



Business
Technology Days

LEAN INNOVATION & DIGITAL TRANSFORMATION

Dr. Roland Huß
ConSol*

Docker für Java Entwickler

Agenda

- Docker Crash Intro
- Integrationstests
- Applikations-Deployment
- Build Integration
- docker-maven-plugin
- Maven Plugin Shootout
- Demo

Roland Huß

- Java seit 1997
- Open Source
 - www.jolokia.org
 - labs.consol.de & ro14nd.de
 - <https://github.com/rhuss>
- Tour 2014
 - JavaZone
 - W-JAX
 - Devovx



ConSol 
Consulting & Solutions

 @ro14nd



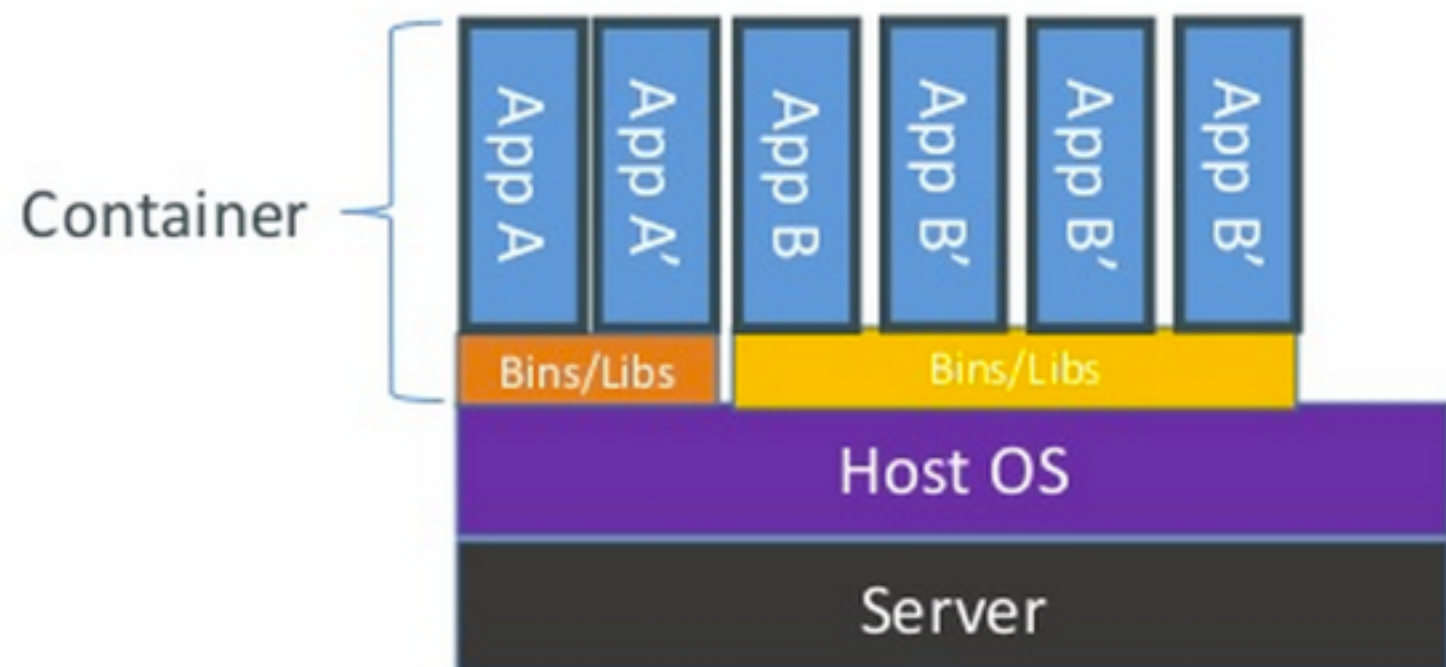
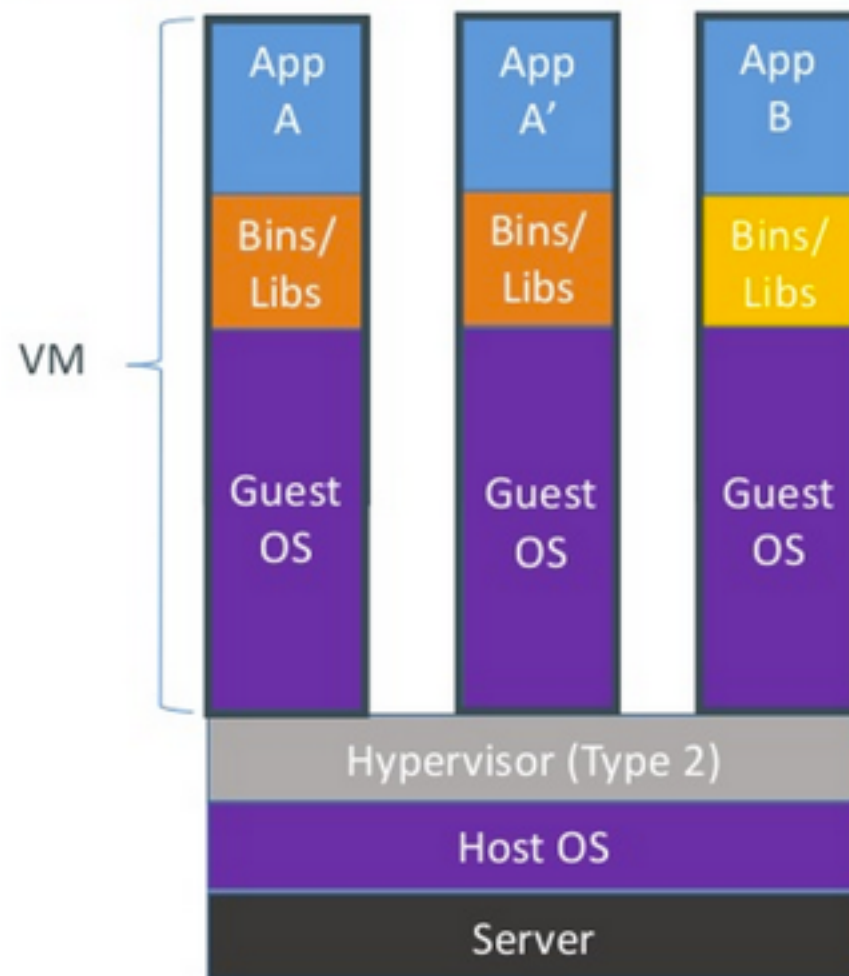
Docker ist eine offene Plattform für
Entwickler und Administratoren um
verteilte Applikationen zu **bauen**,
auszuliefern und zu **betreiben**.

docker.io

Lightweight Container vs.VM

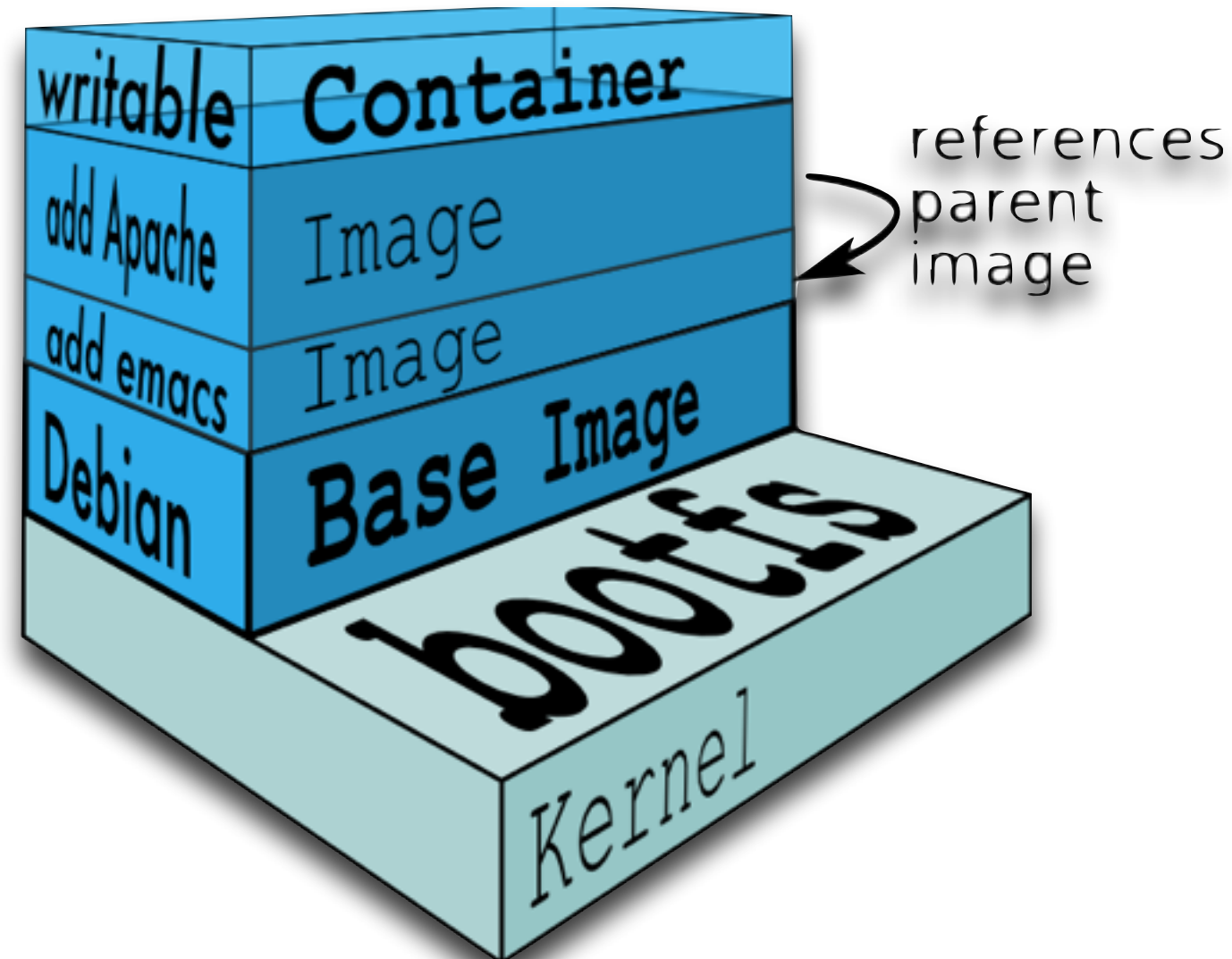
Container sind isoliert, teilen sich aber Kernel und (einige) Dateien

→ schneller & leichter



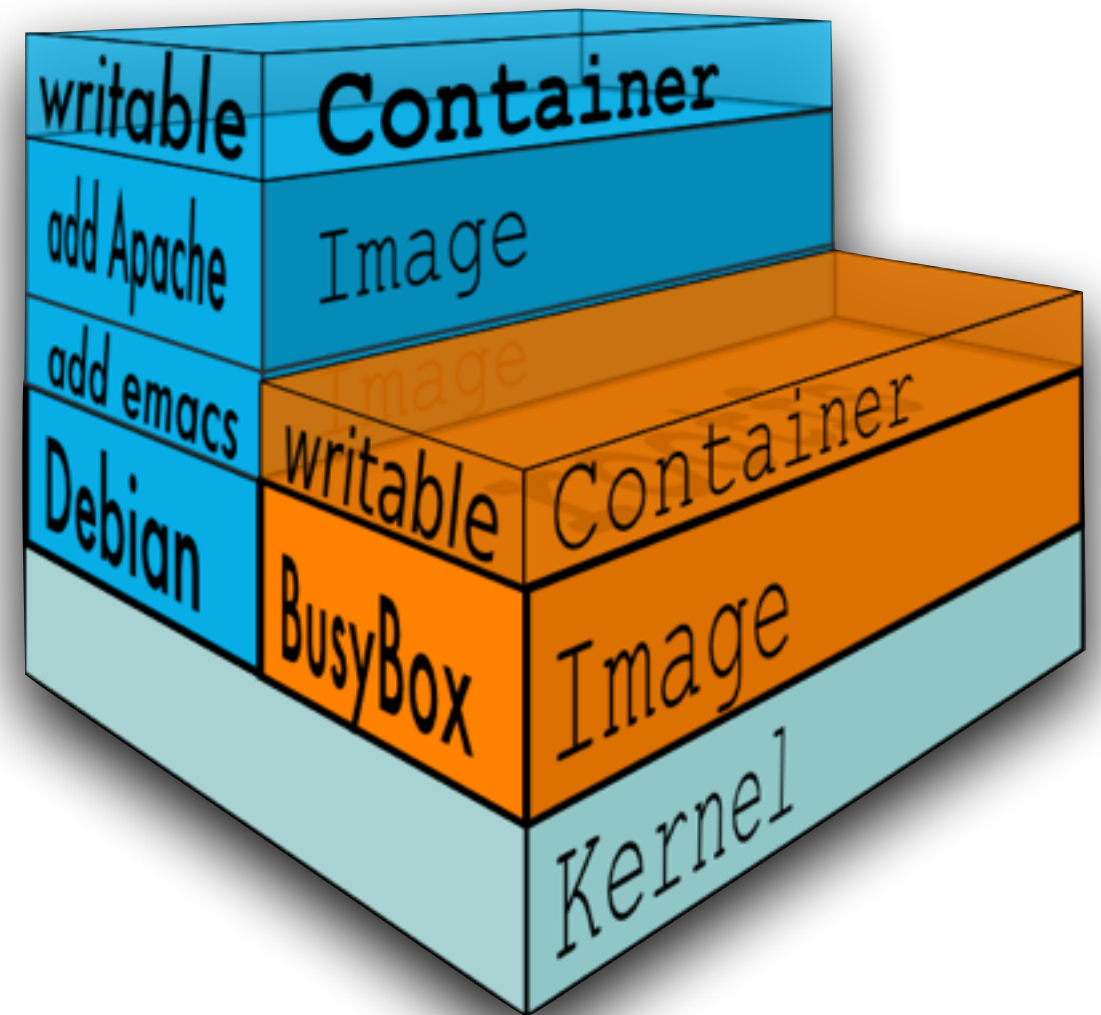
Image

- read-only Dateisystem Schicht
- kann geladen und verteilt werden
- "Blaupause für einen Container"



Container

- read-write
Dateisystem Schicht
- Copy-On-Write
- kann gestartet und gestopped werden
- "Instanz eines Images"



Docker CLI Kommandos

ps

Zeige alle Container

images

Zeige alle Images

run

Erzeuge und starte Container

search

Suche Images in einer Registry

pull

Download von Images

rm

Entfernen eines Container

rmi

Entfernen eines Images

exec

Ausführen eines Kommandos im Container

Docker für Java Entwickler ?

Integrationstests

Applikations-Deployment

Integrationstest

Integrationstests prüfen
Applikationen in einem **realistischen**
Kontext der der
Produktionsumgebung so nahe wie
möglich kommt.

Gute Integrationstests

- **Robust** (aka Wiederholbar)
Laufen entweder immer durch oder schlagen mit dem gleichen Test fehl
- **Autark**
Minimale externe Abhängigkeiten, eigenständig
- **Isoliert**
Parallele Ausführung der gleichen Tests
- **Schnell**
Kurze Turnaround-Zeiten

Externe Testsysteme

~~Robust~~

Test Systeme werden extern verwaltet und konfiguriert.

~~Autark~~

Test Systeme müssen installiert und verfügbar sein.

~~Isoliert~~

Parallele Tests greifen auf das gleiche System zu und stören sich gegenseitig.

~~Schnell~~

Wegen paralleler Nutzung und Netzwerklatenz oft langsam.

aber *nahe* an der Realität !

Simulierte (Mock) Testsysteme

Robust

Kann während des Testlaufs gestartet werden.

Autark

Kann deklarativ konfiguriert werden (z.B. Citrus).

Isoliert

Verschiedene Ports pro Testlauf können konfiguriert werden.

Schnell

Mock Systeme sind aufgrund begrenzter Funktionalität oft schneller.

aber entspricht *nicht* der Wirklichkeit !

Integrationstests mit Docker

Robust

Jeder Testlauf hat einen eigenen Container und Ausführungskontext.

Autark

Keine externen Abhängigkeiten ausser einer Docker Installation.

Isoliert

Perfekte Isolation der Container möglich.

Schnell

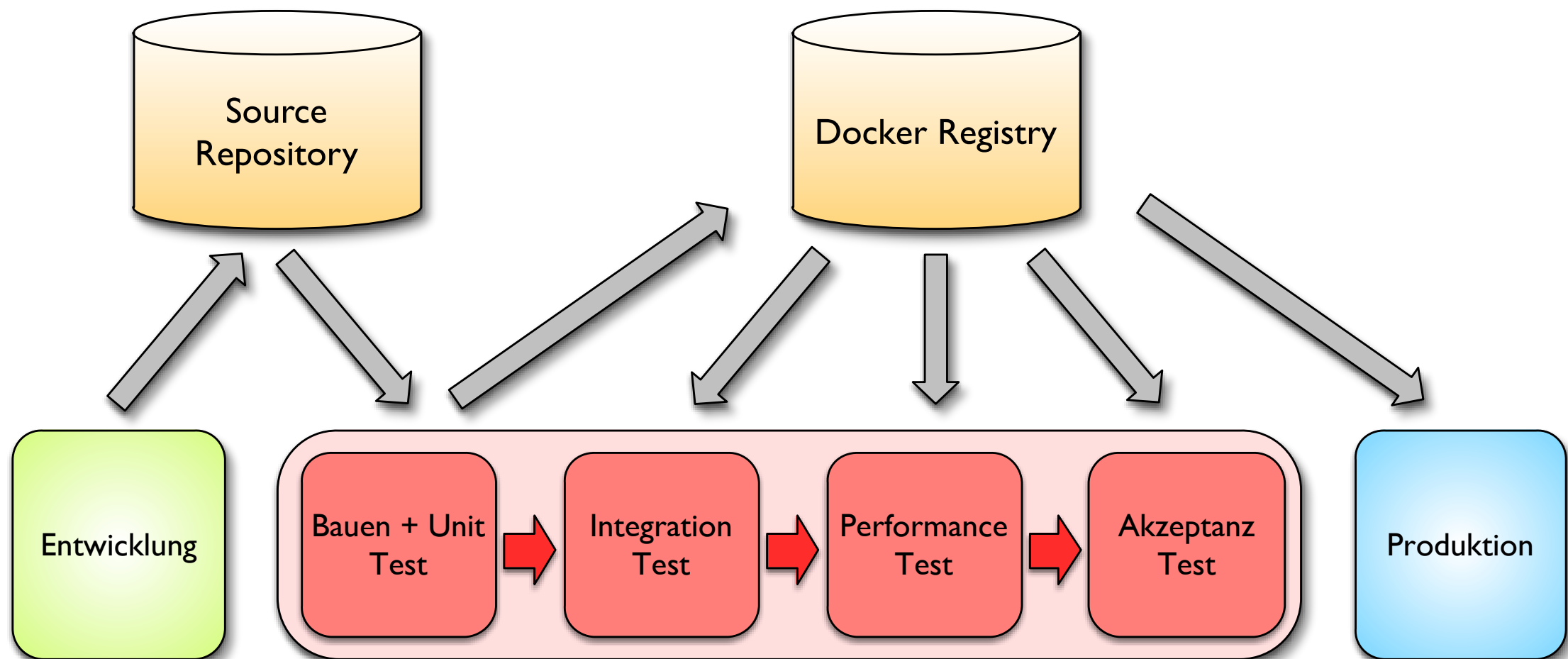
Schneller Container Start dank der Systemlevel Virtualisierung.

und es *kann* die Realität abbilden !

Applikations Deployment

- Standard "Container" Formate für Java Enterprise Anwendungen:
 - Web-Archive (WAR)
 - Enterprise-Archive (EAR)
- Bei Docker wird der Ausführungskontext (Server) mit in den Container gepackt.
- **Immutable Server Pattern**
 - Bei einer neuen Version der Applikation wird auch ein neuer Server deployed

Docker Delivery Pipeline













Container Patterns

- **Datencontainer:**
 - Artefakte werden in einen **Datencontainer** verpackt.
 - Datencontainer wird mit einem **Plattformcontainer** verknüpft.
 - Applikation wird beim Start des Plattformcontainers deployed.
- **Servicecontainer:**
 - Artefakte und Laufzeitumgebung (z.B. Applikationsserver) werden in den **gleichen Container** gepackt.
 - Ideal für Microservices.

Build Integration

- CI Server
 - Pre- und Post-Hooks zum Starten und Stoppen von Docker Container.
- Aufruf der Docker CLI aus dem Build heraus:
 - `exec` Ant-Task
 - `exec-maven-plugin` für Maven
 - mit Groovy aus Gradle heraus
- Dedizierte Maven und Gradle Plugins

docker-maven-plugin

	alexec/docker-maven-plugin Last updated 2 days ago	Java	★ 13	🔗 12
	wouterd/docker-maven-plugin A maven plugin to manage docker containers and images for integration tests. Last updated 3 days ago	Java	★ 13	🔗 3
	etux/docker-maven-plugin Maven plugin to interact with Docker. Last updated on 4 Apr	Java	★ 2	🔗 3
	bibryam/docker-maven-plugin Maven plugin for Docker Last updated on 1 Apr	Java	★ 22	🔗 0
	rhuss/docker-maven-plugin Maven plugin for managing Docker images and containers Last updated 7 hours ago	Java	★ 23	🔗 1
	spotify/docker-maven-plugin A maven plugin for docker Last updated 9 days ago	Java	★ 0	🔗 2
	FitburIO/docker-maven-plugin Docker Maven Plugin Last updated 11 days ago	Java	★ 0	🔗 0
	vjuranek/docker-maven-plugin Maven plugin for Docker Last updated on 23 Mar	Java	★ 1	🔗 0
	imagnatelabs/docker-maven-plugin Docker Maven Plugin Last updated on 2 May	Java	★ 1	🔗 0
	ihr/docker-maven-plugin Maven plugin for Docker orchestration Last updated on 16 Jun		★ 0	🔗 0

WTF or
FTW ?

Die 4 Überlebenden

- **wouterd/docker-maven-plugin**
 - Wouter Danes, ING
- **alexec/docker-maven-plugin**
 - Alex Collins
- **spotify/docker-maven-plugin**
 - Spotify
- **rhuss/docker-maven-plugin**
 - Roland Huß, ConSol

docker-maven-plugin

	<i>wouterd</i>	<i>alexec</i>	<i>spotify</i>	<i>rhuss</i>
<i>Start/Stop</i>	✓	✓	✗	✓
<i>Build/Push</i>	✓	✓	✓	✓
<i>API</i>	jaxrs	docker-java	spotify/docker-client	Apache HC
<i>Image</i>	Dockerfile + Maven Config	Dockerfile + custom YML	Maven config + Dockerfile	Maven config + Assembly + (Dockerfile)
<i>SSL Support</i>	✓	✗	✗	✓

docker-maven-plugin

	<i>wouterd</i>	<i>alexec</i>	<i>spotify</i>	<i>rhuss</i>
<i>Cleanup</i>	✓	✓	✗	✓
<i>Security</i>	Plain	Plain	✗	Encrypted/Plain
<i>URL Wait</i>	✗	✓	✗	✓
<i>Version</i>	2.3	2.1.0	0.0.21	0.10.4
<i>Size LOC</i>	2200	1000	600	2200

rhuss/docker-maven-plugin

- Einfache Konfiguration
- Automatischer Download von Images
- Dynamisches Portmapping
- Maven Artefakte und ihre Abhängigkeiten sollen im Container zur Verfügung stehen
- Upload von Container zu einer Registry
- "Doing it the Maven way"

Goals

docker:start

Starten von Container
(pre-integration-test)

docker:stop

Stoppen von Container
(post-integration-test)

docker:build

Bauen von Images

docker:push

Push Images zu einer Registry

docker:remove

Entfernen von Images

Beispiel Konfiguration

```
<images>
  <image>
    <name>jolokia/integration-test</name>
    <build>
      <from>consol/tomcat-7.0</from>
      <assemblyDescriptor>assembly.xml</assemblyDescriptor>
    </build>
    <run>
      <ports>
        <port>jolokia.port:8080</port>
      </ports>
    </run>
  </image>
</images>
```

Assembly Deskriptor

```
<assembly>
  <dependencySets>
    <dependencySet>
      <includes>
        <include>org.jolokia:jolokia-war</include>
      </includes>
      <outputDirectory>.</outputDirectory>
      <outputFileNameMapping>jolokia.war</outputFileNameMapping>
    </dependencySet>
  </dependencySets>
</assembly>
```

Artefakte im Container

- Assembly Deskriptor des maven-assembly-plugin
 - Build Artefakte
 - Abhängigkeiten
 - beliebige Dateien
- Vordefinierte Deskriptoren
- Daten stehen im Container unter /maven zur Verfügung.

docker-maven-plugin "Shootout"

- Docker Demo Projekt
 - Vanilla PostgreSQL 9 Image
 - HTTP Request Logging Service
 - MicroService mit embedded Tomcat
 - DB Schema wird via Flyway während des Starts gebaut
 - PostgreSQL Container wird über Link angebunden
 - Einfache Integrationstest, der den Service nutzt
- Maven Profile für verschiedene Plugins: wouterd, alexec, spotify, rhuss
- Aufruf: `mvn -Prhuss clean install`
- <https://github.com/rhuss/shootout-docker-maven>

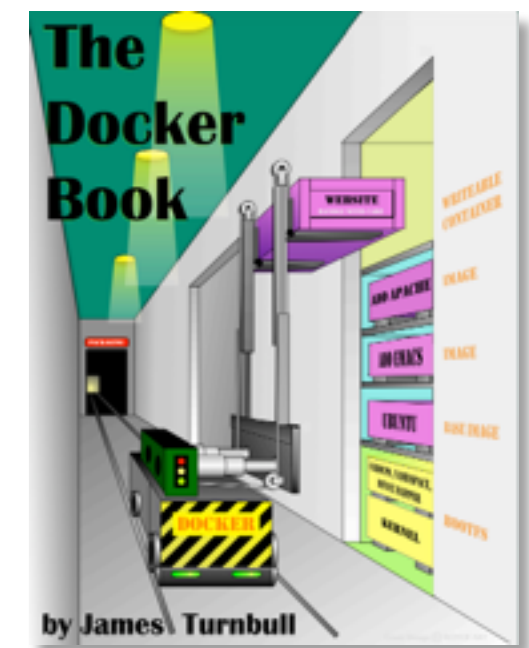
Demo

Zusammenfassung

- Docker ist eine leichtgewichtige Virtualisierungstechnik die den Entwicklungsprozess verbessern kann:
 - Docker kann helfen gute Integrationstests zu entwickeln.
 - Docker bietet eine neues Paradigma für die Auslieferung von Applikationen.
- Für Maven existiert eine (oder mehrere ;-) gute Docker Integration

Referenzen

- index.docker.io - Public Docker Registry
- Entwickler Magazin Spezial Vol.2: Docker
 - Erscheinungsdatum: 17.11.2014
 - http://entwickler.de/docker_spezial
- "The Docker Book"
 - sehr zu empfehlen !
 - <http://www.dockerbook.com/>



Danke !

```
docker_nuke() {  
    docker ps -q | xargs docker stop  
    docker ps -q -a | xargs docker rm  
}  
  
docker_rmi_none() {  
    docker images | grep '<none>' | \  
    awk '{ print $3 }' | \  
    xargs docker rmi  
}  
  
docker_go() {  
    docker run --rm -t -i $@  
}
```



**Franziskanerstraße 38
D-81669 München**

**Tel: +49-89-45841-100
Fax: +49-89-45841-111**

**info@consol.de
www.consol.de**

Docker Advanced

Port Mapping

- Container können Ports bereitstellen
 - werden zur Laufzeit festgelegt
- Exportierte Ports werden auf Host Ports abgebildet.

```
docker run -P
```

Bilde alle exportierten Container Ports auf dynamische Ports aus dem Bereich 49000 ... 49900 ab.

```
docker run -p 8080:8080  
-p 2200:22
```

Bilde Container Port 8080 und 22 auf lokale Ports 8080 und 2200 ab.

```
docker run -p 8080 -p 22
```

Bilde Container Ports 8080 und 22 dynamisch auf lokale Ports aus dem Bereich 49000 ... 49900 ab.

Volumes

- Teilen von Daten zwischen ...
 - ... Container und Container
 - ... Container und Docker Host

```
docker run -v /var/volume1 \  
          -v /var/volume2 \  
          --name DATA busybox true  
docker run -t -i --rm \  
          --volumes-from DATA \  
          --name client1 ubuntu bash
```

Container Linking

- Erzeuge und starte Container mit einem Namen:

```
docker run -d --name redis crosby/redis
```

- Referenziere Container via Name:

```
docker run -t -i --link redis:db ubuntu bash
```

- Verbindungsparameter innerhalb des referenzierenden Containers:

- über /etc/hosts

```
172.17.0.3      db
```

- über Umgebungsvariablen:

```
DB_PORT_5432_TCP=tcp://172.17.0.3:5432
```

```
DB_ENV_PG_VERSION=9.3.5-1.pgdg70+1
```

```
.....
```

Bauen von Images - Run & Commit

- Wähle ein Basis Image aus:

```
docker run -t -i ubuntu bash
```

- Manuelle Installation von Software usw. innerhalb des Containers.
- Stoppe & Speichere Container:

```
docker commit <container-id> <image>  
docker tag <image> <repository>  
docker push <user-name>
```

Bauen von Images - Dockerfile

```
FROM jolokia/java-jolokia:8

MAINTAINER roland@jolokia.org

EXPOSE 8080

RUN wget http://archive.apache.org/tomcat-7/.. -O /tmp/c.tgz
RUN tar xzf /tmp/c.tgz -C /opt
RUN rm /tmp/c.tgz
CMD ["/opt/apache-tomcat-7/bin/catalina.sh", "run" ]
```

Docker Architecture

- Originally based on Linux Container (LXC)
 - since 0.90 :Also own abstraction with libcontainer
- Client-Server Architecture
 - Server communicates via Unix- or INET-Sockets with a REST API
 - Docker Commandos vial CLI (Linux, OS X and Windows)
- Written in Go
- Current version: 1.3.0

For OS X & Windows: boot2docker

- <https://github.com/boot2docker/boot2docker>
- VirtualBox Image: Tiny Core Linux, 24 MB
- OS X:
 - CLI for running and managing the VM
 - Installation via brew (as for for docker)
- Tip: Setting up Port Forwarding for dynamic Ports
 - Howto: <http://bit.ly/1eL7o7I>
 - Script „boot2docker-fwd“: <https://gist.github.com/deinspanjer/9215467>
- Host-Volume sharing not possible