



science + computing

| an atos company



# Der Wal im Windkanal

Docker Container für Scientific Computing

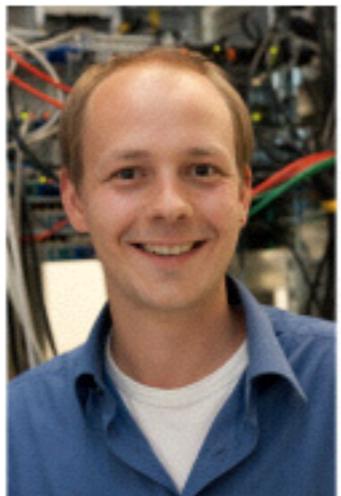
**TUEBIX - Juni 2016**

**Dipl.-Inform. (FH) Holger Gantikow**

science + computing ag

IT-Dienstleistungen und Software für anspruchsvolle Rechnernetze

Tübingen | München | Berlin | Düsseldorf



## Holger Gantikow

Dipl.-Inform. (FH)

**System Engineer**  
science + computing ag STANDARD

72070 Tübingen  
Deutschland

## Persönliches

---

**Ich suche**

neue Kontakte

---

**Ich biete**

Abgeschlossene Diplomarbeit ("Virtualisierung im Kontext von Hocherfügbarkeit"), IT-Know-How, Erfahrung mit Linux, speziell Debian&Red Hat, Windows, Mac OS X, Solaris, \*BSD, HP-UX, AIX, Netzwerkadministration, Netzwerktechnik, Hardware, Asterisk, VoIP-Systeme, Server Administration, Cluster Computing, Hochverfügbarkeit, Virtualisierung, HA, HPC, Autor von Fachartikeln zum Thema Cloud Computing, RHCT, RHSA, Python Programmierung, Neugierde, Flexibilität

---

**Interessen**

IT-spezifisch momentan: Virtualisierung (Xen, ESX, ESXi, KVM), Cluster Computing (HPC, HA), Cloud Computing (speziell: IaaS, HPCaaS), OpenNebula, OpenSolaris, ZFS, XMPP, SunRay ThinClients - ansonsten: Freie Software, Musik, Gitarre, Fotografie



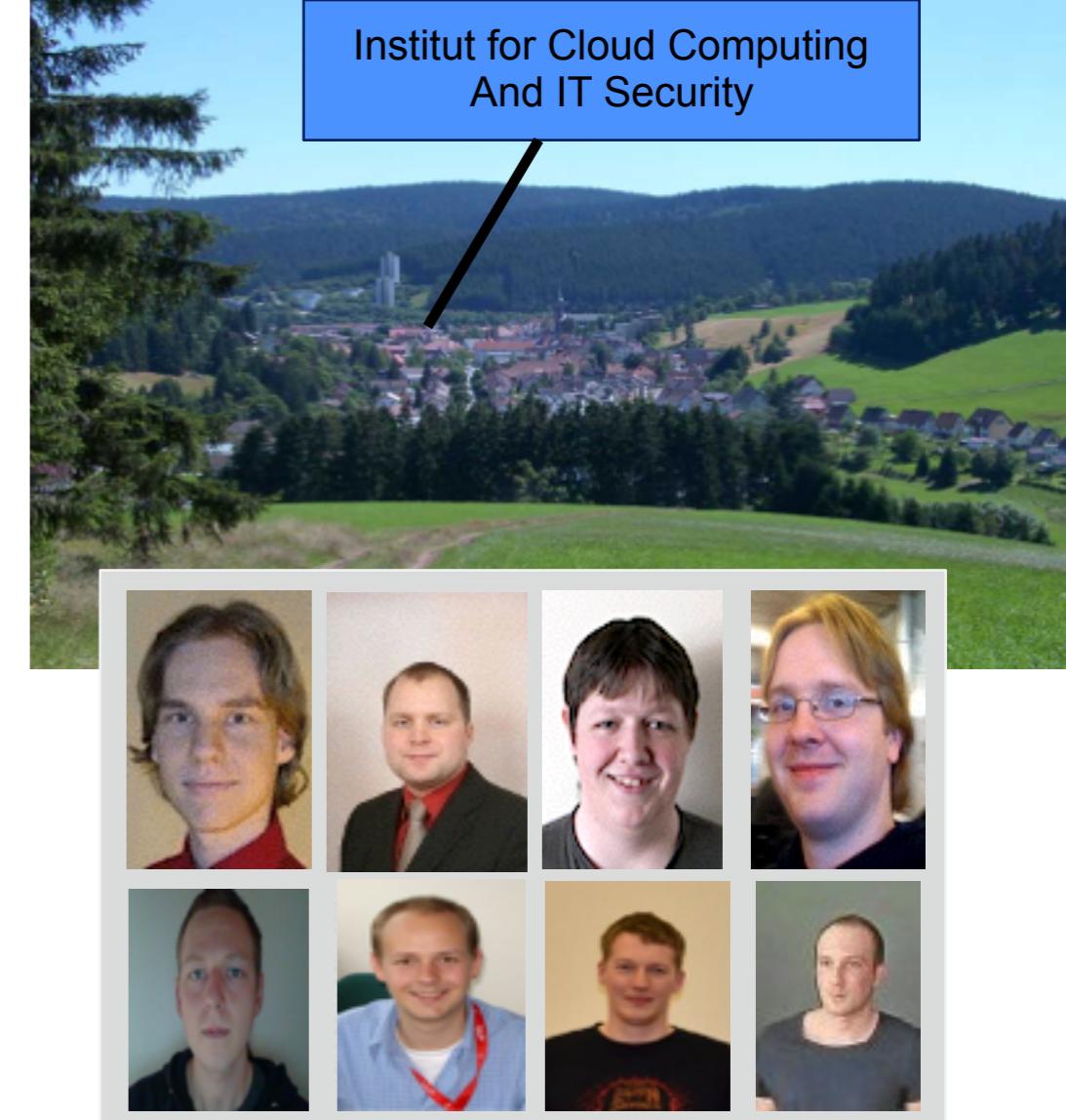
[https://www.xing.com/profile/Holger\\_Gantikow](https://www.xing.com/profile/Holger_Gantikow)

# Institut für Cloud Computing und IT-Sicherheit (IfCCITS)

*previous: Cloud Research Lab*

## facts:

- 10/2015 founded
- Head: Prof. Dr. Ch. Reich
- 5 PhDs, 4 Masters, 6 Bachelors
- <http://www.wolke.hs-furtwangen.de>



## research projects:

- Industrie 4.0 (security, data analysis)
- EU: A4Cloud („accountable Cloud“)
- PET Platform as a Service for Ambient Assisted Living Applications

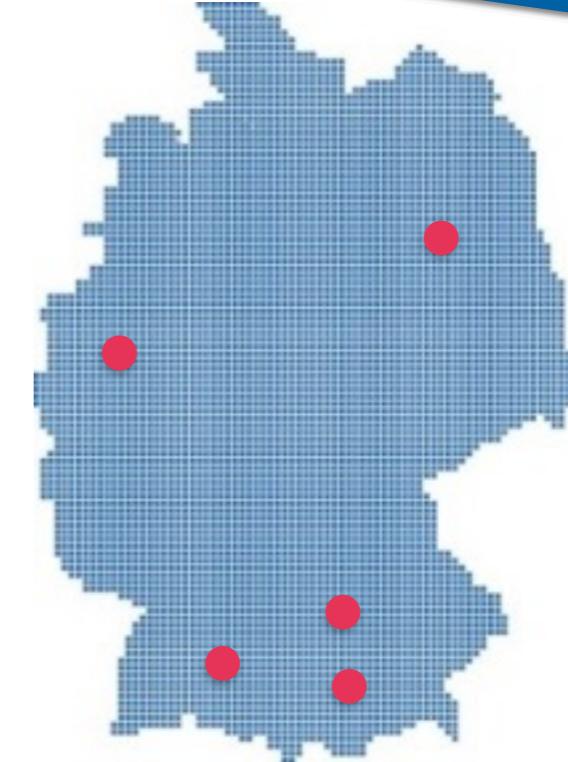
## research topics:

- Distributed Systems
- IT Security
- Cloud Computing
- Industry 4.0; IoT

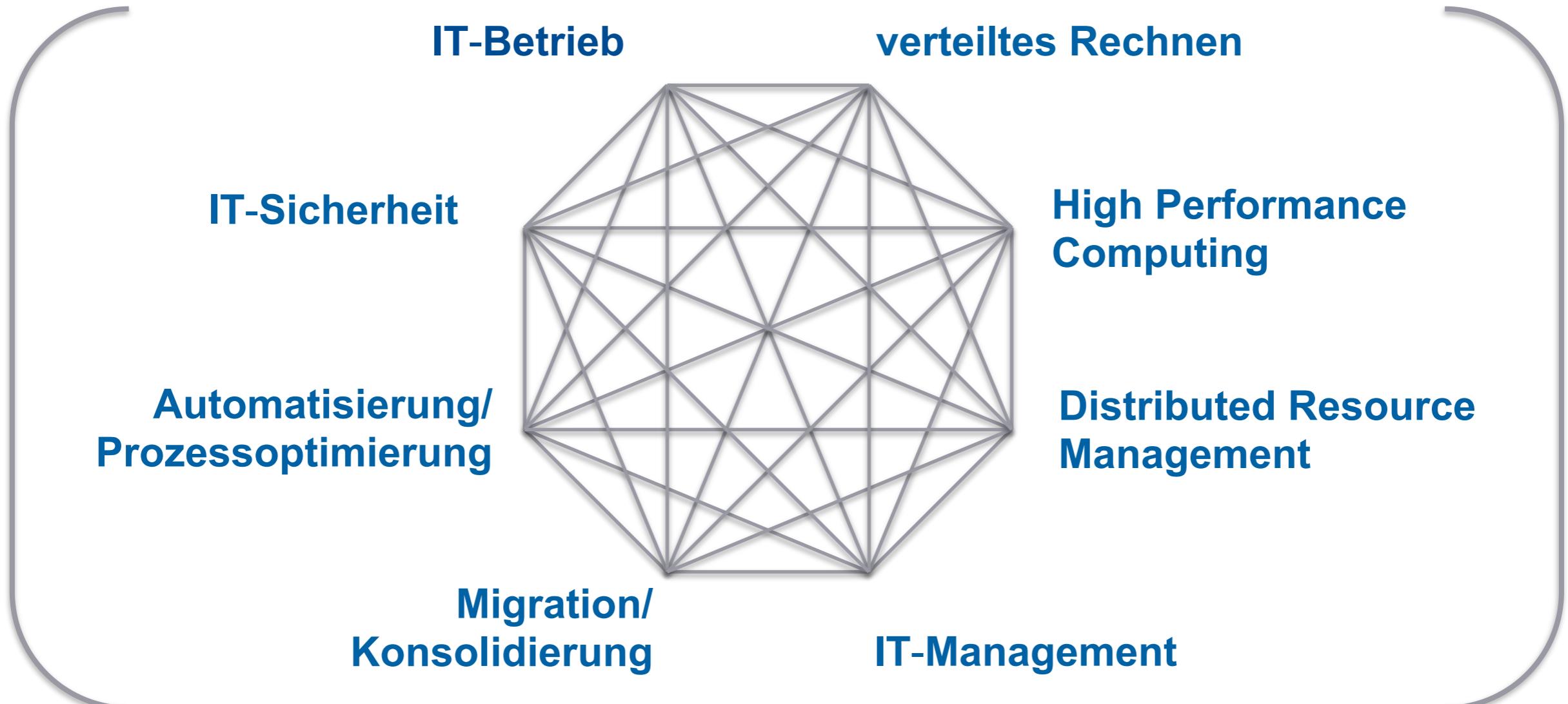
**Gründungsjahr** 1989

**Standorte**  
**Tübingen**  
**München**  
**Berlin**  
**Düsseldorf**  
**Ingolstadt**

**Mitarbeiter** 287  
**Hauptaktionär** Atos SE (100%)  
**davor Bull**  
**Umsatz 2013** 30,70 Mio. Euro

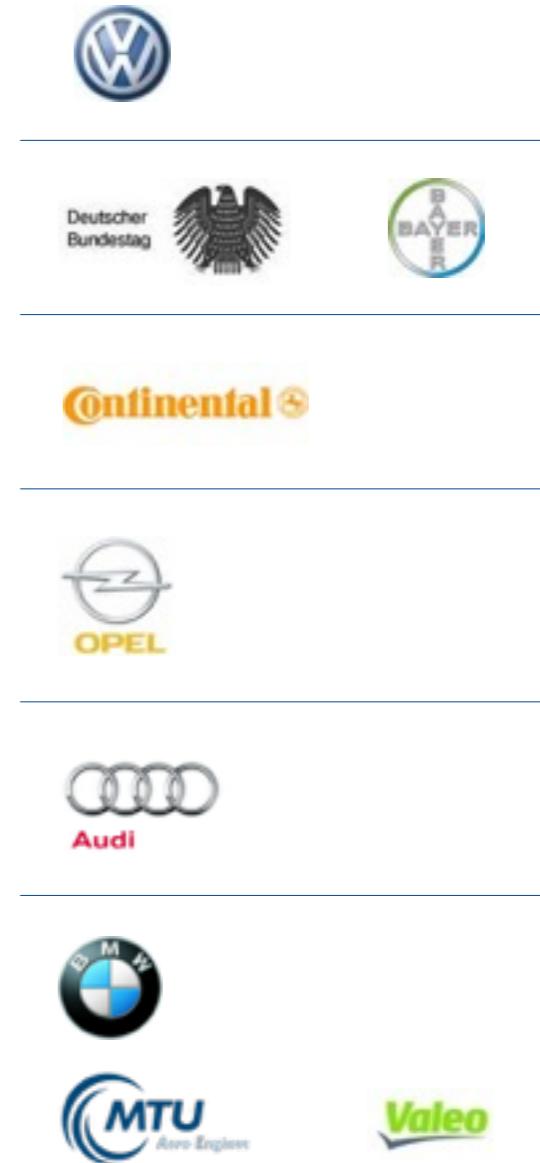
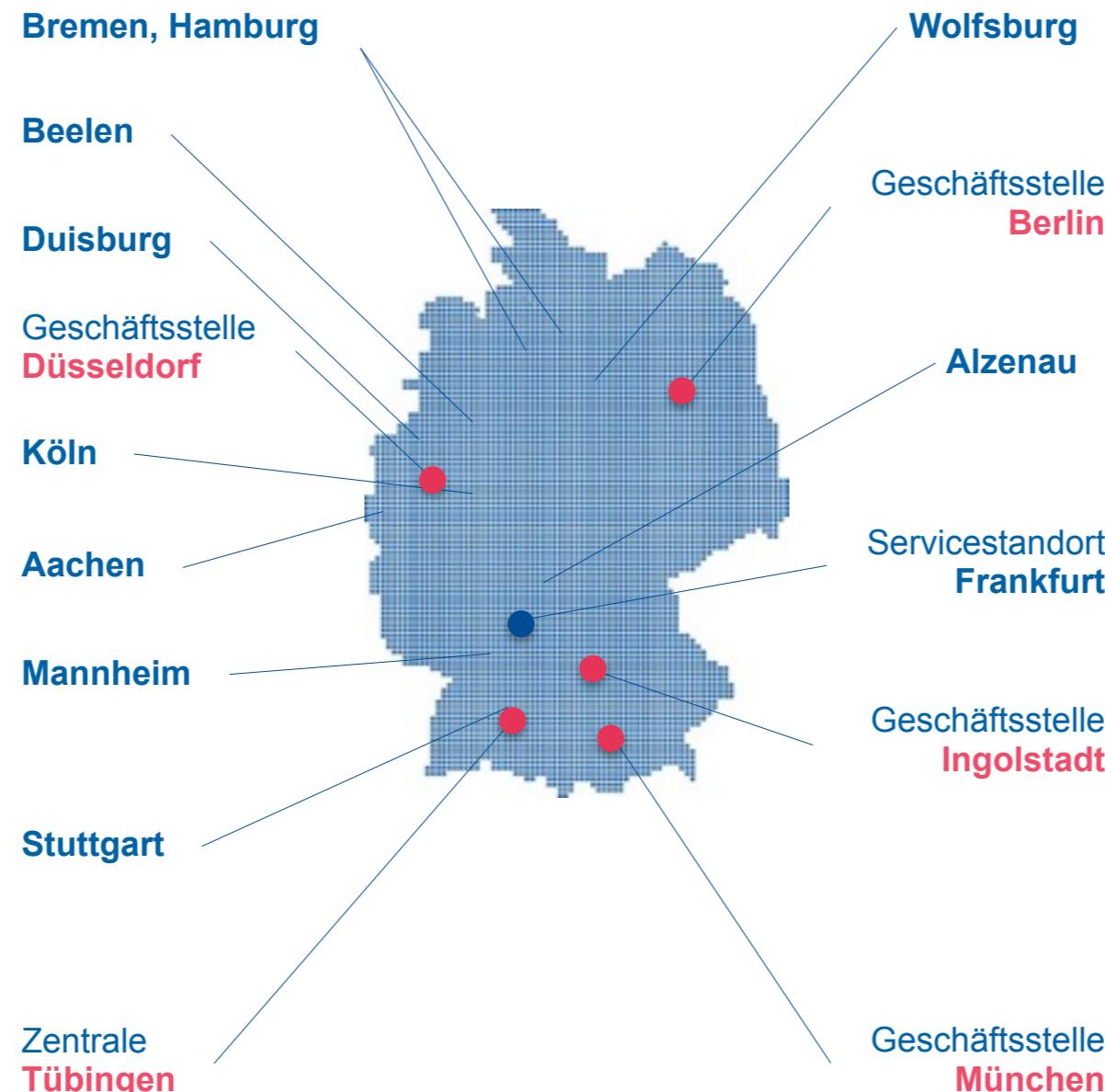


# Anspruchsvolle Rechnernetze effizient nutzen



s+c Kernkompetenzen: **IT Services | Consulting | Software**

# Kunden der science + computing ag



# Agenda

**Teil I: Docker im HPC**

**Teil II: Benchmarks**

**Teil III: Security-Aspekte**

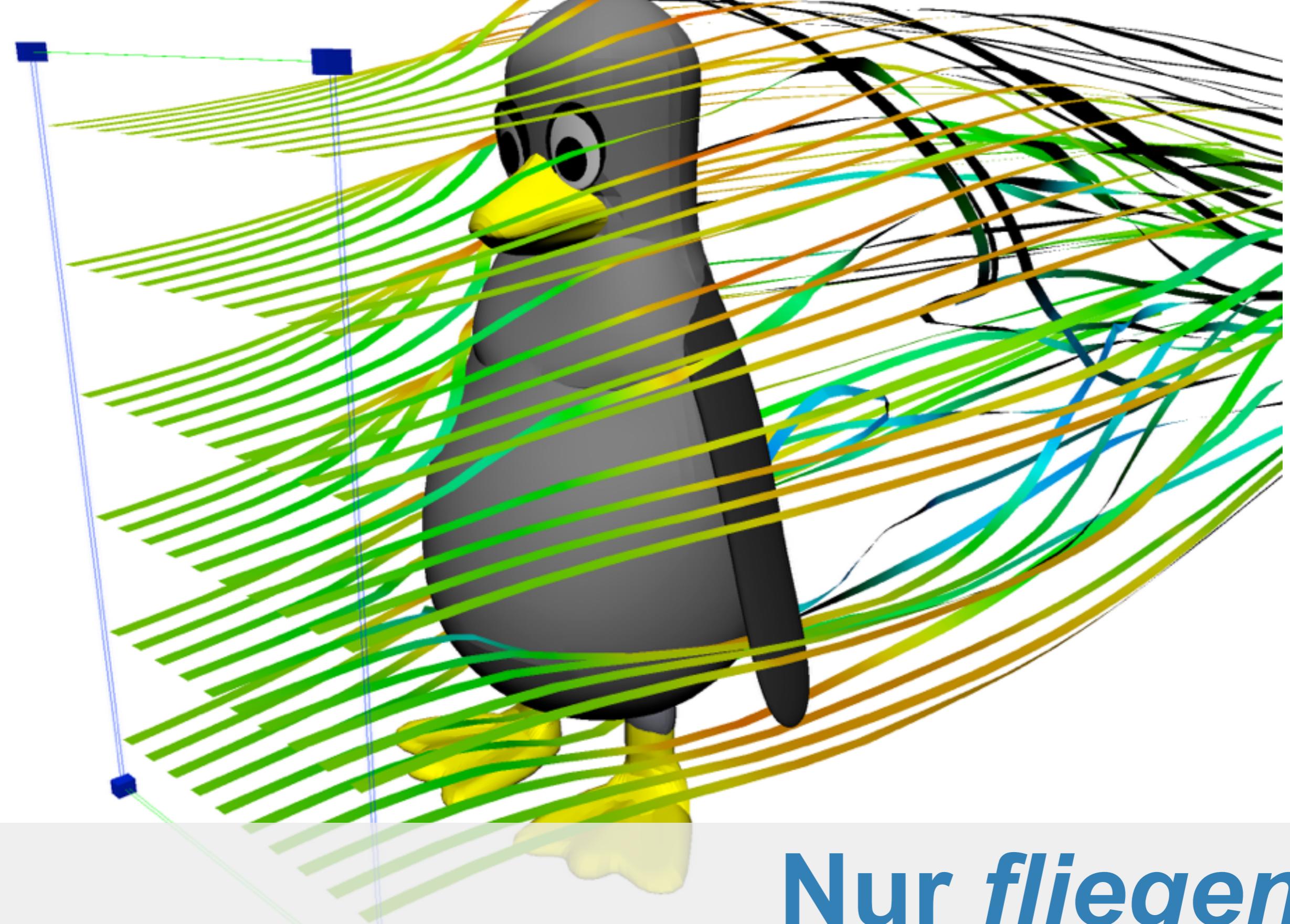
# Teil 0: Einleitung

# Was ist dieses HPC?

# High Performance Computing



Foto: Dieter Both, Bull GmbH



Nur *fliegende*  
Pinguine?

Grafik: Dr. Martin Schulz

# Wissenschaftliches Rechnen

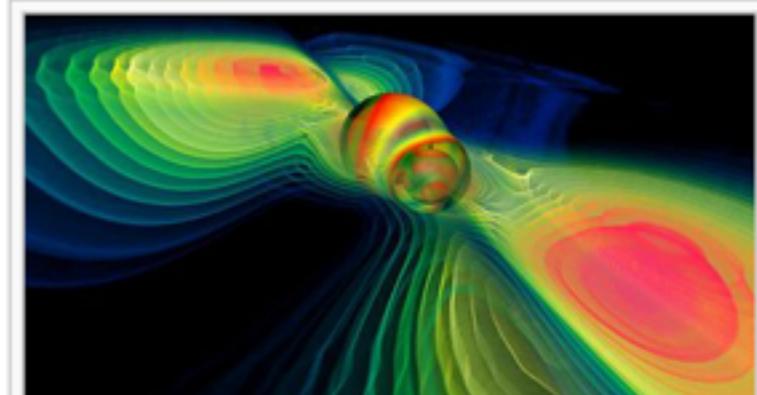
**Wissenschaftliches Rechnen** oder **Simulationswissenschaft** versteht sich als interdisziplinärer Ansatz für die Entwicklung von **Modellen**, **Algorithmen** und **Software**, um mithilfe der **Computersimulation** Fragestellungen aus den **Natur- und Ingenieurwissenschaften** beantworten zu können. Das **Hochleistungsrechnen** ist dabei ein entscheidender Schlüssel. Das wissenschaftliche Rechnen bündelt dazu die **Angewandte Mathematik**, insbesondere die **Numerische Mathematik**, sowie die **Informatik**.<sup>[1]</sup>

Die englische Bezeichnung **Computational Science and Engineering** (CSE) wird auch synonym in der deutschen Sprache verwendet.<sup>[2]</sup> Spezifizierend werden auch andere Begriffe verwendet wie *Scientific Computing* bzw. *Computational Science* (CS) wenn Naturwissenschaften und *Computational Engineering* bzw. *Computational Engineering Science* wenn Ingenieurwissenschaften im Fokus stehen.

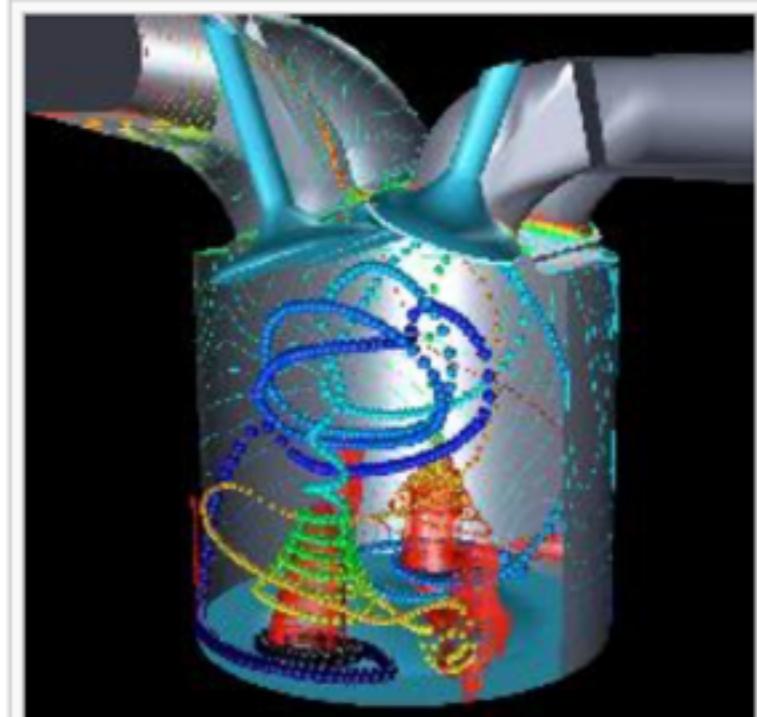
Manchmal wird die **Ingenieurinformatik** mit Computational Engineering gleichgesetzt.<sup>[3]</sup> Jedoch versteht sich die Ingenieurinformatik als eine übergreifende Schnittstelle zwischen Informatik und dem Ingenieurwesen, da Software zunehmend die ingenieurtechnischen Anwendungen durchdringt.<sup>[4]</sup>

## Inhaltsverzeichnis [Verbergen]

- 1 Motivation
- 2 Besondere Förderung der Wissenschaft und Forschung in Deutschland
- 3 Siehe auch
- 4 Literatur
- 5 Weblinks
- 6 Einelnachweise



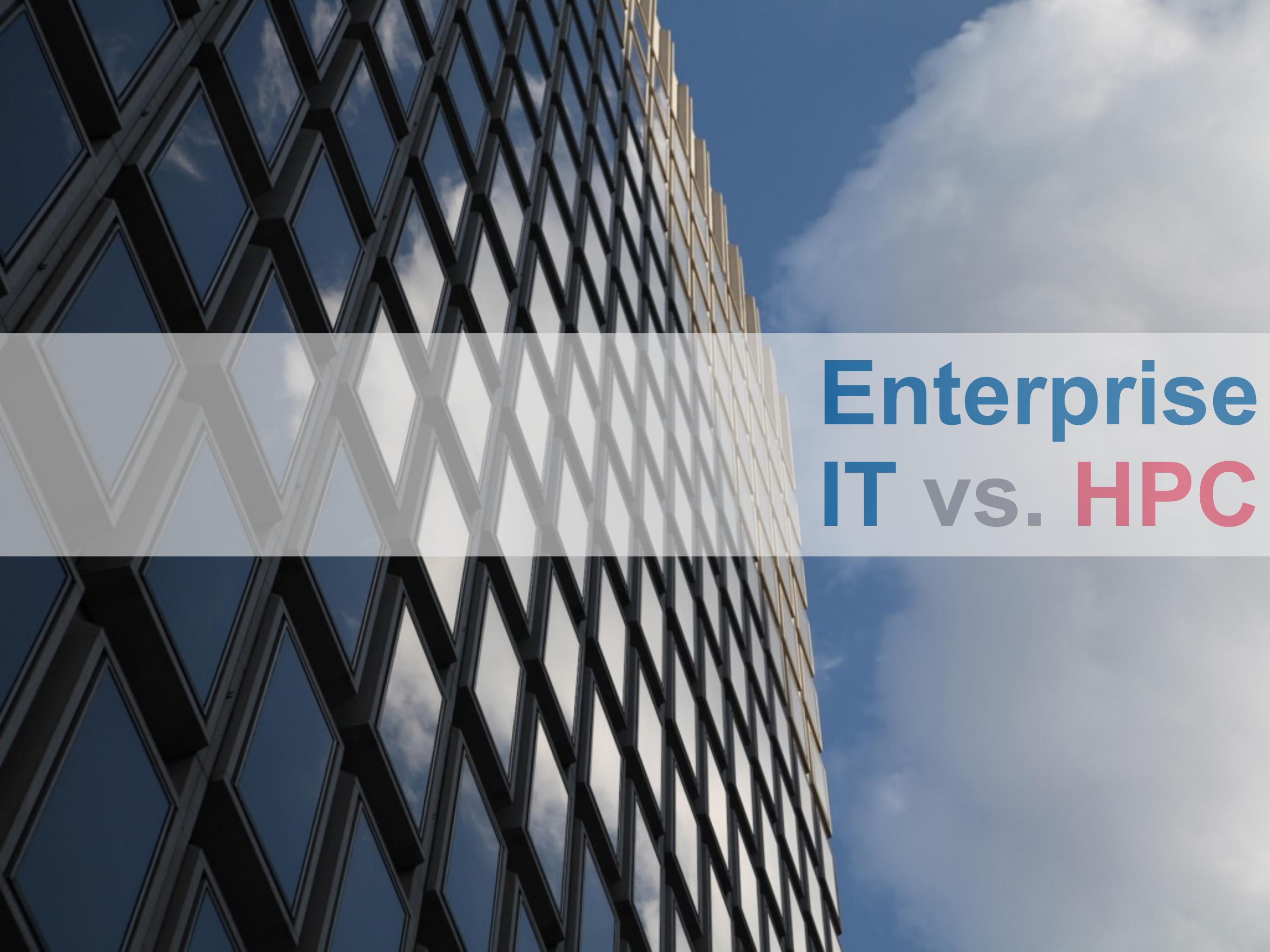
Simulation der Kollision zweier Schwarzer Löcher am Max-Planck-Institut für Gravitationsphysik (2012)



Simulation eines Verbrennungsmotors

## Quelle:

[https://de.wikipedia.org/wiki/Wissenschaftliches\\_Rechnen](https://de.wikipedia.org/wiki/Wissenschaftliches_Rechnen)

The background of the image shows a close-up view of a modern building's exterior. The facade features a grid of large, diamond-shaped windows with dark frames. The windows reflect the bright blue sky and some white clouds. The building's surface is a light-colored, textured material, possibly concrete or a similar stone. The perspective is from a low angle, looking up at the building.

# Enterprise IT vs. HPC

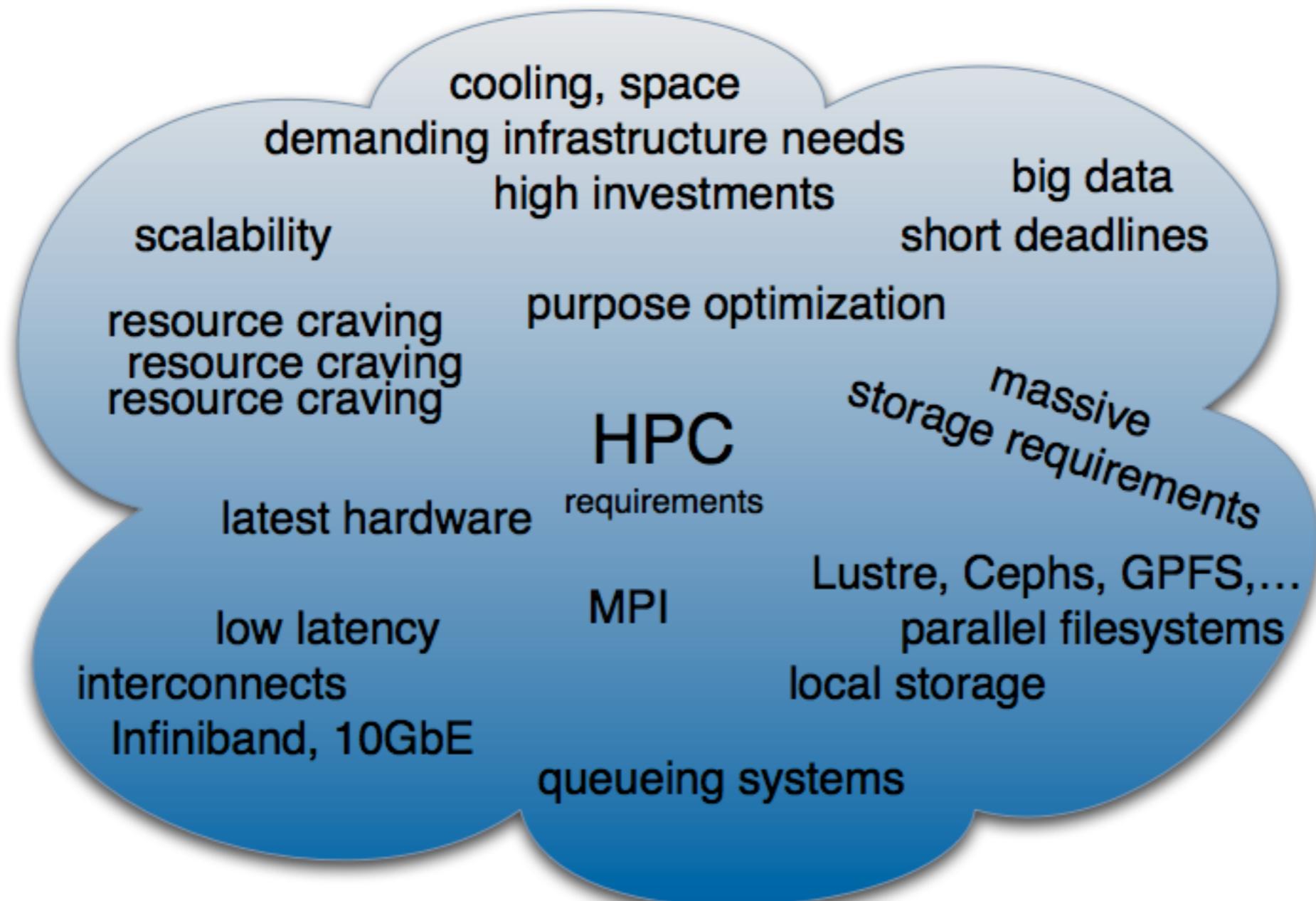
# Enterprise IT vs. HPC

	<b>Enterprise IT</b>	<b>HPC Zentren</b>
<b>Auslastung</b>	< 50%	>>50% (teils bis zu 90%)
<b>Einsatz von Virtualisierung</b>	ja	nein bis selten
<b>Art der Systeme</b>	heterogen	homogen
<b>scale ...</b>	in Konsolidierung via Virtualisierung	out Berechnung verteilt auf mehrere Hosts
<b>Verbindungs-Anforderungen</b>	gemäßigt	Bedarf an latenzarmen Verbindungen mit hohem Durchsatz
<b>Ressourcen-Verbrauch</b>	endlich - meist von Anwenderanzahl abhängig	quasi unendlich - was vorhanden ist wird genutzt
<b>real. Bezugsart (Cloud)</b>	SaaS (Software as a Service)	IaaS (Infrastructure as a Service)
<b>Dateisysteme</b>	zentralisiert	dezentralisiert

## Fazit:

- HPC unterscheidet sich deutlich von Enterprise IT

# HPC-Buzzword-Cloud



# Docker

# DOCKIER



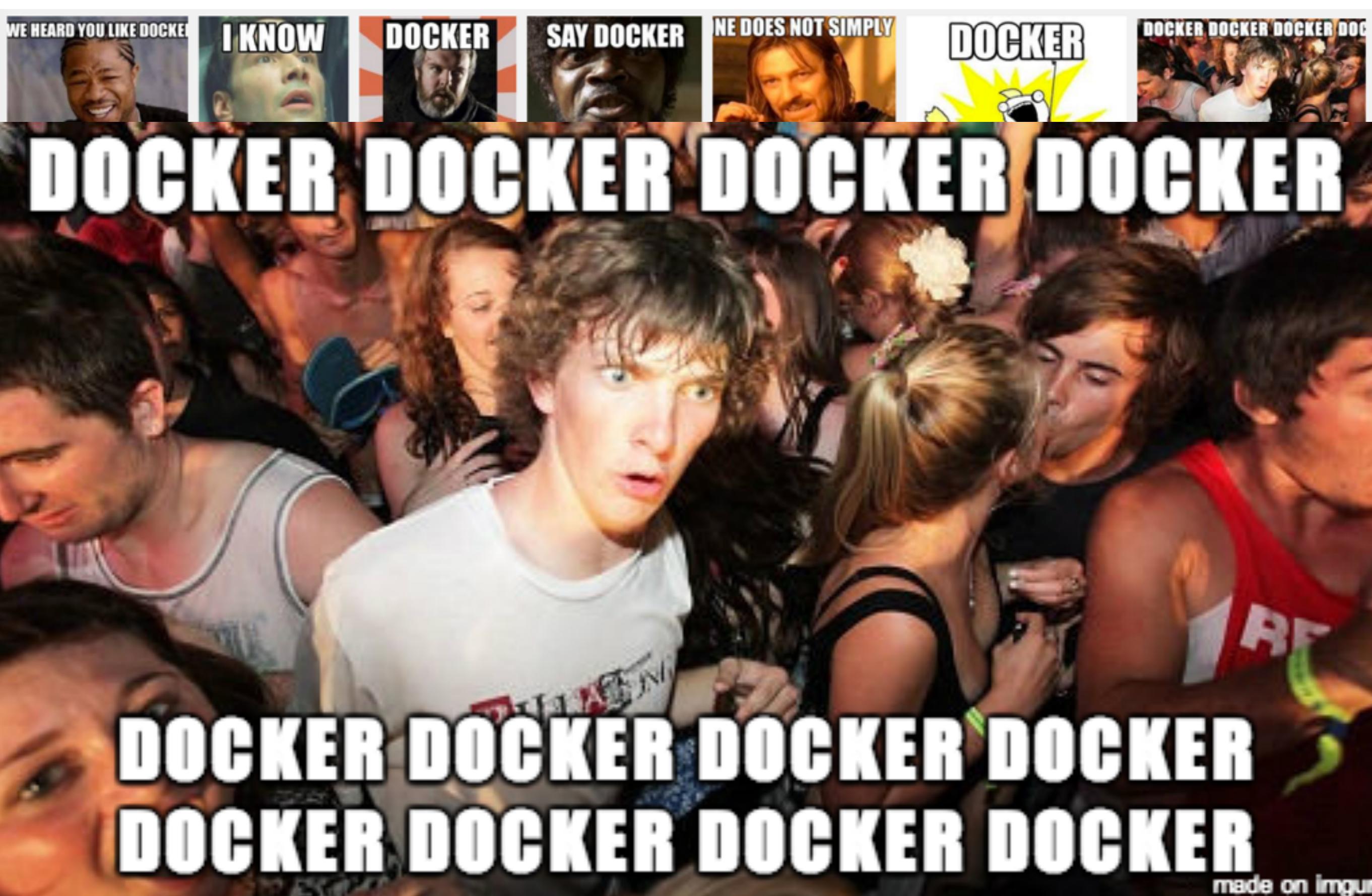
## ALL THE THINGS

Quelle:

<http://cdn.meme.am/instances/500x/59600465.jpg>

„Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications, whether on laptops, data center VMs, or the cloud.“

<https://www.docker.com/whatisdocker>



Quelle: Google „Docker Memes“ ;)

<http://jamespacile.o.github.io/Slides-Dockerize-That-Django-App/img/docker-meme.png>

# Docker 101



# docker

Quelle:

<http://blog.docker.com/wp-content/uploads/2013/06/Docker-logo-011.png>

# Terminologie

## Core Components

### Docker Host

- (Linux) System with Docker Daemon

### Docker Daemon

- The engine, running on the host

### Docker Client

- CLI for interacting with Daemon

## Workflow Components

### Docker Image

- contains application + environment

### Docker Container

- created from image - start, stop, ...

### Docker Registry

- „App Store“ for images
- Public + private repository possible

### Dockerfile

- used for automating image build

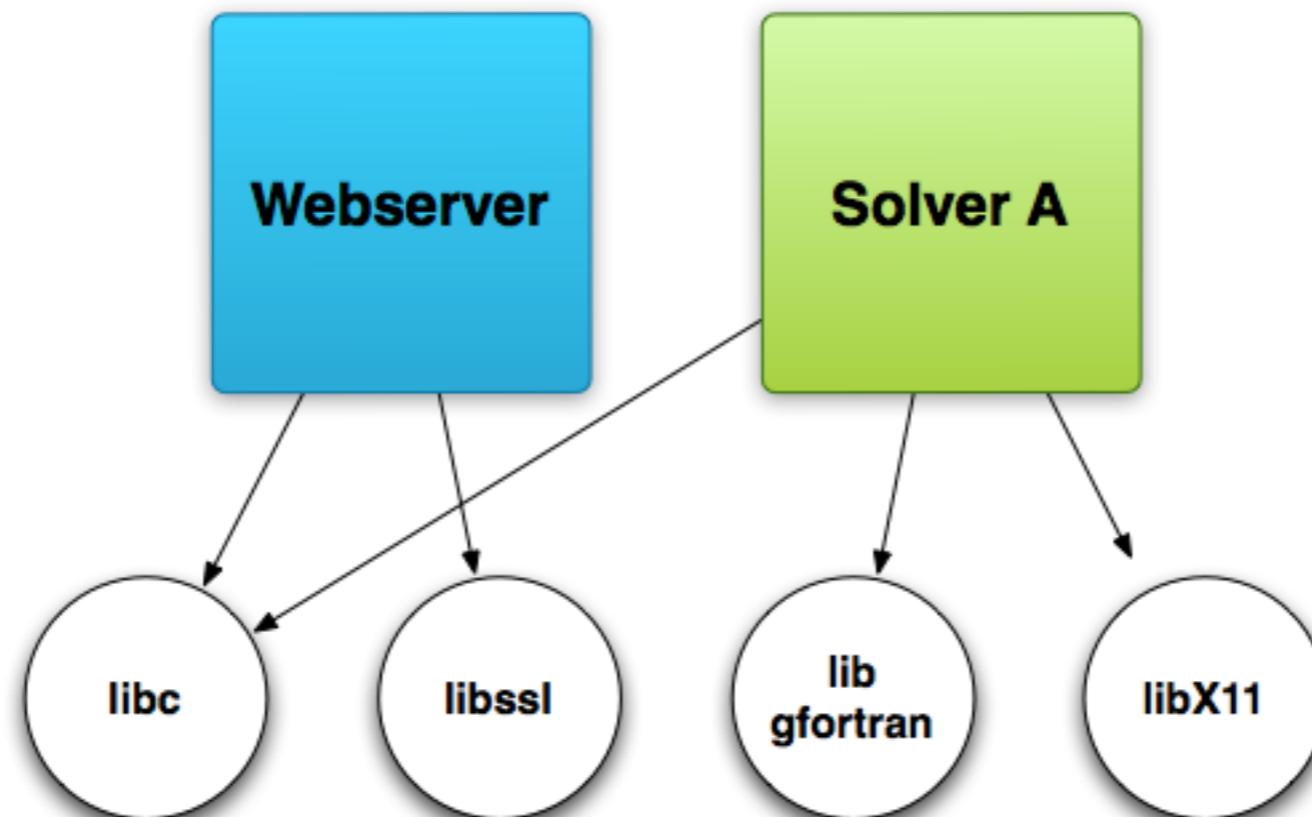
# Was kann Docker für Dich tun?

*Leben retten ;)*

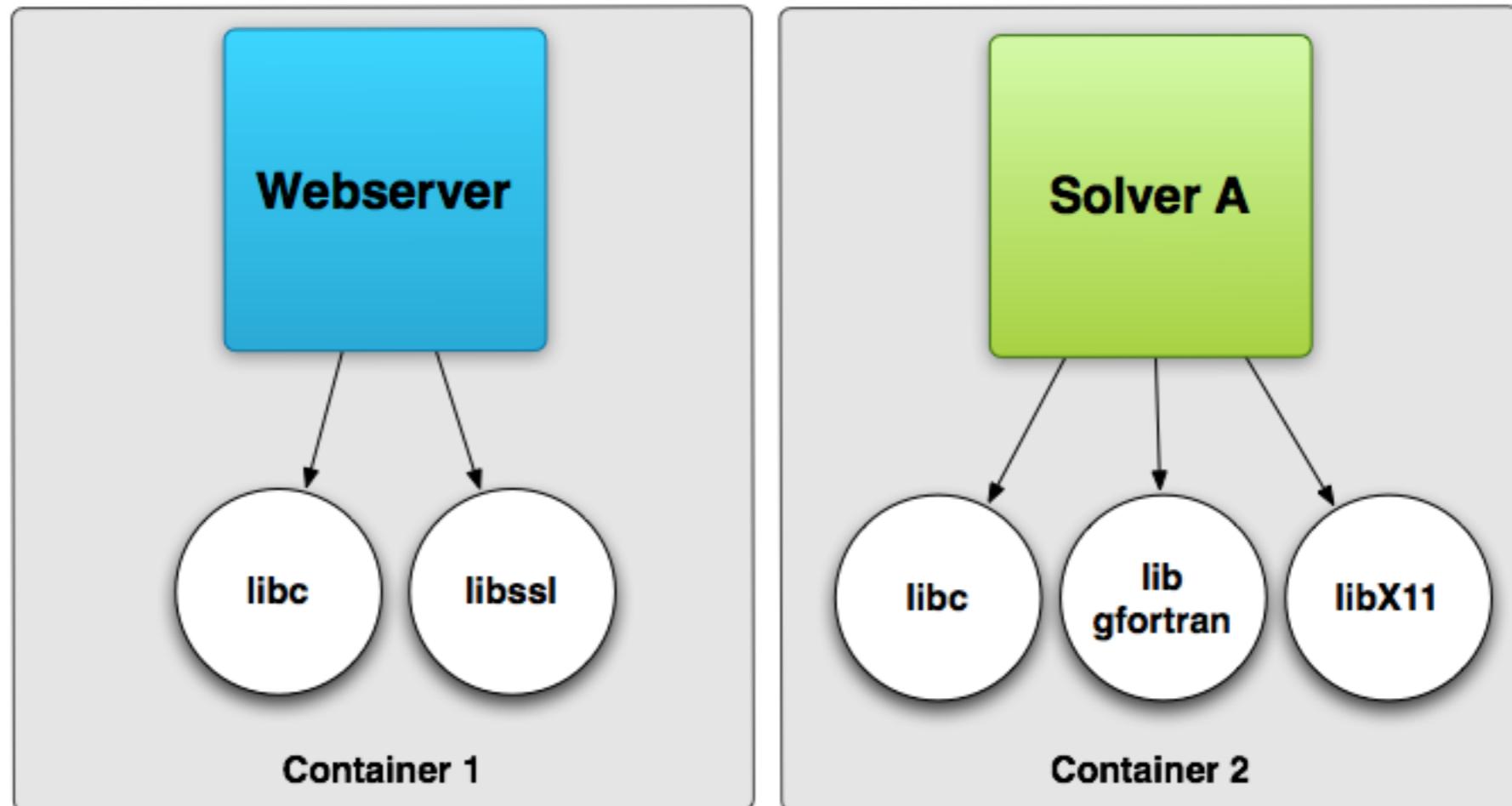
*Devs vs Ops*

# Abhängigkeiten *isolieren*

# Aufbau ohne Docker

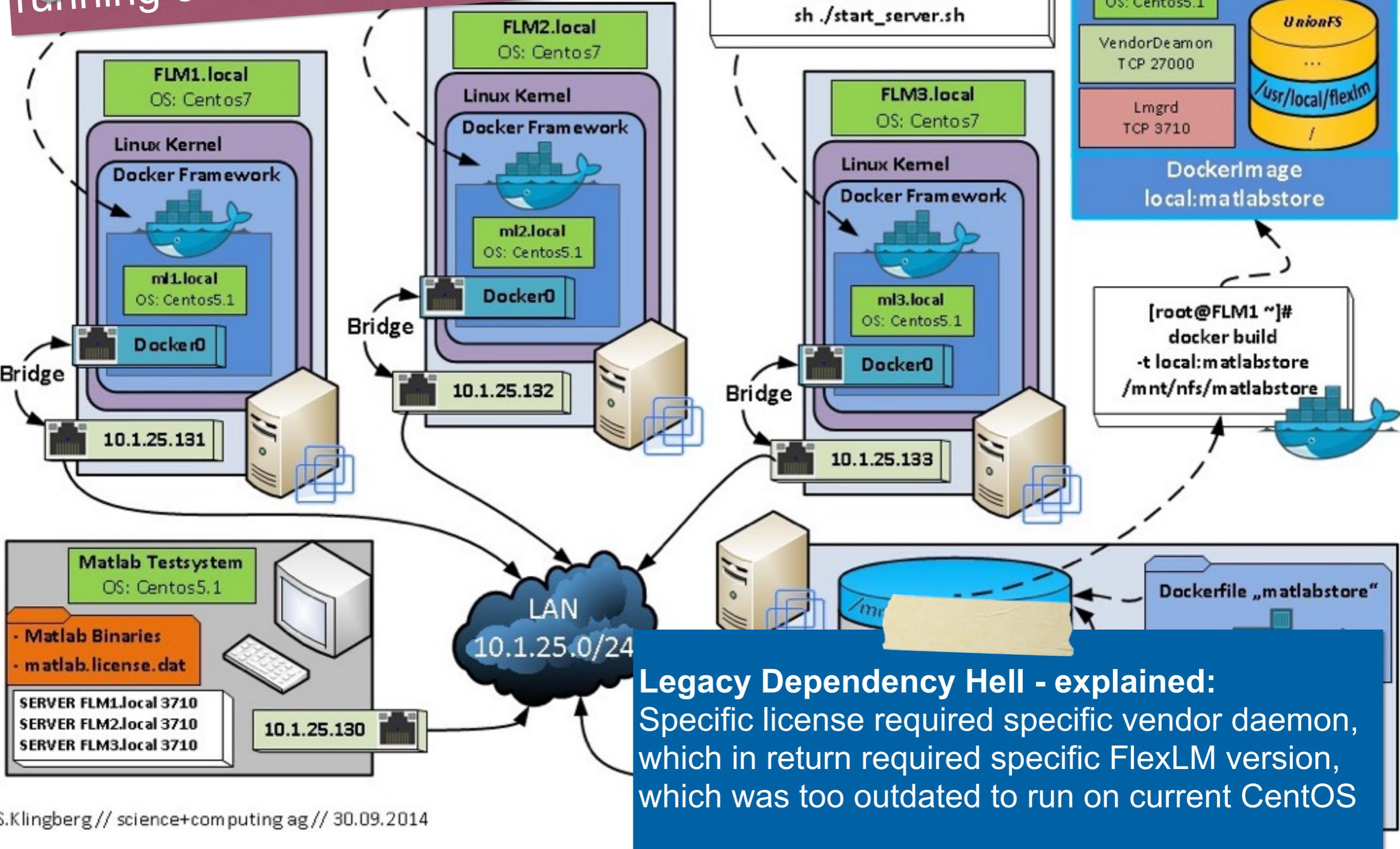


# Aufbau *mit Docker*



# Nicht ganz typisches Beispiel...

# Containerized historic FlexLM „triple“ running on one Docker Host



# Workflow<sup>^2</sup> ;)

shh433 (Seite 1 von 14) ▾

Peltzer et al. *Genome Biology* (2016) 17:60  
DOI 10.1186/s13059-016-0918-z

Genome Biology

SOFTWARE Open Access

 CrossMark

# EAGER: efficient ancient genome reconstruction

Alexander Peltzer<sup>1,2,5\*</sup>, Günter Jäger<sup>1</sup>, Alexander Herbig<sup>1,2,5</sup>, Alexander Seitz<sup>1</sup>, Christian Kniep<sup>4</sup>, Johannes Krause<sup>2,3,5</sup> and Kay Nieselt<sup>1</sup> 

**Abstract**

**Background:** The automated reconstruction of genome sequences in ancient genome analysis is a multifaceted process.

**Results:** Here we introduce EAGER, a time-efficient pipeline, which greatly simplifies the analysis of large-scale genomic data sets. EAGER provides features to preprocess, map, authenticate, and assess the quality of ancient DNA samples. Additionally, EAGER comprises tools to genotype samples to discover, filter, and analyze variants.

**Conclusions:** EAGER encompasses both state-of-the-art tools for each step as well as new complementary tools tailored for ancient DNA data within a single integrated solution in an easily accessible format.

**Keywords:** aDNA, Bioinformatics, Authentication, aDNA analysis, Genome reconstruction

**Background**

In ancient DNA (aDNA) studies, often billions of sequence reads are analyzed to determine the genomic sequence [1–3]. Newly developed

Until today, there have only been a few contributions towards a general framework for this task, such as the collection of tools and respective parameters proposed by Martin Kircher [8]. However, most of these methods have been developed for mitochondrial data in the context of the Neandertal project [1–9], and therefore do

Quelle:  
[https://genomebiology.biomedcentral.com/  
articles/10.1186/s13059-016-0918-z](https://genomebiology.biomedcentral.com/articles/10.1186/s13059-016-0918-z)

Workflows

# Reproduzierbarkeit

<sup>21</sup>

# Science, bitches!

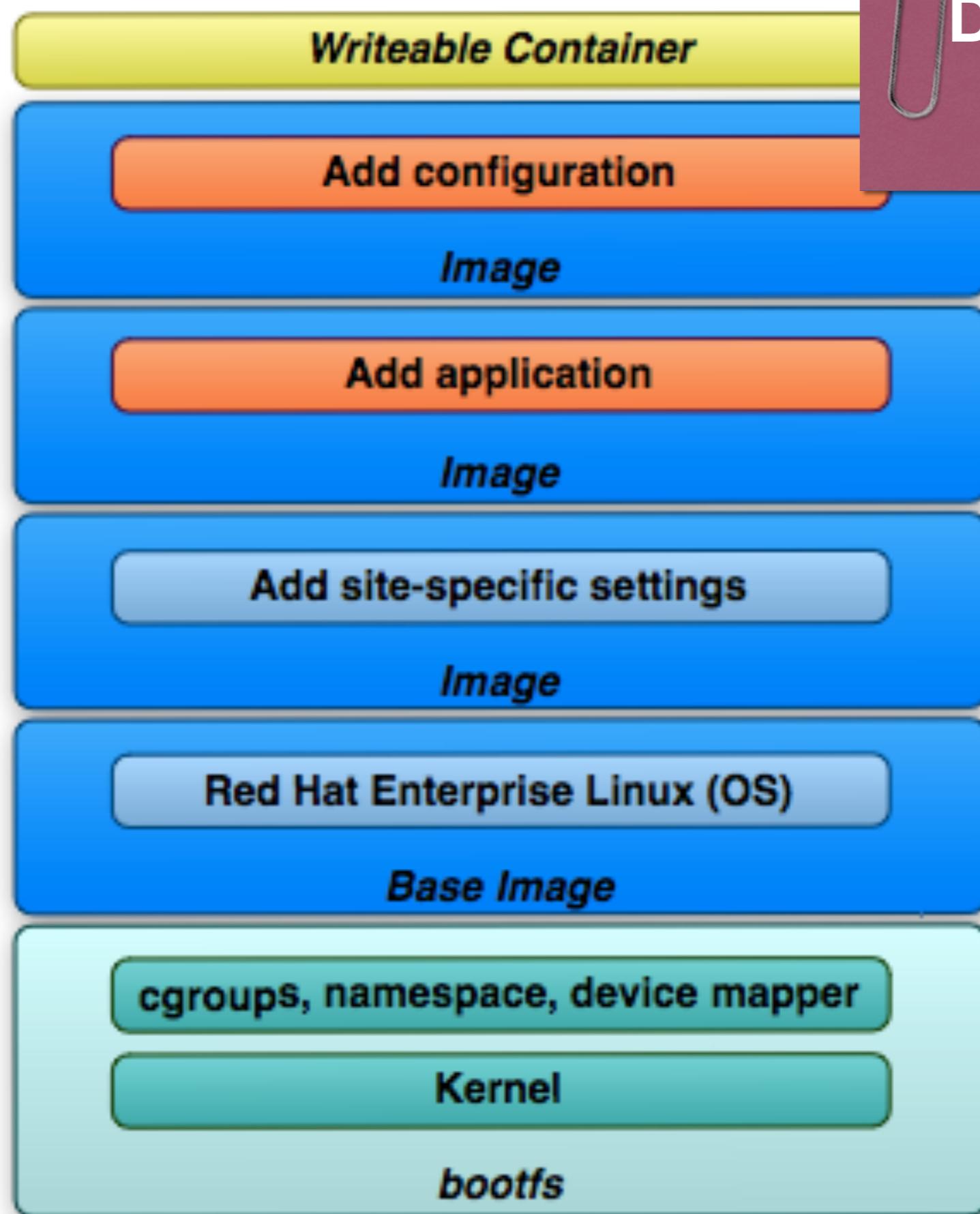


Quelle:

<http://www.critic.co.nz/files/article-3423.jpg>

# Und sonst so...?

# Dateisystem Schichten



Docker Hub

SC.

DOCKER DOCKER DOCKER DOCKER

DOCKER DOCKER DOCKER DOCKER  
DOCKER DOCKER DOCKER DOCKER

“ ” “ ” “ ”

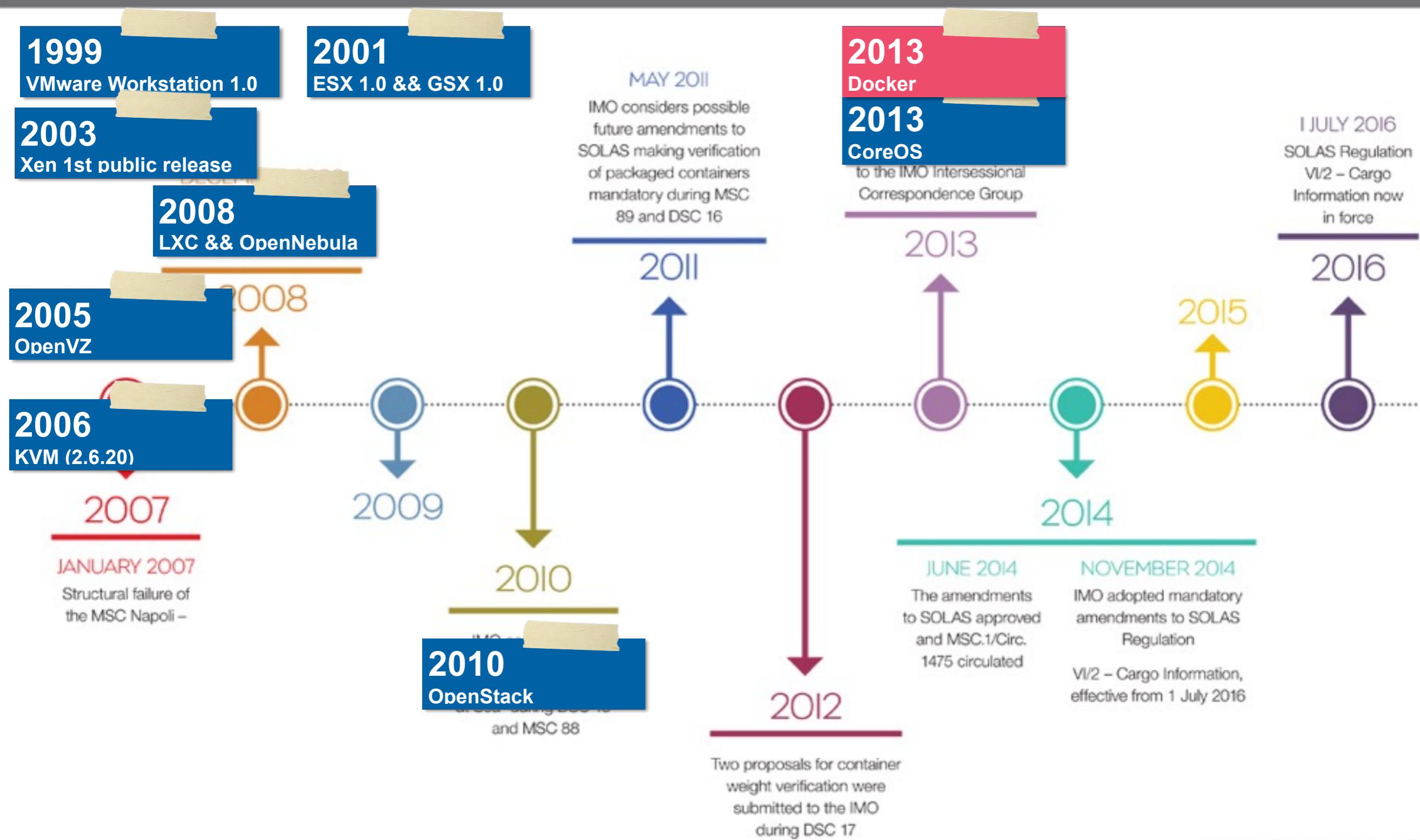
Kubernetes

made on imgur

# Virtualization 2.0?

*Harder, Better, Faster, Stronger?*

# TIMELINE CHART



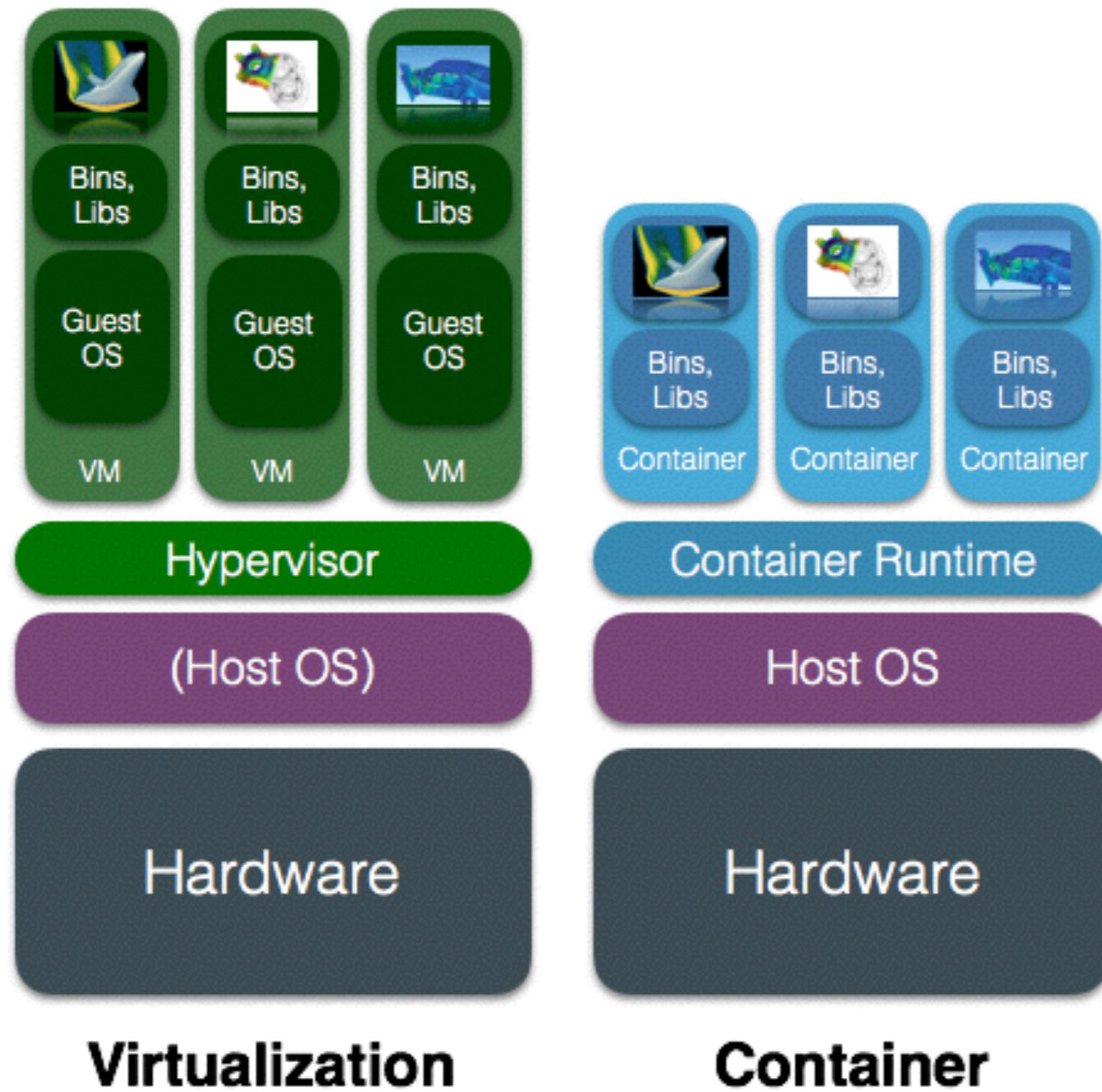
Quelle

<http://www.nepia.com/media/292058/NORTH-Timeline-Chart3.jpg>

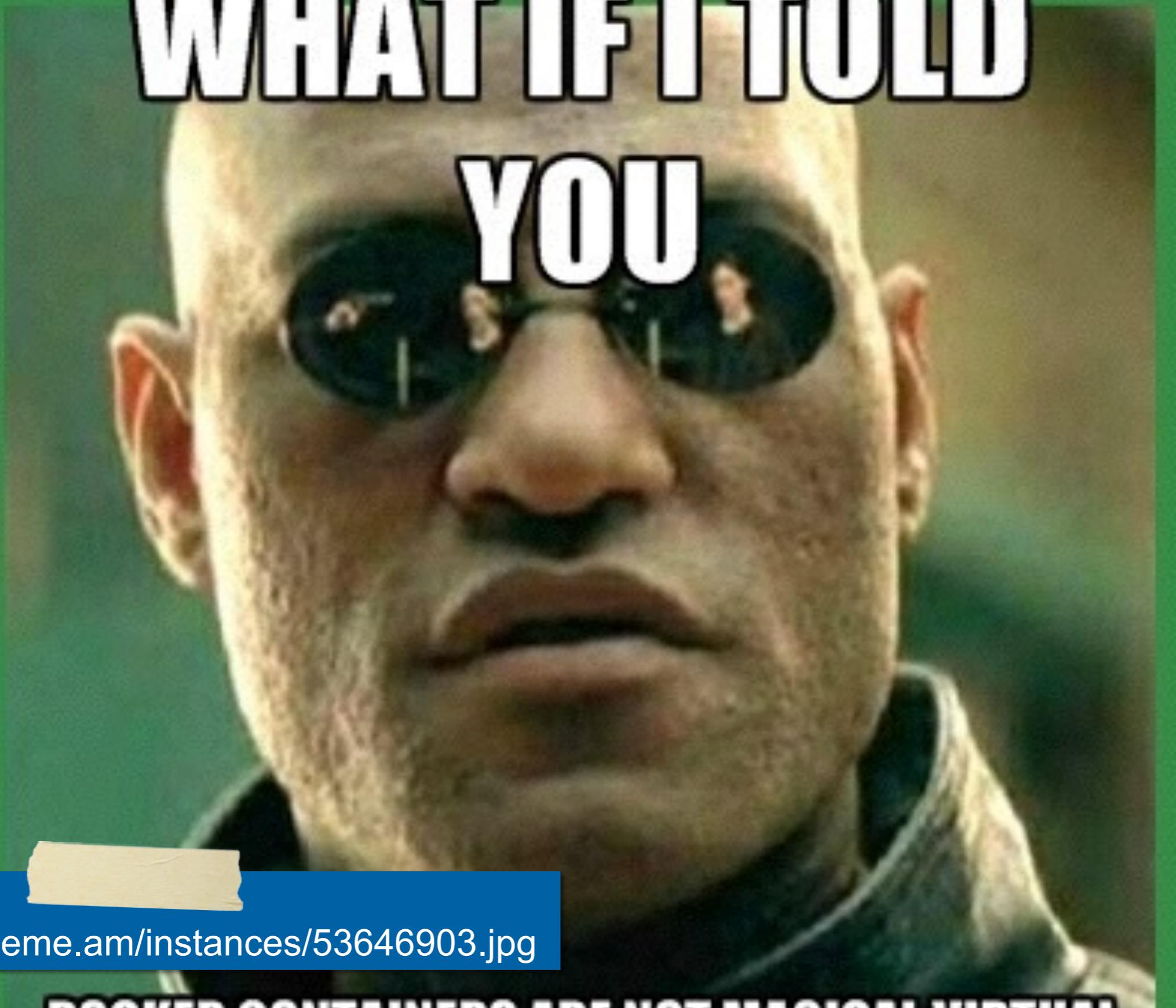
NORTH



# Virtualization *im Vergleich*



# **WHAT IF I TOLD YOU**



**Source:**

<http://cdn.meme.am/instances/53646903.jpg>

**DOCKER CONTAINERS ARE NOT MAGICAL VIRTUAL  
MACHINES**

[memegenerator.net](http://memegenerator.net)

# Was ist anders?

# Teil I: Docker im HPC

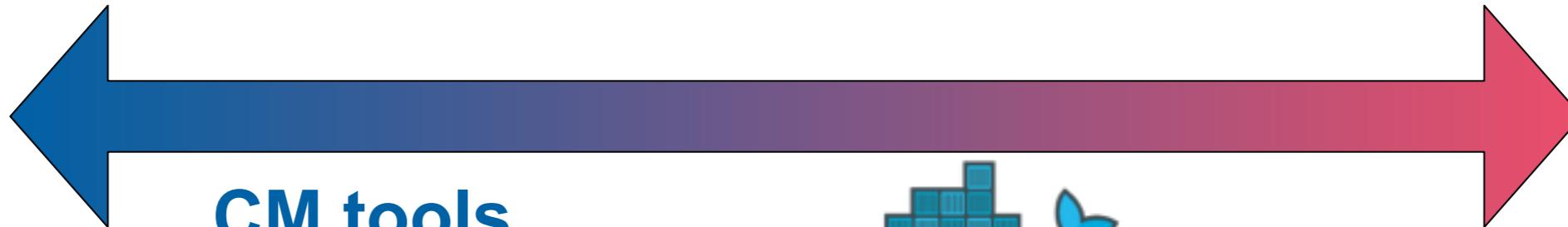
Wo drückt's?

# Saubere Trennung

# Deployment!

# *Deployment im Vergleich*

**less portable,  
minimal overhead**



## Docker

- \* is almost as portable (Windows anyone?) and flexible as VMs
- \* but induces much less overhead than VMs

# Heterogene Cluster homogenisieren

# Umgebung weitergeben?

# weitere Ressourcen?

# Warum nicht *klassische Virtualisierung?*

# Teil II: Benchmarks

# Benchmarks...

## Container-Based Virtualization for HPC

Holger Gantikow<sup>1</sup>, Sebastian Klingberg<sup>1</sup>, Christoph Reich<sup>2</sup>

The screenshot shows a PDF viewer window with two open documents. The top document is titled 'Container-Based Virtualization for HPC' and lists authors Holger Gantikow, Sebastian Klingberg, and Christoph Reich. The bottom document is titled 'An Updated Performance Comparison of Virtual Machines and Linux Containers' by Wes Felter, Alexandre Ferreira, Ram Rajamony, and Juan Rubio. Both documents have similar metadata sections including 'Keywords', 'Abstract', and publication details like 'RC25482 (AUS1407-001) July 21, 2014 Computer Science'.

**Keywords:** Cont

**Abstract:** Expe  
insta  
comp  
This  
Com  
(Doc  
analy

**RC25482 (AUS1407-001) July 21, 2014**  
**Computer Science**

**IBM Research Report**

**An Updated Performance Comparison of Virtual Machines and Linux Containers**

**Wes Felter, Alexandre Ferreira, Ram Rajamony, Juan Rubio**  
IBM Research Division  
Austin Research Laboratory  
11501 Burnet Road  
Austin, TX 78758

# IBM Docker Paper

RC25482 (AUS1407-001) July 21, 2014  
Computer Science

## IBM Research Report

### An Updated Performance Comparison of Virtual Machines and Linux Containers

**Wes Felter, Alexandre Ferreira, Ram Rajamony, Juan Rubio**

IBM Research Division

Austin Research Laboratory

11501 Burnet Road

Austin, TX 78758

USA

Quelle: Google: *ibm docker paper* oder:

[http://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/\\$File/rc25482.pdf](http://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/$File/rc25482.pdf)

# IBM Docker Paper - Aufbau

## 3. Evaluation

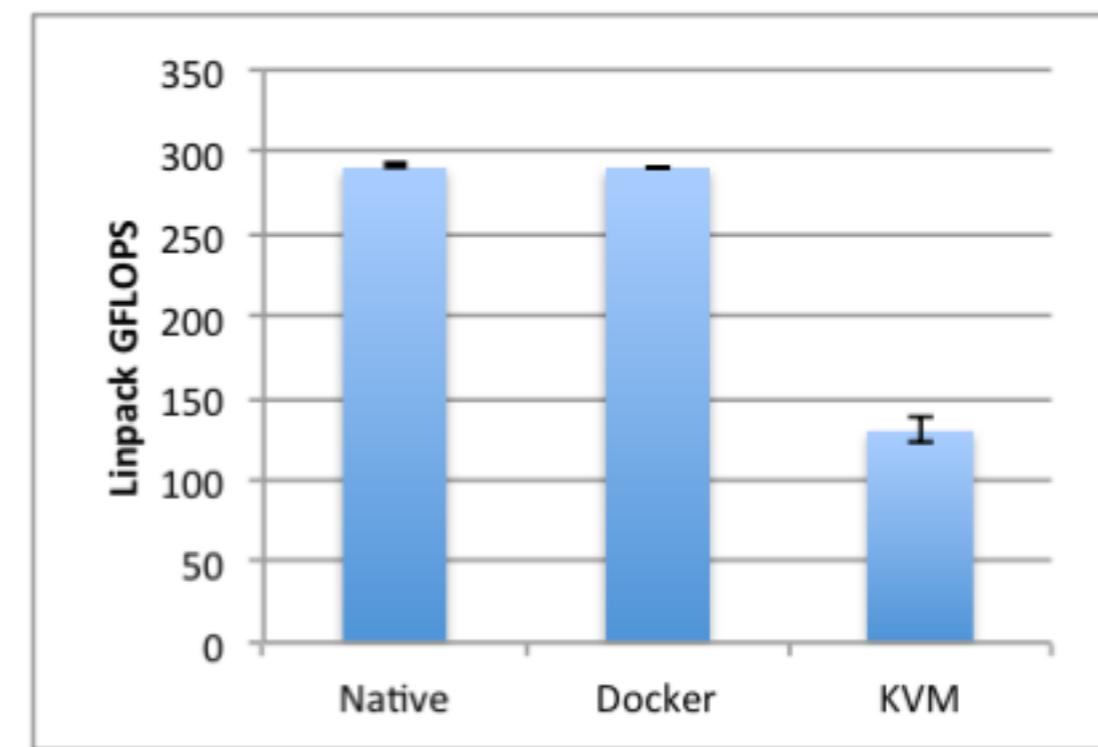
In this paper we seek to isolate and understand the overhead introduced by virtual machines (specifically KVM) and containers (specifically Docker) relative to non-virtualized Linux. The fact that Linux can host both VMs and containers creates the opportunity for an apples-to-apples comparison between the two technologies with fewer confounding variables than many previous comparisons. We attempt such a comparison in this paper.

We do not evaluate the case of containers running inside VMs or VMs running inside containers because we consider such double virtualization to be redundant (at least from a performance perspective). To measure overhead we have configured our benchmarks to saturate the resources of the system under test. Docker containers were not restricted by cgroups so they could consume the full resources of the system under test. Likewise, VMs were configured with 32 vCPUs and adequate RAM to hold the benchmark's working set. We use microbenchmarks to individually measure CPU, memory, network, and storage overhead. We also measure two real server applications: Redis and MySQL.

All of these tests were performed on an IBM® System x3650 M4 server with two 2.4-3.0 GHz Intel Sandy Bridge-EP Xeon E5-2665 processors for a total of 16 cores (plus HyperThreading) and 256 GB of RAM. The two processors/sockets are connected by QPI links making this a non-uniform memory access (NUMA) system. This is a mainstream server configuration that is very similar to those used by popular cloud providers [13].

3.10 (Saucy)

formation to VMs, so the guest OS believes it is running on a 32-socket system with one core per socket. This is a double-edged sword; abstracting the hardware can improve portability but it also eliminates some opportunities for optimization. [13]



**Figure 1.** Linpack performance on two sockets (16 cores). Each data point is the arithmetic mean obtained from ten runs. Error bars indicate the standard deviation obtained over all runs.

Figure 1 shows the performance of Linpack on Linux, Docker, and KVM. A Linpack execution spends the bulk of its time performing mathematical floating point operations. By basing the code on an optimized linear algebra library, the execution gives rise to fairly regular memory accesses

the floating point capability of the core. core, the math library is highly adaptive provided information to tune itself to the

Quelle: Google: *ibm docker paper* oder:

[http://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/\\$File/rc25482.pdf](http://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/$File/rc25482.pdf)

# IBM Docker Paper - Ergebnis

Both VMs and containers are mature technology that have benefited from a decade of incremental hardware and software optimizations. In general, Docker equals or exceeds KVM performance in every case we tested. Our results show that both KVM and Docker introduce negligible overhead for CPU and memory performance (except in extreme cases). For I/O-intensive workloads, both forms of virtualization should be used carefully.

We find that KVM performance has improved considerably since its creation. Workloads that used to be considered very challenging, like line-rate 10 Gbps networking, are now possible using only a single core using 2013-era hardware and software. Even using the fastest available forms of paravirtualization, KVM still adds some overhead to every I/O operation; this overhead ranges from significant when performing small I/Os to negligible when it is amortized over large I/Os. Thus, KVM is less suitable for workloads that are latency-sensitive or have high I/O rates. These overheads significantly impact the server applications we tested.

Although containers themselves have almost no overhead, Docker is not without performance gotchas. Docker volumes have noticeably better performance than files stored in AUFS. Docker's NAT also introduces overhead for workloads with high packet rates. These features represent a tradeoff between ease of management and performance and should be considered on a case-by-case basis.

In some sense the comparison can only get worse for containers because they started with near-zero overhead and VMs have gotten faster over time. If containers are to be widely adopted they must address issues other than

that attempting to exploit NUMA in the cloud may be more effort than it is worth. Limiting each workload to a single socket greatly simplifies performance analysis and tuning. Given that cloud applications are generally designed to scale out and the number of cores per socket increases over time, the unit of scaling should probably be the socket rather than the server. This is also a case against bare metal, since a server running one container per socket may actually be faster than spreading the workload across sockets due to the reduced cross-traffic.

In this paper we created single VMs or containers that consumed a whole server; in the cloud it is more common to divide servers into smaller units. This leads to several additional topics worthy of investigation: performance isolation when multiple workloads run on the same server, live resizing of containers and VMs, tradeoffs between scale-up and scale-out, and tradeoffs between live migration and restarting.

## Source code

The scripts to run the experiments from this paper are available at <https://github.com/thewmf/kvm-docker-comparison>.

## nts

ported in part by the Office of Science,  
Department of Energy under award number

Quelle: Google: ibm docker paper oder:

[http://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/\\$File/rc25482.pdf](http://domino.research.ibm.com/library/cyberdig.nsf/papers/0929052195DD819C85257D2300681E7B/$File/rc25482.pdf)

**"In general, Docker equals or exceeds KVM performance in every case we tested. [...]**

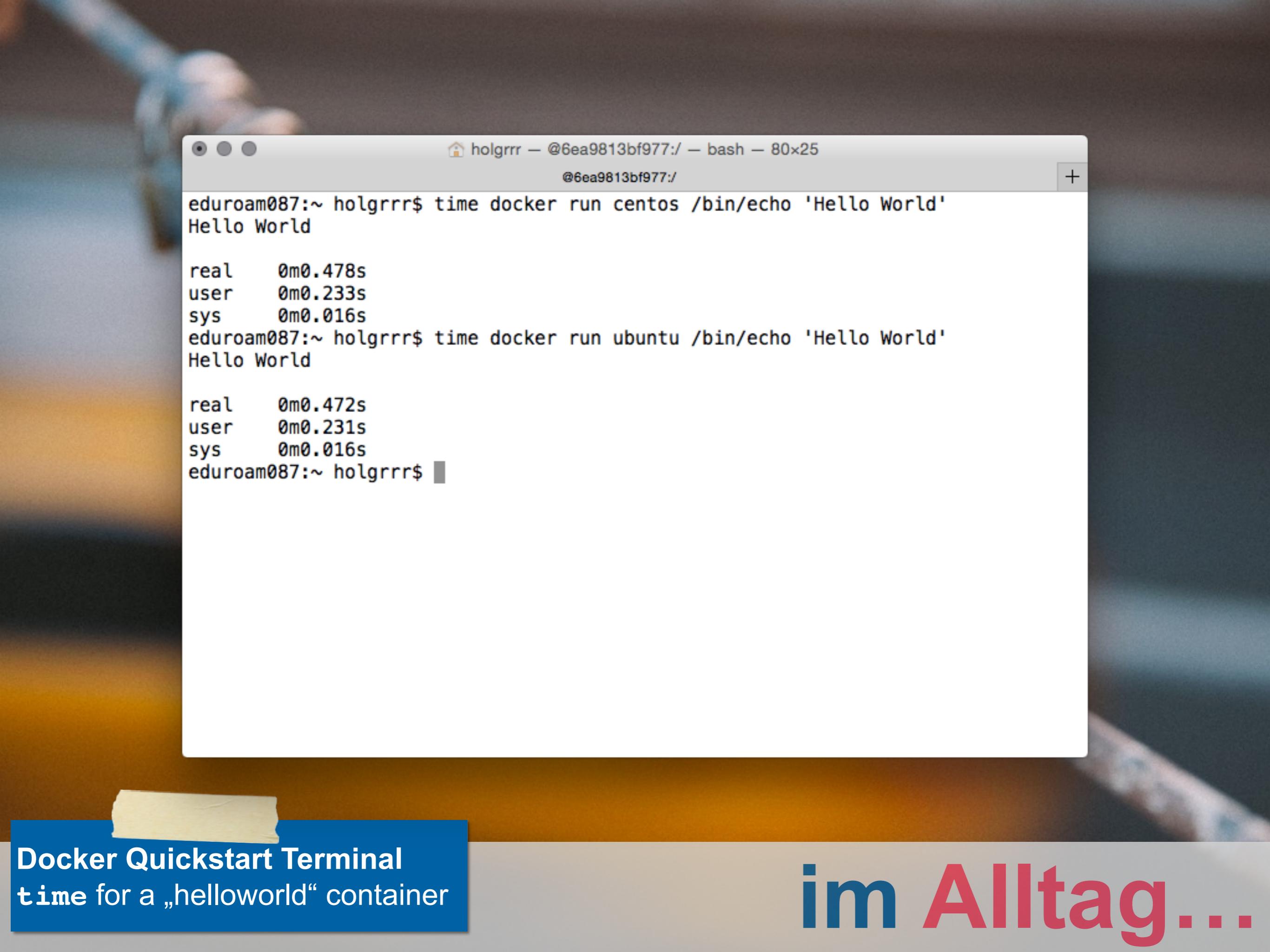
**Even using the fastest available forms of paravirtualization, KVM still adds some overhead to every I/O operation [...].**

**Thus, KVM is less suitable for workloads that are latency-sensitive or have high I/O rates.**

## Container vs. bare-metal:

Although containers themselves have almost no overhead, Docker is not without performance gotchas. Docker volumes have noticeably better performance than files stored in AUFS. Docker's NAT also introduces overhead for work- loads with high packet rates.

These features represent a tradeoff between ease of management and performance and should be considered on a case-by-case basis.



```
holgrrr - @6ea9813bf977:/ - bash - 80x25
@6ea9813bf977:/
eduroam087:~ holgrrr$ time docker run centos /bin/echo 'Hello World'
Hello World

real    0m0.478s
user    0m0.233s
sys     0m0.016s
eduroam087:~ holgrrr$ time docker run ubuntu /bin/echo 'Hello World'
Hello World

real    0m0.472s
user    0m0.231s
sys     0m0.016s
eduroam087:~ holgrrr$
```

Docker Quickstart Terminal  
time for a „helloworld“ container

im Alltag...

# Docker@HPC

## Container-Based Virtualization for HPC

Holger Gantikow<sup>1</sup>, Sebastian Klingberg<sup>1</sup>, Christoph Reich<sup>2</sup>

<sup>1</sup>*science & computing AG, Tübingen, Germany*

<sup>2</sup>*Cloud Research Lab, Furtwangen University, Furtwangen, Germany*

*gantikow@gmail.com, klingber@hs-furtwangen.de, christoph.reich@hs-furtwangen.de*

**Keywords:** Container Virtualization, Docker, High Performance Computing, HPC

**Abstract:** Experts argue that the resource demands of High Performance Computing (HPC) clusters request bare-metal installations. The performance loss of container virtualization is minimal and close to bare-metal, but in comparison has many advantages, like ease of provisioning. This paper presents the use of the newly adopted container technology and its advantages for High Performance Computing, compared to traditional bare-metal installations or the use of VMs. The setup based on Docker (Docker, 2015) shows the possible use in private HPC sites or public clouds. Ending with a detailed risk analysis of Docker HPC installations.

## 1 INTRODUCTION

Applications in the domain of High Performance Computing (HPC) have massive requirements when it comes to resources like CPU, memory, I/O throughput and interconnects. This is the reason why they are traditionally run in a bare-metal setup, directly on physical systems, which are interconnected to so-called clusters.

Such a cluster infrastructure offers the best performance, but of disadvantage is the time for setting up: a) The operating system, usually some Linux flavor, must be installed using automatic mechanisms like PXE and Kickstart to install a basic installation ready to log in and get customized. b) All the applications required for computation and general HPC related libraries, like MPI ((MPI), 2015), have to be installed and fine tuned in the customization phase.

or even different versions of the same one, have conflicting environmental requirements, like a specific Linux version or specific library version (e.g. *libc*). This leads to the risk of putting the consistency of the whole cluster at stake, when adding a new application, or a newer version. Libraries might have to be updated, which might imply a recompile of the whole Linux operating system (OS). This can lead to long downtimes for the cluster, especially if being done in the middle of a job.

Danke!  
@Sebastian Klingberg

# Docker@HPC - Aufbau

memory footprint, as containers share a lot of resources with the host system as opposed to VMs starting a complete OS and in terms of storage required. Startup time is reduced from the time booting a full OS to the few seconds it takes till the container is ready to use.

For reducing storage requirements Docker makes use of *layered file system images*, usually *UnionFS* (Unionfs, 2015) as a space conserving mechanism, which is lacking in several other container solutions. This allows file systems stacked as layers on top of each other (see Figure 4), which enables sharing and reusing of base layers. For example the base installation of a certain distribution, with individually modified overlays stacked on top, can provide the required application and configuration for several different tasks.

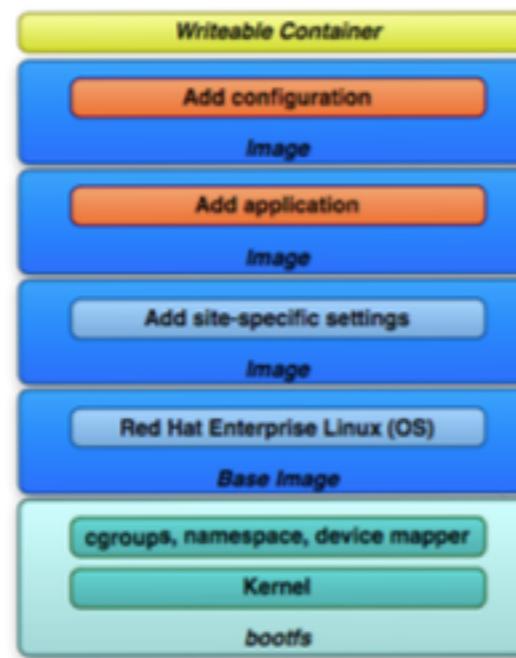


Figure 4: The layers of the Docker file system

## 4 EXPERIMENTAL EVALUATION

Performance-wise, without all the overhead added by hypervisor and VMs, containers as a light-weight virtualization mechanism can achieve almost the same performance as native execution on a bare-metal system does, as other benchmarks (Felter et al., 2014), (Xavier et al., 2013) underline.

As we were interested in the amount of overhead generated by containerizing a HPC workload, we decided to benchmark a real-world ABAQUS example in different scenarios, comparing the containerized execution time to the native execution. ABAQUS (Abaqus FEA, 2015) is frequently used for finite element analysis (FEA) and computer aided engineering (CAE) for example in the aerospace and automotive industries, as it provides wide material modeling capability and multi-physics capabilities.

The application was installed to local disks on a CentOS 7 SunFire X2200 server with the following hardware configuration:

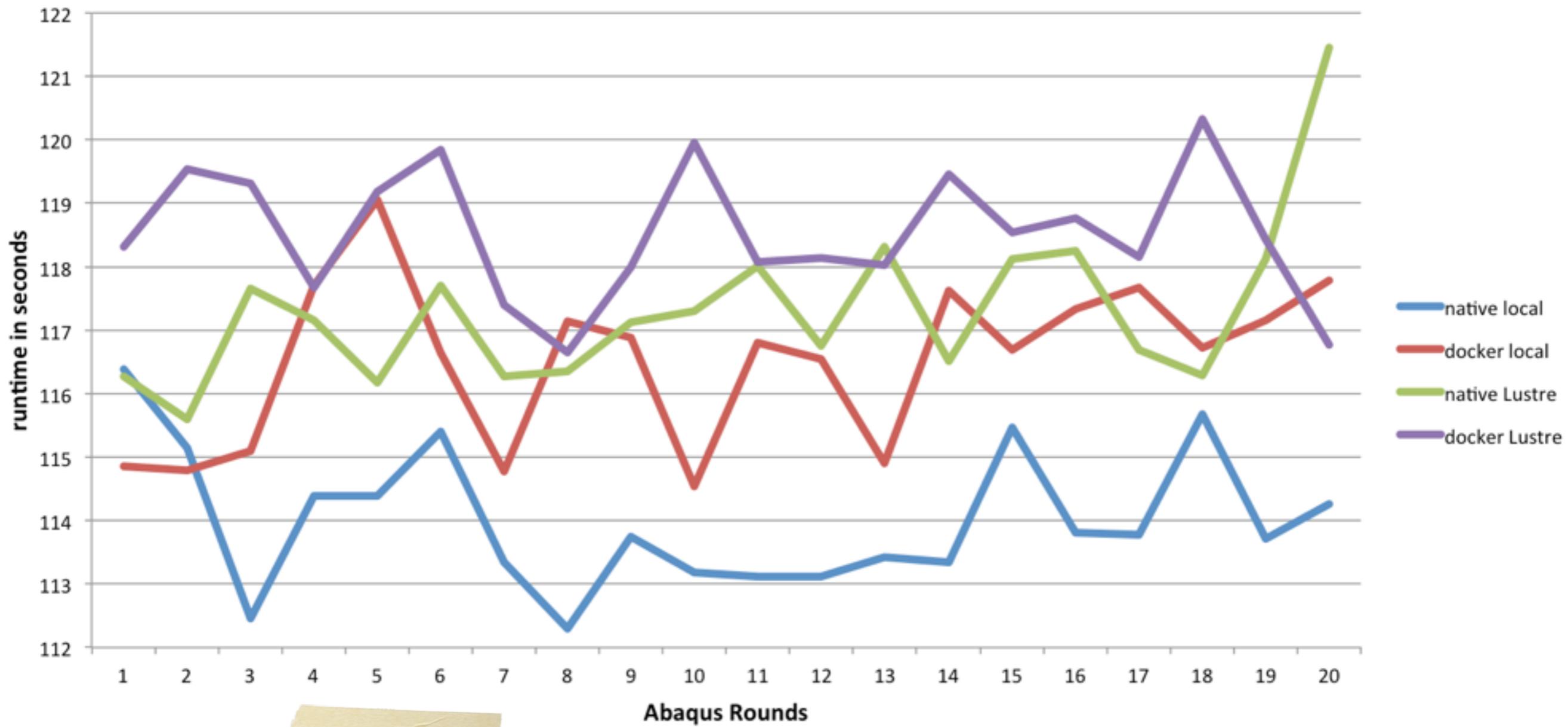
- CPU: 4x Dual-Core AMD Opteron (tm) 2220
- RAM: 16GB
- HDD: local conventional disks without RAID
- Infiniband: Mellanox MT25204

The job used for evaluation is the freely available *s4b* from the *samples.zip* package included in the ABAQUS 6.12-3 installation. It was installed onto local disks and the software licensing was done using a local license file.

As the server provided access to a Lustre parallel

# Docker@HPC - Ergebnis

	A	B	C	D	E	F
1	native local	docker local	native Lustre	docker Lustre		
2	116,50	116,50	117,30	117,30		



Zahlen - average runtime (native vs Docker)  
local disk: 114s vs 116,5s - overhead: 2,21%  
Lustre: 117,3s vs 118,5s - overhead: 1,04%

1,041715506	Overhead
	Groß-/Kleinschreibung
Summe=0	
-	+
80%	

# Teil III: Security-Aspekte

# Zitate



**Quelle:** Surviving the Zombie Apocalypse - Ian Jackson  
<http://xenbits.xen.org/people/iwj/2015/fosdem-security/>

**"Some people make the mistake of thinking of containers as a better and faster way of running virtual machines.**

**From a security point of view, containers are much weaker."**

Dan Walsh,  
SELinux architect(?)

**"Virtual Machines might be more secure today,  
but containers are definitely catching up."**

Jerome Petazzoni,  
Senior Software Engineer at Docker

**"You are absolutely deluded, if not stupid, if you think that a worldwide collection of software engineers who can't write operating systems or applications without security holes, can then turn around and suddenly write virtualization layers without security holes."**

Theo de Raadt,  
OpenBSD project lead

**"Docker's security status is best described as  
'it's complicated'."**

Jerome Petazzoni,  
Senior Software Engineer at Docker

# vulnerabilities

# Surviving the Zombie Apocalypse

## Surviving the Zombie Apocalypse

Security in the Cloud – Containers, KVM and Xen

Ian Jackson <ian.jackson@eu.citrix.com>

FOSDEM 2015

originally based on a talk and research by George Dunlap



# Zombies? - Findings!

Some Free Software VM hosting technologies

Vulnerabilities published in 2014

	Xen PV	KVM+ QEMU	Linux as general container	Linux app container (non-root)
Privilege escalation (guest-to-host)	0	3–5	7–9	4
Denial of service (by guest of host)	3	5–7	12	3
Information leak (from host to guest)	1	0	1	1
				Hosts only application, not guest OS



Frustbringer...

# Docker - Missverständnisse

Jessie Frazelle's Blog

## Docker Containers on the Desktop

February 21, 2015

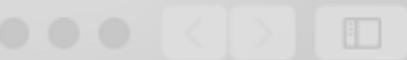
Hello!

If you are not familiar with [Docker](#), it is the popular open source container engine.

Most people use Docker for containing applications to deploy into production or for building their applications in a contained environment. This is all fine & dandy, and saves developers & ops engineers huge headaches, but I like to use Docker in a not-so-typical way.

I use Docker to run all the desktop apps on my computers.

But why would I even want to run all these apps in containers? Well let me



# Docker - Leichtsinn

Jessie Frazelle's Blog

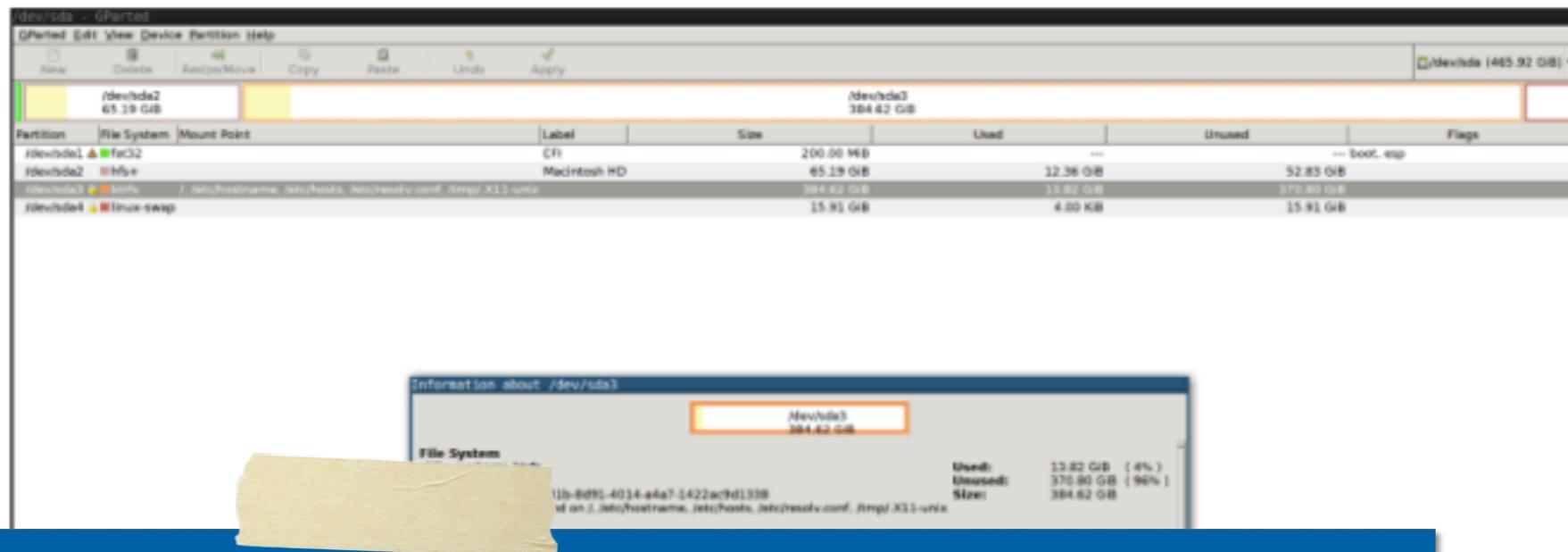
## 7. Gparted

### Dockerfile

Partition your device in a container.

MIND BLOWN.

```
$ docker run -it \
-v /tmp/.X11-unix:/tmp/.X11-unix \ # mount the X11 socket
-e DISPLAY=unix$DISPLAY \ # pass the display
--device /dev/sda:/dev/sda \ # mount the device to partition
--name gparted \
jess/gparted
```



Quelle: Docker Containers on the Desktop

<https://blog.jessfraz.com/posts/docker-containers-on-the-desktop.html>

# Docker - Diskussion

X11 is completely unsecure, the "sandboxed" app has full access to every other X11 client.

if you have docker access you have root access [...]  
`docker run -v /:/tmp ubuntu rm -rf /tmp/*`  
Which will remove all the files on your system.

▲ alexlarsson 10 days ago

This is not sandboxing. Quite the opposite, this gives the apps root access:

First of all, X11 is completely unsecure, the "sandboxed" app has full access to every other X11 client. Thus, its very easy to write a simple X app that looks for say a terminal window and injects key events (say using Xtest extension) in it to type whatever it wants. Here is another example that sniffs the key events, including when you unlock the lock screen: <https://github.com/magcius/keylog>

Secondly, if you have docker access you have root access. You can easily run something like:

`docker run -v /:/tmp ubuntu rm -rf /tmp/*`

Which will remove all the files on your system.

[reply](#)

▲ jdub 9 days ago

Just so everyone knows, this is Alex "I have a weird interest in application bundling systems" Larsson, who is doing some badass bleeding edge work on full on sandboxed desktop applications on Linux. :-)

<http://blogs.gnome.org/alex/2015/02/17/first-fully-sandboxed-applications-on-linux/>

[http://www.youtube.com/watch?v=t-2a\\_XYJPEY](http://www.youtube.com/watch?v=t-2a_XYJPEY)

Like Ron Burgundy, he's... "kind of a big deal".

(Suffer the compliments, Alex.)

[reply](#)

Quelle: Docker containers on the desktop - Discussion  
<https://news.ycombinator.com/item?id=9086751>

# Update: inzwischen *User Namespaces*

"Without user namespaces (`CLONE_NEWUSER`), which Docker currently doesn't use, uid 0 inside a container is the same thing as uid 0 outside it.

If you let Docker run apps as root, which seems to be not uncommon, then it is, in a strong sense, the same as the root user outside the container.

That's why Jessie's gparted process can partition her disk: as long as it can get at the device node, it has full permissions on it.

# NFS *anyone?*

# Docker Image Insecurity

Jonathan

Rudenberg

is a cofounder of

Flynn

and an

architect of the

Tent Protocol.

[GitHub](#)

[Tent](#)

[Twitter](#)

[Email](#)

[Feed](#)

## Docker Image Insecurity

December 23, 2014

Recently while downloading an "official" container image with Docker I saw this line:

```
ubuntu:14.04: The image you are pulling has been verified
```

I assumed this referenced Docker's heavily promoted image signing system and didn't investigate further at the time. Later, while researching the cryptographic digest system that Docker tries to secure images with, I had the opportunity to explore further. What I found was a total systemic failure of all logic related to image security.

Docker's report that a downloaded image is "verified" is based solely on the presence of a signed manifest, and Docker never verifies the image checksum from the manifest. An attacker could provide any image alongside a signed manifest. This opens the door to a number of serious vulnerabilities.

Images are downloaded from an HTTPS server and go through an insecure streaming processing pipeline in the Docker daemon:

```
[decompress] -> [tarsum] -> [unpack]
```

This pipeline is performant but completely insecure. Untrusted input should not be processed before verifying its signature. Unfortunately Docker processes three times before checksum verification is supposed to

# Docker zähmen

# Feingranularer Zugriff

# Wrapper

# Application vs System Container

# *Was sonst noch geschah...*



Docker 1.10, ...

## Kurzfassung:

„Security Scanner“ - <https://github.com/coreosclair>

„Clair is an open source project for the static analysis of vulnerabilities in appc and docker containers.“

**Docker Bench for Security** - <https://github.com/docker/docker-bench-security>

The Docker Bench for Security is a script that checks for dozens of common best-practices around deploying Docker containers in production. [...] an easy way to self-assess their hosts and docker containers against this benchmark.

Und: **Seccomp Profiles, User Namespaces, Authorization Plugins, OpenSCAP/container-compliance, Docker Security Scanning, (ex Notary) ...**

# Further readings...

CIS\_Docker\_1.11.0\_Benchmark\_v1.0.0.pdf (Seite 1 von 161)

Suchen

Center for Internet Security



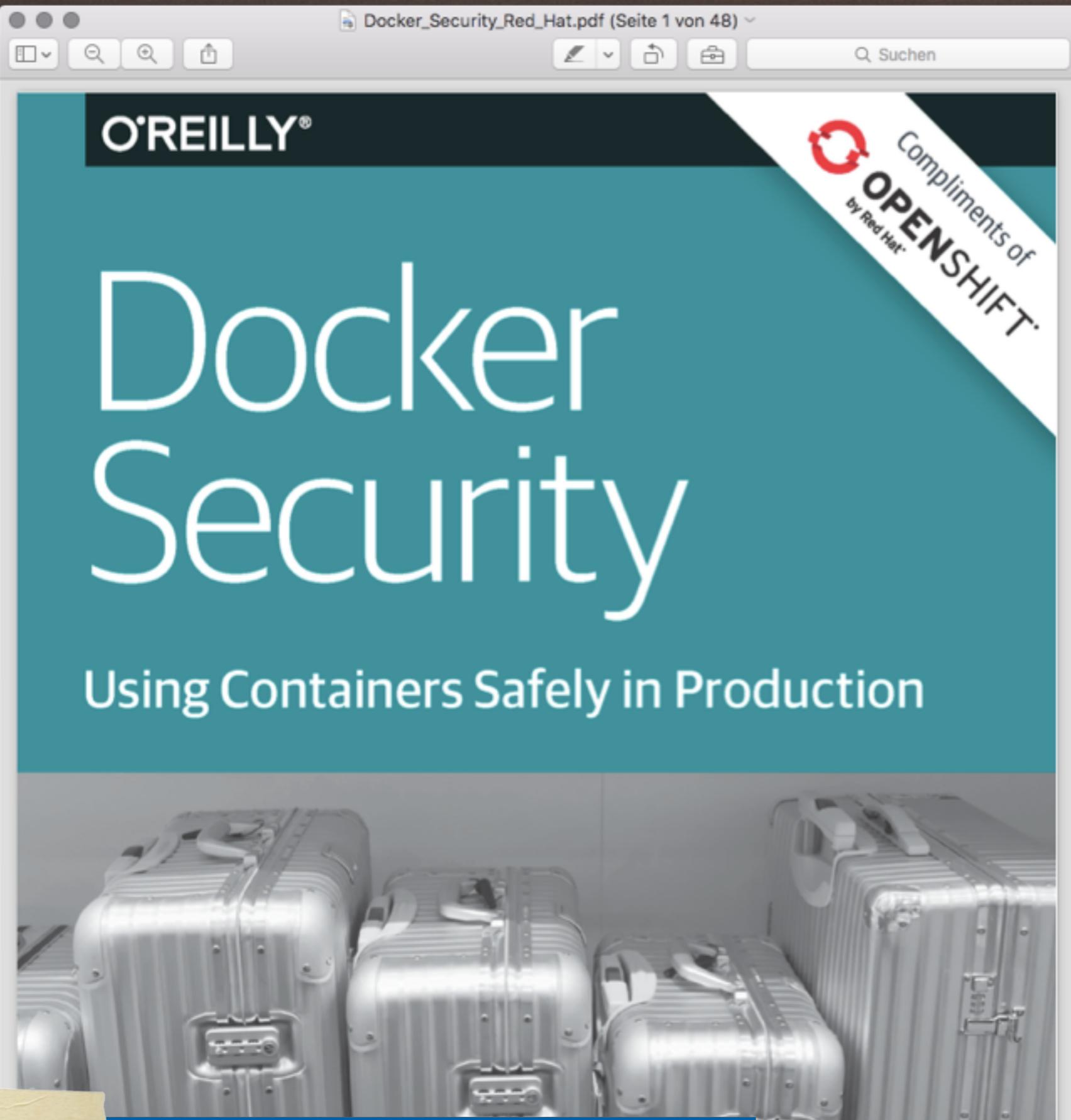
**CIS Docker 1.11.0 Benchmark**

v1.0.0 - 04-12-2016

Quelle

[https://benchmarks.cisecurity.org/tools2/docker/CIS\\_Docker\\_1.11.0\\_Benchmark\\_v1.0.0.pdf](https://benchmarks.cisecurity.org/tools2/docker/CIS_Docker_1.11.0_Benchmark_v1.0.0.pdf)

RTFM



Quelle

<https://www.openshift.com/promotions/docker-security.html>

RTFM

NCC\_Group\_Understanding\_Hardening\_Linux\_Containers-1.0.pdf (Seite 1 von 122)

Suchen

# **NCC Group Whitepaper**

## **Understanding and Hardening Linux Containers**

April 20, 2016 - Version 1.0

**Prepared by**  
Aaron Gratafiori - Technical Director

### **Abstract**

Operating System virtualization is an attractive feature for efficiency, speed and modern application deployment, amid questionable security. Recent advancements of the Linux kernel have coalesced for simple yet powerful OS virtualization via Linux Containers, as implemented by LXC, Docker, and CoreOS Rkt among others. Recent container focused start-ups such as Docker have helped push containers into the limelight. Linux containers offer native OS virtualization, segmented by kernel namespaces, limited through process cgroups and restricted through reduced root capabilities, Mandatory Access Control and user namespaces. This paper discusses these container features, as well as exploring various security mechanisms. Also included is an examination of attack surfaces, threats, and related hardening features in order to properly evaluate container security. Finally, this paper contrasts different container defaults and enumerates strong security recommendations to counter deployment weaknesses- helping support and explain methods for building high-security Linux containers. Are Linux containers the future or merely a fad or fantasy? This paper attempts to answer that question.



Quelle

[https://www.nccgroup.trust/globalassets/our-research/us/whitepapers/2016/april/ncc\\_group\\_understanding\\_hardening\\_linux\\_containers-10pdf/](https://www.nccgroup.trust/globalassets/our-research/us/whitepapers/2016/april/ncc_group_understanding_hardening_linux_containers-10pdf/)

RTFM

### 9.13 Container Defaults

Listed below are the relevant security features for the three major container platforms explored within this paper. Each security feature is covered directly or indirectly within this paper and the title can be clicked, for those which are covered in detail, in order to jump to the relevant section. To avoid any misconceptions, the following parameters are defined as to their use in the table below:

- Default: The security feature is enabled by default.
- Strong Default: The most secure configuration is enabled by default.
- Weak Default: A less secure configuration is enabled by default.
- Optional: The security feature can be optionally configured. This is not a given weakness unless no other equivalent feature can be configured or enabled.
- Not Possible: The security feature cannot be configured in any way, no documentation exists, the feature is still under development, or the feature is not planned to be implemented.

Available Container Security Features, Requirements and Defaults			
Security Feature	LXC 2.0	Docker 1.11	CoreOS Rkt 1.3
User Namespaces	Default	Optional	Experimental
Root Capability Dropping	Weak Defaults	Strong Defaults	Weak Defaults
Procfs and Sysfs Limits	Default	Default	Weak Defaults
Cgroup Defaults	Default	Default	Weak Defaults
Seccomp Filtering	Weak Defaults	Strong Defaults	Optional
Custom Seccomp Filters	Optional	Optional	Optional
Bridge Networking	Default	Default	Default
Hypervisor Isolation	Coming Soon	Coming Soon	Optional
MAC: AppArmor	Strong Defaults	Strong Defaults	Not Possible
MAC: SELinux	Optional	Optional	Optional
No New Privileges	Not Possible	Optional	Not Possible
Container Image Signing	Default	Strong Defaults	Default
Root Interation Optional	True	False	Mostly False

# Teil IV: Zusammenfassung

# Wa(h)lwerbung?



Quelle:

<http://cdn2.spiegel.de/images/image-806145-galleryV9-ygfz.jpg>

# *State of the Union*

Queueing, MPI, ...

## Short summary: **Queueing Systems**

\* Docker „Integration“ (Single Node Containers!) in LSF available, others work on it too.

## **Distributed workloads with MPI**

\* Possible, but requires extra effort like Service Discovery, SDN, but technically works

\* Proof of Concept with MPI-based OpenFOAM Containers with OpenLava available

## **GPUs, Infiniband, SR-IOV** possible

\* Shifter (NERSC)



Quelle:

<https://www.flickr.com/photos/protohiro/3847864550>

# *One more thing...*

## *Docker immer unsicher?*

twitter.com/Ben\_Hall/status/728596633978572801

Startseite Über uns Twitter durchsuchen Hast Du einen Account? Anmelden

X

```
ubuntu@ubuntu:~/ebpf_mapfd_doubleput_exploit$ ./doubleput
starting writev
woohoo, got pointer reuse
writev returned successfully. if this worked, you'll have a root shell in <=60 seconds.
suid file detected, launching rootshell...
we have root privs now...
root@ubuntu:~/ebpf_mapfd_doubleput_exploit# whoami
root
```

Product Hunt LIVE - ProductHunt, Inc. 15m "Everything starts with a great product." —tristanwalker producthunt.com/live/tristan-w...

P tristan walker on Tristan Walker LIVE chat P

>

**Ben Hall** @Ben\_Hall · 20 Std.  
I do enjoy a good Linux Kernel exploit on a Friday afternoon... Especially when @docker protects the host from it :)

Tester, Developer, Trainer 60 87

Quelle

[https://twitter.com/Ben\\_Hall/status/728596633978572801](https://twitter.com/Ben_Hall/status/728596633978572801)

root exploit

Startseite Über uns Twitter durchsuchen Hast Du einen Account? Anmelden

```
ubuntu@ubuntu:~/ebpf_mapfd_doubleput_exploit$ sudo docker run -it \
> -u $(id -u) \
> --security-opt=no-new-privileges \
> -v `pwd`:/exploit \
> ubuntu bash
sudo: unable to resolve host ubuntu
I have no name!@a353bc731b88:$ id
uid=1000 gid=0(root) groups=0(root)
I have no name!@a353bc731b88:$ cd exploit/
I have no name!@a353bc731b88:/exploit$ ./doubleput
suid file detected, launching rootshell...
suidhelper: setuid/setgid: Operation not permitted
I have no name!@a353bc731b88:/exploit$ mkdir: cannot create directory 'fuse_mount': File exists
doubleput: system() failed
doubleput: expected BPF_PROG_LOAD to fail with -EINVAL, got different error: Operation not permitted
I have no name!@a353bc731b88:/exploit$
```

Ben Hall @Ben\_Hall · 20 Std.  
I do enjoy a good Linux Kernel exploit on a Friday afternoon... Especially when @docker protects the host from it :)

Tester, Developer,  
Trainer, Building K  
learning

Quelle

[https://twitter.com/Ben\\_Hall/status/728596633978572801](https://twitter.com/Ben_Hall/status/728596633978572801)

non-root exploit

## Virtualisierung 2.0? Nö ;)

bessere Performance als  
klassische Virtualisierung im HPC

*Sicherheit*  
muss gewährleistet sein



science + computing

| an atos company



Vielen Dank für Ihre Aufmerksamkeit.

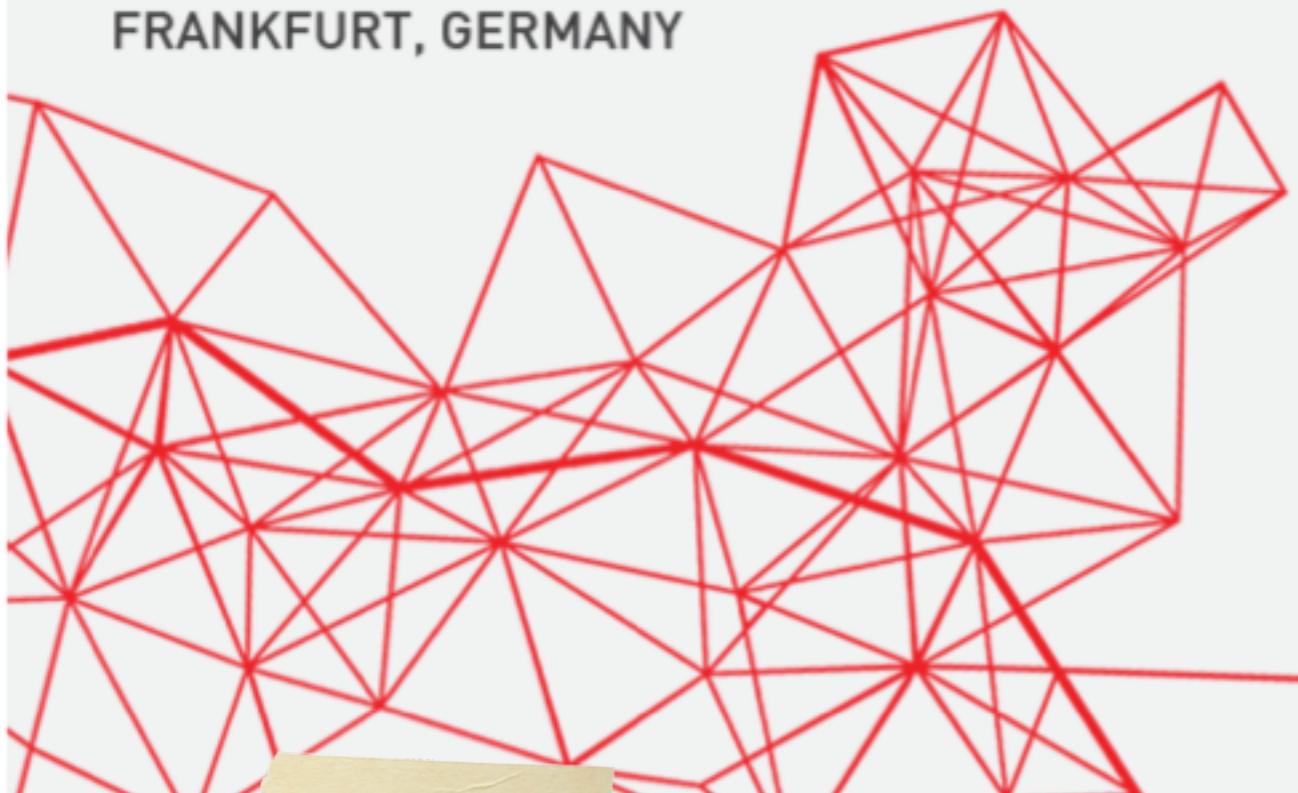
**Holger Gantikow**

science + computing ag  
[www.science-computing.de](http://www.science-computing.de)

Telefon: 07071 9457 - 503  
E-Mail: [h.gantikow@science-computing.de](mailto:h.gantikow@science-computing.de)

# ISC HIGH PERFORMANCE

SUNDAY, JUNE 19 -  
THURSDAY, JUNE 23, 2016  
FRANKFURT, GERMANY



6

REGISTER NOW



TRAVEL & STAY



AGENDA



ISC 2017

# Frage? Antwort!





**science + computing**

| an atos company



<http://www.science-computing.de>  
<https://www.science-computing.de/jobs>