Streaming Analytics Manager 3

Managing Stream Analytics Manager

Date of Publish: 2020-12-15

https://docs.cloudera.com/

Contents

Stream Operations.....3

My Applications View..... 3
Application Performance Monitoring.....3
Exporting and Importing Stream Applications..... 4
Troubleshooting and Debugging a Stream Application.....5
Monitoring SAM Apps and Identifying Performance Issues.....5
Identifying Processor Performance Bottlenecks..... 10
Debugging an Application through Distributed Log Search..... 14
Debugging an Application through Sampling.....16

Stream Operations

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

- Application life cycle management: start, stop, edit, delete
- Application performance metrics
- Troubleshooting, debugging
- Exporting and importing applications

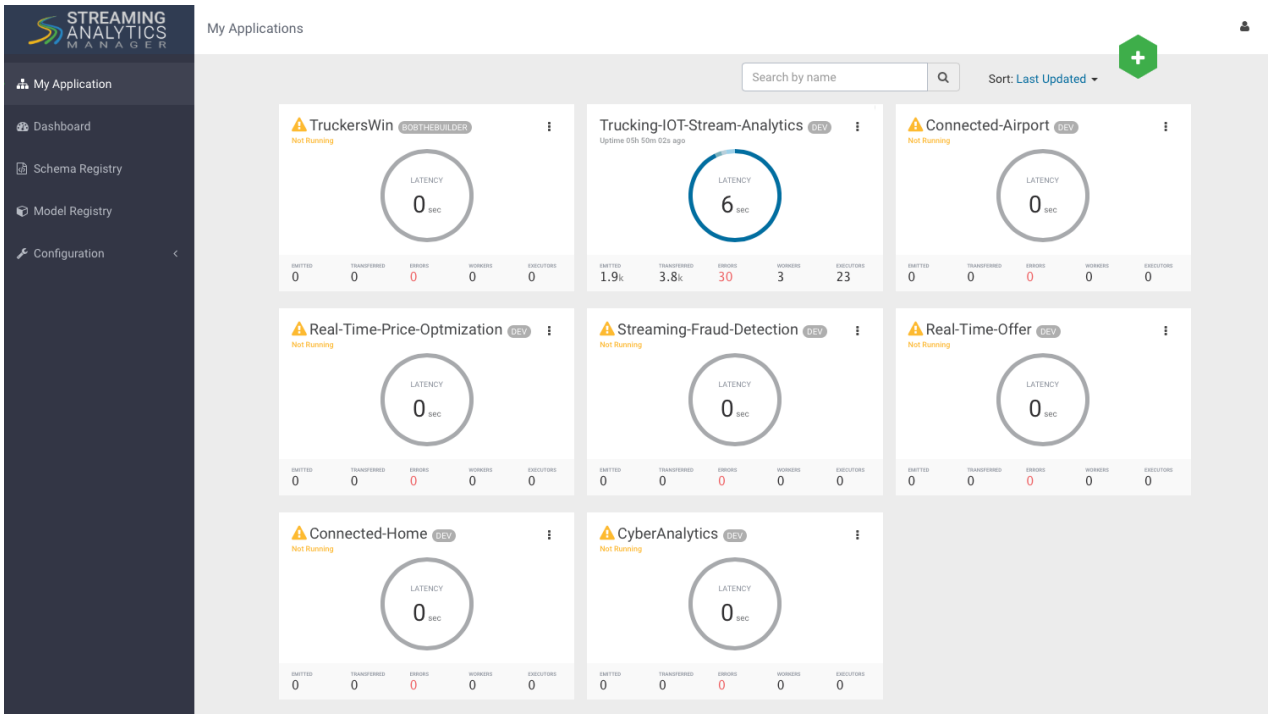
My Applications View

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

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

To access the application dashboard in SAM, click **My Application** tab (the hierarchy icon). The dashboard displays all applications built using Streaming Analytics Manager.

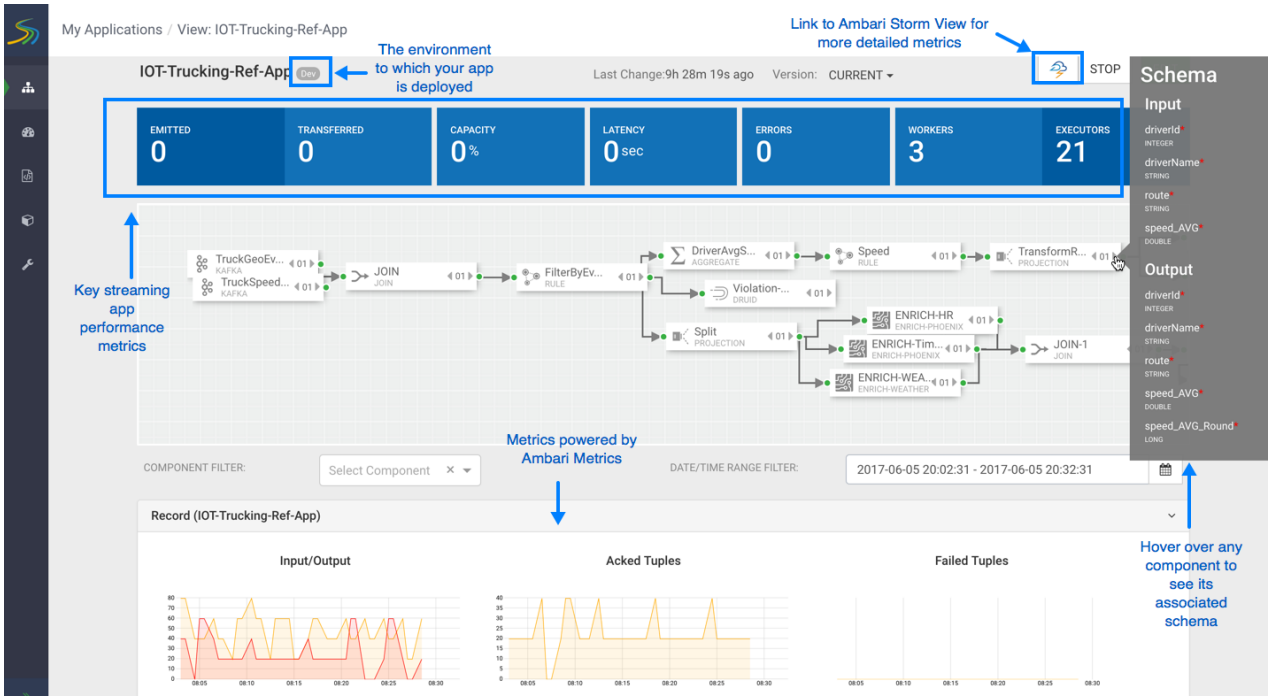
Each stream application is represented by an application tile. Hovering over the application tile displays status, metrics, and actions you can perform on the stream application.



Application Performance Monitoring

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

The following diagram describes elements of the APM view.

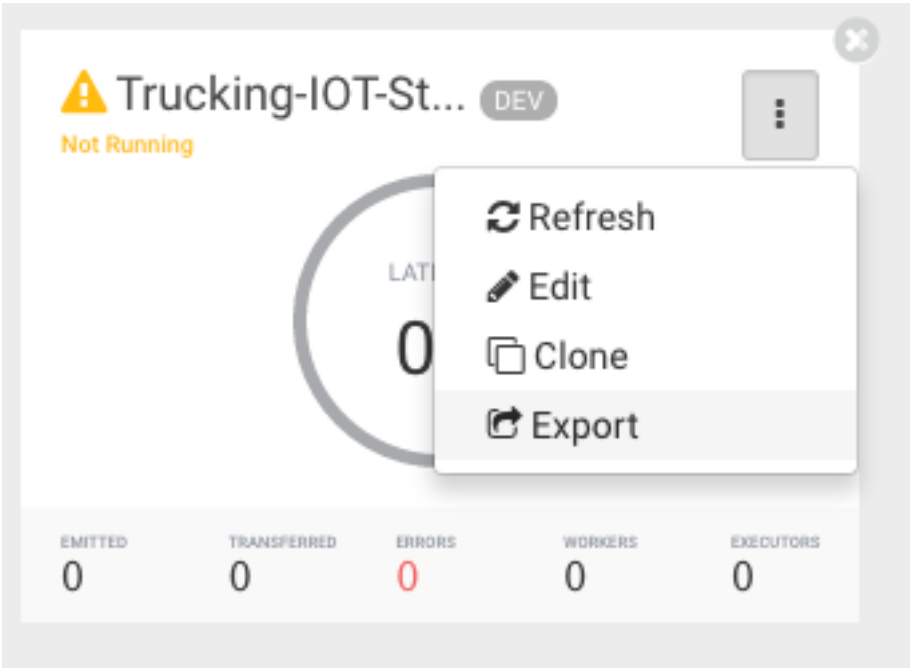


Exporting and Importing Stream Applications

Service pool and environment abstractions combined with import and export capabilities allow you to move a stream application from one environment to another. This task provides instructions about importing a stream application that was exported in JSON format.

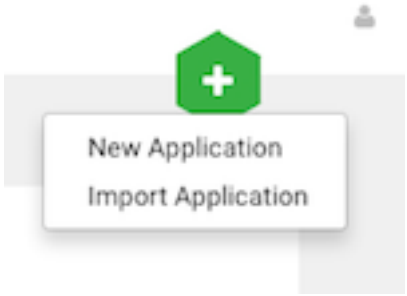
About this task

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

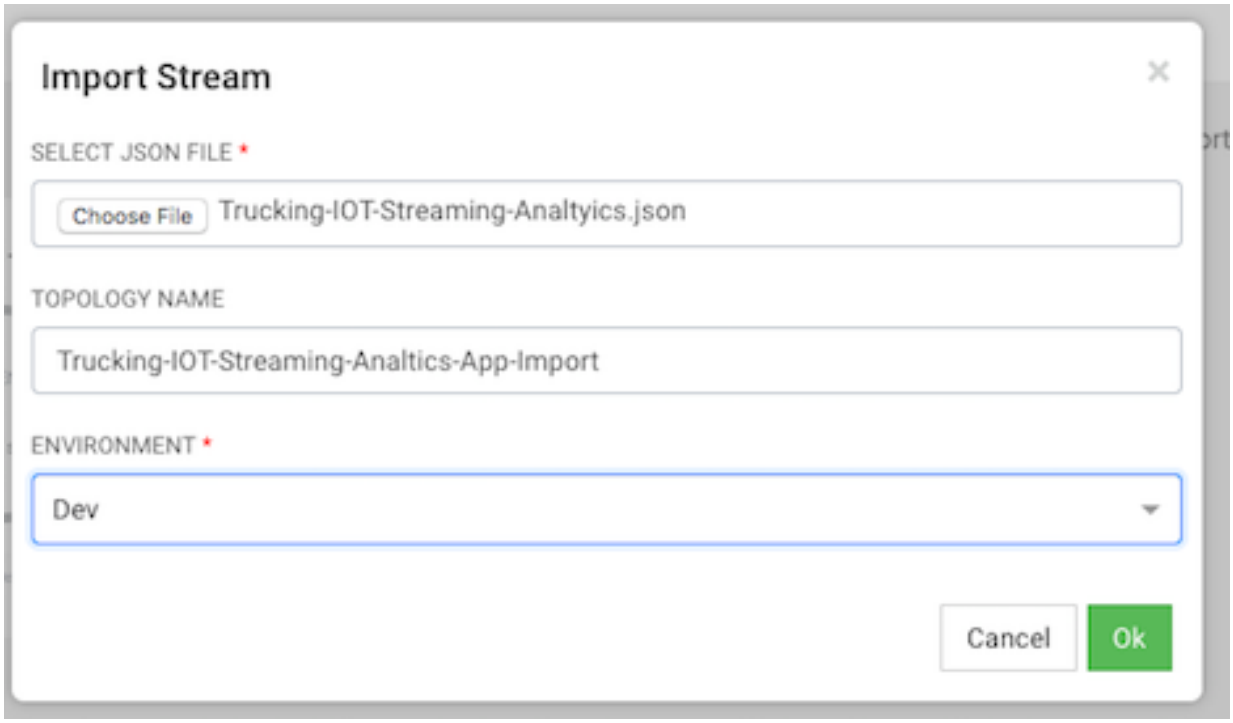


Procedure

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 application, and specify which environment to use.



Troubleshooting and Debugging a Stream Application

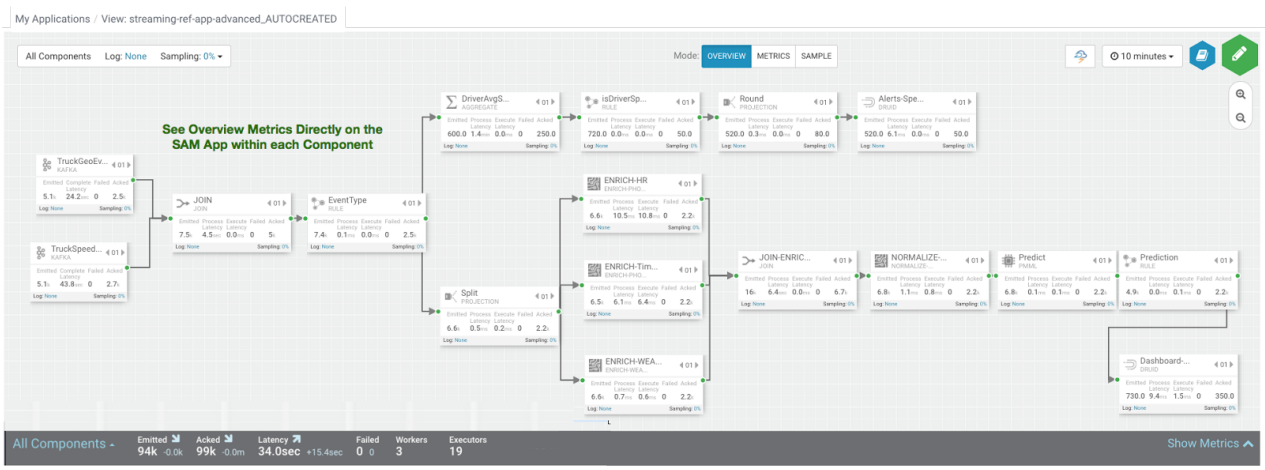
Once we have deployed the streaming app, common actions performed by users such as DevOps, Developers, and Operations teams are the following:

- Monitoring the Application and troubleshooting and identifying performance issues
- Troubleshooting an application through Log Search
- Troubleshooting an application through Sampling

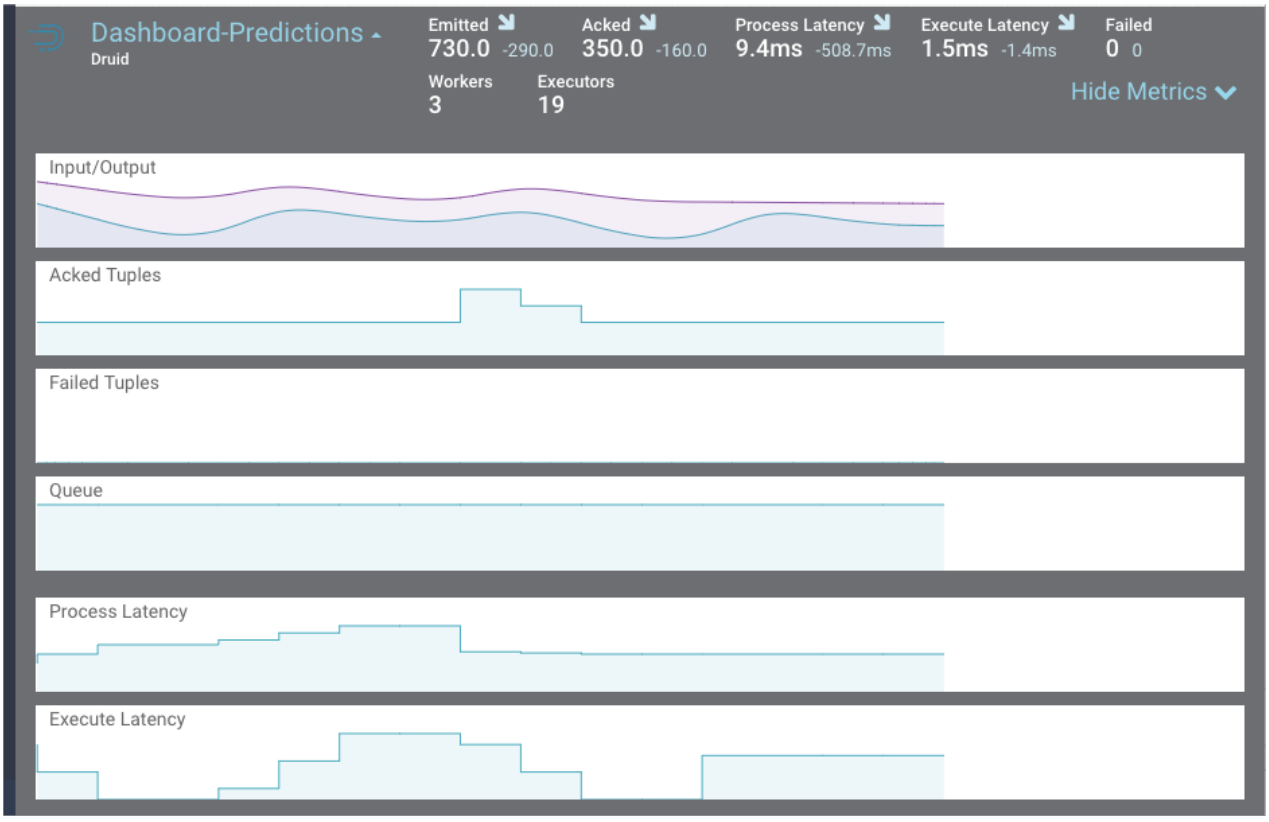
SAM makes performing these tasks easier by using the same visual approach as users have when developing the application. We will walk through these common use cases in the below sections.

Monitoring SAM Apps and Identifying Performance Issues

After deploying SAM and running the test generator for about 30 mins, your Storm Operation Mode of the app renders important metrics within each component on the canvas like below.



You can click on **Show Metrics** to get more details on the metrics and drill down on individual metrics. Note the detailed level metrics for **All Components**, **TruckGeoEvent Kafka** source, and **Dashboard-Predictions** Druid Sink.



Key metrics include the following:

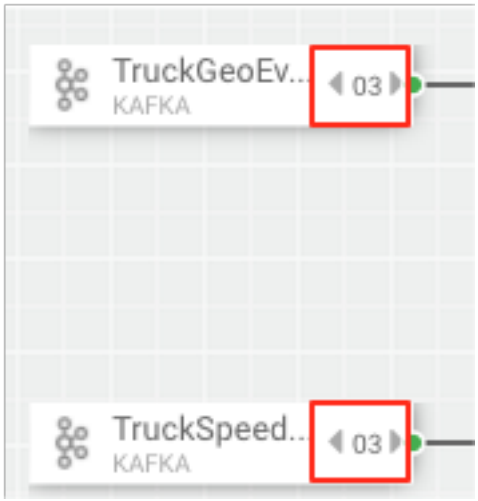
Metric Name	Description
Execute Latency	The average time it takes an event to be processed by a given component

Metric Name	Description
Process Latency	The average time it takes an event to be acked. Bolts that join, aggregate or batch may not Ack a tuple until a number of other Tuples have been received
Complete Latency	How much time an event from source takes to be fully processed and acked by the topology. This metrics is only available for sources (e.g.: Kafka Source)
Emitted	The number of events emitted for the given time period. For example, for a Kafka Source, it is the number of events consumed for the given time period
Acked	The number of events acked for the given time period. For example, for a Kafka Source, it is the number of events consumed and then acked.

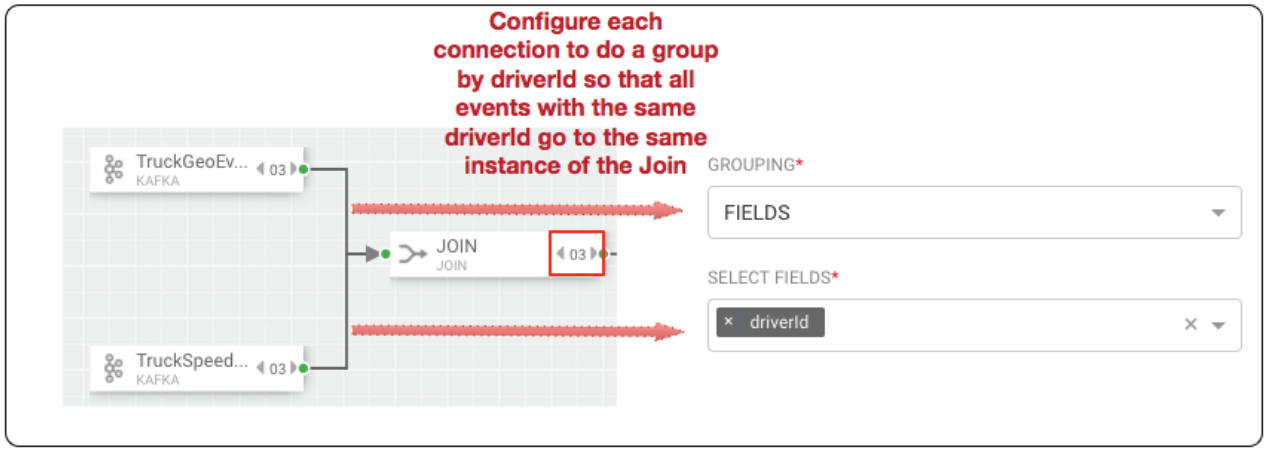
Identifying Throughput Bottlenecks

Looking through the metrics the Source and Sink metrics, we want to increase the throughput such that we emit/ consume more events from the Kafka Topic and send more events to Druid sink over time. We make some changes to the app to increase throughput.

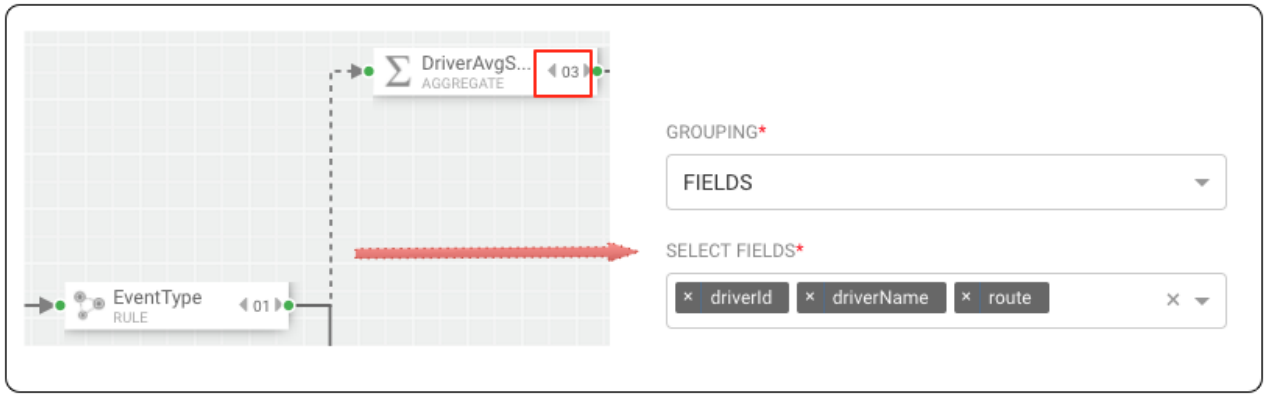
Increase the parallelism of TruckGeoEvent (kafka topic: truck_events_avro) and TruckSpeedEvent (kafka topic: truck_speed_events_avro) from 1 to 3. Note that each of these kafka topics have three partitions.



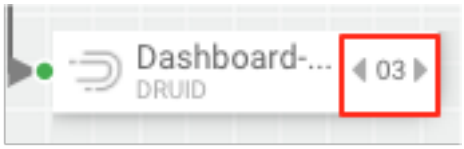
Increase the parallelism of the Join from 1 to 3. Since the join is grouped by driverId, we can configure the connection to use fields grouping to send all events with driverId to the same instance of the Join.



Increase the parallelism of the DriverAvgSpeed aggregate window from 1 to 3. Since the window groups by driverId,driverName and route, we can configure the connection to use fields grouping to send all events with those field values to the same instance of the window.

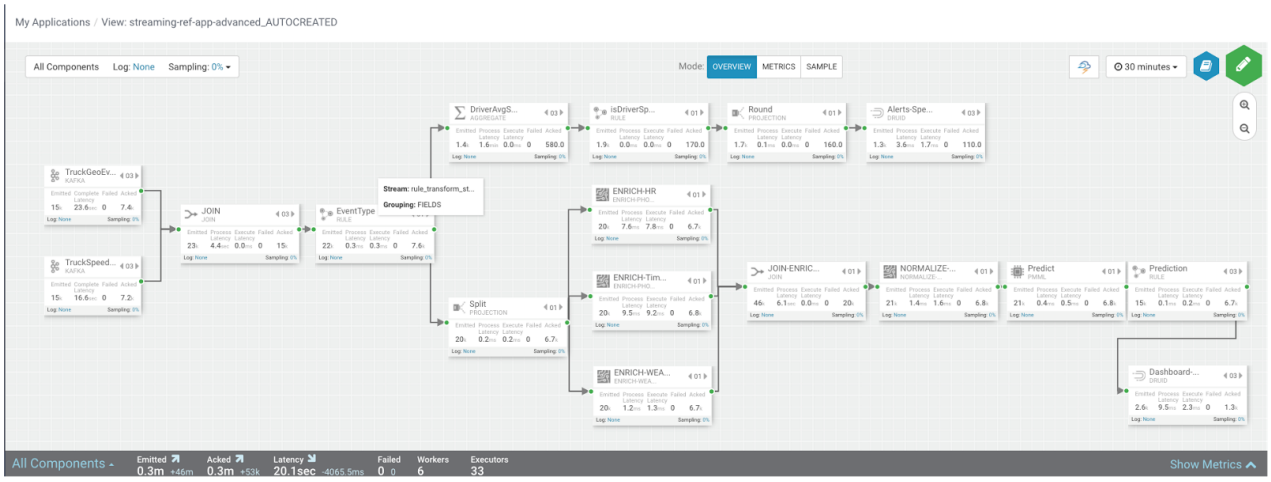


Increase the parallelism of the Dashboard-Predictions Druid sink from 1 to 3 so we can have multiple JVM instances of Druid writing to the cube.



After making these changes, we re-deploy the app using SAM and run the data generator for about 15 minutes and view seeing the following metrics.

SAM's overview and detailed metrics makes it very easy to verify if the performance changes we made had the desired effect.



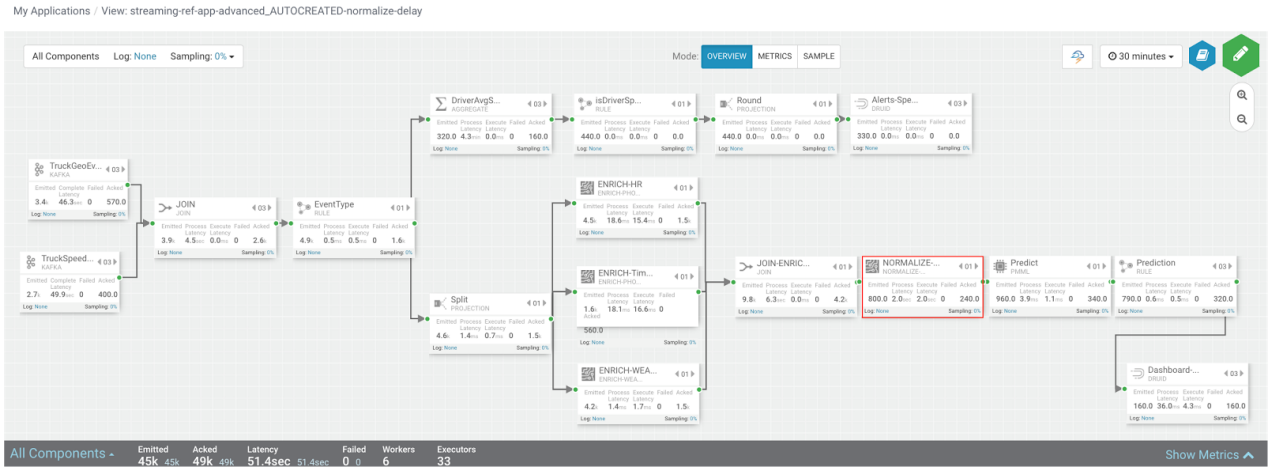
Throughput Improvements for the Kafka Source

The below is the before and after metrics for the TruckGeoEvent Kafka Sink:

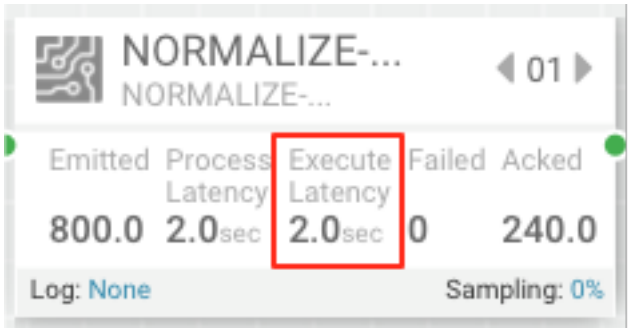
The below is the before and after metrics for the Dashboard-Predictions Druid Sink:

Identifying Processor Performance Bottlenecks

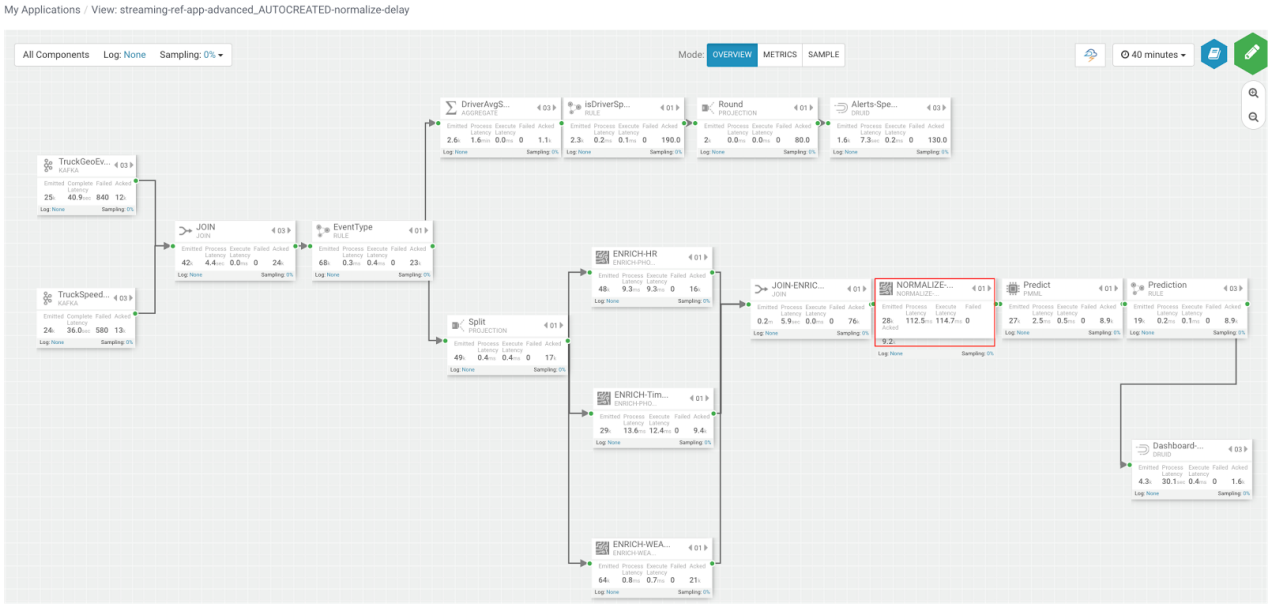
In this scenario, we identify a custom processor that has high latency. After running the data simulator for 30 mins, we view the Overview Metrics of the topology.



Scanning over the metrics, we see that the NORMALIZE-MODEL-FEATURES custom processor has high execute latency of 2 seconds. This means that over the last 30 minutes the average time an event spends in this component is 2 seconds.



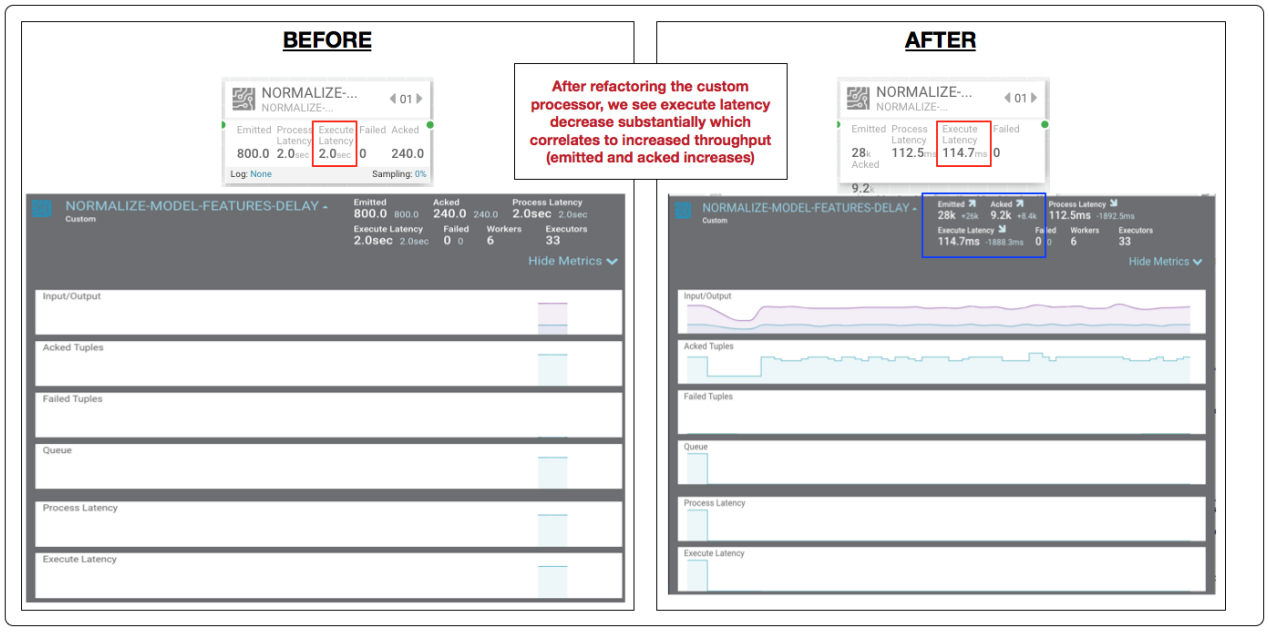
After making changes to the custom processor to address the latench, we re-deploy the app via SAM and run the data generator for about 15 minutes and view seeing the following metrics.



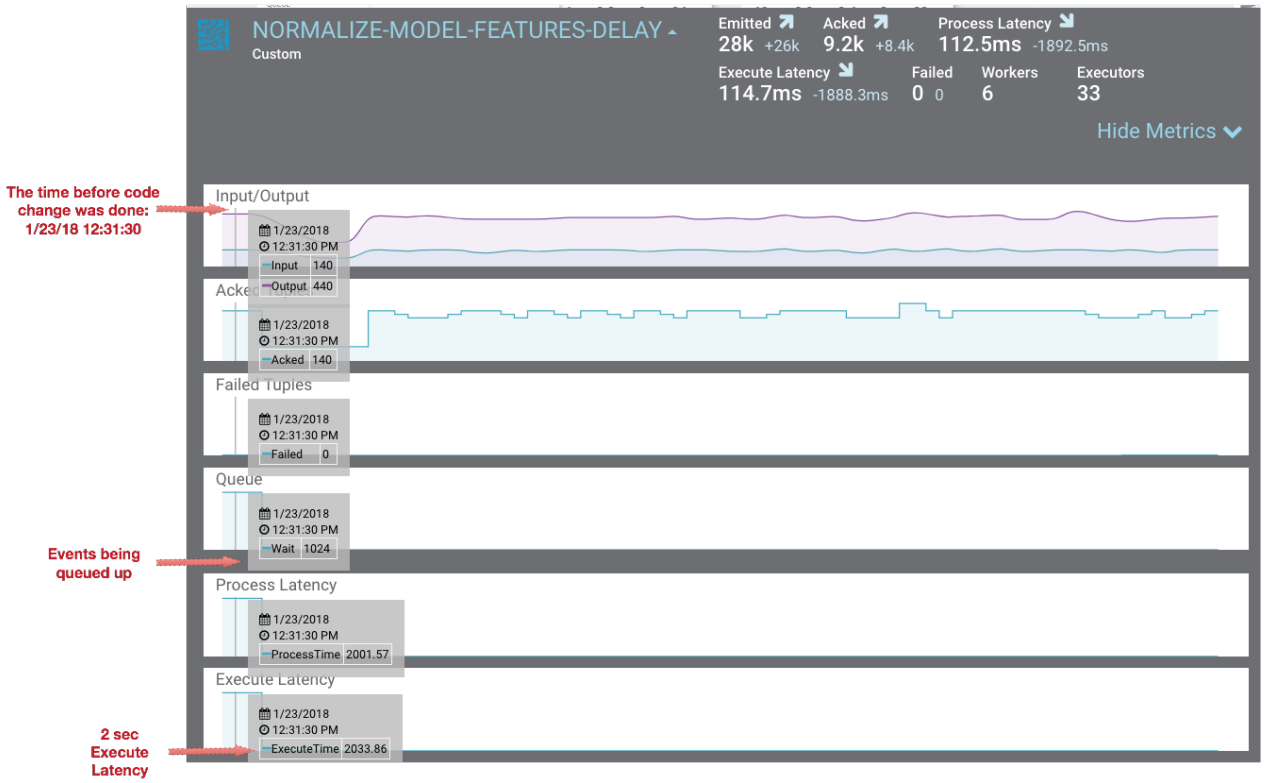
SAM's overview and detailed metrics makes it very easy to verify if the performance changes we made had the desired effect.

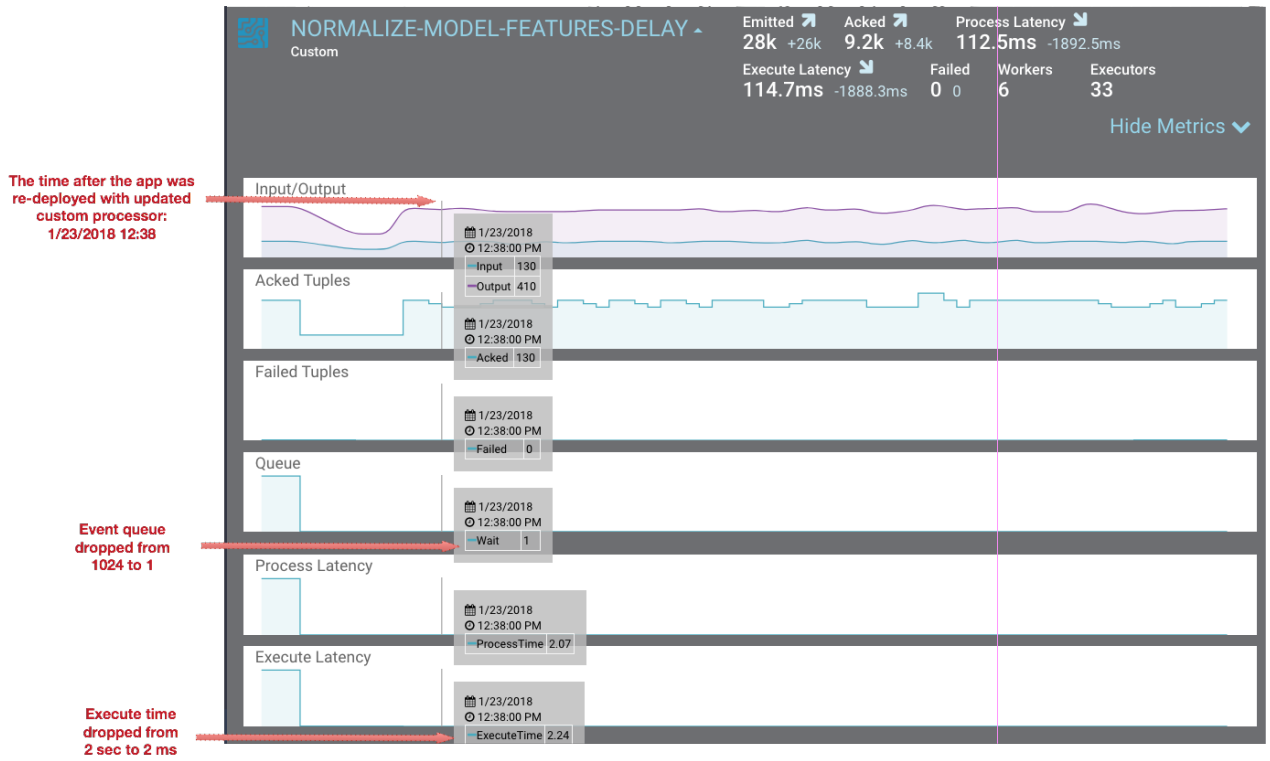
Latency Improvements

The below is the before and after metrics for the NORMALIZE-MODEL-FEATURES custom processor.

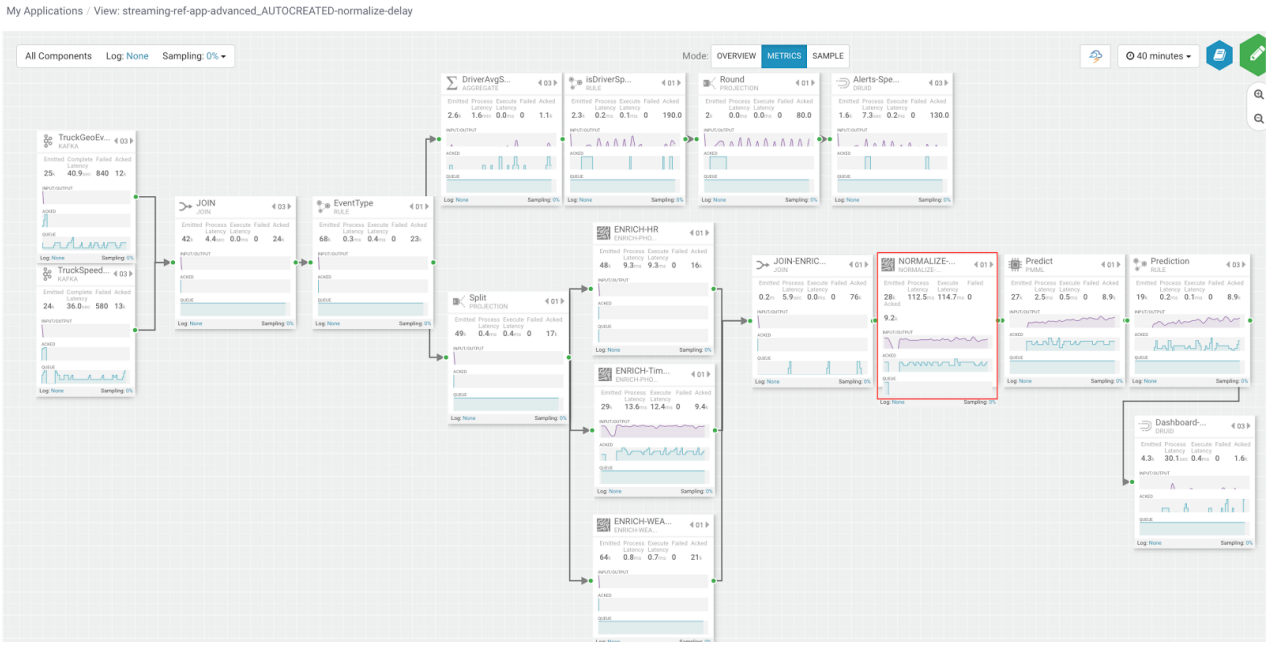


In the metric details view, the graphs provides an easy way to compare metrics before and after the code change.

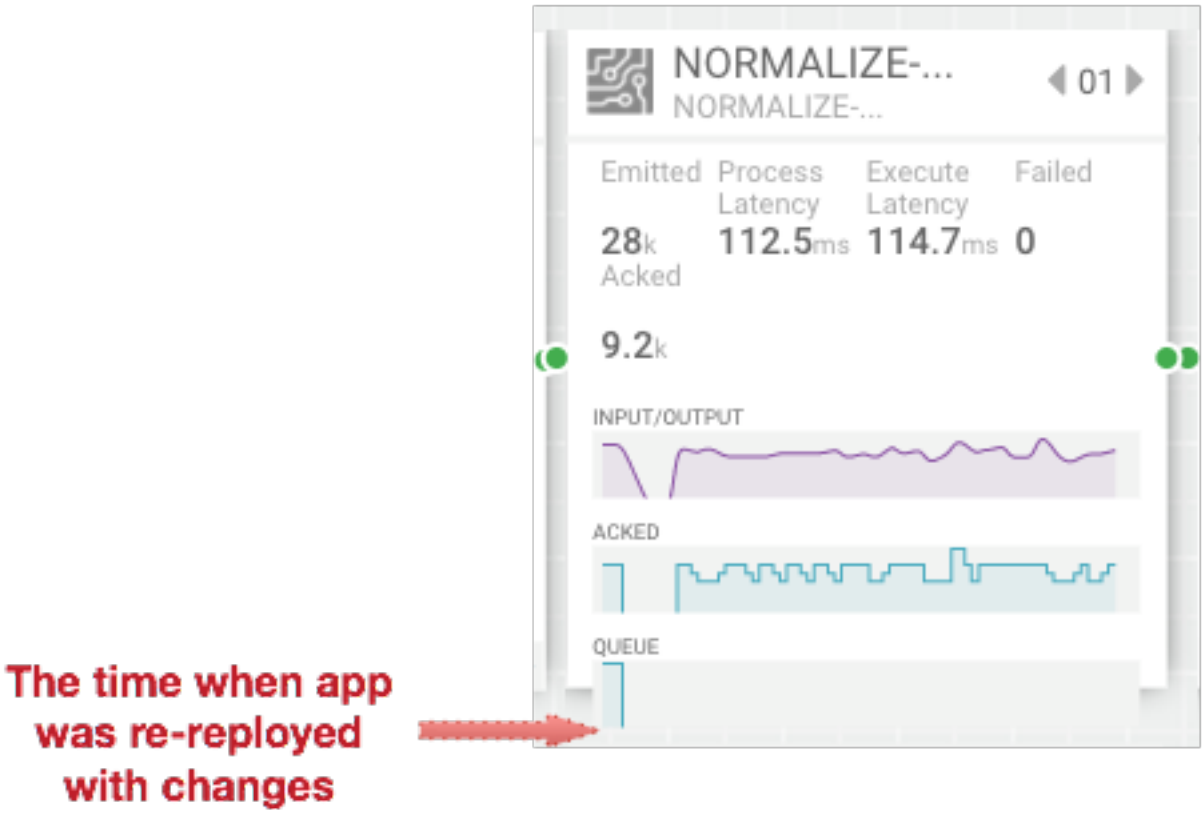




You can also select the Metrics tab to validate the performance improvement.



If you zoom in on the NORMALIZE-MODEL-FEATURES component, you will see that after the code change is made, throughput increases and the wait drops to 0.



Debugging an Application through Distributed Log Search

In a distributed system, searching for logs on different hosts for different components can be extremely tedious and time consuming. With SAM, all the application logs are indexed via the Ambari Log Search Server via Solr. SAM makes it easy to drill into and search for logs for specific components directly from the DAG view. Follow the below steps to use distributed log search:

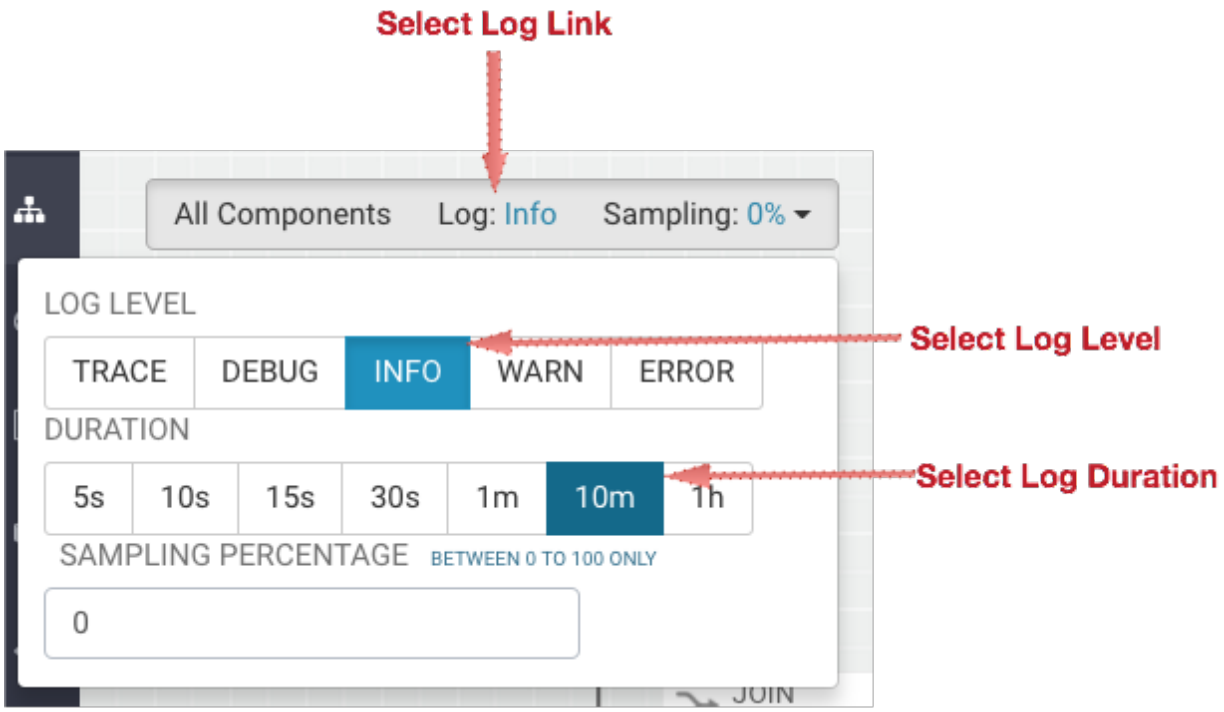
Procedure

1. To enable Log Search in SAM, perform the following actions in Ambari.
   a. In Ambari, select the Log Search service and select 'Log Search UI' from Quick Links.
   b. Select the filter icon on the top right menu.
   c. For the storm_worker component, configure the filter like the following and click Save.

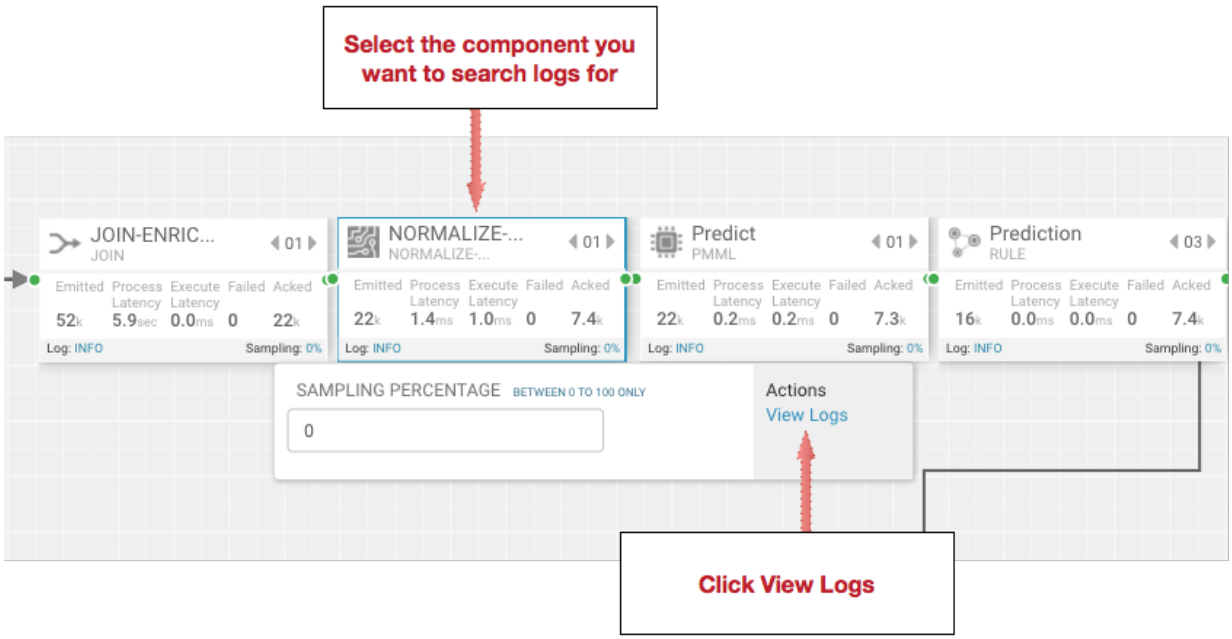
Log Feeder Log Levels Filter

Components	Override	FATAL	ERROR	WARN	INFO	DEBUG	TRACE	UNKNOWN
storm_worker	☐	☑	☑	☑	☑	☑	☑	☐

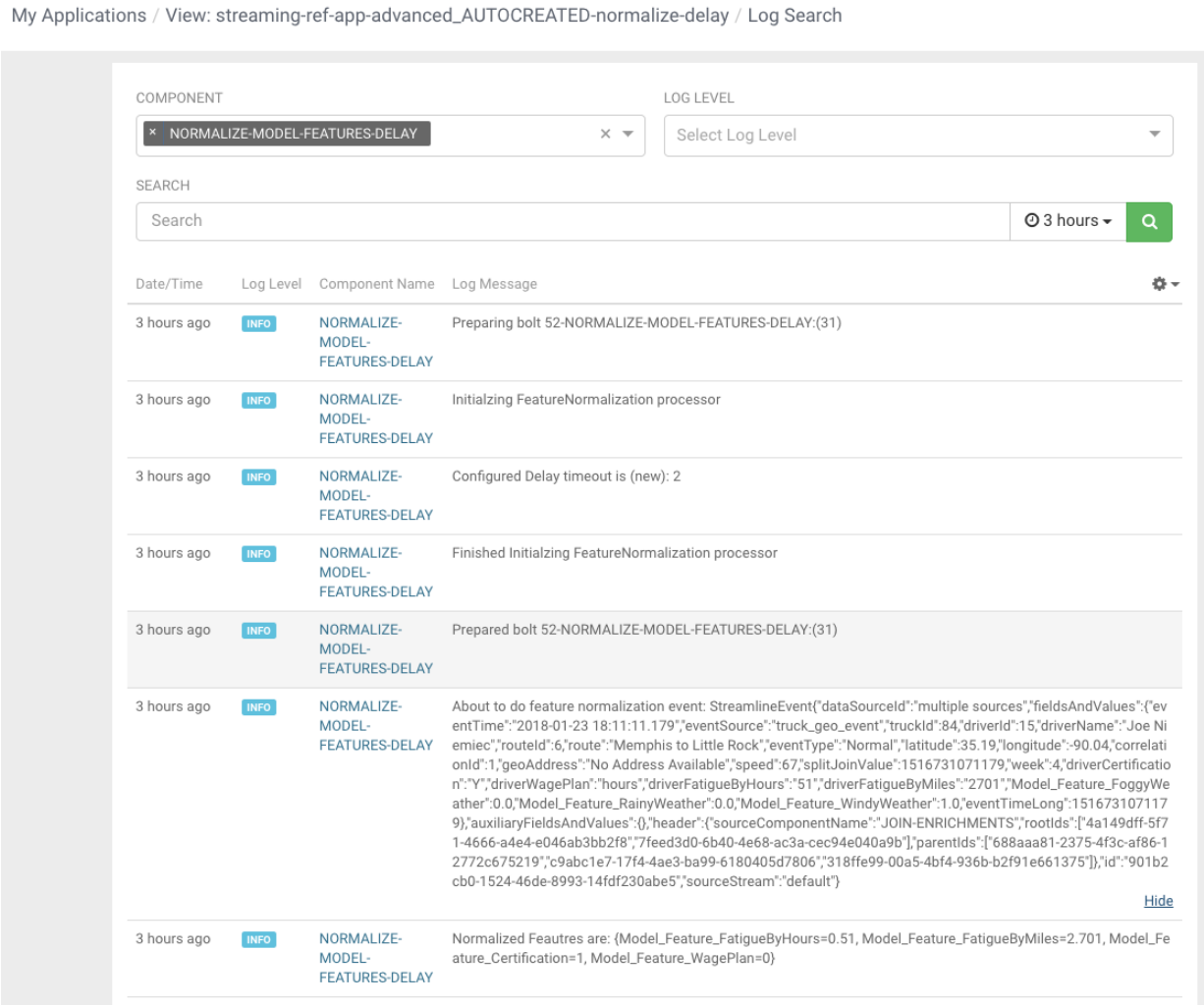
2. In SAM, you can dynamically change the logging level. For example, in SAM view mode of an application, click on the Log link, select the log level and the duration you want that log level.



3. Then click on the component you want to search logs for and under Actions select Logs.



4. This brings you to the Log Search page where you can search by component (s), log level(s) and search for strings using wildcard notation.

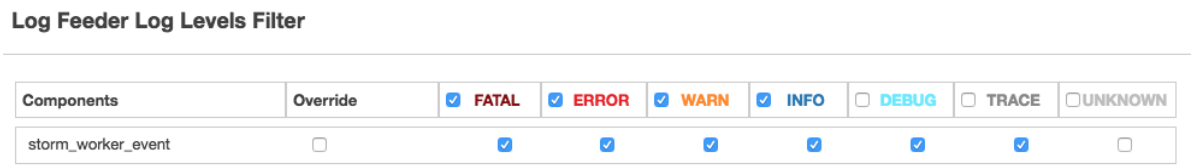


Debugging an Application through Sampling

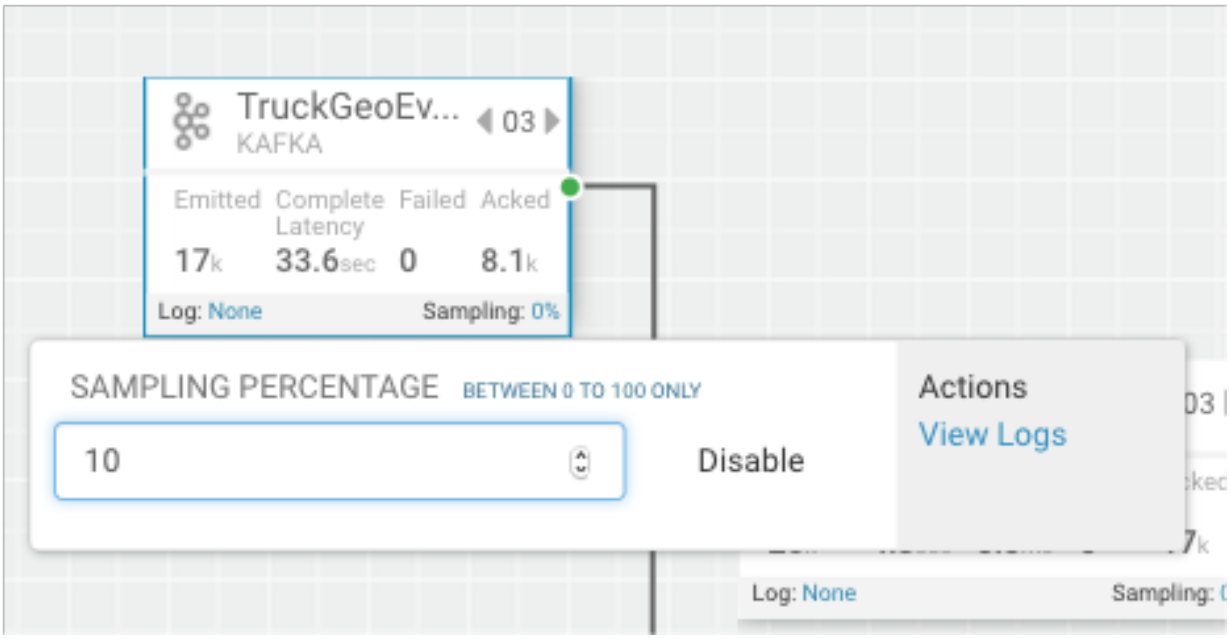
For troubleshooting, a convenient tool is to turn on sampling for a component or for the entire topology based on a sample percentage. Sampling allows you to log the emitted values of a component in the SAM App.

Procedure

1. To enable Log Search in SAM, perform the following actions in Ambari.
   a. In Ambari, select the Log Search service and select ‘Log Search UI’ from Quick Links.
   b. Select the filter icon on the top right menu.
   c. For the storm_worker_event component, configure the filter like the following and click Save.

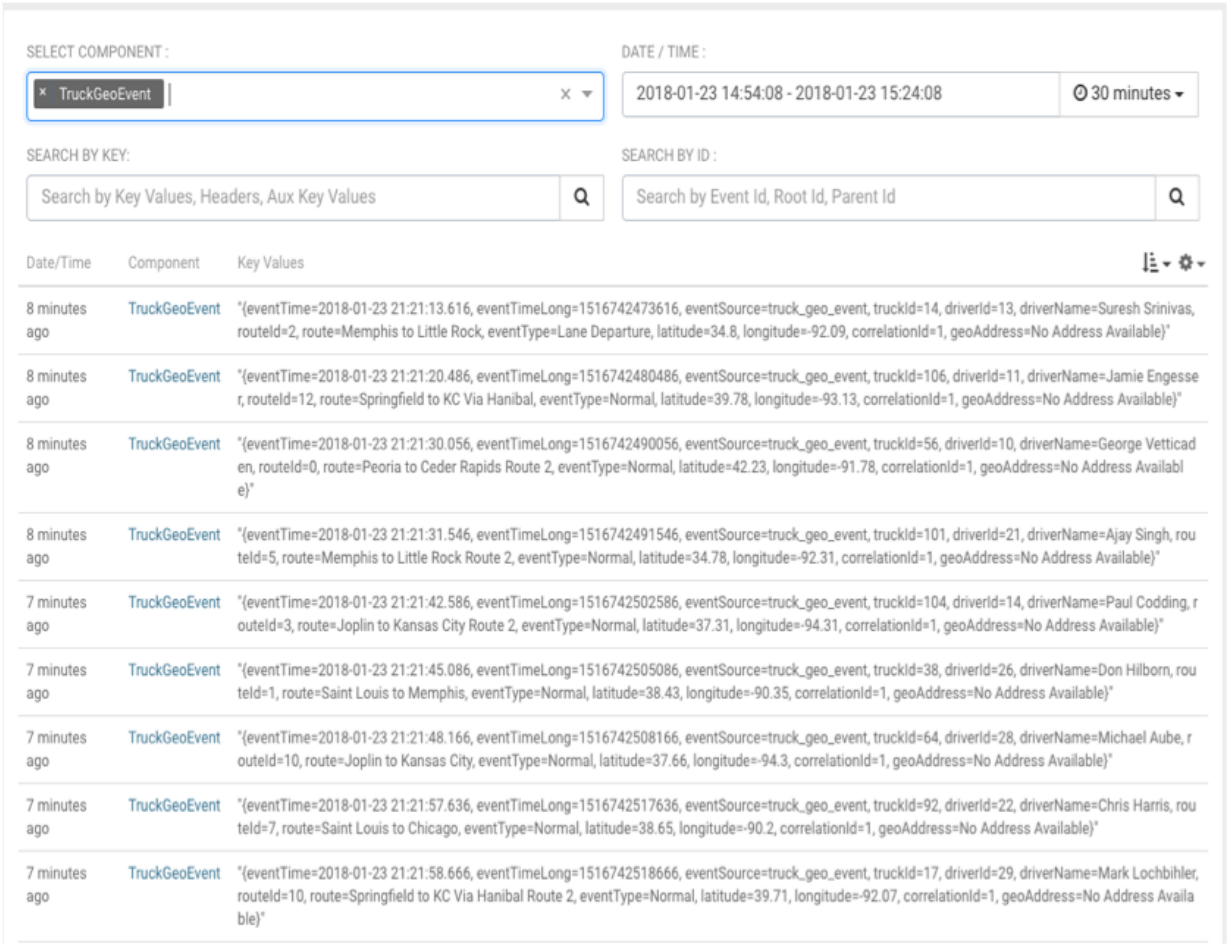


2. In SAM view mode of the App, click on the component you want to turn on sampling for and enter a sampling percentage.



3. Click the ‘SAMPLE’ Tab .

4. Use the Sample Search UI to search for different events that were logged.



Date/Time	Component	Key Values
8 minutes ago	TruckGeoEvent	"{eventTime=2018-01-23 21:21:13.616, eventTimeLong=1516742473616, eventSource=truck_geo_event, truckId=14, driverId=13, driverName=Suresh Srinivas, routeId=2, route=Memphis to Little Rock, eventType=Lane Departure, latitude=34.8, longitude=-92.09, correlationId=1, geoAddress=No Address Available}"
8 minutes ago	TruckGeoEvent	"{eventTime=2018-01-23 21:21:20.486, eventTimeLong=1516742480486, eventSource=truck_geo_event, truckId=106, driverId=11, driverName=Jamie Engesser, routeId=12, route=Springfield to KC Via Hanibal, eventType=Normal, latitude=39.78, longitude=-93.13, correlationId=1, geoAddress=No Address Available}"
8 minutes ago	TruckGeoEvent	"{eventTime=2018-01-23 21:21:30.056, eventTimeLong=1516742490056, eventSource=truck_geo_event, truckId=56, driverId=10, driverName=George Vetticaden, routeId=0, route=Peoria to Ceder Rapids Route 2, eventType=Normal, latitude=42.23, longitude=-91.78, correlationId=1, geoAddress=No Address Available}"
8 minutes ago	TruckGeoEvent	"{eventTime=2018-01-23 21:21:31.546, eventTimeLong=1516742491546, eventSource=truck_geo_event, truckId=101, driverId=21, driverName=Ajay Singh, routeId=5, route=Memphis to Little Rock Route 2, eventType=Normal, latitude=34.78, longitude=-92.31, correlationId=1, geoAddress=No Address Available}"
7 minutes ago	TruckGeoEvent	"{eventTime=2018-01-23 21:21:42.586, eventTimeLong=1516742502586, eventSource=truck_geo_event, truckId=104, driverId=14, driverName=Paul Codding, routeId=3, route=Joplin to Kansas City Route 2, eventType=Normal, latitude=37.31, longitude=-94.31, correlationId=1, geoAddress=No Address Available}"
7 minutes ago	TruckGeoEvent	"{eventTime=2018-01-23 21:21:45.086, eventTimeLong=1516742505086, eventSource=truck_geo_event, truckId=38, driverId=26, driverName=Don Hilborn, routeId=1, route=Saint Louis to Memphis, eventType=Normal, latitude=38.43, longitude=-90.35, correlationId=1, geoAddress=No Address Available}"
7 minutes ago	TruckGeoEvent	"{eventTime=2018-01-23 21:21:48.166, eventTimeLong=1516742508166, eventSource=truck_geo_event, truckId=64, driverId=28, driverName=Michael Aube, routeId=10, route=Joplin to Kansas City, eventType=Normal, latitude=37.66, longitude=-94.3, correlationId=1, geoAddress=No Address Available}"
7 minutes ago	TruckGeoEvent	"{eventTime=2018-01-23 21:21:57.636, eventTimeLong=1516742517636, eventSource=truck_geo_event, truckId=92, driverId=22, driverName=Chris Harris, routeId=7, route=Saint Louis to Chicago, eventType=Normal, latitude=38.65, longitude=-90.2, correlationId=1, geoAddress=No Address Available}"
7 minutes ago	TruckGeoEvent	"{eventTime=2018-01-23 21:21:58.666, eventTimeLong=1516742518666, eventSource=truck_geo_event, truckId=17, driverId=29, driverName=Mark Lochbihler, routeId=10, route=Springfield to KC Via Hanibal Route 2, eventType=Normal, latitude=39.71, longitude=-92.07, correlationId=1, geoAddress=No Address Available}"