DOCKER, TOMCAT AND MICROSERVICES

- @PRossbach

DEV OPS

<=>

YOU BUILD IT

YOU RUN IT

YOU TEST IT

YOU AUTOMATE IT

YOU MEASURE IT

YOU CHANGE IT

YOU SHIP IT

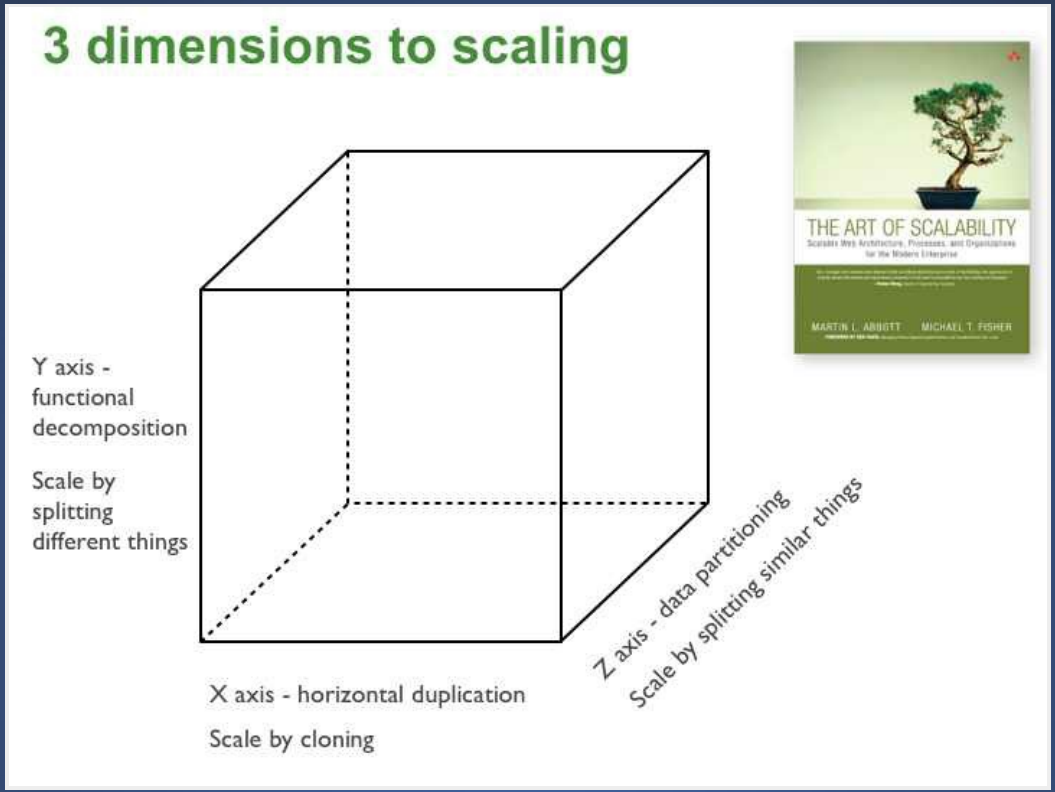
Cloud Ready

MICROSERVICES

- Microservice means
  - developing a single,
  - small,
  - meaningful functional feature as single service.
- Each service has its own process
  - and communicates with lightweight mechanism,
  - deployed in single or multiple servers.
- Each service manages its own data exclusively.

Microservice Architecture – A Quick Guide

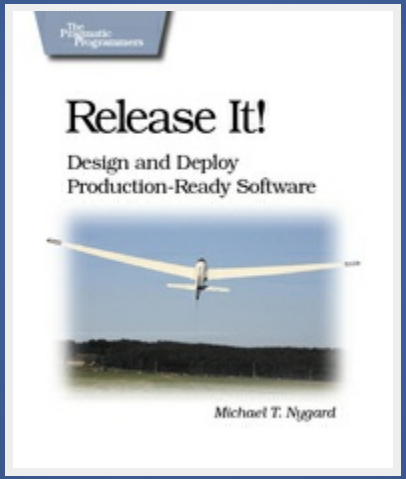
SCALE CUBE



The Art of Scalability

RESILIENT PATTERNS

- Netflix OSS
- MICHAEL NYGARD - Release IT
- Chaos Monkey



THE TWELVE FACTORS --I--

- I. Codebase
  - One codebase tracked in revision control, many deploys
- II. Dependencies
  - Explicitly declare and isolate dependencies
- III. Config
  - Store config in the environment
- IV. Backing Services
  - Treat backing services as attached resources
- V. Build, release, run
  - Strictly separate build and run stages
- VI. Processes
  - Execute the app as one or more stateless processes

THE TWELVE FACTORS --II--

- VII. Port binding
  - Export services via port binding
- VIII. Concurrency
  - Scale out via the process model
- IX. Disposability
  - Maximize robustness with fast startup and graceful shutdown
- X. Dev/prod parity
  - Keep development, staging, and production as similar as possible
- XI. Logs
  - Treat logs as event streams
- XII. Admin processes
  - Run admin/management tasks as one-off processes

The Twelve Factors

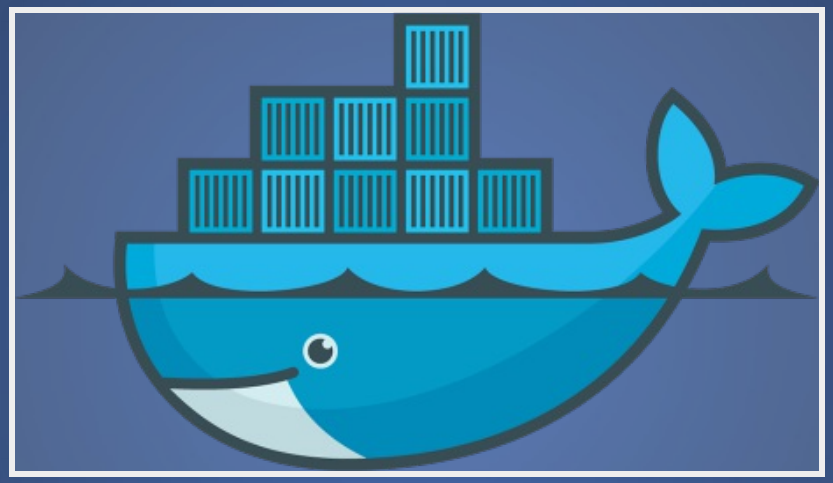
WHAT WE REALLY WANT?
BUILD,
SHIP AND RUN
SOFTWARE EASY PEASY!


HTML
CSS
JS
Restful
Apache
Tomcat
maven
boot2docker
docker
ant
inovex


kubernetes
bee42
Google
VAGRANT

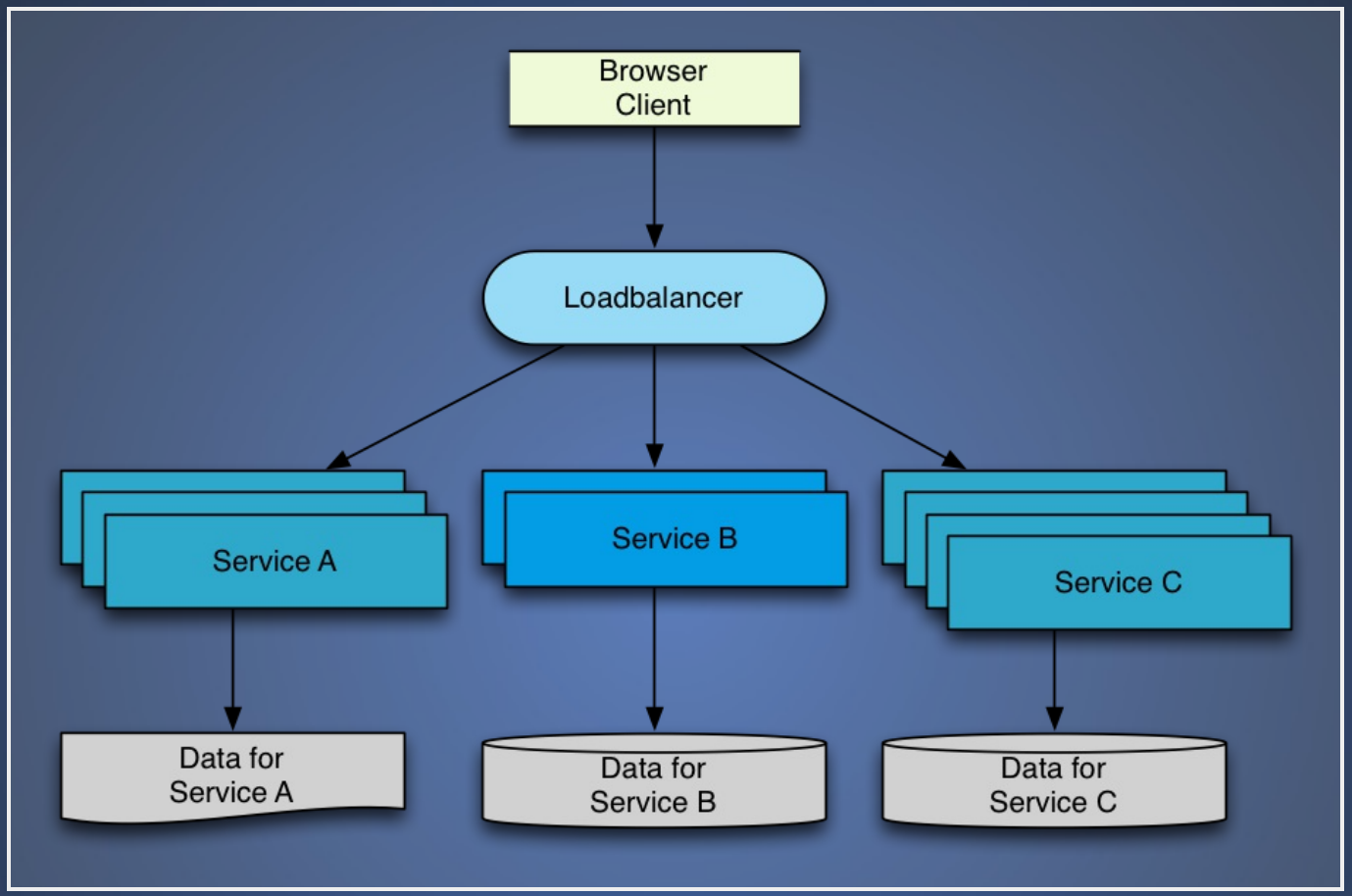
THE DOCKER MANTRA!

- Build, Ship and Run
- Any App,
- Anywhere



Perfect!

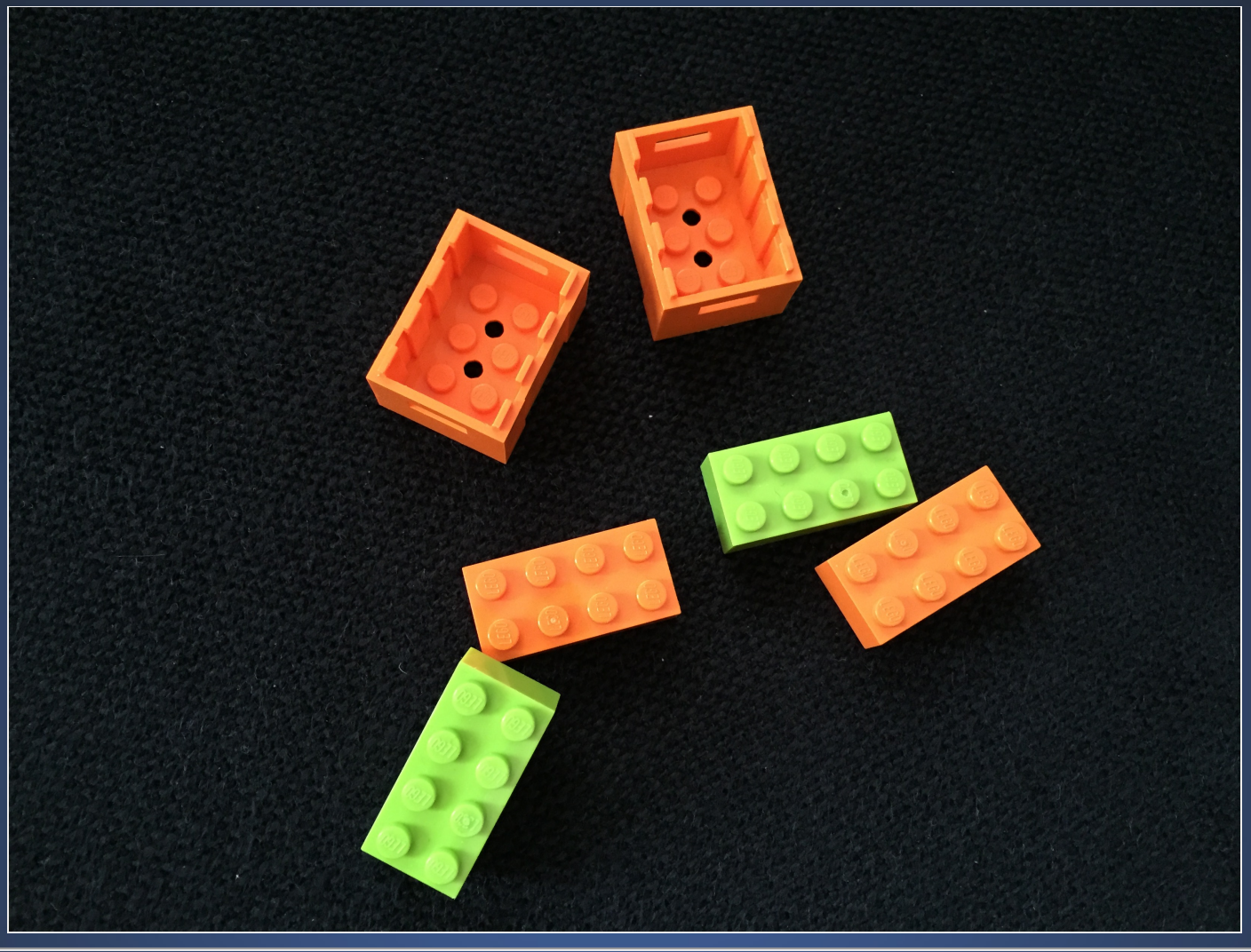
A SIMPLE MICROSERVICE ARCHITECTURE



EXPERIMENT...

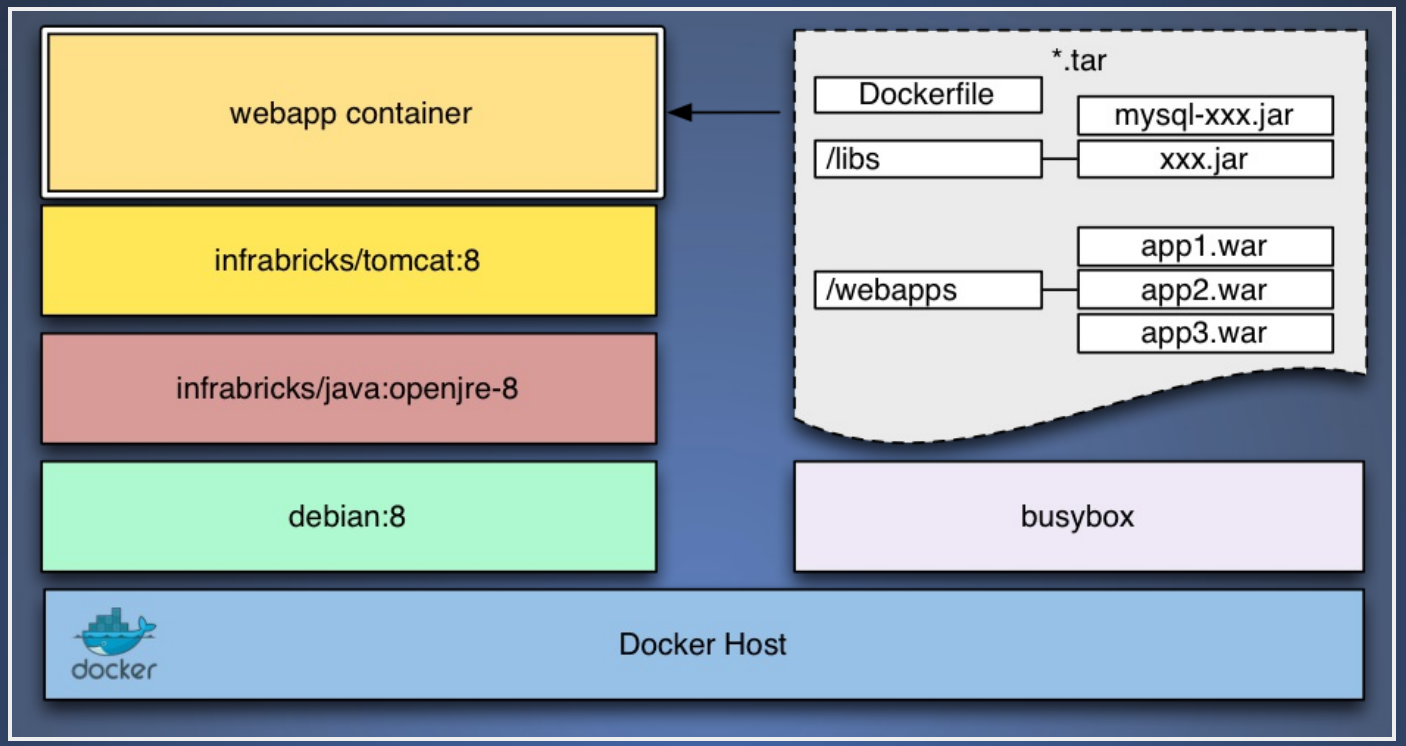
OK, START WITH A SIMPLE PLAN...

- create a Apache Tomcat based microservice
  - `hello world service ABC`
- use as loadbalancer => Nginx
- dockerized it
- scaling possible
- auto config possible => Service Discovery
- Environment setup needs time, **REALLY**!



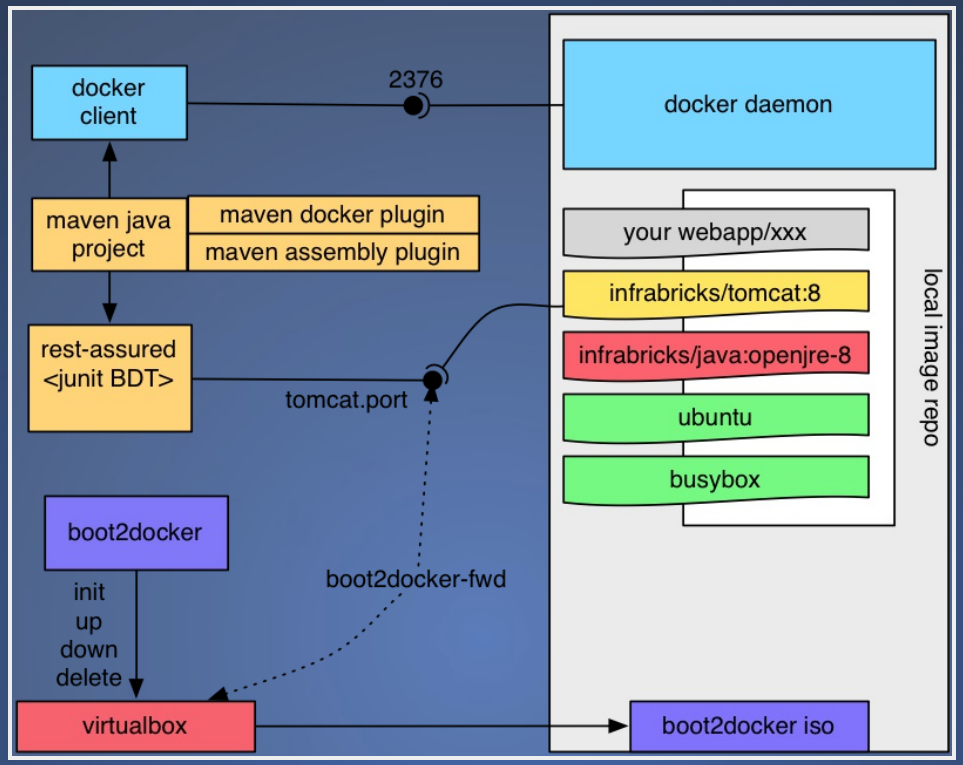
only some new bricks are available

DESIGN OF MY TOMCAT8 DOCKER IMAGE



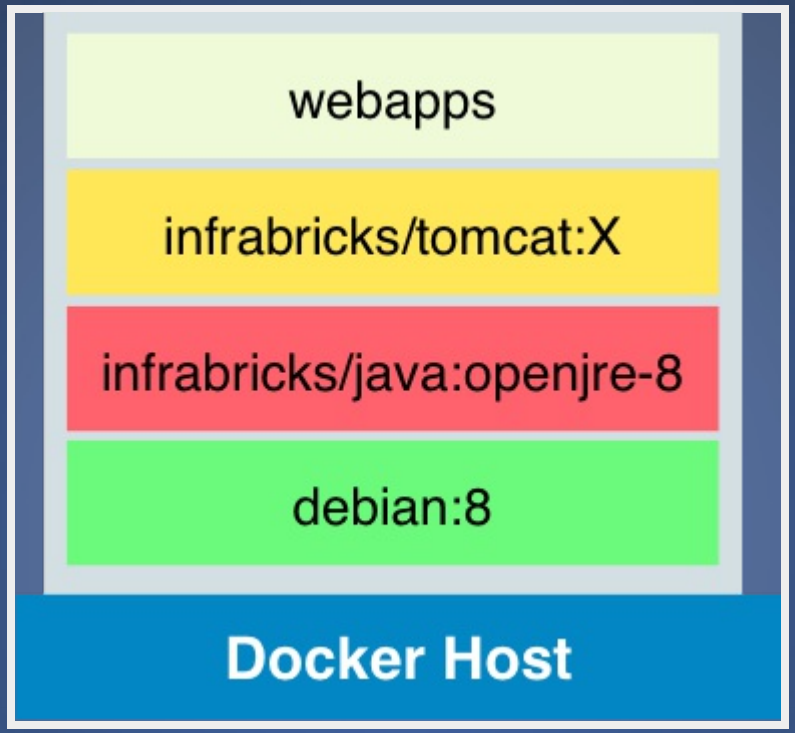
- Dockerbox tomcat 8 project
- More as we need!

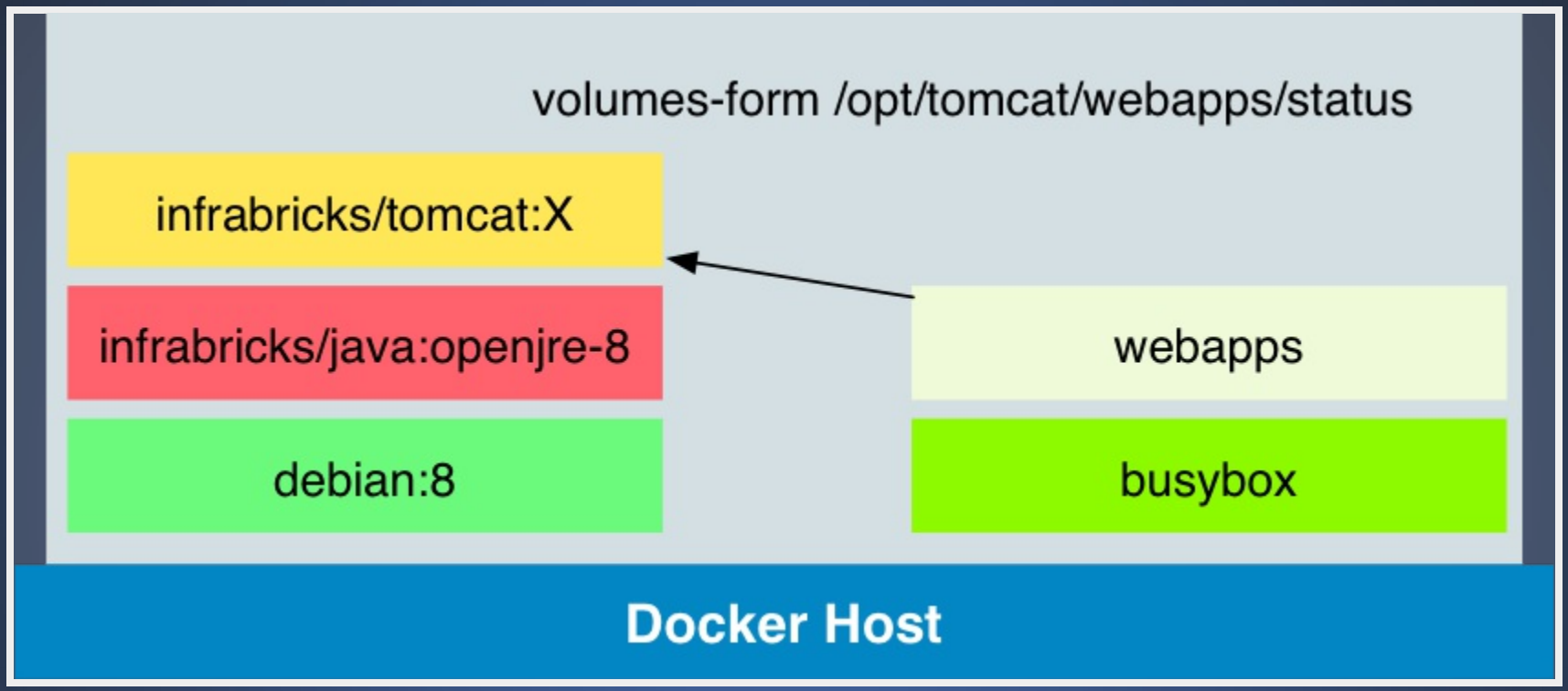
DOCKER MAVEN PLUGIN



- rhuss/docker-maven-plugin
- Docker für Java Entwickler - WJAX von Roland Huss 6.11.2014
- Much more as we need to start!

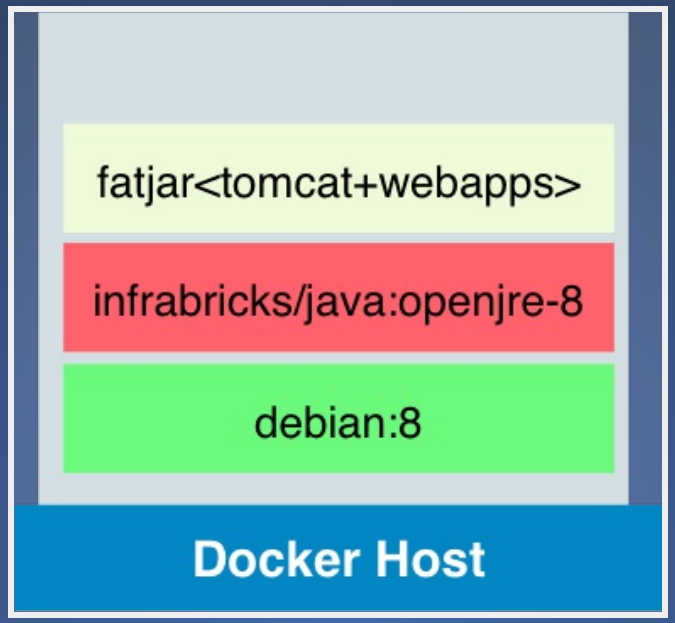
DEVELOPMENT WAR ON TOP





DEVELOPMENT WAR AS VOLUMES

DEVELOPMENT AS FATJAR



- We choose the latest one!
- Use Apache Tomcat Embeded
  - Lesser is more!

BUILD IT

CREATE A MAVEN BUILD ENV

- create an app main project
- add the frameworks
  - Embeded Tomcat 8
  - Jersey 2
  - Metrics
  - Couchdb
- Implement the simple rest service
- Some maven plugins must be used
  - create a fat jar
  - start for debugging a local Tomcat
- Tunning

BUILD IT

CODE REVIEW!

- Tomcat build
- Jersey integration
- Ressources created

EMBEDED TOMCAT MAIN

```
public void start()
  throws ServletException, LifecycleException,MalformedURLException
 {
  String webappDirLocation = ".";
  Tomcat tomcat = new Tomcat();
  String webPort = getConfigParameter(CONFIG_SERVICE_PORT, "8080");
  tomcat.setPort(Integer.valueOf(webPort));

  Context context = tomcat.addWebapp("/hello", new File(
  webappDirLocation).getAbsolutePath());
  Tomcat.addServlet(context, "jersey-container-servlet", resourceConfig());
  context.addServletMapping("/rest/*", "jersey-container-servlet");

  AccessLogStdoutValve valve = new AccessLogStdoutValve();
  String pattern = getConfigParameter(
    CONFIG_SERVICE_PATTERN, ACCESS_LOG_PATTERN);
  valve.setPattern(pattern);
  context.getPipeline().addValve(valve);

  tomcat.start();
  tomcat.getServer().await();
}
```

SHIP IT

DOCKERIZED!

- Choose a Docker Java Image
  - build my own one!
- packaging the results with Docker
- sometimes add a little magic

SIMPLE MICRO DOCKERFILE

```
FROM infrabricks/java:openjre-8
MAINTAINER Peter Rossbach <peter.rossbach@bee42.com>

RUN mkdir /opt/micro
COPY target/helloworld.jar /opt/micro/helloworld.jar
COPY micro.sh /opt/micro/micro.sh
COPY logging.properties /opt/micro/logging.properties

WORKDIR /opt/micro
EXPOSE 8080

CMD [ "/bin/bash", "-c", "/opt/micro/micro.sh" ]
```

```
docker build -t="infrabricks/tomcat:8" .
```

BUILD IT

JAVA DOCKERFILE INFRABRICKS/JAVA:OPENJRE-8

```
FROM debian:8

MAINTAINER Peter Rossbach <peter.rossbach@bee42.com>

ENV DEBIAN_FRONTEND noninteractive

RUN \
  echo "deb http://ftp.us.debian.org/debian sid main" >> /etc/apt/sources.list \
  && apt-get update \
  && apt-get install -yq openjdk-8-jre-headless \
    wget\
    pwgen\
  && apt-get clean autoclean \
  && apt-get autoremove -y \
  && rm -rf /var/lib/{apt,dpkg,cache,log}/
...
```

```
docker build -t="infrabricks/java:openjre-8" .
```

MICRO.SH

```
#!/bin/bash
java -Duser.language=en\
 -Duser.country=US\
 -DCOUCHDB_HOST=couchdb \
 -Djava.util.logging.config.file=/opt/micro/logging.properties\
 -Djava.util.logging.SimpleFormatter.format='%1$tY-%1$tm-%1$tdT%1$tH:%
 -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager \
 -Djava.security.egd=file:/dev/./urandom\
 -jar helloworld.jar
```

JULI with single line output and ISO8601 timestamp

LOGGING.PROPERTIES

```
handlers = java.util.logging.ConsoleHandler
.handlers = java.util.logging.ConsoleHandler
.level=INFO


java.util.logging.ConsoleHandler.level = FINE
java.util.logging.ConsoleHandler.formatter=java.util.logging.SimpleFormatt
java.util.logging.SimpleFormatter.format=%1$tY-%1$tm-%1$tdT%1$tH:%1$tM:%1$

org.apache.tomcat.util.digester.Digester.level=OFF
org.apache.jasper.level=INFO
org.apache.jasper.servlet.TldScanner.level=WARNING
org.apache.catalina.startup.ContextConfig.level=WARNING
```

- SimpleFormatter
- Formatter logging pattern syntax

BUILD IT
RUN IT
SHIP IT

```
$ mvn package
$ docker build -t="infrabricks/helloworld" .
$ docker run -dit -p 8000:8080 infrabricks/helloworld
$ curl -s http://192.168.59.103:8000/hello/rest/helloworld
Hello World from Tomcat Embeded with Jersey!
$ docker login
$ docker push infrabricks/helloworld
```

STATUS REPORT

```
$ curl -s http://192.168.59.103:8000/hello/rest/status
<html>
<body>
<h1>Docker Tomcat status page</h1>
<ul>
<li>Hostname : 5365ac28dc4b</li>
<li>Now : 2014/11/03 13:21:51</li>
</ul>
</body>
</html>
```

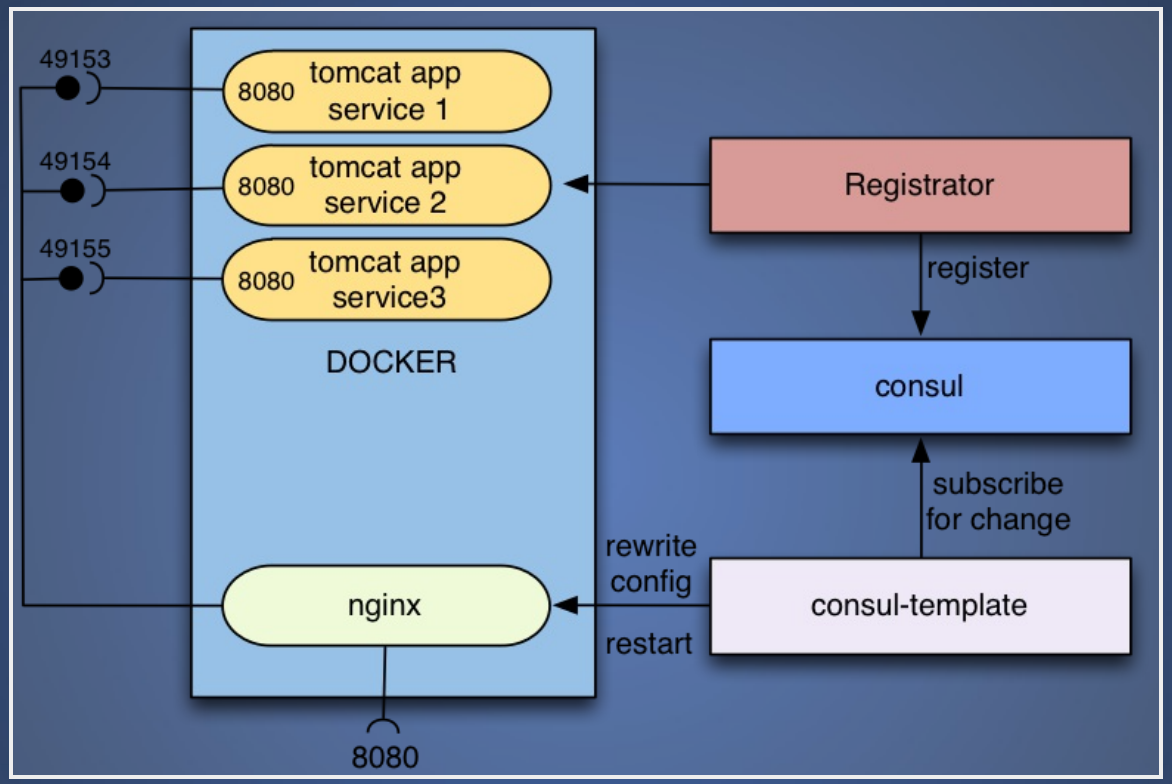
LOOK LATER INSIDE PROJECT SOURCE :)

Happy Hacking to add more...

- use resilient pattern - hystrix
- add monitoring
  - Coda Hale Metrics (hacked)
  - Jolokia
  - Statsd
- add Access Logging (hacked:)
  - stdout logging
- Source helloworld.tar.gz
  - `AccessLogStdoutValve` request logging to stdout
  - `OnelineConsoleReporter` send every 10 sec metrics data
  - CouchDB Access
  - setup details at README.md
- Read Properties from System.env

MICROSERVICE SCALING

AUTOSCALING AND FAILOVER



CONSUL - SERVICE DISCOVERY

- Distributed key value store
- Service Registry
- Multi-Datacenter Support
- ACL
- HTTP and DNS Discovery support
- Simple: curl'able user facing API (HTTP+JSON)
- Secure: optional SSL client cert authentication
- Reliable: properly distributed using Raft

- Written in go
- Use the RAFT Protocol
- Find more info at Consul website

REGISTRATOR

- Registrator automatically registers/deregisters services for Docker containers
  - published ports
  - published metadata from the container environment

- Blog explain registrator
- Github registrator
- ETCD

REGISTRATOR - EXAMPLE

```
$ docker run -d --name redis.0 -p 10000:6379 \
    -e "SERVICE_NAME=db" \
    -e "SERVICE_TAGS=master,backups" \
    -e "SERVICE_REGION=us2" dockerfile/redis
```

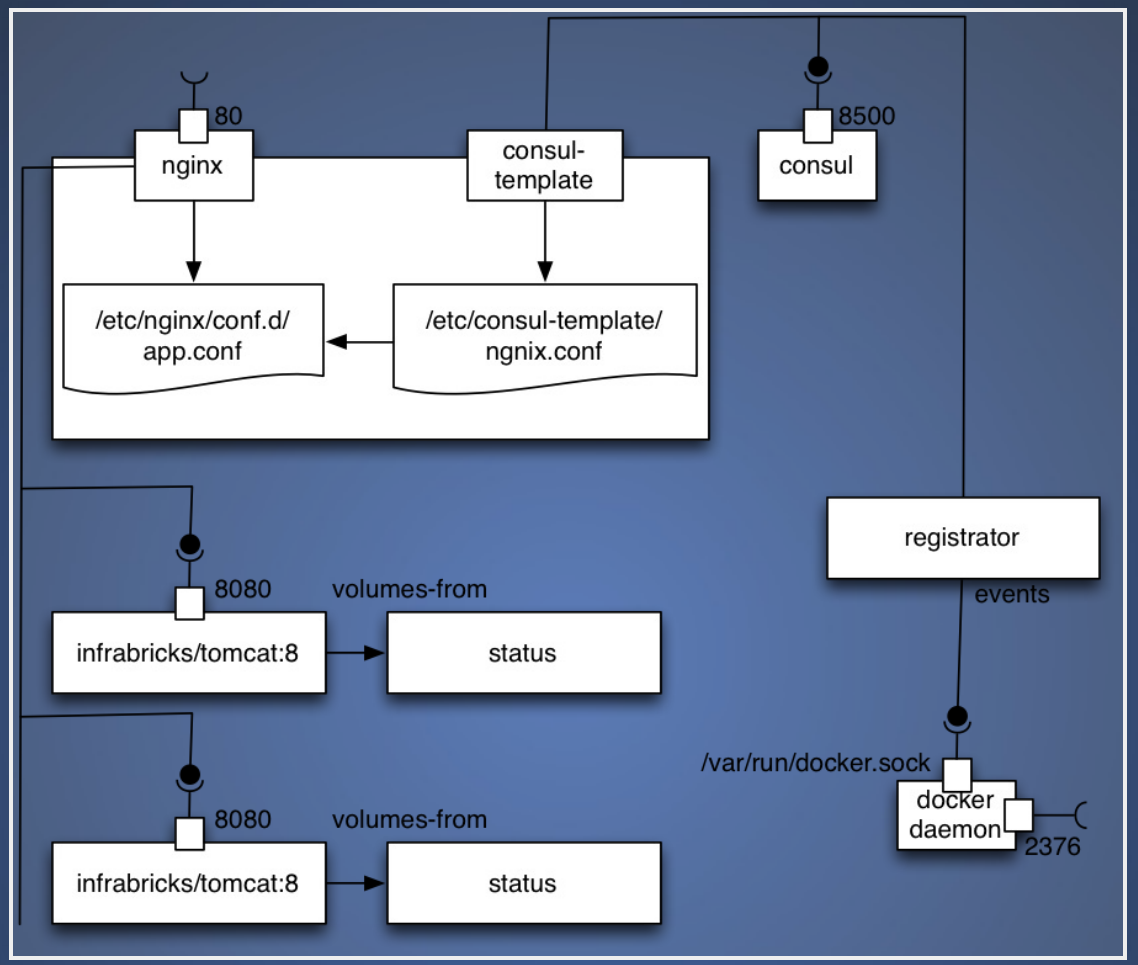
Results in a service:

```
{
    "ID": "hostname:redis.0:6379",
    "Name": "db",
    "Port": 10000,
    "IP": "192.168.1.102",
    "Tags": ["master", "backups"],
    "Attrs": {"region": "us2"}
}
```

DEMO TIME

- setup project
- start tomcat
- loadbalancer

DETAIL OVERVIEW



Build a new machine in a minute

```
docker-machine create -d virtualbox tomcat
```

START TOMCATS

```
cd docker-d/hello
docker-compose up -d
docker-compose scale tomcat=3
```

docker-compose.yml

```
status:
   build: .
tomcat:
  image: infrabricks/tomcat:8
  ports:
    - "8080"
  volumes_from:
    - status
  volumes:
    - tomcat-users.xml:/opt/tomcat/conf/tomcat-users.xml
  environment:
    constraint: zone==dev
    constraint: disk==ssd
    SERVICE_NAME: status
    SERVICE_TAGS: tomcat
    SERVICE_REGION: bee42.1
```

START SERVICE DISCOVERY

```
cd docker.d
docker-compose -f master.yml up -d
```

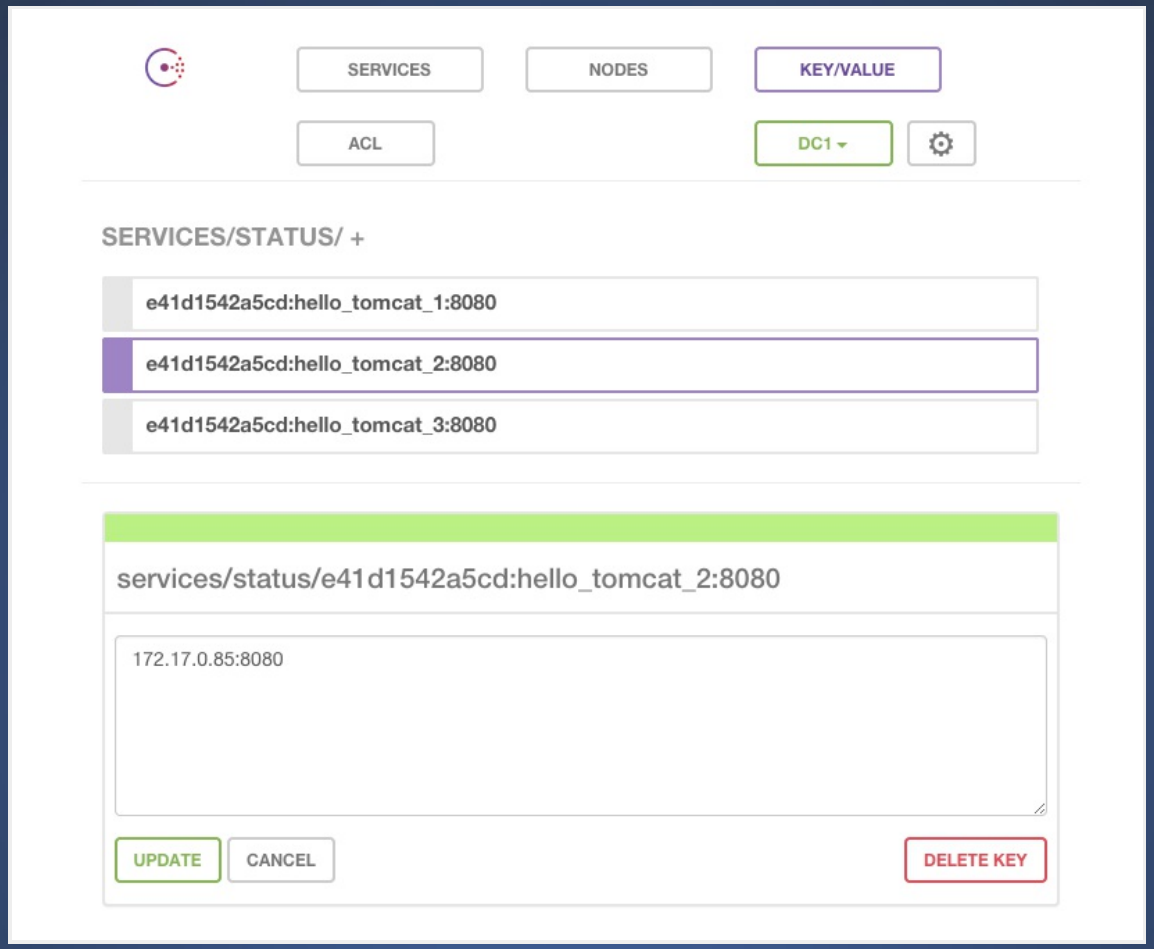
master.yml

```
consul:
  build: consul
  ports:
    - "8400:8400"
    - "8500:8500"
    - "8600:53/udp"
  command: -bootstrap

registrator:
  build: registrator
  links:
    - consul:consul
  volumes:
    - /var/run/docker.sock:/tmp/docker.sock
  command: -internal consul://consul:8500/services

lb:
  build: nginx-lb
  ports:
    - "80:80"
  links:
    - consul:consul
```

CONSUL UI


SERVICES
NODES
KEY/VALUE
ACL
DC1
SERVICES/STATUS/ +
e41d1542a5cd:hello_tomcat_1:8080
e41d1542a5cd:hello_tomcat_2:8080
e41d1542a5cd:hello_tomcat_3:8080
services/status/e41d1542a5cd:hello_tomcat_2:8080
172.17.0.85:8080
UPDATE
CANCEL
DELETE KEY

CONSUL-TEMPLATE

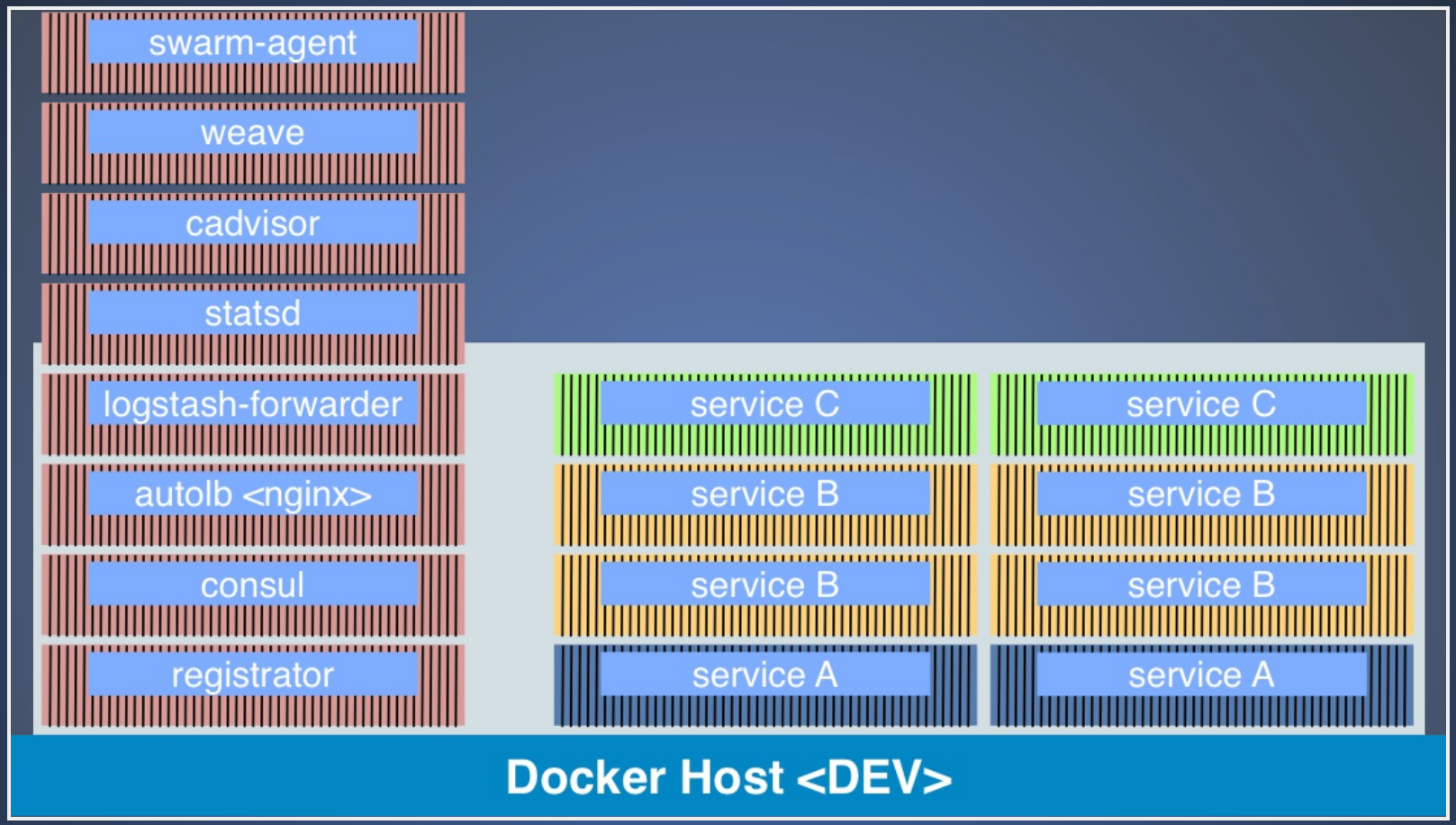
- Use extended GO templates
- Restart service after template changed
- Preferred solutions if you use consul

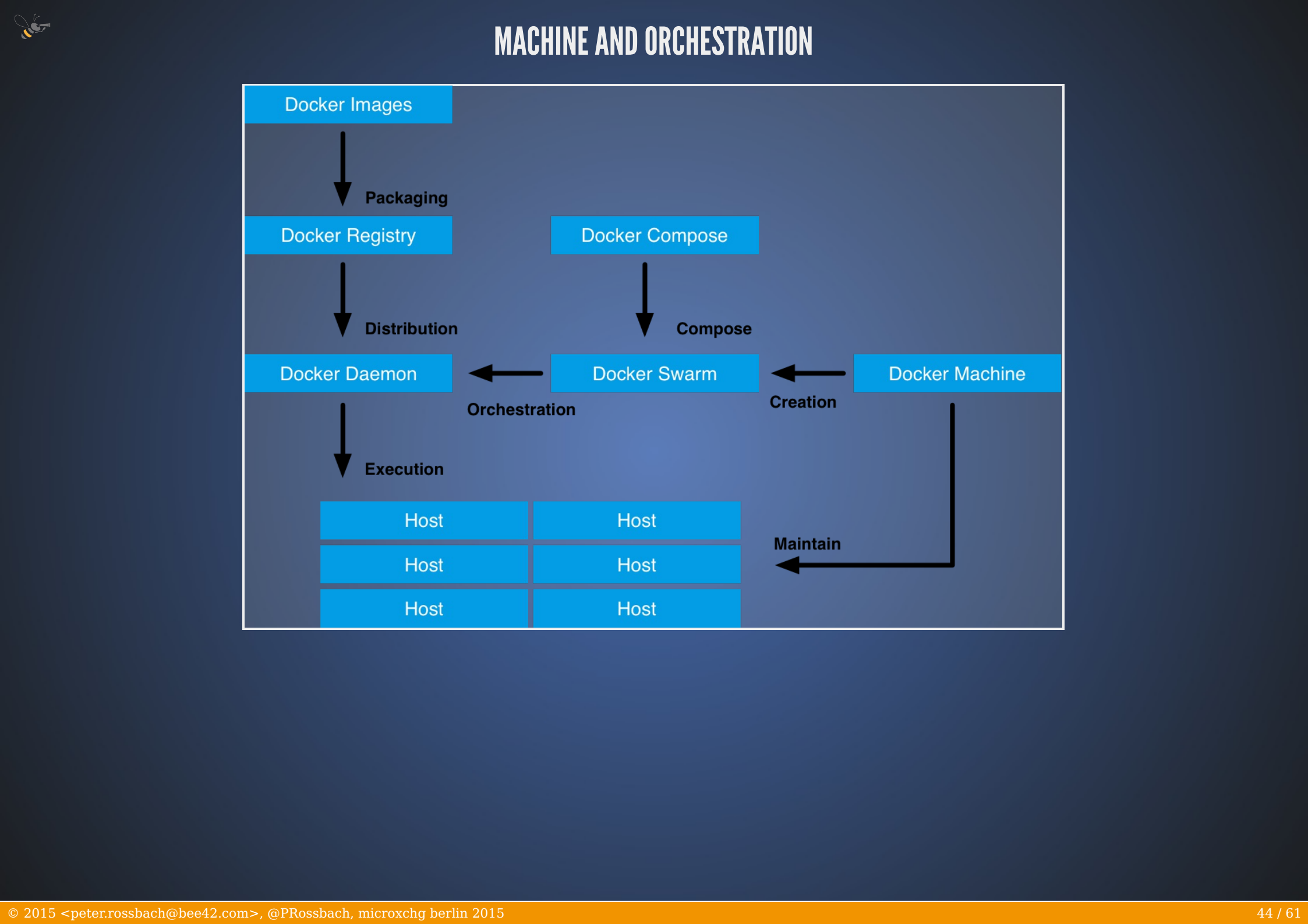
- github consul-template
- Other option: github confd

SUMMARY

- After two days hard work => Success...
- I like to add more infrastructure parts...
- Add speed to your development with simpler tomcat/app setup
- Don't accept huge amount of framework libs...
- Think about infrastructure first...

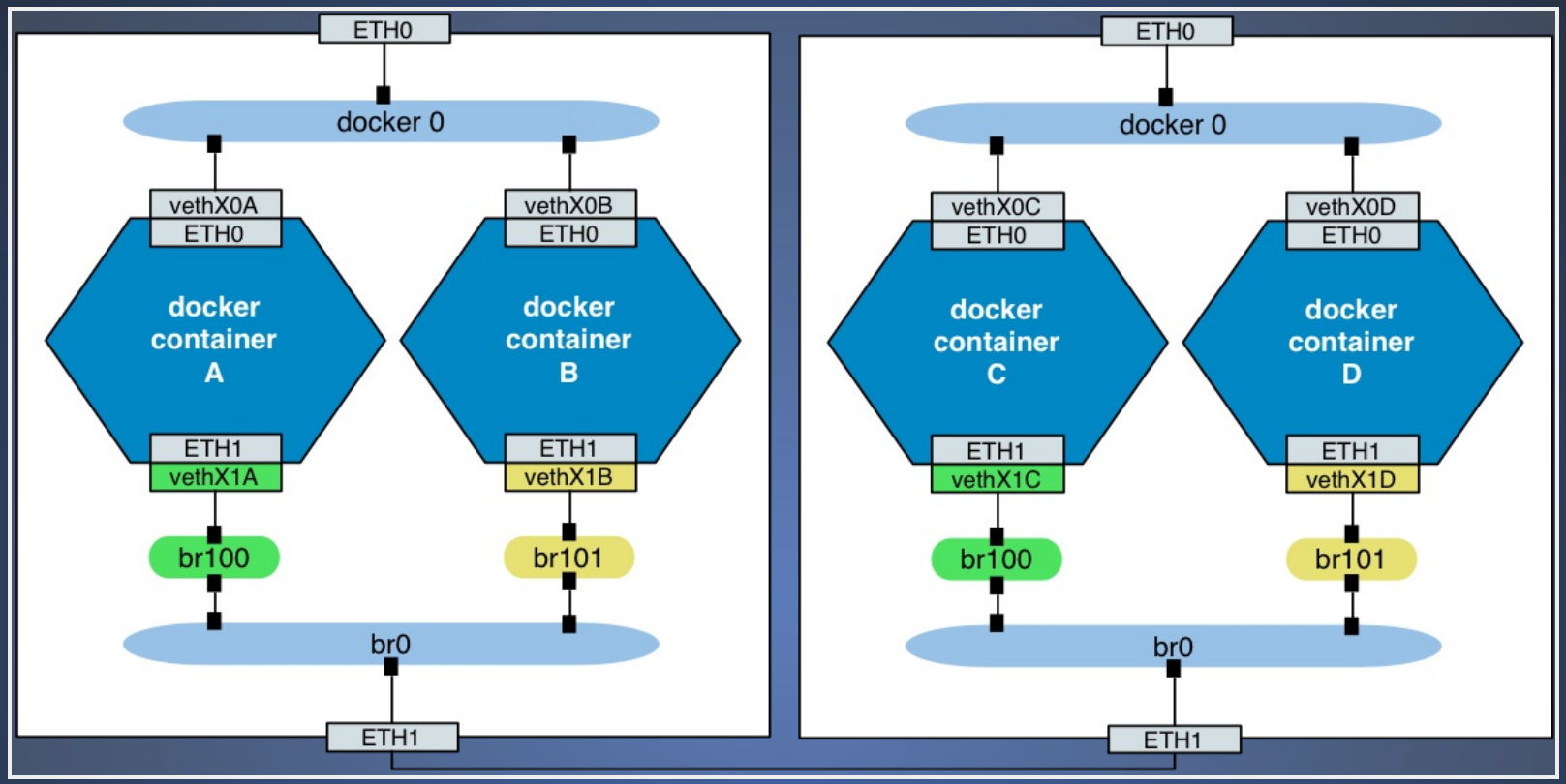
HOW I HANDLE THE GROWING PARTS?





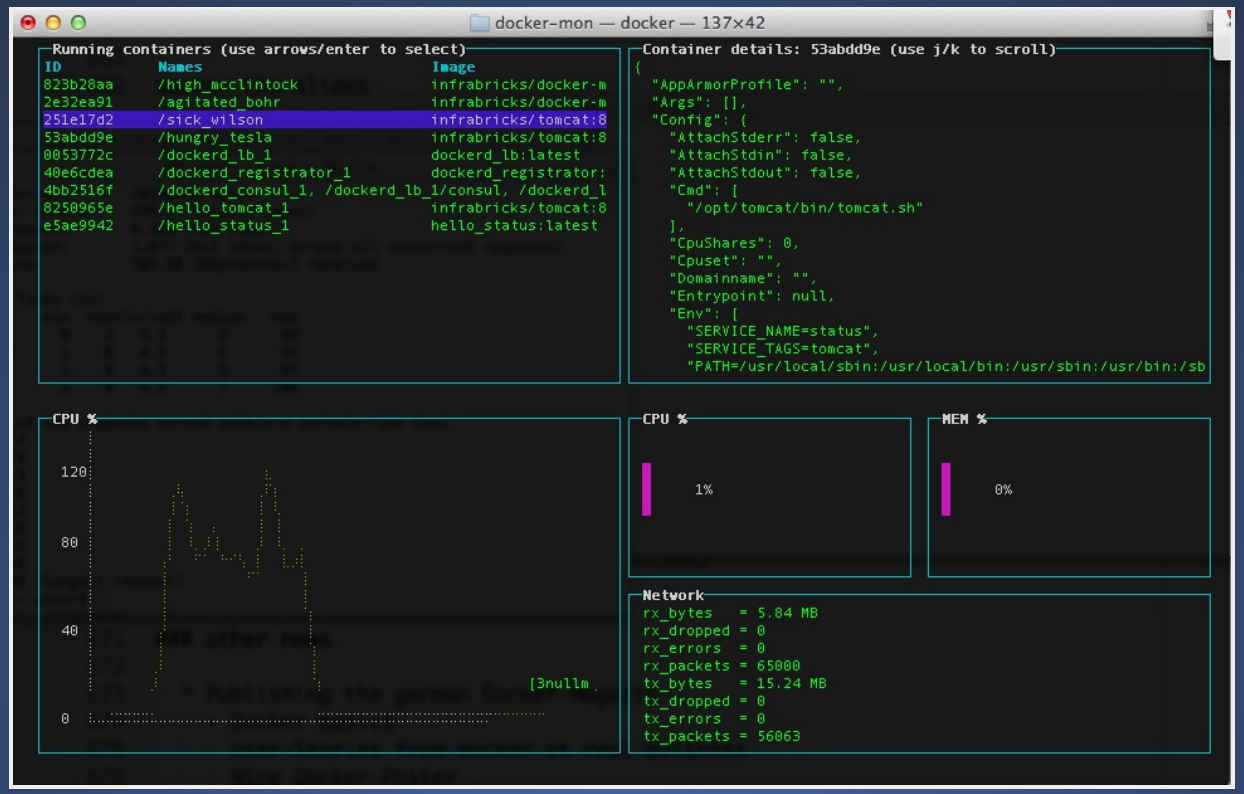
MACHINE AND ORCHESTRATION

NETWORK



More pieces are available :(

DOCKER-MON



- https://github.com/icecrime/docker-mon
- `docker stats <CID>`

CONCLUSION

- Setup the build and prod env is also with Docker tricky
  - A lot of detail work todo.
- Autoscaling env is complex and production needs a more robust solution
- Currently my setup is **WORK in progress**

- Need more experiments!
- Describe the results in my infrabricks docker microservice book...

MY GOLDEN CONTAINER RULES

rule	description
ONE PURPOSE	The container has only one purpose
WORKING	The container is up and running
FIX DEPENDENCIES	The container has controlled dependencies
MINIMAL	The container has only minimal content, nothing more
WHITEBOX	The container source must be available
SECURE	The container must be periodically updated
LIMITED	The container can only consume limited resources
TRUST	The creator of the containers is trustworthy and the content can be validated

JAVA MICROSERVICE SOLUTIONS

- Spring Boot
- DropWizard
- vert.x
- Scala Play, Scala.io and AKKA
- Undertow
- Netty
- Tomcat Embed
- Jetty Embed

SEARCH FOR SIMPLER SOLUTIONS

- node.js
- go libchan

MY PRINCIPLES

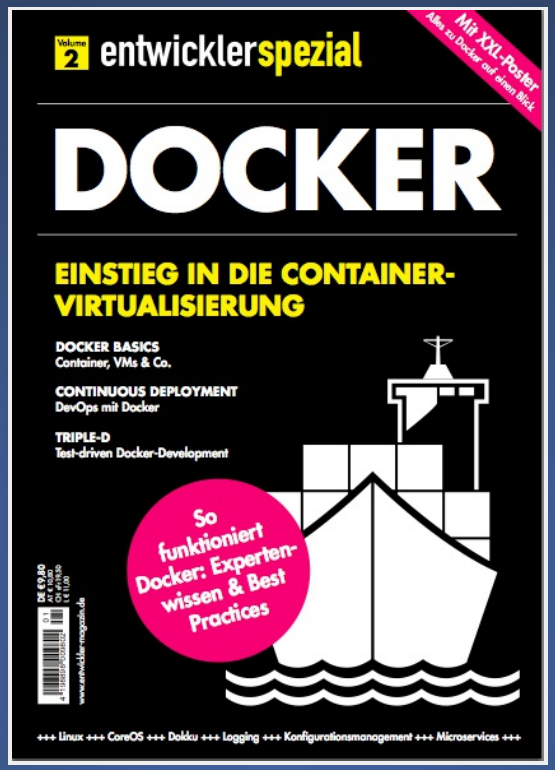
- Less is more
- Simple is better
- Quality costs your money
- Success costs a lot of your time

Perfect is not easy going!

BUILD, SHIP, RUN
ANY APP, ANYWHERE...

TWO MORE THINGS...

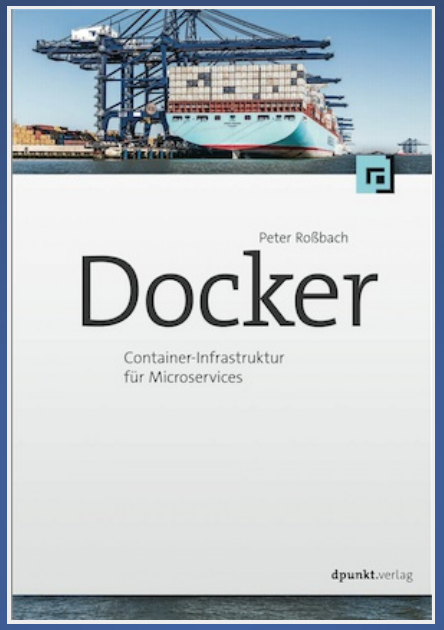
GERMAN: DOCKER ENTWICKLER SPEZIAL MAGAZIN



- 17.11.2014
- Docker Sonderheft

DOCKER

CONTAINER INFRASTRUKTUR FÜR MICROSERVICES



- July 2015
- www.dpunkt.de/docker

BEE42 BOARDING TIME

Work with the beekeeper!

MY OFFER TO YOU!

Service	benefit
Lab:	use my IT-brick knowledge to be faster informed
Consulting:	use my expertise to start smarter
Training:	let me help you to get a docker-brick expert
Products:	use my knowledge of the docker ecosystem

---

peter.rossbach@bee42.com

THE DEVOPS-COMMUNITY NEEDS YOU !

bee42 solutions gmbh starts the implementation of an **Infrabricks line**!

- Peter Rossbach
- Systemarchitect
- Java Developer
- DevOps, Docker-enthusiast and infra-coder with passion
- Apache Tomcat Committer
- Member of the Apache Software Foundation
- Founder of the bee42 solutions gmbh
- Author
  - starts writing the german docker handbook!

We're hiring!

OPEN THE DOORS FOR Q&A...

CONTACT

docker the slides

- start with `docker run -d -p 8000:80 rossbachp/docker-tomcat-micro:microxchg2015`
- open your browser with `http <dockerhost>:8000/docker-tomcat-micro`
- Slides at speakerdeck https://speakerdeck.com/rossbachp

slidefire available at 16.02.2015

- peter.rossbach@bee42.com
- @PRossbach
- Infrabricks Blog
- bee42

GIVE THE NAVIGATOR THE RUDDER!

OTHER NEWS

- Publishing the german Docker-Magazin
  - Docker basics
  - User-Stories from docker at real buisness
  - Nice Docker-Poster
  - Docker Sonderheft
- JAX 2015 20.-24.4.2015
  - DevOps-Days
  - Docker-Workshop
- New DevOpsConf => Berlin 1-3.6.2015 - http://devopsconference.de/2015/
- Docker Conf US 20.-24.6.2015
- Continuous Lifecycle 2015
- Q3 2015: My docker handbook!