striim

Quick Start Tutorial for Streaming Kafka Integration

Striim Technical Guide

MAY 2020

Steve Wilkes, Irem Radzik

Quick Start Tutorial for Streaming Kafka Integration

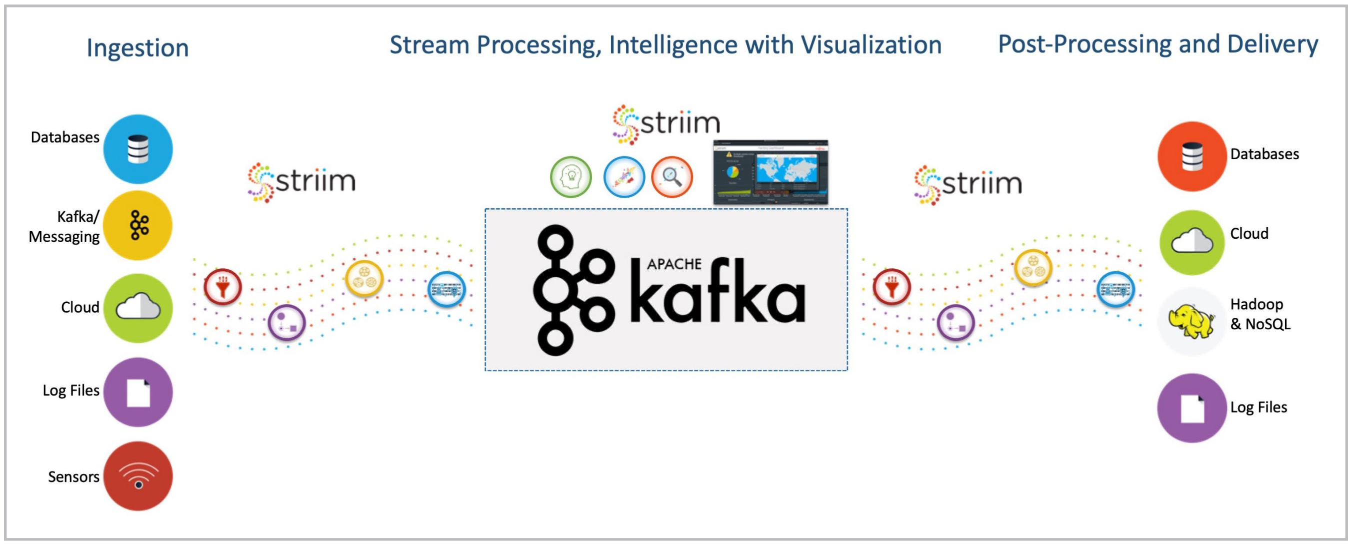
Table of Contents

Introduction

Apache Kafka has seen increasing adoption among enterprises of all sizes as a high-performance, fault-tolerant messaging system. Because it is designed for developers, typically, a team of developers is needed to build, deploy, and maintain any stream processing or analytics applications that use it. At Striim, we believe it should be easy for organizations to make the best use of their messaging system, and people shouldn't have to write Java code to create business solutions around it. The Striim platform makes it easy to stream data into, and out of, Kafka and build analytic applications running on Kafka.

Striim offers a streaming data integration with intelligence software platform that integrates with Kafka as a source and target, and also ships with Kafka built-in. One can optionally start a Kafka cluster when spinning up a Striim cluster, and easily switch between Striim's high-speed in-memory messaging and Kafka. Kafka can become transparent and its capabilities harnessed without having to code to a bunch of APIs.

Striim has offered SQL-query-based processing and analytics for Kafka since 2015. Drag-and-drop UI, pre-built wizards for configuring ingestion into Kafka and custom utilities make Striim the easiest platform to deliver end-to-end streaming integration and analytics applications for Apache Kafka.



Striim offers an end-to-end streaming integration with intelligence platform for Apache Kafka

In this tech guide, we will walk you through Striim's key capabilities and illustrate how they drive value with customer examples. We will also provide step-by-step instructions on how to build an integration solution from a MySQL database to Apache Kafka with in-flight data processing.

Streaming Data Ingestion

When users are considering how to get data into Kafka, they need to determine how to collect source data in a streaming fashion, and how to "massage" and transform that data into the required format on Kafka. Neither of these steps should require any coding yet should be flexible enough to cover a wide range of use cases.

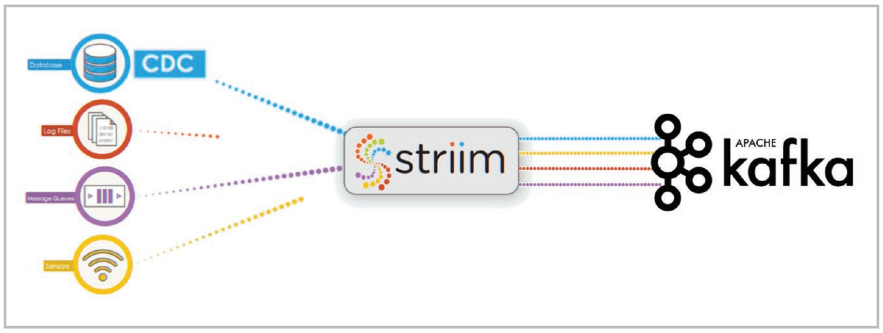
Streaming Data Collection using Log-based CDC

The Striim platform ingests real-time streaming data from a variety of sources out-of-the-box, including databases, files, message queues, and devices. All of these are wrapped in a simple, easy-to-use construct — a data source —

that is configured through a set of properties. This can be done through Striim's SQL-based, TQL scripting language, or the UI. The platform also provides wizards to simplify creating data flows from popular sources to Kafka. The way Striim collects data varies depending on the source, and databases require special treatment.

Databases are often perceived as a record of what has happened in the past, with access to that data through querying. However, that paradigm can change using a technology known as change data capture (CDC). There are different ways change data can be collected from databases. The non-intrusive method uses the transaction log of the database and sees each insert, update and delete as they happen. With this method, Striim can stream out each database operation in real time without impacting the performance of the source database.

Striim takes a similar approach with files. The file reader does not wait for files to be complete before processing them in a batch-oriented fashion. Instead, the reader waits at the end of the file and streams out new data as it is written to the file. As such, it can turn any set of log files into a real-time streaming data source.



Real-time data ingestion into Kafka from many enterprise data sources

Striim ingests data from other sources too, including IoT and device data through TCP/UDP/HTTP/MQTT/AMQP, network information through NetFlow and PCAP, and other message buses such as JMS, MQ Series, and Flume.

The data from all these sources can be delivered "as-is," or go through a series of transformations and enrichments to create exactly the data structure and content needed. Data can even be correlated and joined across sources, before delivery to Kafka.

For easy scalability and high performance in high-volume environments, Striim uses multi-threaded apply with automatic thread management. Striim supports developers with deep visibility into Kafka integration adapters. Organizations can monitor various integration and processing metrics in real time to easily pinpoint any performance bottlenecks.

Data Formatting

Each Kafka consumer can have different requirements for the data format. Since Kafka deals with data at the byte level, it is not aware of the format of data at all, but consumers may need a specific representation — from plain text, or delimited data (think CSVs), to structured XML, JSON or Avro formats.

When writing to Kafka in Striim, users can choose the data format through a simple drop-down list and optional configuration properties, without a single line of code.

Delivering Kafka Data to Enterprise Targets

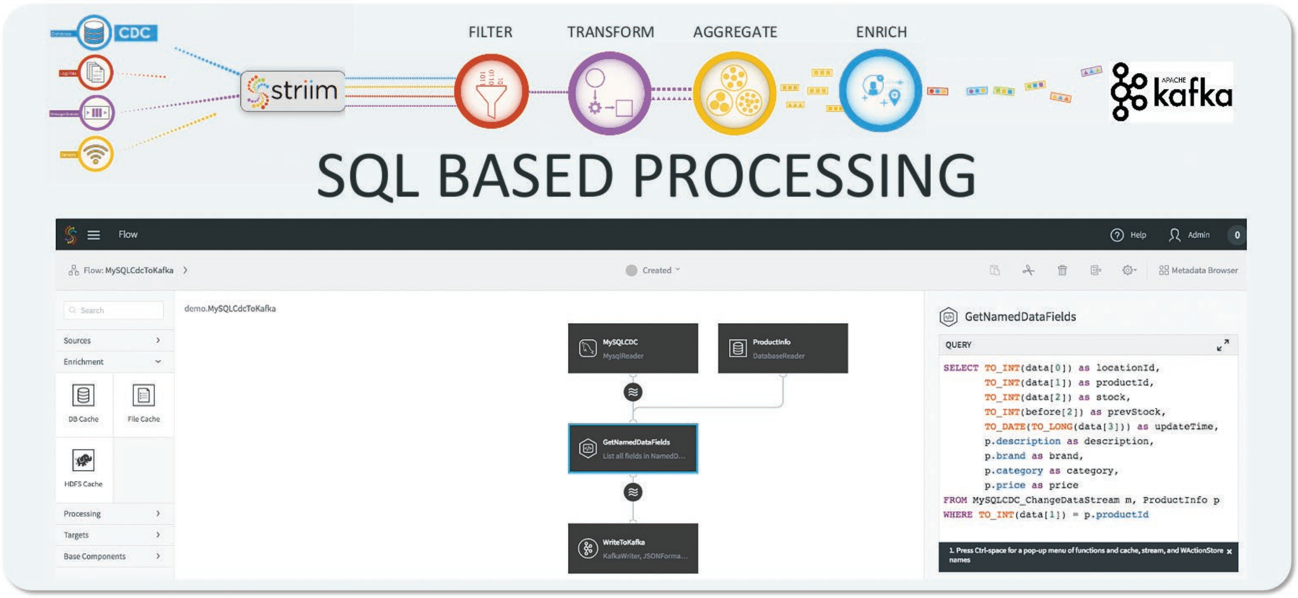
Kafka may be used as the underpinning of stream processing and analytics, or as a data distribution hub. When organizations want to deliver the data from Kafka to somewhere else, as with sourcing data, it should not be difficult or require coding.

The Striim platform can write continuously to a broad range of data targets, including databases, files, message queues, Hadoop environments, and cloud data stores like Azure Blob Storage, Azure SQL Database, Amazon Redshift, and Google BigQuery. The written data format is again configurable and can be applied to the raw Kafka data, or to the results of processing and analytics. This is all achieved through the drag-and-drop UI or scripting language and is simply a matter of choosing the target and configuring properties. When reading from Kafka queues, Striim offers automated mapping of partitions to increase development productivity and accelerate time to market.

A single data flow can write to multiple targets at the same time in real time, with rules encoded as queries in between. This means companies can source data from Kafka and write some of it — or all of it — to HDFS, Azure SQL Database, and the enterprise data warehouse simultaneously.

SQL-Based Stream Processing for Kafka Data

When delivering data to Kafka or writing Kafka data to a downstream target like HDFS, it is essential to consider the structure and content of the data that is being written. Based on the use case, organizations may not require all of the data, only that which matches certain criteria. Users may also need to transform the data through string manipulation or data conversion, or only send aggregates to prevent data overload.



```
SELECT TO_INT(data[0]) as locationId,
       TO_INT(data[1]) as productId,
       TO_INT(data[2]) as stock,
       TO_INT(before[2]) as prevStock,
       TO_DATE(TO_LONG(data[3])) as updateTime,
       p.description as description,
       p.brand as brand,
       p.category as category,
       p.price as price
FROM MySQLCDC_ChangeDataStream m, ProductInfo p
WHERE TO_INT(data[1]) = p.productId
```

1. Press Ctrl-space for a pop-up menu of functions and cache, stream, and WActionStore names

In-stream data processing and preparation for Kafka

Most commonly, users may need to add additional context to the data. A lot of raw data may need to be joined with reference data to make it useful.

In a scenario where CDC is streaming changes from a normalized database, if the database has been designed properly, most of the data fields will be in the form of IDs. This is very efficient for the database, but not very useful for downstream queries or analytics. IoT data can present a similar situation, with device data consisting of a device

ID and a few values, without any meaning or context. In both cases, users may want to enrich the raw data with reference data, correlated by the IDs, to produce a denormalized record with sufficient information.

The key tenets of streaming integration and stream processing — filtering, transformation, aggregation and enrichment — are essential to any data architecture and should be easy to apply to Kafka data without any need for developers or complex APIs.

The Striim platform uses a uniform approach utilizing in-memory continuous queries, with all of the stream processing expressed in a SQL-like language. Anyone with any data background understands SQL, so the constructs are incredibly familiar. Transformations are simple and can utilize both built-in and Java functions, CASE statements and other mechanisms. Filtering is just a WHERE clause.

Aggregations can utilize flexible windows that turn unbounded infinite data streams into continuously changing bounded sets of data. The queries can reference these windows and output data continuously as the windows change. This means a one-minute moving average is just an average function over a one-minute sliding window.

Enrichment requires external data, which is introduced into the Striim platform through the use of distributed caches (otherwise known as a Data Grid). Caches can be loaded with large amounts of reference data, which is stored in-memory across the cluster. Queries can reference caches in a FROM clause the same way as they reference streams or windows, so joining against a cache is simply a JOIN in a query.

Multiple stream sources, windows and caches can be used and combined together in a single query, and queries can be chained together in directed graphs, known as data flows. All of this can be built through the UI or Striim's scripting language and can be easily deployed and scaled across a Striim cluster, without having to write any code.

Streaming Analytics and Data Visualization for Kafka

When the goal is to gain intelligence from the streaming data running in Kafka, delivering that data to Hadoop or a database for analytics is not effective, as it introduces latency. As a result, this solution architecture diminishes the value users get in operational use cases. To support time-sensitive decision making, analytics needs to be done in-memory, as the data is flowing through, surfacing the results of the analytics through visualizations in a dashboard.

Analytics for data in Kafka can involve correlation of data across data streams, looking for patterns or anomalies, making predictions, understanding behavior, or simply visualizing data in a way that makes it interactive and interrogable.

The Striim platform enables Kafka users to perform analytics in-memory, in the same way as they do processing — through SQL-based continuous queries. These queries can join data streams together to perform correlation and look for patterns (or specific sequences of events over time) across one or more data streams utilizing an extensive pattern-matching syntax.

Continuous statistical functions and conditional logic enable anomaly detection, while built-in regression algorithms enable predictions into the future based on current events.

Surface the results of the analytics through visualizations in a dashboard

Analytics can also be rooted in understanding large datasets. Striim customers have used the platform to both train their models and to perform real-time inference and scoring based on existing machine learning models by integrating them into data flows. Streaming data integration with intelligence boosts machine learning initiatives in multiple ways:

- Firstly, users can ingest and pre-process data from Kafka (and other sources) to a desired format to easily extract features for machine learning before continuously loading the data to the analytics environment.
- Secondly, once a model has been constructed and exported, users can easily call the model from Striim's open processor, passing real-time data into it, to infer outcomes continuously.
- Striim can monitor the data evolution and model performance, and trigger model retraining if the model no longer fits the data.
- Finally, using Striim's real-time dashboards, users can see the real-time data and the predictions, with immediate alerts as configured.

The end result is a faster time-to-market for ML solutions and the ability to operationalize the models to real-time business decision making.

Visualizing and interacting with data is a critical component of analytics efforts, as well. The Striim platform UI includes a complete Dashboard Builder that enables rapid development of custom, use-case-specific dashboards and effectively highlights real-time data and the results of analytics. With a rich set of visualizations, and simple query-based integration with analytics results, dashboards can be configured to continually update and enable drill-down and in-page filtering.

Via the interactive, live dashboards, Kafka users can compare live data to historical averages or to a specific date and time in the past, without having to write code. Users can view live data with detailed field and time-based filtering at the page or chart level. In addition, users can search streaming data directly on the dashboard and drill down to detail pages. Striim's charts can be embedded to any custom dashboard or web page to support broad collaboration and distribution of real-time insights.

Delivering an Enterprise-Grade Solution

When Kafka supports business-critical applications and services, it is crucial to ensure what feeds data into Kafka and processes Kafka's data is secure, scalable, and reliable. In the solution components we described earlier, there are multiple major pieces of in-memory technology that have to be integrated seamlessly and tuned in order to be enterprise grade. This means organizations have to consider the scalability, reliability and security of the complete end-to-end architecture with the integrated components, not just their Kafka solution.

Joining streaming data with data cached in an in-memory data grid, for example, requires careful architectural consideration to ensure all pieces run in the same memory space, and joins can be performed without expensive and time-consuming remote calls. Continually processing and analyzing hundreds of thousands, or millions, of events per second across a cluster in a reliable fashion is not a simple task, and can take many years of development time.

The Striim platform has been architected from the ground up to scale and deliver data with security and reliability.

1. **Secure** with built-in authentication, authorization, protection and encryption. Applications can be secured easily through roles, while specific items can be locked down to the stream level using granular security rules. All data can be encrypted in-flight and files can be delivered in encrypted forms using customer provided keys.

2. **High performance and highly scalable** with a distributed, modern architecture that combines highly optimized data serialization and windowing with in-memory computing. Striim enables linear scale-out using a low-cost compute infrastructure to support extreme and varied data volumes and velocity.

3. **Reliable** with an inherently clustered, fault-tolerant architecture and "exactly once" processing. Recovery ensures no events are missed or processed twice, and takes window contents into account.

Customer Examples

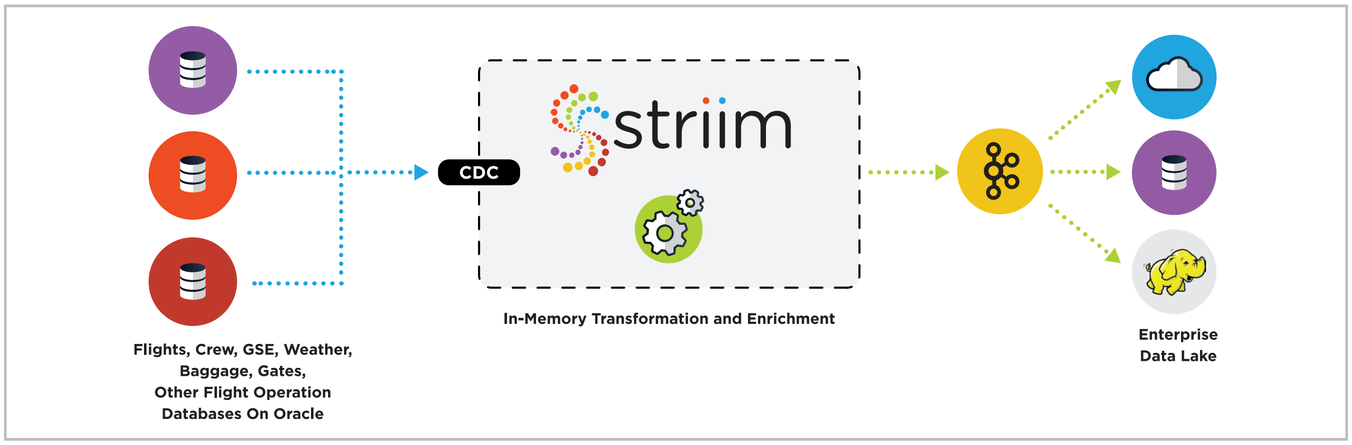
By easily moving high volumes of data to and from Kafka in real-time, Striim enables organizations across a diverse range of industries to gain current and relevant information about their business. Here are a few examples how Striim customers use the technology for strategic initiatives and mission-critical environments:

A global airline wanted to quickly detect events that may impact flight and crew schedules, and optimize schedules based on real-time business and weather events. Striim ingests real-time operational data from the airline's Oracle databases and performs in-memory transformations to deliver the data in a consumable format to the Kafka messaging system in sub-seconds. Using Striim to continuously update its data lake, the airline now has multiple, live KPI dashboards to track different operational stations' progress status.

- Flight Operations teams can make immediate tactical decisions such as speeding the de-icing process by requesting another equipment, assigning a different aircraft, reassigning passengers to other flights and gates, and improving on-time departures.

- Timely decision making for crew scheduling also allows the airlines to comply with government regulations and crew union rules for pilots' and flight attendant's total duty periods and minimizes incidents that lead to paying fines.

- The airline can analyze past decisions and create operational models to better handle future operational emergencies and irregularities.


CDC
striim
In-Memory Transformation and Enrichment
Flights, Crew, GSE, Weather, Baggage, Gates, Other Flight Operation Databases On Oracle
Enterprise Data Lake

Global airlines streams real-time data to Kafka from all critical flight operation systems

A leading European satellite TV provider wanted to improve its customer analytics by using a modern data integration solution that can ingest customer data from multiple different sources in real time. Striim integrates its CRM systems with the analytics solutions using real-time, continuous data pipelines. Striim also performs in-flight, rule-based transformations to prepare the data for analytics. Using Striim, the company now has the following benefits:

- Timely customer behavior analytics using comprehensive and real-time customer data
- The ability to reduce churn and improve customer satisfaction with timely intelligence
- Fast time-to-insight by using pre-processed data.

A large, U.S.-based health insurance company, wanted to gain the agility to transform its business operations. It replaced a brittle, point-to-point application integration architecture with an event-driven application to distribute accurate and timely data across the enterprise. Striim's streaming data integration software was selected for its ability to securely deliver the claims data with very low latency even in high data volume environments. Striim ingests real-time claims data from the claims management system, encrypts and transforms the data in-flight before delivering to an event-driven data hub on Apache Kafka, Azure SQL Database, and Azure Cosmos DB. The data hub solution helps the company:

- Have confidence in the data they share across the enterprise and easily implement new applications and operational changes to improve member services.
- Reduce the cost of change while increasing the productivity of both IT and business teams.
- Comply with security and privacy regulations by having a centralized place to measure data governance and privacy for information assets.

A provider of financial technology applications that offers customers a centralized way to manage their financial tasks, needed to standardize and scale data services for a growing ecosystem of high-volume Oracle databases, microservice applications, reporting, and analytics solutions. Using Striim's CDC capabilities, the company can collect real-time banking data, customer data, and POS data from Oracle databases and continuously stream transactions to Apache Kafka. The developers, DBAs, data architects, and data scientists have increased operational efficiency and reduced the cost of their modern data architecture while laying a foundation to meet the increasing complexity of their customer expectations. Application services receive data in flexible formats, with any changes made upstream at the database level immediately reflected downstream in the data formats delivered to the Kafka messaging infrastructure. Striim's ability to deliver the unique combination of flexible continuous data flows, enterprise-grade scalable clusters, and schema-evolved transactional change data capture has enabled the company to:

- Quickly reap six-figure cost savings over their previous change data capture solution.
- Solve their data warehousing needs using the streaming data pipelines already in place to implement the Kudu integration on a real-time, continuous basis.
- Increase the operational efficiency of the development teams and the availability of the applications.

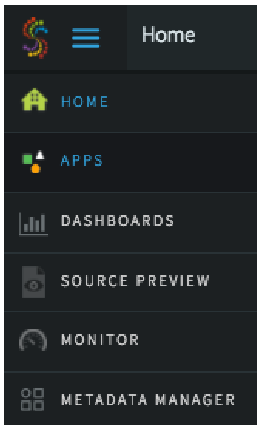
How to Use Striim: Tutorial for Change Data Streaming to Kafka from MySQL Database

Striim's value to Kafka users is not limited to offering enterprise-grade streaming data integration features. With an intuitive, drag-and-drop UI, Striim also makes it fast and easy to build integration and analytics solutions for Kafka. In this section, we will provide you the step-by-step instructions on how to stream transactions from a MySQL database into Apache Kafka using log-based CDC and performing in-flight data processing.

Using Striim's CDC for Ingestion from MySQL Database

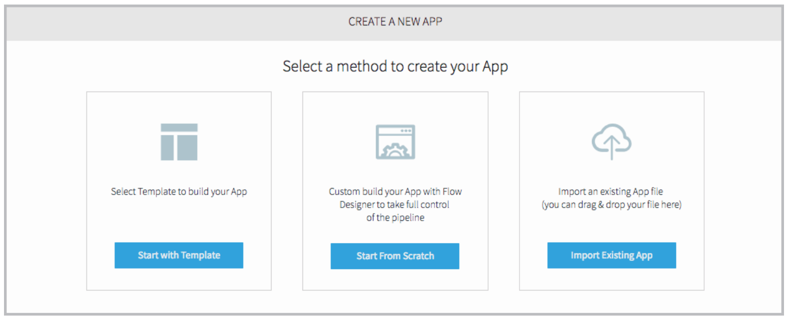
The first step in any streaming integration is sourcing data. We will start with using change data capture (CDC) to stream database DML activity (inserts, updates and deletes) from a MySQL database. The same process would be used to read from Oracle, SQL Server, and MariaDB, but MySQL was chosen so you can easily try this at home using a free trial version of Striim.

Using the Striim Application Templates



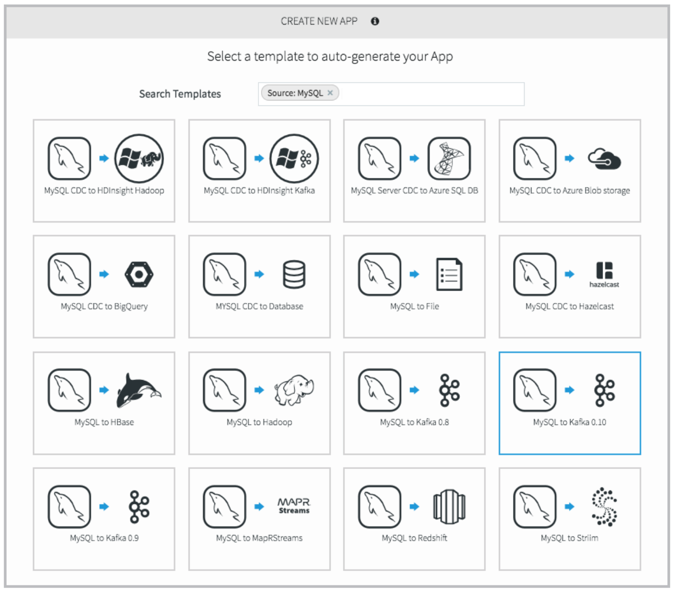
To start building the CDC application, you need to navigate to Apps in either the drop-down menu, or home screen. Once there, click on the Apps link to view all previously created data flow applications. This may be empty if you are first getting started with Striim. To create the application, click "Add App" in the top right-hand corner. This will give you three options for creating a new application:

- Start with Template
- Start from Scratch
- Import Existing App


CREATE A NEW APP
Select a method to create your App
Select Template to build your App
Start with Template
Custom build your App with Flow Designer to take full control of the pipeline
Start From Scratch
Import an existing App file (you can drag & drop your file here)
Import Existing App

The Different Options for Creating Applications Using Striim

In this case we are going to start with a template. There are a lot of templates for many combinations of sources and targets. You can narrow the search by entering terms into the Search Templates box.


CREATE NEW APP

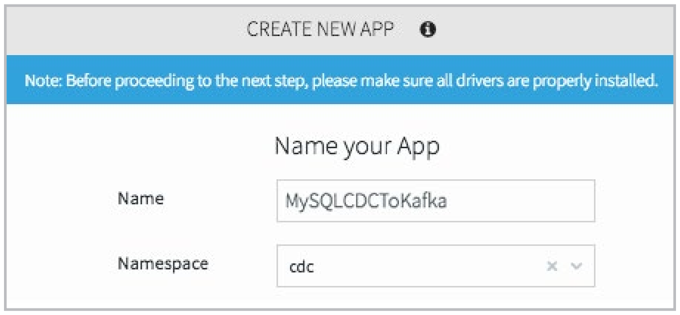
Select a template to auto-generate your App

Search Templates: Source: MySQL

Striim Provides Many Pre-defined Application Templates

Since we are sourcing from MySQL, you can narrow the search using MySQL. Our first dataflow will write into Kafka, so select the appropriate Kafka version as a target. You are now able to name your application. Within Striim applications have fully qualified names that enable the same name (for different departments, or users) to exist in different namespaces. A namespace is really just a way to keep related things together.

This application is MySQLCDCToKafka, and we'll put it in the 'cdc' namespace.


CREATE NEW APP

Note: Before proceeding to the next step, please make sure all drivers are properly installed.

Name your App

Name: MySQLCDCToKafka

Namespace: cdc

Creating a New Application Using Striim

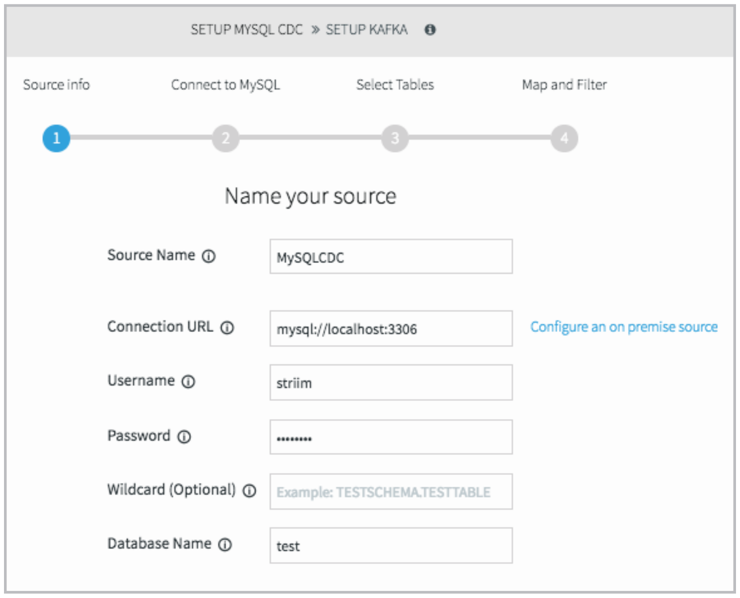
When you click "Save," the application is created, and the wizard will walk you through the steps of the process:

1. Enter the Data Source info
2. Test the connection and CDC capabilities
3. Select the required tables (Do optional mapping and filtering of data)
4. Enter the Target info

You can move forwards and backwards through these steps as necessary if you need to make any corrections.

Enter the Data Source Info

The first step is to enter the required information to connect to the data source. This will vary by source, but you will always need to choose a unique (within the namespace) data source name. In the case of MySQL, you need to enter a connection URL, username, password, and database name.



Name your source

Source Name	MySQLCDC
Connection URL	mysql://localhost:3306
Username	striim
Password	••••••••
Wildcard (Optional)	Example: TESTSCHEMA.TESTTABLE
Database Name	test

The First Step in the Wizard is Configuring the Data Source

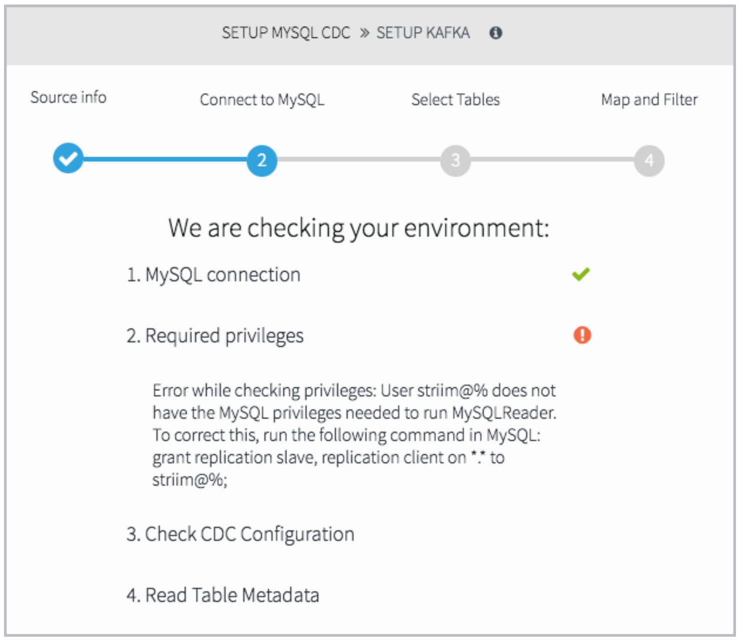
Note: If you are using Striim in the cloud and have configured an on-premise agent to collect data to send to the cloud, you will need to click on 'configure an on-premise source'. Don't worry about the password, it is automatically encrypted by our server.

To validate the information and move to the next step, click on the "Next" button at the bottom right.

Testing the Connection

The Striim platform not only verifies that the database connection information you entered is correct, but also checks that the database user has the correct privileges, and that CDC is setup correctly. This step varies from data source to data source. If anything is wrong, you will be notified and told how to fix the problem.

For example, when Next was clicked previously, the 'striim' user did not have the correct privileges to receive CDC information from the MySQL database. And the results of step two were as follows.


SETUP MYSQL CDC » SETUP KAFKA

We are checking your environment:

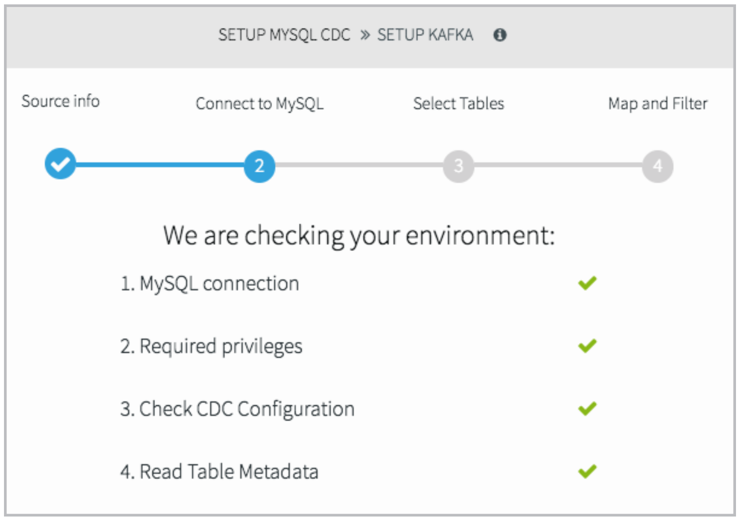
1. MySQL connection ✓
2. Required privileges !

Error while checking privileges: User striim@% does not have the MySQL privileges needed to run MySQLReader. To correct this, run the following command in MySQL: grant replication slave, replication client on *.* to striim@%;

3. Check CDC Configuration
4. Read Table Metadata

The Second Step in the Wizard is Testing the Connection and Showing How to Fix Issues

As you can see, the explanation includes the steps required to fix the issue — an additional two permissions were needed by the 'striim' user to obtain the CDC information. Once 'Replication Client' and 'Replication Slave' privileges were granted, the UI permits you to move to the next step.


SETUP MYSQL CDC » SETUP KAFKA

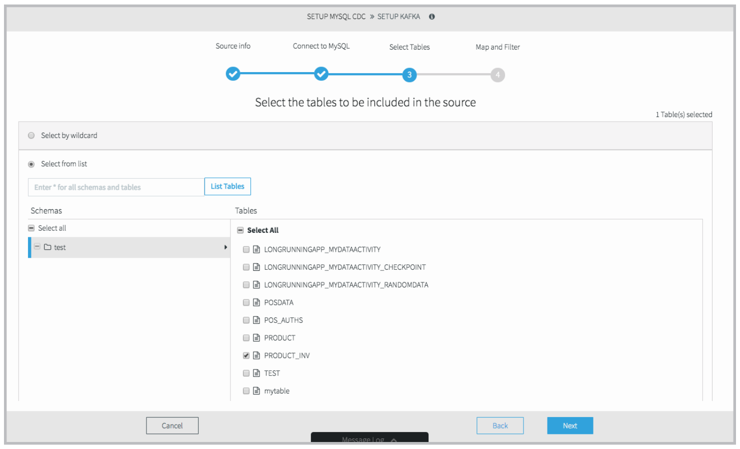
We are checking your environment:

1. MySQL connection ✓
2. Required privileges ✓
3. Check CDC Configuration ✓
4. Read Table Metadata ✓

The Second Step in the Wizard Once All Issues Are Removed

Select the Required Tables

The Striim platform enables you to selectively choose which database schemas and tables to use CDC for. These can be chosen from a list, or through wildcard selection.



Select the tables to be included in the source

1 Table(s) selected

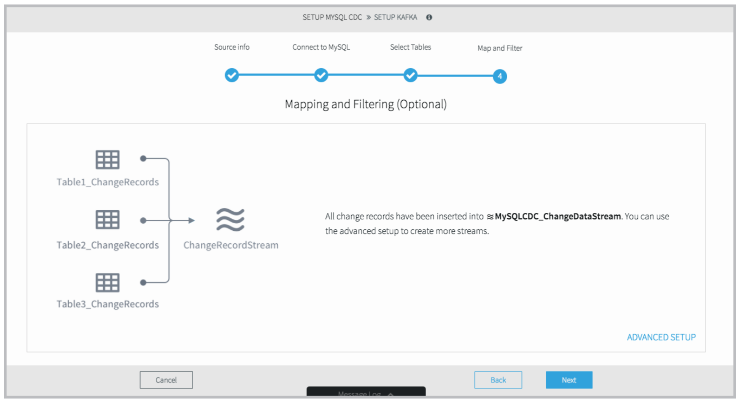
- ○ Select by wildcard
- ● Select from list

Enter * for all schemas and tables | List Tables

Schemas	Tables
Select all	Select All
test	☐ LONGRUNNINGAPP_MYDATAACTIVITY
	☐ LONGRUNNINGAPP_MYDATAACTIVITY_CHECKPOINT
	☐ LONGRUNNINGAPP_MYDATAACTIVITY_RANDOMDATA
	☐ POSDATA
	☐ POS_AUTHS
	☐ PRODUCT
	☑ PRODUCT_INV
	☐ TEST
	☐ mytable

The Third Step in the Wizard is Selection Tables for CDC

In this case we are just going to select the PRODUCT_INV table in the test database schema. Clicking Next takes us to an optional screen where you can do more complex data selection.



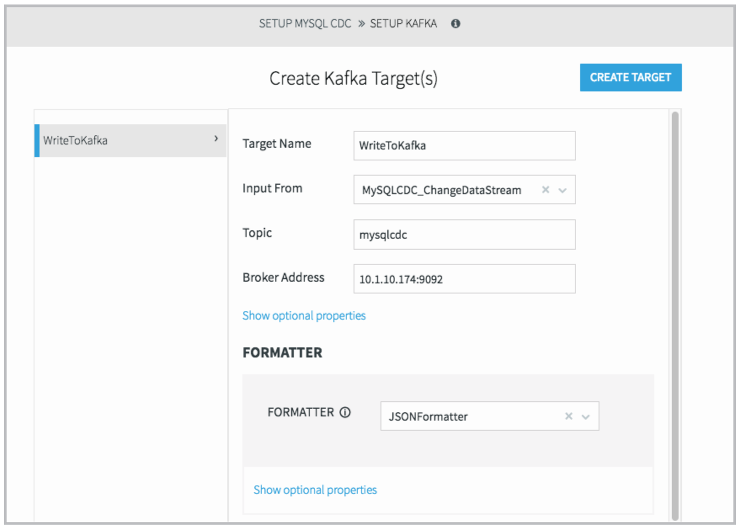
Mapping and Filtering (Optional)

All change records have been inserted into ≋ **MySQLCDC_ChangeDataStream**. You can use the advanced setup to create more streams.

You Can Optionally Do Advanced Data Mapping, if Necessary

Enter the Target Info

The final step is to specify how we want to write data to the target. When we selected the wizard, we chose to write data to Apache Kafka. To complete this step, we simply need to enter a few connection properties including the target name, topic, and broker URL.



Target Name: WriteToKafka
Input From: MySQLCDC_ChangeDataStream
Topic: mysqlcdc
Broker Address: 10.1.10.174:9092
FORMATTER: JSONFormatter

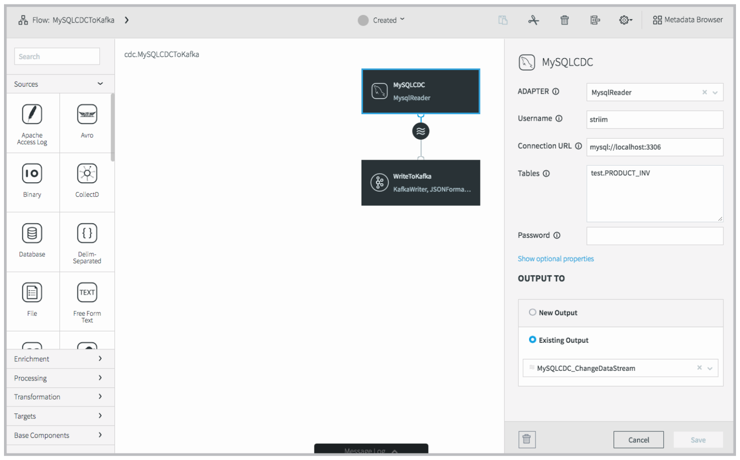
The Final Wizard Step is Configuring the Target

The final step, before clicking "Next" one last time, is to choose the format of the output data. The Striim platform supports JSON, Delimited, XML, Avro and free text formats for Kafka topics. In this case we are selecting the JSONFormatter. This has a number of optional properties, but we are just going to go with the defaults.

Using the wizard, we quickly built an integration application that collects data continuously from MySQL using change data capture, and writes that in real time to Apache Kafka in JSON format. After clicking "Next" you are shown the data flow, and can now edit, run and test the data flow as necessary.

Deploying and Starting the Data Flow

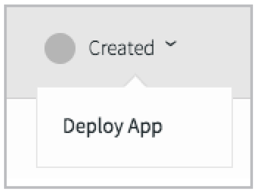
The resulting data flow can now be modified, deployed and started through the UI.



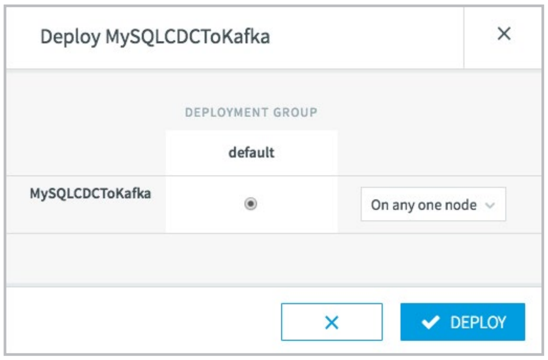
ADAPTER: MysqlReader
Username: striim
Connection URL: mysql://localhost:3306
Tables: test.PRODUCT_INV
Password:
OUTPUT TO: New Output [✓ Existing Output]
MySQLCDC_ChangeDataStream

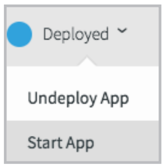
The Wizard Generates an End-to-End Data Flow

This is achieved through the state control button directly above the data flow. The initial state of the data flow is 'created'. This means it is just a definition of what the application is supposed to do. In order to turn this into runtime objects, the application needs to be deployed.



Click on the 'Created' dropdown and select 'Deploy App' to show the Deploy UI.

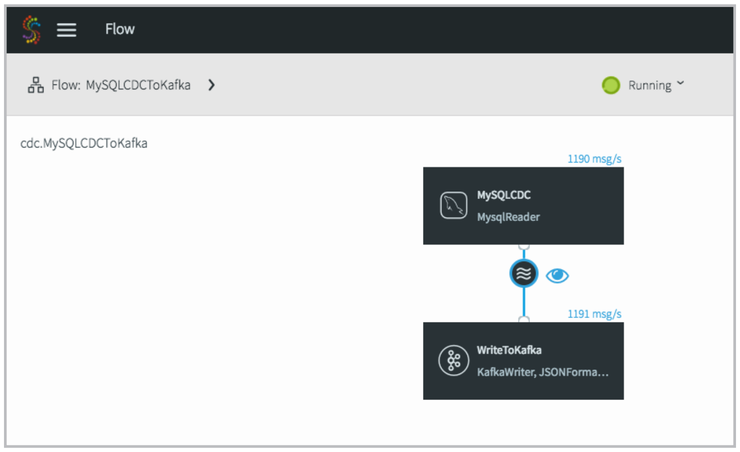




Striim data flows can be deployed very flexibly. The default is to deploy the data flow to the least used node in a Striim cluster. However, the application can be deployed to all, or just some of the nodes (in predefined groups) as necessary. You can even split applications into sub-flows and deploy pieces on different parts of the cluster if required. In this case it's easy, we'll just deploy this whole application to one node. After deployment the application is ready to start, by selecting Start App.

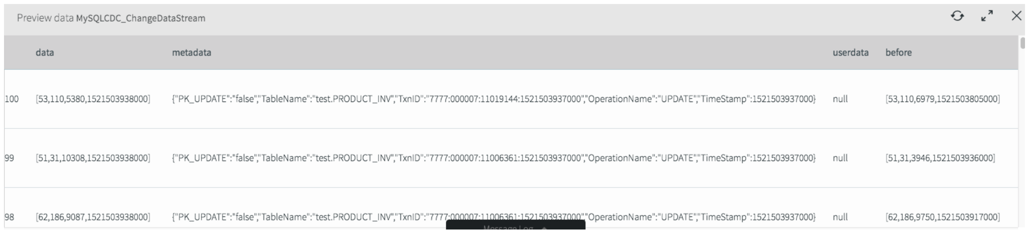
Testing the Data Flow

Assuming you have activity on the MySQL table selected in during table selection, you should see data flowing in the UI, indicated by a number of msgs/s.



Testing the Streaming Data Flow

If you now click on the data stream in the middle and click on the eye icon, you can preview the data flowing between MySQL and Kafka.

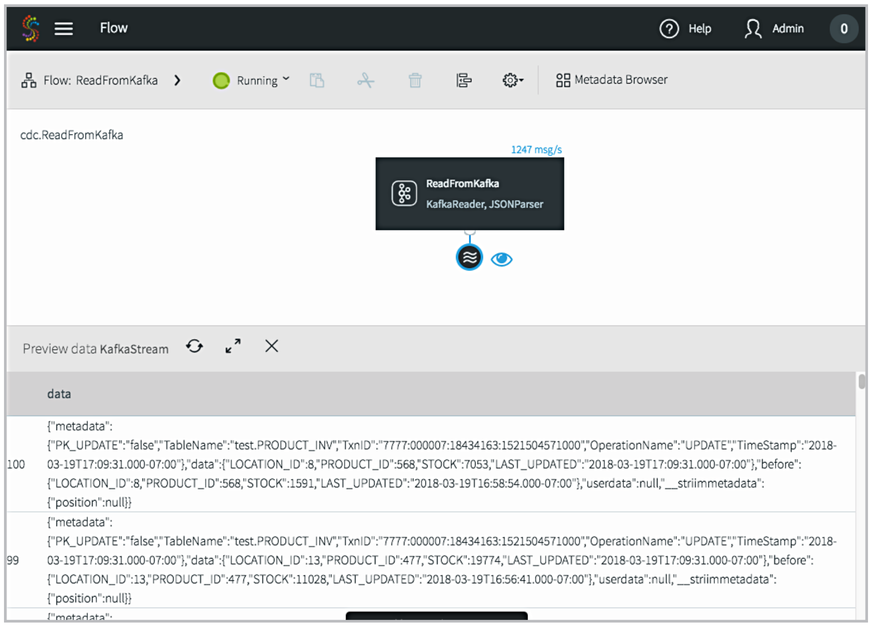


	data	metadata	userdata	before
100	[53,110,5380,1521503938000]	{"PK_UPDATE":"false","TableName":"test.PRODUCT_INV","TxnID":"7777:000007:11019144:1521503937000","OperationName":"UPDATE","TimeStamp":1521503937000}	null	[53,110,6979,1521503805000]
99	[51,31,10308,1521503938000]	{"PK_UPDATE":"false","TableName":"test.PRODUCT_INV","TxnID":"7777:000007:11006361:1521503937000","OperationName":"UPDATE","TimeStamp":1521503937000}	null	[51,31,3946,1521503936000]
98	[62,186,9087,1521503938000]	{"PK_UPDATE":"false","TableName":"test.PRODUCT_INV","TxnID":"7777:000007:11006361:1521503937000","OperationName":"UPDATE","TimeStamp":1521503937000}	null	[62,186,9750,1521503917000]

Previewing the Data Flowing from MySQL to Kafka

Here you can see the data, metadata (these are all updates) and before values (what the data was before the update).

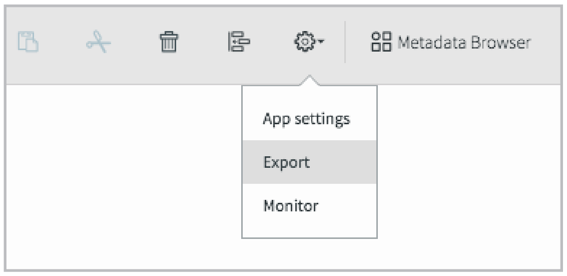
You can write a Kafka Consumer to see what this looks like in JSON format or use a simple Striim application to read from Kafka, then use the built-in preview mechanism to see what it looks like.



	data
100	{"metadata": {"PK_UPDATE":"false","TableName":"test.PRODUCT_INV","TxnID":"7777:000007:18434163:1521504571000","OperationName":"UPDATE","TimeStamp":"2018-03-19T17:09:31.000-07:00"},"data":{"LOCATION_ID":8,"PRODUCT_ID":568,"STOCK":7053,"LAST_UPDATED":"2018-03-19T17:09:31.000-07:00"},"before": {"LOCATION_ID":8,"PRODUCT_ID":568,"STOCK":1591,"LAST_UPDATED":"2018-03-19T16:58:54.000-07:00"},"userdata":null,"__striimmetadata": {"position":null}}
99	{"metadata": {"PK_UPDATE":"false","TableName":"test.PRODUCT_INV","TxnID":"7777:000007:18434163:1521504571000","OperationName":"UPDATE","TimeStamp":"2018-03-19T17:09:31.000-07:00"},"data":{"LOCATION_ID":13,"PRODUCT_ID":477,"STOCK":19774,"LAST_UPDATED":"2018-03-19T17:09:31.000-07:00"},"before": {"LOCATION_ID":13,"PRODUCT_ID":477,"STOCK":11028,"LAST_UPDATED":"2018-03-19T16:56:41.000-07:00"},"userdata":null,"__striimmetadata": {"position":null}}
	{"metadata":

Testing the Message Format Using a Striim Kafka Reader

Exporting as A Script



With the Striim Platform you are not forced to use the UI, even though it is an incredibly fast and convenient way of doing things). You can instead use our scripting language.

Any streaming data flow application can be exported as a TQL script, and any TQL script can be imported then worked with in the UI.

If we go back to the original MySQLCDCToKafka application and click on the configuration dropdown on the top right of the toolbar, you can export it.

The script for the application looks like this:

```
CREATE APPLICATION MySQLCDCToKafka;
CREATE SOURCE MySQLCDC USING MysqlReader  (
  Username: ‘striim’,
  Password: ‘Vx/QMzVeRTMY1rxZH+zycQ==’,
  ConnectionURL: ‘mysql://localhost:3306’,
  Tables: ‘test.PRODUCT_INV’,
  DatabaseName: ‘test’,
  Password_encrypted: true
 )
OUTPUT TO MySQLCDC_ChangeDataStream ;
CREATE TARGET WriteToKafka USING KafkaWriter VERSION ‘0.8.0’ (
  Mode: ‘Sync’,
  Topic: ‘mysqlcdc’,
  brokerAddress: ‘localhost:9092’
 )
FORMAT USING JSONFormatter  (
 )
INPUT FROM MySQLCDC_ChangeDataStream;
END APPLICATION MySQLCDCToKafka;
```

Transforming and Enriching Change Data

What we set up so far is the real-time data ingestion from a MySQL database and its delivery to Kafka. In this section, we are going to process and enrich data-in-motion using continuous queries written in Striim’s SQL-based stream processing language before delivering to Kafka. Using a SQL-based language is intuitive for data processing tasks, and most common SQL constructs can be utilized in a streaming environment. The main differences between using SQL for stream processing, and its more traditional use as a database query language, are that all processing is in-memory, and data is processed continuously, such that every event on an input data stream to a query can result in an output.

The first thing we are going to do with the data is extract fields we are interested in and turn the hierarchical input data into something we can work with more easily.

Transforming Streaming Data With SQL

You may recall the data we saw in the previous section looked like this:

data	before	metadata
[86,466,869,1522184531000]	[86,466,918,1522183459000]	{ “PK_UPDATE”:”false”, “TableName”:”test.PRODUCT_INV”, “TxnID”:”7777:000009:48657361:1522184531000”, “OperationName”:”UPDATE”, “TimeStamp”:1522184531000 }

This is the structure of our generic CDC streams. Since a single stream can contain data from multiple tables, the column values are presented as arrays which can vary in size. Information regarding the data is contained in the metadata, including the table name and operation type.

The PRODUCT_INV table in MySQL has the following structure:

LOCATION_ID int(11) PK

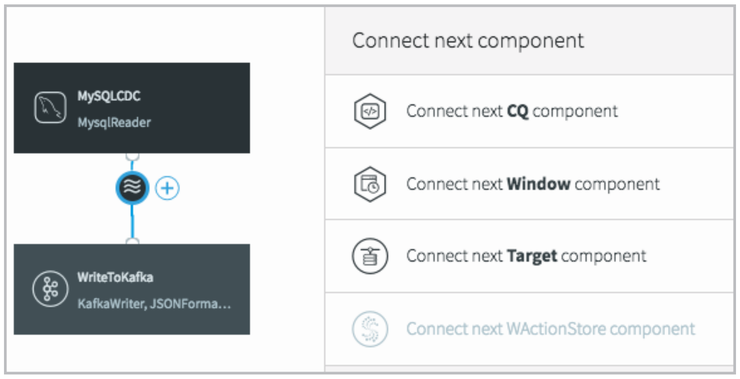
PRODUCT_ID int(11) PK

STOCK int(11)

LAST_UPDATED timestamp

The first step in our processing is to extract the data we want. In this case, we only want updates, and we're going to include both the *before* and *after* images of the update for stock values.

To do the processing, we need to add a continuous query (CQ) into our dataflow. This can be achieved in a number of ways in the UI, but we will click on the datastream, then on the plus (+) button, and select "Connect next **CQ** component" from the menu.


MySQLCDC
MysqlReader
WriteToKafka
KafkaWriter, JSONForma...
Connect next component
Connect next **CQ** component
Connect next **Window** component
Connect next **Target** component
Connect next WActionStore component

Connect Next CQ Component to Add to Our First Continuous Query

As with all components in Striim, we need to give the CQ a name, so let's call it "ExtractFields". The processing query defaults to selecting everything from the stream we were working with.

```
SELECT * FROM MySQLCDC_ChangeDataStream m;
```

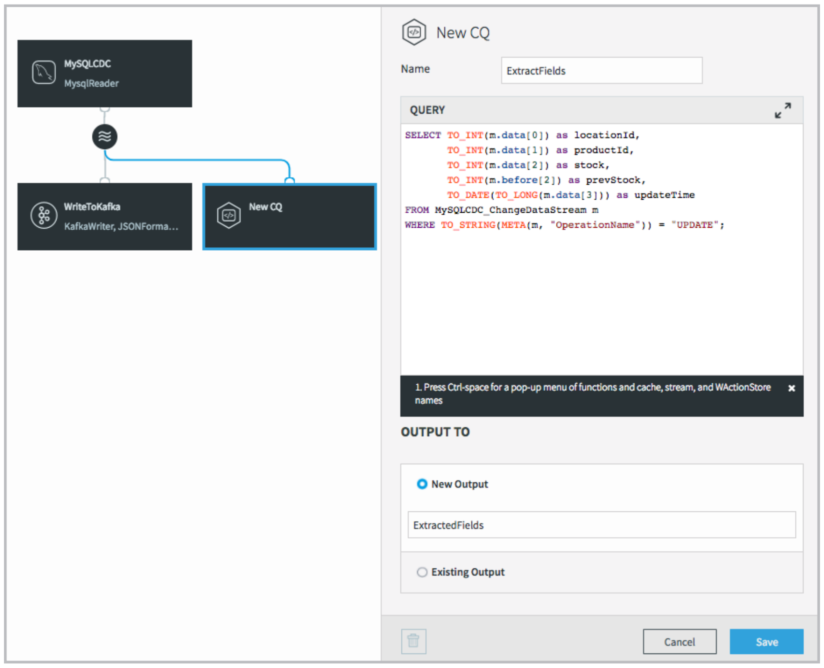
But we want only certain data, and to restrict things to updates. When selecting the data we want, we can apply transformations to convert data types, access metadata, and many other data manipulation functions. This is the query we will use to process the incoming data stream:

```
SELECT TO_INT (m. data [0]) as locationId,
       TO_INT (m. data [1]) as productId,
       TO_INT (m. data [2]) as stock,
       TO_INT (m. before [2]) as prevStock,
       TO_DATE ( TO_LONG (m. data [3]) as updateTime
FROM MySQLCDC_ChangeDataStream m
WHERE TO_STRING ( META (m,  "OperationName" )) =  "UPDATE" ;
```

Notice the use of the data array (what the data looks like after the update) in most of the selected values, but the use of the before array to obtain the prevStock.

We are also using the metadata extraction function (META) to obtain the operation name from the metadata section of the stream, and a number of type conversion functions (TO_INT for example) to force data to be of the correct data types. The date is actually being converted from a LONG timestamp representing milliseconds since the EPOCH.

The final step before we can save this CQ is to choose an output stream. In this case we want a new stream, so we'll call it "ExtractedFields".



Name: ExtractFields

```
SELECT TO_INT(m.data[0]) as locationId,
       TO_INT(m.data[1]) as productId,
       TO_INT(m.data[2]) as stock,
       TO_INT(m.before[2]) as prevStock,
       TO_DATE(TO_LONG(m.data[3])) as updateTime
FROM MySQLCDC_ChangeDataStream m
WHERE TO_STRING(META(m, "OperationName")) = "UPDATE";
```

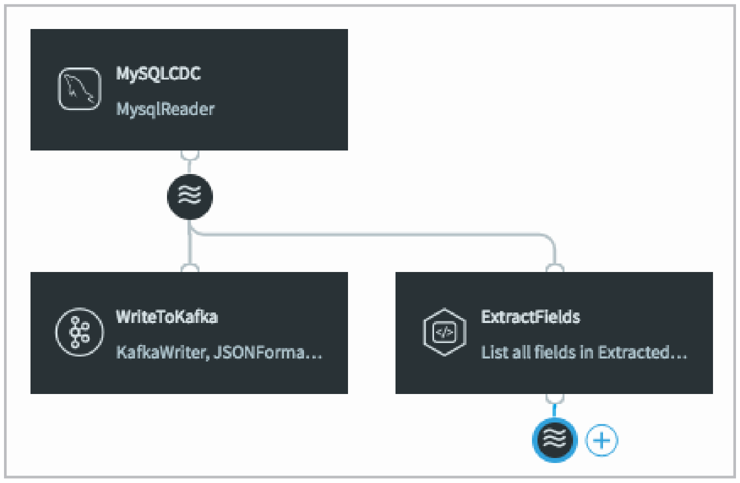
1. Press Ctrl-space for a pop-up menu of functions and cache, stream, and WActionStore names

[✓ New Output] ExtractedFields

Existing Output

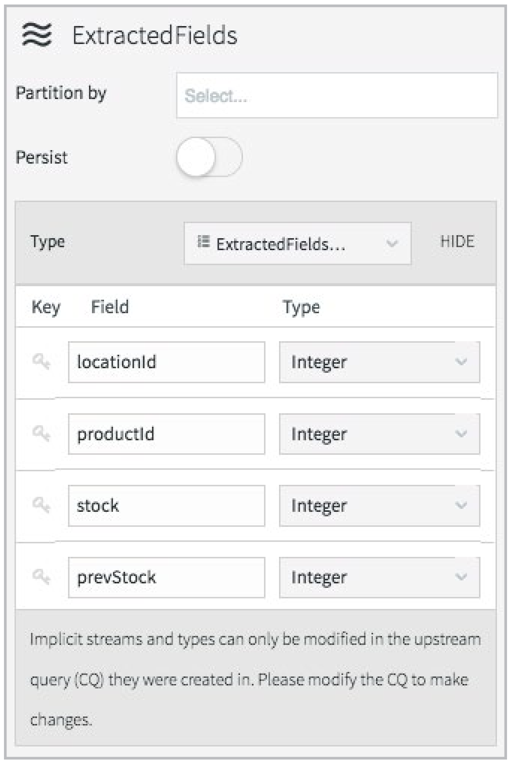
Data-flow with Newly Added CQ

When we click on Save, the query is created alongside the new output stream, which has a data type to match the projections (the transformed data we selected in the query).



After Clicking Save, the New CQ and Stream Are Added

The data type of the stream can be viewed by clicking on the stream icon.



ExtractedFields

Partition by: Select...

Persist

Type: ExtractedFields... HIDE

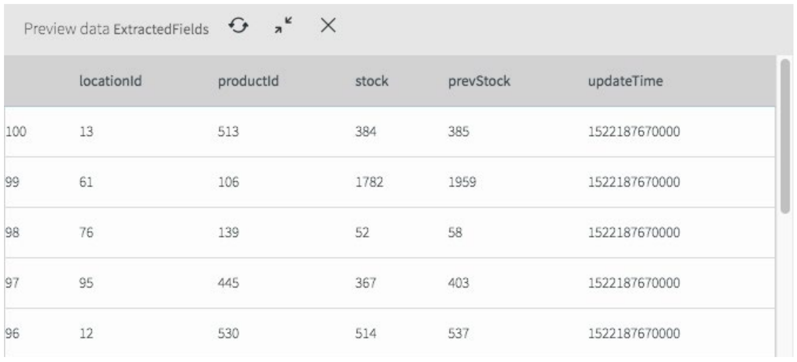
Key	Field	Type
	locationId	Integer
	productId	Integer
	stock	Integer
	prevStock	Integer

Implicit streams and types can only be modified in the upstream query (CQ) they were created in. Please modify the CQ to make changes.

Stream Properties Showing Generated Type Division

There are many different things you can do with streams themselves, such as partition them over a cluster, or switch them to being persistent (which utilizes our built-in Apache Kafka), but that is a subject for a later blog.

If we deploy and start the application then we can see what the data now looks like in the stream.



Preview data ExtractedFields

	locationId	productId	stock	prevStock	updateTime
100	13	513	384	385	1522187670000
99	61	106	1782	1959	1522187670000
98	76	139	52	58	1522187670000
97	95	445	367	403	1522187670000
96	12	530	514	537	1522187670000

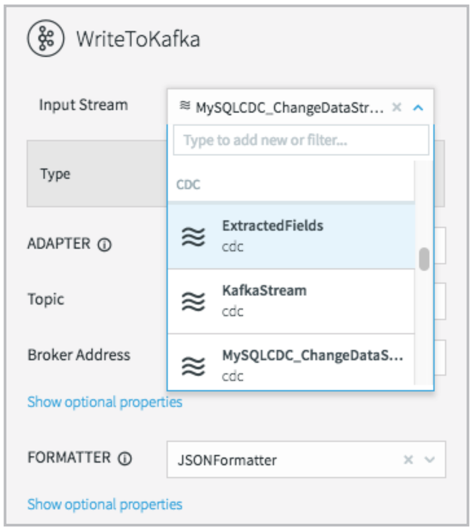
Extracted Fields Viewed by Previewing Data Streams

As you can see it looks very different from the previous view and now only contains the fields we are interested in for the remainder of the application.

But at the moment, this new stream currently goes nowhere, while the original data is still being written to Kafka.

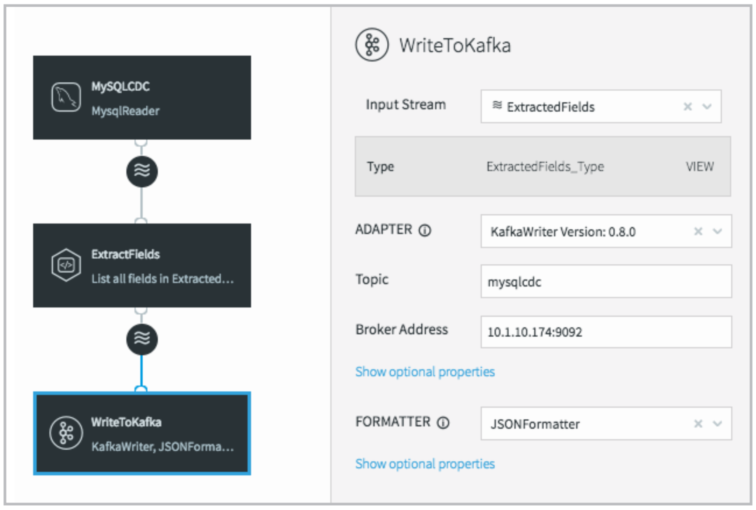
Delivering Transformed Data to Apache Kafka

To feed the transformed data to Kafka, all we need to do is change the input stream for the WriteToKafka component.



Changing the Kafka Writer Input Stream

This changes the data flow, making it a continuous linear pipeline, and ensures our new simpler data structure is what is written to Kafka.

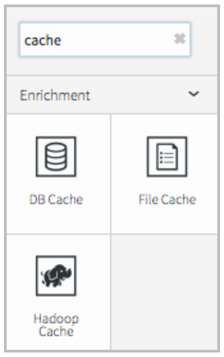


Input Stream: ExtractedFields
Type: ExtractedFields_Type VIEW
ADAPTER: KafkaWriter Version: 0.8.0
Topic: mysqlcdc
Broker Address: 10.1.10.174:9092
FORMATTER: JSONFormatter

Linear Data Flow Including Our Process CQ Before Writing to Kafka

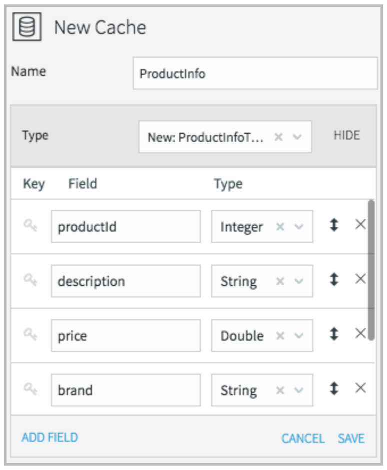
Utilizing Caches for Enrichment

Now that we have the data in a format we want, we can start to enrich it. Since the Striim platform is a high-speed, low latency, SQL-based stream processing platform, reference data also needs to be loaded into memory so that it can be joined with the streaming data without slowing things down. This is achieved through the use of the Cache component. Within the Striim platform, caches are backed by a distributed in-memory data grid that can contain millions of reference items distributed around a Striim cluster. Caches can be loaded from database queries, Hadoop, or files, and maintain data in-memory so that joining with them can be very fast.



A Variety of In-Memory Caches Are Available for Enrichment

In this example we are going to use two caches — one for product information loaded from a database, and another for location information loaded from a file.


New Cache

Name: ProductInfo

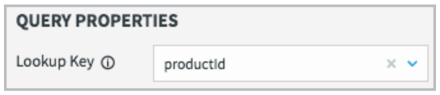
Type: New: ProductInfoT... HIDE

Key	Field	Type
	productId	Integer
	description	String
	price	Double
	brand	String

Setting the Name and Datatype for the ProductInfo Cache

All caches need a name, data type, lookup key, and can optionally be refreshed periodically. We'll call the product information cache "ProductInfo," and create a data type to match the MySQL PRODUCT table, which contains details of each product in our CDC stream. This is defined in MySQL as:

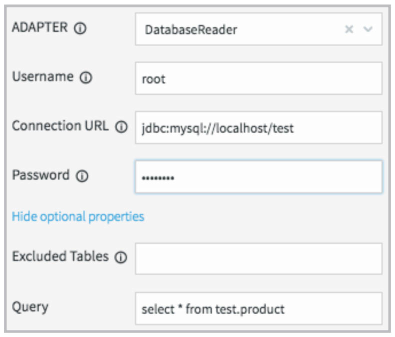
PRODUCT_ID int(11) PK
DESCRIPTION varchar(255)
PRICE decimal(8,2)
BRAND varchar(45)
CATEGORY varchar(45)
WORN varchar(45)


QUERY PROPERTIES

Lookup Key: productId

The lookup key for this cache is the primary key of the database table, or productId in this case.

All we need to do now is define how the cache obtains the data. This is done by setting the username, password, and connection URL information for the MySQL database, then selecting a table, or a query to run to access the data.



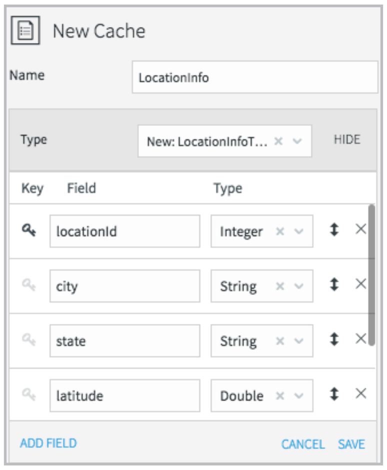
ADAPTER	DatabaseReader
Username	root
Connection URL	jdbc:mysql://localhost/test
Password	••••••••
Excluded Tables	
Query	select * from test.product

Configuring Database Properties for the ProductInfo Cache

When the application is deployed, the cache will execute the query and load all the data returned by the query into the in-memory data grid; ready to be joined with our stream.

Loading the location information from a file requires similar steps. The file in question is a comma-delimited list of locations in the following form:
Location ID, City, State, Latitude, Longitude, Population

We will create a File Cache called “LocationInfo” to read and parse this file, and load it into memory assigning correct data types to each column.



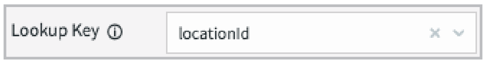
Name: LocationInfo

Type: New: LocationInfoT…

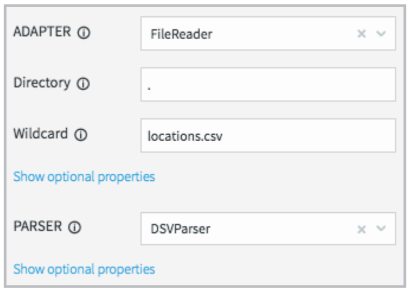
Key	Field	Type
✓	locationId	Integer
	city	String
	state	String
	latitude	Double

Setting the Name and Datatype for the LocationInfo Cache

The lookup key is the location id.


Lookup Key: locationId

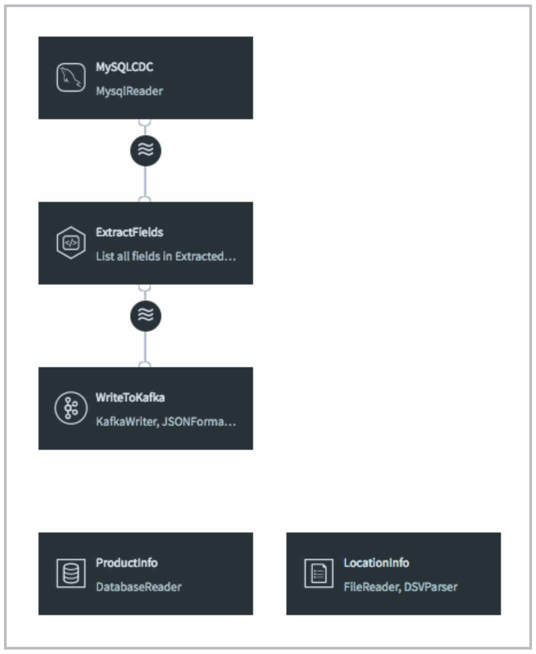
We will be reading data from the “locations.csv” file present in the product install directory “.” Using the DSVParser. This parser handles all kinds of delimited files. The default is to read comma-delimited files (with optional header and quoted values), so we can keep the default properties.



ADAPTER	FileReader
Directory	.
Wildcard	locations.csv
Show optional properties	
PARSER	DSVParser
Show optional properties	

Configuring FileReader Properties for the LocationInfo Cache

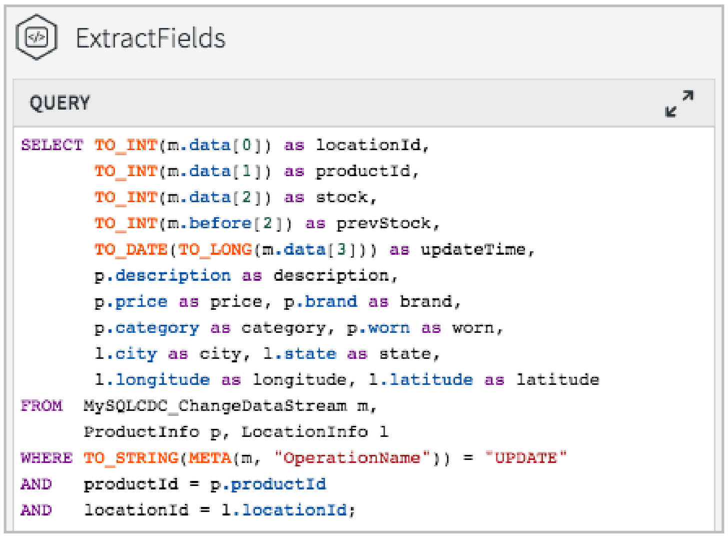
As with the database cache, when the application is deployed, the cache will read the file and load all the data into the in-memory data grid ready to be joined with our stream.



Dataflow Showing Both Caches Currently Ready to be Joined

Joining Streaming and Cache Data for Enrichment With SQL

The final step is to join the data in the caches with the real-time data coming from the MySQL CDC stream. This can be achieved by modifying the ExtractFields query we wrote earlier.


ExtractFields

QUERY

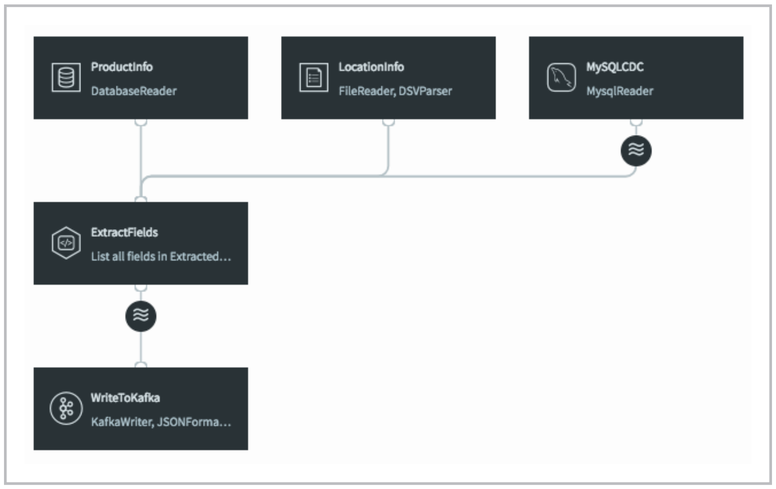
```
SELECT TO_INT(m.data[0]) as locationId,
       TO_INT(m.data[1]) as productId,
       TO_INT(m.data[2]) as stock,
       TO_INT(m.before[2]) as prevStock,
       TO_DATE(TO_LONG(m.data[3])) as updateTime,
       p.description as description,
       p.price as price, p.brand as brand,
       p.category as category, p.worn as worn,
       l.city as city, l.state as state,
       l.longitude as longitude, l.latitude as latitude
FROM   MySQLCDC_ChangeDataStream m,
       ProductInfo p, LocationInfo l
WHERE TO_STRING(META(m, "OperationName")) = "UPDATE"
AND    productId = p.productId
AND    locationId = l.locationId;
```

Full Transformation and Enrichment Query Joining the CDC Stream with Cache Data

All we are doing here is adding the ProductInfo and LocationInfo caches into the FROM clause, using fields from the caches as part of the projection, and including joins on productId and locationId as part of the WHERE clause.

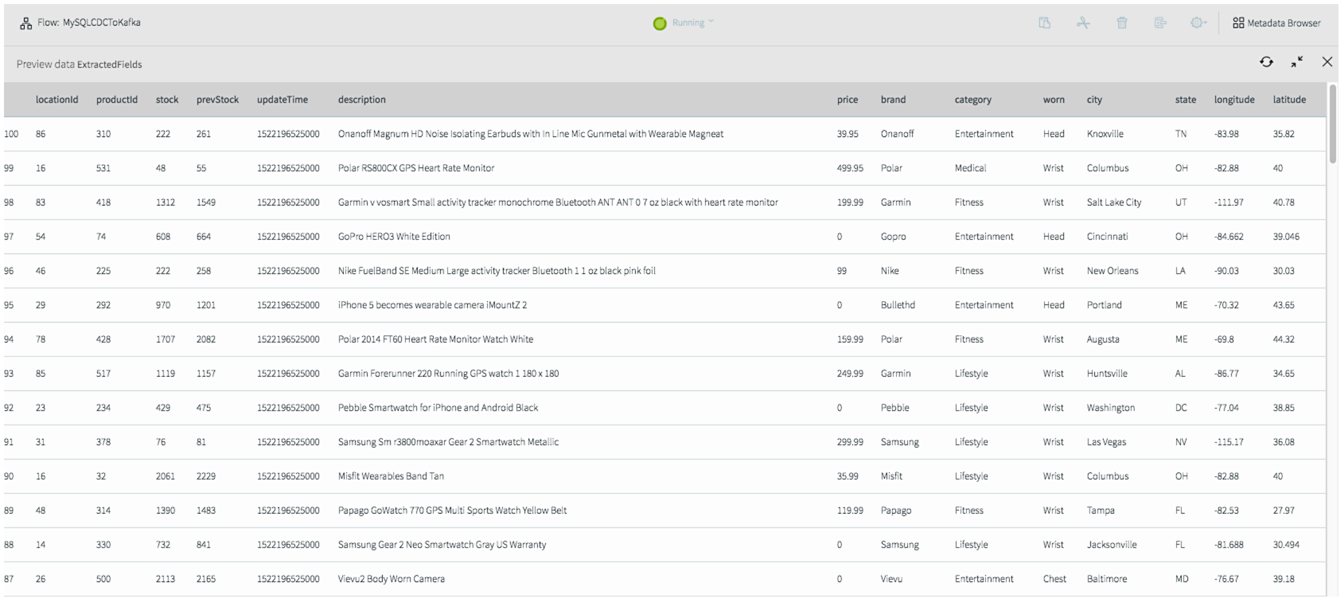
The result of this query is to continuously output enriched (denormalized) events for every CDC event that occurs for the PRODUCT_INV table. If the join was more complex — such that the ids could be null, or not match the cache entries — we could change to use a variety of join syntaxes, such as OUTER joins, on the data. We will cover this topic in a subsequent blog.

When the query is saved, the dataflow changes in the UI to show that the caches are now being used by the continuous query.



Dataflow After Joining Streaming Data with Caches in the CQ

If we deploy and start the application, then preview the data on the stream prior to writing to Kafka we will see the fully-enriched records.



	locationId	productId	stock	prevStock	updateTime	description	price	brand	category	worn	city	state	longitude	latitude
100	86	310	222	261	1522196525000	Onanoff Magnum HD Noise Isolating Earbuds with In Line Mic Gunmetal with Wearable Magneat	39.95	Onanoff	Entertainment	Head	Knoxville	TN	-83.98	35.82
99	16	531	48	55	1522196525000	Polar RS800CX GPS Heart Rate Monitor	499.95	Polar	Medical	Wrist	Columbus	OH	-82.88	40
98	83	418	1312	1549	1522196525000	Garmin v vosmart Small activity tracker monochrome Bluetooth ANT ANT 0 7 oz black with heart rate monitor	199.99	Garmin	Fitness	Wrist	Salt Lake City	UT	-111.97	40.78
97	54	74	608	664	1522196525000	GoPro HERO3 White Edition	0	Gopro	Entertainment	Head	Cincinnati	OH	-84.662	39.046
96	46	225	222	258	1522196525000	Nike FuelBand SE Medium Large activity tracker Bluetooth 1 1 oz black pink foil	99	Nike	Fitness	Wrist	New Orleans	LA	-90.03	30.03
95	29	292	970	1201	1522196525000	iPhone 5 becomes wearable camera iMountZ 2	0	Bulleth d	Entertainment	Head	Portland	ME	-70.32	43.65
94	78	428	1707	2082	1522196525000	Polar 2014 FT60 Heart Rate Monitor Watch White	159.99	Polar	Fitness	Wrist	Augusta	ME	-69.8	44.32
93	85	517	1119	1157	1522196525000	Garmin Forerunner 220 Running GPS watch 1 180 x 180	249.99	Garmin	Lifestyle	Wrist	Huntsville	AL	-86.77	34.65
92	23	234	429	475	1522196525000	Pebble Smartwatch for iPhone and Android Black	0	Pebble	Lifestyle	Wrist	Washington	DC	-77.04	38.85
91	31	378	76	81	1522196525000	Samsung Sm r3800moaxar Gear 2 Smartwatch Metallic	299.99	Samsung	Lifestyle	Wrist	Las Vegas	NV	-115.17	36.08
90	16	32	2061	2229	1522196525000	Misfit Wearables Band Tan	35.99	Misfit	Lifestyle	Wrist	Columbus	OH	-82.88	40
89	48	314	1390	1483	1522196525000	Papago GoWatch 770 GPS Multi Sports Watch Yellow Belt	119.99	Papago	Fitness	Wrist	Tampa	FL	-82.53	27.97
88	14	330	732	841	1522196525000	Samsung Gear 2 Neo Smartwatch Gray US Warranty	0	Samsung	Lifestyle	Wrist	Jacksonville	FL	-81.688	30.494
87	26	500	2113	2165	1522196525000	Vievu2 Body Worn Camera	0	Vievu	Entertainment	Chest	Baltimore	MD	-76.67	39.18

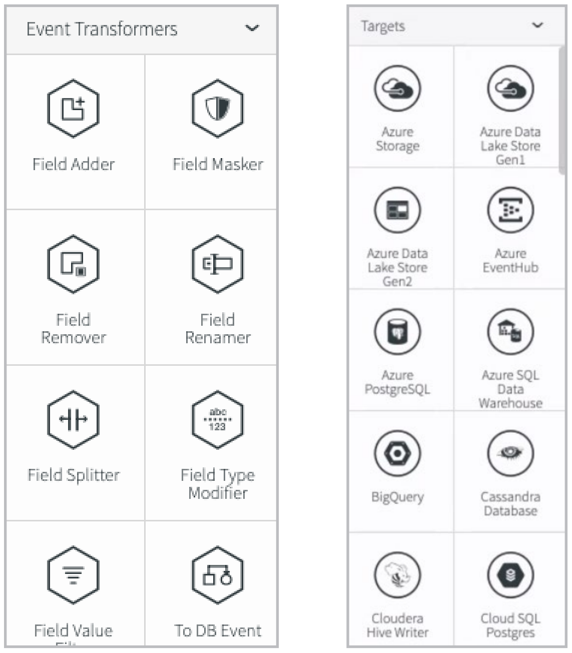
Results of Previewing Data After Transformation and Enrichment

The data delivered to Kafka as JSON looks like this.

```
{
  “locationId“:9,
  “productId“:152,
  “stock“:1277,
  “prevStock“:1383,
  “updateTime“:”2018-03-27T17:28:45.000-07:00”,
  “description“:”Dorcy 230L ZX Series Flashlight”,
  “price“:33.74,
  “brand“:”Dorcy”,
  “category“:”Industrial”,
  “worn“:”Hands”,
  “city“:”Dallas”,
  “state“:”TX”,
  “longitude“:-97.03,
  “latitude“:32.9
}
```

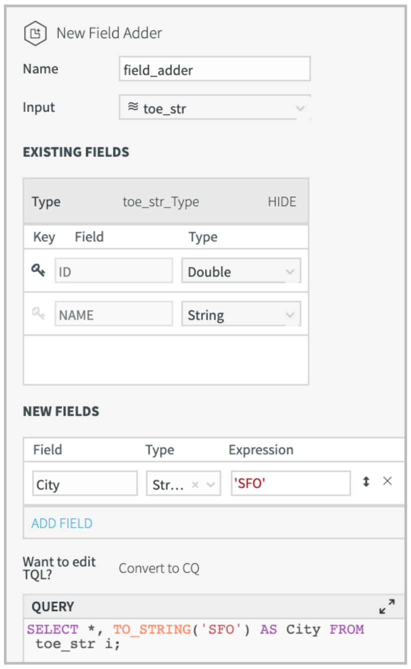
Stream Processing with Zero Coding

As you can see, it is very straightforward to use the Striim platform to not only integrate streaming data sources using CDC with Apache Kafka, but also to leverage SQL-based stream processing and enrich the data-in-motion without slowing the data flow. Striim's flow designer provides various processors, enrichers, transformers, and targets as shown below to complete your pipeline, in some cases with zero coding.



Flow Designer Event Transformers and Targets

All you need to do is to configure the applet to set up in-flight data transformation, filtering, masking, or enrichment steps to your data flow.


New Field Adder

Name	field_adder
Input	≋ toe_str

EXISTING FIELDS

Type toe_str_Type HIDE

Key	Field	Type
🔑	ID	Double
	NAME	String

NEW FIELDS

Field	Type	Expression
City	Str...	'SFO'

ADD FIELD

Want to edit TQL? Convert to CQ

QUERY

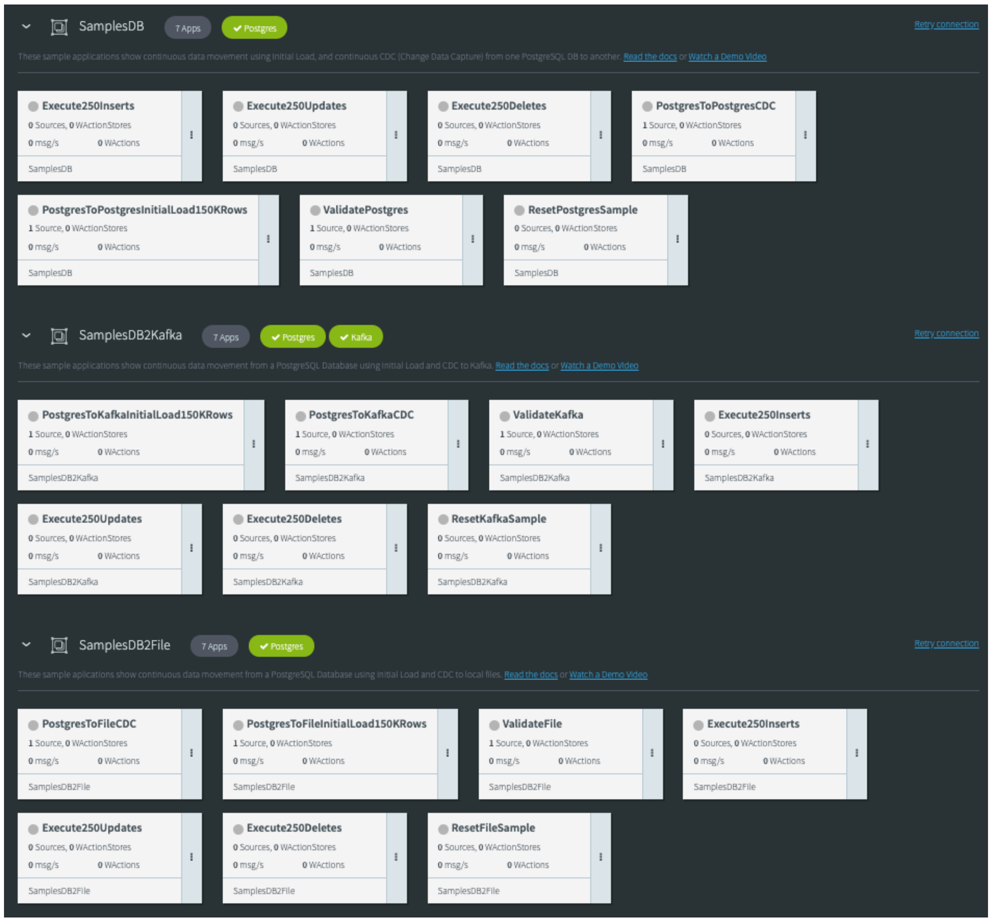
```
SELECT *, TO_STRING('SFO') AS City FROM
 toe_str i;
```

Flow Designer Event Transformer for Adding New Fields

Using Striim for Other Database Sources

Striim's CDC feature supports many sources beyond MySQL. The popular sources are Oracle, SQL Server, MySQL, PostgreSQL, HPE NonStop, and MariaDB. Striim's easy-to-use CDC template wizards automate the creation of applications for these popular data sources as well.

In addition, Striim offers pre-built integration applications for bulk loading and CDC from PostgreSQL source databases to target systems including PostgreSQL database, Kafka, and files. You can start these applications in seconds by going to the Applications section of the Striim platform.



SamplesDB 7 Apps ✓ Postgres Retry connection

These sample applications show continuous data movement using Initial Load, and continuous CDC (Change Data Capture) from one PostgreSQL DB to another. Read the docs or Watch a Demo Video

Execute250Inserts
0 Sources, 0 WActionStores
0 msg/s 0 WActions
SamplesDB

Execute250Updates
0 Sources, 0 WActionStores
0 msg/s 0 WActions
SamplesDB

Execute250Deletes
0 Sources, 0 WActionStores
0 msg/s 0 WActions
SamplesDB

PostgresToPostgresCDC
1 Source, 0 WActionStores
0 msg/s 0 WActions
SamplesDB

PostgresToPostgresInitialLoad150KRows
1 Source, 0 WActionStores
0 msg/s 0 WActions
SamplesDB

ValidatePostgres
1 Source, 0 WActionStores
0 msg/s 0 WActions
SamplesDB

ResetPostgresSample
0 Sources, 0 WActionStores
0 msg/s 0 WActions
SamplesDB

SamplesDB2Kafka 7 Apps ✓ Postgres ✓ Kafka Retry connection

These sample applications show continuous data movement from a PostgreSQL Database using Initial Load and CDC to Kafka. Read the docs or Watch a Demo Video

PostgresToKafkaInitialLoad150KRows
1 Source, 0 WActionStores
0 msg/s 0 WActions
SamplesDB2Kafka

PostgresToKafkaCDC
1 Source, 0 WActionStores
0 msg/s 0 WActions
SamplesDB2Kafka

ValidateKafka
1 Source, 0 WActionStores
0 msg/s 0 WActions
SamplesDB2Kafka

Execute250Inserts
0 Sources, 0 WActionStores
0 msg/s 0 WActions
SamplesDB2Kafka

Execute250Updates
0 Sources, 0 WActionStores
0 msg/s 0 WActions
SamplesDB2Kafka

Execute250Deletes
0 Sources, 0 WActionStores
0 msg/s 0 WActions
SamplesDB2Kafka

ResetKafkaSample
0 Sources, 0 WActionStores
0 msg/s 0 WActions
SamplesDB2Kafka

SamplesDB2File 7 Apps ✓ Postgres Retry connection

These sample aplications show continuous data movement from a PostgreSQL Database using Initial Load and CDC to local files. Read the docs or Watch a Demo Video

PostgresToFileCDC
1 Source, 0 WActionStores
0 msg/s 0 WActions
SamplesDB2File

PostgresToFileInitialLoad150KRows
1 Source, 0 WActionStores
0 msg/s 0 WActions
SamplesDB2File

ValidateFile
1 Source, 0 WActionStores
0 msg/s 0 WActions
SamplesDB2File

Execute250Inserts
0 Sources, 0 WActionStores
0 msg/s 0 WActions
SamplesDB2File

Execute250Updates
0 Sources, 0 WActionStores
0 msg/s 0 WActions
SamplesDB2File

Execute250Deletes
0 Sources, 0 WActionStores
0 msg/s 0 WActions
SamplesDB2File

ResetFileSample
0 Sources, 0 WActionStores
0 msg/s 0 WActions
SamplesDB2File

Striim's Pre-built Sample Integration Applications

Conclusion

While Kafka serves many enterprises as a business-critical fault-tolerant messaging system, getting data in and out of Kafka, and performing true stream processing and analytics for Kafka data is not a trivial task. If organizations want to make the most of Kafka, they shouldn't have to architect and build a massive infrastructure, nor should they need an army of developers to craft their required processing and analytics solutions. The Striim platform offers an enterprise-grade software platform that enables fast-time-to-market for integration and analytics solutions for Kafka. By using Striim, data scientists, business analysts and other IT and data professionals can focus on delivering business value.

striim

Connect with us:

www.striim.com/blog/

www.linkedin.com/company/striim

www.facebook.com/striim

www.twitter.com/striimteam

www.striim.com/youtube

For more information, or to schedule a free trial, please contact us at **info@striim.com** or at **+1 (650) 241-0680**

www.striim.com