TRANSAKTIONALE VERARBEITUNG MIT **APACHE CAMEL**

oepfelbaum
BANKING TECHNOLOGY EXPERTS

UM WAS GEHTS?

Event Driven Architecture hat viele Vorteile,
birgt aber auch Gefahren

Asynchrone Kommunikation:
- Sender wartet nicht auf Antwort

Keine Garantie, dass jede Message im Ziel ankommt

Apache Camel als Integrationssoftware

Kafka sendet nach Prinzip **"Fire and Forget"**

Empfang der Messages nicht garantiert

→ **Informationen gehen verloren!**

oepfelbaum
BANKING TECHNOLOGY EXPERTS

KEY CONCEPTS

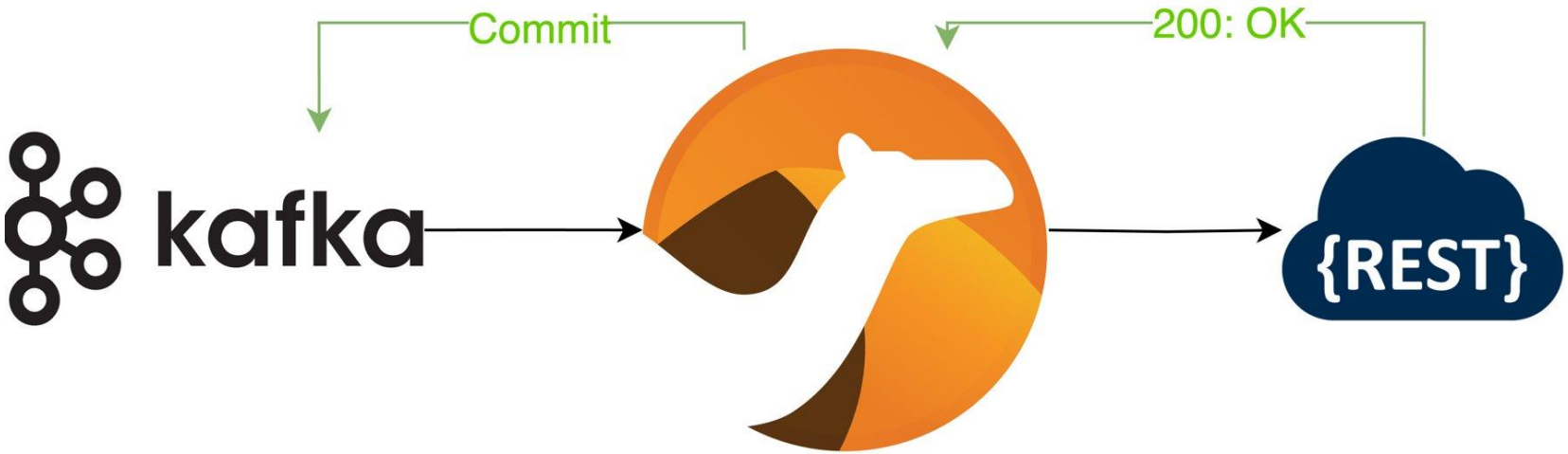
1. **Kein Commit**, bis die Message korrekt verarbeitet wurde

2. Nicht verarbeitbare Messages in die **Dead Letter Topic** zurücksenden

3. **Redelivery/Retries**

oepfelbaum
BANKING TECHNOLOGY EXPERTS

SZENARIO: HAPPY PATH

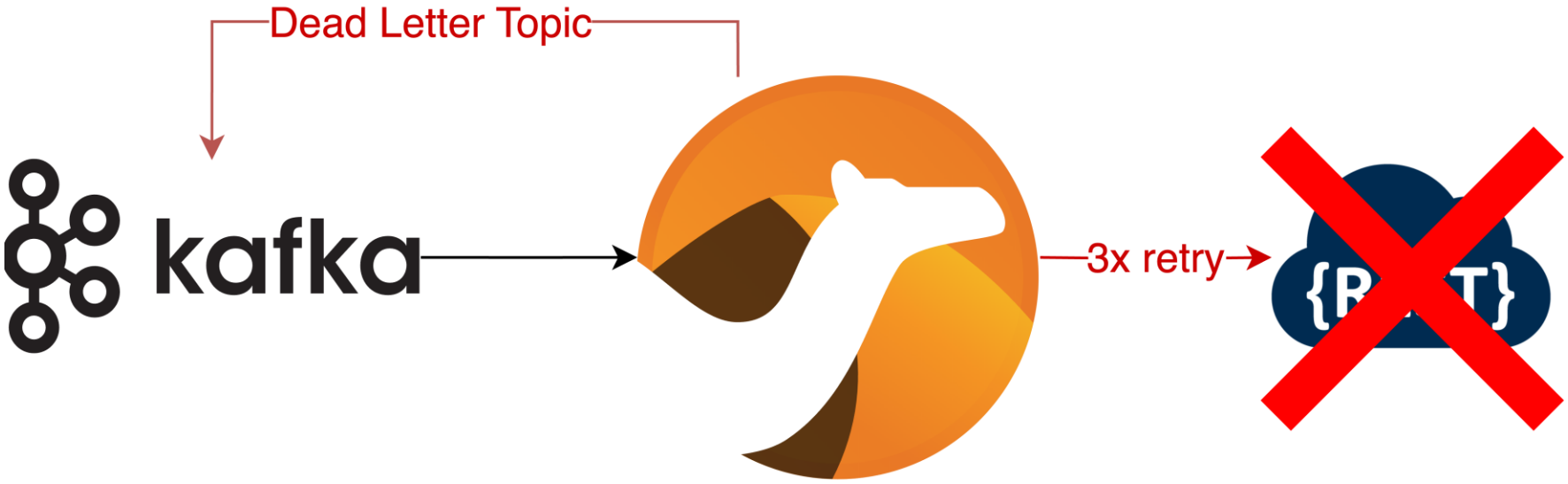


Message erreicht Endpoint

Acknowledge/Commit kommt zurück

oepfelbaum
BANKING TECHNOLOGY EXPERTS

SZENARIO: REST API OFFLINE

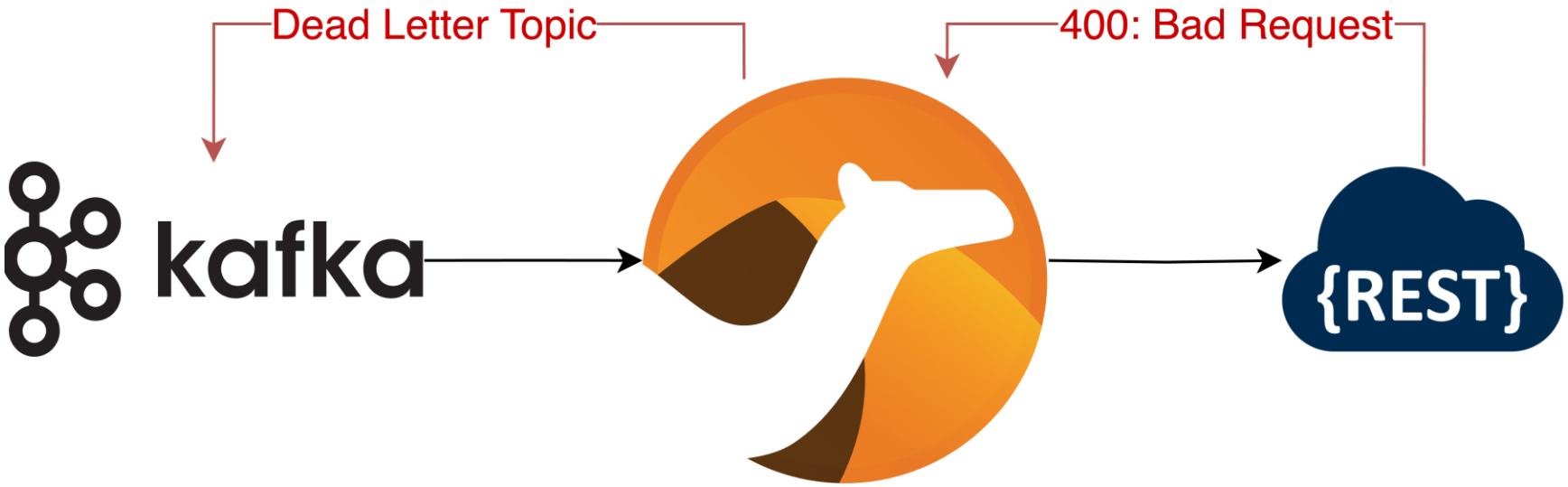


Camel versucht zuerst **Redelivery**

Message geht **zurück in DLT**

oepfelbaum
BANKING TECHNOLOGY EXPERTS

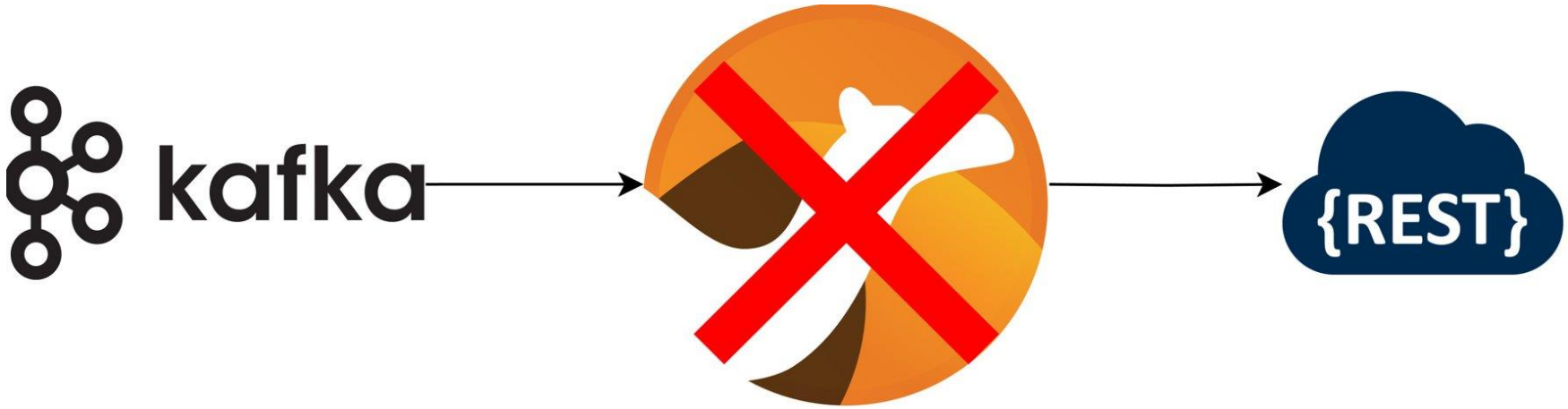
SZENARIO: 400 BAD REQUEST



Redelivery macht keinen Sinn

Message geht direkt in die **Dead Letter Topic**

SZENARIO: CAMEL CRASH



Ohne Transaktion: Kafka committed Message, die nicht ankommt

Mit Transaktion: Kafka verschickt Message erneut

DETAILS ZUR IMPLEMENTATION

Konfiguration **komplett in Apache Camel** umsetzbar

→ Apache Kafka sowie REST-Server bleiben unberührt

KAFKA CONSUMER

from("kafka") →

to("rest") →

```
// Kafka Consumer
from("kafka:" + kafkaTopic +
        "?brokers=" + kafkaBrokers +
        "&groupId=" + kafkaGroupId +
        "&maxPollRecords=1" +
        "&autoOffsetReset=earliest" +
        "&autoCommitEnable=false" +
        "&allowManualCommit=true")
    .log("Consuming message from Kafka")
    // Copy Header to a Property, because the Header will be filtered
    .setProperty(KafkaConstants.MANUAL_COMMIT, header(KafkaConstants.MANUAL_COMMIT))
    .choice()
        .when(body().contains("MQ"))
            .log("Sending message to MQ")
            .to("jms:queue:" + jmsSinkQueue)
        .when(body().contains("REST"))
            .log("Sending message to REST API")
            .setHeader(Exchange.HTTP_METHOD, constant(HttpMethods.POST))
            .setHeader("Content-Type", constant("text/plain"))
            .to(restSinkAddress)
    .end()

    // Commit Kafka Offset when Camel Exchange is completed
    .onCompletion().onCompleteOnly()
        .process(exchange -> {

            KafkaManualCommit manual = exchange.getProperty(KafkaConstants.MANUAL_COMMIT, KafkaManualCommit.class);
            if (manual != null) {
                manual.commit();
                System.out.println("Kafka offset committed");
            }
        })
    .end();
```

Autocommit deaktivieren

Commit manuell ausführen

opfelbaum
BANKING TECHNOLOGY EXPERTS

ERROR HANDLER

```
errorHandler(deadLetterChannel("kafka:DeadLetterChannel")
        .maximumRedeliveries(3)
        .retryAttemptedLogLevel(LoggingLevel.INFO)
);
```

```
// No Redelivery for 4xx errors
onException(HttpOperationFailedException.class)
        .onWhen(exchange -> {
            HttpOperationFailedException exception =
                    exchange.getProperty(Exchange.EXCEPTION_CAUGHT, HttpOperationFailedException.class);

            return HttpStatus.valueOf(exception.getStatusCode()).is4xxClientError();
        })
        .handled(true)
        .to("log:4xx?level=ERROR");
```

400 Error:
Ohne Redelivery
direkt in die DLT

oepfelbaum
BANKING TECHNOLOGY EXPERTS

IBM MQ CONSUMER

```
// JMS MQ Consumer
from("jms:queue:" + jmsSourceQueue + "?transacted=true")
        .log("Consuming message from MQ")
        .choice()
            .when(body().contains("MQ"))
                .log("Sending message to MQ")
                .to("jms:queue:" + jmsSinkQueue)
            .when(body().contains("REST"))
                .log("Sending message to REST API")
                .setHeader(Exchange.HTTP_METHOD, constant(HttpMethods.POST))
                .setHeader("Content-Type", constant("text/plain"))
                .to(restSinkAddress)
        .end();
```

Support für **alle JMS Implementationen**!

CONCLUSION

Problem ist gelöst, aber...

Potenzielle Drawbacks:

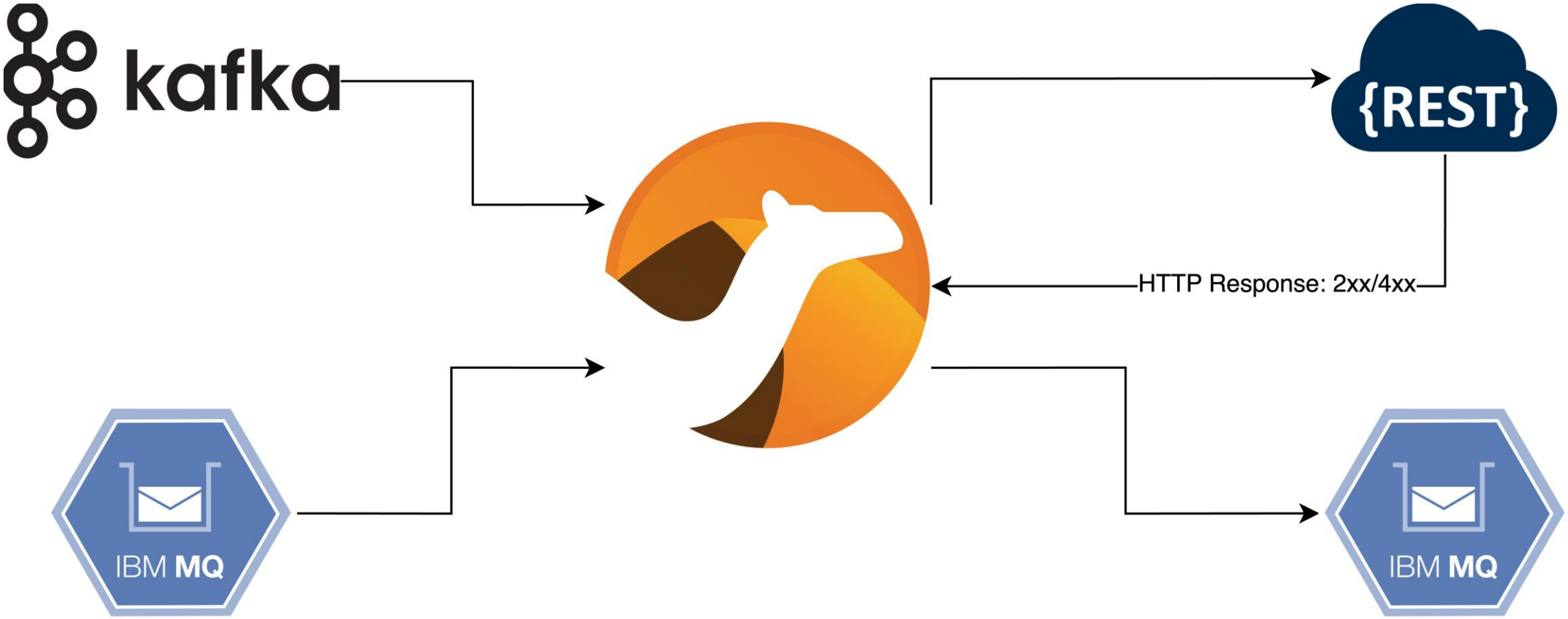
- Sind die Messages noch **asynchron**?
- Lediglich eine **"at-least-once"**-Delivery
- Nur Sicherung zwischen **2 Services**

Q&A

Bei weiteren Fragen:

- lukas.yu@oepfelbaum.com
- +41 78 849 76 78

DEMO



https://github.com/Oepfelbaum/transactional-camel

oepfelbaum
BANKING TECHNOLOGY EXPERTS