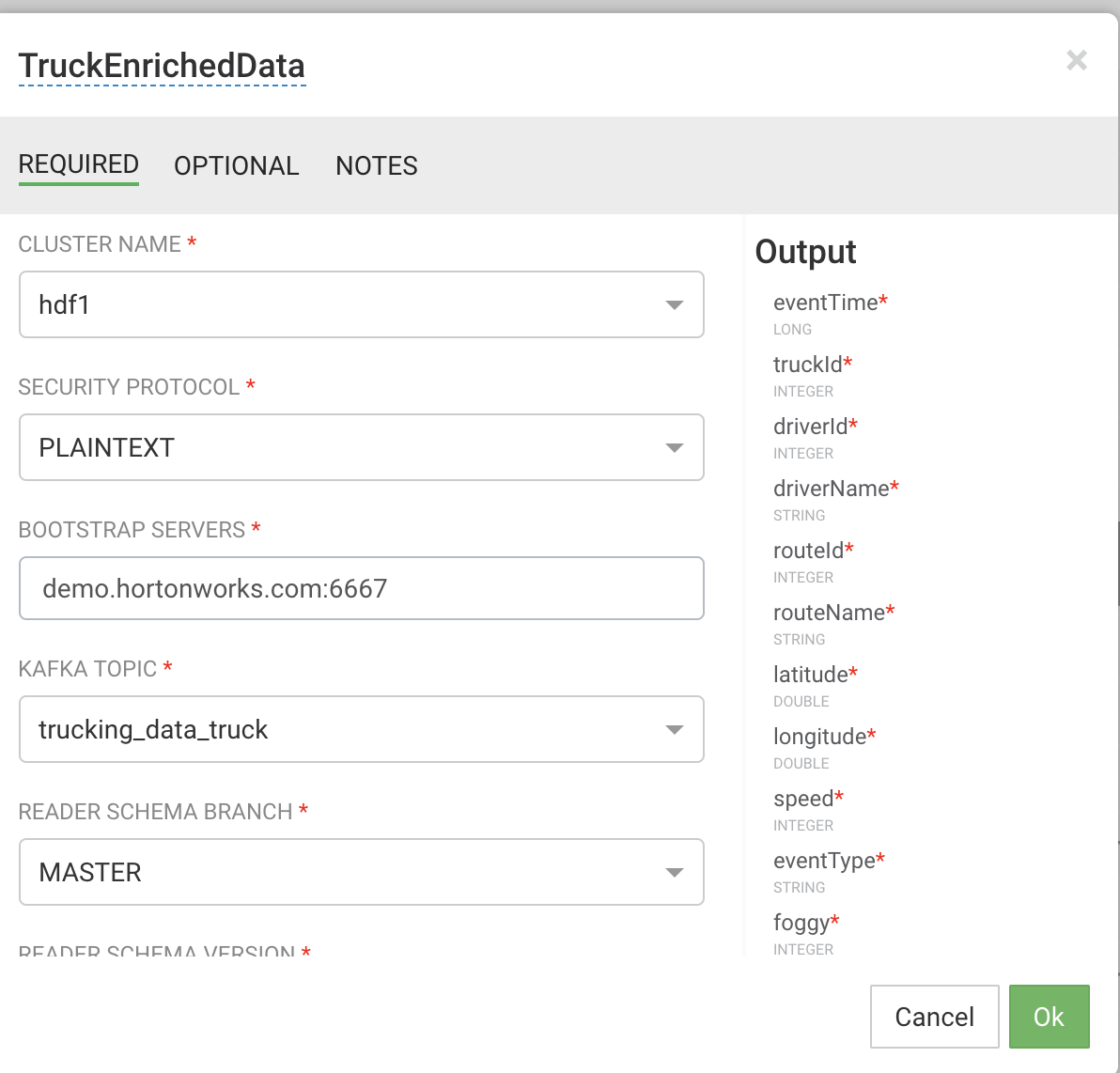
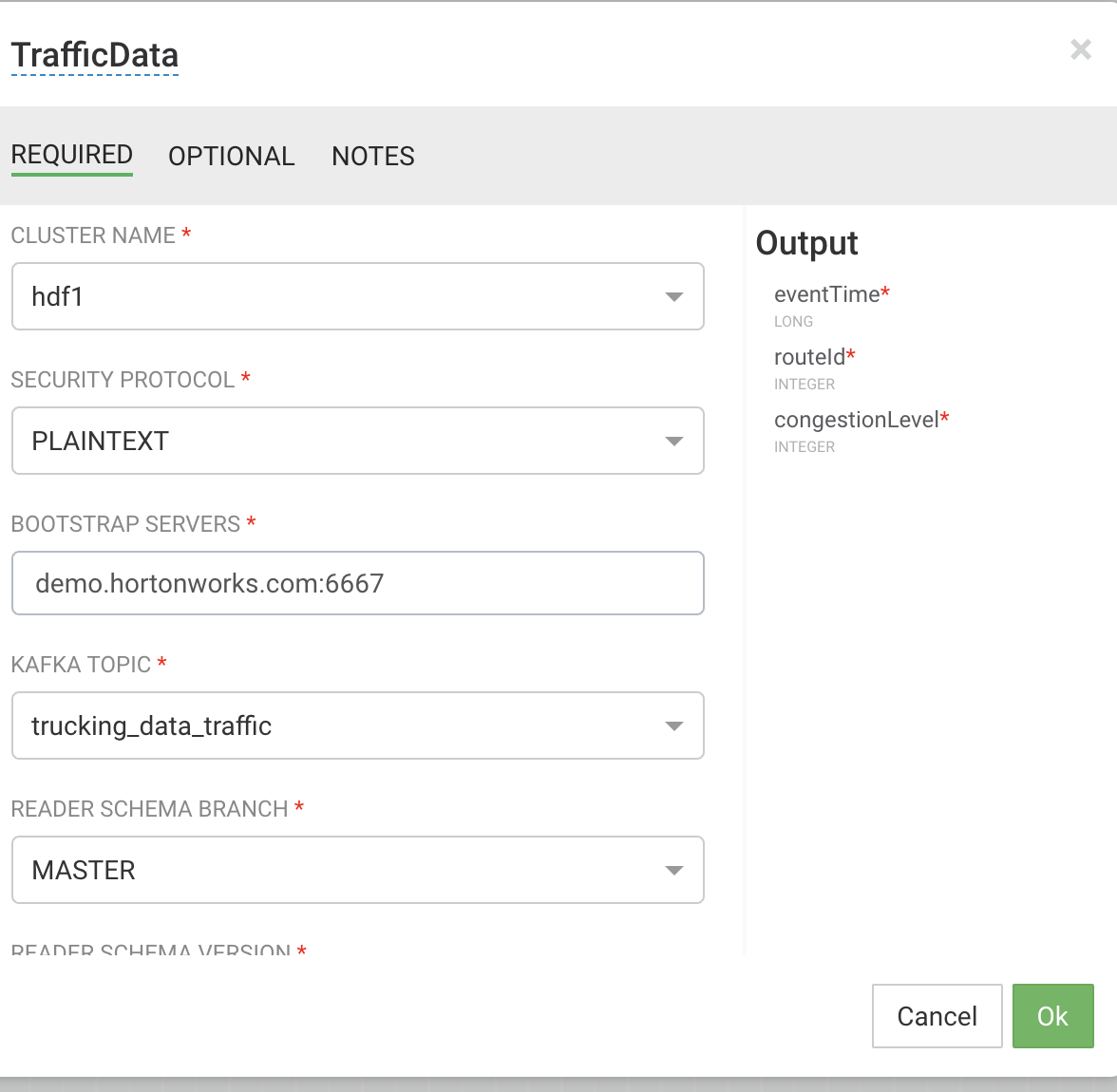
1. From amabari, launch SAM
2. Click on the spanner icon (left pane) and create a new service pool using auto add. Specify the url as shown in the UI.
3. Click on the spanner icon (left pane) and choose to add an environment. Choose all the services from the service pool.
4. Click my application (first icon in the left pane) and click + (top right corner)
5. Give a name without space (application name with space is not supported)
6. In the environment drop down choose the environment you just created.
7. Drag and drop a kafka source from the left pane and set is as below



Set the CONSUMER GROUP ID to trucking\_data\_truck\_enriched1

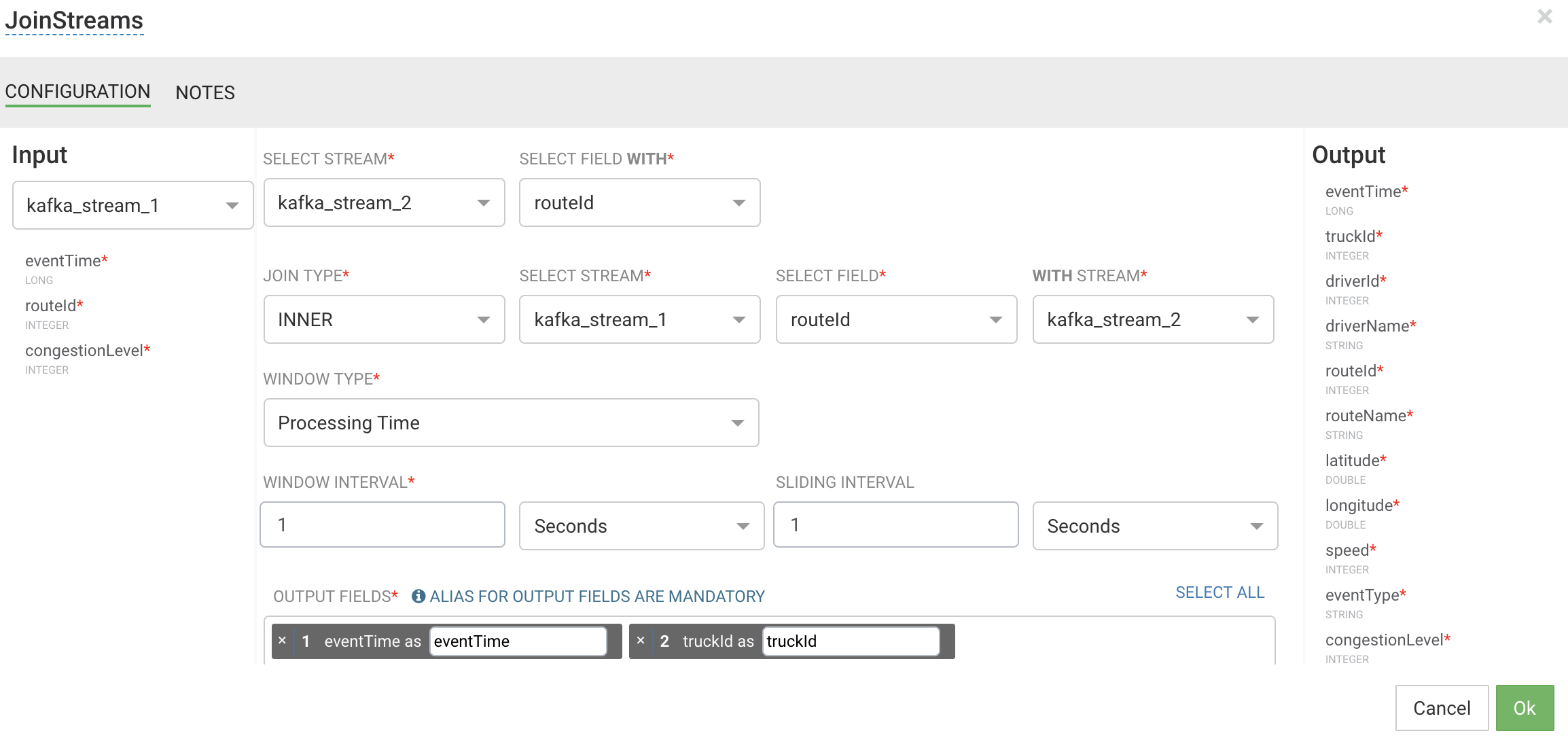
Set the bootstrap server to your server (default port is 6667)

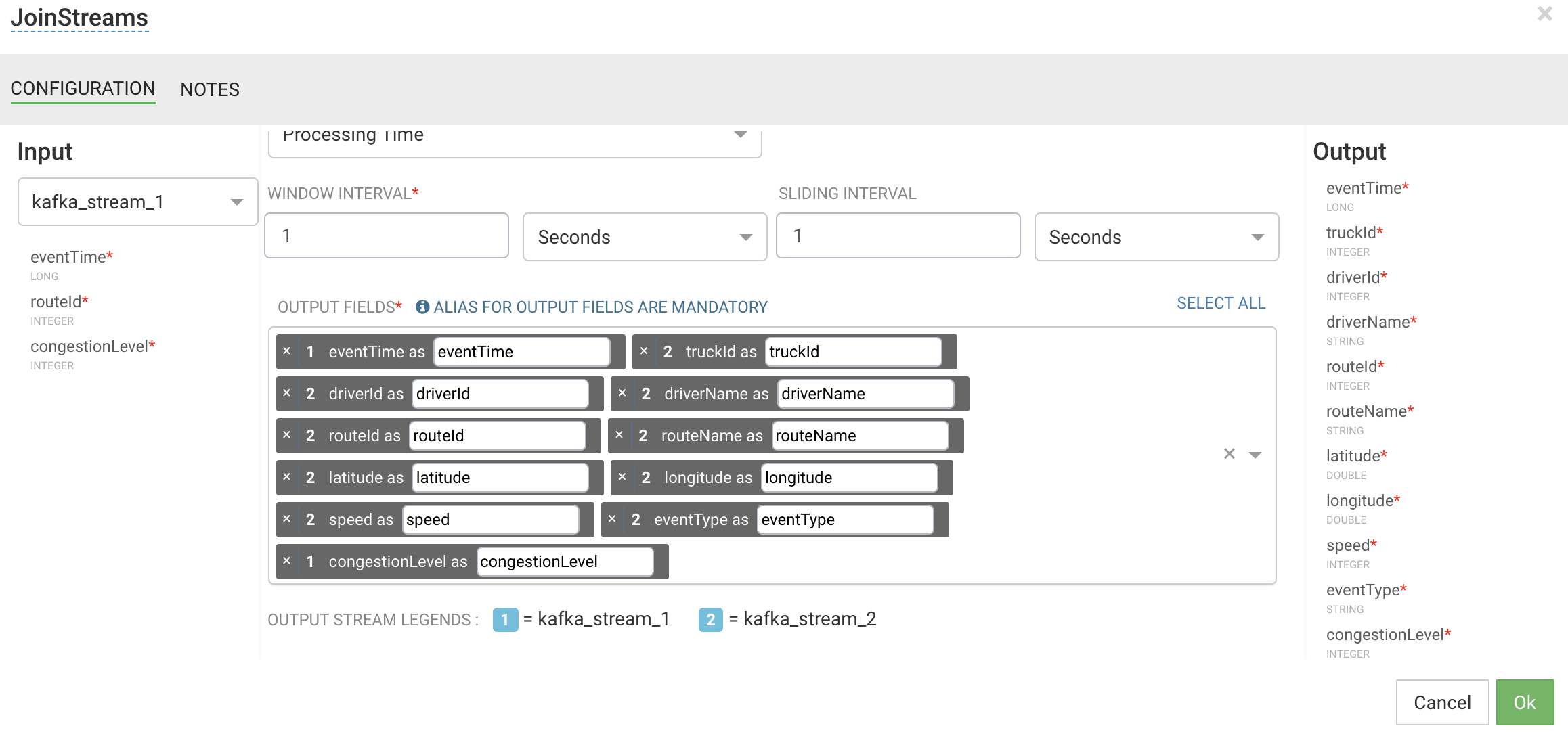
1. Drag and drop another kafka source and set the properties to below



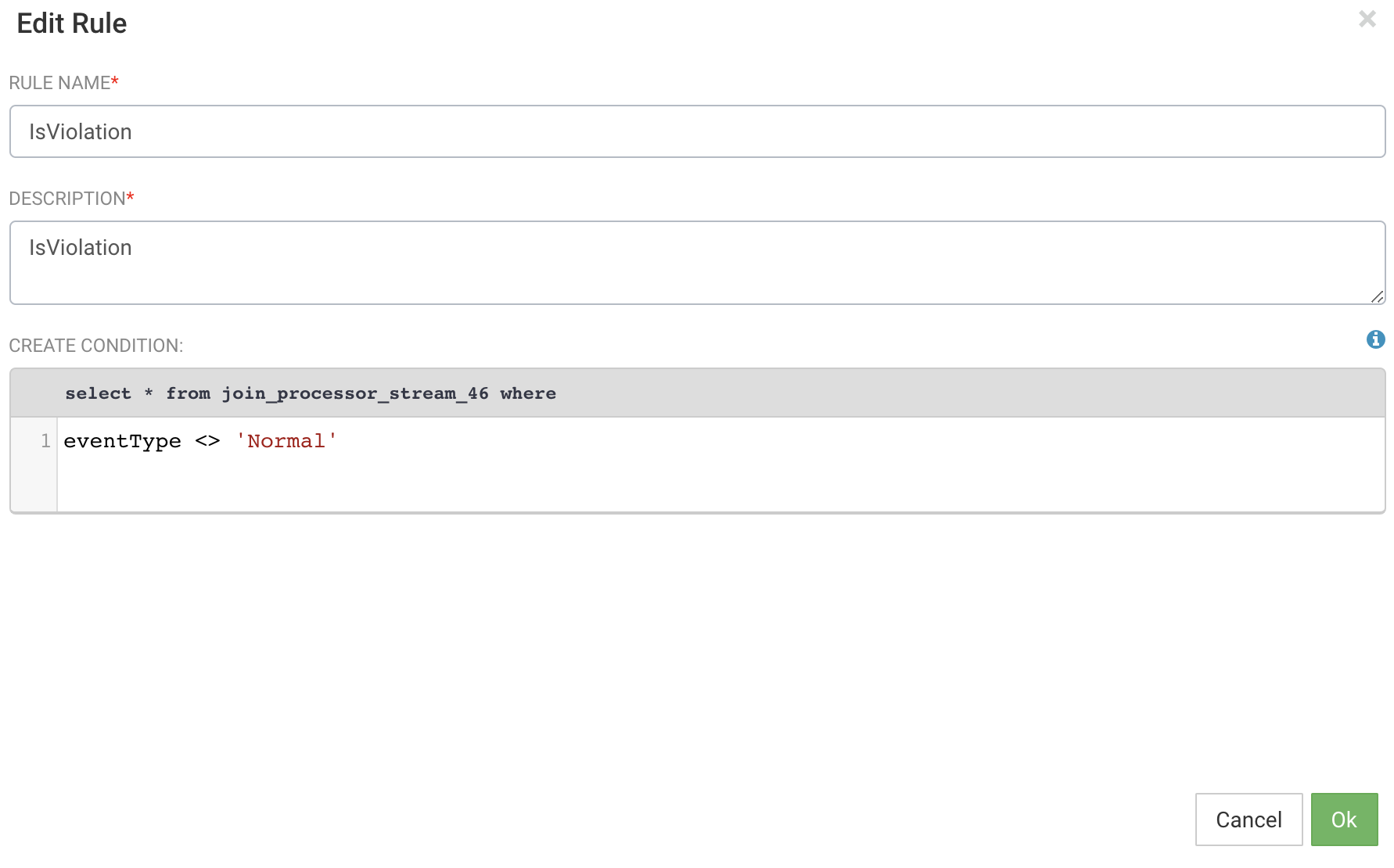
Set the consumer group id to trucking\_data\_traffic1

1. Drag and drop a join to the canvas and connect the two kafka sources to the join and set the properties below

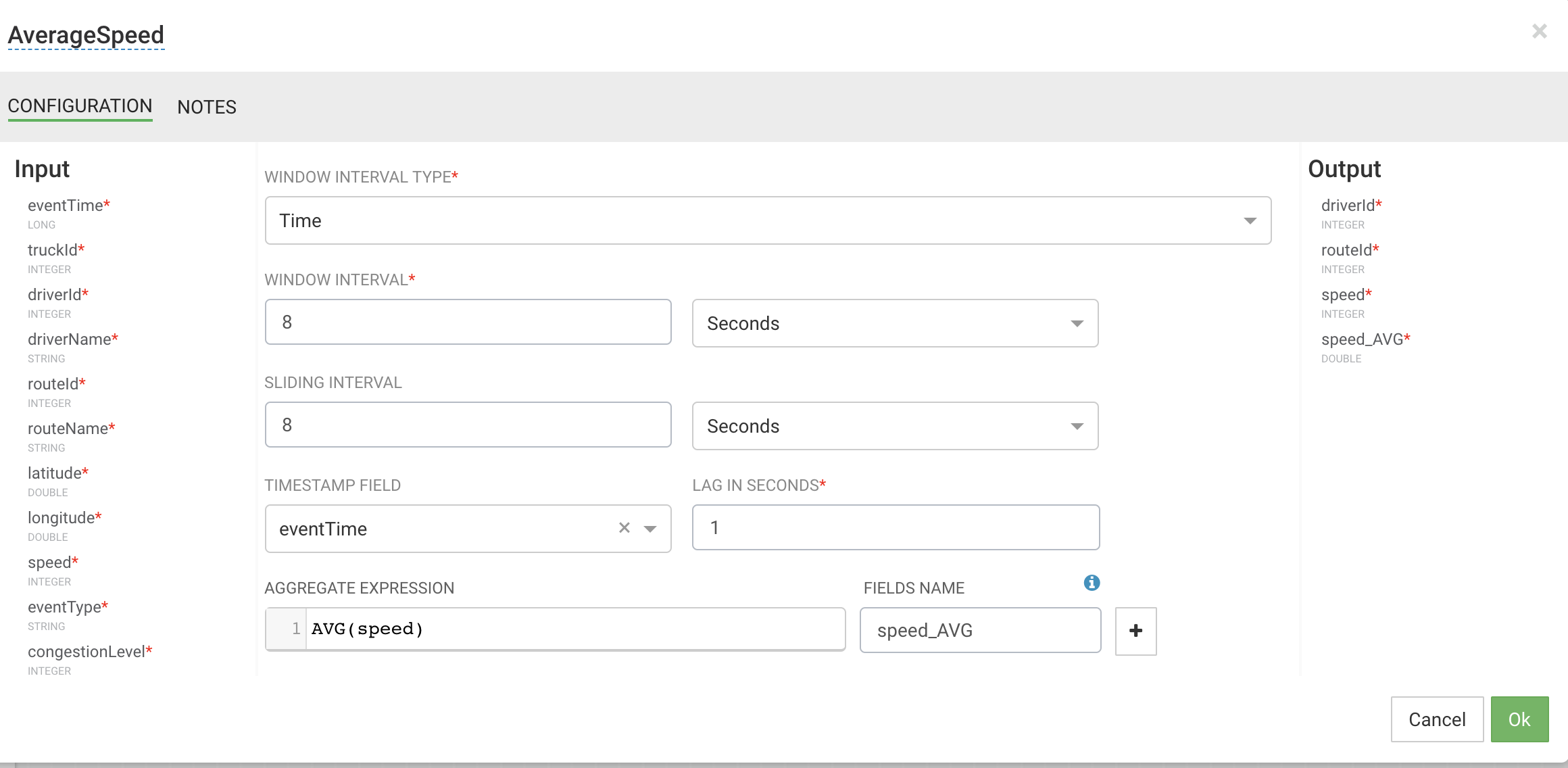




1. Add a rule processor from the left pane and connect the join to the rule and then double click on add new rules and set it to the following



1. Drop the aggregate processor to the canvas and connect the above rule to the aggregate. Choose the aggregate fields as driverid, routeid and speed and the average function to AVG(speed)



1. Drop a druid sink onto the canvas and connect the average to the druid sink. Set the following properties on the druid sink

Cluster name: hdf1

Name of the indexing service: druid/overlord

Service discovery path: /druid/discovery

Data source name: average-speed-cube-01

Zookeeper connect string: demo.hortonworks.com:2181. (change the hostname to your hostname)

Dimensions: driverid,routeid,speed,speed\_AVG

Timestamp field: processingTime

Window period: PT1M

Index retry period: PT1M

Segment granularity: second

Query granularity:second

1. Drop a hdfs sink onto the canvas and connect to the average and set below properties

Cluster name: hdf1

Hdfs url: hdfs://demo.hortonworks.com:8020. (change hostname to your host)

Path: /apps/trucking/average-speed. (change permisions on /. Hdfs dfs -chmod 777 /.

Flush count : 1000

Rotation policy : file based rotation

Rotation size multiplier:500

Rotation size unit: KB

Output field: driverid, routeid,speed speed\_AVG

1. Drop a druid sink and connect to the violation rule and set the following

Cluster name: hdf1

Name of the indexing service: druid/overlord

Service discovery path: /druid/discovery

Data source name: violation-events-cube-01

Zookeeper connect string: demo.hortonworks.com:2181. (change the hostname to your hostname)

Dimensions: eventTime,routeid,congestionLevel,trucked,driverid,driverName,routeName,latitude,longitude,speed,eventType

Timestamp field: processingTime

Window period: PT1M

Index retry period: PT1M

Segment granularity: second

Query granularity:second

1. Drop a hdfs sink and connect to the violation rule and set to the following

Cluster name: hdf1

Hdfs url: hdfs://demo.hortonworks.com:8020. (change hostname to your host)

Path: /apps/trucking/violation-events. (change permisions on /. Hdfs dfs -chmod 777 /.

Flush count : 1000

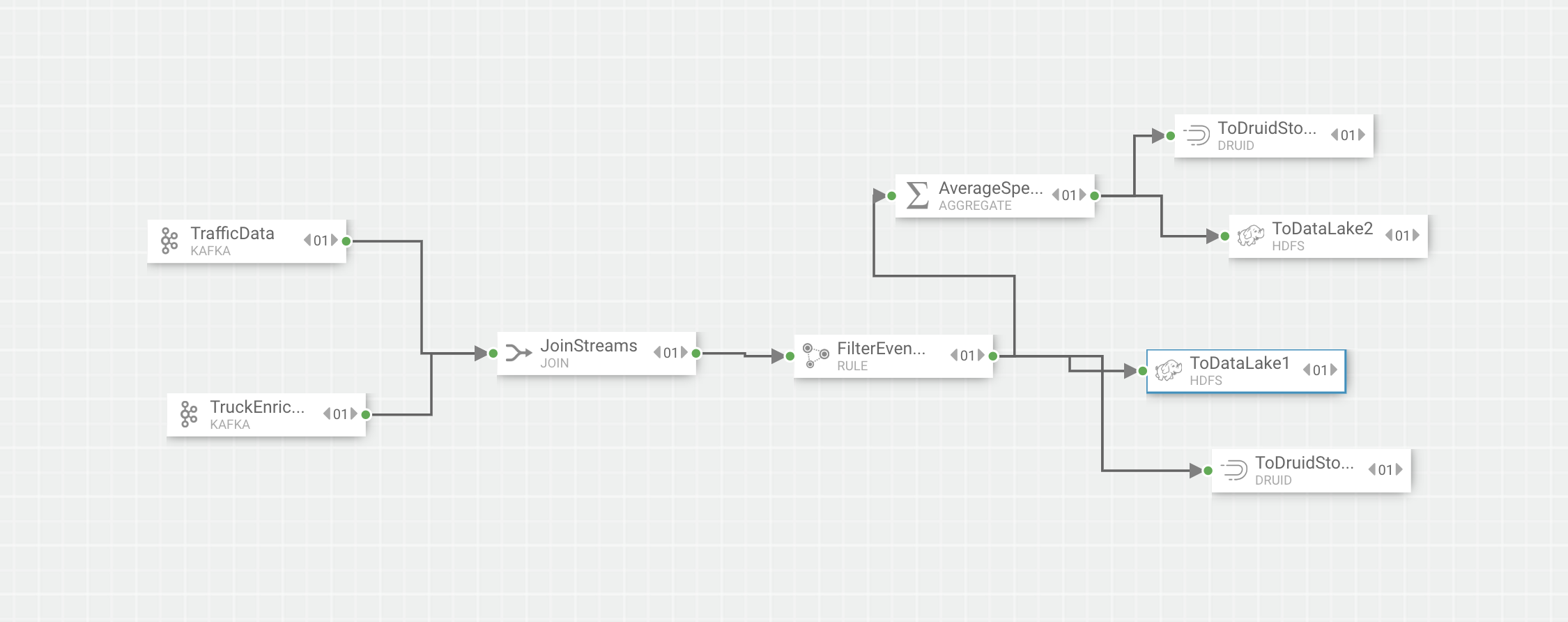
Rotation policy : file based rotation

Rotation size multiplier:500

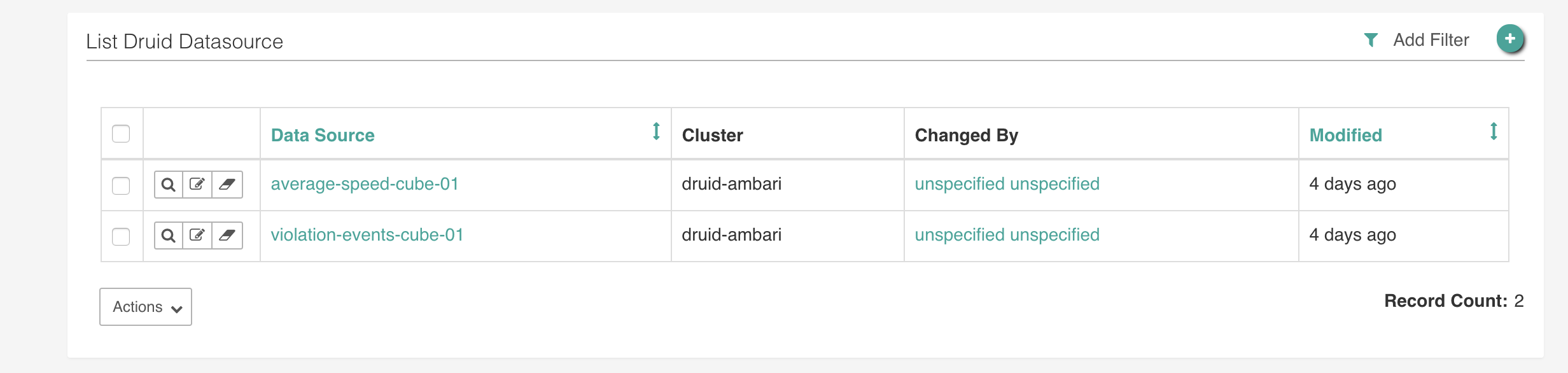
Rotation size unit: KB

Output field: eventTime,routeid,congestionLevel,trucked,driverid,driverName,routeName,latitude,longitude,speed,eventType

1. Your final flow should look like below



1. The finished application has been uploaded to <https://github.com/vnarayaj/nifitrucking>
2. Start the nifi flow and the SAM application
3. From ambari login to superset. Click on sources drop down and select refresh druid metadata. You should see two cubes



1. Click on the violation events cube
2. Set the query and chart options as below and run the query

