1.– 4. September 2014
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Container verschiffen

Continuous Delivery mit Docker

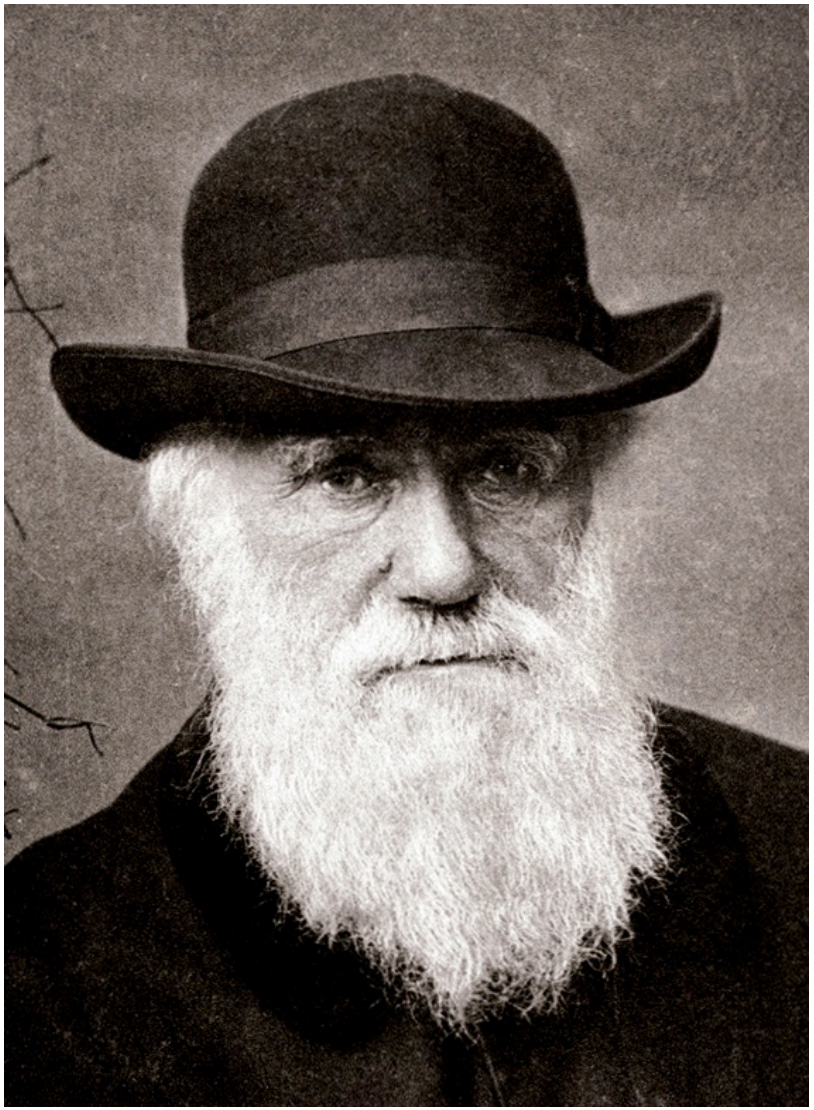
Dr. Halil-Cem Gürsoy

adesso AG

Der Speaker

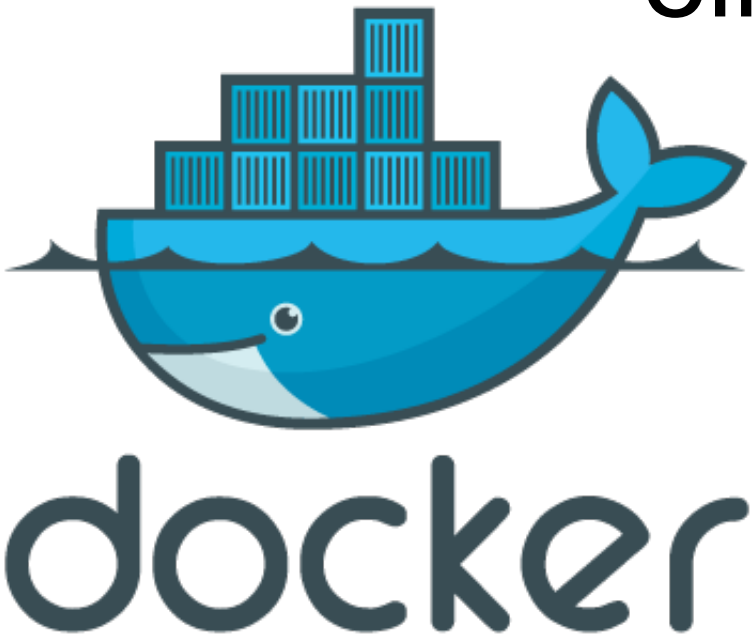
- Principal SW Architect @ adesso AG
- ~15 Software-Entwicklung
- Diverse Artikel / Speaker auf Konferenzen
- „Cloud“ / Verteilte System
  - Persistenz
  - Build & Deployment

Warum eigentlich das ganze?




http://www.flickr.com/photos/jpmartineau/501718334/

Definition
und **Provisionierung**
eines Linux-Containers



Container vs. VM

- Container sind keine VMs! **Leichtgewichtig**!

- LXC – **L**inu**X** **C**ontainers
  - System-level Virtualisation, *cgroups, namespaces usw.*

- Container
  - Kernel-Sharing
  - Prozesse in einer isolierten Umgebung (Netzwerk, Ressourcen usw.)
  - Innerhalb der Containers wie in einer VM
  - Außerhalb des Containers nur ein normaler Prozess

Container vs. VMs

Starte VM: schnell.

Starte Container: sehr schnell.

Wirklich sehr sehr schnell!

Docker - Installation

- Kernel sharing != Windows
  - Unterstützt wird Linux (>= 3.8 Kernel, Ubuntu bevorzugt)
  - OSX (via *boot2docker*)

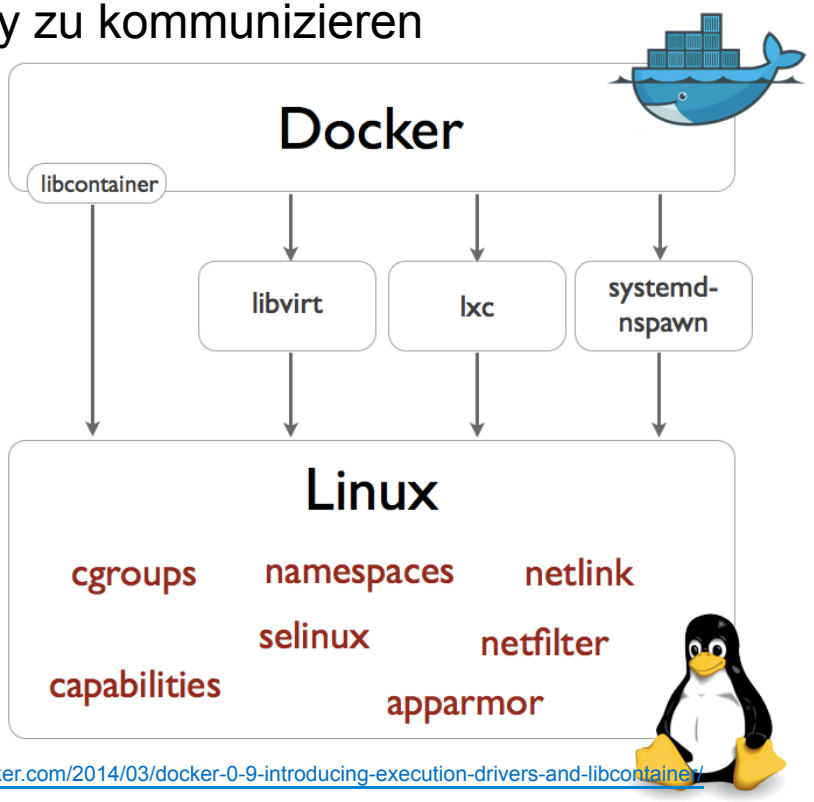
- Diverse I/PaaS-Platformen
  - Amazon EC2 (in normalen VMs)
  - Google CE (‚*container optimized image*')
  - OpenStack (‚*first class citizens*')
  - CloudFoundry (‚*Decker*')

- Mit Vagrant

- Chef, Puppet

Docker internals

- Filesystem
  - AUFS – Another union file system
  - Spezialisiert auf die Anforderungen von Docker
  - Inzwischen auch Unterstützung für andere Filesysteme
  - „Git-Like“ Befehle um mit Index/Repository zu kommunizieren
- cgroups
  - Kapselung von Gruppen von Prozessen
  - Isolierung von Ressourcen
- Docker ist „nur“ ein Interface zu *libcontainer*
  - LXC ist optional



Source: http://blog.docker.com/2014/03/docker-0-9-introducing-execution-drivers-and-libcontainer/

Docker - Wording

- Container
  - Laufzeit
  - Wird aus einem Image gestartet
- Image
  - Ein „Template“ um einen Container zu starten
  - Images werden in einem Repository verwaltet
  - Lokal, auf einem Server oder
- Image bauen
  - Interaktiv
  - Dockerfile
  - Packer

Warum Images / Container?

- *Self containing*

- Unabhängig von der Umgebung
  - > Ein Image für alle Stages
  - > Entwicklerfreundlich
  - > OPS-Freundlich

- Dev kümmert sich um das Innenleben
- Ops kümmert sich im die „Anschlüsse“

- Nur abhängig von Architektur
  - > 64/32 bit

www.flickr.com/photos/imaphotog/205492402/

Images

- Zentrales Repository
  - > pull/push
- Export / Import
  - > Archiv-Datei
- Ein Artefakt wie jedes andere!

Docker basics - Eigene Images

```
docker pull ubuntu
docker images
docker run ubuntu -i -t /bin/bash
root@8c13505e2526:/#
     […]
root@8c13505e2526:/# exit
docker commit 8c13505e2526 my-ubuntu:1.0
docker push someserver:5000/my-ubuntu:1.0
```

Docker basics - Dockerfile

```
FROM ubuntu:precise
MAINTAINER Halil-Cem Guersoy hcguersoy@gmail.com
RUN apt-get install -y wget

docker build -t ubuntu-wget .
docker inspect ubuntu-wget
docker push somereposerver:5000/ubuntu-wget
docker run -i -t ubuntu-wget /bin/bash
docker ps -a
```

Dockerfile für App-Deployment?

- Lowlevel, vergleichbar mit *Kickstart*
- Distro-Spezifisch
- Ggf besser: „echte“ Provisionierungstools
  - Chef
  - Puppet
  - Ansible

- Packer
  - Docker Support noch in Entwicklung
  - „Vagrant-Like“
  - Nutzt Provisionierungstools wie Puppet

Puppet-Integration

- Dockerfile
- ...oder mit Packer

```
ADD ./ /puppetwrk
RUN puppet apply -v --modulepath=/puppetwrk/modules/
                    /puppetwrk/manifests/nagios.pp
```

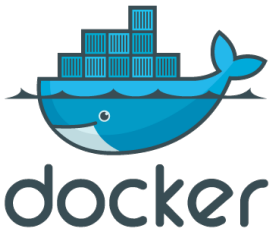
- Image wird mit Hilfe von Puppet provisioniert
- Komplexe Images möglich
- Application Deployment

Docker für App-Container

Java
Tomcat
Web Application

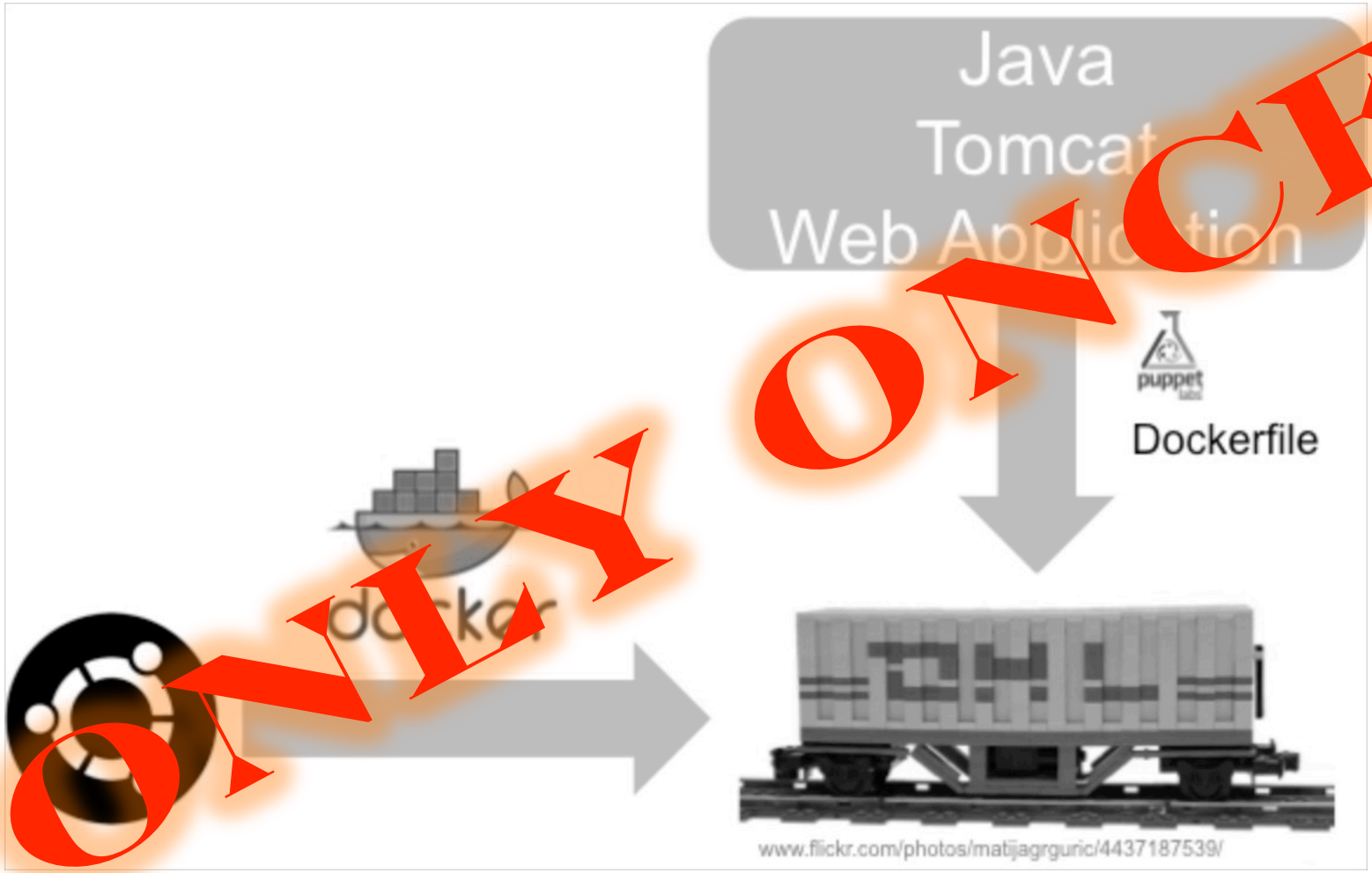
puppet labs

Dockerfile



www.flickr.com/photos/matijagrguric/4437187539/

Docker für App-Container



Docker in Continuous Delivery

- Images zur Softwareverteilung
  - Basis-Images mit z.B. Application Server oder Datenbank-Installation
  - App Images enthalten nur die Änderungen gegenüber den Basis-Images
  - z.B. Artefakte und Konfiguration
- Nur 1x deployen, konfigurieren usw.

```
└─3588c05c7d24 Virtual Size: 510.5 MB Tags: cdwrkshp/java7-tc7:latest
  └─7cd0471cb71e Virtual Size: 510.5 MB
    ├─54dc3c2d593c Virtual Size: 517 MB
    │ └─094cc7c59c79 Virtual Size: 523.1 MB
    │   └─688a08b73f44 Virtual Size: 523.1 MB
    │     └─52e46ca36cf0 Virtual Size: 523.1 MB Tags: angularjs-springmvc:1.1.0-42
    ├─7cc63bc98af7 Virtual Size: 517 MB
    │ └─8bf219561a14 Virtual Size: 523.1 MB
    │   └─ebe7ec7d34bb Virtual Size: 523.1 MB
    │     └─23205cc10892 Virtual Size: 523.1 MB Tags: angularjs-springmvc:1.1.0-41
    ├─d420a23f980b Virtual Size: 517 MB
    │ └─4afb0caceae6 Virtual Size: 523.1 MB
    │   └─aed914cc5e2c Virtual Size: 523.1 MB
    │     └─d17e5a80a367 Virtual Size: 523.1 MB Tags: angularjs-springmvc:1.1.0-40
```

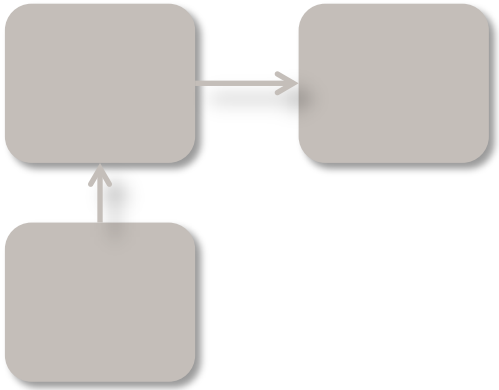
Container verbinden

- Eine Applikation besteht nicht nur aus einem Container
  - Runtime-Umgebung, z.B. App-Server (falls nicht schon tot), Java, Rails usw.
  - Datenbank, Webserver, Load Balancer usw.

- Container untereinander bekannt machen: *Links*

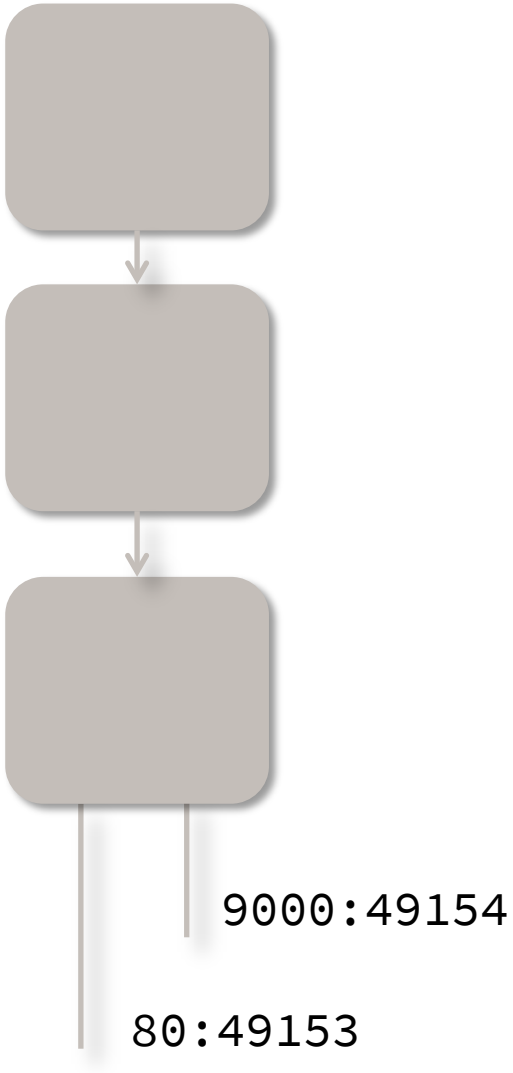
```
docker run -d -P --name db42
docker run -d -P --name app42 --link db42:db
```

```
--link [name]:[alias]
```



- Alias und Ports
  - in Umgebungsvariablen
  - /etc/hosts: `172.10.36.12  db`

Container verbinden



- Container exponieren ihre Ports
- Werden von Docker auf den Host gemappt beginnend ab 49153
- Oder es wird ein Port zugewiesen
- `Expose` im Dockerfile
- Ports werden „isoliert“
- Keine Port-Konflikte

Container ...


www.flickr.com/photos/matijagrguric/4437187539/


www.flickr.com/photos/matijagrguric/4437187539/


www.flickr.com/photos/matijagrguric/4437187539/


www.flickr.com/photos/matijagrguric/4437187539/


www.flickr.com/photos/matijagrguric/4437187539/


www.flickr.com/photos/matijagrguric/4437187539/


www.flickr.com/photos/matijagrguric/4437187539/


www.flickr.com/photos/matijagrguric/4437187539/


www.flickr.com/photos/matijagrguric/4437187539/

Docker und CI/CD-Tools

- Jenkins

Jenkins

- Jenkins mit Docker

- Shell
  - Shell „*Glue code*“ zur Interaktion mit lokalem oder remote Docker

- *Docker Build Step Plugin, Docker build publish Plugin*

- Jenkins Parametrizied Build Plugin
  - Durchreichen von notwendigen Informationen
  - Git / SVN Revision
  - Image ID
  - Container ID
  - Port-Mapping

Jenkins Skalieren

- Docker Container als Jenkins Slaves

- Docker Cloud Plugin
  - Docker Container starten
  - Jenkins Slave in Docker Container starten

- Skalierung
- Start - Build - Destroy

- Warum?
  - Saubere Trennung der Build-Prozesse
  - Skalierung
  - Keine Alterungseffekte

Andere Ci/CD-Tools

- Atlassian Bamboo
  - Diverse Dockerfiles für „Installation“
  - https://bitbucket.org/atlassianlabs/atlassian-docker
- JetBrains TeamCity
  - Diverse Plugins zur Build-Ausführung in Containern
  - TeamCity.Virtual Plugin https://github.com/jonnyzzz/TeamCity.Virtual
- drone.io
  - Build-System
  - Basiert vollständig auf Docker
  - https://github.com/drone/drone / http://drone.io

Docker for real life

- Docker bietet eine Remote API
  - Ist „fast RESTfull“
  - Starten / Stoppen usw. usw.
  - Öffnet Möglichkeit zur Verwaltung von zentralen Docker-Hosts

- CoreOS
  - Orchestrierung von Docker-Instanzen / Clusteraufbau
- Shipyard
  - UI zur Verwaltung von Docker-Hosts
  - REST API – einfacher als Docker Remote API
- DockerUI
  - Recht einfache GUI


Demo time
http://www.flickr.com/photos/schwaber/3238977103/

http://www.flickr.com/photos/an_untrained_eye/6630719431/

adesso

IT MUSS NEU GEDACHT WERDEN – NEW SCHOOL OF IT

New-School-of-IT.de

adesso GROUP

Vielen Dank für Ihre Aufmerksamkeit.

Kontakt:
halil-cem.guersoy[at]adesso.de
Twitter: @hgutwit
G+ https://plus.google.com/+HalilCemGürsoy