Hortonworks DataFlow

Hortonworks Streaming Analytics Manager

(April 24, 2017)

docs.hortonworks.com

Hortonworks DataFlow: Hortonworks Streaming Analytics Manager

Table of Contents

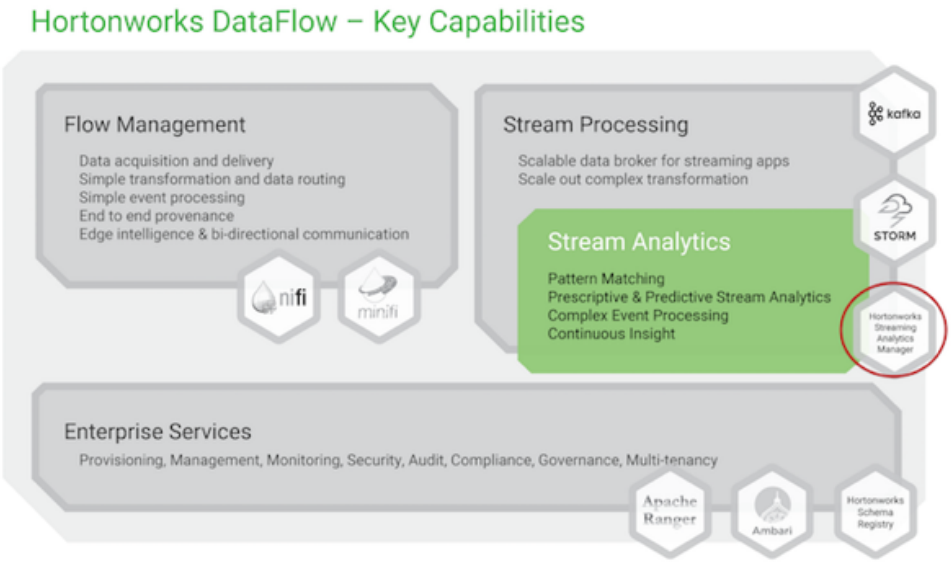
1. Overview

Important

Streaming Analytics Manager is a technical preview release and is under development. Do not use this in a production deployment. If you have any questions about Streaming Analytics Manager, contact Support by logging a case on the Hortonworks Support Portal, and provide feedback on your experiences through the Hortonworks Community Forums.

The Hortonworks DataFlow Platform (HDF) provides flow management, stream processing, and enterprise services for collecting, curating, analyzing and acting on data in motion across on-premise data centers and cloud environments.

As the following diagram illustrates, Hortonworks Streaming Analytics Manager is an application within the stream processing suite of the HDF platform:



Use Streaming Analytics Manager to design, develop, deploy and manage streaming analytics apps with a drag-and-drop visualization paradigm. Streaming Analytics Manager allows you to build streaming analytics apps for event correlation, context enrichment, complex pattern matching, and analytic aggregations. You can create alerts and notifications when insights are discovered.

Streaming Analytics Manager is agnostic to the underlying streaming engine, and it can support multiple streaming substrates such as Storm, Spark Streaming, Flink, etc. The first streaming engine fully supported is Apache Storm.

This overview chapter describes fundamental concepts related to Streaming Analytics Manager:

- Stream processing and flow management capabilities
- Streaming Analytics Manager modules

- Streaming Analytics Manager taxonomy
- Streaming Analytics Manager personas

1.1. Stream Processing and Flow Management Capabilities

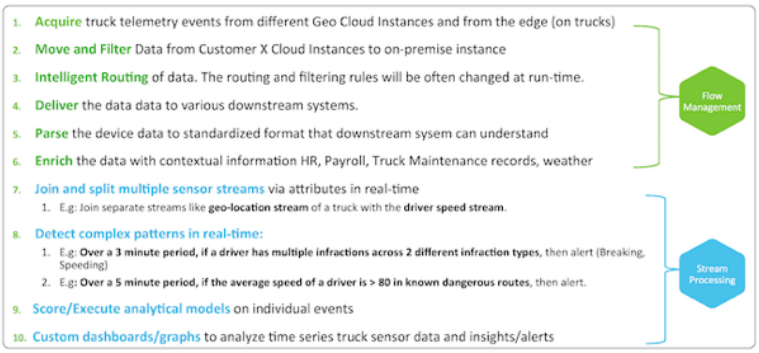
To build real-world data in motion apps such as those based on the Internet of Things (IoT), you need both flow management and stream processing capabilities. What is the difference between the two?

Data in motion apps typically have the following key requirements:

- **Acquisition of Data** from data sources within the data center, and across cloud environments and edge devices.
- **Moving and Filtering of Data** from edge devices (such as telematic panels on trucks), and across cloud environments and core data centers.
- **Intelligent and Dynamic Routing of Data** across regional data centers to core processing data centers.
- **Delivering Data** to different downstream systems.
- **Joining and Splitting Streams of Data** as they move.
- **Detecting complex patterns** in the streams of data.
- **Scoring/Executing Analytics Models** within the stream.
- **Creating Custom Dashboards** to visualize and analyze the streams and insights.

To explain how flow management and stream processing relate to these requirements, we employ a fictitious use case for trucking company X, which installed sensors on its fleet of trucks. These sensors emit streams of event data such as speed, braking frequency, and geo-code location. In this use case, the trucking company is building an IoT trucking app that monitors trucks in real time.

The following diagram illustrates how each of these requirements would be implemented in the context of stream processing and flow management:



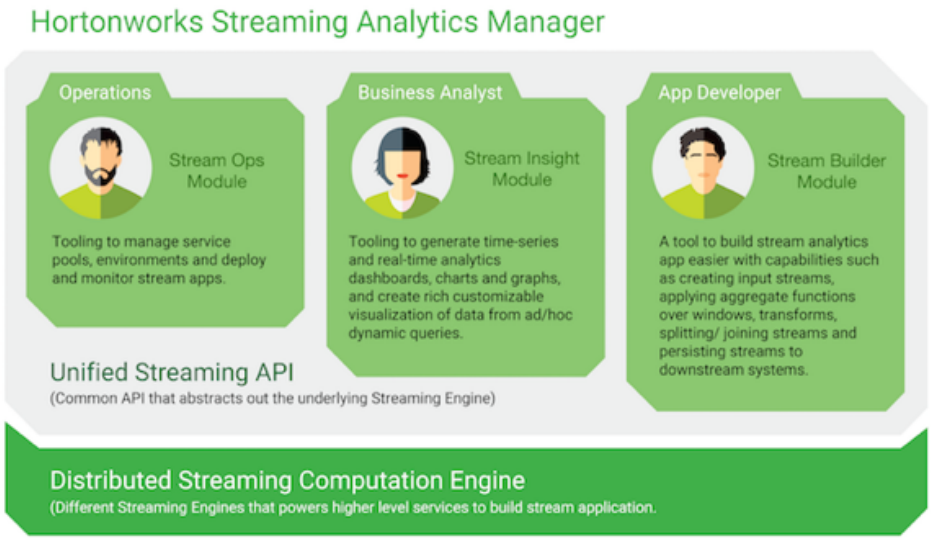
As part of the stream processing suite available in HDF, Streaming Analytics Manager provides capabilities for implementing the requirements outlined in blue in the previous diagram.

To summarize, Streaming Analytics Manager provides the following core capabilities:

- Building stream apps, using the following primitives:
  - Connecting to streams
  - Joining streams
  - Forking streams
  - Aggregating over windows
  - Extensibility: adding custom processors and user-defined-functions (UDFs)
  - Stream analytics: descriptive, predictive, and prescriptive
  - Rules engine
  - Transformations
  - Filtering and routing
  - Notifications and alerts
- Deploying stream apps:
  - Deploying the stream app on a supported streaming engine:
  - Monitoring the stream app with application-specific metrics.
- Exploring and analyzing streaming data; discovering insights:
  - Creating dashboards of streaming data
  - Exploring streaming data
  - Creating streaming cubes

1.2. Streaming Analytics Manager Modules

Streaming Analytics Manager is composed of several different modules that cater to different user personas. The following diagram illustrates Streaming Analytics Manager modules



You can think of Streaming Analytics Manager as an application that operates on top of a streaming engine (a "gray box") that allows you build stream apps faster on top of your selected streaming substrate. The unified streaming API provides the abstraction that allows you to plug in different streaming engines.

1.3. Streaming Analytics Manager Taxonomy

The following table describes the taxonomy for Streaming Analytics Manager. This taxonomy will be used throughout the rest of this guide.

Term	Description
Streaming Analytics Manager	The name of the graphical application for building, deploying, and managing stream apps.
Stream App	A streaming application built using Streaming Analytics Manager.
My Applications	The landing page for the Streaming Analytics Manager application. The Dashboard has a list of stream apps.
App Tile	Located on the dashboard, an app tile provides metrics, status and lifecyle actions for a stream. Each stream app is displayed as an app tile.
Stream Builder	The Streaming Analytics Manager tool that is used to build stream apps.
Builder Canvas	The canvas of the Stream Builder, on which stream apps are built. The canvas includes a palette of Builder components that can be used to build a stream app.
Builder Components	Building blocks available on the Builder Canvas palette, which can be used to build stream apps. There are four types of Builder components: • Source Builder Component: for creating streams from data sources such as Kafka topics or HDFS files. • Processor Builder Component: for manipulating and processing events in a stream, such as routing, applying transformations, performing windowing operations, and applying rules.

Term	Description
	• Sink Builder Component: for sending events to other systems such as HBase, HDFS, and Kafka. • Custom Builder Component: for creating custom requirements and adding them to the canvas palette.
Tile component	A component tile that has been moved onto the Builder Canvas, configurable for use in a specific stream app.
Connectors	Define connections between component tiles, directing a flow of tuples and how they flow (such as shuffle grouping).
Stream Operation	A view showing a running stream app, providing metrics for the app. After you use Stream Builder to build and deploy a stream app, the Stream Operation view allows you to monitor the running app.
Service Pool	A pool of services that can be used to create different environments. Services can come from two sources: • Ambari-managed cluster: if you specify an Ambari URL, a service pool is populated with all of the services managed by that Ambari Instance; for example, Storm, HDFS, HBase, and Kafka. • Custom service pool: for services not managed by Ambari, you can create a custom service and add that to a pool. Examples include Elastic Service and the Schema Registry Service.
Environment	A set of services you choose from one or more service pools. The environment is then associated with a stream app, which uses those services in that environment for various configurations.
Stream Insight Superset	The name of the Stream Insight module within SAM for Business Analysts to create dashboards and visualizations
Insight Data Source	A analytics cube powered by Druid where events can be streamed into for rollups/aggregations/analytics
Insight Slice	An insight visualization that can be created from a cube. An insight can be added to a dashboard
Insight Dashboard	Consists of a set of insight slices. Dashboards are created by the Business analyst in the Stream Insights Superset module.

1.4. Streaming Analytics Manager Personas

Four main modules within Streaming Analytics Manager offer services to different personas in an organization:

User Persona	Module	Module Features and Functionality
IT Engineer, Operations Engineer, Platform Engineer, Platform Operator	Stream Management	• Create service pools and environments. • Provision, manage and monitor stream apps. • Scale out or scale in stream apps based on resource consumption.
Application Developer	Stream Builder	• The Stream Builder tool assists in building analytic-focused stream apps.

User Persona	Module	Module Features and Functionality
		• The tool creates streams for event correlation, context enrichment, complex pattern matching, and aggregation. It can create alerts and notifications when patterns are detected and insights are discovered. • The interface uses a drag-and-drop visual programming paradigm.
Business Analyst, Data Analyst	Stream Insight Superset	• The Stream Insight tool assists in generating time-series and real-time analytics dashboards, charts, and graphs of metrics, alerts and notifications. • The tool provides interactive, ad-hoc analytics. You can issue ad-hoc queries, perform multidimensional analyses, and visualize the results in rich configurable dashboards. • The tool offers a self-service ability to create alerts and notification dashboards based on insights derived from the real-time streaming data flows.
SDK Developer	Unified Streaming API	• The unified streaming API abstracts out the underlying streaming engine, making it more straightforward to implement custom components. Initial support is for Storm.

The following subsections describe responsibilities for each persona. For additional information, see the following chapters in this guide:

Persona	Chapter Reference
IT Engineer, Operations Engineer, Platform Engineer, Platform Operator	Installing and Configuring Streaming Analytics Manager Managing Stream Apps
Application Developer	Running the Sample App Building an End-to-End Stream App
Business Analyst, Data Analyst	Creating Visualizations: Insight Slices
SDK Developer	Adding Custom Builder Components

1.4.1. Platform Operator Persona

A platform operator typically manages the Streaming Analytics Manager platform, and provisions various services and resources for the application development team. Common responsibilities of a platform operator include:

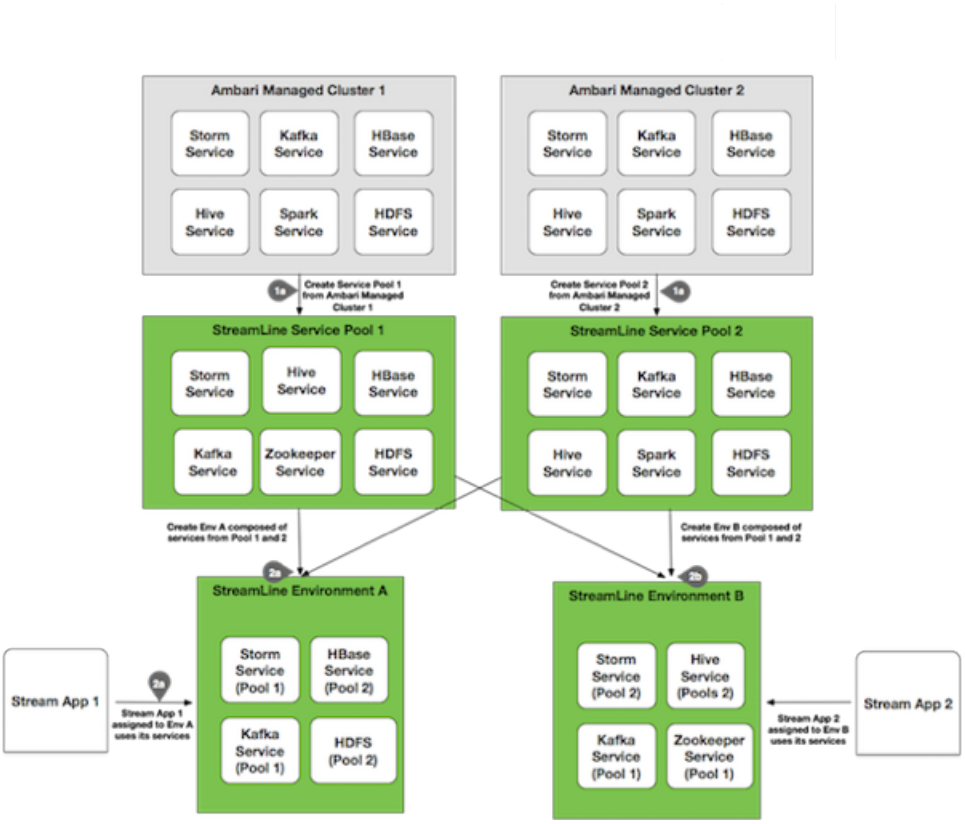
- Installing and managing the Streaming Analytics Manager application platform.
- Provisioning and providing access to services (e.g big data services like Kafka, Storm, HDFS, HBase) for use by the development team when building stream apps.
- Provisioning and providing access to environments such as development, testing, and production, for use by the development team when provisioning stream apps.

1.4.1.1. Services, Service Pools and Environments

To perform these responsibilities, a platform operator works with three important abstractions in Streaming Analytics Manager:

- **Service** is an entity that an application developer works with to build stream apps. Examples of services could be a Storm cluster that the stream app will be deployed to, a Kafka cluster that is used by the stream app to create a streams, or a HBase cluster that the stream app writes to.
- **Service Pool** is a set of services associated with an Ambari managed cluster
- **Environment** is a named entity that represents a set of services chosen from different service pools. A stream app is assigned to an environment and the app can only use the services associated with an environment.

The following diagram illustrates these constructs:



The Service, Service Pool, and Environment abstractions provide the following benefits:

1. **Simplicity and ease of use**: An application developer can use the Service abstraction without needing to focus on configuration details. For example, to deploy a stream app to a Storm cluster, the developer does not need to consider how to configure the Storm cluster (Nimbus host, ports, and so on). Instead, the developer simply selects the Storm service from the environment associated with the app. The service abstract out all the details/complexities.

2. **Ease of propagating a stream app between environments**: With Service as an abstraction, it is easy for the stream operator or application developer to move a stream app from one environment to another. They simply export the stream app and import it into a different environment.

For more information about creating and managing the Streaming Analytics Manager environment, see "Managing Stream Apps."

1.4.2. Application Developer Persona

The application developer uses the Stream Builder component to design, implement, deploy, and debug stream apps in Streaming Analytics Manager.

The following subsections describe component building blocks and schema requirements. For more information about building a stream app, see "Running the Sample Application" and "Building an End-to-End Stream App."

1.4.2.1. Component Building Blocks

Stream Builder offers several building blocks for stream apps: sources, processors, sinks, and custom components.

1.4.2.1.1. Sources

Source builder components are used to create data streams. SAM has the following sources:

- Kafka
- Azure Event Hub
- HDFS

1.4.2.1.2. Processors

Processor builder components are used to manipulate events in the stream.

The following table lists processors that are available with Streaming Analytics Manager.

Processor Name	Description
Join	• Joins two streams together based on a field from each stream. • Two join types are supported: inner and left. • Joins are based on a window that you can configure based on time or count.

Rule	• Allows you to configure rule conditions that route events to different streams.
	• Standard conditional operators are supported for rules.
	• Configuring a rule has two modes:
	  • General: Guided rule creation using drop-down menus.
	  • Advanced: Write complex SQL to construct a rule.

Processor Name	Description
	• Rules are translated to SQL to be applied on the stream.
Aggregate	• Performs functions over windows of events. • Two types of windows are supported: tumbling and sliding. • You can create window criteria based on time interval and count. • Window functions supported out of the box include: stddev, stddevp, variance, variancecep, avg, min, max, sum, count. The system is extensible to add custom functions as well.
Projection	• Applies transformations to the events in the stream • Extensive set of OOO functions and the ability to add your own functions
Branch	• Performs a standard if-else construct for routing.
PMML	• Executes a PMML model that is stored in the Model Registry. PMML has been minimally tested as part of the Tech Preview, and should not be used.

1.4.2.1.3. Sinks

Sink builder components are used to send events to other systems.

Streaming Analytics Manager supports the following sinks:

- Kafka
- Druid
- HDFS
- HBase
- Hive
- JDBC
- OpenTSDB
- Notification (OOO support Kafka and the ability to add custom notifications)
- Cassandra
- Solr

1.4.2.1.4. Custom Components

For more information about developing custom components, see SDK Developer Persona.

1.4.2.2. Schema Requirements

Unlike NiFi (the flow management service of the HDF platform), Streaming Analytics Manager requires a schema for stream apps. More specifically, every Builder component requires a schema to function.

The primary data stream source is Kafka, which uses the HDF Schema Registry.

The Builder component for Apache Kafka is integrated with the Schema Registry. When you configure a Kafka source and supply a Kafka topic, Streaming Analytics Manager calls the Schema Registry. Using the Kafka topic as the key, Streaming Analytics Manager retrieves the schema. This schema is then displayed on the tile component, and is passed to downstream components.

1.4.3. Analyst Persona

A business analyst uses the Streaming Analytics Manager Stream Insight module to create time-series and real-time analytics dashboards, charts and graphs; and create rich customizable visualizations of data.

Stream Insight Key Concepts

The following table describes key concepts of the Stream Insights module.

Stream Insight Concept	Description
Analytics Engine	• Stream Insight analytics engine is powered by Druid, an open source data store designed for OLAP queries on event data. • Data can be streamed into the Analytics engine via the Druid/Analytics Engine Sink that app developers can use when building streaming apps. The analytics engine sink can stream data into new/existing insight cubes.
Insight Data Source	• A insight data source is powered by Druid that represents the store for streaming data. The cube can be queried to do rollups, aggregations and other powerful analytics
Insight Slice	• A visualization that can be created from asking questions of the data source. An insight can be added to the dashboard
Dashboard	• Consists of a set of slices. Dashboards are created by the Business analysts to perform descriptive analytics

A business analyst can create a wide array of visualizations to gather insights on streaming data.

The platform supports over 30+ visualizations the business analyst can create. For visualization examples, see the Gallery of Superset Visualizations.

For more information, see "Creating Insight Slices."

1.4.4. SDK Developer Persona

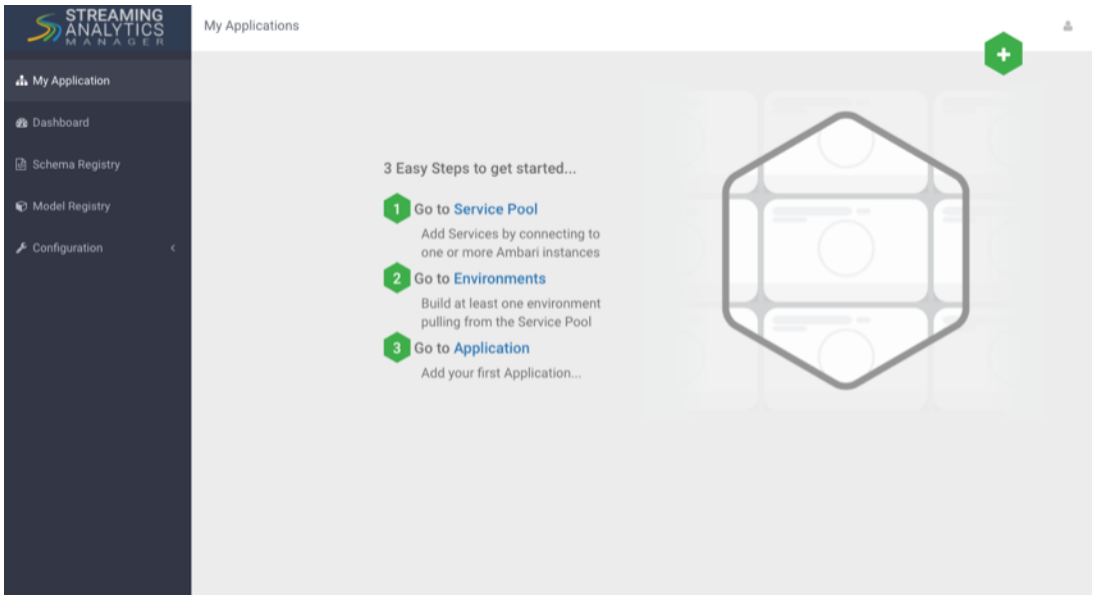
Streaming Analytics Manager supports the development of custom functionality through the use of its SDK. For more information, see "Adding Custom Builder Components."

2. Streaming Analytics Manager Environment Setup and Managing Stream Apps

The information in this chapter focuses on the following operational tasks, suited for operational persona:

- Creating service pools
- Creating environments

As the following diagram illustrates, when accessing the Streaming Analytics Manager for the first time there are three simple steps to perform before application developers can build streaming apps. Subsequent subsections walk through each of these steps.

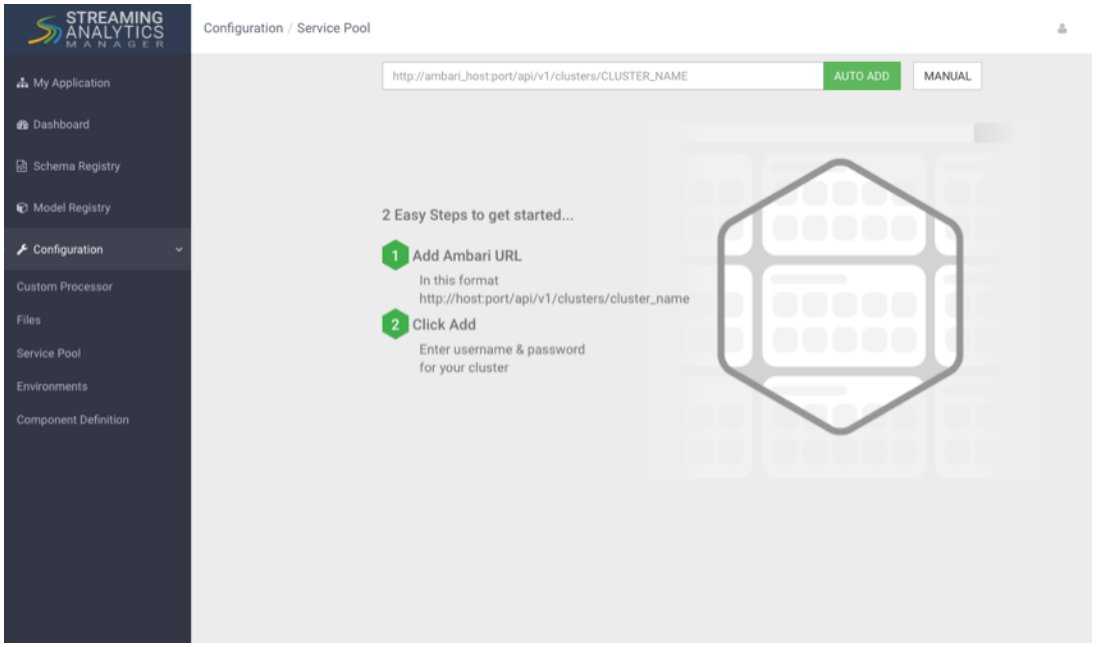


2.1. Managing Service Pools

Service is an entity that an application developer works with to build stream apps. Examples of services include a Storm cluster that the stream app will be deployed to, a Kafka cluster that is used by the stream app to create a streams, or an HBase cluster that the stream app writes to.

A **Service Pool** is a set of services associated with an Ambari managed cluster. To manage service pools, hover over the Configuration tab (the wrench icon) and select the Service Pool menu item.

The Service Pool dashboard lists all existing service pools, and allows you to create new service pools.

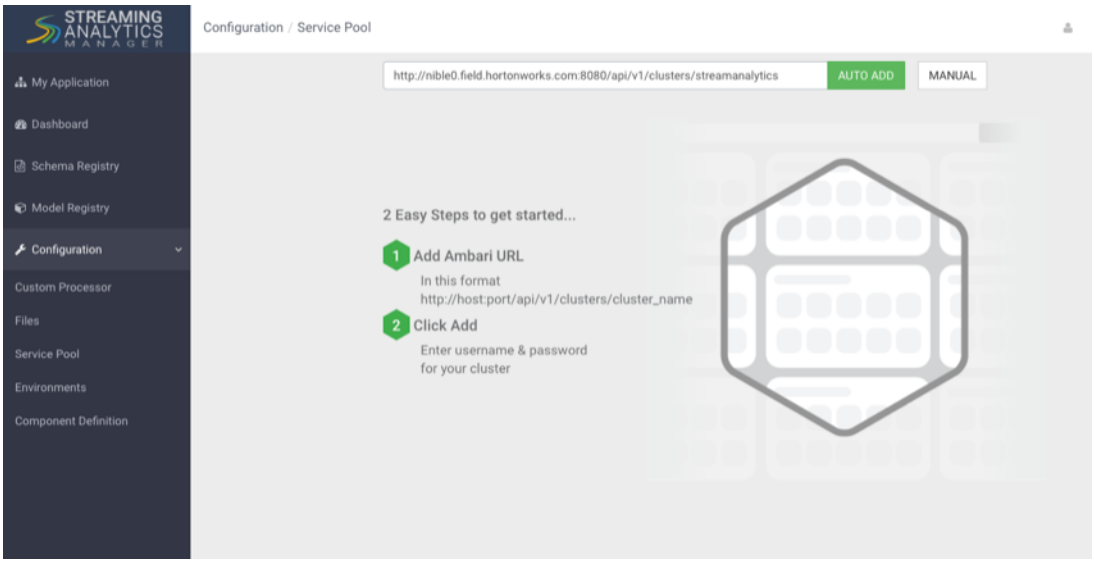


2.1.1. Adding a New Service Pool

To add a new service pool:

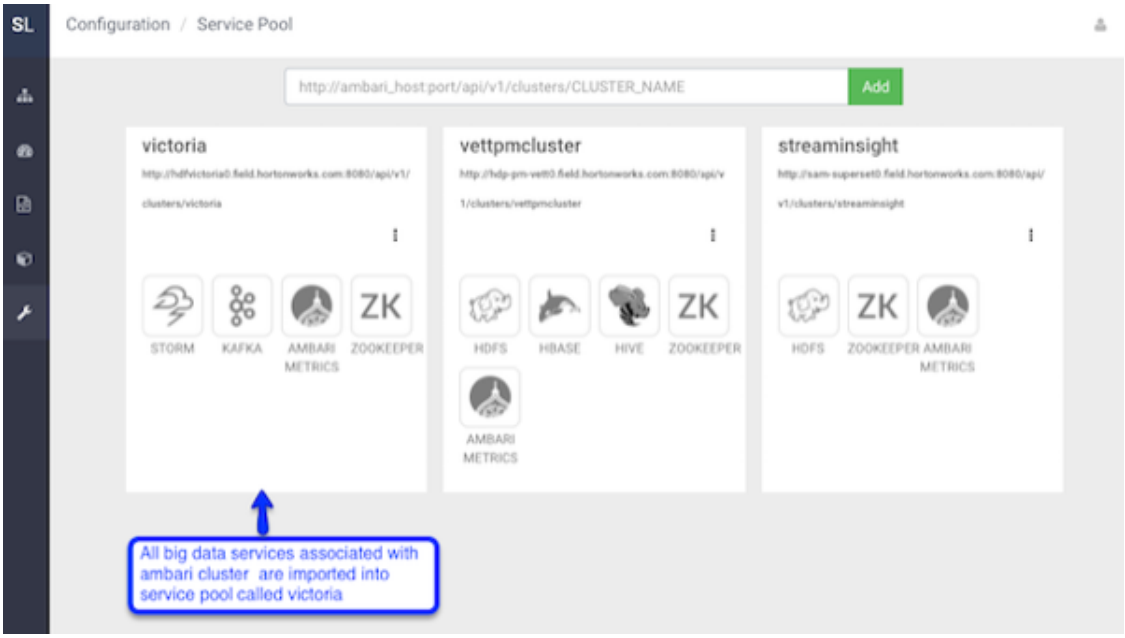
1. Enter the rest endpoint URL for the Ambari managed cluster, and click "Auto Add".

   The syntax of this URL has the following form: `http://[AMBARI_HOST]:8080/api/v1/clusters/[AMBARI_CLUSTER_NAME]`.



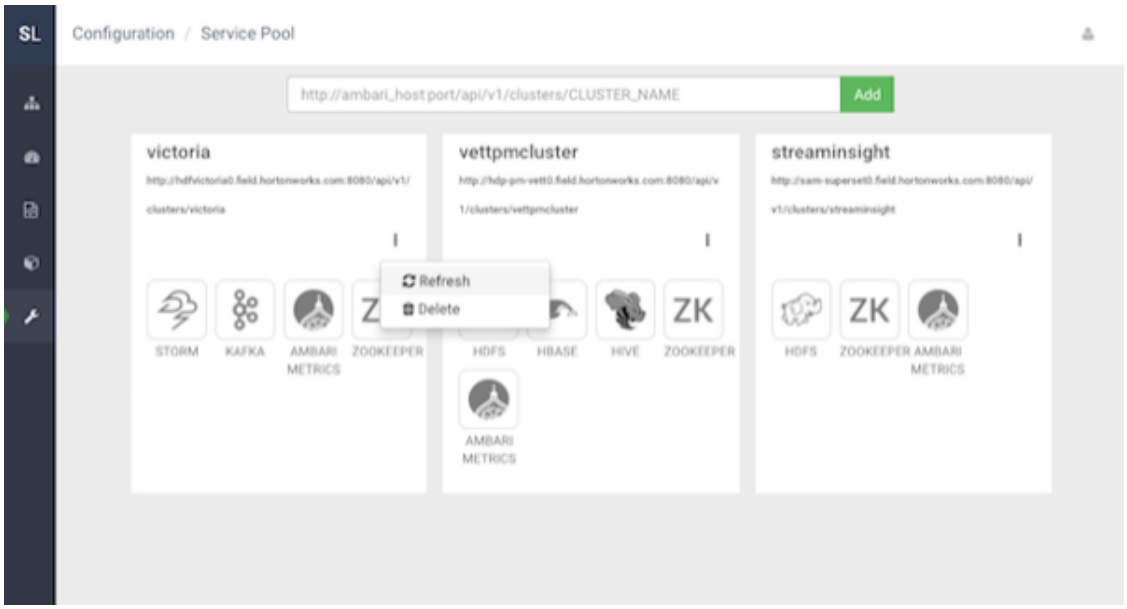
2. Ambari prompts for credentials; enter a valid username and password.

3. Streaming Analytics Manager retrieves all of the services and creates a new pool. The name of the service pool is the name of the Ambari cluster.



2.1.2. Updating Service Pools

When a service pool is created, all of the configuration to manage and connect to the big data services in the pool are imported from Ambari into SAM. If a configuration associated with a service is changed in Ambari, the service pool can adopt the new configuration by refreshing the pool as indicated in the following diagram.

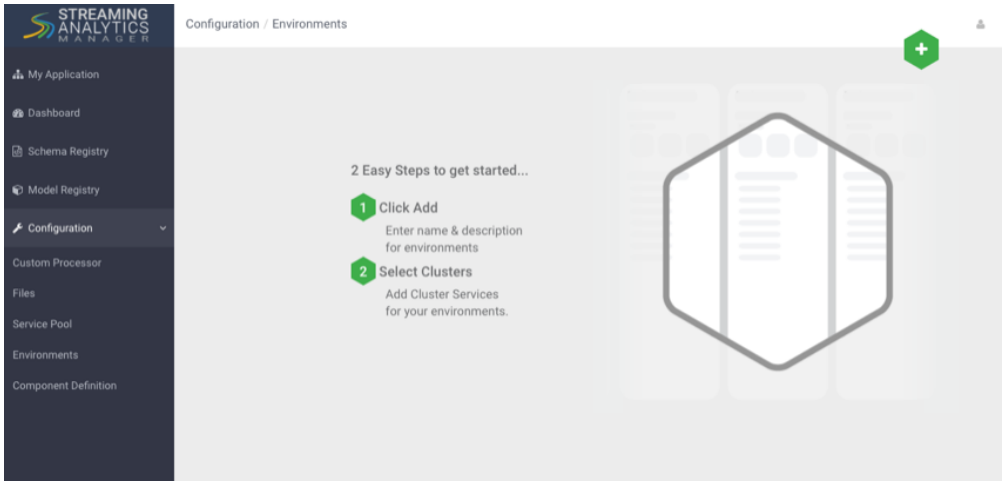


2.2. Managing Environments

Environment is a named entity that represents a set of services chosen from different service pools. A stream app is assigned to an environment; the app can only use the services associated with that environment.

To manage environments, hover over the Configuration Tab (the wrench icon) and select the Environments menu item.

The Environments dashboard lists all existing environments, and allows you to create a new Environment.

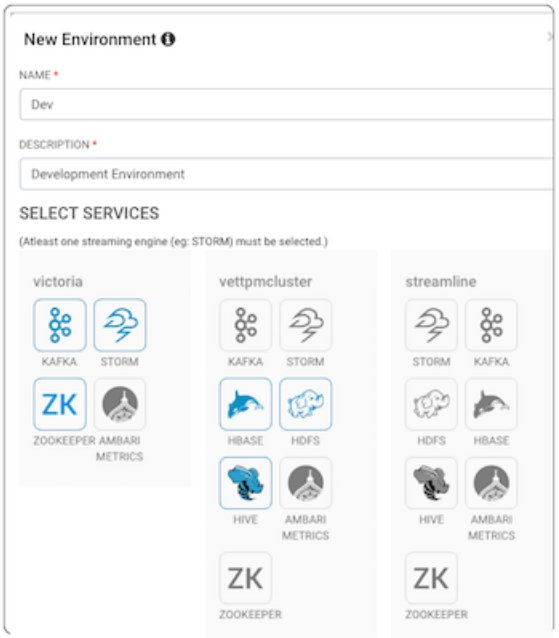


2.2.1. Create New Environment

To add a new environment:

1. Click the + icon.

2. Name the environment, choose the services that you want in the environment, and click "Save".

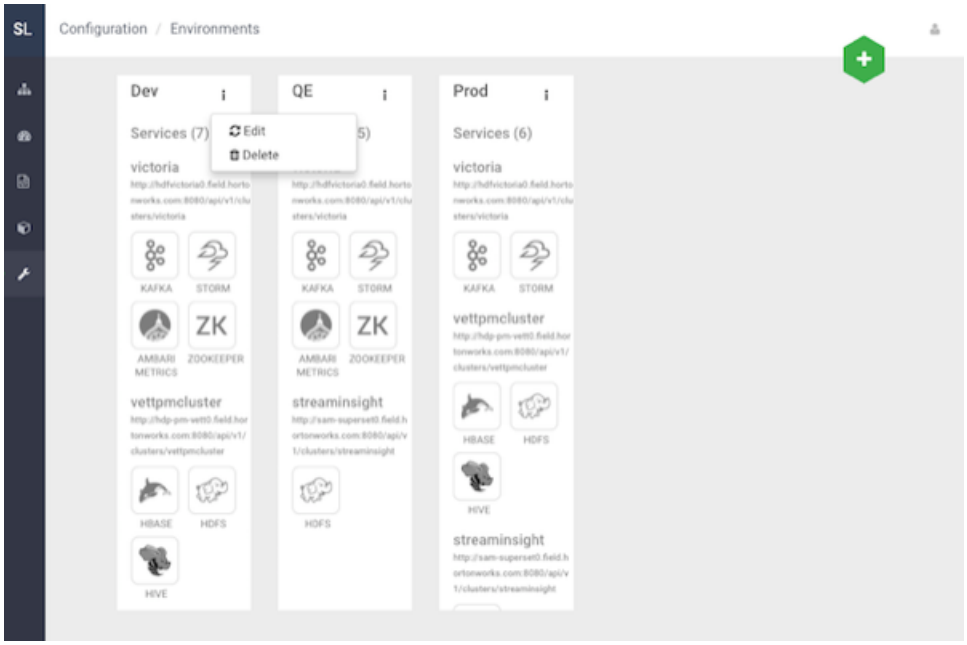


After an Environment is created, an app developer can create a new stream app, associate it with the environment, and use the big data services with the app. See *Building an End-to-End Stream App* for more details.

2.2.2. Updating and Deleting Environments

You can update and delete environments by clicking the options icon in each environment box, as shown in the following diagram.

When an environment is associated with an application, it cannot be deleted or updated.



3. Building an End-to-End Stream App

To build a complex streaming analytics app from scratch, we will work with a fictitious use case: a Trucking company with a large fleet of trucks that it wants to monitor in real time. Their monitoring app has the following requirements:

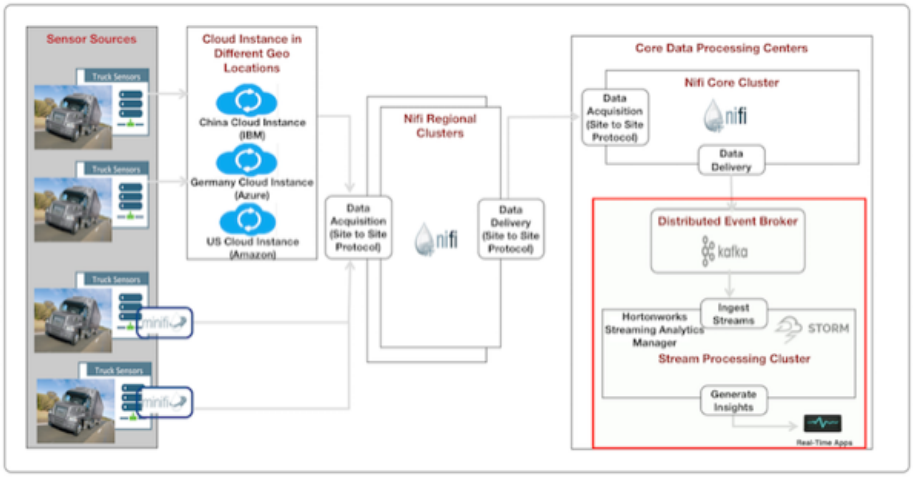
1. Outfit each truck with two sensors that emit event data such as timestamp, driver ID, truck ID, route, geographic location, and event type.
   - The geo event sensor emits geographic information (latitude and longitude coordinates) and events such as excessive braking or speeding.
   - The speed sensor emits the speed of the vehicle.
2. Emit events to an IoT gateway, and stream them into a file.
3. Ingest the data (using NiFi), and then route, transform, enrich and deliver the data to a downstream component (Kafka).
4. Connect to the two streams of data to do analytics on the stream.
5. Join the two sensor streams using attributes in real-time; for example, join the geo-location stream of a truck with the speed stream of a driver.
6. Filter the stream on only events that are infractions/violations.
7. All infraction events need to be available for descriptive analytics (dash-boarding, visualizations, etc.) by a business analyst. The analyst needs the ability to do analysis on the streaming data.
8. Detect complex patterns in real-time; for example:

   Over a three-minute period, detect if average speed of a driver is more than 80 miles per hour on routes known to be dangerous.
9. When each of the preceding rules fires, create alerts and make them instantly accessible.

3.1. Reference Architecture

The reference architecture for the Trucking use case is shown in the following image.

The below sections walks you through how to implement all nine requirements. Requirements 1-3 will be done using Nifi and Schema Registry and requirements 4 through 9, will be implemented using the new Streaming Analtyics Manager.



3.2. Registering Schemas in Schema Registry

To learn more about the Hortonworks Schema Registry, see the Hortonworks Schema Registry User Guide.

3.2.1. Truck Sensor Log File Schemas

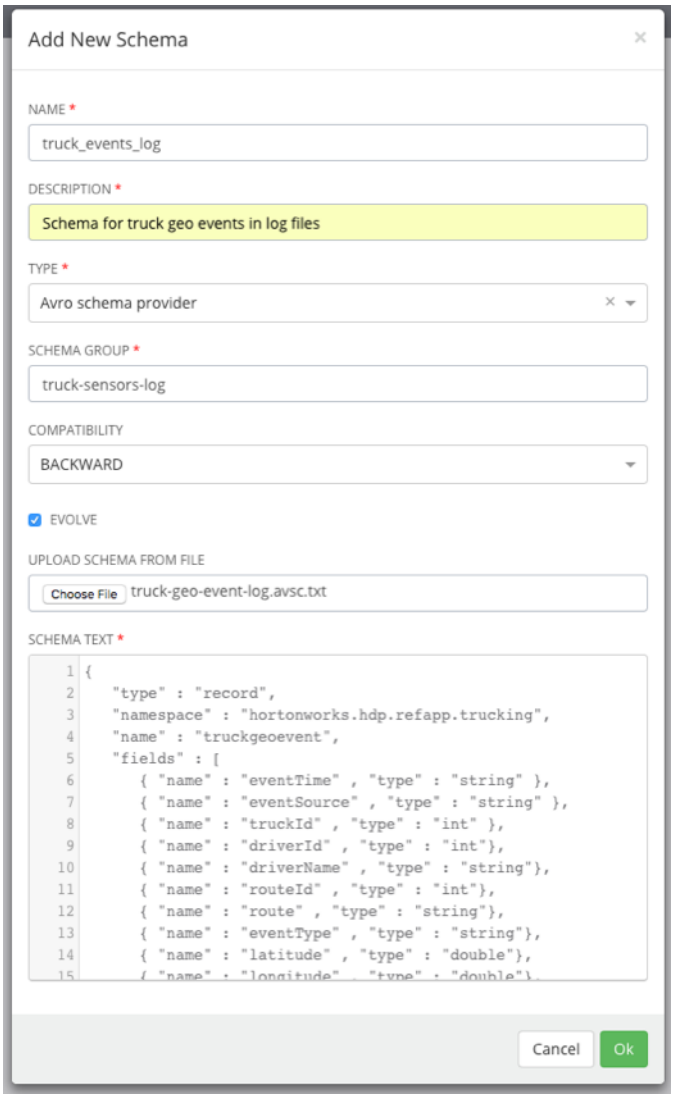
The trucking application streams raw events from the two sensors to a file. To accommodate files from the two sensor, you must register two schemas with Schema Registry. To register a schema, perform the following steps in the Schema Registry UI:

1. Select the Registry Service in Ambari and under QuickLinks select the 'Registry UI' to go the Schema Registry UI

2. Click the "+" button to add a schema, the schema group, and schema metadata for the Geo Event Sensor:

   - `Name = truck_events_log`
   - `Description = Schema for truck geo events in log files`
   - `Type = Avro schema provider`
   - `Schema Group = truck-sensors-log`
   - `Compatibility: BACKWARD`
   - Check the evolve checkbox
   - To setup the schema, do the following
     - Download this schema and save it locally

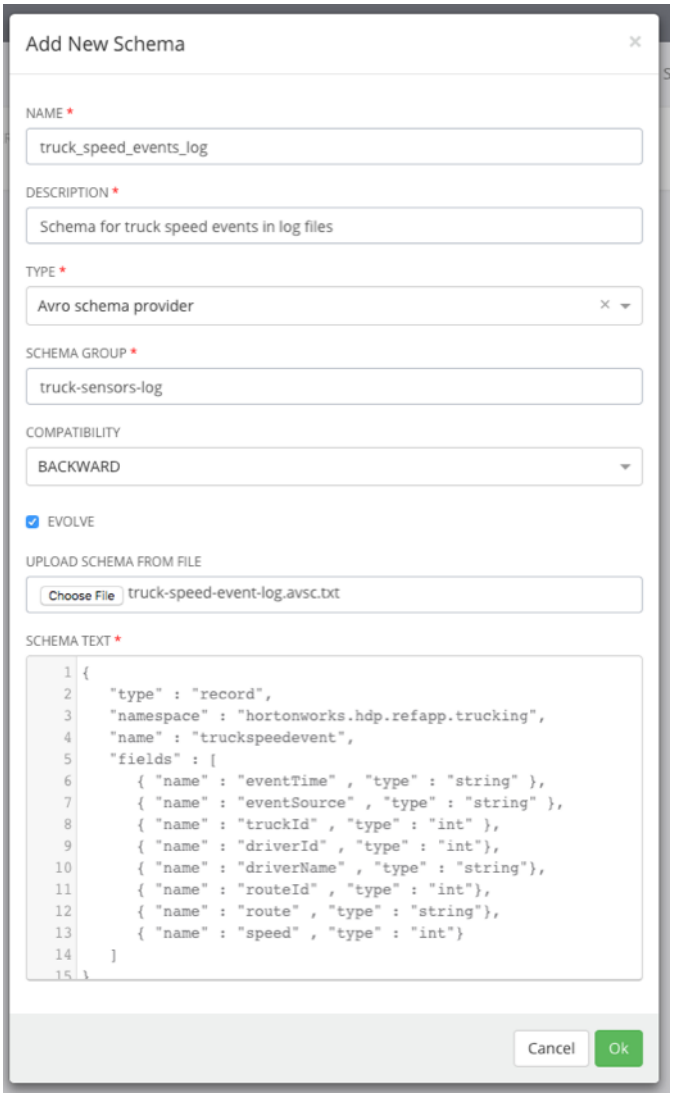
- Click the Choose File button and select the locally saved schema



3. Click the "+" button to add a schema, schema group, and schema metadata for the Speed Sensor:

   - `Name = truck_speed_events_log`
   - `Description = Schema for truck speed events in log files`
   - `Type = Avro schema provider`
   - `Schema Group = truck-sensors-log`
   - `Compatibility: BACKWARD`
   - Check the evolve checkbox
   - To setup the schema, do the following

- Download this schema and save it locally
- Click the Choose File button and select the locally saved schema



3.2.2. Truck Sensor Kafka Schemas

NiFi ingests the streams from the two sensors and routes the events to two different Kafka topic. During the process, NiFi enriches the data and publishes the enriched data to the Kafka topic.

Perform the following steps to create the 2 Kafka topics:

1. Log into the node where Kafka broker is running
2. Create the Kafka topics using the following commands:

```
cd /usr/hdf/current/kafka-broker/bin/
```

```
./kafka-topics.sh \
--create \
--zookeeper <zookeeper-host>:2181 \
--replication-factor 2 \
--partition 3 \
--topic truck_events_avro


./kafka-topics.sh \
--create \
--zookeeper <zookeeper-host>:2181 \
--replication-factor 2 \
--partition 3 \
--topic truck_speed_events_avro
```

Register the schemas for the two Kafka topics.

1. Click the "+" button to add a schema, schema group and schema metadata for the Geo Event Sensor kafka topic:

   - `Name = truck_events_avro:v`

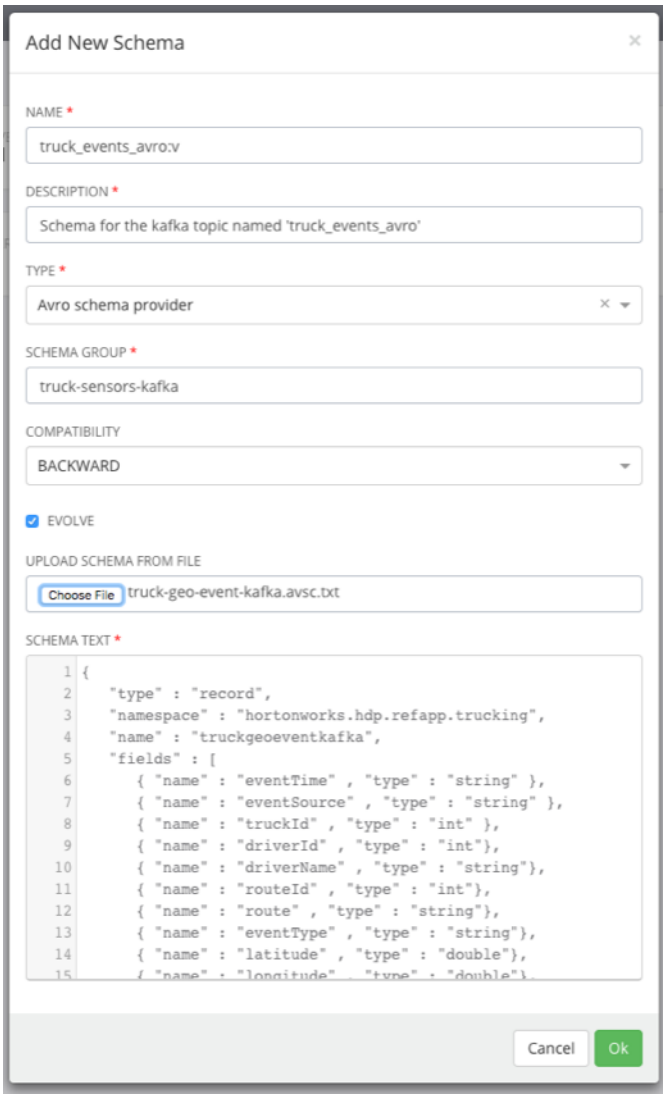
     

     Note

     The name of the schema must be the name of the kafka topic plus a suffix. The suffix is either a :v or :k because Kafka has the concept of a key and a value. A schema can be defined for either the key or the value or both. The suffix indicates what the schema is for. In this case, the :v indicates that this schema is for the values that go into the Kafka topic.

   - `Description = Schema for the kafka topic named 'truck_events_avro'`
   - `Type = Avro schema provider`
   - `Schema Group = truck-sensors-kafka`
   - `Compatibility: BACKWARD`
   - Check the evolve checkbox
   - To setup the schema, do the following
     - Download this schema and save it locally

- Click the Choose File button and select the locally saved schema



2. Click the "+" button to add a schema,, schema group (exists from previous step), and schema metadata for the Speed Event Sensor kafka topic:

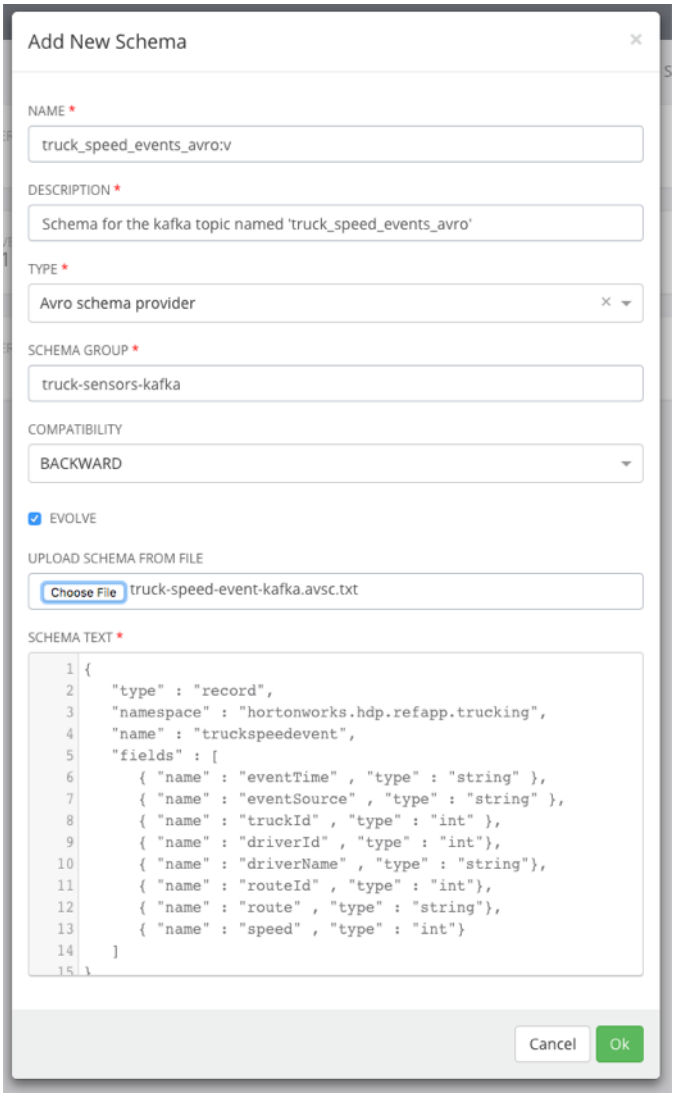
   - `Name = truck_speed_events_avro:v`

Note

The name of the schema must be the name of the kafka topic plus a suffix. The suffix is either a :v or :k because Kafka has the concept of a key and a value. A schema can be defined for either the key or the value or both. The suffix indicates what the schema is for. In this case, the :v indicates that this schema is for the values that go into the Kafka topic.

   - `Description = Schema for the kafka topic named 'truck_speed_events_avro'`

- `Type = Avro schema provider`
- `Schema Group = truck-sensors-kafka`
- `Compatibility: BACKWARD`
- Check the evolve checkbox
- To setup the schema, do the following
  - Download this schema and save it locally
  - Click the Choose File button and select the locally saved schema

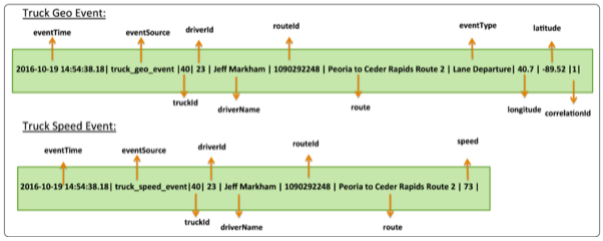


If you want to create these schemas programmatically using the Schema Registry client via REST rather than through the UI, examples can be found here.

3.3. Data Producer App generates events with schema key

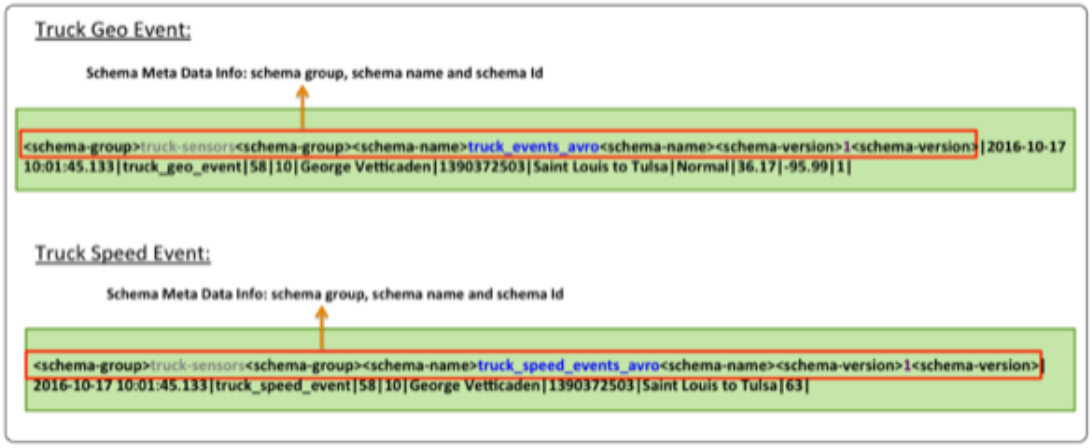
3.3.1. Raw Truck Event Stream

The following is a sample of a raw event generated by the trucking application:



3.3.2. Raw Truck Event Streams with Schema Meta Information

The following are the raw events with the schema metadata embedded in the event (annotated in red) streamed to files.



3.4. Nifi: Create a Flow App

To make things easier to setup, you can import the Nifi Template for this flow from the TP Bundle located under `$BUNDLE_LOC/Hortonworks-Streaming-Analytics-Manager-TP-Bundle/Nifi-and-Schema-Registry-Integration/Nifi-Templates` The below steps walks through the imported flow and what you have to configure/modify for your env.

3.4.1. Schema Registry Controller Service

1. Click on Flow Configuration Settings icon and select Controller Services tab

2. You will see the SchemaRegistryService. Edit the properties to configure the Schema Registry URL based on your env. You can find this value in the Sreaming Analytics Manager Service in Ambari for config called registry.url. An example of what url would look like is the following (http://$REGISTRY_SERVER:7788/api/v1)

3. Enable the controller service

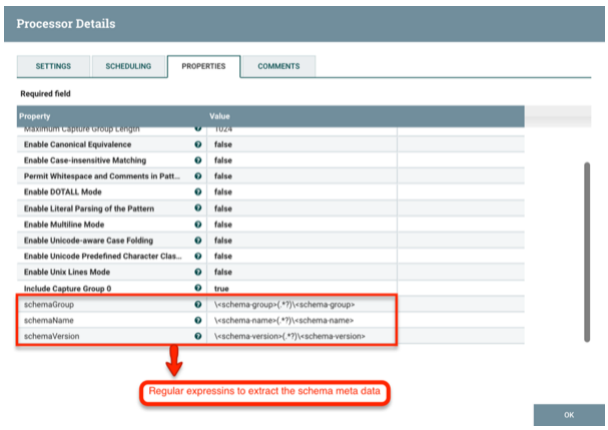
3.4.2. NiFi Ingests the Event

The first step in the NiFi flow is to ingest the raw events with the schema metadata. To perform this task, use a simple Tail processor to tail the file which contains both the geo events and speed events.

Make sure you modify the file property in the `TailFile` processor to point to the file that you will generate events to with the simulator in the subsequent section (e.g: /tmp/truck-sensor-data/telemetry-device-4.txt)

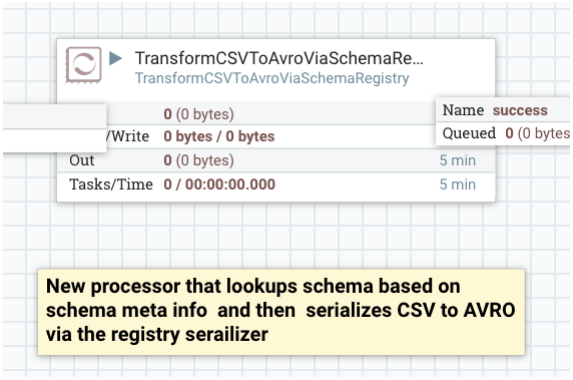
3.4.3. NiFi extracts Schema Metadata from the event

To extract the schema metadata into flowfile attributes, use the `ExtractText` processor with regular expressions to store the schema group, schema name, and schema version as attributes within the FlowFile.



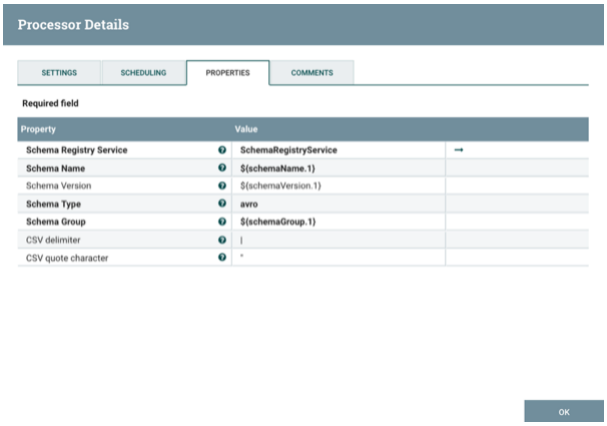
3.4.4. NiFi serializes an event to Avro using Schema Registry

The next step uses the new `TransformCSVToAvroViaSchemaRegistry` processor to serialize the CSV into Avro using Schema Registry.



The mandatory properties for the processor are:

- SchemaRegistryService: Reference to the schema registry controller service
- The schema metadata properties:
  - Schema Group
  - Schema Name
  - Schema Version
- Schema Type: The format you being using for serialization. Avro is the only supported type.

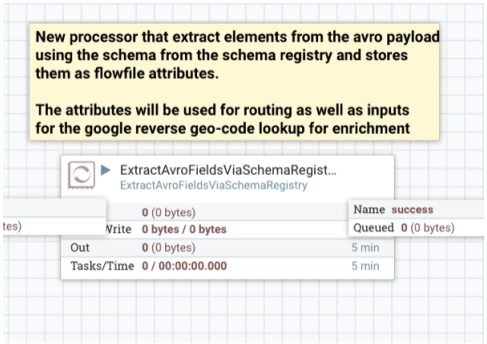


To serialize the CSV into Avro, the processor does the following:

- Using the schema metadata properties, the processor fetches the schema and the serializer from Schema Registry.
- With the schema, the processor creates an Avro GenericObject and serializes the object.

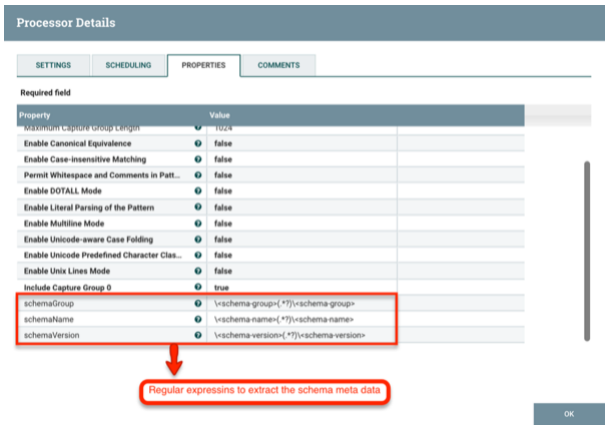
3.4.5. Extract values from the Avro object as attributes using Schema Registry

The next step is to extract values from the serialized Avro object using the schema, and to store them as flowfile attributes using the new `ExtractAvroFieldsWithSchemaRegistry` processor.



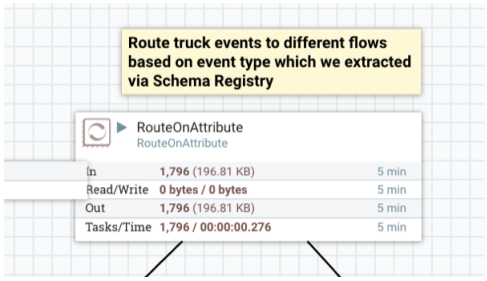
The static properties are the same as `TransformCSVToAvroViaSchemaRegistry` but this processor supports dynamic properties. With dynamic properties, the property name is the field name within the schema from which you want to retrieve a value, and the property value is the flowfile element name in which you want to store the value. In the below example, you extract the values of 3 fields from the truck_events_log schema:

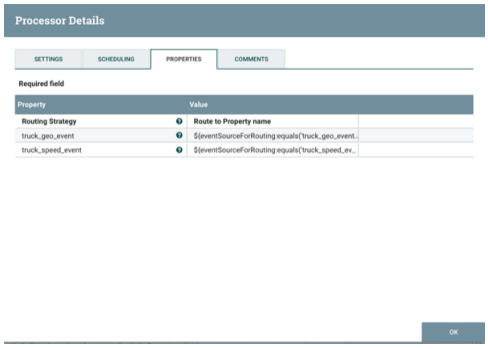
- eventSource – The value of this field is stored in a flowfile attribute called eventSourceForRouting. This flow file attribute is used for routing decisions downstream.
- latitude – The value of this field is stored in a flowfile attribute called latitudeForGeoCodeEnrichment. This flow file attribute is used as an input to the Google GeoCode API downstream.
- longitude – The value of this field is stored in a flowfile attribute called longitudeForGeoCodeEnrichment. This flow file attribute is used as an input to the Google GeoCode API downstream.



3.4.6. NiFi Makes Routing Decisions Based on Extracted Values

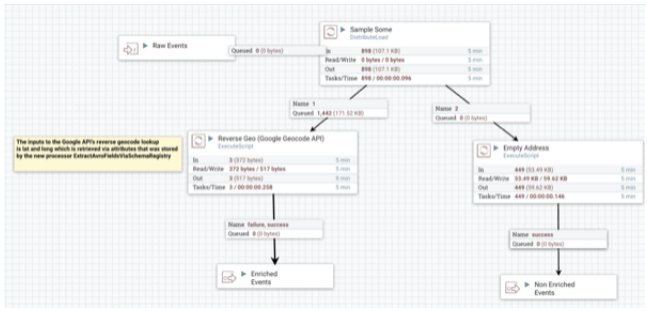
Now that you have saved the inputs for routing based decisions, use the RouteOnAttribute processor to route messages from the truck Geo Events and Speed Events sensors to different flows.



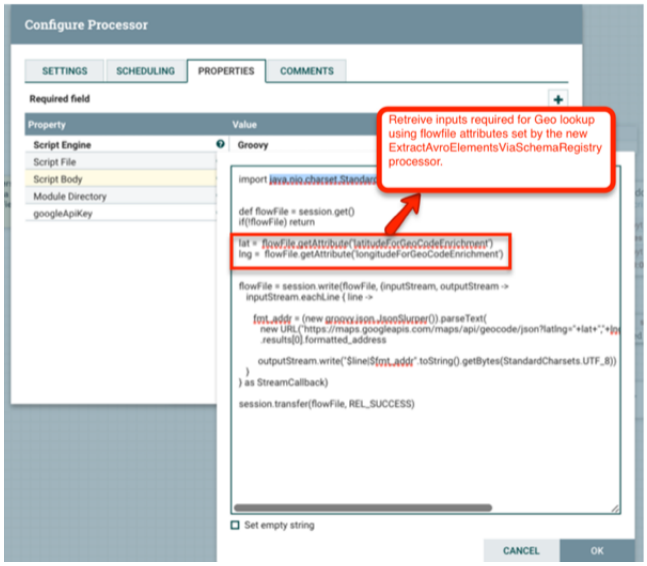


3.4.7. Enrich Geo Event with Address using the Google Geo-Code API

Next, use the Google Geo-Code API to enrich the truck geo events with the address associated with the coordinates in the event.

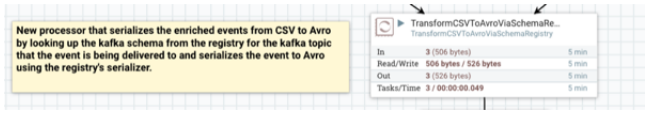


In the `ExecuteScript` processor groovy script below, note that we are retrieving the longitude and latitude values from the flowFile attributes to pass into the Geo-Code API.



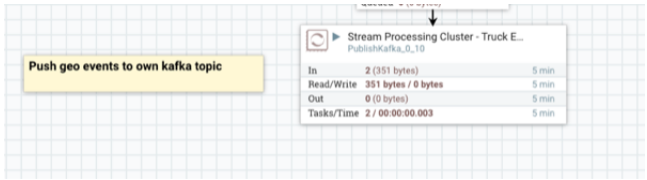
3.4.8. NiFi Serializes the Enriched Event using Schema Registry

Now that you have enriched the truck geo event data, publish the event into a Kafka topic. Because the Kafka topic has been registered with Schema in Schema Registry you can serialize the enriched event using the registered schema, using the new processor TransformCSVToAvroViaSchemaRegistry.



3.4.9. Publish to Kafka

Now that you have serialized the event with the schema registered for the Kafka topic, publish the serialized Avro event into Kafka using the `PublishKafka` processor.



Ensure that you have created two Kafka topics in your Kafka Cluster and modify the 2 Kafka Publisher processors to point to your kafka cluster by configuring correctly the 'Kafka Brokers" property.

3.4.10. Start the Nifi Flow

Start all the Nifi processors in the flow you imported and modified with respect to your environment.

3.5. Streaming Analytics Manager: Create a Stream Analytics App

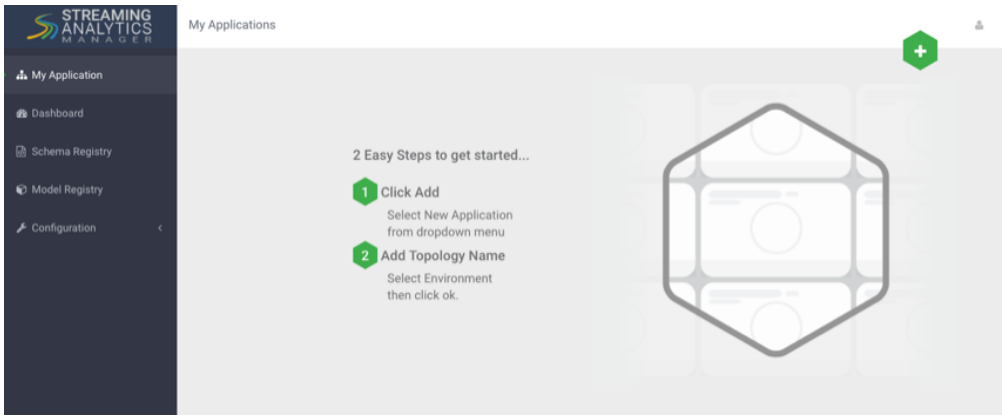
3.5.1. Create a Service Pool and Environment

Before you create an app, you have to create a Service Pool and then an Environment that you associate with an app. See the sections above on "Streaming Analytics Manager Environment Setup and Managing Stream Apps"

3.5.2. Create Your First App

The Streaming Analtyics Manager provides capabilities to the app developer for building streaming apps. You can go to the Stream Builder UI by select the "Streaming Analytics Manager" service in Ambari and under Quick Links select "SAM UI".

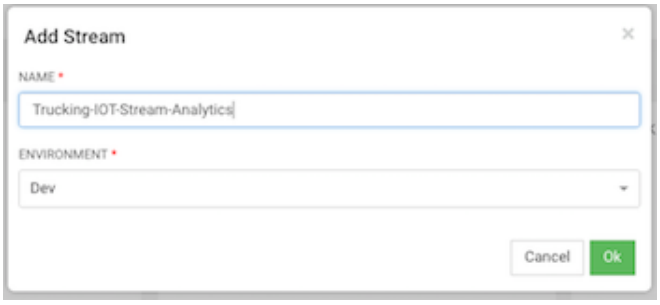
As the following diagram illustrates, creating a new stream app requires two steps: clicking the + icon, and then providing a unique name for the stream app and associating the app with an Environment.



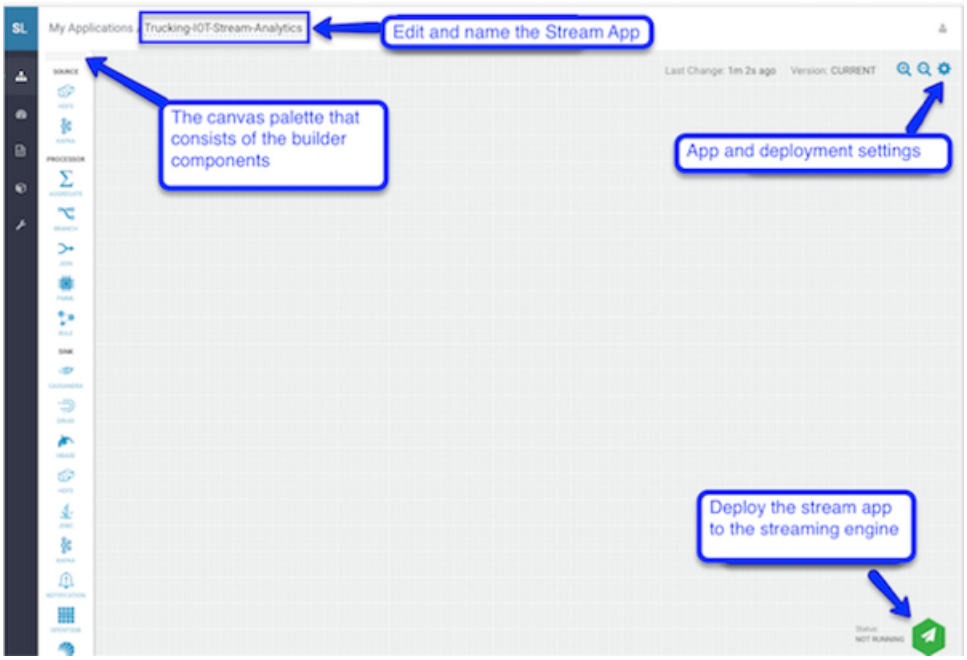
To create a new stream app:

1. Click the + icon on the "My Applications" dashboard and choose "New Application."

2. Specify the name of the stream app and the environment that the stream app will use. The name of the stream app should not have any spaces.



3. Streaming Analytics Manager displays the Stream Builder canvas. Builder components on the canvas palette are the building blocks used to build stream apps.



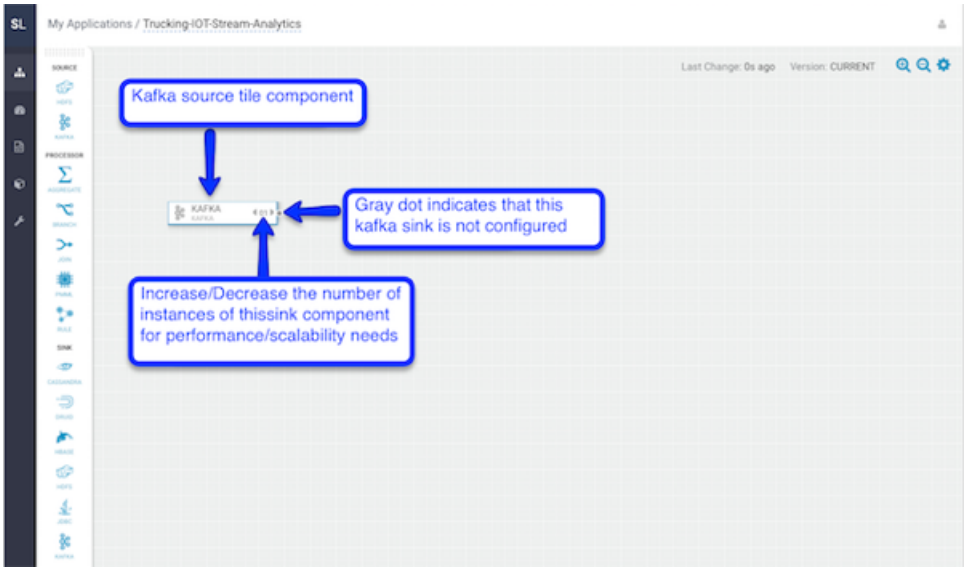
3.5.3. Creating and Configuring the Kafka Source Stream

The first step in building a stream app is to drag builder components to the canvas. As described in the Streaming Analytics Manager Overview chapter, Stream Builder offers four types of builder components: sources, processors, sinks, and custom components.

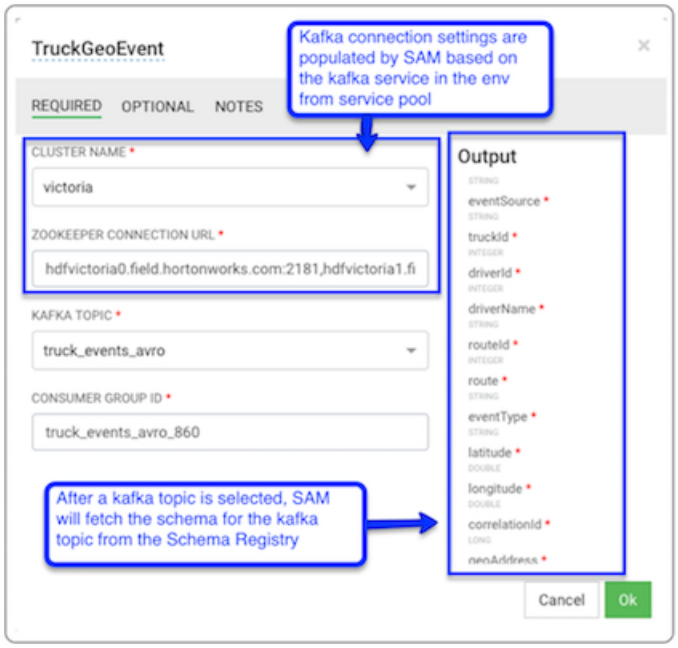
When you drag a builder component onto the canvas, it becomes a tile component in the new app. Configure the tile components by double-clicking each tile. In addition, you can set the number of run-time instances for a tile component by clicking the up arrow on the tile. When a tile component is configured correctly, Stream Builder displays a green dot.

Complete the following instructions to start building a stream app. The below steps walks through how to implement Requirement 4 of the use case.

1. A stream app must start with a Kafka source. Drag the Kafka builder component onto the canvas, creating a Kafka tile component:



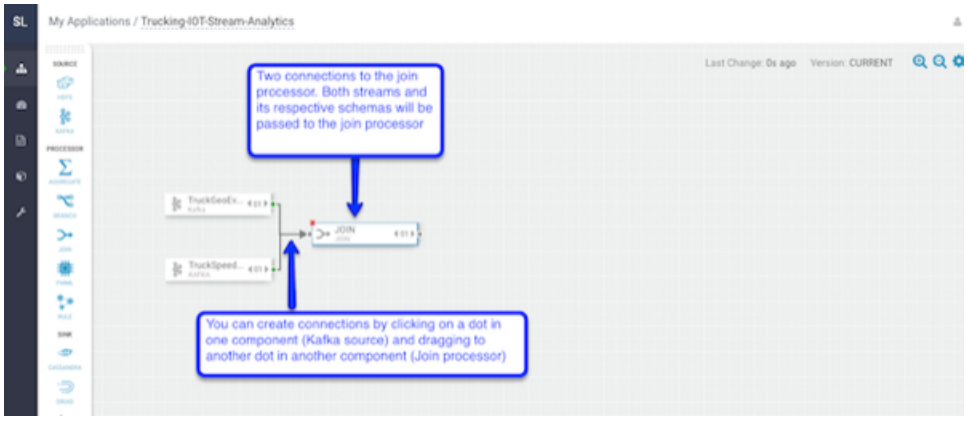
2. Double-click on the tile to begin configuring Kafka. After you specify a Kafka topic name, Streaming Analytics Manager communicates with the Schema Registry and displays the schema:



3.5.4. Connecting Components

To pass a stream of events from one component to the next, create a connection between the two components. In addition to defining data flow, connections allow you to pass a schema from one component to another.

The following example shows two connections: a connection from Kafka sink TruckGeoStream to the join processor, and a connection from the Kafka sink TruckSpeedStream to the same join processor.



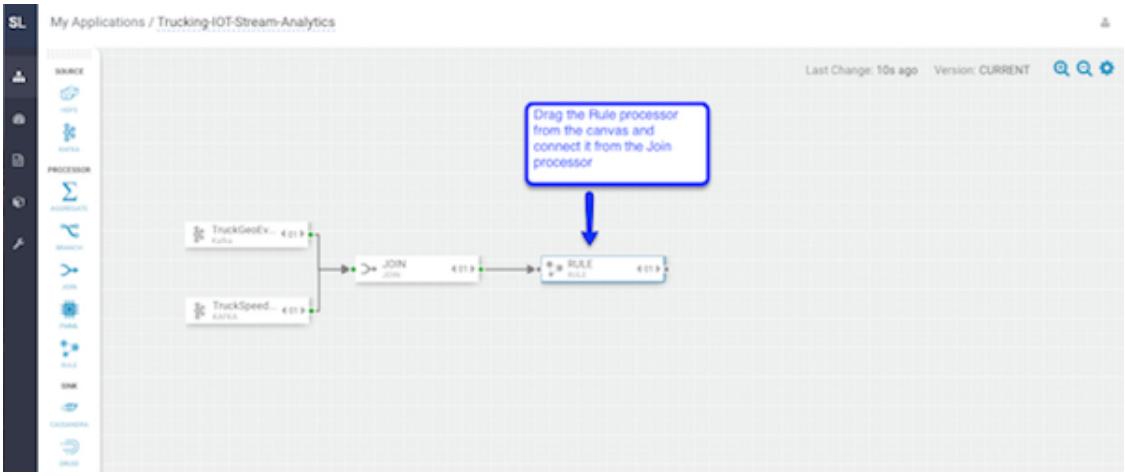
3.5.5. Joining Multiple Streams

Joining multiple streams is an important capability in Streaming Analytics Manager. After double clicking on the join processor, the following image shows how to configure a join processor that joins the truck geo-event stream with the speed event stream, based on Requirement 5 of the use case.

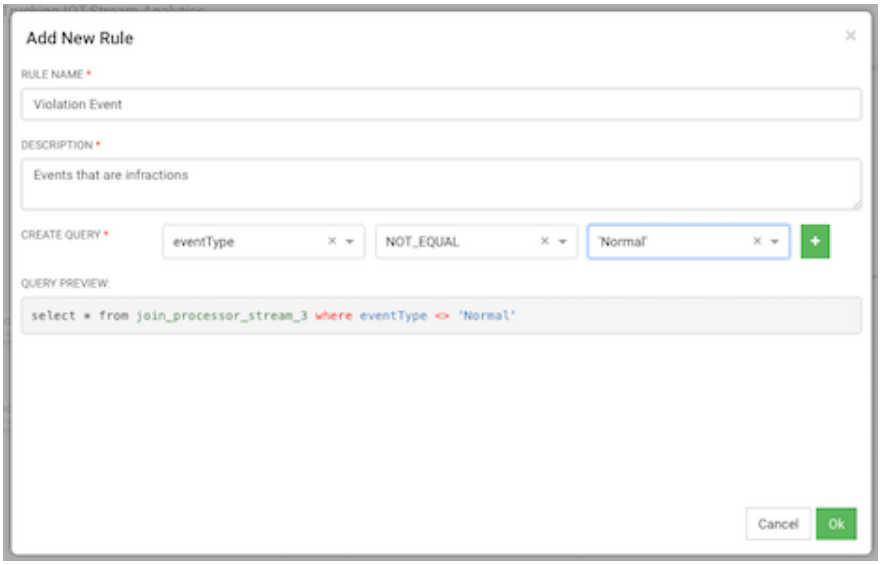
3.5.6. Filtering Events in a Stream using Rules/SQL

Streaming Analytics Manager provides powerful capabilities to filter events in the stream. It uses a Rules Engine, which translates rules into SQL queries that operate on the stream of data. The following steps demonstrate this, implementing Requirement 6 of the use case.

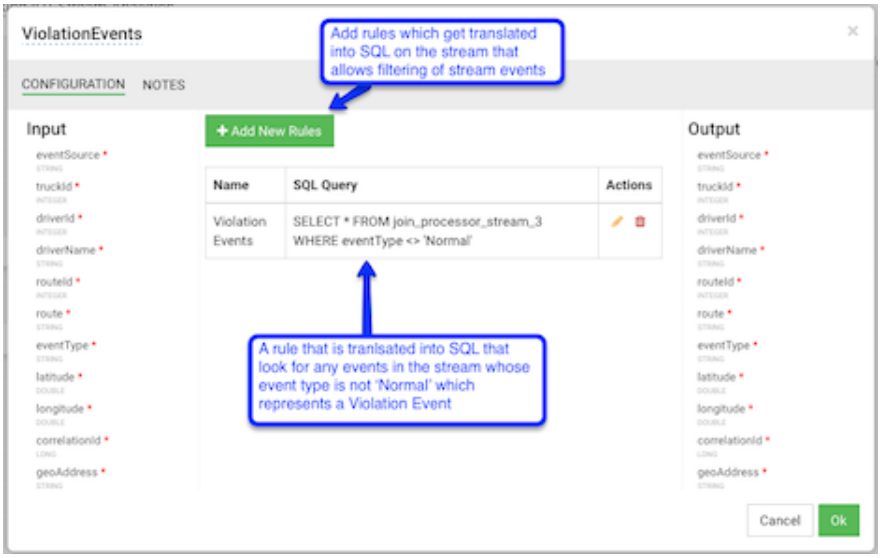
1. Drag the Rule processor to the canvas and connect it from the join processors



2. Double click on the Rule processor, click the "Add new Rules" button, and create a new rule:



3. Click OK to save the new rule.

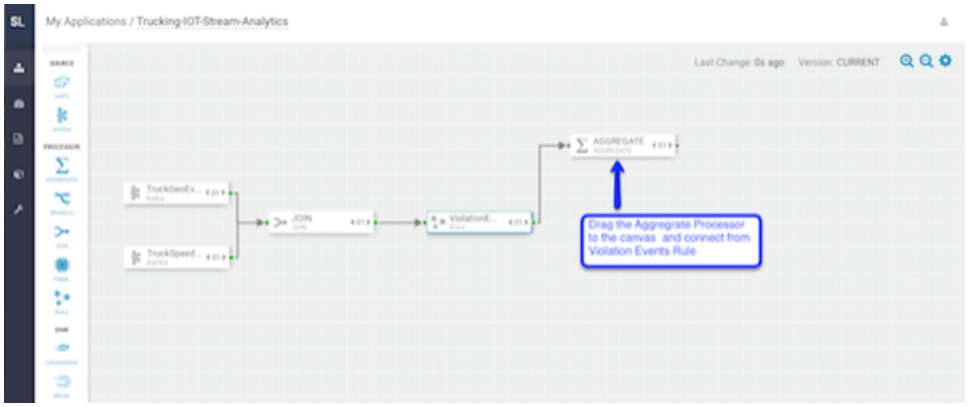


3.5.7. Using Aggregate Functions over Windows

Windowing is the ability to split an unbounded stream of data into finite sets based on specified criteria such as time or count, so that you can perform aggregate functions (such as sum or average) on the bounded set of events. Streaming Analytics Manager exposes these capabilities using the Aggregate processor. The Aggregate processor supports two window types, tumbling and sliding windows. The creation of a window can be based on time or count.

The following images show how to use the Aggregate processor to implement Requirement 8 of the use case.

1. Drag the Aggregate processor to the canvas and connect it from the Rule processor:



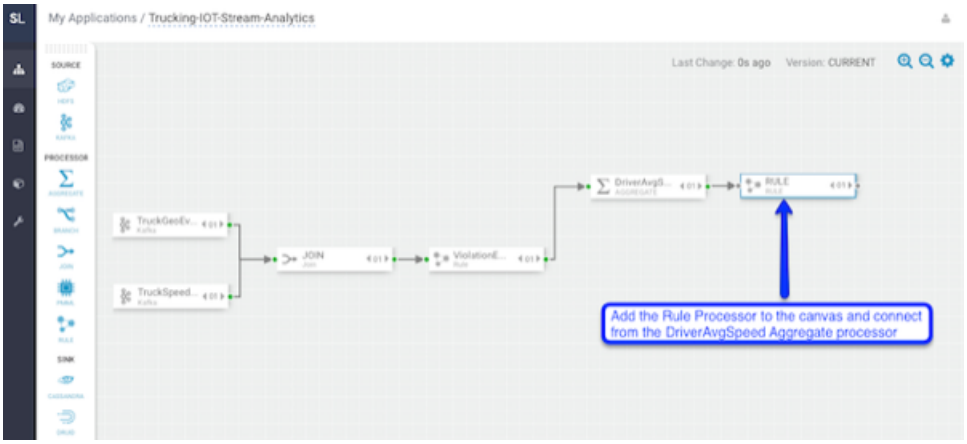
2. Double-click on the Aggregate processor, and configure it to calculate the average speed of driver over a three-minute duration.

3.5.8. Implementing Business Rules on the Stream

In the preceding section, you used the Aggregate processor to calculate the average speed of drivers over three-minute intervals.

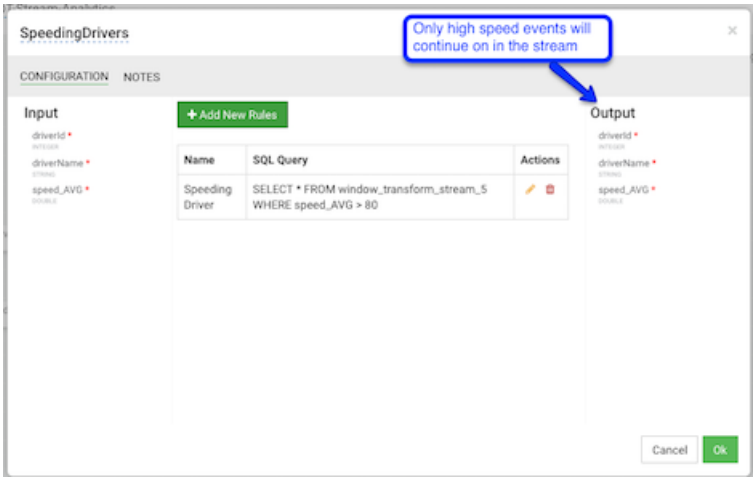
Next you will implement the business rule to detect high speeding drivers. "High speed" is defined as greater than 80 miles per hour over a three-minute time window. This step partially implements Requirement 8 of the use case.

1. Drag the Rule processor onto the canvas and connect to it from the DriverAvgSpeed Aggregate processor:



2. Configure the business rule as follows:

3. The fully configured business rule should look similar to the following. Only high speed events continue on in the stream.



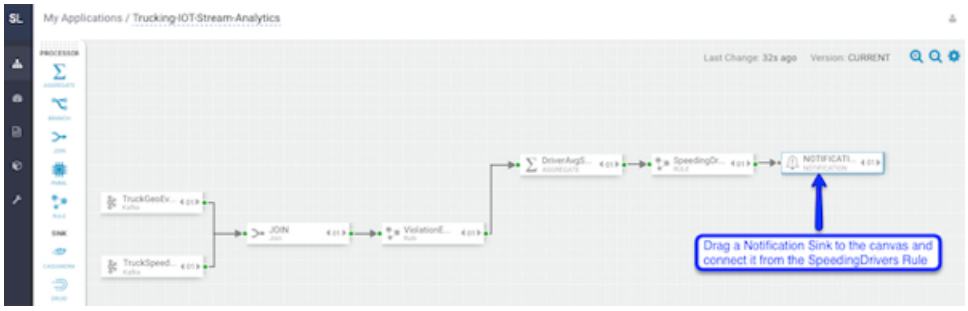
3.5.9. Creating Alerts with Notifications Sink

The Notifications sink allows you to create alerts.

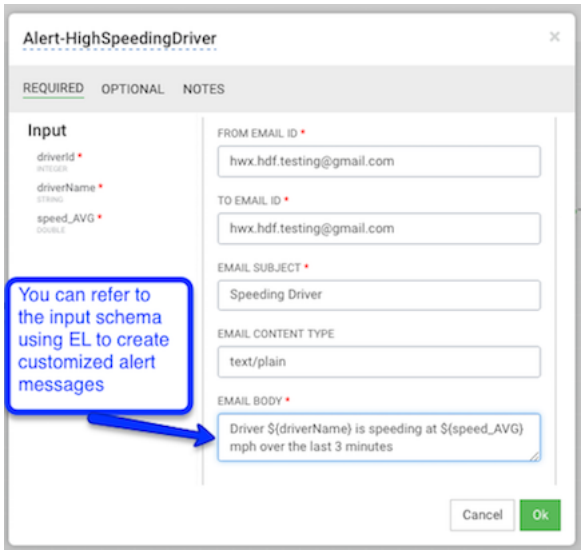
The Notification sink supports email alerts, and it is extensible– you can can plug in other types of notifications.

The following steps demonstrate how to create email alerts when drivers are speeding.

1. Drag the Notifications sink to the canvas and connect to it from the SpeedingDrivers Rule.



2. Configure the Notifications sink to generate email alerts for high speeding drivers.



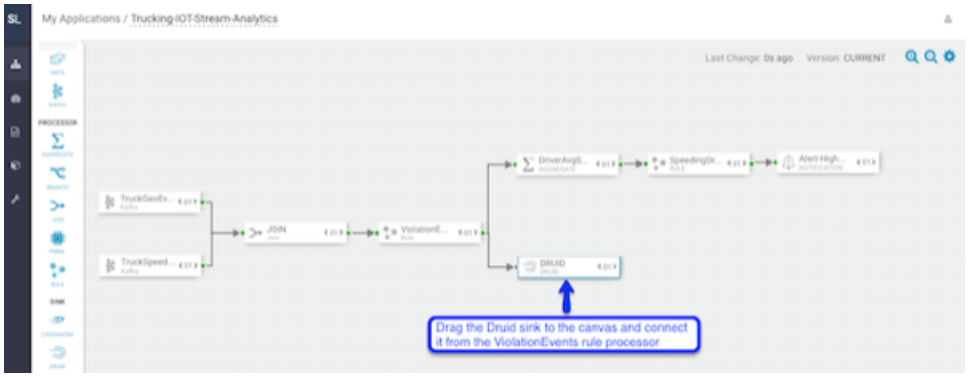
3.5.10. Streaming Violation Events to an Analytics Engine/ Druid for Descriptive Analytics

So far, you have implemented each of the requirements in the use case except Requirement 7:

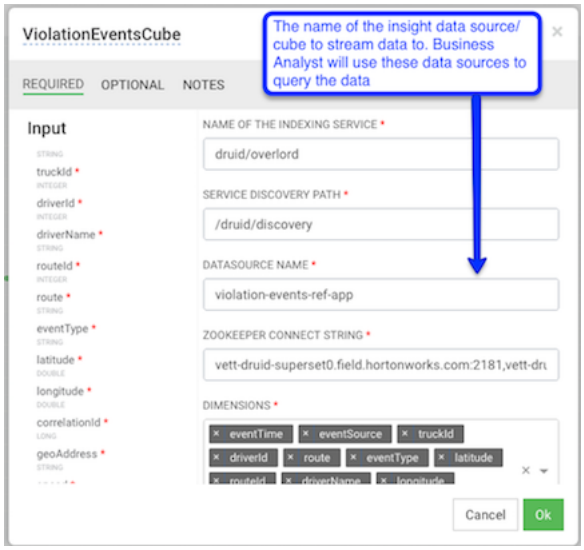
All infraction events need to be available for descriptive analytic (dash-boarding, visualizations, etc.) by a business analyst. The analyst needs the ability to do analysis on the streaming data.

The analytics engine in Streaming Analytics Manager is powered by Druid. The following steps show how to stream data into Druid, so that a business analyst can use the Stream Insight Superset module to generate descriptive analytics.

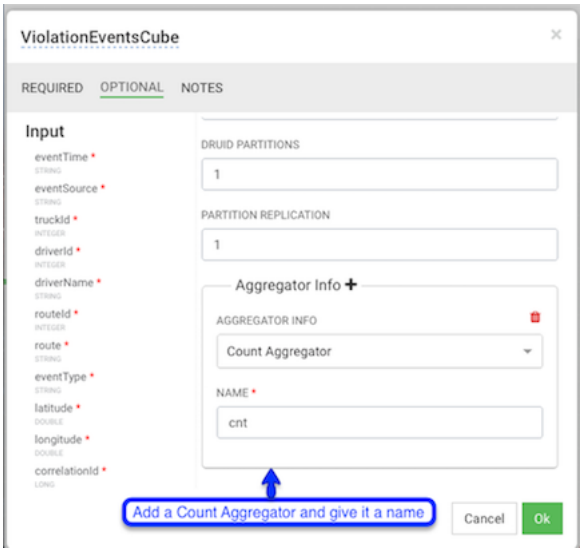
1. Drag the Druid processor to the canvas and connect it from the ViolationEvents Rule processor:



2. Configure the Druid processor. You can edit the ZooKeeper connect string in the advanced section of the Druid Service in Ambari, under the property `druid.zk.service.host`.



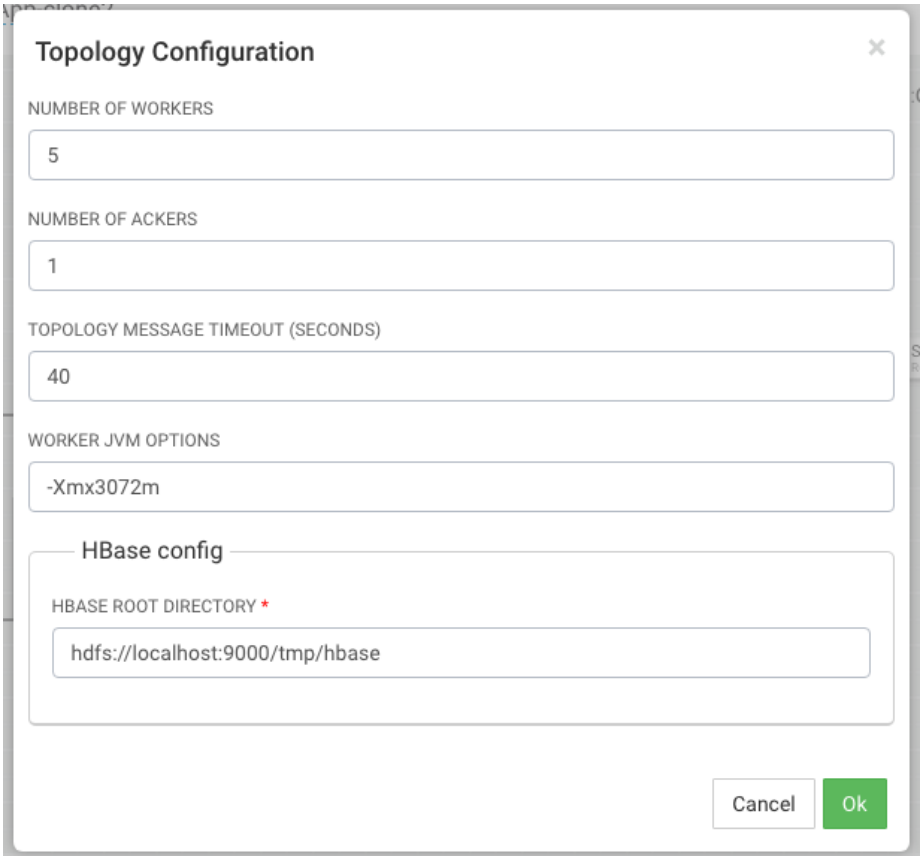
3. Configure the following settings, under the "OPTIONAL" menu:



3.5.11. Deploying a Stream App

3.5.11.1. Configure Deployment Settings

Before deploying the application, it is important to configure deployment settings such as JVM size, number of ackers, and number of workers. Because this topology uses a number of joins and windows, you should increase the JVM heap size for the workers. Click the gear icon on the top right corner of the canvas, and increase the number of workers (e.g: 5) and increase the JVM heap memory (-Xmx3072m)



3.5.11.2. Deploy the App

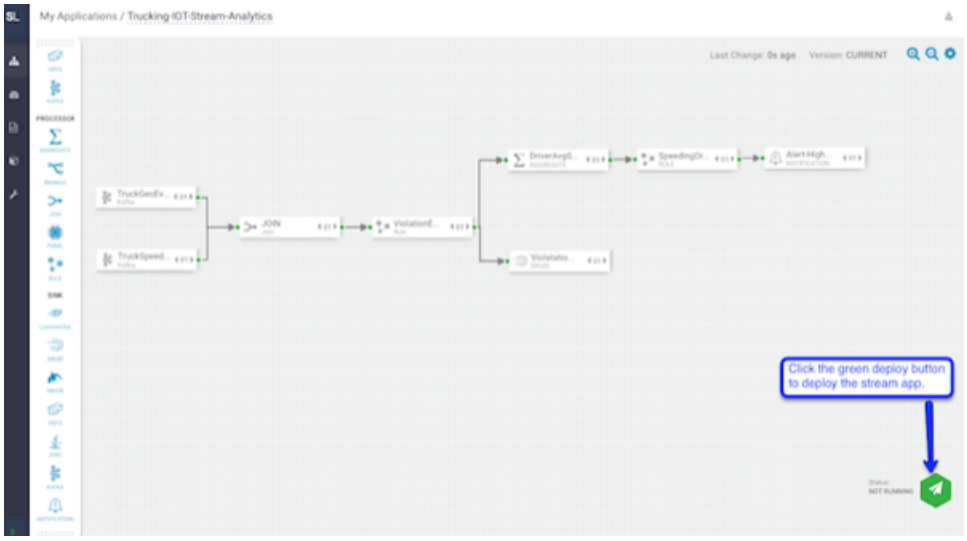
After the app's deployment settings has been configured, click the Deploy button on the lower right of the canvas. During the deployment process, Streaming Analytics Manager completes the following tasks:

1. Construct the configurations for the different big data services used in the stream app.

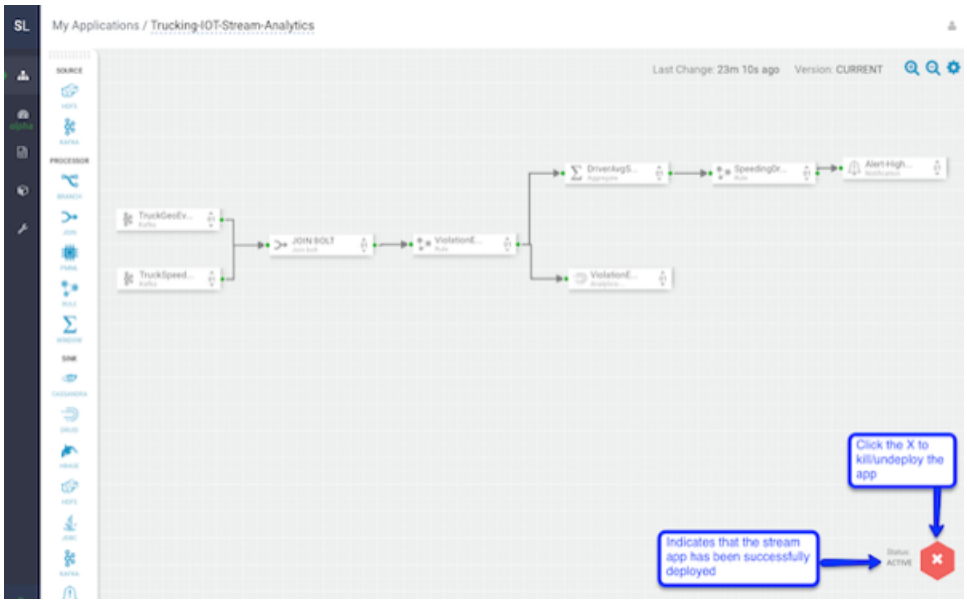
2. Create a deployable jar of the streaming app.

3. Upload and deploy the app jar to streaming engine server.

The stream app is deployed to a Storm cluster based on the Storm Service defined in the Environment associated with the app.



After the application has been deployed successfully, Streaming Analytics Manager notifies you and updates the status to Active, as shown in the following diagram.



3.6. Running the Stream Simulator

Now that we have developed and deployed the Nifi Flow App and the Stream Analytics App, we are ready to run a data simulator that generates truck geo events and sensor events for the apps to process..

To generate the raw truck events with schema metadata information, perform the following steps:

1. Download the Data-Loader.zip. Unzip it and copy it to the node where you are running NiFi. Lets call the directory you unzip it to as: $DATA_LOADER_HOME. Then perform the following:

```
tar -zxvf $DATA_LOADER_HOME/routes.tar.gz

nohup java -cp \
data-loader-jar-with-dependencies.jar \
hortonworks.hdp.refapp.trucking.simulator.SimulationRunnerApp \
20000 \
hortonworks.hdp.refapp.trucking.simulator.impl.domain.transport.Truck \
hortonworks.hdp.refapp.trucking.simulator.impl.collectors.
FileEventWithSchemaInfoCollector \
1 \
/root/workspace/Data-Loader/routes/midwest/ \
10000 \
/tmp/truck-sensor-data/telemetry-device-4.txt &
```

2. Tail the file (`/tmp/truck-sensor-data/telemetry-device-4.txt`) to which you are writing, to view the generated

3.7. Stream Operations

Previous sections showcased features of the Stream Builder that allow app developers to build and deploy streaming applications.

The Stream Operation view provides management of the stream apps, including the following:

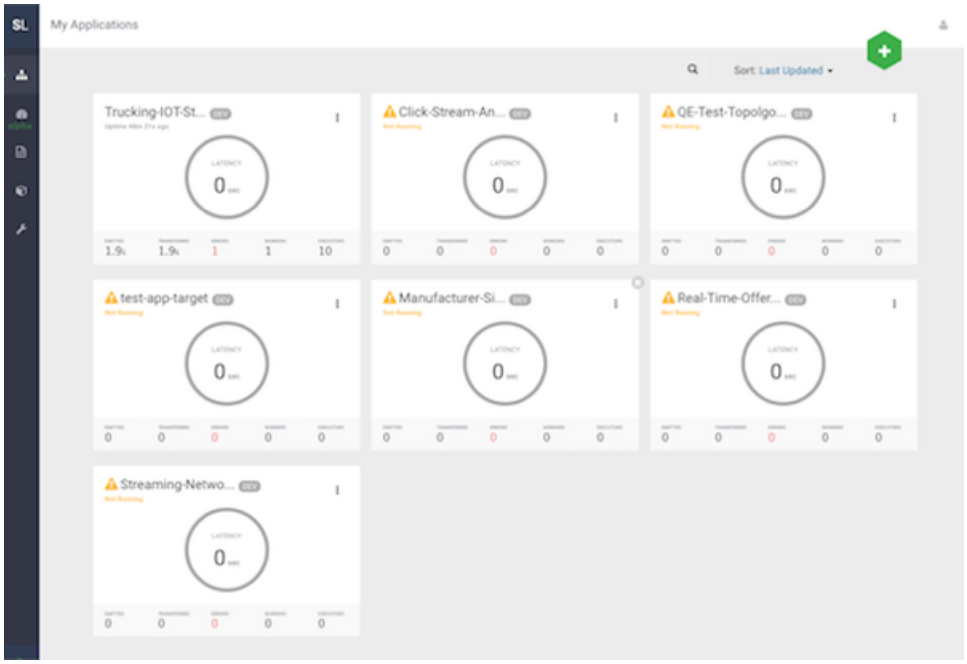
- Application lifecycle management: start, stop, edit, delete
- Application performance metrics
- Troubleshooting, debugging
- Exporting and importing apps

3.7.1. My Applications View

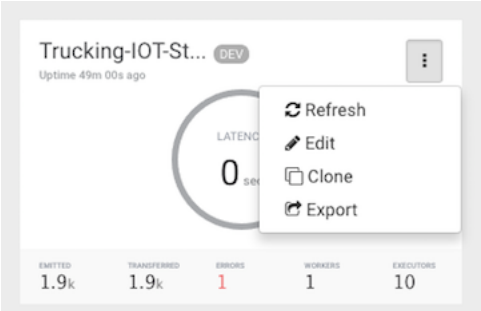
Once a stream app has been deployed, the Stream Operations displays operational views of the app.

One of these views is called "My Application" dashboard.

To access the application dashboard in Streaming Analytics Manager, click “My Application” tab (the hierarchy icon). The dashboard displays all apps built using Streaming Analytics Manager:



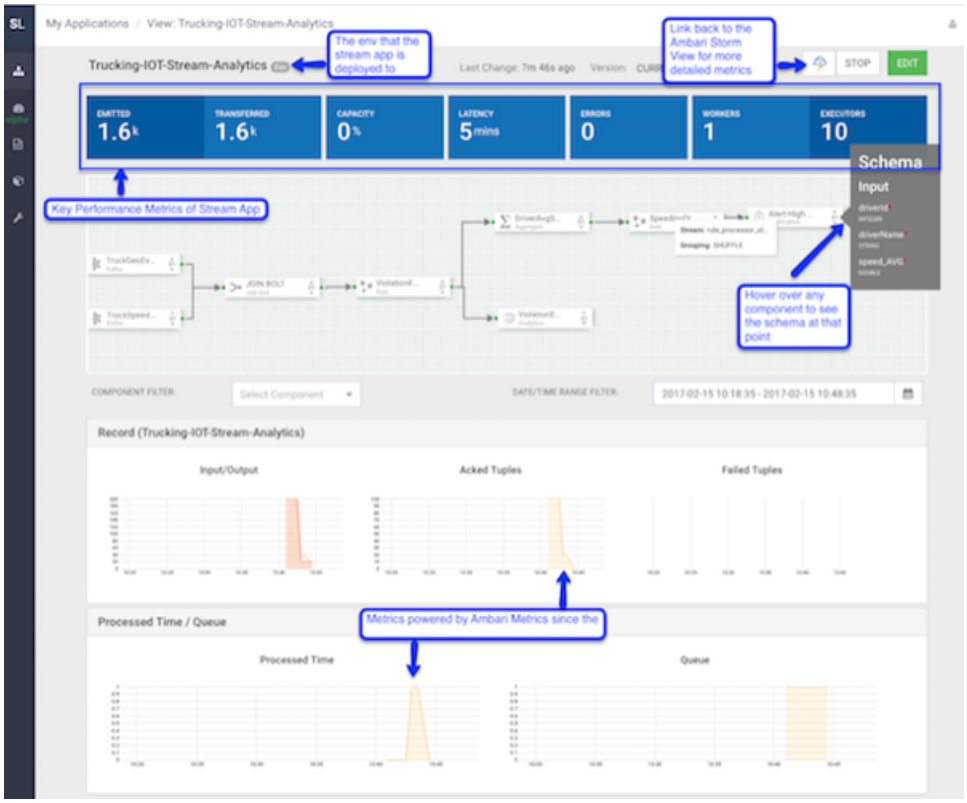
Each stream app is represented by an app tile. Hovering over the app tile provides status, metrics, and actions you can perform on the stream app.



3.7.2. Application Performance Monitoring

To view application performance metrics (APM) for the application, clicking on the app name on the app tile.

The following diagram describes elements of the APM view.

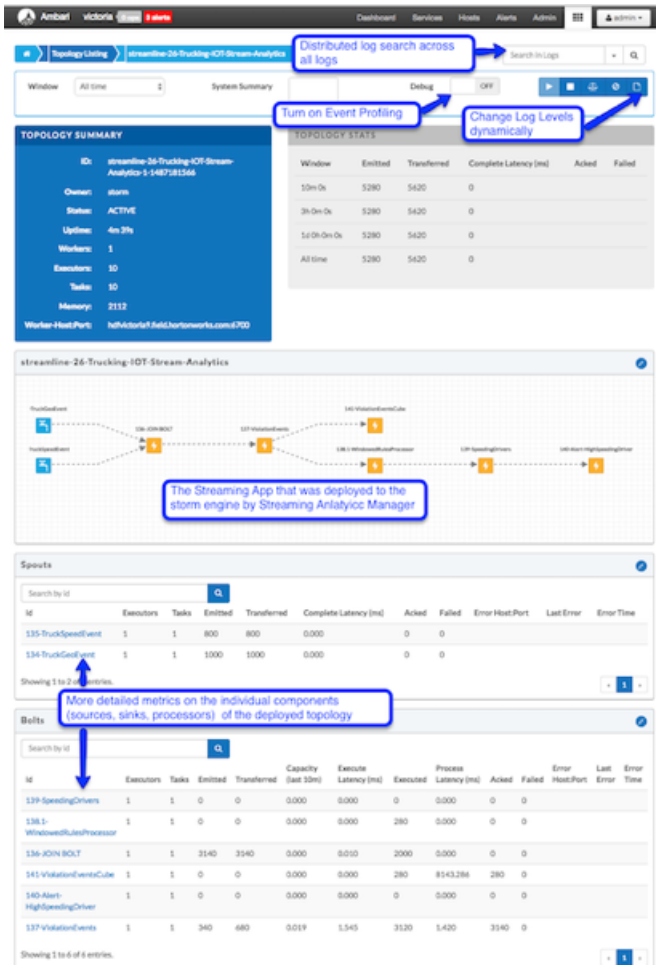


3.7.3. Troubleshooting and Debugging a Stream App

At the top right corner of the APM, there is a Storm icon that takes you to the Storm Ambari view.

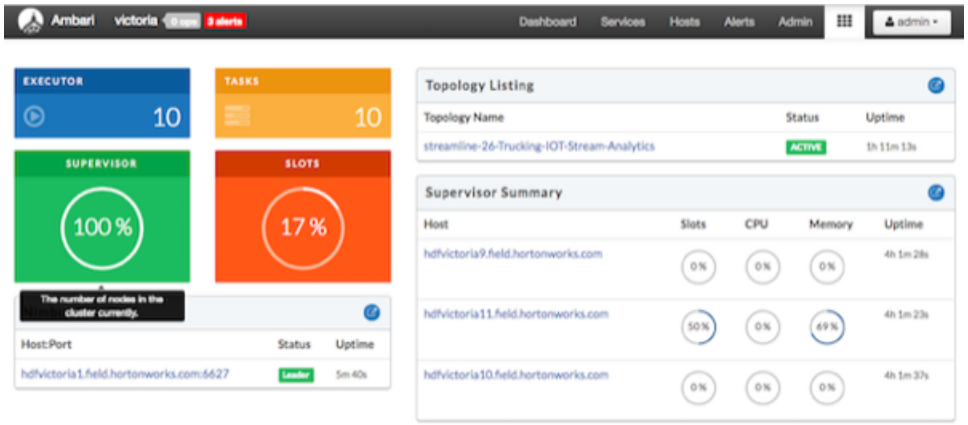
The Storm Ambari View provides the following capabilities for deeper troubleshooting and debugging:

- Topology View and Metrics: shows a visual representation of the deployed topology and topology level Metrics.
- Distributed Log Search: allows users to search all logs across supervisor machines for a topology; results can include zipped logs.
- Dynamic Log Levels: allows Users and Administrators to dynamically change the log level settings for a running topology.
- Topology Event Inspector: allows viewing of tuples flowing through the topology along with the ability to turn on/off debug events without having to stop/restart the entire topology.
- Dynamic Worker Profiling: allows users to request worker profile data directly from the Storm UI (Heap Dumps, JStack Output, JProfile).



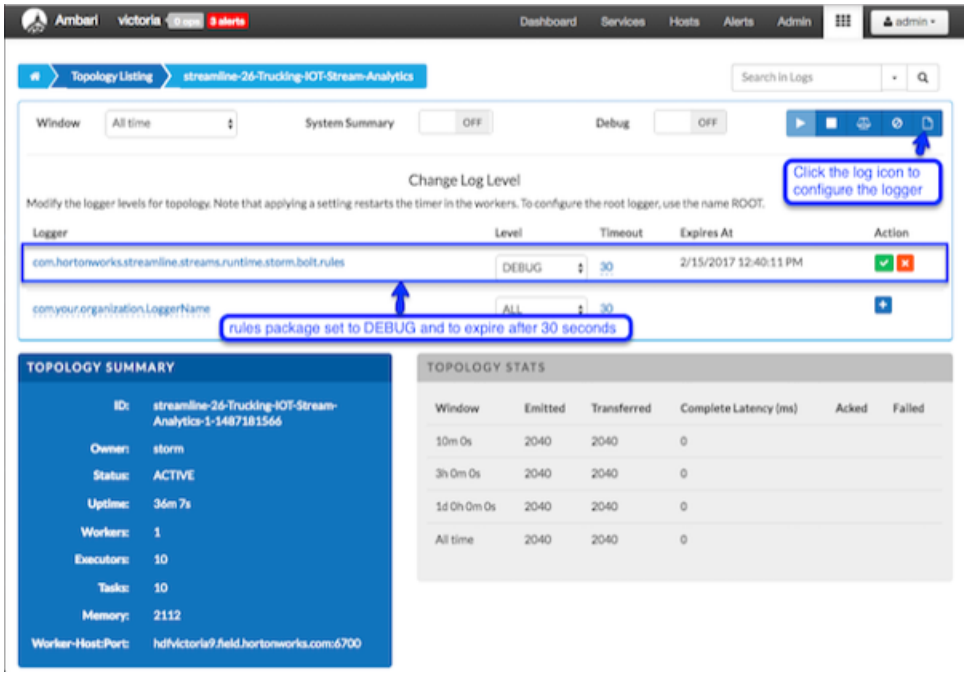
3.7.3.1. Streaming Engine Infrastructure Metrics

The following dashboard shows infrastructure metrics for the streaming engine used; in this case, it shows details about the Storm cluster.



3.7.3.2. Changing Log Levels Dynamically and with Expiration Policies

When debugging a stream application, the ability to change the log dynamically is a powerful troubleshooting feature. However, since typical stream applications handle millions of events per second, changes to log levels can impact performance unless safeguards such as expiration policies are defined. The following diagram shows how to change log levels with expiration policies.

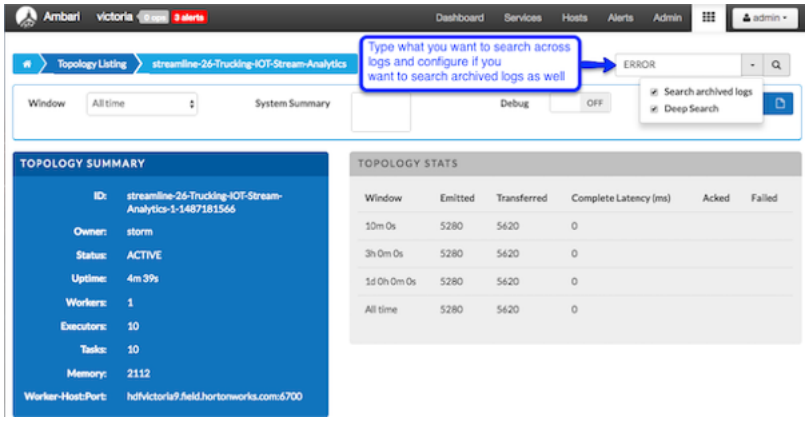


3.7.3.3. Distributed Log Search

Storm is a distributed streaming engine, which means that many worker nodes can be used to power the streaming application. Because it has a distributed architecture, logs are distributed across the cluster on many worker nodes. Searching for log data across workers can be a painful process. With distributed log search, however, you can search across all logs located across all worker nodes.

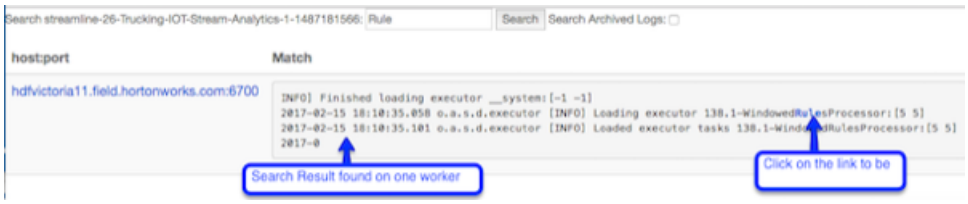
The following steps describe how to use distributed log search.

1. Type your search string in the distributed log search text box:



2. Review the results.

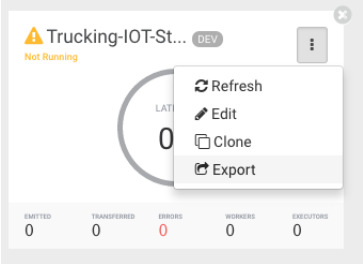
3. Click on the link to navigate to the exact location in the log file.



3.7.4. Exporting and Importing Stream Apps

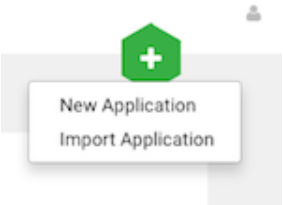
Service pool and environment abstractions combined with import and export capabilities allow you to move a stream app from one environment to another easily.

To export a stream app, click the Export icon on the "My Application" dashboard. This downloads a JSON file that represents your streaming app.

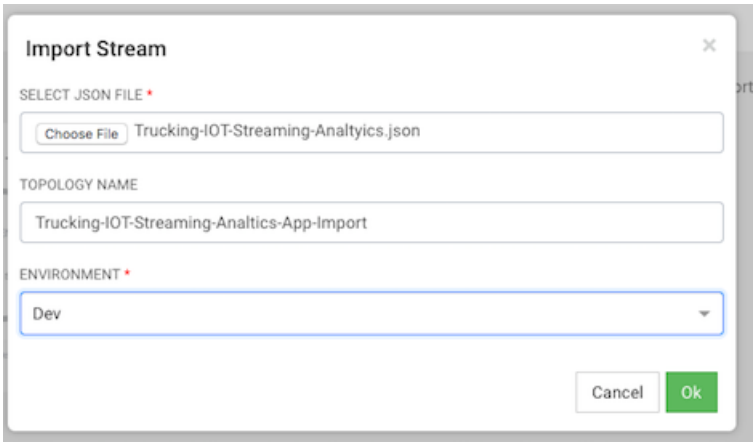


To import a stream app that was exported in JSON format:

1. Click on the + icon in My Applications View and select import application:



2. Select the JSON file that you want to import, provide a unique name for the app and specify which environment to use.



You can import a sample app from the TP Bundle located under `$BUNDLE_LOC/Hortonworks-Streaming-Analytics-Manager-TP-Bundle/Stream-App-Examples`.

4. Creating Visualizations: Insight Slices

A business analyst can create a wide array of visualizations to gather insights on streaming data. The platform supports over 30+ visualizations the business analyst can create. For visualization examples, see the Gallery of Superset Visualizations.

The general process for creating and viewing visualizations is as follows:

1. Whenever you add new data sources to Druid via a Stream App, perform the **Refresh Druid Metadata** action on the SuperSet menu

2. Using the Superset Stream Insight UI, create one or more "slices". A slice is one business visualization assoicated with a data source (e.g: Druid cube)

3. Using the Dashboard menu, add the slices to your dashboard and organize their layout.

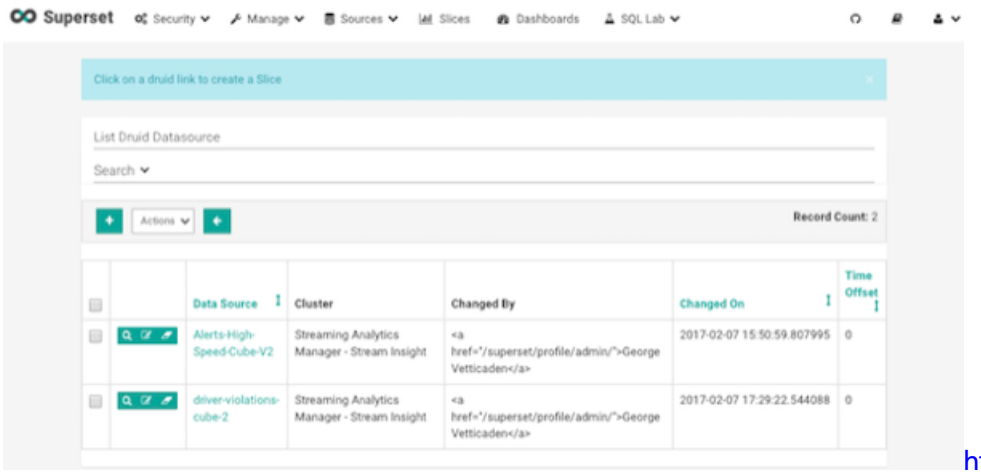
4.1. Creating Insight Slices

The following steps demonstrate a typical flow for creating a slice:

1. Choose **Slices** on the Menu.

2. Click + to create a new Slice.

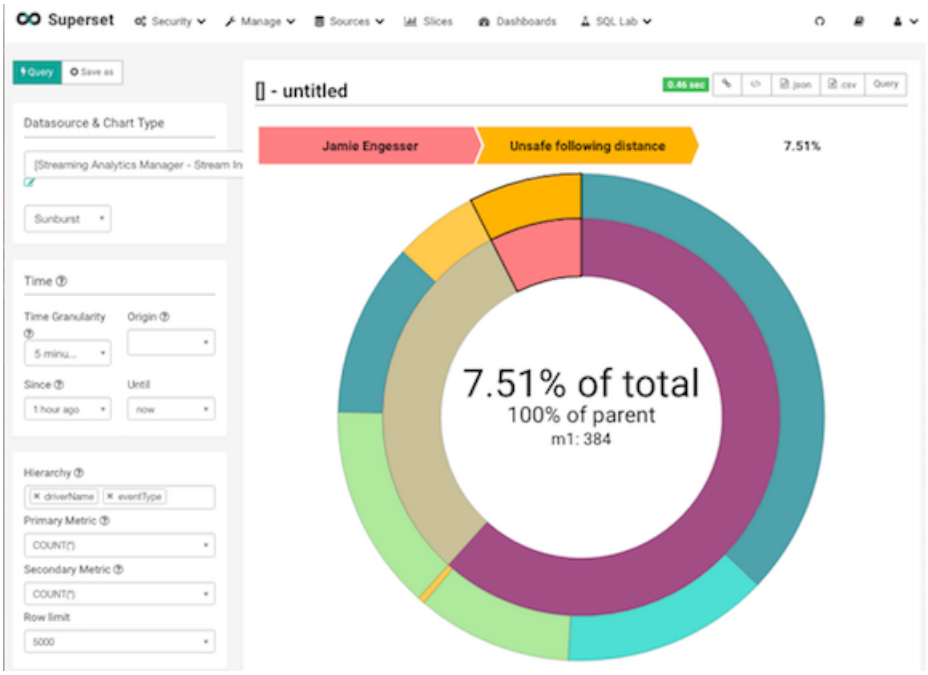
3. Select the Druid Data Source that you want to use for the new visualization:



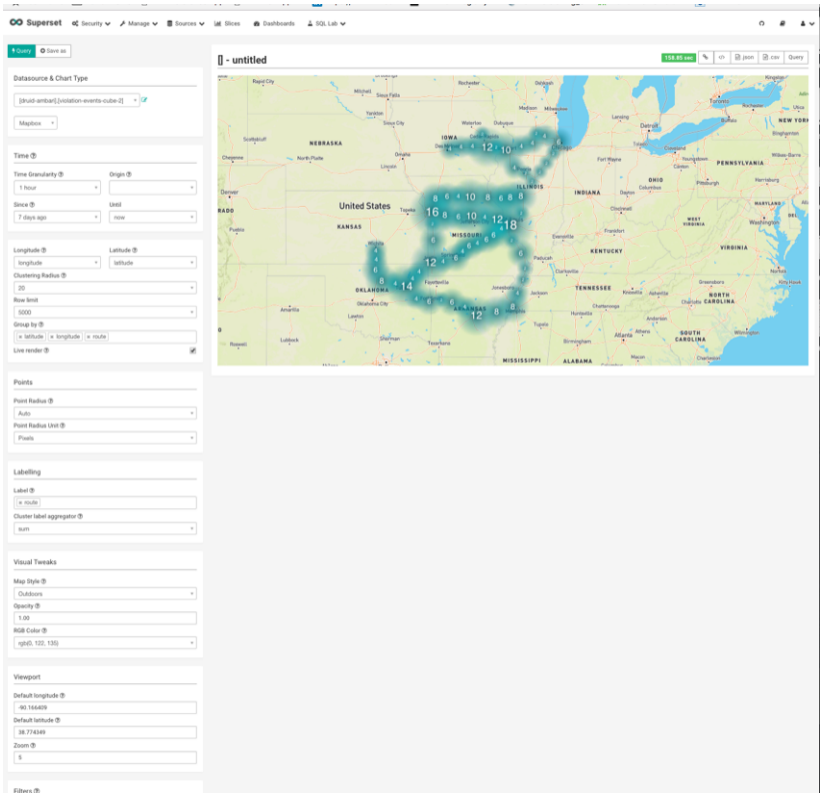
https://www.mapbox.com/

4. Select a Chart Type from the menu.

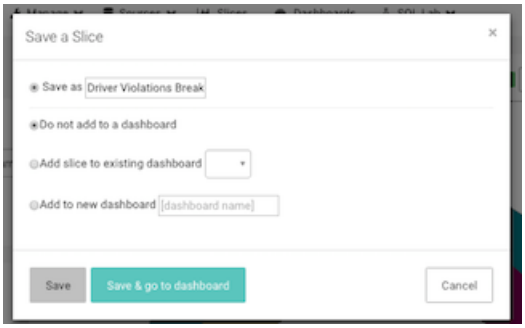
   This example creates a "Sunburst" visualization. Configure the chart and click **Execute Query**.



5. Another visualization could be integration with MapBox



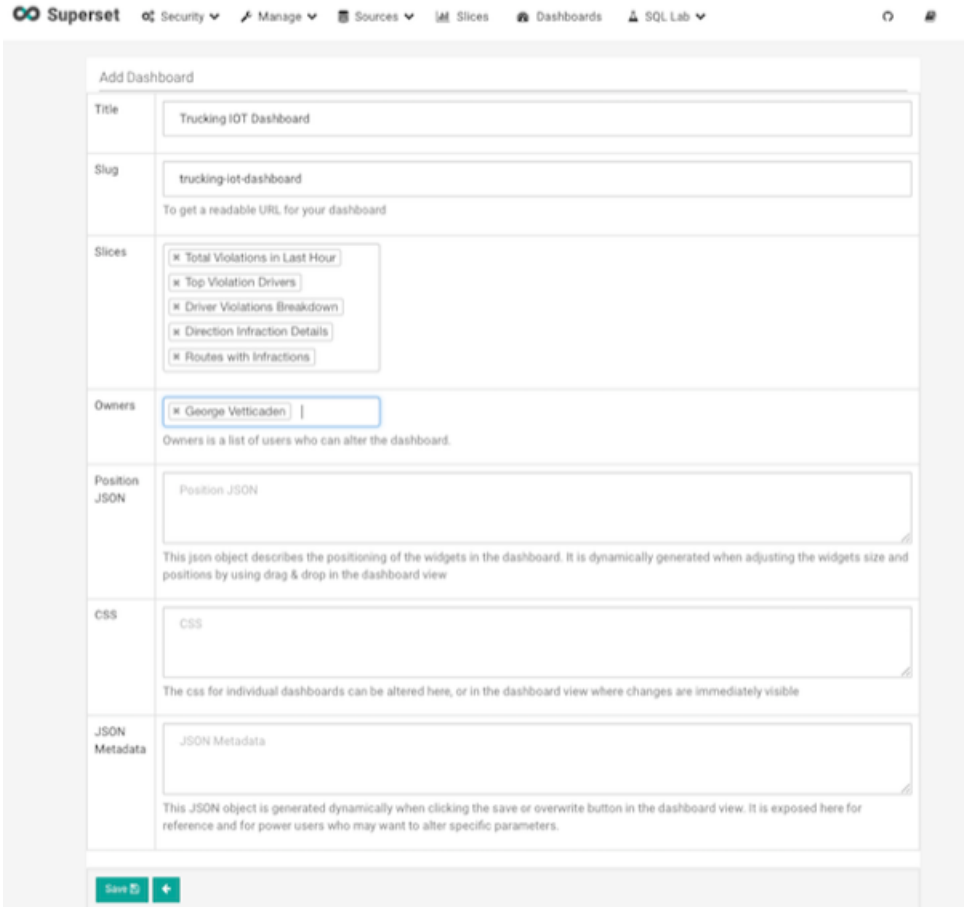
6. To save the slice, specify a name and name and click **Save**.



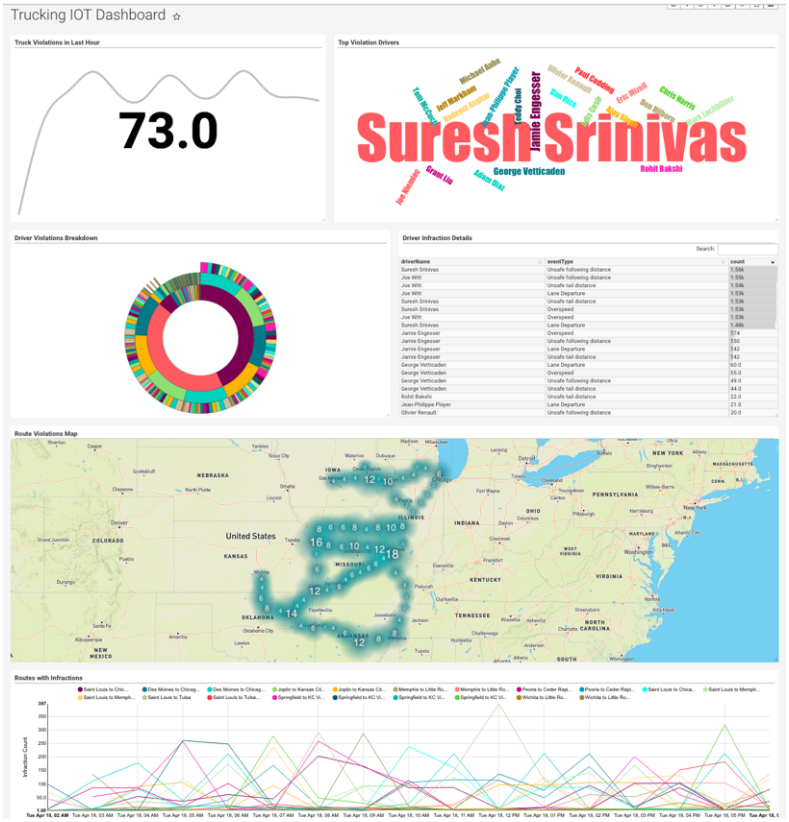
4.2. Adding Insight Slices to a Dashboard

After you create slices, you can organize them into a dashboard:

1. Click the Dashboard menu item.
2. Click + to create a new Dashboard.
3. Configure the dashboard: specify a name and the slices to include in the dashboard.



4. Arrange the slices on the dashboard as desired, and then click **Save**. The following image shows a sample dashboard.



5. Adding Custom Builder Components

Streaming Analytics Manager supports the ability to add custom components using the Streaming Analytics Manager SDK.

Follow these steps to build custom components:

1. Implement a custom processor using the SDK, and package it into a jar file with all of its dependencies.

   a. Create a new maven project using this maven pom file as an example.

   b. To implement a custom processor, implement the following interface:

      `org.apache.streamline.streams.runtime.CustomProcessorRuntime`.

      See the WebSocketSink class for an example.

   c. Package the jar file with all dependencies, by running the following commands:

      ```
      mvn clean package
      mvn assembly:assembly
      ```

   d. In the target directory you should have an uber jar that ends with `jar-with-dependencies.jar`. You will use this jar file in the next step.

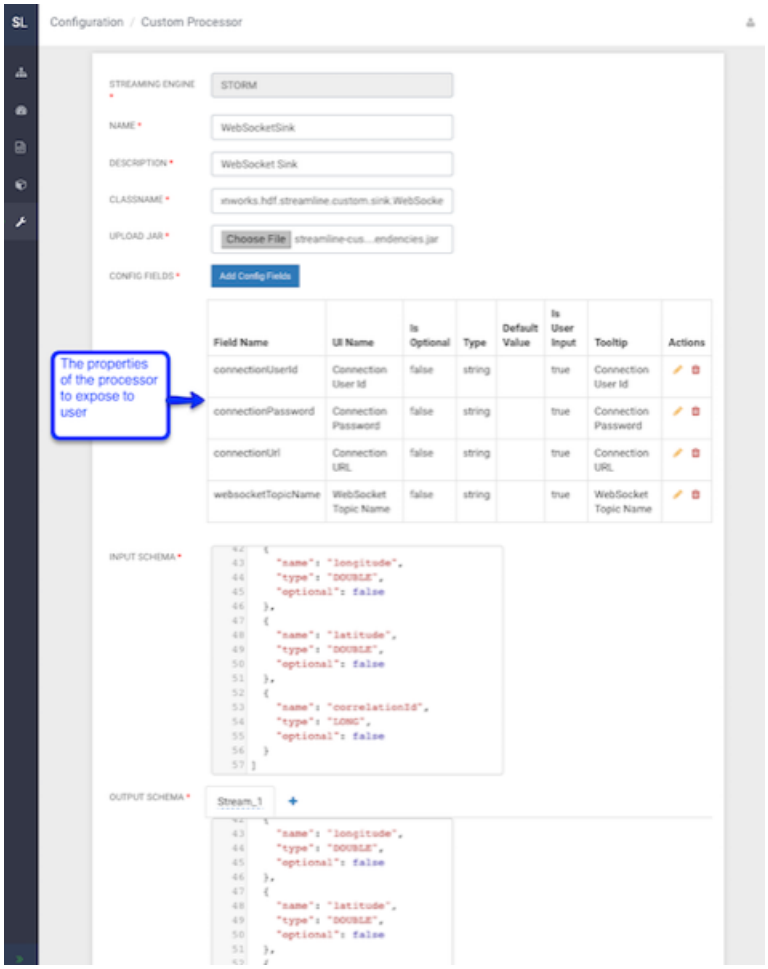
2. Register the custom processor in Streaming Analytics Manager.

   a. In Streaming Analytics Manager under the Configuration tab, select "Custom Processors."

   b. Click the + icon to add a new processor.

   c. Enter details for the custom processor, and upload the processor.

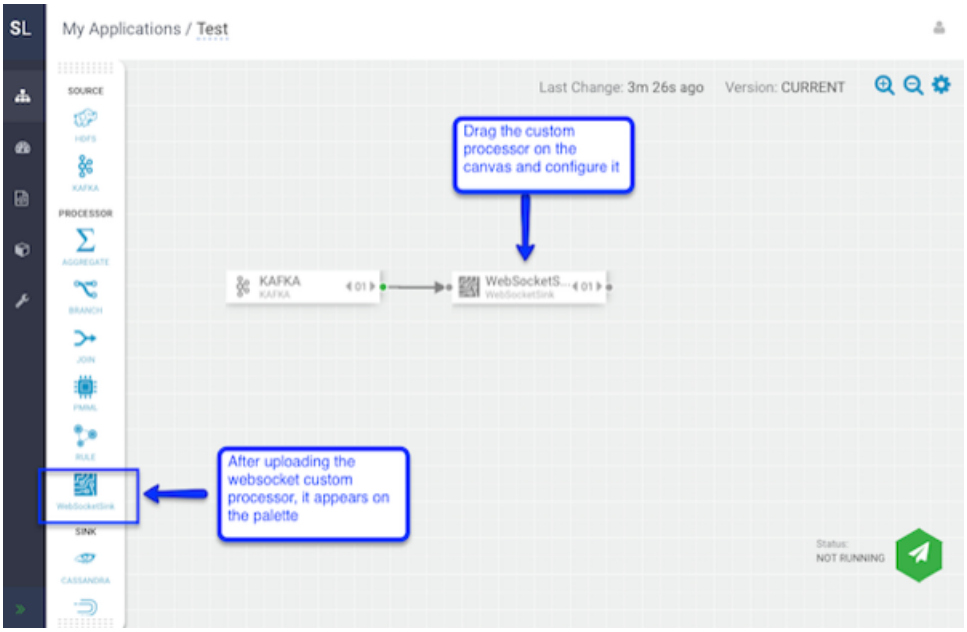
      For example, the `WebSocketSink` was configured as follows:



d. It might take a few minutes to upload the jar file to the server. Do not navigate away until you see a response. If you do not see a response, return to the Custom Processor page again; do not click Save again.

3. After the processor is registered, create a new stream app.

   On the canvas palette you should see the new processor. Drag the processor onto the canvas and configure it.



4. When you double-click on the WebSocketSink processor the following user fields are exposed. Notice that the configuration is based on the "Config Fields" settings specified during the registration process: