



science + computing

| an atos company



Linux Containers in a Nutshell

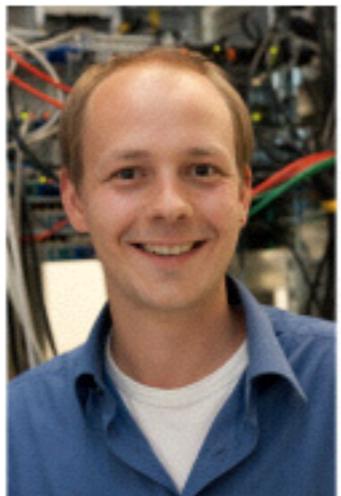
ISC HPC - Workshop: Linux Containers to Optimise IT Infrastructure for HPC & BigData

June 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 Hocherfügbarekeit"), 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

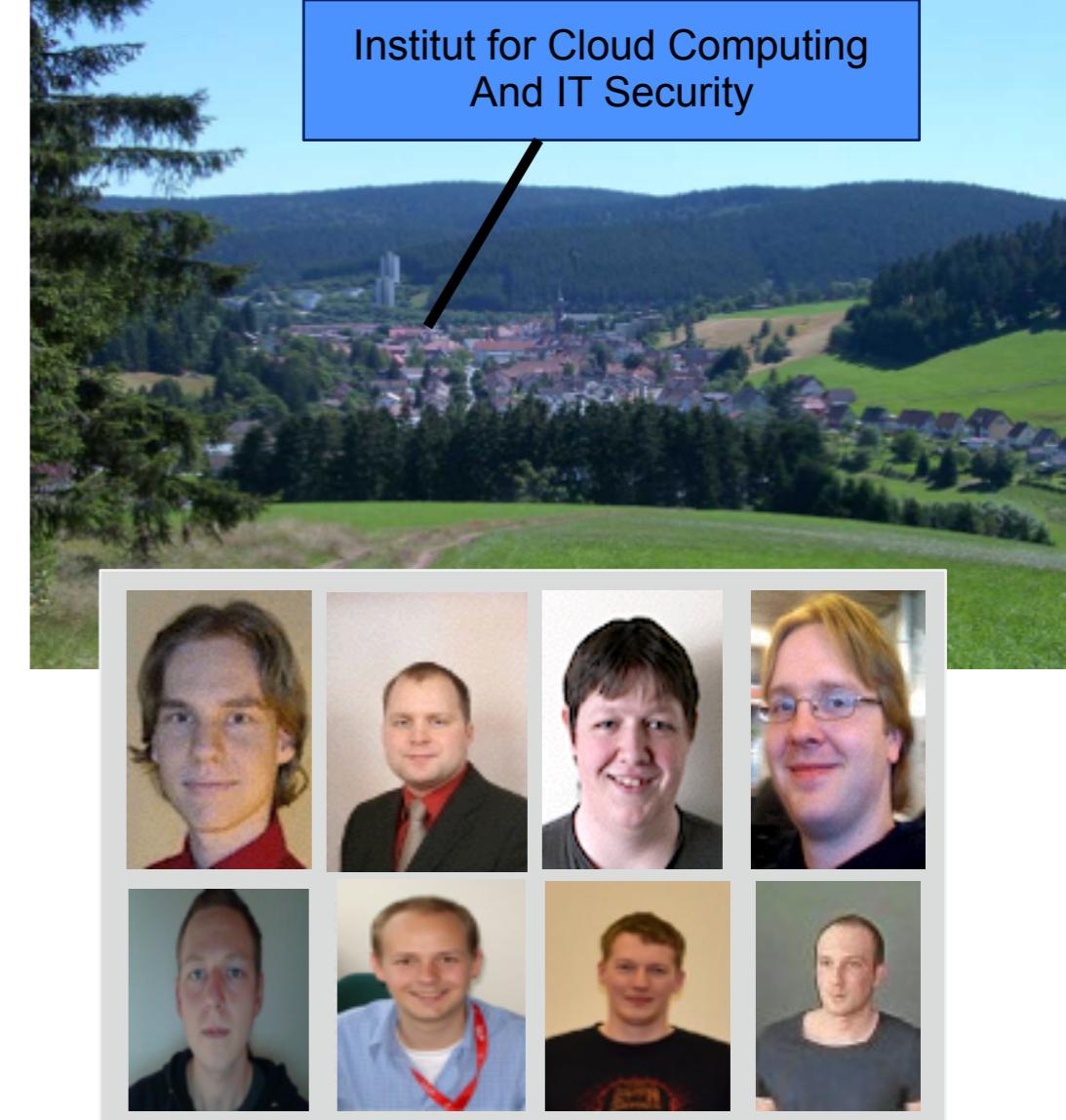
IT-spezifisch momentan: Virtualisierung (Xen, ESX, ESXi, KVM), Cluster Computing (HPC, HA), Cloud Computing (speziell: IaaS, HPCaaS), OpenNebula, OpenSolaris, ZFS, XMPP, SunRay ThinClients - ansonsten: Freie Software, Musik, Gitarre, Fotografie

https://www.xing.com/profile/Holger_Gantikow
also on LinkedIn, Twitter, \$SOCIALNETWORK

Institut für Cloud Computing und IT-Sicherheit (IfCCITS) *previous: Cloud Research Lab*

facts:

- founded 10/2015
- 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:

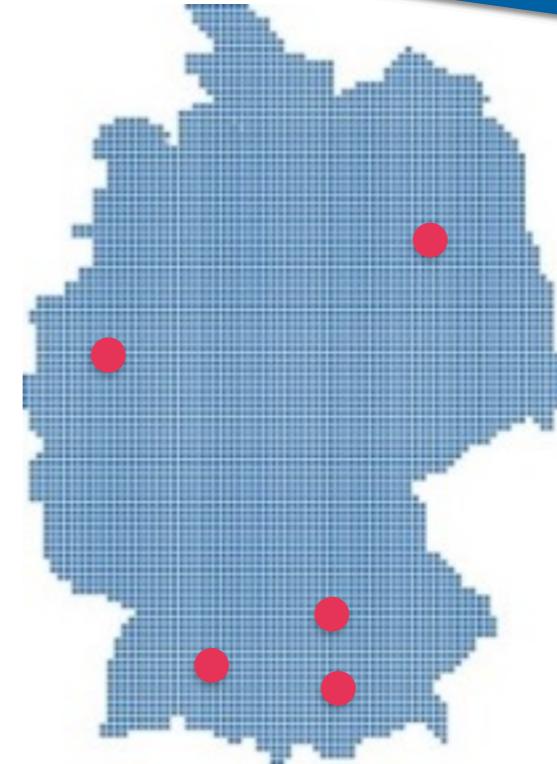
- Distributed Systems
- IT Security
- Cloud Computing
- Industry 4.0; IoT

<http://www.wolke.hs-furtwangen.de>

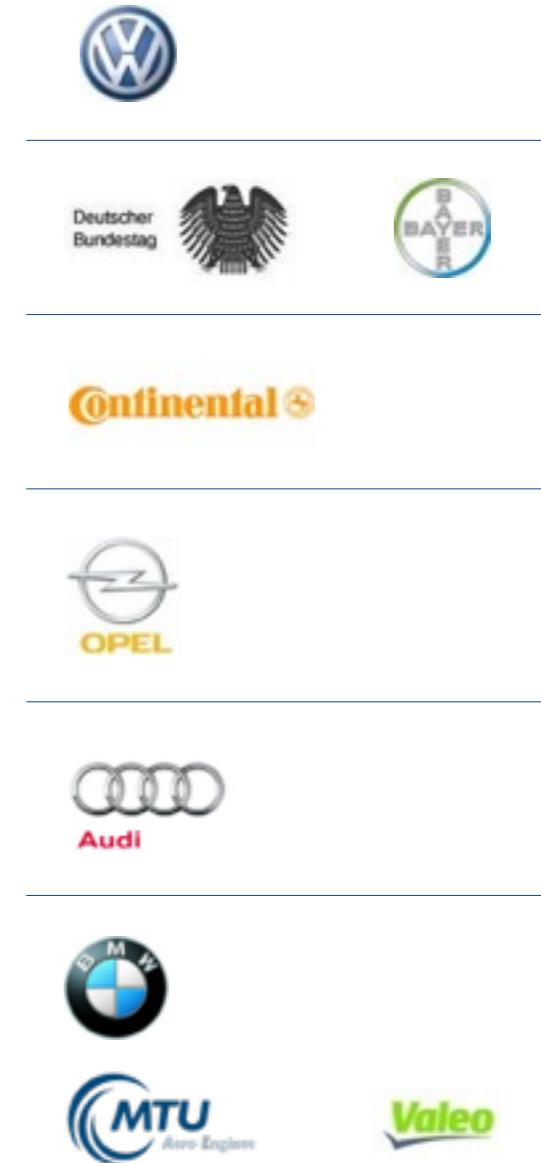
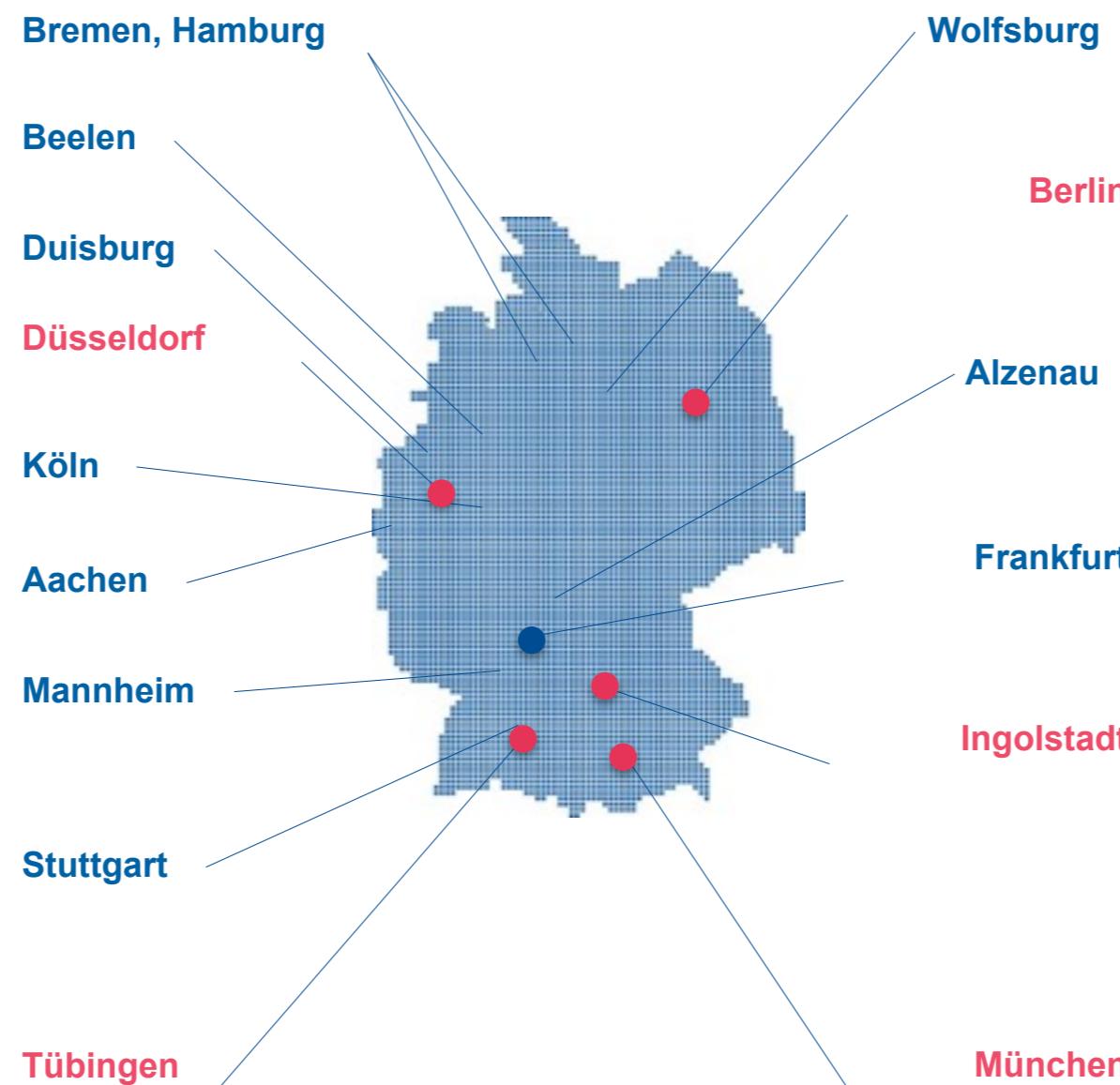
science + computing ag

Our focus:
IT-services and software for technical
computation environments

Founded	1989
Offices	Tübingen
	Munich
	Berlin
	Düsseldorf
	Ingolstadt
Employees	287
Owner	Atos SE (100%)
	before: Bull
an. turnover (2013)	30,7 Mio. Euro



Customers





science + computing

| an atos company

soon going to be...

Atos

Trusted partner for your Digital Journey

Agenda

Part I: What is Docker?

Part II: Why Docker matters

Part III: Little bit of Security

Part IV: What's new?

Part V: Getting started

Part 0: A few questions...

Raise your hand!

Who...

...has heard of Docker?

...knows what Docker is?

...has tried Docker?

...uses Docker?

...uses Docker in production?

...with additional tools?

Part I: **What is Docker?**

DOCKER



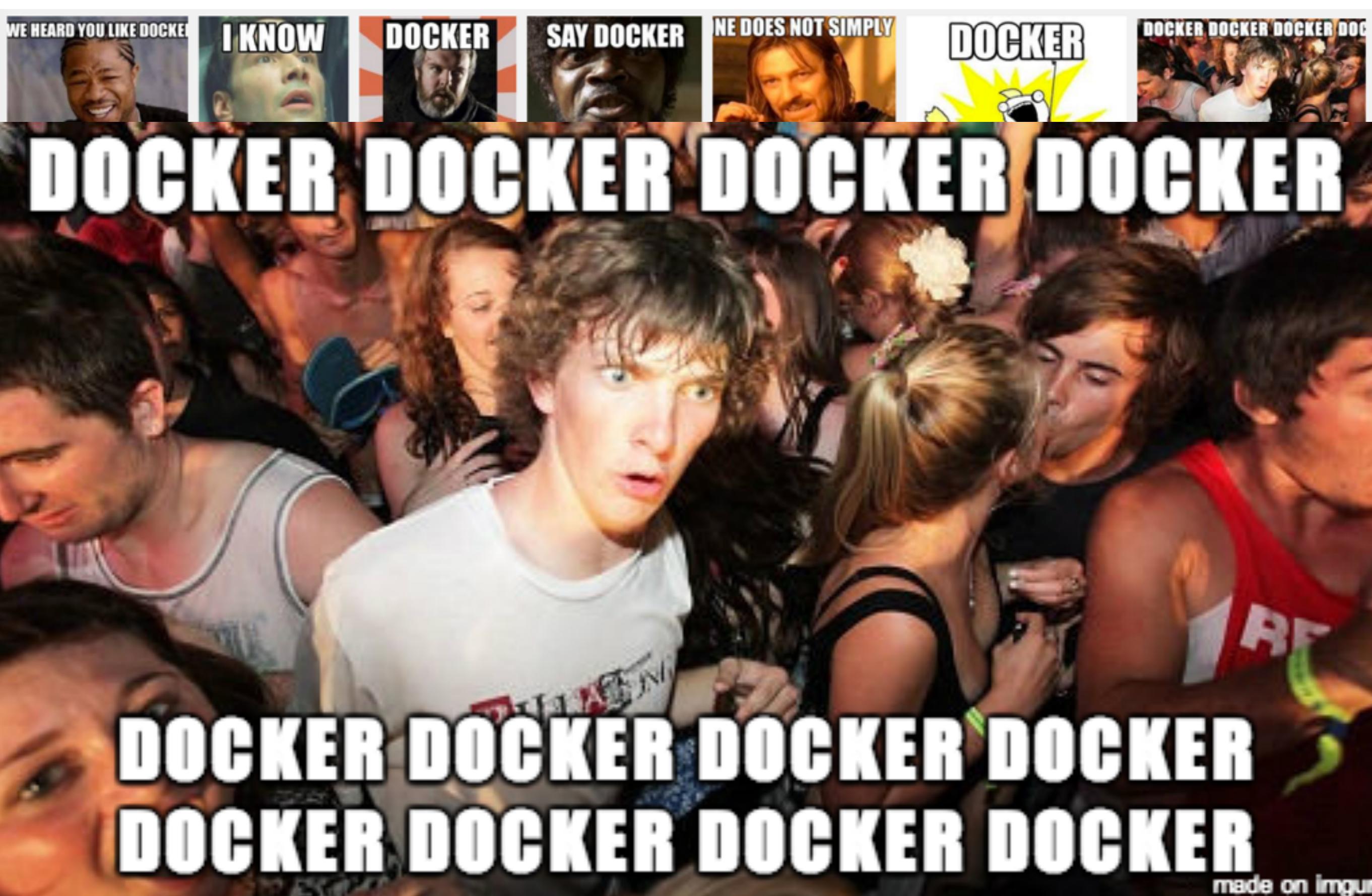
ALL THE THINGS

Source: Docker all the things

<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>



made on imgur

Source: Google „Docker Memes“ ;)

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

Docker 101



docker

Source:

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

Both Linux Kernel Features

Namespaces

several subsystems ns-aware

- illusion of running in isolation

Control Groups (cgroups)

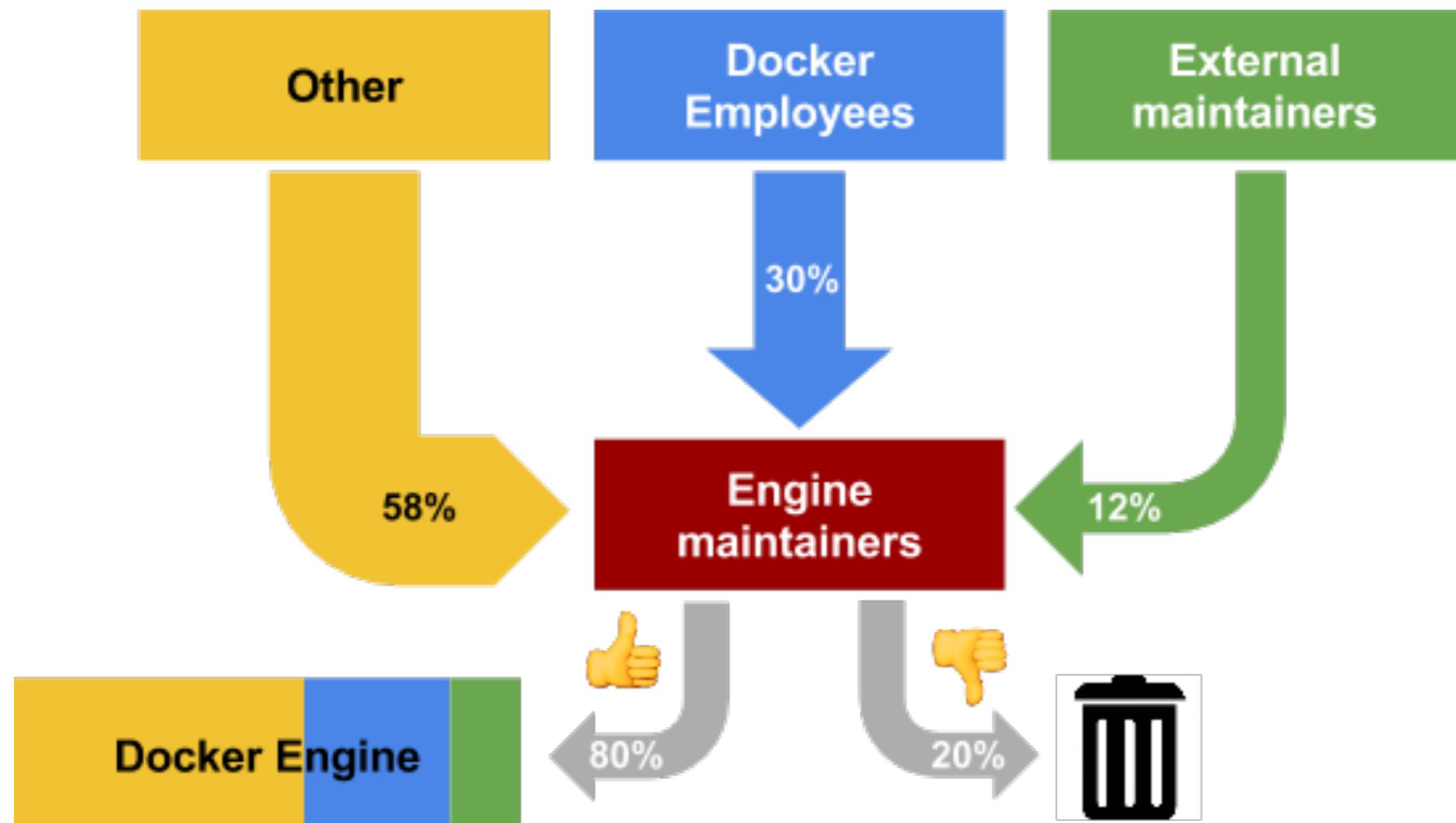
several resources controllable

- limits resource usage

Namespace	Description	Controller	Description
pid	Process ID	blkio	Access to block devices
net	Network Interfaces, Routing Tables, ...	cpu	CPU time
ipc	Semaphores, Shared Memory, Message Queues	devices	Device access
mnt	Root and Filesystem Mounts	memory	Memory usage
uts	Hostname, Domainname	net_cls	Packet classification
user	UserID and GroupID	net_prio	Packet priority

Docker History

Open Source @Docker



Source: Open Source at Docker, Part 1: The People

<https://blog.docker.com/2016/05/open-source-docker-part-1-people/>

Terminology

Docker Components

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

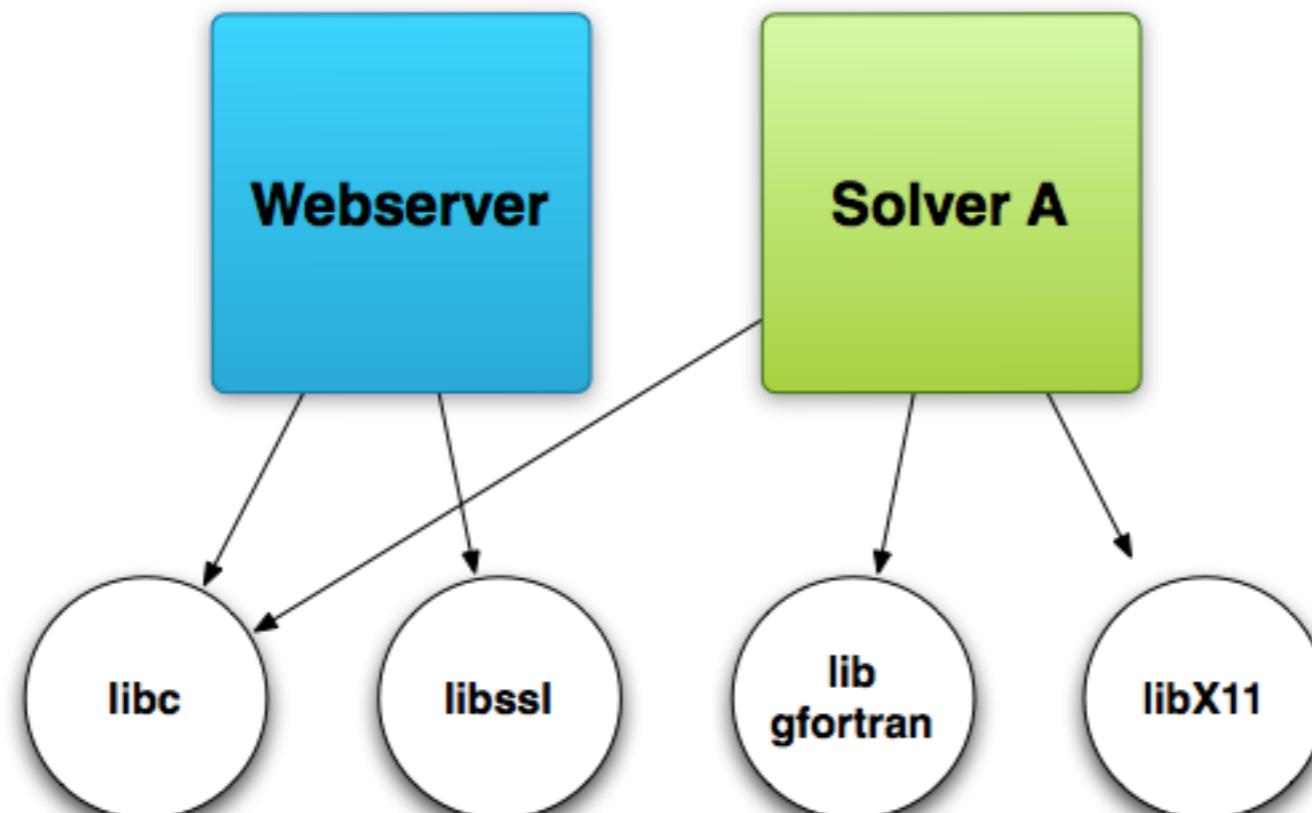
What can Docker do for you?

Save lives ;)

Devs vs Ops

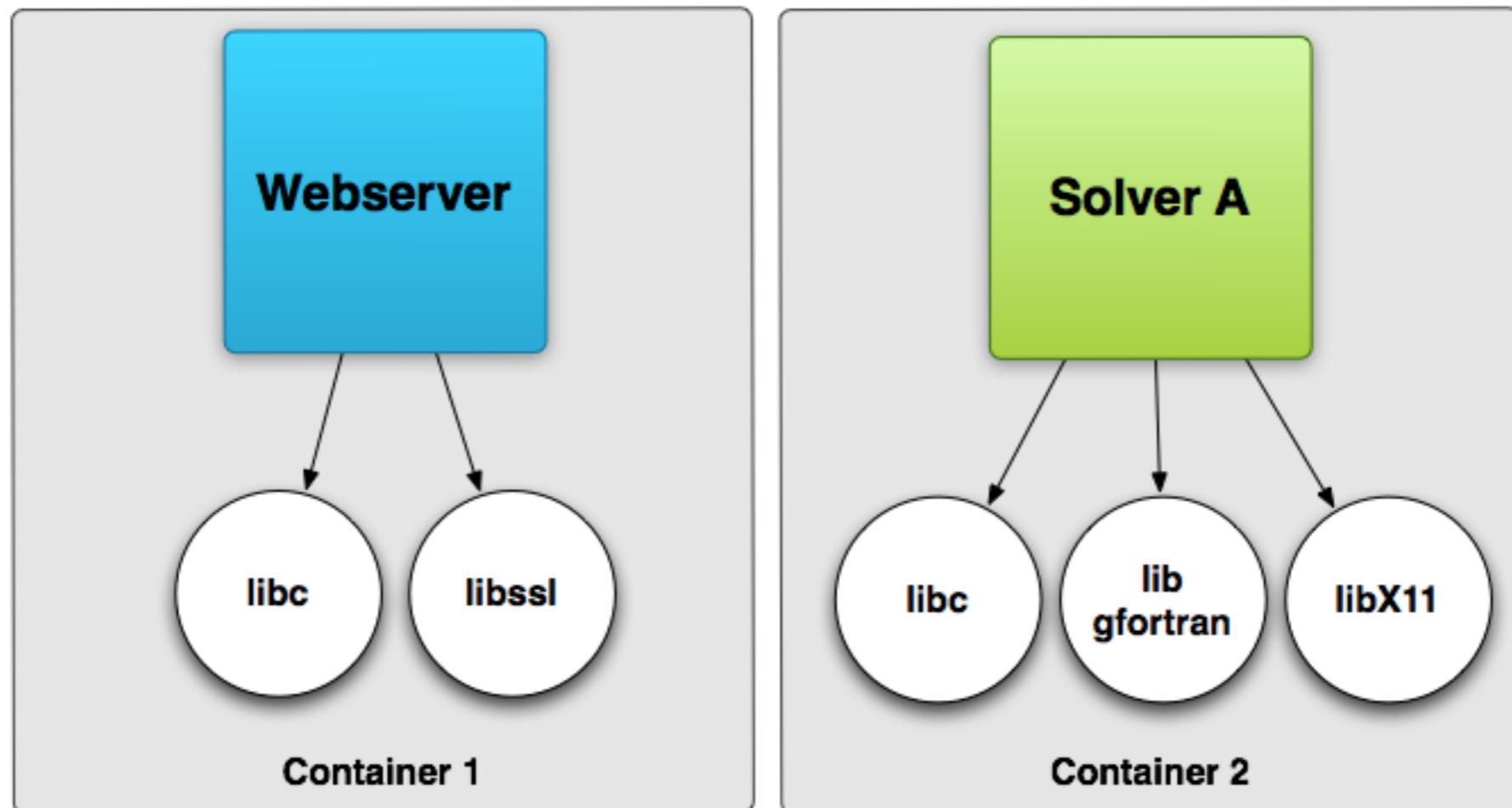
isolate dependencies

Classic setup (without Docker)



Things tend to *break* when updating libraries/OS or applications
Problems with legacy code on current OS and vice versa
Even different versions of same application can have conflicting requirements.

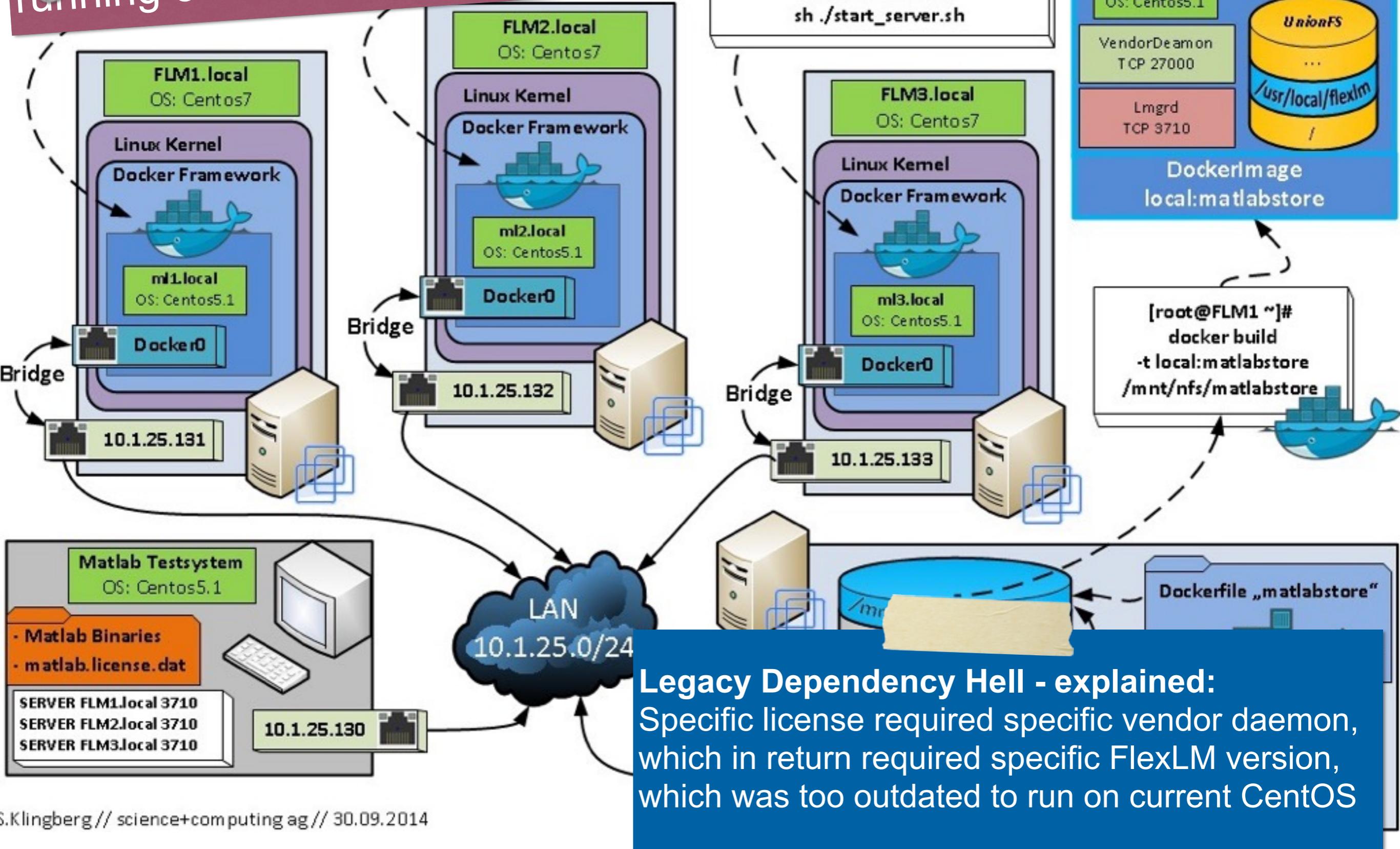
Containerized setup



Easily shareable with 3rd parties (colleague, customers, provider, ...)
Use same *environment* on Laptop, Workstation, Cluster, Cloud VM, bare-metal Cloud!
Only Docker *runtime* required.

Not exactly typical example...

Containerized historic FlexLM „triple“ running on one Docker Host



Workflow^{^2} ;)

Docker highly useable for „containing“ + sharing workflows, such as in genome sequencing, where pipelines with complex dependencies need to be shared.
Added benefit: „reproducibility“ - data can be reprocessed using the exact same toolstack + versions originally used. Great for verifying results - even by a third party.

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^{1,2,5*}, Günter Jäger¹, Alexander Herbig^{1,2,5}, Alexander Seitz¹, Christian Kniep⁴, Johannes Krause^{2,3,5} and Kay Nieselt¹ 

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

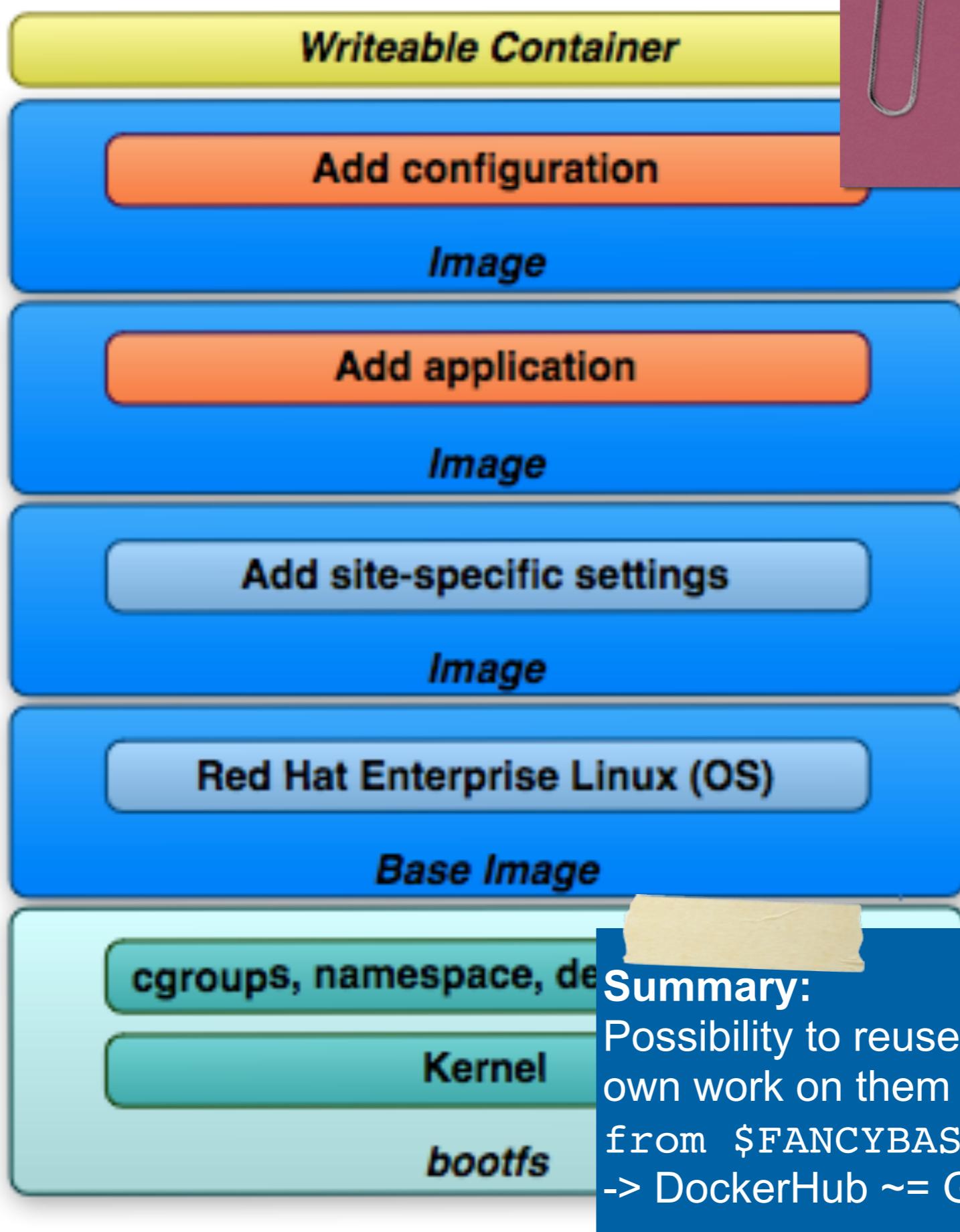
In ancient DNA (aDNA) studies, often billions of sequence reads are analyzed to determine the genomic sequence [1–3]. Newly developed

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

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

Workflows

image layers



Docker Hub

Docker Toolbox

Docker Compose

Docker Swarm

Docker Machine

Docker Universal Control Plane

Docker Trusted Registry

Docker Cloud

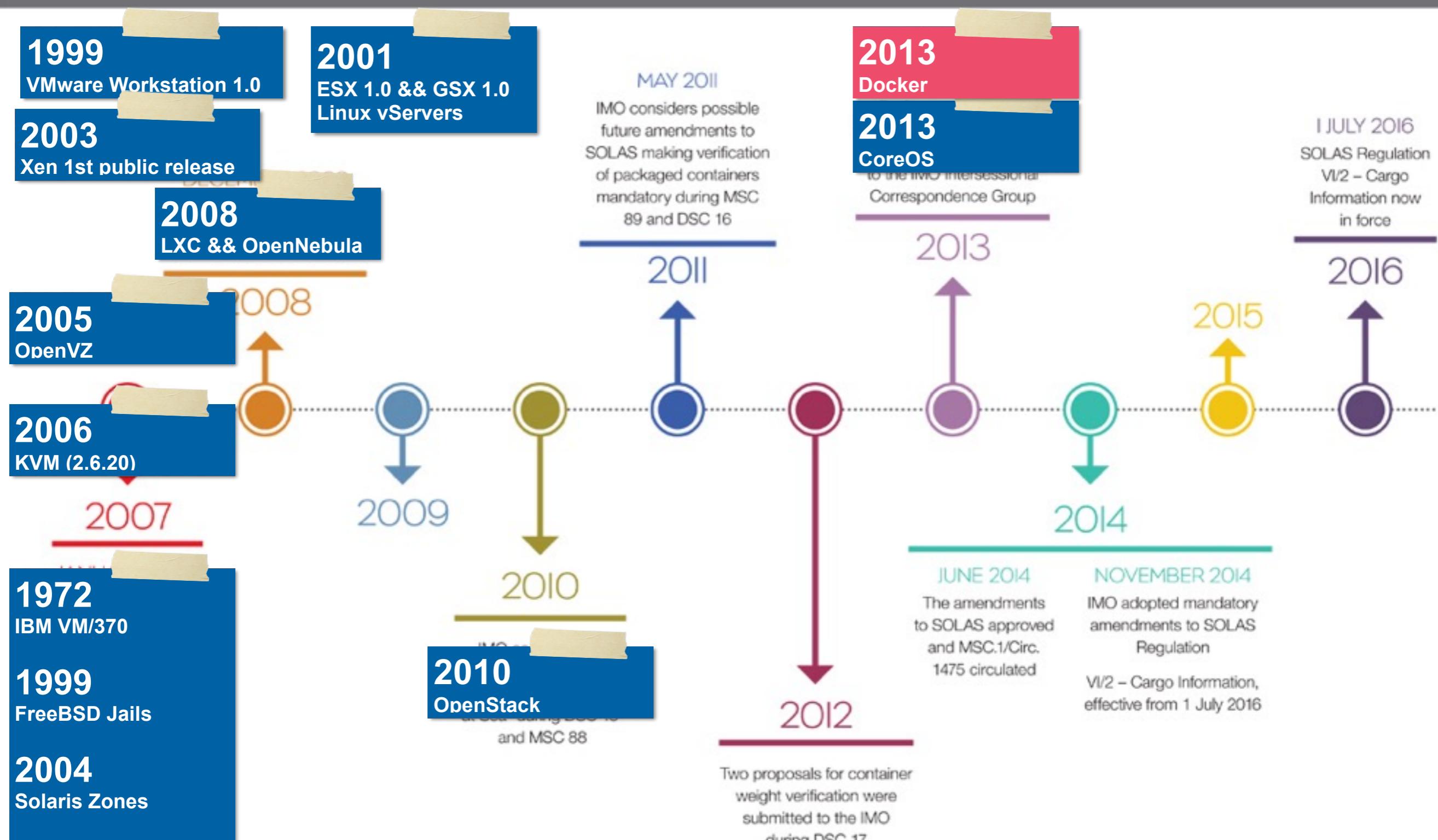
Docker „XYZ“ ;)

Kubernetes

Virtualization 2.0?

Harder, Better, Faster, Stronger?

TIMELINE CHART



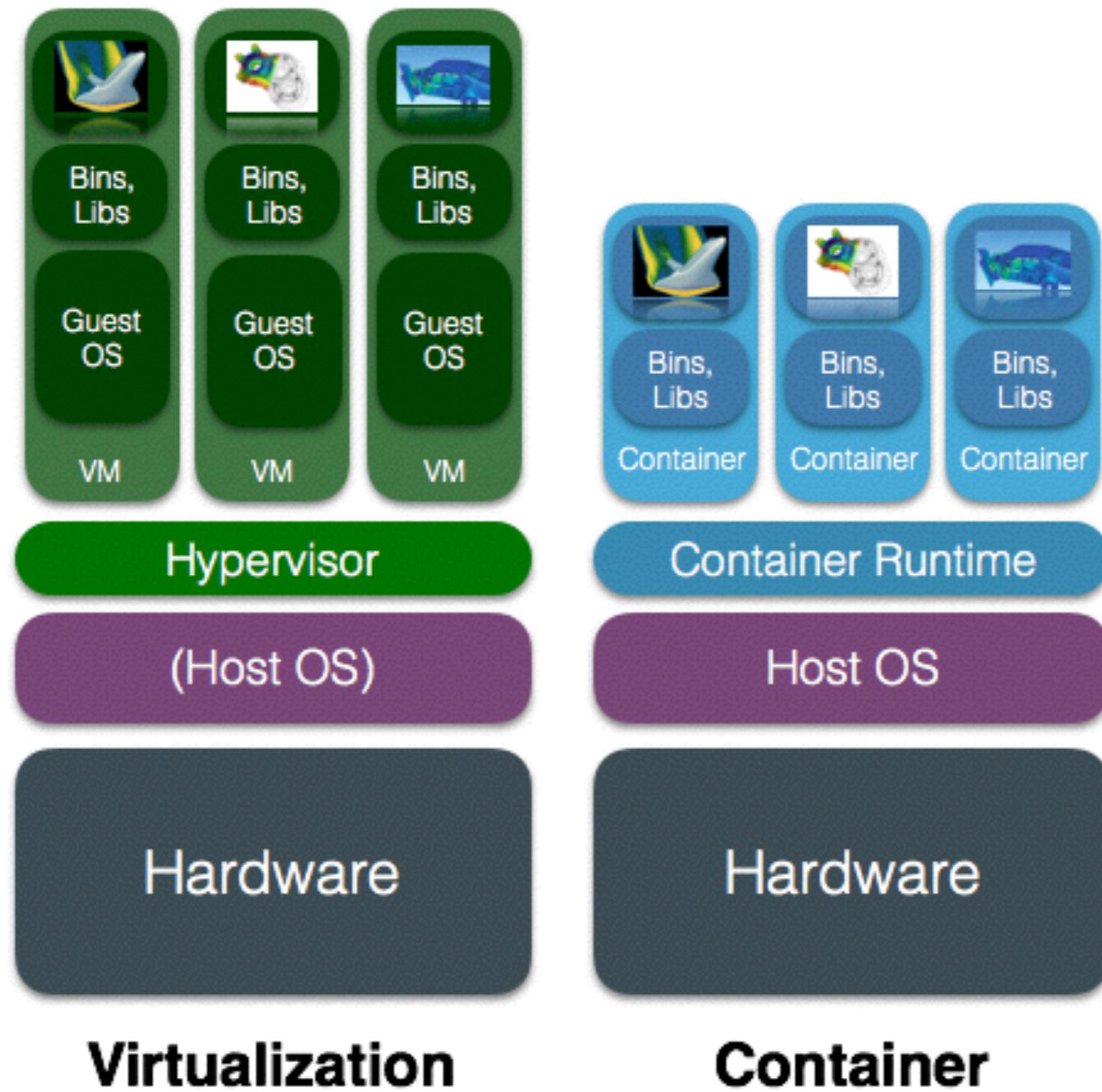
Source

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

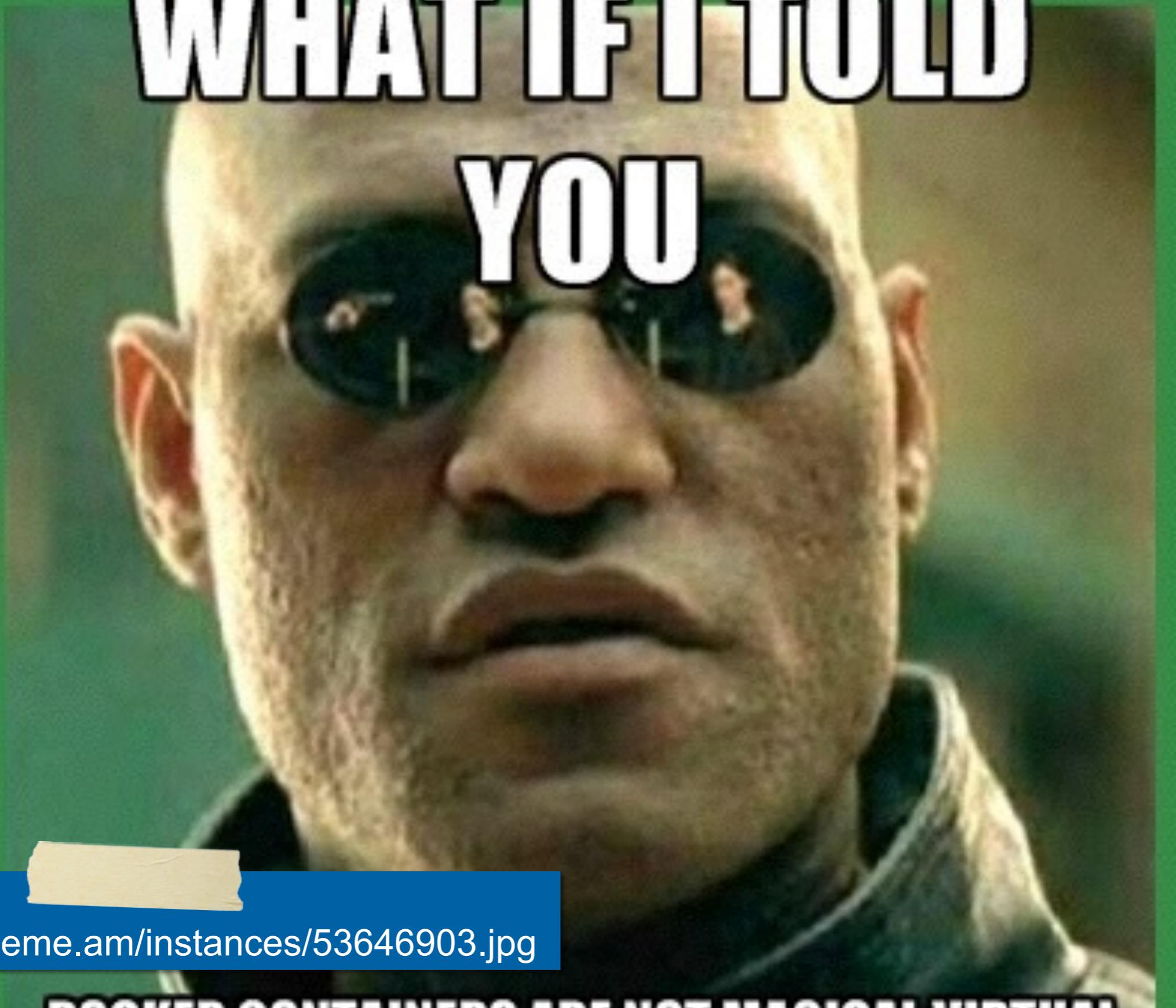
NORTH



Virtualization *compared*



WHAT IF I TOLD YOU



Source:

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

**DOCKER CONTAINERS ARE NOT MAGICAL VIRTUAL
MACHINES**

memegenerator.net

What's different?

compared to VMs?

lightweight vs fat requirement when shipping application

compared to other „containerizers“?

systemd-nspawn, LXC, rkt

Docker: (finally) containers for the masses, by offering easy to use tools + workflows.

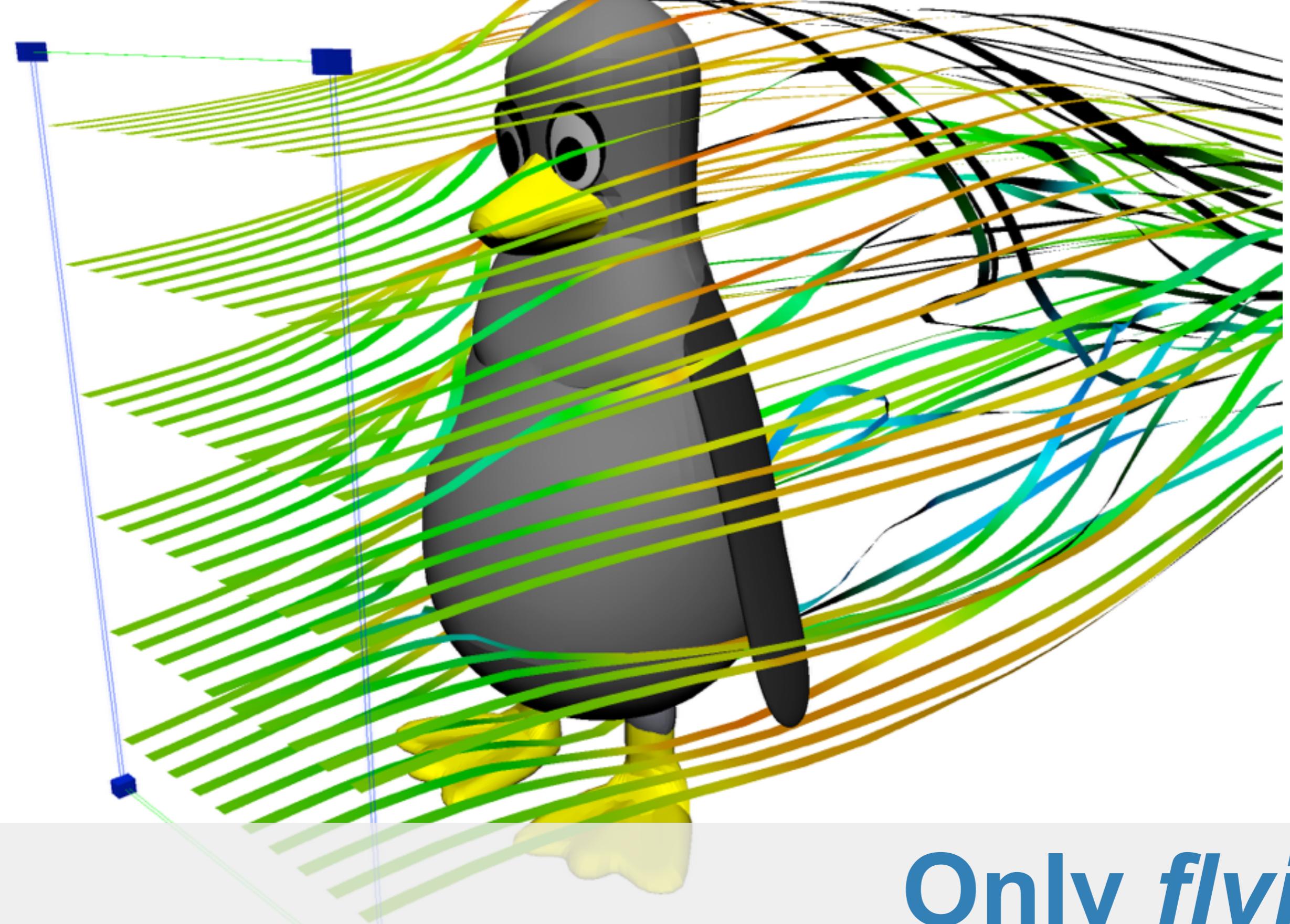
- * **systemd-nspawn**: "only" „chroot on steroids“
- * **LXC**: focus rather system containerization than application containerization
- * **rkt**: most similar to Docker, interesting features - might be less production-ready. Yet.

Part II: Why Docker matters... ...for computing?

High Performance Computing



Source: Dieter Both, Bull GmbH



Only *flying*
penguins?

Source: Dr. Martin Schulz

We all got issues...

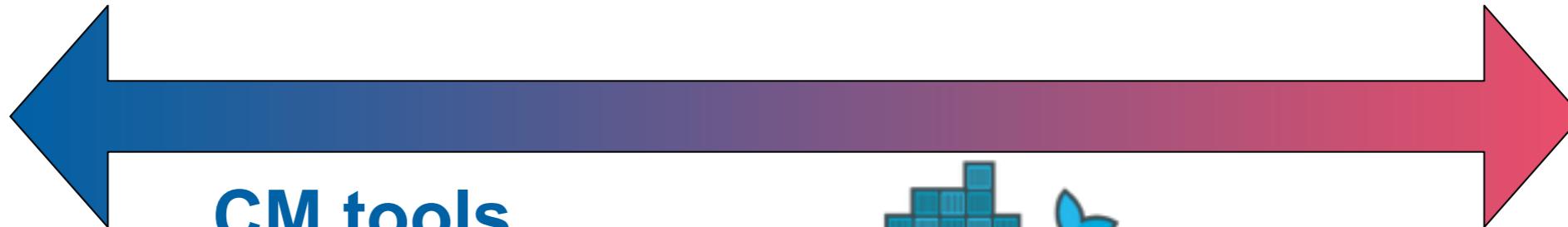
clean slate
for applications

deployment!

docker pull vs yum install

Deployment compared

**less portable,
minimal overhead**



**manual
configuration**

**traditional
VMs**

Docker

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

homogenize heterogeneous Clusters

passing on an environment

The screenshot shows a web browser window with the URL openfoam.com in the address bar. The page content is as follows:

OpenFOAM The open source CFD toolbox 

Home Products Services Download Code Documentation News

About us Contact Jobs Legal FOLLOW US ON  twitter

Releases

- Overview
- Current Release
- Installation
 - ... Source
 - ... Binary
 - ... Windows
- Release History

OpenFOAM® Binary Installation

OpenCFD Limited uses [Docker Hub](#) to distribute pre-compiled versions of OpenFOAM+ for Linux, Mac OS X and Windows, including a complete development environment.

Why Docker for OpenFOAM+?

Docker containers enable binaries compiled on a given Linux environment to be run on other platforms without any performance degradation. Docker also operates on Windows and Mac OS X wrapped in a light-weight Virtual Box.

An image of OpenFOAM+ contains binaries and source code. The Docker environment provides:

- A complete development environment to compile local modifications and create executables.
- A consistent behaviour of the OpenFOAM+ across all platforms

Check if your (Linux or Windows or Mac OS X) system is supported by visiting <https://docs.docker.com/engine/installation/>

Installing Docker

On Linux

- Please follow the instructions on <https://docs.docker.com/engine/installation/>
- On Ubuntu 15.10:

```
sudo apt-get install docker-engine
sudo service docker start
```

Source: Docker for OpenFOAM+
<http://www.openfoam.com/download/install-binary.php>

ISV packages

Reproducibility

²¹

Science, bitches!



Source:

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

additional resources

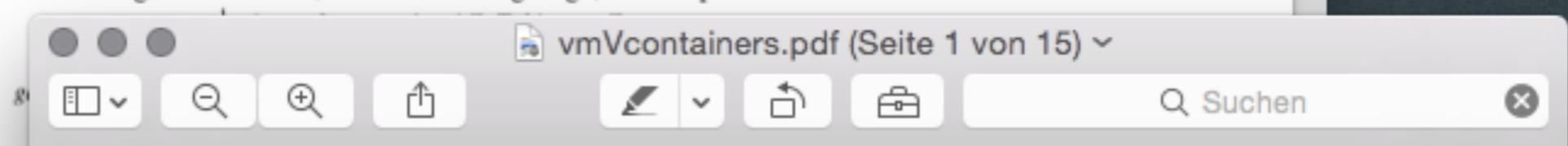
Why not use
classical virtualization?

well...

Benchmarks...

Container-Based Virtualization for HPC

Holger Gantikow¹, Sebastian Klingberg¹, Christoph Reich²



Keywords: Cont

Abstract: Expe
insta
comp
This
Com
(Doc
analy

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

1 INTROD

Applications in High Performance Computing (HPC) come to resource management and interconnection are traditionally run on physical systems called clusters.

Such a cluster has good performance, but often it turns up: a) The operati

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

Source: 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 - Structure

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]

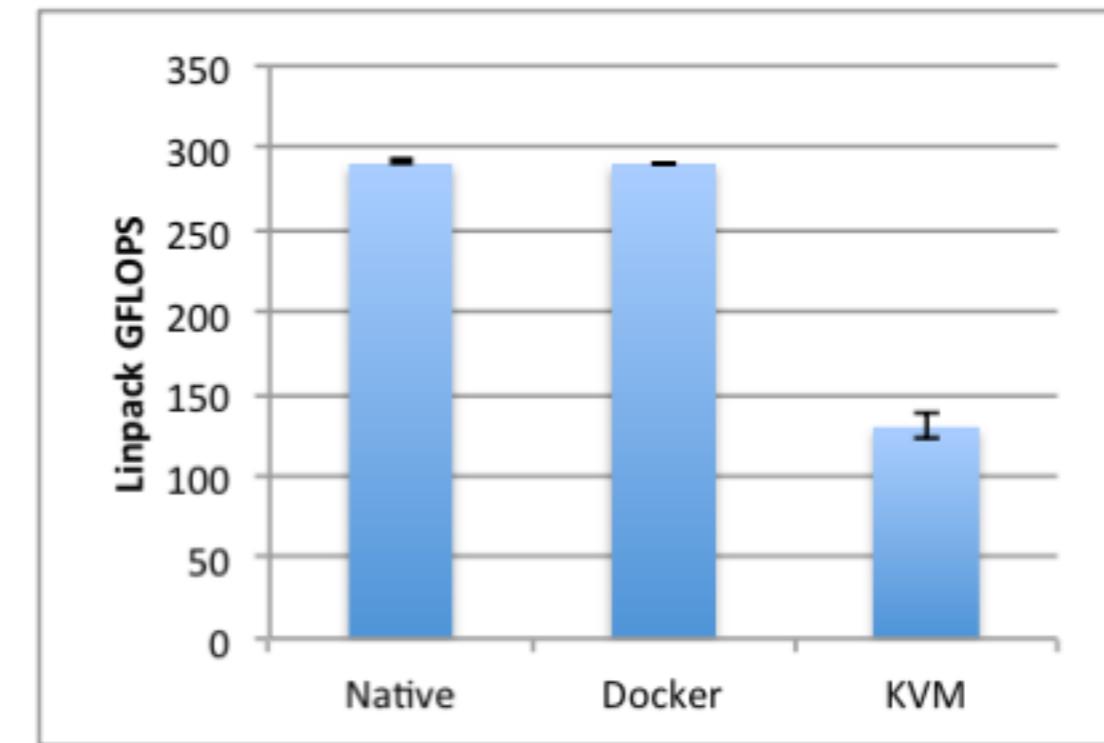


Figure 1. Linpack performance on two sockets (16 cores). Each data point is the arithmetic mean obtained from ten runs. Error bars indicate the standard deviation obtained over all runs.

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

Source: 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 - Results

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

Source: 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.

Docker@HPC

Container-Based Virtualization for HPC

Holger Gantikow¹, Sebastian Klingberg¹, Christoph Reich²

¹*science & computing AG, Tübingen, Germany*

²*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.

Nowadays, there are several ways to overcome these disadvantages. One way is to use virtual machines (VMs), which are virtual environments running on top of a host OS. Another way is to use container technology, which runs directly on the host OS. In this paper, we will focus on container technology and its advantages for HPC.

The rest of the paper is organized as follows: In section 2, we will introduce Docker, a popular container technology. In section 3, we will compare Docker with VMs and bare-metal installations. In section 4, we will present a case study showing the use of Docker in a private HPC site. Finally, in section 5, we will conclude the paper with a summary and an outlook.

© 2015 Sebastian Klingberg. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Thanks!
@Sebastian Klingberg

Docker@HPC - Setup

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.

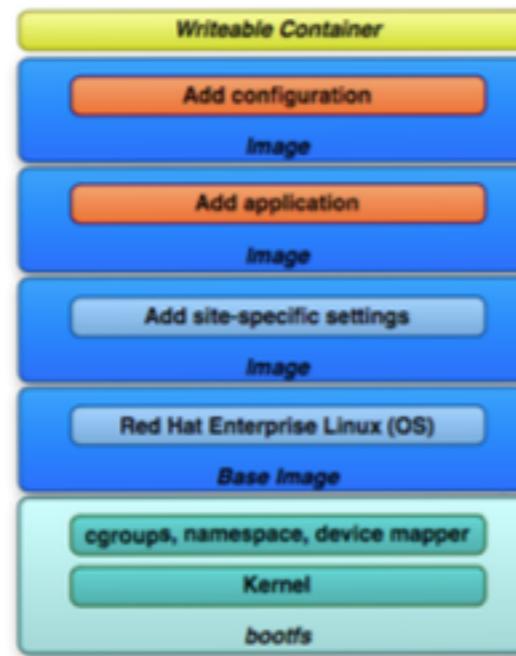


Figure 4: The layers of the Docker file system

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.

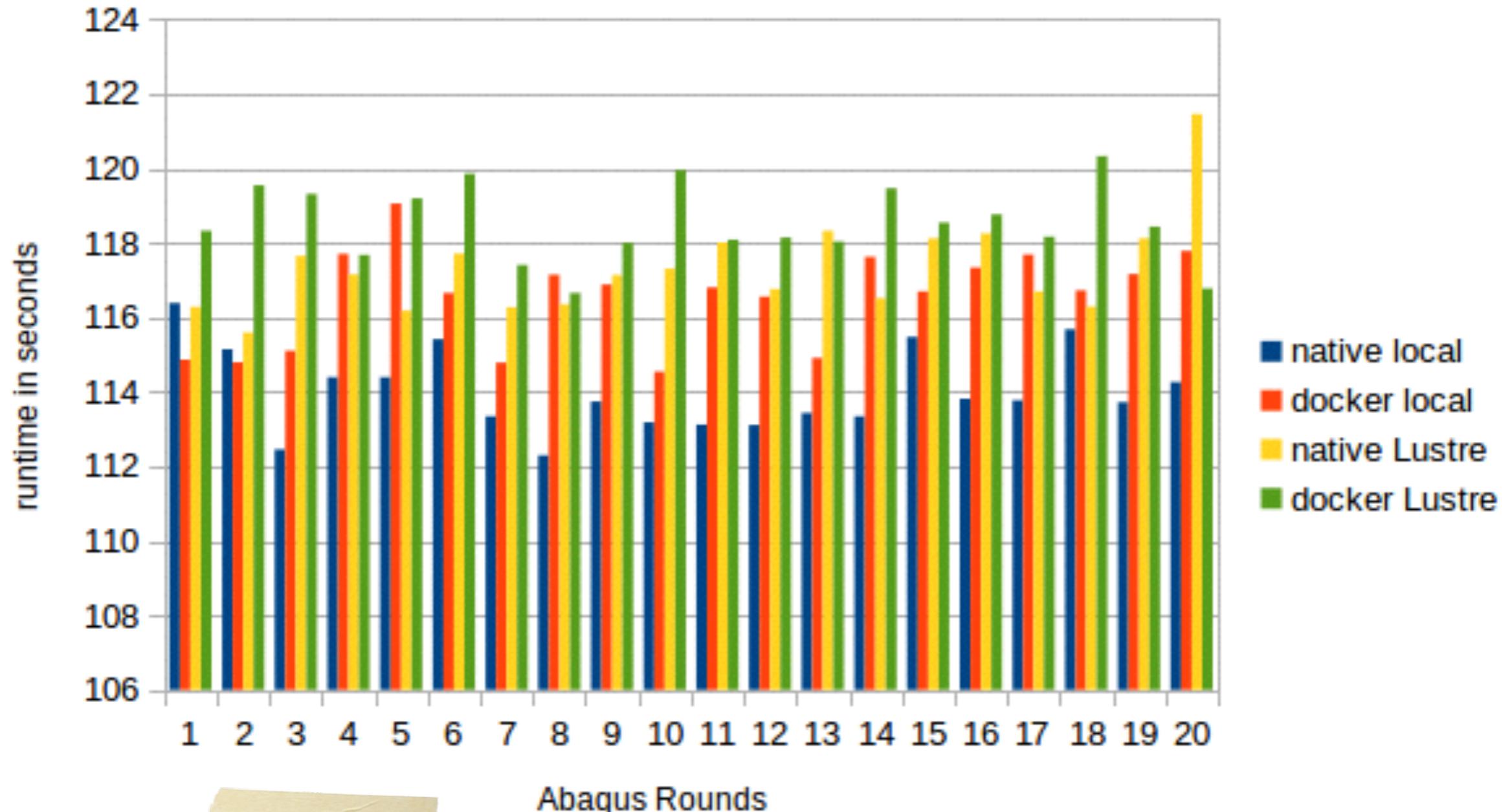
The application was installed to local disks on a CentOS 7 SunFire X2200 server with the following hardware configuration:

- 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 - Results



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

```
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

everyday life...

Part III: **Little bit of Security**

Opinions matter... Quotes



Source: 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

..., numbers count
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



Sources of Frustration

Docker - *misunderstandings*

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

[Read More](#)

[View](#)

[Edit](#)

[Delete](#)

by [Jess Frazelle](#) | 2015-02-21 | [OS X](#)

Source: Docker Containers on the Desktop

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

Docker - carelessness

Jessie Frazelle's Blog

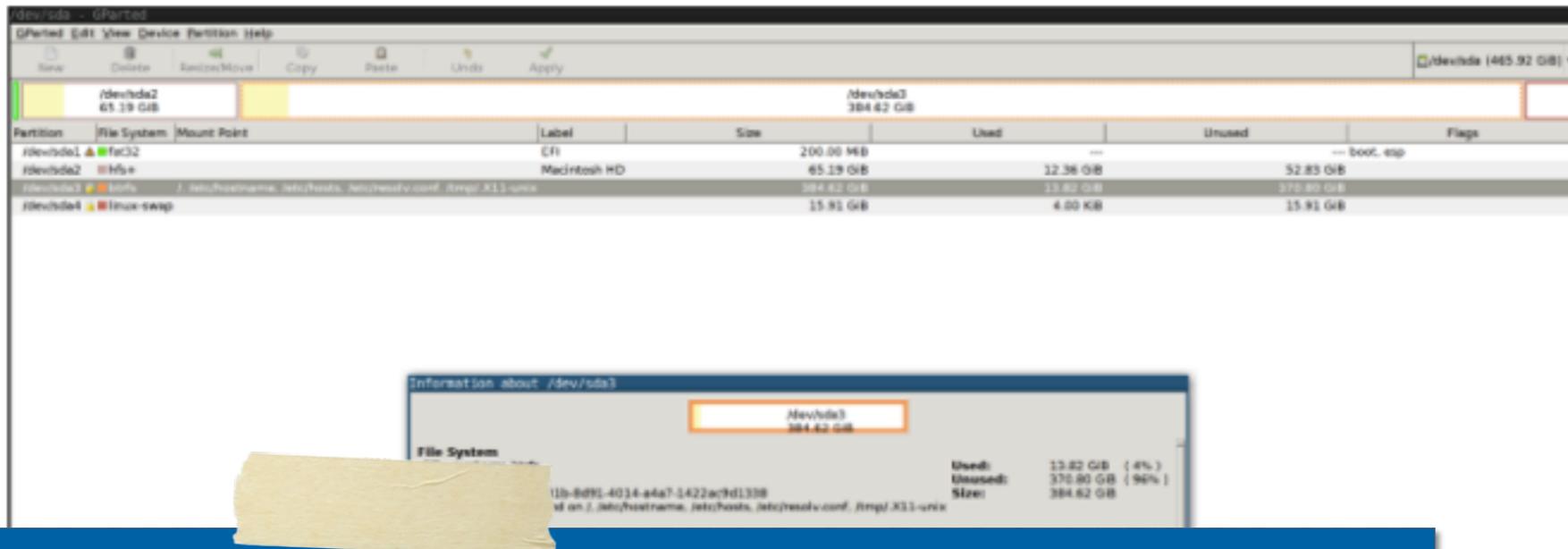
7. Gparted

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
```



Source: Docker Containers on the Desktop

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

Docker - discussion

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

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

(Suffer the compliments, Alex.)

[reply](#)

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

Update: *User Namespaces* available!

"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?

```
[badguy@docker ~]# cd /home/goodguy
bash: cd: /home/goodguy: Permission denied
```

```
[badguy@docker ~]# id badguy && id goodguy
uid=1234(badguy) gid=1234(badguy) groups=1234(badguy),1337(docker)
uid=1000(goodguy) gid=1000(goodguy) groups=1000(goodguy)
```

```
[badguy@docker ~]# docker run -it -v /home:/nfs3home -u 1000 busybox sh
/ $ id
uid=1000 gid=0(root)
/ $ touch /nfs3home/goodguy/badguy_WAS_HERE && exit
```

taming Docker

fine-grained access

wrapper scripts

instead of access to docker-command

Application Container

instead of System Container

*own registry
instead of Docker Hub*

Part IV: **What's new?**

Security!

A Look Back at One Year of Docker Security

By [Diogo Mónica](#) | April 20, 2016

Share this: [!\[\]\(6e52e0e576d5316b311a34170fc34b02_img.jpg\) 25](#) [!\[\]\(2ca0ec41619a742b35f8f015ae6eefa0_img.jpg\) 566](#) [!\[\]\(ef28d680ad565f80f1c1e2997086a541_img.jpg\) 0](#) [!\[\]\(29cbfe8c41a11af2155cacfb4a277a13_img.jpg\) 15](#) [!\[\]\(f1aa7f47d45cc3145969b10fbc6df135_img.jpg\) 101](#) [!\[\]\(dddf983daa41151fe6c4e30bf5fe1669_img.jpg\) 3](#) [!\[\]\(93e23b823774376eada96f150f06279d_img.jpg\) 1](#)

Container Security, docker, Docker security, operational security, security

Security is one of the most important topics in the container ecosystem right now, and over the past year, our team and the community have been hard at work adding new security-focused features and improvements to the Docker platform.

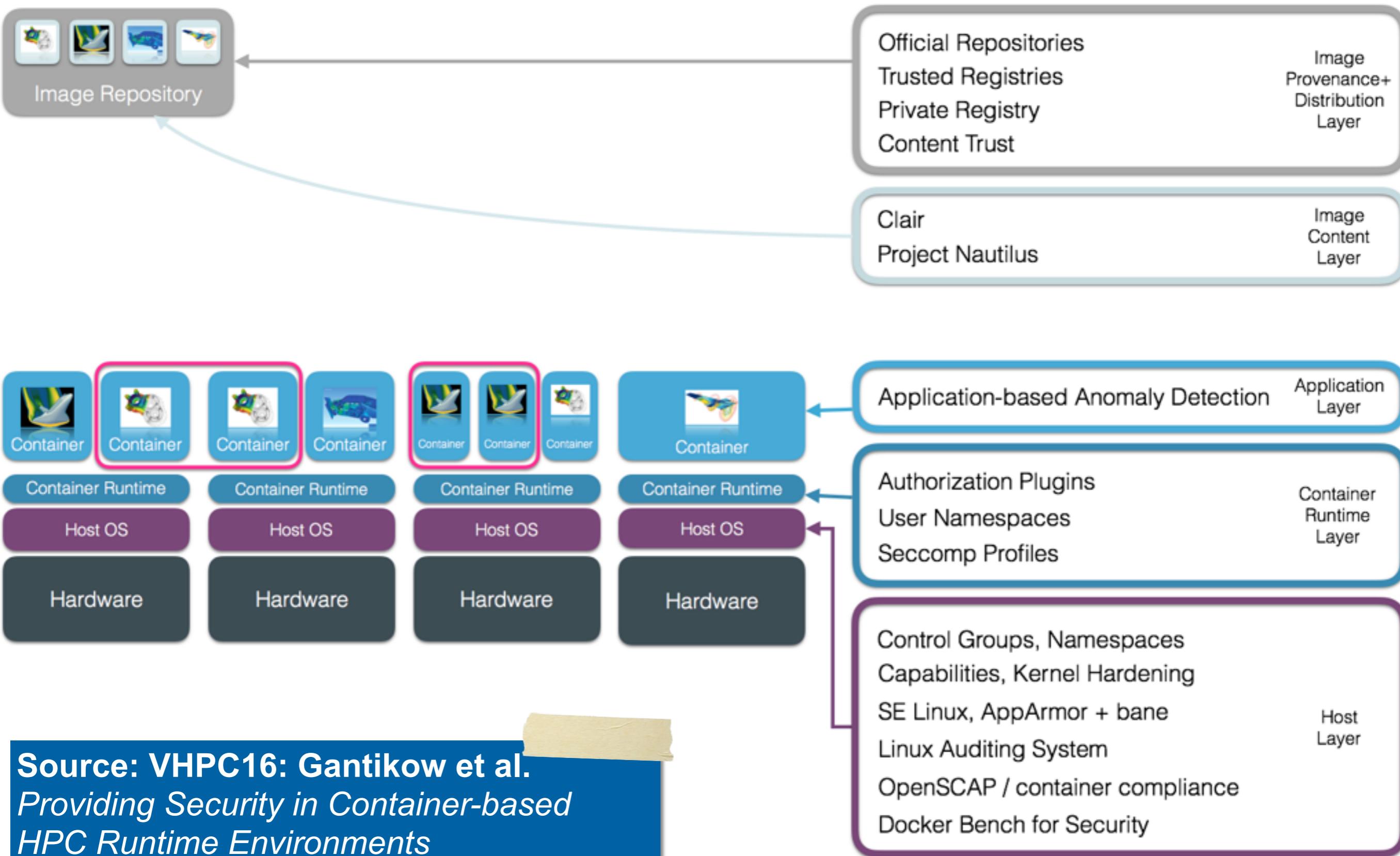
Security should be part of the Platform

As companies transition more and more of their infrastructures to public and private clouds, they have started to realize that security simply can't be bolted on as an afterthought, and instead must be fundamentally built into the platform.

We are incredibly happy that a year after the first [Docker Security White Paper](#) and the first [CIS Benchmark for Docker 1.6](#), there continues to be strong industry validation of our efforts, most recently in the form of a new [Docker 1.11 CIS Benchmark](#) and a [feature evaluation of the Docker Engine](#), as part of NCC Group's whitepaper on hardening Linux containers.

Source: A Look Back at One Year of Docker Security
<https://blog.docker.com/2016/04/docker-security/>

Security on many layers...

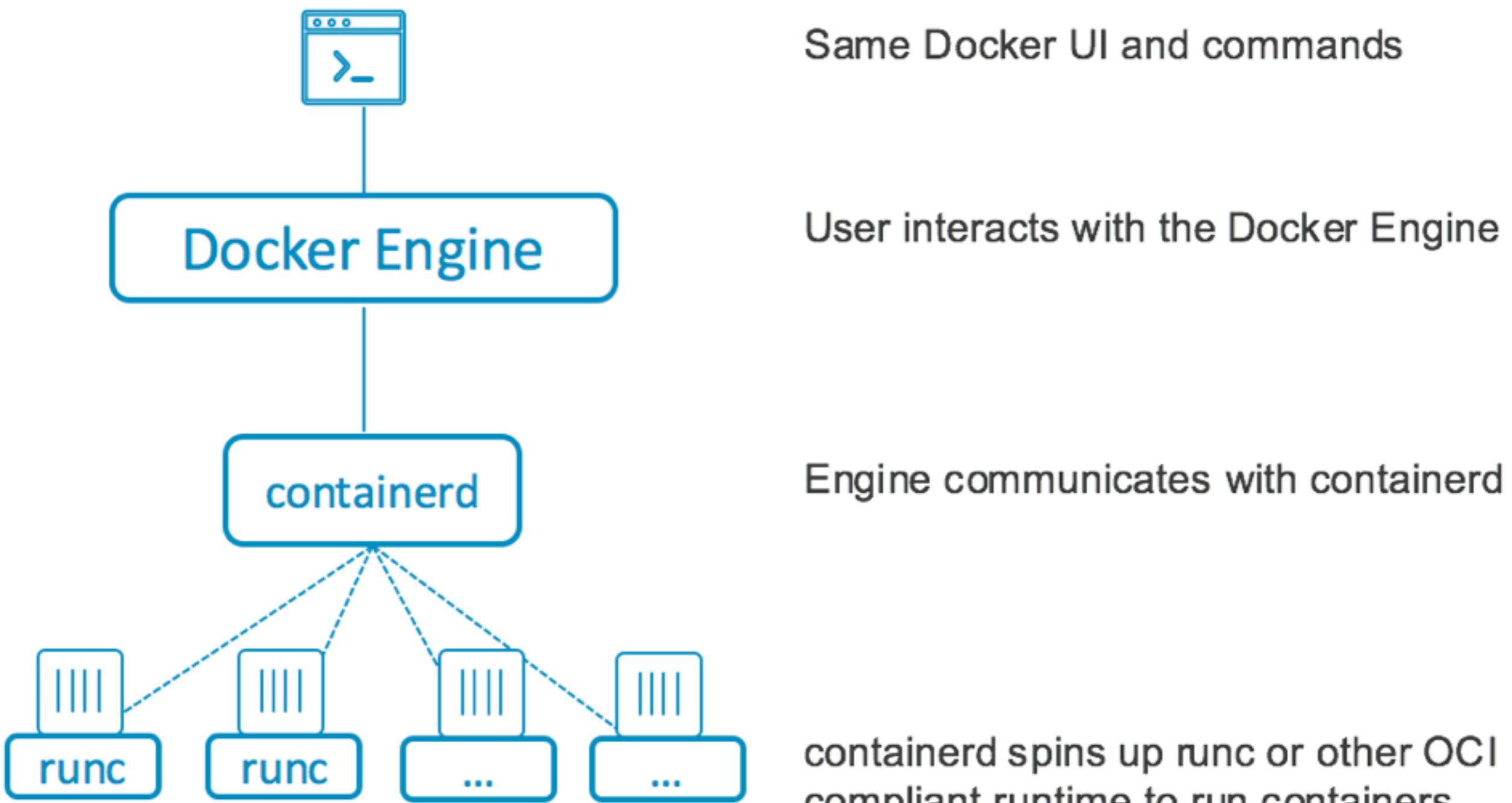


Features

new architecture

runC + containerd

plugable Architecture



Source: Docker 1.11: The first runtime built on containerd and based on OCI technology

<https://blog.docker.com/2016/04/docker-engine-1-11-runc/>

networking

Multi-Host Networking

Multi-Host Networking

You can create a new network with `docker network create`. In this example, we'll create a network called "frontend" and run an nginx container inside it:

```
$ docker network create frontend  
$ docker run -itd --net=frontend --name web nginx
```

You can use networks to separate your applications, or even separate individual components of your application. For example, we could run a web application in a network called "app" and then use the `docker network connect` command so our Nginx container can forward connections to it.

```
$ docker network create app  
$ docker run -itd --name myapp --net=app <my application  
container>  
$ docker network connect app web
```

Now Nginx should be able to connect to your application using the hostname "myapp.app"

Source: Multi-Host Docker Networking is now ready for production
<https://blog.docker.com/2015/11/docker-multi-host-networking-ga/>

DockerCon 2016



SOLD OUT

Join the Waitlist



LiveStream Schedule

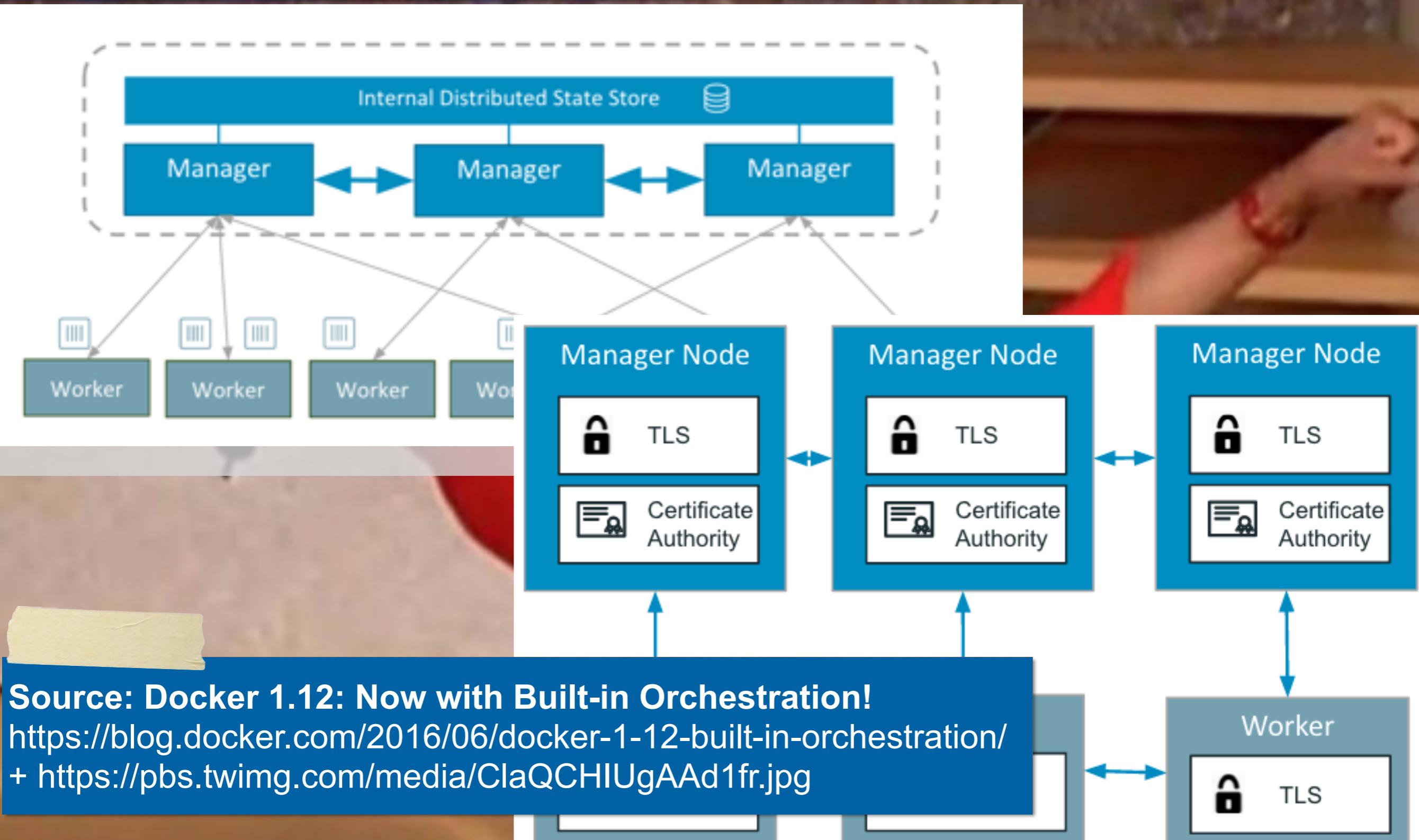
DockerCon 2016 - Quickupdate

Docker Store, AWS&Azure, DAB-files, service command, orchestration

June 21: 4:45pm (PDT)

June 19-21, 2016 | Seattle

YOU GET ORCHESTRATION!



EVERYONE GETS ORCHESTRATION!

Part V: **Getting started**

Additional Material

Installing Docker

Linux

apt-get install docker

or

yum install docker

or

`$ curl -sSL https://get.docker.com/ | sh`

And remember to:
start the docker service ;)
add user to docker groups vs sudo



By [Patrick Chanezon](#)

March 24, 2016



Beta, docker for mac, docker for windows

Share this:

 416

 0

 14

 102

 2.3k

 30

 2

To

celebrate Docker's third birthday, today we start a limited availability [beta program for Docker for Mac and Docker for Windows](#), an integrated, easy-to-deploy environment for building, assembling, and shipping applications from Mac or Windows. Docker for Mac and Windows contain many improvements over Docker Toolbox.

- Faster and more reliable: no more VirtualBox! The Docker engine is running in an Alpine Linux distribution on top of an xhyve Virtual Machine on Mac OS X or on a Hyper-V VM on Windows, and that VM is managed by the Docker application. You don't need `docker-machine` to run Docker for Mac and Windows.
- Tools integration: Docker for Mac is a Mac application and Docker for Windows is a Windows application, including a native user interface and auto-update capability. The Docker tool set comes bundled with it:
[Docker command line](#), [Docker Compose](#), and [Docker Notary command](#)

Source: Docker for Mac and Windows Beta [...]

<https://blog.docker.com/2016/03/docker-for-mac-windows-beta/>

Windows



Docker Subscription

Docker Hub

Docker Trusted Registry

Docker Engine

Docker Kitematic

Docker Toolbox

Docker Registry

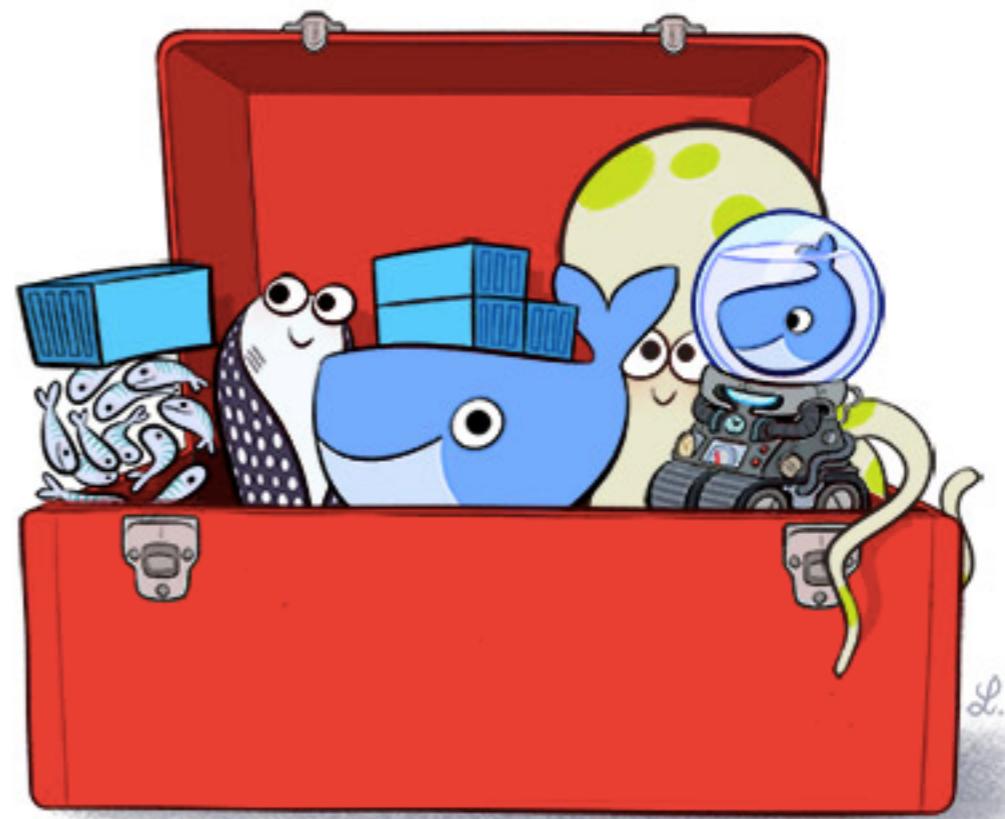
Docker Machine

Docker Swarm

Docker Compose

Docker Toolbox

[Getting Started Guide \(Mac\)](#) | [Getting Started Guide \(Windows\)](#) | [Contribute to Toolbox](#)



[Download \(Mac\)](#)

[Download \(Windows\)](#)

Compatible with Mac OS X 10.8+ and Windows 7+

get Docker Toolbox

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

up... and running!



Docker Toolbox Installer
<https://www.docker.com/toolbox>

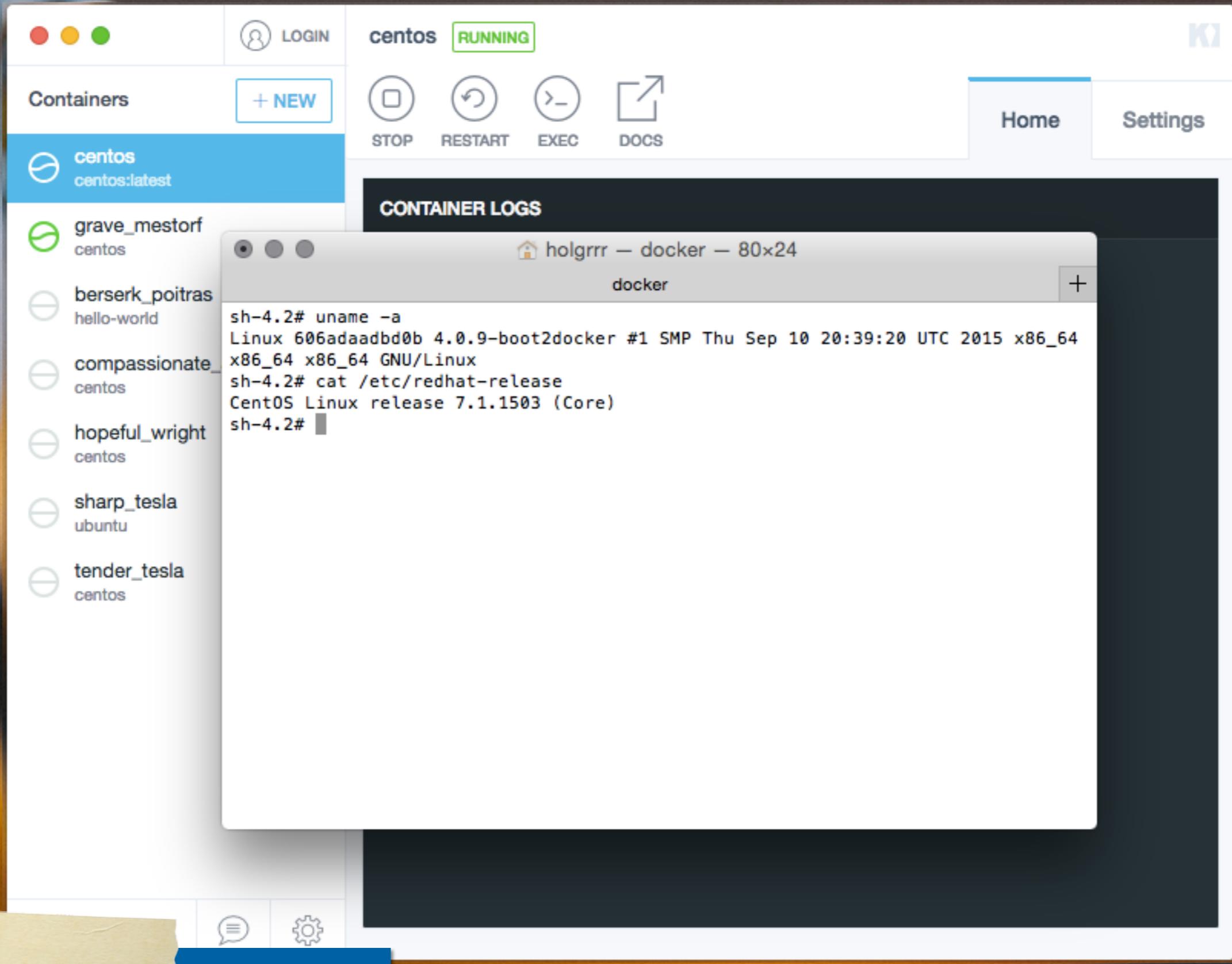
up... and running!

The screenshot shows the Kitematic (Beta) interface. At the top, there are three colored dots (red, yellow, green), a user icon labeled "LOGIN", and a "K" logo. Below the header, there's a search bar with the query "centos". To the left, under "Containers", there's a list of local containers: "grave_mestorf centos", "berserk_poitras hello-world", "compassionate_albatt...", "hopeful_wright centos", "sharp_tesla ubuntu", and "tender_tesla centos". A "+ NEW" button is also present. The main area is titled "Other Repositories" and displays several Docker images from various repositories:

- official centos**: The official build of CentOS. Likes: 1383. CREATE button.
- nathonfowlie centos-jre**: Latest CentOS image with the JRE pre-installed. Likes: 2. CREATE button.
- pdericson centos**: Docker image for CentOS. Likes: 0. CREATE button.
- blacklabelops centos**: Blacklabelops Centos 7.1.503 base image without systemd. Built daily! Likes: 0. CREATE button.
- nimmis java-centos**: This is docker images of CentOS 7 with different versions of java. Likes: 5. CREATE button.
- torusware speedus-centos**: Always updated official CentOS docker image with Torusware Speedus Lite acceleration... Likes: 7. CREATE button.
- layerworx centos**: CentOS container with etcd, etcdctl, confd and consul. Likes: 0. CREATE button.
- jdeathe centos-ssh**: CentOS-6 6.6 x86_64 / EPEL/IUS Repos / OpenSSH / Supervisor. Likes: 0. CREATE button.

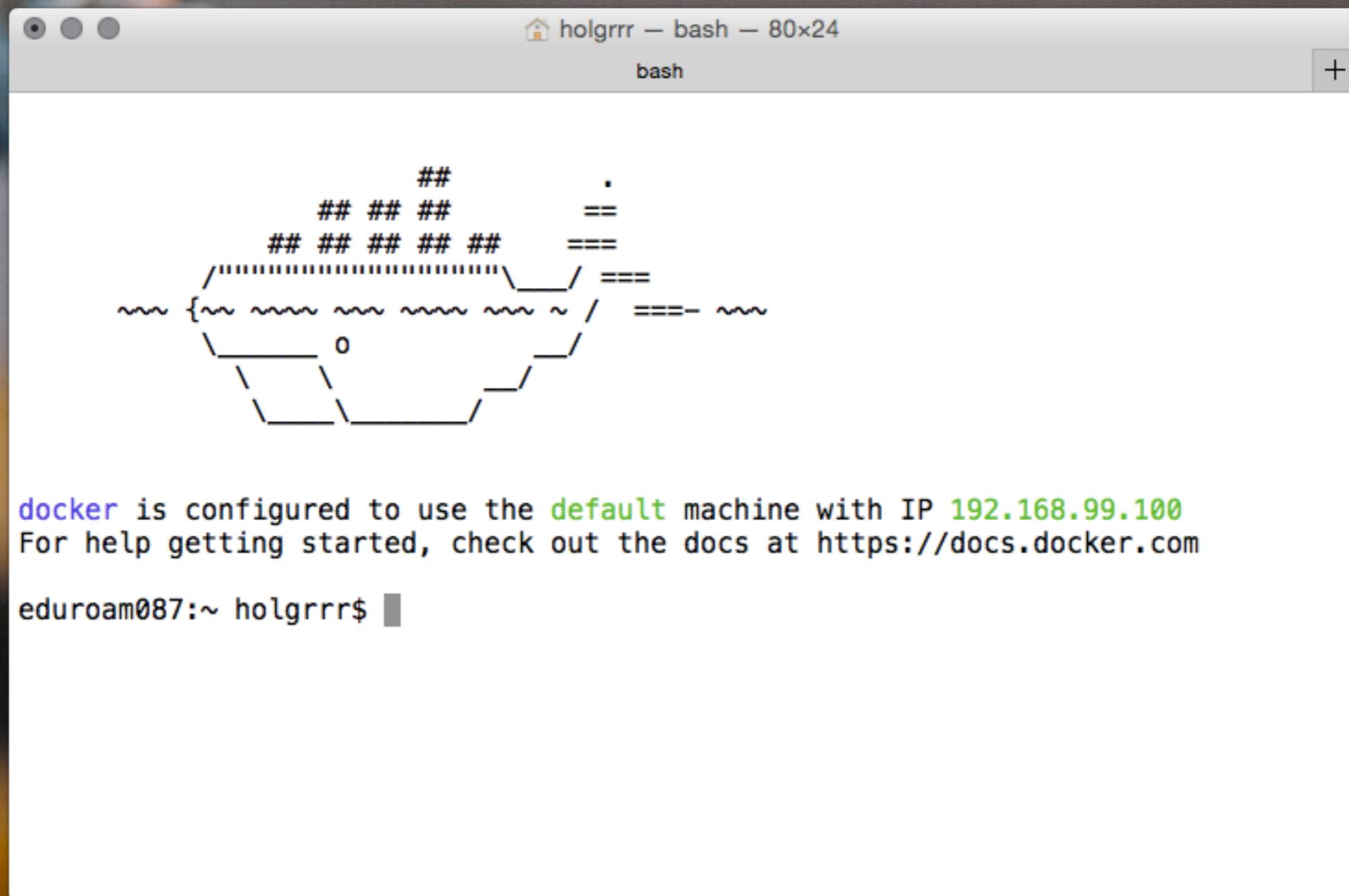
Kitematic (Beta)
included in Toolbox

up... and running!



Kitematic (Beta)
access to CentOS container

up... and running!!



Docker Quickstart Terminal included in Toolbox

up... and running!

Running Docker

```
holgrrr ~ bash - 80x24
bash
\-----o-----/
docker is configured to use the default machine with IP 192.168.99.100
For help getting started, check out the docs at https://docs.docker.com

eduroam189:~ holgrrr$ ps aux | grep -i virt
holgrrr      684  3,1 5,0 2981776 417052 ?? S Sa04pm 27:51.21 /Applications/VirtualBox.app/Contents/MacOS/VBoxHeadless --comment default --startvm a94f9654-4a32-4d9c-861f-812f815cf8c1 --vrde config
holgrrr      687  0,0 0,1 2467204  4640 ?? S Sa04pm  0:02.31 /Applications/VirtualBox.app/Contents/MacOS/VBoxNetDHCP --ip-address 192.168.99.1 --lower-ip 192.168.99.100 --mac-address 08:00:27:B0:16:2D --netmask 255.255.255.0 --network HostInterfaceNetworking-vboxnet3 --trunk-name vboxnet3 --trunk-type netadp --upper-ip 192.168.99.254
holgrrr      667  0,0 0,1 2510244   7904 ?? S Sa04pm  1:25.55 /Applications/VirtualBox.app/Contents/MacOS/VBoxSVC --auto-shutdown
holgrrr      664  0,0 0,1 2451312   4484 ?? S Sa04pm  0:13.96 /Applications/VirtualBox.app/Contents/MacOS/VBoxXPCOMIPCD
holgrrr      5784  0,0 0,0 2432772    644 s005 S+  9:21pm  0:00.00 grep -i virt
eduroam189:~ holgrrr$
```

running VirtualBox VM
in background - required

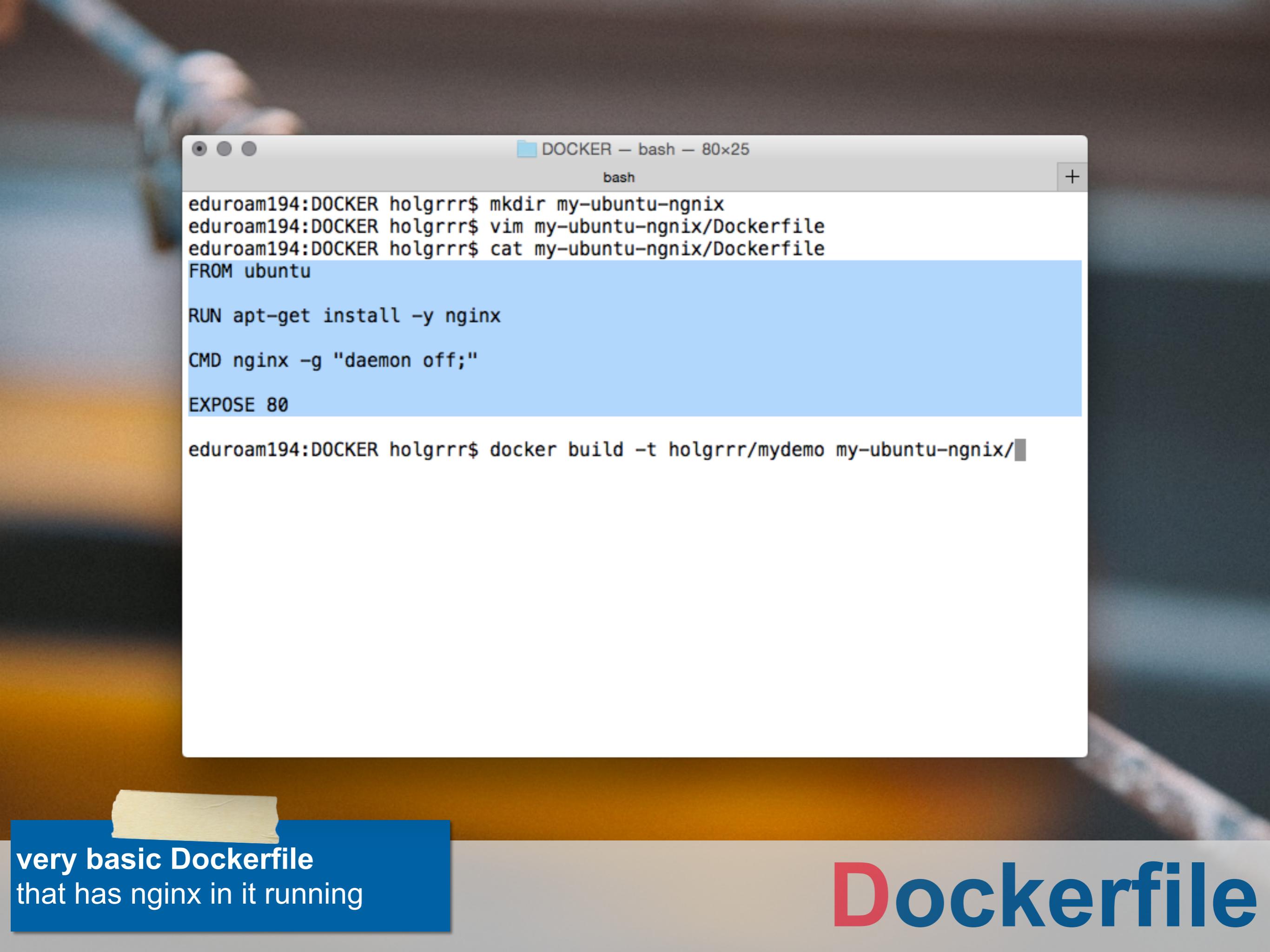
up... and running?

```
holgrrr — @6ea9813bf977:/ — docker — 80x25
@6ea9813bf977:/
eduroam087:~ holgrrr$ uname -a
Darwin eduroam087.uni-tuebingen.de 14.3.0 Darwin Kernel Version 14.3.0: Mon Mar
23 11:59:05 PDT 2015; root:xnu-2782.20.48~5/RELEASE_X86_64 x86_64
eduroam087:~ holgrrr$
eduroam087:~ holgrrr$
eduroam087:~ holgrrr$ docker run -ti centos /bin/bash
[root@6ea9813bf977 /]# uname -a
Linux 6ea9813bf977 4.0.9-boot2docker #1 SMP Thu Sep 10 20:39:20 UTC 2015 x86_64
x86_64 x86_64 GNU/Linux
[root@6ea9813bf977 /]# cat /etc/redhat-release
CentOS Linux release 7.1.1503 (Core)
[root@6ea9813bf977 /]#
```

Docker Quickstart Terminal
run centos container, get bash

up... and running!

Dockerfile 101



```
eduroam194:DOCKER holgrrr$ mkdir my-ubuntu-nginx
eduroam194:DOCKER holgrrr$ vim my-ubuntu-nginx/Dockerfile
eduroam194:DOCKER holgrrr$ cat my-ubuntu-nginx/Dockerfile
FROM ubuntu

RUN apt-get install -y nginx

CMD nginx -g "daemon off;"

EXPOSE 80

eduroam194:DOCKER holgrrr$ docker build -t holgrrr/mydemo my-ubuntu-nginx/
```

very basic Dockerfile
that has nginx in it running

Dockerfile

```
eduroam194:DOCKER holgrrr$ vim my-ubuntu-nginx/Dockerfile
eduroam194:DOCKER holgrrr$ cat my-ubuntu-nginx/Dockerfile
FROM ubuntu

RUN apt-get install -y nginx

CMD nginx -g "daemon off;"

EXPOSE 80

eduroam194:DOCKER holgrrr$ docker build -t holgrrr/mydemo my-ubuntu-nginx/
Sending build context to Docker daemon 2.048 kB
Step 0 : FROM ubuntu
--> 91e54dfb1179
Step 1 : RUN apt-get install -y nginx
--> Using cache
--> 619cb69cc4bc
Step 2 : CMD nginx -g "daemon off;"
--> Using cache
--> 1f832d657d86
Step 3 : EXPOSE 80
--> Using cache
--> 0919ada223a8
Successfully built 0919ada223a8
eduroam194:DOCKER holgrrr$
```

Building a new ubuntu image
including nginx

Dockerfile

```
DOCKER — docker — 80x25
docker
eduroam194:DOCKER holgrrr$ cat my-ubuntu-nginx/Dockerfile
FROM ubuntu

RUN apt-get install -y nginx

CMD nginx -g "daemon off;"

EXPOSE 80

eduroam194:DOCKER holgrrr$ docker build -t holgrrr/mydemo my-ubuntu-nginx/
Sending build context to Docker daemon 2.048 kB
Step 0 : FROM ubuntu
--> 91e54dfb1179
Step 1 : RUN apt-get install -y nginx
--> Using cache
--> 619cb69cc4bc
Step 2 : CMD nginx -g "daemon off;"
--> Using cache
--> 1f832d657d86
Step 3 : EXPOSE 80
--> Using cache
--> 0919ada223a8
Successfully built 0919ada223a8
eduroam194:DOCKER holgrrr$ docker run -P holgrrr/mydemo
```

Building a new image

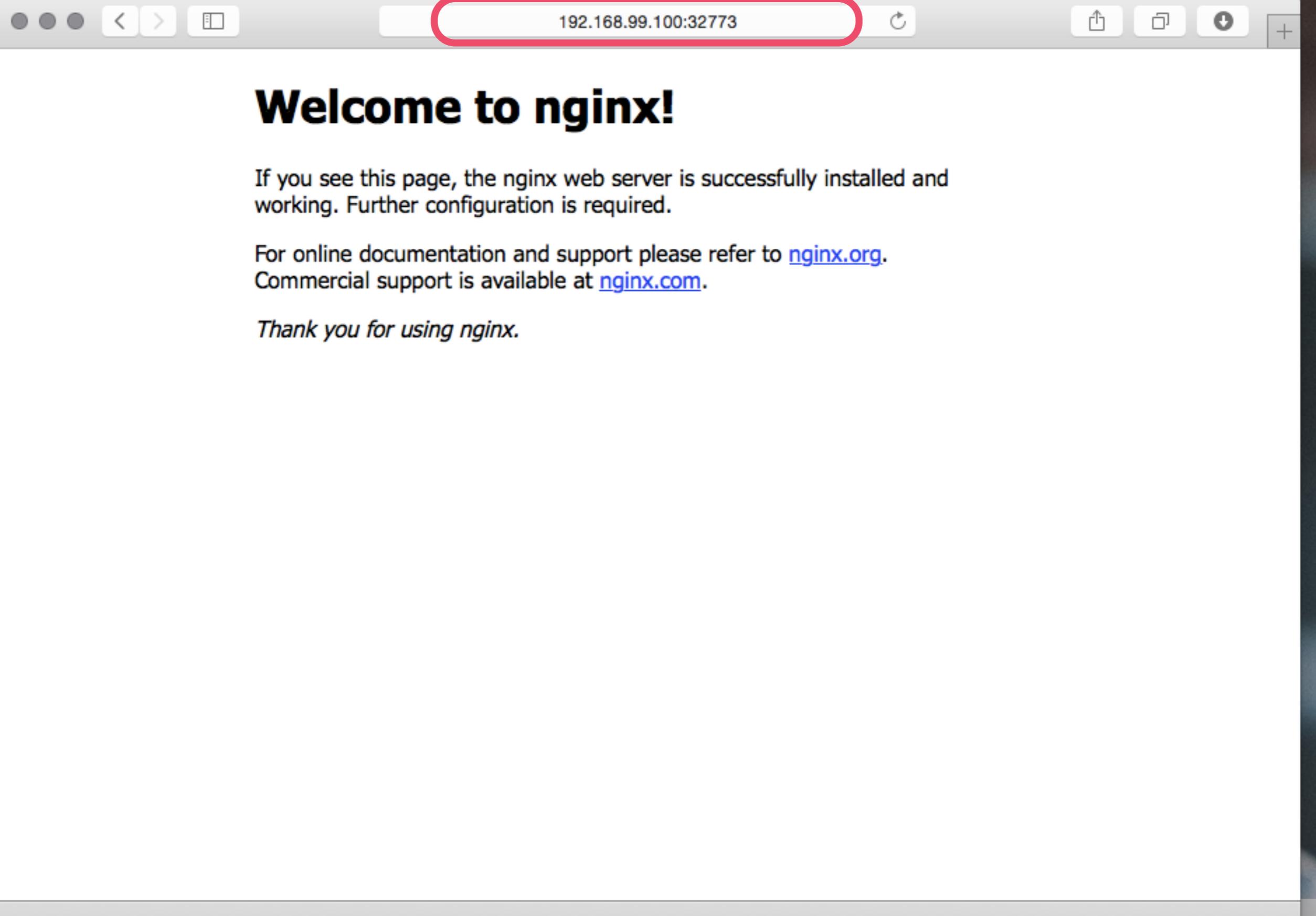
The -P flag tells Docker to map any required network ports container to our host.

Dockerfile

Checking host IP

192.168.99.100 is used

Dockerfile



Dockerfile

Teil V: **Conclusion**

Summary:

Docker has a lot to offer for computing and scientific workflows.

- * easy deployment of applications and dependencies
- * configuration isolation that goes along with it
- * containing legacy applications
- * at almost no performance loss, performing close to bare-metal
- * easy to share it with others + reproducibility
- * possibility to use the same image on all machines -
 - * laptop, workstation, cluster, cloud
- * possibility to increased dynamic utilization of resources, "cluster-sharing"
- * many aspects covered by VMs, but Docker provides easy tools+workflows

But still:

- * integration in queuing systems „improvable“
- * cloud orchestration vs HPC
- * ISV support yet (mostly) lacking
- * higher organizational costs (security, ...) - but there is no such thing as free lunch ;)

Source:

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

Proof of Concept

Motivation & Forschungsziel



„[...] Design und Integration einer umfassenden Lösung für BatchJob-basiertes, paralleles Technical Computing über Interconnect-basiertes SM–MIMD in HPC Clustersystemen, durch den Einsatz verteilter Linux-Container ...“



Bachelorthesis
im Studiengang
Computer Networking Bachelor

Container-basiertes
High Performance Computing
Proof of Concept

Referent : Prof. Dr. Christoph Reich
Hochschule Furtwangen University

Koreferent : Michael Heinrich
science + computing AG, Tübingen

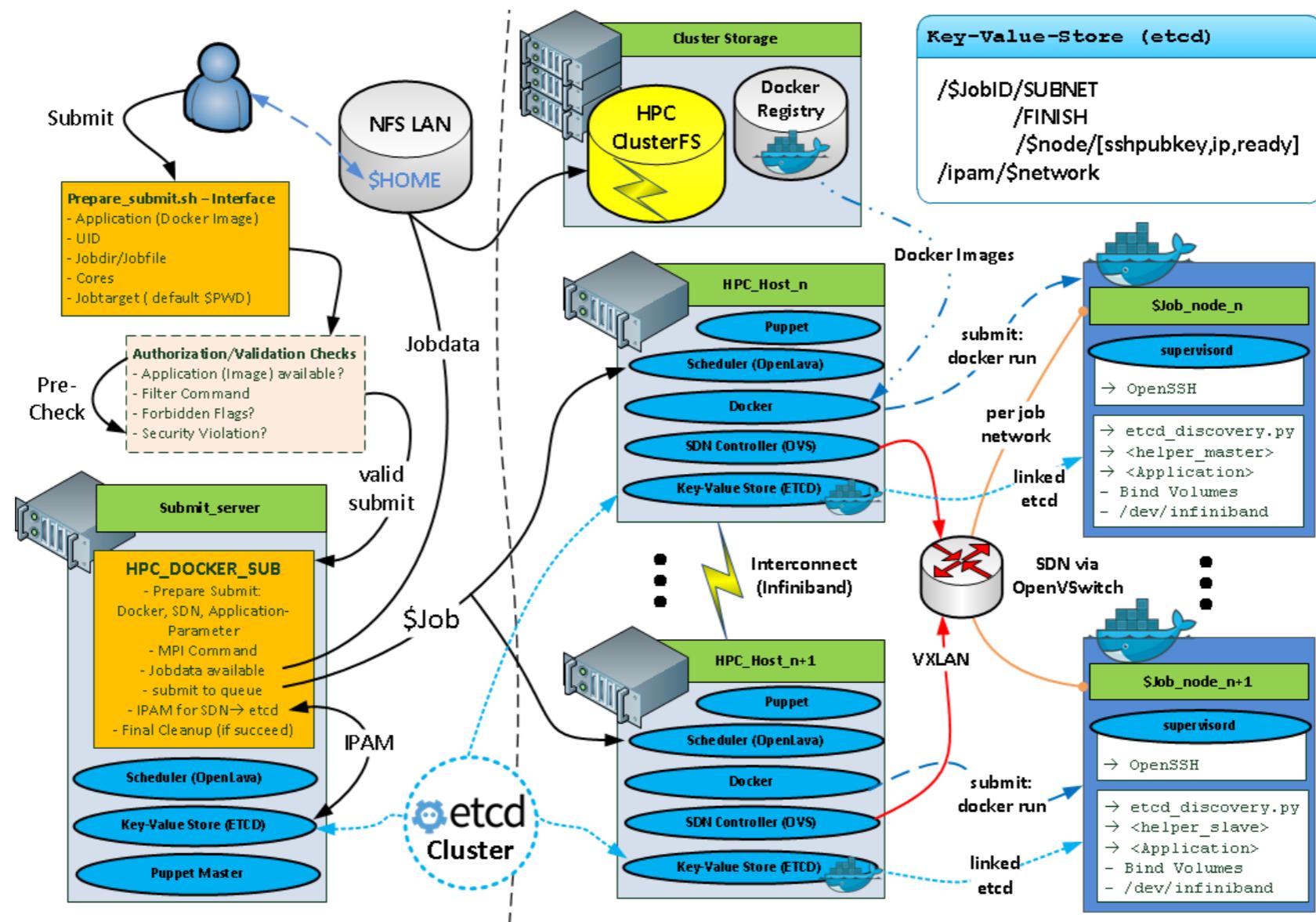
Vorgelegt am : 29.02.2016

Vorgelegt von : Sebastian Klingberg

Proof of Concept Integration of Docker into „regular“ CAE-HPC Environment
* featuring OpenLava + OpenFoam

Proof of Concept

Fertiges Konzept



Overall Conclusion

Virtualization 2.0? No, something different!

Pretty good for: Portable Deployment,
Sharing, Re-use of components

better performance for computing than
classic virtualization

***Security* has to be ensured**
***getting started* is easy**

One more thing...

Containers might actually improve security...

Startseite Über uns

Twitter durchsuchen



Hast Du einen Account? Anmelden

```
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
```



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 :)



60

87

...

Tester, Developer,
Trainer, Building...

60

87



Source

https://twitter.com/Ben_Hall/status/728596633978572801

root exploit

```
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$
```

Source

https://twitter.com/Ben_Hall/status/728596633978572801

non-root exploit



science + computing

| an atos company



Thanks a lot for your attention!

Holger Gantikow

science + computing ag
www.science-computing.de

Telefon: 07071 9457 - 503
E-Mail: h.gantikow@science-computing.de

Questions?

Answers!





science + computing

| an atos company



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