Kubernetes 101

Oleg Fiksel | Sean Mitchell | Michael Siebertz

Email: oleg@fiksel.info | sean@srm.im | mail@michaelsiebertz.de

Matrix: @oleg:fiksel.info | @sean:fiksel.info | @captain.vsan:matrix.org

2018-11-04 OpenRheinRuhr 2018

MOTIVATION

Our motivation to do this talk

MOTIVATION

Our motivation to do this talk

Provide an overview of container architechture on the example of Docker and Kubernetes.

AGENDA

AGENDA

- Docker Fundamentals

AGENDA

- Docker Fundamentals
- Kubernetes from 10.000 feet

AGENDA

- Docker Fundamentals
- Kubernetes from 10.000 feet
- Live Demo

OLEG FIKSEL (DOCKER FUNDAMENTALS)

[1] Accenture Interactive
[2] DevSecOps Workshop

OLEG FIKSEL (DOCKER FUNDAMENTALS)

About me:

- DevOps Engineer @ Accenture Interactive [1] (DevOps Chapter)
- **Main topics**
  - Big fan of CI (especially GitLab) and a Gentoo user
  - Learning Golang and like to automate eveything
  - Continuous Security (DevSecOps) [2]

[1] Accenture Interactive
[2] DevSecOps Workshop

SEAN MITCHELL (KUBERNETES FROM 10.000 FEET)

[1] Grandcentrix

SEAN MITCHELL (KUBERNETES FROM 10.000 FEET)

About me:

- Site Reliability Engineer (SRE) @ Grandcentrix [1]
- **Main topics**
  - Support developers building and deploying projects
  - Ensure customers' cloud infrastructure runs
  - Home Automation geek

[1] Grandcentrix

MICHAEL SIEBERTZ (LIVE DEMO)

[1] HRS

MICHAEL SIEBERTZ (LIVE DEMO)

About me:

- DevOps Engineer @ HRS [1]
- **Main topics**
  - Linux-User for long time
  - Trying to automate as much as possible
  - Part of Central DevOps Team

[1] HRS

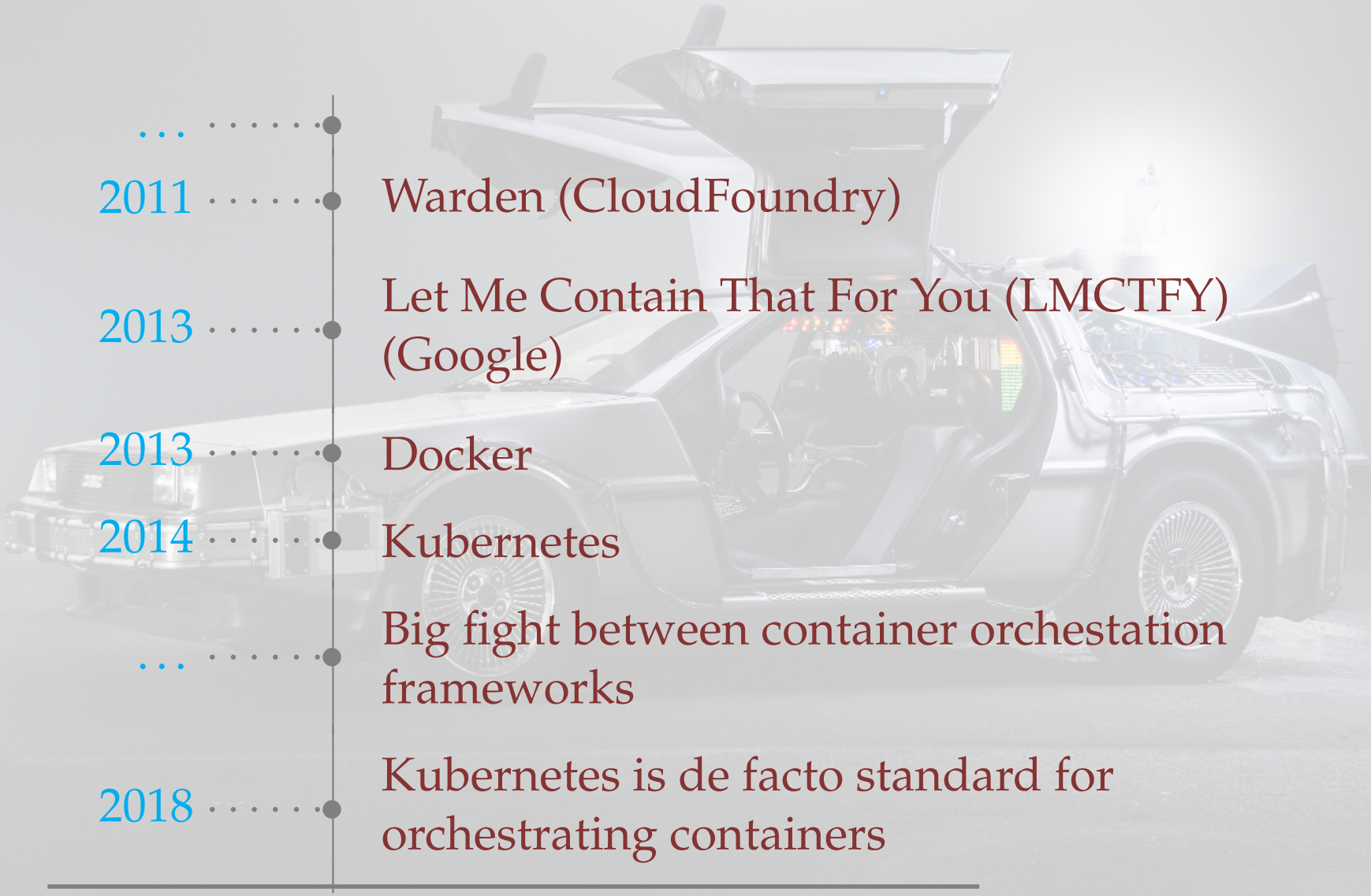
Container fundamentals

A BIT OF HISTORY

A BIT OF HISTORY 1/2

- 1979 — chroot
- 2000 — FreeBSD Jails
- 2001 — Linux VServer
- 2004 — Solaris Containers
- 2005 — OpenVZ
- 2006 — Process Containers (Google)
- 2008 — LXC
- ... —

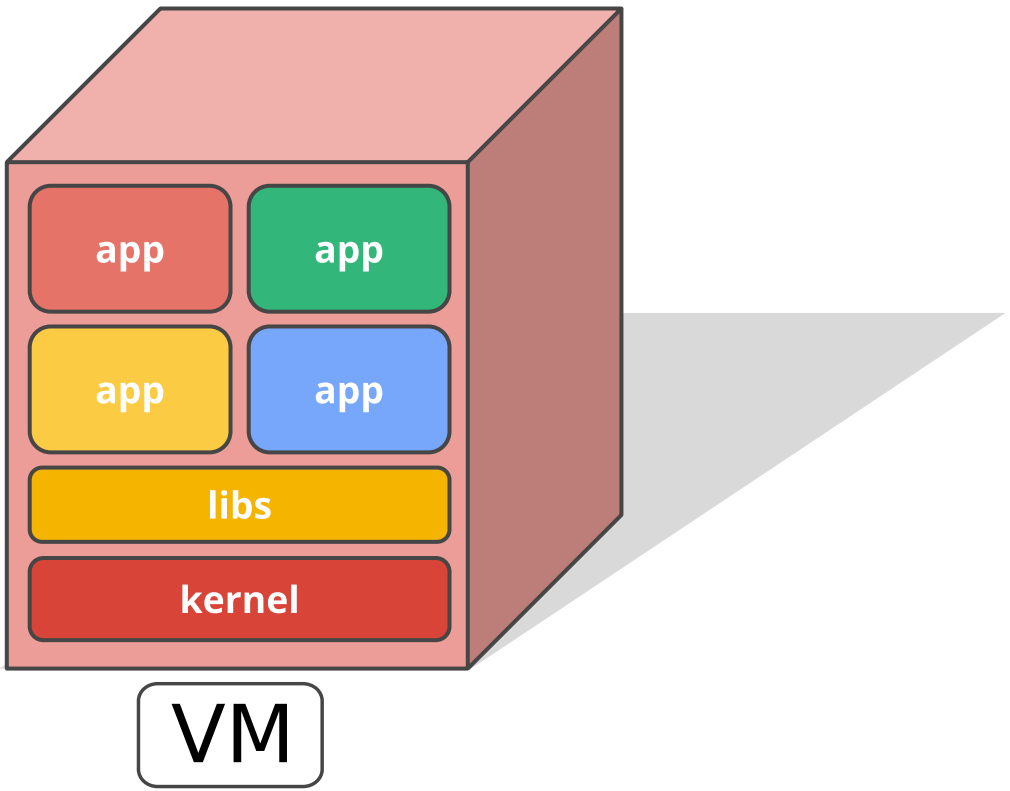
A BIT OF HISTORY 2/2



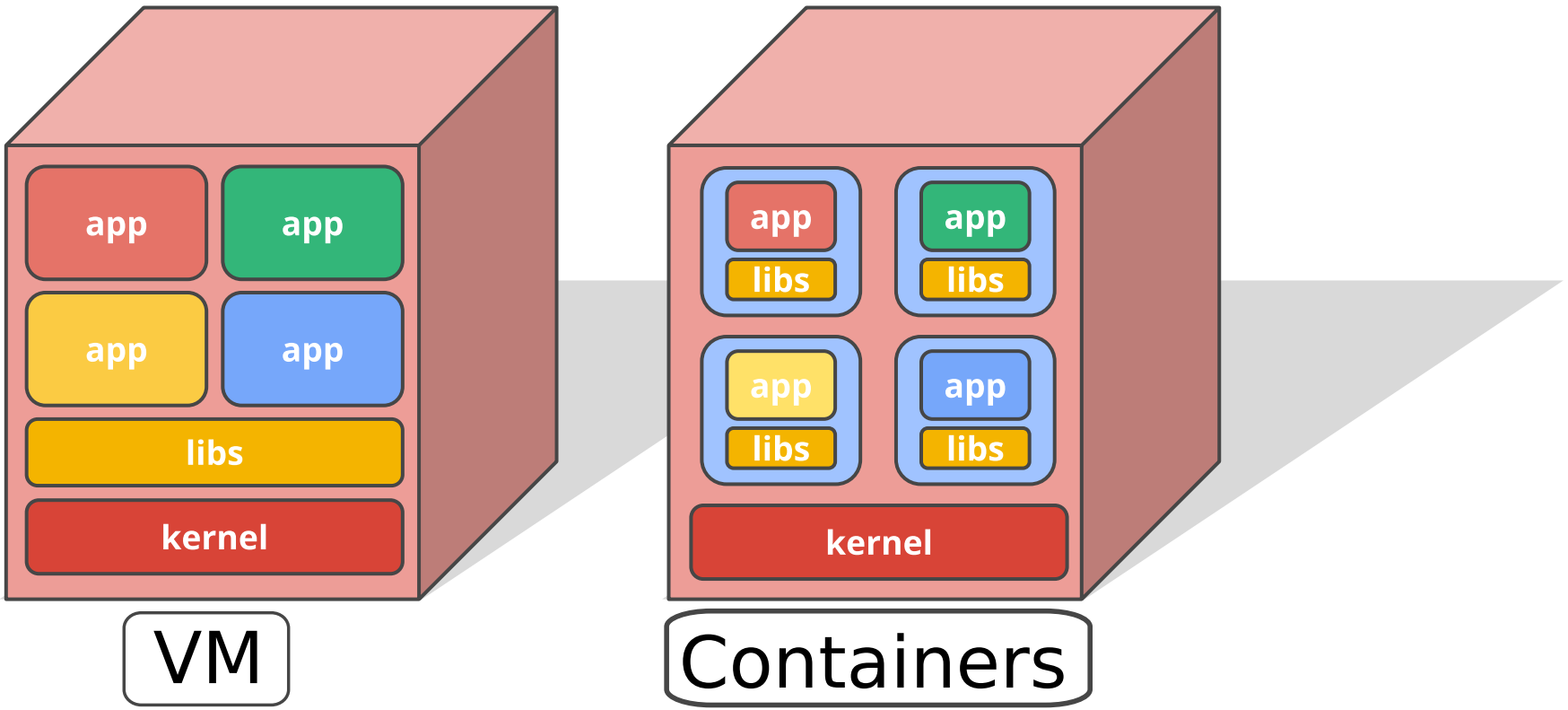
- ...
- 2011 — Warden (CloudFoundry)
- 2013 — Let Me Contain That For You (LMCTFY) (Google)
- 2013 — Docker
- 2014 — Kubernetes
- ... — Big fight between container orchestation frameworks
- 2018 — Kubernetes is de facto standard for orchestrating containers

VMs and Containers

VMs and Containers



VMs and Containers



VMs and Containers

VMs and Containers

- Storage

VMs and Containers

- Storage
  - VM: Binary Image

VMs and Containers

- Storage
  - VM: Binary Image
  - Container: Layered plain FS

VMs and Containers

- Storage
  - VM: Binary Image
  - Container: Layered plain FS
- Emulation

VMs and Containers

- Storage
  - VM: Binary Image
  - Container: Layered plain FS
- Emulation
  - VM: BIOS, Hardware, etc.

VMs and Containers

- Storage
  - VM: Binary Image
  - Container: Layered plain FS
- Emulation
  - VM: BIOS, Hardware, etc.
  - Container: No emulation, just separation

VMs and Containers

- Storage
  - VM: Binary Image
  - Container: Layered plain FS
- Emulation
  - VM: BIOS, Hardware, etc.
  - Container: No emulation, just separation
- Networking

VMs and Containers

- Storage
  - VM: Binary Image
  - Container: Layered plain FS
- Emulation
  - VM: BIOS, Hardware, etc.
  - Container: No emulation, just separation
- Networking
  - VM: NIC emulation

VMs and Containers

- Storage
  - VM: Binary Image
  - Container: Layered plain FS
- Emulation
  - VM: BIOS, Hardware, etc.
  - Container: No emulation, just separation
- Networking
  - VM: NIC emulation
  - Container: Uses network stack of the host

VMs and Containers

- Storage
  - VM: Binary Image
  - Container: Layered plain FS
- Emulation
  - VM: BIOS, Hardware, etc.
  - Container: No emulation, just separation
- Networking
  - VM: NIC emulation
  - Container: Uses network stack of the host
- Runtime

VMs and Containers

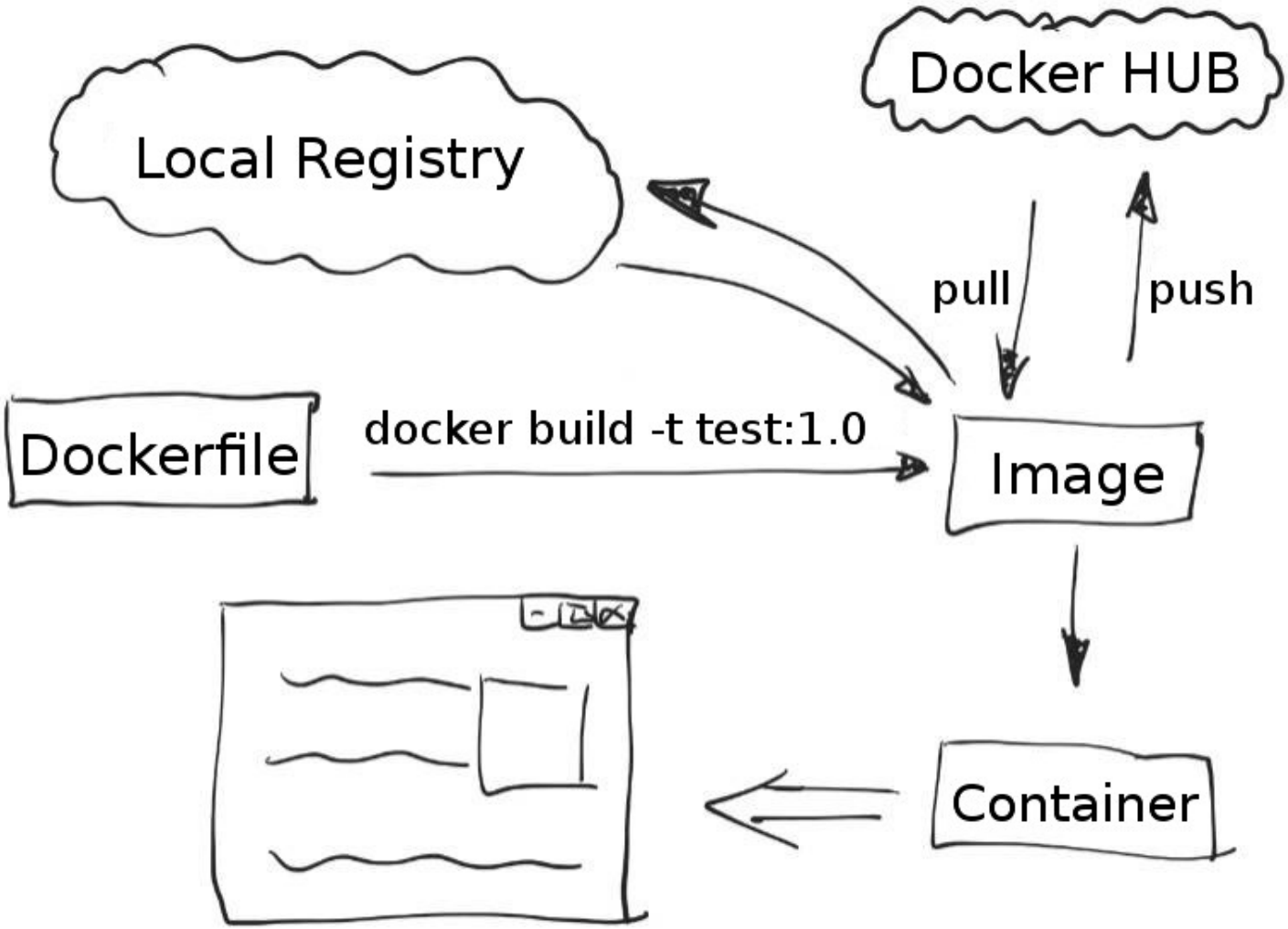
- Storage
  - VM: Binary Image
  - Container: Layered plain FS
- Emulation
  - VM: BIOS, Hardware, etc.
  - Container: No emulation, just separation
- Networking
  - VM: NIC emulation
  - Container: Uses network stack of the host
- Runtime
  - VM: stateful

VMs and Containers

- Storage
  - VM: Binary Image
  - Container: Layered plain FS
- Emulation
  - VM: BIOS, Hardware, etc.
  - Container: No emulation, just separation
- Networking
  - VM: NIC emulation
  - Container: Uses network stack of the host
- Runtime
  - VM: stateful
  - Container: stateless

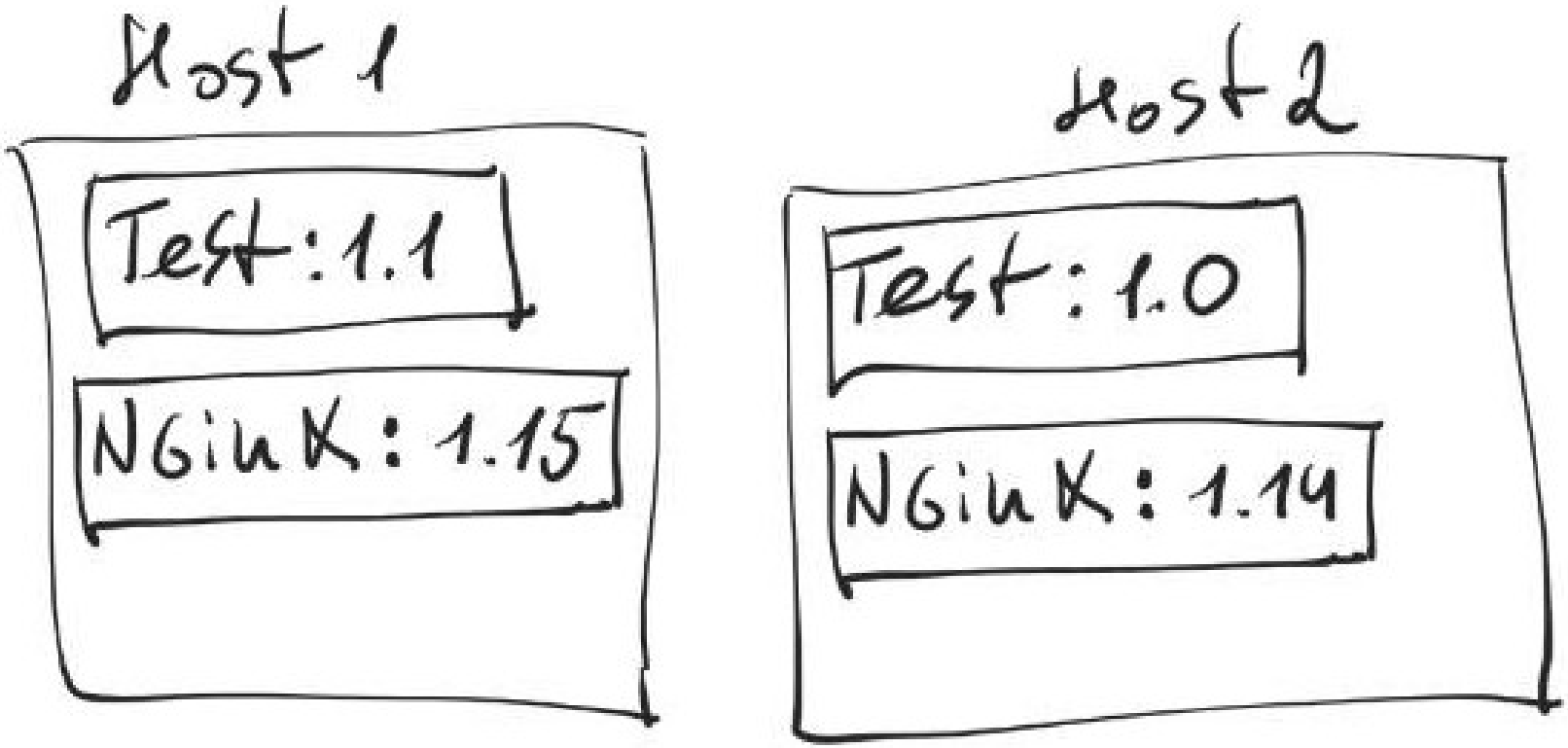
CONTAINER LIFECYCLE - OVERVIEW

CONTAINER LIFECYCLE - OVERVIEW



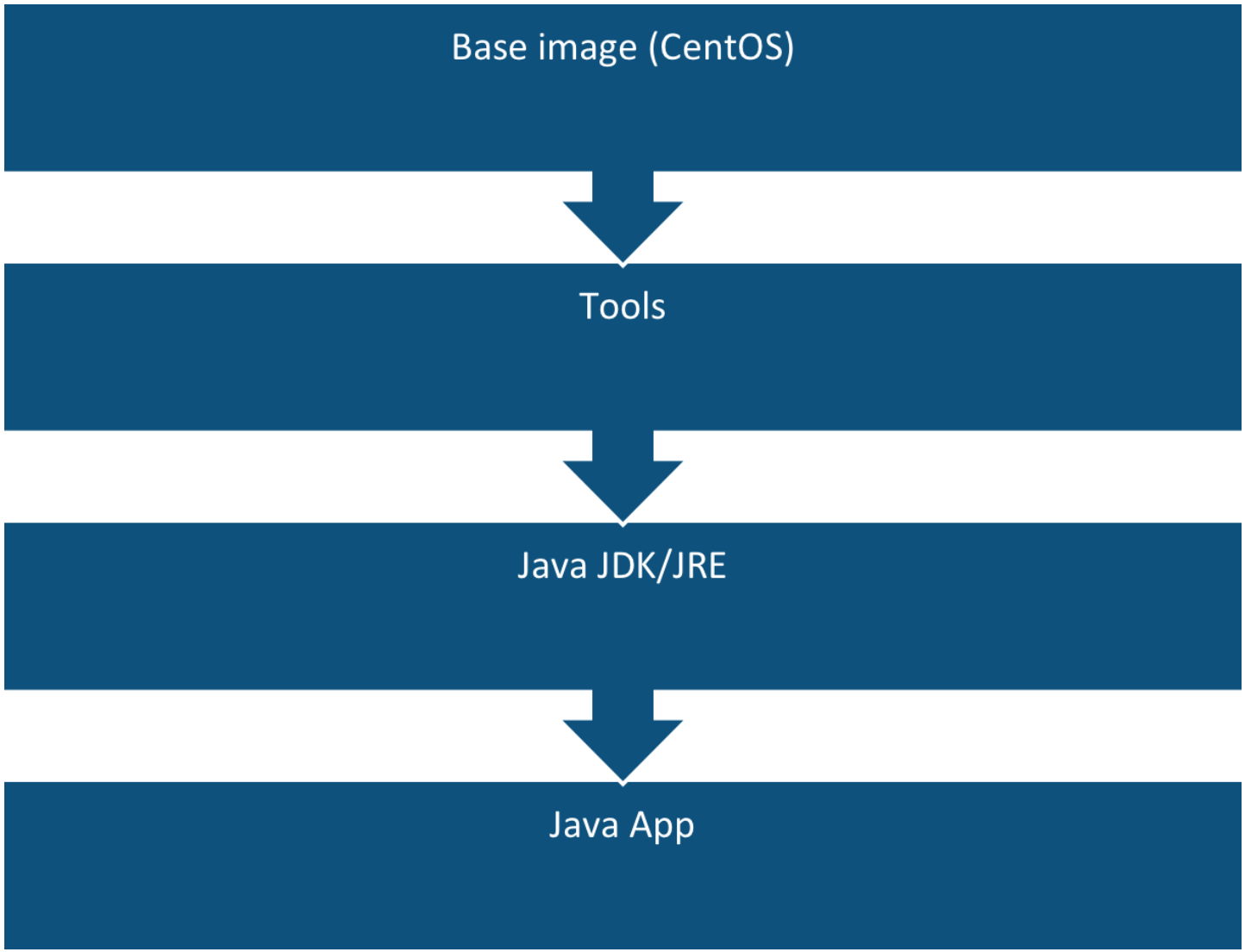
CONTAINER LIFECYCLE - IMAGE VERSIONS

CONTAINER LIFECYCLE - IMAGE VERSIONS

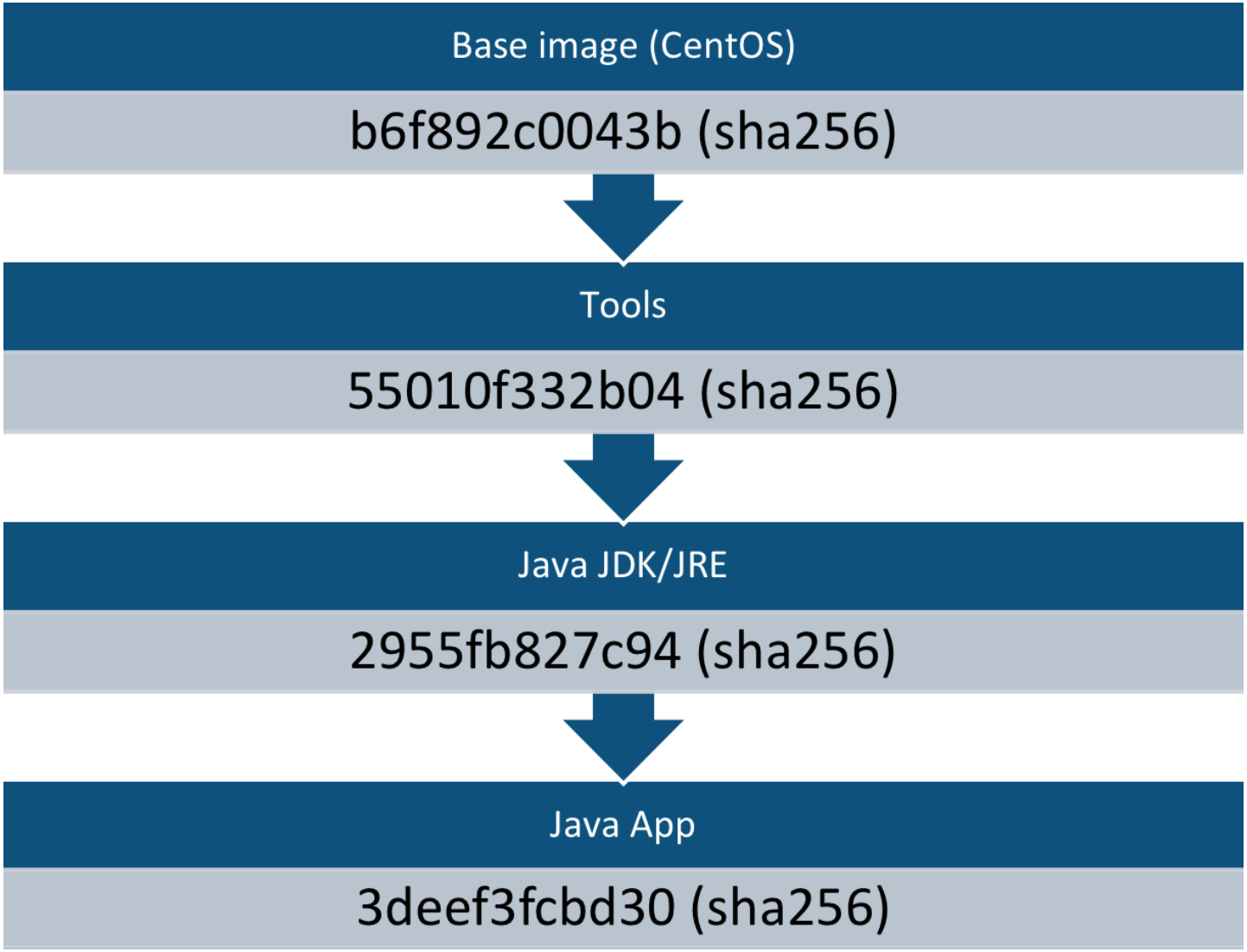


LAYERED FILE SYSTEM

LAYERED FILE SYSTEM



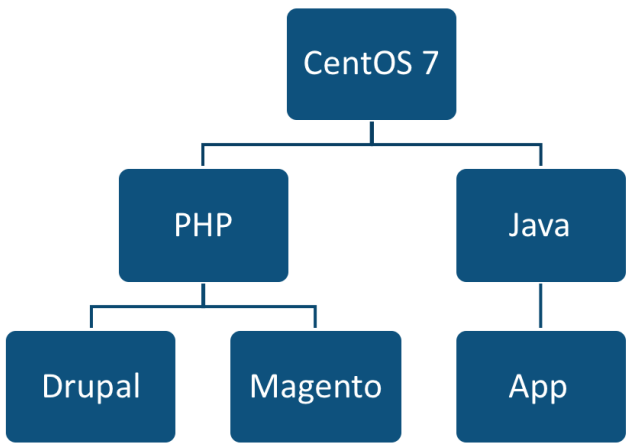
LAYERED FILE SYSTEM



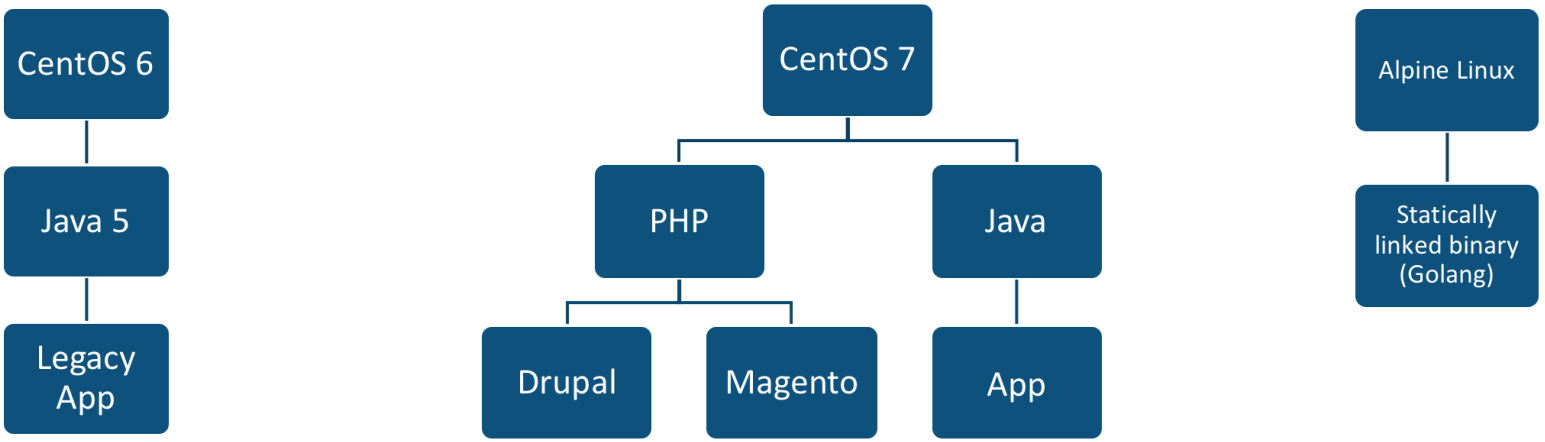
LAYERED FILE SYSTEM - IMAGE DEPENDENCIES

LAYERED FILE SYSTEM - IMAGE DEPENDENCIES

LAYERED FILE SYSTEM - IMAGE DEPENDENCIES



LAYERED FILE SYSTEM - IMAGE DEPENDENCIES



LAYERED FILE SYSTEM - BACKENDS

LAYERED FILE SYSTEM - BACKENDS

- aufs (legacy)
- overlayfs (legacy)
- devicemapper
- overlayfs2
- btrfs

PERSISTENT STORAGE (VOLUMES)

PERSISTENT STORAGE (VOLUMES)

- Containers start without state

PERSISTENT STORAGE (VOLUMES)

- Containers start without state
- State is provided via mounts

PERSISTENT STORAGE (VOLUMES)

- Containers start without state
- State is provided via mounts
  - Binds (local FS)

PERSISTENT STORAGE (VOLUMES)

- Containers start without state
- State is provided via mounts
  - Binds (local FS)
  - Docker volumes

PERSISTENT STORAGE (VOLUMES)

- Containers start without state
- State is provided via mounts
  - Binds (local FS)
  - Docker volumes
  - Docker Storage plugins

SEPARATION AND RESOURCE LIMITING

SEPARATION AND RESOURCE LIMITING

- cgroups

SEPARATION AND RESOURCE LIMITING

- cgroups
  - Linux kernel feature

SEPARATION AND RESOURCE LIMITING

- cgroups
  - Linux kernel feature
  - Limiting: CPU, memory, disk I/O, network

SEPARATION AND RESOURCE LIMITING

- cgroups
  - Linux kernel feature
  - Limiting: CPU, memory, disk I/O, network
- namespaces

SEPARATION AND RESOURCE LIMITING

- cgroups
  - Linux kernel feature
  - Limiting: CPU, memory, disk I/O, network
- namespaces
  - Linux kernel feature

SEPARATION AND RESOURCE LIMITING

- cgroups
  - Linux kernel feature
  - Limiting: CPU, memory, disk I/O, network
- namespaces
  - Linux kernel feature
  - process isolation
    Each container "thinks" it's the only one in the system
    (*ps uax*)

SUMMARY

SUMMARY

- The is no question "Use container or not?".
  The question is "Where to start?"

SUMMARY

- The is no question "Use container or not?".
  The question is "Where to start?"
- Docker is the standard de facto (for now) for running containers

SUMMARY

- The is no question "Use container or not?".
  The question is "Where to start?"
- Docker is the standard de facto (for now) for running containers
- Docker alone is not enough to run containers big time

SUMMARY

- The is no question "Use container or not?".
  The question is "Where to start?"
- Docker is the standard de facto (for now) for running containers
- Docker alone is not enough to run containers big time
- Next up: Container Orchestration (Kubernetes)

Q & A

Thanks!

Oleg Fiksel | Sean Mitchell | Michael Siebertz

Email: oleg@fiksel.info | sean@srm.im | mail@michaelsiebertz.de

Matrix: @oleg:fiksel.info | @sean:fiksel.info | @captain.vsan:matrix.org