# Kafka: Our Trusty Database Companion

Franz Neubert

# Me

- Software Engineer at Otto
- Javascript, Scala, Kafka, AWS

@Scarysize

/Scarysize

OTTO

# Tracking @ otto.de

## What

- User journey
- User interaction with features
- Personalized content

# Why

- Measure performance of features & improve the shop in a data driven way
- Personalize shopping experience e.g. via recommendations
- Capture general business KPIs

## Who

- Teams decide what to track
- Tracking is offered as a service to other teams
    - Server- and client-side APIs
    - Preprocessing of tracking data
    - Access to enriched data for analysis

## How

- Page impressions are tracked with additional information in **labels**
- Labels are represented by key-value pairs
- There is no fixed set of available labels

- Examples:
  - `san_SearchTerm`
  - `order_BasketItems`
- **1415** different labels in 25 groups (prefixes)

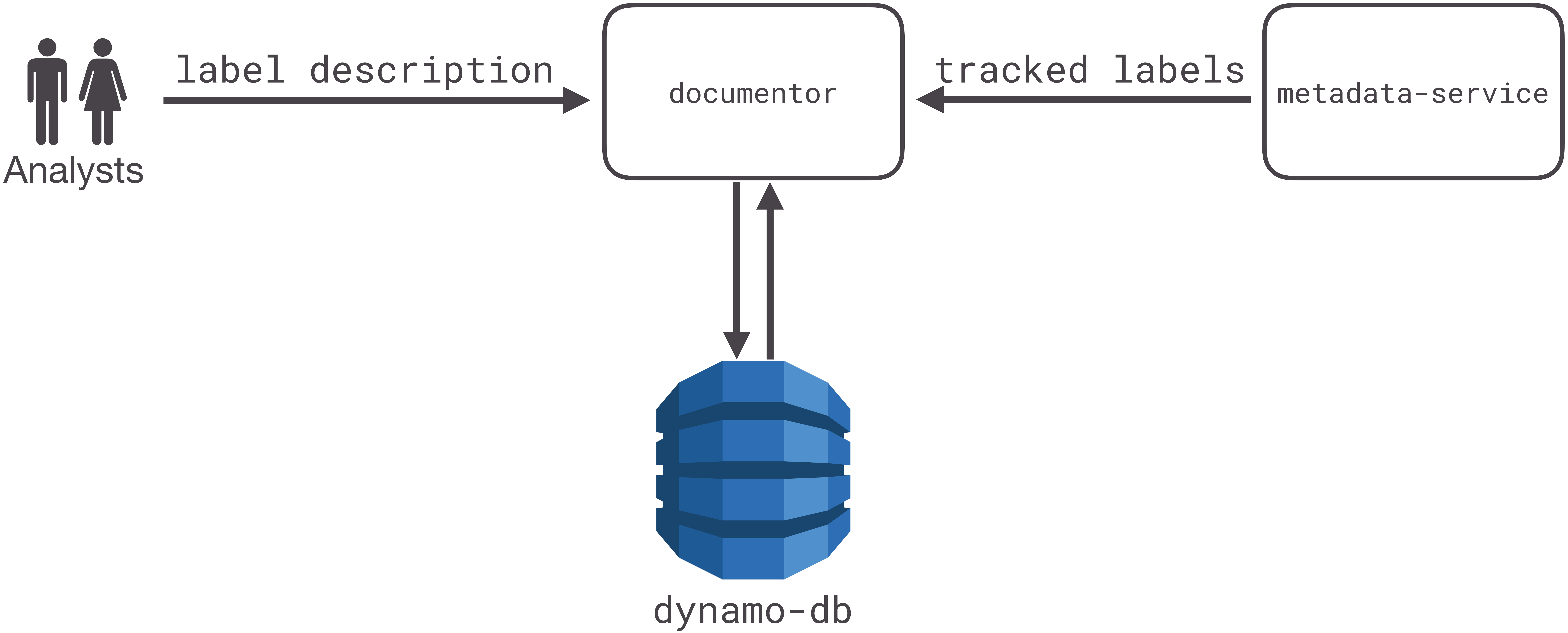
# Stack

- AWS
- Kafka Cluster with 6 Brokers, distributed across 3 availability zones
- About 20 Scala Services
- Handling ~400k req/min (client side tracking)
  - Peaking ~1.5m req/min

- Data transport
- Application state persistence
- Access to tracking data
- Key-value storage

# Documentation Service

- Enable analysts to document labels
- Make documentation data accessible to other services
- Restore live back-ups into develop environment

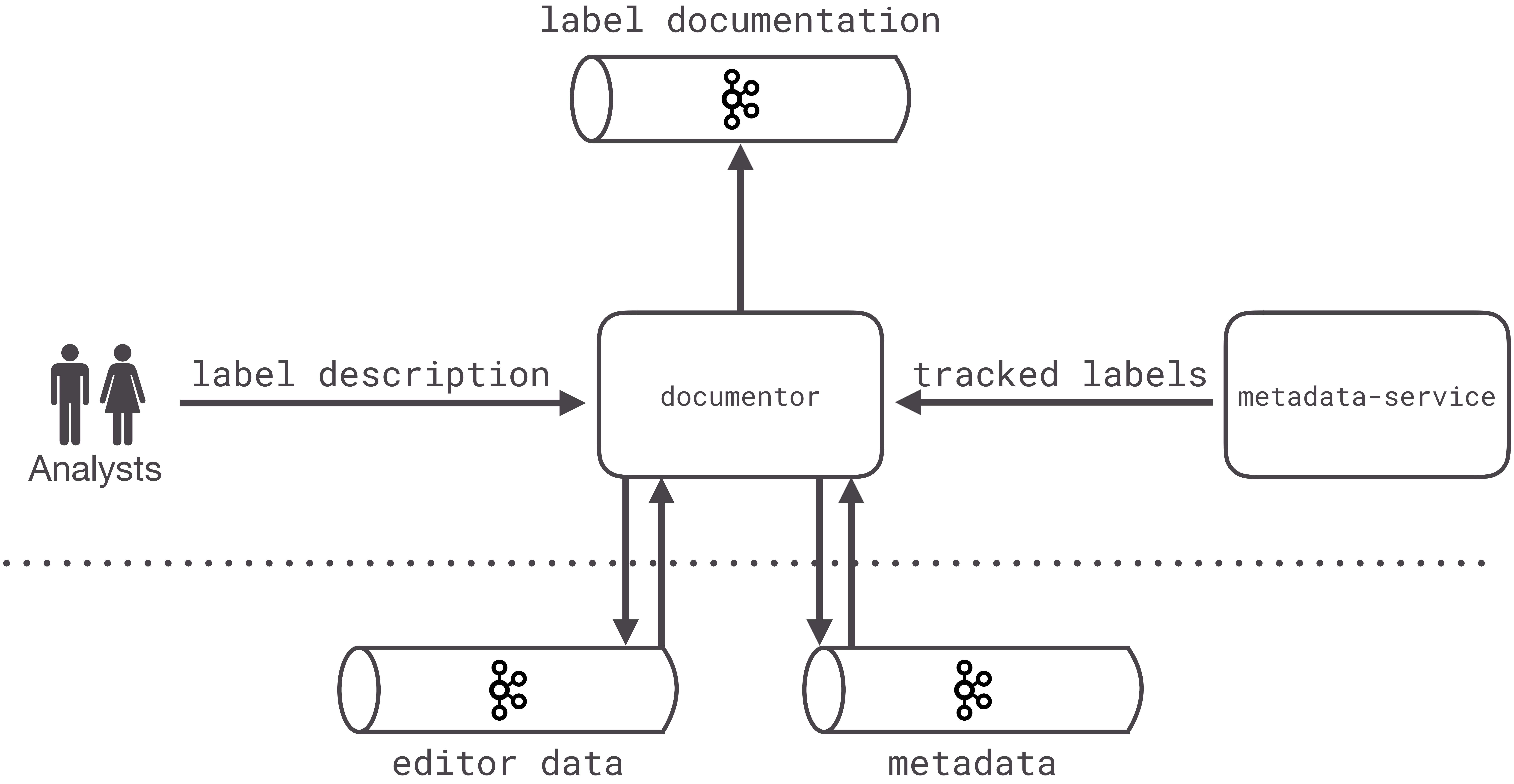

Analysts
label description
documentor
tracked labels
metadata-service
dynamo-db

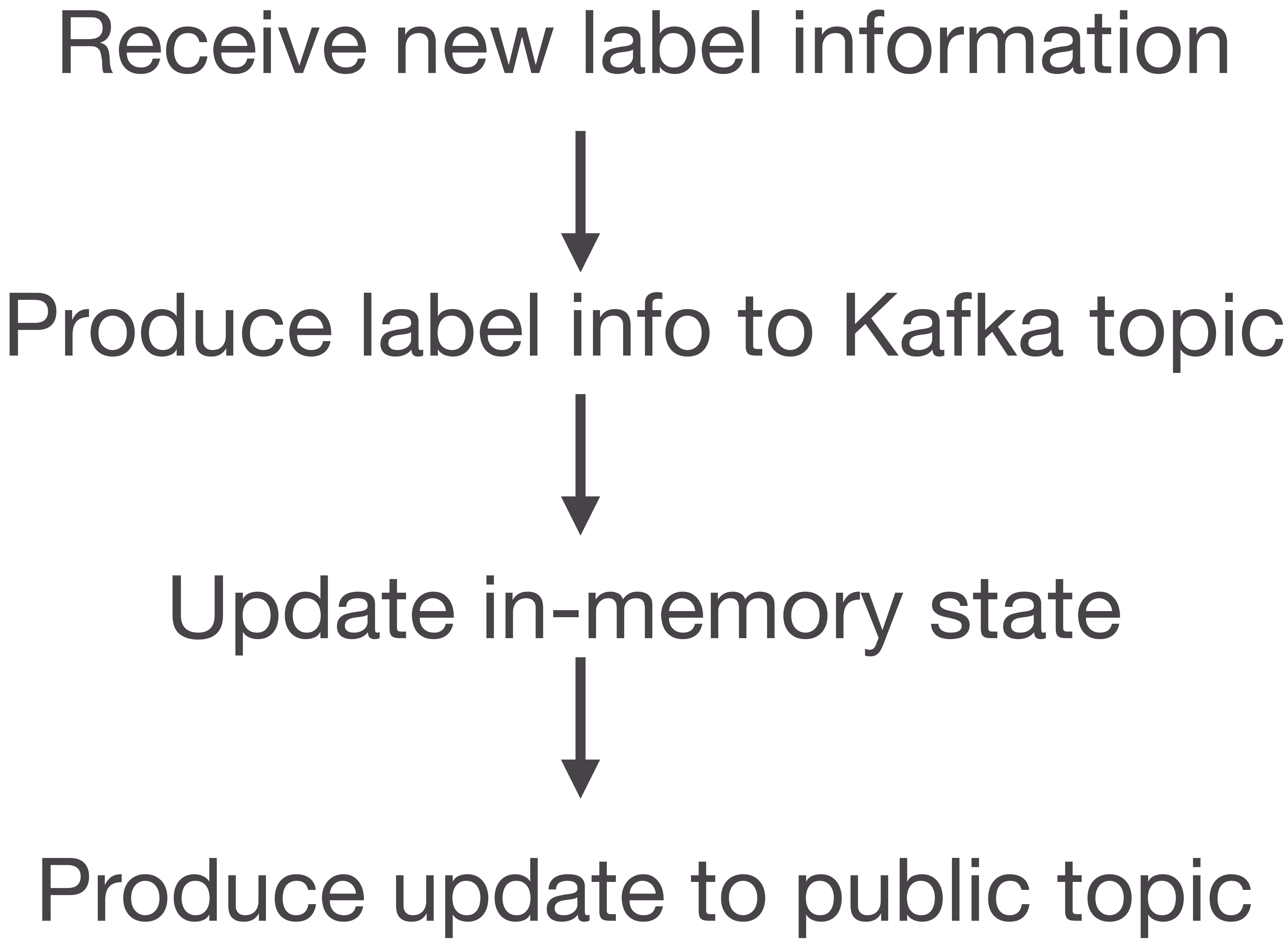
- Easy API via Java AWS SDK
- Straightforward  back-ups
- Simple to operate

- Easy API via Java AWS SDK
    - Funny behaviour with Java Boolean types
- Straightforward  back-ups
    - No managed cross-region back-ups or access to back-up data
- Simple to operate
    - Complicated pricing options

- DynamoDB felt like an additional piece of infrastructure we needed to **understand** and **manage**

- We already operate a Kafka cluster and have **high expertise** in working with it


label documentation
label description
documentor
tracked labels
metadata-service
Analysts
editor data
metadata



Receive new label information

Produce label info to Kafka topic

Update in-memory state

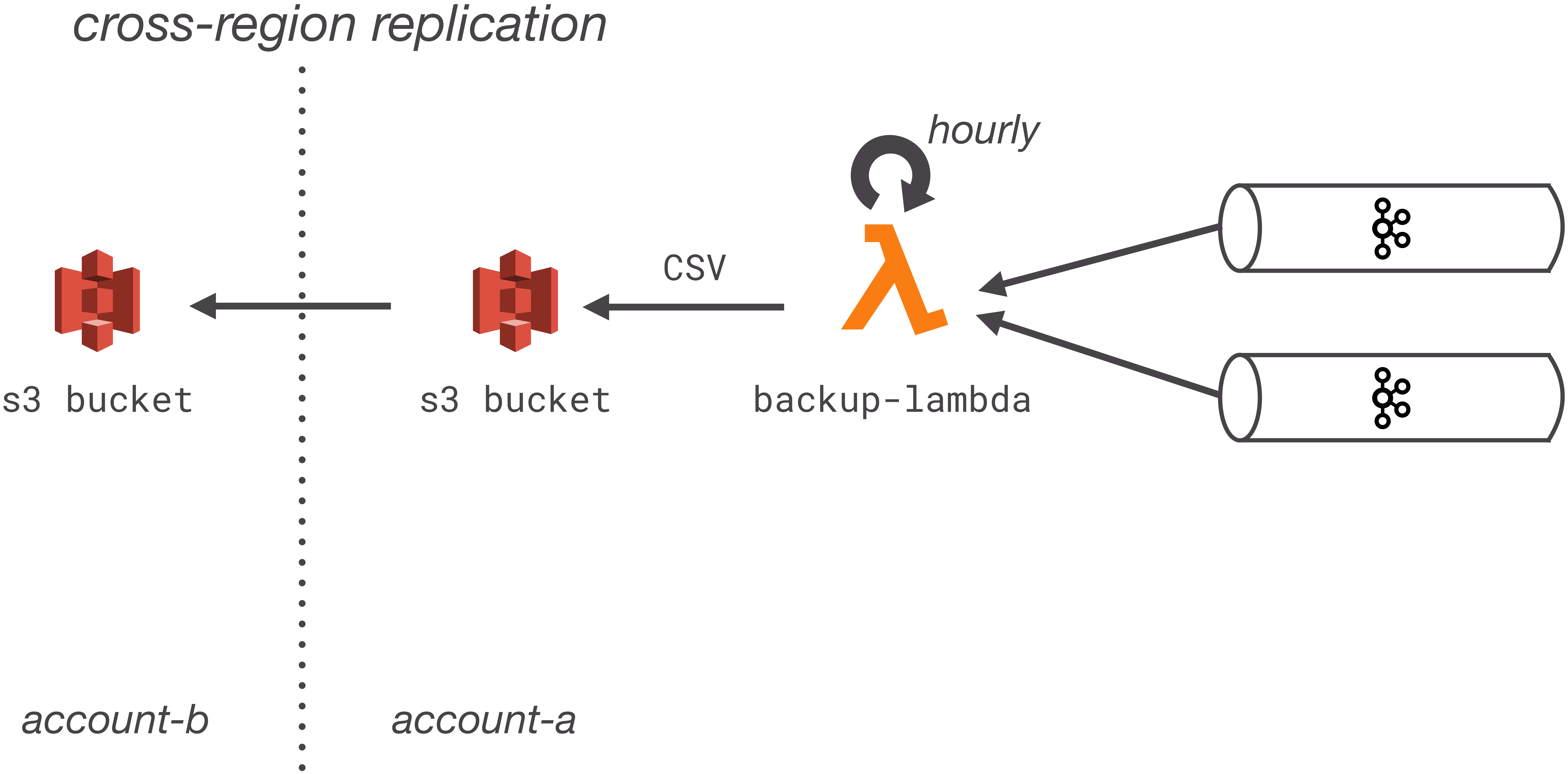
Produce update to public topic

- `cleanup.policy: compact`
- Enables log compaction
- Old messages will be compacted instead of being deleted
- Label name used as the key
- → We can use Kafka as a simple key-value store

# Data Access

- Access to documentation data via a separate topic
- Keeps internal format separate from “public” API
- Clients can consume updates at their own pace
- Updates propagate instantly to downstream consumers, no waiting for a scheduled HTTP call

# Back-up



*cross-region replication*

*hourly*

CSV

s3 bucket

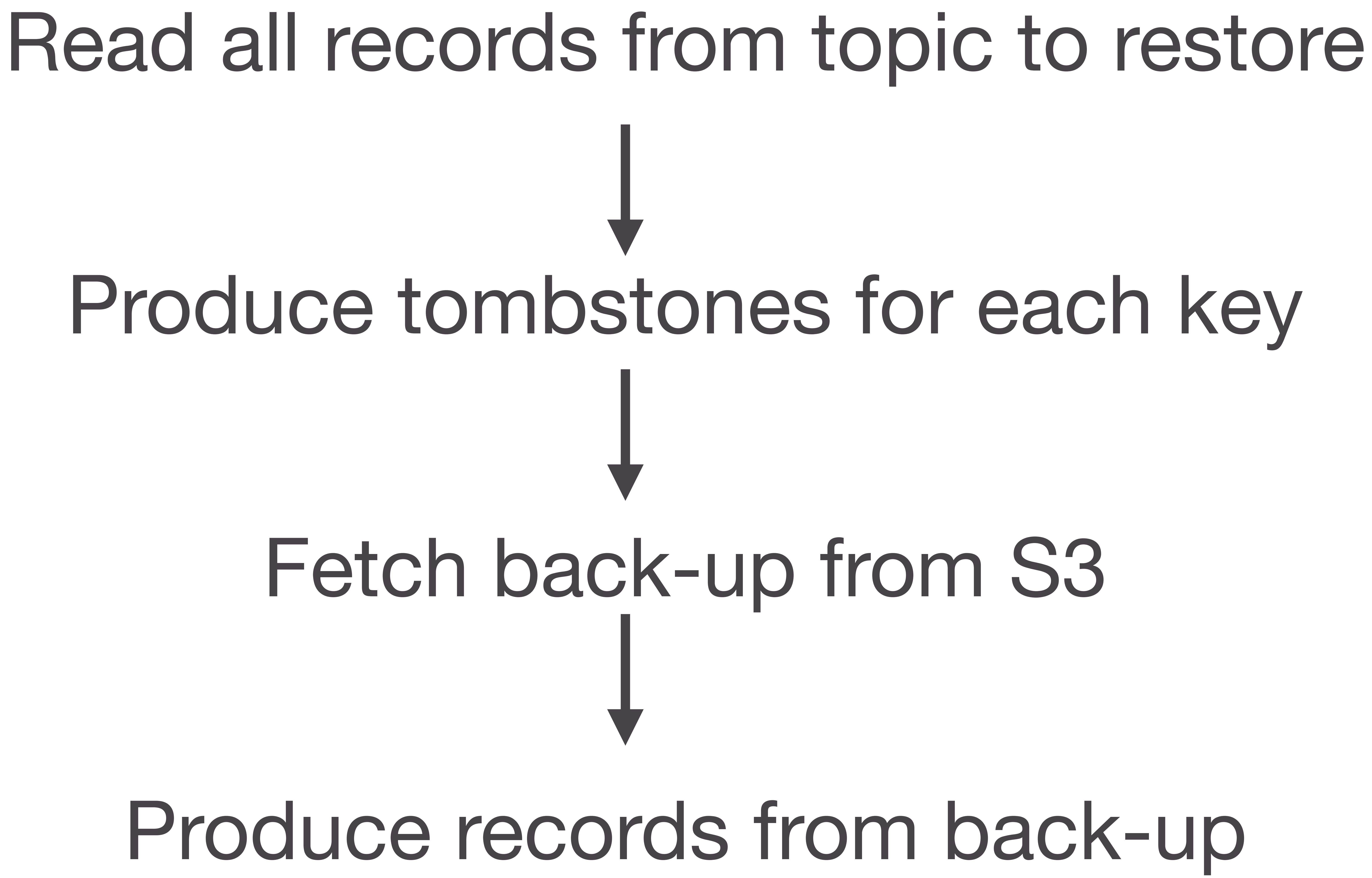
s3 bucket

backup-lambda

*account-b*

*account-a*

# Restore



Read all records from topic to restore

Produce tombstones for each key

Fetch back-up from S3

Produce records from back-up

# Outlook

- Receive metadata via Kafka not HTTP
- Use event sourcing in order to maintain a change log for each documented label
- Provide more information about labels
  - Deprecation dates, start of use dates etc.

# Thanks

Franz Neubert

@Scarysize

/Scarysize