Turning the wheel - Streaming 4.4 billion events with Apache Kafka

Streaming customer, policy and vehicle information via Apache Kafka and MongoDB – and what we learnt on the way …

Simon Aubury

April 2019

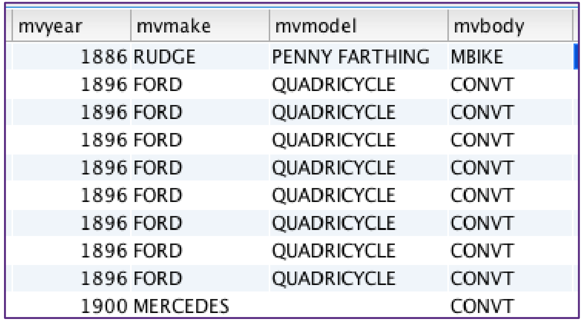
Agenda

Tonight

- Context - what problem are we trying to solve
- Architecture of our data flow
- Kafka & Kafka Connect
- Challenges .. and solutions

We’ve a lot of data

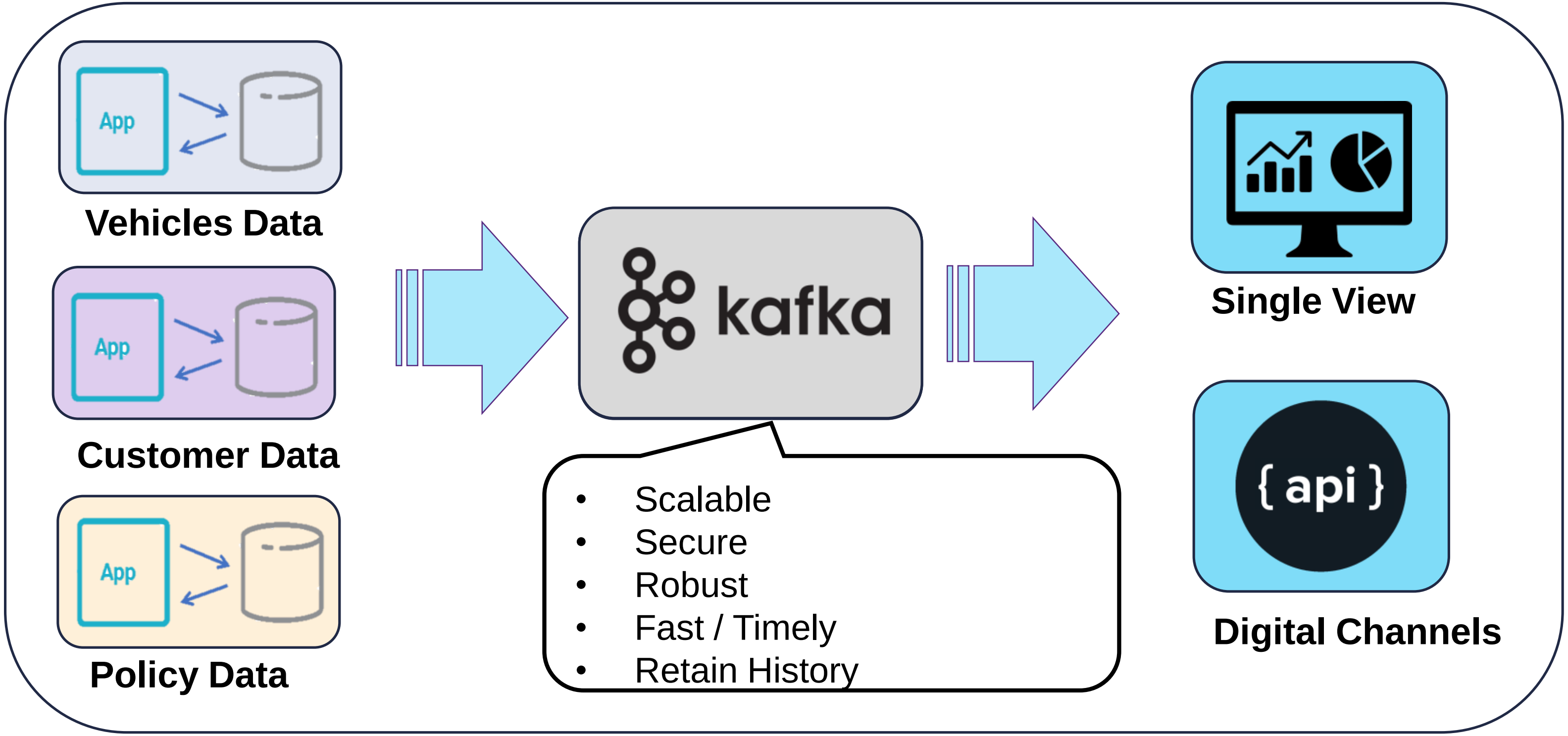
Context #1



mvyear	mvmake	mvmodel	mvbody
1886	RUDGE	PENNY FARTHING	MBIKE
1896	FORD	QUADRICYCLE	CONVT
1896	FORD	QUADRICYCLE	CONVT
1896	FORD	QUADRICYCLE	CONVT
1896	FORD	QUADRICYCLE	CONVT
1896	FORD	QUADRICYCLE	CONVT
1896	FORD	QUADRICYCLE	CONVT
1896	FORD	QUADRICYCLE	CONVT
1896	FORD	QUADRICYCLE	CONVT
1900	MERCEDES		CONVT

We’ve a lot of systems

Context #2



Vehicles Data

Customer Data

Policy Data

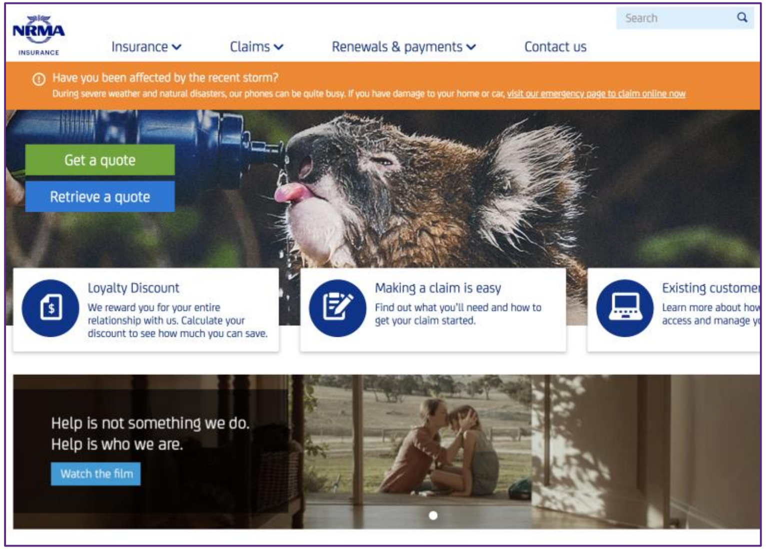
- Scalable
- Secure
- Robust
- Fast / Timely
- Retain History

Single View

Digital Channels

We want to tie it together

Context #3



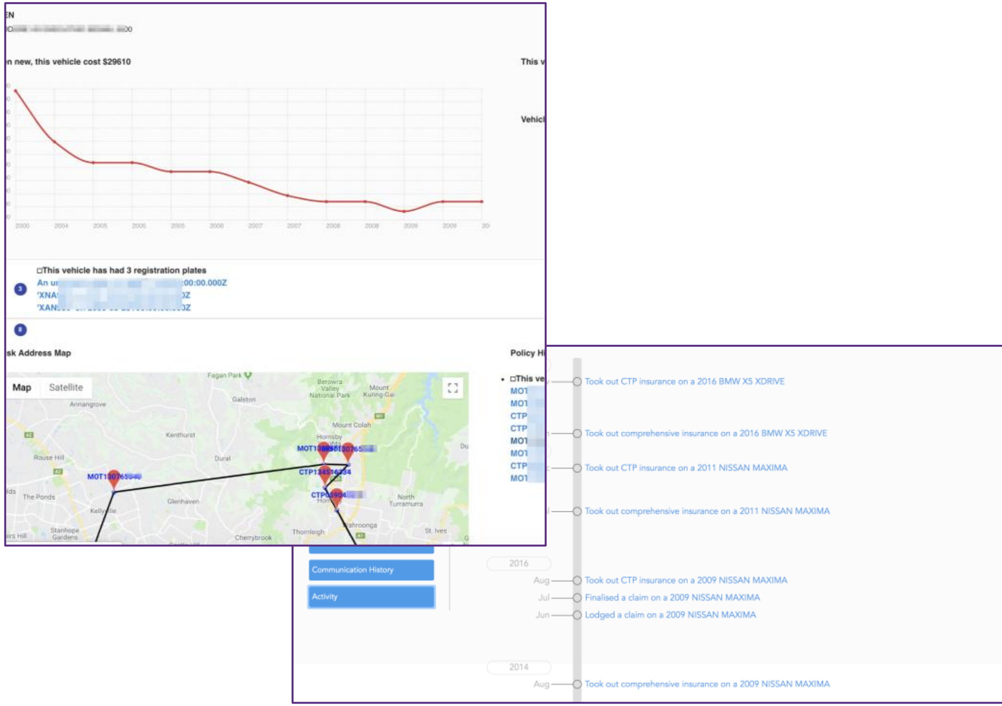
Have you been affected by the recent storm?
During severe weather and natural disasters, our phones can be quite busy. If you have damage to your home or car, visit our emergency page to claim online now

Loyalty Discount
We reward you for your entire relationship with us. Calculate your discount to see how much you can save.

Making a claim is easy
Find out what you'll need and how to get your claim started.

Existing custome
Learn more about hov
access and manage y

Help is not something we do.
Help is who we are.



EN
O[redacted]00

n new, this vehicle cost $29610

This v

Vehicl

□This vehicle has had 3 registration plates
An ur[redacted]:00:00.000Z
'XNA[redacted]Z
'XAN[redacted]Z

sk Address Map

Policy Hi

- □This ve
MOT[redacted]
MOT[redacted]
CTP[redacted]
CTP[redacted]
MOT[redacted]
MOT[redacted]
CTP[redacted]
MOT[redacted]

Communication History
Activity

Took out CTP insurance on a 2016 BMW X5 XDRIVE
Took out comprehensive insurance on a 2016 BMW X5 XDRIVE
Took out CTP insurance on a 2011 NISSAN MAXIMA
Took out comprehensive insurance on a 2011 NISSAN MAXIMA

2016
Aug Took out CTP insurance on a 2009 NISSAN MAXIMA
Jul Finalised a claim on a 2009 NISSAN MAXIMA
Jun Lodged a claim on a 2009 NISSAN MAXIMA

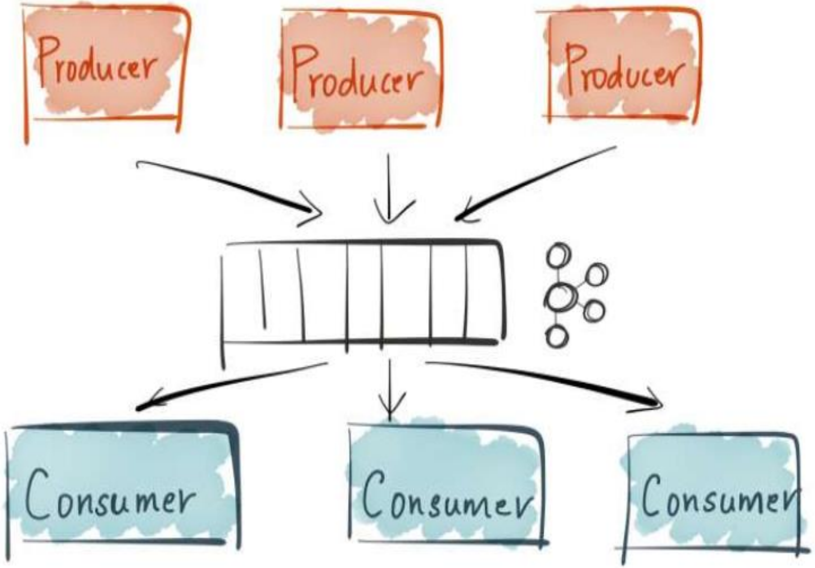
2014
Aug Took out comprehensive insurance on a 2009 NISSAN MAXIMA

What is Kafka?

A very quick into

Apache Kafka : *Unified, high-throughput, low-latency platform for handling real-time data feeds*

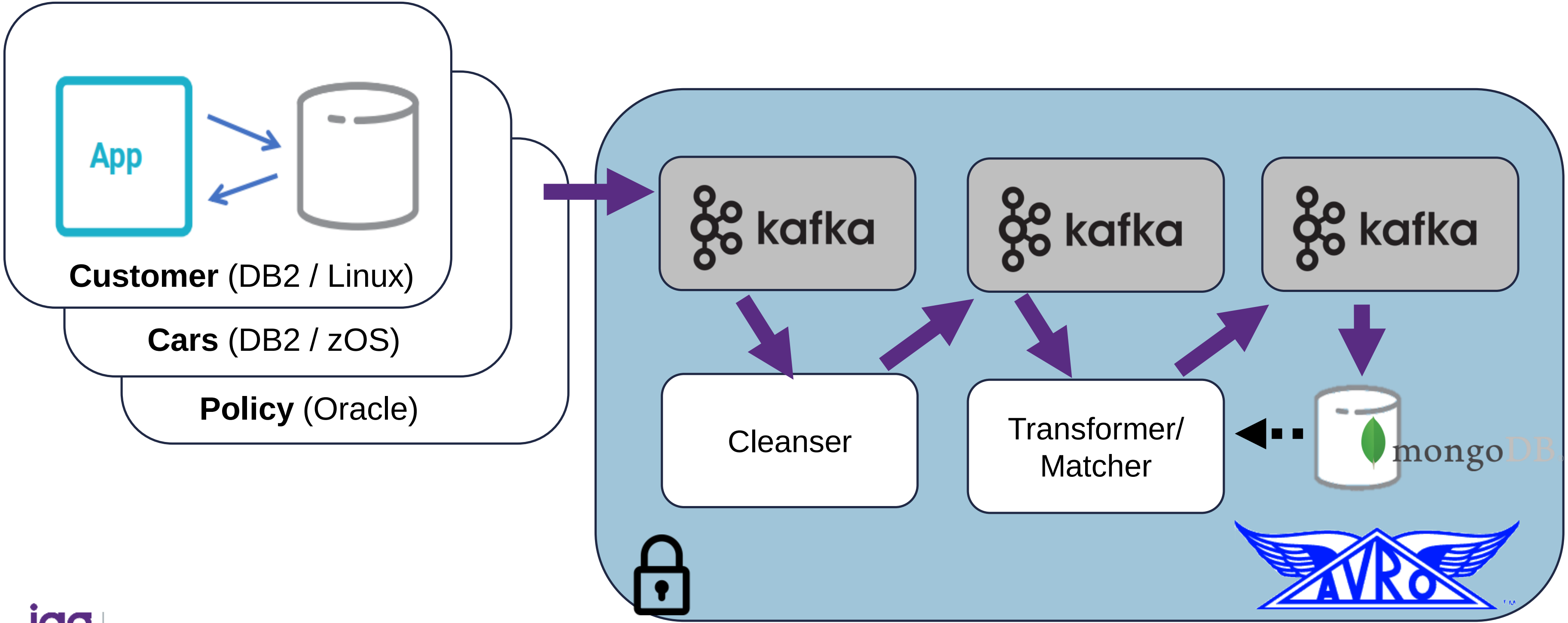
- Originally developed by LinkedIn, open sourced in early 2011
- “The global commit log thingy”
- Kafka maintains feeds of messages in topics
- Appends ; ordered, immutable sequence



Credit: https://www.confluent.io/blog/stream-data-platform-1/

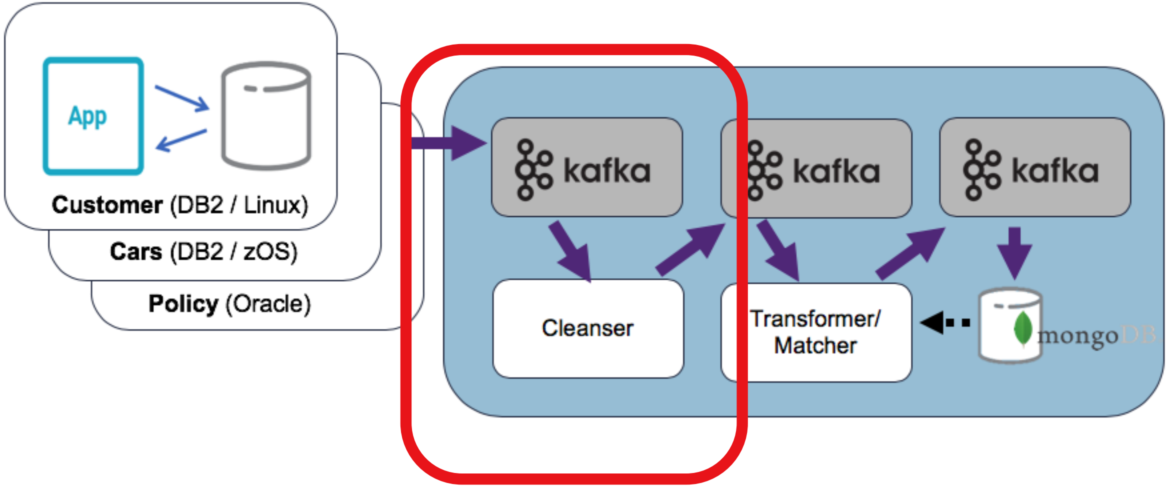
Architecture of our data flow

Lots of boxes



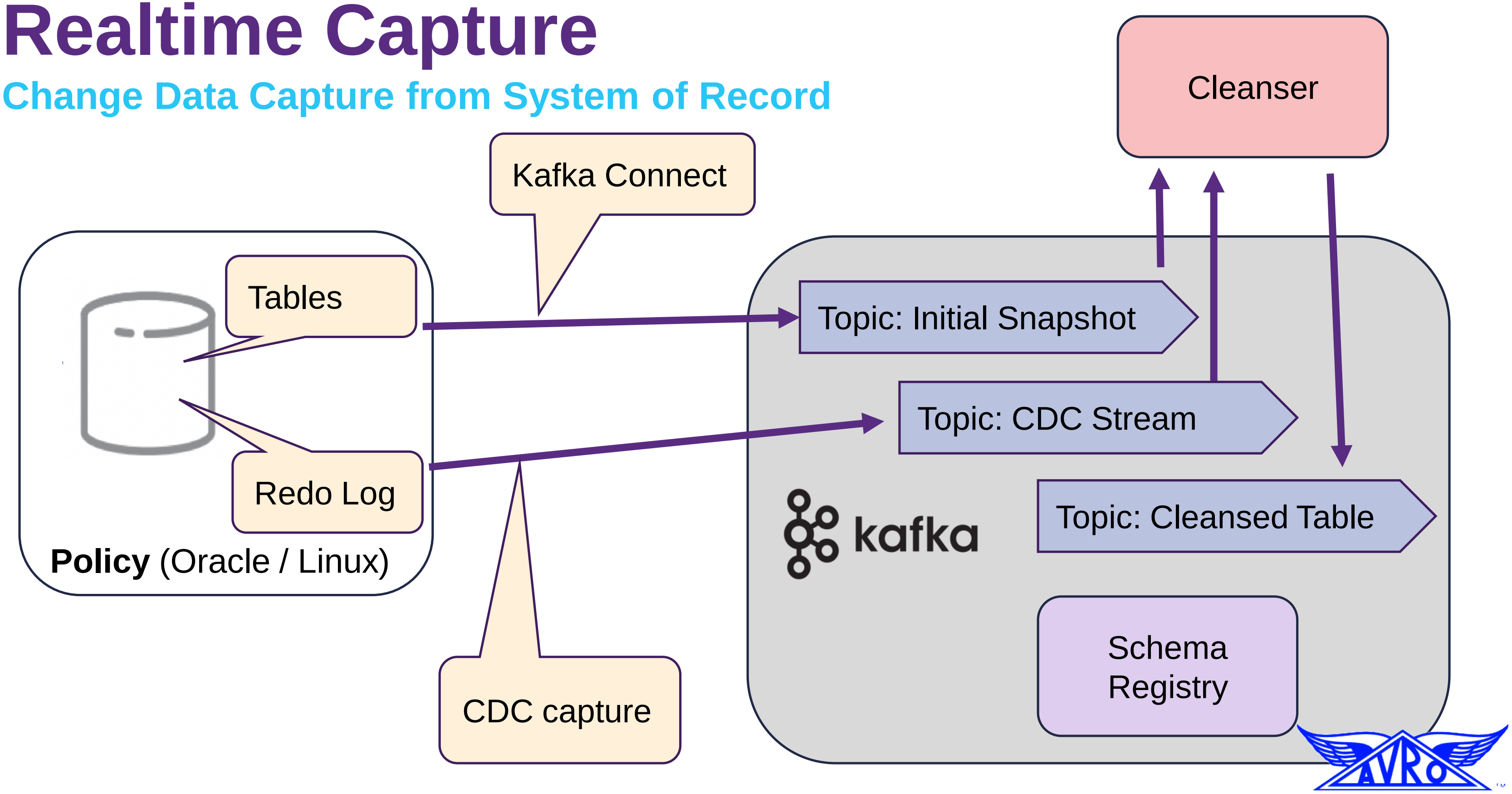
Part 1 - Extract

Source System Low Touch Data Acquisition



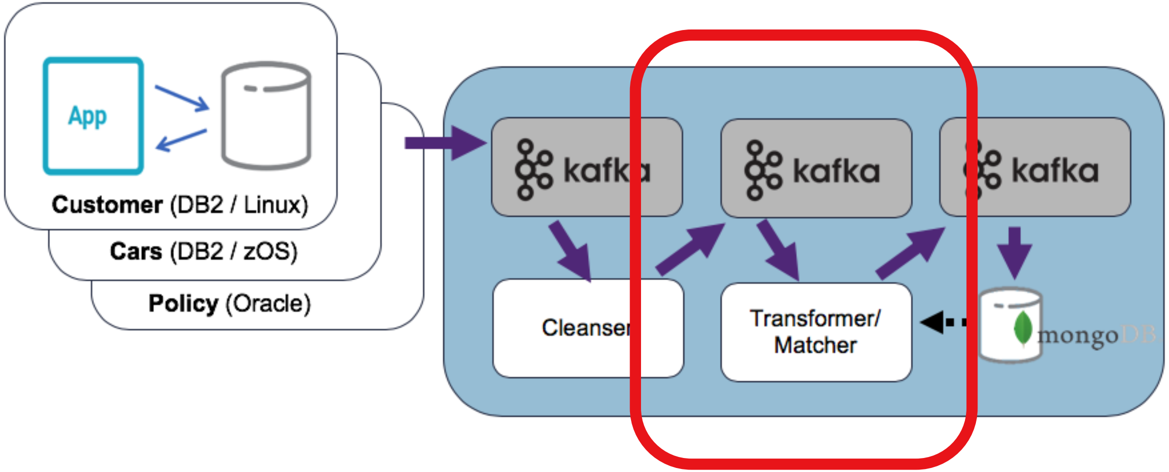
Realtime Capture

Change Data Capture from System of Record

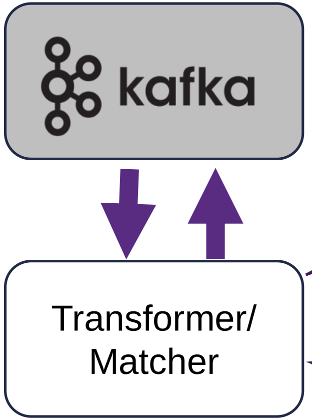


Part 2 – Transformation & Matching

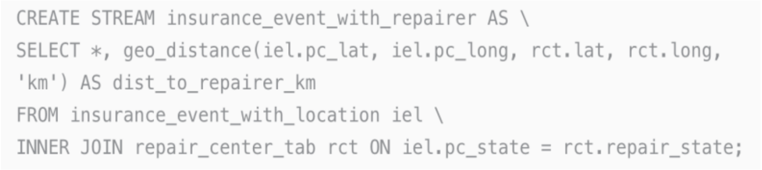
Finding stuff



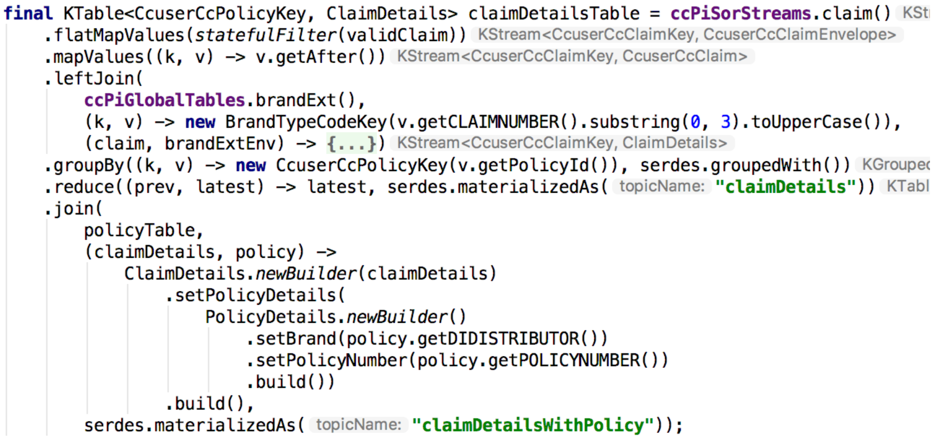
Transform & Match



Transformer/ Matcher



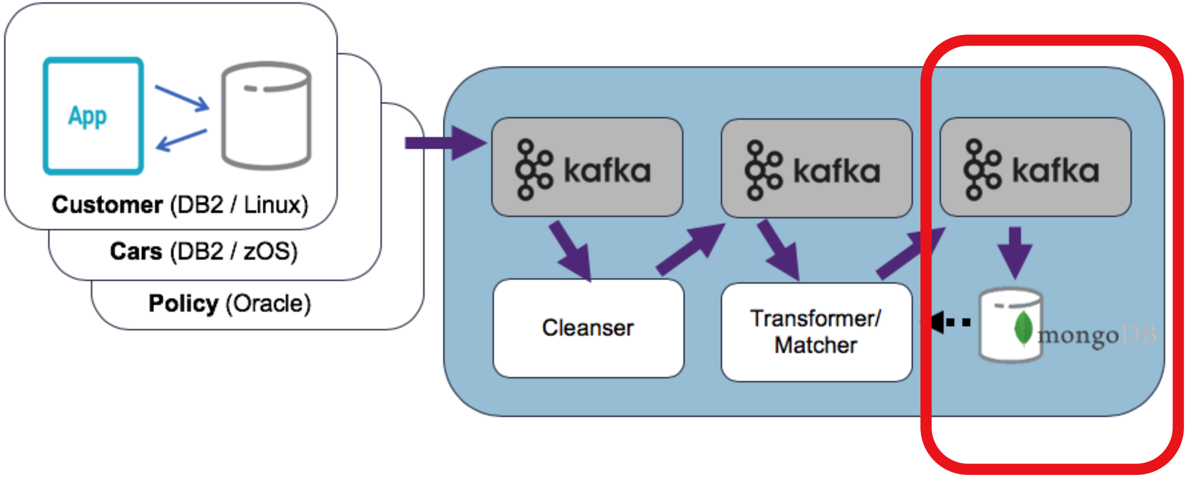
```
CREATE STREAM insurance_event_with_repairer AS \
SELECT *, geo_distance(iel.pc_lat, iel.pc_long, rct.lat, rct.long,
'km') AS dist_to_repairer_km
FROM insurance_event_with_location iel \
INNER JOIN repair_center_tab rct ON iel.pc_state = rct.repair_state;
```



```
final KTable<CcuserCcPolicyKey, ClaimDetails> claimDetailsTable = ccPiSorStreams.claim() KSt
    .flatMapValues(statefulFilter(validClaim)) KStream<CcuserCcClaimKey, CcuserCcClaimEnvelope>
    .mapValues((k, v) -> v.getAfter()) KStream<CcuserCcClaimKey, CcuserCcClaim>
    .leftJoin(
        ccPiGlobalTables.brandExt(),
        (k, v) -> new BrandTypeCodeKey(v.getCLAIMNUMBER().substring(0, 3).toUpperCase()),
        (claim, brandExtEnv) -> {...}) KStream<CcuserCcClaimKey, ClaimDetails>
    .groupBy((k, v) -> new CcuserCcPolicyKey(v.getPolicyId()), serdes.groupedWith()) KGroupe
    .reduce((prev, latest) -> latest, serdes.materializedAs( topicName: "claimDetails")) KTabl
    .join(
        policyTable,
        (claimDetails, policy) ->
            ClaimDetails.newBuilder(claimDetails)
                .setPolicyDetails(
                    PolicyDetails.newBuilder()
                        .setBrand(policy.getDIDISTRIBUTOR())
                        .setPolicyNumber(policy.getPOLICYNUMBER())
                        .build())
                .build(),
        serdes.materializedAs( topicName: "claimDetailsWithPolicy"));
```

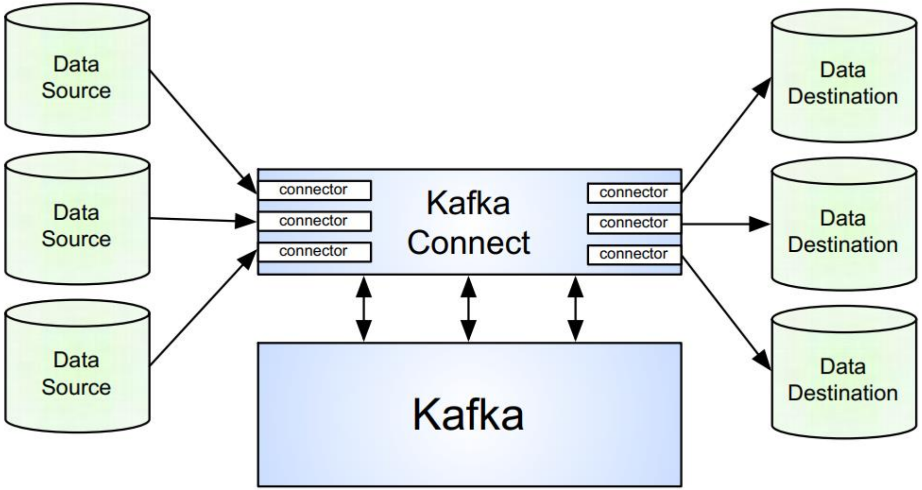
Part 3 – Serving Layer

Sinking Results



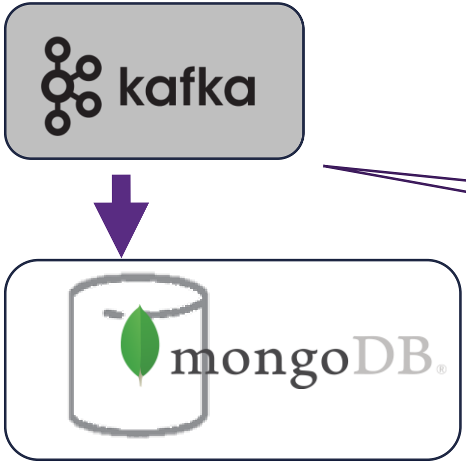
Kafka Connect

- Distributed, scalable, fault-tolerant service designed to reliably stream data between Kafka and other data systems
- **Source Connectors** import data from another system (e.g. a relational database into Kafka)
- **Sink Connectors** export data (e.g. the contents of a Kafka topic to an HDFS file).



Kafka Connect Sink

Writing to MongoDB Serving Layer



Kafka Connect

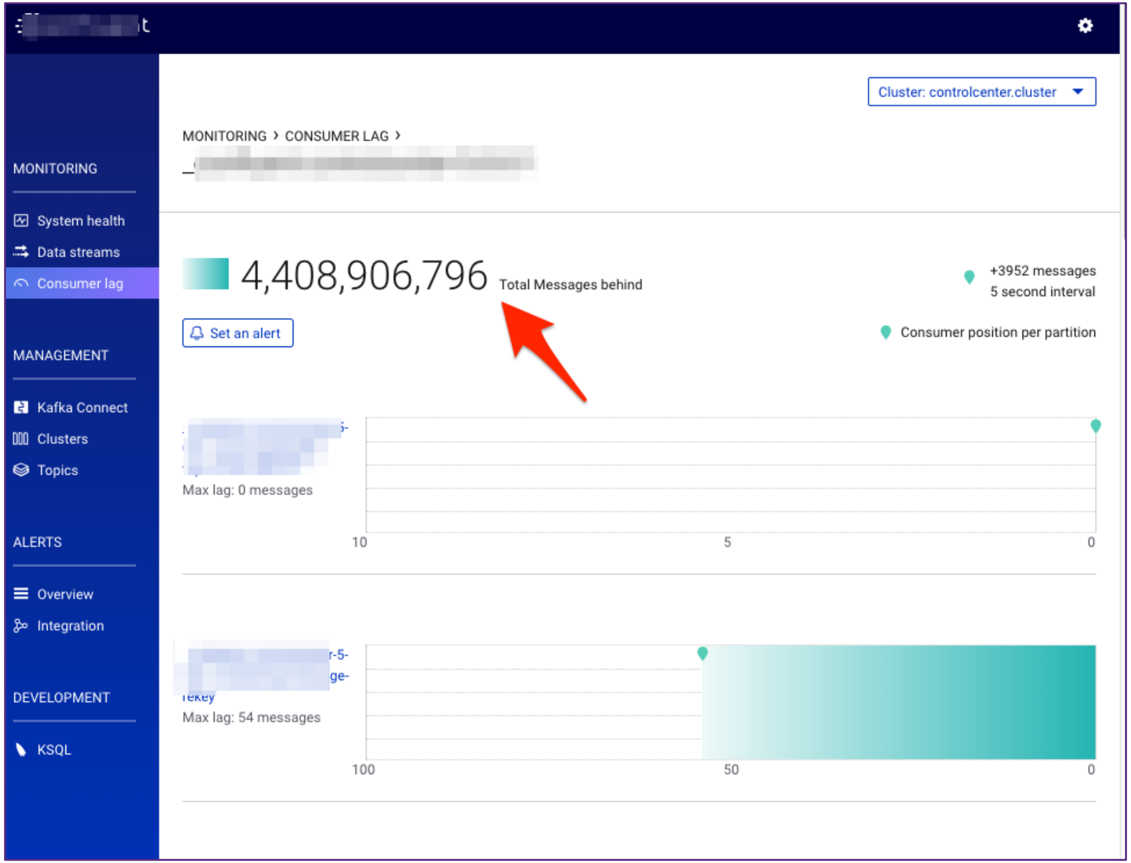
```
{
   "connector.class":
         "at.grahsl.kafka.connect.mongodb.MongoDbSinkConnector",
   "topics": ”sva-prod",
   "mongodb.connection.uri": "mongodb://sva-prod.dataeng.internal:27017/",
   "mongodb.collection": ”sva-vehicle
}
```

What did we discover?

Slow to fast … to *really* fast!

Challenge

Lots of data



Cluster: controlcenter.cluster

MONITORING › CONSUMER LAG ›

[redacted]

4,408,906,796 Total Messages behind

+3952 messages
5 second interval

Consumer position per partition

[redacted]

Max lag: 0 messages

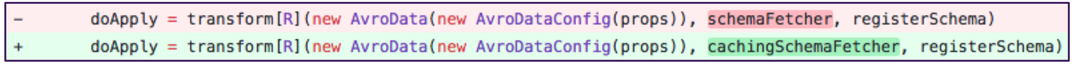
[redacted]

Max lag: 54 messages

Hot code #1

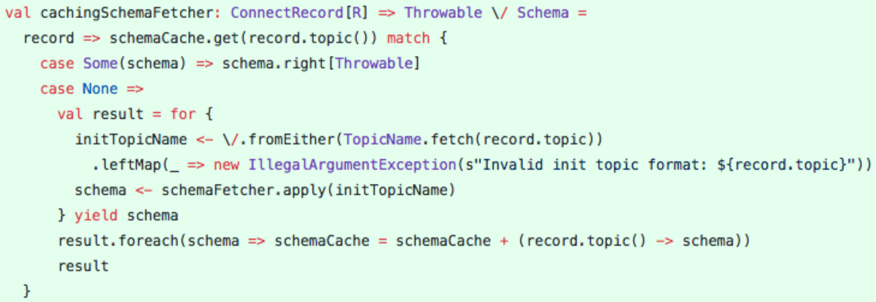
A bit of caching

Before: very slow transform
30 records / sec / table



```
-       doApply = transform[R](new AvroData(new AvroDataConfig(props)), schemaFetcher, registerSchema)
+       doApply = transform[R](new AvroData(new AvroDataConfig(props)), cachingSchemaFetcher, registerSchema)
```

After: add cache for schema lookup
200 records / sec / table

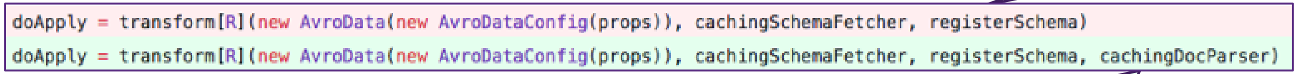


```
val cachingSchemaFetcher: ConnectRecord[R] => Throwable \/ Schema =
  record => schemaCache.get(record.topic()) match {
    case Some(schema) => schema.right[Throwable]
    case None =>
      val result = for {
        initTopicName <- \/.fromEither(TopicName.fetch(record.topic))
          .leftMap(_ => new IllegalArgumentException(s"Invalid init topic format: ${record.topic}"))
        schema <- schemaFetcher.apply(initTopicName)
      } yield schema
      result.foreach(schema => schemaCache = schemaCache + (record.topic() -> schema))
      result
  }
```

Hot code #2

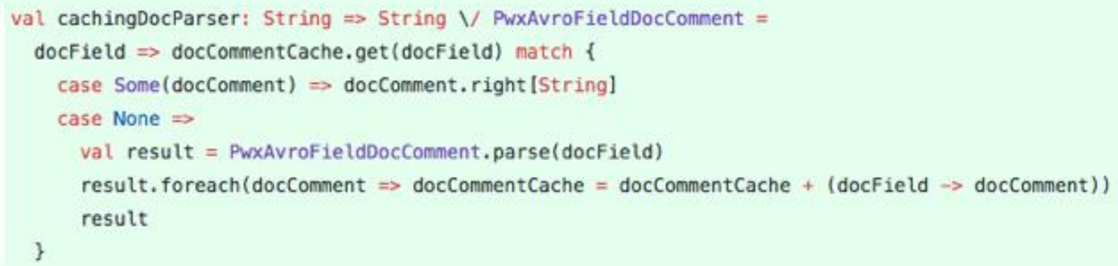
A bit more caching

Before: still slow transform
200 records / sec / table



```
doApply = transform[R](new AvroData(new AvroDataConfig(props)), cachingSchemaFetcher, registerSchema)
doApply = transform[R](new AvroData(new AvroDataConfig(props)), cachingSchemaFetcher, registerSchema, cachingDocParser)
```

After: add cache for field metadata
5,500 records / sec / table



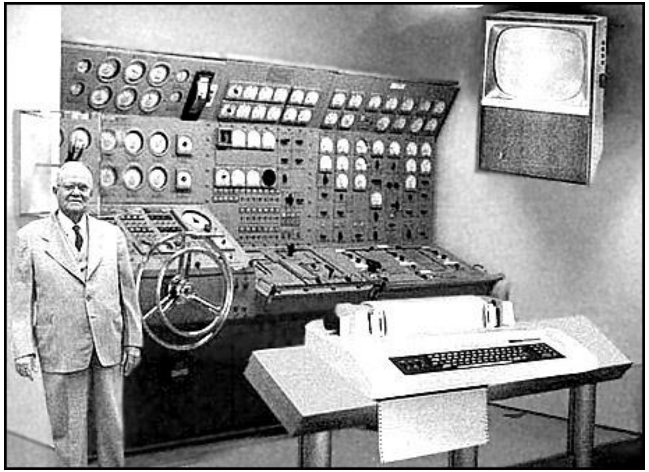
```
val cachingDocParser: String => String \/ PwxAvroFieldDocComment =
  docField => docCommentCache.get(docField) match {
    case Some(docComment) => docComment.right[String]
    case None =>
      val result = PwxAvroFieldDocComment.parse(docField)
      result.foreach(docComment => docCommentCache = docCommentCache + (docField -> docComment))
      result
  }
```

Horizontal scaling?

Theory

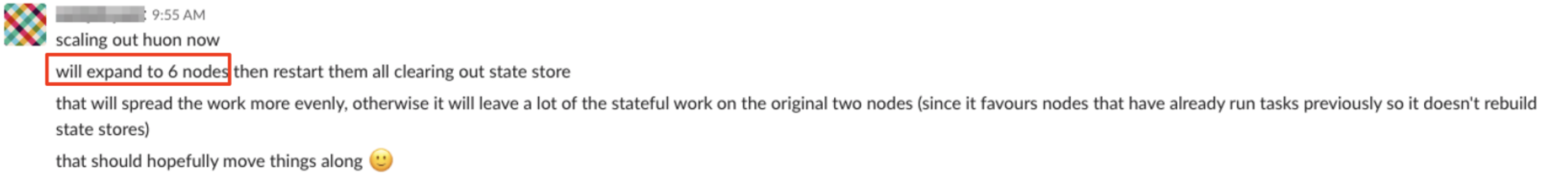
To scale out, you simply start another instance of your stream processing application, e.g. on another machine. The instances of your application will become aware of each other and automatically begin to share the processing work.

https://www.confluent.io/blog/elastic-scaling-in-kafka-streams/



Horizontal scaling … scales horizontally!

Testing



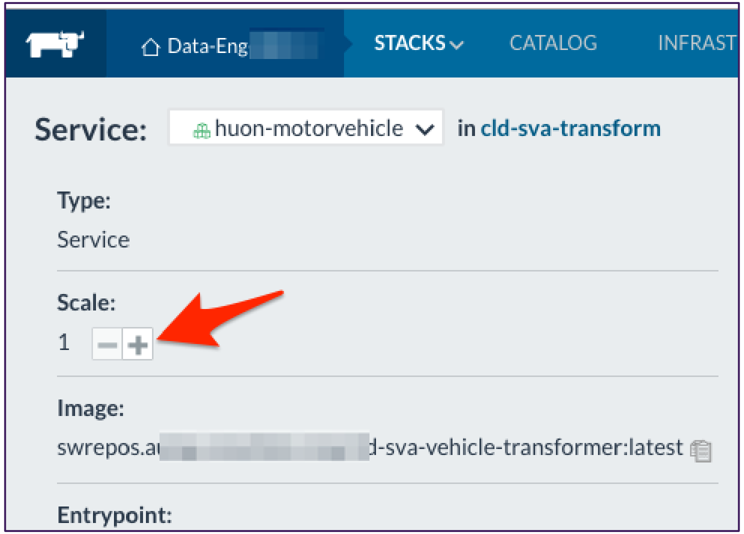
[redacted] 9:55 AM

scaling out huon now

will expand to 6 nodes then restart them all clearing out state store

that will spread the work more evenly, otherwise it will leave a lot of the stateful work on the original two nodes (since it favours nodes that have already run tasks previously so it doesn't rebuild state stores)

that should hopefully move things along 🙂



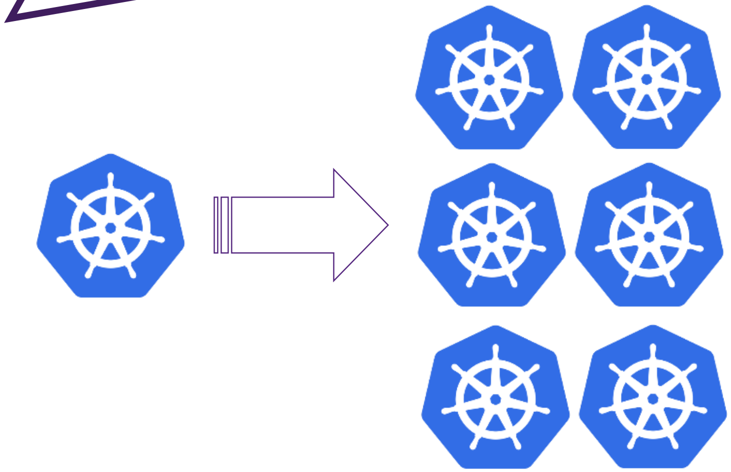
Service: huon-motorvehicle in cld-sva-transform

Type:
Service

Scale:
1 − +

Image:
swrepos.a[redacted]d-sva-vehicle-transformer:latest

Entrypoint:



Horizontal scaling … meet efficient code

Reality

30 records / sec

Approx 100,000 / hr

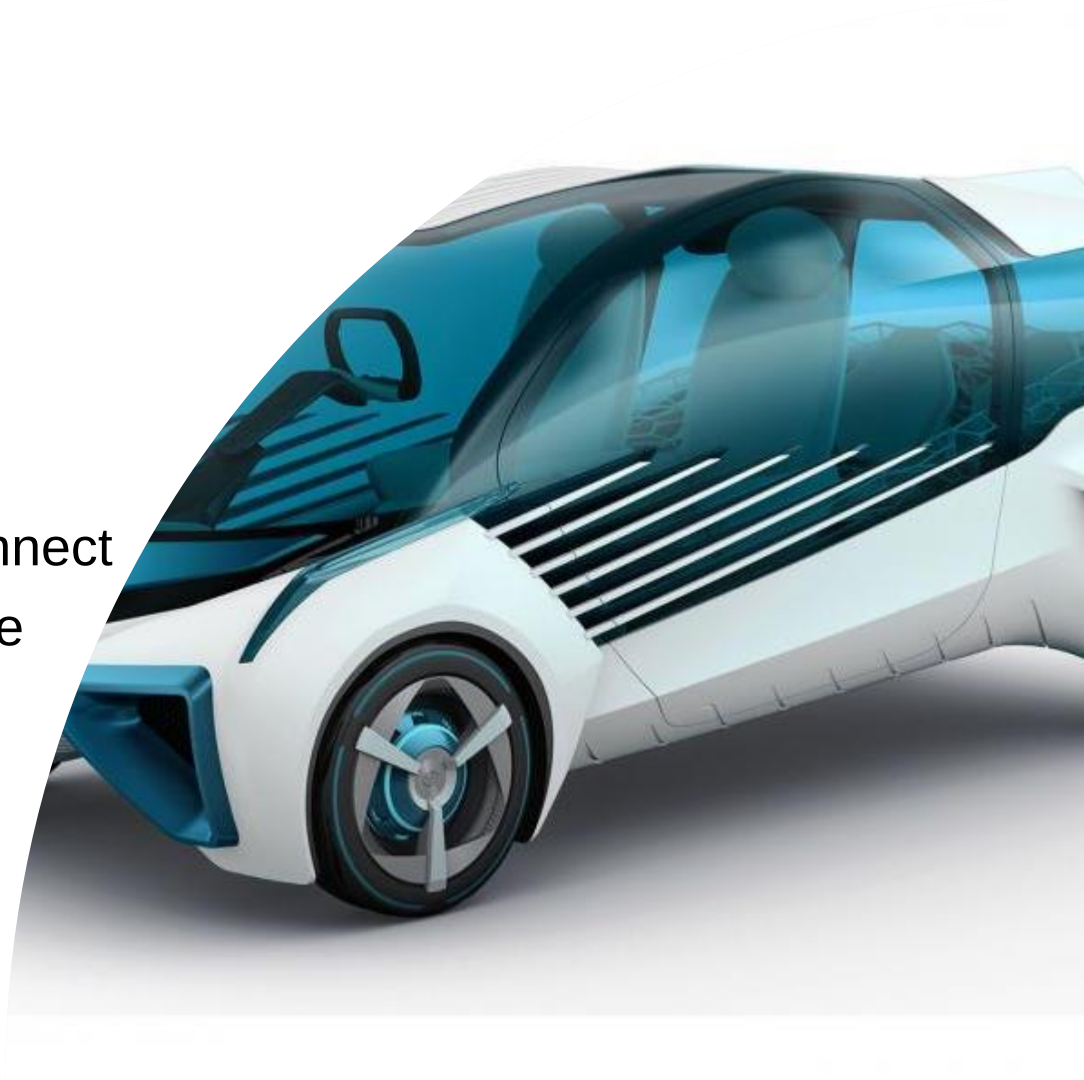
33,500 records / sec

Approx 130 mill / hr

Summary

What did we cover again?

- Architecture of our data flow
  - Extract - Low touch CDC
  - Transformation & Match
  - Serving ; Kafka & Kafka Connect
- Solutions for high performance



Questions?

iag

Australia

NRMA INSURANCE

SGIO

SGIC

CGU

wfi

swann insurance

New Zealand

STATE

ami

NZI

nac INSURANCE

LANTERN insurance

swann insurance

Asia

NZI

AmAssurance

KURNIA INSURANS

ASURANSI PAROLAMAS

SBIGeneral INSURANCE

AAA ASSURANCE
Right to peace of mind