

Building a SAM Application

Date of Publish: 2020-12-15



Contents

Building an Application.....	3
Launch the Stream Builder UI.....	3
Add a New Stream Application.....	3
Add a Source.....	4
Connect Components.....	6
Join Multiple Streams.....	7
Filter Events in a Stream.....	7
Use Aggregate Functions over Windows.....	9
Deploying a Stream App.....	10
Configure Deployment Settings.....	10
Deploy the App.....	11

Building an Application

Prerequisites

- You have integrated SAM
- You have set up appropriate environments and service pools

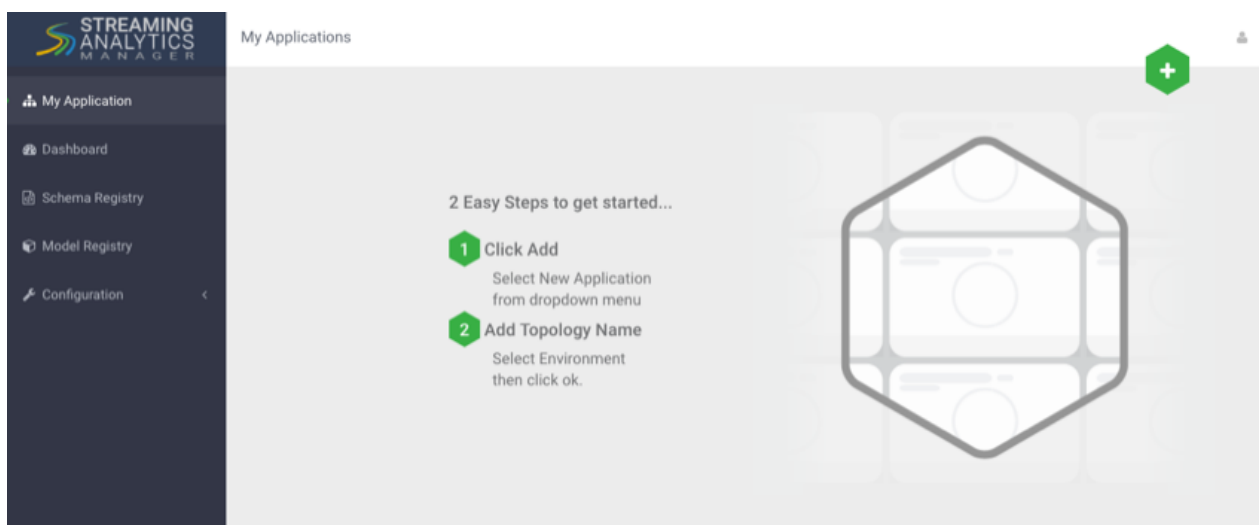
Launch the Stream Builder UI

Procedure

1. In Ambari, select **Streaming Analytics Manager** from the left-hand **Services** pane.
2. Under **Quick Links**, select **SAM UI**.

Results

The SAM Stream Builder UI displays. You can return at any time by clicking **My Applications** from the left-hand menu.



Add a New Stream Application

Procedure

1. Specify the name of the stream application and the environment you want to use.



Note:

The name of the stream app should not have any spaces.

2. SAM displays the Stream Builder canvas. Builder components on the canvas palette are the building blocks you use to build stream apps. Refer to the *HDF Overview* for information about each component building block.



Add a Source

As described in the *HDF Overview*, Stream Builder offers four types of builder components: sources, processors, sinks, and custom components. Start building your application by adding a source.

Before you begin

You have configured Schema Registry and integrated with SAM.

Procedure

1. Drag a source builder component, Kafka for example, onto the canvas. This creates a Kafka tile component:

The screenshot displays the SAM interface for an application named "IOT-Trucking-Ref-App". The left sidebar contains a navigation menu with categories: SOURCE, EVENT HUBS, HDFS, KAFKA, PROCESSOR, AGGREGATE, BRANCH, and JOIN. The main workspace shows a "Kafka source tile" with a label "KAFKA" and a value of "01". A gray dot on the right side of the tile indicates that the build component is not configured. A blue arrow points to the tile with the text "Click the arrows to increase or decrease the number of builder component instances for performance and scalability needs". Another blue arrow points to the gray dot with the text "Gray dot indicates that the build component is not configured". The top right of the workspace shows "Last Change: 2m 59s ago" and "Version: CURRENT".

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

Kafka connection settings are populated by SAM based on the Kafka service in Environment from the Service Pool

REQUIRED
OPTIONAL
NOTES

CLUSTER NAME *

streamanalytics

SECURITY PROTOCOL *

PLAINTEXT

BOOTSTRAP SERVERS *

secure-fenton-hdf5.field.hortonworks.com:6667,secure

KAFKA TOPIC *

truck_events_avro

CONSUMER GROUP ID *

truck_geo_event_1|

Output

eventSource
STRING

truckId*
INTEGER

driverId*
INTEGER

driverName*
STRING

routeId*
INTEGER

route*
STRING

eventType*
STRING

latitude*
DOUBLE

longitude*
DOUBLE

correlationId*
LONG

geoAddress
STRING

After you select a Kafka topic, SAM fetches the topic schema from Schema Registry

Cancel

Ok

3. Add the additional components you want to use to develop your stream app.

Results

When you have added and correctly configured your stream app components, the component tile displays a green dot on the left. You cannot connect a source to different processors or sinks until it is correctly configured.

Connect Components

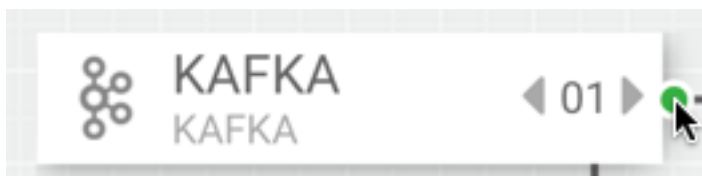
Once you have added and configured your source, add additional processors and sinks to the canvas. To pass a stream of events from one component to the next, create a connection between the two components. In addition to defining data flow, connections allow you to pass a schema from one component to another.

Before you begin

You have added and configured at least one source.

Procedure

1. Click the green dot to the left of your source component.



2. Drag your cursor to the component tile to which you want to connect.

Join Multiple Streams

Joining multiple streams is an important SAM capability. You accomplish this by adding the Join processor to your stream application.

Procedure

1. Drag a Join processor onto your canvas and connect it to a source.
2. Double click the Join tile to open the **Configuration** dialog.
3. Configure the Join processors according to your streaming application requirements.

Example

The screenshot shows the 'JOIN' configuration dialog with the following settings:

- Input:** kafka_stream_1
- JOIN TYPE:** INNER
- SELECT STREAM:** kafka_stream_2
- SELECT FIELD:** driverId
- WITH STREAM:** kafka_stream_1
- WINDOW INTERVAL TYPE:** Time
- WINDOW INTERVAL:** 05
- SLIDING INTERVAL:** 5
- OUTPUT FIELDS:** eventTime, eventSource, truckId, driverId, driverName, routeId, route, eventType, latitude, longitude, correlationId, geoAddress, speed

Annotations in the image include:

- 'Join stream_1 on field driverId' pointing to the input and select field dropdowns.
- 'Inner join with stream_2 on driverId' pointing to the JOIN TYPE and SELECT STREAM dropdowns.
- 'Wait 5 seconds for streams to catch up before the join occurs' pointing to the top right of the dialog.
- 'The output of the joins' pointing to the OUTPUT FIELDS list.

Filter Events in a Stream

You can use SAM to filter events in the stream. You accomplish this by using Rule processor, which translates rules into SQL queries that operate on the stream of data.

Procedure

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



2. Double click the Rule processor, click the + **Add New Rules** button, and create a new rule:

Add New Rule

RULE NAME*
Violation Event

DESCRIPTION*
Events that are infractions from drivers and trucks

CREATE QUERY*
eventType × NOT_EQUAL × 'Normal' × +

QUERY PREVIEW:
eventType <> 'Normal'

Cancel Ok

3. Click **Ok** to save the new rule.

Example

Event Type

CONFIGURATION NOTES

Input

- eventTime* (STRING)
- eventSource* (STRING)
- truckId* (INTEGER)
- driverId* (INTEGER)
- driverName* (STRING)
- routeId* (INTEGER)
- route* (STRING)
- eventType* (STRING)
- latitude* (DOUBLE)
- longitude* (DOUBLE)
- correlationId* (LONG)

Output

- eventTime* (STRING)
- eventSource* (STRING)
- truckId* (INTEGER)
- driverId* (INTEGER)
- driverName* (STRING)
- routeId* (INTEGER)
- route* (STRING)
- eventType* (STRING)
- latitude* (DOUBLE)
- longitude* (DOUBLE)
- correlationId* (LONG)

Rules Table:

Name	Condition	Actions
Violation Event	eventType <> 'Normal'	

Annotations:

- Click to add rules which get translated into SQL on the stream and allows filtering of stream events (points to '+Add New Rules' button)
- A rule that is translated into SQL that looks for any event in the stream with an event type not equal to Normal, which represents a Violation Event (points to the rule in the table)

Buttons: Cancel, Ok

Use 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. In SAM, you accomplish this using the Aggregate processor. The Aggregate processor supports two window types, tumbling and sliding windows. You can create a window based on time or count.

Procedure

1. Drag the Aggregate processor to the canvas and connect it to the stream application you are building.
2. Double click the Aggregate tile to configure it according to your stream application requirements.

Example

The screenshot shows the configuration window for a stream application named "DriverAvgSpeed". The window is divided into several sections:

- Input:** A list of input fields including truckId, driverId, driverName, routeId, route, eventType, latitude, longitude, correlationId, geoAddress, and speed.
- SELECT KEYS:** A field where the keys "driverId", "driverName", and "route" are selected. An annotation points to this field with the text "The fields to group by".
- WINDOW INTERVAL TYPE:** Set to "Time".
- WINDOW INTERVAL:** Set to "3" Minutes.
- SLIDING INTERVAL:** Set to "3" Minutes.
- TIMESTAMP FIELD:** Set to "processingTime".
- Output:** A list of output fields including driverId, driverName, route, and speed_AVG. An annotation points to this section with the text "At the end of the window, this is the new schema that will be output to the stream: the average speed of every driver".

Buttons for "Cancel" and "Ok" are visible at the bottom right of the window.

Deploying a Stream App

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).

Topology Configuration ✕

NUMBER OF WORKERS
5

NUMBER OF ACKERS
1

TOPOLOGY MESSAGE TIMEOUT (SECONDS)
40

WORKER JVM OPTIONS
-Xmx3072m

HBase config

HBASE ROOT DIRECTORY *
hdfs://localhost:9000/tmp/hbase

Cancel Ok

Deploy the App

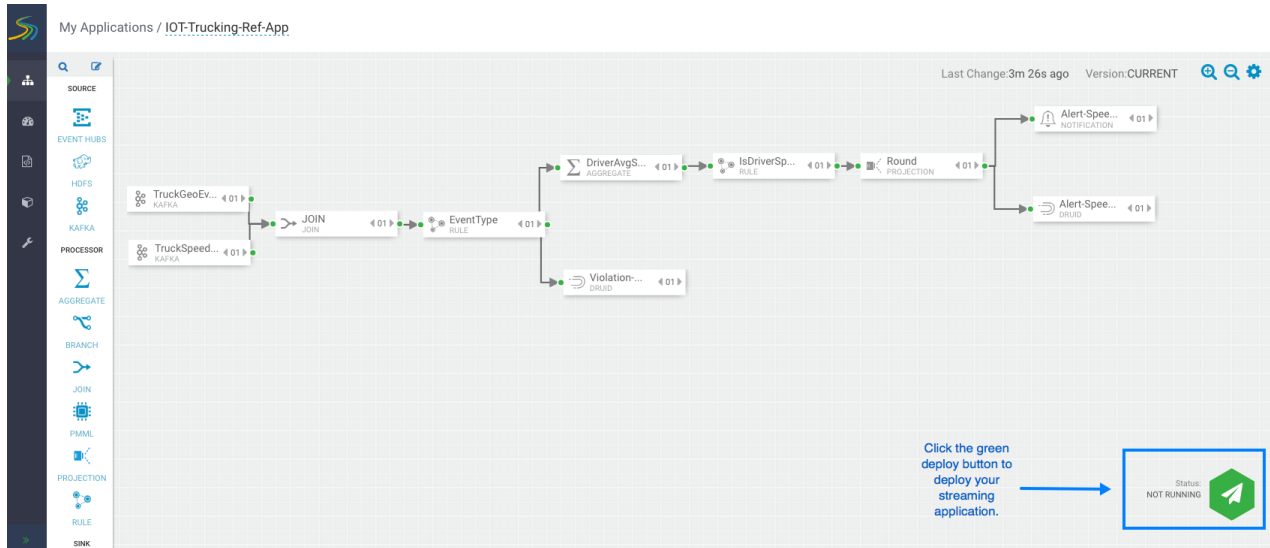
After the app's deployment settings has been configured, click the Deploy button on the lower right of the canvas. During the deployment process, Streaming Analytics Manager completes the following tasks:

Procedure

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.

Results

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.

