

---

# Vergleich eines Usecases mit Serverless Technologie gegenüber Spring Boot Technologie am Beispiel von Instant Payments

---

**Silas Hoffmann**

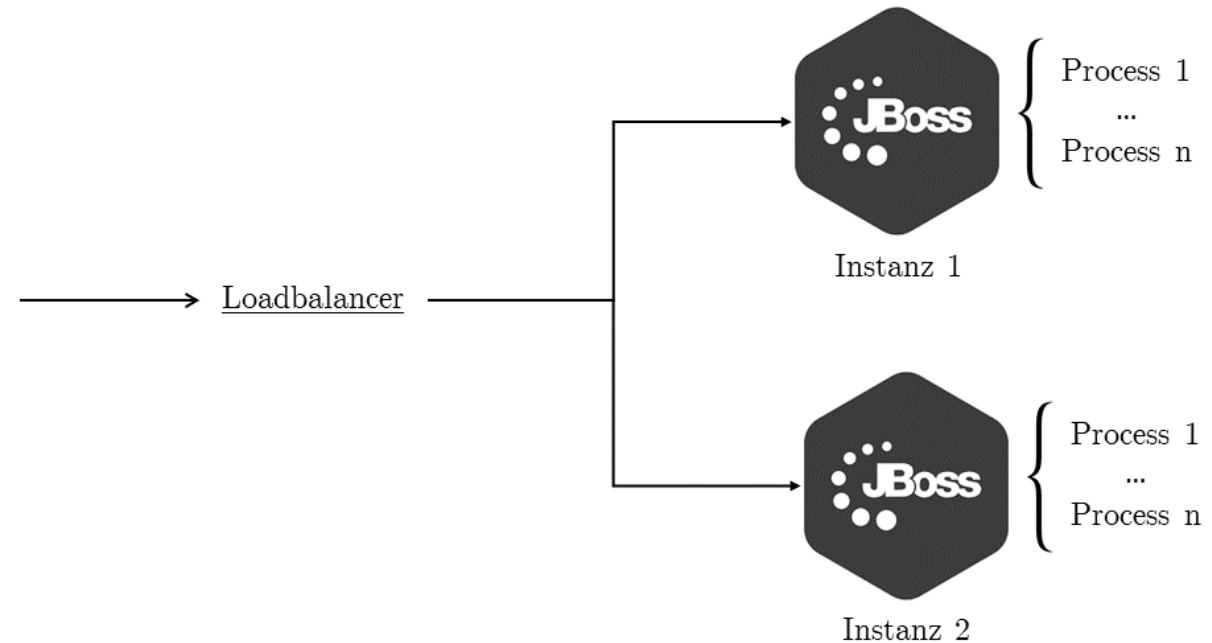
16 September 2021

# Inhalt

- Ist-Analyse
- Zielsetzung
- Vorgehensmodell
- Implementierung des Prototypen
- Optimierungspotenzial

# Ist-Analyse

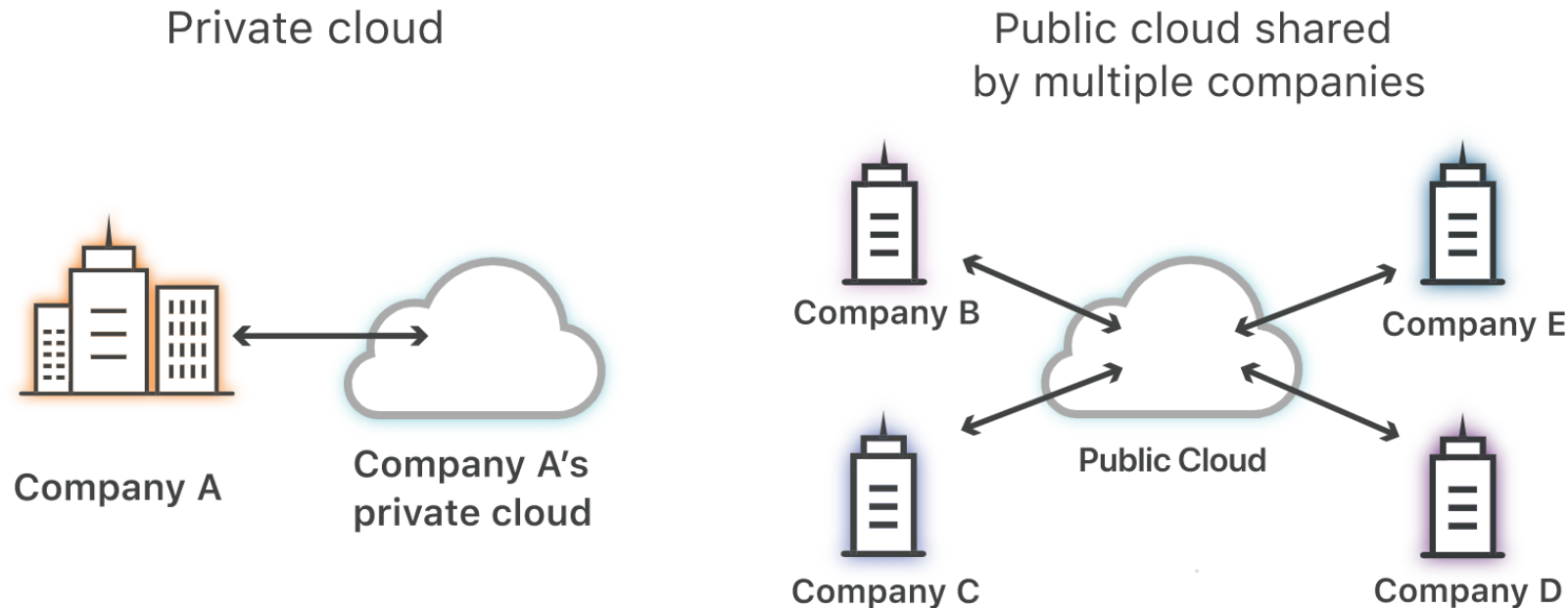
- Verwendet Application Server
- Aufteilung der Last durch Loadbalancer
- Request-Queue bei Überlauf befüllt
- Dynamische Prozessanzahl
- Monolith: Probleme
  - Skalierte Entwicklung
  - Unabhängiges Deployment
  - Skalierung innerhalb einer Produktivumgebung



Quelle: Hoffmann – Bsc. Thesis

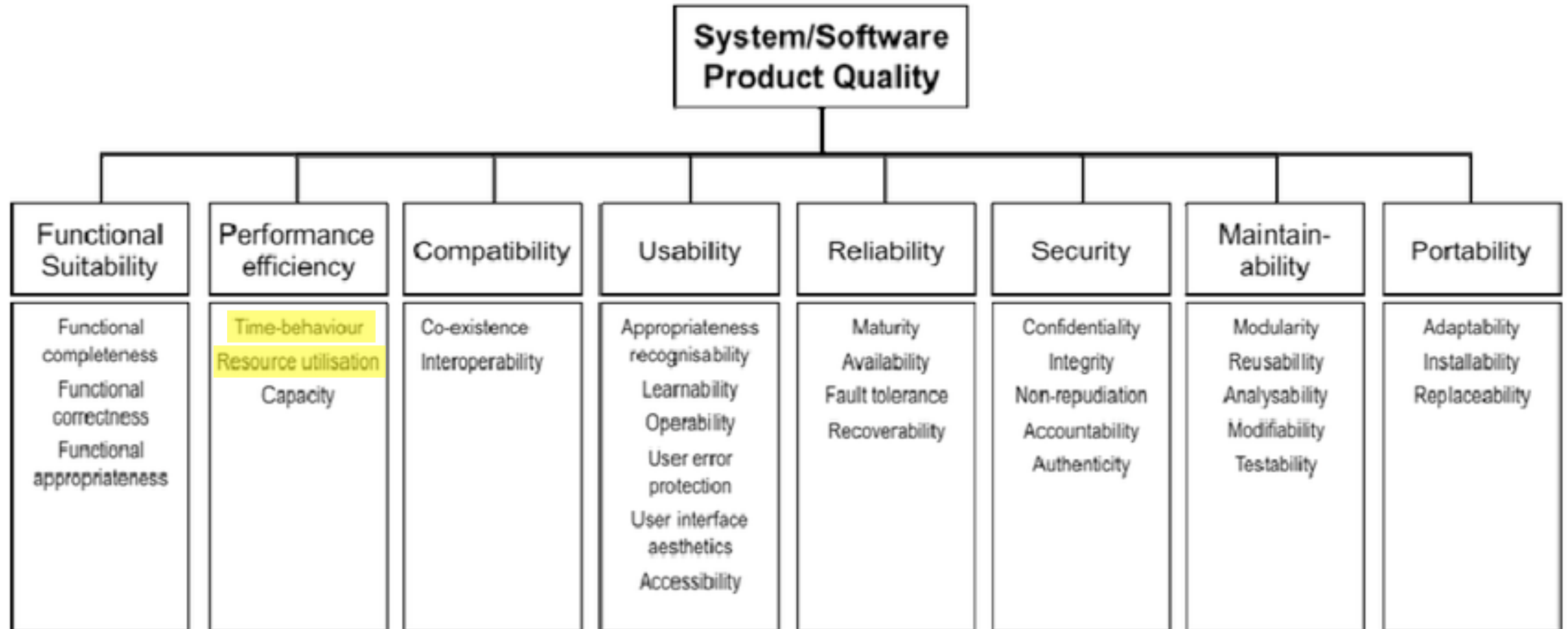
# Zielsetzung

- Cloudfähigkeit von Spring Boot und Serverless Tech. Vergleichen
- Container Startupzeiten / Verarbeitungsgeschwindigkeiten evaluieren
  - Startupzeiten müssen absolut minimal sein um fachliche Timeout bei Instant-Payments zu vermeiden (End-to-End max. 7 Sekunden)



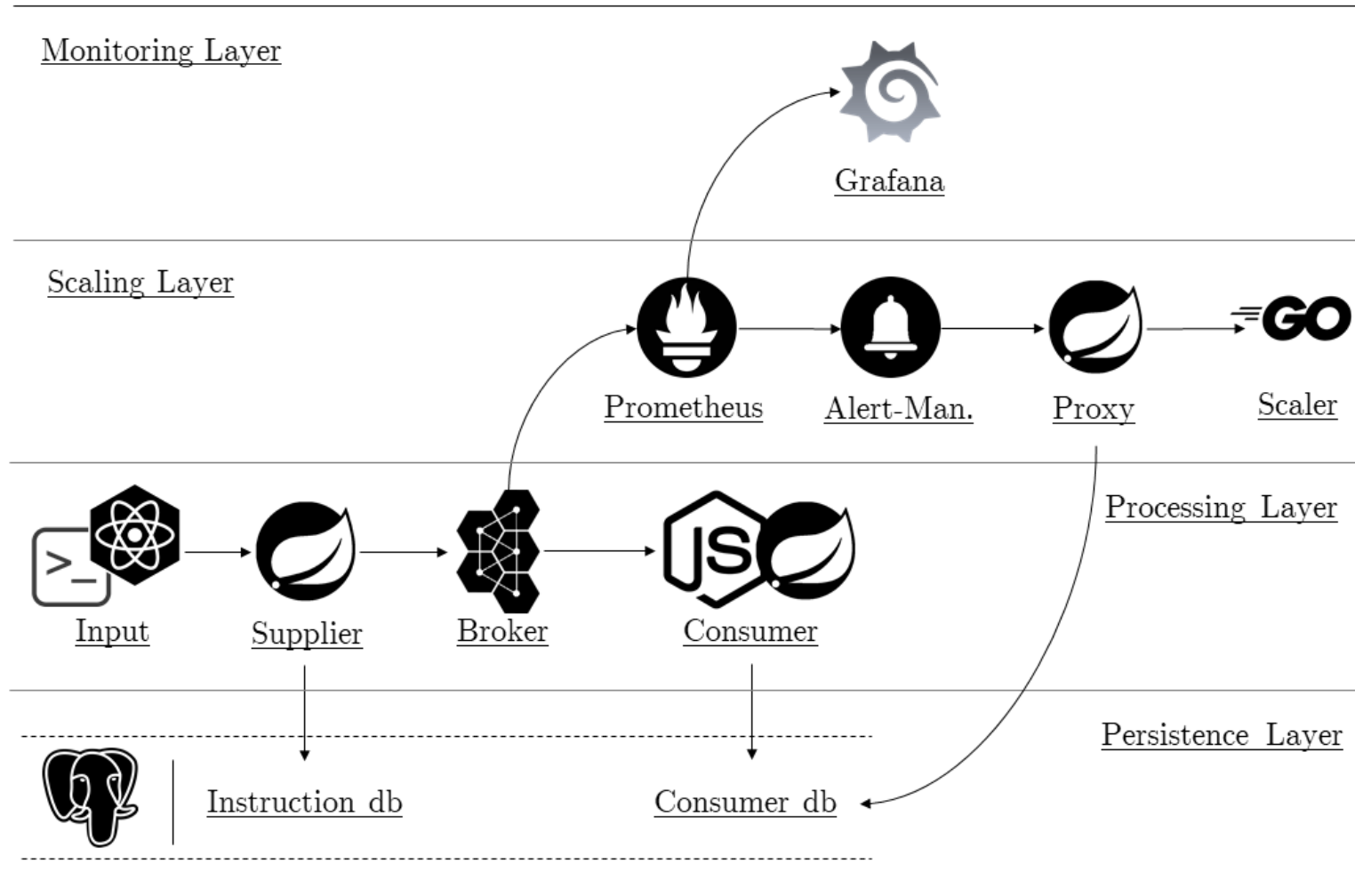
Quelle: <https://www.cloudflare.com/de-de/learning/cloud/what-is-a-public-cloud/>

# Vorgehensmodell



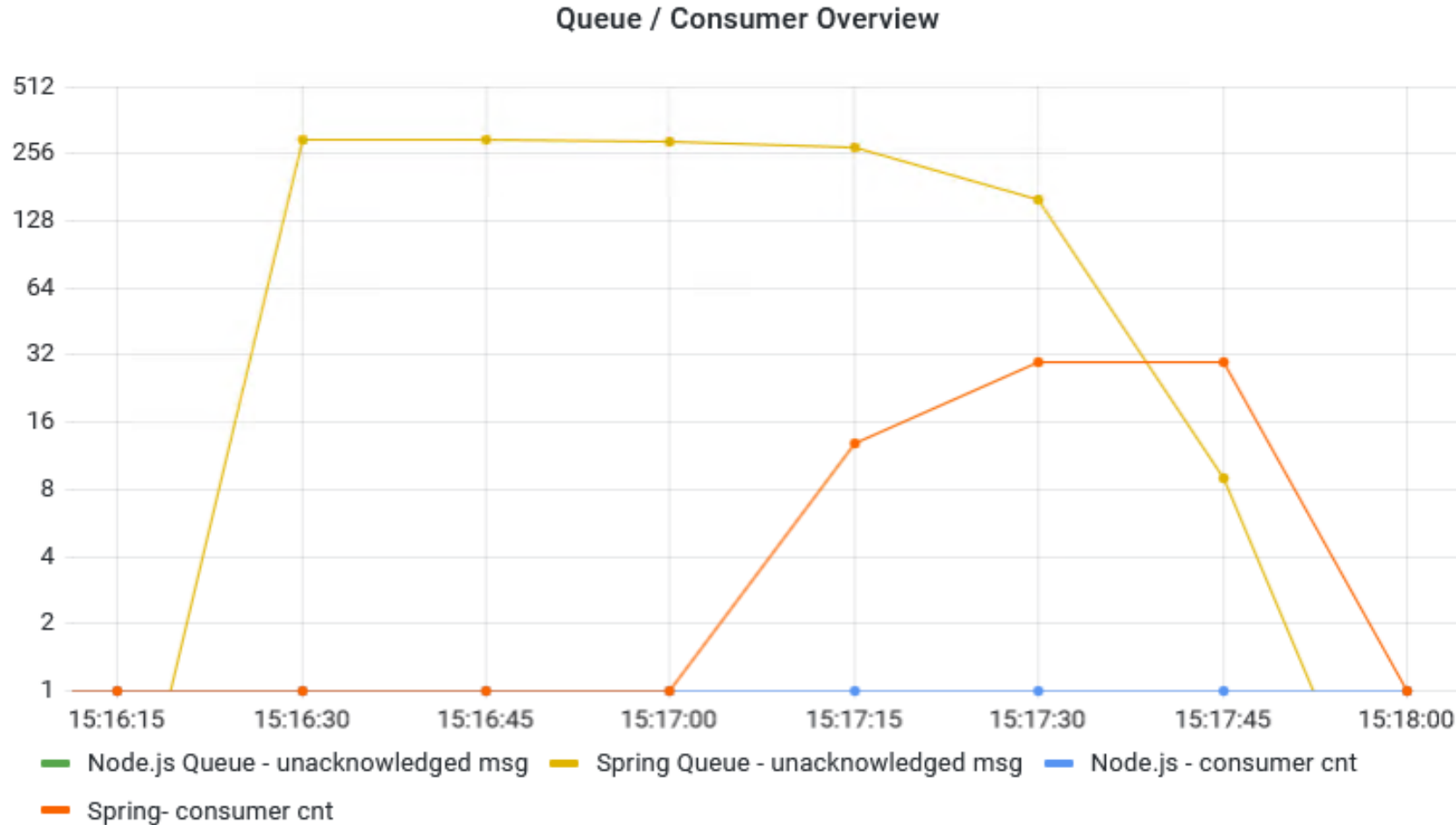
Quelle: ISO/IEC 25010

# Implementierung des Prototypen



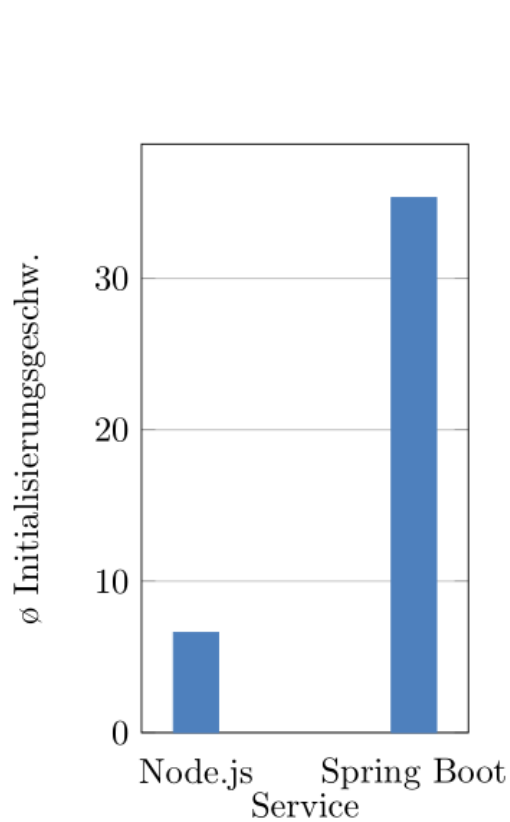
Quelle: Hoffmann – Bsc. Thesis

# Implementierung des Prototypen

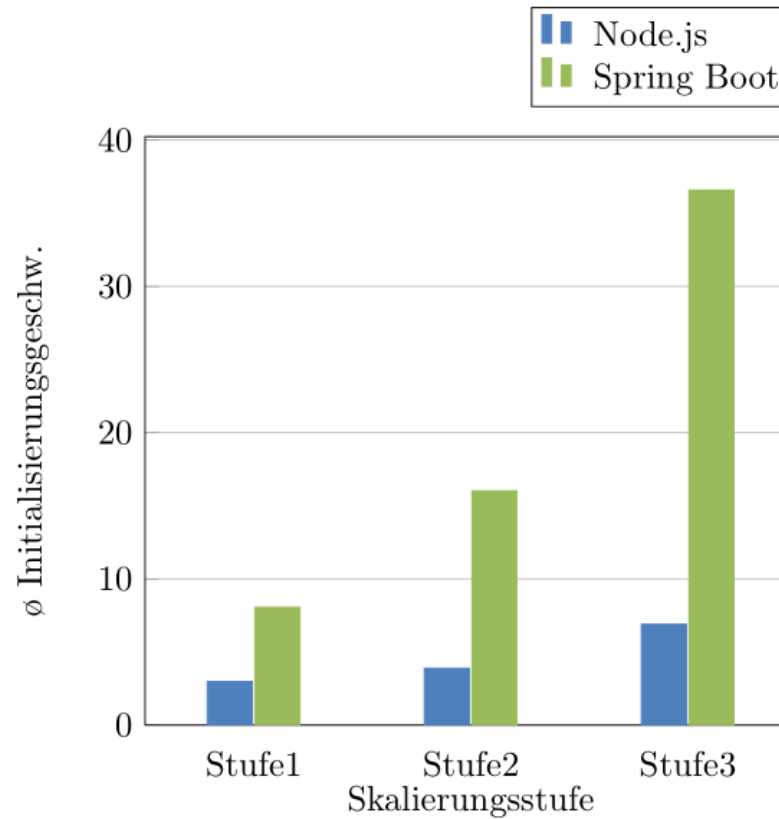


Quelle: Hoffmann – Bsc. Thesis

# Ergebnisanalyse / Fazit

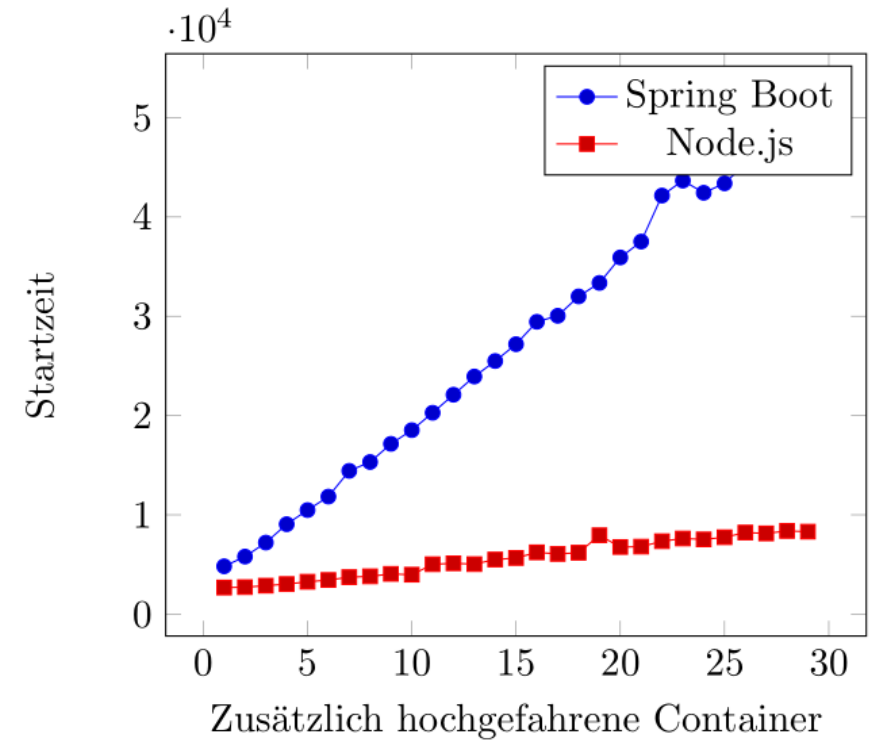


- Node.js: 6,9 Sek.
- Spring: 36,6 Sek.



Stufen ab denen Skalierer neue Instanzen startet

- Stufe 1: 15 Msg. -> 5 Container
- Stufe 2: 30 Msg. -> 10 Container
- Stufe 3: 100 Msg. -> 30 Container



Quelle: Hoffmann – Bsc. Thesis

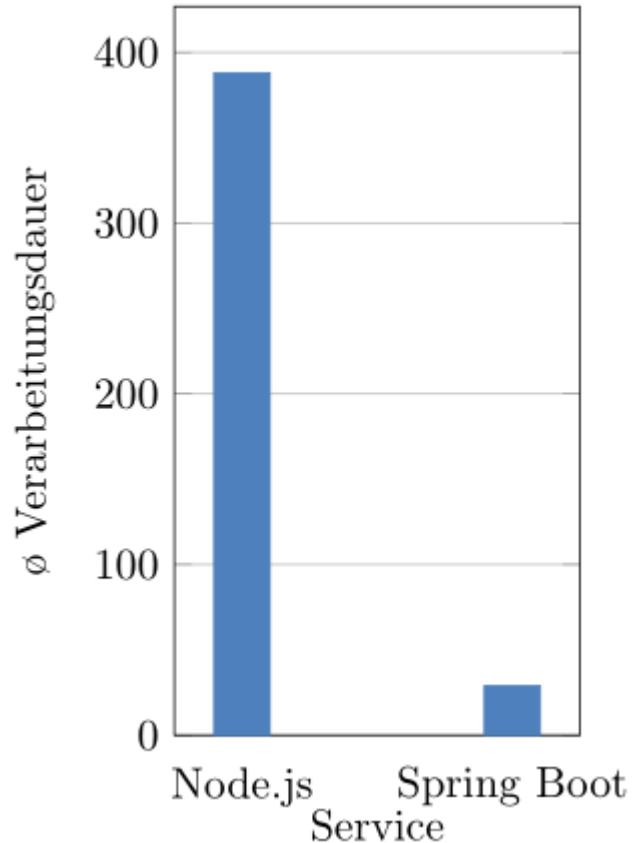
Linearer Anstieg:

- Node.js: 194 Millis pro Container
- Spring: 1611 Millis pro Container

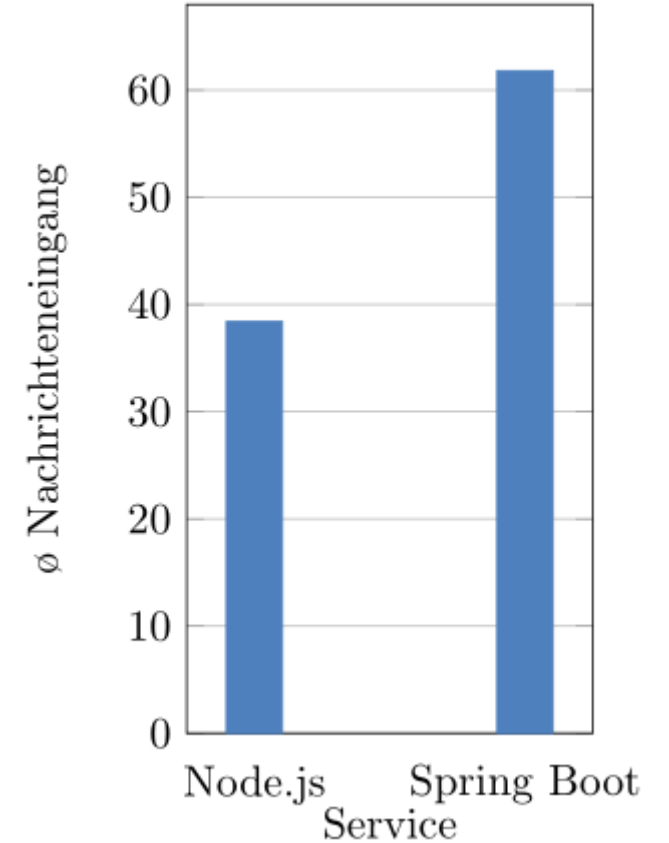


# Ergebnisanalyse / Fazit

- Node.js mit besserem Skalierungsverhalten
- Spring Boot mit besserer Verarbeitungsgeschwindigkeit
- Unterschied beim Nachrichteneingang vernachlässigbar



- Node.js: 388 Millis.
- Spring: 29 Millis.



- Node.js: 38,4 Sek.
- Spring: 61,8 Sek.

Quelle: Hoffmann – Bsc. Thesis

# Optimierungspotenzial

Docker	Spring
Ressourcenoptimierung	Spring-Bean - Optimierung der Initialisierungsphase
Ausführungsreihenfolge	
Design For Failure (chaos monkey etc.)	

---

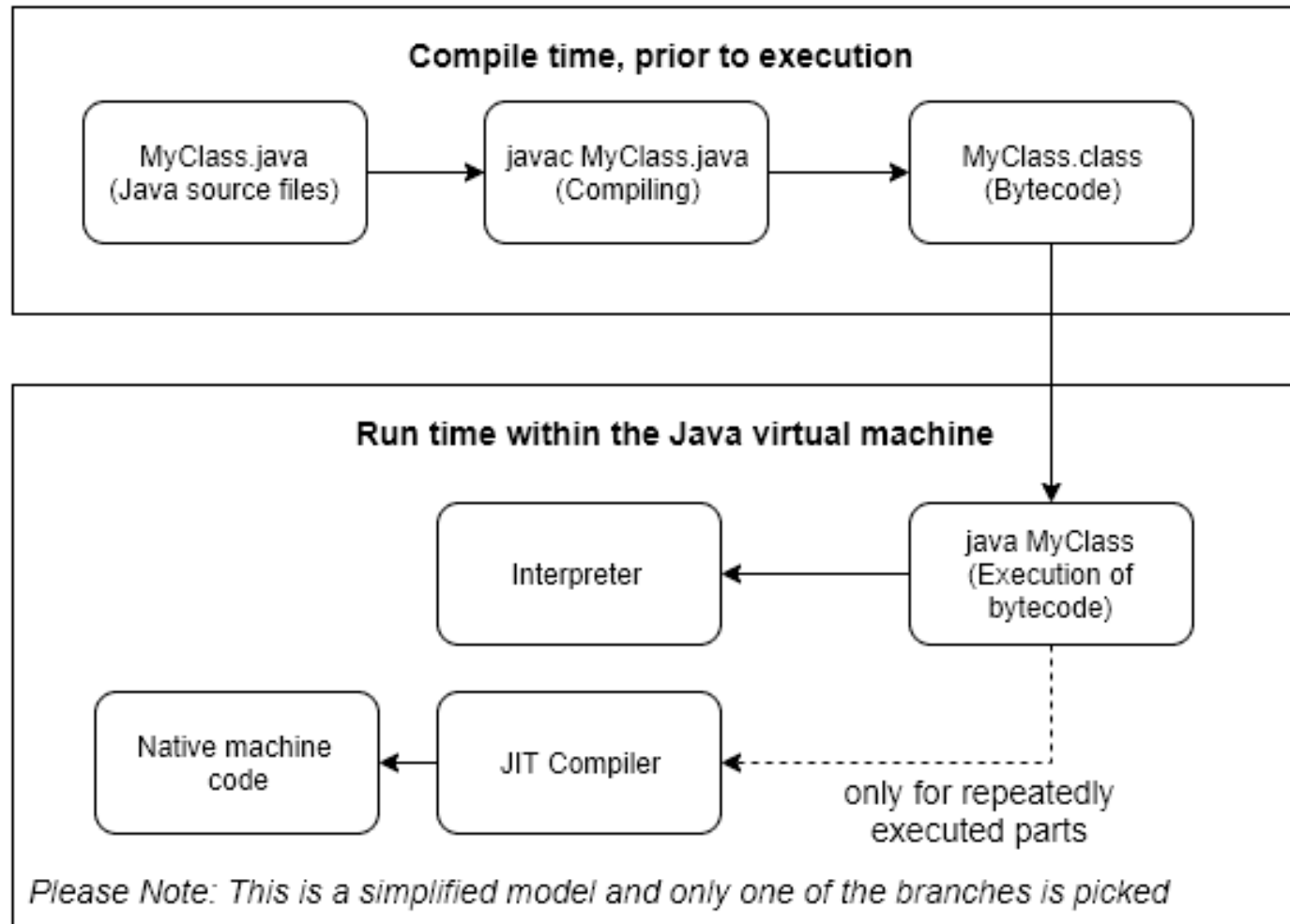
# Vergleich eines Usecases mit Serverless Technologie gegenüber Spring Boot Technologie am Beispiel von Instant Payments

---

**Silas Hoffmann**

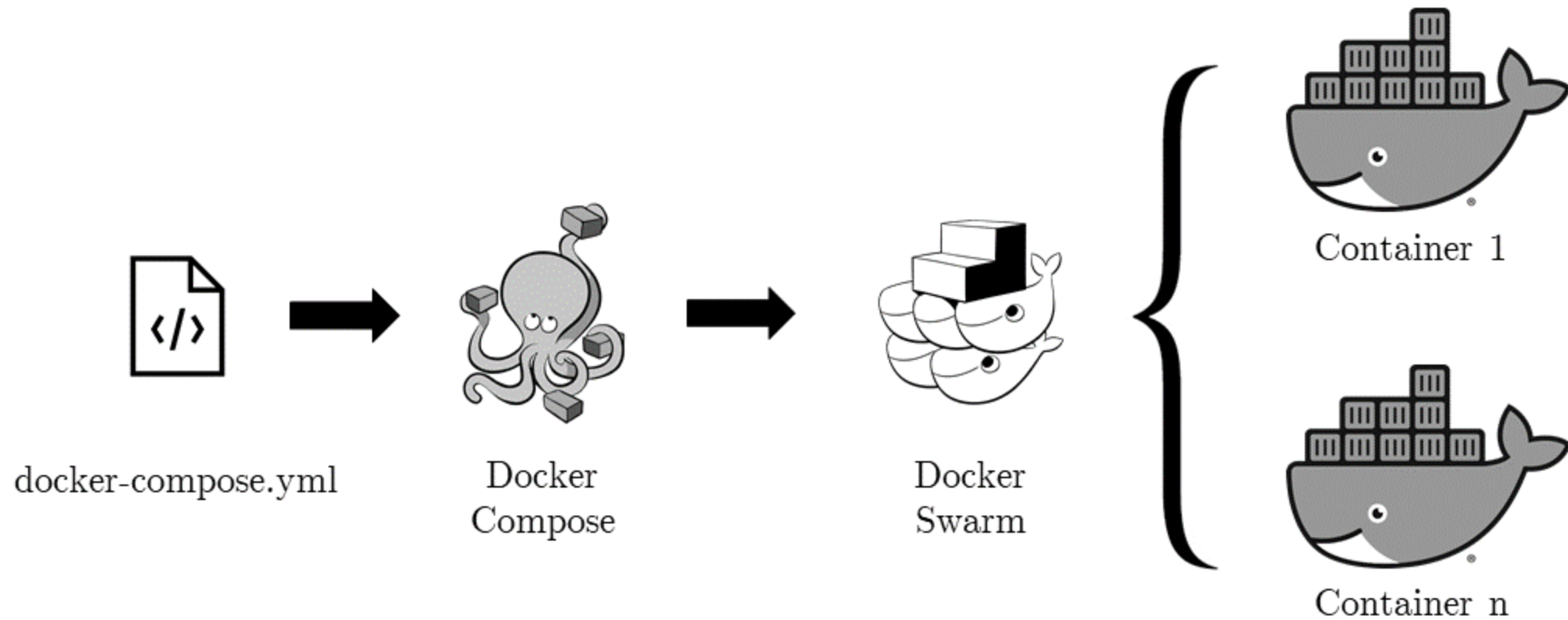
16 September 2021

# JVM (JIT Compiler)



<https://rieckpil.de/whatis-graalvm/>

# Implementierung des Prototypen



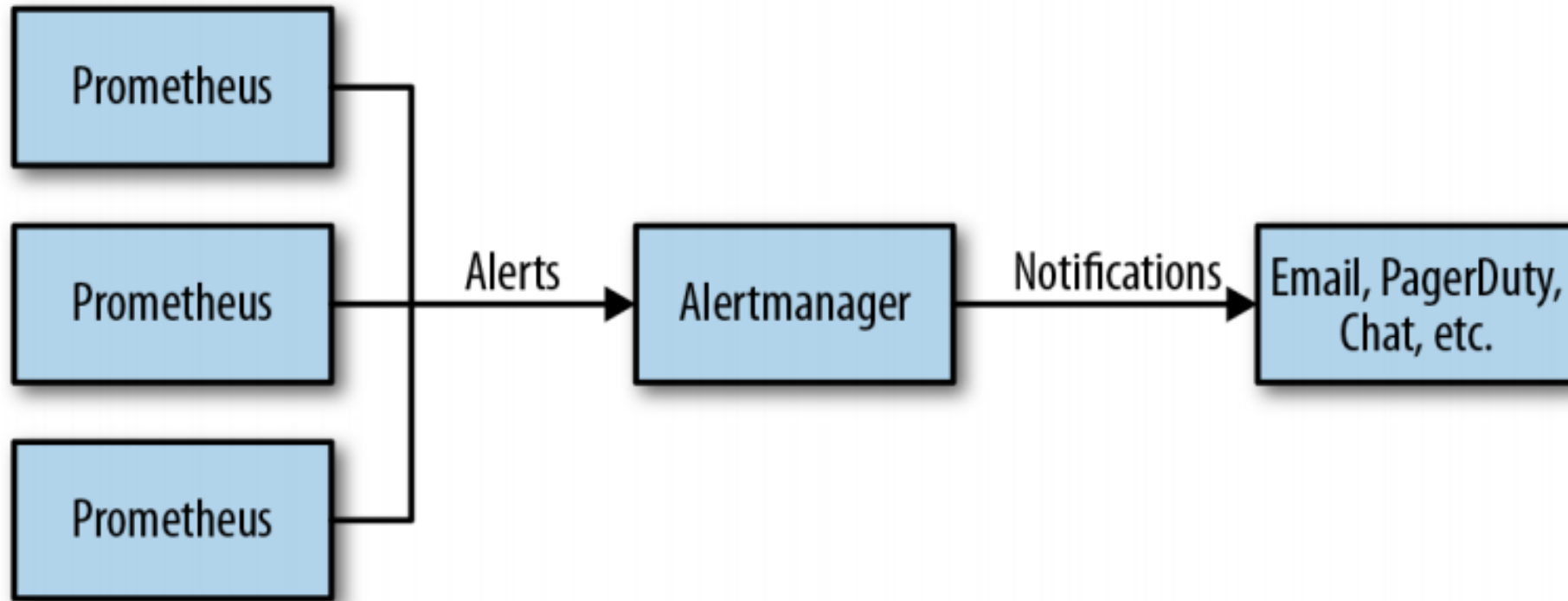
Quelle: Hoffmann – Bsc. Thesis

# Skalierung - Regelsatz

$\frac{QL3}{QB2 < MC}$	UP $abs(CB0 - CB3)$	UP $abs(CB1 - CB3)$	UP $abs(CB2 - CB3)$	OK –
$\frac{QL2}{QB1 < MC \leq QB2}$	UP $abs(CB0 - CB2)$	UP $abs(CB1 - CB2)$	OK –	DOWN $abs(CB2 - CB3)$
$\frac{QL1}{QB0 < MC \leq QB1}$	UP $abs(CB0 - CB1)$	OK –	DOWN $abs(CB1 - CB2)$	DOWN $abs(CB1 - CB3)$
$\frac{QL0}{MC \leq QB0}$	OK –	DOWN $abs(CB0 - CB1)$	DOWN $abs(CB0 - CB2)$	DOWN $abs(CB0 - CB3)$
	$\frac{CL0}{CB0 == CC}$	$\frac{CL1}{CB0 < CC \leq CB1}$	$\frac{CL2}{CB1 < CC \leq CB2}$	$\frac{CL3}{CB2 < CC \leq CB3}$

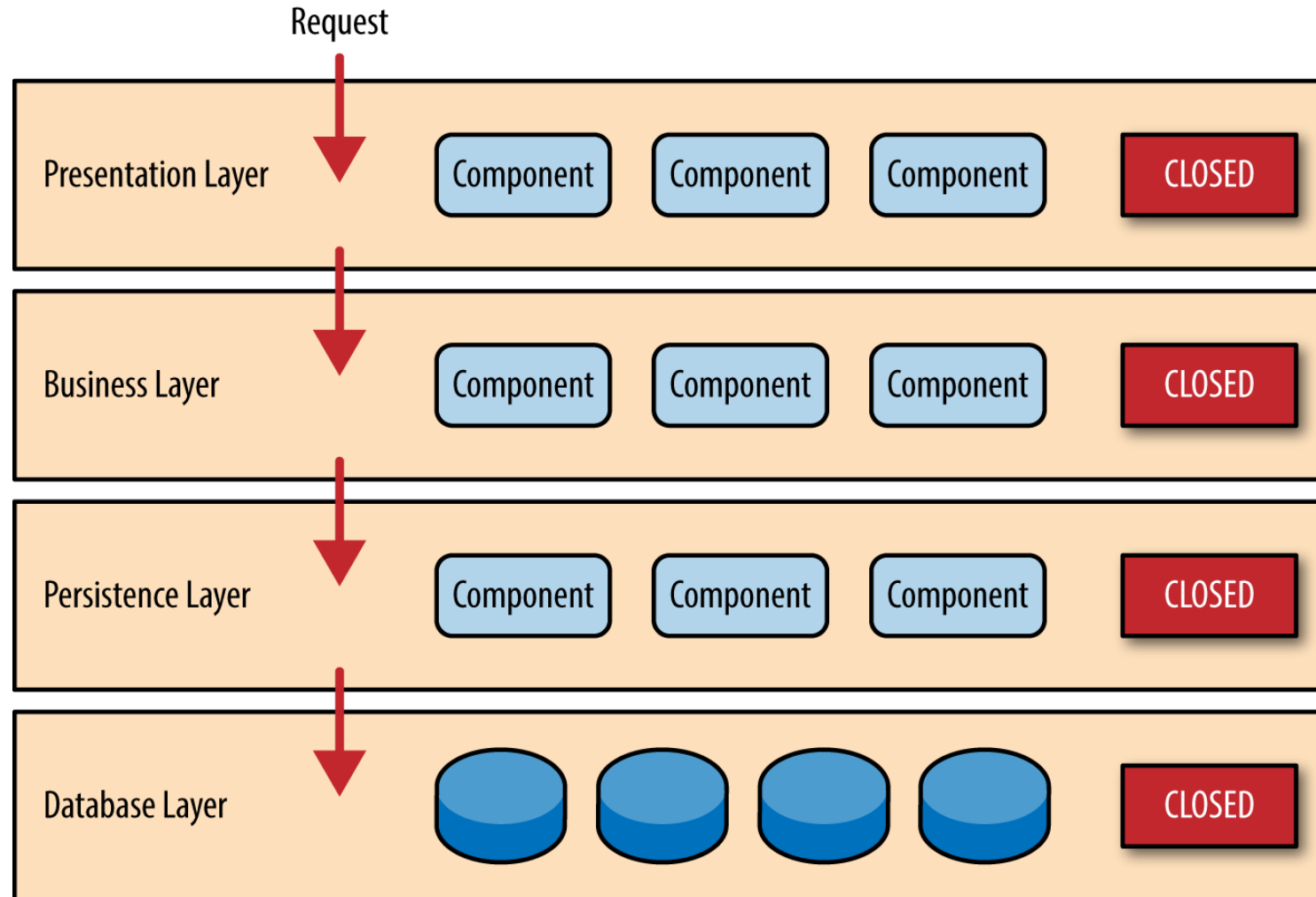
CB0=1    CB2=10    QB0=15    QB2=100    CC: Container Count  
 CB1=5    CB3=30    QB1=30                    MC: Message Count

# Prometheus / Alertmanager



Quelle: Brazil – Prometheus: Up & Running (S. 291)

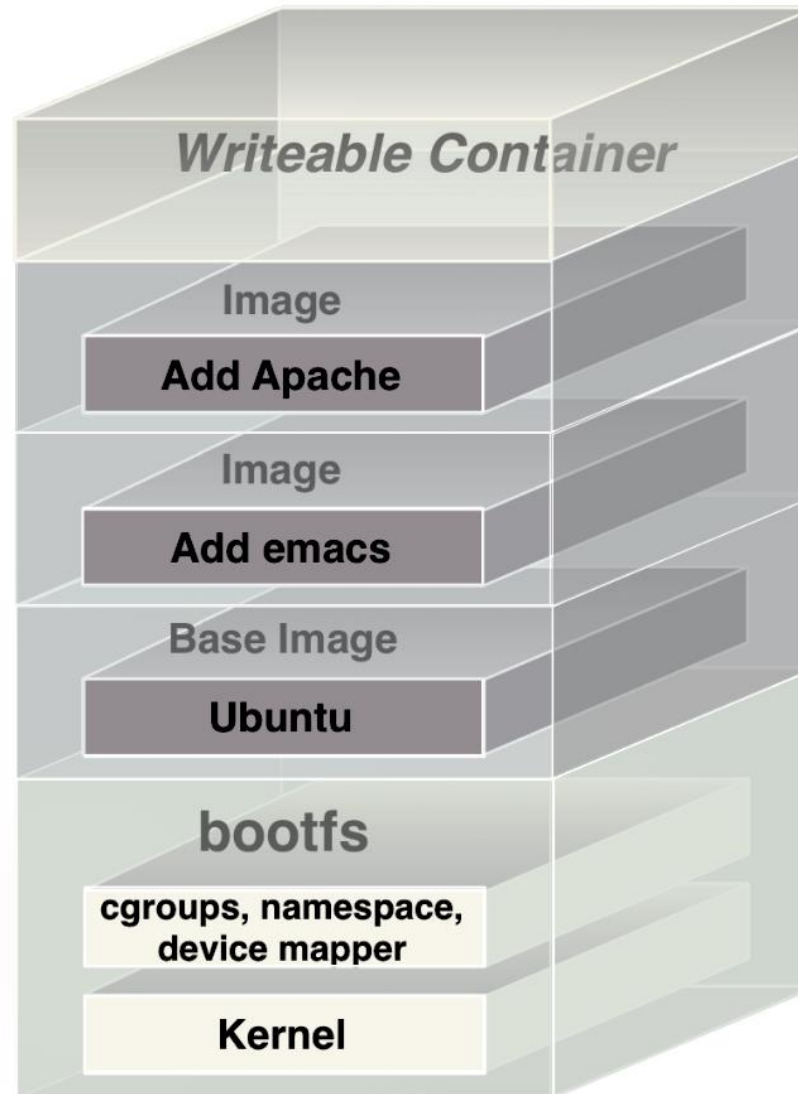
# Tier - Modell



Quelle: <https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html>

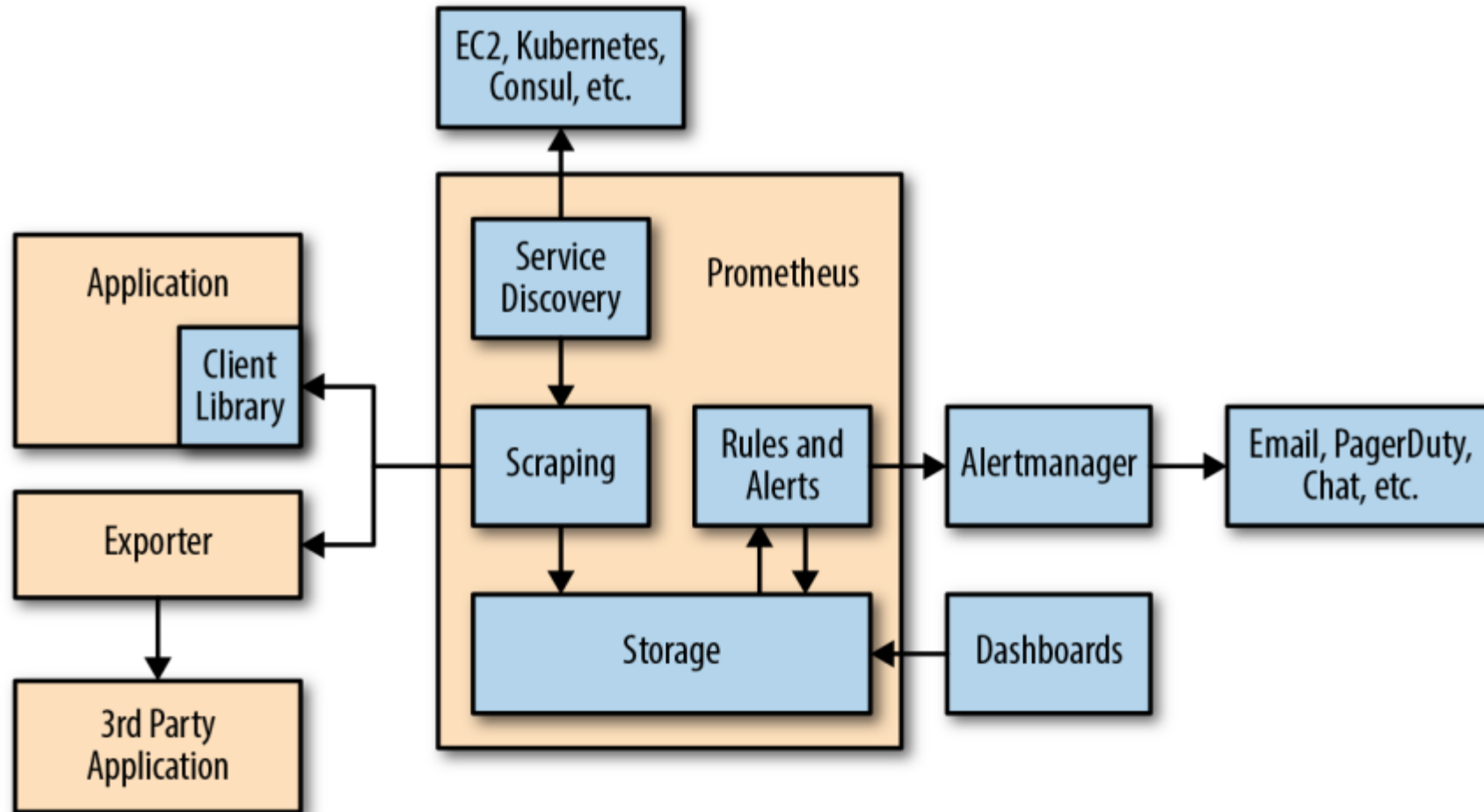


# Docker - Aufbau

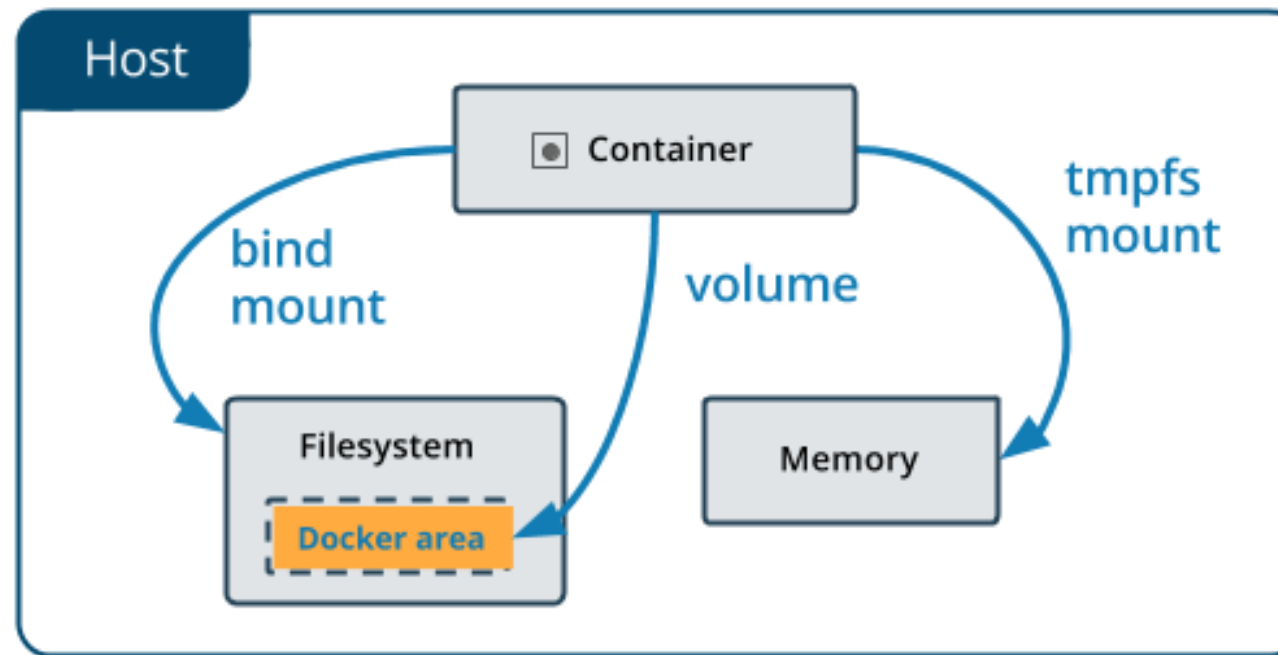


Quelle: J. Turnbull – The Docker Book (S. 72)

# Prometheus - Architecture

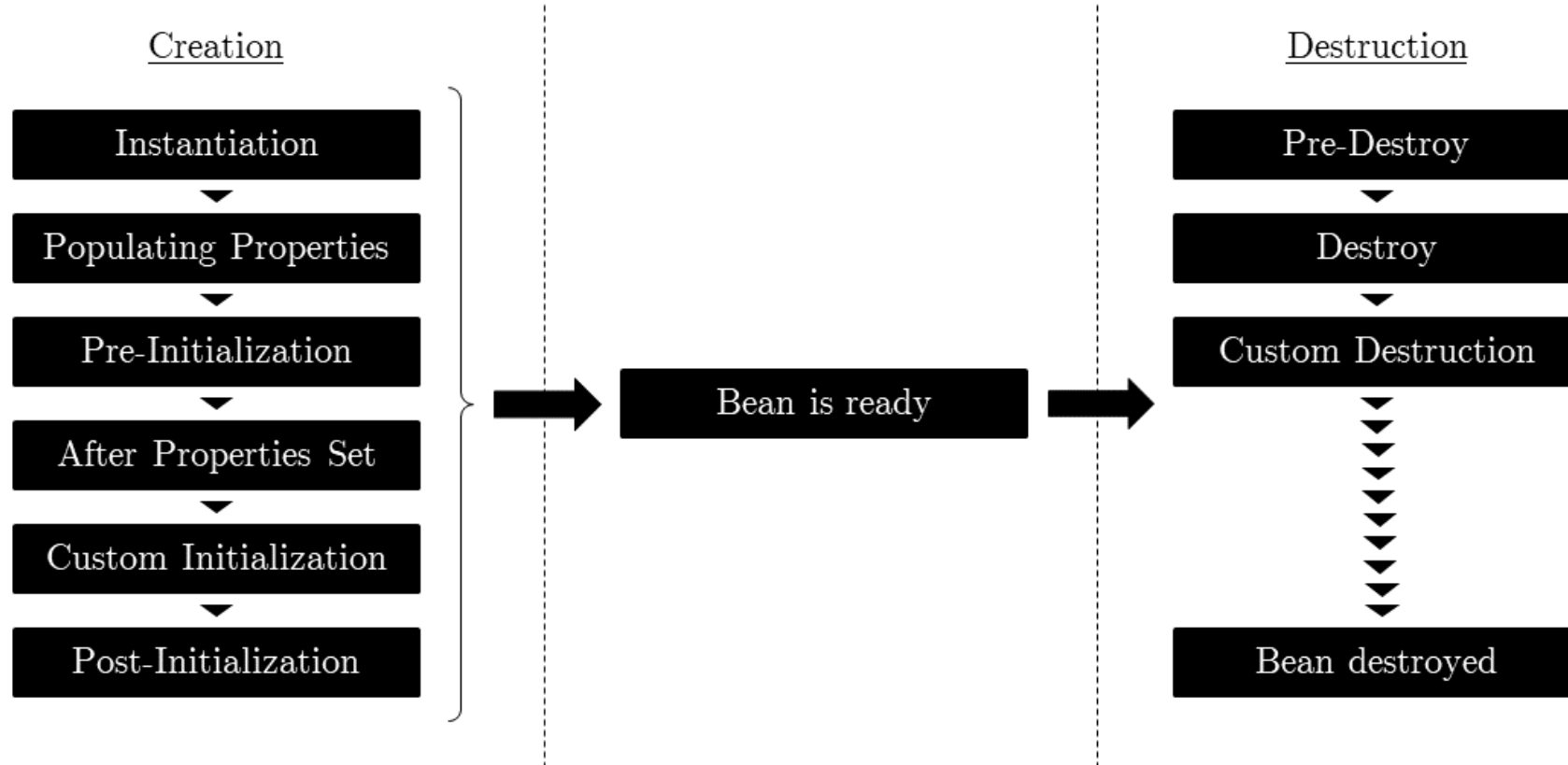


# Docker – Types of mounts



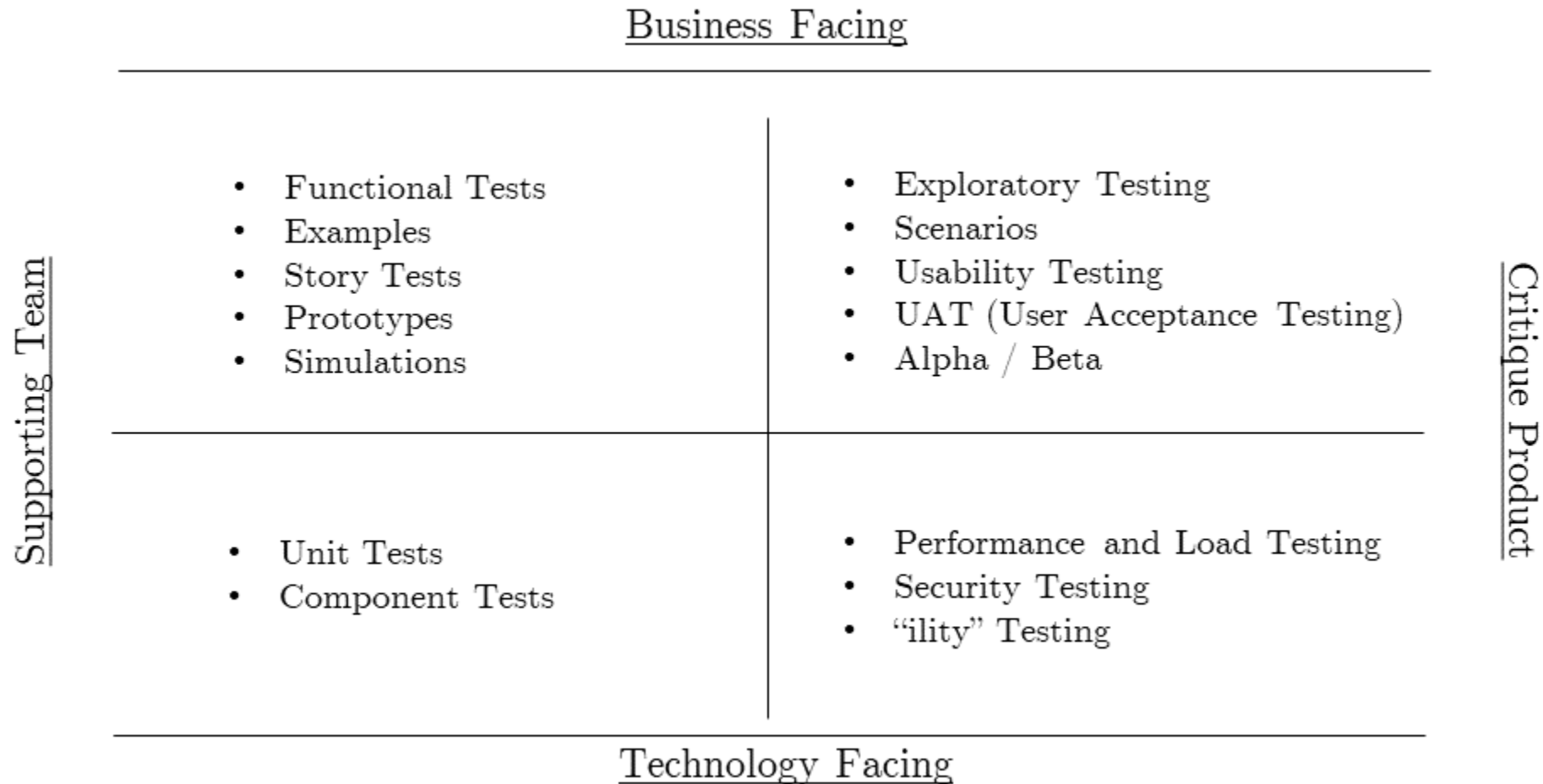
Quelle: Docker Documentation - Kapitel /storage/volumes/

# Spring Bean - Lifecycle



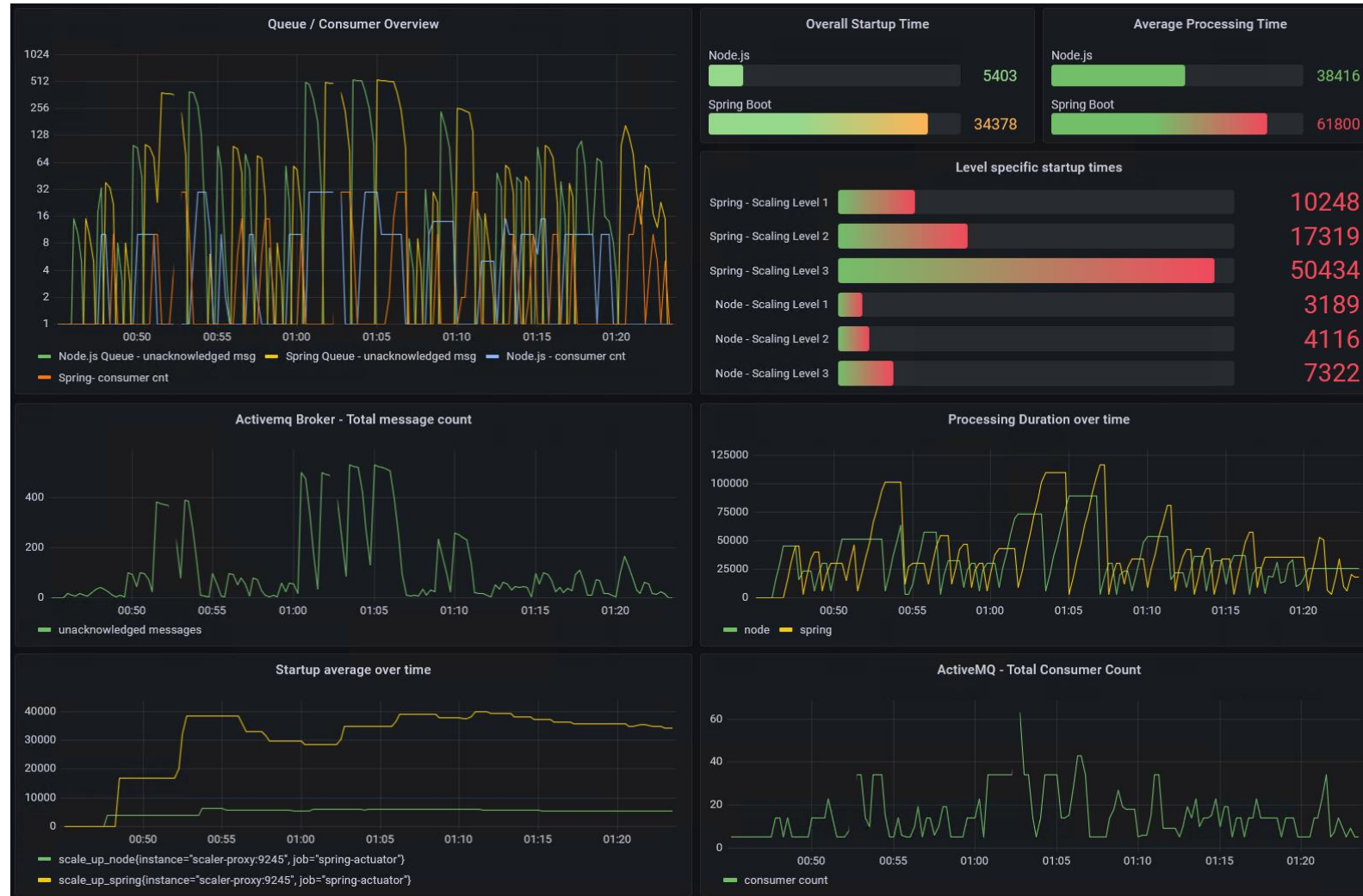
Quelle: Hoffmann – Bsc. Thesis

# Agile Testing Quadrants





Quelle: Hoffmann – Bsc. Thesis

# Implementierung des Prototypen



Quelle: Hoffmann – Bsc. Thesis

# Activemq - Dashboard













Home | Queues | Topics | Subscribers | Connections | Network | Scheduled | Send

Support

Queue Name  Create Queue Name Filter  Filter

### Queues:

Name ↑	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views	Operations
nodeack	0	1	205	205	Browse Active Consumers Active Producers  	Send To Purge Delete
nodequeue	0	1	2765	2765	Browse Active Consumers Active Producers  	Send To Purge Delete
persistencequeue	0	1	5530	5530	Browse Active Consumers Active Producers  	Send To Purge Delete
springack	0	1	263	263	Browse Active Consumers Active Producers  	Send To Purge Delete
springqueue	0	1	2765	2765	Browse Active Consumers Active Producers  	Send To Purge Delete

#### Queue Views

- Graph
- XML

#### Topic Views

- XML

#### Subscribers Views

- XML

#### Useful Links

- Documentation
- FAQ
- Downloads
- Forums

Copyright 2005-2015 The Apache Software Foundation.

Quelle: Hoffmann – Bsc. Thesis

# Input UI

xpath

iban



payment

Randomize messages



Quantity

25



Timespan

0



Start Batch

Quelle: Hoffmann – Bsc. Thesis



# Grafana - PromQL

▼ **spring-queue-size** (Prometheus) ? 📄 👁 🗑 ⋮

[Metrics browser >](#)

```
org_apache_activemq_Broker_QueueSize{brokerName="localhost", destinationName="springqueue", destinationType="Queue", instance="activemq:8080", job="services"}
```

**Legend** ⓘ Spring Queue - unacknow ... **Min step** ⓘ  **Resolution** 1/1 ▼

**Format** Time series ▼ ☐ Instant ☐ Prometheus ⓘ ☐ Exemplars ⓘ

Quelle: Hoffmann – Bsc. Thesis

# Prometheus - Datasource

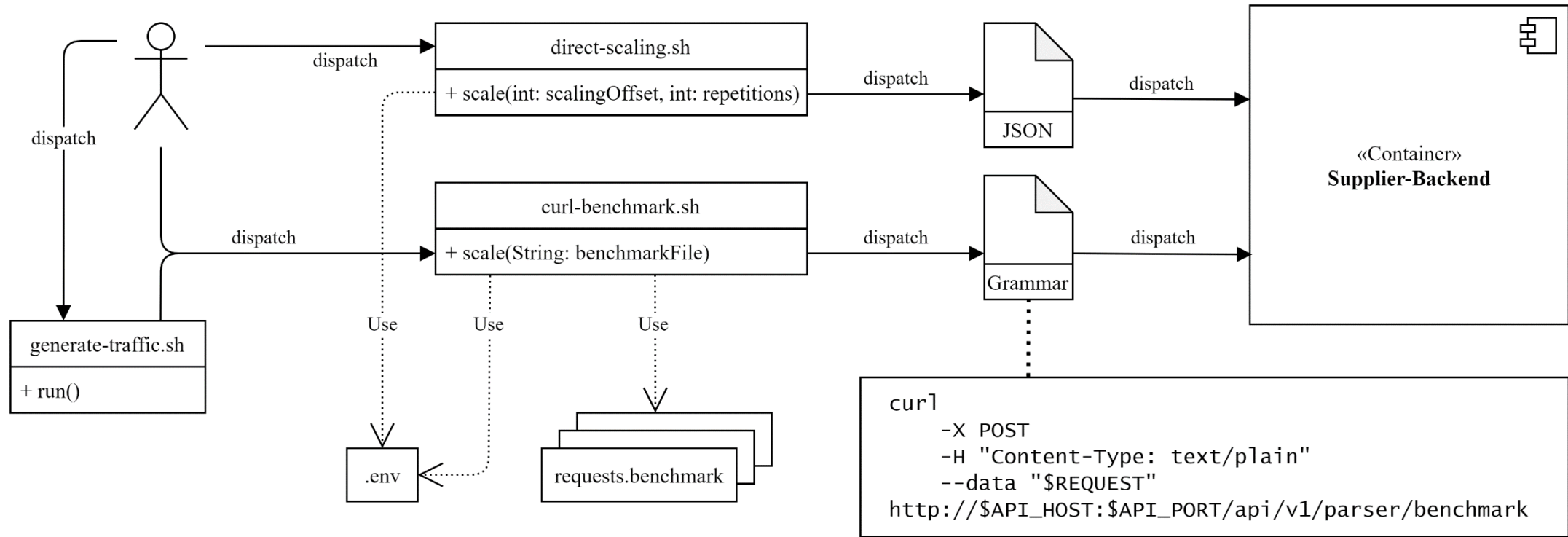
Name	<input type="text" value="Prometheus"/>	Default	<input checked="" type="checkbox"/>
------	---	---------	-------------------------------------

## HTTP

URL	<input type="text" value="http://prometheus:9090"/>	
Access	<input type="text" value="Server (default)"/>	<a href="#">Help &gt;</a>
Whitelisted Cookies	<input type="text" value="New tag (enter key to add)"/>	
Timeout	<input type="text"/>	

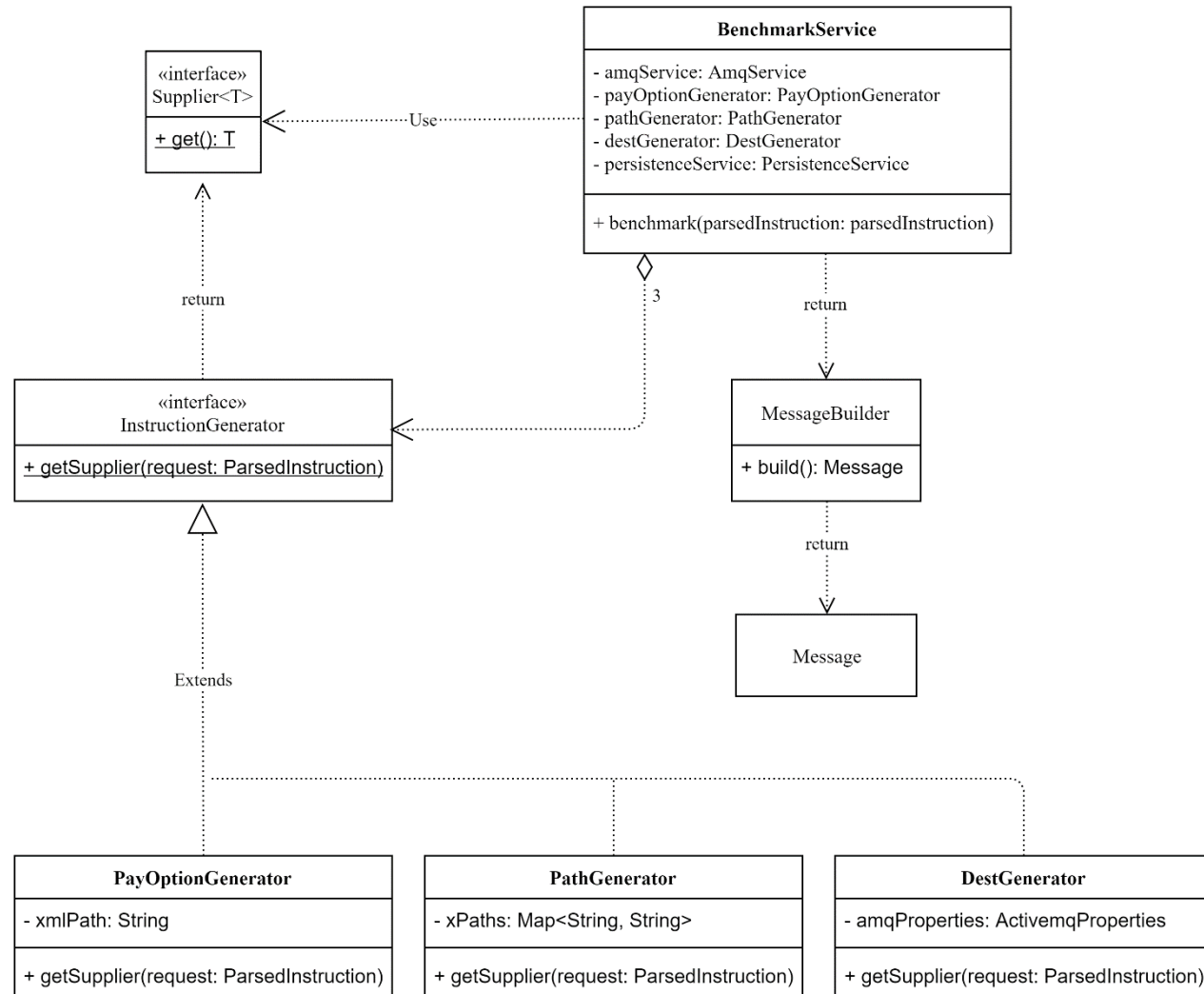
Quelle: Hoffmann – Bsc. Thesis

# Input - UML



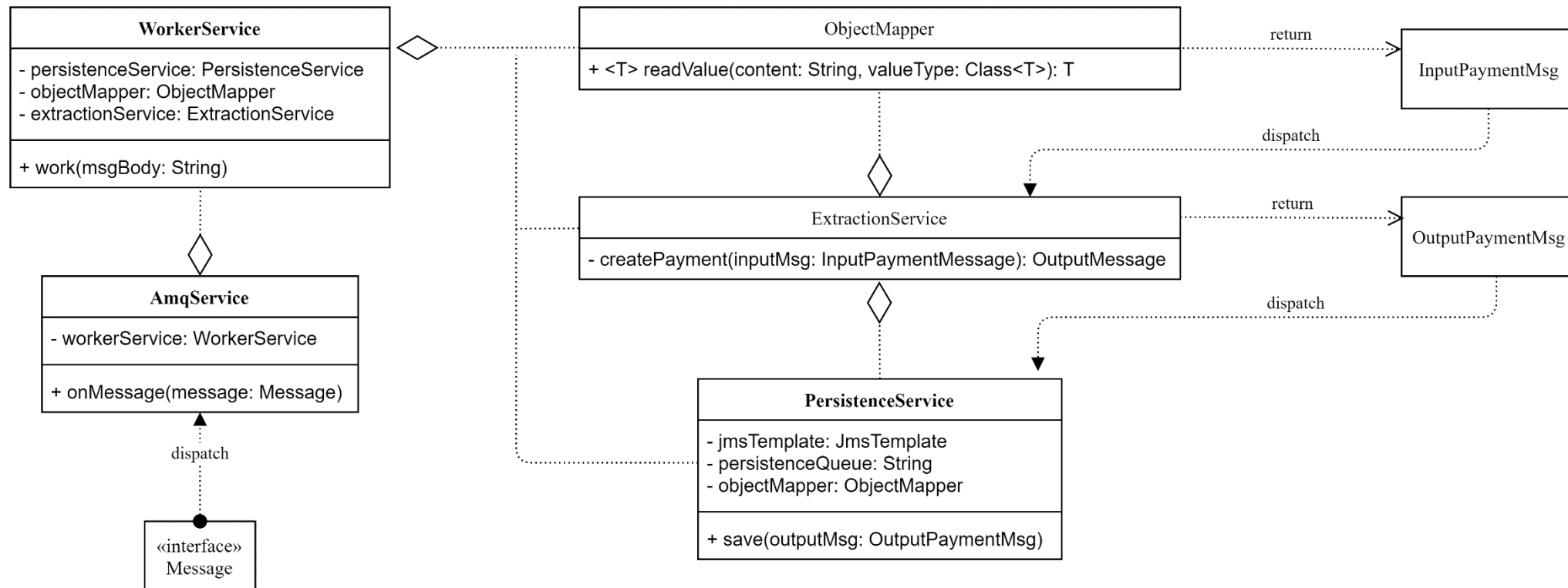
Quelle: Hoffmann – Bsc. Thesis

# Supplier - UML



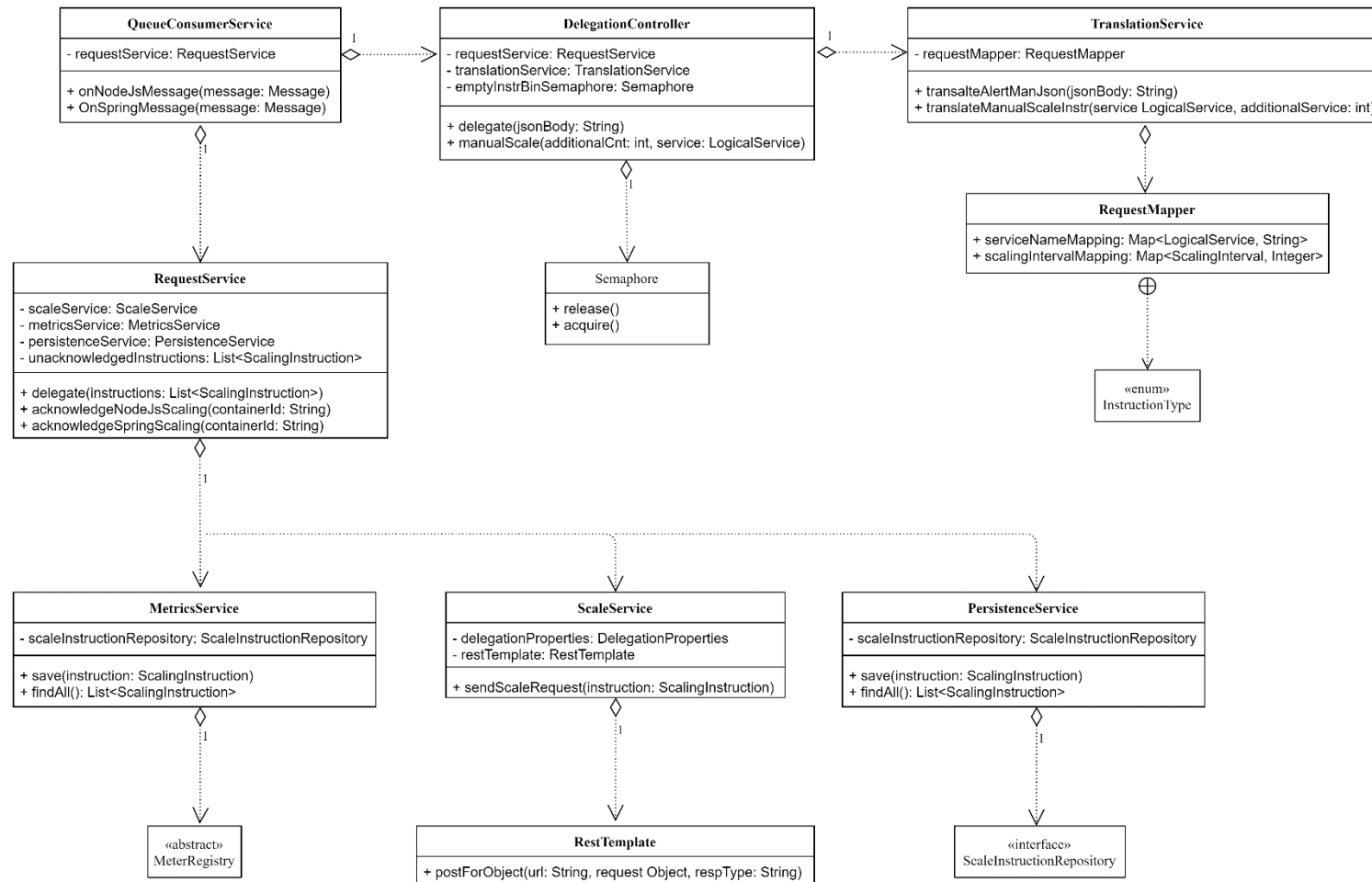
Quelle: Hoffmann – Bsc. Thesis

# Consumer - UML



Quelle: Hoffmann – Bsc. Thesis

# Scaler Proxy - UML



Quelle: Hoffmann – Bsc. Thesis