Joker<?>

Change Data Capture Pipelines With Debezium and Kafka Streams

Gunnar Morling

Software Engineer

@gunnarmorling

Red Hat

Debezium

What's Change Data Capture?
Use Cases

1

2

Kafka Streams with Quarkus

Supersonic Subatomic Java
The Kafka Streams Extension

Debezium + Kafka Streams = 🧡

Data Enrichment
Auditing
Expanding Partial Update Events
Aggregate View Materialisation

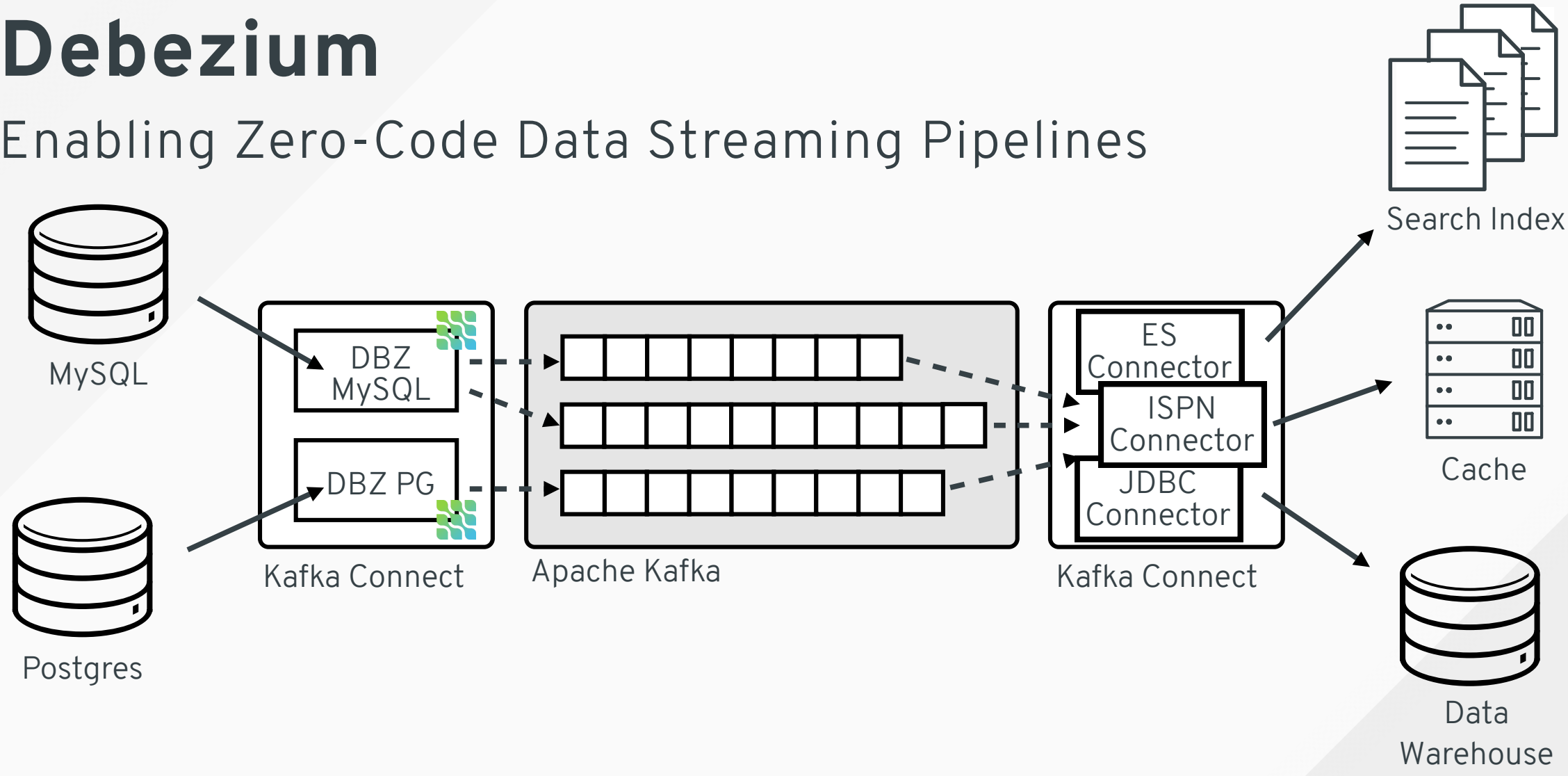
3

Gunnar Morling

- Open source software engineer at Red Hat
  - **Debezium**
  - Quarkus
  - Hibernate
- **Spec Lead** for Bean Validation 2.0 JAKARTA EE
- Other projects: **Layrry**, Deptective, MapStruct
- Java Champion 

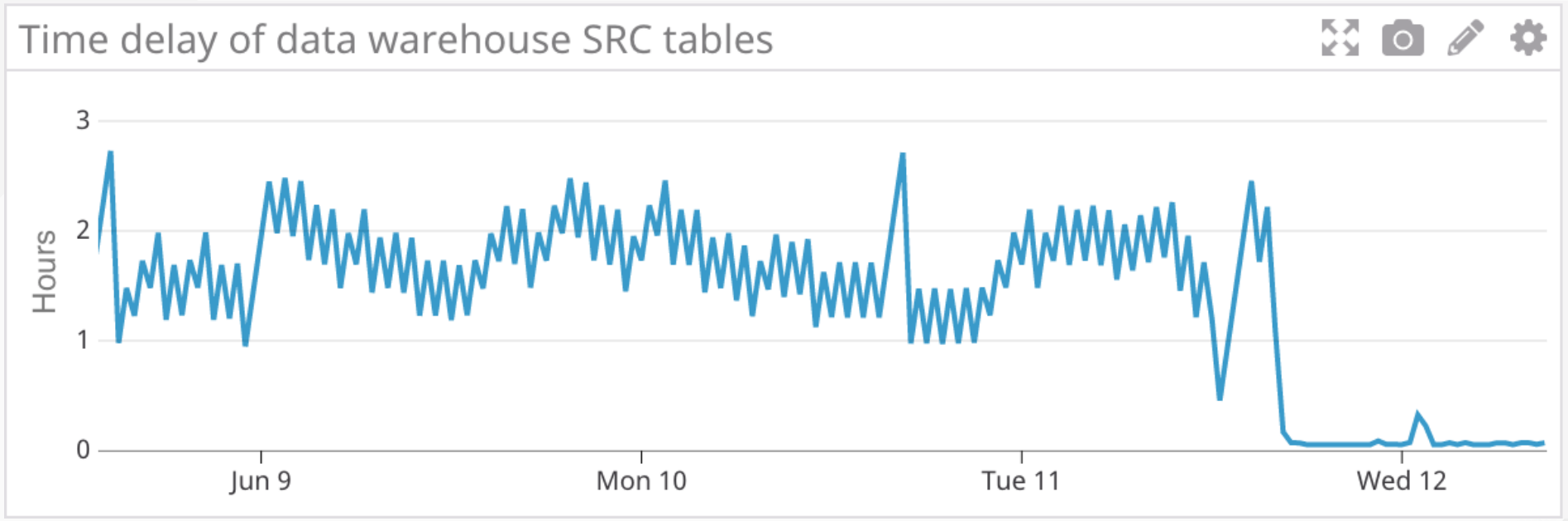
Debezium

Enabling Zero-Code Data Streaming Pipelines



Debezium

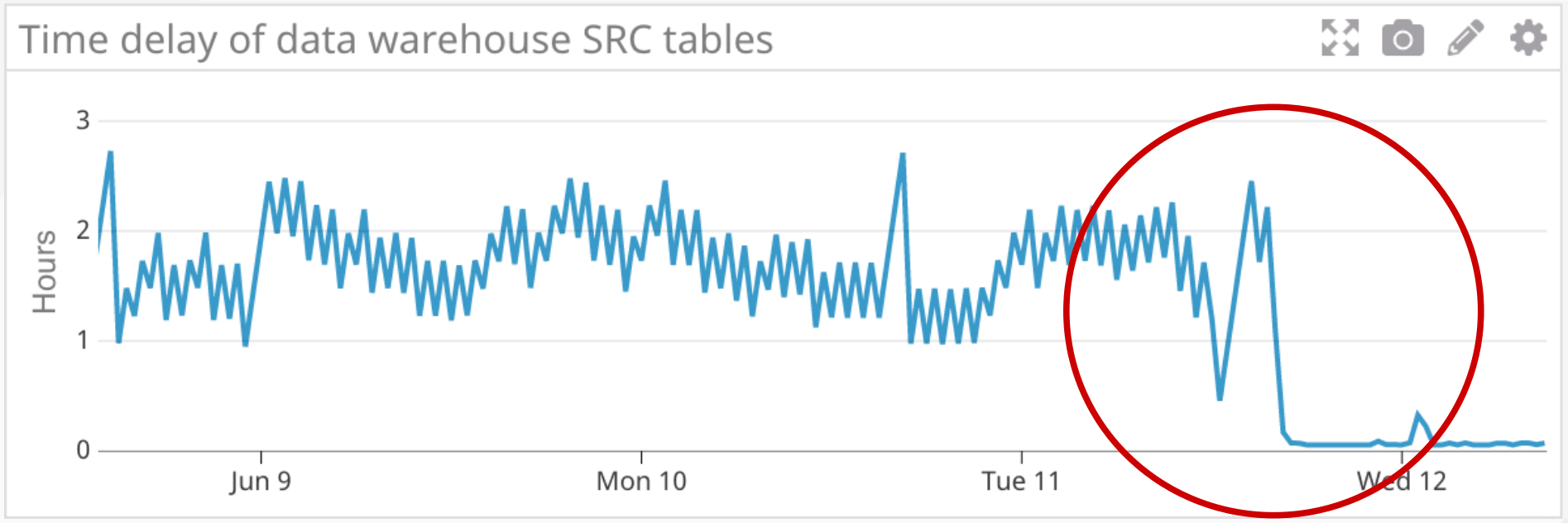
Low-Latency Change Data Streaming



https://medium.com/convoy-tech/

Debezium

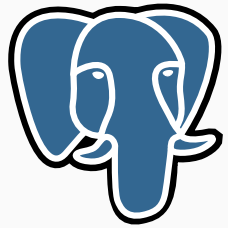
Low-Latency Change Data Streaming



https://medium.com/convoy-tech/

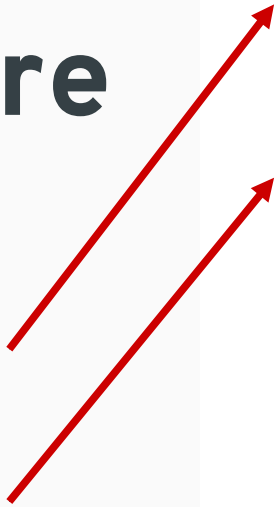
Debezium Connectors

- **MySQL**
- **Postgres**
- **MongoDB**
- **SQL Server**
- Incubating
  - Cassandra
  - Oracle
  - Db2
  - Vitess



Change Event Structure

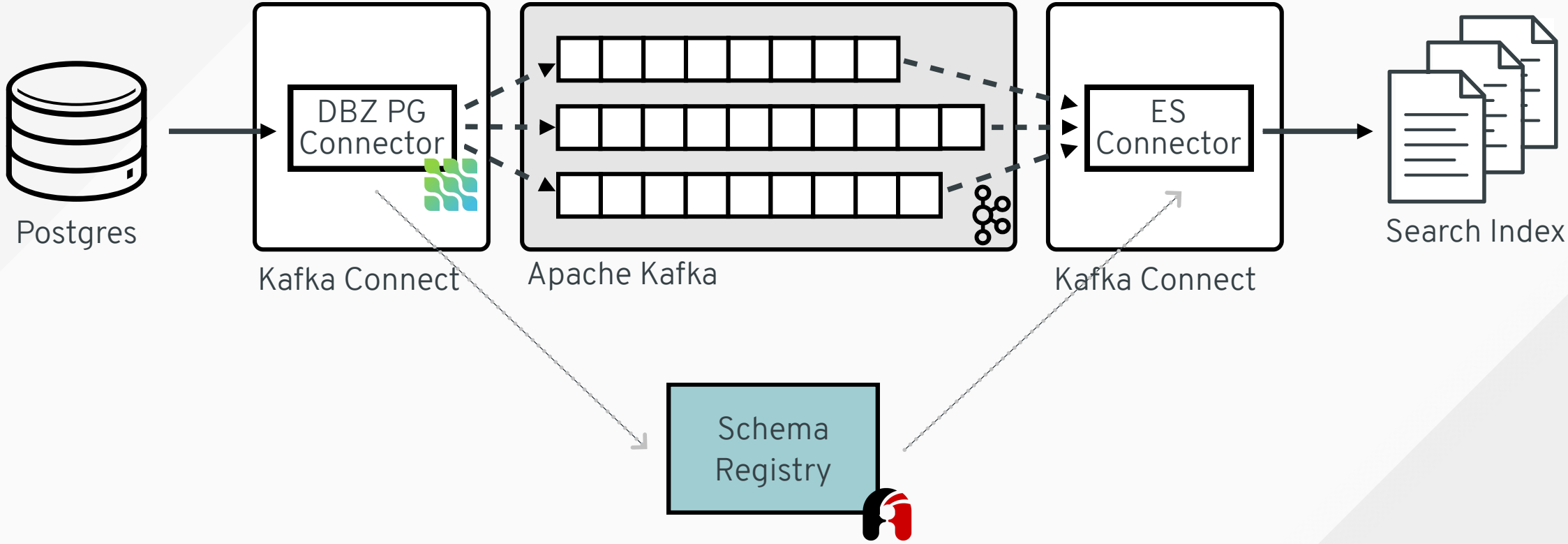
- Key: **Primary key** of table
- Value: Describing the change event
  - **Old** row state
  - **New** row state
  - **Metadata**
- **Serialization**
  - JSON
  - Avro
  - Schema Registry



```
{
  "before": null,
  "after": {
    "id": 1004,
    "first_name": "Anne",
    "last_name": "Kretchmar",
    "email": "annek@noanswer.org"
  },
  "source": {
    "name": "dbserver1",
    "server_id": 0,
    "ts_sec": 0,
    "file": "mysql-bin.000003",
    "pos": 154,
    "row": 0,
    "snapshot": true,
    "db": "inventory",
    "table": "customers"
  },
  "op": "c",
  "ts_ms": 1486500577691
}
```

Schema Registry

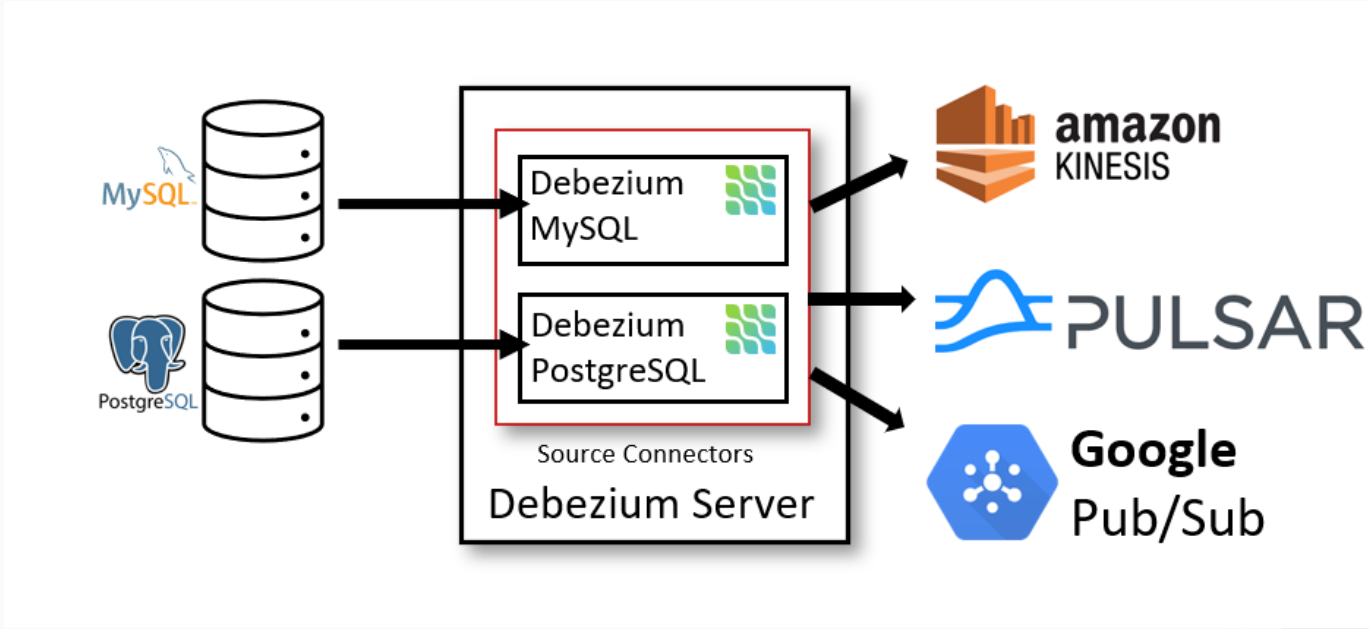
Managing Schema Evolution



Debezium

Three Ways of Using it

- Kafka Connect
- Embedded library
- Debezium Server

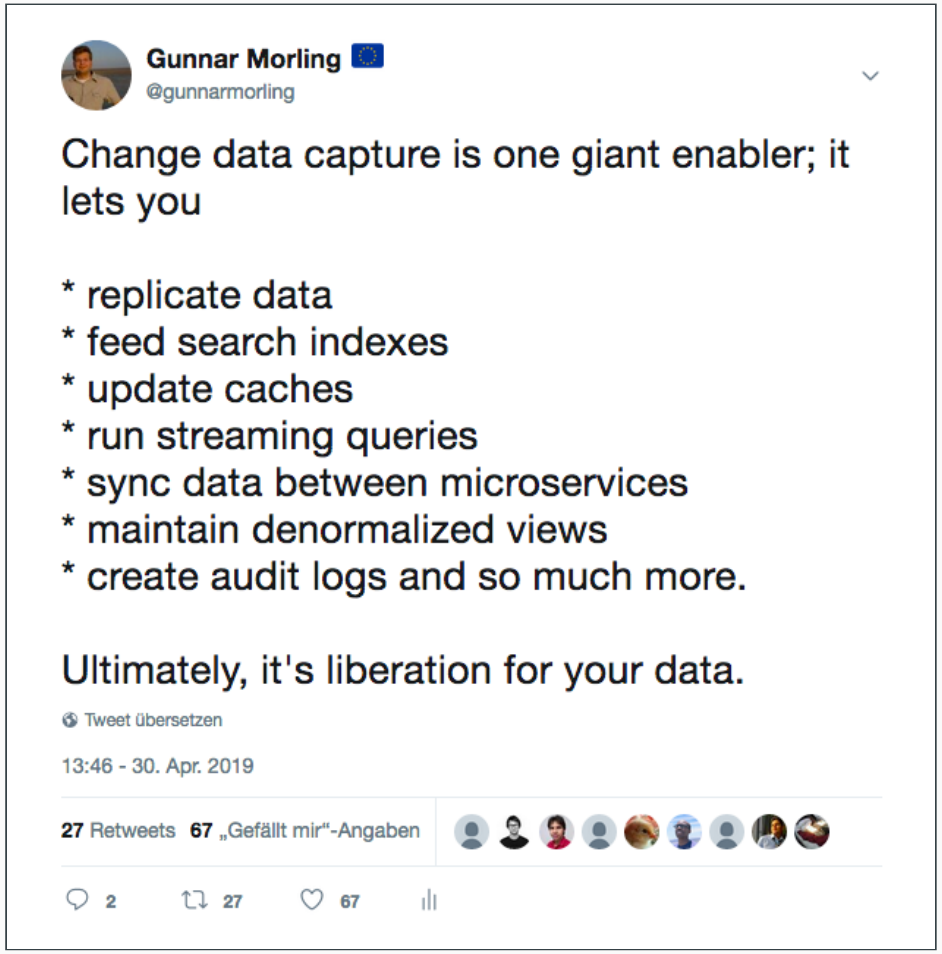


Meme idea: Robin Moffatt

Log- vs. Query-Based CDC

	Query-Based	Log-Based
All data changes are captured	-	+
No polling delay or overhead	-	+
Transparent to writing applications and models	-	+
Can **capture deletes** and **old record state**	-	+
Installation/Configuration	+	-

CDC – "Liberation for Your Data"



Gunnar Morling
@gunnarmorling

Change data capture is one giant enabler; it lets you

* replicate data
* feed search indexes
* update caches
* run streaming queries
* sync data between microservices
* maintain denormalized views
* create audit logs and so much more.

Ultimately, it's liberation for your data.

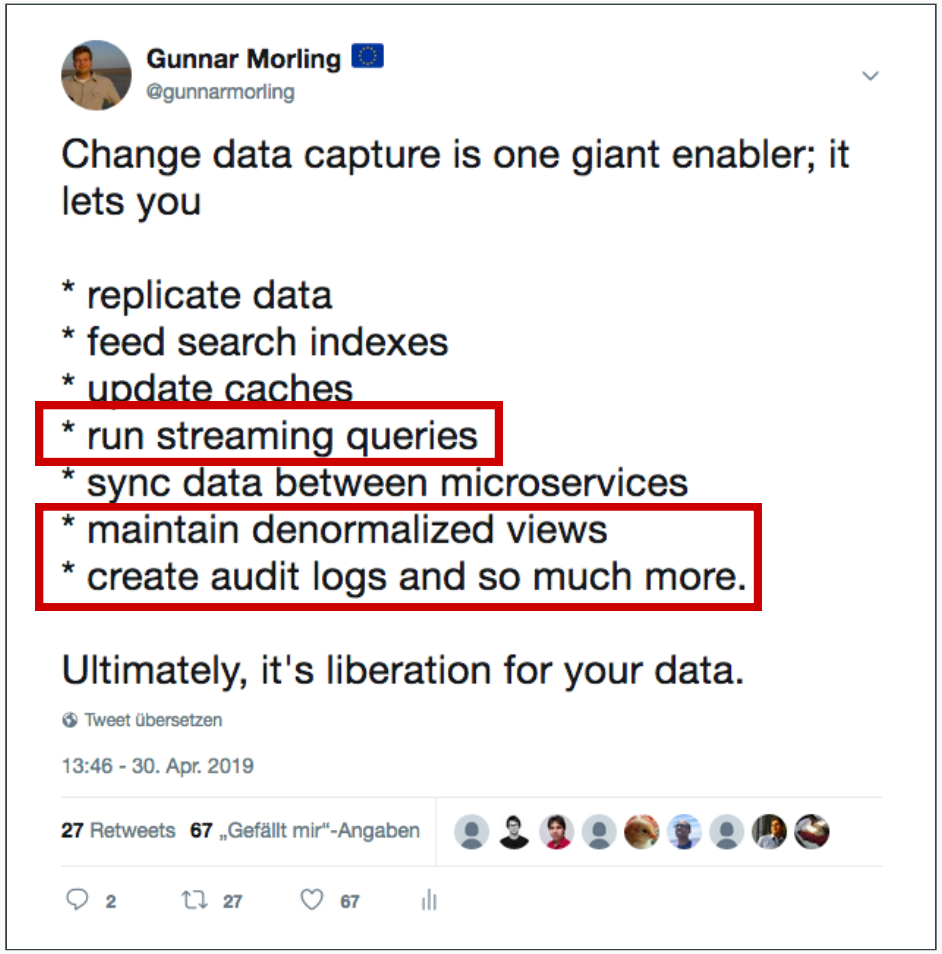
Tweet übersetzen

13:46 - 30. Apr. 2019

27 Retweets 67 „Gefällt mir"-Angaben

2 27 67

CDC – "Liberation for Your Data"



Gunnar Morling
@gunnarmorling

Change data capture is one giant enabler; it lets you

* replicate data
* feed search indexes
* update caches
* run streaming queries
* sync data between microservices
* maintain denormalized views
* create audit logs and so much more.

Ultimately, it's liberation for your data.

Tweet übersetzen

13:46 - 30. Apr. 2019

27 Retweets 67 „Gefällt mir"-Angaben

2 27 67

Debezium

What's Change Data Capture?
Use Cases

1

2

Kafka Streams with Quarkus

Supersonic Subatomic Java
The Kafka Streams Extension

3

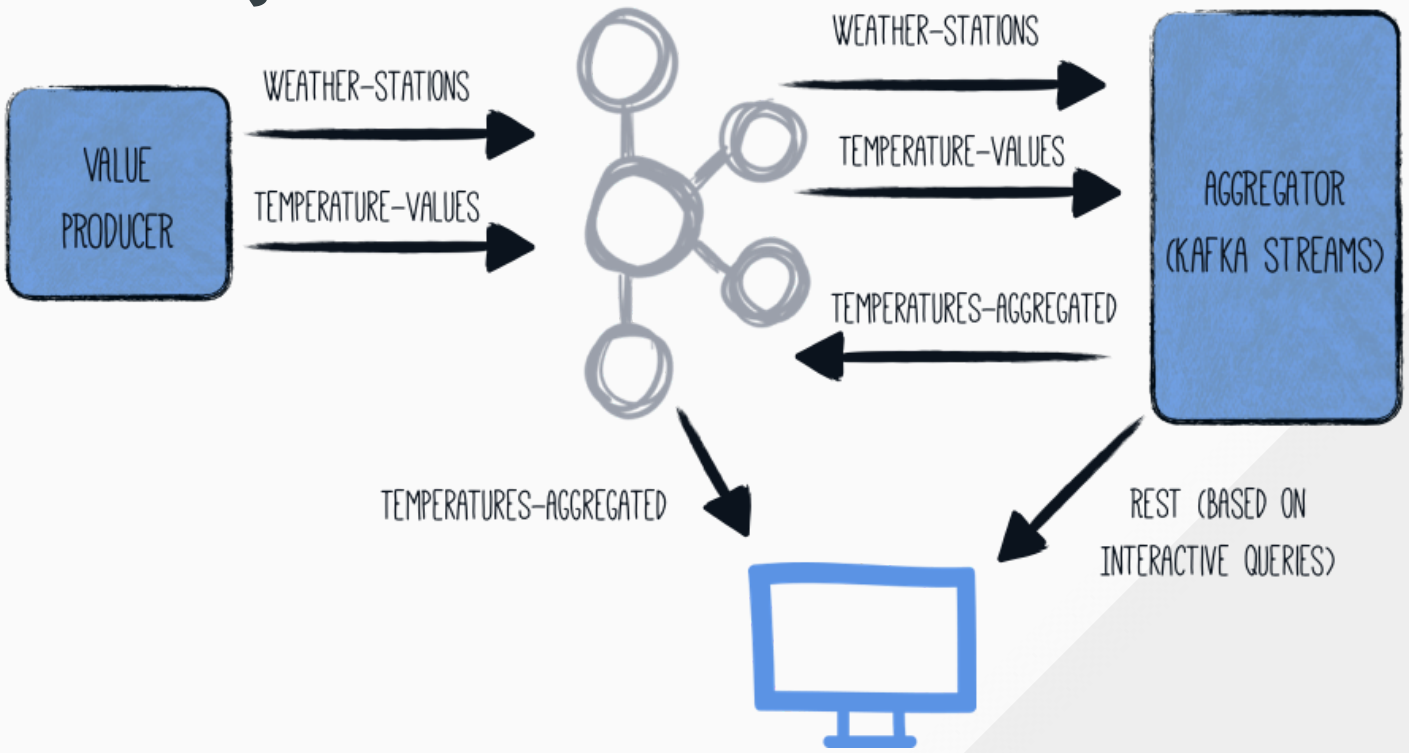
Debezium + Kafka Streams = 🧡

Data Enrichment
Auditing
Expanding Partial Update Events
Aggregate View Materialisation

Kafka Streams

Streaming Queries on Kafka Topics

- Java API for **stateful stream processing**
- Rich set of **Operators**
- **Scaling out** to multiple JVMs
- **Interactive queries**



Kafka Streams

```
public static void main(final String[] args) throws Exception {
  Properties props = new Properties();
  props.put(StreamsConfig.APPLICATION_ID_CONFIG, "wordcount-application");
  props.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "kafka-broker1:9092");
  props.put(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass());
  props.put(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass());

  StreamsBuilder builder = new StreamsBuilder();
  KStream<String, String> textLines = builder.stream("TextLinesTopic");
  KTable<String, Long> wordCounts = textLines
    .flatMapValues(textLine -> Arrays.asList(textLine.toLowerCase().split("\\W+")))
    .groupBy((key, word) -> word)
    .count(Materialized.<String, Long, KeyValueStore<Bytes, byte[]>>as("counts-store"));
  wordCounts.toStream().to("WordsWithCountsTopic", Produced.with(Serdes.String(), Serdes.Long()));

  KafkaStreams streams = new KafkaStreams(builder.build(), props);
  Runtime.getRuntime().addShutdownHook(new Thread(streams::close));

  streams.start();
}
```

Kafka Streams

```
public static void main(final String[] args) throws Exception {
  Properties props = new Properties();
  props.put(StreamsConfig.APPLICATION_ID_CONFIG, "wordcount-application");
  props.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "kafka-broker1:9092");
  props.put(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass());
  props.put(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass());

  StreamsBuilder builder = new StreamsBuilder();
  KStream<String, String> textLines = builder.stream("TextLinesTopic");
  KTable<String, Long> wordCounts = textLines
    .flatMapValues(textLine -> Arrays.asList(textLine.toLowerCase().split("\\W+")))
    .groupBy((key, word) -> word)
    .count(Materialized.<String, Long, KeyValueStore<Bytes, byte[]>>as("counts-store"));
  wordCounts.toStream().to("WordsWithCountsTopic", Produced.with(Serdes.String(), Serdes.Long()));

  KafkaStreams streams = new KafkaStreams(builder.build(), props);
  Runtime.getRuntime().addShutdownHook(new Thread(streams::close));

  streams.start();
}
```

Kafka Streams

```
public static void main(final String[] args) throws Exception {
  Properties props = new Properties();
  props.put(StreamsConfig.APPLICATION_ID_CONFIG, "wordcount-application");
  props.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "kafka-broker1:9092");
  props.put(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass());
  props.put(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass());

  StreamsBuilder builder = new StreamsBuilder();
  KStream<String, String> textLines = builder.stream("TextLinesTopic");
  KTable<String, Long> wordCounts = textLines
    .flatMapValues(textLine -> Arrays.asList(textLine.toLowerCase().split("\\W+")))
    .groupBy((key, word) -> word)
    .count(Materialized.<String, Long, KeyValueStore<Bytes, byte[]>>as("counts-store"));
  wordCounts.toStream().to("WordsWithCountsTopic", Produced.with(Serdes.String(), Serdes.Long()));

  KafkaStreams streams = new KafkaStreams(builder.build(), props);
  Runtime.getRuntime().addShutdownHook(new Thread(streams::close));

  streams.start();
}
```

Kafka Streams

```
public static void main(final String[] args) throws Exception {
  Properties props = new Properties();
  props.put(StreamsConfig.APPLICATION_ID_CONFIG, "wordcount-application");
  props.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "kafka-broker1:9092");
  props.put(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass());
  props.put(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass());

  StreamsBuilder builder = new StreamsBuilder();
  KStream<String, String> textLines = builder.stream("TextLinesTopic");
  KTable<String, Long> wordCounts = textLines
    .flatMapValues(textLine -> Arrays.asList(textLine.toLowerCase().split("\\W+")))
    .groupBy((key, word) -> word)
    .count(Materialized.<String, Long, KeyValueStore<Bytes, byte[]>>as("counts-store"));
  wordCounts.toStream().to("WordsWithCountsTopic", Produced.with(Serdes.String(), Serdes.Long()));

  KafkaStreams streams = new KafkaStreams(builder.build(), props);
  Runtime.getRuntime().addShutdownHook(new Thread(streams::close));

  streams.start();
}
```

Quarkus

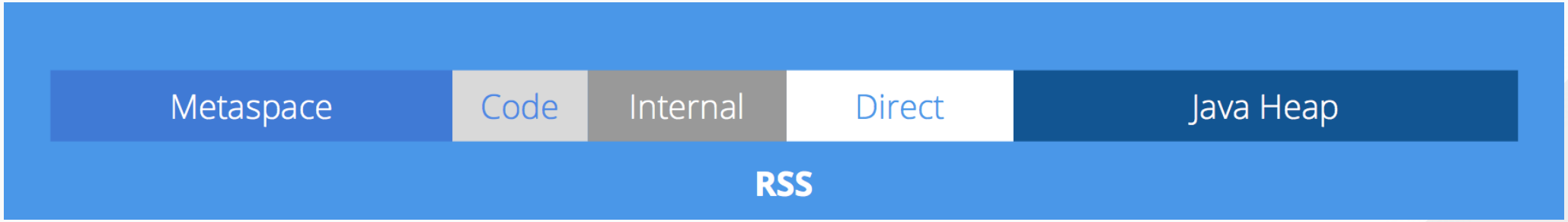
Supersonic Subatomic Java

“ A Kubernetes Native Java stack tailored for OpenJDK HotSpot and GraalVM, crafted from the best of breed Java libraries and standards.

Quarkus

The Truth About Java and Containers

- Designed for **high through-put** (requests/s)
- **Startup overhead:** # of classes, bytecode, JIT
- **Memory overhead:** # of classes, metadata, compilation




$ ./my-native-java-rest-app
Quarkus started in 0.008s
Memory (RSS) in Megabytes*
*Tested on a single-core machine
REST
Quarkus + Native (via GraalVM)
12 MB
Quarkus + JVM (via OpenJDK)
73 MB
Traditional Cloud-Native Stack
136 MB
REST + CRUD
Quarkus + Native (via GraalVM)
28 MB
Quarkus + JVM (via OpenJDK)
145 MB
Traditional Cloud-Native Stack
209MB
BOOT + First Response Time
REST
Quarkus + Native (via GraalVM)
0.016 Seconds
Quarkus + JIT (via OpenJDK)
0.943 Seconds
Traditional Cloud-Native Stack
4.3 Seconds
REST + CRUD
Quarkus + Native (via GraalVM)
0.042 Seconds
Quarkus + JIT (via OpenJDK)
2.033 Seconds
Traditional Cloud-Native Stack
9.5 Seconds

Quarkus

What Does a Framework Do at Start-up Time?

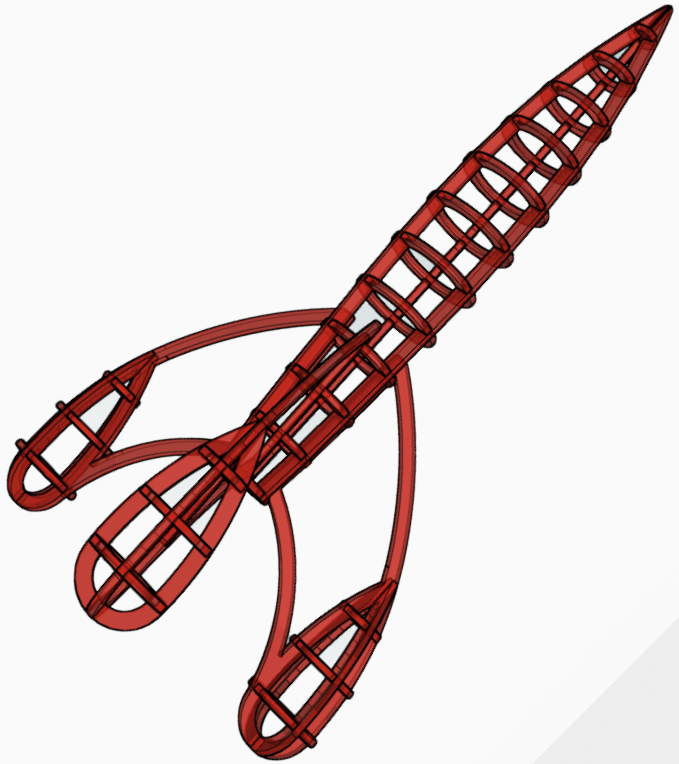
Parse **config files**
Classpath & classes scanning for annotations, getters or other metadata
Build framework **metamodel objects**
Prepare reflection and build proxies

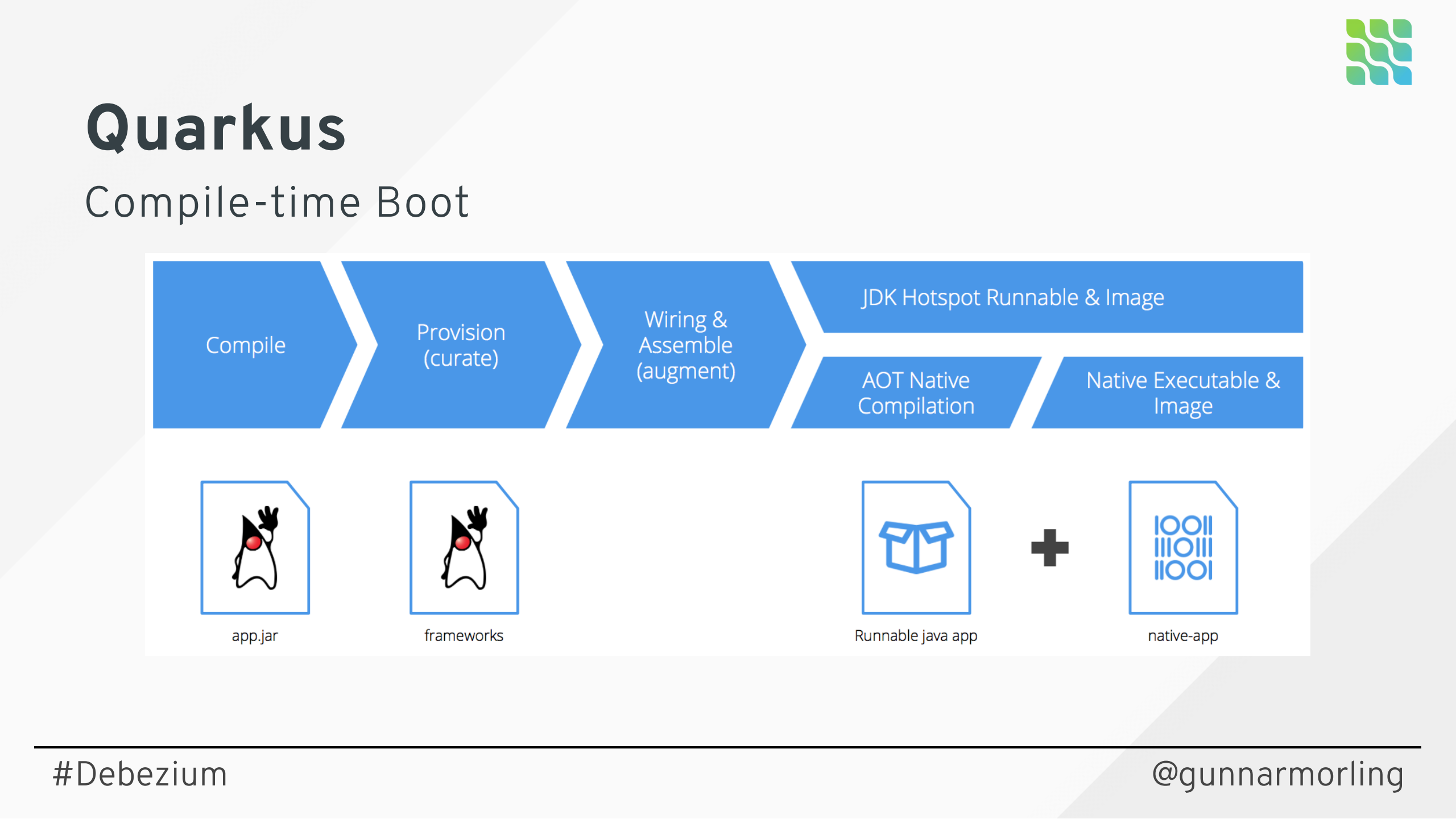
Start and open IO, threads, etc.

Quarkus

Compile-time Boot

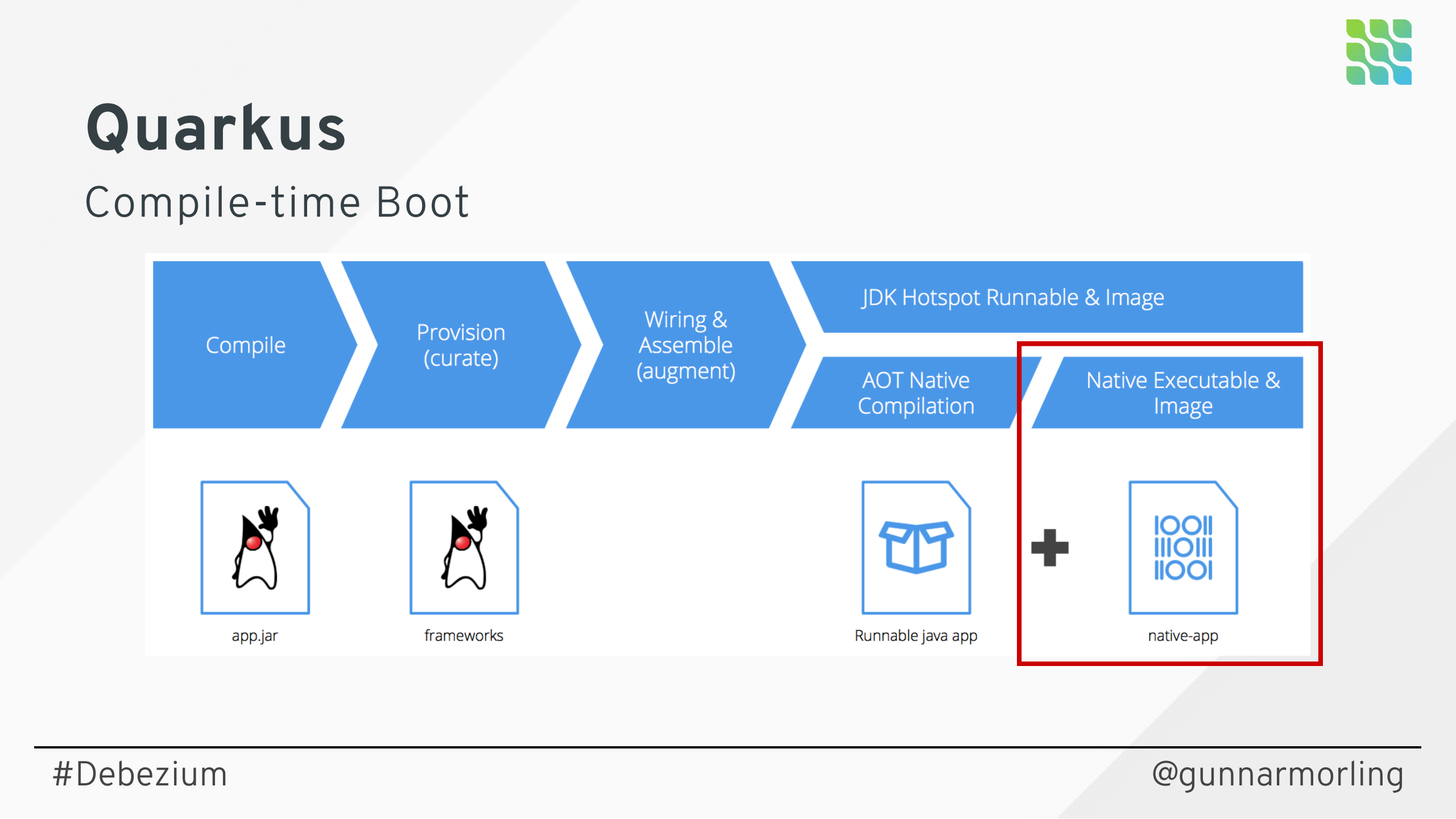
- Do the work **once**, not at each start
- All the bootstrap classes are **no longer loaded**
- **Less time** to start, **less memory** used
- Less or no reflection nor dynamic proxies





Quarkus

Compile-time Boot



Quarkus

Compile-time Boot

Quarkus

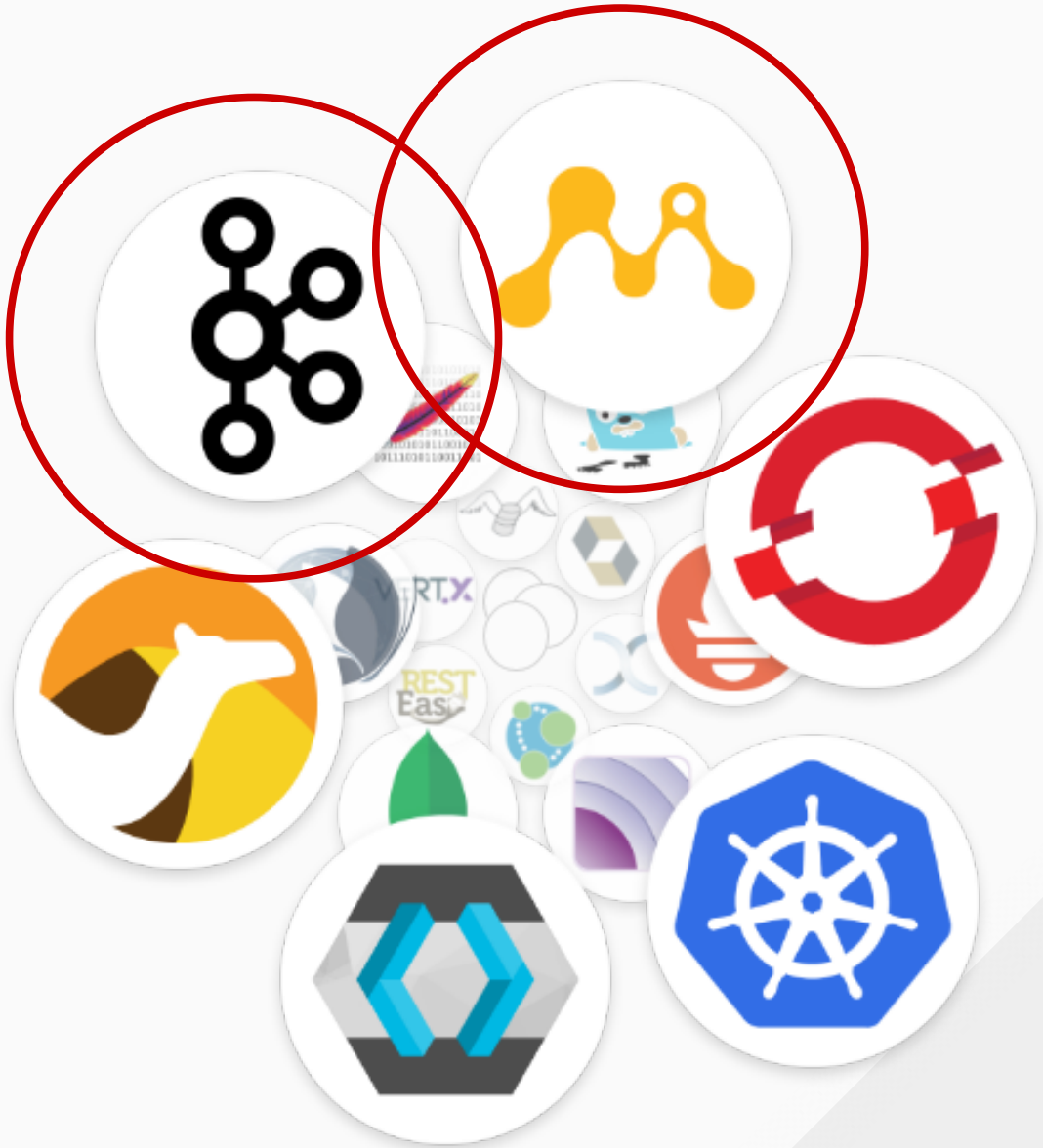
Supersonic Subatomic Java

- **Developer joy**
- **Imperative and Reactive**
- Best-of-breed **libraries**

Quarkus

Supersonic Subatomic Java

- **Developer joy**
- **Imperative and Reactive**
- Best-of-breed **libraries**



Quarkus

Extensions

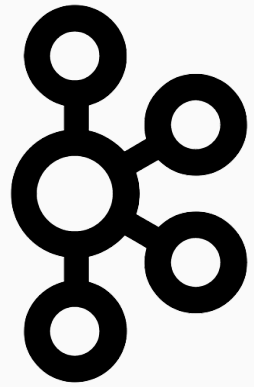
- **Units of** Quarkus **distribution**
- Configure, boot and **integrate frameworks** into Quarkus
- **Cater for GraalVM AOT** and closed-world assumption

```
private void registerDefaultExceptionHandler(
        BuildProducer<ReflectiveClassBuildItem> reflectiveClasses) {

    reflectiveClasses.produce(
            new ReflectiveClassBuildItem(
                    true, false, false, LogAndFailExceptionHandler.class));
}
```

Quarkus

The Kafka Streams Extension

- **Management** of topology
- **Health** checks
- **Metrics**
- **Dev Mode**
- Support for **native binaries** via GraalVM
  - Reflection on Serdes, exception handlers etc.
  - RocksDB



Debezium

What's Change Data Capture?
Use Cases

1

2

Kafka Streams with Quarkus

Supersonic Subatomic Java
The Kafka Streams Extension

Debezium + Kafka Streams = 🧡

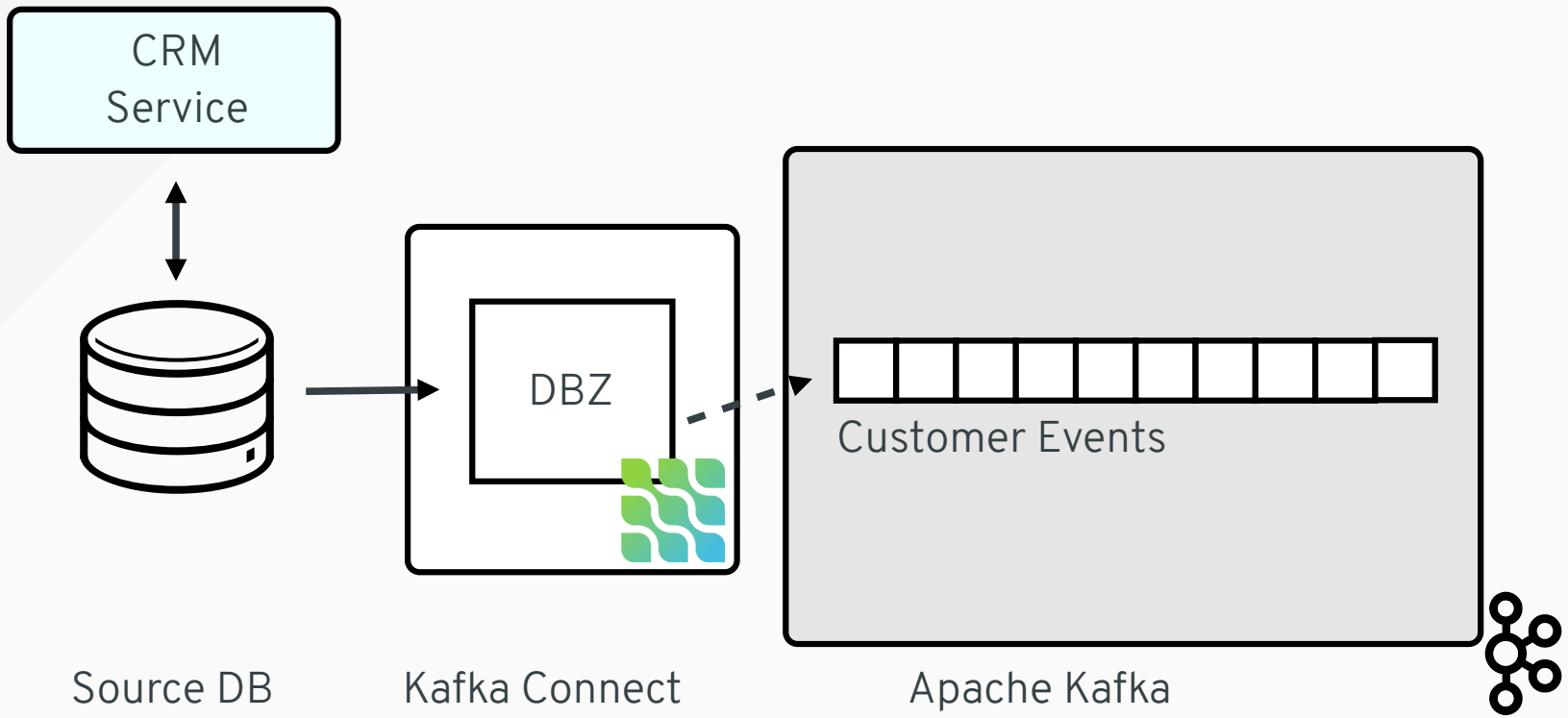
Data Enrichment
Auditing
Expanding Partial Update Events
Aggregate View Materialisation

3

Data Enrichment - Demo

Auditing

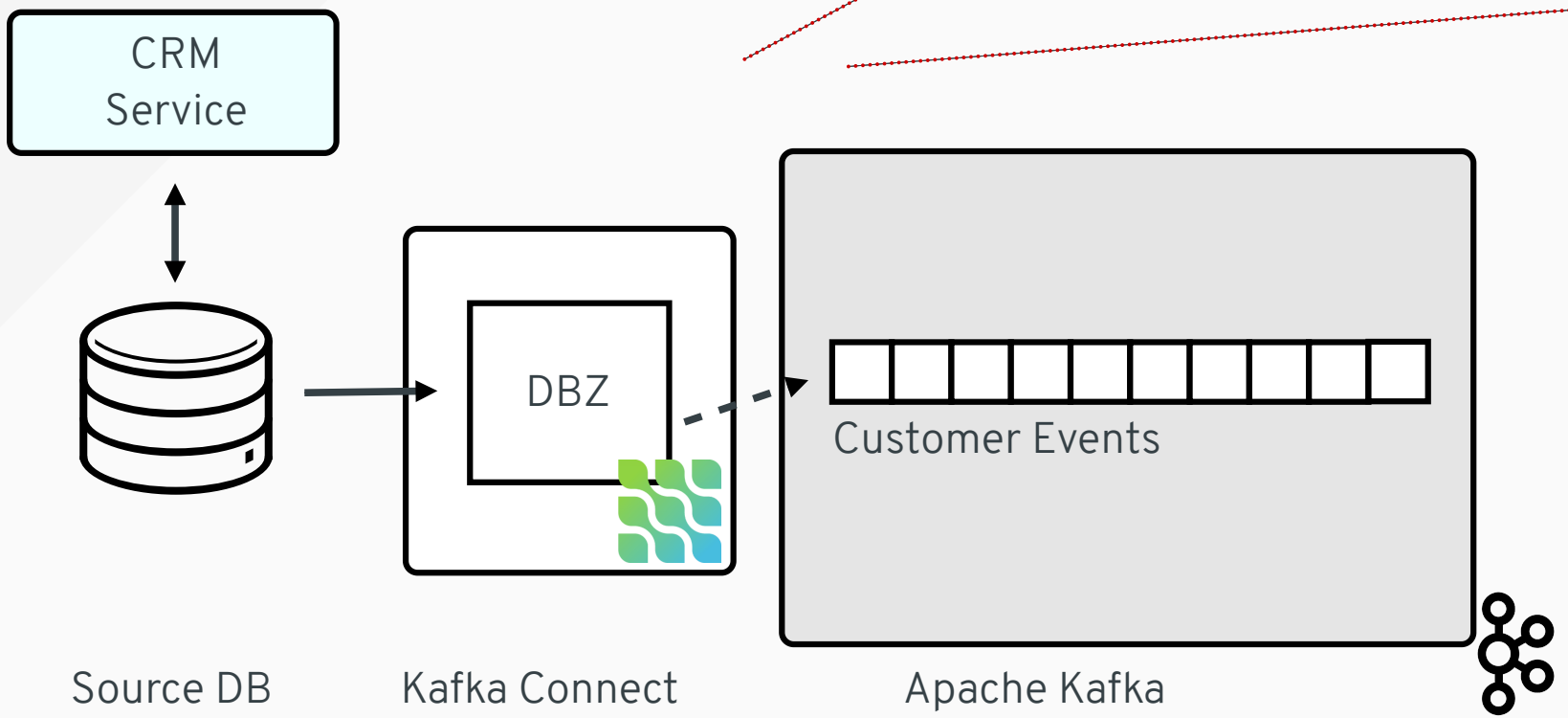
Auditing


CRM
Service
DBZ
Customer Events
Source DB
Kafka Connect
Apache Kafka

Auditing

"Transactions" table

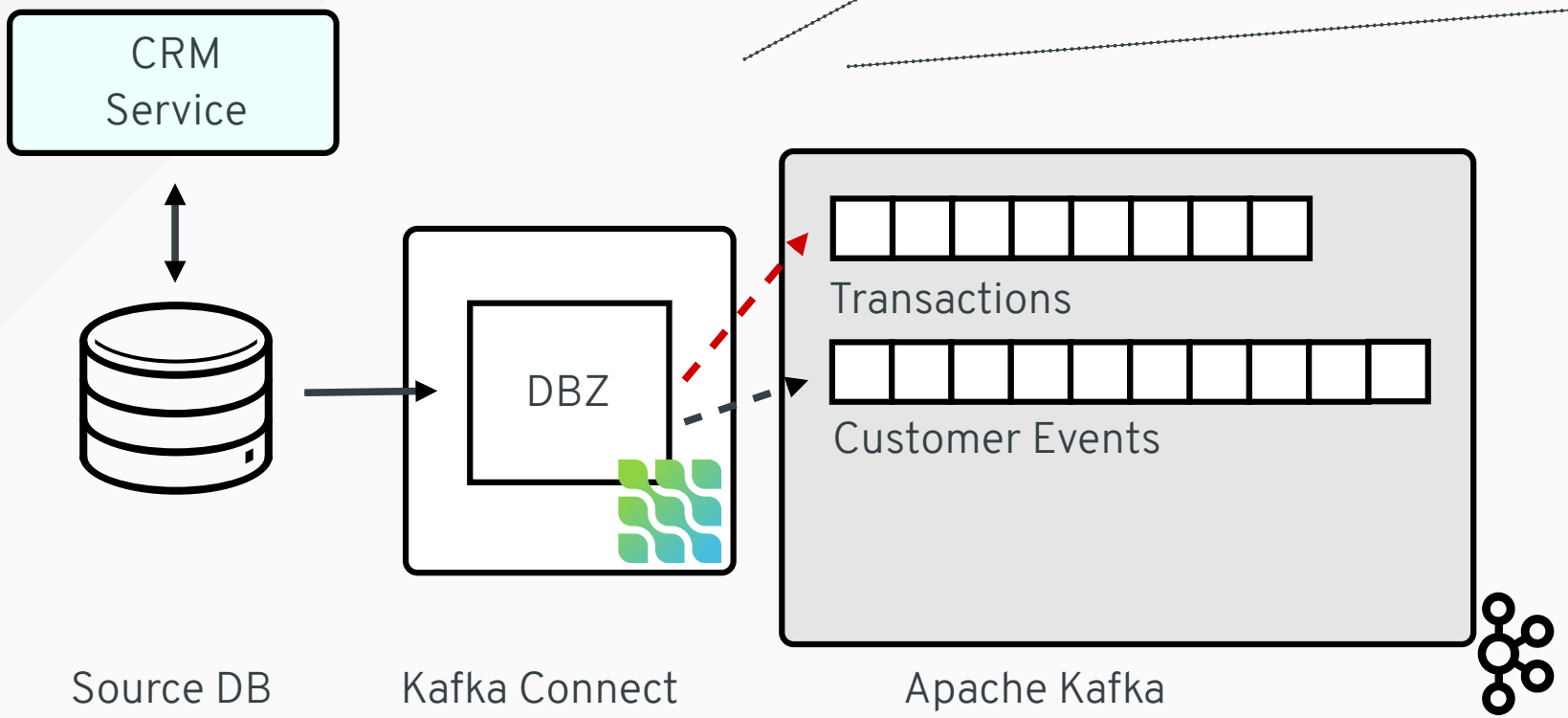
Id	User	Use Case
tx-1	Bob	Create Customer
tx-2	Sarah	Delete Customer
tx-3	Rebecca	Update Customer



Auditing

"Transactions" table

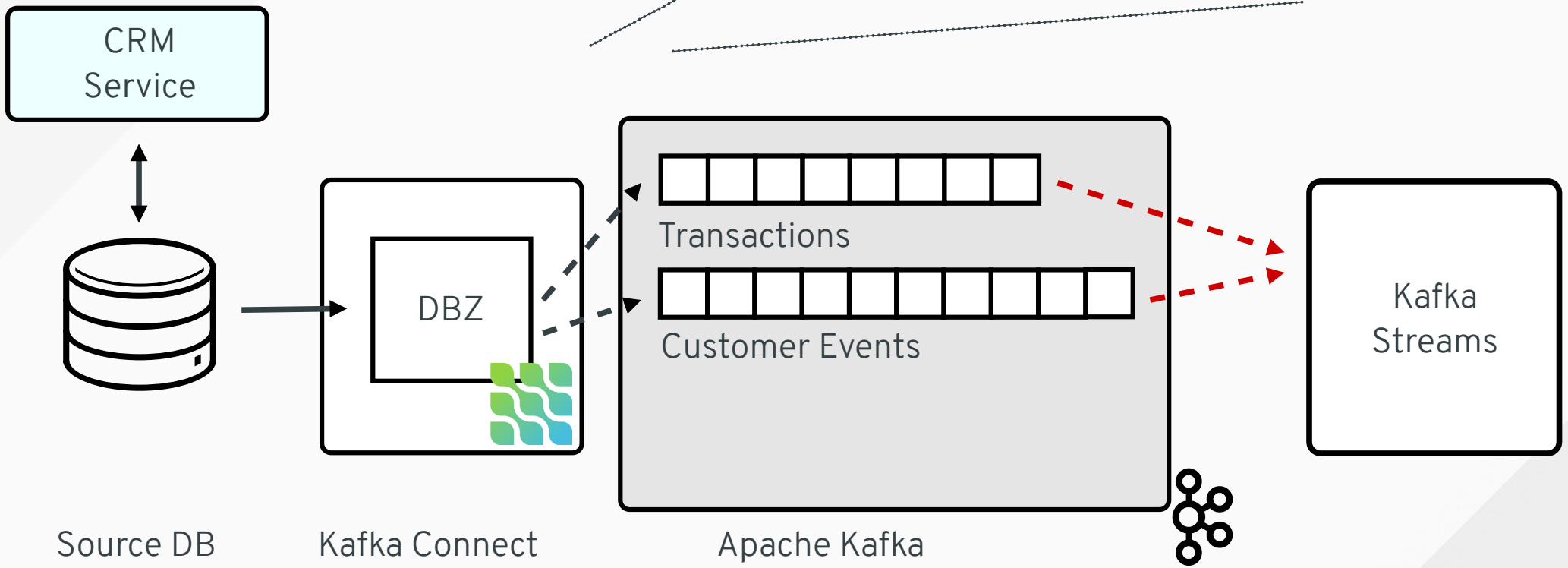
Id	User	Use Case
tx-1	Bob	Create Customer
tx-2	Sarah	Delete Customer
tx-3	Rebecca	Update Customer



Auditing

"Transactions" table

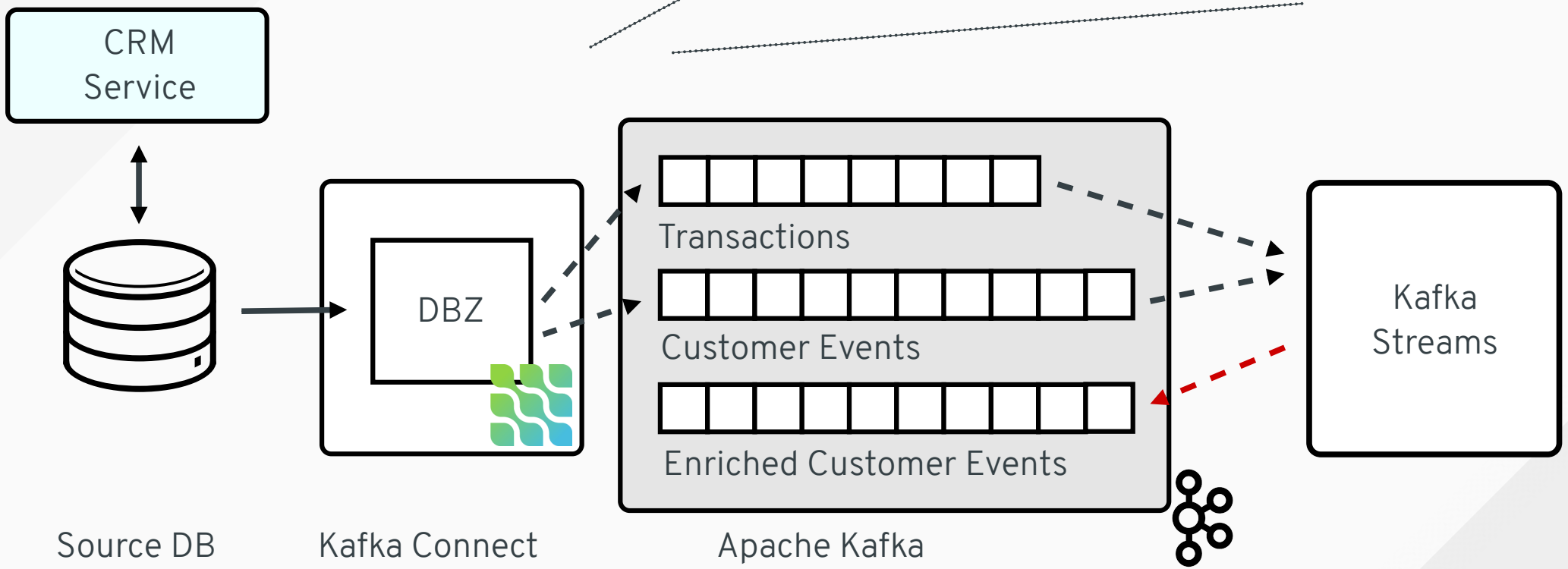
Id	User	Use Case
tx-1	Bob	Create Customer
tx-2	Sarah	Delete Customer
tx-3	Rebecca	Update Customer



Auditing

"Transactions" table

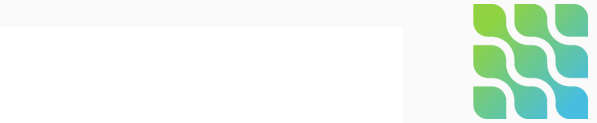
Id	User	Use Case
tx-1	Bob	Create Customer
tx-2	Sarah	Delete Customer
tx-3	Rebecca	Update Customer



Auditing

```
{
  "before": {
    "id": 1004,
    "last_name": "Kretchmar",
    "email": "annek@example.com"
  },
  "after": {
    "id": 1004,
    "last_name": "Kretchmar",
    "email": "annek@noanswer.org"
  },
  "source": {
    "name": "dbserver1",
    "table": "customers",
    "txId": "tx-3"
  },
  "op": "u",
  "ts_ms": 1486500577691
}
```

Customers



```
{
  "id": "tx-3"
}
```

```
{
  "before": null,
  "after": {
    "id": "tx-3",
    "user": "Rebecca",
    "use_case": "Update customer"
  },
  "source": {
    "name": "dbserver1",
    "table": "transactions",
    "txId": "tx-3"
  },
  "op": "c",
  "ts_ms": 1486500577691
}
```

Transactions

```
{
  "before": {
    "id": 1004,
    "last_name": "Kretchmar",
    "email": "annek@example.com"
  },
  "after": {
    "id": 1004,
    "last_name": "Kretchmar",
    "email": "annek@noanswer.org"
  },
  "source": {
    "name": "dbserver1",
    "table": "customers",
    "txId": "tx-3"
  },
  "op": "u",
  "ts_ms": 1486500577691
}
```

Customers

```
{
  "id": "tx-3"
}
```

```
{
  "before": null,
  "after": {
    "id": "tx-3",
    "user": "Rebecca",
    "use_case": "Update customer"
  },
  "source": {
    "name": "dbserver1",
    "table": "transactions",
    "txId": "tx-3"
  },
  "op": "c",
  "ts_ms": 1486500577691
}
```

Transactions

```
{
  "before": {
    "id": 1004,
    "last_name": "Kretchmar",
    "email": "annek@example.com"
  },
  "after": {
    "id": 1004,
    "last_name": "Kretchmar",
    "email": "annek@noanswer.org"
  },
  "source": {
    "name": "dbserver1",
    "table": "customers",
    "txId": "tx-3"
  },
  "op": "u",
  "ts_ms": 1486500577691
}
```

Customers

Auditing


{
"before": {
"id": 1004,
"last_name": "Kretchmar",
"email": "annek@example.com"
},
"after": {
"id": 1004,
"last_name": "Kretchmar",
"email": "annek@noanswer.org"
},
"source": {
"name": "dbserver1",
"table": "customers",
"txId": "tx-3",
"user": "Rebecca",
"use_case": "Update customer"
},
"op": "u",
"ts_ms": 1486500577691
}
Enriched Customers

Auditing

- Non-trivial join implementation
  - **no ordering** across topics
  - need to **buffer change events** until TX data available
- bit.ly/debezium-auditlogs

```
@Override
public KeyValue<JsonObject, JsonObject>
    transform(JsonObject key, JsonObject value) {

  boolean enrichedAllBufferedEvents =
      enrichAndEmitBufferedEvents();

  if (!enrichedAllBufferedEvents) {
    bufferChangeEvent(key, value);
    return null;
  }

  KeyValue<JsonObject, JsonObject> enriched =
      enrichWithTxMetaData(key, value);
  if (enriched == null) {
    bufferChangeEvent(key, value);
  }

  return enriched;
}
```


Expanding Partial
Update Events

Expanding Partial Update Events

Examples

- MongoDB update events (**"patch"**)
- Postgres
  - Replica identity **not FULL**
  - **TOAST-ed** columns
- Cassandra update events
- MySQL with row image minimal

```
{
  "before": { ... },
  "after": {
    "id": 1004,
    "first_name": "Dana",
    "last_name": "Kretchmar",
    "email": "annek@noanswer.org",
    "biography":
      "__debezium_unavailable_value"
  },
  "source": { ... },
  "op": "u",
  "ts_ms": 1570448151611
}
```

Expanding Partial Update Events

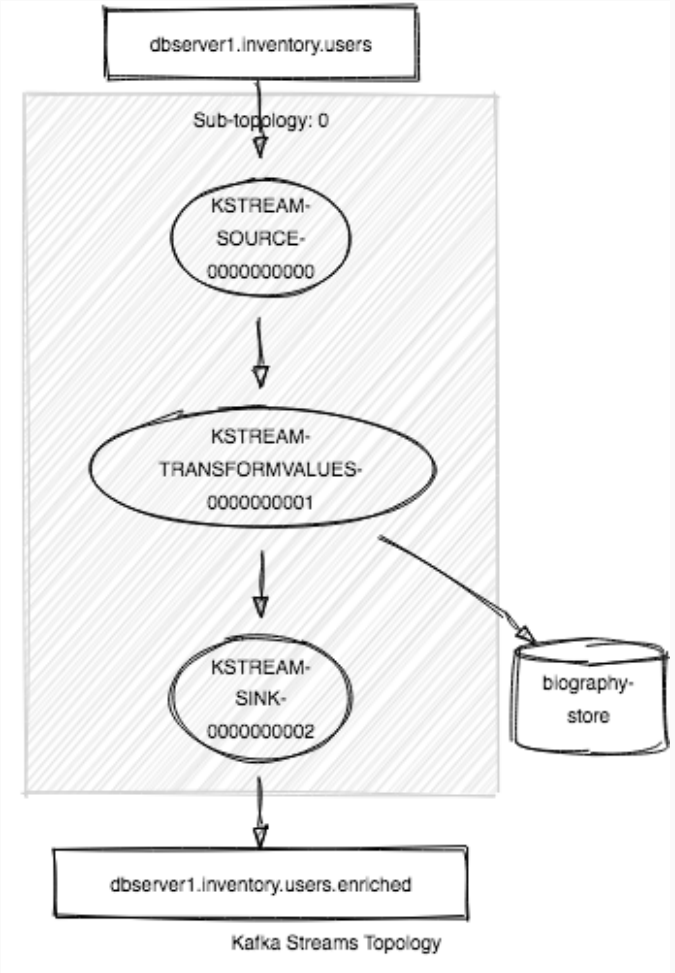
Examples

- MongoDB update events (**"patch"**)
- Postgres
  - Replica identity **not FULL**
  - **TOAST-ed** columns
- Cassandra update events
- MySQL with row image minimal

```
{
  "before": { ... },
  "after": {
    "id": 1004,
    "first_name": "Dana",
    "last_name": "Kretchmar",
    "email": "annek@noanswer.org",
    "biography":
      "__debezium_unavailable_value"
  },
  "source": { ... },
  "op": "u",
  "ts_ms": 1570448151611
}
```

Expanding Partial Update Events

Topology



https://zz85.github.io/kafka-streams-viz/

Expanding Partial Update Events

Obtaining missing values from a state store

```
class ToastColumnValueProvider implements ValueTransformerWithKey<JsonObject, JsonObject, JsonObject>

  private KeyValueStore<JsonObject, String> biographyStore;

  @Override
  public void init(ProcessorContext context) {
    biographyStore = (KeyValueStore<JsonObject, String>) context.getStateStore(
        TopologyProducer.BIOGRAPHY_STORE);
  }

  @Override
  public JsonObject transform(JsonObject key, JsonObject value) {
    // ...
  }
}
```

Expanding Partial Update Events

Obtaining missing values from a state store

```
class ToastColumnValueProvider implements ValueTransformerWithKey<JsonObject, JsonObject, JsonObject>

  private KeyValueStore<JsonObject, String> biographyStore;

  @Override
  public void init(ProcessorContext context) {
    biographyStore = (KeyValueStore<JsonObject, String>) context.getStateStore(
        TopologyProducer.BIOGRAPHY_STORE);
  }

  @Override
  public JsonObject transform(JsonObject key, JsonObject value) {
    // ...
  }
}
```

Expanding Partial Update Events

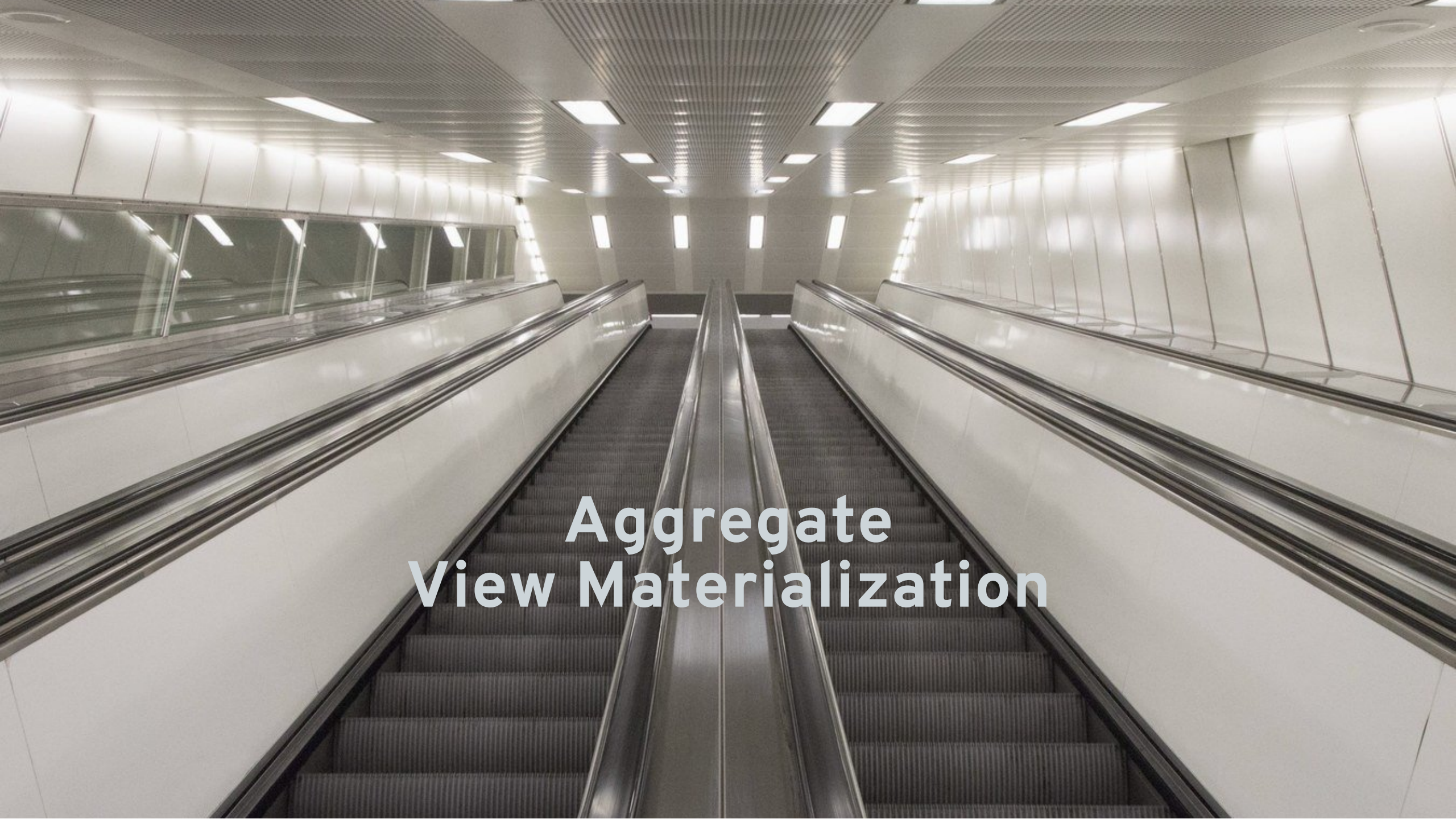
Obtaining missing values from a state store

```
JsonObject payload = value.getJsonObject("payload");
JsonObject newRowState = payload.getJsonObject("after");
String biography = newRowState.getString("biography");

if (isUnavailableValueMarker(biography)) {
  String currentValue = biographyStore.get(key);
  newRowState = Json.createObjectBuilder(newRowState)
    .add("biography", currentValue)
    .build();
  // ...
}
else {
  biographyStore.put(key, biography);
}
return value;
```

Expanding Partial Update Events

Obtaining missing values from a state store

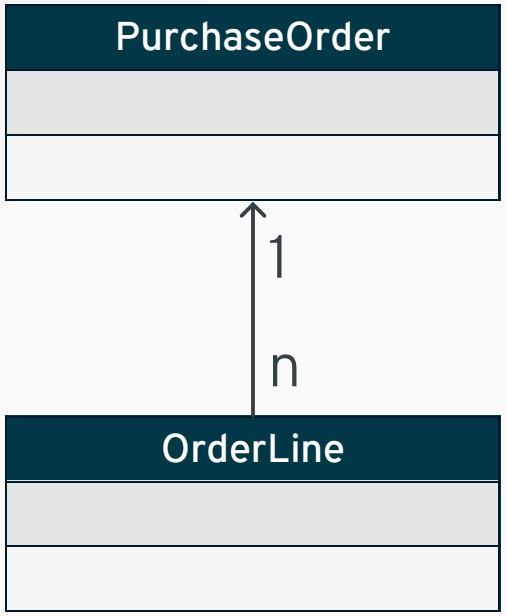
```
JsonObject payload = value.getJsonObject("payload");
JsonObject newRowState = payload.getJsonObject("after");
String biography = newRowState.getString("biography");

if (isUnavailableValueMarker(biography)) {
  String currentValue = biographyStore.get(key);
  newRowState = Json.createObjectBuilder(newRowState)
    .add("biography", currentValue)
    .build();
  // ...
}
else {
  biographyStore.put(key, biography);
}
return value;
```

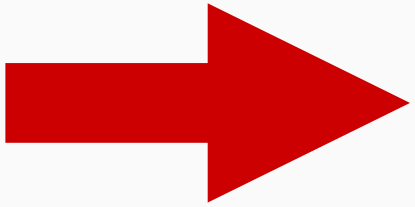


Aggregate View Materialization

Aggregate View Materialization

From Multiple Topics to One View





```
{
  "purchaseOrderId" : "order-123",
  "orderDate" : "2020-08-24",
  "customerId": "customer-123",
  "orderLines" : [
   {
     "orderLineId" : "orderLine-456",
     "productId" : "product-789",
     "quantity" : 2,
     "price" : 59.99
   },
   {
     "orderLineId" : "orderLine-234",
     "productId" : "product-567",
     "quantity" : 1,
     "price" : 14.99
   }
```

Aggregate View Materialization

Non-Key Joins (KIP-213)

```
KTable<Long, OrderLine> orderLines = ...;
KTable<Integer, PurchaseOrder> purchaseOrders = ...;

KTable<Integer, PurchaseOrderWithLines> purchaseOrdersWithOrderLines = orderLines
  .join(
    purchaseOrders,
    orderLine -> orderLine.purchaseOrderId,
    (orderLine, purchaseOrder) -> new OrderLineAndPurchaseOrder(orderLine, purchaseOrder))
  .groupBy(
    (orderLineId, lineAndOrder) -> KeyValue.pair(lineAndOrder.purchaseOrder.id, lineAndOrder))
  .aggregate(
    PurchaseOrderWithLines::new,
    (Integer key, OrderLineAndPurchaseOrder value, PurchaseOrderWithLines agg) -> agg.addLine(value),
    (Integer key, OrderLineAndPurchaseOrder value, PurchaseOrderWithLines agg) -> agg.removeLine(value)
   );
```

Aggregate View Materialization

Non-Key Joins (KIP-213)

```
KTable<Long, OrderLine> orderLines = ...;
KTable<Integer, PurchaseOrder> purchaseOrders = ...;

KTable<Integer, PurchaseOrderWithLines> purchaseOrdersWithOrderLines = orderLines
  .join(
    purchaseOrders,
    orderLine -> orderLine.purchaseOrderId,
    (orderLine, purchaseOrder) -> new OrderLineAndPurchaseOrder(orderLine, purchaseOrder))
  .groupBy(
    (orderLineId, lineAndOrder) -> KeyValue.pair(lineAndOrder.purchaseOrder.id, lineAndOrder))
  .aggregate(
    PurchaseOrderWithLines::new,
    (Integer key, OrderLineAndPurchaseOrder value, PurchaseOrderWithLines agg) -> agg.addLine(value),
    (Integer key, OrderLineAndPurchaseOrder value, PurchaseOrderWithLines agg) -> agg.removeLine(value)
   );
```

Aggregate View Materialization

Non-Key Joins (KIP-213)

```
KTable<Long, OrderLine> orderLines = ...;
KTable<Integer, PurchaseOrder> purchaseOrders = ...;

KTable<Integer, PurchaseOrderWithLines> purchaseOrdersWithOrderLines = orderLines
  .join(
    purchaseOrders,
    orderLine -> orderLine.purchaseOrderId,
    (orderLine, purchaseOrder) -> new OrderLineAndPurchaseOrder(orderLine, purchaseOrder))
  .groupBy(
    (orderLineId, lineAndOrder) -> KeyValue.pair(lineAndOrder.purchaseOrder.id, lineAndOrder))
  .aggregate(
    PurchaseOrderWithLines::new,
    (Integer key, OrderLineAndPurchaseOrder value, PurchaseOrderWithLines agg) -> agg.addLine(value),
    (Integer key, OrderLineAndPurchaseOrder value, PurchaseOrderWithLines agg) -> agg.removeLine(value)
   );
```

Aggregate View Materialization

Non-Key Joins (KIP-213)

```
KTable<Long, OrderLine> orderLines = ...;
KTable<Integer, PurchaseOrder> purchaseOrders = ...;

KTable<Integer, PurchaseOrderWithLines> purchaseOrdersWithOrderLines = orderLines
  .join(
    purchaseOrders,
    orderLine -> orderLine.purchaseOrderId,
    (orderLine, purchaseOrder) -> new OrderLineAndPurchaseOrder(orderLine, purchaseOrder))
  .groupBy(
    (orderLineId, lineAndOrder) -> KeyValue.pair(lineAndOrder.purchaseOrder.id, lineAndOrder))
  .aggregate(
    PurchaseOrderWithLines::new,
    (Integer key, OrderLineAndPurchaseOrder value, PurchaseOrderWithLines agg) -> agg.addLine(value),
    (Integer key, OrderLineAndPurchaseOrder value, PurchaseOrderWithLines agg) -> agg.removeLine(value)
   );
```

Aggregate View Materialization

Non-Key Joins (KIP-213)

```
KTable<Long, OrderLine> orderLines = ...;
KTable<Integer, PurchaseOrder> purchaseOrders = ...;

KTable<Integer, PurchaseOrderWithLines> purchaseOrdersWithOrderLines = orderLines
  .join(
    purchaseOrders,
    orderLine -> orderLine.purchaseOrderId,
    (orderLine, purchaseOrder) -> new OrderLineAndPurchaseOrder(orderLine, purchaseOrder))
  .groupBy(
    (orderLineId, lineAndOrder) -> KeyValue.pair(lineAndOrder.purchaseOrder.id, lineAndOrder))
  .aggregate(
    PurchaseOrderWithLines::new,
    (Integer key, OrderLineAndPurchaseOrder value, PurchaseOrderWithLines agg) -> agg.addLine(value),
    (Integer key, OrderLineAndPurchaseOrder value, PurchaseOrderWithLines agg) -> agg.removeLine(value)
   );
```

Aggregate View Materialization

Awareness of Transaction Boundaries

- TX metadata in change events (e.g. dbserver1.inventory.orderline)

```
{
  "before": null,
  "after": { ... },
  "source": { ... },
  "op": "c",
  "ts_ms": "1580390884335",
  "transaction": {
    "id": "571",
    "total_order": "1",
    "data_collection_order": "1"
  }
}
```

Aggregate View Materialization

Awareness of Transaction Boundaries

- Topic with **BEGIN/END markers**
- Enable consumers to buffer all events of one transaction

```
{
  "transactionId" : "571",
  "eventType" : "begin transaction",
  "ts_ms": 1486500577125
}
```

BEGIN

```
{
  "transactionId" : "571",
  "ts_ms": 1486500577691,
  "eventType" : "end transaction",
  "eventCount" : [
    {
      "name" : "dbserver1.inventory.order",
      "count" : 1
    },
    {
      "name" : "dbserver1.inventory.orderLine",
      "count" : 5
    }
  ]
}
```

END

Bonus:Single Message Transformations

- **Stateless transformations** in Kafka Connect
  - Routing
  - Format conversions (types, masking, names etc.)
  - Filtering
- **In Debezium:** content-based router/filter, outbox router, etc.

```
transforms=route
transforms.route.type=io.debezium.transforms.ContentBasedRouter
transforms.route.language=jsr223.graal.js
transforms.route.topic.expression=value.after.orderType == 'B2B' ? 'b2b_orders' : 'b2c_orders'
```

Takeaways

Debezium and Kafka Streams

- Kafka Streams can take CDC to the next level
- Many use cases
  - Data **enrichment**
  - Creating **aggregated events**
  - **Streaming queries**
  - **Interactive query** services for legacy databases
- Quarkus is the perfect platform

Takeaways

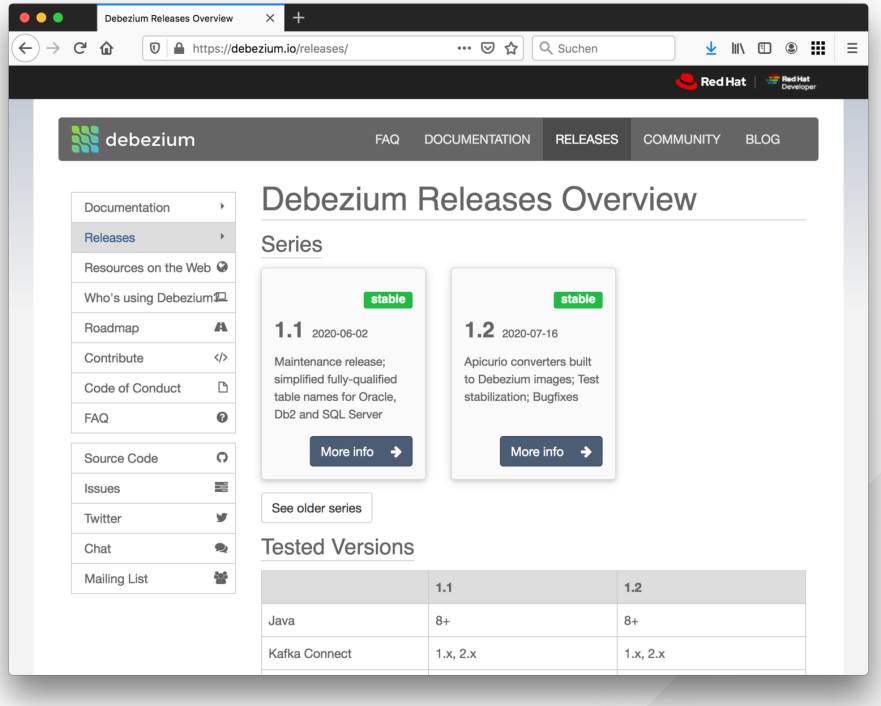
Debezium and Kafka Streams

- Kafka Streams can take CDC to the next level
- Many use cases
  - Data **enrichment**
  - Creating **aggregated events**
  - **Streaming queries**
  - **Interactive query** services for legacy databases
- Quarkus is the perfect platform

Debezium + Kafka Streams = ❤️

Resources

- **Website:** debezium.io
- **Examples:** github.com/debezium/debezium-examples/
- **Latest news:** @debezium



Q&A

gunnar@hibernate.org

@gunnarmorling

Red Hat