A gentle introduction to Stream Processing

Nicolas Fränkel

@nicolas_frankel

Me, myself and I

- Developer
- Developer Advocate

Schedule

1. Why streaming?
2. The fun of Open Data
3. Demo!

In a time before our time…

Data was neatly stored in SQL databases

What SQL implies

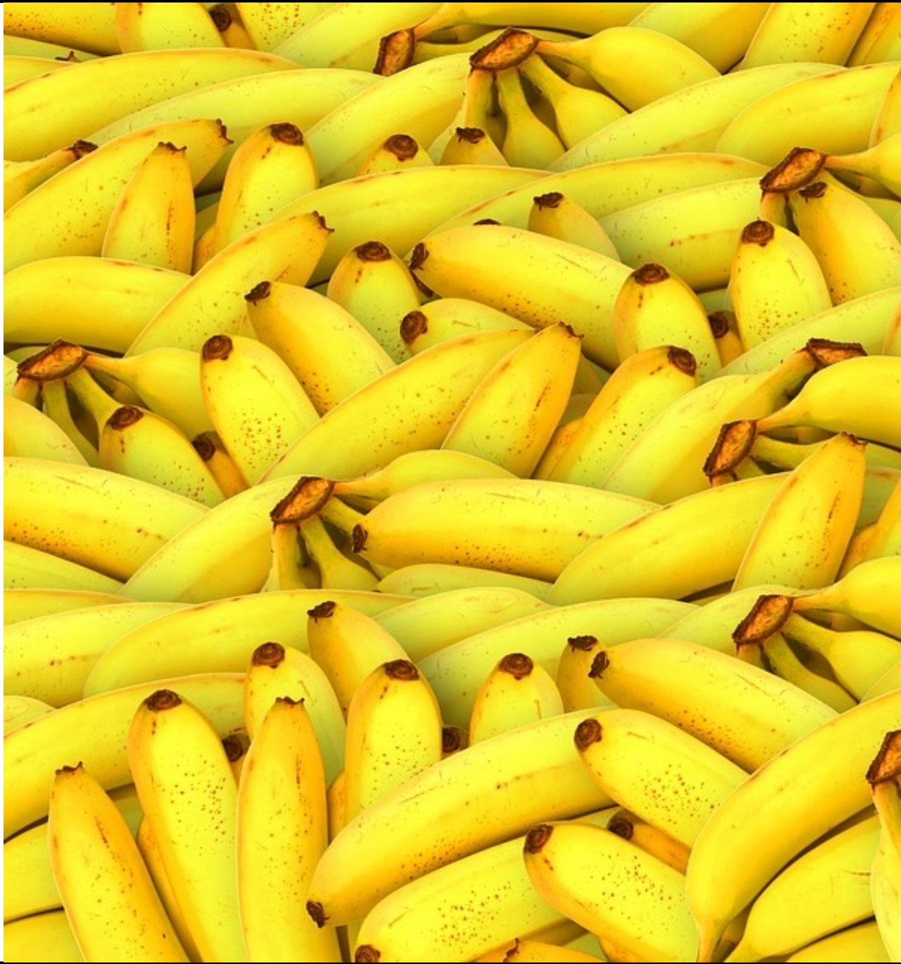
- Deduplication of data
  - Normal forms
  - Joins
- Data quality
  - Constraints

Writes vs. reads

- Normalized vs. denormalized
- Correct vs. fast

The need for Extract Transform Load

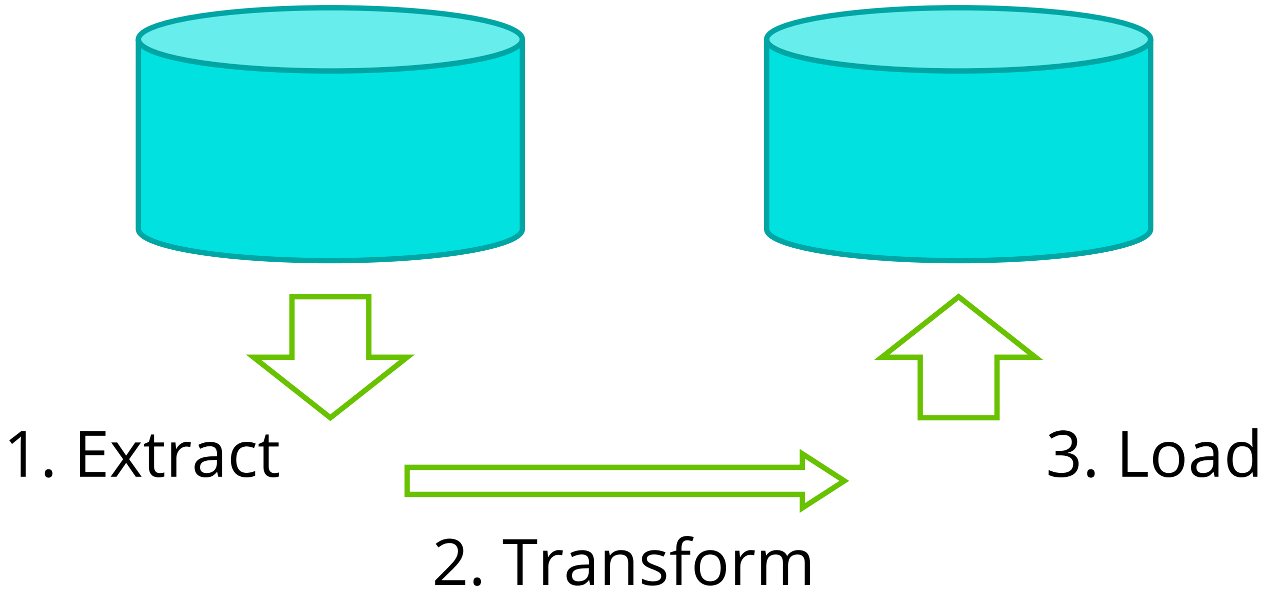
- Analytics
  - Supermarket sales in the last hour?
- Reporting
  - Banking account annual closing



The need for ETL

- Different actors
- With different needs
- Using the same database?

The batch model



Batches are everywhere!

Properties of batches

- Scheduled at regular intervals
  - Hourlys
  - Daily
  - Yearly
  - etc.
- Run in a specific amount of time

Oops

- When the execution time overlaps the next execution schedule
- When the batch fails mid-execution
- When the space taken by the data exceeds the storage capacity
- etc.

Chunking!

- Keep a cursor
  - And only manage “chunks” of data
- What about new data coming in?

Big data!

- Parallelize everything
  - Map - Reduce
  - Hadoop
- NoSQL
  - Schema on Read vs. Schema on Write

Event-Driven Programming

“In programming and software design, an event is **an action or occurrence** recognized by software, often originating asynchronously from the external environment, that may be handled by the software. Computer events can be generated or triggered by the system, by the user, or in other ways.”

-- *Wikipedia*

Make everything event-based!

Benefits

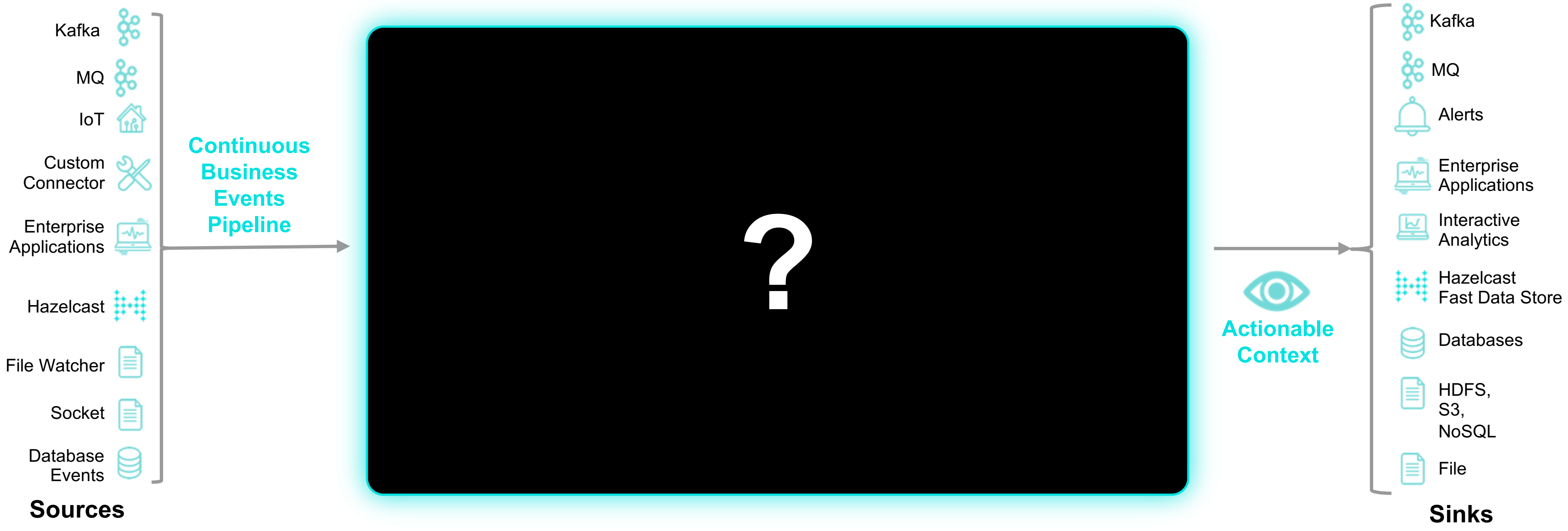
- Memory-friendly
- Easily processed
- Pull vs. push
  - Very close to real-time
  - Keeps derived data in-sync

From finite datasets to infinite

Stateful streams

- Aggregation
- Windowing

Data Sources and Sinks



Analytics and Decision Making

- Real-time dashboards
- Statistics
- Predictions
  - Push stream through ML model
- Complex-Event-Processing

“Event” storage systems

- Apache Kafka
- Apache Pulsar

Apache Kafka

- Distributed
- On-disk storage
- Messages sent and read from a topic
- Consumer can keep track of the offset

In-memory stream processing engines

- On-premise
  - Apache Flink
  - Hazelcast Jet
- Cloud-based
  - Amazon Kinesis
  - Google Dataflow
- Apache Beam
  - Abstraction over some of the above

Hazelcast Platform

- Apache 2 Open Source
- Unified batch/streaming API
- Hazelcast Jet Enterprise offering

Pipeline

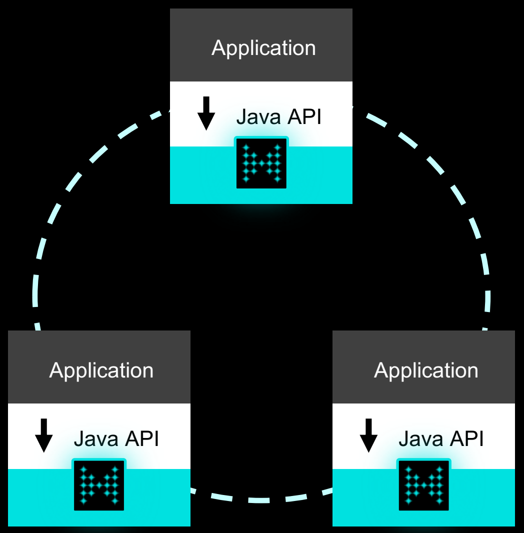
- Declarative code that defines and links sources, transforms, and sinks
- Platform-specific SDK
- Client submits pipeline to the SPE

Job

- Running instance of pipeline in SPE
- SPE executes the pipeline
  - Code execution
  - Data routing
  - Flow control

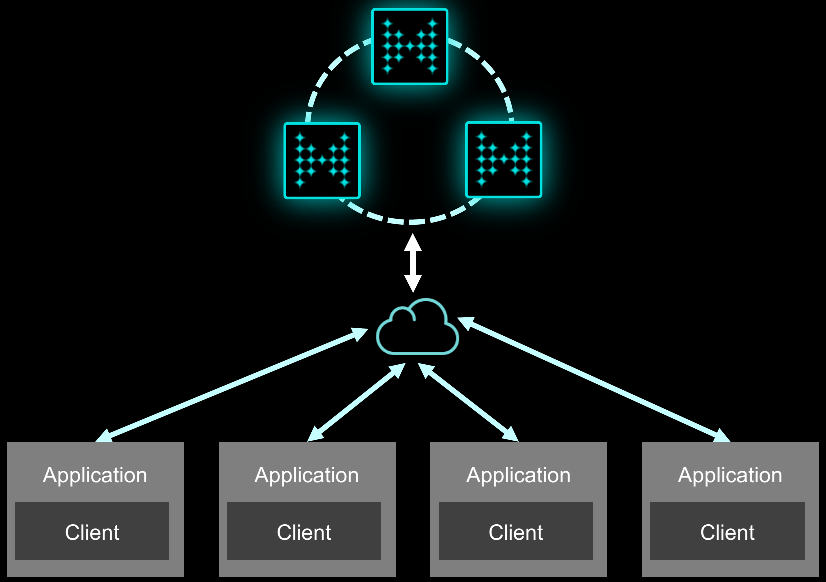
Hazelcast Deployment Options

Embedded Mode



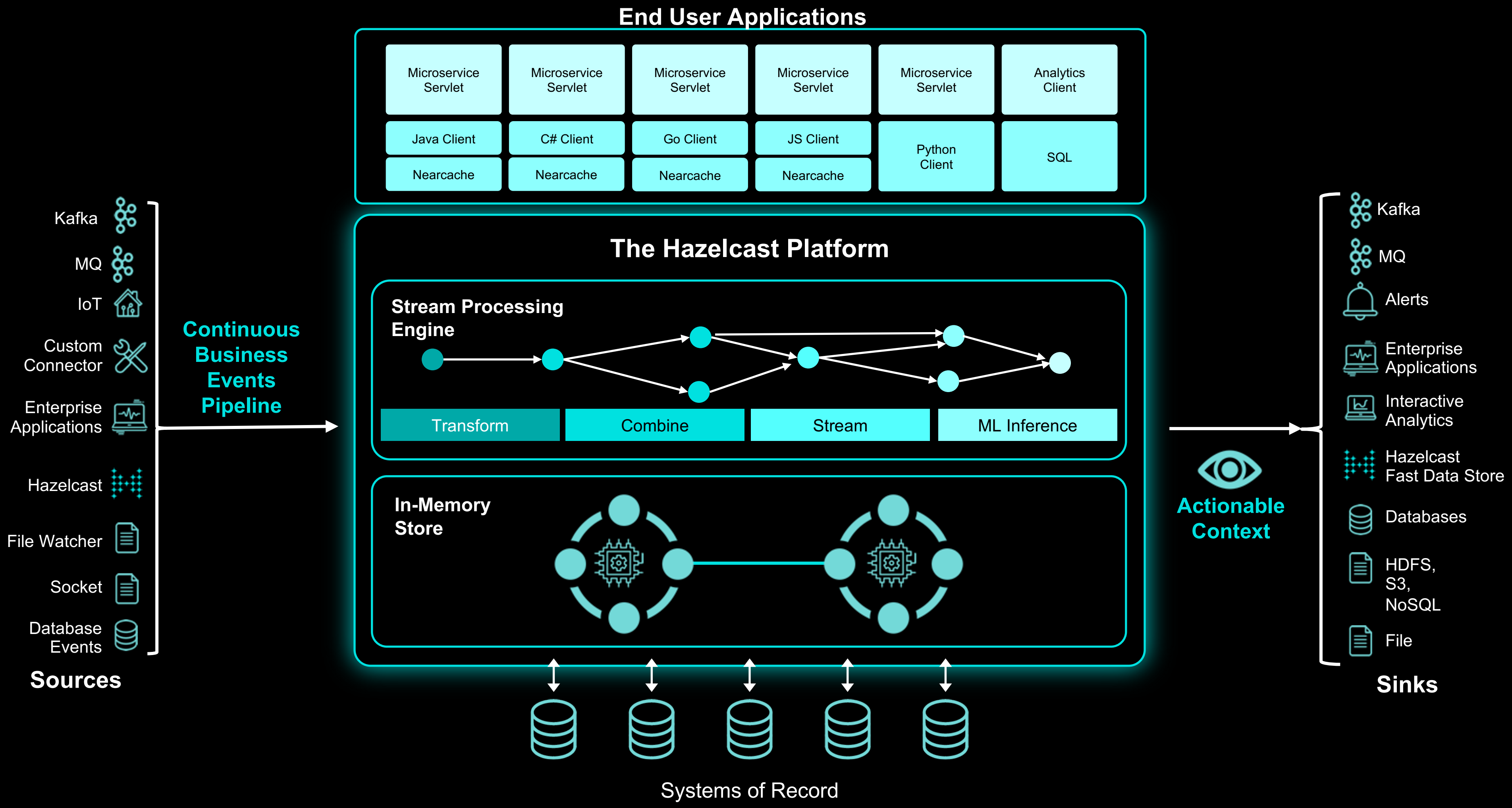
Great for microservices,
OEM and ops simplification

Client-Server Mode



Great for scale-up or scale-out deployments with cluster lifecycle decoupled from app servers

Clients available in Java (Scala, Clojure, etc.), Node.js, C#/.NET, C++, Python, and Go



HAZELCAST

@nicolas_frankel

Open Data

« **Open data** is the idea that some data should be freely available to everyone to use and republish as they wish, without restrictions from copyright, patents or other mechanisms of control. »

--https://en.wikipedia.org/wiki/Open_data

Some Open Data initiatives

- France:
  - https://www.data.gouv.fr/fr/
- Switzerland:
  - https://opendata.swiss/en/
- European Union:
  - https://data.europa.eu/euodp/en/data/

Challenges

1. Access
2. Format
3. Standard
4. Data correctness

Access

- Access data interactively through a web-service
- Download a file

Format

Does Open Data mean Open Format?

- PDF
- CSV
- XML
- JSON
- etc.

Standard

- Let’s pretend the format is XML
  - Which grammar is used?
- A shared standard is required
  - Congruent to a domain

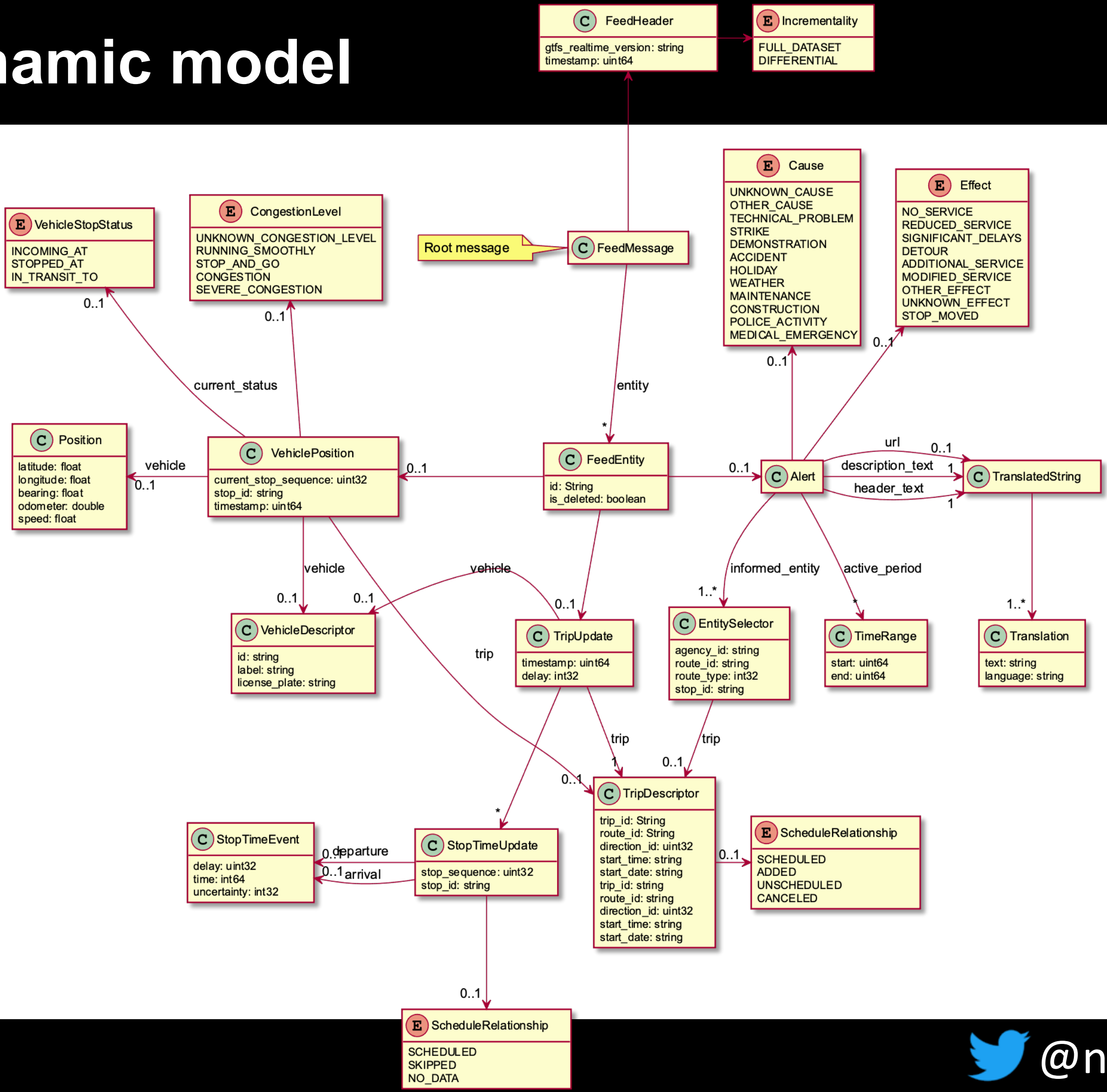
Data correctness

```
"32.TA.66-43","16:20:00","16:20:00","8504304"
"32.TA.66-44","24:53:00","24:53:00","8500100"
"32.TA.66-44","25:00:00","25:00:00","8500162"
"32.TA.66-44","25:02:00","25:02:00","8500170"
"32.TA.66-45","23:32:00","23:32:00","8500170"
```

A standard for Public Transport

- General Transit Feed Specification (GTFS)
- " […] defines a **common format for public transportation schedules and associated geographic information**. GTFS feeds let public transit agencies publish their transit data and developers write applications that consume that data in an interoperable way."
- Based on two kinds of data:
  - "**Static**" *e.g.* stops
  - **Dynamic** *e.g.* position

GTFS dynamic model



A data provider

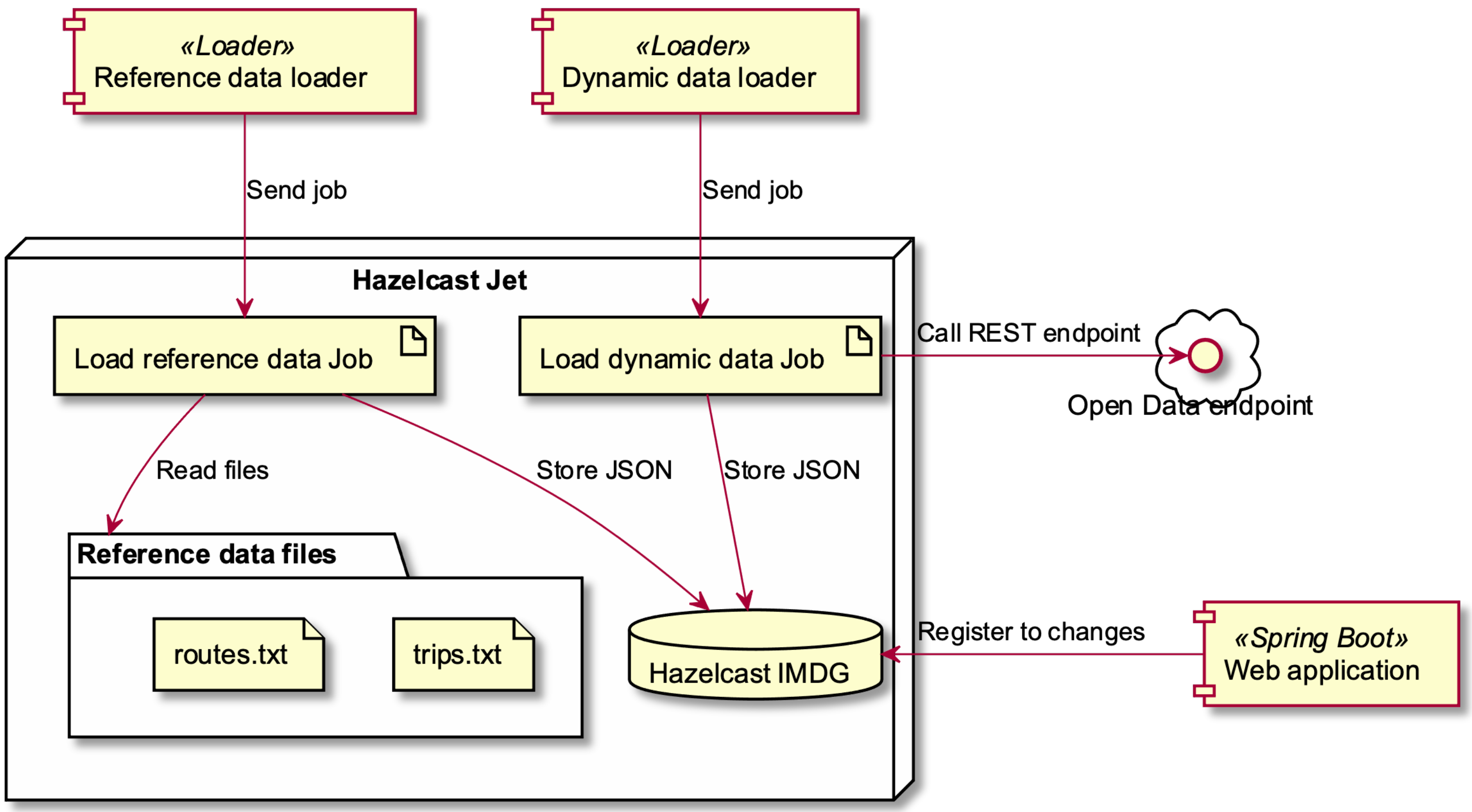
"511 is your phone and web source for Bay Area traffic, transit, carpool, vanpool, and bicycling information. It's FREE and available whenever you need it – 24 hours a day, 7 days a week – from anywhere in the nine-county Bay Area"

-- https://511.org/open-data

The dynamic data pipeline

1. Source: web service
2. Split into trip updates
3. Transform to JSON
4. Filter out malformed data
5. Enrich with stop times, trip, routes and stops data
6. Transform hours into timestamp
7. “Flatten” JSON
8. Peek sample (for debugging purpose)
9. Transform into map entry
10. Sink: Hazelcast IMDG map

Architecture overview



Talk is cheap, show me the code!

Recap

- Streaming has a lot of benefits
- Open Data has a lot of untapped potential
- Get cool stuff done!

Thanks a lot!

- https://blog.frankel.ch/
- @nicolas_frankel
- https://jet-start.sh/
- https://bit.ly/jet-train
- https://slack.hazelcast.com/
- https://training.hazelcast.com/