QAWARE
SOFTWARE ENGINEERING

# Vom Monolithen in die Cloud.

**Mario-Leander Reimer, QAware GmbH**
mario-leander.reimer@qaware.de

Linux Stammtisch
München, 30. Januar 2018

# Mario-Leander Reimer

Cheftechnologe, QAware GmbH

- Entwickler && Architekt
- 20+ Jahre Erfahrung
- #CloudNativeNerd
- Open Source Enthusiast


**Kontakt:**

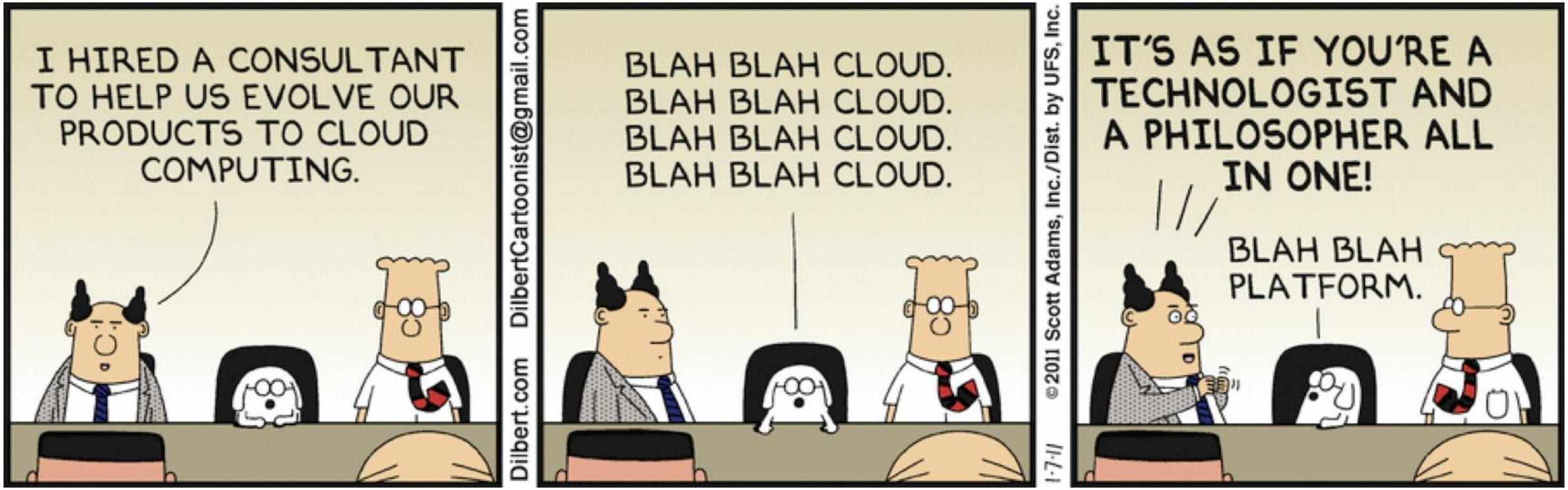
Phone: +49 89 23 23 15 121

Mail: mario-leander.reimer@qaware.de

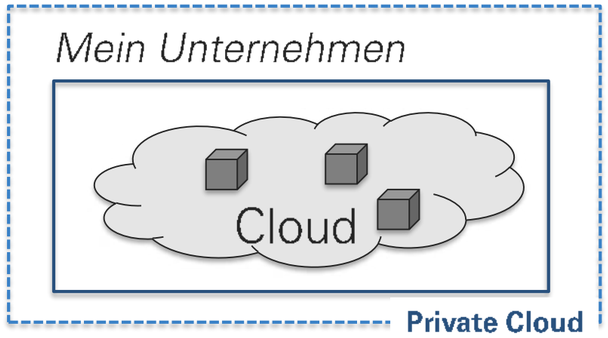
Twitter: @LeanderReimer

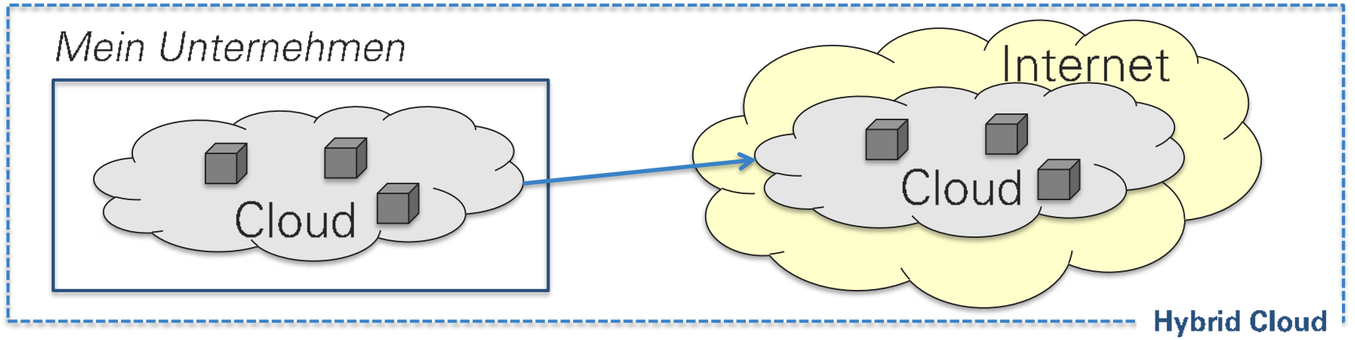
Github: https://github.com/lreimer

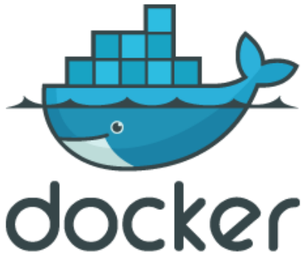
# Wir gehen in die Cloud!



# Gesagt, getan!



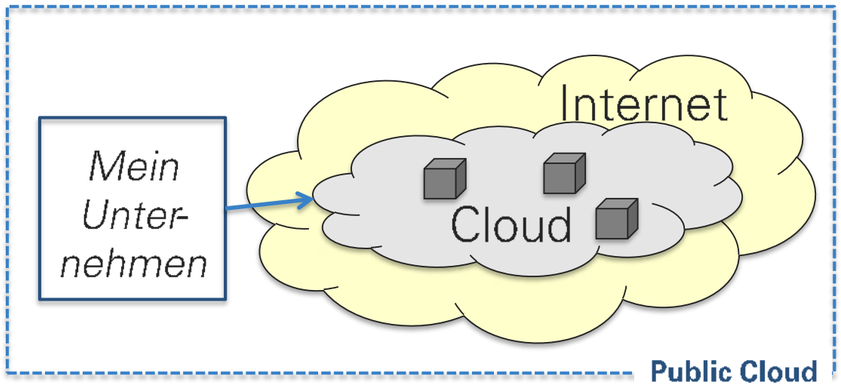



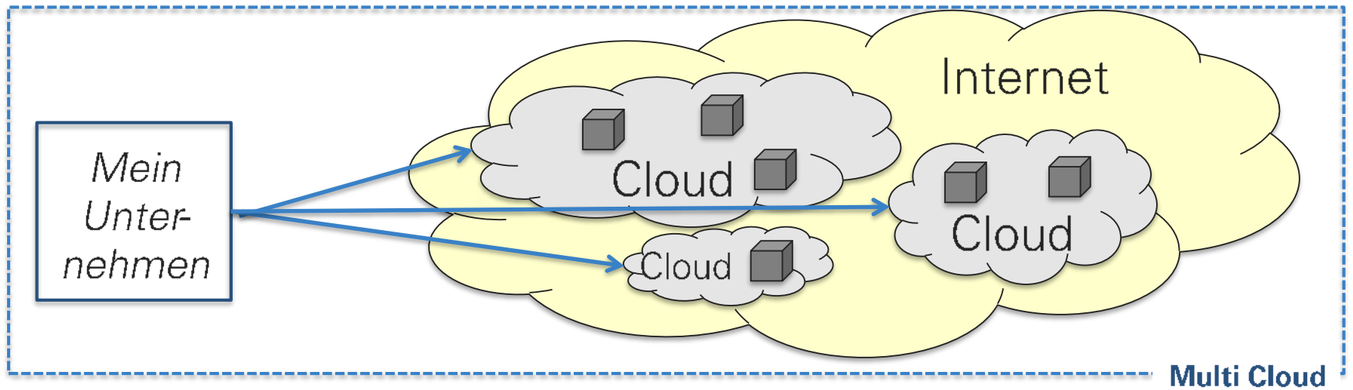









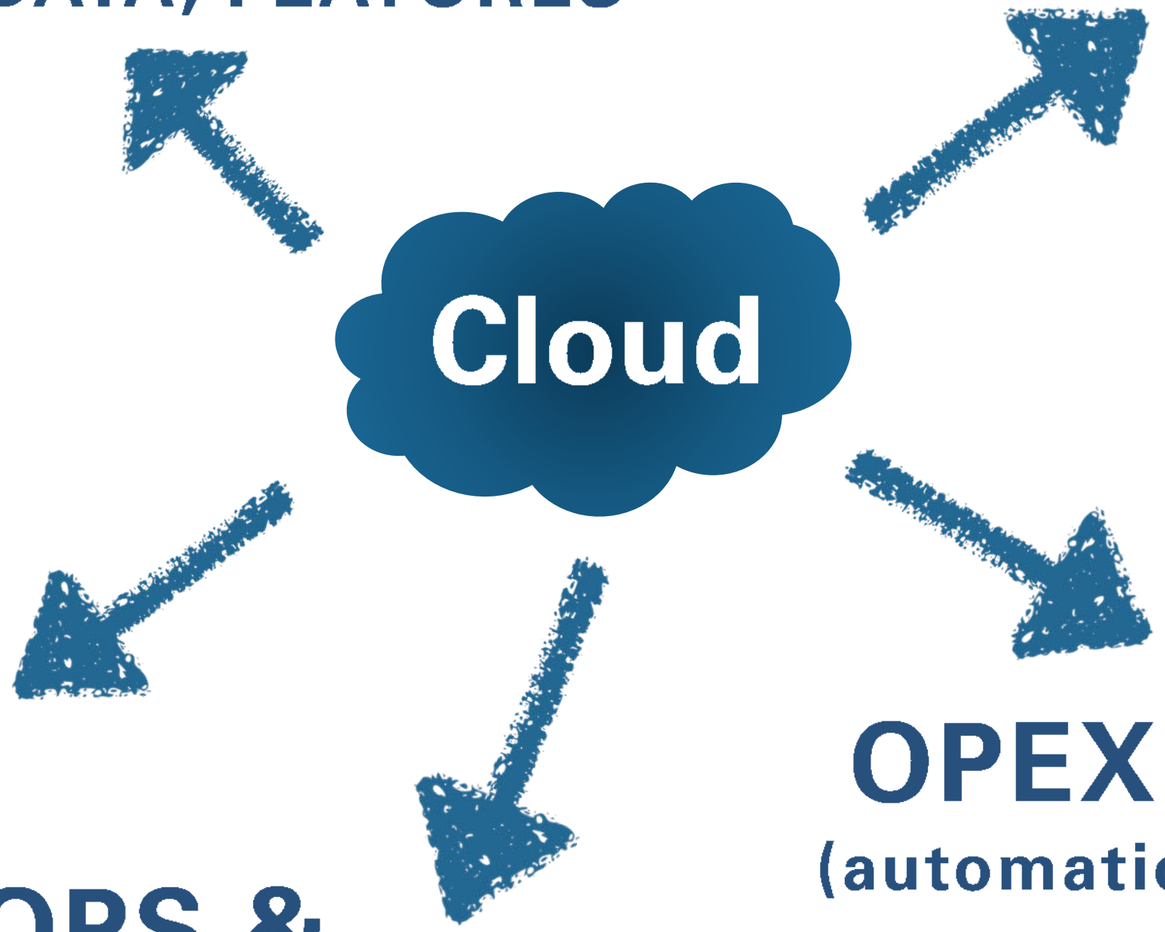



DISRUPT

INDUSTRIALIZE

# HYPERSCALE
TRAFFIC, DATA, FEATURES

# ANTIFRAGILITY

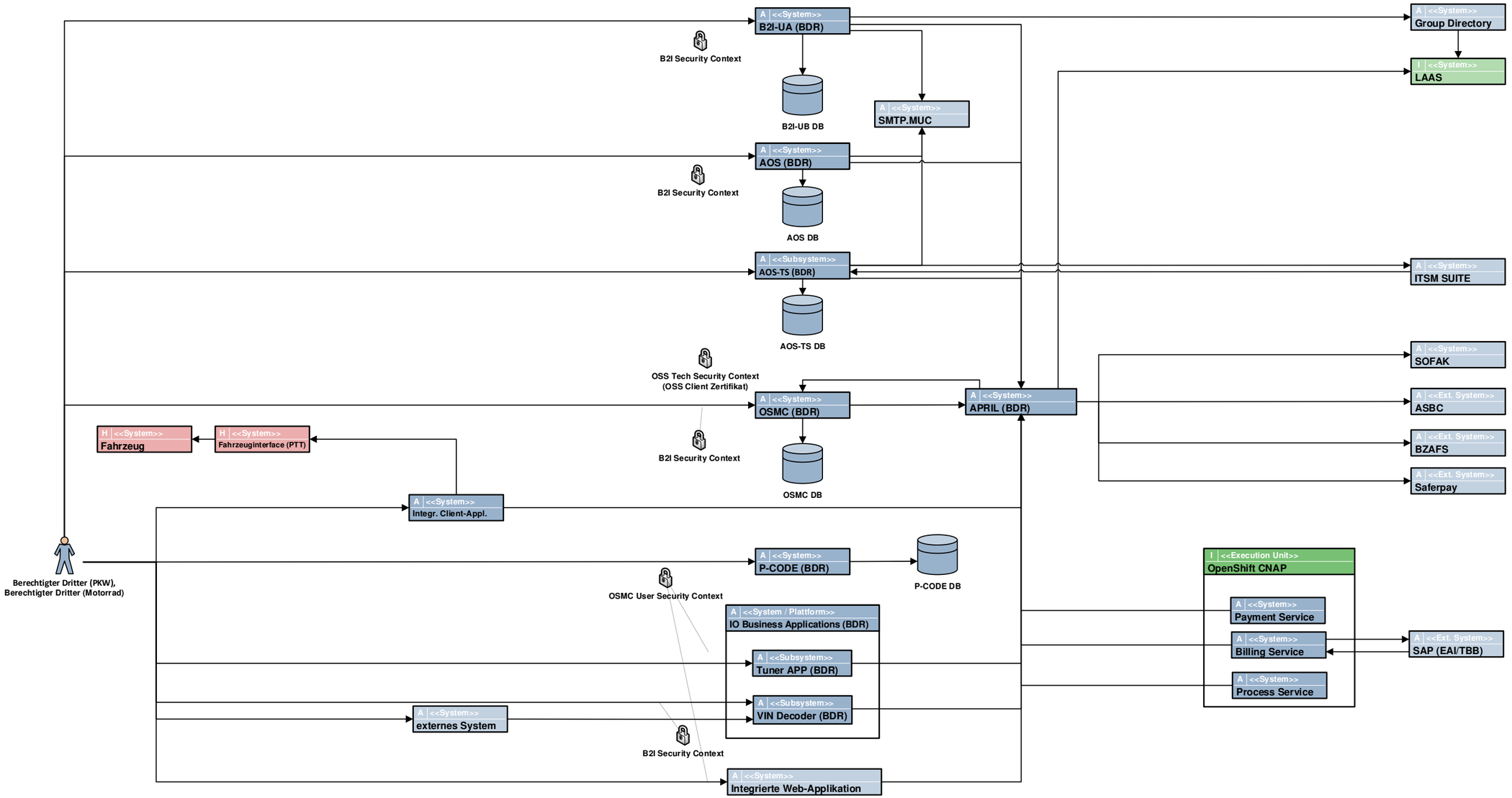



# SPEED

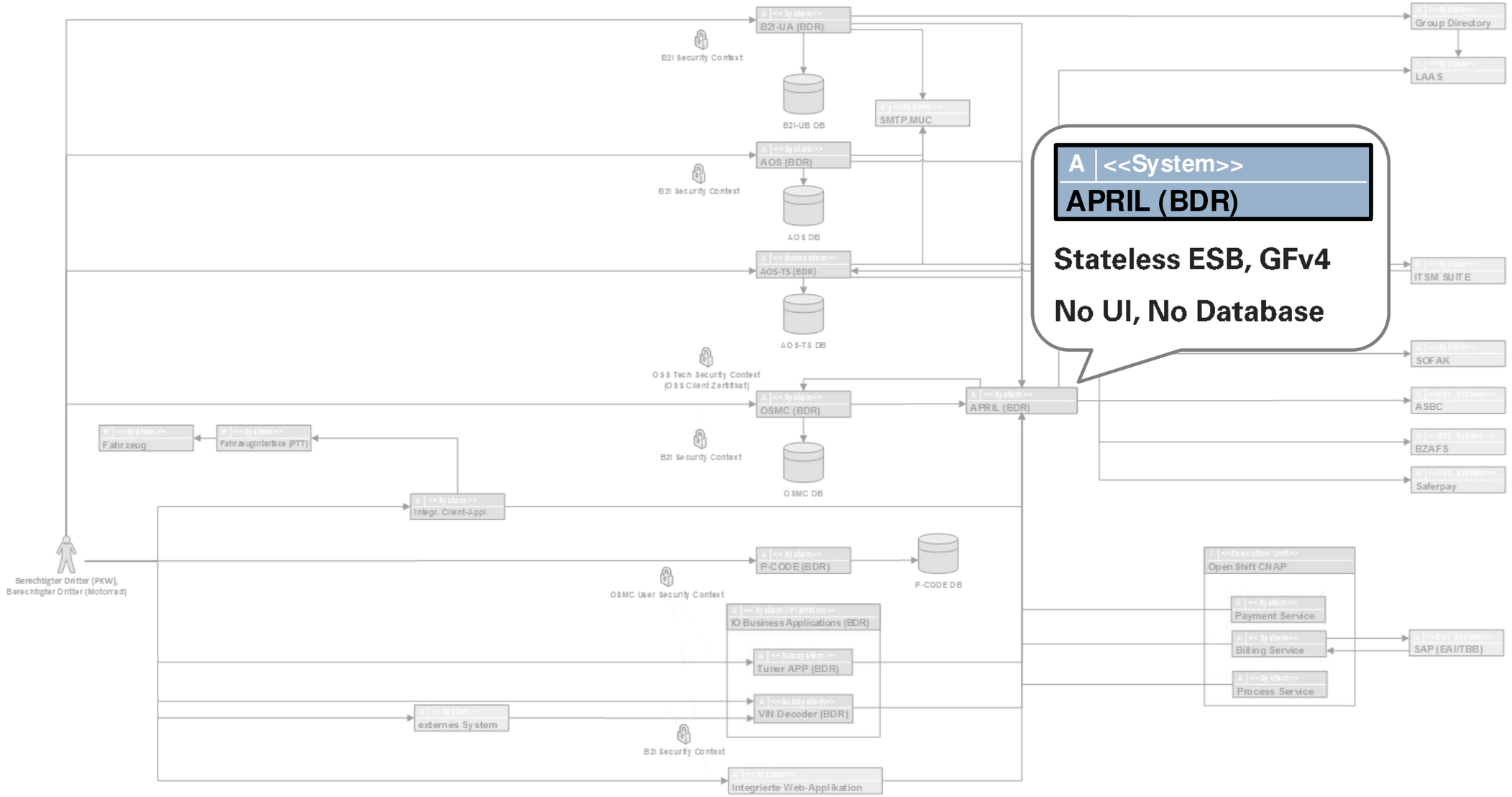
# OPEX SAVINGS
(automation & utilization)

# DEVOPS &
CONTINUOUS DELIVERY

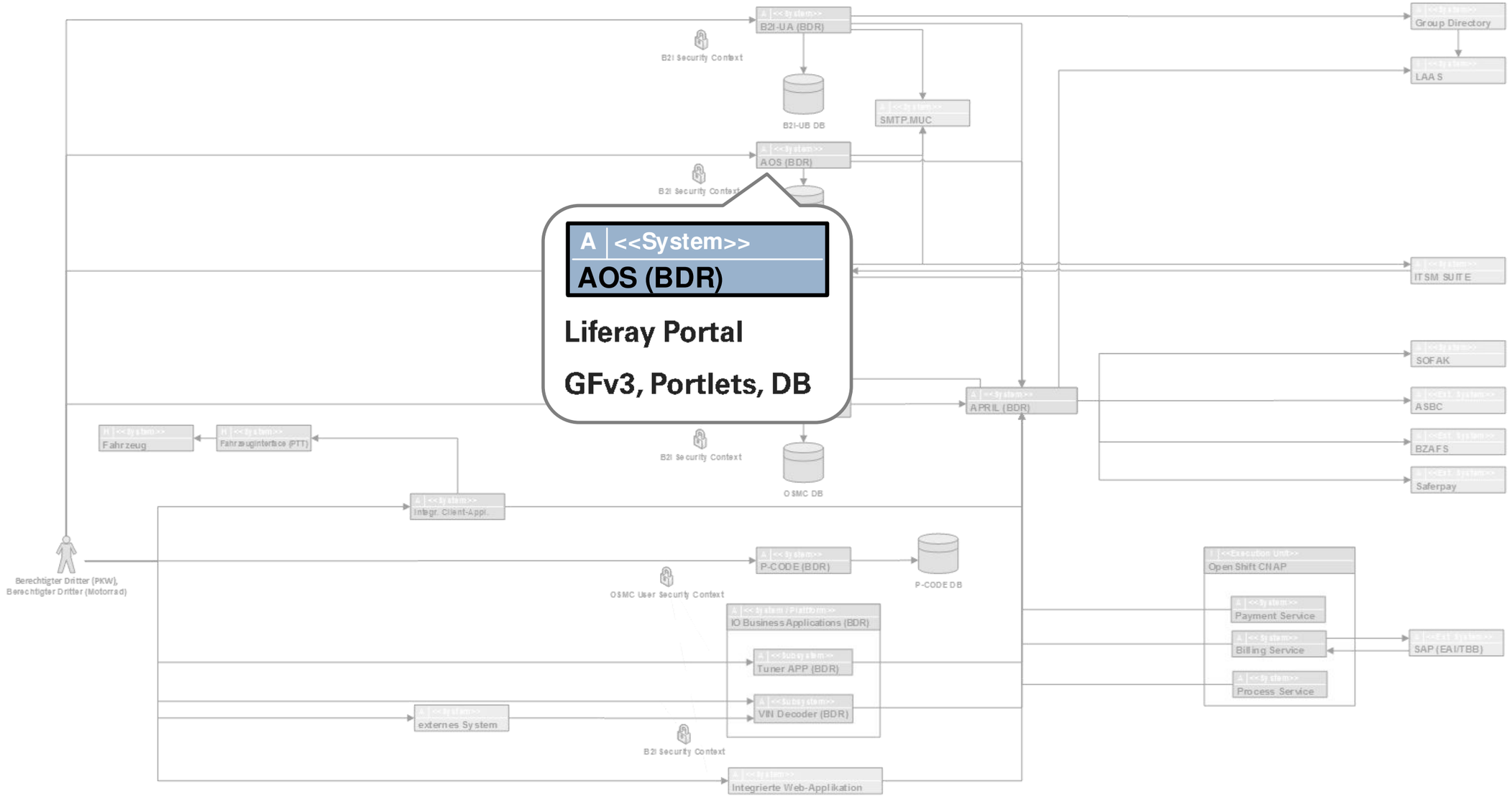
# Hyperscale, Antifragilität und Continuous Delivery als wesentliche Treiber für die Evolution unserer Systeme.



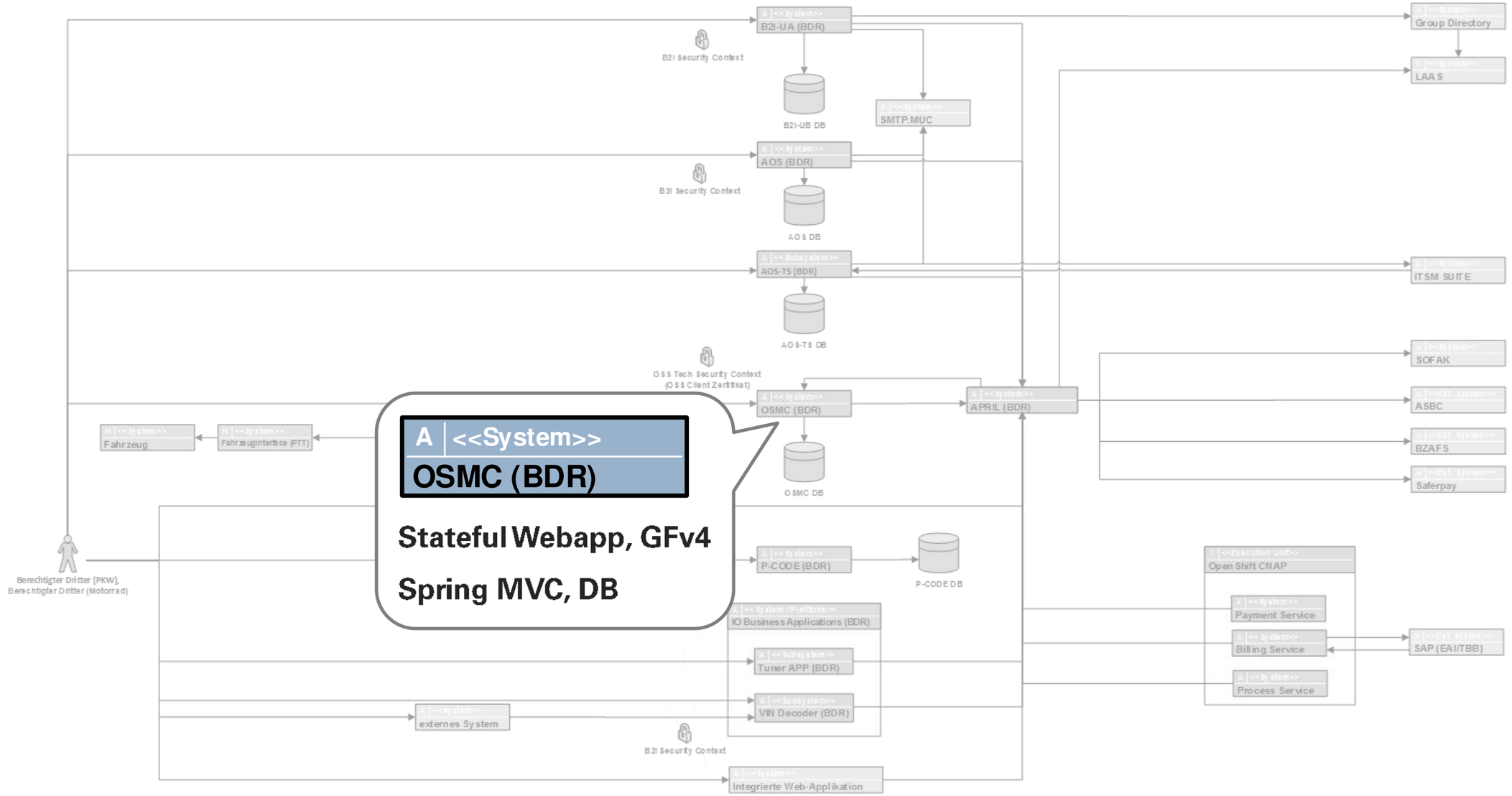
# Hyperscale, Antifragilität und Continuous Delivery sind wesentliche Treiber für die Evolution unserer Systeme.



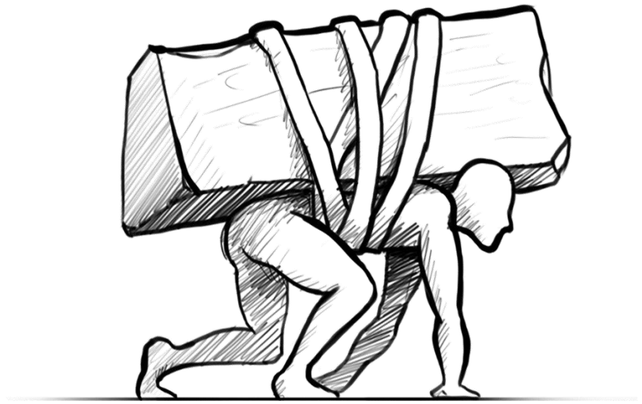
# Hyperscale, Antifragilität und Continuous Delivery sind wesentliche Treiber für die Evolution unserer Systeme.



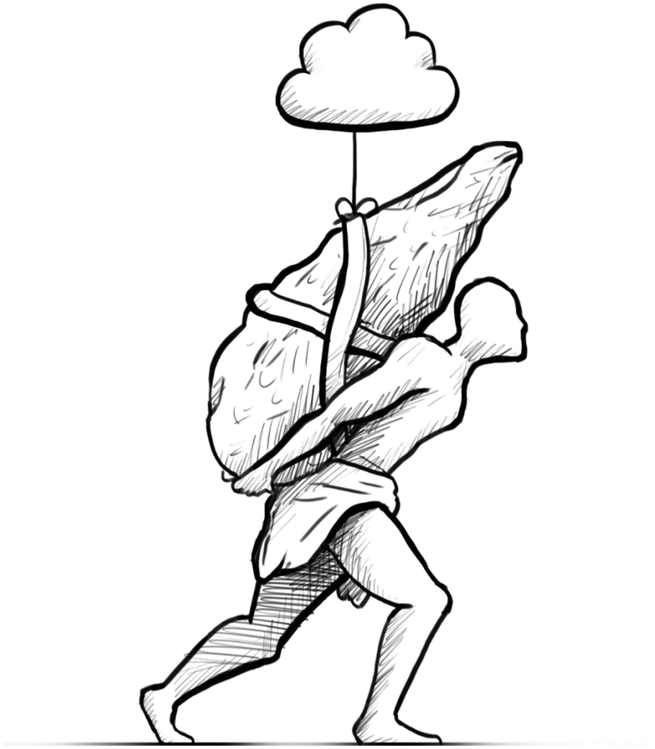
# Hyperscale, Antifragilität und Continuous Delivery sind wesentliche Treiber für die Evolution unserer Systeme.



# Lassen sich unsere Bestandssysteme mit vertretbarem Aufwand in Richtung Cloud entwickeln?



- Monolithic Deployment
- Traditional Infrastructure



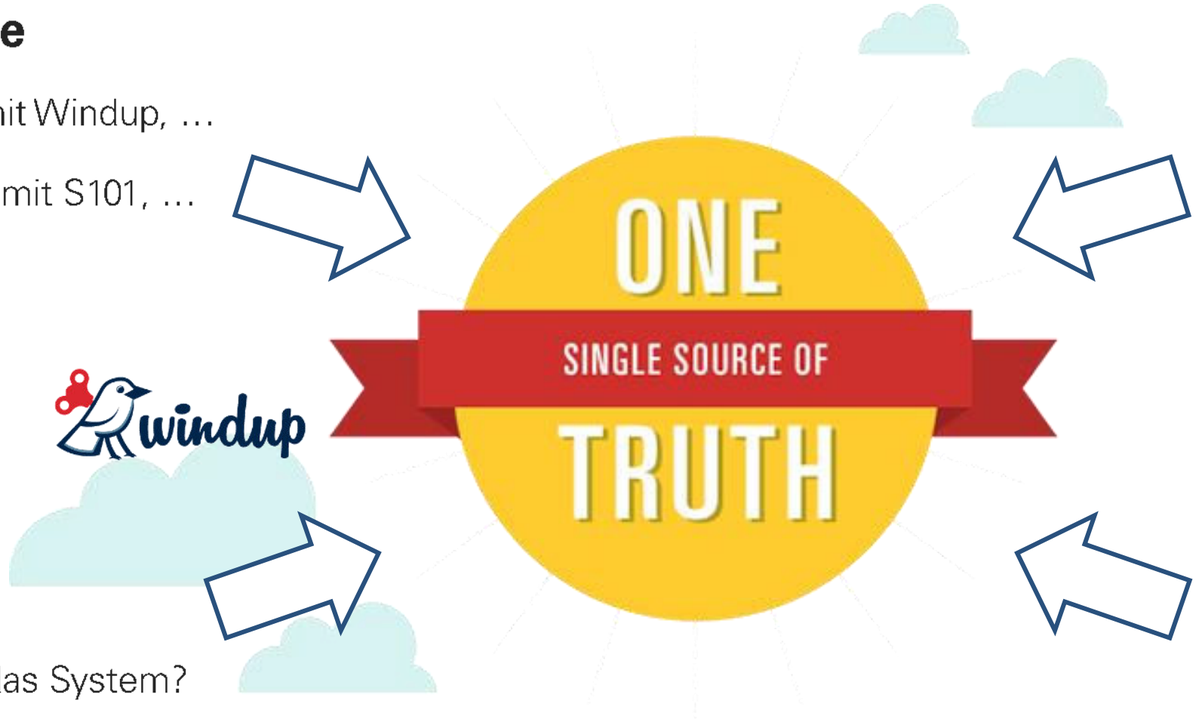
- Containerization
- 12-Factor App Principles

- Microservices
- Cloud-native Apps

# Fakten schaffen. Es braucht eine belastbar Aufwands- und Ressourcenprognose.

## Software-Analyse

- Statische Analyse mit Windup, …
- Architektur Analyse mit S101, …




## Proof of Concepts

- Migration von APRIL
- Migration der B2I-UA

## Fragenkatalog

- Welche Komplexität hat das System?
- Welche Musterlösung verwendet das System?
- Welche Technologien werden verwendet?

## QAware Know How

- Erfahrung aus anderen Migrationen
- Kontenrahmen
- Eingespieltes Team

# Fragebogen: Typische Fragen und ihre Motivation.

1. **Technology Stack** (e.g. OS, appserver, jvm)
2. **Benötigte Resourcen** (memory, CPU cores)
3. **Writes to storage** (local/remote storage, write mode, data volume)
4. **Spezielle Anforderungen** (native libs, special hardware)
5. **Inbound und Outbound Protocols** (protocol stack, TLS, multicast, dynamic ports)
6. **Ability to Execute** (regression/load tests, business owner, dev knowhow, release cycle, end of life)
7. **Client Authentifizierung** (e.g. SSO, login, certificates)

- What images to provide?
- How many applications will be hard or inefficient to schedule (> 3 GB RAM, > 2 cores)?
- What storage solutions to provide?
- What applications will be hard or impossible to be containerized?
- Are there any non cloud-friendly protocols?
- How risky is the migration? Is the migration maybe not required?
- What IAM and security mechanisms haveto be ported to the cloud?

# All Migration Issues Report

The Migration Issues report provides a concise summary of all issues that require attention.

## Analysis Detail

| Issue by Category | Incidents Found | Story Points per Incident | Level of Effort | Total Story Points |
|---|---|---|---|---|
| **Optional** | 295 | | | 54 |
| Maven POM | 205 | 0 | Info | 0 |
| Java APIs for local file system | 36 | 1 | Trivial change or 1-1 library swap | 36 |
| Hard-coded IP Address Detected | 28 | 0 | Info | 0 |
| Used class java.net.URL/URI with local path | 18 | 1 | Trivial change or 1-1 library swap | 18 |
| Unparsable XML File | 5 | 0 | Info | 0 |
| Web XML | 2 | 0 | Info | 0 |
| Dynamic class instantiation | 1 | 0 | Info | 0 |

| Issue by Category | Incidents Found | Story Points per Incident | Level of Effort | Total Story Points |
|---|---|---|---|---|
| **Potential Issues** | 494 | | | 494 |
| Detected local file system paths | 482 | 1 | Trivial change or 1-1 library swap | 482 |
| Detected URL with local file system path | 12 | 1 | Trivial change or 1-1 library swap | 12 |

# Anwendungen können über fünf verschiedene Wege migriert werden.

1. **Rehost:** Die Anwendung as-is auf neue Hosts umziehen

2. **Refactor**: Die Anwendung möglichst ohne funktionale Änderung so anpassen, dass sie auf einer Cloud-Plattform (Kubernetes, OpenShift, DC/OS) und der darauf verfügbaren Ausführungsplattform (Container, OS, JDK, AppServer) läuft.

3. **Revise**: Die Anwendung so anpassen, dass sie die Möglichkeiten und Vorteile der Cloud-Plattform möglichst gut ausnutzt. Die wesentlichen Design Prinzipien Cloud-nativer Apps werden dabei soweit vertretbar berücksichtigt.

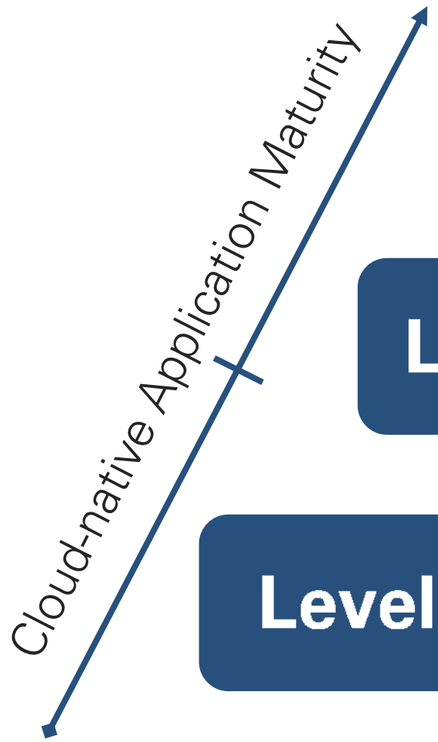
4. **Rebuild**: Anwendung neu bauen oder im großen Stil umbauen als Cloud Native Anwendung, geschnitten in Microservices und API getrieben

5. **Replace**: Anwendungen abmanagen oder durch neue Anwendungen ersetzen

# Wichtige Design Prinzipien Cloud-nativer Applikationen dienen als Leitplanken für benötigte Umbauten.

- **Design for Distribution**: Containers; microservices; API getriebene Entwicklung.
- **Design for Automation**: Automatisierung von Dev & Ops Tasks.
- **Design for Resiliency**: Fehlertolerant und selbstheilend.
- **Design for Elasticity**: Skaliert dynamisch und reagiert auf Stimuli.
- **Design for Performance**: Responsive; Concurrent; Ressourcen effizient.
- **Design for Delivery**: Kurze Roundtrips und automatisierte Provisionierung.
- **Design for Diagnosability**: Cluster-weite Logs, Metriken und Traces.
- **Design for Security:** Abgesicherte Endpunkte, API-Gateways, E2E-Encryption.

# Ein gestuftes Vorgehen macht die Migration planbar, die technologischen Risiken werden leichter beherrschbar.




**Level 3: Cloud Native**

- Loosely coupled microservices.
- Scales dynamically based on load.

**Level 2: Cloud Resilient**

- Fault tolerant and resilient design.
- Metrics and monitoring built-in.

**Level 1: Cloud Friendly**

- Adheres to the 12-factor app principles.
- Cloud friendly app server runtime.

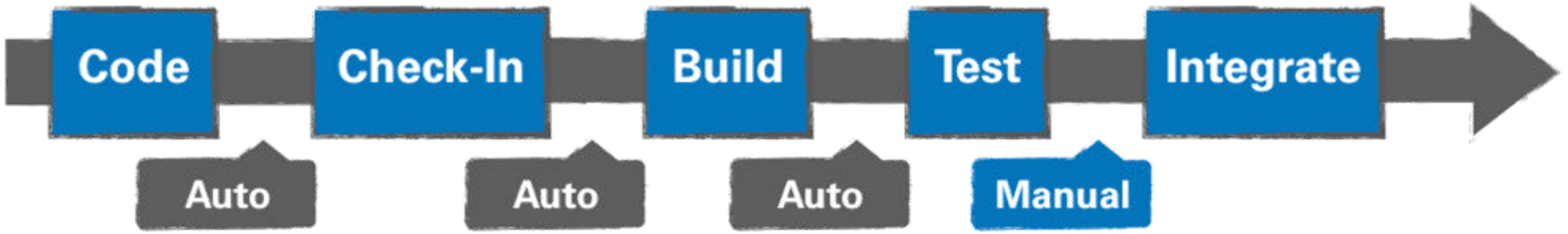
**Level 0: Cloud Ready**

- Executed as self-contained image.
- Runs on virtualized HW and file system.

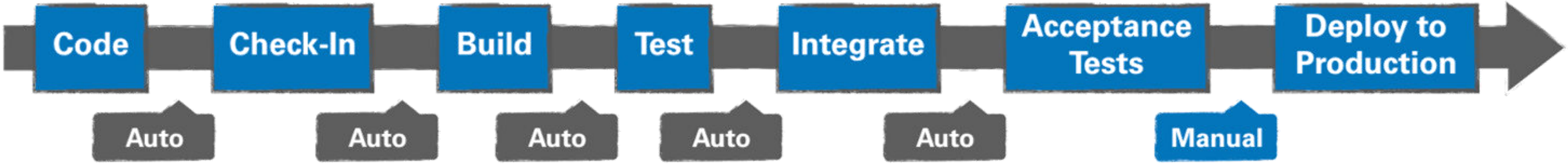
https://www.opendatacenteralliance.org/docs/architecting_cloud_aware_applications.pdf

# Delivery

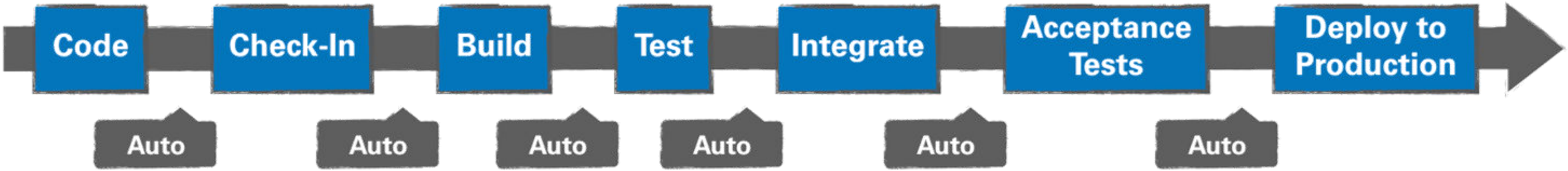
## Continuous Integration

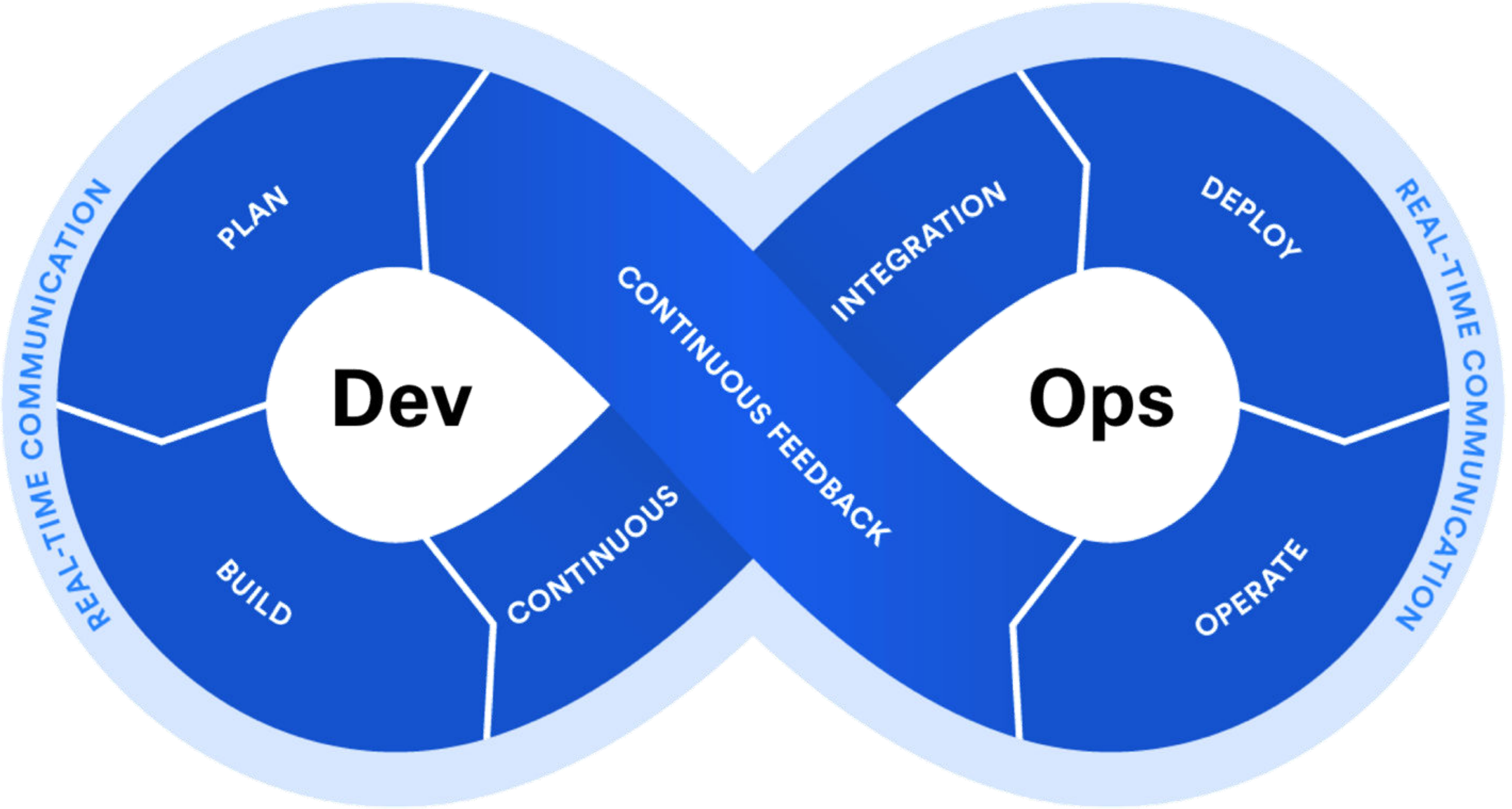



## Continuous Delivery



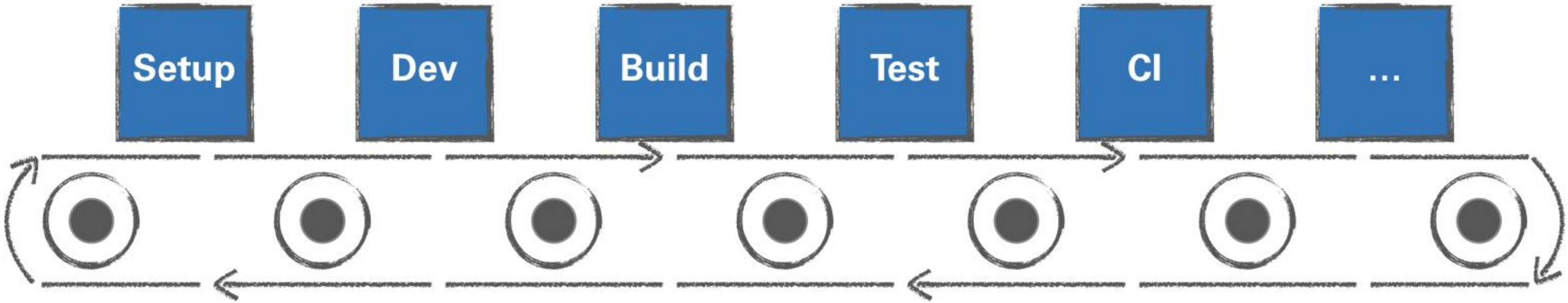

## Continuous Deployment




REAL-TIME COMMUNICATION
PLAN
CODE
BUILD
CONTINUOUS
Dev
CONTINUOUS FEEDBACK
INTEGRATION
DEPLOY
OPERATE
Ops
REAL-TIME COMMUNICATION

# Software Industrialisierung ist eine Schlüsselanforderung für erfolgreiches DevOps und Continuous Delivery.

- Hoher Automatisierungsgrad von arbeits-intensiven und wiederkehrenden Tasks
- Bessere Software-Qualität durch eine abgestimmte Tool-Chain
- Mehr Produktivität und Zufriedenheit der Entwickler-Teams
- Bessere Kosten-Effizienz und Wettbewerbsfähigkeit



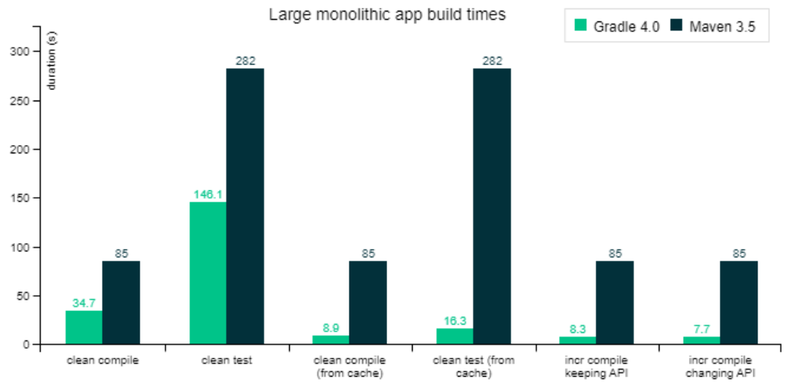
# Durch die Evolution der Build Toolchain werden schnelle Roundtrips und effiziente Feature Entwicklung möglich.

SUBVERSION → Bitbucket

- Nutzung einer Agile Delivery Tool Chain
- Migration aller Repositories von SVN nach Git mit voller Historie innerhalb 1 Woche
- Anpassung und Migration aller Jenkins Build-Jobs auf neue Build Infrastruktur
- *More improvements to come …*

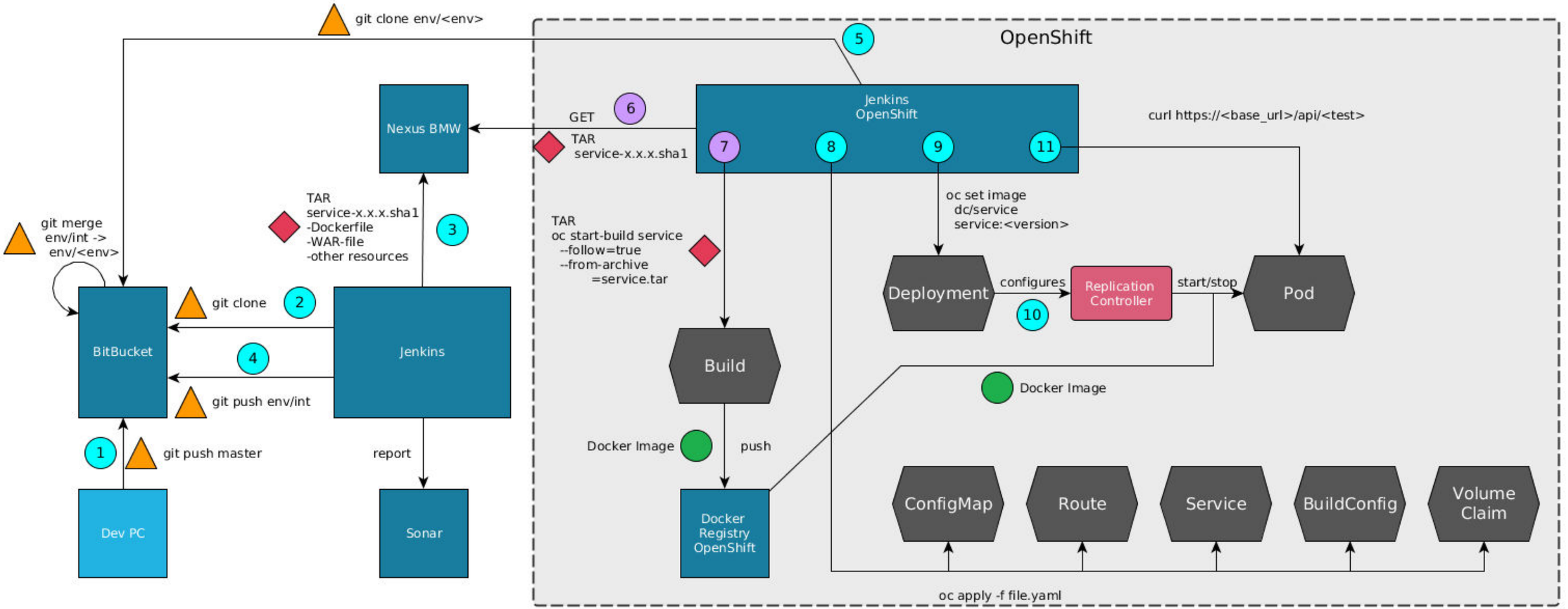
Maven™ → Gradle

- Drastisch reduzierte Build-Zeiten für mehr Produktivität und Qualität
- Migration von Bestandsprojekten mit Augenmaß




https://gradle.org/gradle-vs-maven-performance/

# Unsere Continuous Integration & Deployment Pipeline.



# Distribution

# Simple Dockerfile for Payara Micro.

```
FROM payara/micro:174

# copy the WAR file into deployments directory
COPY target/april-bdr-runtime-1.5.0-SNAPSHOT.war /opt/payara/deployments/

USER root
RUN mkdir -p /april/logs && chown -R payara:payara /april
USER payara

ENTRYPOINT ["java", "-server", "-Dcom.bmw.mastersolutions.gf.domain.dir=/april",
                               "-Dcom.bmw.iap.april.gf.project.data.shared=/april/data",
                               "-Dcom.bmw.mastersolutions.gf.project.logs=/april/logs",
                               "-jar", "/opt/payara/payara-micro.jar"]
CMD ["--deploymentDir", "/opt/payara/deployments", "--noCluster"]
```

# A docker-compose.yml for building and running locally.

```
version: ‘2’

services:
  april-bdr-runtime:
    build: .
    image: "april-bdr-runtime:1.5.0"
    volumes:
    - ./src/test/glassfish/data:/april/data
    - ./target/glassfish/logs:/april/logs
    ports:
    - "8080:8080"
```

**Use volumes to mount local host directories into the container**

# More sophisticated Dockerfile for Payara Server.

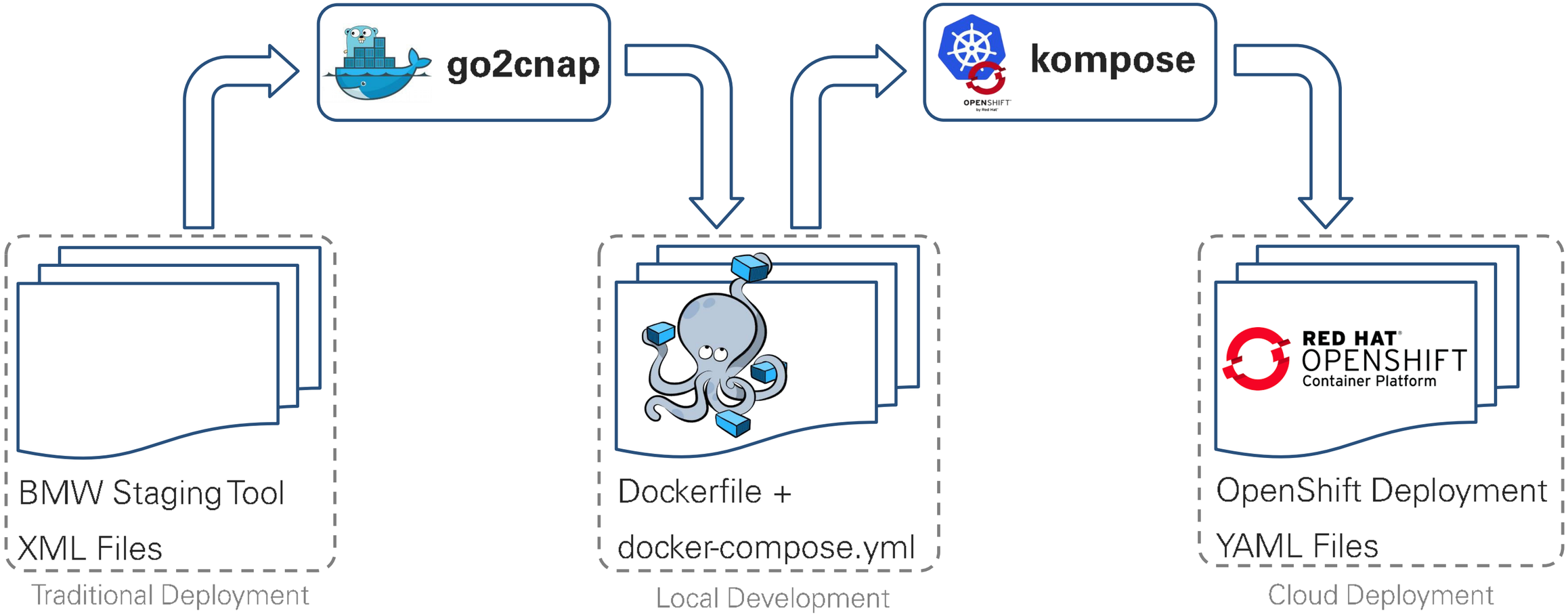
```
FROM payara/server-full:173

COPY *.asadmin /tmp/
RUN $AS_ADMIN start-domain $PAYARA_DOMAIN && \
    $AS_ADMIN $AS_ADMIN_LOGIN multimode --file /tmp/jvm_options.asadmin && \
    $AS_ADMIN $AS_ADMIN_LOGIN multimode --file /tmp/payara_optimization.asadmin && \
    $AS_ADMIN stop-domain $PAYARA_DOMAIN

COPY target/april-bdr-runtime-1.5.0-SNAPSHOT.war $DEPLOY_DIR
RUN ${PAYARA_PATH}/generate_deploy_commands.sh

# RUN $AS_ADMIN start-domain --dry-run --postbootcommandfile $DEPLOY_COMMANDS $PAYARA_DOMAIN
COPY start-domain.sh $PAYARA_PATH/start-domain.sh
ENTRYPOINT $PAYARA_PATH/start-domain.sh
```

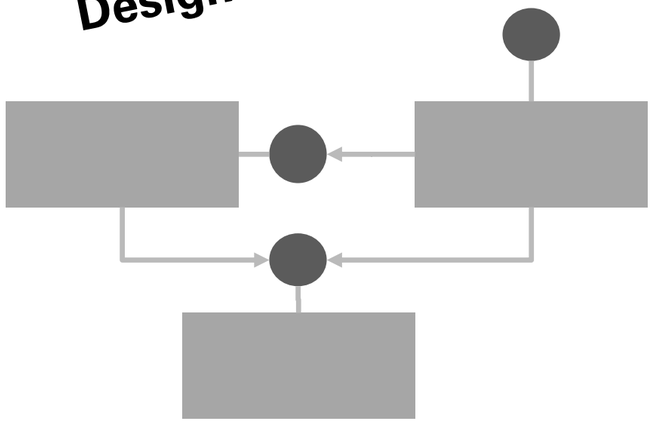
# Industrialisierte Migration aller Deployment Artefakte für eine schnelle und einheitliche Containerisierung.



# Cloud-native Anwendungsentwicklung: Komponenten entlang das kompletten Software Lebenszyklus.

DESIGN | BUILD | RUN

**Design Components**



1:1

**Dev Components**

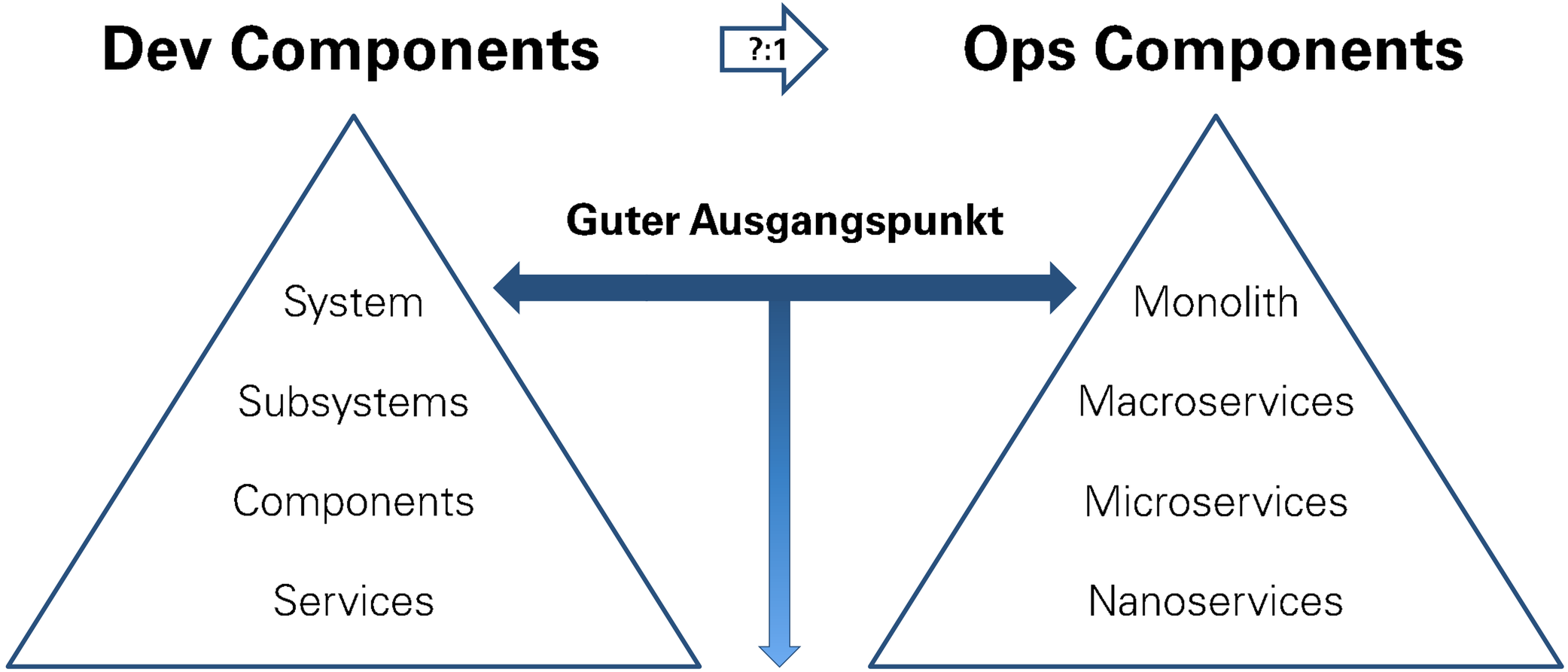

?:1

**Ops Components**

- Complexity unit
- Data integrity unit
- Coherent and cohesive feature unit
- Decoupled unit

- Planning & Assignment unit
- Knowledge unit
- Development unit
- Integration unit

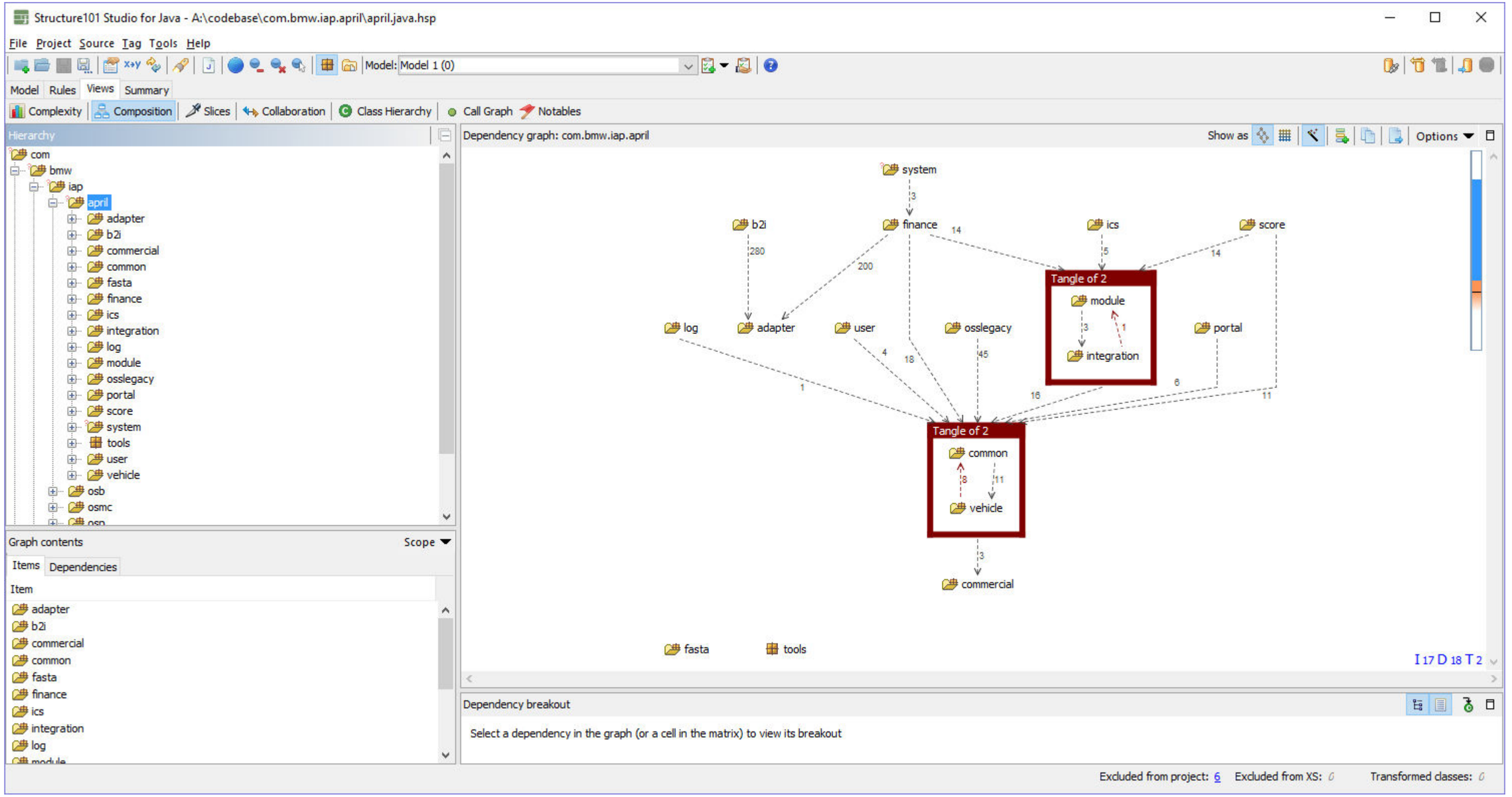
- Release unit
- Deployment unit
- Runtime unit (crash, slow-down, access)
- Scaling unit



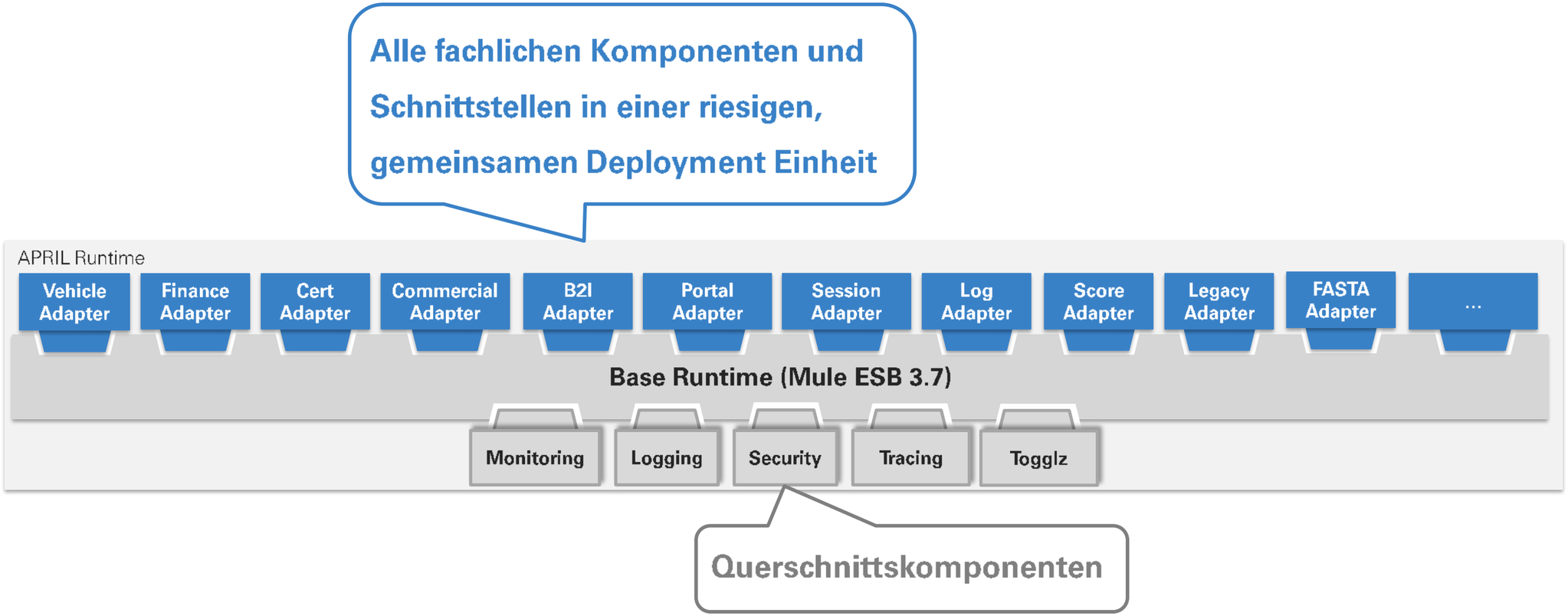

**Decomposition Trade-Offs**

| | |
|---|---|
| + Flexiblere Skalierung möglich | — Verteilungsschulden: Latenz |
| + Runtime Isolation (Crash, Slow-Down, …) | — Steigende Infrastruktur Komplexität |
| + Unabhängige Releases, Deployments, Teams | — Steigende Troubleshooting Komplexität |
| + Bessere Ressourcennutzung | — Steigende Integration-Komplexität |

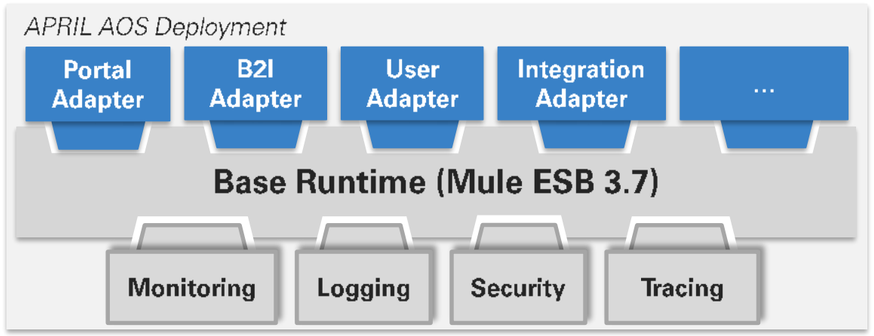
# Logische Sicht auf eine fachliche Paketstruktur.

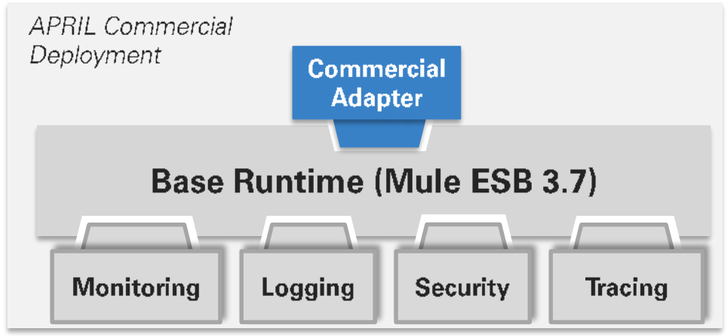


# „Decomposing the Monolith“ am Beispiel von APRIL.

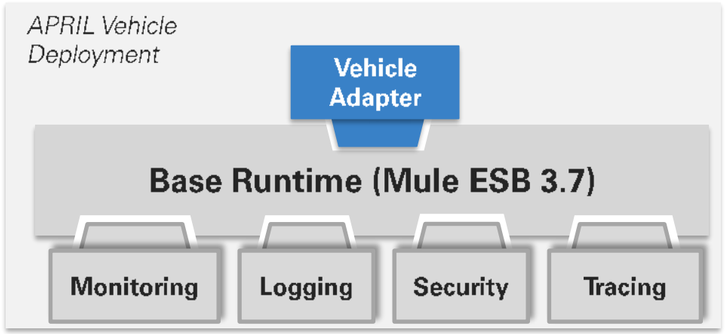


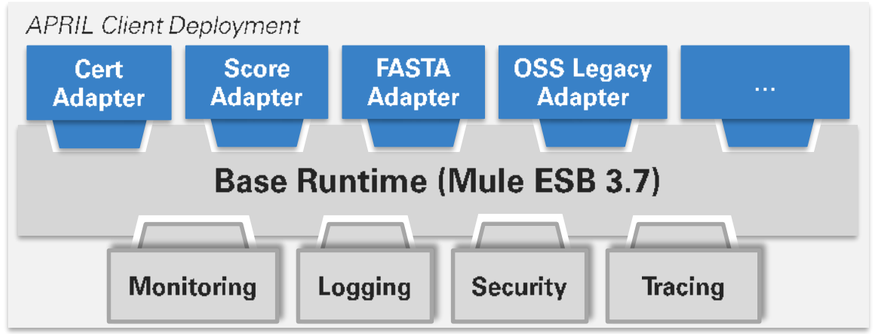
# „Decomposing the Monolith“ am Beispiel von APRIL.

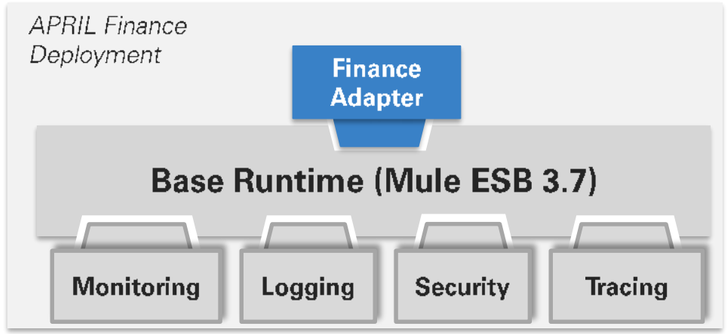



*Ein Deployment pro System-Context*

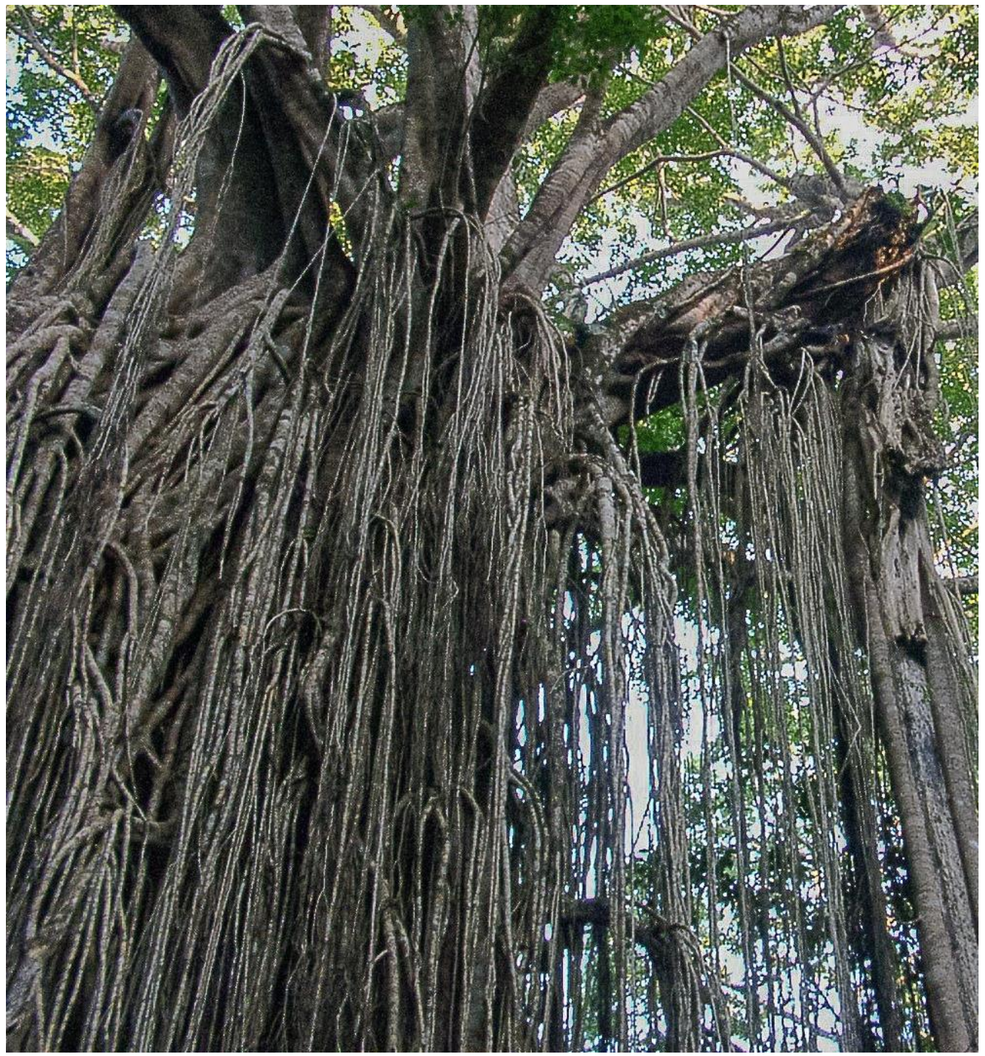


# Schrittweise Evolution und Cloud-nativer Neubau mit dem Strangler Pattern.

- **Transform**: ein Teil der Bestandsfunktionalität wird extrahiert und in ein neues, modernes System eingebaut.
- **Coexist**: beide Systeme koexistieren für eine Zeit. Aufrufe der alten Funktionalität werden umgeleitet.
- **Eliminate**: alte Funktionalität wird aus dem Bestandssystem entfernt sobald diese nicht mehr genutzt wird.
- Eignet sich besonders für Web- oder API-Monolithen.
- Problematisch bei Non-RESTful URL Strukturen.



https://martinfowler.com/bliki/StranglerApplication.html

# Performance

# Define Resource Constraints carefully.

```yaml
resources:
  # CPU is specified in units of cores
  # Memory is specified in units of bytes

  # required resources for a Pod to be scheduled and started
  requests:
    memory: "128Mi"
    cpu: "1"

  # the Pod will be restarted if limits are exceeded
  # so be careful not to set them too low!
  limits:
    memory: "1Gi"
    cpu: "2"
```

# Tame and Tune your JVM!

```
-XX:+UnlockExperimentalVMOptions -XX:+UseCGroupMemoryLimitForHeap
-XX:ParallelGCThreads=2  -XX:ParallelGCThreads=2

-server
-Xmx320m -Xss256k -XX:MaxMetaspaceSize=160m -XX:CompressedClassSpaceSize=32m

# Do not use G1GC?
-XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:NewRatio=1 -XX:+CMSParallelRemarkEnabled

# Use for small heaps on 64-bit VMs
-XX:+AggressiveOpts -XX:+UseCompressedOops -XX:+UseCompressedClassPointers

# optional
-XX:+UnlockDiagnosticVMOptions -XX:NativeMemoryTracking=summary
```

Since jdk8_131

Extra memory settings

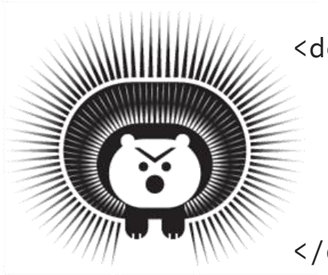
GC tuning.

Fancy tuning.

Diagnostics.

# Resiliency

# Retrofitting resiliency using Netflix Hystrix is easy.



```
<dependencies>
    <dependency>
        <groupId>com.netflix.hystrix</groupId>
        <artifactId>hystrix-core</artifactId>
        <version>${hystrix.version}</version>
    </dependency>
</dependencies>
```

- Use Netflix Hystrix for the resilient (synchronous) call of any external system
- Circuit Breaker and Bulk Heading implementation
- Easy integration with any JEE7 application
  - Can be used easily with Jersey Client for REST Calls
  - Can be integrated easily with JSR 236 Concurrency API via HystrixConcurrencyStrategy
- Integrates seemlessly with Dropwizard Metrics

# Diagnosability

# Liveness and Readiness Probes for Metrics endpoints.

```
# container will receive requests if probe succeeds
readinessProbe:
  httpGet:
    path: /admin/ping
    port: 8080
  initialDelaySeconds: 30
  timeoutSeconds: 5

# container will be killed if probe fails
livenessProbe:
  httpGet:
    path: /admin/healthcheck
    port: 8080
  initialDelaySeconds: 60
  timeoutSeconds: 10
```

# Retrofitting metrics, health and admin endpoints using the Dropwizard Metrics library in 30 minutes.

```
<dependencies>
    <dependency>
        <groupId>io.dropwizard.metrics</groupId>
        <artifactId>metrics-core</artifactId>
        <version>${metrics.version}</version>
    </dependency>
</dependencies>
```

- Usage of Dropwizard Metrics to retrofit metrics, health and admin endpoints
- Easy integration with any JEE7 application
- Definition of Custom Health Checks
- Used as Liveness und Readiness Probes

```
<!--
http://metrics.dropwizard.io/3.1.0/manual/servlets/
-->

<servlet>
   <servlet-name>adminServlet</servlet-name>
   <servlet-class>
      com.codahale.metrics.servlets.AdminServlet
   </servlet-class>
</servlet>

<servlet-mapping>
   <servlet-name>adminServlet</servlet-name>
   <url-pattern>/admin/*</url-pattern>
</servlet-mapping>
```

**QAware GmbH München**

Aschauer Straße 32

81549 München

Tel.: +49 (0) 89 23 23 15 – 0

Fax: +49 (0) 89 23 23 15 – 129

twitter.com/qaware

linkedin.com/qaware

github.com/qaware

xing.com/qaware

slideshare.net/qaware

youtube.com/qawaregmbh