OSP

Otto Group Solution Provider

Serverless – Hype oder Mehrwert?

Kundenselektion für Rabattaktionen auf dem OTTO Marktplatz

Video Streaming

Scaling up the Prime Video audio/video monitoring service and reducing costs by 90%

The move from a distributed microservices architecture to a monolith application helped achieve higher scale, resilience, and reduce costs.

Marcin Kolny
Mar 22, 2023

At Prime Video, we offer thousands of live streams to our customers. To ensure that customers seamlessly receive content, Prime Video set up a tool to monitor every stream viewed by customers. This tool allows us to automatically identify perceptual quality issues (for example, block corruption or audio/video sync problems) and trigger a process to fix them.

Our Video Quality Analysis (VQA) team at Prime Video already owned a tool for audio/video quality inspection, but we never intended nor designed it to run at high scale (our target was to monitor thousands of concurrent streams and grow that number over time). While onboarding more streams to the service, we noticed that running the infrastructure at a high scale was very expensive. We also noticed scaling bottlenecks that prevented us from monitoring thousands of streams. So, we took a step back and revisited the architecture of the existing service, focusing on the cost and scaling bottlenecks.

Most popular

"We're just beginning to build the future of live sports streaming"
Feb 07, 2023

Prime Video announces Amazon Research Awards recipients for fall 2022
Apr 17, 2023

Empathetic by design: How Amélie Werner prioritizes her team to drive innovation for customers
Apr 05, 2023

Kurz zu mir

Alexander Lehmann

- seit 2022 Softwareentwickler bei OSP
- mit 12 Jahren das erste Mal programmiert
- Architekt aus Leidenschaft
  - Cloud / Serverless
  - DDD
- Turniertänzer
- Segelflieger
- Skipper

OSP
Otto Group Solution Provider

Kontext

Business Kontext

Technischer Kontext

Anwendungs-fall

Kontext

Business Kontext

Technischer Kontext

Anwendungs-fall

Beispiel

Bausteine

Patterns

Lessons Learned

Kontext

Business Kontext

Technischer Kontext

Anwendungs-fall

Beispiel

Bausteine

Patterns

Lessons Learned

Fazit

Hype oder Mehrwert?

Kontext

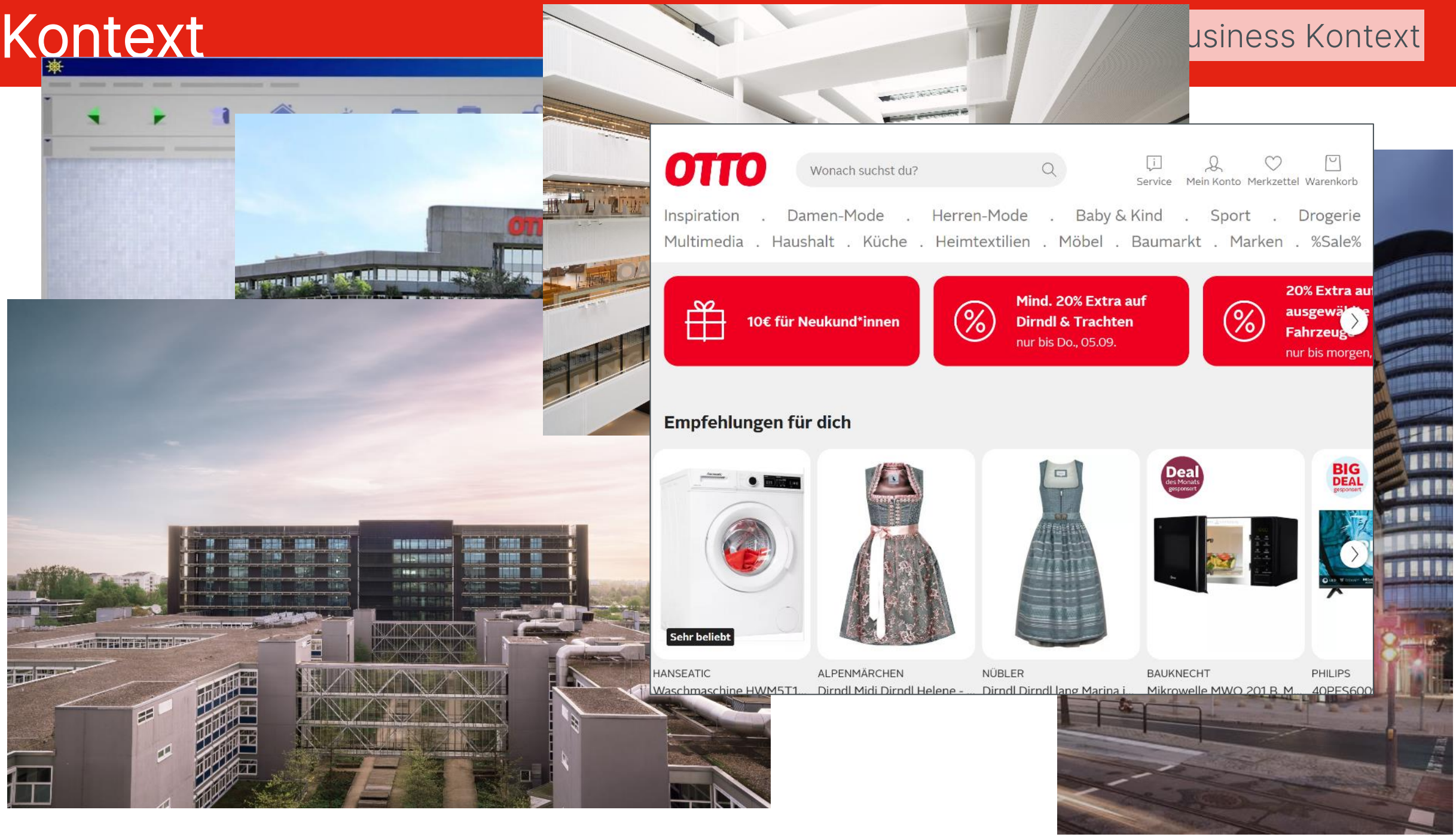
Business Kontext

Technischer Kontext

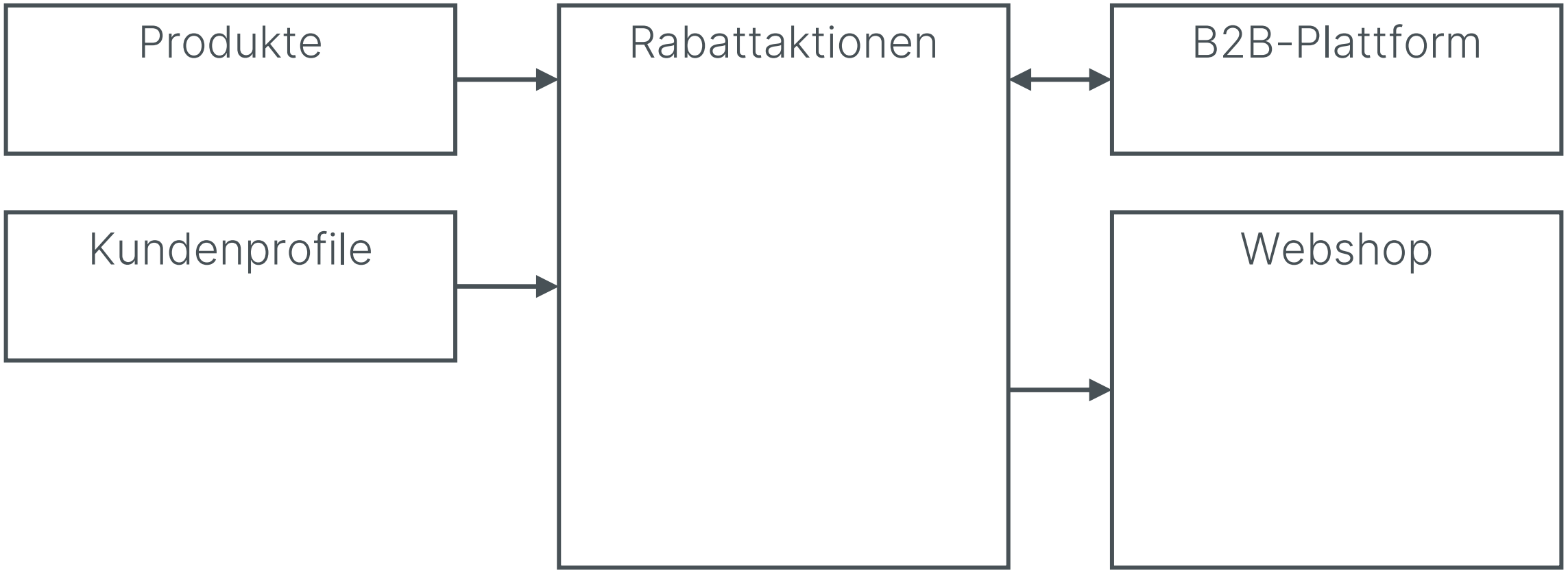
Anwendungs-fall

Kontext

usiness Kontext



Kontext


Produkte
Kundenprofile
Rabattaktionen
B2B-Plattform
Webshop

Kontext

Business Kontext

Technischer Kontext

Anwendungs-fall

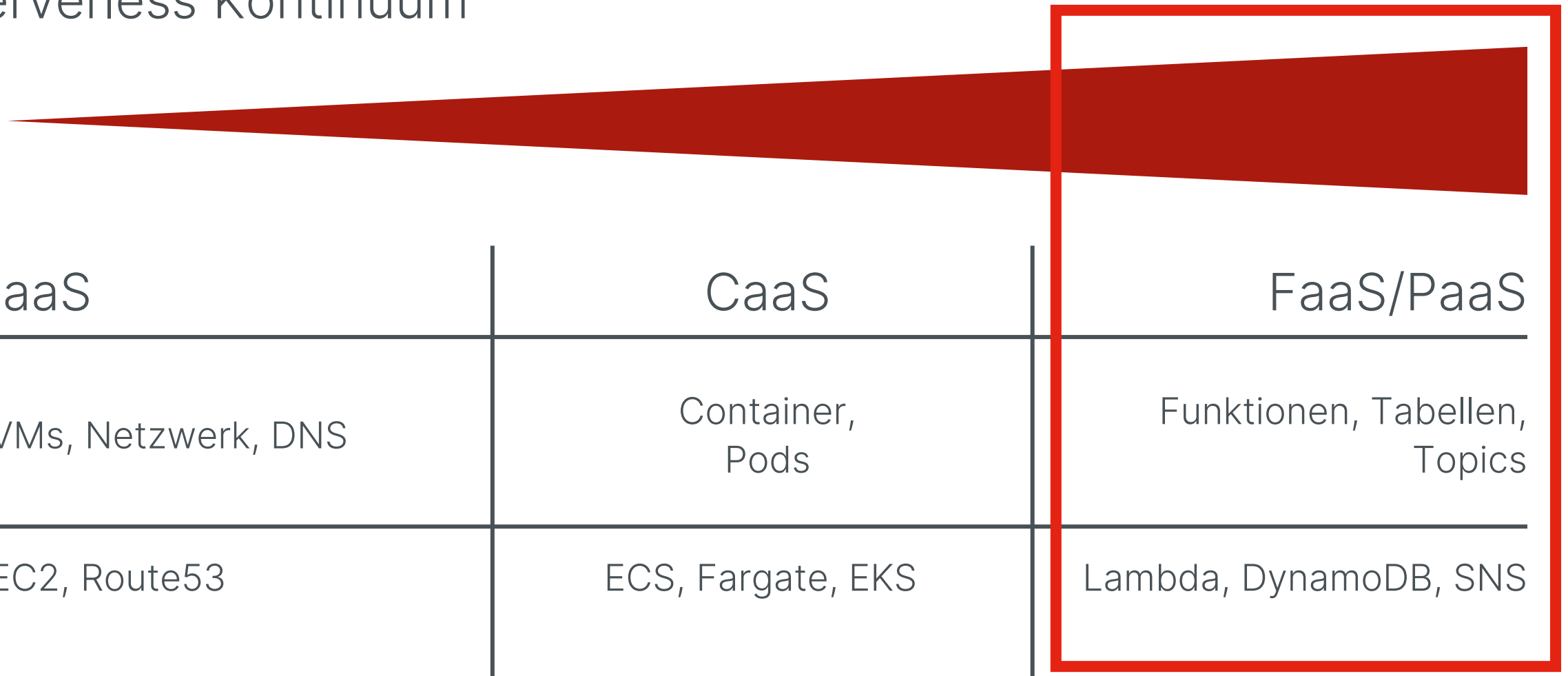
Kontext

Schlagworte

- Cloud Provider: AWS
- Code & Pipelines: Github
- Trunk Based Development
- Continuous Deployment
- Serverless

Kontext

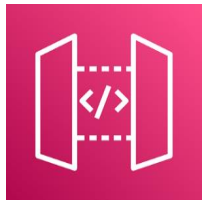
Serverless Kontinuum



IaaS	CaaS	FaaS/PaaS
VMs, Netzwerk, DNS	Container, Pods	Funktionen, Tabellen, Topics
EC2, Route53	ECS, Fargate, EKS	Lambda, DynamoDB, SNS

Kontext

Tech Stack (Backend)



API Gateway

Lambda

Python

Typescript

DynamoDB

OpenSearch

SNS

SQS

Kafka

OSP
Otto Group Solution Provider

Kontext

Business Kontext

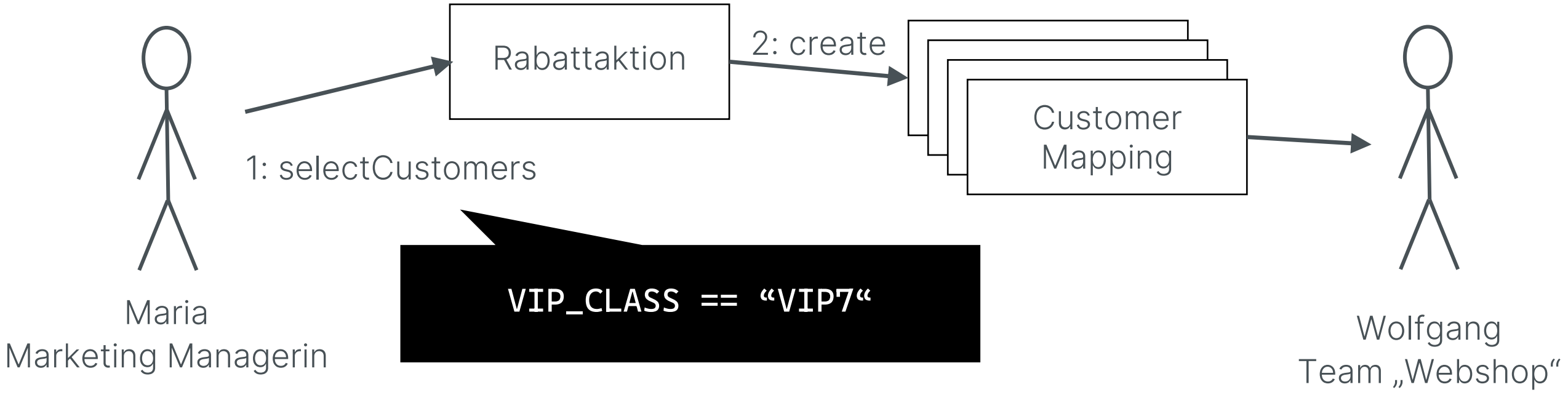
Technischer Kontext

Anwendungs-fall

Kontext

Anwendungs-fall

Ein Marketing Manager wählt Kunden für eine Rabattaktion



Kontext

Business Kontext

Technischer Kontext

Anwendungs-fall

Kontext

Business Kontext

Technischer Kontext

Anwendungs-fall

Beispiel

Bausteine

Patterns

Lessons Learned

Fazit

Hype oder Mehrwert?

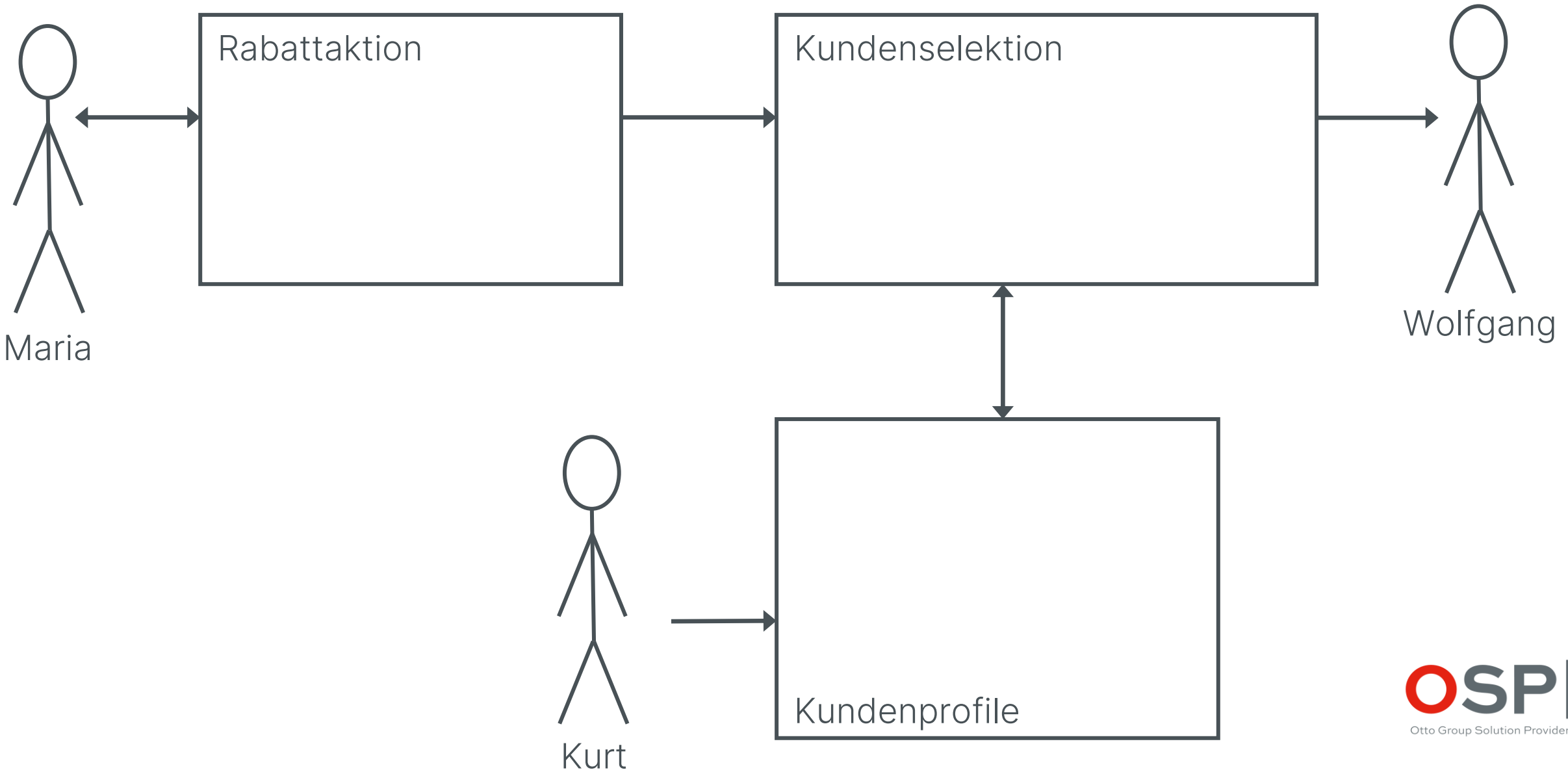
Beispiel

Bausteine

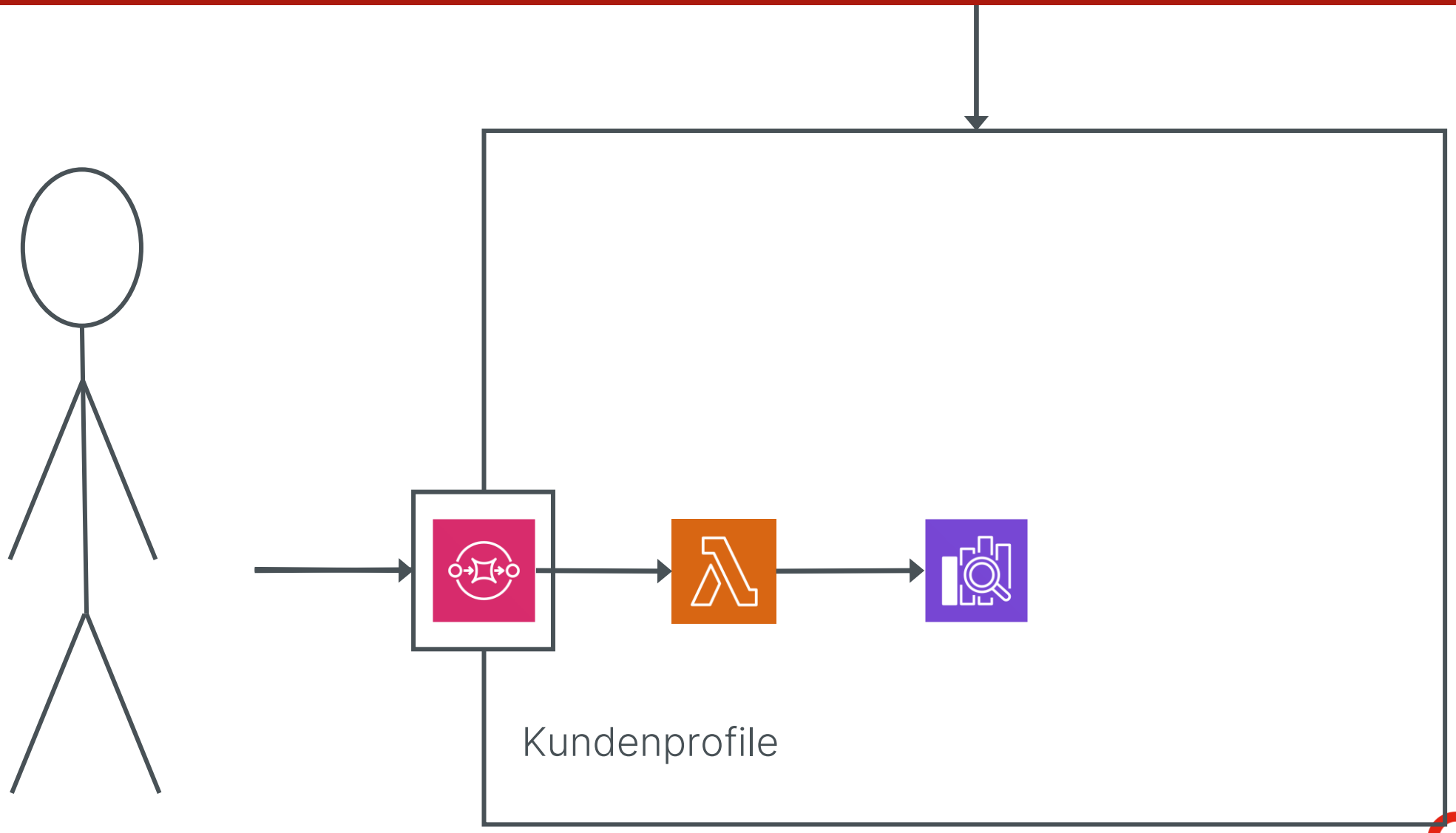
Patterns

Lessons Learned

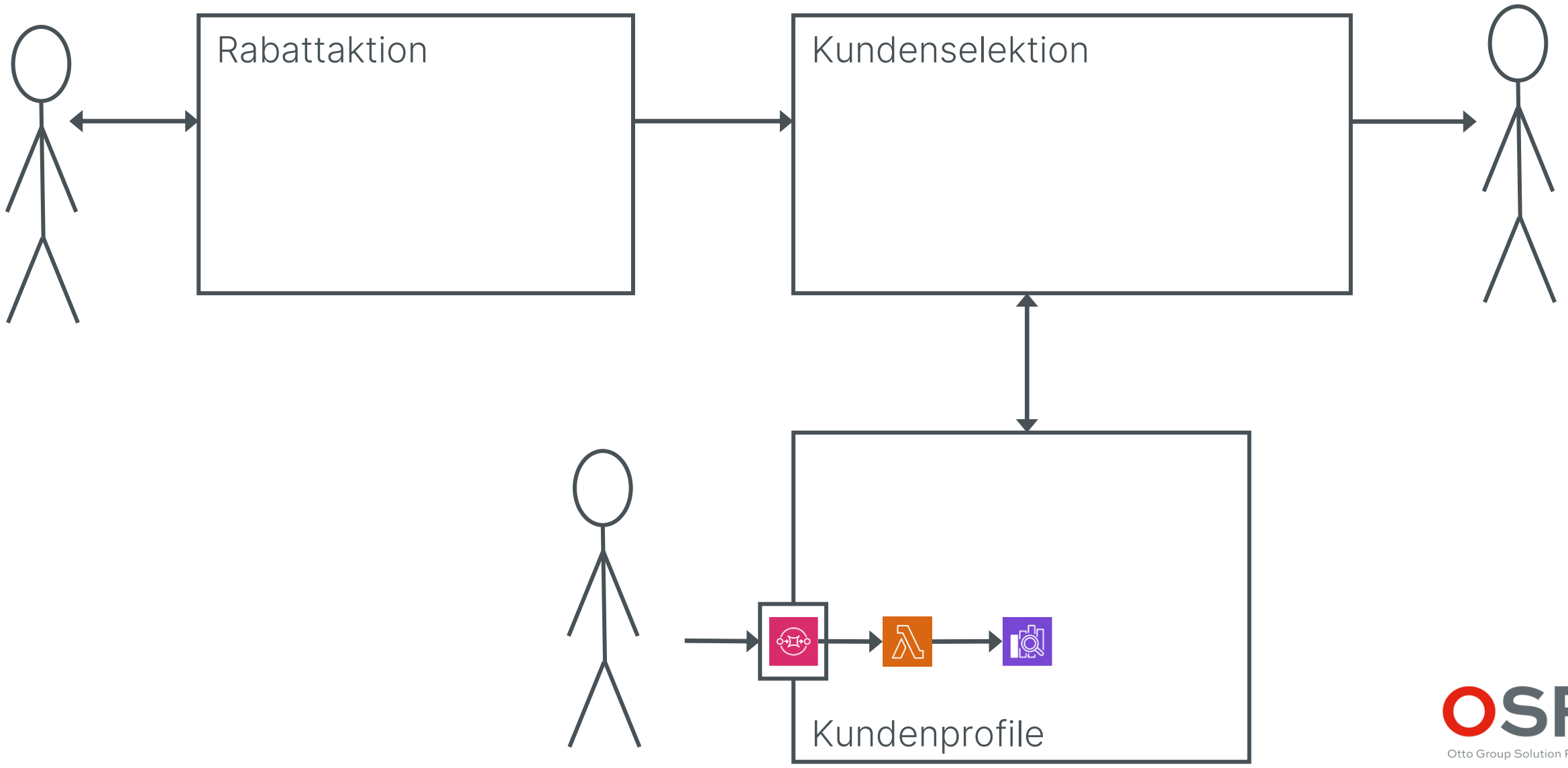
Beispiel


Rabattaktion
Kundenselektion
Maria
Wolfgang
Kurt
Kundenprofile

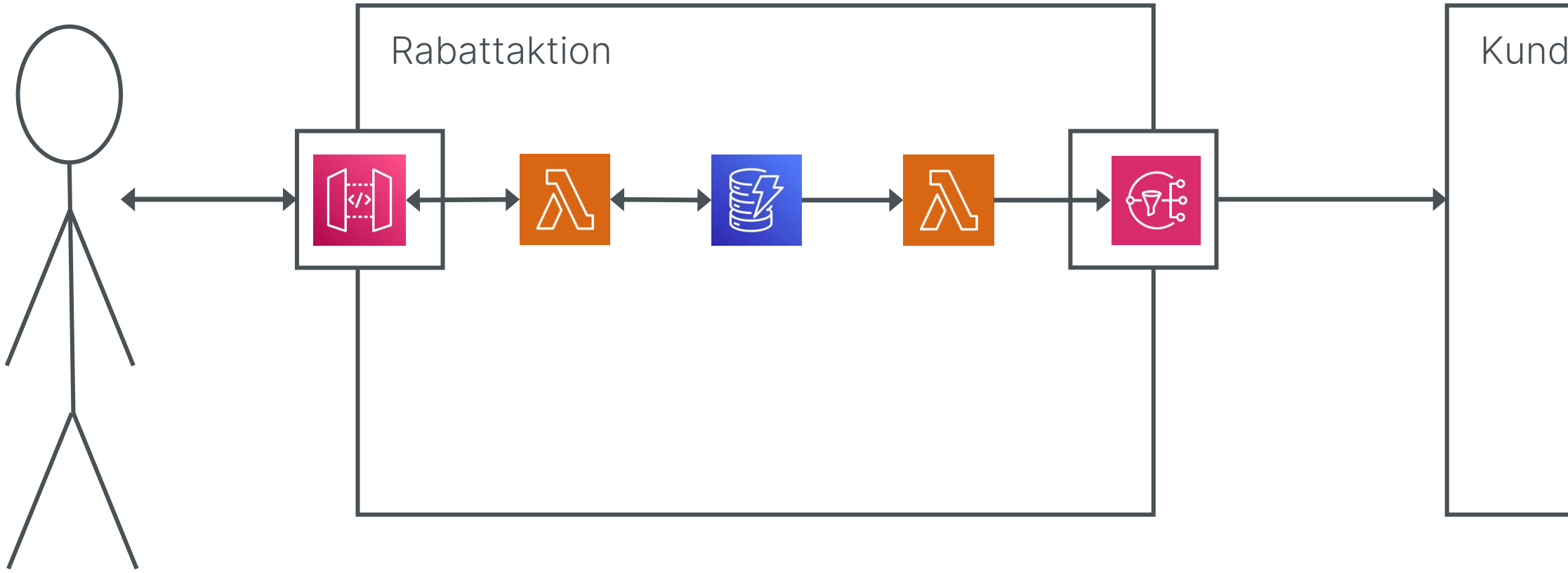
Beispiel



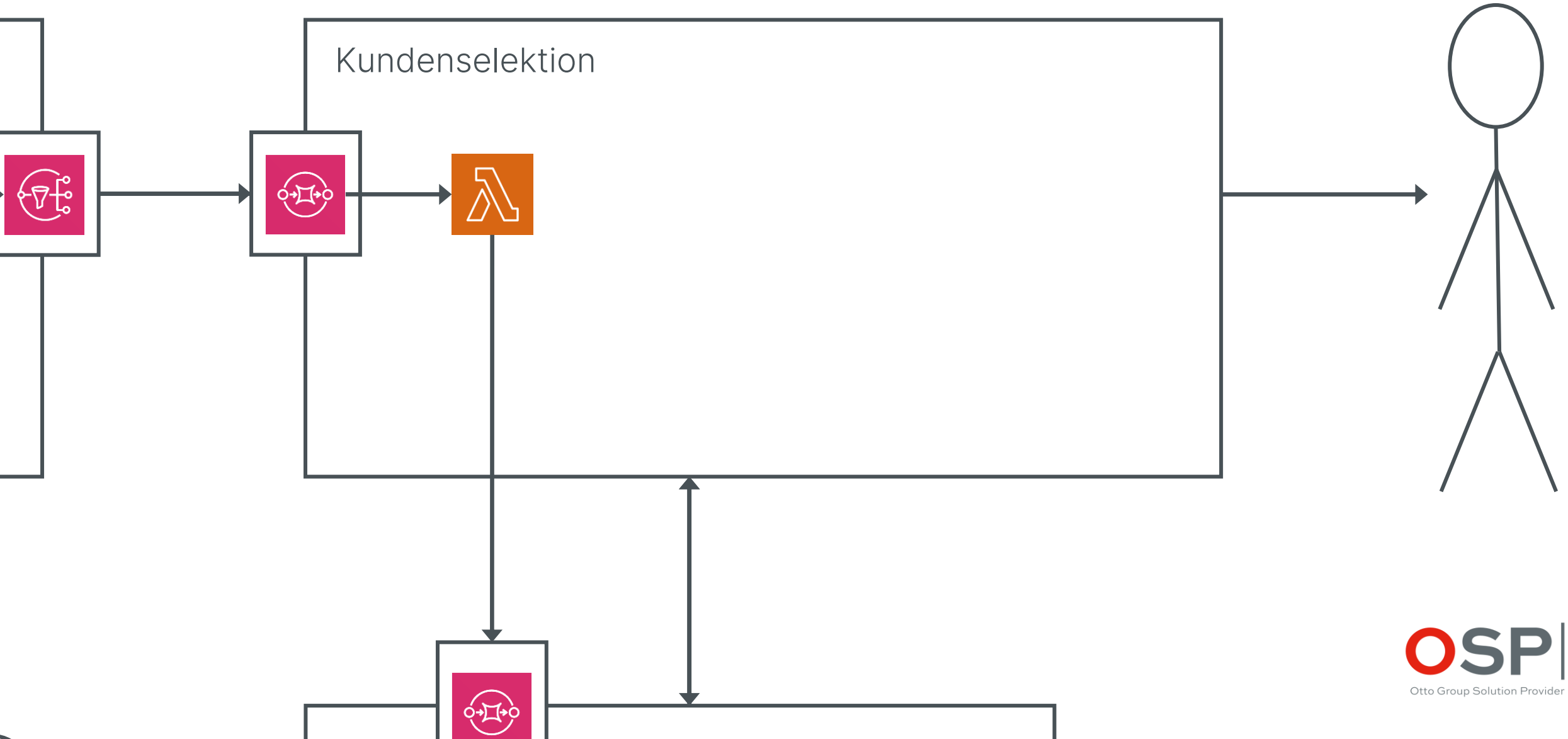
Beispiel



Beispiel

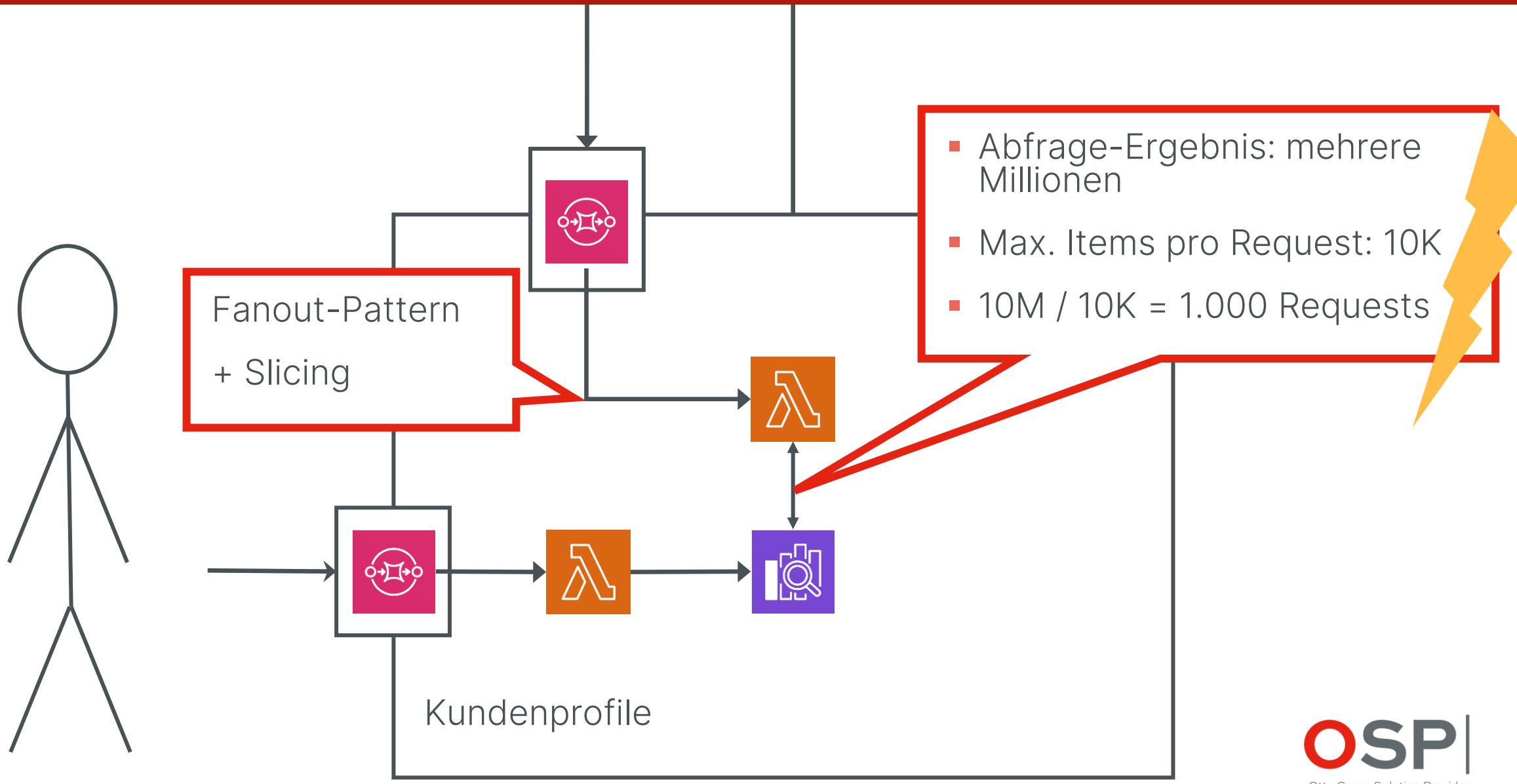


Beispiel

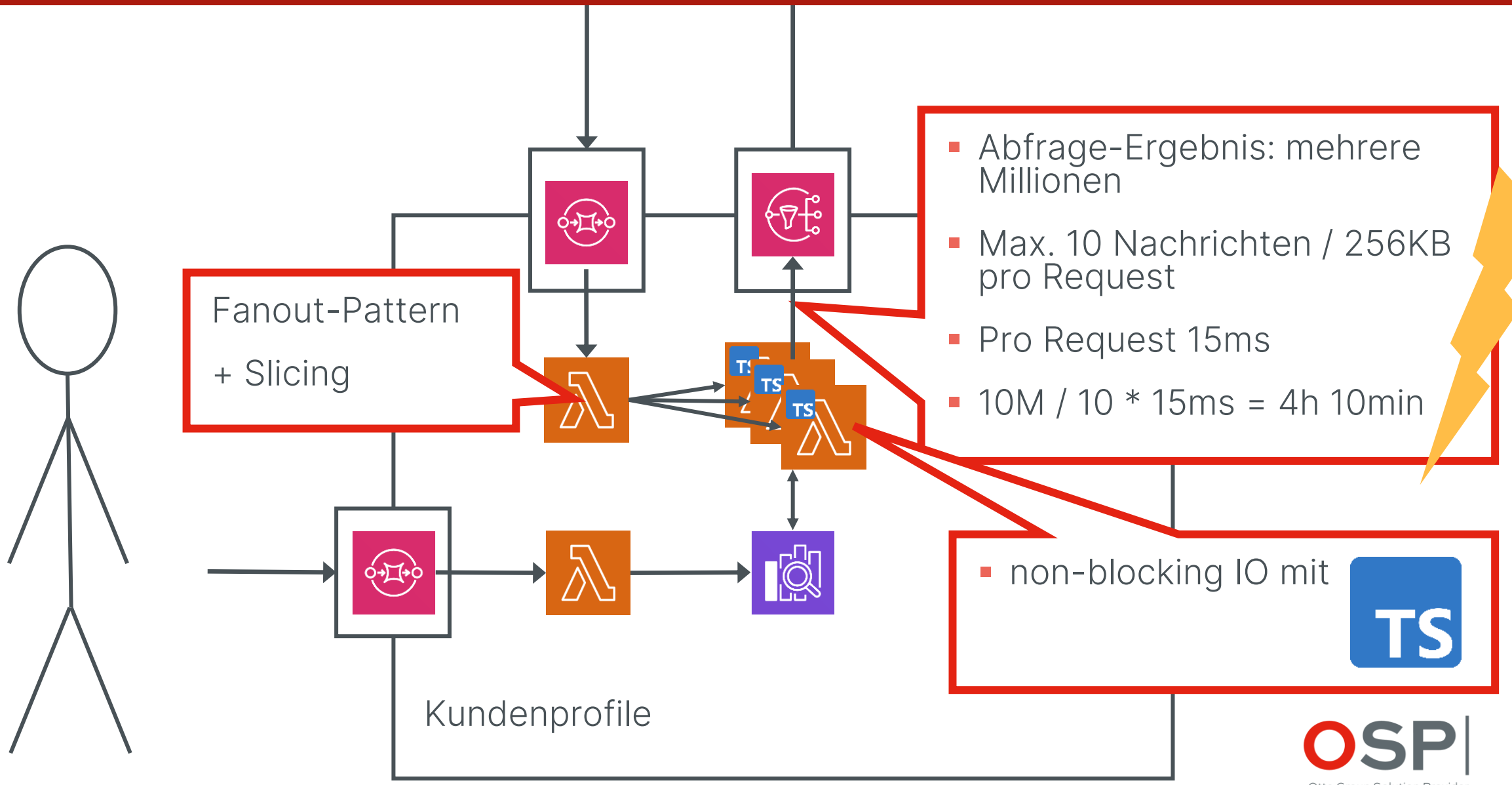


OSP
Otto Group Solution Provider

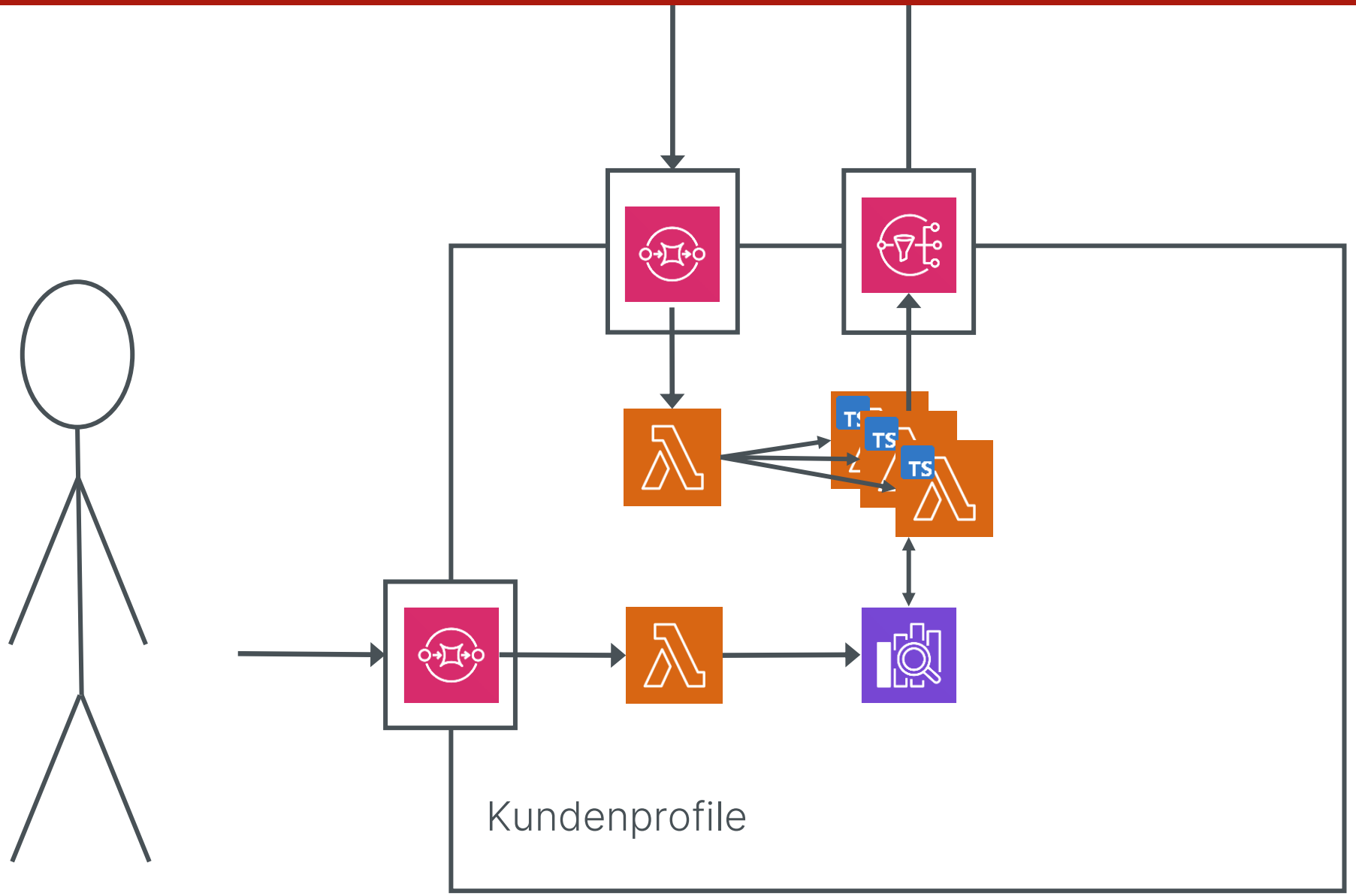
Beispiel


Fanout-Pattern
+ Slicing
Abfrage-Ergebnis: mehrere Millionen
Max. Items pro Request: 10K
10M / 10K = 1.000 Requests
Kundenprofile

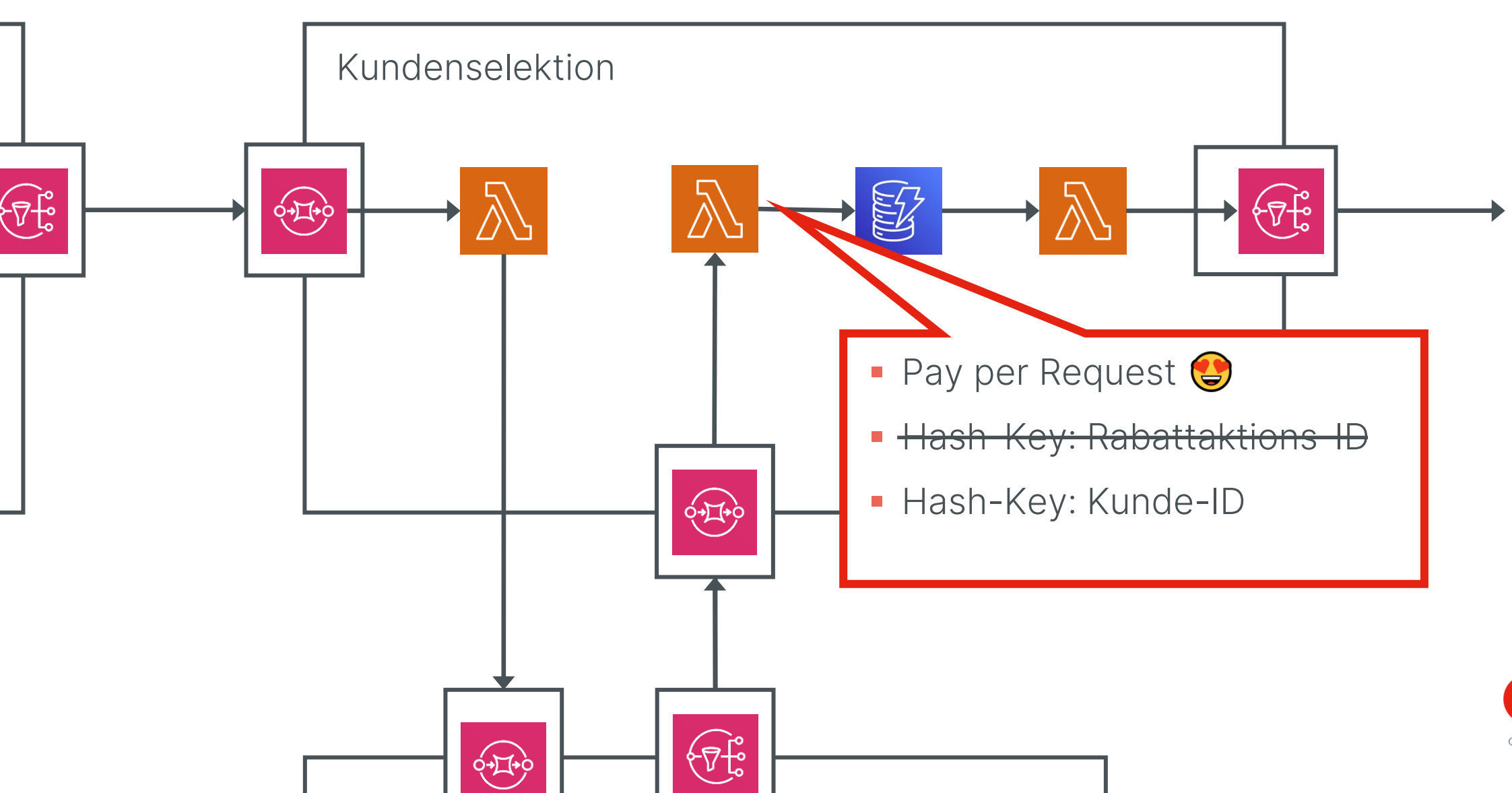
Beispiel


Fanout-Pattern
+ Slicing
Kundenprofile
TS
TS
TS
Abfrage-Ergebnis: mehrere Millionen
Max. 10 Nachrichten / 256KB pro Request
Pro Request 15ms
10M / 10 * 15ms = 4h 10min
non-blocking IO mit
TS

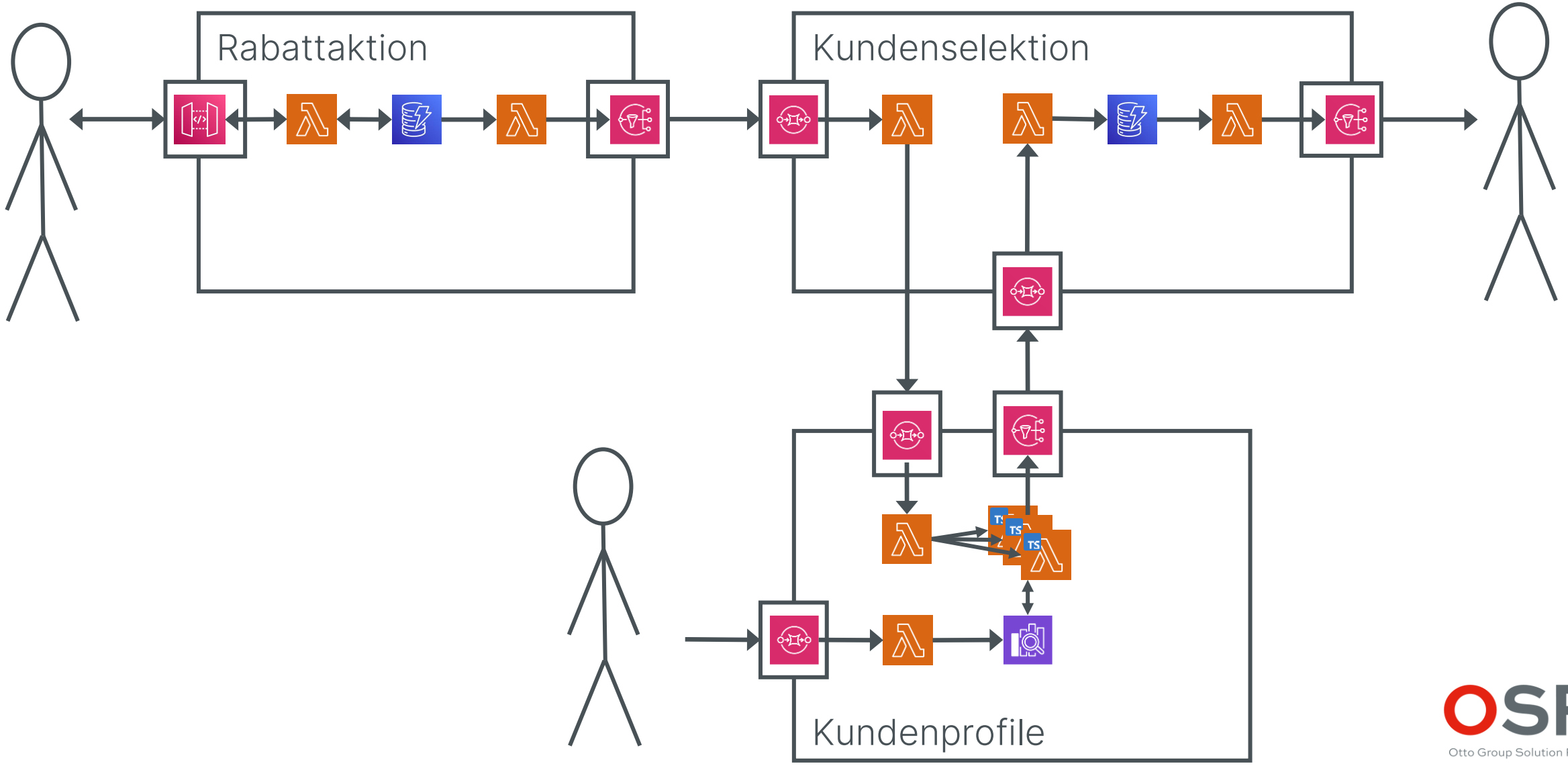
Beispiel



Beispiel


Kundenselektion
Pay per Request 😍
~~Hash-Key: Rabattaktions-ID~~
Hash-Key: Kunde-ID

Beispiel



Beispiel

Bausteine

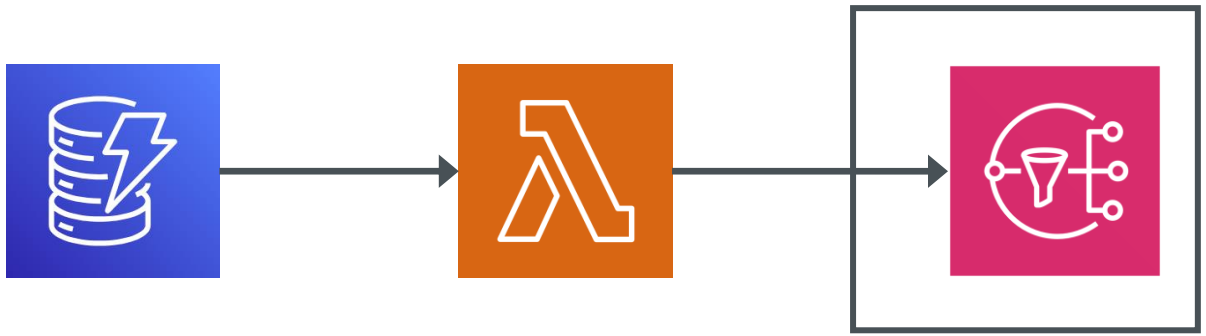
Patterns

Lessons Learned

Beispiel

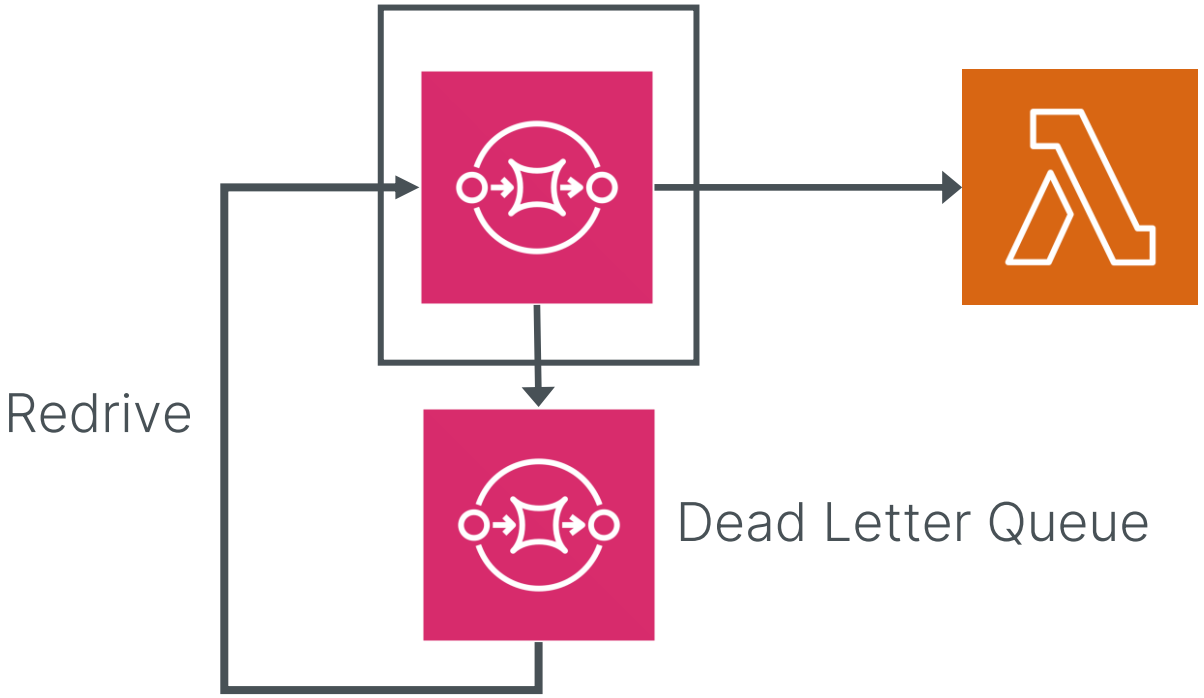
API Pattern

Beispiel



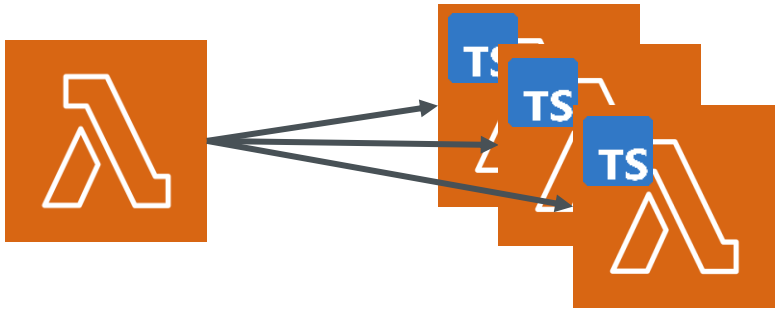
Outbox Pattern

Beispiel



Input Queue Pattern

Beispiel



Fanout Pattern

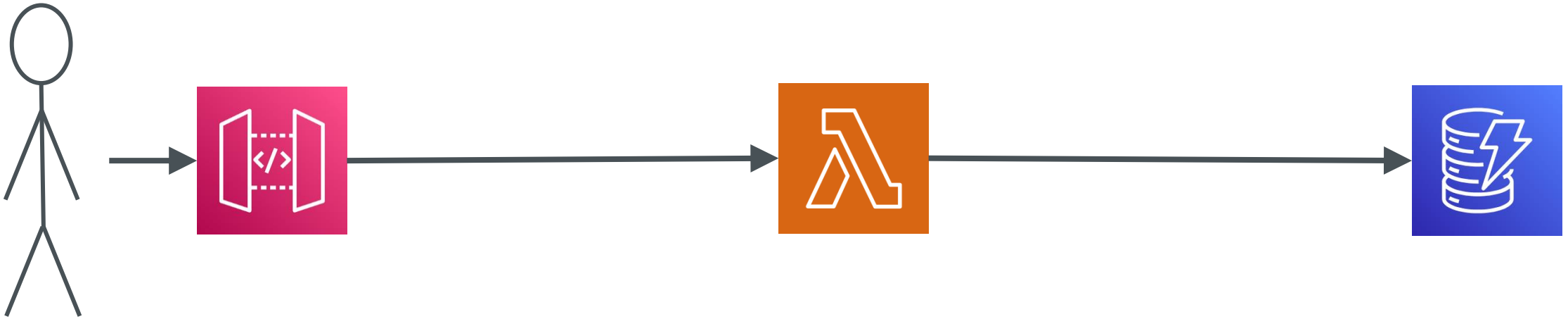
Beispiel

Bausteine

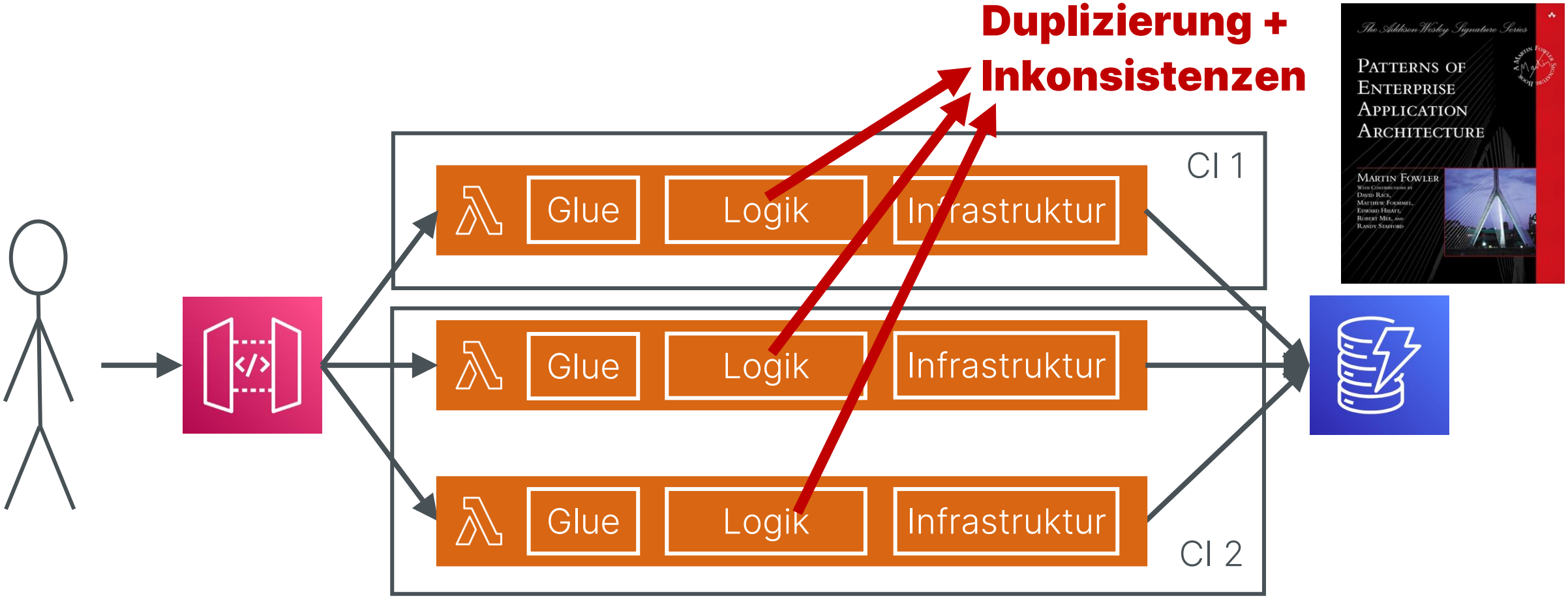
Patterns

Lessons Learned

Beispiel

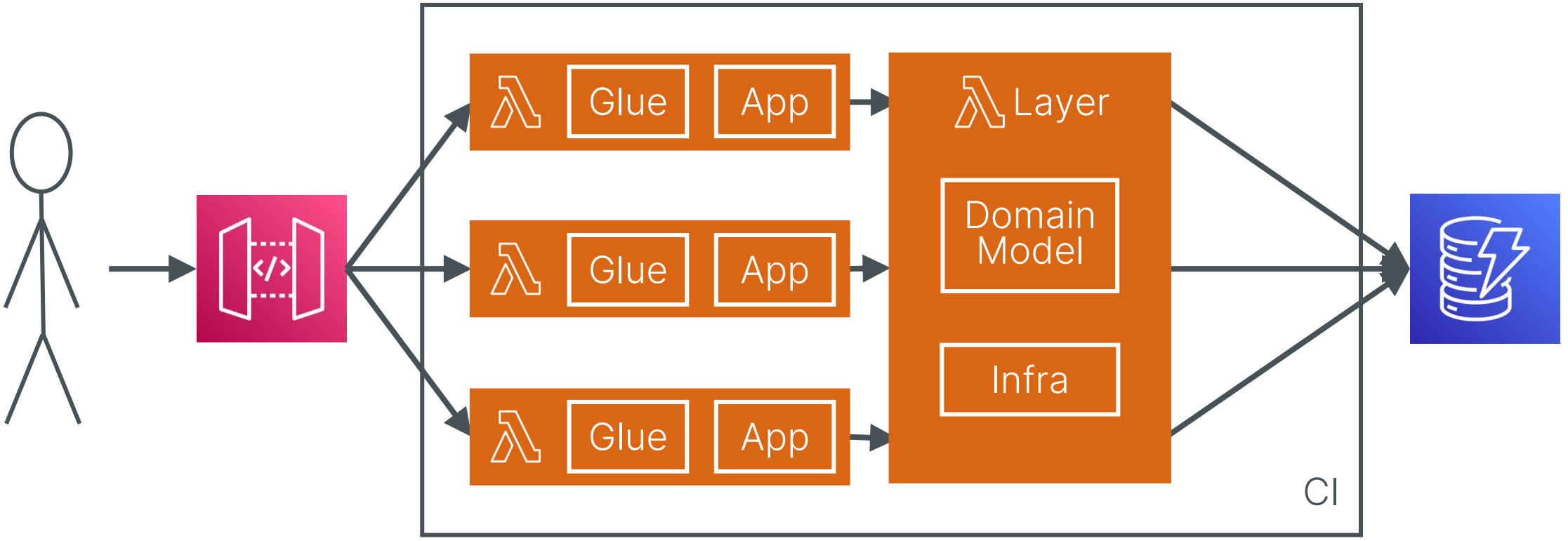


Beispiel

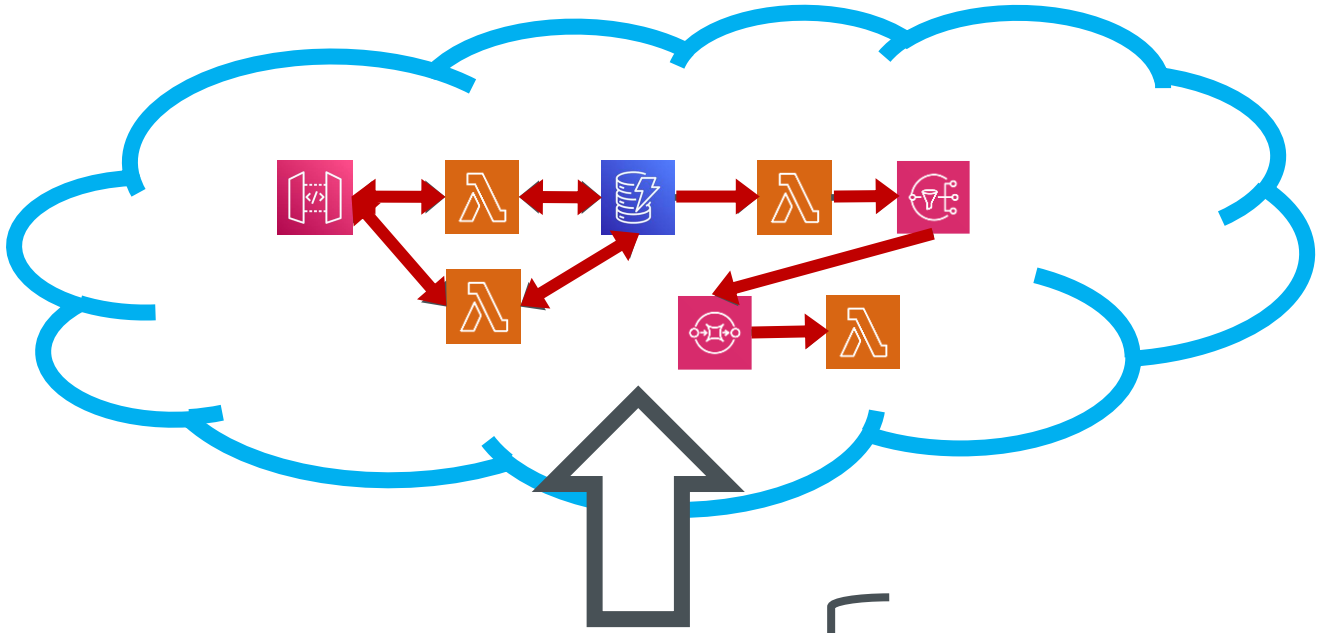

Duplizierung +
Inkonsistenzen
CI 1
Glue
Logik
Infrastruktur
Glue
Logik
Infrastruktur
Glue
Logik
Infrastruktur
CI 2

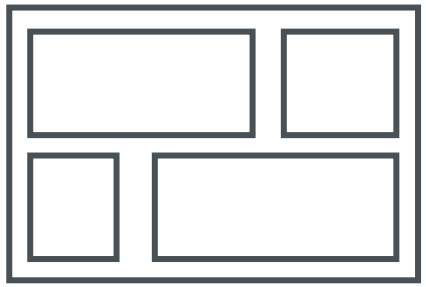
The Addison Wesley Signature Series
Patterns of Enterprise Application Architecture
Martin Fowler

Beispiel



Beispiel





IaC Framework

Glue Code (IaC)

- Permissions
- Backups
- Monitoring
- Resilienz

Beispiel

Bausteine

Patterns

Lessons Learned

Kontext

Business Kontext

Technischer Kontext

Anwendungs-fälle

Beispiel

Bausteine

Patterns

Lessons Learned

Fazit

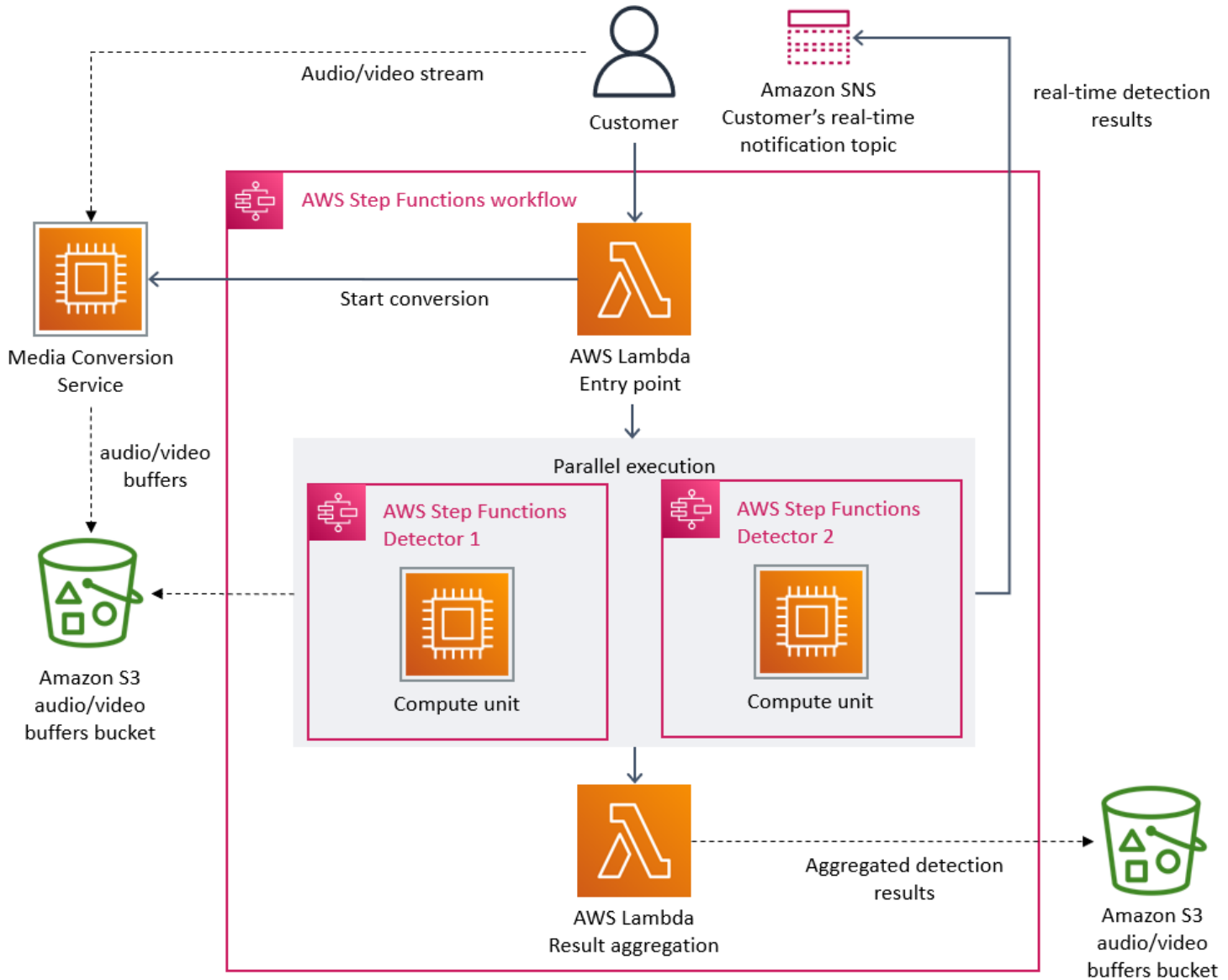
Hype oder Mehrwert?

Fazit

Hype oder Mehrwert?

Fazit

Hype oder Mehrwert?



Fazit

❌ Serverless schlecht geeignet ❌

- Hohe Last auf CPU, GPU oder RAM (bspw. Video-Verarbeitung)
- Sehr gut vorhersehbare/kontinuierliche Last
- Sehr latenzkritische Applikationen (p=0,99)

Fazit

✓ Serverless gut geeignet ✓

- Business-Applikationen (ein Mensch klickt auf Buttons) bzw. eventgetriebene Applikationen
- Unvorhersehbare Peak-Loads
- Hohe Anforderungen an Verfügbarkeit, Resilienz und Skalierbarkeit
- Start-Ups: Pay per Request → keine Fixkosten

Fazit

❓ Konsequenzen von Serverless ❓

- Man nutzt ein battle-tested Application Framework
- HTTP-Requests everywhere
- Skalierbar by-default
- Verschiedene Technologien leicht kombinierbar
- Super geringer Betriebsaufwand
- Betriebskosten, Performance und Umweltfreundlichkeit gehen Hand-in-Hand

OSP
Otto Group Solution Provider

Vielen Dank für Eure Aufmerksamkeit!