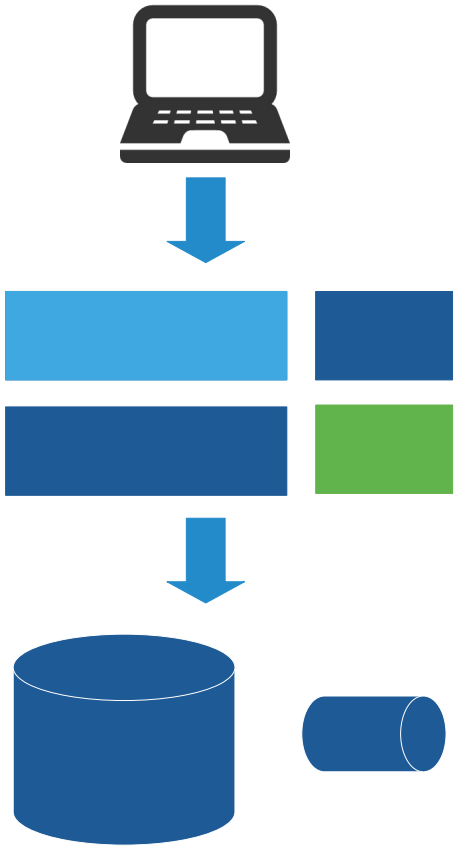
Container Basics

Agenda

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

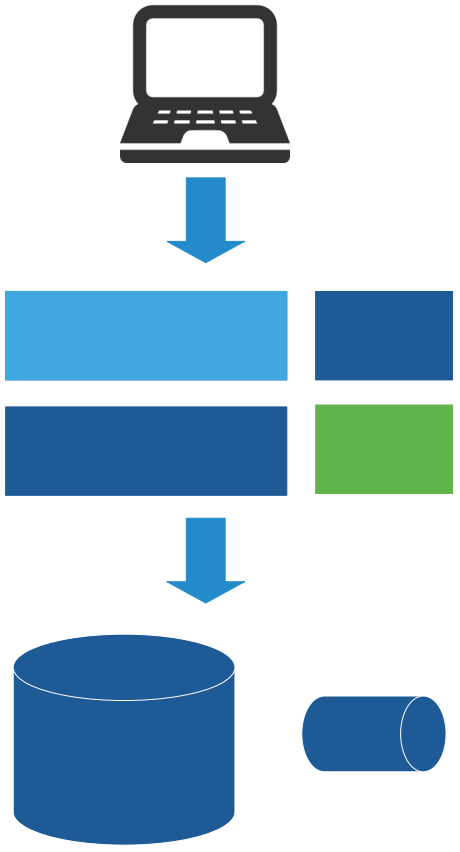
Developments in IT

1995

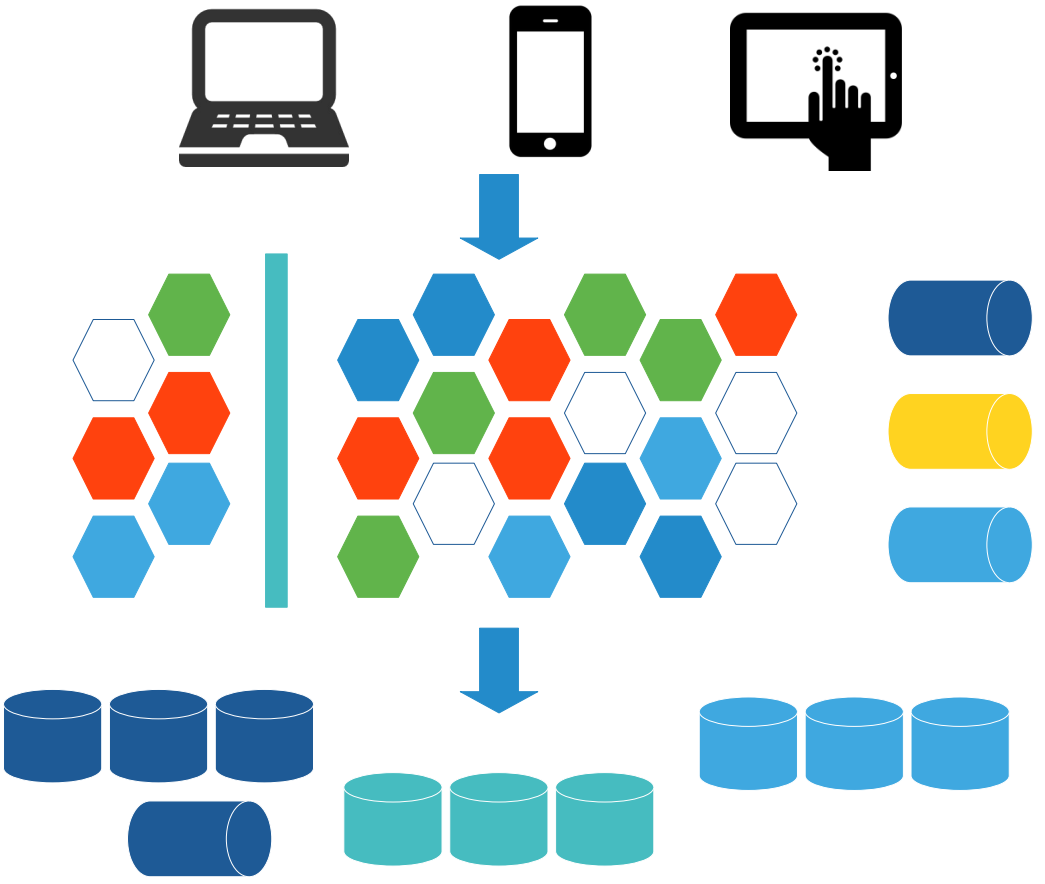


Developments in IT

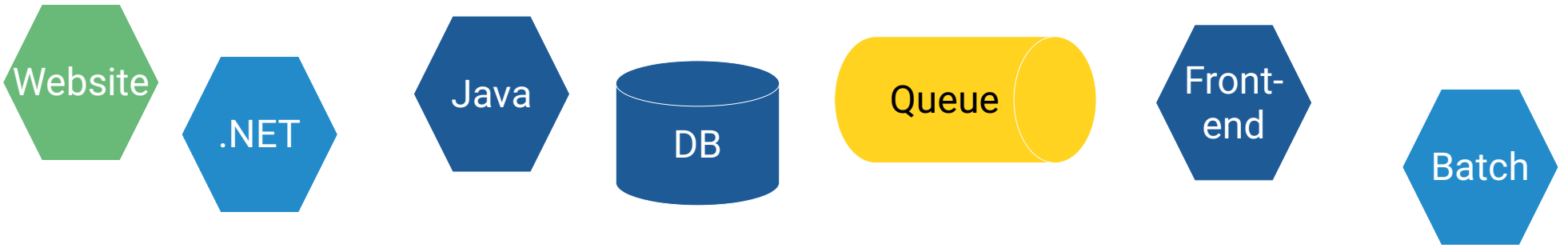
1995



today



Plethora of Combinations



Large number of services

Interactions between services

Large number of environments

Fast and easy migration

Matrix from Hell

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

Goods Traffic before 1960

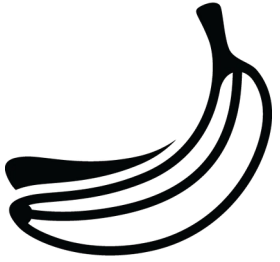

A 164


A 157

Plethora of Combinations

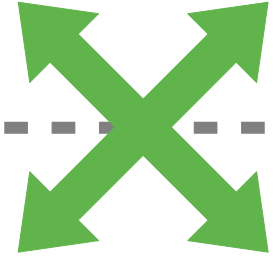






Large number of goods

Interaction between goods



Large number of routes

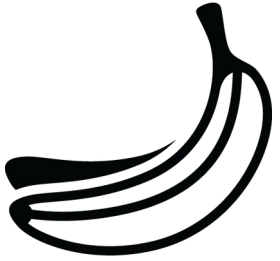
Fast and smooth transport




Solution





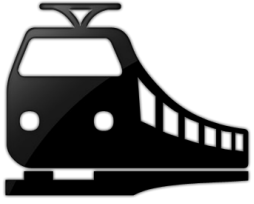


Large number of goods

Interaction between goods

Large number of routes

Fast and smooth transport




Applications in Containers

Website

.NET

Java

DB

Queue

Front-end

Batch

Large number of services

Large number of environments

Interactions between services

Fast and easy migration

Local

UAT

Prod

Pre Prod

Dev

Matrix Reloaded

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

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

Hello World

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

What happend?

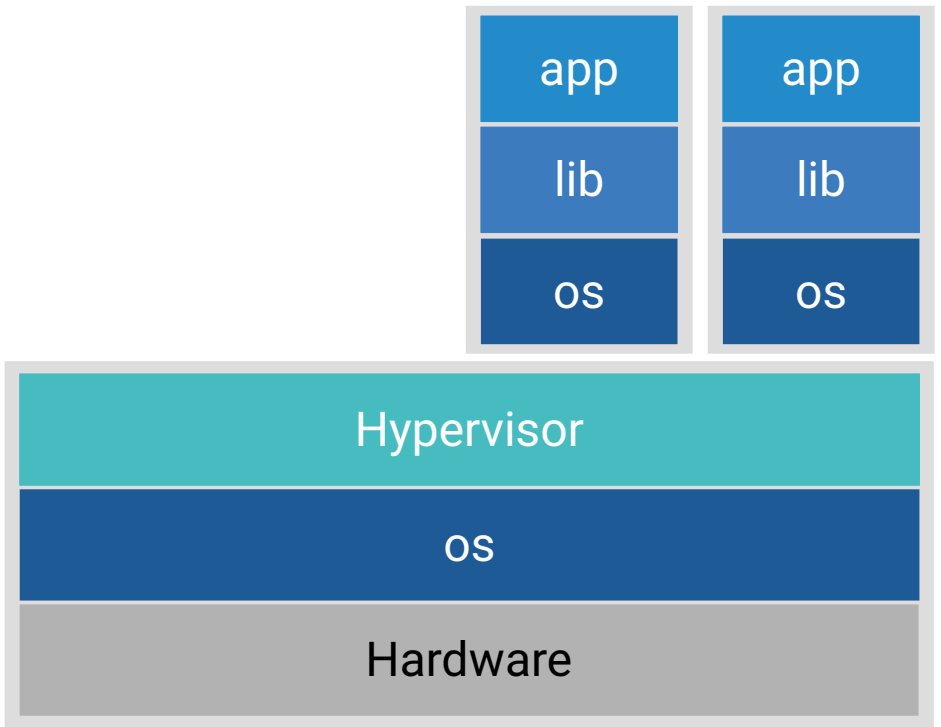
- Container start
- Allocation of filesystem
- Mount of a read/write filesystem layer
- Attachment of networking layer
- Execution of `echo` command
- Output to my console
- Container stop

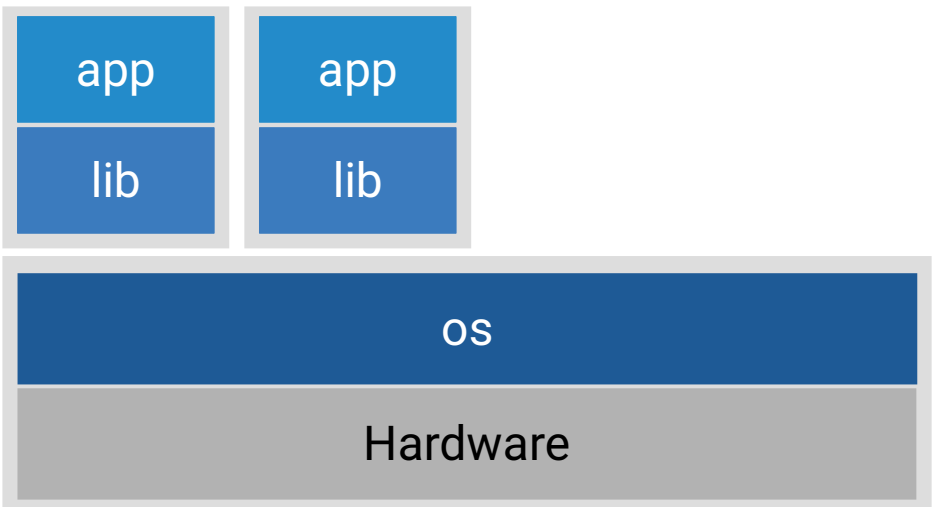
What is a Container?

Nothing new!

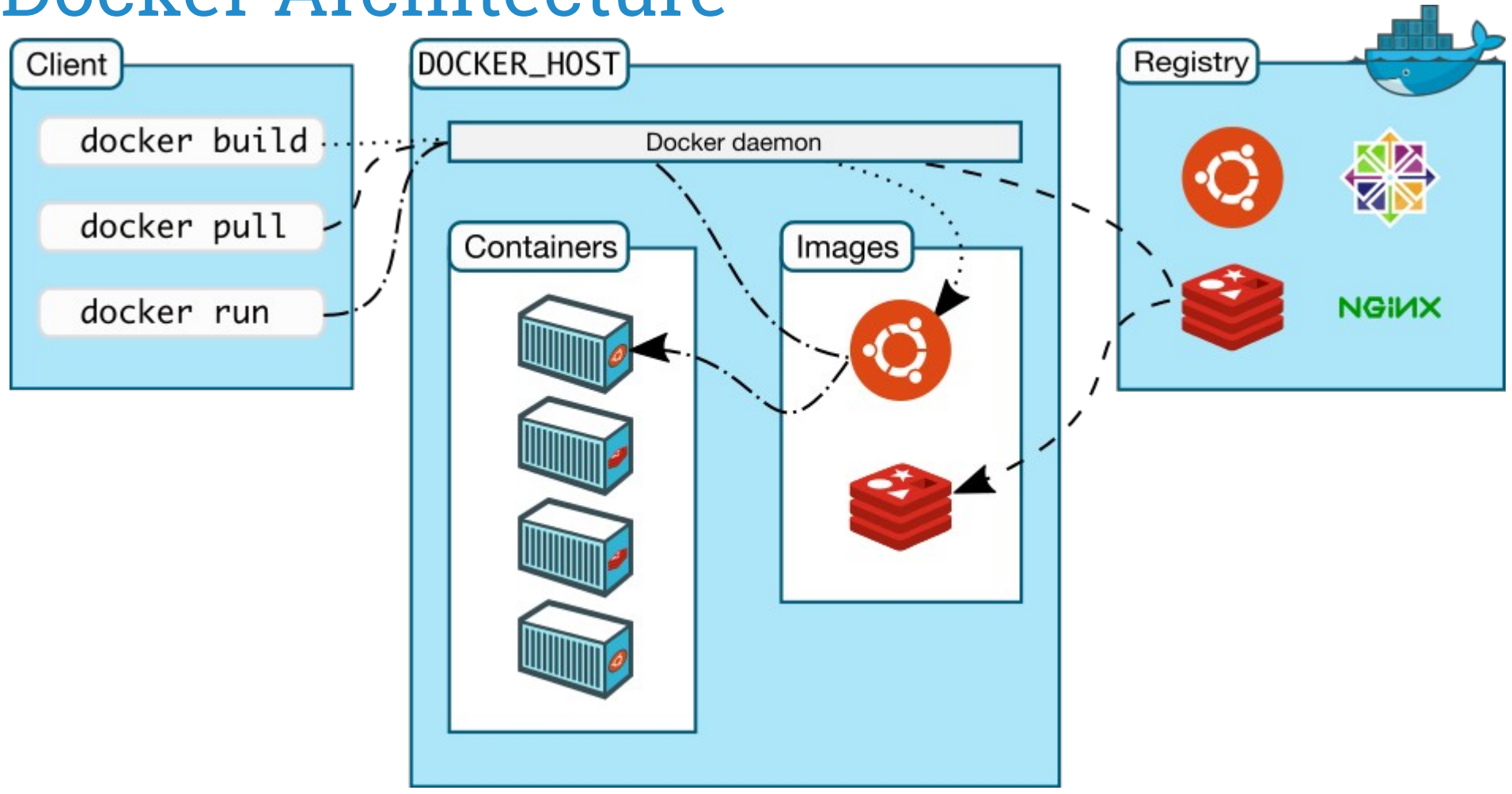
LXC, VServer, Free BSD Jails, Google...

Virtualiziation vs. Containerization

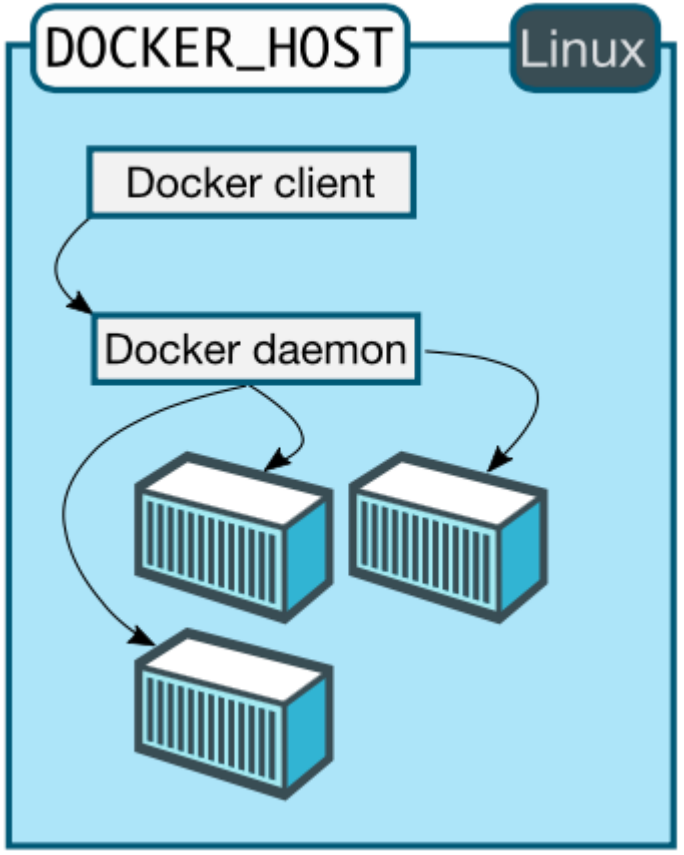


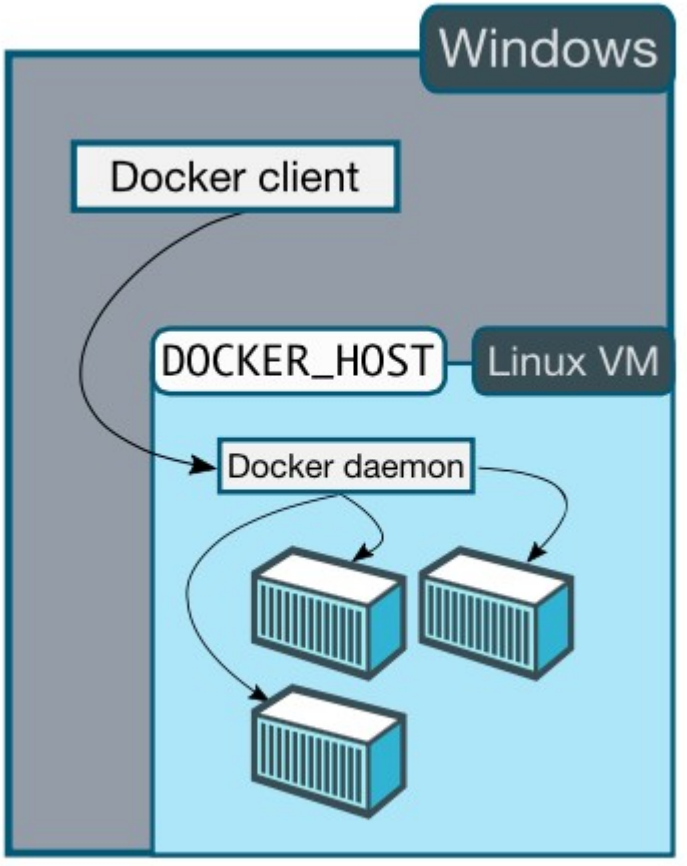


Docker Architecture



Docker Architecture on Windows



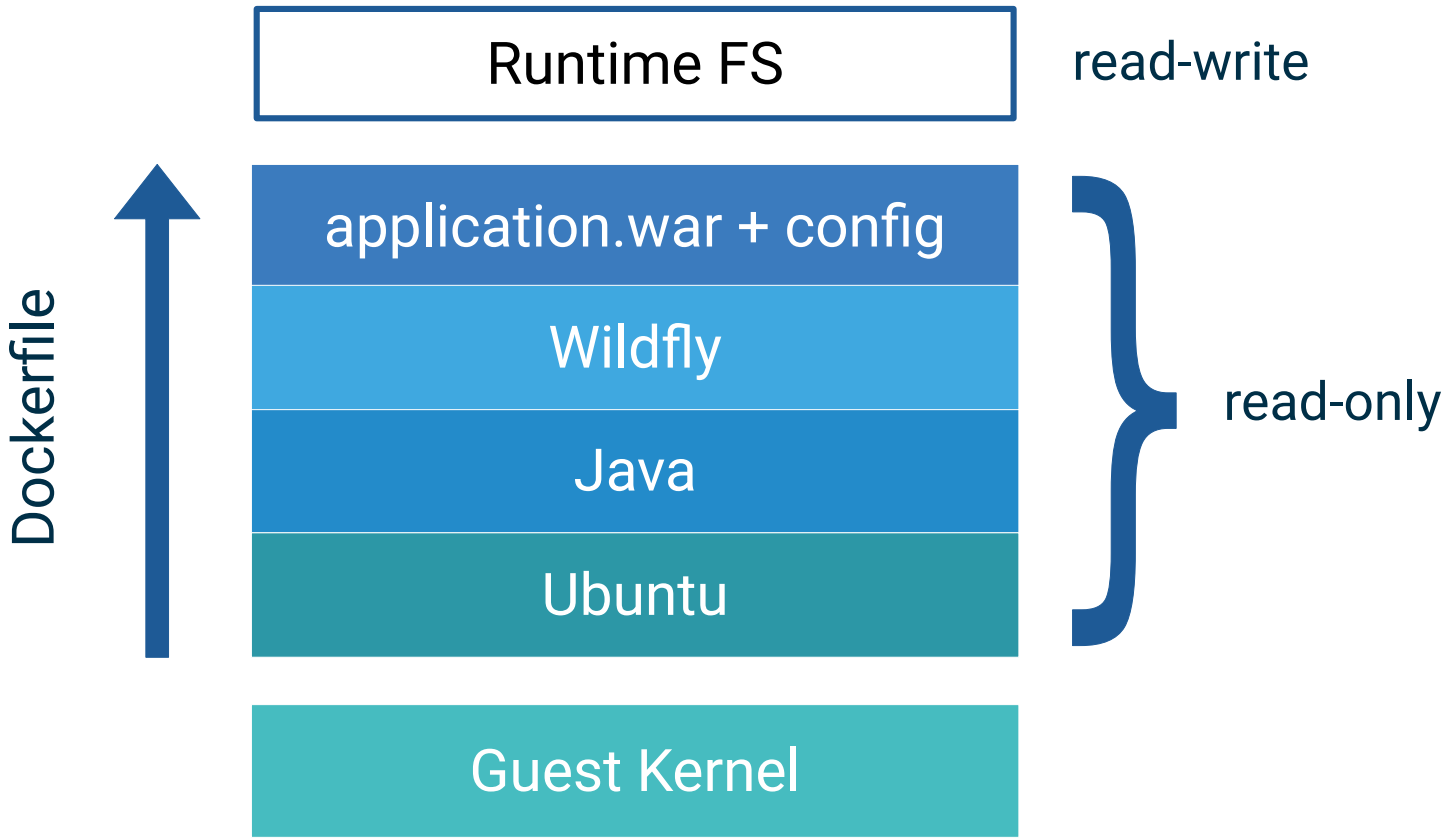


Docker Hub

- 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
application.war + config
Wildfly
Java
Ubuntu
read-only
Dockerfile
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

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!

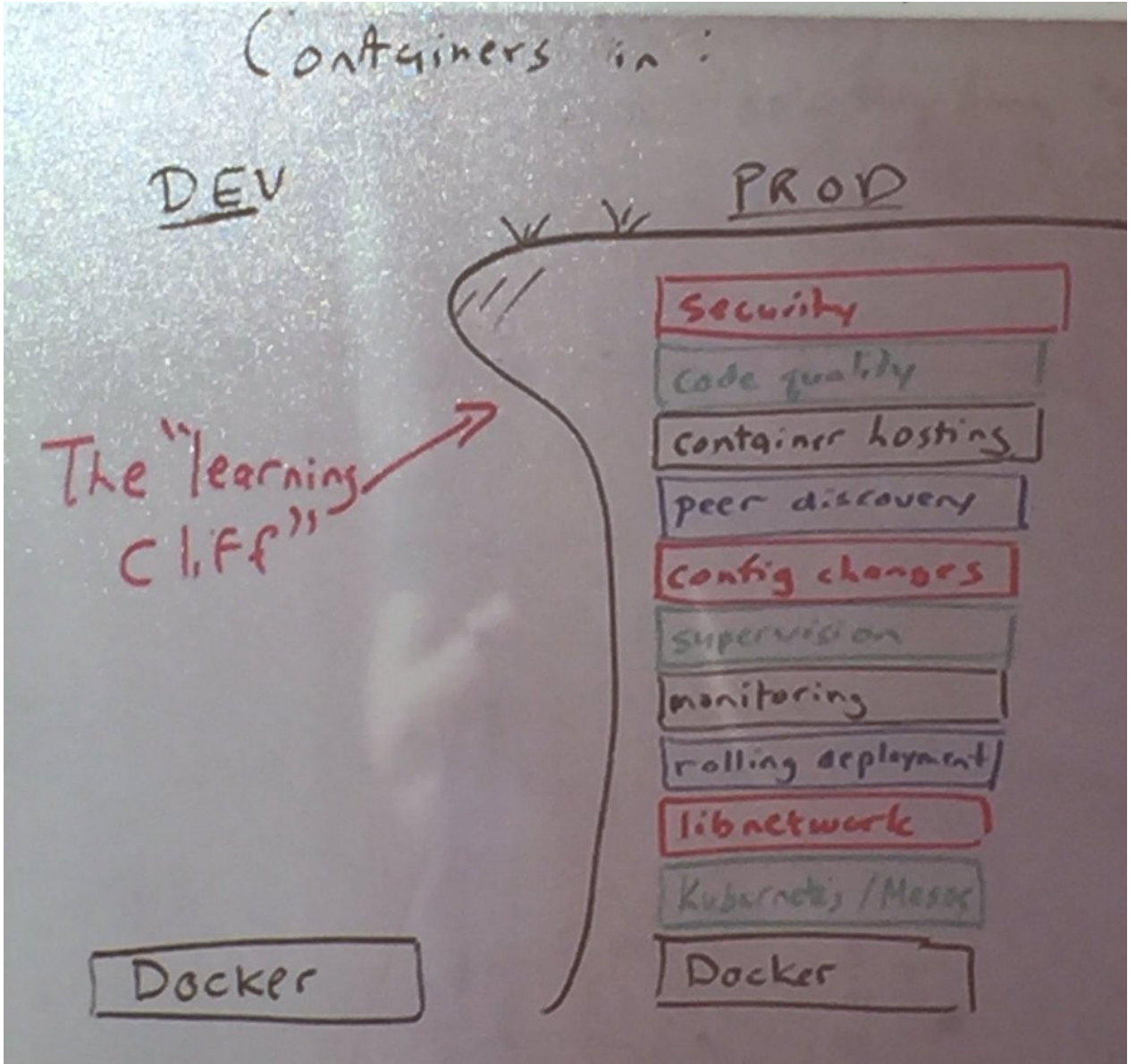
```
FROM jboss/wildfly

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 changers
supervision
monitoring
rolling deployment
libnetwork
Kubernetes / Mesos
Docker
Docker

Challenges: Security

Vulnerability Analysis of 2500 Docker Hub Images:

- Number of newly introduced vulnerabilities is rapidly increasing
- Certified images are the most vulnerable
- Official images are the least vulnerable (still 42% are)
- The most severe vulnerabilities originate from two of the most popular scripting languages, JavaScript and Python

Source: https://arxiv.org/pdf/2006.02932.pdf
June 11, 2020: Katrine Wist, Malene Helsem, Danilo Gligoroski

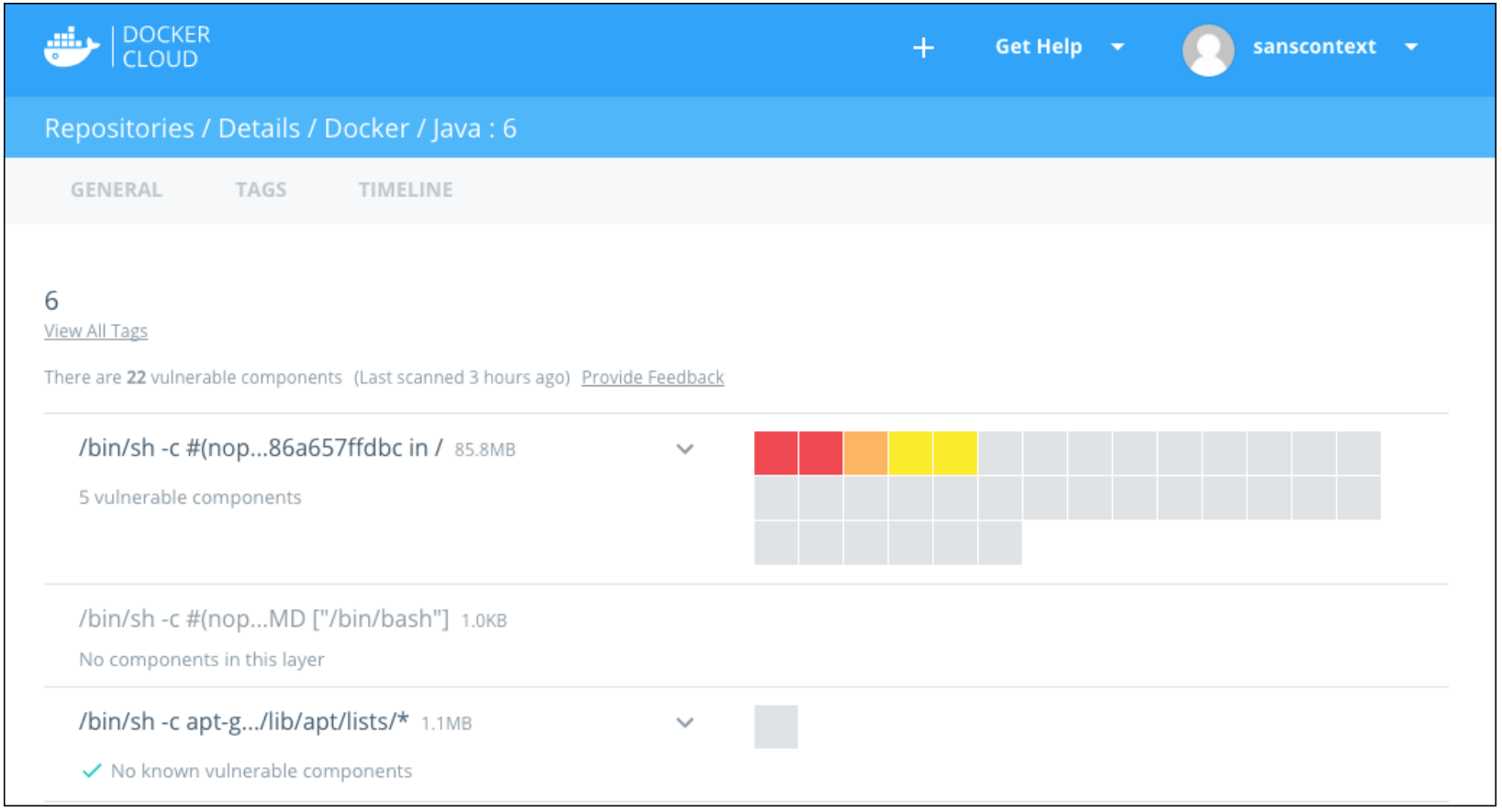
Challenges: Security

How to minimize risk:

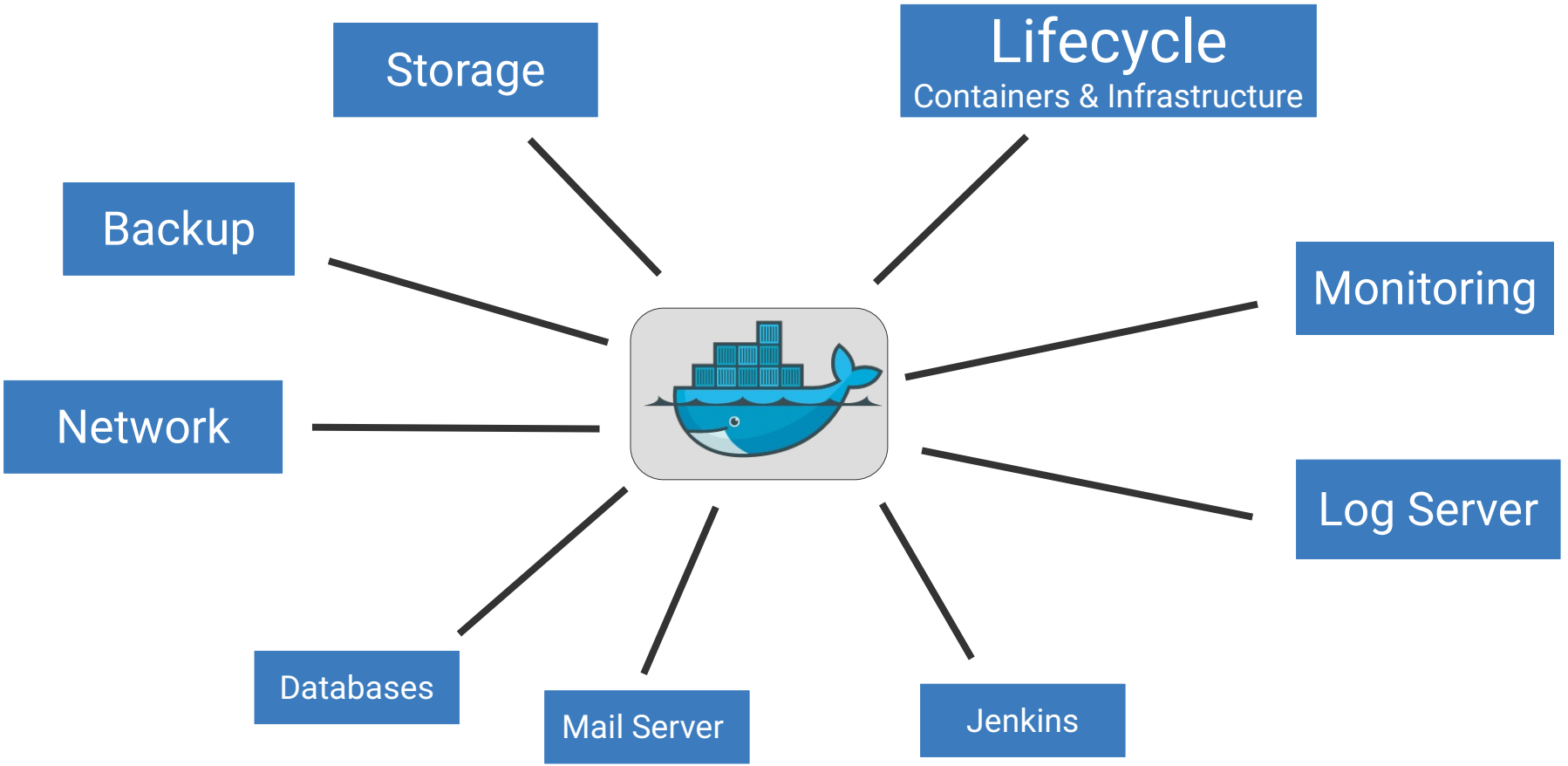
- Treat processes as if they were running on the host
- Do not let processes run as root
- Only open necessary ports
- Build automatically and often

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

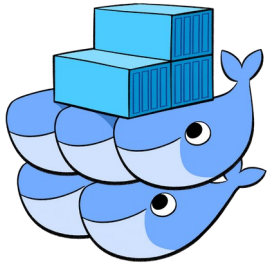
Docker Security Scanning



Challenges: Operations

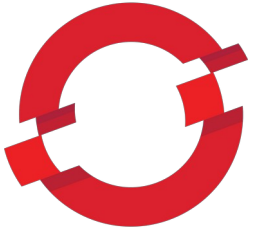


Deployment and Orchestration



MESOS

RANCHER



OPENSHIFT

Thank you!