[Overview 2](#_Toc382232924)

[Pre-requisites 2](#_Toc382232925)

[Configuration 2](#_Toc382232926)

[Algorithm 3](#_Toc382232927)

[Design 3](#_Toc382232928)

[DAG 3](#_Toc382232929)

[Operators 3](#_Toc382232930)

[YahooFinanceCSVInputOperator 3](#_Toc382232931)

[DerbySqlStreamOperator 4](#_Toc382232932)

[AlertEscalationOperator 4](#_Toc382232933)

[SmtpOutputOperator 4](#_Toc382232934)

[Functionality 4](#_Toc382232935)

[Launching the application 4](#_Toc382232936)

[Monitoring the application 5](#_Toc382232937)

[Operations 11](#_Toc382232938)

[Stopping the application 15](#_Toc382232939)

[Conclusions 16](#_Toc382232940)

Application ‘Yahoo Finance with Alert’

# Overview

The ‘Yahoo finance with Alerts’ demo application is one of the simplest applications to understand massive computational ability of the DataTorrent platform. This application gets Yahoo finance feed with different parameters for a given set of tickers. Based on the value of a parameter, an alert can be generated. The frequency of the alerts can also be controlled.

The application samples yahoo finance ticker every 500ms. All data points are streamed into Derby SQL. While a user is able to get the streaming quotes from ‘Yahoo finance’, he needs to perform further analysis based on these quotes. This is what helps the user make trading decisions.

In order that the user can make decisions in real time, this application demonstrates the ability of the DataTorrent platform to perform massive real time computations from various pieces of information which can be consolidated and then processed. This application checks for the last traded price for a ticker symbol and generates an alert if the price drops below a certain threshold.

An SQL query is run on the inputs to compare the last traded price with the threshold. All the above is done in real time.

# Pre-requisites

* Derby SQL
* SMTP

# Configuration

* If the user wants to select the ticker symbols for which the alerts are to be raised, it can be done by modifying the dt-site.xml file as below –

<!-- Yahoo Finance application with Alert generation demo -->

<property>

<name>com.datatorrent.demos.yahoofinance.ApplicationWithAlert.tickerSymbols</name>

<value>*<desired ticker symbols>*</value>

</property>

<property>

<name>com.datatorrent.demos.yahoofinance.ApplicationWithAlert.tickerThreshold</name>

<value>*<desired threshold values>*</value>

</property>

Note : The threshold values should be same in number and should be in the same order as the ticker symbols.

* Please ensure that an SMTP server is appropriately setup.

As per the configured values for your SMTP server, please set the following in the configuration file –

1. Please open DataTorrent configuration file ~/.dt/dt-site.xml in any editor.
2. Set the values for the following in the dt-site.xml file as needed -

<property>

<name>com.datatorrent.demos.yahoofinance.ApplicationWithAlert.From</name>

<value>*<Sender’s email address>* </value>

</property>

<property>

<name>com.datatorrent.demos.yahoofinance.ApplicationWithAlert.To</name>

<value><*Recipient’s email address*></value>

</property>

<property>

<name>com.datatorrent.demos.yahoofinance.ApplicationWithAlert.smtpHost</name>

<value>*<smtp host>*</value>

</property>

<property>

<name>com.datatorrent.demos.yahoofinance.ApplicationWithAlert.smtpPort</name>

<value>*<smtp port>*</value>

</property>

<property>

<name>com.datatorrent.demos.yahoofinance.ApplicationWithAlert.smtpUser</name>

<value>*<smtp Username>*</value>

</property>

<property>

<name>com.datatorrent.demos.yahoofinance.ApplicationWithAlert.smtpPasswd</name>

<value>*<smtp password>*</value>

</property>

You can optionally set the ‘useSsl parameter as well.

The ‘alertInterval’ is set to 1 minute by default. This means that an alert would not be generated within one minute of a previous alert for a given ticker symbol. This can also be overridden by setting it in the dt-site.xml file as –

<property>

<name>com.datatorrent.demos.yahoofinance.ApplicationWithAlert.alertInterval</name>

<value>*<desired value for alert Interval in milliseconds>*</value>

</property>

Note : **The smtp parameters are a must to be set in case the user wants to receive alert over emails. If not set, the application will display the alerts on the console.**

# Algorithm

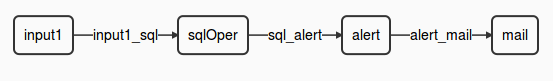
The ‘Yahoo finance with Alert’ application written for DataTorrent platform uses this algorithm:

1. Get the last traded price for the specified tickers
2. Compare the last traded price with a threshold
3. If the last traded price is less than the threshold, initiate action to generate alert
4. Check if an alert was raised for this exact condition within the past 1 minute.
5. If not, raise an email alert.

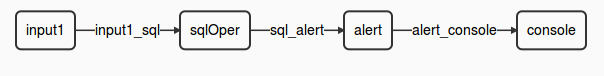
# Design

The DAG for the application is shown below followed by a description of the operators.

## DAG



In case the smtp parameters are not specified, the dag will look as follows –



This view can be seen in the ‘logicalDAG’ widget in the console.

## Operators

The ‘Yahoo finance with Alert’ application involves multiple computational steps and the DataTorrent platform provides operators to perform each of these. Specifically, this application consists of the following operators –

|  |  |  |
| --- | --- | --- |
| **Type of Operator** | **Name of the operator in DAG** | **Library** |
| [YahooFinanceCSVInputOperator](#YahooFinanceCSVInputOperator) | input1 | Custom operator |
| [DerbySqlStreamOperator](#DerbySqlStreamOperator) | sqlOper | lib/math/SumKeyVal |
| Yahoofinance[AlertEscalationOperator](#AlertEscalationOperator) | alertOPer | Custom operator |
| [SmtpOutputOperator](#SmtpOutputOperator) | mailOper | lib/math/RangeKeyVal |
| [ConsoleOutputOperator](#ConsoleOutputOperator) | consoleOperator | lib/io/ConsoleOutputOperator |

### [YahooFinanceCSVInputOperator](https://github.com/DataTorrent/Malhar/demos/src/main/java/com/datatorrent/demos/yahoofinance/YahooFinanceCSVInputOperator.java)

This operator gets data from Yahoo finance and emits it in the form of a hashmap. It prepares the URL for the specified parameters and uses the same to get the data from Yahoo finance. The instance ‘input1’ of this operator gets data for a ‘symbol’(***s0***) (for example - YHOO, GOOG, FB, etc) and its ‘Last Traded Price’(***l1***). This information is sent to the ‘sqlOper’ operator for further processing.

### [DerbySqlStreamOperator](https://github.com/DataTorrent/Malhar/library/src/main/java/com/datatorrent/lib/streamquery/DerbySqlStreamOperator.java)

This is an embedded Derby SQL input operator. This accepts inputs from the YahooFinanceCSVInputOperator instances. This operator creates a database schema and creates a table – for the input that it receives. This table is populated with data coming in from the YahooFinanceCSVInputOperator. This operator then runs a query to find if the last traded price for the share of a particular specified ticker is below the specified threshold. For example if the last traded price of “AAPL” is < 450 USD. In such a case, it will signal the AlertEscalationOperator to raise an alert.

### YahoofinanceAlertEscalationOperator

This is a custom operator which extends the AlertEscalationOperator. This operator checks if an alert has been raised in the past one minute for the specified ticker symbol. If the condition is the same and the alert has been raised within the specified alertInterval, it will not raise another alert till a minute (i.e. the alertInterval – as configured. Default is one minute) has passed since the last alert. This operator thus, decides whether it is the right time to raise an alert or not. If it is, it emits the data to the SmtpOutputOperator to generate an Smtp alert.

### [SmtpOutputOperator](https://github.com/DataTorrent/Malhar/library/src/main/java/com/datatorrent/lib/io/SmtpOutputOperator.java)

This operator is used to generate an email notification in case of the average dropping below the threshold.

[ConsoleOutputOperator](https://github.com/DataTorrent/Malhar/library/src/main/java/com/datatorrent/lib/io/ConsoleOutputOperator.java):

This operator comes into the picture when smtp is not configured. The alterts will be displayed on the console in such a case.

#### Streams

|  |  |
| --- | --- |
| **Stream** | **Description** |
| input1\_sql | Carries a hashmap with - symbol and last traded price for the symbol read |
| sql\_alert | Carries information needed for the alert to be generated |
| alert\_mail | Carries alert information for generating an smtp alert |
| alert\_console | Carries alert information for to be displayed on the console |

# Functionality

The detailed functionality of each operator is covered above. Note that in this application, we do use operators which are statefull as well as some which are stateless.

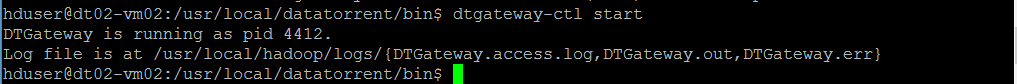
|  |  |  |
| --- | --- | --- |
| **Type of Operator** | **Statefull** | **Partitionable** |
| [YahooFinanceCSVInputOperator](#YahooFinanceCSVInputOperator) | N | Y |
| [DerbySqlStreamOperator](#DerbySqlStreamOperator) | N | Y |
| [AlertEscalationOperator](#AlertEscalationOperator) | N | Y |
| [SmtpOutputOperator](#SmtpOutputOperator) | N | Y |

# Launching the application

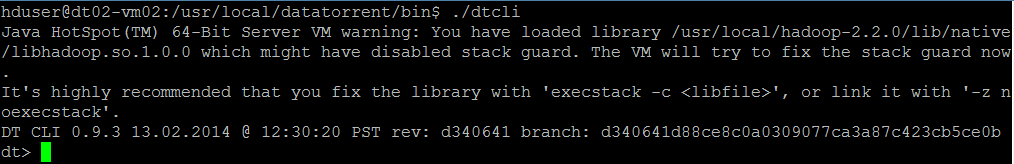
In order to be able to launch sample Demo Applications, please make sure you have completed following steps.

In case you have not started ‘dtGateway’, start the process as follows ...

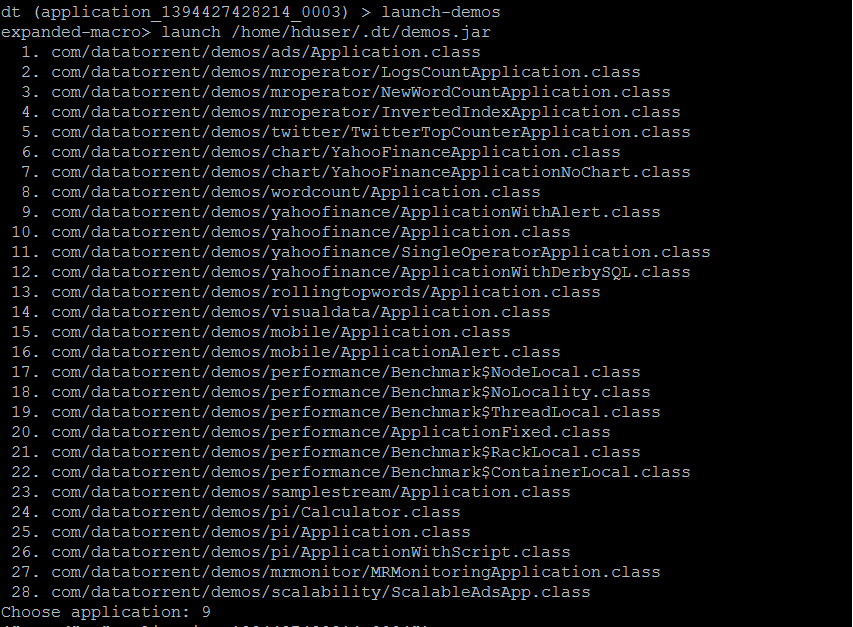
1. Start the DataTorrent Gateway process. This can be done from the command line as –



1. Launch DataTorrent Command Line Interface (dtcli)



1. Launch demo application Jar. (Here, we assume that the Hadoop services are up and running and the demos are being launched in a cluster mode.)



The user can now select and launch the application.

The jar is in ~/.dt/demos.jar and can also be launched as "launch ~/.dt/demos.jar

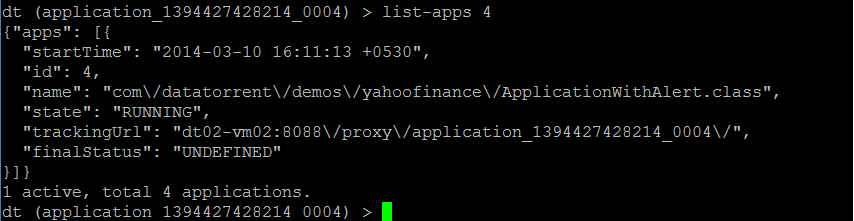
Please refer to DataTorrent Quick Start Guide for additional details.

# Monitoring the application

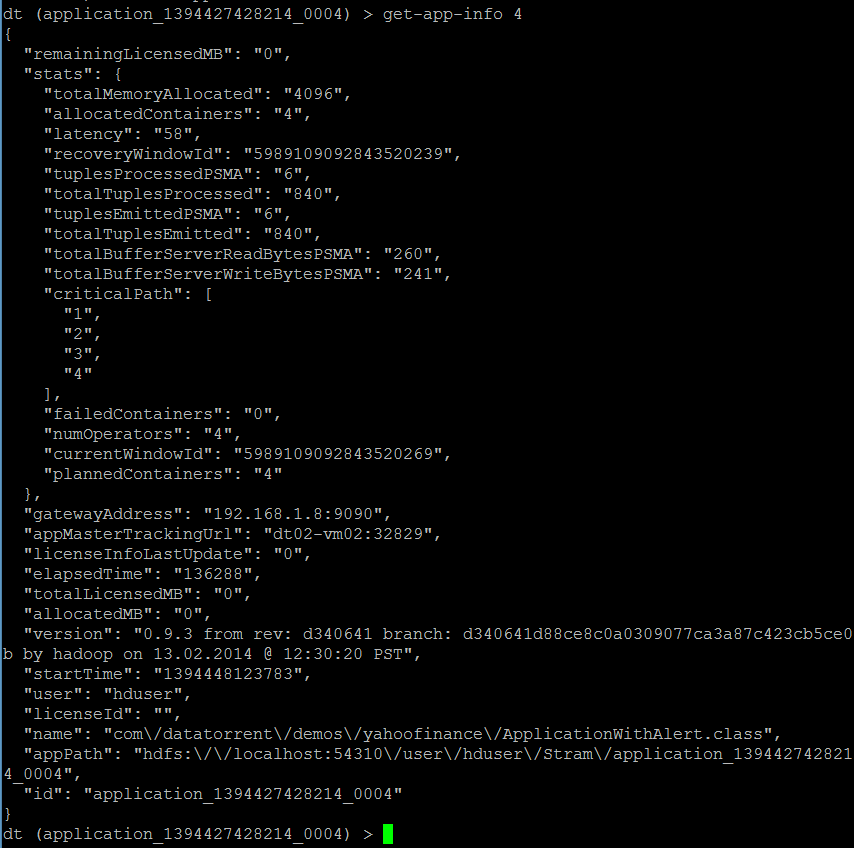
Once launched, the application can be monitored/managed from either the command line viz. ‘dtcli’ or the GUI viz. DataTorrent Gateway Console. The console and cli get data from same webservices layer. Following are some of the sample operations that are typically performed while monitoring the application –

1. List the application -

#### Using ‘dtcli’



1. Get detailed information about a given application –



1. Connect to the application –

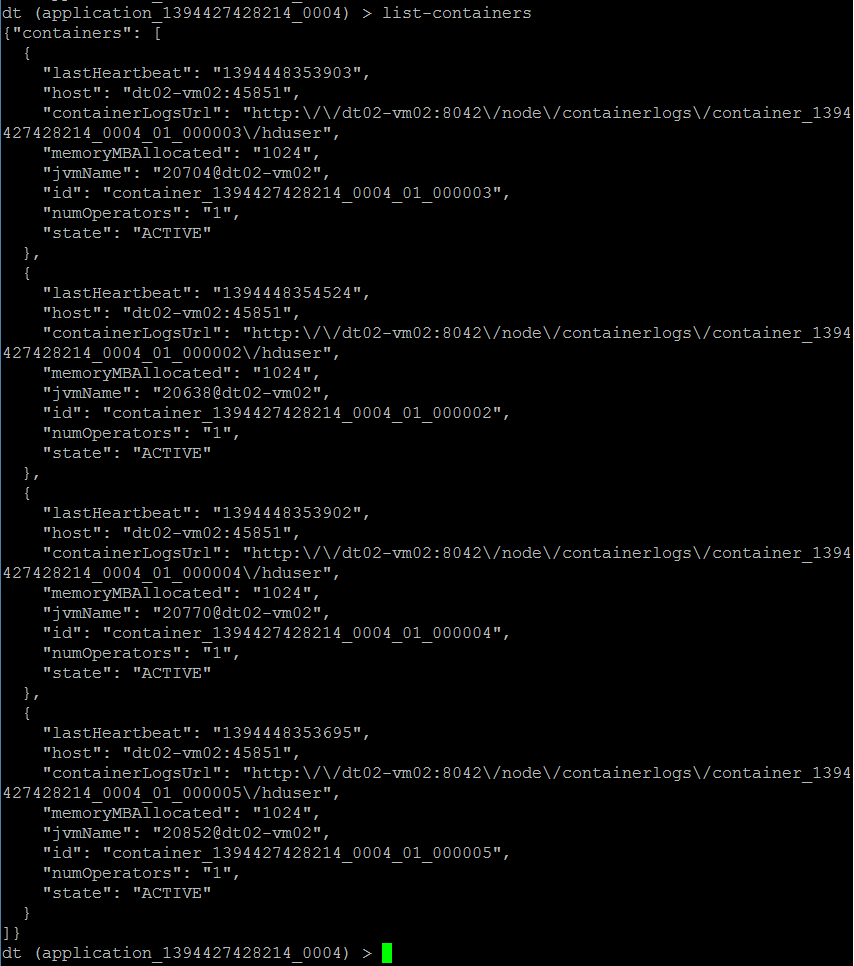


1. Show logical plan



1. List the containers :

This lists the containers for the application that you are connected to, at this point in time.

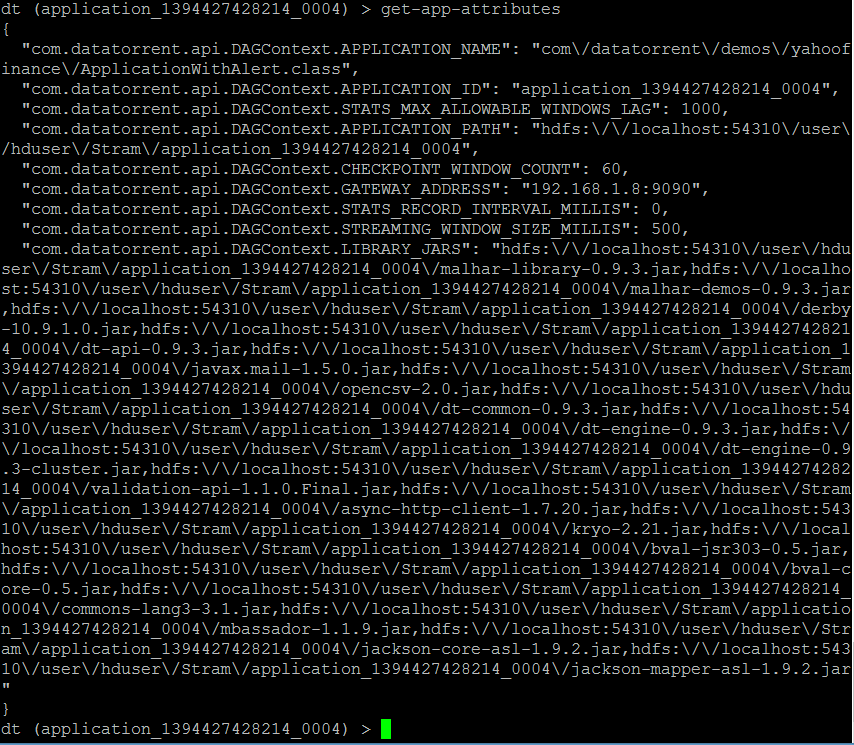


1. List all the operators –

(Note that the screenshot below captures only the first 2. Others are displayed in the same manned in order)



1. Get application attributes :

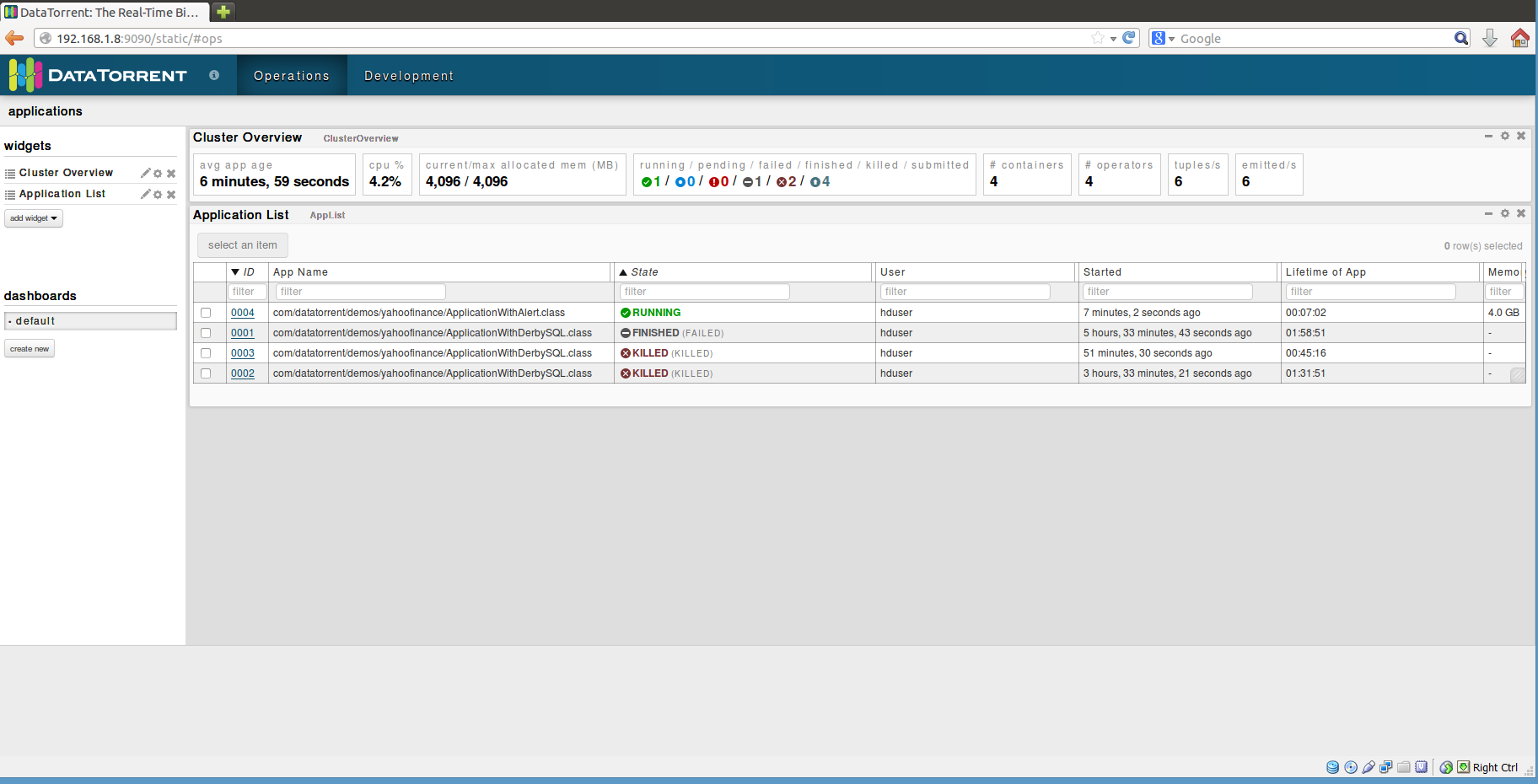


1. Similarly the user can use other dtcli to monitor/manage the application.

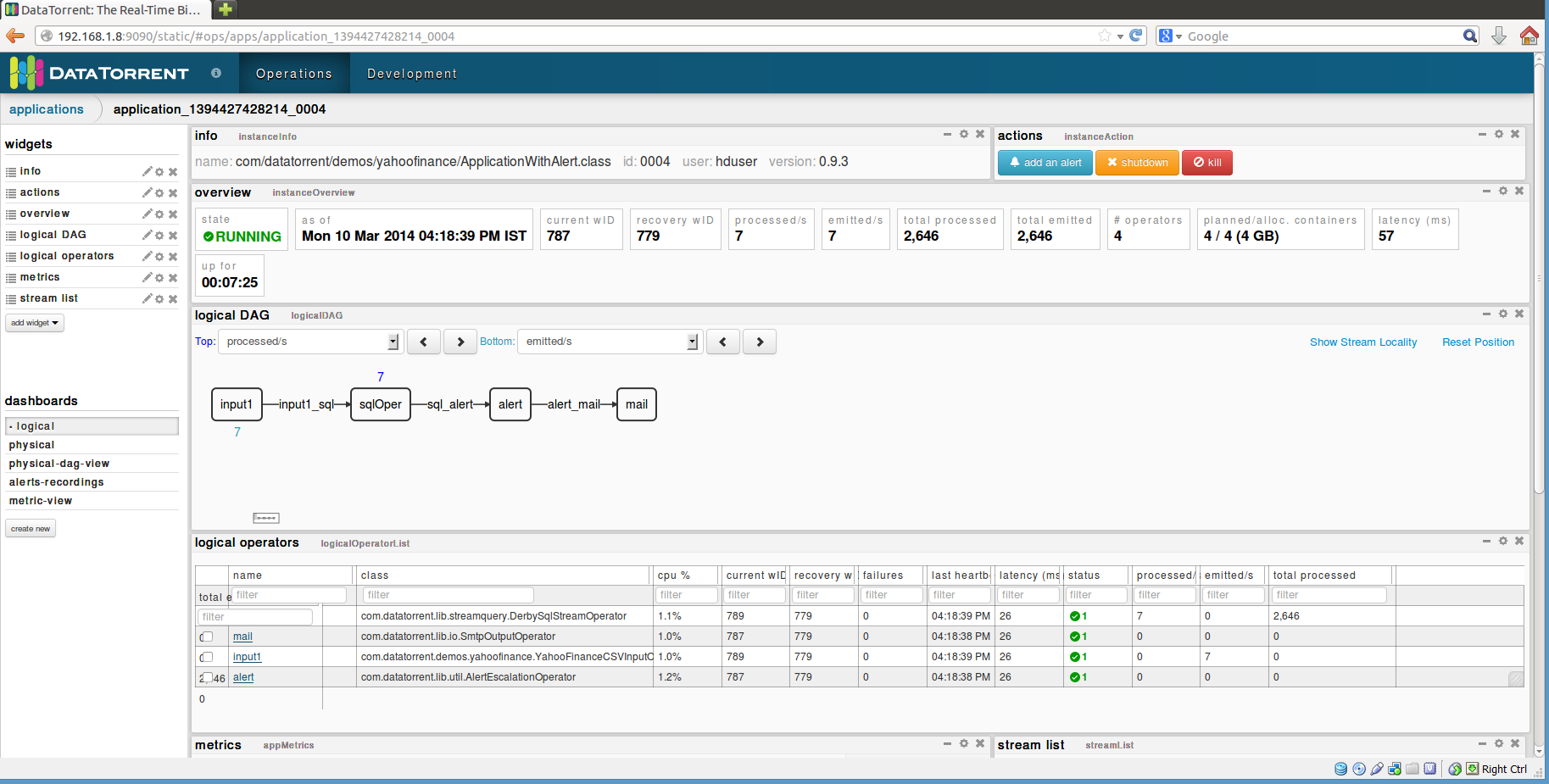
#### Using ‘Gateway Console’

# Operations

1. On the Gateway console, you will see the apps as below –



1. In order to get detailed about the application, click on ‘ID’ for the application and you will get to the screen below –

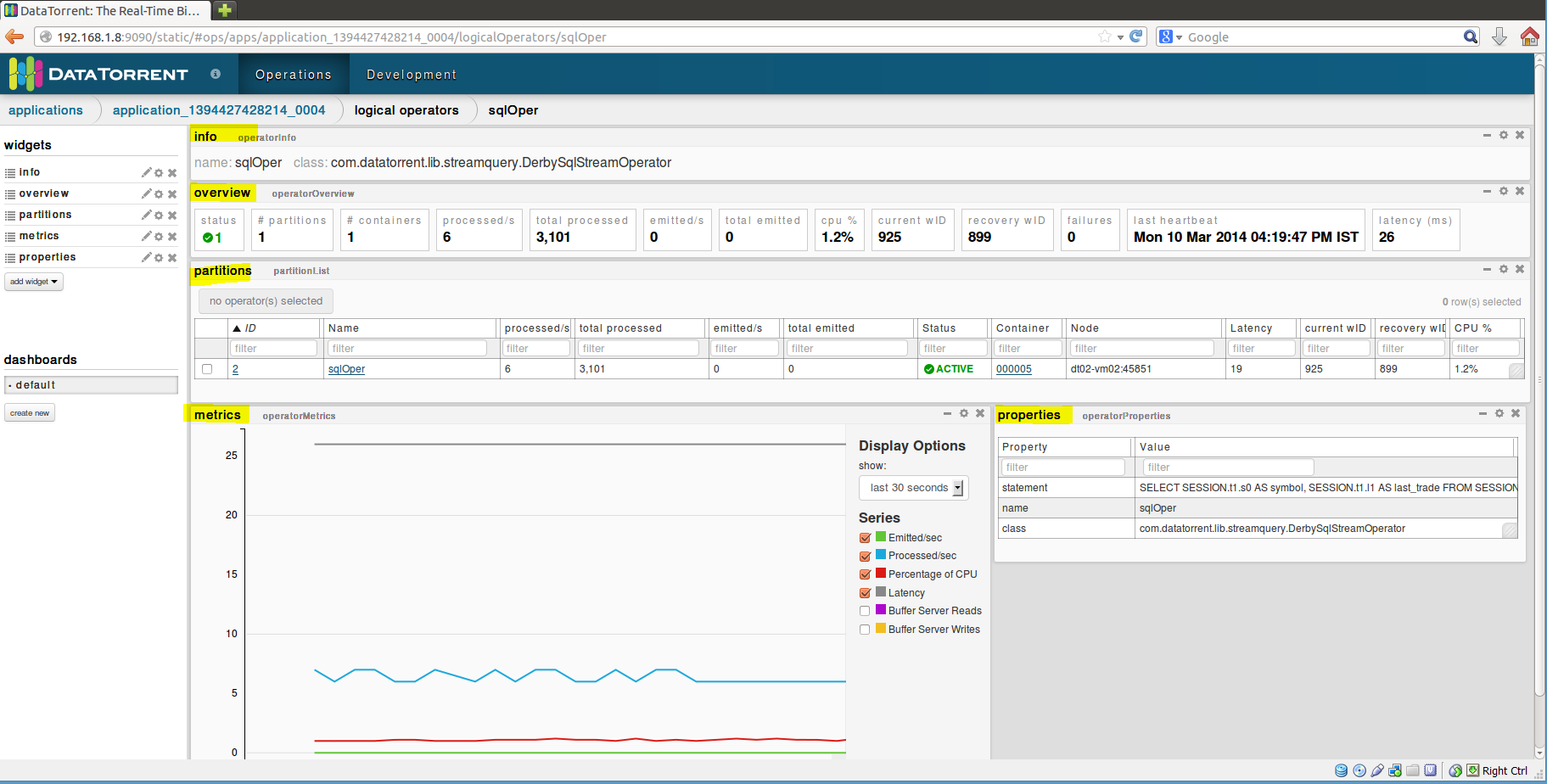


As you can see, it shows you a lot of details about the application, such as –

* Overview
* Logical DAG
* Logical operators
* Streams (not captured in the screenshot, but can be seen whe you scroll downwards)
* Various mertices, etc.

Additionally, you can select additional widgets as well as rearrange the widgets as per your wish.

1. Clicking on any of the operators will give you the details for that particular operator. For example, the screenshot below is for the ‘StockTickInput’ operator.

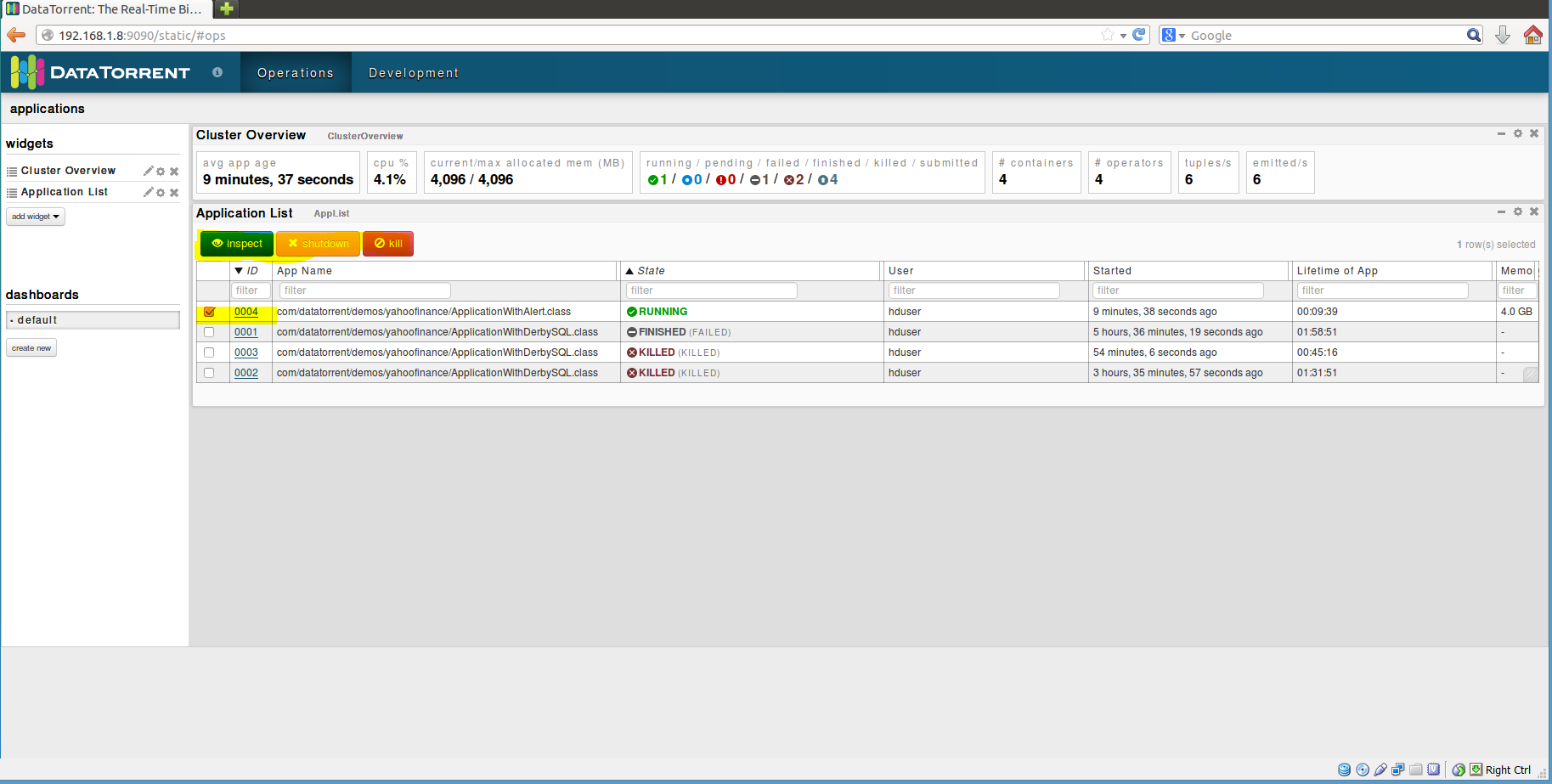


You can see detailed information about –

* partitions
* Metrices
* Properties

You can add more widgets here as well

1. Selecting an application from the list of applications also enables the operations to control the application viz. inspect, shutdown, kill. The user can click on any of the buttons to perform the corresponding operation.

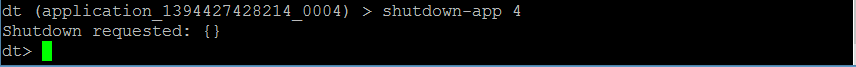


# Stopping the application

From dtcli

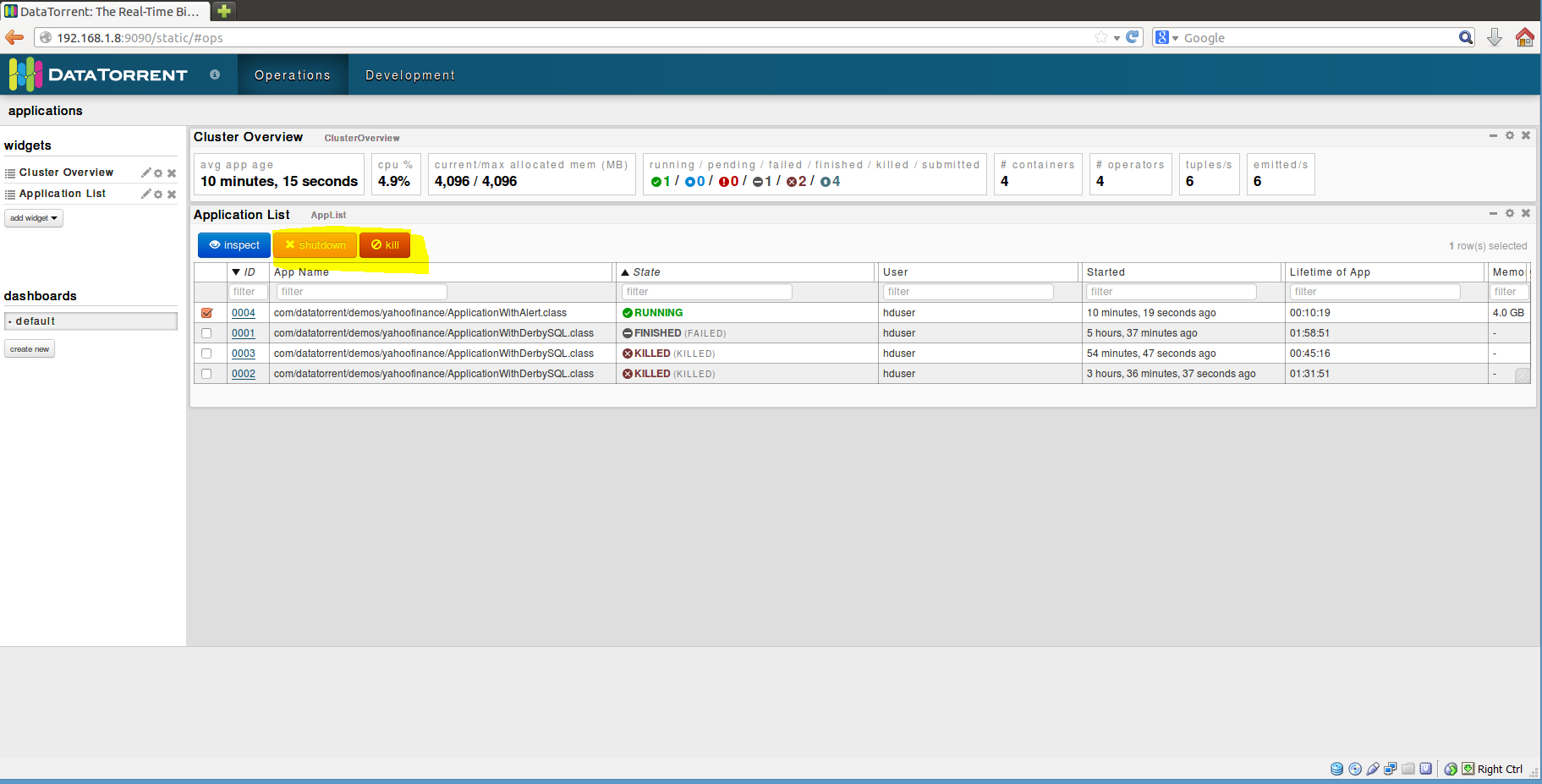
The user can use the ‘shutdown-app’ command to shut down the application from the dtcli.

Since this is a platform for streaming applications, in most of the cases, the application would not be shutting down The user will have to kill the application. In order to achieve this, a kill-app command is provided which can be used.



From DT Gateway Console

An application can be either shutdown or killed from the DT Gateway console. You can do so when you select an application from the list of applications. Once the user does this, the ‘shutdown’ and ‘kill buttons get activated along with the ‘inspect’ button. The user can then shutdown or kill the application using these buttons.



# Conclusions

This application demonstrates the ability of the DataTorrent platform to generate alerts based on various conditions and also control the frequency of the alerts.

The alerts are displayed on the console as below –

