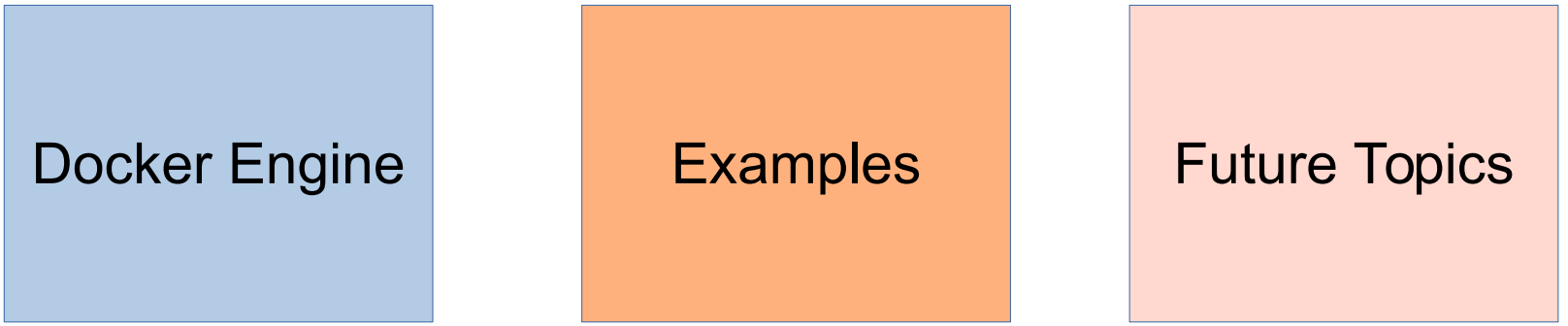
Docker

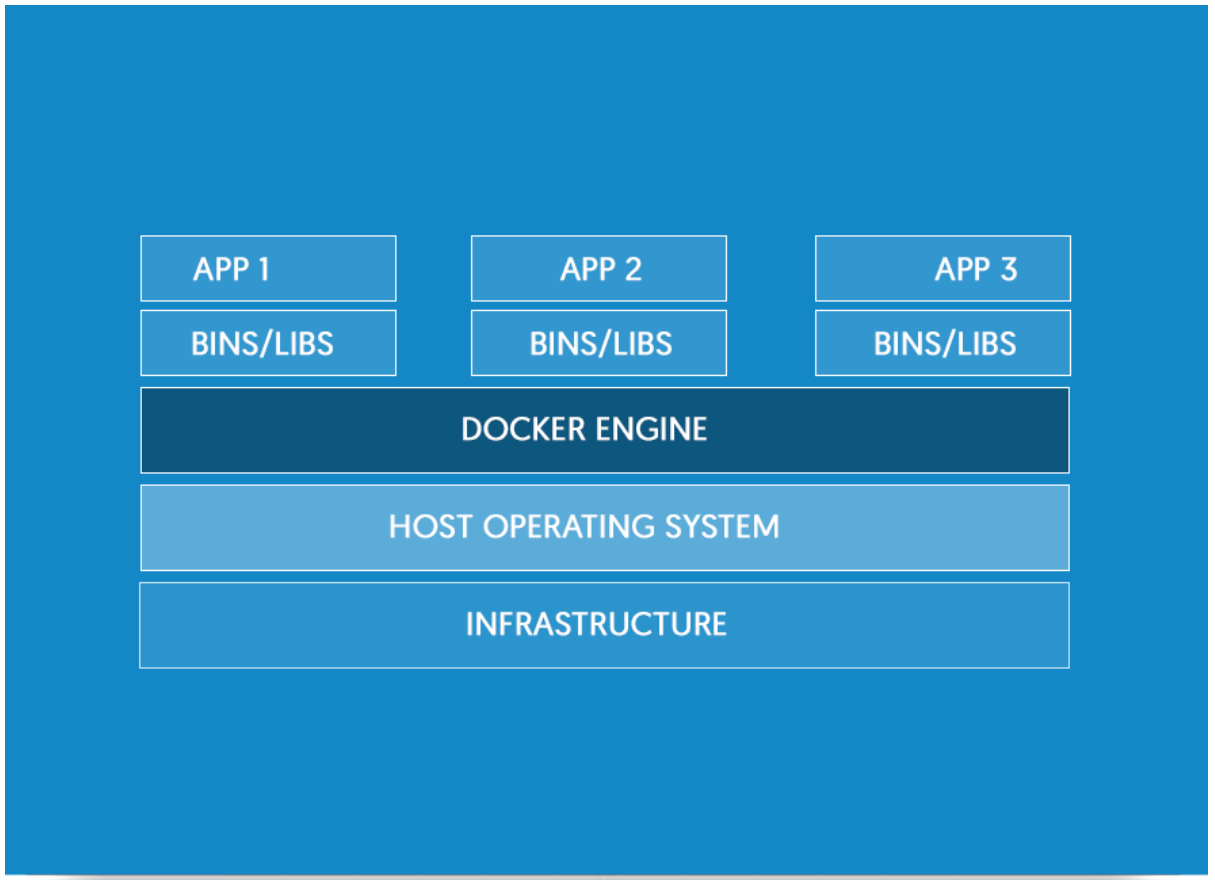
Biel, 09.11.2016

Benjamin Fankhauser

Berner
Fachhochschule


Docker Engine
Examples
Future Topics

Docker Project



- Open Source

- Linux Only (Andere OS via vorkonfigurierte VM)

- Organisation ähnlich wie Github

Docker

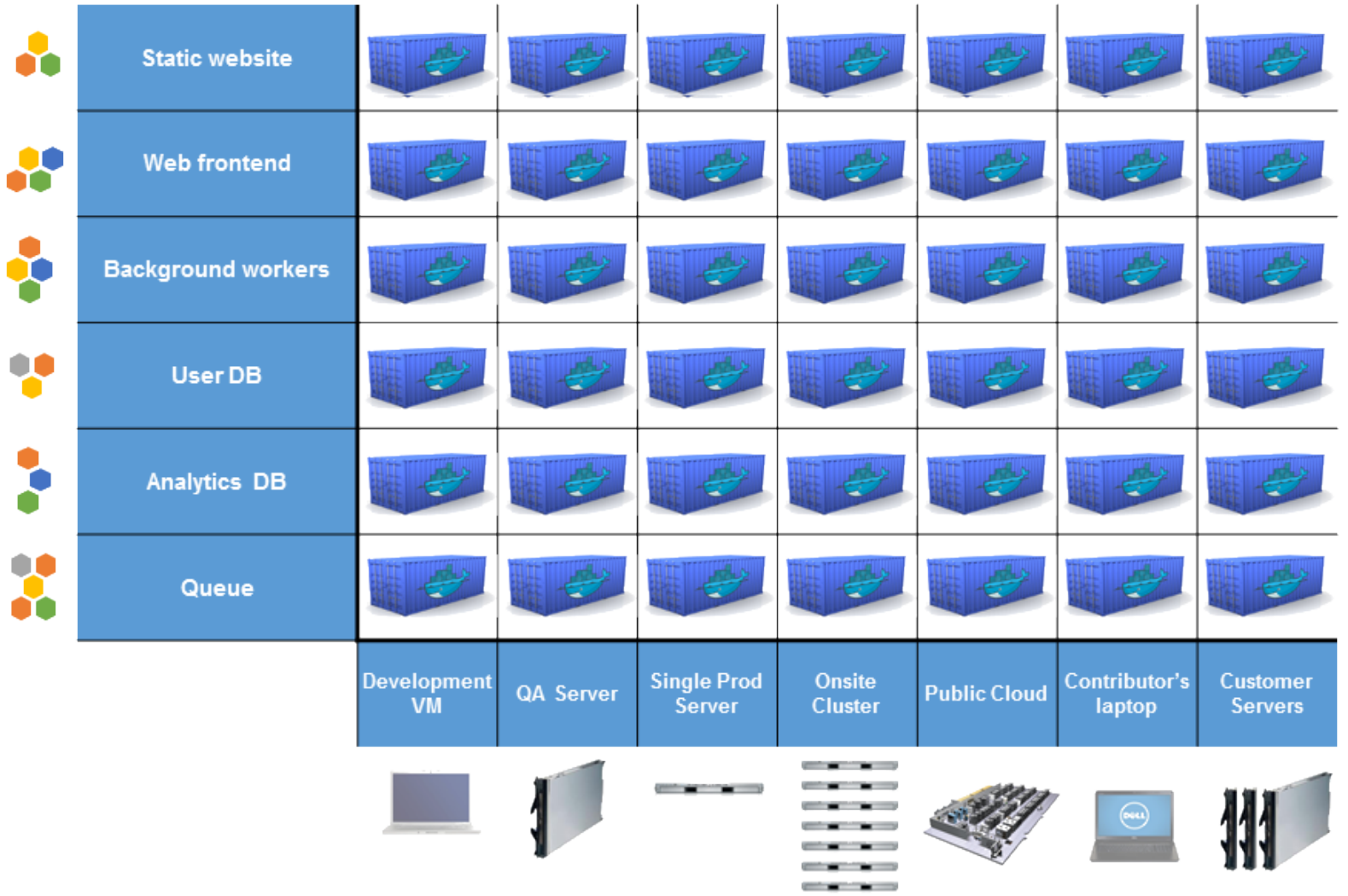
Inhalt

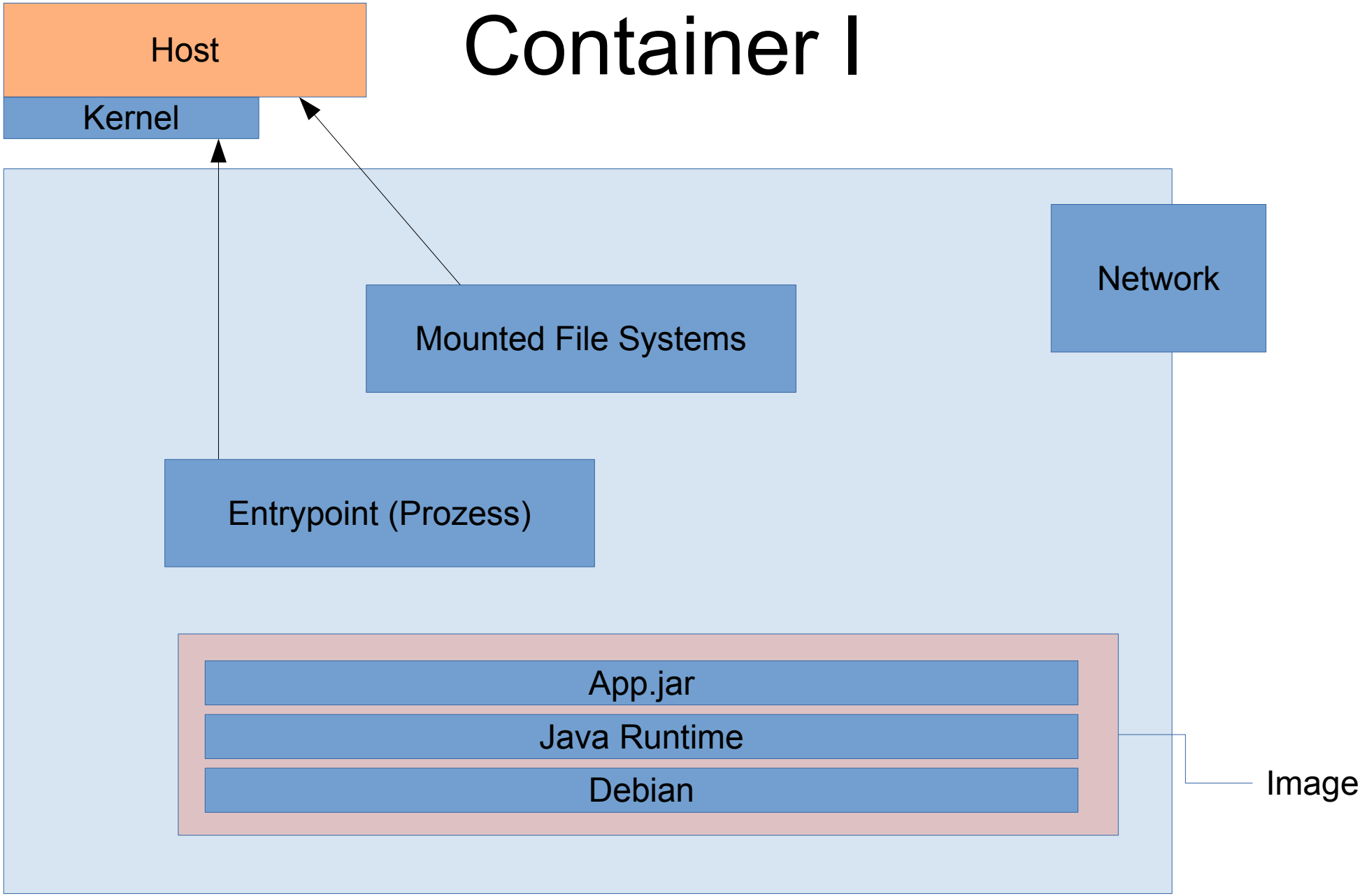
- Container und Images
- Technische Umsetzung
- Image Verwaltung
- Security

Container

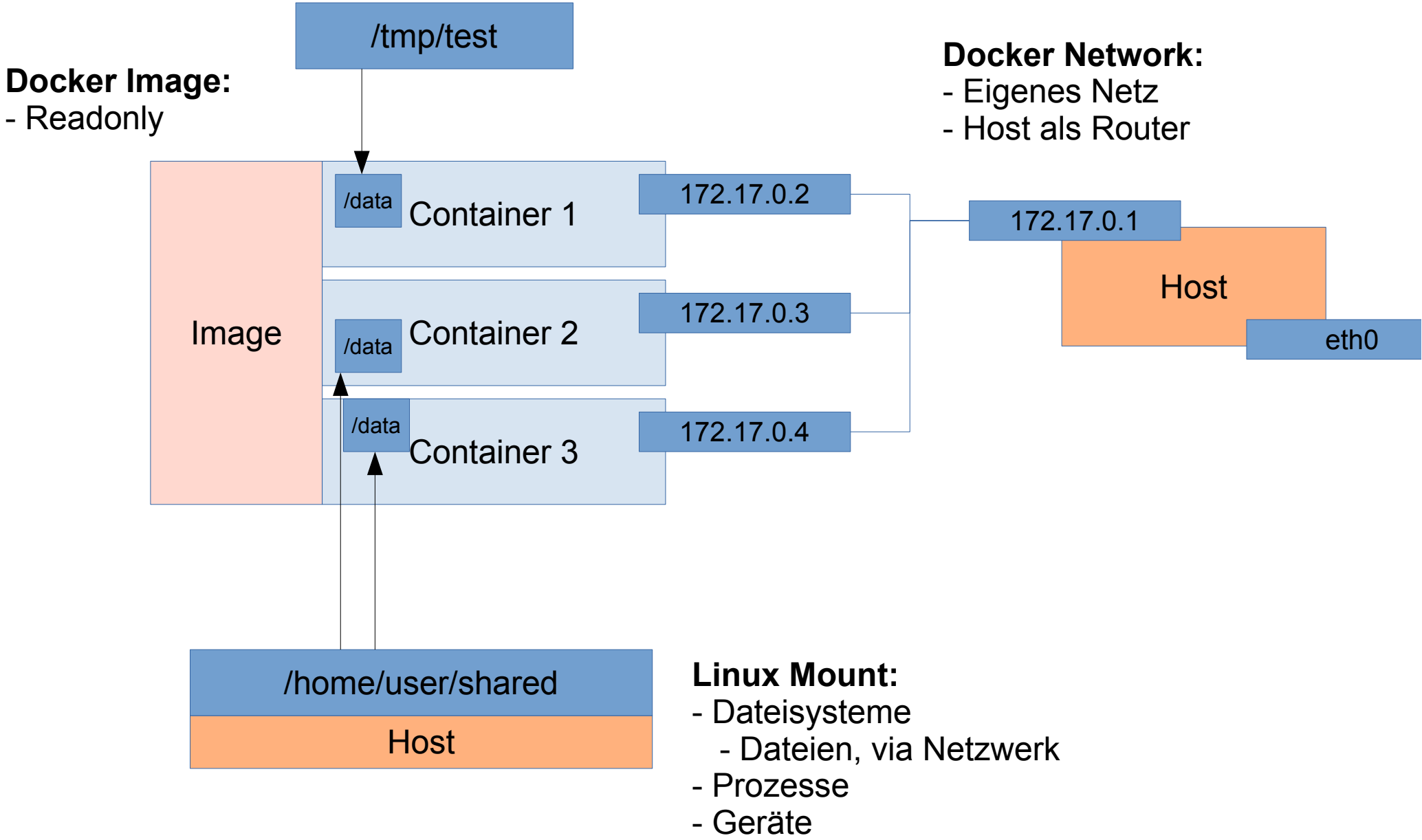
Quelle:
http://paislee.io/content/images/2015/03/blue_whale_container_ship.jpg
http://docker.com

Container


Static website
Web frontend
Background workers
User DB
Analytics DB
Queue
Development VM
QA Server
Single Prod Server
Onsite Cluster
Public Cloud
Contributor's laptop
Customer Servers

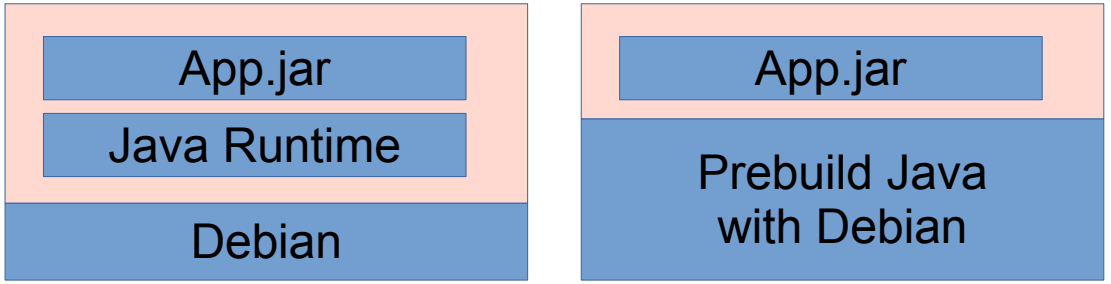

Container I
Host
Kernel
Network
Mounted File Systems
Entrypoint (Prozess)
App.jar
Java Runtime
Debian
Image

Container II



Image

Verschiedene Wege zum selben Ziel



Images sind Readonly und werden gemeinsam benutzt

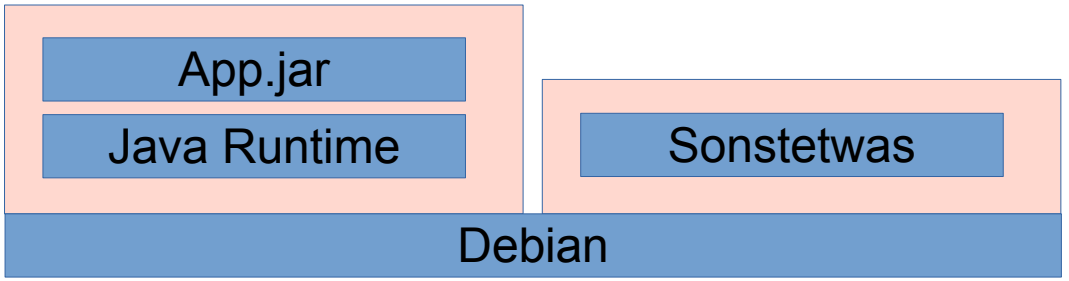


Image Layers

```
[fsb1@fy03 ~]$ dockviz images --tree
├─<missing> Virtual Size: 123.0 MB
│ └─<missing> Virtual Size: 123.0 MB Tags: [redacted]/debian:latest, debian:latest
│   ├─c613455bff48 Virtual Size: 418.7 MB Tags: [redacted]/mariadb-vtvote:latest
│   └─<missing> Virtual Size: 167.3 MB
│     └─<missing> Virtual Size: 289.9 MB
│
└─<missing> Virtual Size: 4.8 MB
  ├─<missing> Virtual Size: 11.4 MB
  │ └─<missing> Virtual Size: 11.4 MB
  │   └─<missing> Virtual Size: 11.4 MB
  │     └─f8103909759b Virtual Size: 167.1 MB Tags: frolvlad/alpine-oraclejdk8:slim
  │       └─b863e0eb478d Virtual Size: 167.1 MB
  │         ├─2482d902b932 Virtual Size: 200.8 MB
  │         │ └─758cd2466b14 Virtual Size: 234.5 MB
  │         │   └─77e64eee6fc6 Virtual Size: 234.5 MB
  │         ├─3223c6766ffe Virtual Size: 200.8 MB
  │         │ └─da97fb8f40e1 Virtual Size: 234.5 MB
  │         │   └─d675720f7ba6 Virtual Size: 234.5 MB
  │         ├─13be904e8623 Virtual Size: 181.3 MB
  │         │ └─b4366036a309 Virtual Size: 195.5 MB
  │         │   └─ae4abd2f09e1 Virtual Size: 195.5 MB Tags: vtvote/vote-tenant-client:latest
  │         ├─bf3f999e5aaa Virtual Size: 200.8 MB
  │         │ └─468b2f7d7fc1 Virtual Size: 234.5 MB
  │         │   └─0178d78c913f Virtual Size: 234.5 MB Tags: [redacted]:[redacted]/vote-web:latest
  │         └─e9f693c90ae6 Virtual Size: 181.3 MB
  │           └─ed3160c48541 Virtual Size: 195.4 MB
  │             └─4b0251738835 Virtual Size: 195.4 MB Tags: springio/gs-spring-boot-docker:latest
```

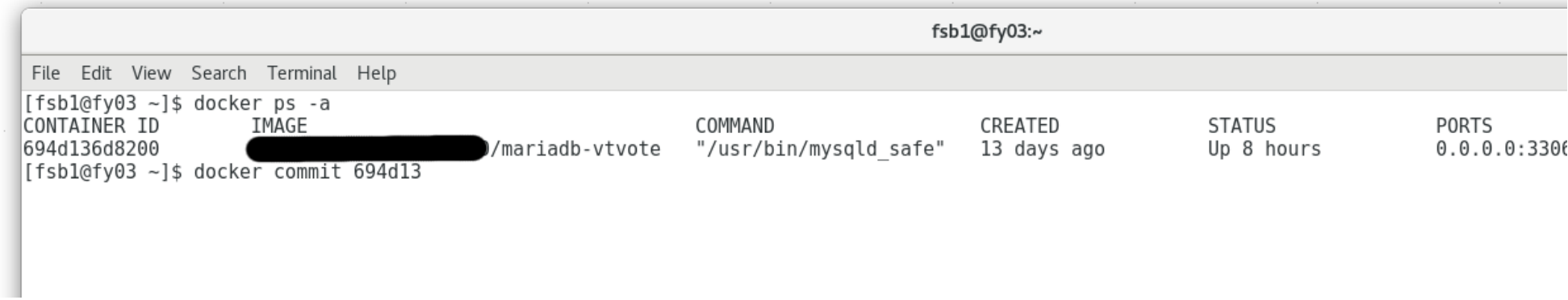
Images Bauen

Via Dockerfile (Automatisiert)

Dockerfile

```
1 FROM frolvlad/alpine-oraclejdk8:slim
2 VOLUME /tmp
3 ADD vote-web-0.0.1-SNAPSHOT.war voteweb.war
4 RUN sh -c 'touch /voteweb.war'
5 ENTRYPOINT ["java","-jar","/voteweb.war"]
```

Via “docker commit <containerid>” (von Hand konfiguriert)



Beispiel Maven Build

```
<plugin>
     <groupId>com.spotify</groupId>
     <artifactId>docker-maven-plugin</artifactId>
     <configuration>
          <imageName>${docker.image.prefix}/${project.artifactId}</imageName>
          <dockerDirectory>${project.basedir}/src/main/docker</dockerDirectory>
          <resources>
               <resource>
                    <targetPath>/</targetPath>
                    <directory>${project.build.directory}</directory>
                    <include>${project.build.finalName}.war</include>
               </resource>
          </resources>
     </configuration>
</plugin>
```

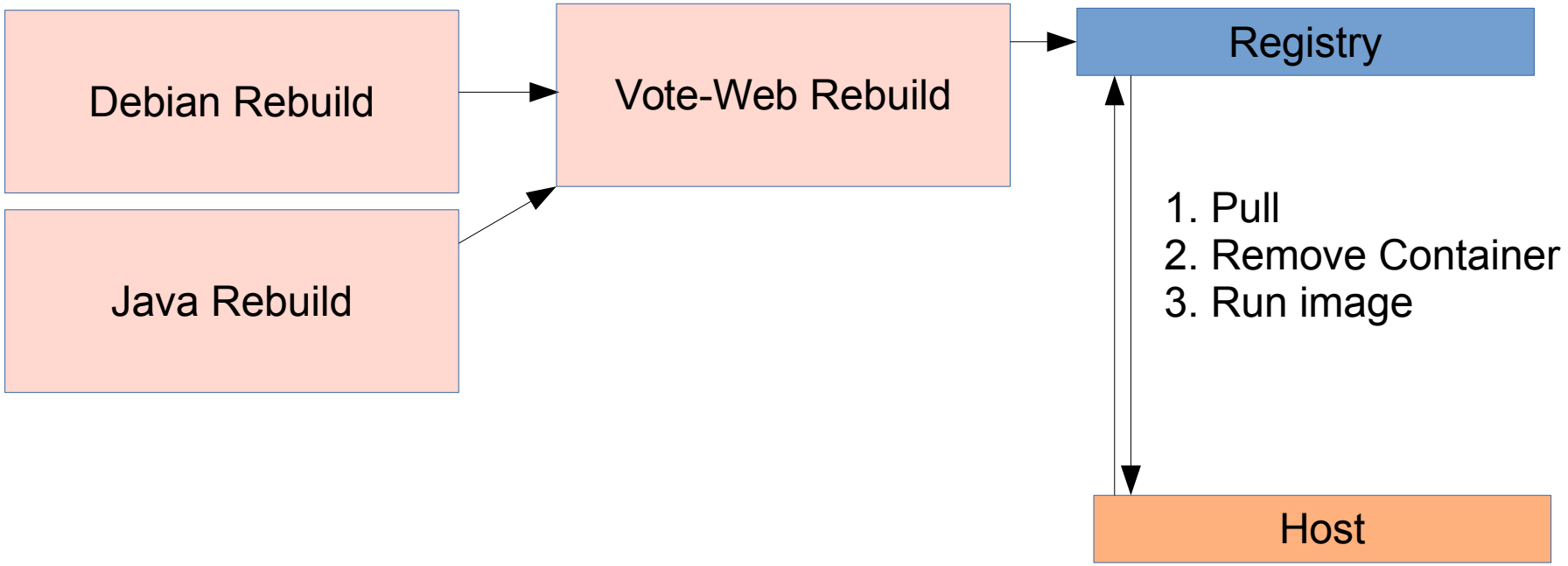
Images verteilen

- Hub: öffentliches Repo
- Registry: Software für privates Repo
  - Verteilung nur über verifiziertes TLS
  - Hashes zur Integrität der Abhängigkeiten
  - Verteilt Differenzen

- Save und Load
  - Verteilung muss selber sichergestellt werden
  - Grosse Dateien (Verteilt Alles)

Wartbarkeit von Images

- Applikation und Daten trennen
- Automatisiert Images updaten

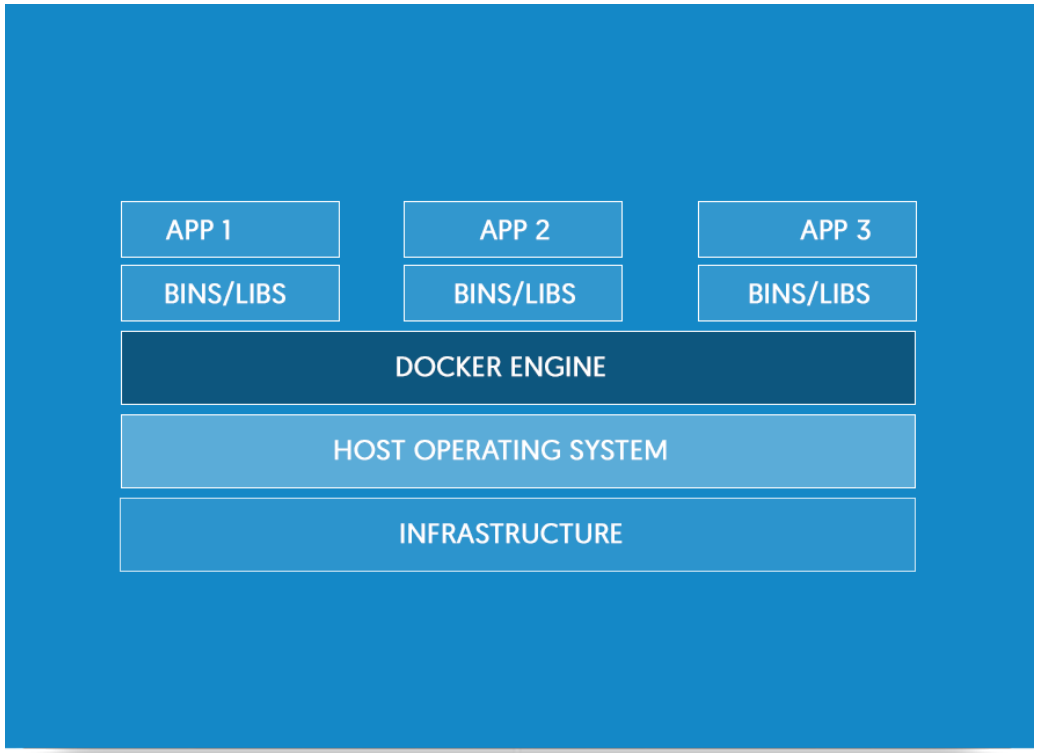


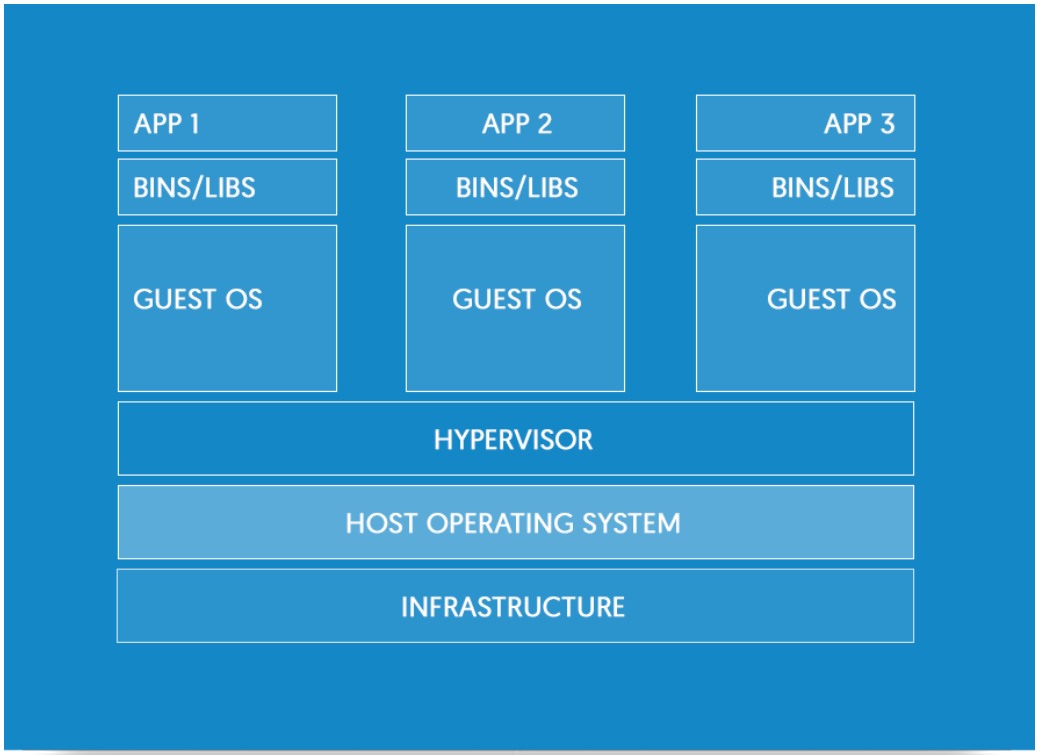
Security

- Container und Security
  - Libs sind unabhängig
    - Updates können ohne Abhängigkeiten installiert werden
    - Verwendete Dienste und Libs sind auf ein minimum reduziert
  - Nur ein Kernel
    - Container Root ist Kernel Root
    - Resourcen können blokiert werden
  - Flexible Erweiterungen

Sicherer Betrieb ist nicht die Aufgabe des Containers.

Docker Engine





Docker

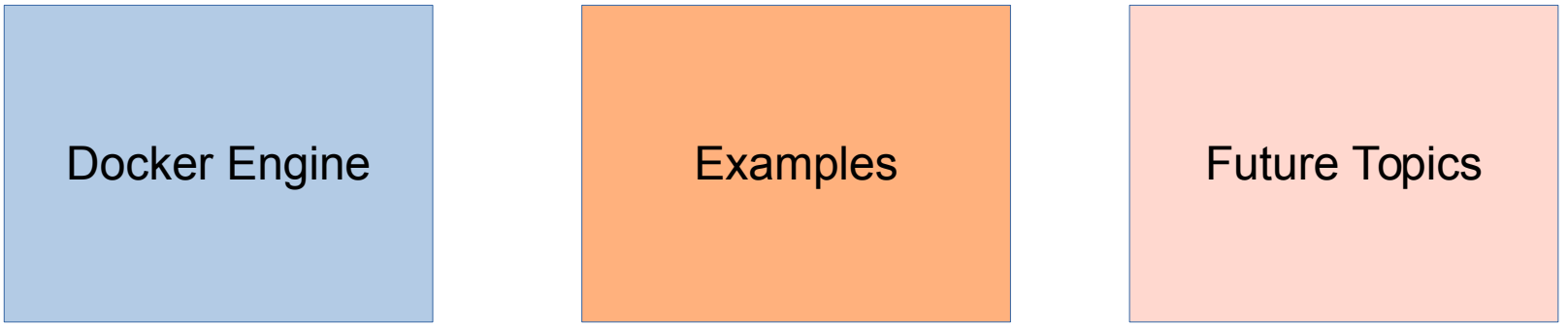
VM

Wiso Docker?

- Container mit Zusatz
  - Container und Images (Docker Engine)
  - Verteilung (Docker Hub)
  - Netzwerk (Docker Network)
  - Remote Management (Docker Machine)
  - Cluster Management (Docker Swarm)
  - Community

Docker Machine

- Remote Management
  - Lokale VM für Windows und Mac
  - Container auf einem Server
  - Container in proprietärer Cloud (eigene Driver)


Docker Engine
Examples
Future Topics

Docker Commands

- Docker run <image> → container
  - (pull,create,start)
- Docker stop <container>
- Docker start <container>
- Docker rm <container>

- Docker ps [-a]
- Docker images

docker Cheat Sheet

REBELLABS by ZEROTURNAROUND

Glossary

Layer - a set of read-only files to provision the system

Image - a read-only layer that is the base of your container. Might have a parent image

Container - a runnable instance of the image

Registry / Hub - central place where images live

Docker machine - a VM to run Docker containers (Linux does this natively)

Docker compose - a utility to run multiple containers as a system

Useful one-liners

Download an image

```
docker pull image_name
```

Start and stop the container

```
docker [start|stop] container_name
```

Create and start container, run command

```
docker run -ti --name container_name
    image_name command
```

Create and start container, run command, destroy container

```
docker run --rm -ti image_name command
```

Example filesystem and port mappings

```
docker run -it --rm -p 8080:8080 -v
/path/to/agent.jar:/agent.jar -e
JAVA_OPTS="-javaagent:/agent.jar"
tomcat:8.0.29-jre8
```

Docker cleanup commands

Kill all running containers

```
docker kill $(docker ps -q)
```

Delete dangling images

```
docker rmi $(docker images -q -f
    dangling=true)
```

Remove all stopped containers

```
docker rm $(docker ps -a -q)
```

Docker machine commands

Use docker-machine to run the containers

Start a machine

```
docker-machine start machine_name
```

Configure docker to use a specific machine

```
eval "$(docker-machine env machine_name)"
```

Docker compose syntax

docker-compose.yml file example

```
version: "2"
services:
web:
  container_name: "web"
  image: java:8 # image name
  # command to run
  command: java -jar /app/app.jar
  ports: # map ports to the host
    - "4567:4567"
  volumes: # map filesystem to the host
    - ./myapp.jar:/app/app.jar
mongo: # container name
  image: mongo # image name
```

Create and start containers

```
docker-compose up
```

Interacting with a container

Run a command in the container

```
docker exec -ti container_name command.sh
```

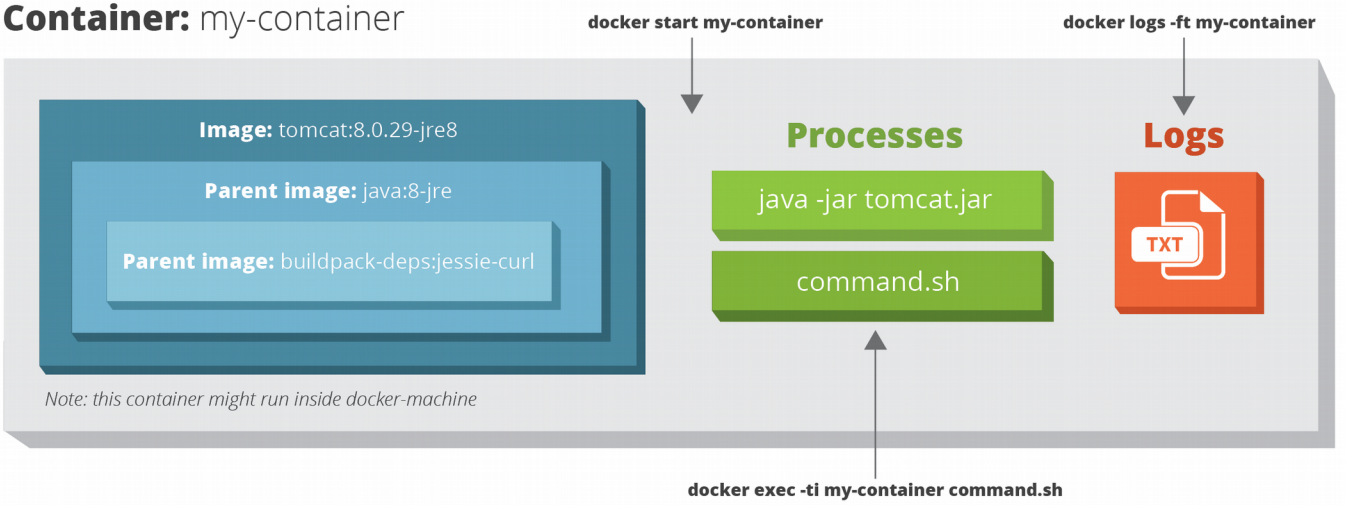
Follow the container logs

```
docker logs -ft container_name
```

Save a running container as an image

```
docker commit -m "commit message" -a "author"
    container_name username/image_name:tag
```

Container: my-container



Examples

- Aktueller Build auf der Dev Platform
- Entwickler DB direkt von der Registry

- Images aktuell halten

- Docker Netzwerk

- Docker Management mit Docker

- Docker-compose (Komplettes vote-web mit Loadbalancing)

Future Topics

- Cluster Management with Rancher
- Docker Overlay Networks
- Docker Cloud

“Docker-Container in der Cloud”

Quelle: http://www.shiplilly.com/wp-content/uploads/2013/09/Funny-shipping.jpg

Fragen?

Quellen

- http://paislee.io/content/images/2015/03/blue_whale_container_ship.jpg
- http://docker.com
- https://zeroturnaround.com/rebellabs/docker-commands-and-best-practices-cheat-sheet/
- http://www.shiplilly.com/wp-content/uploads/2013/09/Funny-shipping.jpg