

The background is a solid dark blue. In the top-left corner, there are several overlapping squares and rectangles in shades of light blue and teal. In the bottom-right corner, there are several overlapping triangles and quadrilaterals in shades of teal and light blue.

Container Technologie Docker

1. Einführung in Container

1. Was sind Container?
2. Anwendungsbeispiele
3. Chancen und Herausforderungen

2. Techlab

Agenda



©Håkan Dahlström

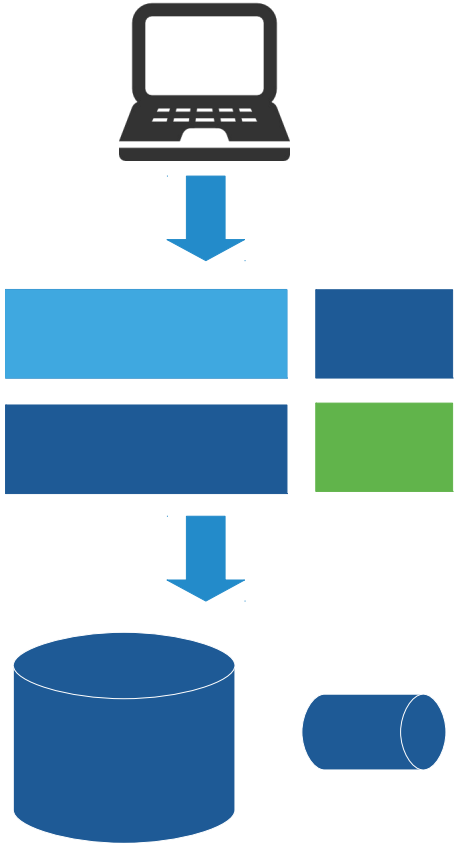
Source: <https://www.flickr.com/photos/dahlstroms/3144199355>

Container Hype

in aller Munde

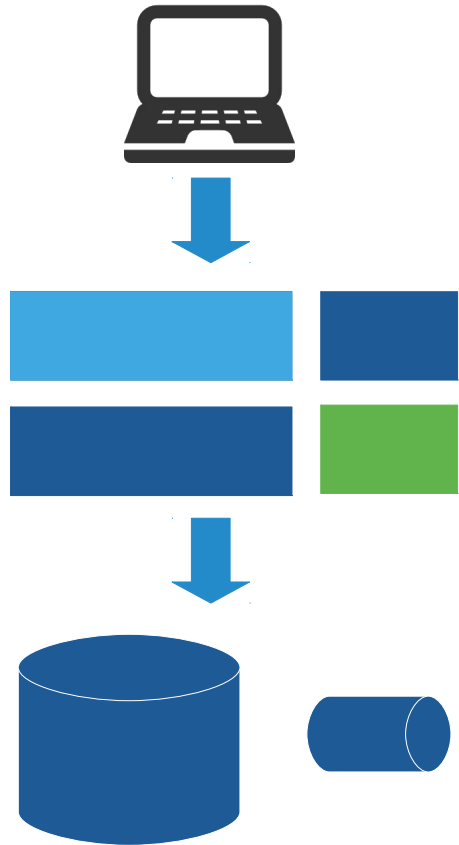
Wie hat sich die IT entwickelt?

1995

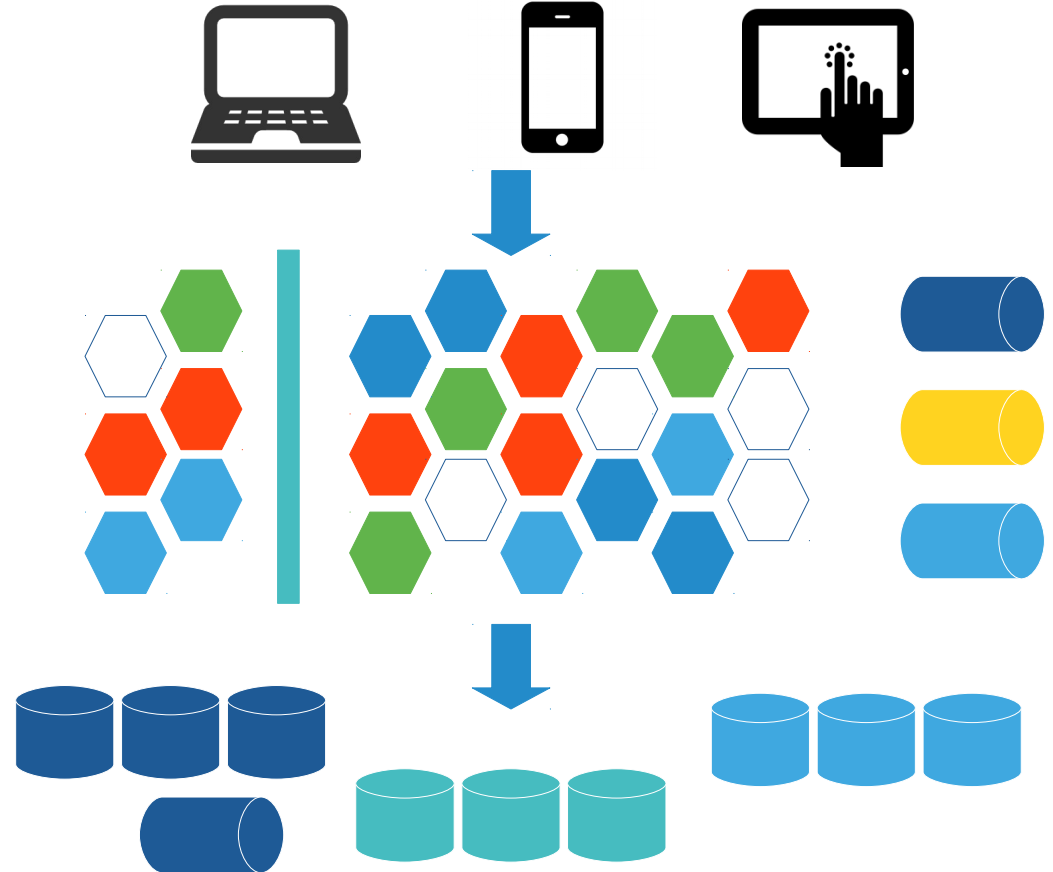


Wie hat sich die IT entwickelt?

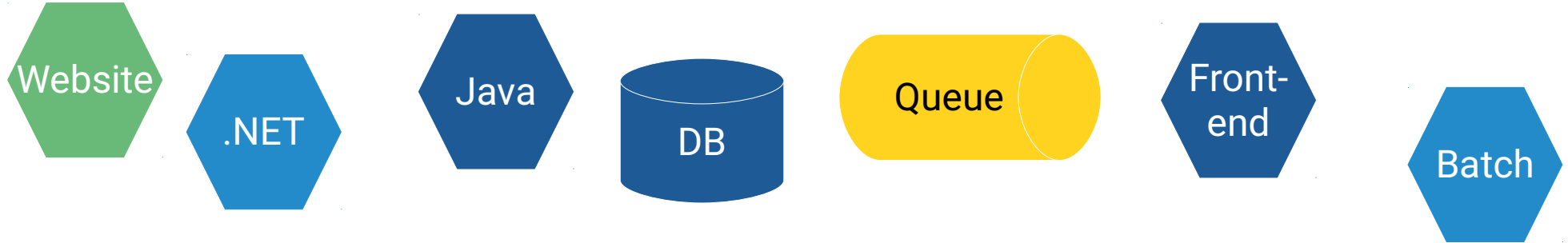
1995



2018



Unmenge an Kombinationen

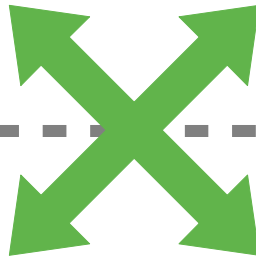


Grosse Anzahl Services

Gewollte Interaktionen

Grosse Anzahl Umgebungen

Schnelle und einfache Migrationen



Local

UAT

Prod

Pre Prod

DEV

Matrix from Hell

Website	?	?	?	?	?	?	?
Frontend	?	?	?	?	?	?	?
Webservice	?	?	?	?	?	?	?
Datenbank	?	?	?	?	?	?	?
Queue	?	?	?	?	?	?	?
Application	?	?	?	?	?	?	?
	Entwickler PC	Dev	Test	Prod	Cloud	Customer Server	...

Gütertransport

vor 1960



A 164



A 157

Unmenge an Kombinationen

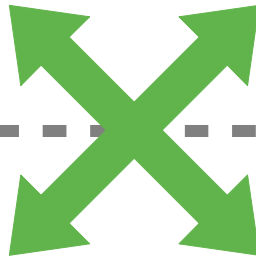
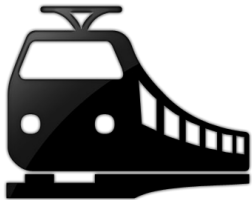


Grosse Anzahl Güter

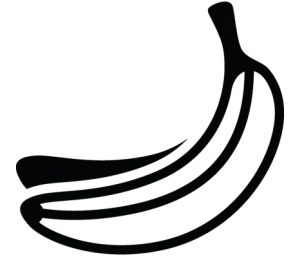
Interaktion zwischen Gütern

Grosse Anzahl Transportwege

Schneller und reibungsloser
Transport



Die Lösung



Grosse Anzahl Güter



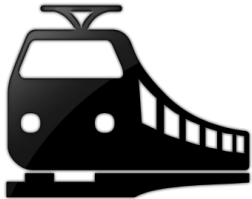
Grosse Anzahl Transportwege



Interaktion zwischen Gütern



Schneller und reibungsloser
Transport



Applikationen in Container

Website

.NET

Java

DB

Queue

Front-end

Batch

Grosse Anzahl Services

Grosse Anzahl Umgebungen



Gewollte Interaktionen

Schnelle und einfache Migrationen

Local






























UAT

Prod

Pre Prod

DEV

Matrix Reloaded

Website							
Frontend							
Webservice							
Datenbank							
Queue							
Application							
	Entwickler PC	Dev	Test	Prod	Cloud	Customer Server	...

Was bringt uns nun das Ganze?

- Standardisierter Mechanismus für Builden, Deployen und Betreiben von Applikationen
- Isolation der Applikationen
- Saubere Definition der Schnittstellen gegenüber der Betriebsplattform
- Möglichkeit zur Vereinheitlichung der Workflows
- Dev- / Test Umgebung analog Prod

Hello World Docker Beispiel

```
docker run fedora-minimal /bin/echo "Hello world"
```



Was ist jetzt im Hintergrund passiert?

- Start Container
- Allokation Filesystem
- Mount Read/Write-Filesystem Layer
- Anhängen Netzwerk Layer
- Ausführen ``echo``-Befehl
- Output an meine Console
- Stop Container
- ...

In weniger als einer
Sekunde

The background is a solid blue color. In the top-left and top-right corners, there are decorative geometric shapes made of triangles and squares in various shades of blue and teal. The text "Was sind Container?" is centered in the upper half of the image.

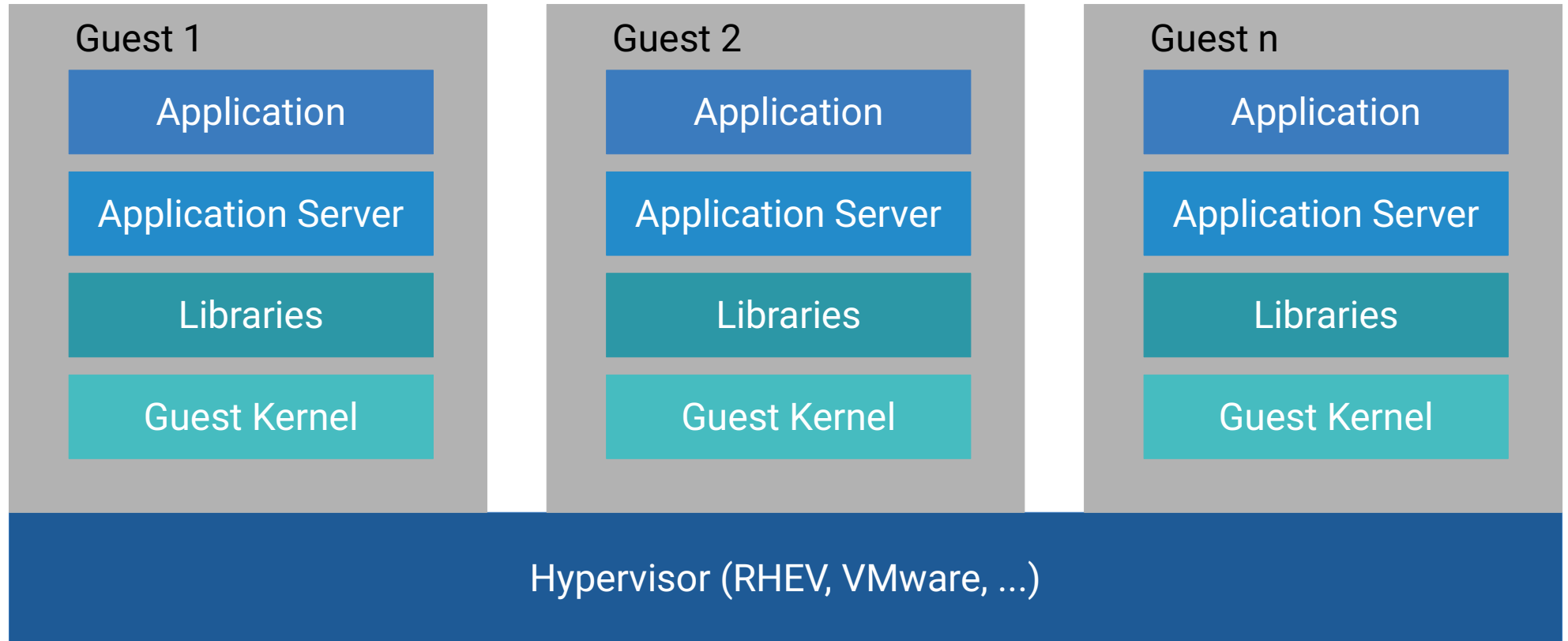
Was sind Container?

The background is a solid blue color. In the top-left and top-right corners, there are decorative geometric shapes made of overlapping triangles and squares in various shades of blue and teal.

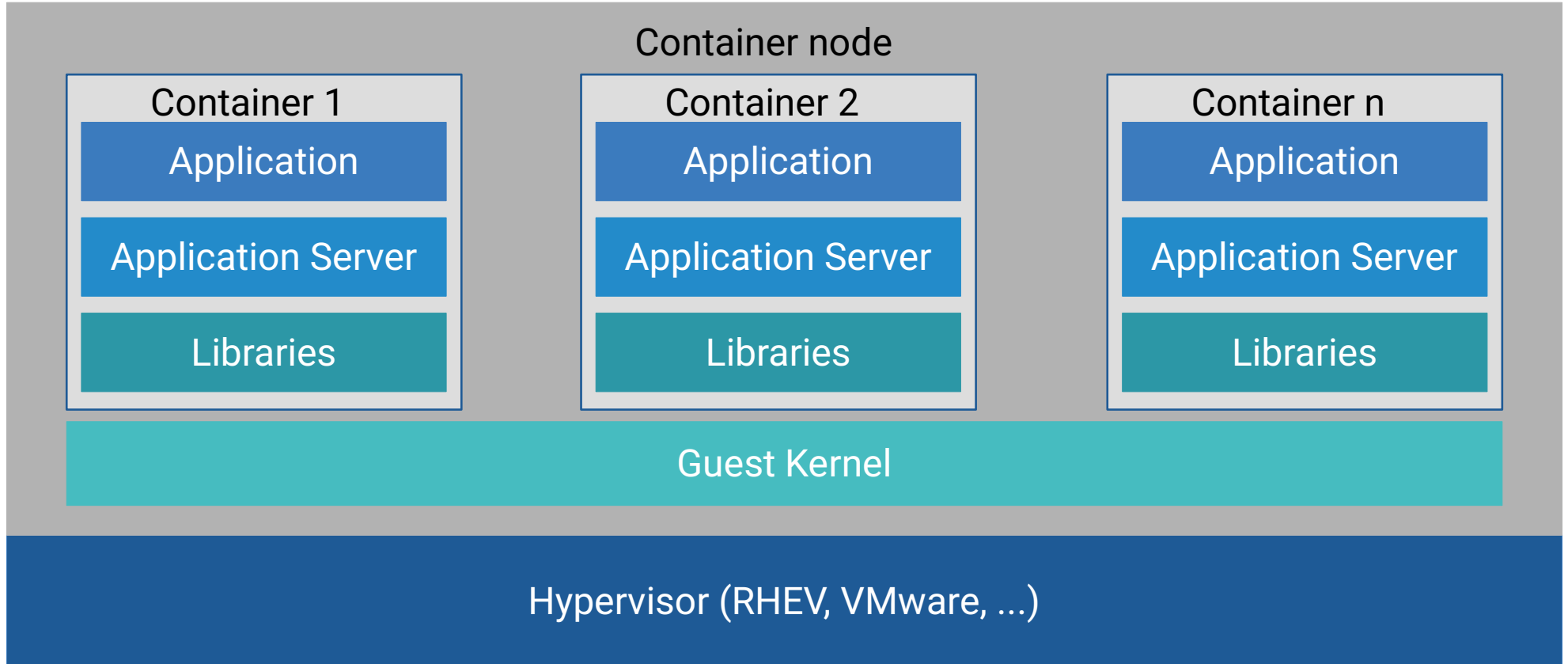
Nichts neues!

LXC, VServer, Free BSD Jails, Google...

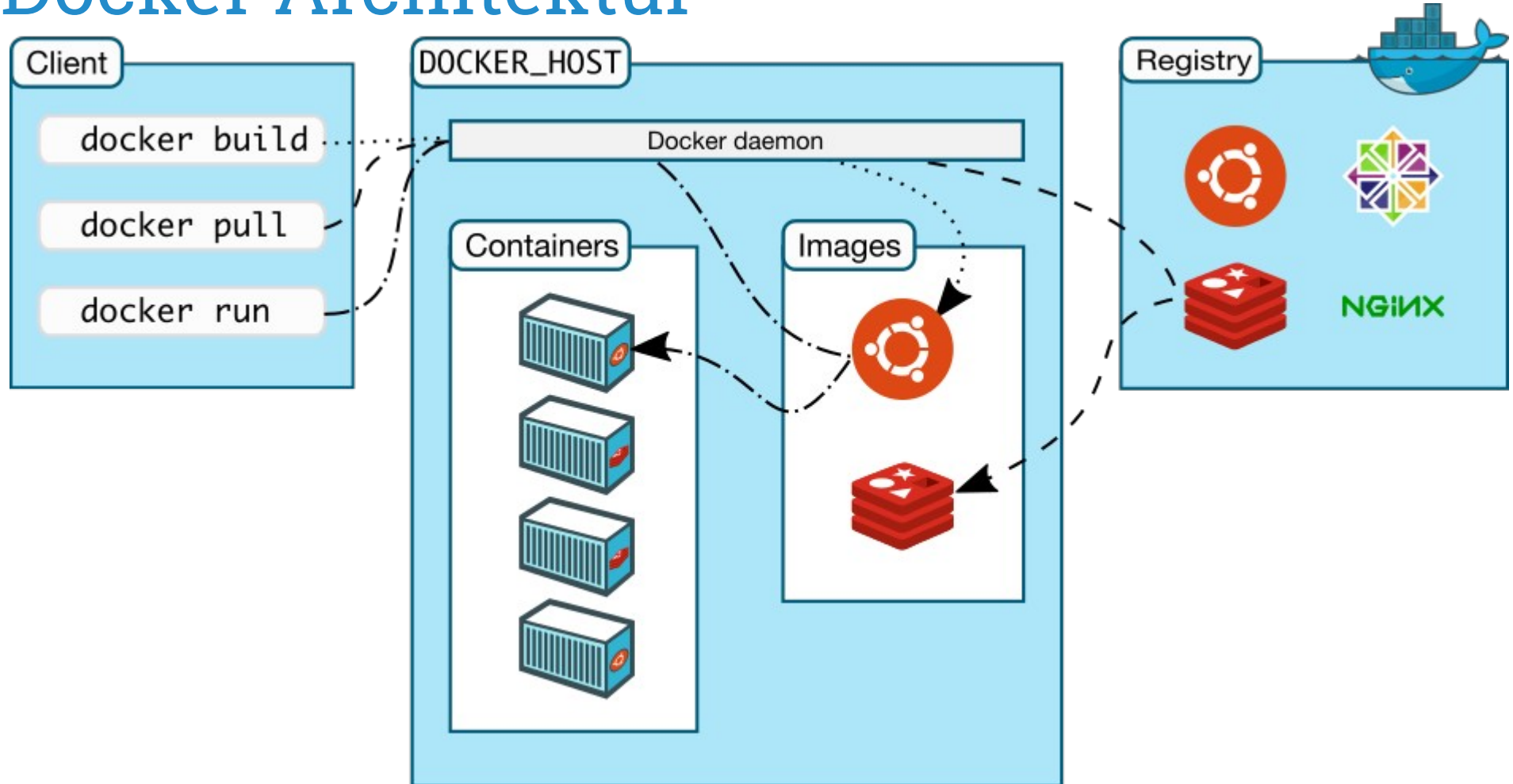
Klassische Virtualisierung



Container Virtualisierung



Docker Architektur



Docker Hub

- Public Registry
- Vielzahl an offiziellen Images verfügbar
- Buildplattform für eigene Images
- Auto Build und Integration mit GitHub

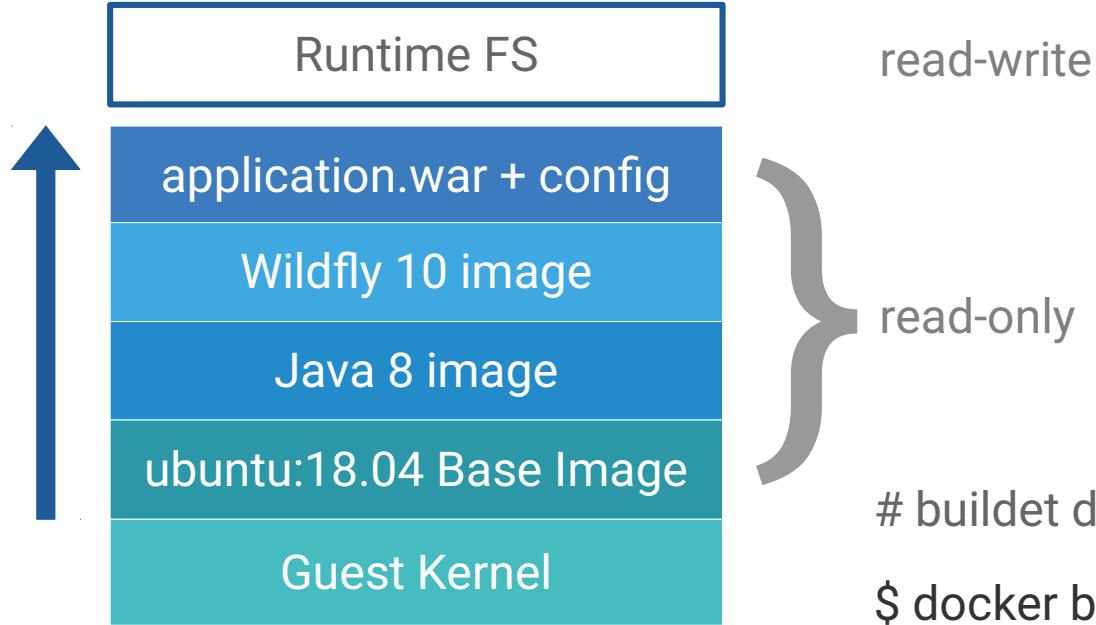
Hinter den Kulissen

- Docker ist in GO implementiert
- Namespaces als isolierte Umgebung für Container
- Control Groups
 - Limitiert Ressourcen für Prozesse
- Union File System
 - Ermöglicht Filesystem Layering

Docker Container



Hierarchische Dockerfiles



buildet das Dockerfile unter ./

\$ docker build .

container von image app starten

\$ docker run -p 8080:8080 app

Container und deren Infrastruktur

- Container sind unveränderbar
- Aktualisierung eines Containers erfolgt über Austauschen
- Sowohl Applikation wie auch System Patches
- Kein lokales Filesystem für Applikationsdaten
- Persistent Storage

Dockerfile (1)

```
FROM centos7
```

```
MAINTAINER Thomas Philipona <philipona@puzzle.ch>
```

```
EXPOSE 8080
```

```
# Install Java
```

```
RUN INSTALL_PKGS="tar unzip bc which lsof java-1.8.0-openjdk  
java-1.8.0-openjdk-devel" && \
```

```
    yum install -y $INSTALL_PKGS && \
```

```
    yum clean all -y
```


Dockerfile (2)

...

```
USER 1001
```

```
# Add application source to Dockerfile
```

```
ADD . /opt/app-root/src/
```

```
# build application and copy to correct location
```

```
RUN sh /opt/app-root/src/gradlew build && \
```

```
    cp -a  
    /opt/app-root/src/build/libs/springboots2idemo*.jar /opt/app-  
root/application.jar
```

Dockerfile (3)

...

```
CMD ["java", "-Xmx64m", "-Xss1024k", "-jar", "/opt/app-root/application.jar"]
```

Best Practices Dockerfiles (1)

- Container sollen zustandslos sein, Zustand nur in gemounteten Volumes
- Ein Prozess pro Container
- Anzahl Layers möglichst klein halten
- .dockerignore File verwenden
- Loggen via stdout → Plattform macht Aggregation / Rotation

Best Practices Dockerfiles (2)

- Trusted Base Images verwenden
- Automatisches Rebuild → Security Patches
- Multiline Argumente sortieren

```
RUN apt-get update && apt-get install -y \  
    bzip2 \  
    curl \  
    git \  
    mercurial \  
    subversion
```



Anwendungsbeispiele

LaTeX Rendering Infrastruktur

Installation von LaTeX Rendering Infrastruktur je nach Betriebssystem sehr aufwändig

Rendering Infrastruktur als Docker Container →

Standardisierung, Nachvollziehbarkeit, Dokumentation über installation der Infrastruktur, schnell

```
docker run pdflatex -output-directory output /input.tex
```

Buildinfrastruktur für Applikationen

Java, JavaScript, Ruby on Rails, Node, ...

Installation von Buildtools in verschiedenen Versionen für unterschiedliche Projekte

Exakte Umgebung die explizit für Applikation definiert wird

Wiederverwendbar, schnell, isoliert.

Java EE 7 Applikation Wildfly 10

War und Config ins Image hinzufügen und Go!

```
FROM jboss/wildfly  
  
ADD app-web.war /opt/jboss/wildfly/standalone/deployments/  
ADD standalone.xml /opt/jboss/wildfly/standalone/configuration/
```

Und und und...

The top corners of the slide feature decorative geometric shapes. The top-left corner has a cluster of overlapping triangles in light blue, teal, and dark blue. The top-right corner has a similar cluster with a prominent teal diamond and light blue triangles. The rest of the slide has a solid blue background.

Chancen und Herausforderungen

Vorteile von Containern

- Leichtgewichtig und schnell
- Standardisierung
- Einfach zu gebrauchen und zu erweitern
- Grosse Community Docker Hub, Tool Ökosystem
- Vielzahl an Docker Images verfügbar
- Kann «Works on my Machine» Probleme ausmerzen

Containers in :

DEV

PROD

The "learning cliff" →

Docker

Security

code quality

container hosting

peer discovery

Config changes

supervision

monitoring

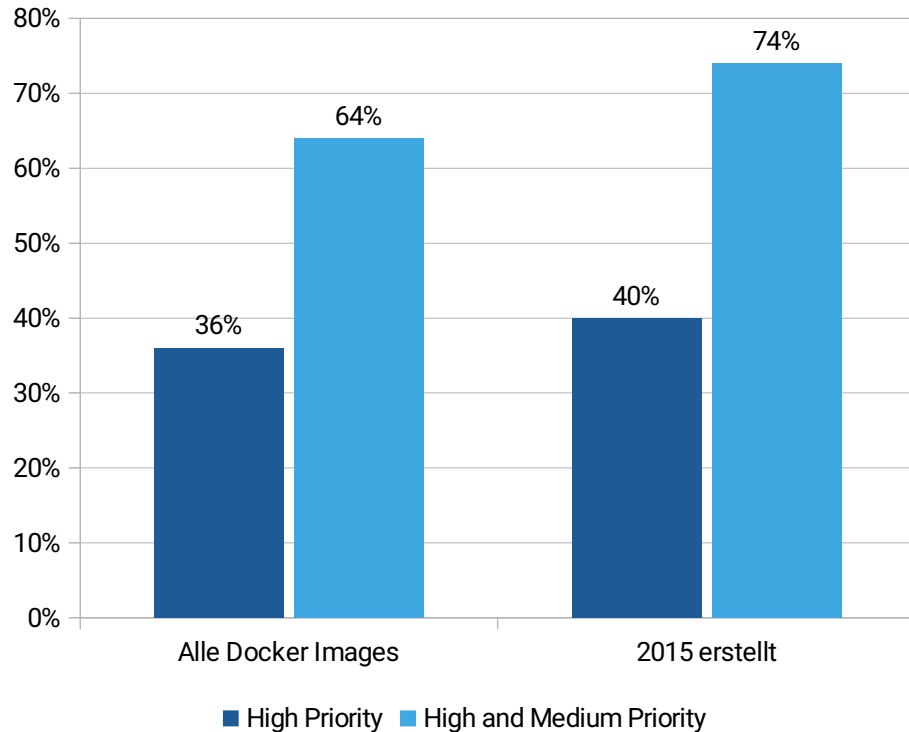
rolling deployment

libnetwork

Kubernetes / Mesos

Docker

Herausforderungen: Sicherheit (1/3)



64% aller Docker Images auf Docker Hub der offiziellen Repositories haben Sicherheitslücken:

ShellShock (Bash)

Heartbleed (OpenSSL)

Poodle (OpenSSL)


...

Quelle: <http://www.banyanops.com/blog/analyzing-docker-hub/>
Mai 2015: Jayanth Gummaraju, Tarun Desikan and Yoshio Turner

Herausforderungen: Sicherheit (2/3)


- Prozesse behandeln als würden sie auf Host laufen
- Prozesse nicht als root laufen lassen
- Nur notwendige Ports öffnen
- SELinux oder AppArmor im Container einsetzen

Docker Security Scanning

 DOCKER CLOUD

+

Get Help

 sanscontext

Repositories / Details / Docker / Java : 6

GENERAL

TAGS

TIMELINE

6

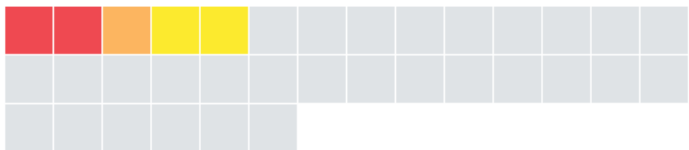
[View All Tags](#)

There are **22** vulnerable components (Last scanned 3 hours ago) [Provide Feedback](#)

/bin/sh -c #(nop...86a657ffdbc in / 85.8MB

▼

5 vulnerable components




/bin/sh -c #(nop...MD ["/bin/bash"] 1.0KB

No components in this layer

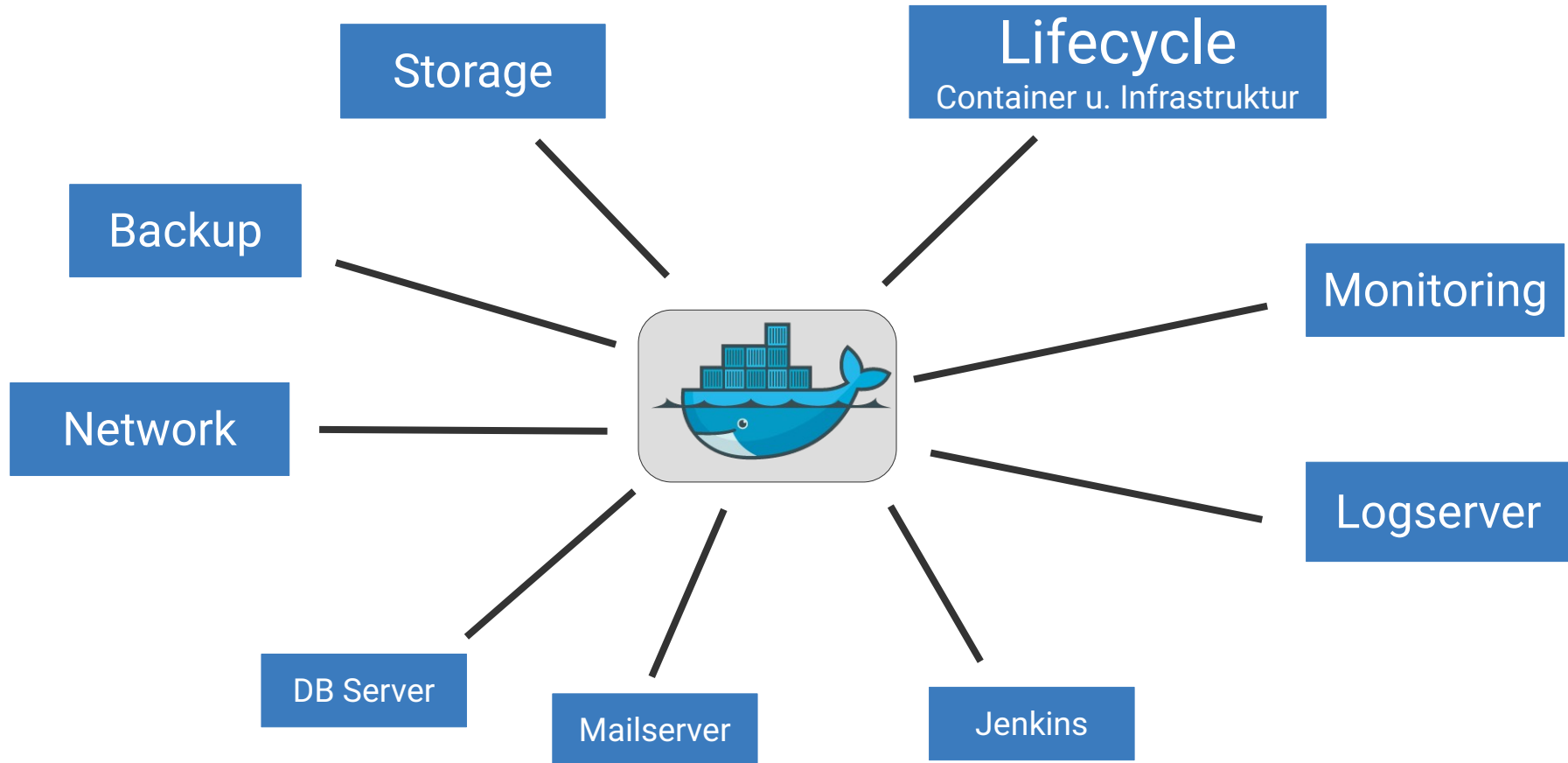
/bin/sh -c apt-g.../lib/apt/lists/* 1.1MB

▼

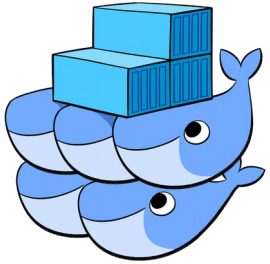
No known vulnerable components



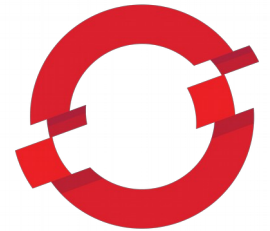
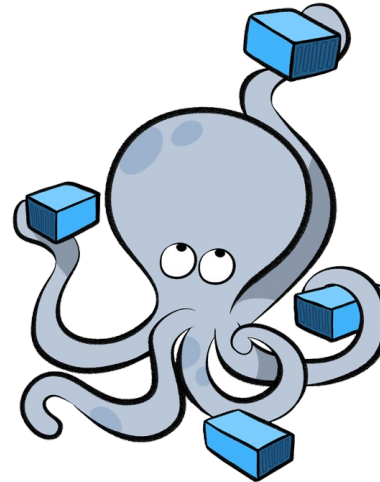
Herausforderungen: Betrieb



Deployment und Orchestration



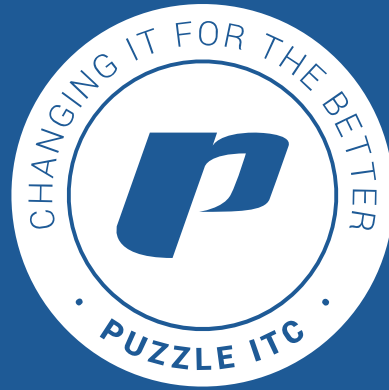
MESOS



OPENSIFT

The image features a solid blue background. In the top-left and top-right corners, there are decorative geometric patterns composed of overlapping triangles and squares in various shades of blue and teal. The word "Techlab" is centered in the middle of the image in a white, serif font.

Techlab



Thank you!