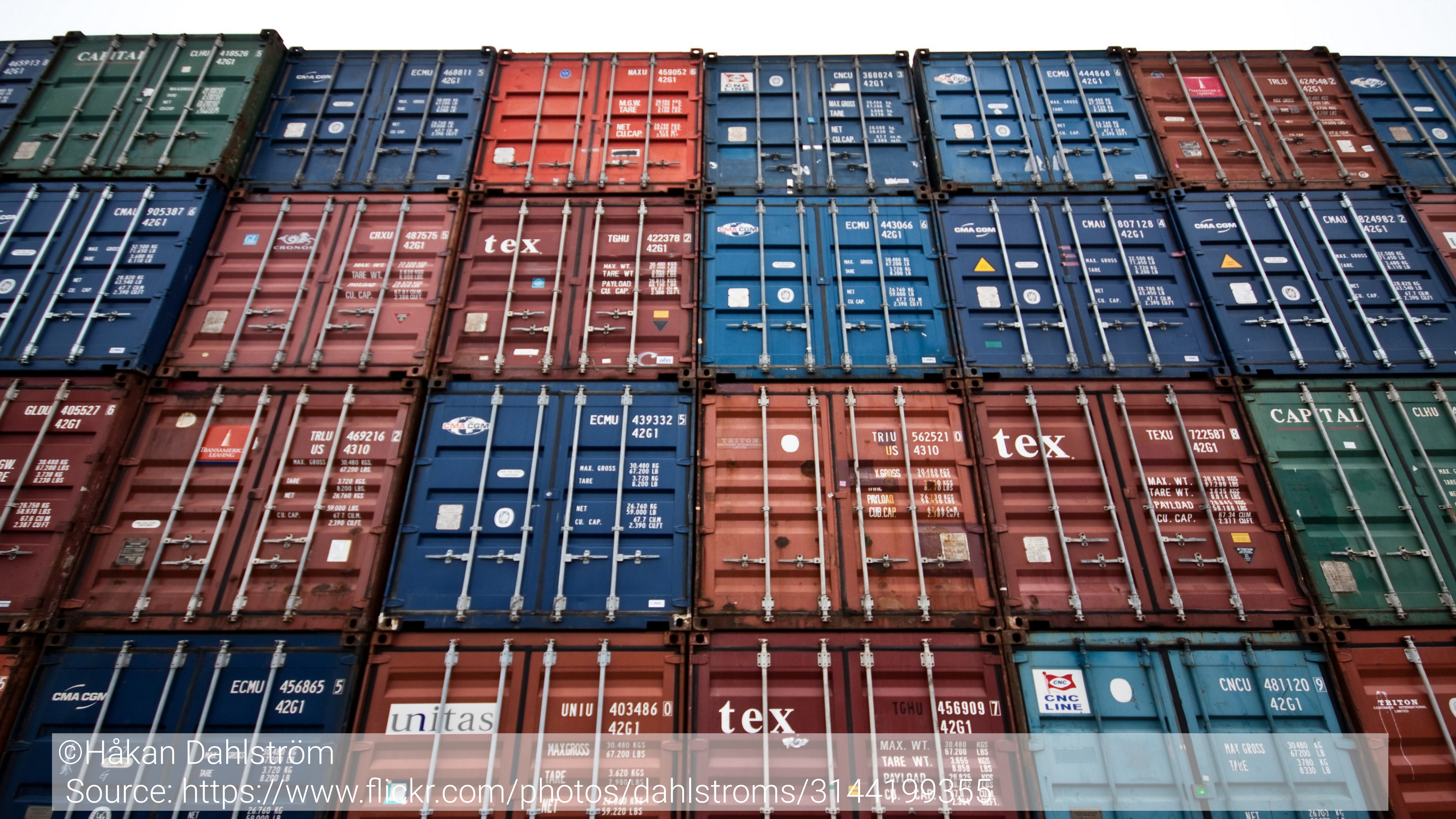
Container Technology Docker

1. Introduction to Containers
    1. What is a Container?
    2. Examples of Use
    3. Chances and Challenges
2. Techlab

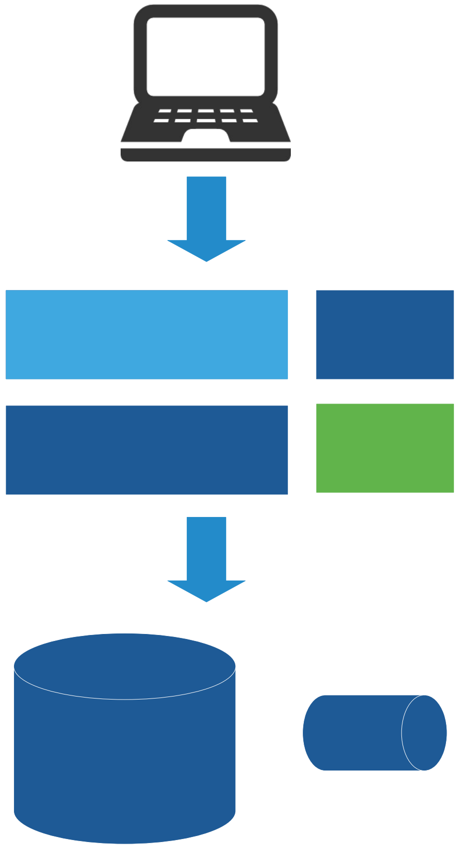
Agenda



©Håkan Dahlström
Source: https://www.flickr.com/photos/dahlstroms/3144199355

Developments in IT

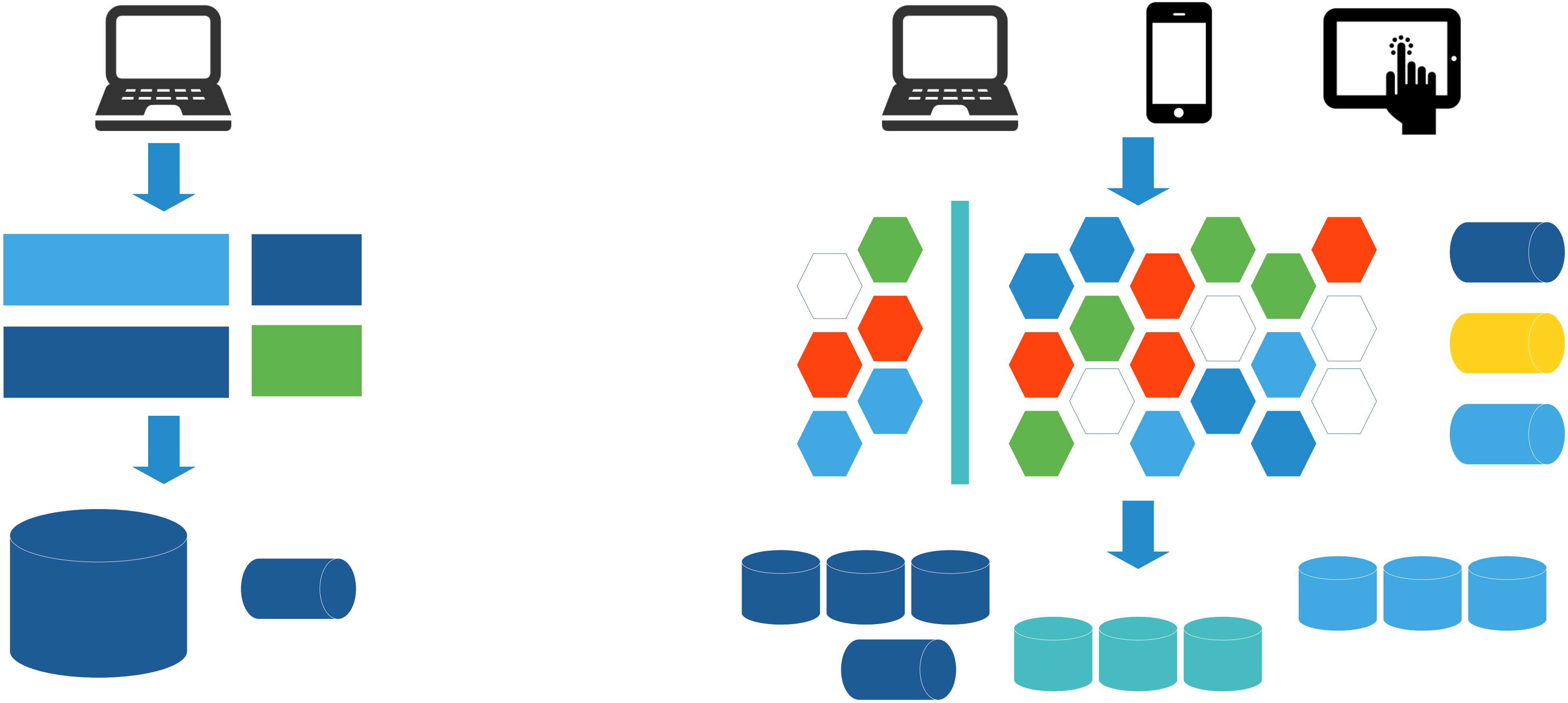
1995



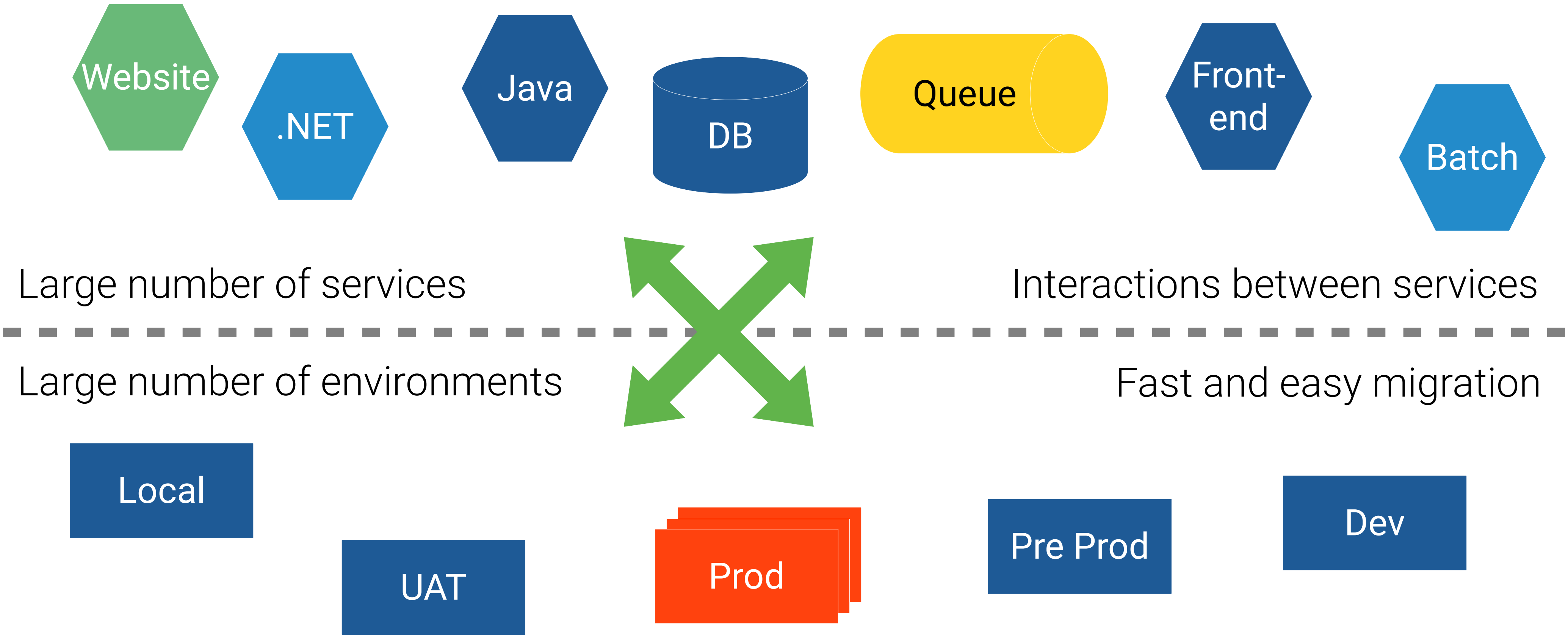
Developments in IT

1995

today



Plethora of Combinations



Website

.NET

Java

DB

Queue

Front-end

Batch

Large number of services

Interactions between services

Large number of environments

Fast and easy migration

Local

UAT

Prod

Pre Prod

Dev

Matrix from Hell

Website	?	?	?	?	?	?	?
Frontend	?	?	?	?	?	?	?
Web-Service	?	?	?	?	?	?	?
Database	?	?	?	?	?	?	?
Queue	?	?	?	?	?	?	?
Application	?	?	?	?	?	?	?
	Dev PC	Dev	Test	Prod	Cloud	Customer server	...

Goods Traffic before 1960

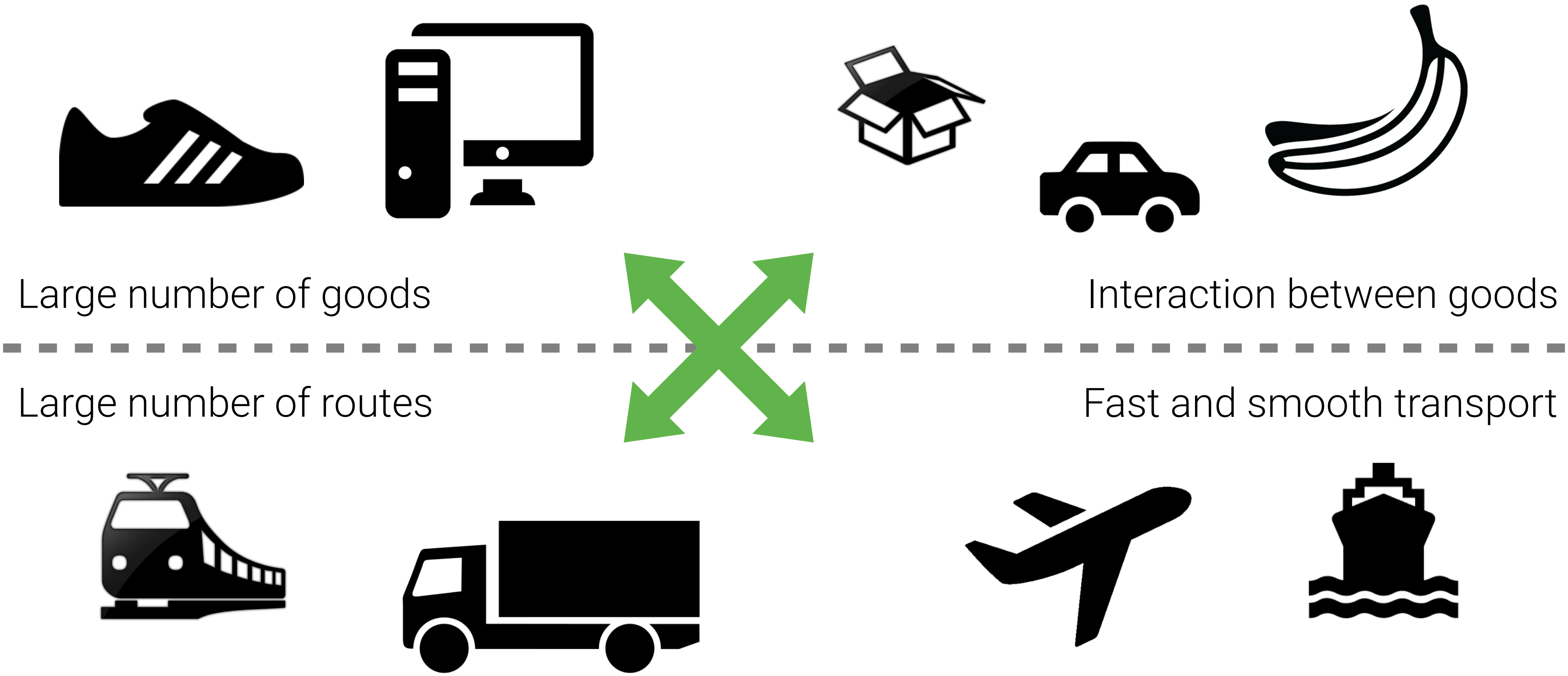
A 164

02159

3 28 39

A 157

Plethora of Combinations



Large number of goods

Interaction between goods

Large number of routes

Fast and smooth transport

Solution

Large number of goods

Large number of routes

Interaction between goods

Fast and smooth transport

Applications in Containers

Website

.NET

Java

DB

Queue

Front-end

Batch

Large number of services

Interactions between services

Large number of environments

Fast and easy migration

Local

UAT

Prod

Pre Prod

Dev

Matrix Reloaded

	Dev PC	Dev	Test	Prod	Cloud	Customer Server	...
Website							
Frontend							
Web-Service							
Database							
Queue							
Application							

Purpose

- Standardized mechanism for building, deploying and operating applications
- Isolation of applications
- Clear definition of interfaces between application and platform
- Potential to unifiy workflows
- Dev and test environments analog to production

CRIME
0800

Hello World Docker Example

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

What happend in the background?

- Container start
- Allocation of filesystem
- Mount of a read/write filesystem layer
- Attachment of networking layer
- Execution of `echo` command
- Output to my console
- Container stop

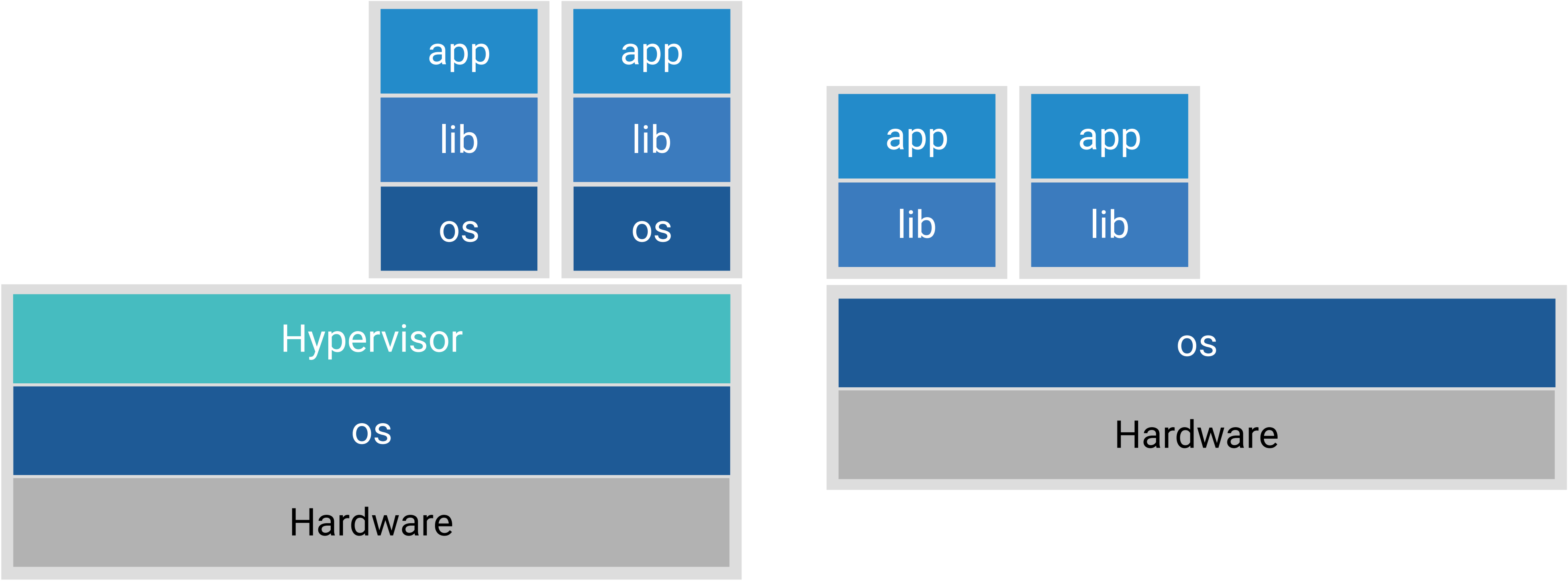
<ls

What is a Container?

Nothing new!

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

Virtualiziation vs. Containerization



app

lib

os

app

lib

os

Hypervisor

os

Hardware

app

lib

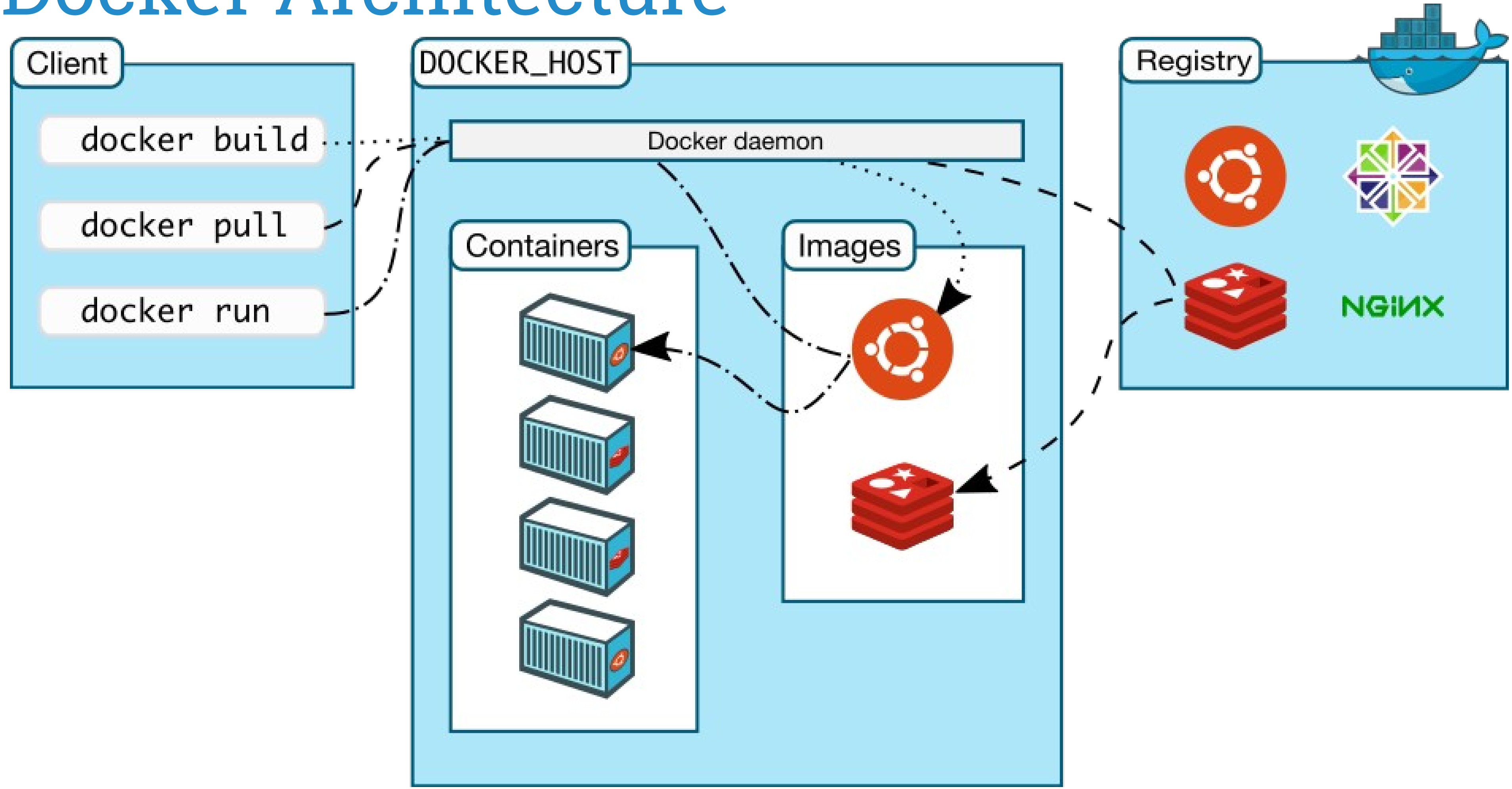
app

lib

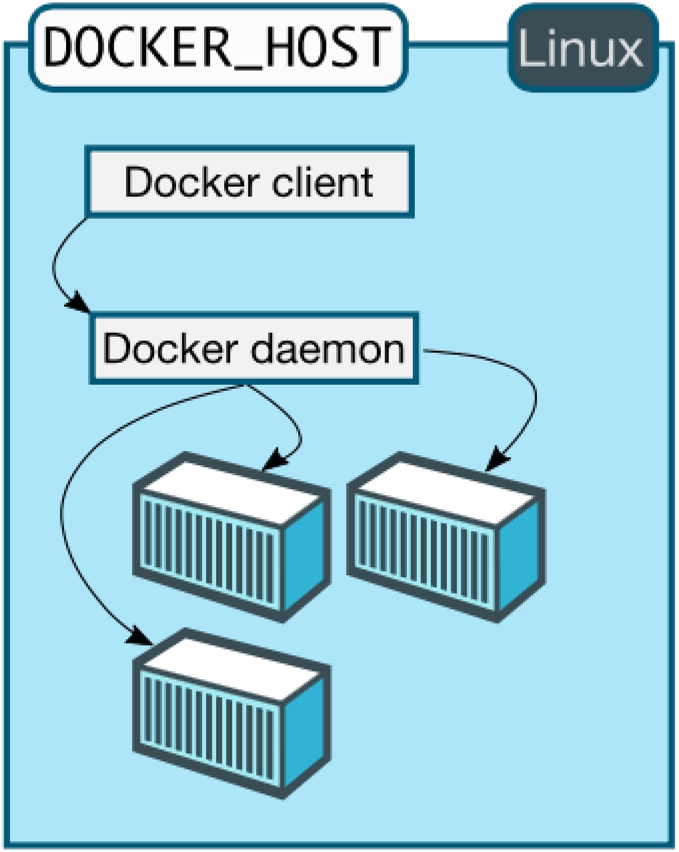
os

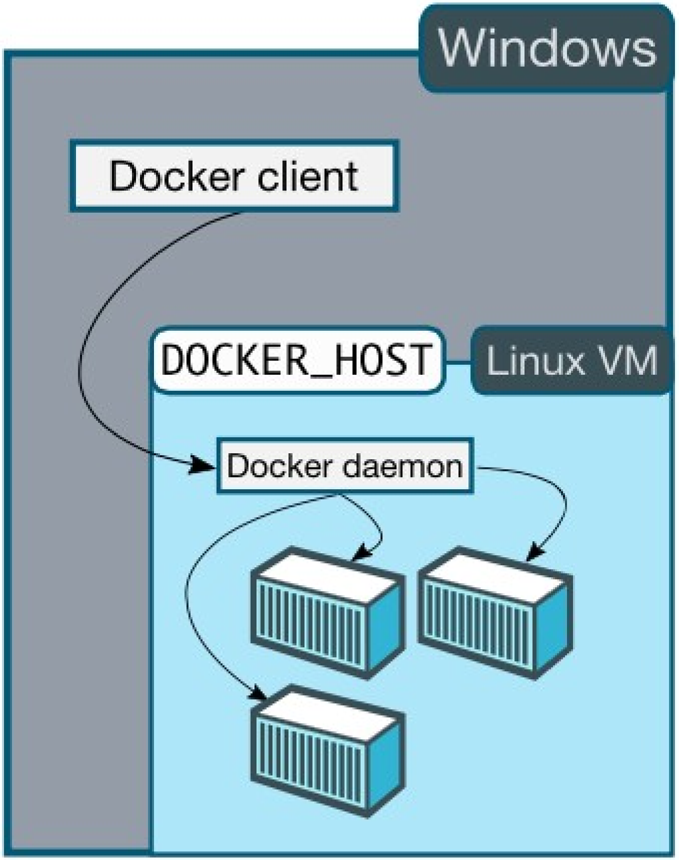
Hardware

Docker Architecture



Docker Architecture on Windows



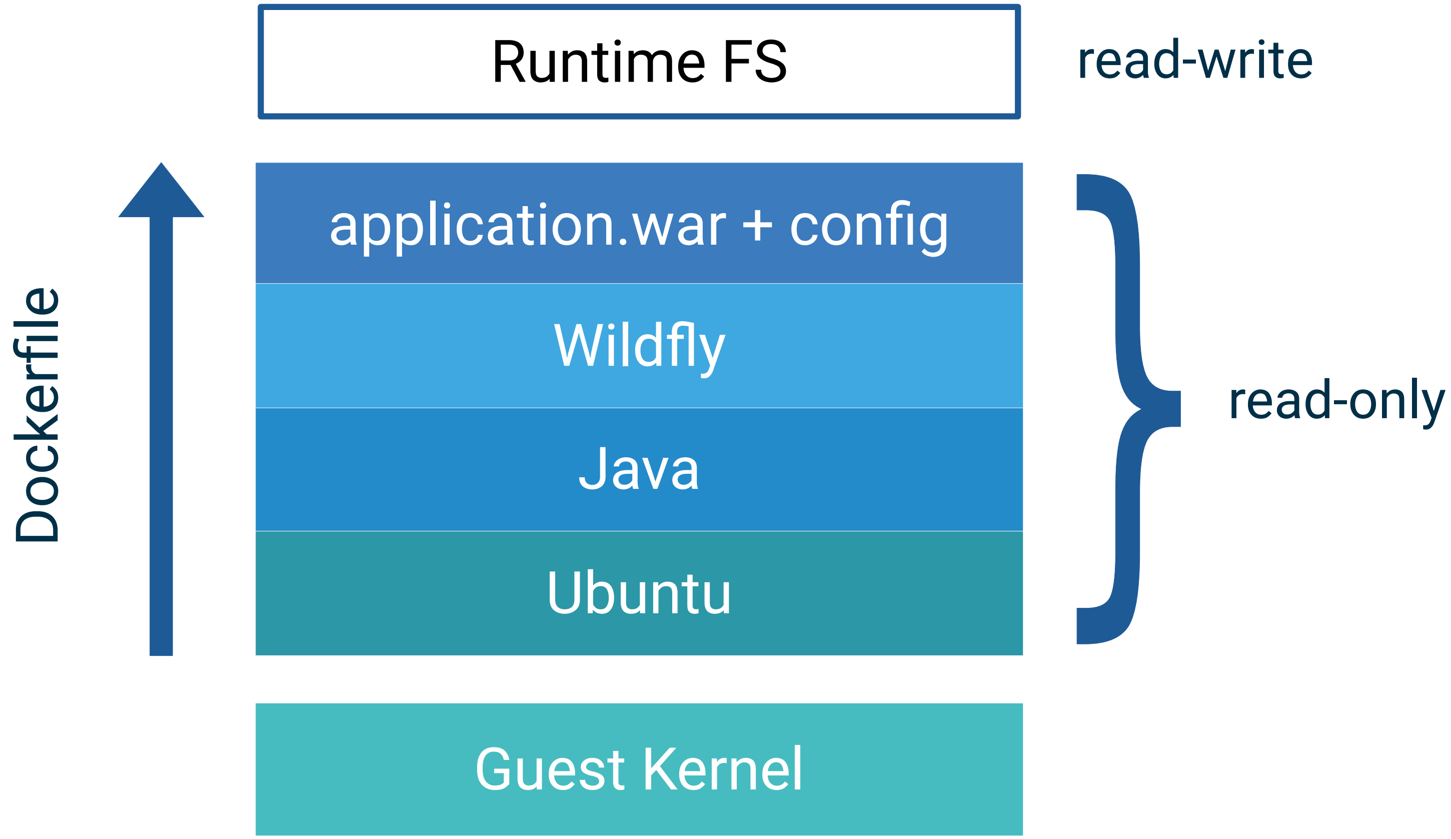


Docker Hub

- Public registry
- Lots of (official) container images
- Build platform for your own images
- Auto build and integration into GitHub
- ...

Behind the scenes

- Docker is implemented in GO
- Namespaces (visibility)
- Control Groups (resources)
- Union File System


Runtime FS
read-write
Dockerfile
application.war + config
Wildfly
Java
Ubuntu
read-only
Guest Kernel

Container Concepts

- Containers are immutable
- Update a container by replacing it
  - applies to applications and system patches
- No local filesystem for data
- 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 image
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"]
```

Dockerfile Best Practices (1/2)

- Containers should be stateless, state should only exist in mounted volumes
- One process per container
- Keep number of layers small
- Use a `.dockerignore` file
- Log to stdout → let platform handle aggregation and rotation

Dockerfile Best Practices (2/2)

- Only use trusted base images
- Automatic rebuild → security patches
- Sort multiline arguments:

```
RUN apt-get update && apt-get install -y \
  bzr \
  cvs \
  git \
  mercurial \
  subversion
```

Examples of Use

LaTex Rendering

Installation of a LaTex rendering infrastructure can be a pain (depending on OS)...

Put it into a container!

Standardization, traceability, installation documentation, fast

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

Build infrastructure

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

Build tools in different versions for varying projects have to be installed...

Put it in a container (-:

Define the exact development environment for each application.

Reusable, fast, isolated.

Java EE application with WildFly

Add your .war file and configuration into the image and let's go!

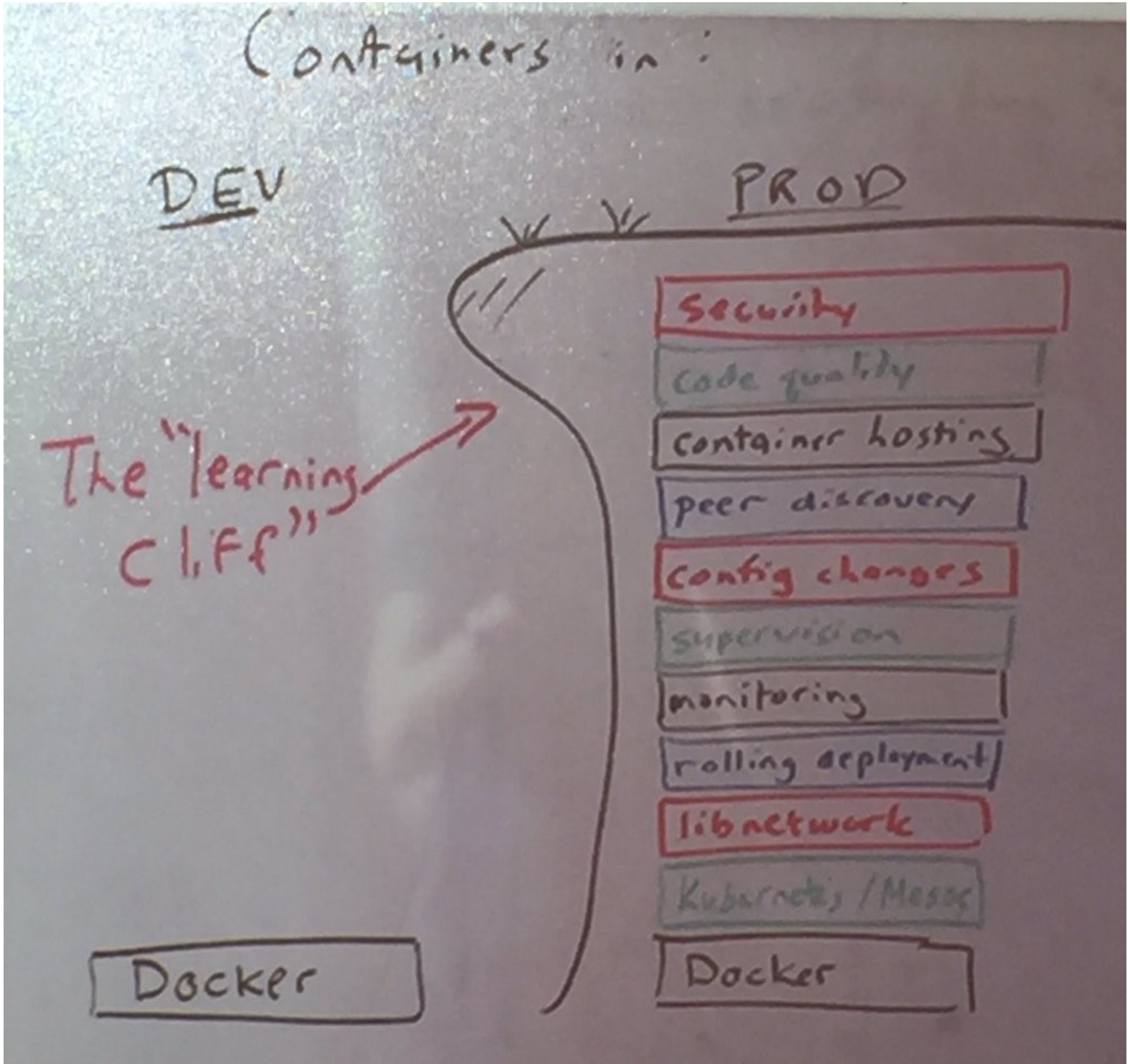
```
FROM jboss/wildfly

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

And so on…

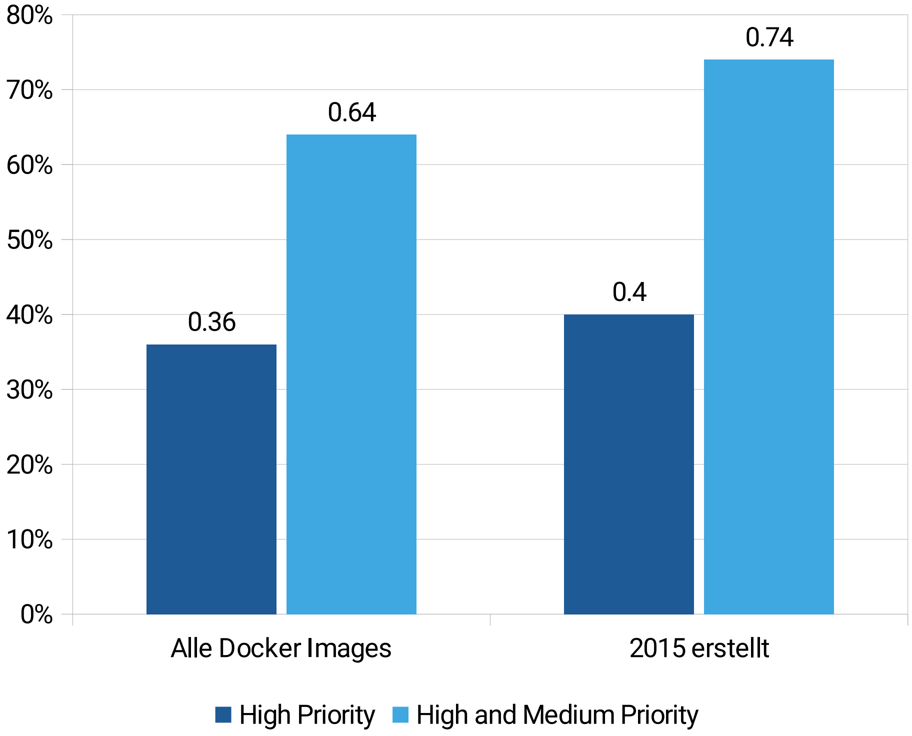
Chances and Challenges

Advantages

- Lightweight and fast
- Standardized
- Easy to use and extend
- Large community
- Lots of available images
- "Works on my machine" was yesterday


Containers in:
DEV
PROD
The "learning cliff"
security
code quality
container hosting
peer discovery
config changes
supervision
monitoring
rolling deployment
libnetwork
Kubernetes/Mesos
Docker
Docker

Challenges: Security



"Over 30% of Official Images in Docker Hub Contain High Priority Security Vulnerabilities":

ShellShock (Bash)

Heartbleed (OpenSSL)

Poodle (OpenSSL)

...

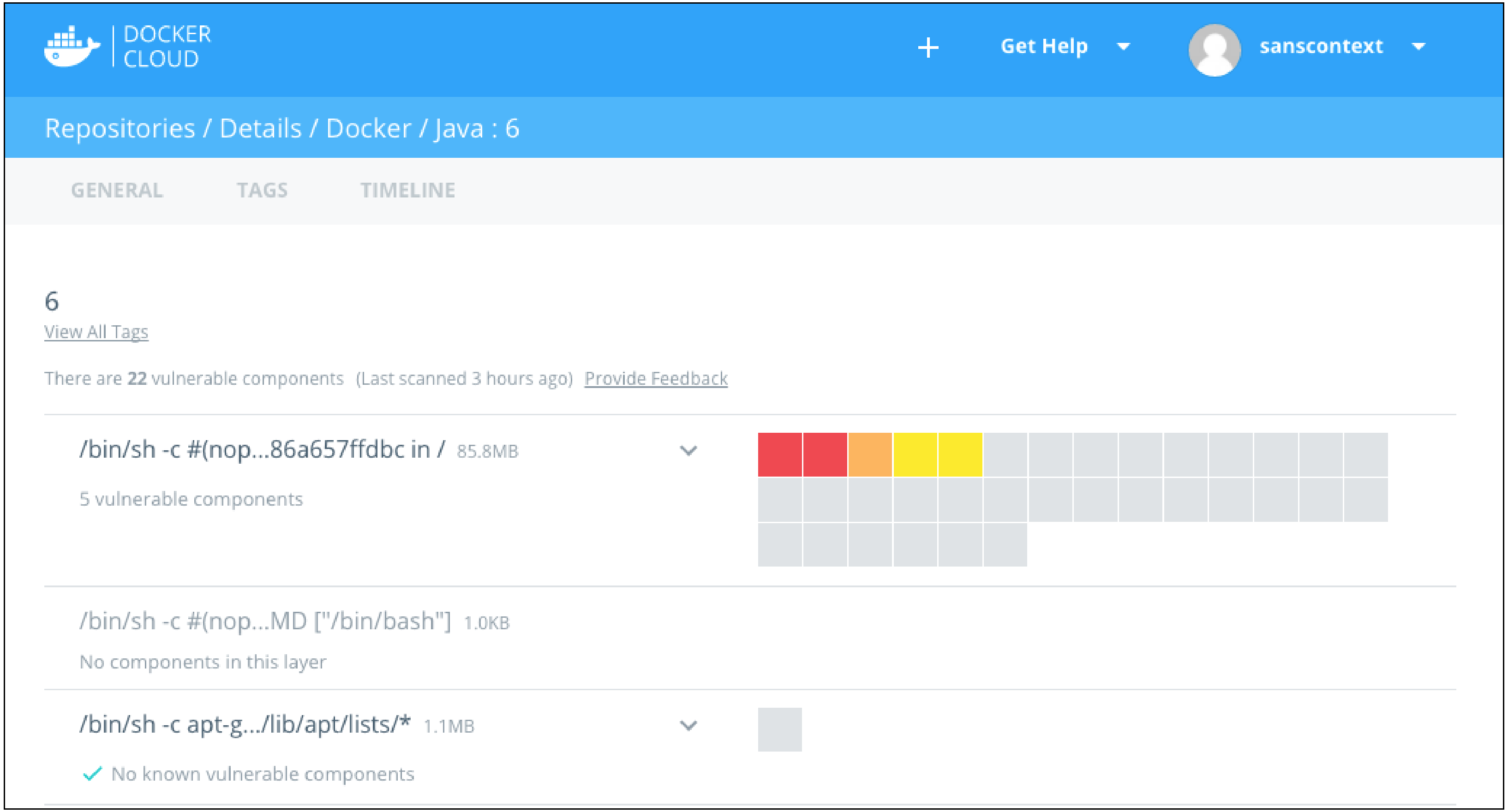
Source: http://www.banyanops.com/blog/analyzing-docker-hub/
May 2015: Jayanth Gummaraju, Tarun Desikan and Yoshio Turner

Challenges: Security

- Treat processes as if they were running on the host
- Do not let processes run as root
- Only open necessary ports

Quellen: https://opensource.com/business/14/7/docker-security-selinux
http://www.pro-linux.de/news/1/22315/docker-erh%C3%B6ht-die-sicherheit.html

Docker Security Scanning



Repositories / Details / Docker / Java : 6

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

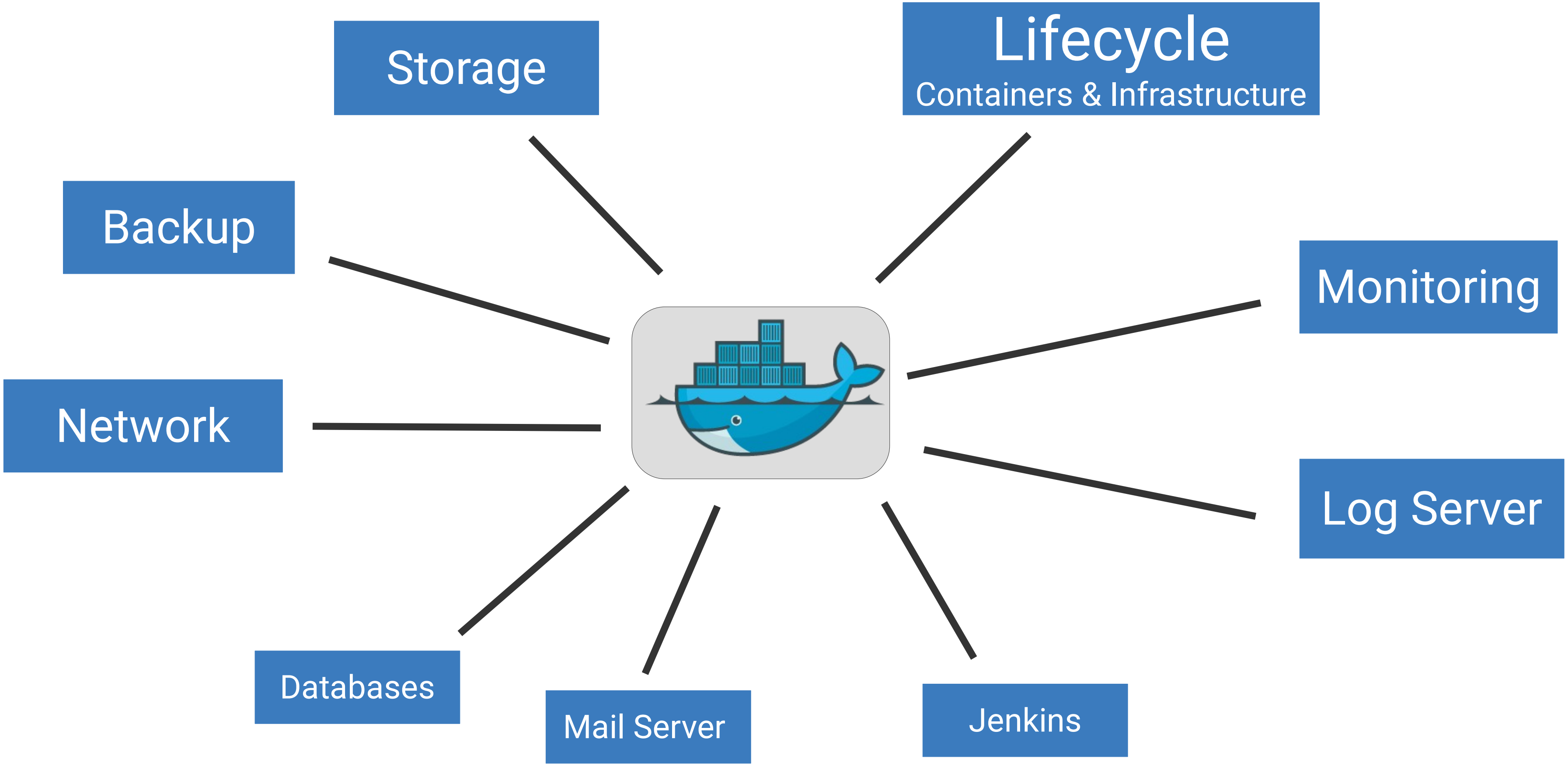
/bin/sh -c #(nop...MD ["/bin/bash"] 1.0KB

No components in this layer

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

✓ No known vulnerable components

Challenges: Operations



Storage

Lifecycle
Containers & Infrastructure

Backup

Monitoring

Network

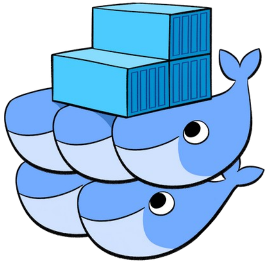
Log Server

Databases

Mail Server

Jenkins

Deployment and Orchestration



Thank you!