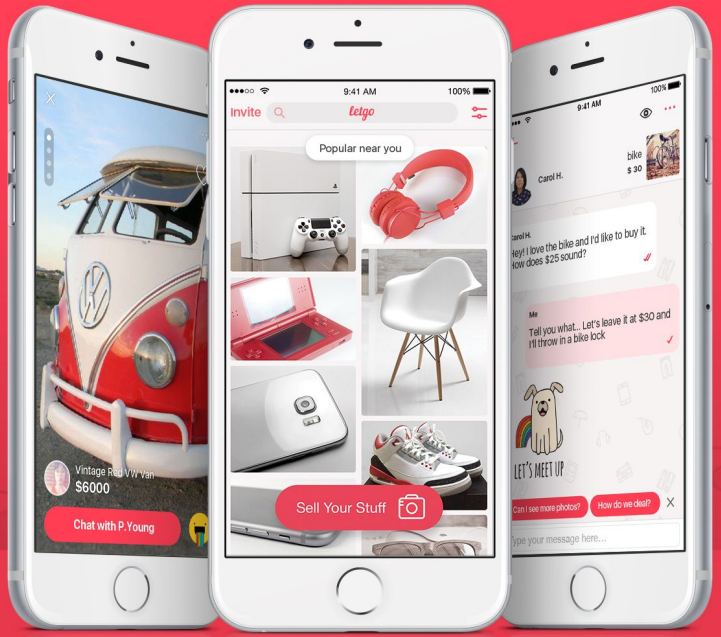
How to create a self-service data platform with guarantees by leveraging Avro schemas

Letgo Data Team

letgo



Founded in 2015

Phenomenal growth

130 MM downloads

Focused on USA and Turkey

Offices in Barcelona and NY

Letgo Data Platform in numbers

500GB
Data daily

850M
Events Processed Daily

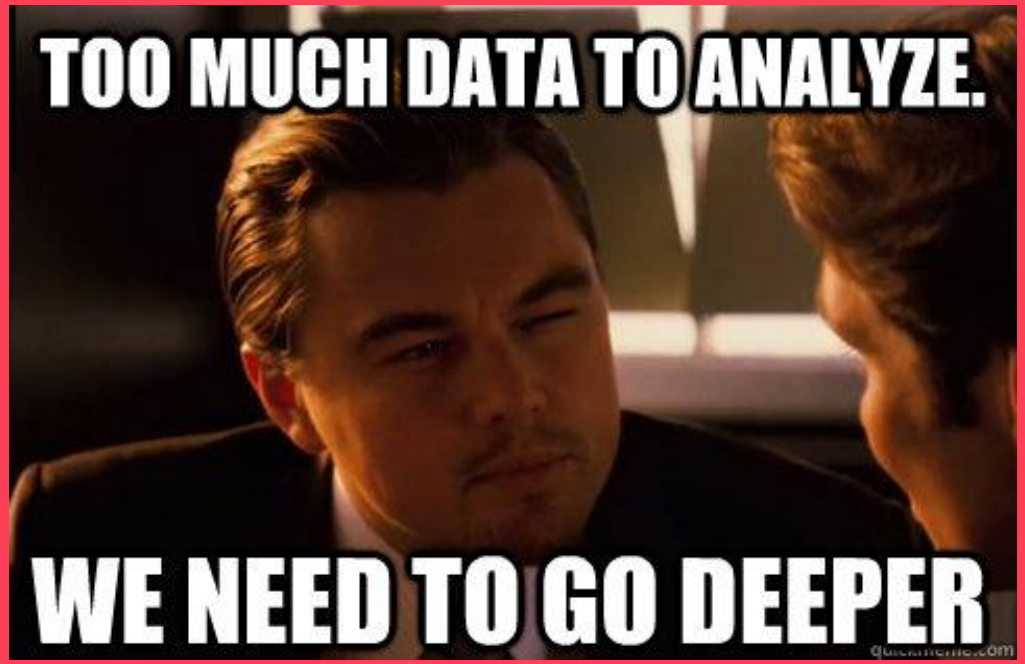
10K
Events Processed per Second

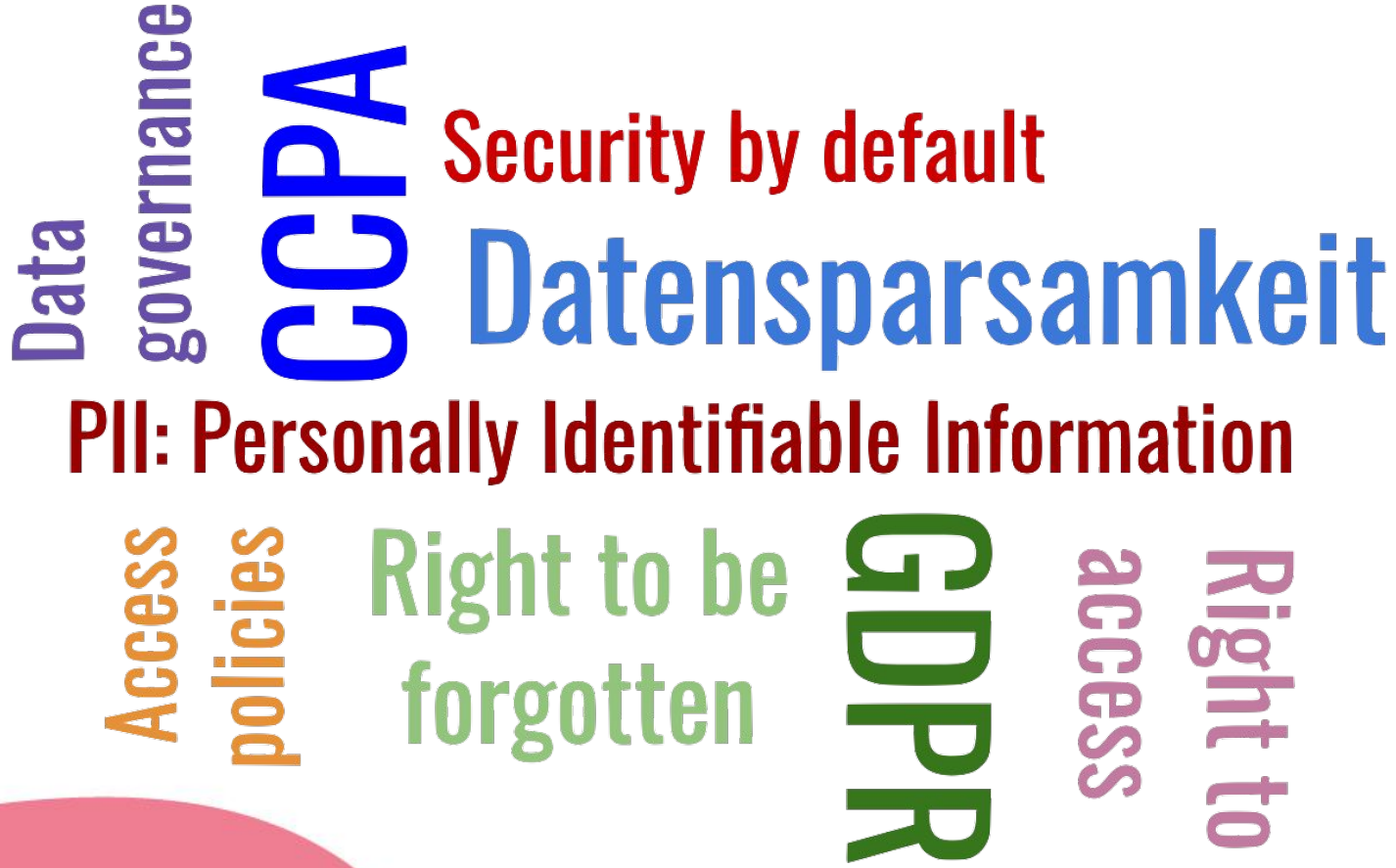
600+
Event Types

200TB
Storage (S3)

< 1sec
NRT Processing Time

Challenges




Data governance
CCPA
Security by default
Datensparsamkeit
PII: Personally Identifiable Information
Access policies
Right to be forgotten
GDPR
Right to access

User privacy rights

- **Right of access**: we are required to provide to the users a copy of their processed personal data upon request
- **Right to be forgotten**: users have the right to request that personal information be removed from our systems
- We need to know:
  - what information represents data related to users
  - where the information of a specific user resides in the data lake

Identifying personal information in Domain Events

- The same field can have different names in different events
- Not all events contain a *user_id*, need to join with other events

Two user ids?

```
"data":{
    "id":"00000-00000",
    "type":"message_sent",
    "conversation_id":"11111-11111",
    "from_user":"luke",
    "to_user":"han",
    "content":"Hello!",
    "sent_at": 1566901277000
}
```

No user id at all?

```
"data":{
    "id":"00000-00000",
    "type":"account_updated",
    "account_id":"11111-11111",
    "reason":"whatever",
    "updated_at":1566901277000
}
```

Usability vs security

- Employees should only access sensible data that they need to fulfill their business needs
- A Data Protection Officer (DPO) should explicitly allow access to sensible data when applicable

Architecture Overview

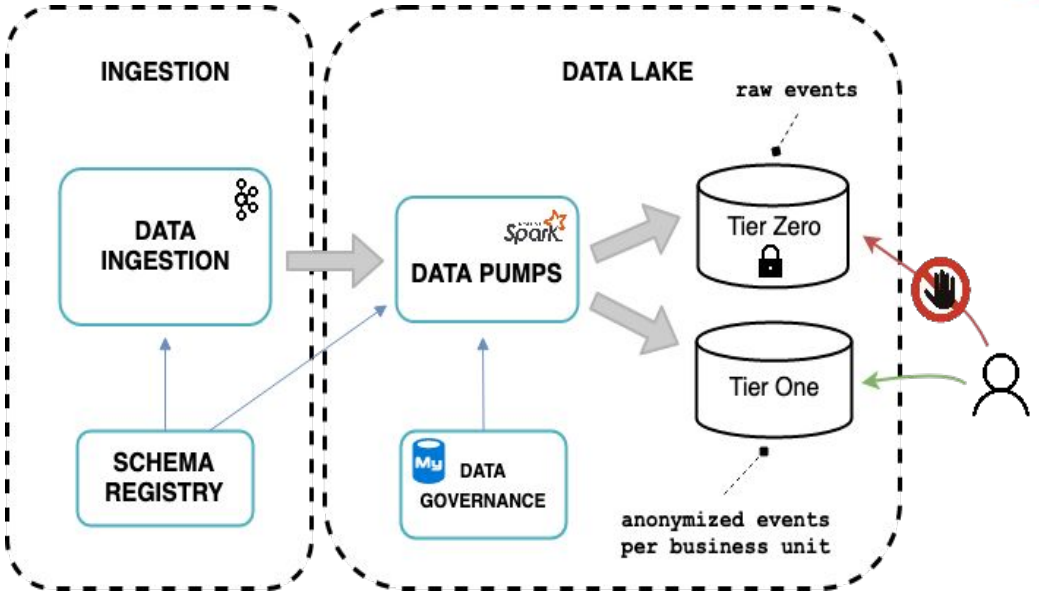
High-level architecture

Defining principles:

- Self-service platform
- Compliance to privacy laws
- Good development guarantees
- Minimal maintenance
- Cost containment by design

Main technologies:

- Spark for data processing
- Kafka and Kafka Connect for data transport
- Avro for data governance



Apache Avro

- Serialization protocol
- Schema-based
- Binary
  - Fast parsing!
  - Compact
  - Tools to inspect :(
- Polyglot: java, scala, js, python...
- Enforces structure of events
- Evolvable schemas!
- Tip: aim for full compatibility when evolving schemas

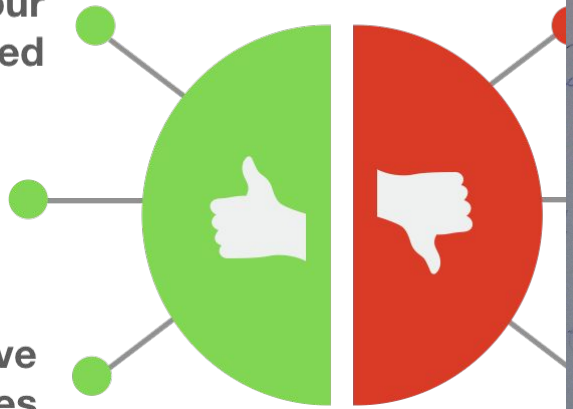
```
{
  "type":"record",
  "namespace":"com.letgo",
  "name":"ProductVisited",
  "fields":[
    {
      "name":"user_id",
      "type":"string",
      "doc":"Id of the visiting user",
    },
    {
      "name":"product_id",
      "type":"string",
      "doc":"Id of the visited product",
    },
    ...
  ]
}
```

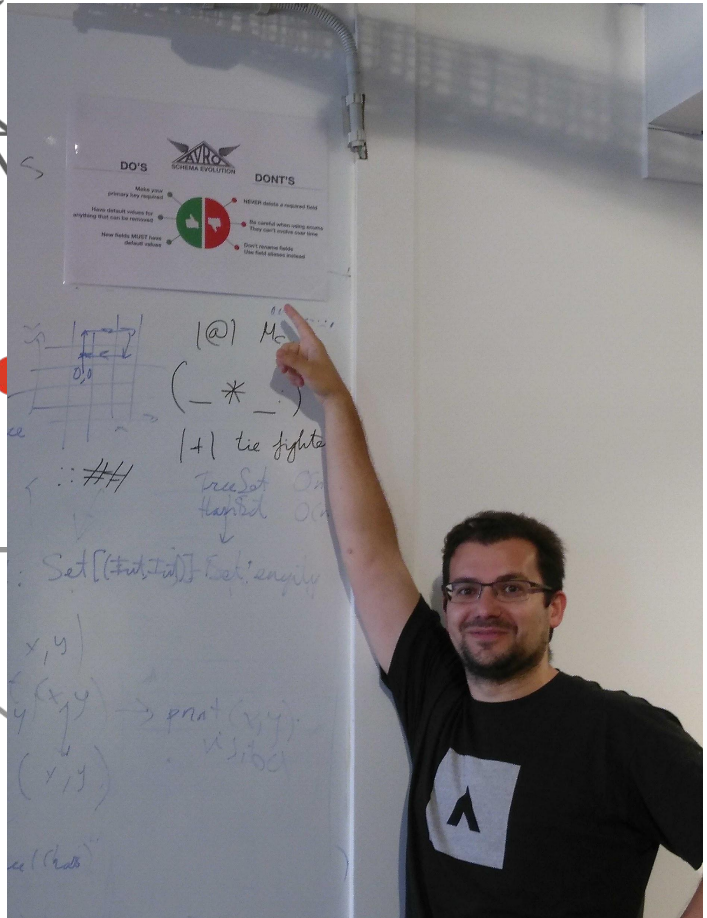
DO'S

Make your primary key required

Have default values for anything that can be removed

New fields MUST have default values





Apache Avro - tagging PII fields

- Out of the box schemas enforce name and type of fields, but don't hold metadata about the business context
- Need a way to know what each field represents...

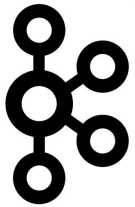
```
{
  "type":"record",
  "name":"ProductCreated",
  "fields":[
    {
      "name":"id",
      "type":"string"
    },
    {
      "name":"user_id",
      "type":"string"
    }
  ]
}
```

Apache Avro - tagging PII fields

- Out of the box schemas enforce name and type of fields, but don't hold metadata about the business context
- Need a way to know what each field represents...
- **Solution**: define custom tags to identify entities of interest!

```
{
  "type":"record",
  "name":"ProductCreated",
  "fields":[
    {
      "name":"id",
      "type":"string",
      "letgo.properties": {
        "entity": "product/id",
        "linked_to": "user_id"
      }
    },
    {
      "name":"user_id",
      "type":"string",
      "letgo.properties": {
        "entity": "user/id",
      }
    }
  ]
}
```

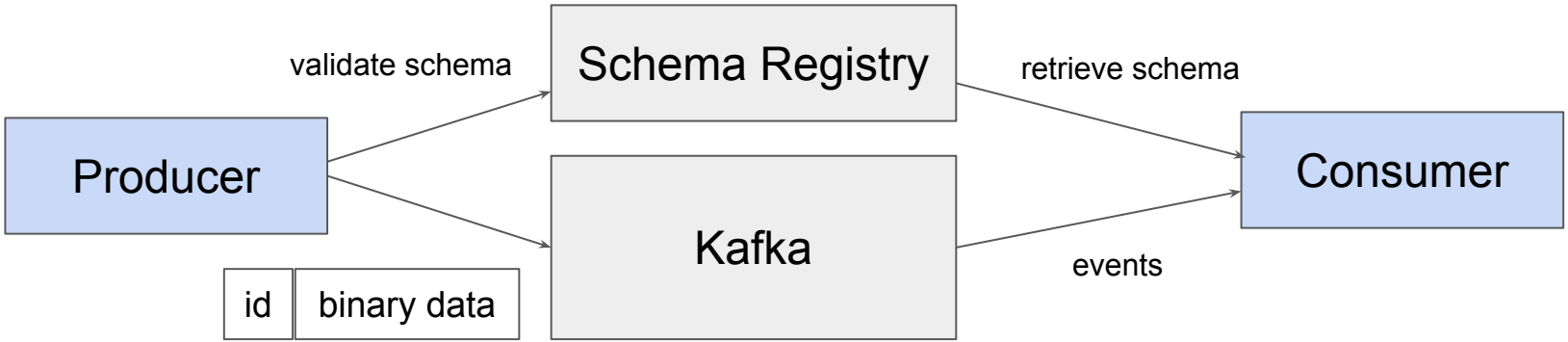
Apache Kafka



- Distributed commit log open sourced by LinkedIn
- Its creators founded Confluentic to provide commercial support
- Kafka offers
  - exactly-once delivery of events
  - high-throughput (millions of events per second)
  - high-reliability
- How?
  - for Kafka, payloads are just byte arrays
  - no parsing, zero-copy optimizations
- So, Kafka is blind regarding schemas and schema validation...

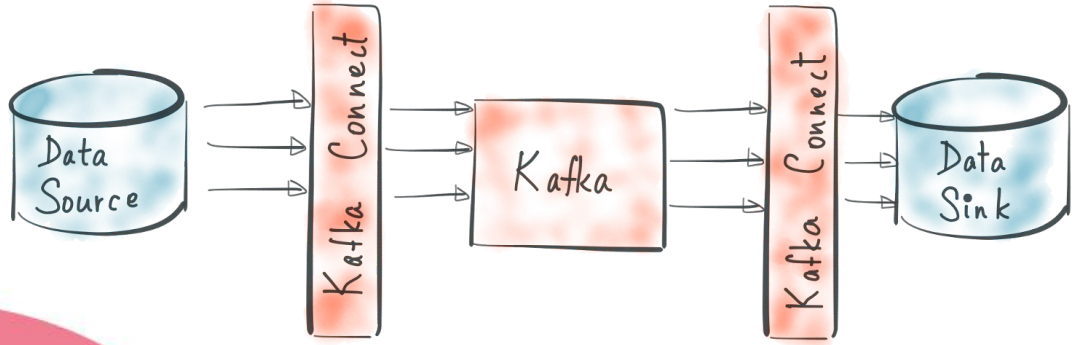
Kafka - Schema Registry

- Component developed by Confluentic (also OSS)
- Used at the edges of Kafka
- Enforces schema compatibility



Kafka Connect

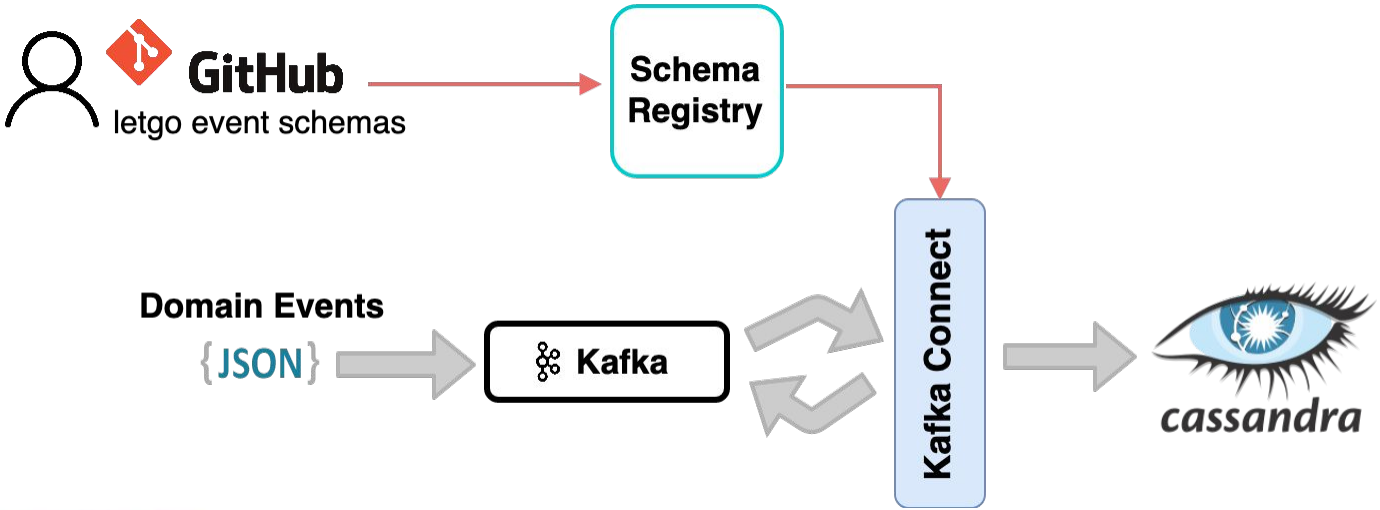
- Framework for connecting Kafka with external systems
  - **Source** connectors for moving data into Kafka
  - **Sink** connectors to move data out of Kafka
- Reliable, distributed, scalable, fault tolerant and low latency
- Standardized way to move data around inside the Kafka ecosystem without reinventing the wheel



Ingestion Platform

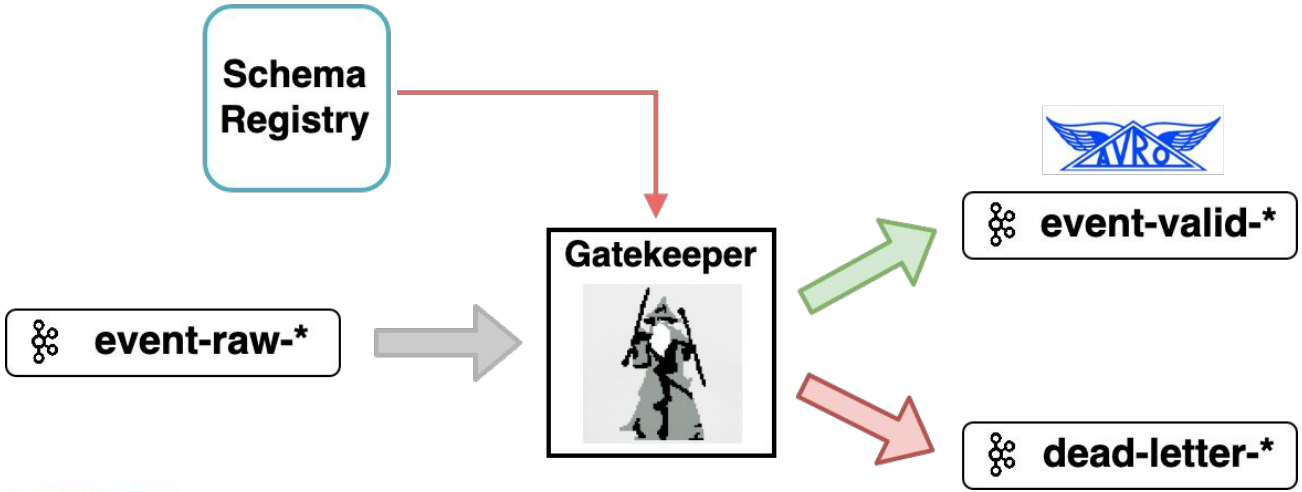
Ingestion platform architecture

- Raw events are ingested and validated with Kafka and connectors
- Cassandra table with TTL for data landing and deduplication



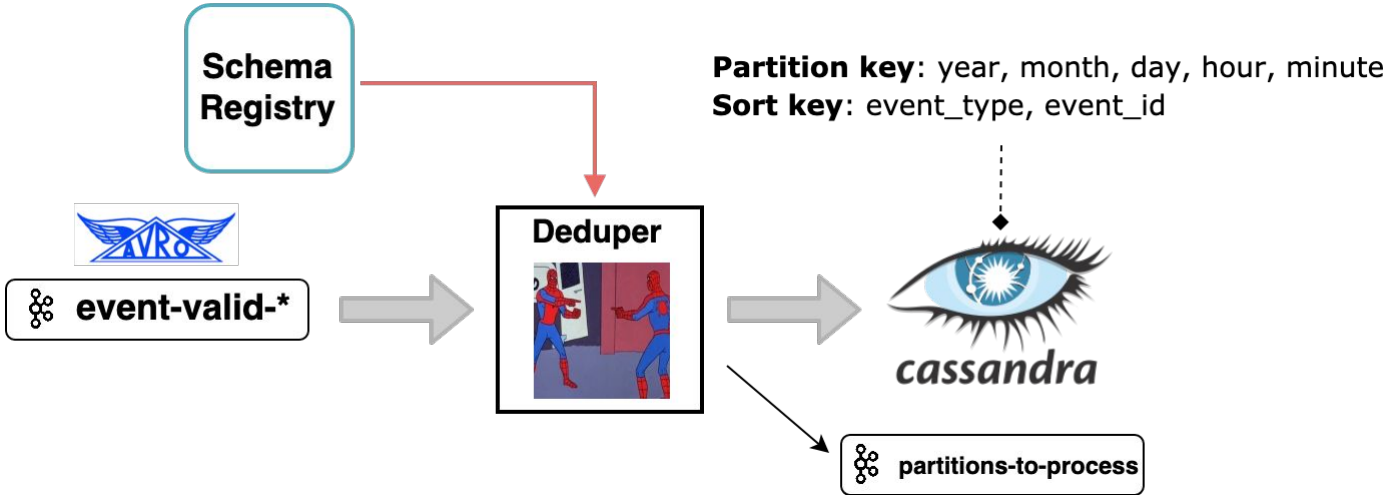
Gatekeeper

- Kafka connector sink that validates raw events based on schemas from the Schema Registry
- Publishes valid events serialized as Avro to valid Kafka topics

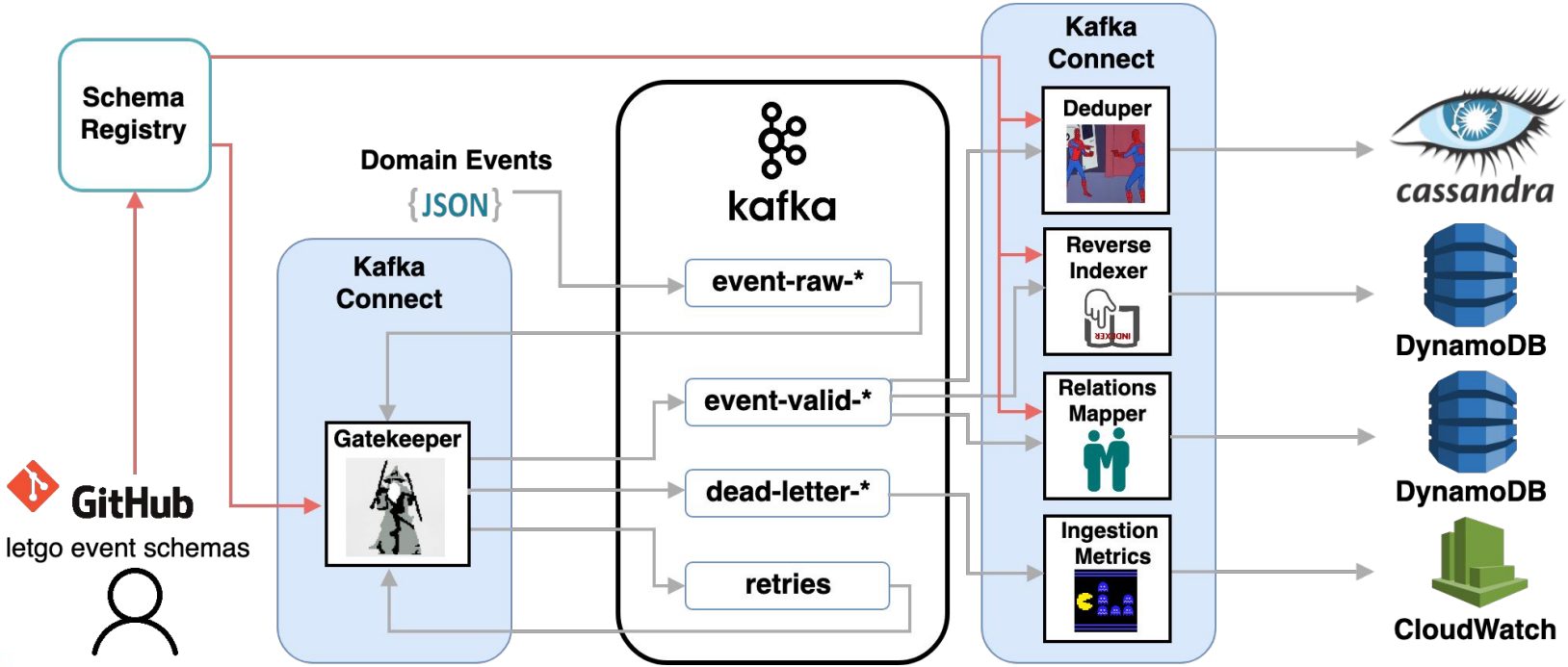


Deduper

- Kafka connector sink that lands valid Avro events into Cassandra
- Duplicated events hit the same partition key and get stored only once



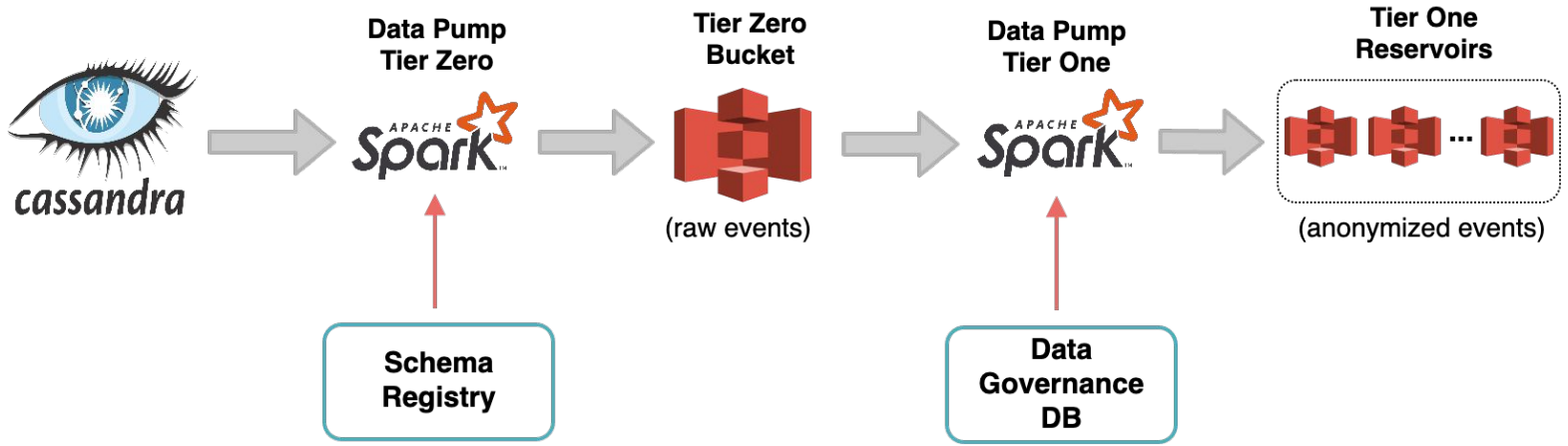
Ingestion platform review - the whole picture



Data Lake

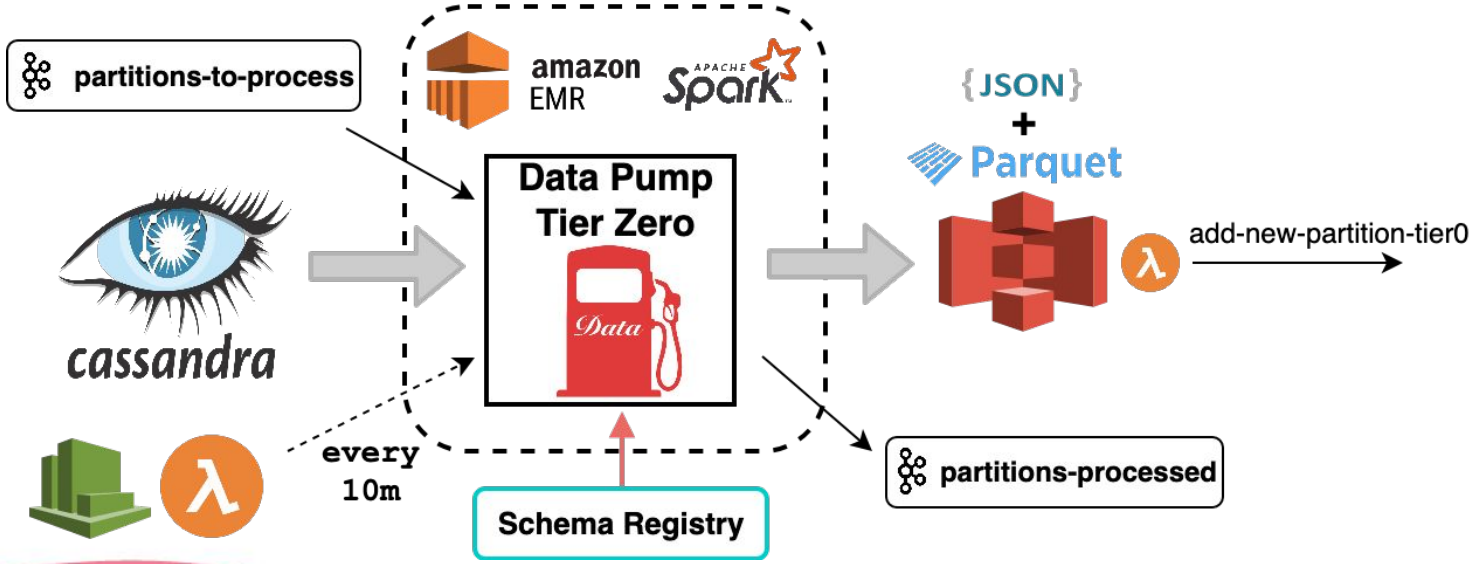
Data Lake architecture

- Events are loaded from Cassandra, deserialized and dumped to S3 Tier Zero (no access)
- Then they are copied to anonymized Tier One reservoirs with TTL



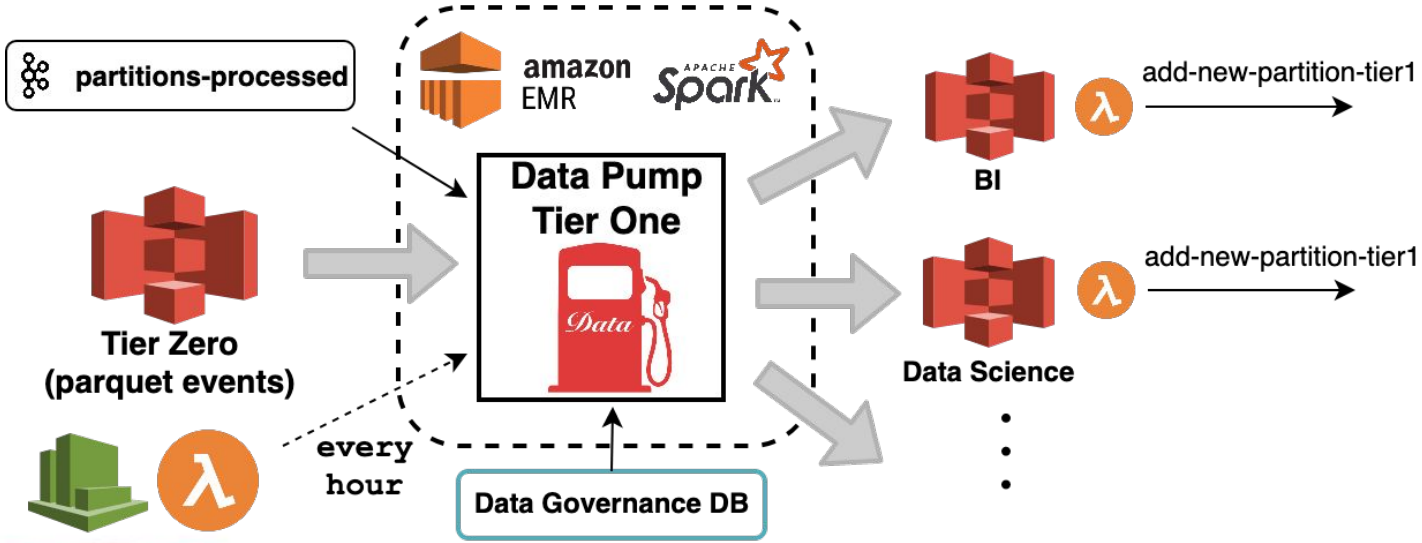
Data Pump Tier Zero

- Lives in a separate isolated account that nobody can access
- Partitions automatically reprocessed if late events appear



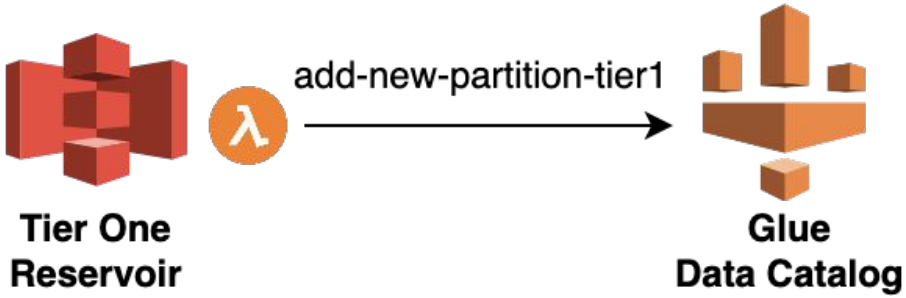
Data Pump Tier One

- New data is loaded from tier zero and sent to different “reservoirs”
- Each reservoir is anonymized as per data governance rules



Data catalog

- When data arrives to S3 it’s not yet visible in queries
- A lambda is triggered when a new object appears in S3 registering the new partition in the Glue Catalog
- If a partition is recomputed in tier zero, the change is propagated automatically to all reservoirs

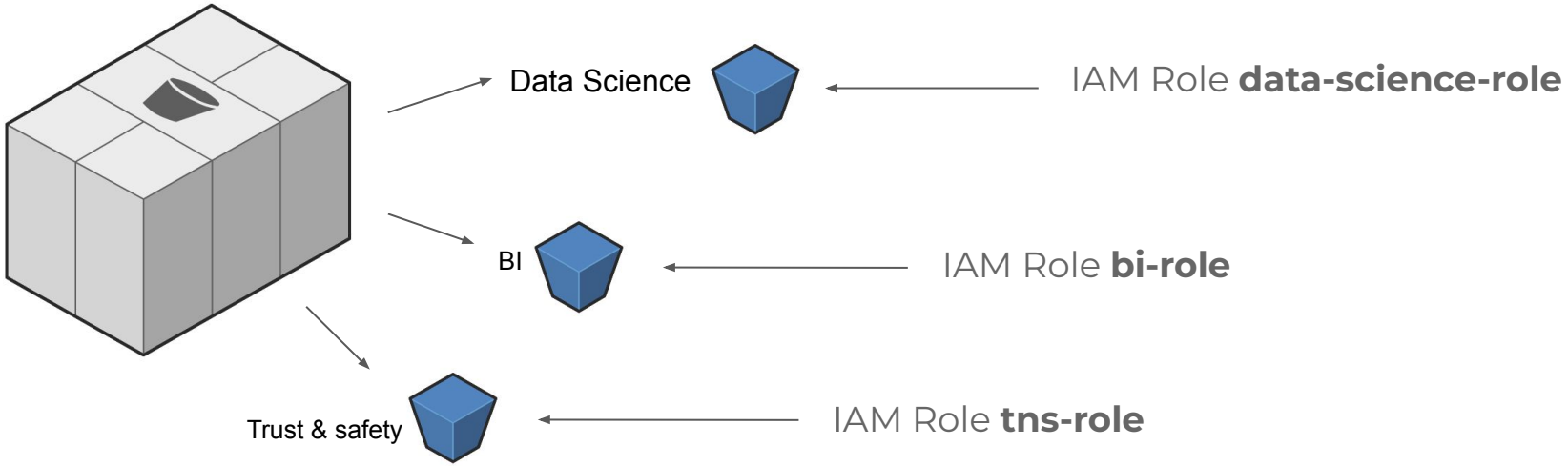


Data governance rules

- Rules defined for **fields**, **tags, tables** and/or **reservoirs** (in this order or priority)
- Possible actions for table columns:
  - **SHOW**: the column values are left as they are
  - **HASH**: the column values are anonymized through a hash function
  - **DROP**: the column is removed from the table
- Default action: DROP all fields tagged as PII
- UI to request access which must be approved by DPO

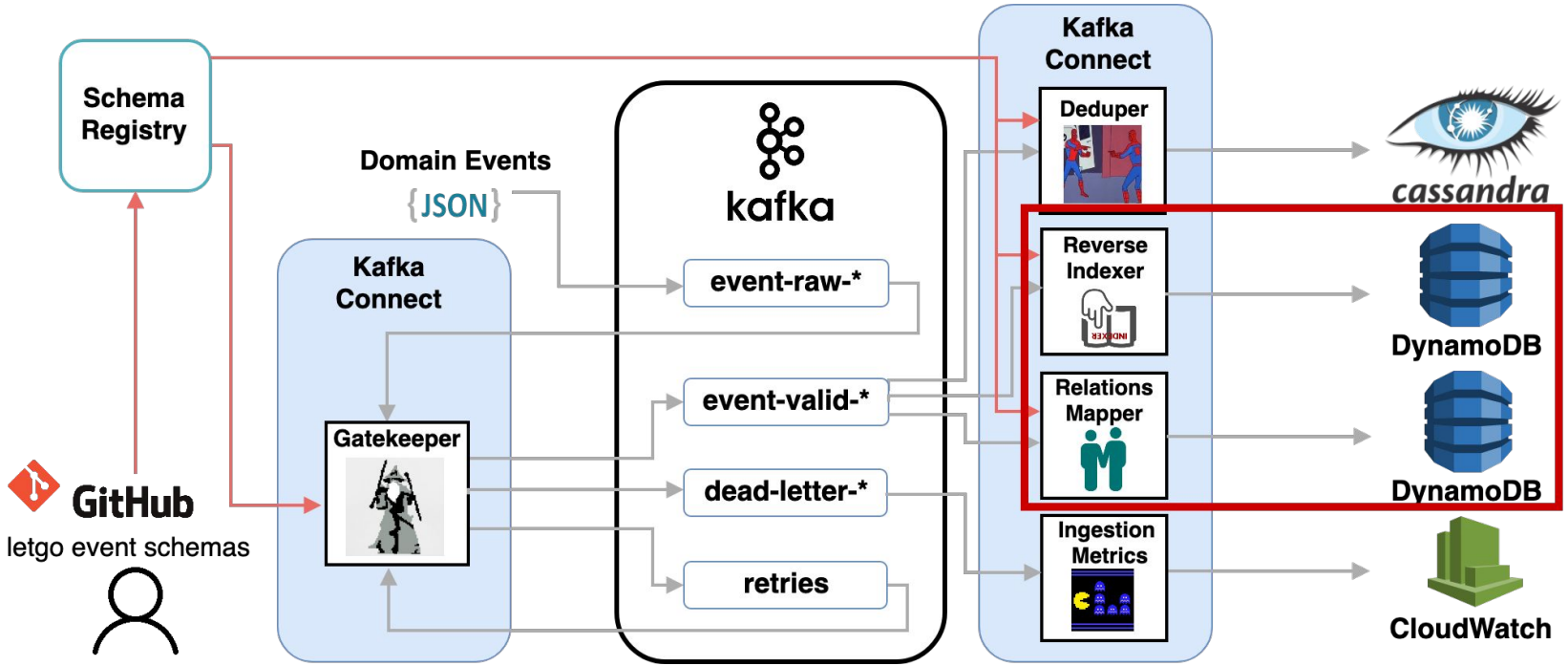
Data Access Policies

- Employees can only access their own anonymized reservoir
- IAM roles for authentication and authorization



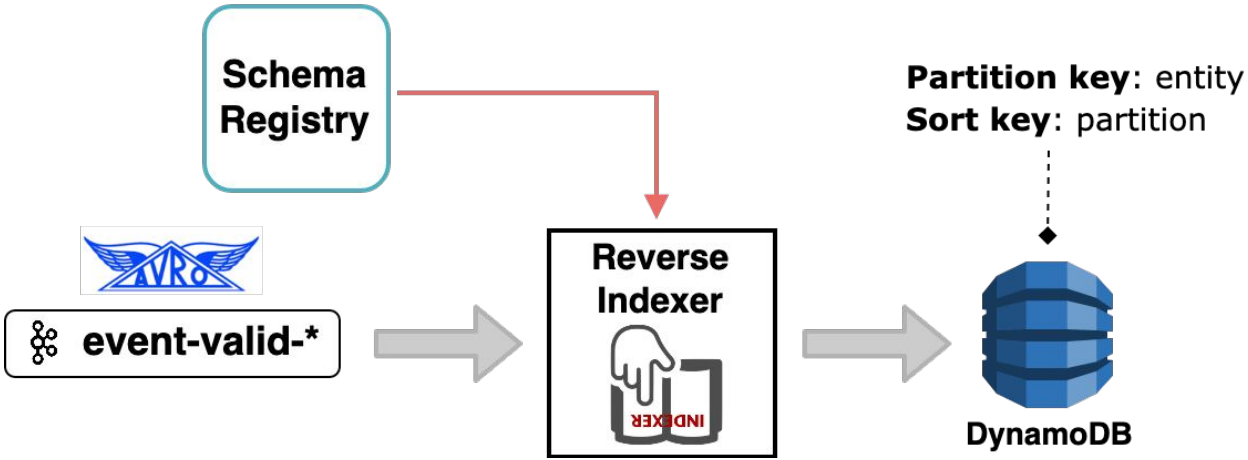
Access & Delete

Ingestion platform architecture (again)



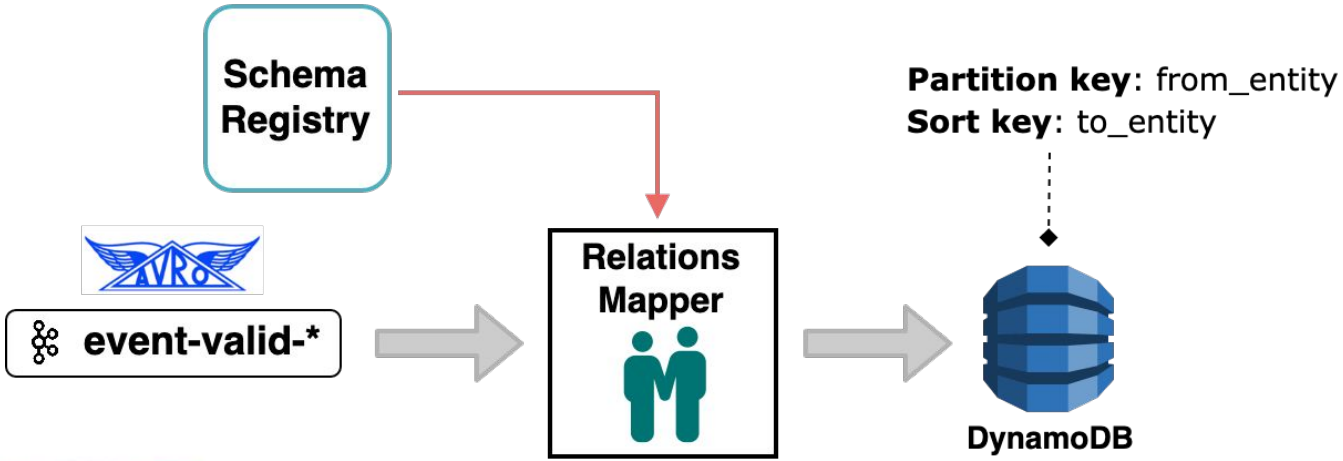
Reverse indexer

- Kafka connector sink that indexes partitions relevant to each entity
- Example: “lookup all Data Lake partitions containing events for user ‘*luke*’”



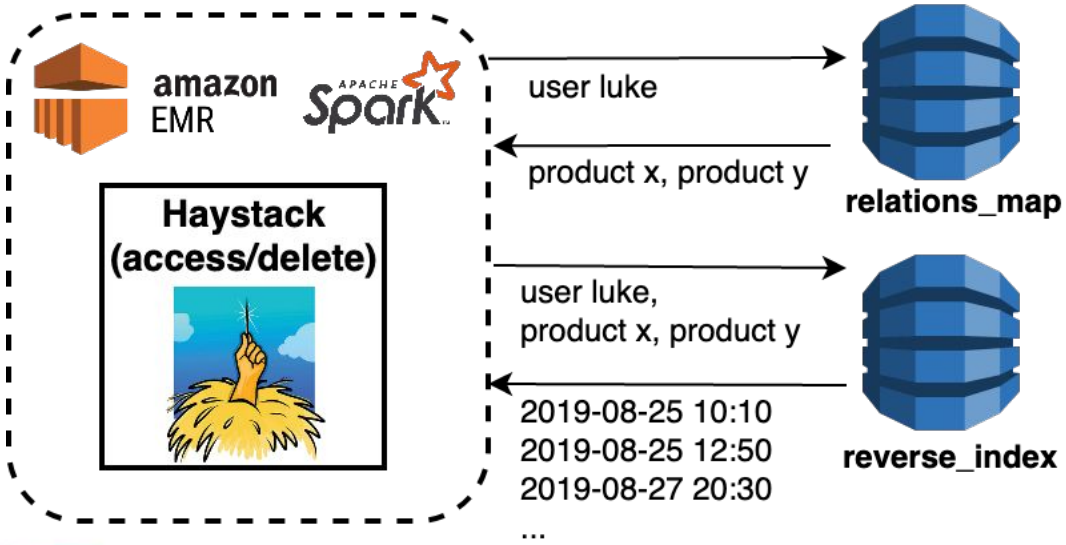
Relations mapper

- Kafka connector sink that stores relations between entities
- Allows to build a graph of relations among entities
- Example: “lookup all products, payments, etc. connected to user ‘*luke*’”



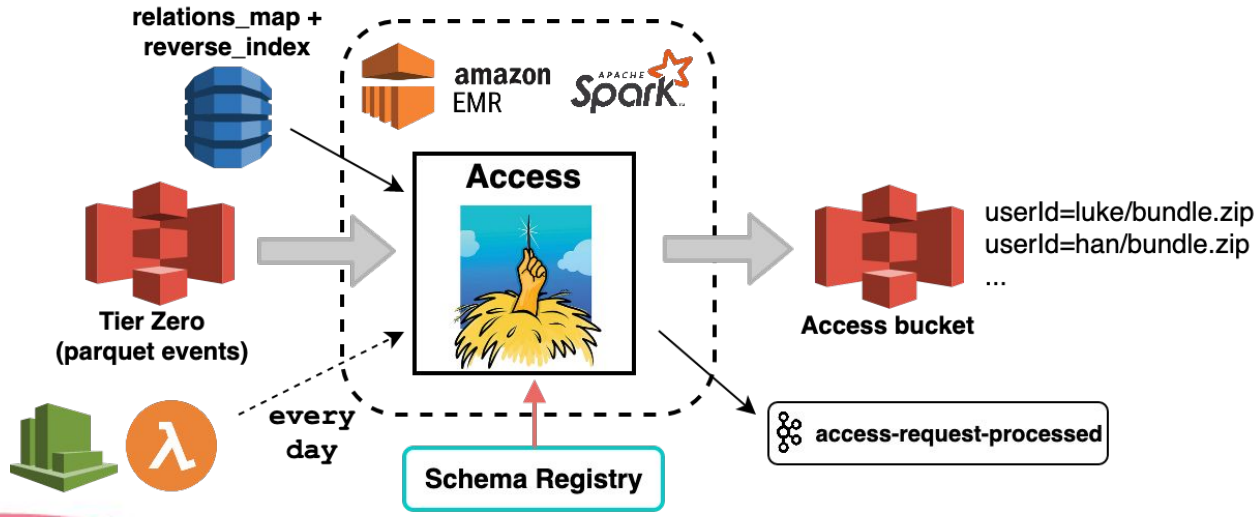
Identifying user's data in the Data Lake

- It's like finding the needle in the haystack...
- Avoid full scan with an **inverted index** and a **map of relations**



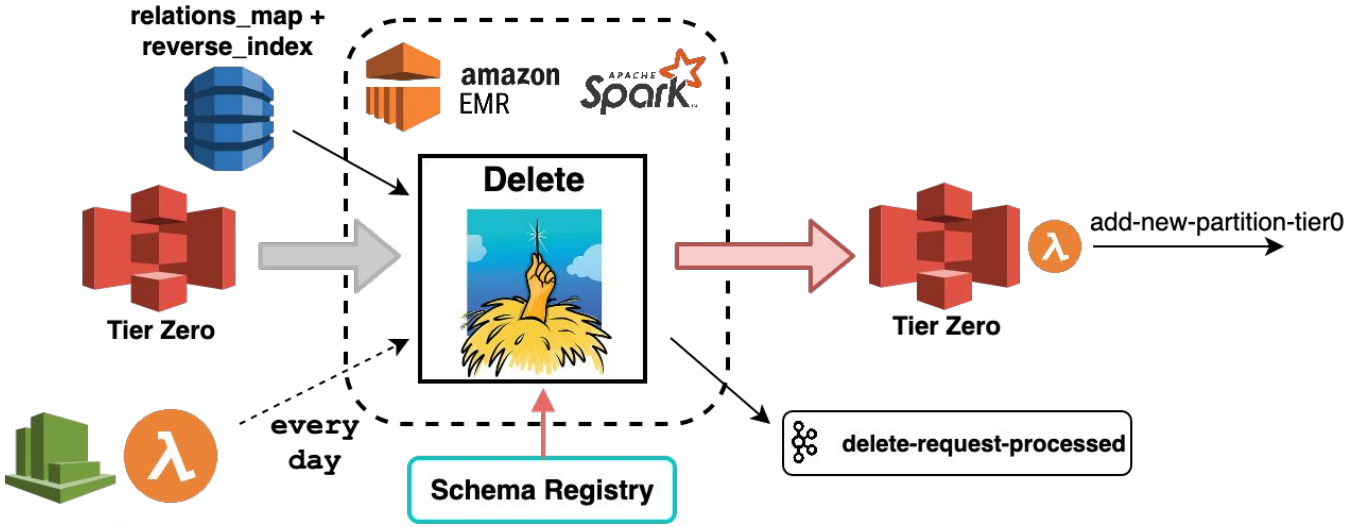
Access job

- Load relevant partitions for a batch of requests and save data of interest in a zip file for each one
- Ephemeral EMR cluster for ad-hoc computation



Delete job

- Recomputes relevant partitions removing/anonymizing data
- A copy is created in a separate bucket for law enforcement agencies



Questions?