Rules and Statistical analysis

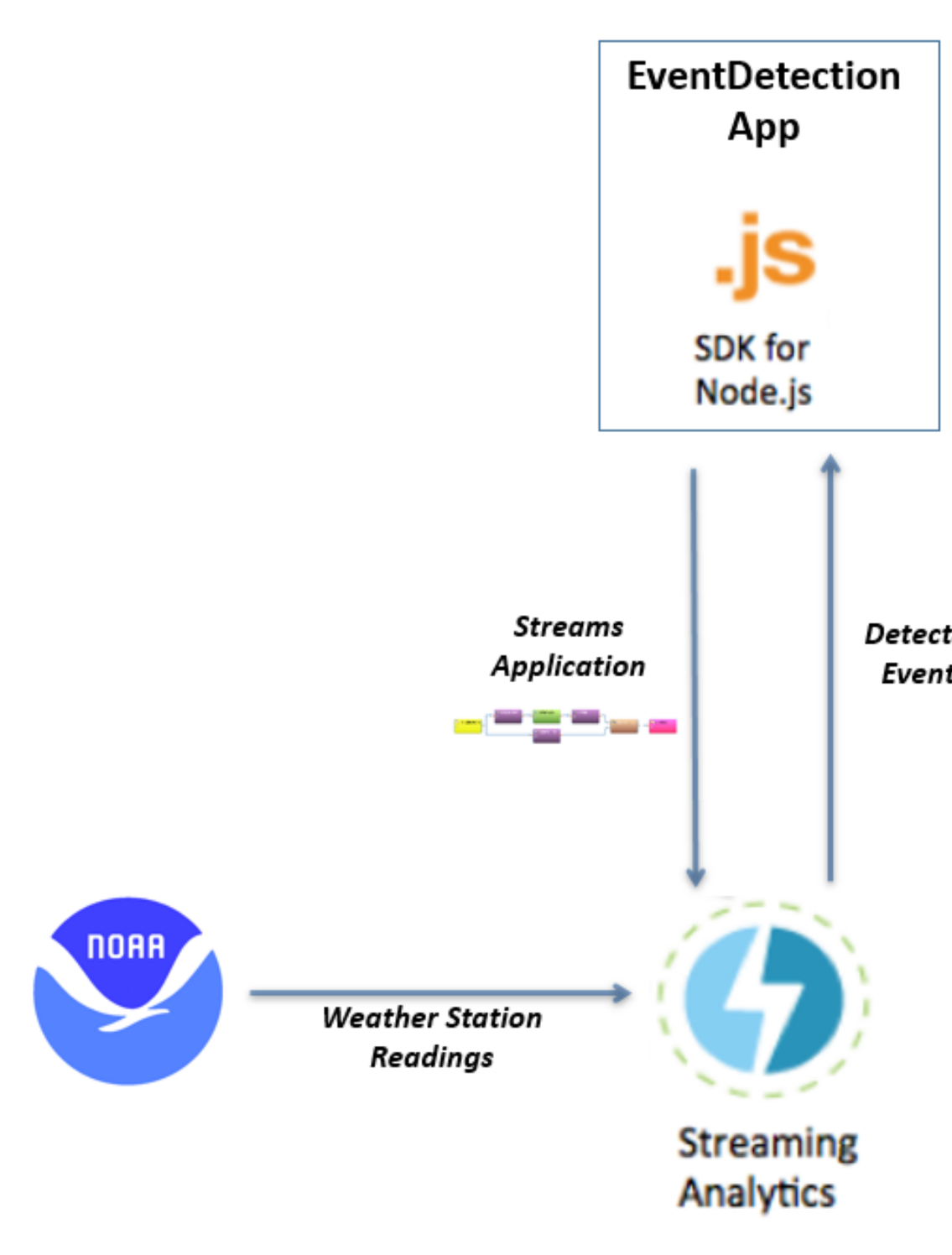
Detect Complex Events in Real-time with Streaming Analytics in the IBM Cloud

MikeBranson
Published on April 23, 2018 / Updated on November 26, 2019

Do you want to perform complex event detection on information from real-time data sources and act quickly when events are found? It's easier than you might think when you use the Streaming Analytics in IBM Cloud. To show how easy, I built a starter app that:

- Uses the Streaming Analytics service from a Node.js web app in the IBM Cloud
- Ingests a stream of data into Streaming Analytics
- Recognizes patterns and detects events in the data stream
- Sends the results of the analysis to the web app

This tutorial explains how to obtain, run, and extend this starter app, called EventDetection. Here's a graphical overview of the solution components:



The EventDetection app is implemented via the SDK for Node.js runtime. The app provides a simple web UI to display status and results of the analysis. The Node.js app is bound to an instance of the Streaming Analytics service. The app controls the service via the Streaming Analytics REST API.

Analytics are performed by an IBM Streams application that implements the event detection against a stream of weather data. A Streams application bundle is submitted by the Node.js app to deploy the Streams application to the Streaming Analytics instance running in IBM Cloud.

The data being analyzed is weather data available on the public Internet from the National Oceanic and Atmospheric Administration (NOAA). After the Streams application is deployed into the Streaming Analytics instance, it continuously ingests and analyzes the weather data until the Streams application is stopped.

When the Streams application detects events, they're sent to the Node.js app for display on the app's web UI.

Please Note: NOAA data is available for a certain number of past hours. The application analyzes this past data first and then picks up real-time data as it is published by NOAA. For this reason, the application detects a large number of events right away and then detects events more sporadically, as new data is posted. Most NOAA weather stations post new readings hourly, but some updates might come more frequently.

Step 1: Deploy the Sample

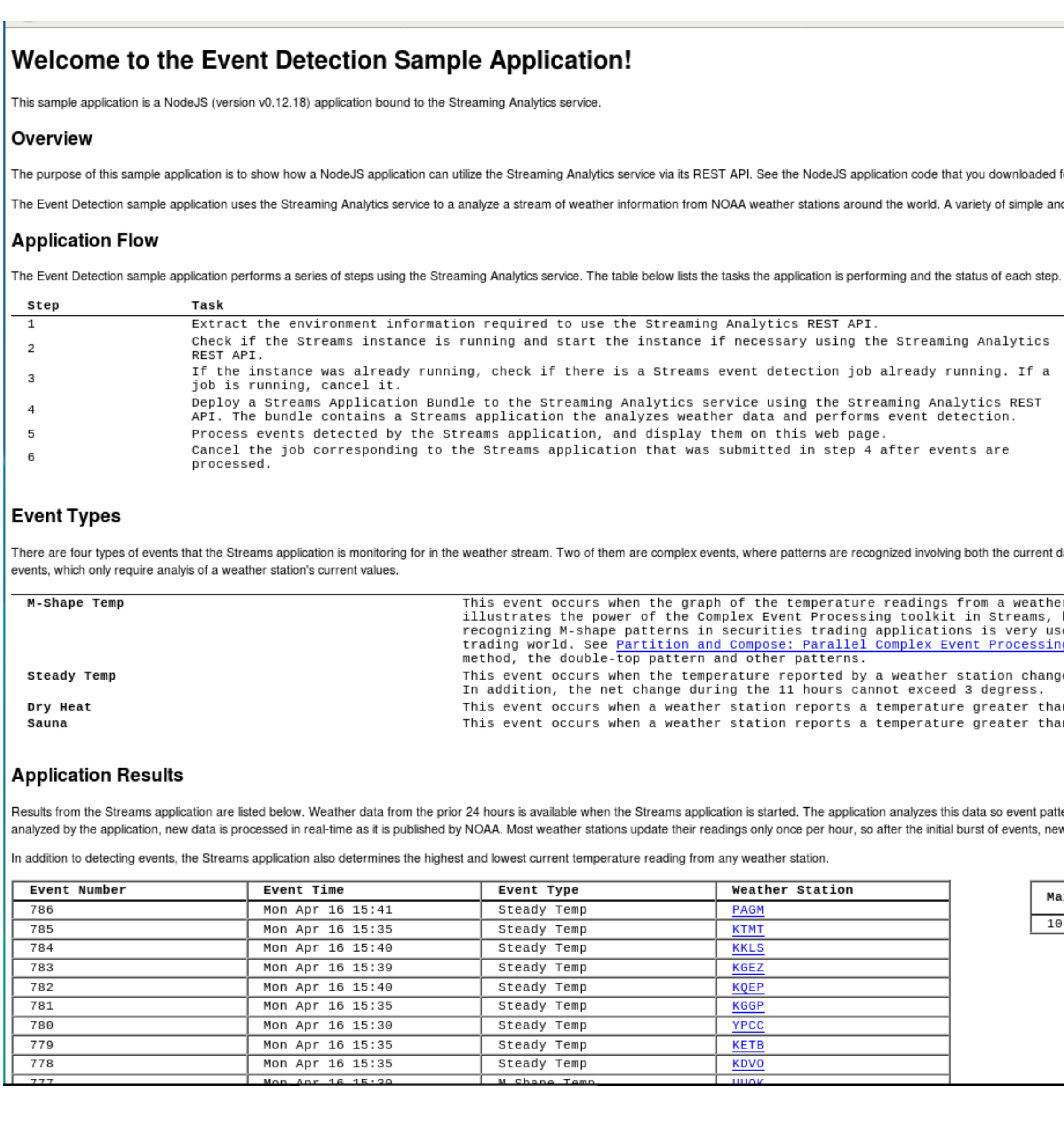
To run the Event Detection sample, you need to access the IBM Streams Samples Catalog, and find the EventDetectionV2 sample.

After locating the sample, follow the step-by-step instructions in its README.md file to run the sample.

Step 2: View the Running Sample

The last step of the deployment instructions in the README.md file has you visiting the URL of the Node.js application in your web browser. When you bring up the web app in your browser, you will see a basic web page with the title: **Welcome to the Event Detection Sample Application!** as shown in the figure below.
The web page is broken up into a few different sections:

- The **Application Flow** section lists the steps that are being performed by the application and their status.
- The **Event Types** section defines the types of events the application is detecting.
- The **Application Results** section displays the events as they are detected. It also displays the highest and lowest temperature currently reported.



Welcome to the Event Detection Sample Application!

This sample application is a NodeJS (version v0.12.18) application bound to the Streaming Analytics service.

Overview

The purpose of this sample application is to show how a NodeJS application can utilize the Streaming Analytics service via its REST API. See the NodeJS application code that you downloaded f

The Event Detection sample application uses the Streaming Analytics service to a analyze a stream of weather information from NOAA weather stations around the world. A variety of simple and

Application Flow

The Event Detection sample application performs a series of steps using the Streaming Analytics service. The table below lists the tasks the application is performing and the status of each step.

Step	Task
1	Extract the environment information required to use the Streaming Analytics REST API.
2	Check if the Streams instance is running and start the instance if necessary using the Streaming Analytics REST API.
3	If the instance was already running, check if there is a Streams event detection job already running. If a job is running, cancel it.
4	Deploy a Streams Application Bundle to the Streaming Analytics service using the Streaming Analytics REST API. The bundle contains a Streams application the analyzes weather data and performs event detection.
5	Process events detected by the Streams application, and display them on this web page.
6	Cancel the job corresponding to the Streams application that was submitted in step 4 after events are processed.

Event Types

There are four types of events that the Streams application is monitoring for in the weather stream. Two of them are complex events, where patterns are recognized involving both the current d
events, which only require analyis of a weather station's current values.

M-Shape Temp	This event occurs when the graph of the temperature readings from a weathe illustrates the power of the Complex Event Processing toolkit in Streams, recognizing M-shape patterns in securities trading applications is very us trading world. See Partition and Compose: Parallel Complex Event Processin method, the double-top pattern and other patterns.
Steady Temp	This event occurs when the temperature reported by a weather station chang In addition, the net change during the 11 hours cannot exceed 3 degress.
Dry Heat	This event occurs when a weather station reports a temperature greater tha
Sauna	This event occurs when a weather station reports a temperature greater tha

Application Results

Results from the Streams application are listed below. Weather data from the prior 24 hours is available when the Streams application is started. The application analyzes this data so event patte
analyzed by the application, new data is processed in real-time as it is published by NOAA. Most weather stations update their readings only once per hour, so after the initial burst of events, new

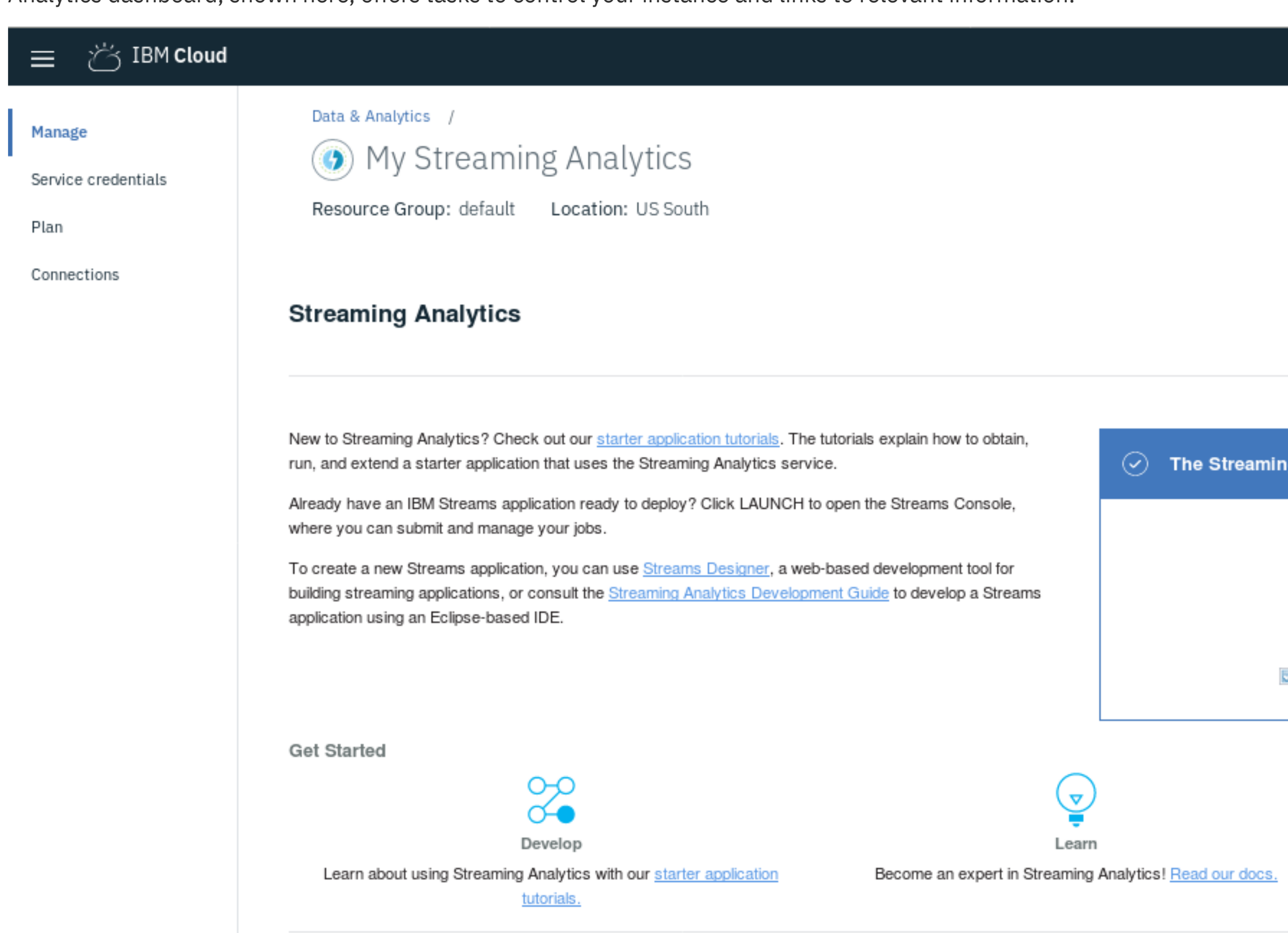
In addition to detecting events, the Streams application also determines the highest and lowest current temperature reading from any weather station.

Event Number	Event Time	Event Type	Weather Station
786	Mon Apr 16 15:41	Steady Temp	PAGM
785	Mon Apr 16 15:35	Steady Temp	KTMT
784	Mon Apr 16 15:40	Steady Temp	KKLS
783	Mon Apr 16 15:39	Steady Temp	KGEZ
782	Mon Apr 16 15:40	Steady Temp	KQEP
781	Mon Apr 16 15:35	Steady Temp	KGGP
780	Mon Apr 16 15:30	Steady Temp	YPCC
779	Mon Apr 16 15:35	Steady Temp	KETB
778	Mon Apr 16 15:35	Steady Temp	KDVO

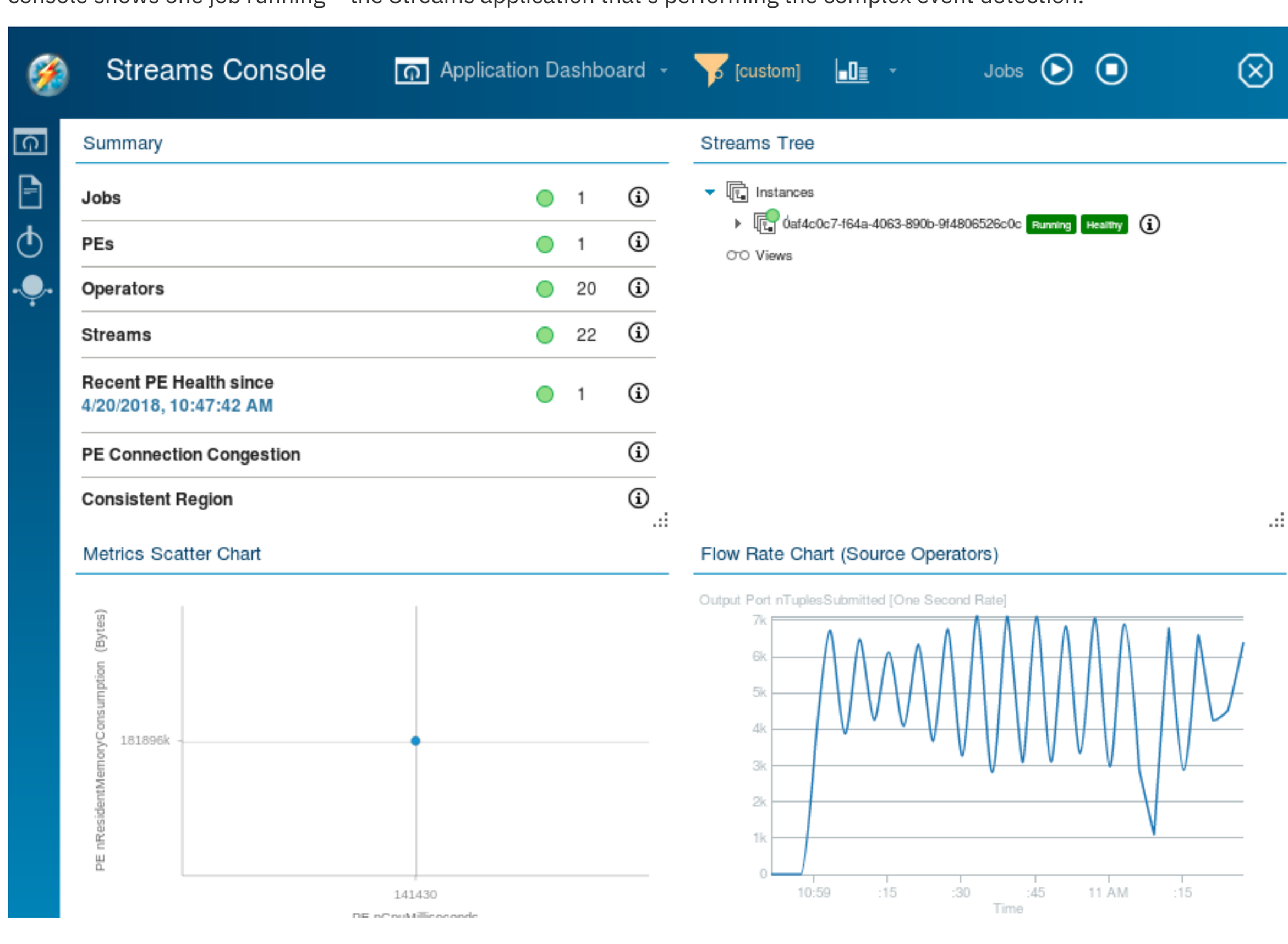
Ma
10

Step 3: Explore the Running Streams Application

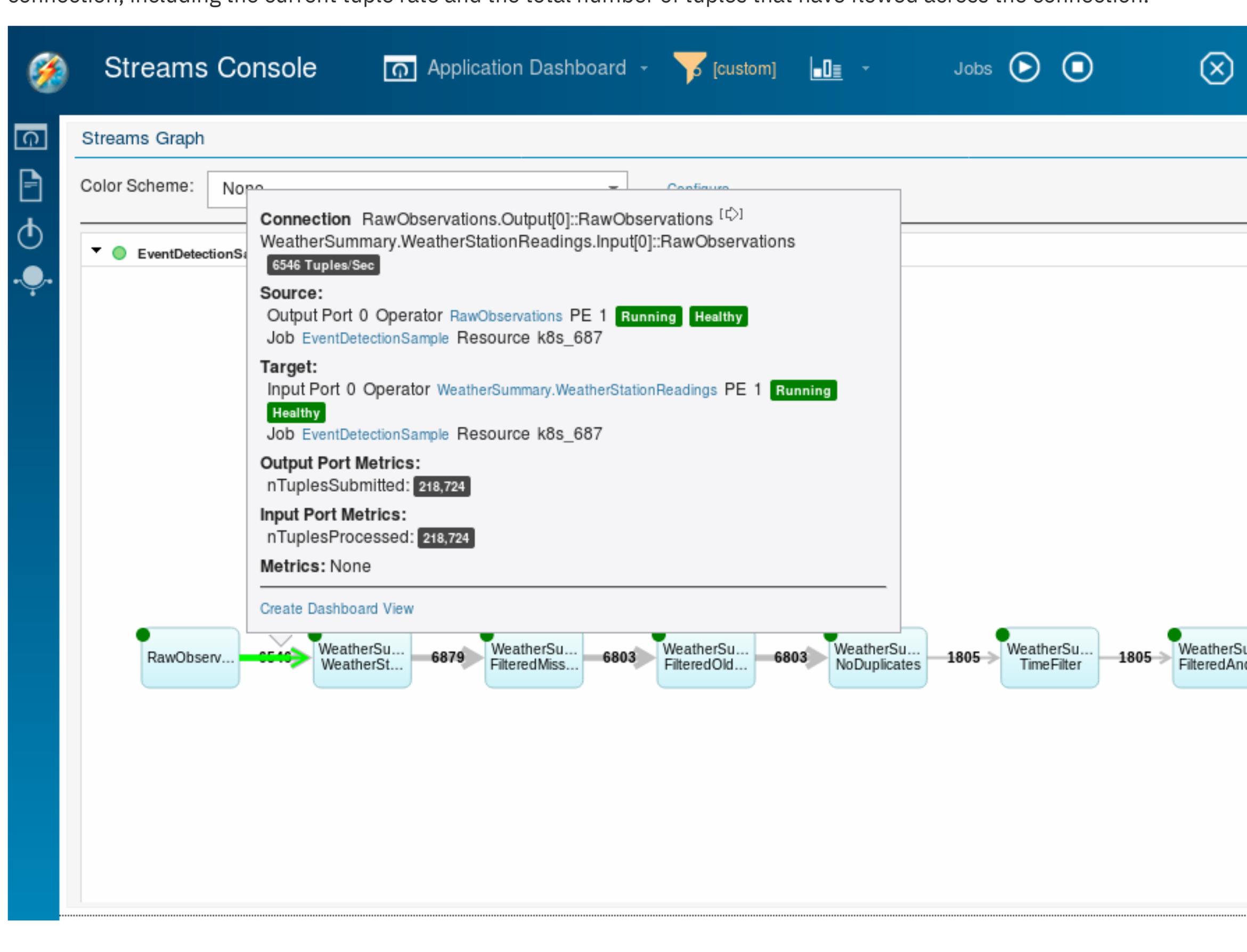
1. In the IBM Cloud web portal, bring up the service dashboard for your Streaming Analytics service. The Streaming Analytics dashboard, shown here, offers tasks to control your instance and links to relevant information:


IBM Cloud
Manage
Service credentials
Plan
Connections
Data & Analytics /
My Streaming Analytics
Resource Group: default Location: US South
Streaming Analytics
New to Streaming Analytics? Check out our starter application tutorials. The tutorials explain how to obtain, run, and extend a starter application that uses the Streaming Analytics service.
Already have an IBM Streams application ready to deploy? Click LAUNCH to open the Streams Console, where you can submit and manage your jobs.
To create a new Streams application, you can use Streams Designer, a web-based development tool for building streaming applications, or consult the Streaming Analytics Development Guide to develop a Streams application using an Eclipse-based IDE.
The Streamin
Get Started
Develop
Learn about using Streaming Analytics with our starter application tutorials.
Learn
Become an expert in Streaming Analytics! Read our docs.

2. Click the **LAUNCH** button on the dashboard to display the Streaming Analytics console. In the image below, the console shows one job running – the Streams application that's performing the complex event detection:


Streams Console
Application Dashboard
[custom]
Jobs
Summary
Jobs 1
PEs 1
Operators 20
Streams 22
Recent PE Health since 4/20/2018, 10:47:42 AM 1
PE Connection Congestion
Consistent Region
Streams Tree
Instances
0af4c0c7-f64a-4063-890b-9f4806526c0c Running Healthy
Views
Metrics Scatter Chart
PE nResidentMemoryConsumption (Bytes)
181896k
141430
Flow Rate Chart (Source Operators)
Output Port nTuplesSubmitted [One Second Rate]
7k
6k
5k
4k
3k
2k
1k
0
10:59
:15
:30
:45
11 AM
:15
Time

3. Maximize the Streams Graph pane in the upper right of the console to show the flow graph of the Streams application. The graph shows the live status of your Streams application as it runs. You can hover over the operators in the graph or the connections between them to get more-detailed information. In the following screenshot, hovering over a connection between the first two operators in the graph displays information about the status of that connection, including the current tuple rate and the total number of tuples that have flowed across the connection.


Streams Console
Application Dashboard
[custom]
Jobs
Streams Graph
Color Scheme: None
EventDetectionSa
Connection RawObservations.Output[0]::RawObservations
WeatherSummary.WeatherStationReadings.Input[0]::RawObservations
6546 Tuples/Sec
Source:
Output Port 0 Operator RawObservations PE 1 Running Healthy
Job EventDetectionSample Resource k8s_687
Target:
Input Port 0 Operator WeatherSummary.WeatherStationReadings PE 1 Running
Healthy
Job EventDetectionSample Resource k8s_687
Output Port Metrics:
nTuplesSubmitted: 218,724
Input Port Metrics:
nTuplesProcessed: 218,724
Metrics: None
Create Dashboard View
RawObserv...
WeatherSu... WeatherSt...
6879
WeatherSu... FilteredMiss...
6803
WeatherSu... FilteredOld...
6803
WeatherSu... NoDuplicates
1805
WeatherSu... TimeFilter
1805
WeatherSu FilteredAn

Please Note: This Streams application can show tuple rates of zero for long periods of time. As mentioned earlier, the weather data stream is "bursty". There is a lot of data to process when the app is started, but after that new data becomes available infrequently, so no data is processed in some time periods. To consistently see nonzero tuple rates in the graph, restart your Node.js app and switch over to the Streams graph immediately. Once the job restarts, you'll see nonzero tuple rates for at least a few minutes as the Streams application processes the initial burst of data.

Step 4: Review the Node.js Code

The EventDetection app is a complete yet simple application that requires no customization to run. To understand the app, examine its code:

1. Open the app.js file to view the application logic. The code in app.js is organized around six major steps:
   - **Step 1** – Extract the environment information required to use the Streaming Analytics REST API.
   - **Step 2** – Check if the Streams instance is running and start the instance if necessary via the Streaming Analytics REST API.
   - **Step 3** – If the instance was already running, check if a Streams event-detection job is already running. If a job is running, cancel it.
   - **Step 4** – Deploy a Streams Application Bundle to the Streaming Analytics service by using the Streaming Analytics REST API. The bundle contains a Streams application that analyzes weather data and performs event detection.
   - **Step 5** – Process events detected by the Streams application, and display them on this web page.
   - **Step 6** – Cancel the job corresponding to the Streams application after 1,500 events are processed.
2. Skim the code to identify the where these steps above are performed.
   - Some of the steps involve calling the Streaming Analytics REST API, making this sample a good example of how to use the API from Node.js.
   - Step 5 is implemented as a POST handler. The Streams application sends event messages to Node.js using an HTTPPost operator, and the POST handler in the Node.js app parses them and updates the web UI accordingly.

Step 5: Review the Streams Application Code

The Streams application used is a complete Streams application that requires no customization to run. The source code that you downloaded (or cloned or forked) contains the application's source code as well as its prebuilt .sab file. To understand the Streams application, examine its code. Below, we look at the details behind two key pieces of the Streams app:

1. Open the EventDetection.spl file (located in the project's spl subdirectory). The source code for the application is written in SPL, a language oriented to data streams and operators that act upon them.
2. Skim the code to locate the operator declarations and compare them to the flow graph that you saw in the Streaming Analytics console. You'll take a more detailed look at a couple of the operators in the remainder of this section.
3. Examine the code for one of the operators that detects a complex event. The code snippet below shows an operator called MatchRegex, which is used to detect patterns on a series of data tuples in a stream. The code comments describe the nature of the M-shape pattern that the operator will detect:

```
//
// The first complex event is called "M-shape".  It triggers when the graph of the temperature for a
// weather station form's an M shape over a period of time.
//
// Detecting M shape patterns in weather data is not that useful, but recognizing an M shape in financial
// trading is valuable and is referred to as a "double-top" stock pattern.
//
// See http://hirzels.com/martin/papers/debs12-cep.pdf for more information on this complex event
// detection method, the double-top pattern and other patterns.
//
stream<WeatherSummary weatherValues, rstring event> TempMEvent = MatchRegex(WeatherSummary)
{
  param
    pattern     : ". rise+ drop+ rise+ drop* deep";
    partitionBy : stationCode;
    predicates  : {
      rise = tempInF>First(tempInF)  && tempInF>=Last(tempInF),
      drop = tempInF>=First(tempInF) && tempInF<Last(tempInF),
      deep = tempInF<First(tempInF)   && tempInF<Last(tempInF) };
  output
    TempMEvent : weatherValues=WeatherSummary, event="M-Shape Temp";
}
```

   The declaration of the operator defines the pattern that you're trying to detect by using regular-expression syntax with a set of predicates that are also defined in the operator declaration. The operator looks for the M-shape of the temperature at a weather station based upon the set of values that have been reported by that weather station. This MatchRegex operator consumes the WeatherSummary stream defined earlier in the SPL code and produces a stream called TempMEvent. The operator partitions the weather station's readings into separate groupings, by the weather station's ID, and maintains the state necessary to detect the event for each weather station.

4. Next, examine a sequence of two operators used to send events back to the Node.js app:

```
//
// Send events to the the application user interface by converting them to json and HTTPPost-ing
// to the Node.js app
//
stream <rstring jsonString> JSONOutput = com.ibm.streamsx.json::TupleToJSON(OutputEvents)
{
}

() as HttpEvents = HTTPPost(JSONOutput) {
  param
    headerContentType : "application/json";
    url : ((rstring) getSubmissionTimeValue("route"));
}

//
// Print events to stdout so they can be viewed in a log for debug purposes.
//
() as HttpEventsDbg = Custom(JSONOutput) {
  logic
    onTuple JSONOutput: {
      println(JSONOutput.jsonString);
    }
    onPunct JSONOutput: {
      println(currentPunct());
    }
}
```

   The first operator in the preceding snippet converts a tuple in a stream into a JSON string. This operator consumes a stream called OutputEvents defined earlier in the SPL code and produces a stream called JSONOutput. The next operator, called HTTPPost, consumes the JSONOutput stream and sends the JSON string to the route for the Node.js app via an HTTP POST.

Step 6: Customize or Extend the Sample

Now that you're familiar with the starter app, you can modify the application's source code to customize it or extend it in any of several interesting ways:

- To make the app run longer, modify the Node.js code in app.js to change the event_target variable's value from 1500 to a higher number.
- To define a new complex event for the Streams application to detect:
  1. Update the SPL code to add another MatchRegex operator to the flow to detect a pattern that you want to look for.
  2. Update the operators after your new MatchRegex operator in the SPL code so that your new event type gets sent back to the Node.js application.

To modify the app:

1. Plan your modifications.
2. Change the Node.js and/or SPL source code to reflect your desired customizations.
3. If you have modified the SPL code, you must recompile it in a Streams development environment and replace the .sab file that you downloaded with this updated version. To learn how to develop and compile a Streams app, see the Streaming Analytics Development Guide
4. Deploy (push) the modified Node.js application to the IBM Cloud.

Conclusion

Complex event detection against a real-time data stream is possible using the Streaming Analytics service in the IBM Cloud. The application that you worked through in this tutorial will get you started. You can change to the data streams you want to analyze, define the events you want to detect, and act on those events to accomplish your goals.

TAGS FEATURED, SAMPLES, CLOUD

by MikeBranson