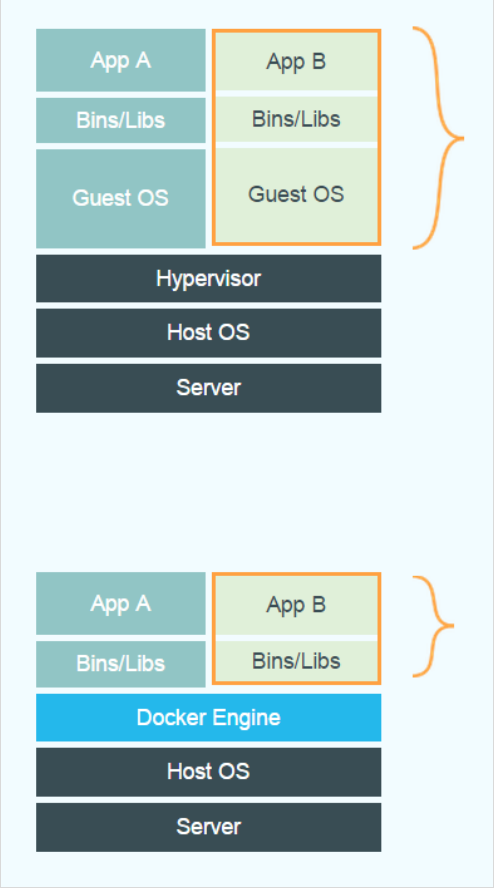
Your Host

Rainer Stropek

Developer, Entrepreneur
Azure MVP, MS Regional Director
Trainer bei IT-Visions

Contact

software architects gmbh
rainer@timecockpit.com
Twitter: @rstropek



What is Docker?

Virtual machines vs. Docker

Each VM runs its own guest operating system

Container reuse the host operating system

Container run in user space

https://www.docker.com/whatisdocker/

What's Docker?

Container virtualization

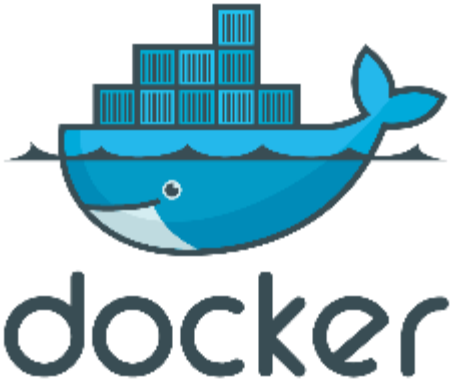
Container run in user space and use kernel of host
Has been existing in Linux for quite a while
Docker builds on Linux Containers (LXC) and makes it easy to use and consume

Advantages?

Fast, small, and agile (e.g. Docker in Docker)

Disadvantages?

Security (less isolated)



What's Docker?

Command line tool, REST services
Docker client can manage remote Docker daemon

Container packaging format

Dockerfiles for image creation from source code

Version management for images
Images can be based on images

Docker Hub: Platform to exchange images and Dockerfiles
Publishing on Docker Hub is not in scope of this talk

Docker in Windows

Boot2Docker
Run lightweight Linux in VirtualBox

Compile Docker client on Windows
Written in GO

Container virtualization in Windows
Announced for next version of Windows Server

Use Azure to play with Docker
Existing VM image (Docker on Ubuntu server) in Azure marketplace
Use Docker container to run Azure tools (e.g. https://registry.hub.docker.com/u/kmouss/azure-cli/)

Demo

Docker in Azure

- Create Ubuntu server with Docker in Microsoft Azure
  - Using the Azure portal
  - Using Azure XPlat tools
- Connect to Docker daemon remotely

```
// Connect to Docker client in Azure
// (see also https://github.com/rstropek/DockerVS2015Intro)

// Try to connect to remote docker daemon
docker --tls=true \
  -H tcp://dockersamplehost.cloudapp.net:4243 \
  info

// Try to start a docker container remotely
docker --tls -H tcp://dockersamplehost.cloudapp.net:4243 \
  run -i -t ubuntu /bin/bash

// Set environment variable to shorten command line
export DOCKER_HOST=tcp://dockersamplehost.cloudapp.net:4243
docker –tls info
```

Remote Docker

Container

Working with containers

Docker CLI

Documentation

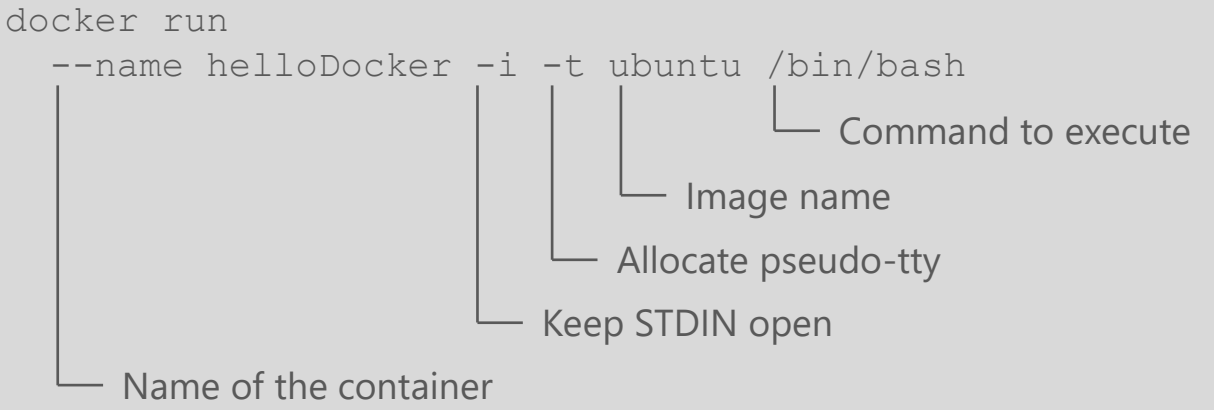
http://docs.docker.com/reference/commandline/cli

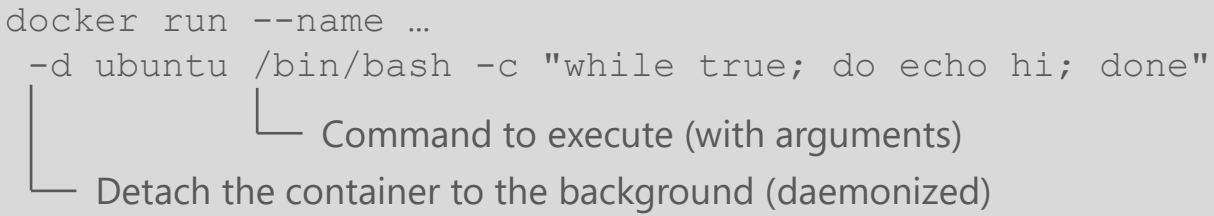
Important Commands for Containers

`docker run` – Run a command in a new container
`docker ps` – List containers
`docker start/stop` – Restarts/stops a container
`docker rm` – Removes container(s)
`docker attach` – Attach to running container
`docker top` – Display processes running in container
`docker exec` – Run a command in a container

Docker CLI

Starting Containers





Interactive container

Daemonized container

Running in the background

`--rm` removes container when it exits

```
# Check if docker is running
docker info

# Start interactive container
docker run --name helloDocker -i -t ubuntu /bin/bash
  echo Hello > helloTechorama.txt
  exit

# List containers
docker ps
docker ps –a
docker ps --no-trunc -aq

# Restart container
docker start helloDocker

# Attach to container
docker attach helloDocker

# Remove container
docker rm helloDocker
# Remove all containers
docker rm `docker ps --no-trunc -aq`
```

Demo

Interactive Container

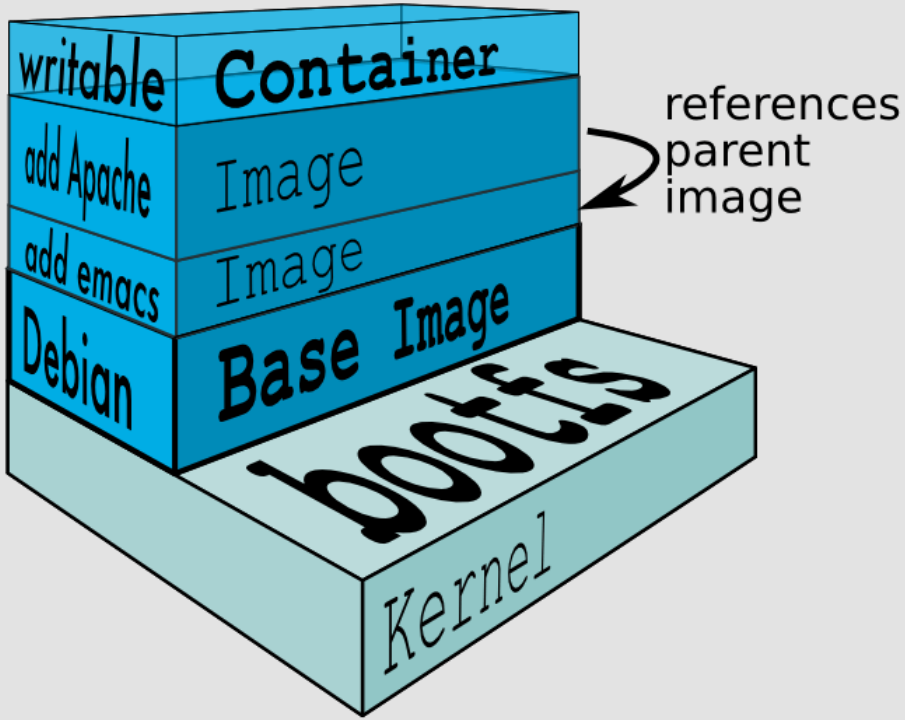
```
# Start demonized container and get logs
docker run --name backgroundContainer -d ubuntu /bin/bash \
  -c "while true; do echo hello world; sleep 1; done"

# Get the logs (-f for continuous monitoring)
docker logs backgroundContainer

# Check the processes in docker container
docker top backgroundContainer

# Open interactive shell in running container
docker exec -i -t backgroundContainer /bin/bash
```

Demo

Daemonized Container

Images

Working with images



File System Layers

Rootfs stays read-only

Union-mount file system
- over the read-only file system
  - Multiple file systems stacked on top of each other

Only top-most file system is writable
- Copy-on-write

Docker CLI

Important Commands for Images

`docker images` – List all images
`docker search` – Search for image on Docker Hub
`docker pull` – Pulls an image from the registry (Docker Hub)
`docker commit` – Create image from container
`docker inspect` – Get low-level information on container or image

Docker CLI

Building Images from Containers

```
docker commit
  -m="Techorama image" --author="Rainer Stropek"
```

Message

Author of the image

```
  templateContainer rstropek/ubuntu:withFile
```

Target repository:tag

Name of the container

```
# Start interactive container
docker run --name templateContainer -i -t ubuntu /bin/bash
  echo "Hello Techorama!" > helloWorld.txt
  exit

# Build image from container
docker commit -m="Techorama image" --author="Rainer" \
  templateContainer rstropek/ubuntu:withFile

# Remove container
docker rm -f templateContainer

# Create new container from new image
docker run --name newContainer -i -t rstropek/ubuntu:withFile \
  /bin/bash

# Remove image
docker rmi <image>

# Run DockerUI in container
# https://github.com/crosbymichael/dockerui
docker run -d -p 9000:9000 --privileged \
  -v /var/run/docker.sock:/var/run/docker.sock \
  dockerui/dockerui
```

Demo

Create Image

Dockerfiles

Creating images from source

Dockerfiles

```
# Version 0.0.1
FROM nginx
MAINTAINER Rainer Stropek "rainer@timecockpit.com"
ENV REFRESHED_AT 2014-02-22
RUN apt-get -qq update
```

Execute command in new layer on top of the image and commit the result

```
COPY *.html /usr/share/nginx/html/
```

Copy files to the filesystem of the container

```
docker build -t staticweb .
```

Tag for the image

Dockerfile location

Documentation

https://docs.docker.com/reference/builder/

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

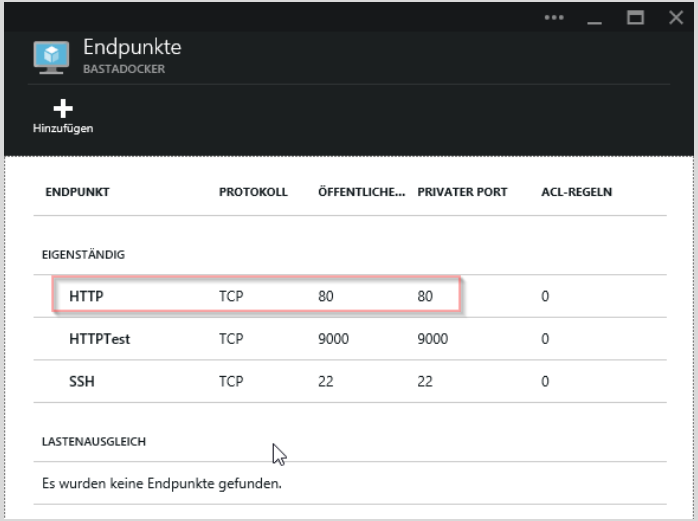
See Dockerfile for nginx

Docker CLI

Exposing ports

```
docker run --name staticwebcontainer \
  -d -p 80:80 staticweb
```

- Expose port 80
- Run daemonized



```
# Get sample code from GitHub
git clone https://github.com/rstropek/DockerVS2015Intro.git

# Build website
cd dockerDemos/01-staticWeb/app
npm install
grunt
cd ..

# Build image from Dockerfile
docker build -t staticweb .
docker run --name staticwebcontainer -d -p 80:80 staticweb

# Change website content and rebuild container

# Run a second container, run a third container (linked)
docker run -i -t --link <cont1>:sweb1 --link <cont2>:sweb2
ubuntu /bin/bash
  apt-get install curl
  curl http://sweb1
```

Demo

Dockerfile

```
# Run grunt inside a docker container
docker run --rm -v ~/DockerVS2015Intro/dockerDemos/01-
staticWeb/app:/data killercentury/nodejs-bower-grunt grunt


# Run daemonized grunt inside a docker container
docker run -d -v ~/DockerVS2015Intro/dockerDemos/01-
staticWeb/app:/data killercentury/nodejs-bower-grunt grunt
watch


# Run nginx webserver inside daemonized container
docker run -d -p 80:80 -v ~/DockerVS2015Intro/dockerDemos/01-
staticWeb/app:/usr/share/nginx/html nginx
```

Demo

Automated build

```
# Run grunt inside a docker container
docker run --rm
```

└ Remove the container when it exists

```
  -v ~/DockerVS2015Intro/dockerDemos/01-staticWeb/app:/data
```

└ Mount host volume (`host:container`)

```
  dockerfile/nodejs-bower-grunt
```

└ Use existing image

```
  grunt
```

└ Run `grunt`

Demo

Run Grunt (build) in Container

ASP.NET in Docker

Running ASP.NET in Docker

```
FROM microsoft/aspnet
MAINTAINER Rainer Stropek "rainer@timecockpit.com"
ENV REFRESHED_AT 2015-01-02

ENV SOURCE_DIR /app/src

RUN mkdir -p $SOURCE_DIR
WORKDIR $SOURCE_DIR

COPY refreshAndRunSample.sh $SOURCE_DIR/
RUN chmod a+x $SOURCE_DIR/refreshAndRunSample.sh

RUN apt-get -qqy install git
RUN git init \
 && git pull https://github.com/aspnet/Home.git \
 && cd  samples/HelloMvc/ \
 && kpm restore

ENTRYPOINT ["/app/src/refreshAndRunSample.sh"]
```

Dockerfile

Base image:
https://registry.hub.docker.com/u/microsoft/aspnet/

Run container

```
docker run -d -t
  -p 80:5004 myaspnet
```

Application Scenarios

Running continuous integration in containers

Rebuild complex runtime environment on my laptop

Identical environment for dev, test, and prod

Cost reduction in the cloud

High density hosting (e.g. multiple versions)

Split software into multiple, independent services

Micro-services, see Manfred's session tomorrow