

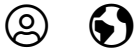
Cloudera uses cookies to provide and improve our site services.

By using this site, you consent to use of cookies as outlined in

Accept cookies

Cloudera's Privacy and Data Policies

(<https://www.cloudera.com/legal/policies.html>).



(/)

Tutorials (/tutorials.html)

Getting Started with CDF Sandbox

Ready to Get Started?

Download Sandbox (</downloads/hortonworks-sandbox.html>)

Introduction

In this tutorial, you will learn how to deploy a modern real-time streaming application. This application serves as a reference framework for developing a big data pipeline, complete with a broad range of use cases and powerful reusable core components. You will explore the NiFi Dataflow application, Kafka topics, Schemas and SAM topology.

Prerequisites

Downloaded and deployed the **Cloudera DataFlow (CDF)**

([https://www.cloudera.com/downloads/hortonworks-sandbox/hdf.html?](https://www.cloudera.com/downloads/hortonworks-sandbox/hdf.html?utm_source=mktg-tutorial)

[utm_source=mktg-tutorial](https://www.cloudera.com/downloads/hortonworks-sandbox/hdf.html?utm_source=mktg-tutorial)) Sandbox

Outline

Concepts

Overview of Trucking IoT Ref App

Step 1: Explore Dataflow Application

Step 2: View Schema Registry

Step 3: Analyze Stream Analytics Application

Step 4: View the Storm Engine that Powers SAM

Summary

Further Reading

Appendix A: Trucking IoT GitHub Repo (<https://github.com/orendain/trucking-iot/tree/master>)

Concepts

Stream Analytics Manager (SAM)

Stream Analytics Manager is a drag and drop program that enables stream processing developers to build data topologies within minutes compared to traditional practice of writing several lines of code. A topology is a **directed acyclic graph (DAG)** of processors. Now users can configure and optimize how they want each component or processor to perform computations on the data. They can perform windowing, joining multiple streams together and other data manipulation. SAM currently supports the stream processing engine known as Apache Storm, but it will later support other engines such as Spark and Flink. At that time, it will be the users choice on which stream processing engine they want to choose.

Apache Storm

Apache Storm is the current backend computational processing engine for Stream Analytics Manager. After the user builds their SAM topology, all the actual processing of data happens in a Storm topology, which is also a **DAG**, but is comprised of spouts and bolts with streams of tuples representing the edges.

A **spout** ingests the data usually from a Kafka Topic into the topology while **bolts** do all the processing. Thus, all the same components from the SAM topology are represented in the Storm topology, but as appropriate spouts and bolts.

Storm is the Open Source distributed, reliable, fault-tolerant system that handles real time analytics, scoring machine learning models, continuous static computations and enforcing Extract, Transform and Load (ETL) paradigms.

Schema Registry

Schema Registry (SR) stores and retrieves Avro Schemas via RESTful interface. SR stores a version history containing all schemas. Serializers are provided to plug into Kafka clients that are responsible for schema storage and retrieve Kafka messages sent in Avro format.

Kafka

Apache Kafka is an open source publish-subscribe based messaging system responsible for transferring data from one application to another.

Overview of Trucking IoT Ref App

The Trucking IoT Reference Application is built using Cloudera DataFlow Platform.

The Trucking IoT data comes from a truck events simulator that is ingested by Apache NiFi, NiFi sends the data to Kafka topics which are then ingested by Stream Analytics Manager (SAM). A more in depth explanation of the pipeline will be explained as you explore the NiFi Dataflow application, Schema Registry and SAM.

Step 1: Explore Dataflow Application

1. Open the NiFi UI <http://sandbox-hdf.hortonworks.com:9090/nifi/> (<http://sandbox-hdf.hortonworks.com:9090/nifi/>)
2. The NiFi Dataflow application template: `Trucking IoT Demo` will appear on the canvas.
4. Select NiFi configuration icon . Click on the Controller Services tab.
5. The **HortonworksSchemaRegistry** should be enabled. If it's not enabled then select the lightning bolt symbol next to **HortonworksSchemaRegistry**.

NiFi Flow Configuration

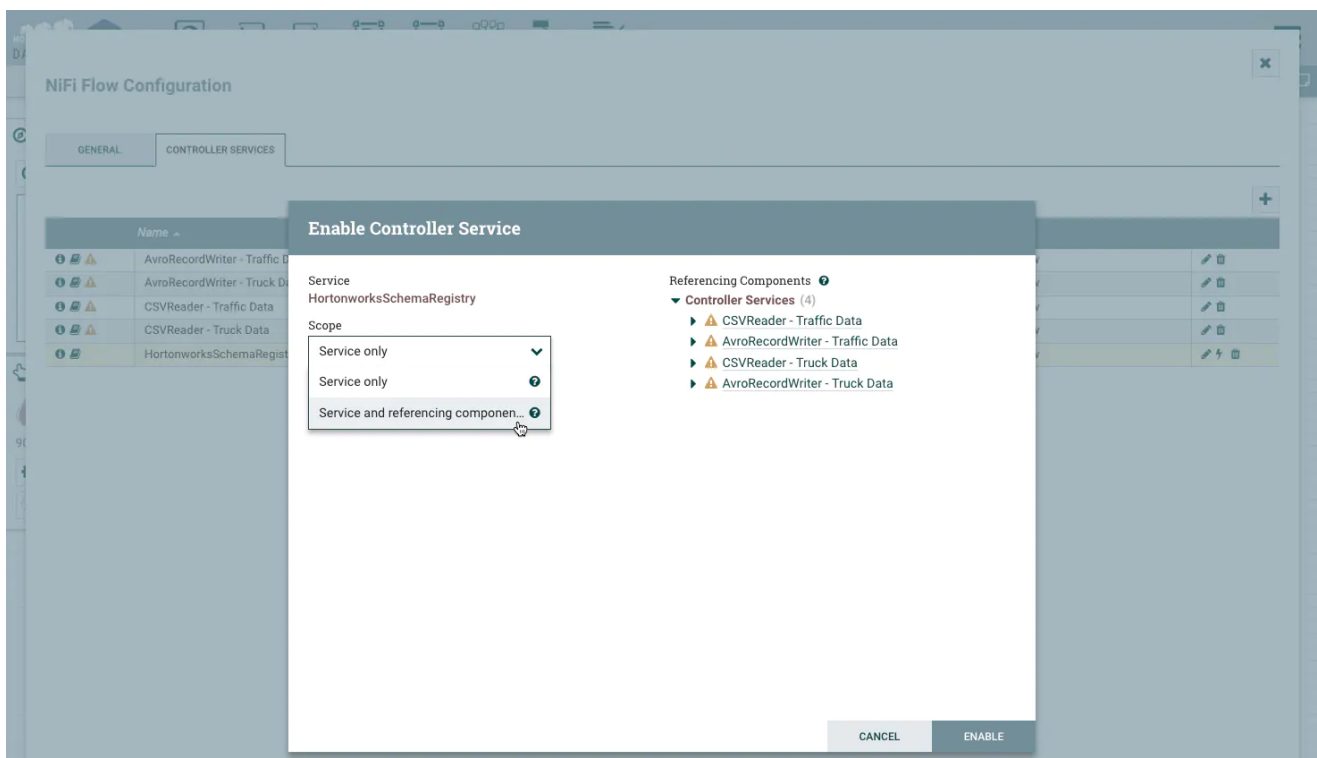
GENERAL

CONTROLLER SERVICES

+

	Name	Type	Bundle	State	Scope	
	AvroRecordWriter - Traffic...	AvroRecordSetWriter 1.2...	org.apache.nifi - nifi-recor...	 Enabled	NiFi Flow	
	AvroRecordWriter - Truck ...	AvroRecordSetWriter 1.2...	org.apache.nifi - nifi-recor...	 Enabled	NiFi Flow	
	CSVReader - Traffic Data	CSVReader 1.2.0.3.0.2.0-76	org.apache.nifi - nifi-recor...	 Enabled	NiFi Flow	
	CSVReader - Truck Data	CSVReader 1.2.0.3.0.2.0-76	org.apache.nifi - nifi-recor...	 Enabled	NiFi Flow	
	HortonworksSchemaRegi...	HortonworksSchemaRegi...	org.apache.nifi - nifi-hwx-s...	 Enabled	NiFi Flow	

6. In the "Enable Controller Service" window, under "Scope", select "Service and referencing components". Then click ENABLE.



All controller services referencing **HortonworksSchemaRegistry** will also be enabled. Head back to the NiFi Dataflow.

Overview of the **7 processors** in the NiFi Flow:

GetTruckingData - Simulator generates TruckData and TrafficData in bar-delimited CSV

RouteOnAttribute - filters the *TrafficData* and *TruckData* into separate data feeds

Data Name	Data Fields
TruckData	eventTime, truckId, driverId, driverName, routeId, routeName, latitude, longitude, speed, eventType
TrafficData	eventTime, routeId, congestionLevel

TruckData side of Flow

EnrichTruckData - tags on three fields to the end of *TruckData*: "foggy", "rainy", "windy"

ConvertRecord - reads incoming data with "CSVReader" and writes out Avro data with "AvroRecordSetWriter" embedding a "trucking_data_truck_enriched" schema onto each flowfile.

PublishKafka_1_0 - stores Avro data into Kafka Topic "trucking_data_truck_enriched"

TrafficData side of Flow

ConvertRecord - converts CSV data into Avro data embedding a "trucking_data_traffic" schema onto each flowfile

PublishKafka_1_0 - stores Avro data into Kafka Topic "trucking_data_traffic"

Overview of **5 controller services** used in the NiFi Flow:


AvroRecordSetWriter - Enriched Truck Data - writes contents of RecordSet in Binary Avro Format (trucking_data_truck_enriched schema)

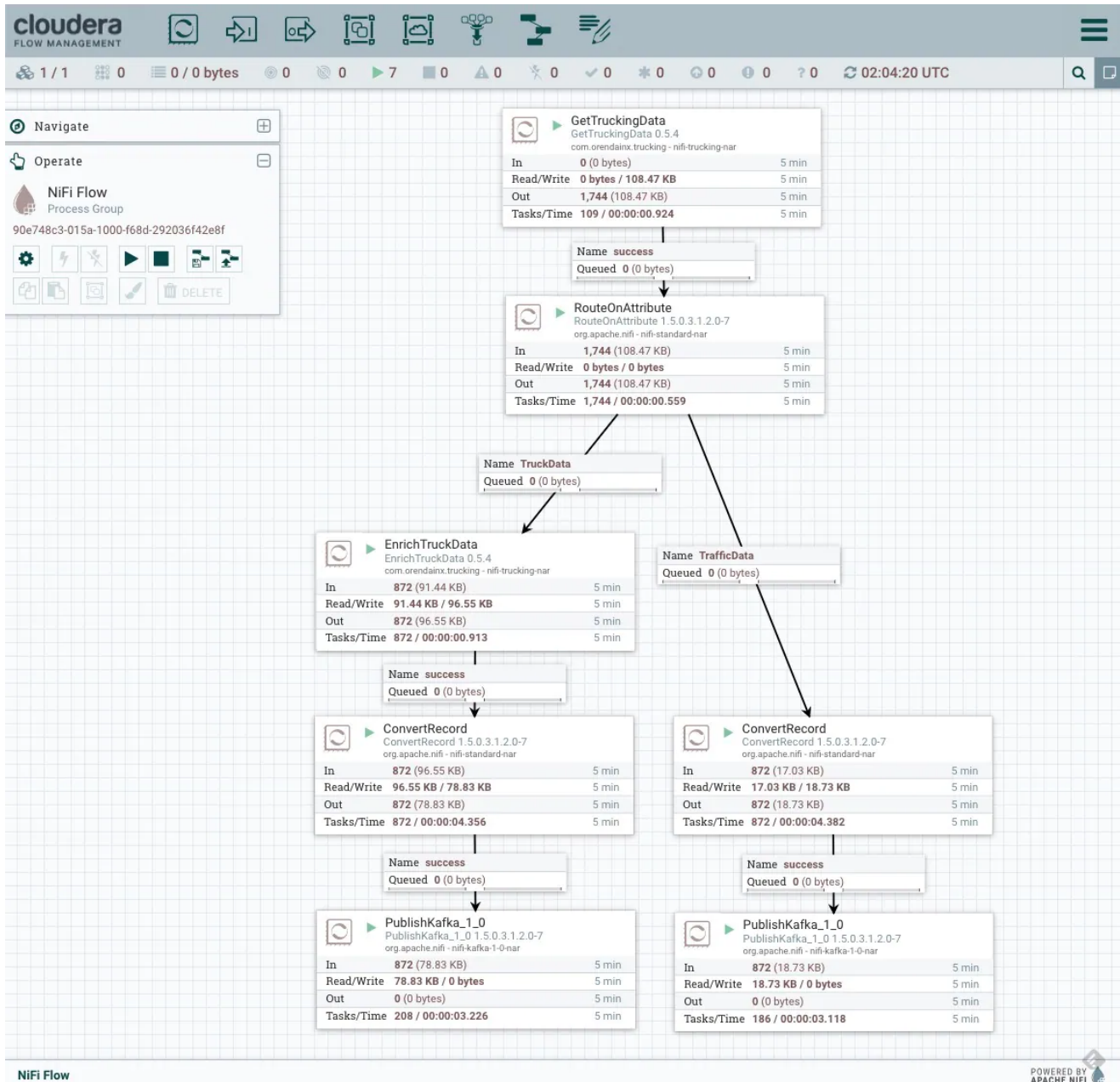
AvroRecordSetWriter - Traffic Data - writes contents of RecordSet in Binary Avro Format (trucking_data_traffic schema)


CSVReader - Enriched Truck Data - returns each row in CSV file as a separate record (trucking_data_truck_enriched schema)

CSVReader - Traffic Data - returns each row in CSV file as a separate record (trucking_data_traffic schema)

HortonworksSchemaRegistry - provides schema registry service for interaction with Cloudera Schema Registry

7. Press **command+A** or **control+A** to select all the processors in the NiFi Dataflow and click on the start button .









8. To reduce resource consumption and footprint, when the **PublishKafka_1_0** processors reach about 500 input records, click on the stop button . This will take approximately 1 - 2 minutes.

9. Stop NiFi service: **Ambari -> NiFi -> Service Actions -> Stop**

Step 2: View Schema Registry

1. Open the Schema Registry UI at <http://sandbox-hdf.hortonworks.com:7788/> (<http://sandbox-hdf.hortonworks.com:7788/>)

SCHEMA REGISTRY		All Schemas					
		Search by name		Sort: Last		Updated ▾	
	trucking_data_driverstats	TYPE	GROUP	BRANCH	SERIALIZER & DESERIALIZER		
	BACKWARD	avro	truck...	1 	0	▾	
	trucking_data_joined	TYPE	GROUP	BRANCH	SERIALIZER & DESERIALIZER		
	BACKWARD	avro	truck...	1 	0	▾	
	trucking_data_truck_enriched	TYPE	GROUP	BRANCH	SERIALIZER & DESERIALIZER		
	BACKWARD	avro	truck...	1 	0	▾	
	trucking_data_traffic	TYPE	GROUP	BRANCH	SERIALIZER & DESERIALIZER		
	BACKWARD	avro	truck...	1 	0	▾	
	trucking_data_truck	TYPE	GROUP	BRANCH	SERIALIZER & DESERIALIZER		
	BACKWARD	avro	truck...	1 	0	▾	

Overview of the essential **schemas** in the Schema Registry:

trucking_data_joined - model for truck event originating from a truck's onboard computer (EnrichedTruckAndTrafficData)

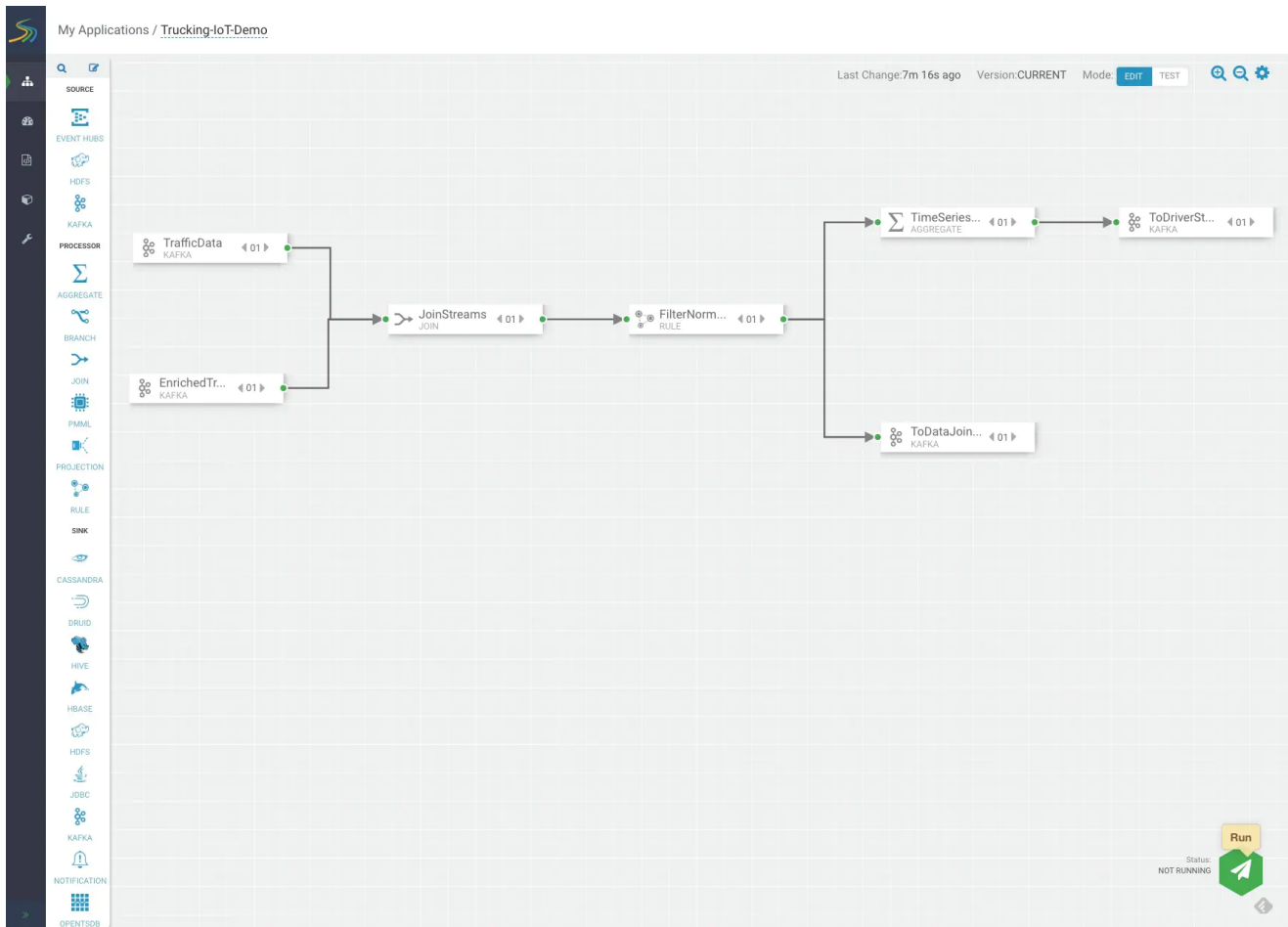
trucking_data_truck_enriched - model for truck event originating from a truck's onboard computer (EnrichedTruckData)

trucking_data_traffic model for eventTime, routeld, congestionLevel (TrafficData)

Step 3: Analyze Stream Analytics Application

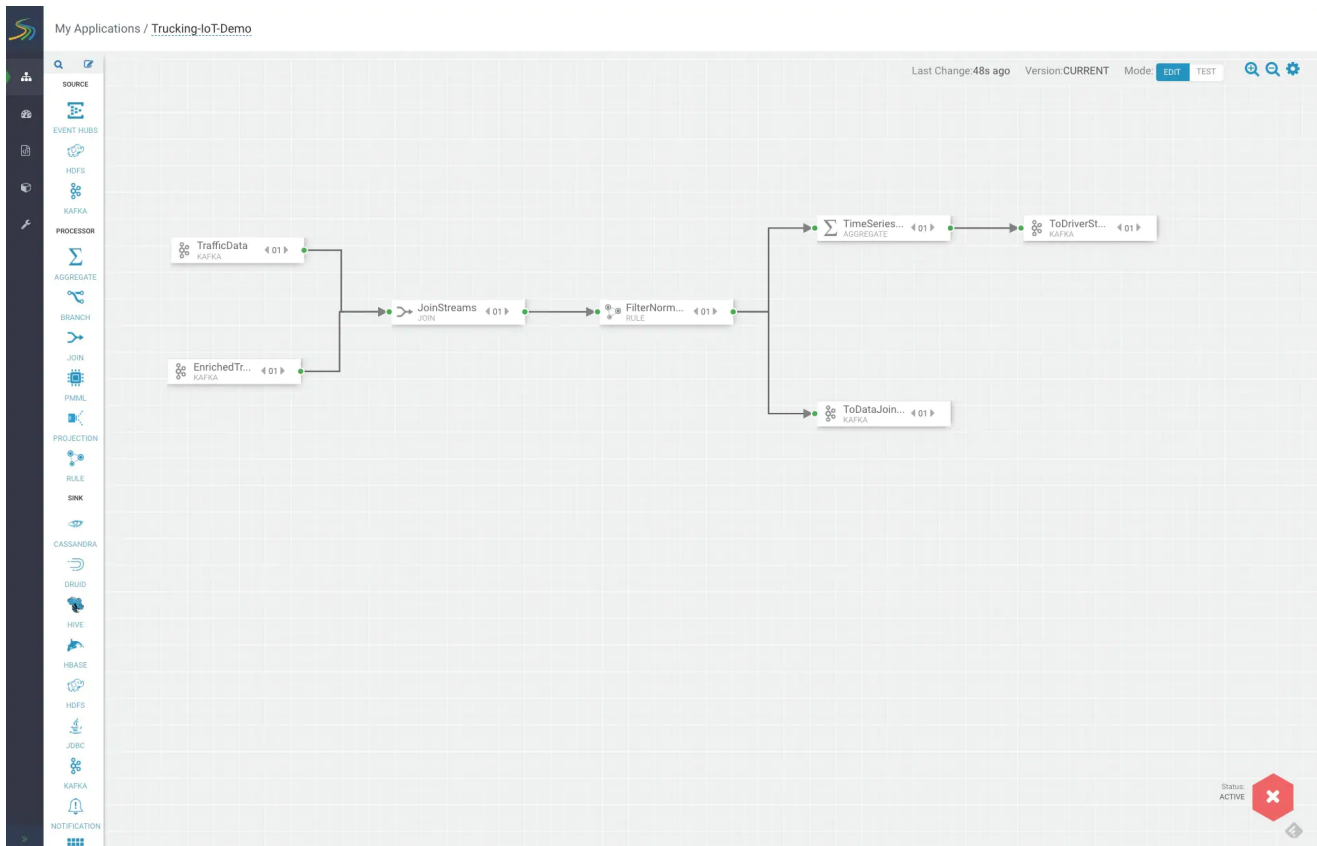
1. Open Stream Analytics Manager (SAM) at <http://sandbox-hdf.hortonworks.com:7777/> (<http://sandbox-hdf.hortonworks.com:7777/>)

2. Click on the Trucking-IoT_Demo, then the green pencil on the right top corner. This should show an image similar to the one below, click on the **Run** button to deploy the topology:



A window will appear asking if you want to continue deployment, click **Ok**.

3. You will receive a notification that the SAM topology application deployed successfully and your topology will show Active Status in the bottom right corner.



Overview of the SAM Canvas:

My Applications: Different Topology Projects

1st Left Sidebar: My Applications, Dashboard, Schema Registry, Model Registry, Configuration

2nd Left Sidebar: Different stream components (source, processor, sink)

Gear Icon: Configure topology settings

Status Icon: Start or Stop Topology

Overview of SAM topology:

TrafficData is the source component name, which pulls in data from the Kafka topic "trucking_data_traffic".

EnrichedTruckData is the source component name, which pull is data from the Kafka topic "trucking_data_truck_enriched"

JoinStreams joins streams "TrafficData" and "EnrichedTruckData" by "routeId".

FilterNormalEvents checks if non "Normal" eventType's occur, then it will emit them.

TimeSeriesAnalysis computes the average of 10 samples of speed across a 10 second window period, calculates the sum across the same window period as before for foggy, rainy, windy and eventTime individually.

ToDriverStats stores the input from "TimeSeriesAnalysis": driverId, routeId, averageSpeed, totalFog, totalRain, totalWind, and totalViolations into Kafka topic "trucking_data_driverstats".

ToDataJoined stores the input from "FilterNormalEvents": eventTime, congestionLevel, truckId, driverId, driverName, routeId, routeName, latitude, longitude, speed, eventType, foggy, rainy, and windy into Kafka topic "trucking_data_joined".

Step 4: View the Storm Engine that Powers SAM

1. From Ambari, click on **Storm > Storm UI**
2. Click on Topology Name: **streamline-1-Trucking-IoT-Demo** under **Topology Summary**

Storm UI

Cluster Summary

Version	Supervisors	Used slots	Free slots	Total slots	Executors	Tasks
1.2.1.3.2.0.0-520	1	1	1	2	9	9

Nimbus Summary

Search:

Host	Port	Status	Version	UpTime
sandbox-hdf.hortonworks.com	6627	Leader	1.2.1.3.2.0.0-520	20m 51s

Showing 1 to 1 of 1 entries

Topology Summary

Search:

Name	Owner	Status	Uptime	Num workers	Num executors	Num tasks	Replication count	Assigned Mem (MB)	Scheduler Info
streamline-1-Trucking-IoT-Demo	storm	ACTIVE	13m 17s	1	9	9	1	832	

Showing 1 to 1 of 1 entries

Supervisor Summary

Search:

Host	Id	Uptime	Slots	Used slots	Avail slots	Used Mem (MB)	Version
sandbox-hdf.hortonworks.com (log)	6a52fdb1-13b1-4195-89a3-02bafab9f12a	2h 12m 48s	2	1	1	832	1.2.1.3.2.0.0-520

Showing 1 to 1 of 1 entries

Nimbus Configuration

Show 20 entries

Search:

Key	Value
backpressure.disruptor.high.watermark	0.9
backpressure.disruptor.low.watermark	0.4
backpressure.znode.timeout.secs	30
backpressure.znode.update.freq.secs	15

3. Overview of the Storm Topology

Storm UI

Search streamline-1-Trucking-IoT-Demo-1-1536269100: Search Search Archived Logs: ☐

Topology summary

Name	Id	Owner	Status	Uptime	Num workers	Num executors	Num tasks	Replication count	Assigned Mem (MB)	Scheduler Info
streamline-1-Trucking-IoT-Demo	streamline-1-Trucking-IoT-Demo-1-1536269100	storm	ACTIVE	14m 4s	1	9	9	1	832	

Topology actions

[Activate](#) [Deactivate](#) [Rebalance](#) [Kill](#) [Debug](#) [Stop Debug](#) [Change Log Level](#)

Topology stats

Window	Emitted	Transferred	Complete latency (ms)	Acked	Failed
10m 0s	1360	1440	1572.167	120	186
3h 0m 0s	119120	127520	4001.569	1160	740
1d 0h 0m 0s	119120	127520	4001.569	1160	740
All time	119120	127520	4001.569	1160	740

Kafka Spouts Lag

Id	Topic	Partition	Latest Offset	Spout Committed Offset	Lag
1-TrafficData	trucking_data_traffic	0	568	568	0
2-EnrichedTruckData	trucking_data_truck_enriched	0	568	568	0

Spouts (All time)

Search:

Id	Executors	Tasks	Emitted	Transferred	Complete latency (ms)	Acked	Failed	Error Host	Error Port	Last error	Error Time
1-TrafficData	1	1	1640	1640	3628.100	600	220				
2-EnrichedTruckData	1	1	2160	2160	4401.714	560	520				

Showing 1 to 2 of 2 entries

Bolts (All time)

Search:

Id	Executors	Tasks	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)	Acked	Failed	Error Host	Error Port	Last error	Error Time
3-JoinStreams	1	1	27880	27860	0.000	0.011	1900	2893.067	1880	0				
4-FilterNormalEvents	1	1	60180	68600	0.000	0.048	26000	0.024	25980	0				
5-TimeSeriesAnalysis	1	1	9680	9680	0.000	0.142	8180	44.660	4820	0				
6-ToDataJoined	1	1	16380	16380	0.000	0.308	8180	53.873	8200	0				
7-ToDriverStats	1	1	80	80	0.000	0.000	0	0.000	20	0				
_acker	1	1	1120	1120	0.000	0.002	81920	0.008	81860	0				
_eventlogger	1	1	0	0	0.000	0.000	0	0.000	0	0				

You can see the total number of **Emitted** (1360) and **Transferred** (1440) tuples after 10m 0s under **TOPOLOGY STATS** for the entire topology. You can also see individual emitted and transferred tuples for each individual Spout and Bolt in the topology increase. If we hover over one of the spouts or bolts on the graph, we can see how much data they process and their latency.

Topology Summary

Topology Stats

Topology Static Visualization

Spout

Bolts

Topology Configuration

Summary

Congratulations! You deployed the Trucking IoT demo that processes truck event data by using the NiFi data flow application to separate the data into two flows: *TruckData* and *TrafficData*. These two flows are then transmitted into two Kafka robust queues tagged with Schema Registry schemas: *trucking_data_traffic* and *_trucking_data_truck_enriched*. Stream Analytics Manager's (SAM) topology pulls in this data to join the two streams (or flows) by *routeId*, and filter's non-normal events which then get split into two streams. One stream is sent to a Kafka sink directly the other stream is then further filtered with an aggregate processor then sent to a different Kafka sink.

Further Reading


Apache NiFi User Guide (<https://nifi.apache.org/docs.html>)

Kafka Documentation (<https://kafka.apache.org/documentation/>)

Schema Registry (<https://docs.confluent.io/current/schema-registry/docs/index.html>)

Stream Analytics Manager User Guide

(https://docs.hortonworks.com/HDPDocuments/HDF3/HDF-3.0.2/bk_streaming-analytics-manager-user-guide/content/ch_sam-manage.html)

f (<https://www.facebook.com/cloudera/>) 
(<https://twitter.com/cloudera>**)** **in**
(<https://www.linkedin.com/company/cloudera>**)**

Partners (</partners.html>)

Resources (</resources.html>)

Community (<http://community.cloudera.com>)

Documentation (<http://docs.cloudera.com>)

Careers (</about/careers.html>)

Contact Us (</contact-sales.html>)

US: +1 888 789 1488 (tel:18887891488)

Outside the US: +1 650 362 0488 (tel:16503620488)

 English

© 2021 Cloudera, Inc. All rights reserved. **Terms & Conditions** (</legal/terms-and-conditions.html>) | **Privacy Policy and Data Policy** (</legal/policies.html>) | **Unsubscribe / Do Not Sell My Personal Information** (</unsubscribe.html>)
Apache Hadoop (<http://hadoop.apache.org/>) and associated open source project names are trademarks of the **Apache Software Foundation** (<http://apache.org/>). For a complete list of trademarks, **click here** (</legal/terms-and-conditions.html#trademarks>).