

# TECHORAMA

MAY 12-13 2015



docker

...and why it is relevant for developers

# Agenda

This is a session for developers who heard about Docker and wonder why it would be relevant for them.

What is different about Docker? Why could it be useful for web developers? How can it be used on Azure?

This is a Docker intro session in which I demo the application of Docker in a web development scenario.

# Your Host

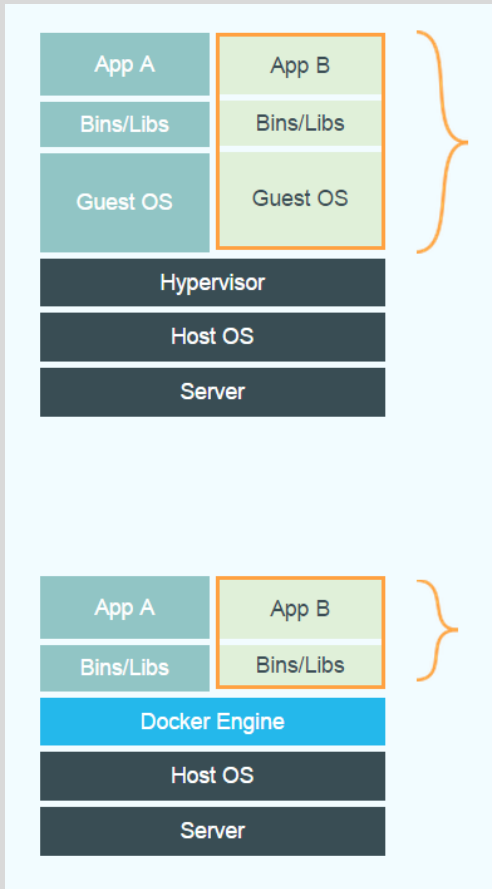
## Rainer Stropek

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

## Contact

software architects gmbh  
[rainer@timecockpit.com](mailto:rainer@timecockpit.com)  
Twitter: @rstropek





## Virtual Machines

## Docker Container

# What is Docker?

Virtual machines vs. Docker

Each VM runs its own guest operating system

Containers reuse the host operating system  
Containers run in user space

# What's Docker?

## Container virtualization

Container run in user space and use kernel of host

Has been existing in Linux for quite a while

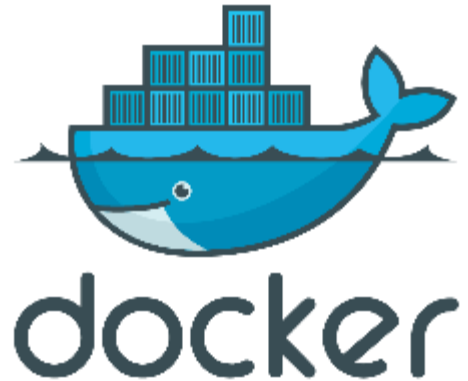
Docker builds on Linux Containers (LXC) and makes it easy to use and consume

## 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 run
```

```
--name helloDocker -i -t ubuntu /bin/bash
```

└── Name of the container

└── Keep STDIN open

└── Allocate pseudo-tty

└── Image name

└── Command to execute

```
docker run --name ...
```

```
-d ubuntu /bin/bash -c "while true; do echo hi; done"
```

└── Command to execute (with arguments)

└── Detach the container to the background (daemonized)

# 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

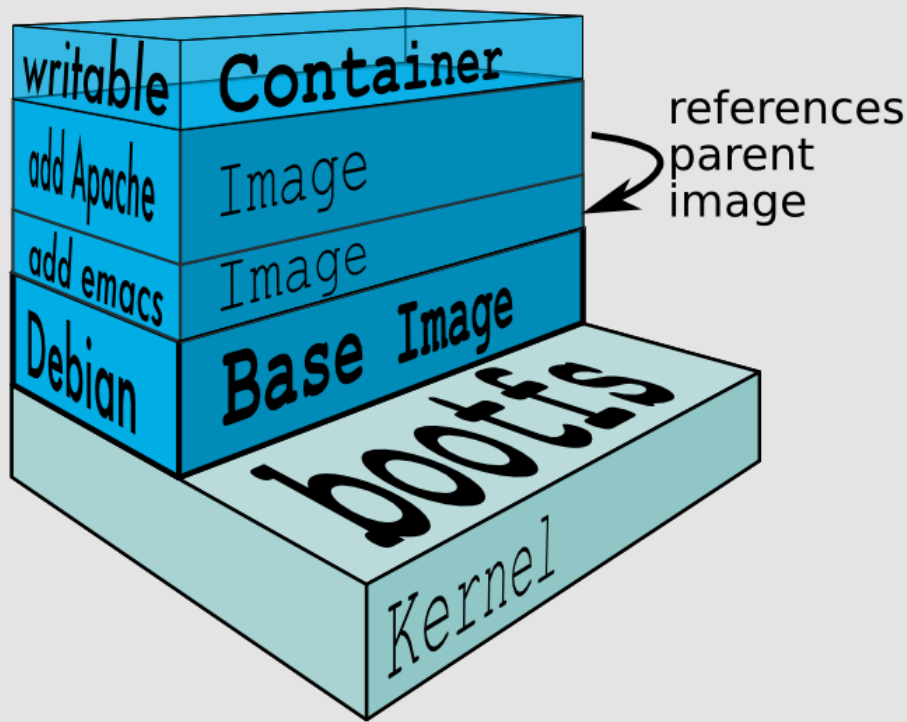
# Open interactive shell in running container
docker exec -i -t backgroundContainer /bin/bash
```

# 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"
```



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



```
# 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

```
# 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

# Dockerfiles

## Documentation

<https://docs.docker.com/reference/builder/>  
[https://registry.hub.docker.com/\\_/nginx/](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

ENDPUNKT	PROTOKOLL	ÖFFENTLICHE...	PRIVATER PORT	ACL-REGELN
HTTP	TCP	80	80	0
HTTPTest	TCP	9000	9000	0
SSH	TCP	22	22	0

LASTENAUSGLEICH

Es wurden keine Endpunkte gefunden.

```
# 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:/var/www/html dockerfile/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)

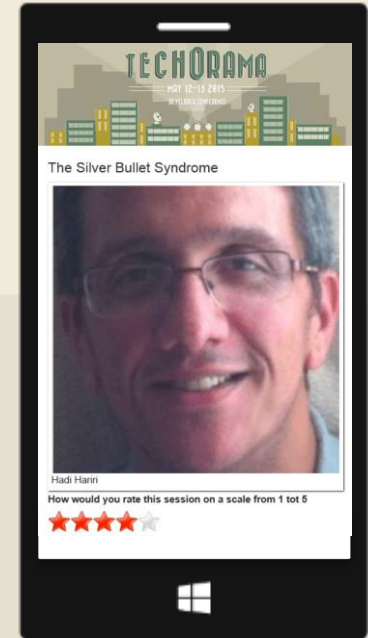
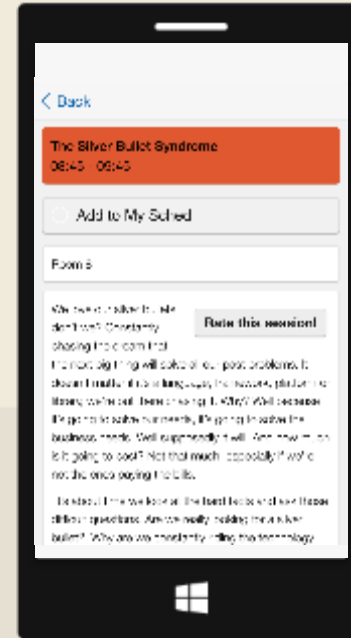
Split software into multiple, independent services

Micro-services, see Manfred's session tomorrow

# Provide your instant session feedback!

To help us in our journey to host the best developer conference in Europe, we want your feedback.

Rate all sessions via the Techorama schedule page!  
<http://techorama2015.sched.org>



More information on <http://www.techorama.be/techorama-2015/feedback/>

# A big thank you to our partners

## Platinum Partners



## Gold Partners



## Silver & Social Partners

