Docker – Container in OpenStack

DOST 2015

24. Juni 2015

Christian Baumann
Linux Consultant
B1 Systems GmbH
baumann@b1-systems.de

Einleitung – Docker, OpenStack

Docker?

OpenStack?

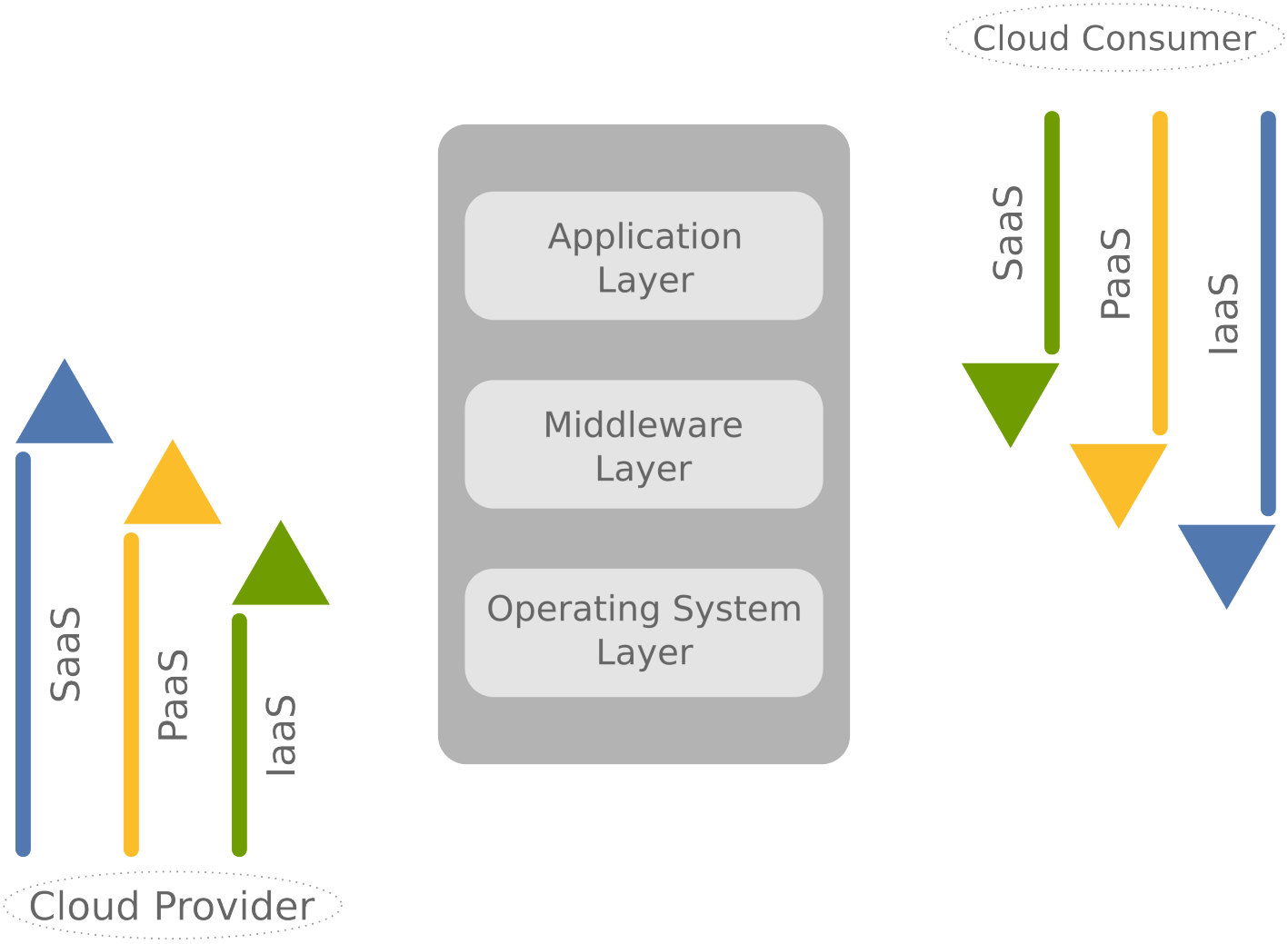
Typen von Virtualisierung

- OS-Virtualisierung
- Emulation
- Hypervisor-Virtualisierung
  - Typ 1
  - Typ 2
- Paravirtualisierung

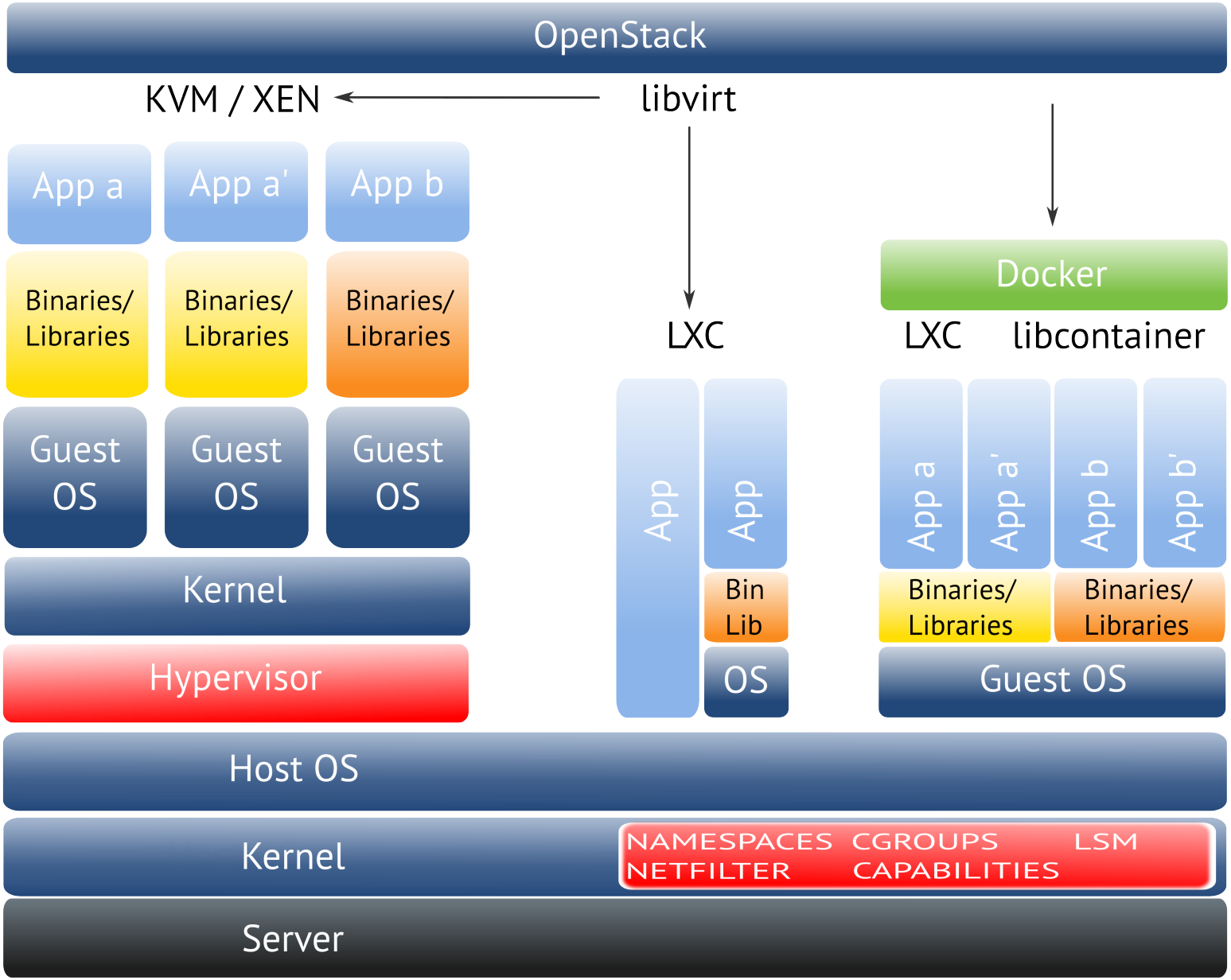
Virtualisierung – Timeline

- 1979 Unix chroot Syscall
- 1990s Xenoserver Research Cambridge
- 1999 VMware Workstation 1.0
- 2001 VMware ESX 1.0
- 2002 Xen OpenSource
- 2003 Xen 1.0
- 2006 Amazon EC2
- 2007 KVM im Linux Kernel
- 2008 Rackspace Cloud
- 2008 LXC in Linux Kernel
- 2013 Docker

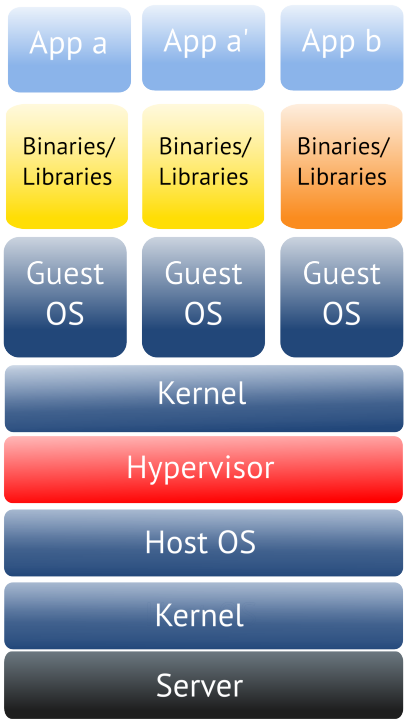
Service-Pyramide



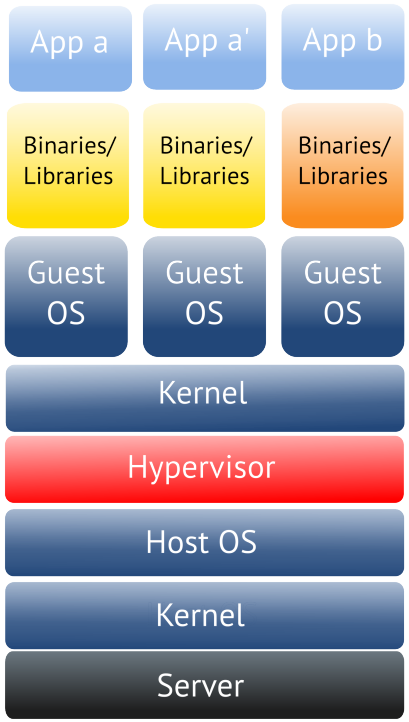
The Big Picture


OpenStack
KVM / XEN
libvirt
App a
App a'
App b
Binaries/ Libraries
Binaries/ Libraries
Binaries/ Libraries
Guest OS
Guest OS
Guest OS
Kernel
Hypervisor
LXC
App
App
Bin Lib
OS
Docker
LXC
libcontainer
App a
App a'
App b
App b'
Binaries/ Libraries
Binaries/ Libraries
Guest OS
Host OS
Kernel
NAMESPACES
CGROUPS
LSM
NETFILTER
CAPABILITIES
Server

Virtualisierung – Eigenschaften



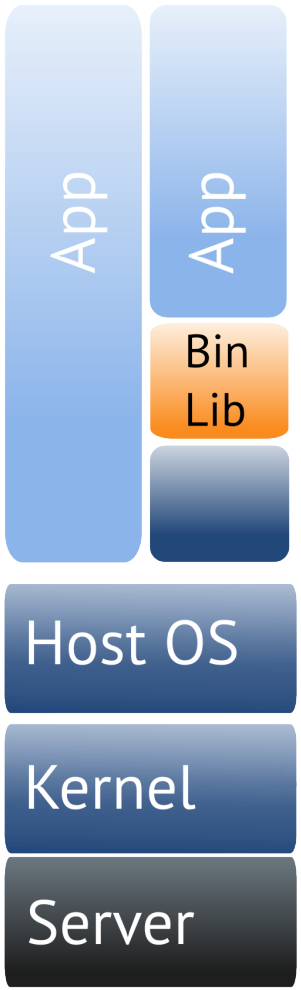
- maximale Isolation
- vollwertige Hardware-Umgebung
- Bereitstellungszeit: 5-30 Minuten
- Größe: GBs
- Performance-Overhead durch Hypervisor
- Management-Overhead



Definition einer VM – SLES12.xml

```
<domain type='kvm'>
  <name>SLES12</name>
  <uuid>1b9661f5....</uuid>
  <title>SLES12_PROD_VM</title>
  <memory unit='KiB'>1048576</memory>
  <currentMemory'>1048576</currentMemory>
    <type arch='x86_64'>hvm</type>
    <boot dev='hd'/>
...
    <disk type='file' device='disk'>
      <source file='SLES12_PROD.qcow2'/>
...
    <interface type='network'>
    <mac address='52:54:00:bc:d7:59'/>
...
    <timer name='rtc' tickpolicy='catchup'/>
    <timer name='pit' tickpolicy='delay'/>
    <timer name='hpet' present='no'/>
```

LXC – *LinuX Containers*

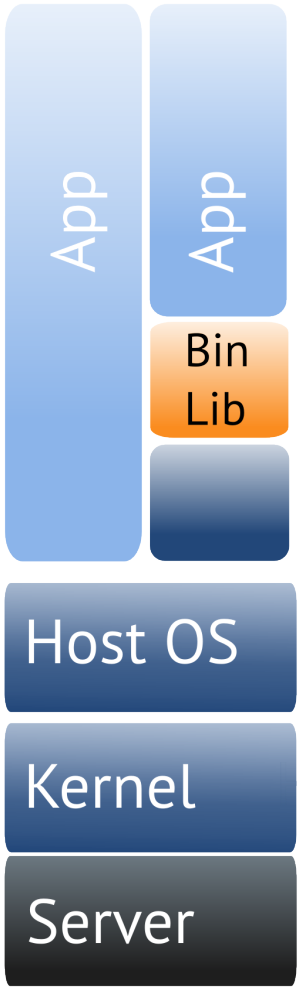


LXC APP Container:

```
1 Prozess direkt vom Host
kaum Isolation
Bereitstellung: sofort
Größe: ~0
Performance: nativ
Overhead: ~0
```

LXC OS Container:

```
Init Prozess o.ä.
gute Isolation
Bereitstellung: nativ
Größe: MBs
Performance: nativ
Overhead: gering
```



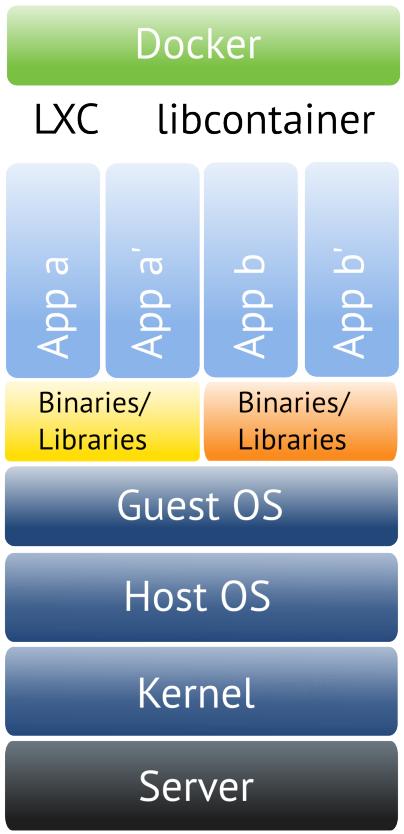
Definition LXC APP Container:

```
<domain type='lxc'>
  <name>sh</name>
  <uuid>58f081cd-09f9-4a4a-9f37-5d3bc687bf53</uuid>
  <memory unit='KiB'>908288</memory>
  <os> <type>exe</type> <init>/bin/sh</init> </os>
...
```

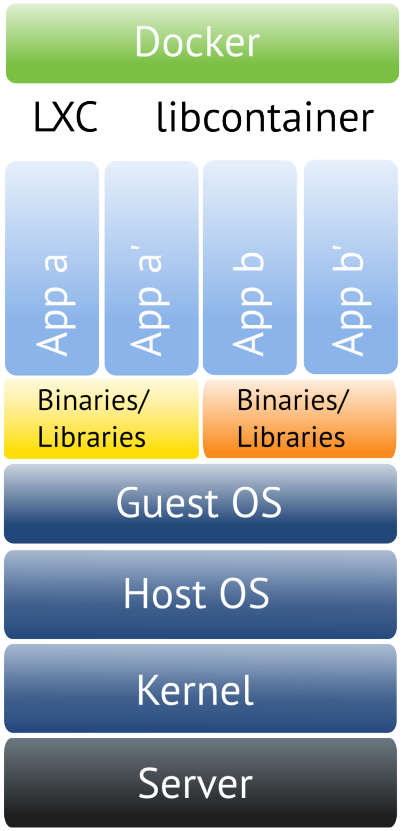
Definition LXC OS Container:

```
  <os> <type>exe</type> <init>/sbin/init</init> </os>
...
  <filesystem type='mount' accessmode='passthrough'>
      <source dir='/srv/lxc-sles12'/>
      <target dir='/'/>
    </filesystem>
```

Docker – Eigenschaften



- Applikationsfokus:
  „*Build, Ship and Run Applications*“
- nutzt Container-Technologien (LXC/libcontainer)
- gute Isolation der Container
- Bereitstellung: Sekunden – Minuten
- Größe: MBs



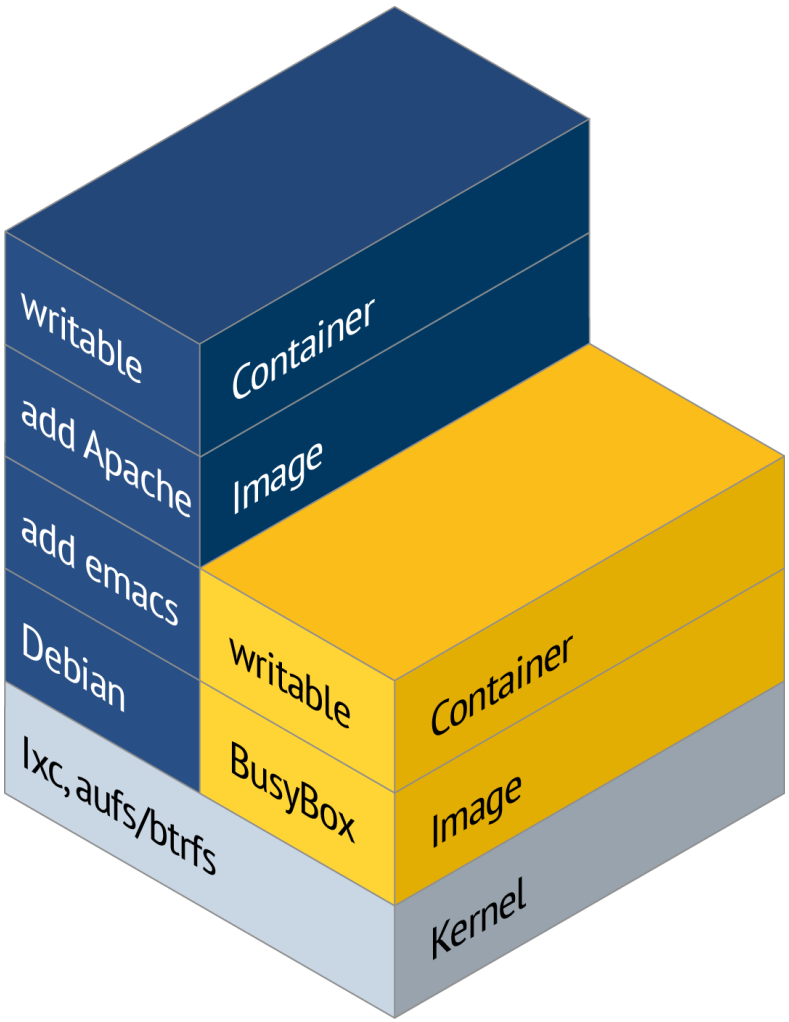
Definition einer Docker-App:

```
FROM debian:latest
RUN apt-get install -y emacs
RUN apt-get install -y apache2
EXPOSE 80
ADD www /var/www/site
CMD /usr/sbin/apache2ctl -D FOREGROUND
```

build und run der Docker-App:

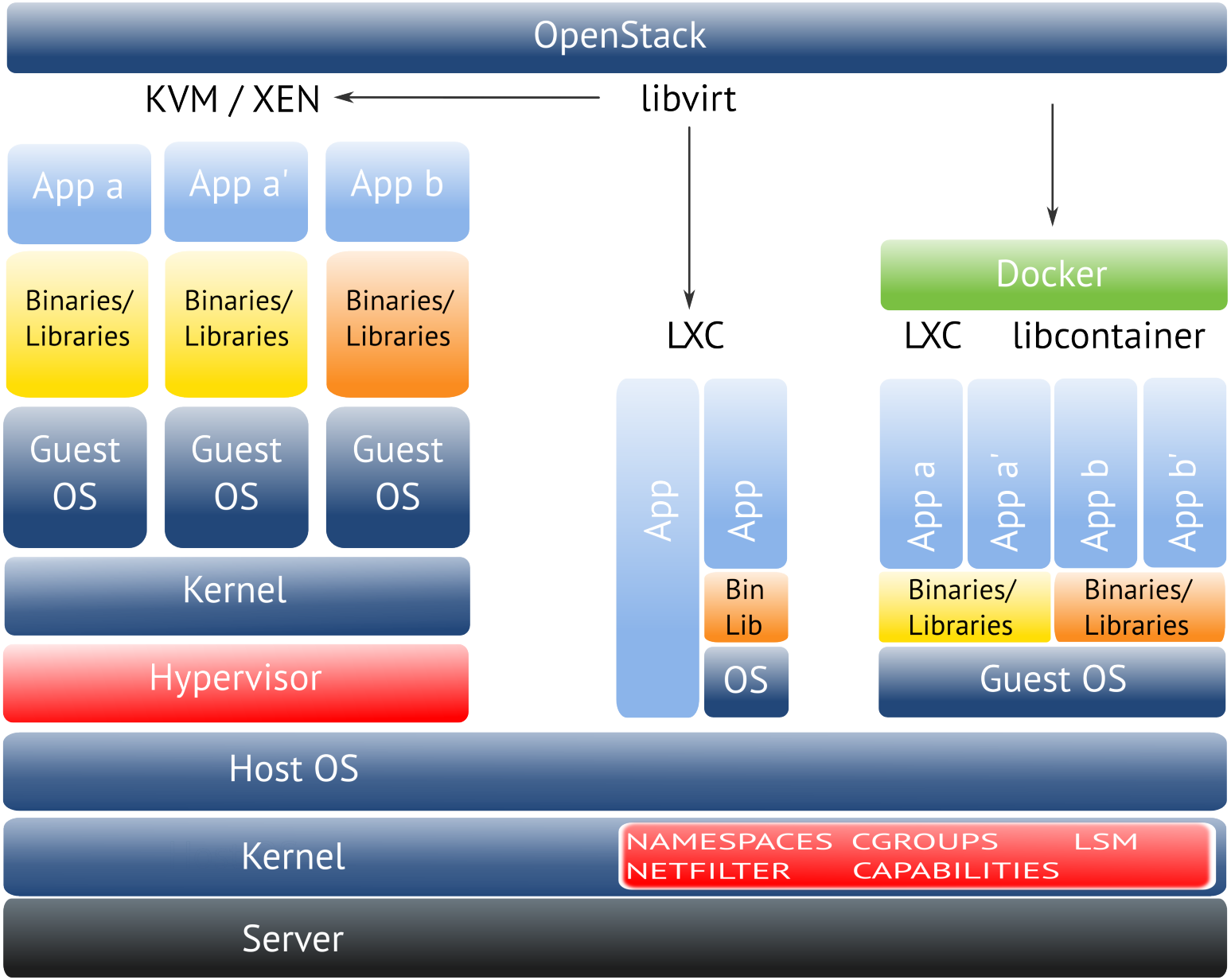
```
$ docker build -t baumann/myapache .
$ docker run -P -d baumann/myapache
```

Docker Layered Images



build eines *Docker Layered Image*:

```
$ docker build -t baumann/myapache
Uploading context 10.24 kB
Uploading context
Step 1 : FROM debian
 ---> bbba202fe96b
Step 2 : CMD apt-get install emacs
 ---> Using cache
 ---> 31182097be13
Step 3 : CMD  apt-get install ap..
 ---> Using cache
 ---> 2a5ffc17324d
....
Successfully built 2a5ffc17324d
```


OpenStack
KVM / XEN
libvirt
App a
App a'
App b
Binaries/ Libraries
Binaries/ Libraries
Binaries/ Libraries
Guest OS
Guest OS
Guest OS
Kernel
Hypervisor
LXC
App
App
Bin Lib
OS
Docker
LXC
libcontainer
App a
App a'
App b
App b'
Binaries/ Libraries
Binaries/ Libraries
Guest OS
Host OS
Kernel
NAMESPACES
CGROUPS
LSM
NETFILTER
CAPABILITIES
Server

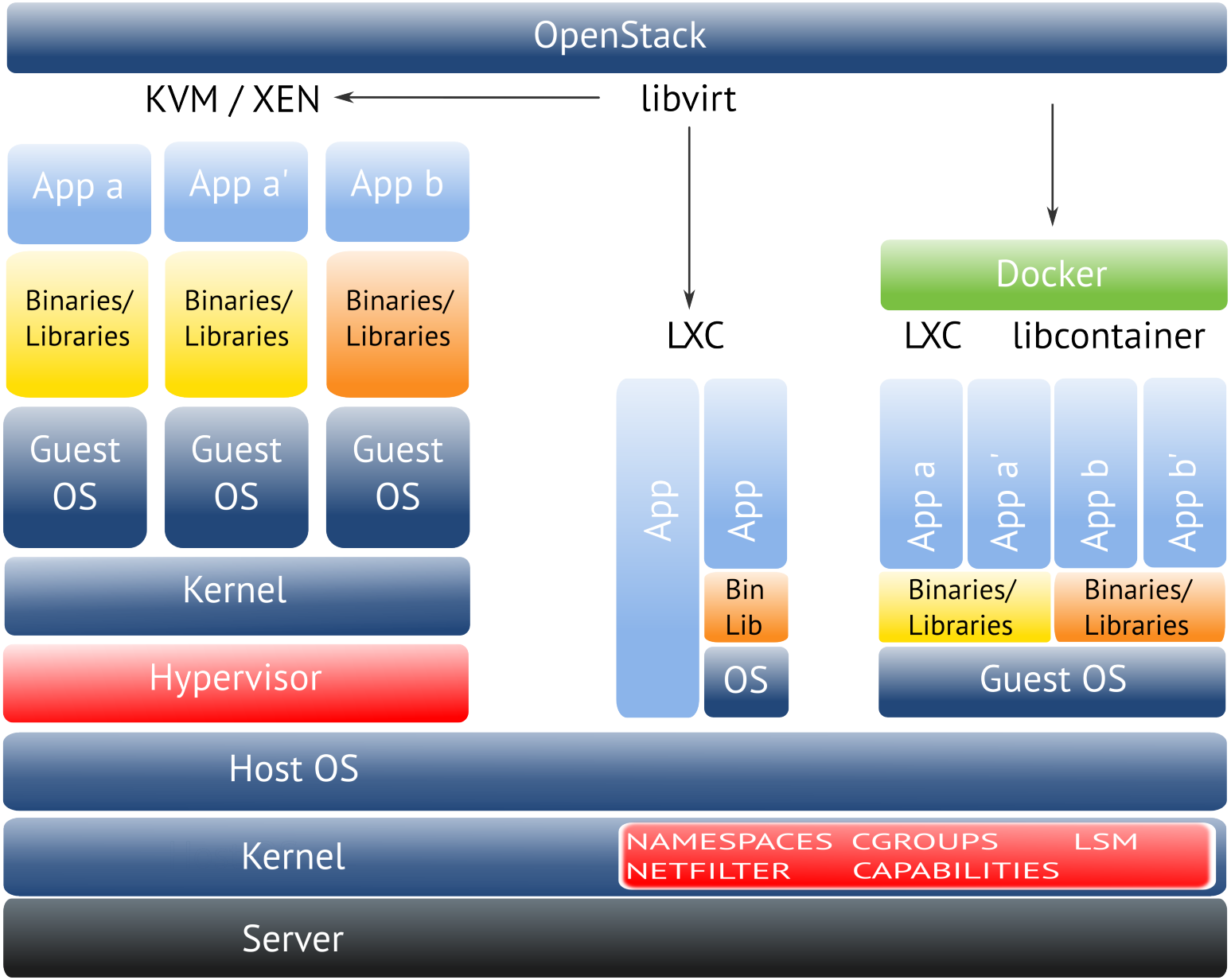
Docker – Build, Ship and Run Applications

Container Execution Environments

- Docker kann derzeit drei *Execution Driver* für die Erstellung von Containern nutzen:

LXC	bewährte leichtgewichtige Prozessvirtualisierung im Linux-Kernel
Libcontainer	neue Eigenentwicklung von Docker zur Nutzung der Kernel-Features
libvirt-lxc	LXC-Treiber mittels `libvirt`

- Libcontainer ist Default seit Version 1.0
- Ein einzelner Kernel sorgt für die Limitierung und Isolation der Ressourcen.
  - ⇒ Emulation von Hardware entfällt
  - ⇒ Performance-Overhead wird minimiert


OpenStack
KVM / XEN
libvirt
App a
App a'
App b
Binaries/ Libraries
Binaries/ Libraries
Binaries/ Libraries
Guest OS
Guest OS
Guest OS
Kernel
Hypervisor
LXC
App
App
Bin Lib
OS
Docker
LXC
libcontainer
App a
App a'
App b
App b'
Binaries/ Libraries
Binaries/ Libraries
Guest OS
Host OS
Kernel
NAMESPACES
CGROUPS
LSM
NETFILTER
CAPABILITIES
Server

Linux Containers (LXC)

- leichtgewichtiger Virtualisierungsmechanismus (Prozessvirtualisierung) mit modularem Design
- verwaltet Ressourcen und Prozessgruppen voneinander isoliert in so genannten *Containern*
- nutzt vorhandene Kernel-Features wie Capabilities, Control Groups und Namespaces
- „*chroot on steroids*“ (`http://lxc.sourceforge.net`)
- seit 2.6.29 Teil des Upstream Kernels (und damit aller gängigen Distributionen)
- freie Software (größtenteils GNU-LGPL-Lizenz)

libcontainer

- Go-Implementation, um Linux-Kernel-Namespaces ohne weitere Abhängigkeiten für Container zu nutzen
- vollständig unabhängig von LXC
- verwaltet als Teil von Docker die nötigen Kernel-Features:
  - Capabilities
  - Control Groups
  - Namespaces
  - Apparmor-/SElinux-Profile
  - Netzwerkschnittstellen
  - Firewall-Regeln

Capabilities

- teilen die Rechte eines privilegierten Benutzers (`root`) in bestimmte Teilbereiche auf
- können für unterschiedliche Prozesse individuell gesetzt werden
  $\Rightarrow$ geringere Gefahr eines Missbrauchs von Rechten
- da Container = Prozesse besteht die Möglichkeit, bestimmte Rechte innerhalb eines Containers einzuräumen
- minimieren Wechselwirkungen (z. B. beim Mounten des root-Dateisystems mit `read only` beim Herunterfahren eines Containers)

CGroups

- fassen Prozesse in einer hierarchischen Struktur zusammen.
- ermöglichen es, den Zugriff dieser Prozesse auf Ressourcen zu beschränken oder zu unterbinden und so die Nutzung dieser Ressourcen zu limitieren.
- Typische Ressourcen sind Prozessoren, Arbeitsspeicher, Netzwerkressourcen oder `Kernel Namespaces`.
- Die hierarchische Struktur ermöglicht es, Eigenschaften von Subsystemen auf mehreren Ebenen zu unterteilen.
- Zu jeder dieser Ebenen lässt sich die Nutzung der möglichen Ressourcen weiter einschränken.
- Prozesse (Container) lassen sich dann gezielt in die Tasklist der betreffenden CGroup aufnehmen.

Chroot

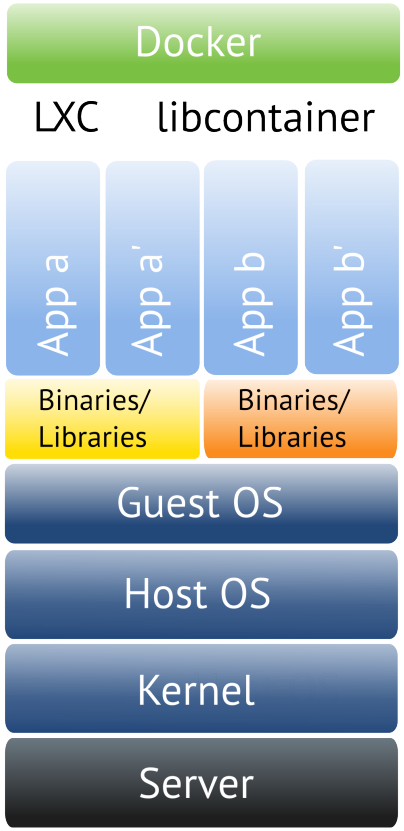
- Funktion unter Unix-Systemen, um im laufenden System ein neues Rootverzeichnis zuzuweisen
- primär für das Aufsetzen virtueller Umgebungen entwickelt
- einfache Möglichkeit, nicht vertrauenswürdige Programme zu isolieren (aka. „Sandbox“)
- einfacher Jail-Mechanismus – aus dem aber durchaus ausgebrochen werden kann
- Prozesse im Chroot-Verzeichnis können nicht mehr auf Dateien außerhalb dieses Verzeichnisses zugreifen
- Programm muss im Chroot-Verzeichnis eine komplette Umgebung vorfinden (Platz für temporäre Dateien, Konfigurationsdateien, Programmbibliotheken, . . .)
- keine Ressourcen-Limitierung

Kernel Namespaces

- Integration von Namespaces im Kernel mit Version 2.6.19
- Namespaces helfen, Prozesse voneinander zu isolieren
- mögliche Eigenschaften zum Definieren eines Namensraumes:

`ipc`	System-V-Interprozess-Kommunikation (IPC)
`uts`	Hostname
`mnt`	Mountpoints und Dateisysteme
`pid`	Prozesse
`net`	Netzwerk-Stack
`user`	Benutzer (UIDs)

Bestandteile – Das Docker-Universum



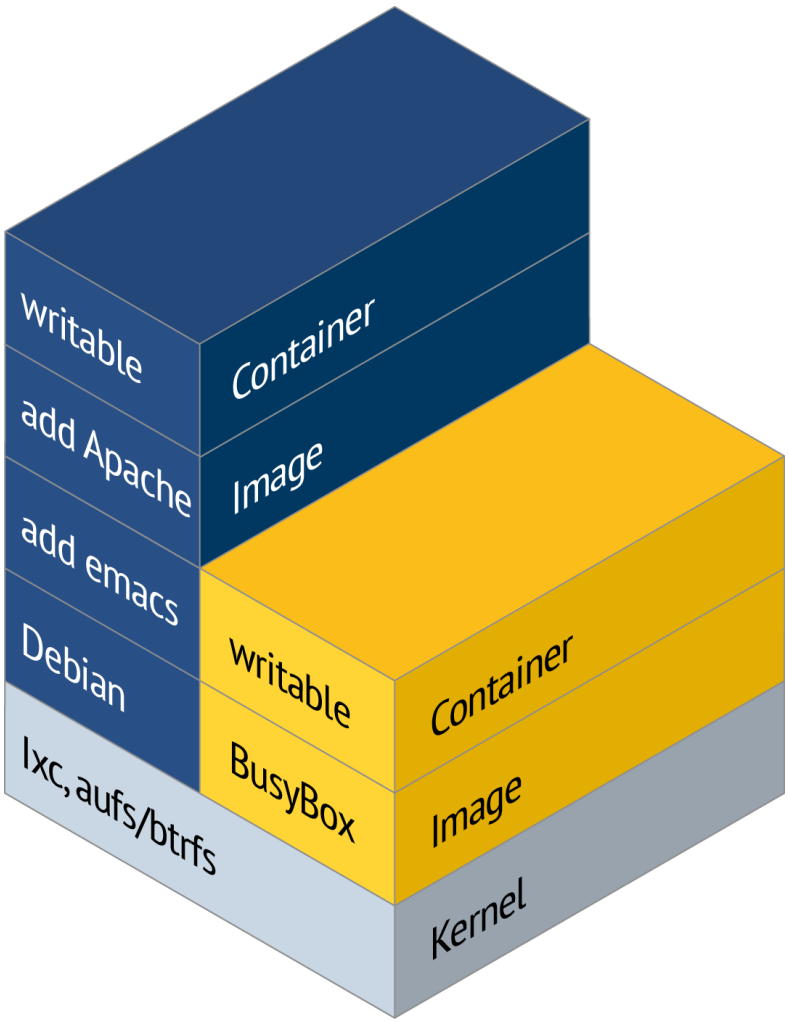
Definition einer Docker-App:

```
FROM debian:latest
RUN apt-get install -y emacs
RUN apt-get install -y apache2
EXPOSE 80
ADD www /var/www/site
CMD /usr/sbin/apache2ctl -D FOREGROUND
```

build und run der Docker-App:

```
$ docker build -t baumann/myapache .
$ docker run -P -d baumann/myapache
```

Docker Layered Images



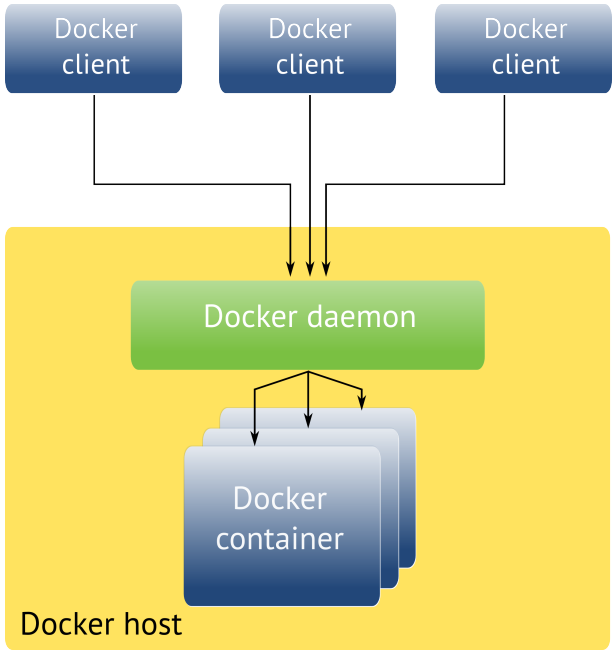
build eines *Docker Layered Image*:

```
$ docker build -t baumann/myapache
Uploading context 10.24 kB
Uploading context
Step 1 : FROM debian
 ---> bbba202fe96b
Step 2 : CMD apt-get install emacs
 ---> Using cache
 ---> 31182097be13
Step 3 : CMD  apt-get install ap..
 ---> Using cache
 ---> 2a5ffc17324d
....
Successfully built 2a5ffc17324d
```

Images, Container und Registries

- *Docker Images*
  - Read-Only-Templates
  - Basis für Container
- *Docker Container*
  - wird auf Basis eines Image und Dockerfile erzeugt
  - hält zur Laufzeit die Änderungen vor (Read-/Write-Layer)
  - Änderungen können in einem neuen Image übernommen („committed“) werden.
- *Docker Registries*
  - Image Repository
  - privat und öffentlich
  - zentrale Ressource: *Docker Hub*

Bestandteile – Die Docker-Engine



Die Docker Engine besteht aus zwei Teilen:

Server Daemon für den Serverprozess zur Verwaltung der Container.

Client Client zur (Fern-)Steuerung des Daemons.

Docker Projekte

Docker ist kollaborativ, modular und erweiterbar:

Atomic	atomar aufgebautes Hostsystem für Docker-Container
CoreOS	minimales Linux als Container-Betriebssystem
Machine	Docker Rollout Werkzeug
Swarm	nativer Docker Cluster
Compose	Multi Tier Applikations Rollout
boot2docker	Docker auf MacOS und Windows
Packer	Erstellung fertiger Images mittels Templates
	...

Docker und OpenStack

Docker im OpenStack Kontext:

novadocker	Docker als Backend in Nova
heat docker ressource	Docker Einbindung in Heat
Dockenstack	Devstack auf Docker
Kolla	OpenStack in Docker
Magnum	Container Orchestration in OpenStack
	...

Vielen Dank für Ihre Aufmerksamkeit!

Bei weiteren Fragen wenden Sie sich bitte an info@b1-systems.de
oder +49 (0)8457 - 931096