

Continuous Delivery mit Docker

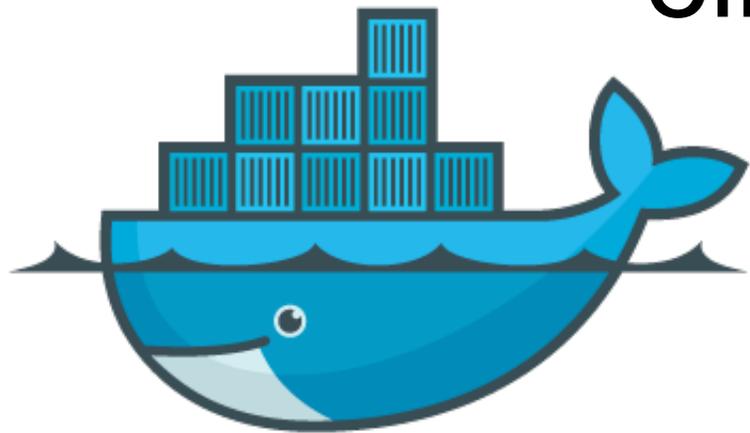
Berlin Expert Days 2014

Dr. Halil-Cem Gürsoy
adesso AG



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

Definition und Provisionierung eines Linux-Containers



docker

Container vs. VM

- ▶ Container sind keine VMs!
- ▶ LXC – **LinuX Containers**
 - > chroot & cgroups
- ▶ Container
 - > Linux System läuft innerhalb eines Linux-Systems
 - > Gruppe von Prozessen in einer isolierten Umgebung
 - > Innerhalb der Containers wie in einer VM
 - > Außerhalb des Containers nur ein normaler Prozess
 - > Netzwerk, Ressourcen usw. vollständig isoliert
- ▶ Kaum Footprint

Starte VM: schnell.

Starte Container: sehr schnell.

Wirklich sehr sehr schnell!

- ▶ Kernel sharing != Windows
 - > Unterstützt werden Linux (≥ 3.8 Kernel, Ubuntu bevorzugt)
 - > OSX (Client, boot2docker)
- ▶ Diverse IaaS-Plattformen
 - > Amazon EC2
 - > Google CE
 - > OpenStack
- ▶ Mit Vagrant
 - > Provisioner „Docker“
- ▶ Chef, Puppet



- ▶ 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

- ▶ cgroup
 - > Kapselung von Gruppen von Prozessen
 - > Isolierung von Ressourcen

- ▶ Docker ist „nur“ ein Interface zu LXC

- ▶ Container
 - > Laufzeit
 - > Wird aus einem Image gestartet

- ▶ Image
 - > Ein „Template“ um einen Container zu starten
 - > Images werden in einem Repository verwaltet
 - > Lokal oder auf einem Server

- ▶ Image bauen
 - > Interaktiv
 - > Dockerfile
 - > Packer

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-Deploymen?

- ▶ Lowlevel, vergleichbar mit Kickstart
- ▶ Distro-Spezifisch
- ▶ Besser: „echte“ Provisionierungstools
 - > Chef
 - > Puppet
- ▶ Packer
 - > Docker Support noch in Entwicklung
 - > Langfristig richtiger Ansatz zur Integration
 - > „Vagrant-Like“



Puppet-Integration

- ▶ Dockerfile
- ▶ ...oder mit Packer

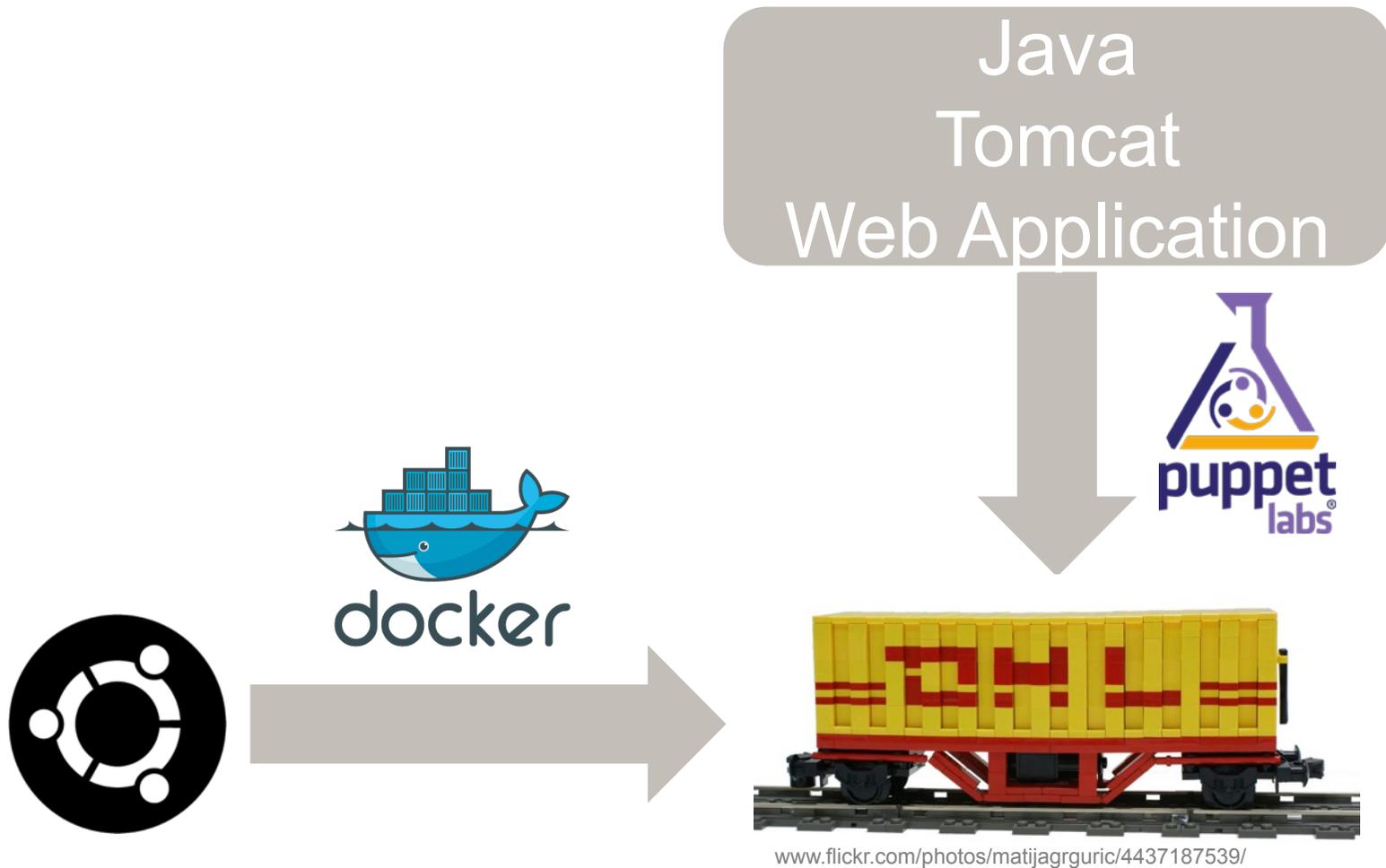
```
ADD ./ /puppetwrk
```

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

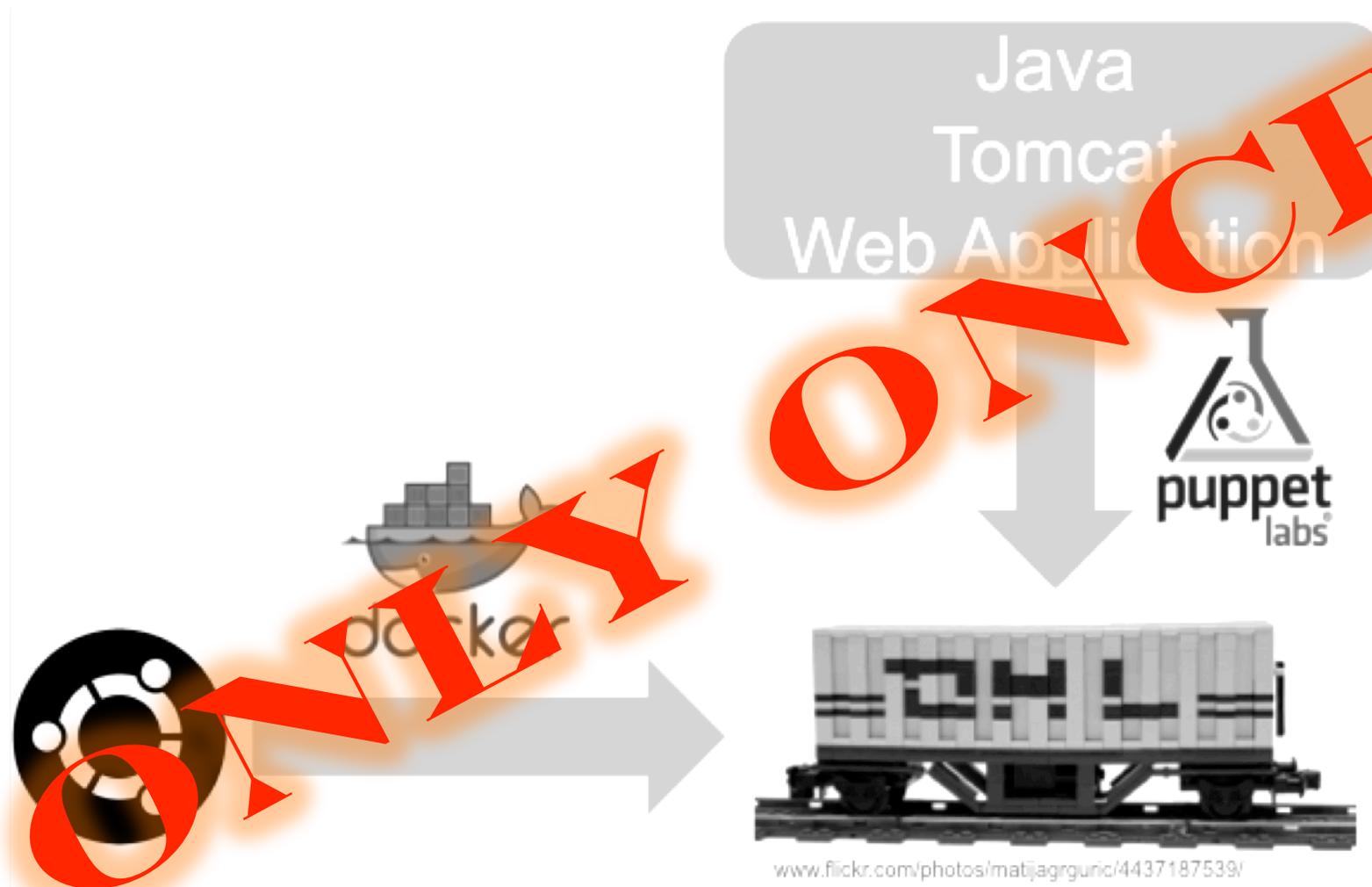
- ▶ Image wird mit Hilfe von Puppet provisioniert
- ▶ Komplexe Images möglich
- ▶ Application Deployment



Docker für App-Container



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
```

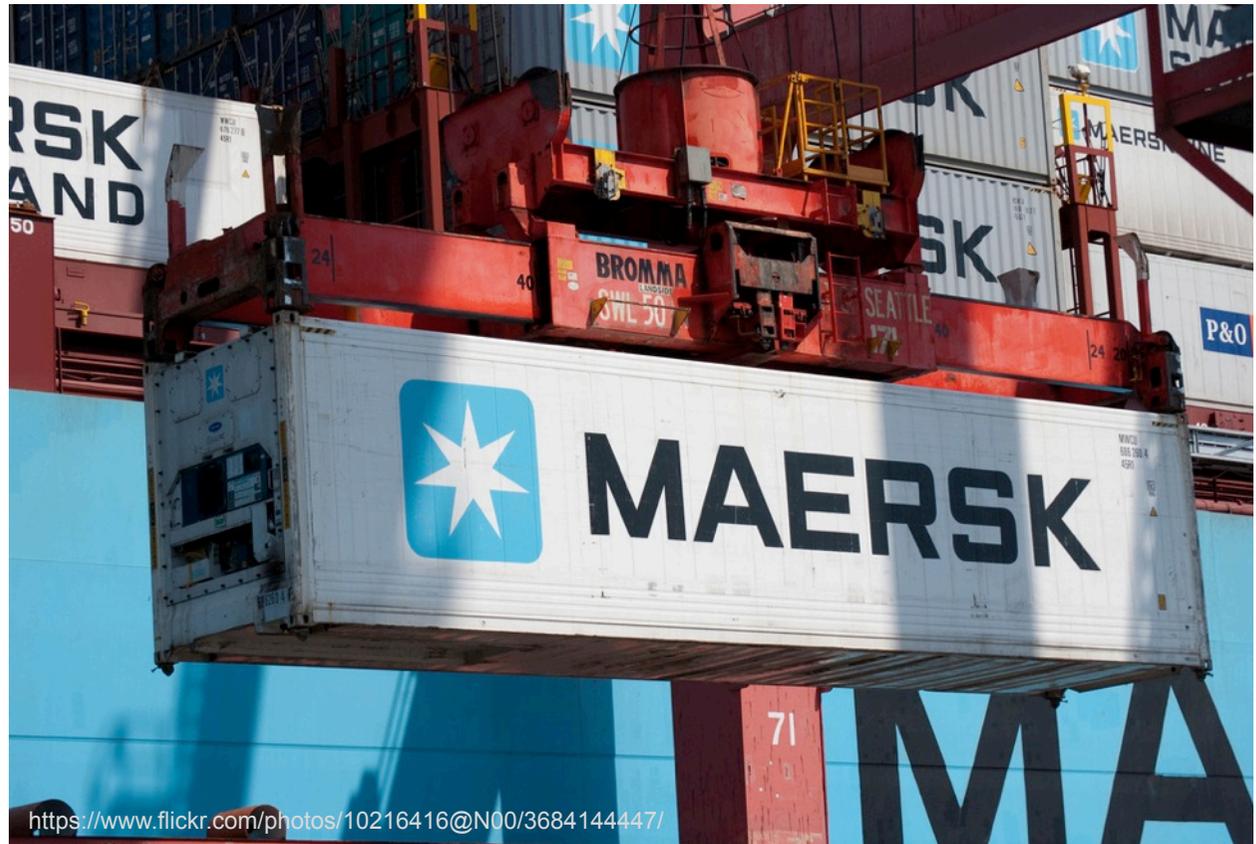
Warum Images / Container?

- ▶ *Self containing*
- ▶ Unabhängig von der Umgebung
 - > Ein Image für alle Stages
 - > Entwicklerfreundlich
 - > OPS-Freundlich
- ▶ Dev kümmert sich um das Innenleben
- ▶ Ops kümmert sich um die „Anschlüsse“
- ▶ Nur abhängig von Architektur
 - > 64/32 bit



Images

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



Docker und CI/CD-Tools

► Jenkins



Build Pipeline: Angular Demo Pipeline





- ▶ Jenkins mit Docker
- ▶ Shell
 - > Shell „Glue code“ zur Interaktion mit lokalem oder remote Docker
- ▶ Jenkins Parametrized Build Plugin
 - > Durchreichen von notwendigen Informationen
 - > Git / SVN Revision
 - > Image ID
 - > Container ID
 - > Port-Mapping
- ▶ Docker Executor Plugin

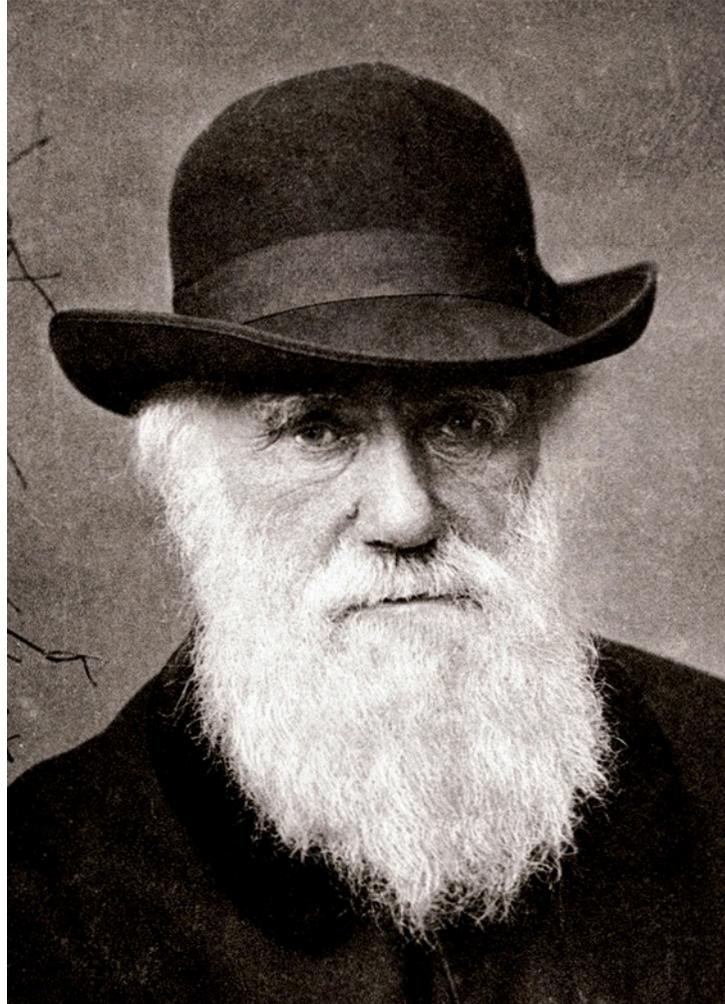
- ▶ Docker Container als Jenkins Slaves
- ▶ Docker Cloud Plugin
 - > Docker Container starten
 - > Jenkins Slave in Docker Container starten
- ▶ Skalierung
- ▶ Start – Build - Destroy
- ▶ Warum?
 - > Saubere Trennung der Build-Prozesse
 - > Buildspezifische Images
 - > Skalierung

Demo time

- ▶ 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 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

Warum eigentlich das ganze?





Vielen Dank für Ihre Aufmerksamkeit.

Kontakt:

halil-cem.guersoy[at]adesso.de

Twitter: [@hgutwit](https://twitter.com/hgutwit)

G+ <https://plus.google.com/+HalilCemGürsoy>

www.adesso.de
info@adesso.de

