docker

für Entwickler

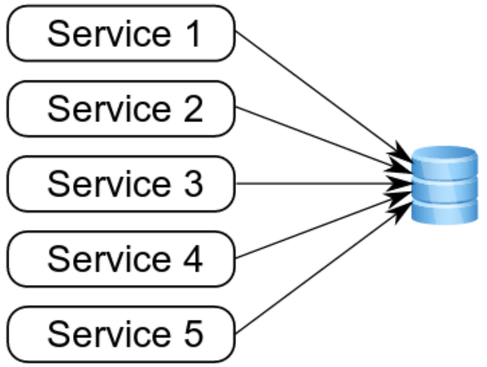
Eric Weikl

BTD 7, 23.05.2014

Das Problem...

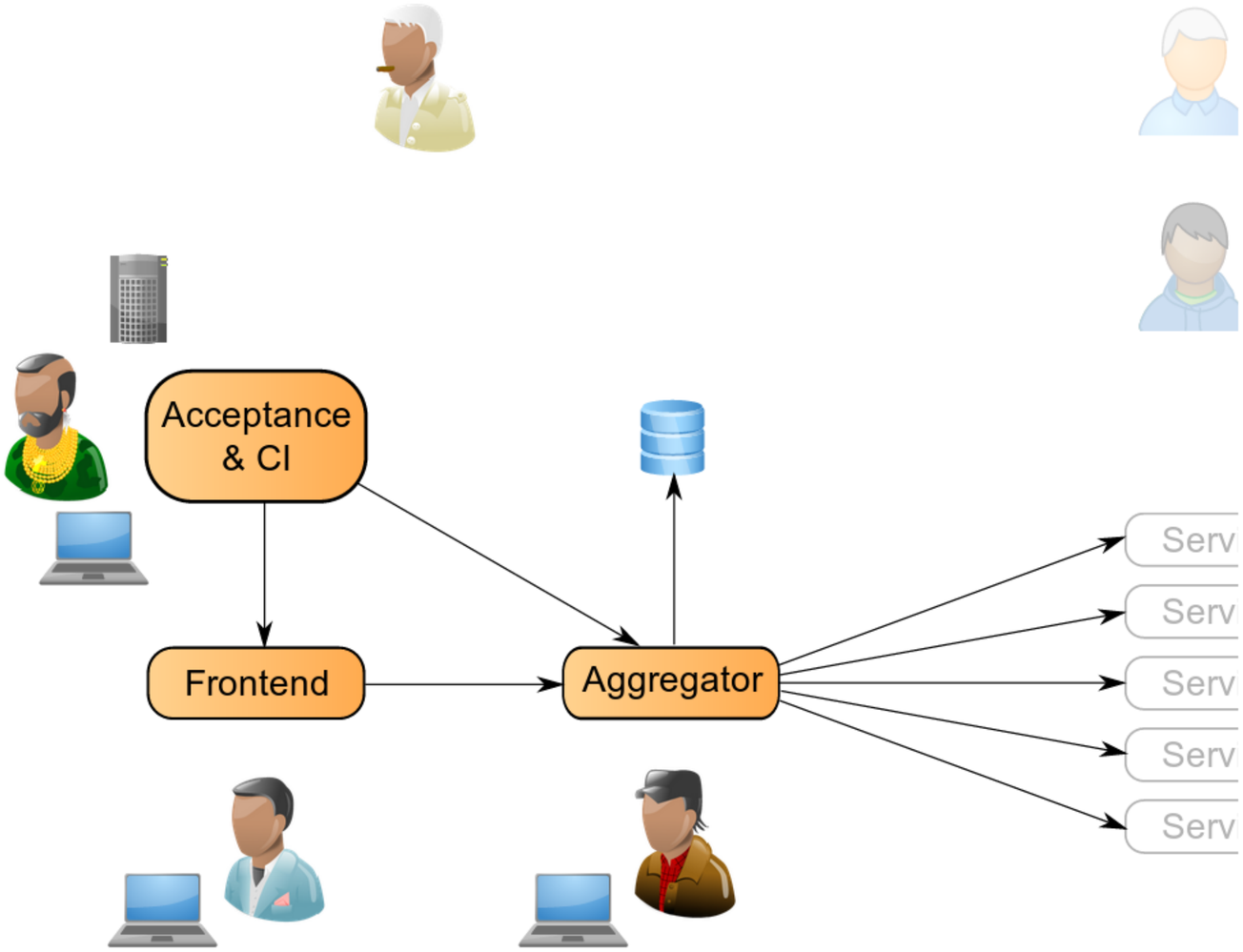
Team A

Team B



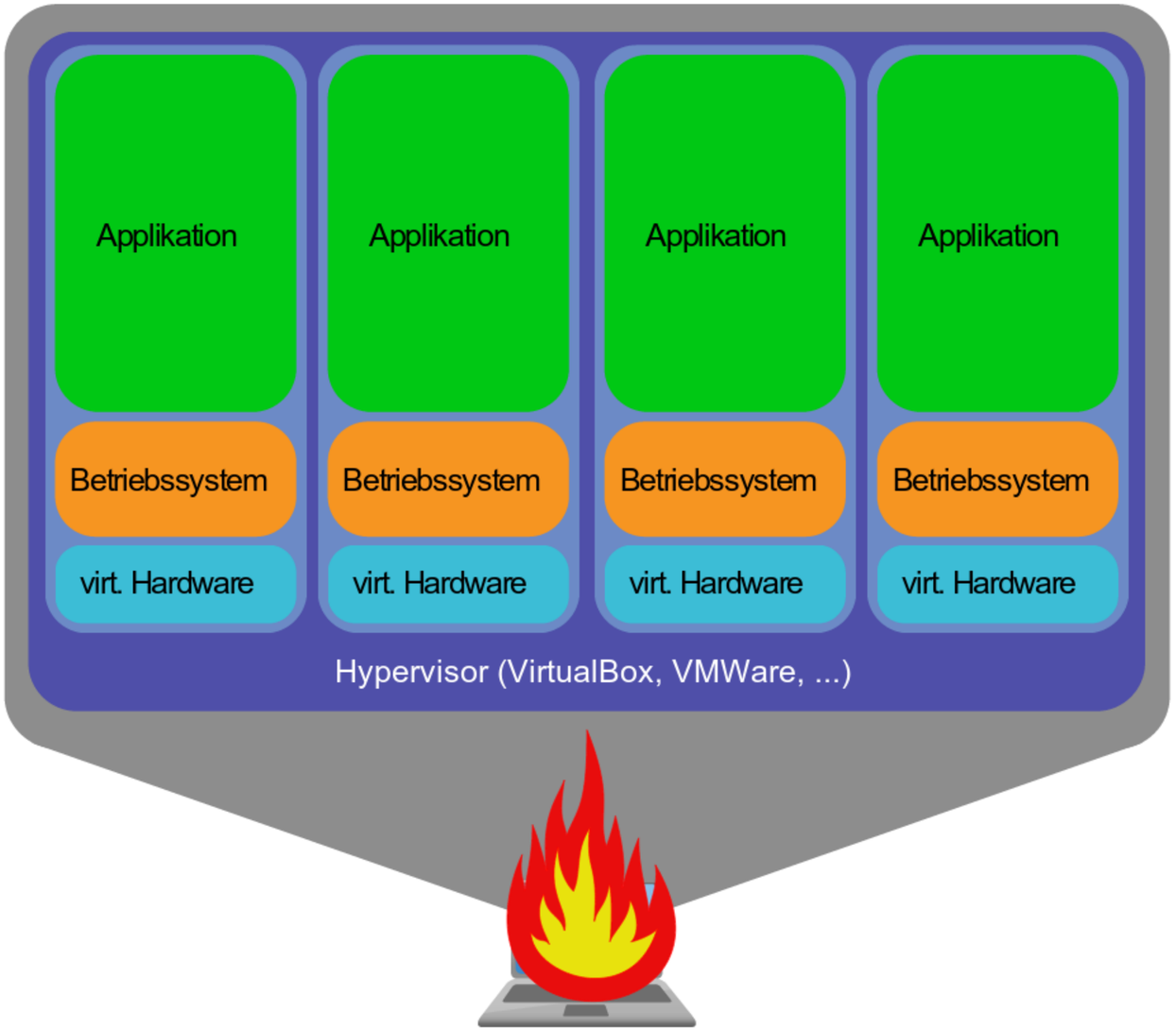
A-Team

Te

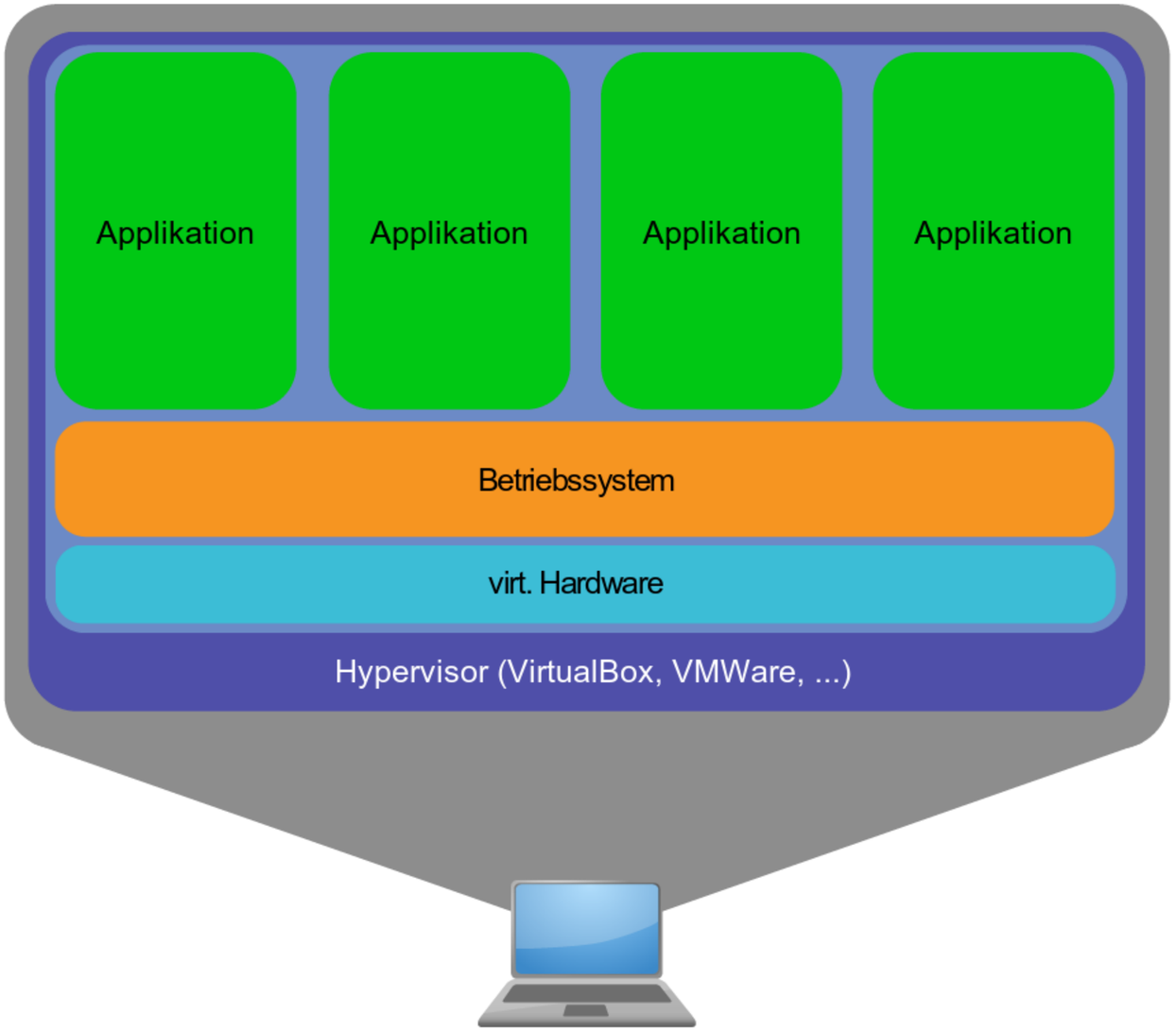

Acceptance
& CI
Frontend
Aggregator
Serv
Serv
Serv
Serv
Serv

Probleme mit Virtualisierung

Wie viele VMs?



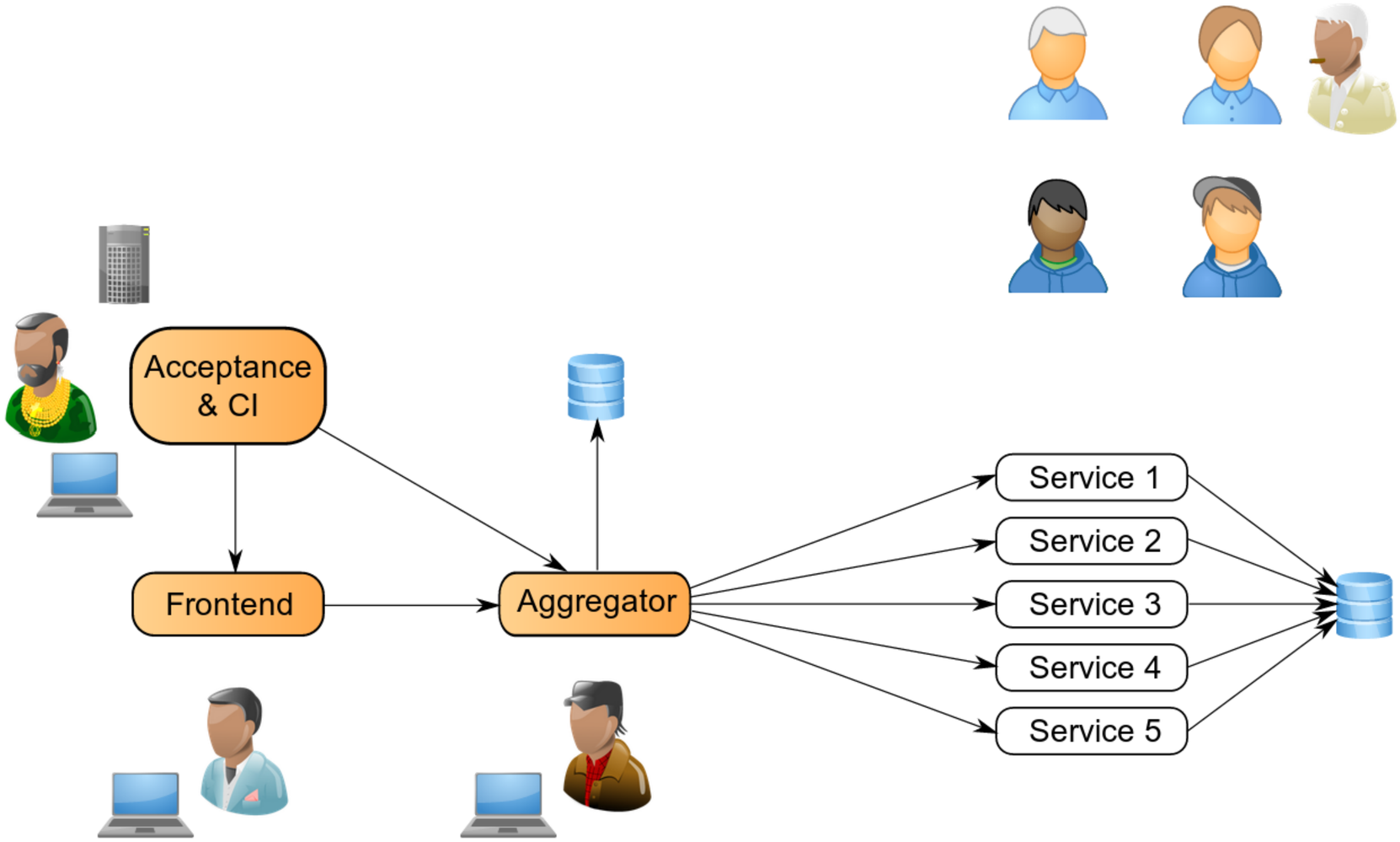
Wie viele VMs?



Puppet vs. Golden Images

A-Team

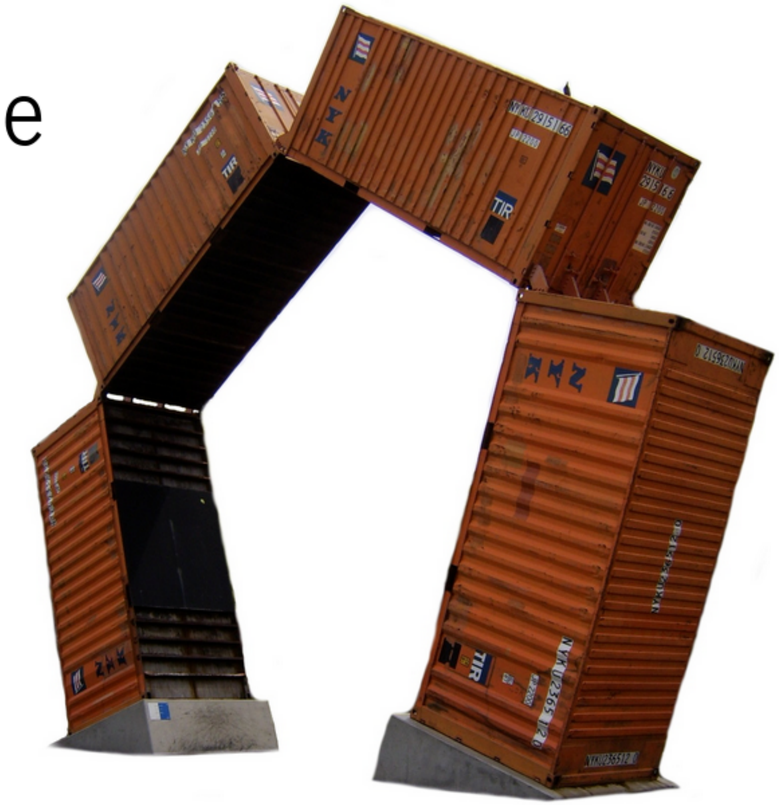
Team B


Acceptance & CI
Frontend
Aggregator
Service 1
Service 2
Service 3
Service 4
Service 5


docker

Container

= isoliertes Dateisystem
+ virtuelles Netzwerk-Interface
+ Prozessisolation
+ CPU / Memory (opt.)



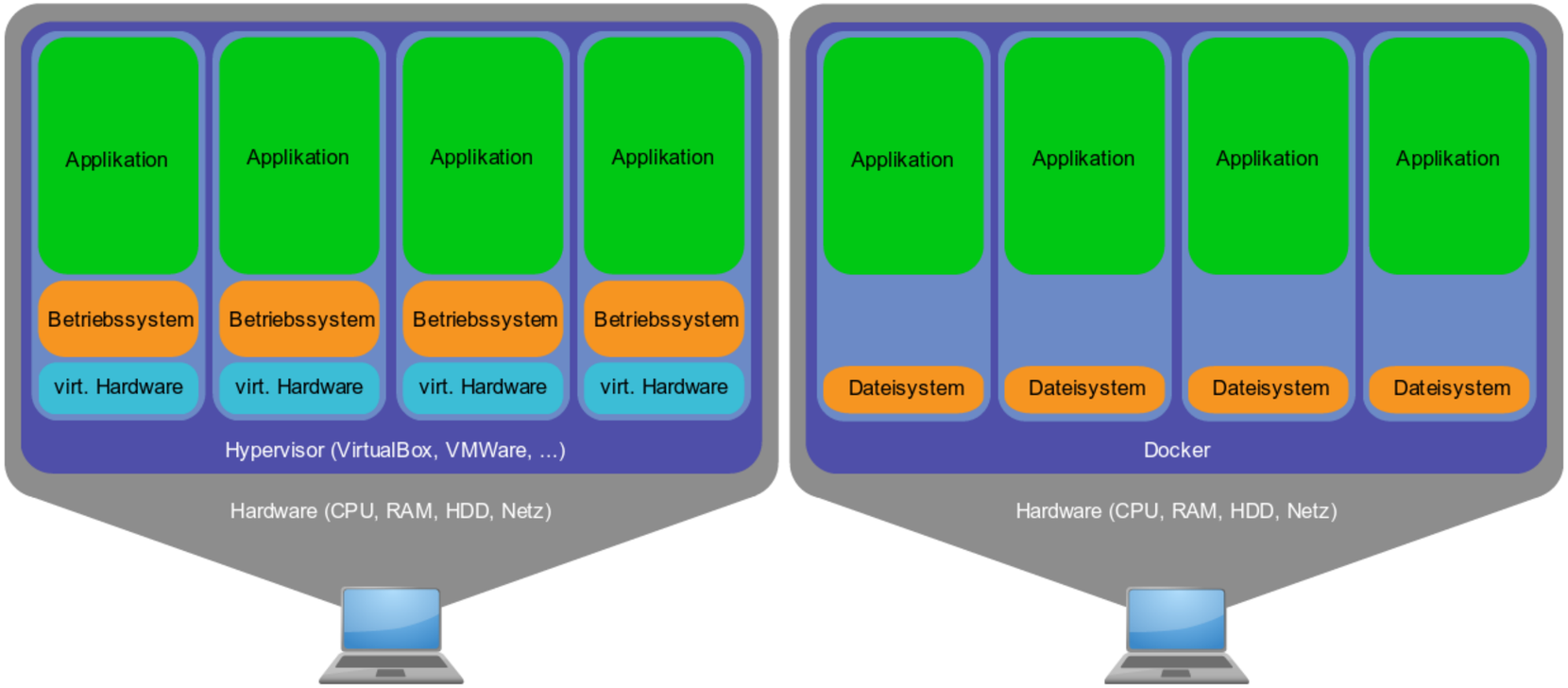
"Build once, run anywhere"*

* Solange anywhere == Linux mit aktuellem Kernel ist

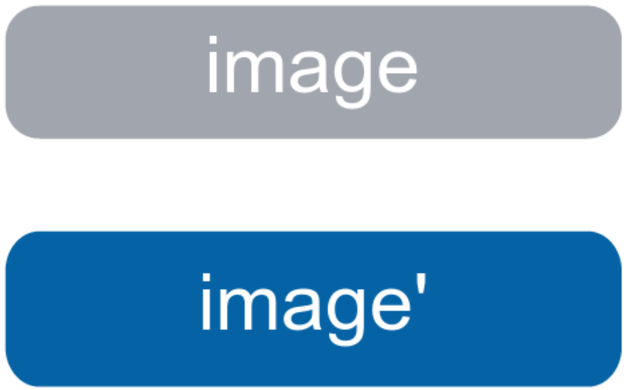
Unterschiede

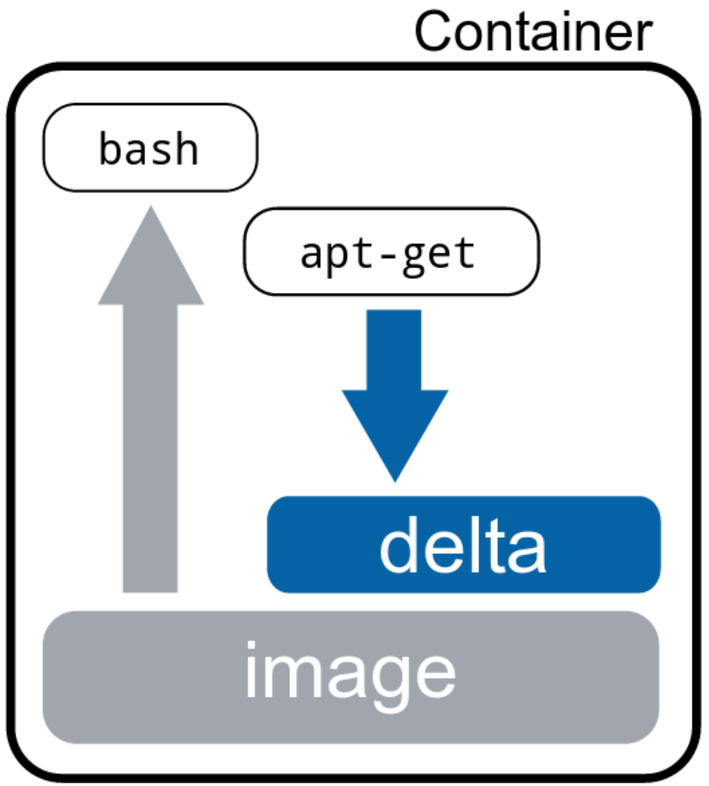
	VM	Container
Hardware	Virtualisiert	Host
Betriebssystem	Pro VM	1 Kernel, Dateisystem pro Container
Snapshots	Dateisystem & RAM	Dateisystem
Läuft bis	Shutdown	Hauptprozess endet

Vergleich Ressourcen

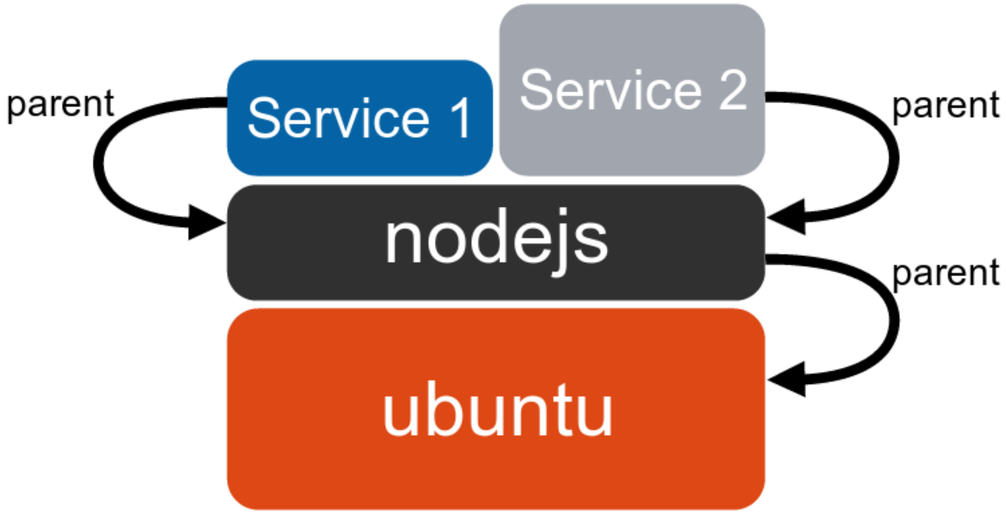


Docker Images





Gestapelte Images



Demo

(http://ericweikl.github.io/docker-playground/)

Dockerfile

```
FROM <image>
MAINTAINER <you>

RUN <step 1>
RUN <step 2>
RUN <step 3>

ADD <source> <target>
```

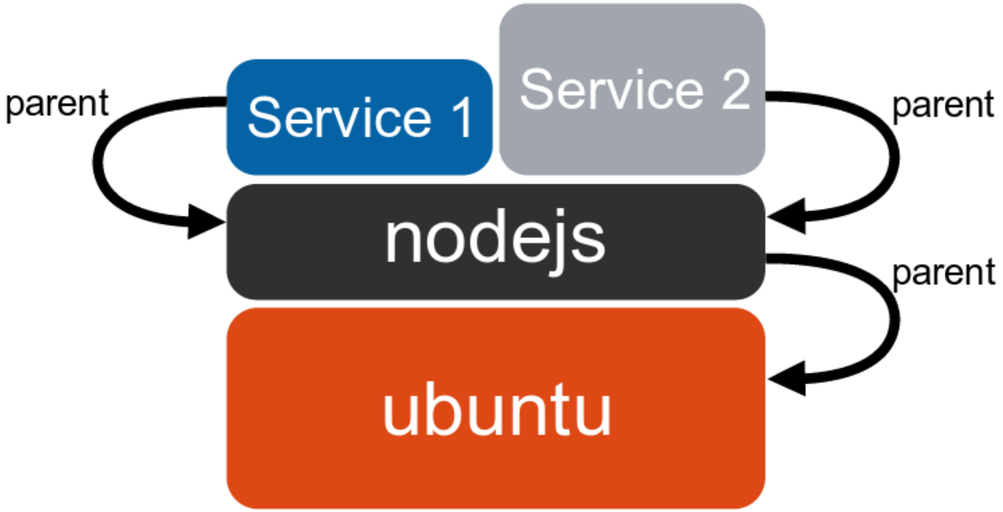
Beispiel Dockerfile

```
FROM ubuntu:trusty
MAINTAINER Eric Weikl <eric.weikl@tngtech.com>

RUN echo 'Hello World from Dockerfile!' > /hello.txt
```

```
$ docker build -t ericweikl/helloworld .
```

Docker bei Team B



NodeJS Image

```
FROM ubuntu:trusty
MAINTAINER Team B <team-b@example.com>

ENV NODE_PATH /usr/local/lib/node_modules

RUN apt-get install -y nodejs npm
```

```
$ docker build -t teamb/nodejs .
```

Service Image

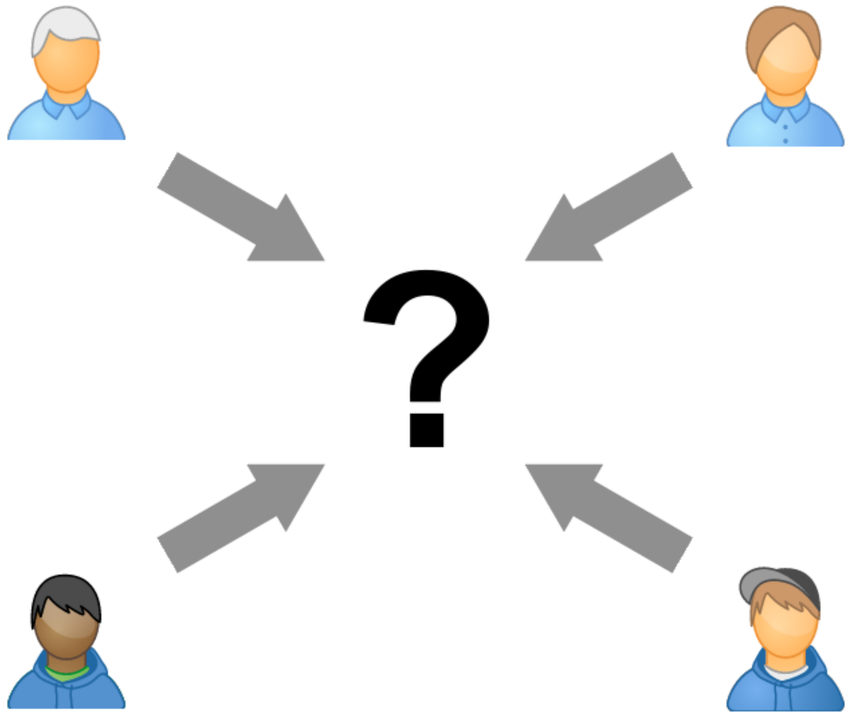
```
FROM teamb/nodejs
MAINTAINER Team B <team-b@example.com>

RUN npm install --global winston && npm install --global mysql

RUN mkdir -p /opt/service1/logs
ADD src/ /opt/service1/
ENTRYPOINT ["nodejs", "/opt/service1/main.js"]
```

```
$ docker run -d \
    -v /var/log/service1:/opt/service1/logs \
    -p 8001:8000 \
    teamb/service1 \
    dev
```

Wie tauscht Team B Images aus?



Docker Registry

Docker Index

http://index.docker.io

ubuntu:trusty

teamb/service1

Private Registry

http://registry.example.com:5000

registry.example.com:5000/teamb/service1

```
$ docker push registry.example.com:5000/teamb/service1
```

Team B Build Pipeline

1. Unit-Tests ausführen
2. Image bauen
3. Integrationstests mit Containern ausführen
4. Erfolg: Image hochladen
5. Fehlschlag: Container kann auf CI-Server eingesehen werden

Was hat das A-Team gelernt?

- Mit Docker können Entwickler ganze Umgebungen auf Ihrer Maschine aufbauen
- Container sind ein leichtgewichtiger, pragmatischer Ersatz für Virtualisierung
- Images bauen aufeinander auf
- Images können über eine Registry mit anderen geteilt werden
- Die Isolation von Containern senkt die Bedeutung vieler Hilfsmittel

Wie macht das A-Team weiter?

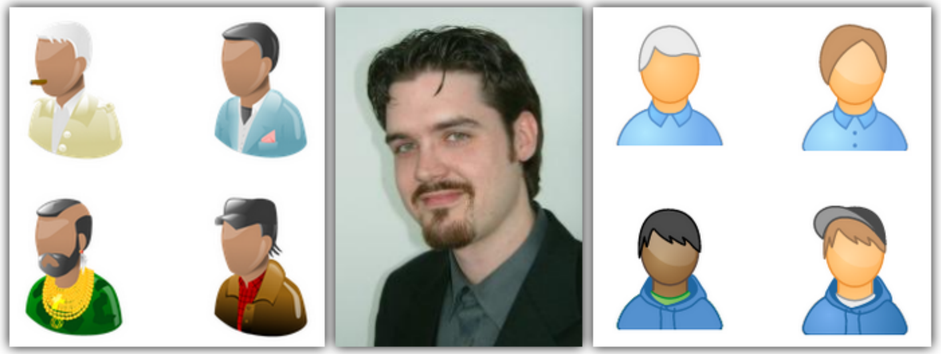
- Eine VM mit Docker (Vagrant mit Puppet)
- Images von Team B aus Registry beziehen
- Java-Teile wie bisher testen
- Mittelfristig: eigene Images für Java-Teile und DB
- Langfristig: Docker Deployment auf Produktion

Wie machen **Sie** weiter?

```
$ git clone https://github.com/ericweikl/docker-playground
$ cd docker-playground
$ vagrant up
```

- Welcher Teil Ihrer Infrastruktur hat Sie schon immer genervt?
- Bauen Sie Ihr erstes Image!

Vielen Dank!



eric.weikl@tngtech.com
https://www.xing.com/profile/Eric_Weikl
@ericweikl

TNG TECHNOLOGY CONSULTING

Bilder

- Flames (modifiziert) — http://commons.wikimedia.org/wiki/File:Flames_by_mimooh.svg
- CD/DVD Stock Image — https://www.flickr.com/photos/seftonbillington/4664509637
- Marionette puppet silhouette free icon — Icon made by Freepik from www.flaticon.com
- Container Arch (modifiziert) — https://www.flickr.com/photos/radiomacguys/459903013