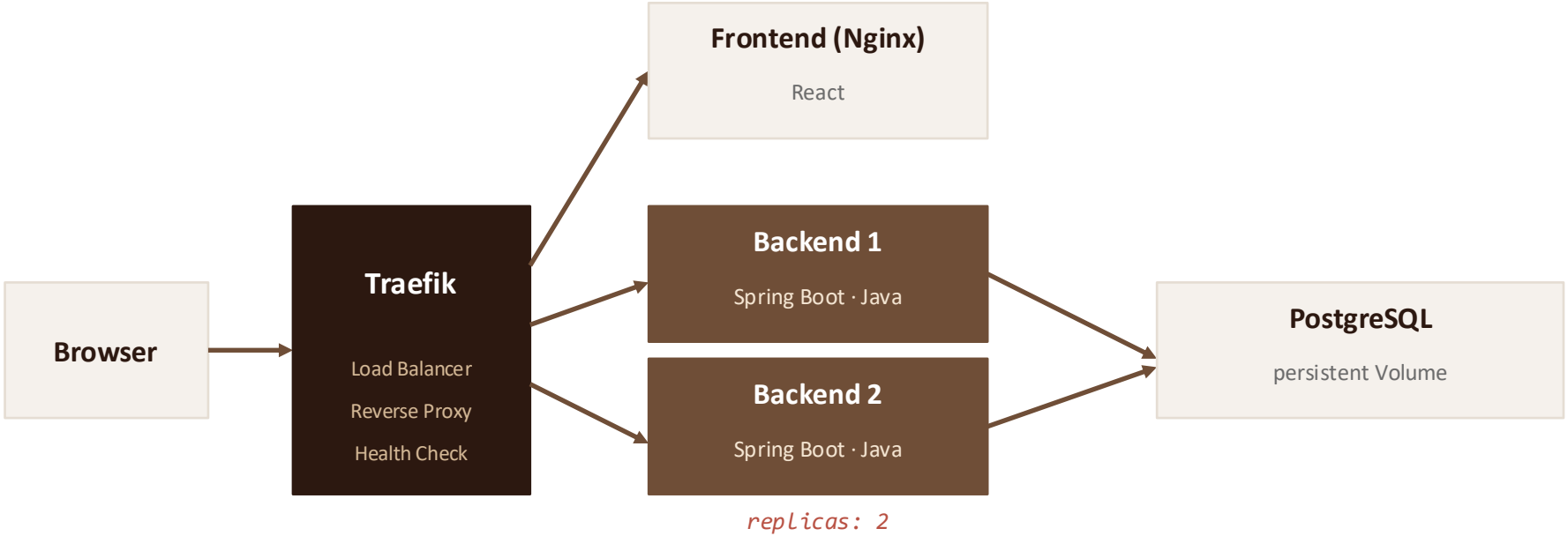
CoffeeCluster

Distributed Systems Challenge Task

Marc Fisch · OST Rapperswil · DSy FS 2026

Architektur



Tech-Stack

FRONTEND

React 19

Vite 7 · TypeScript 5 · Node 24

LOAD BALANCER

Traefik v3.7.1

L7 · Docker-native · Active HC

BACKEND

Spring Boot 4.0.3

Java 25 · JPA · Actuator

AUTH

JJWT 0.13

HS256 · BCrypt · Stateless

DATABASE

PostgreSQL 18

Persistent Volume

CONTAINER

Docker Compose

Multi-Stage · Alpine

Demo

01

docker compose up

Ein Befehl, der das ganze Cluster startet.

02

Login + JWT

Register, Token im Network-Tab beobachten.

03

Round-Robin sichtbar

Drei Coffees → Container-IDs wechseln im Badge.

04

Failover live

docker stop backend-1 → App läuft weiter.

05

Load-Test mit hey

2000 Requests, 100 parallel, gegen /api/logs.

Load Test

```
hey -n 2000 -c 100 -H "Authorization: Bearer …" http://localhost/api/logs
```

2000	100	100%	897	293
Requests	Concurrent Workers	Success Rate	req/s · avg	ms · p95 Latency

Interpretation

Keine fehlgeschlagenen Requests. Traefik + zwei stateless Spring-Boot-Replicas verkraften 100 parallele Clients ohne Drops. Der Engpass ist nicht das Backend. Es ist die geteilte **PostgreSQL-Instanz**.

Take-aways

5

Pflicht-Komponenten erfüllt

Frontend · LB · 2×Service · JWT · DB

1

Befehl startet alles

docker compose up -d --build

0

Manuelle Setup-Schritte

Auch DB-Schema kommt via Hibernate

100%

Erfolgsrate im Load-Test

2000 Requests, 100 parallel

Fragen?

Marc Fisch · OST Rapperswil · DSy FS 2026