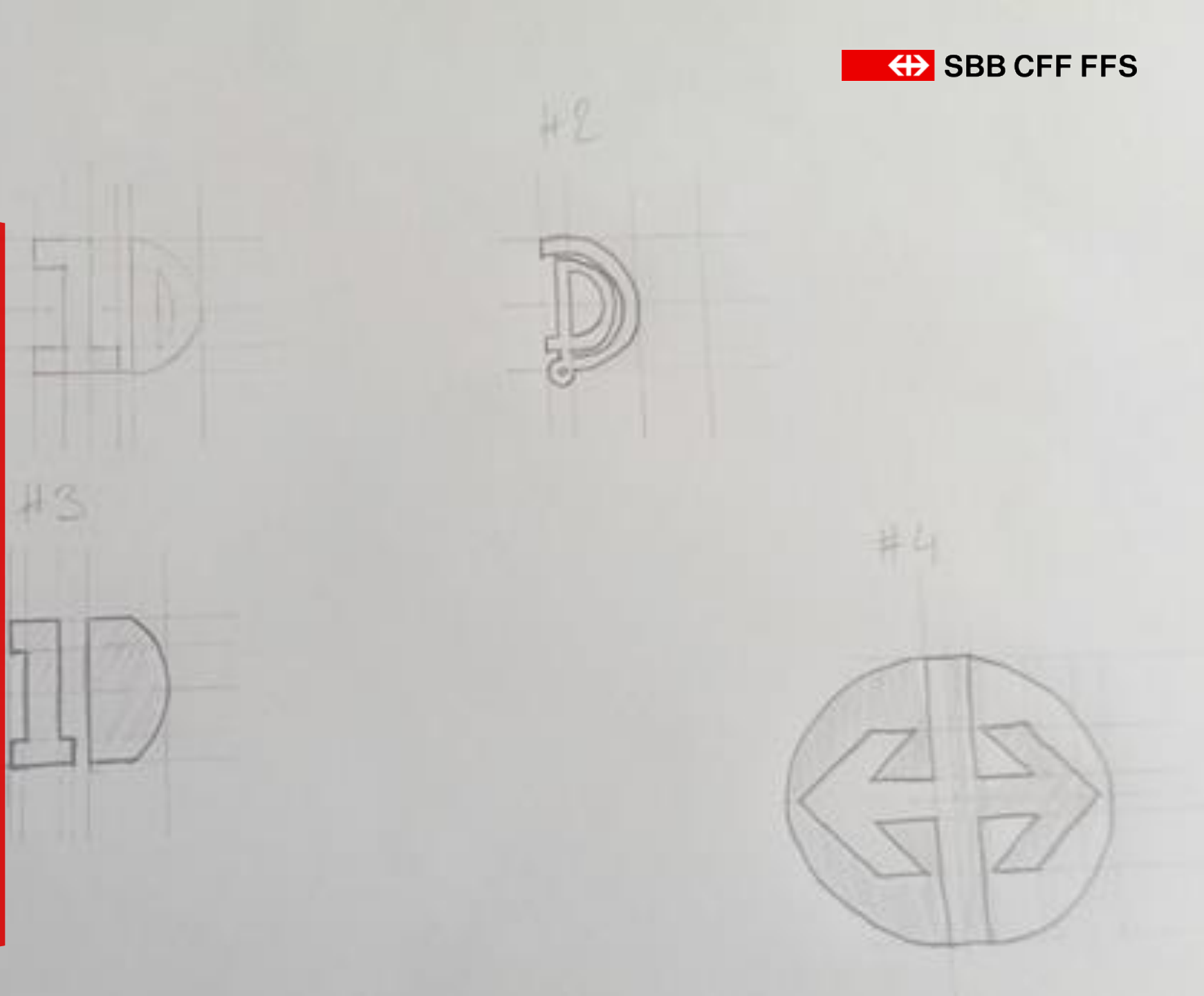
SBB CFF FFS

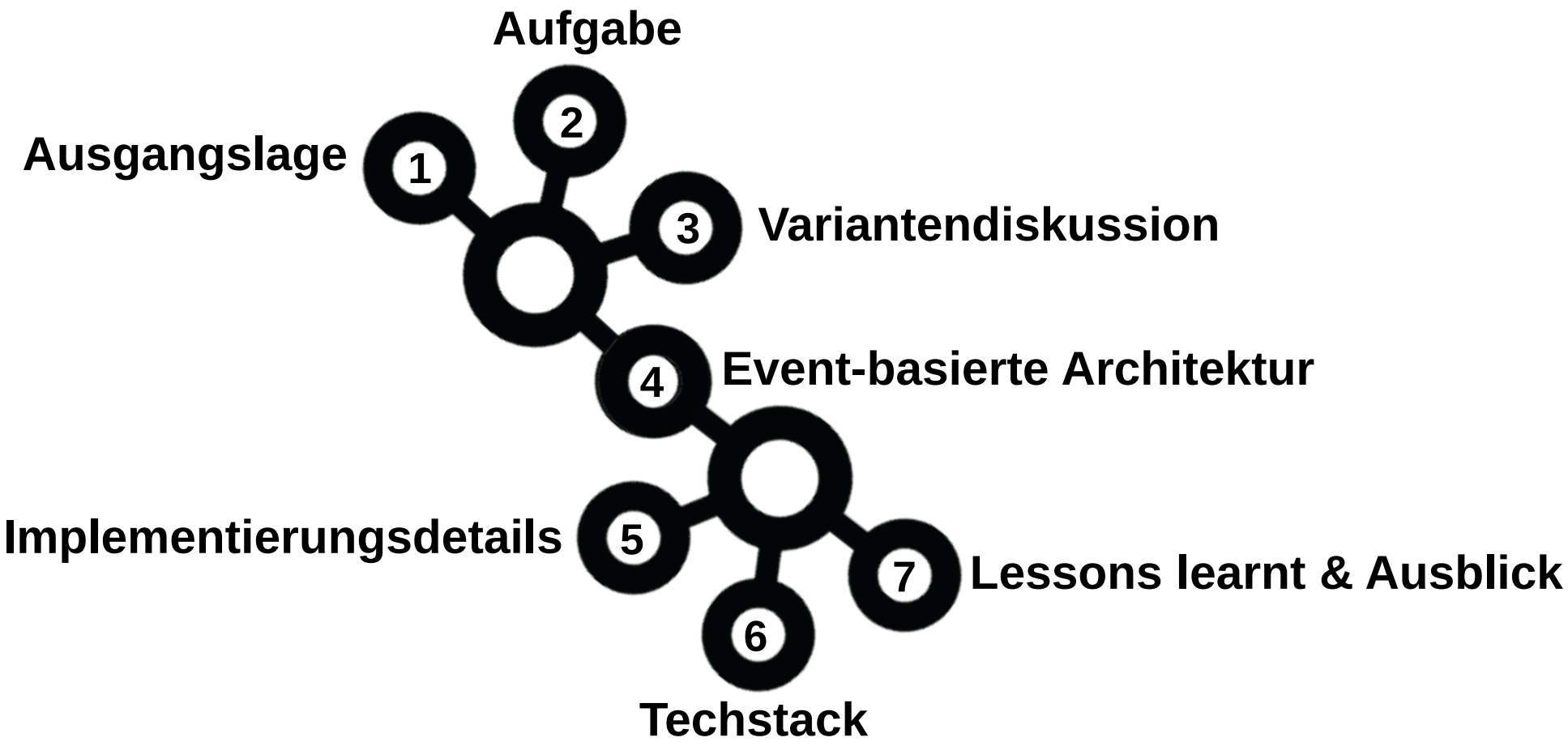
Event-basierte Architektur mit Apache Kafka

12. Dez. 2018 @jugch

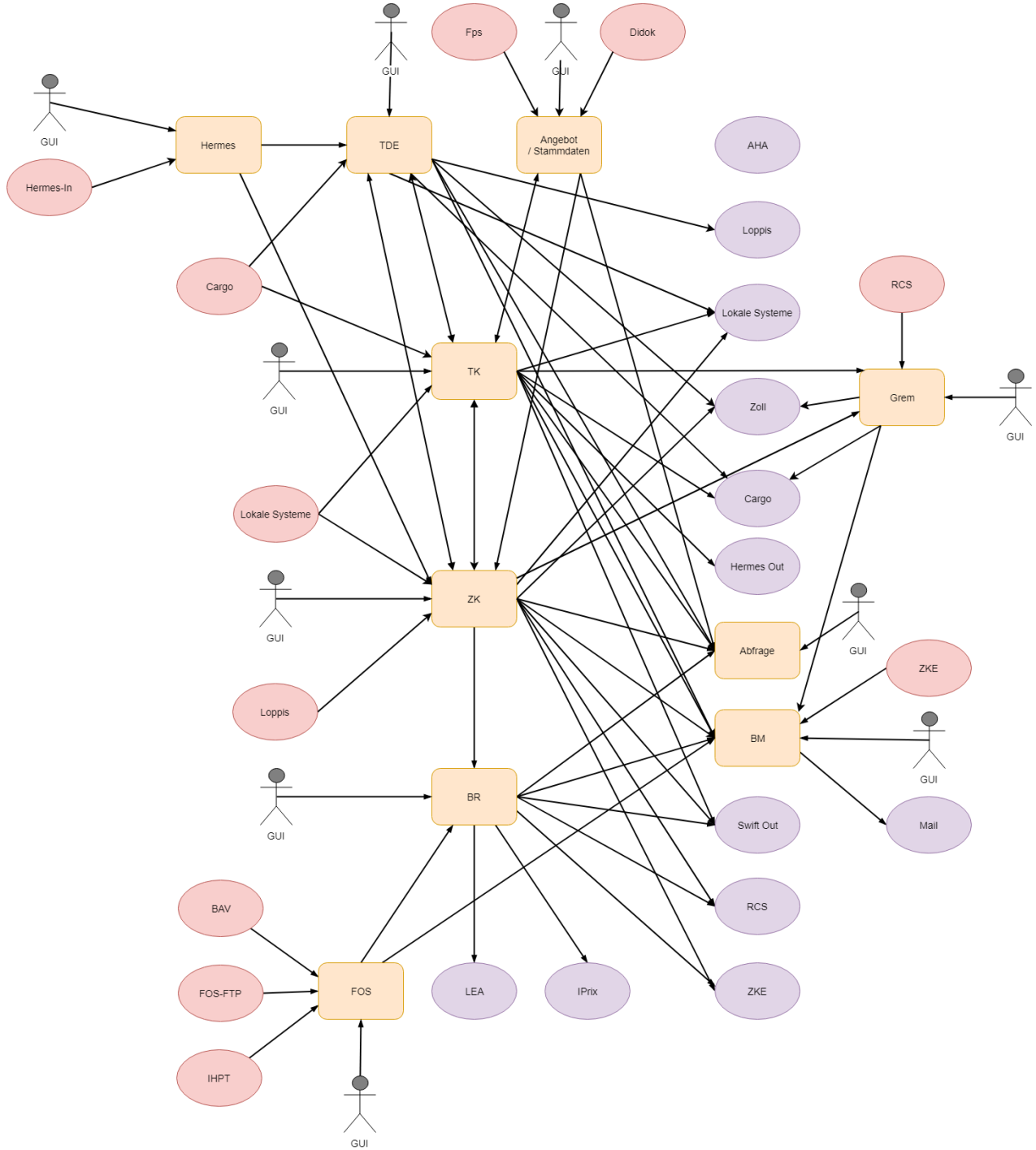
Korhan Gülseven
Renato Löffel



SBB CFF FFS


Aufgabe
Ausgangslage
1
2
3
Variantendiskussion
4
Event-basierte Architektur
Implementierungsdetails
5
6
7
Lessons learnt & Ausblick
Techstack

Ausgangslage

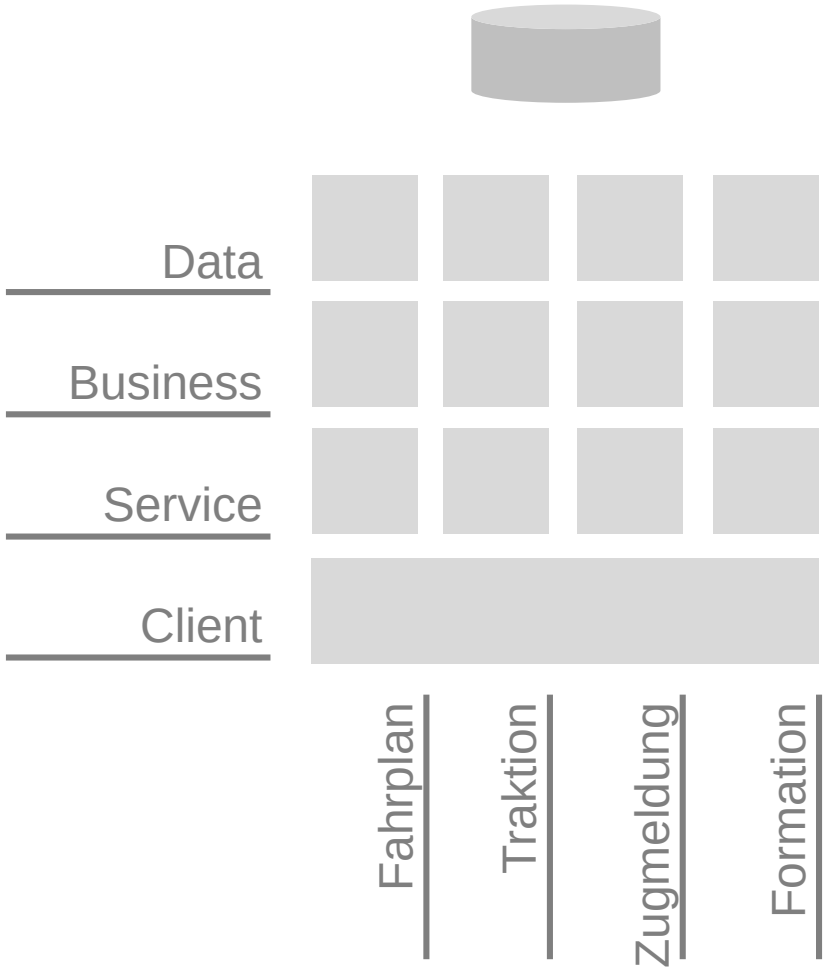

GUI
Fps
GUI
Didok
GUI
Hermes
TDE
Angebot
/ Stammdaten
AHA
Hermes-In
Loppis
Cargo
RCS
Lokale Systeme
TK
GUI
Zoll
Grem
GUI
Cargo
Lokale Systeme
Hermes Out
ZK
GUI
GUI
Abfrage
ZKE
Loppis
BM
GUI
BR
GUI
Swift Out
Mail
BAV
RCS
FOS-FTP
FOS
LEA
IPrix
ZKE
IHPT
GUI

Aufgabenstellung
Ablösung der Produktionsdurchführung CIS Infra

Top Qualitätsziele
1. Änderbarkeit / Wartbarkeit
2. Performance
3. Zuverlässigkeit / Verfügbarkeit

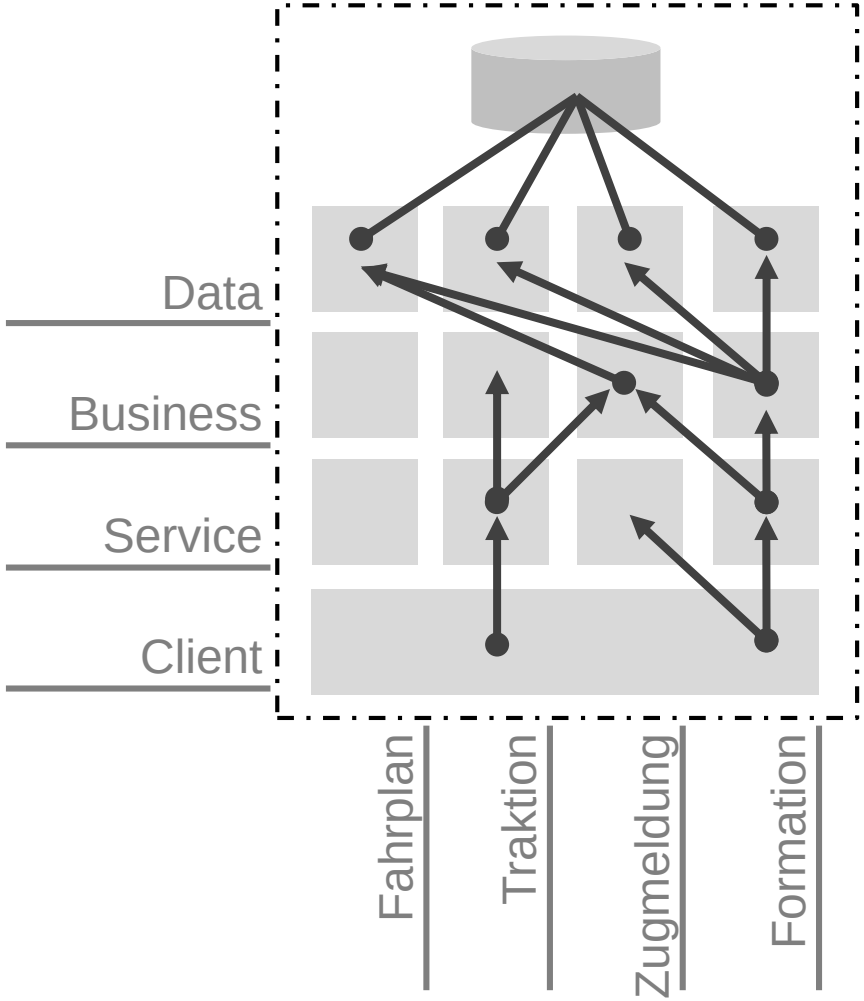
SBB CFF FFS

Monolith

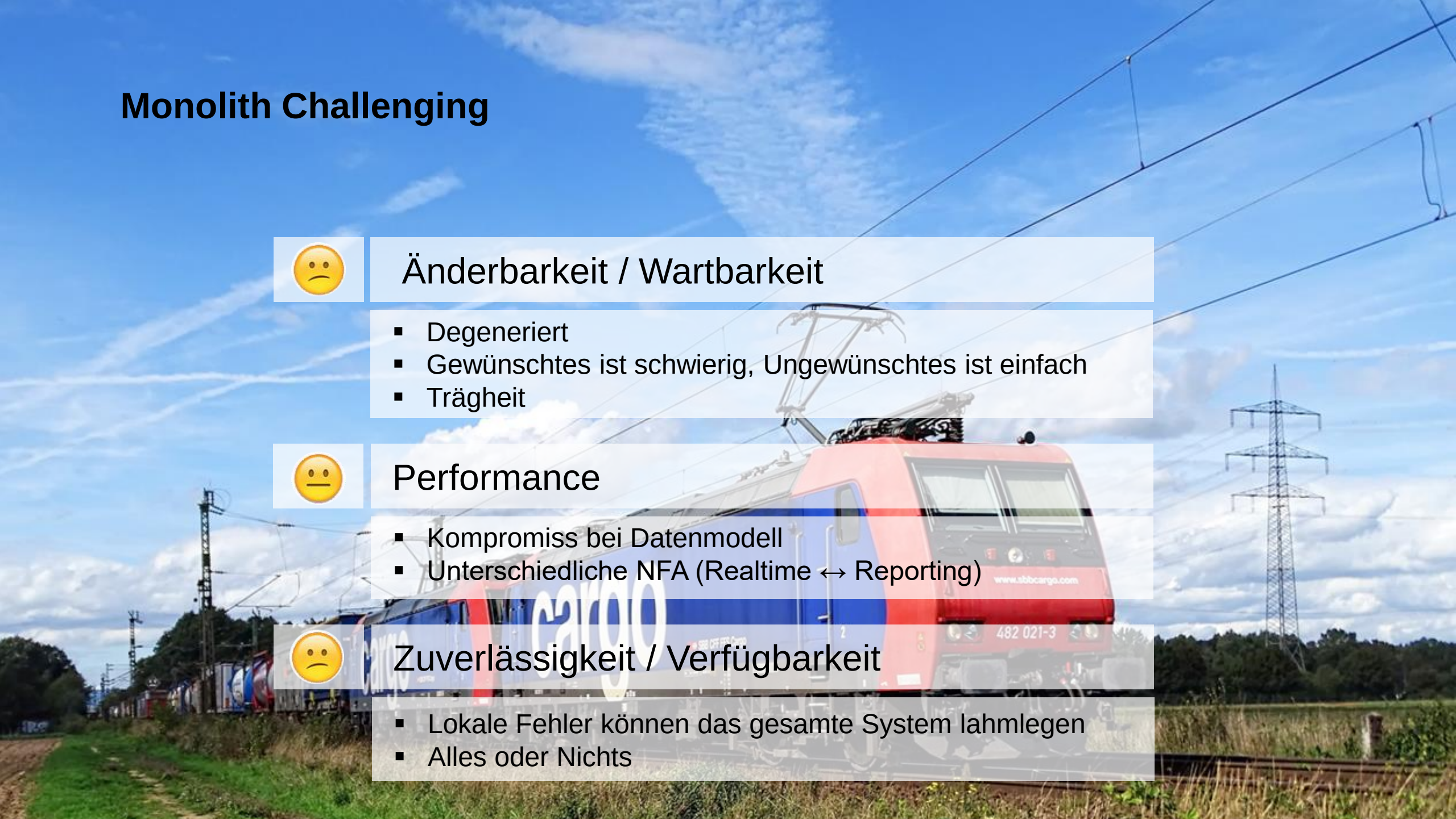


SBB CFF FFS

Monolith



Monolith Challenging



😕 Änderbarkeit / Wartbarkeit

- Degeneriert
- Gewünschtes ist schwierig, Ungewünschtes ist einfach
- Trägheit

😐 Performance

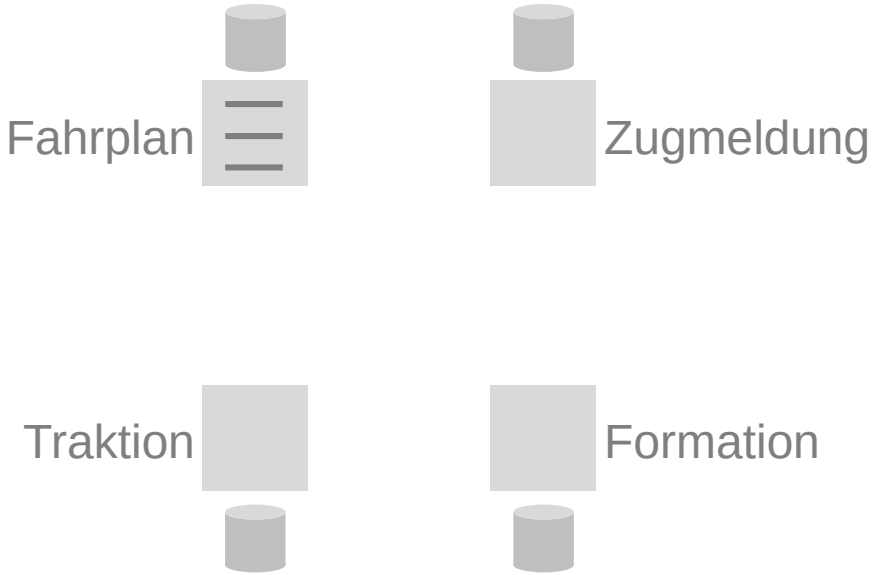
- Kompromiss bei Datenmodell
- Unterschiedliche NFA (Realtime ↔ Reporting)

😕 Zuverlässigkeit / Verfügbarkeit

- Lokale Fehler können das gesamte System lahmlegen
- Alles oder Nichts

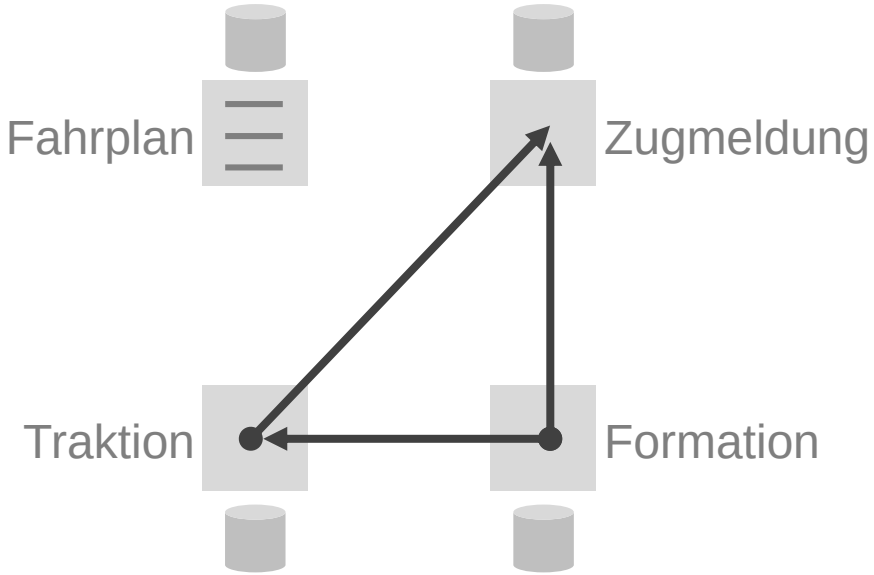
SBB CFF FFS

Microservice Request/Response



SBB CFF FFS

Microservice Request/Response



Microservice Request/Response Challenging

Änderbarkeit / Wartbarkeit

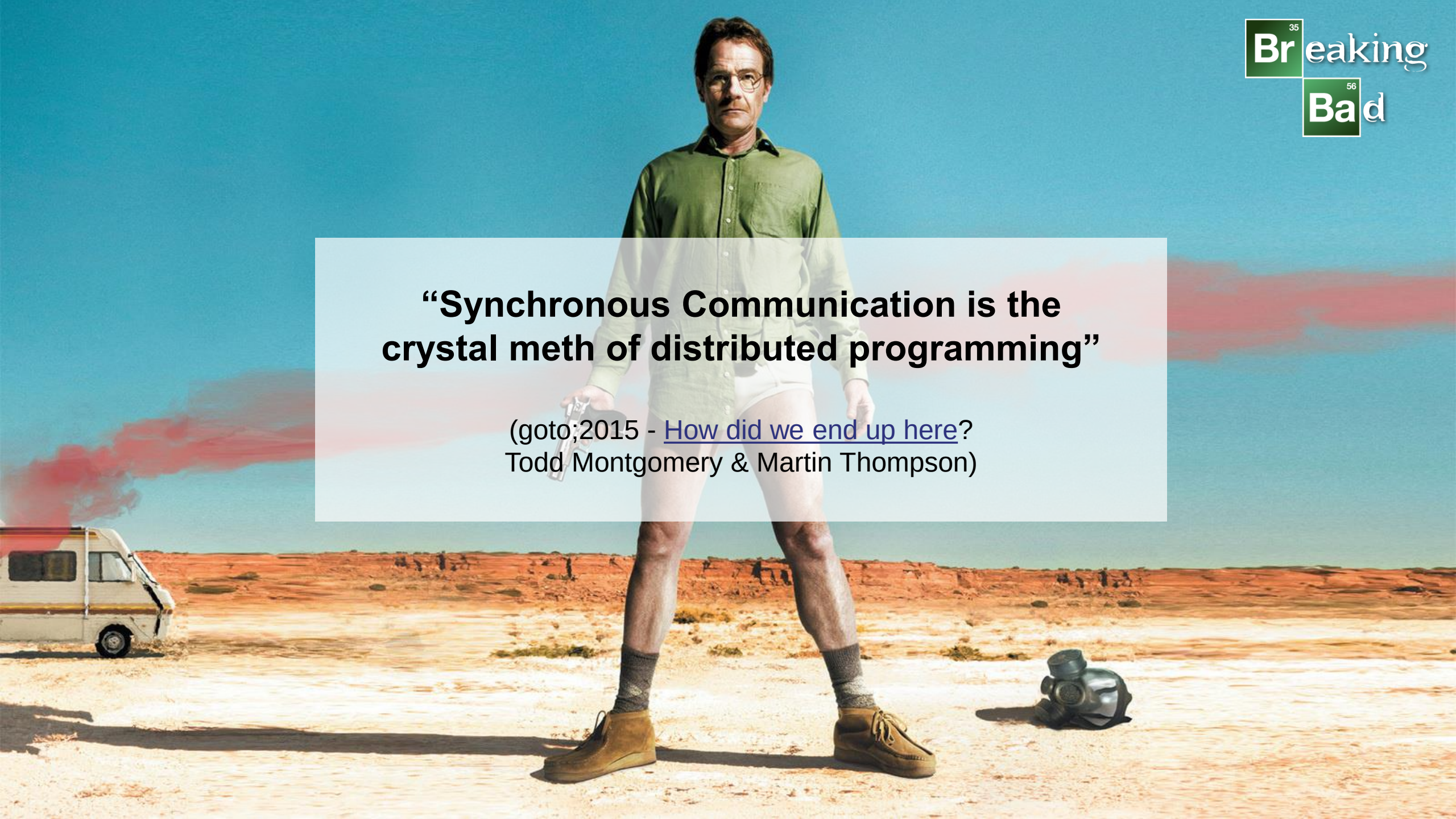
- Starke Isolation
- Kleine Einheiten

Performance

- Innerhalb eines Moduls optimal
- Modulübergreifend schwach, Response-/Latenzzeiten werden kumuliert

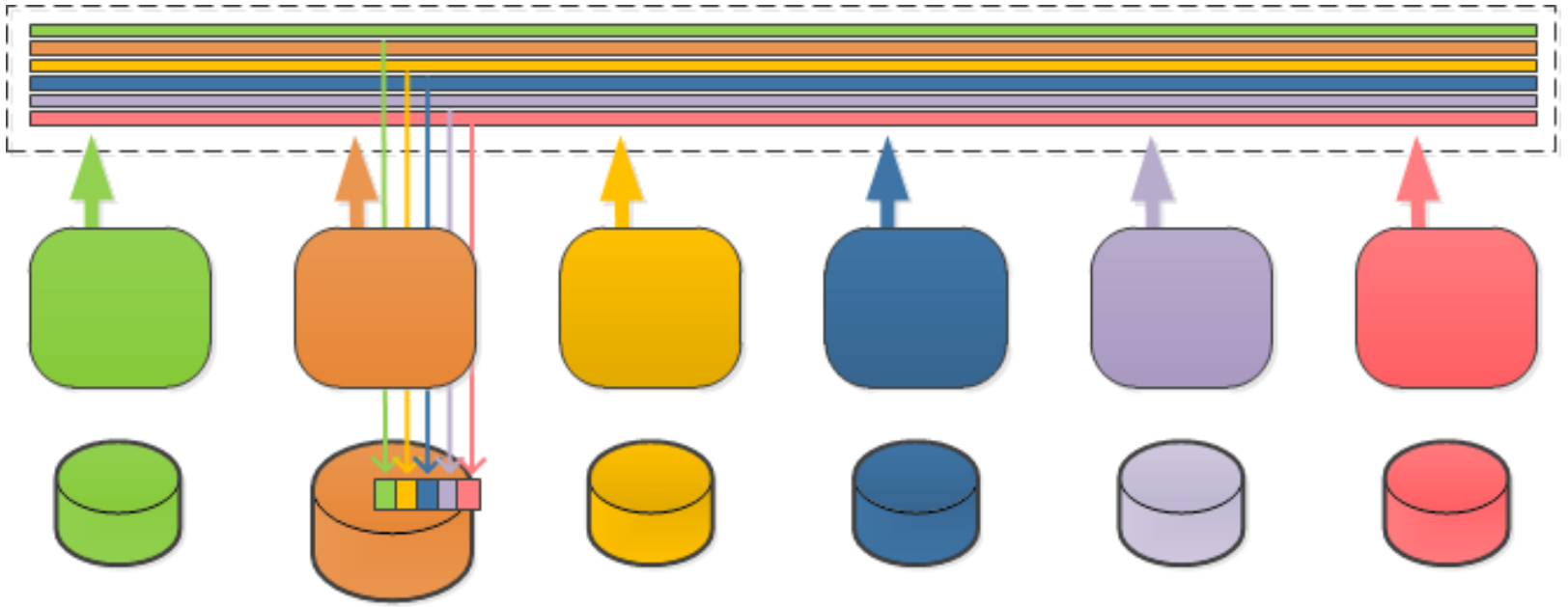
Zuverlässigkeit / Verfügbarkeit

- Verfügbarkeit eines Moduls ist direkt von anderen abhängig
- Daten sind nicht da, wo sie benötigt werden.


Breaking Bad
“Synchronous Communication is the crystal meth of distributed programming”
(goto;2015 - How did we end up here?
Todd Montgomery & Martin Thompson)

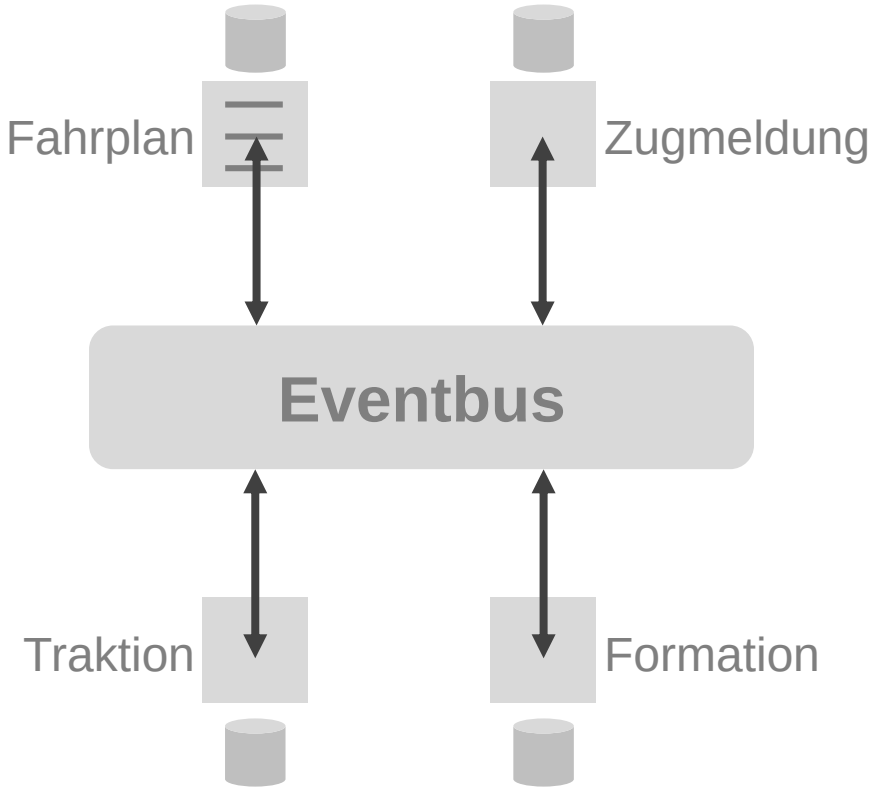
SBB CFF FFS

Exkurs: event-basierte Integration



SBB CFF FFS

Microservice event-basiert



Microservice event-basiert Challenging

Änderbarkeit / Wartbarkeit

- Analog zu Microservice Request/Response

Performance

- Realtime-Event-Processing
- Laufende Datenverteilung

Zuverlässigkeit / Verfügbarkeit

- Maximale Isolierung in der Laufzeit

Event-basierte Architektur

Maximale Entkopplung (Raum und Zeit)

Event-basierte Architektur

Verteiltes System

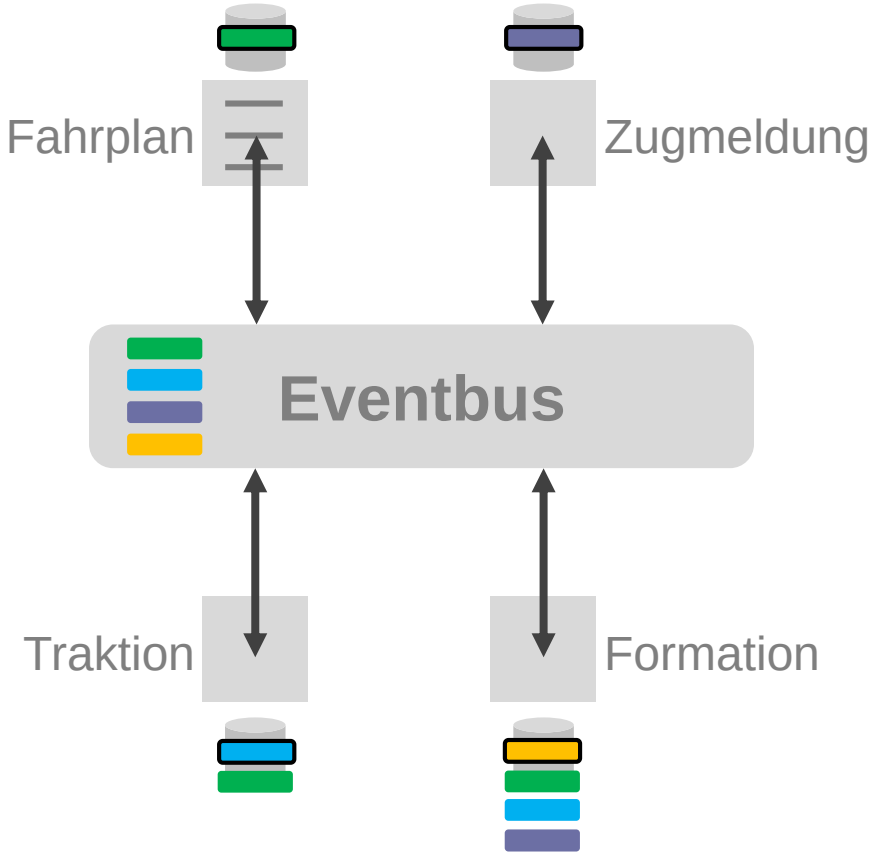
- CAP
- Keine verteilten Transaktionen
- Keine Garantien (Reihenfolge, Duplikat)
- Versionierung

Zustand

- Redundanz

SBB CFF FFS

Verteilter State



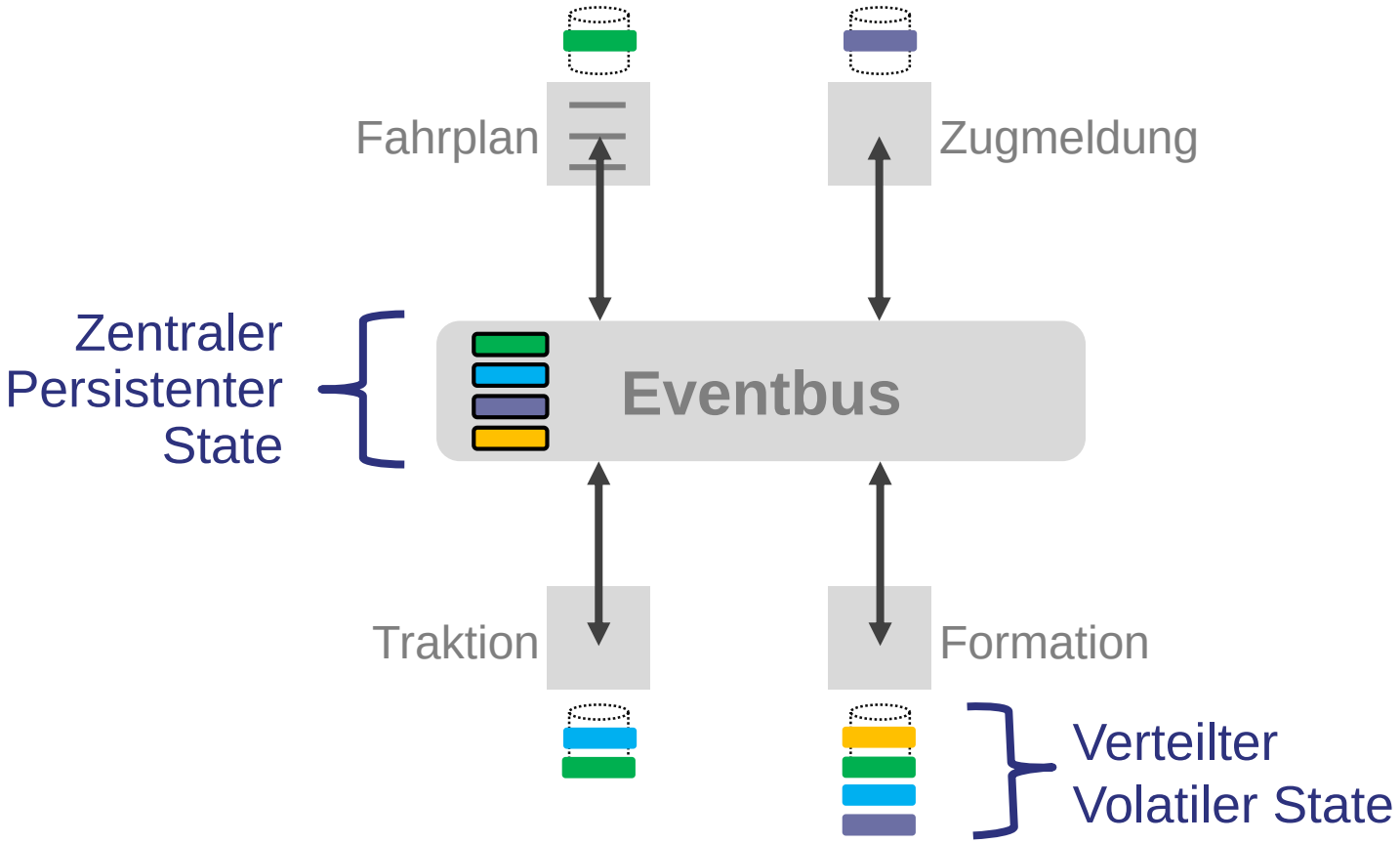
Primary

Replica

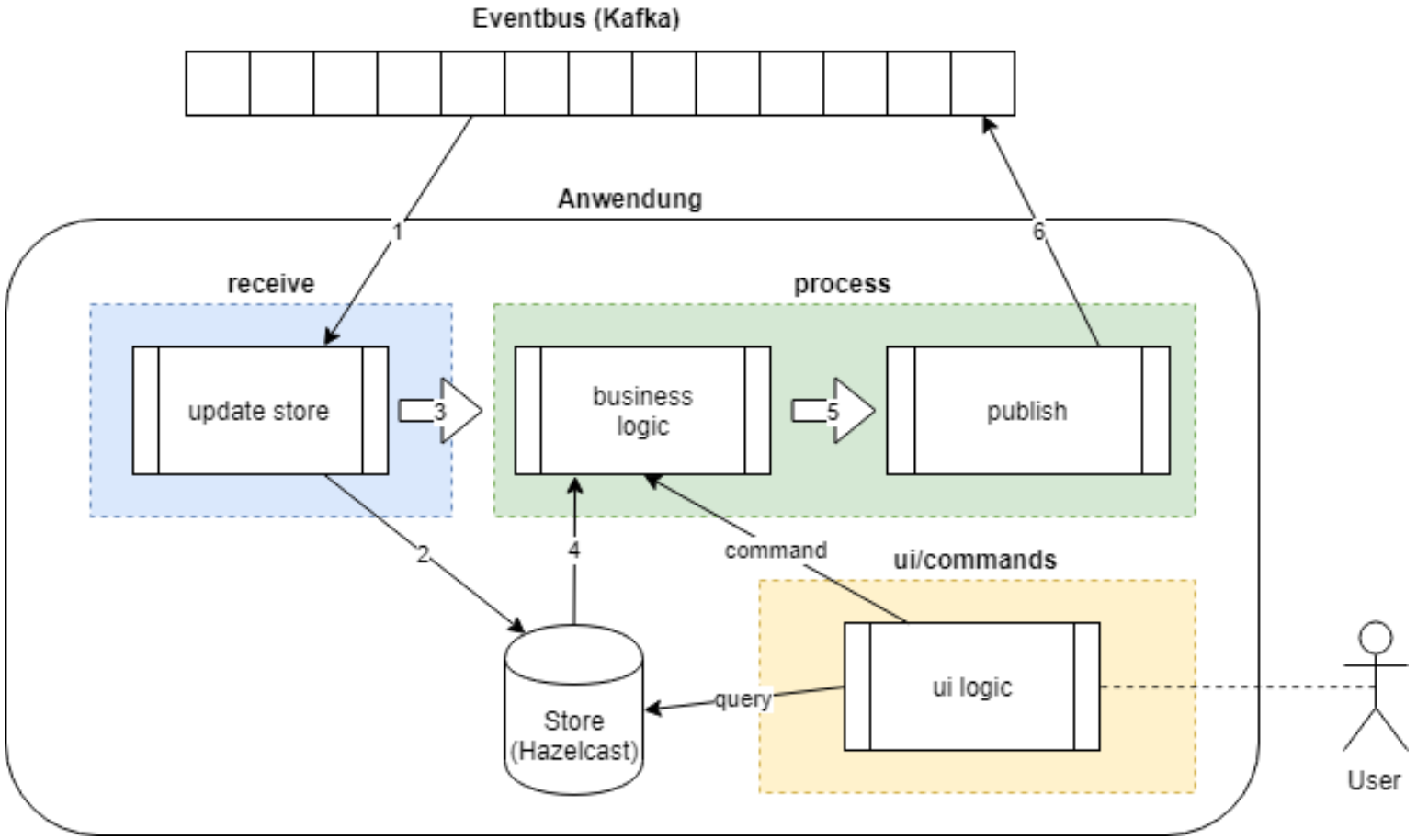
Änderung im Primary muss in allen Replicas nachgeführt werden!

SBB CFF FFS

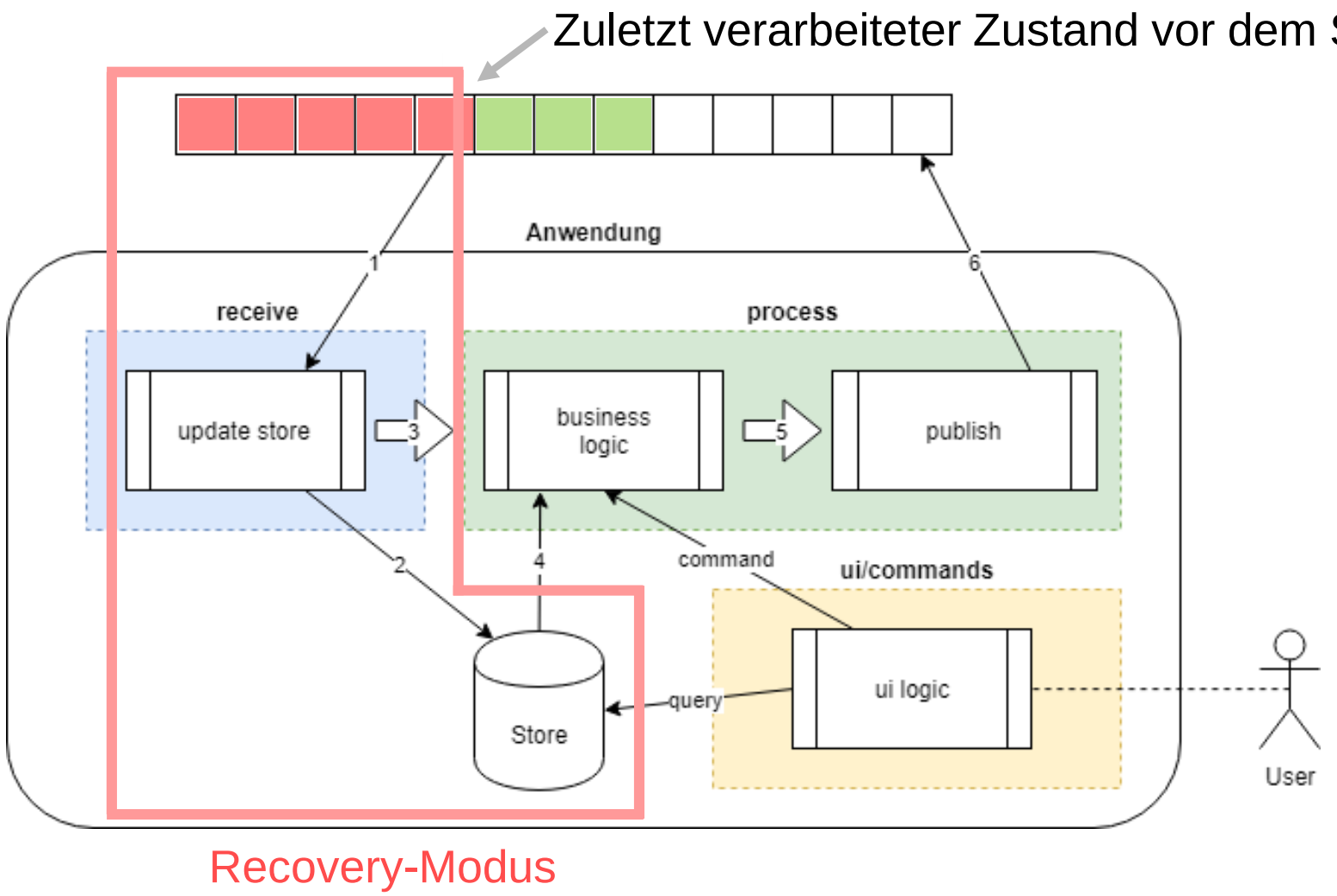
Zentraler State – Single Source of Truth



Verarbeitungsmodell

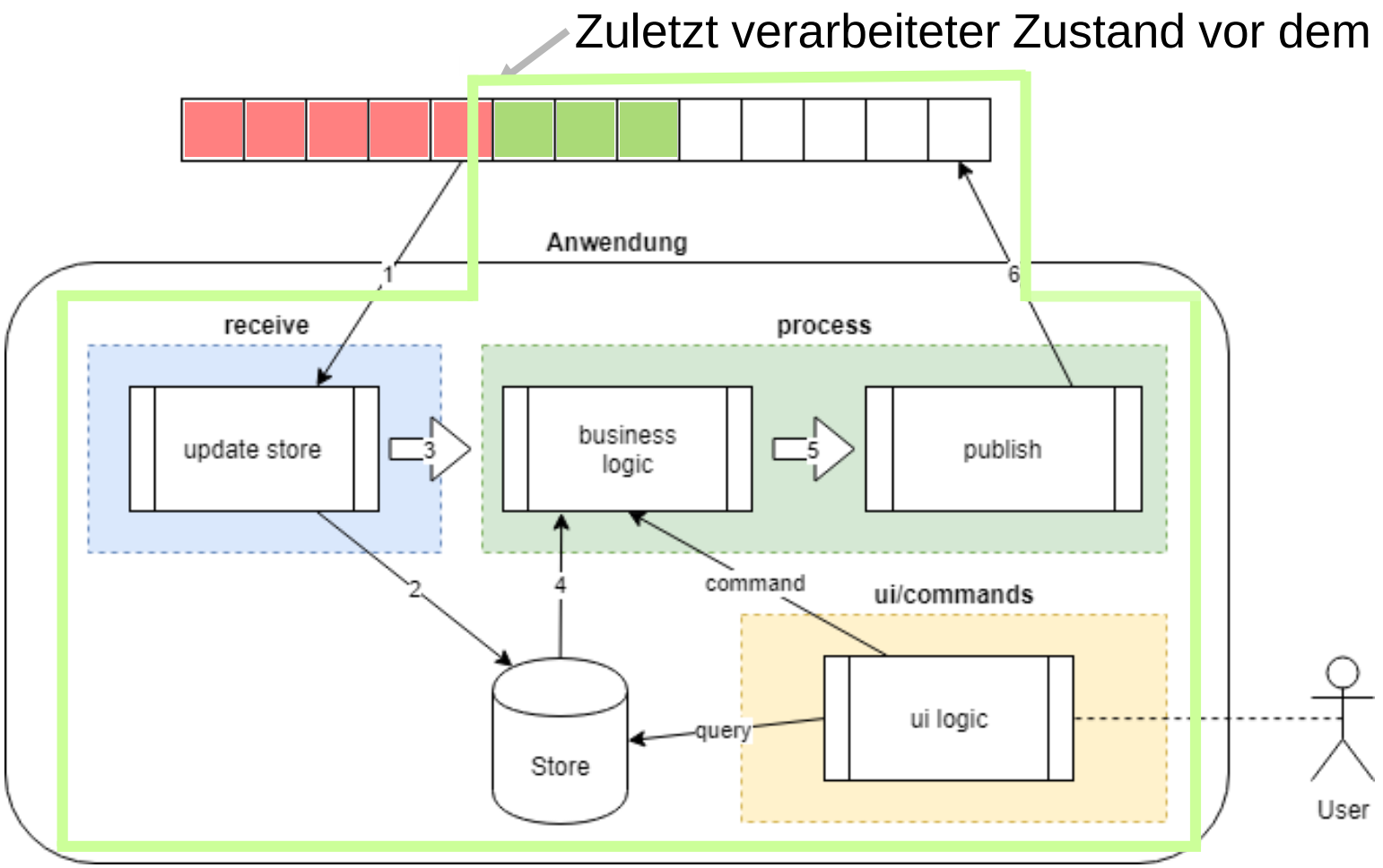


Verarbeitungsmodell: State Recovery nach einem Cold-Start



SBB CFF FFS

Verarbeitungsmodell: State Recovery nach einem Cold-Start



Normal-Modus

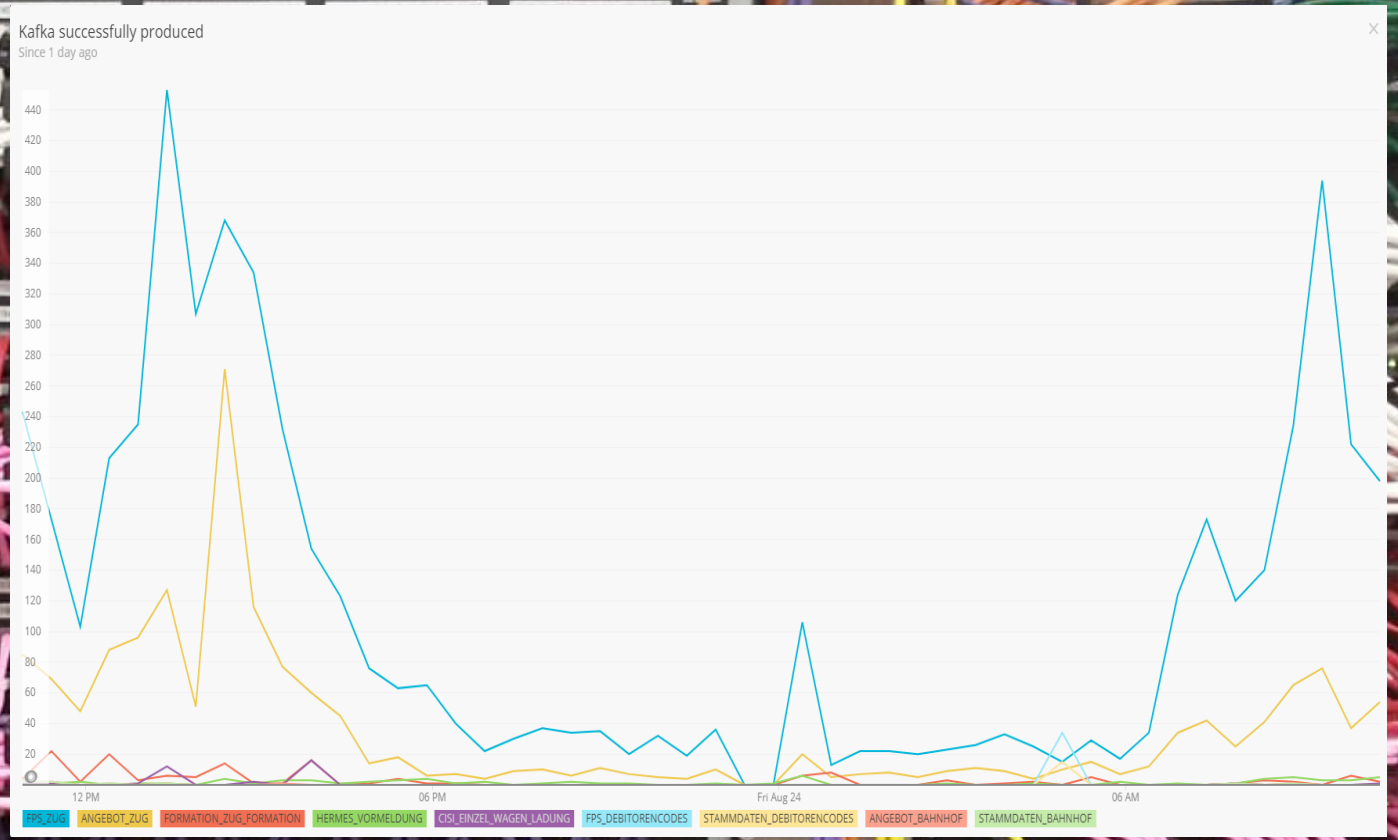
https://stackoverflow.com/questions/52666551/event-sourcing-with-kafka-for-microservices

Boilerplate «Kafka/Event-Sourcing» in ein Framework extrahiert.

- Recovery Detection
- Orchestrierung receive/process
- Duplicate Check
- Concurrency → Optimistic Locking
- Fehlerhandling → Dead Letter Topic
- Monitoring/Tracing

Techstack

- Kafka Cluster auf OpenShift
- Hazelcast Embedded/Clustered
- Spring Kafka
- New Relic
- Zipkin



Techstack: Benefits

Kafka

- Persistenz & Messaging
- Skalierung mittels Partitionen
- Streaming Plattform
- Keine Datentransformation

– Technisches vs Fachliches Log (Retention/ Compaction)

Hazelcast

- Schnell (In Memory)
- Keine Datentransformation
- Natürlicher Partner mit Kafka (Key, Value)

– Reduzierte Query Möglichkeiten

Spring Kafka

- Kafka Consumer/Producer the spring way

Lessons Learnt & Ausblick

- Kafka Streaming & Interactive Queries
- Aufbau & Betrieb Kafka (Projektrisiko)
- Hazelcast Query Tool?

- Agilität der event-basierten Architektur
- Stabilität & Ökosystem & Toolunterstützung
- Performance
- Produktive Daten

SBB CFF FFS

Fragen

→ Kontakt:

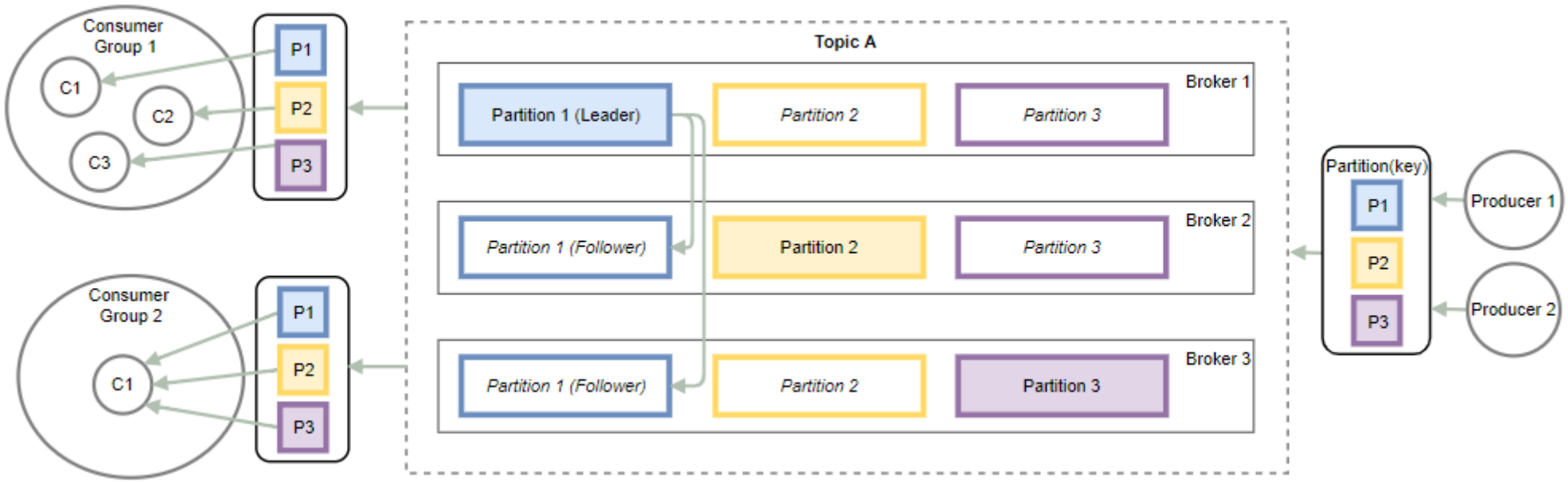
- korhan.guelseven@sbb.ch
- renato.loeffel@adesso.ch

SBB CFF FFS

Backup

SBB CFF FFS

Kafka: Skalierung & Failover



SBB CFF FFS

Concurrency & Optimistic Locking