[Overview 2](#_Toc382326426)

[Pre-requisites 2](#_Toc382326427)

[Configuration 2](#_Toc382326428)

[Algorithm 2](#_Toc382326429)

[Design 3](#_Toc382326430)

[DAG 3](#_Toc382326431)

[Operators 3](#_Toc382326432)

[YahooFinanceCSVInputOperator 3](#_Toc382326433)

[DerbySqlStreamOperator 4](#_Toc382326434)

[ConsoleOutputOperator 4](#_Toc382326435)

[Functionality 5](#_Toc382326436)

[Launching the application 5](#_Toc382326437)

[Monitoring the application 6](#_Toc382326438)

[Operations 11](#_Toc382326439)

[Stopping the application 14](#_Toc382326440)

[Conclusions 15](#_Toc382326441)

Application ‘Yahoo Finance with DerbySQL’

# Overview

The ‘Yahoo finance with Derby SQL’ 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 which are then combined to get the PE\_Ratio and PB\_Ratio for that symbol.

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 analyzes and provides the user with the useful ratios based on the following information received in real time -

* Last Traded Price
* Earnings per Share
* Book Value

An SQL query is run on the inputs to calculate the ratios. All the above is done for each ticker symbol and in real time.

# Pre-requisites

* Derby SQL

# Configuration

There is no particular configuration that needs to be done to run the ‘Yahoo finance with Derby SQL’ demo application on the DataTorrent platform. If the user wants to select the ticker symbols for which the ratios are to be calculated, it can be done by modifying the dt-site.xml file as below –

<!-- Yahoo Finance application with Derby SQL -->

<property>

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

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

</property>

# Algorithm

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

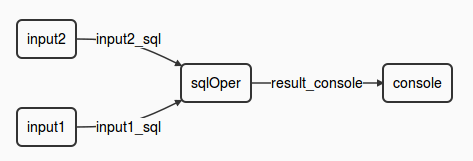
1. Get last traded price from Yahoo finance for specified ticker symbols.
2. Get earnings per share and book value from Yahoo finance for specified ticker symbols.
3. Consolidate information coming from independent sources into a database
4. Extract relevant information needed to calculate the ratios
5. Calculate ratios and display it on the console.

# Design

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

## DAG

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



## Operators

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

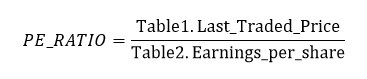
|  |  |  |
| --- | --- | --- |
| **Type of Operator** | **Name of the operator in DAG** | **Library** |
| [YahooFinanceCSVInputOperator](#YahooFinanceCSVInputOperator) | input1Oper | Custom operator |
| [YahooFinanceCSVInputOperator](#YahooFinanceCSVInputOperator) | Input2Oper | Custom operator |
| [DerbySqlStreamOperator](#DerbySqlStreamOperator) | sqlOper | lib/math/SumKeyVal |
| [ConsoleOutputOperator](#ConsoleOutputOperator) | consoleOperator | lib/io/ConsoleOutputOperator |

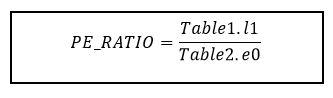
### [YahooFinanceCSVInputOperator](https://github.com/DataTorrent/Malhar/demos/src/main/java/com/datatorrent/demos/yahoofinance/YahooFinanceCSVInputOperator.java)

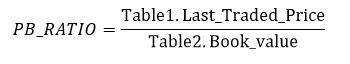
This operator gets data from Yahoo finance and emits it in the form of a hashmap. It prepares the URL for the specified parameters and uses the same to get the data from Yahoo finance. There are 2 instances of this operator viz. ‘input1Operator’ and input2Operator’. One of the instances gets data for a ‘symbol’(***s0***) (for example - YHOO, GOOG, FB, etc) and its ‘Last Traded Price’(***l1***). The other instance captures information about the ‘symbol’(***s0***), ‘Earnings / share’(***e0***) and the ‘Book Value’(***b4)***. Each of these instances sends this information to the ‘sqlOper’ operator for further processing.

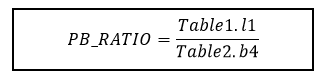
### [DerbySqlStreamOperator](https://github.com/DataTorrent/Malhar/library/src/main/java/com/datatorrent/lib/streamquery/DerbySqlStreamOperator.java)

This is an embedded Derby SQL input operator. This accepts inputs from the YahooFinanceCSVInputOperator instances. This operator created a database schema and creates tables – one for each input that it receives. Each of these tables is populated with data coming in from the two inputs. This operator then runs a query to calculate the PE\_RATIO and PB\_RATIO for a given symbol. These are calculated as below –









Note : Both the above calculations are done for a given symbol.

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

The various ratios calculated by the DerbySQLStream operator are displayed on the console by the console output operator.

#### Streams

|  |  |
| --- | --- |
| **Stream** | **Description** |
| input1\_sql | Carries a hashmap with - symbol and last traded price for the symbol read |
| Input2\_sql | Carries a hashmap with – symbol, earnings per share and book value for the symbol read |
| result\_console | Carries symbol, PE\_Ratio and PB\_Ratio to the console for displaying |

# Functionality

The detailed functionality of each operator is covered above.

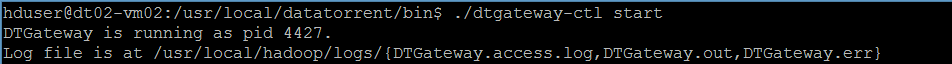
|  |  |  |
| --- | --- | --- |
| **Type of Operator** | **Statefull** | **Partitionable** |
| [YahooFinanceCSVInputOperator](#YahooFinanceCSVInputOperator) | N | Y |
| [DerbySqlStreamOperator](#DerbySqlStreamOperator) | N | Y |
| ConsoleOutputOperator | N | Y |

# Launching the application

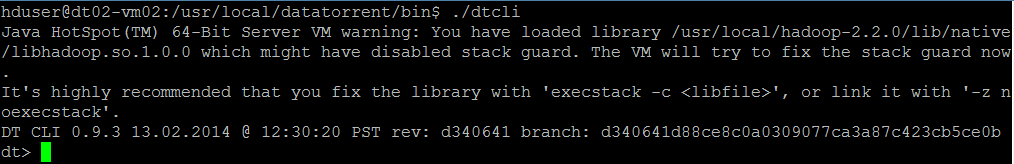
In order to be able to launch sample Demo Applications, please make sure you have completed following steps.

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

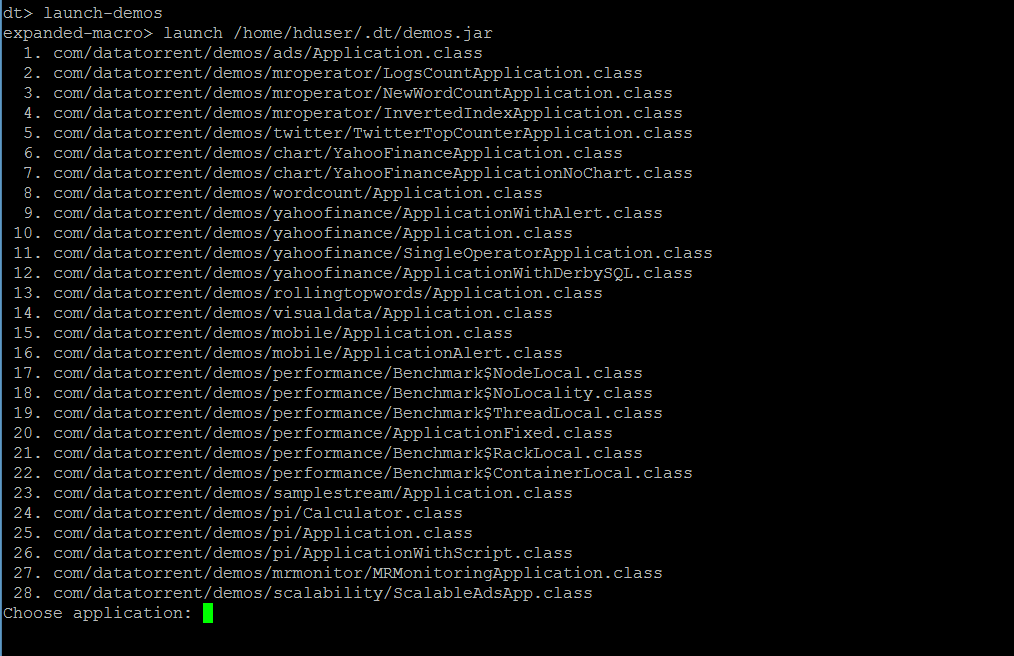
1. Start the DataTorrent Gateway process. This can be done from the command line as –



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

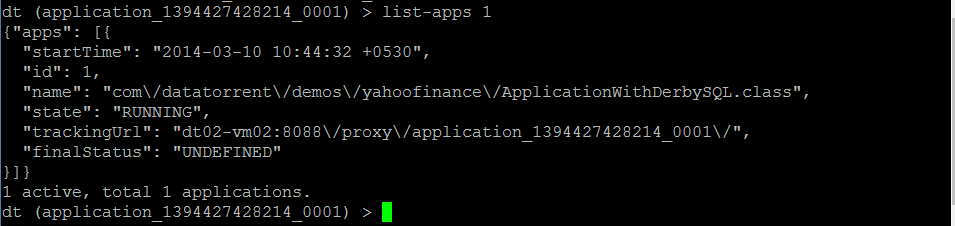
Please refer to DataTorrent Quick Start Guide for additional details.

# Monitoring the application

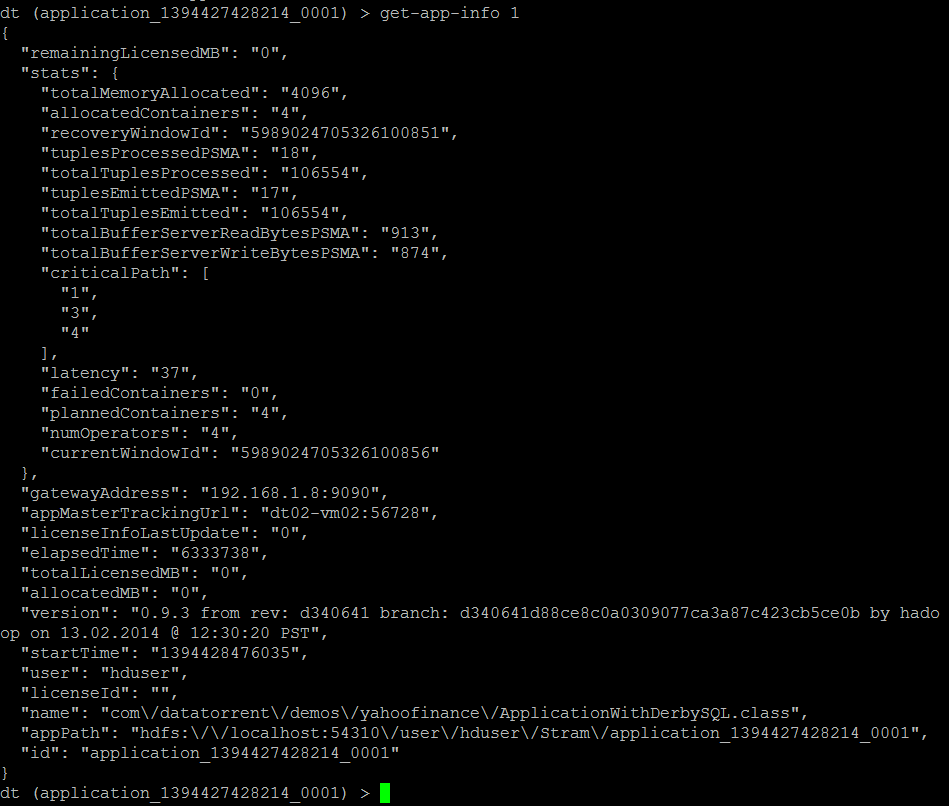
Once launched, the application can be monitored/managed from either the command line viz. ‘dtcli’ or the GUI viz. DataTorrent Gateway Console. The console and cli get data from same webservices layer. Following are some of the sample operations that are typically performed while monitoring the application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

1. List the application -

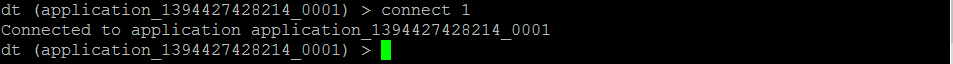
#### Using ‘dtcli’



1. Get detailed information about a given application –



1. Connect to the application –

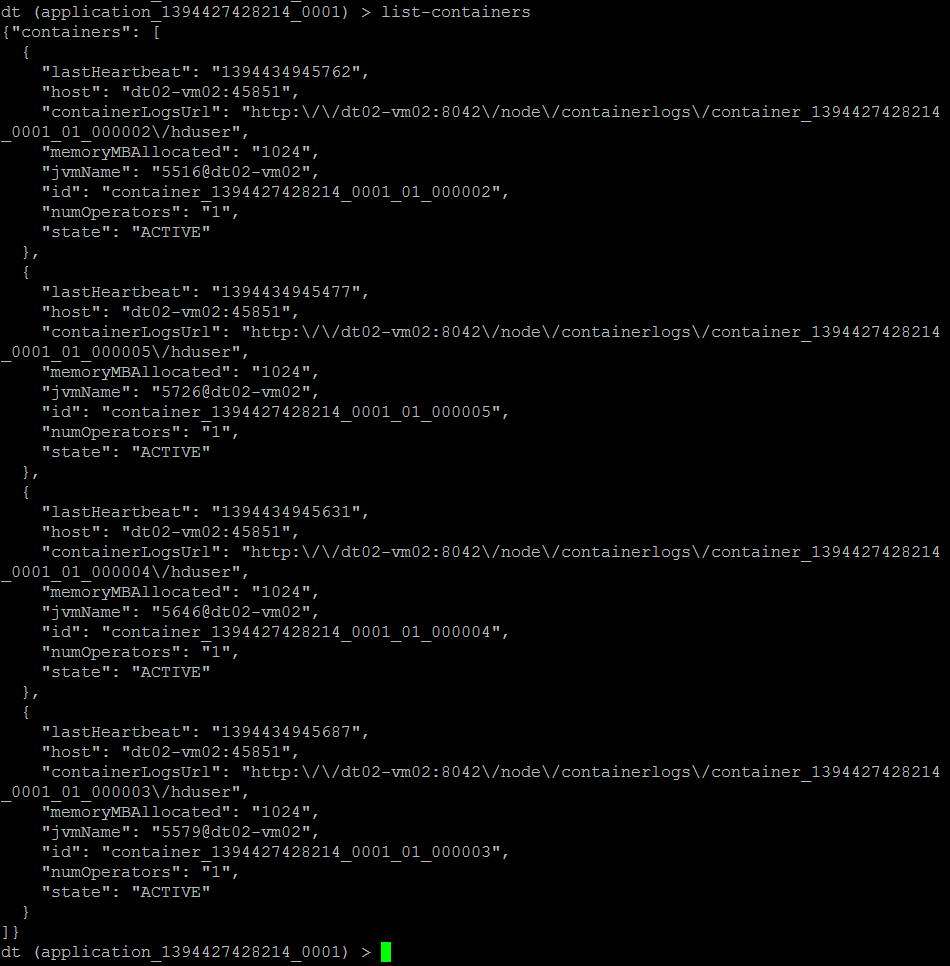


1. Show logical plan



1. List the containers :

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

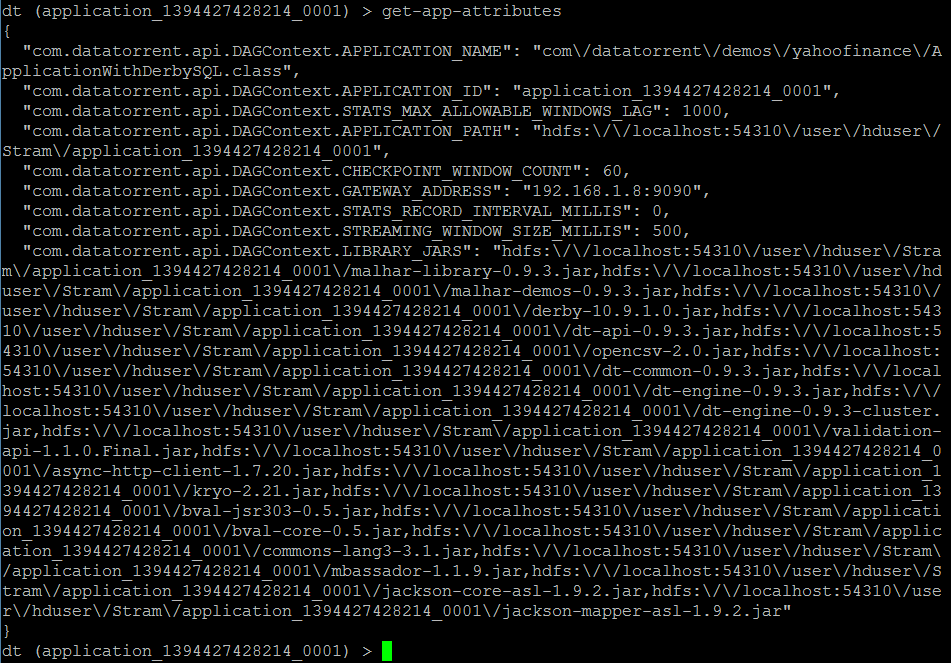


1. List all the operators –

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



1. Get application attributes :

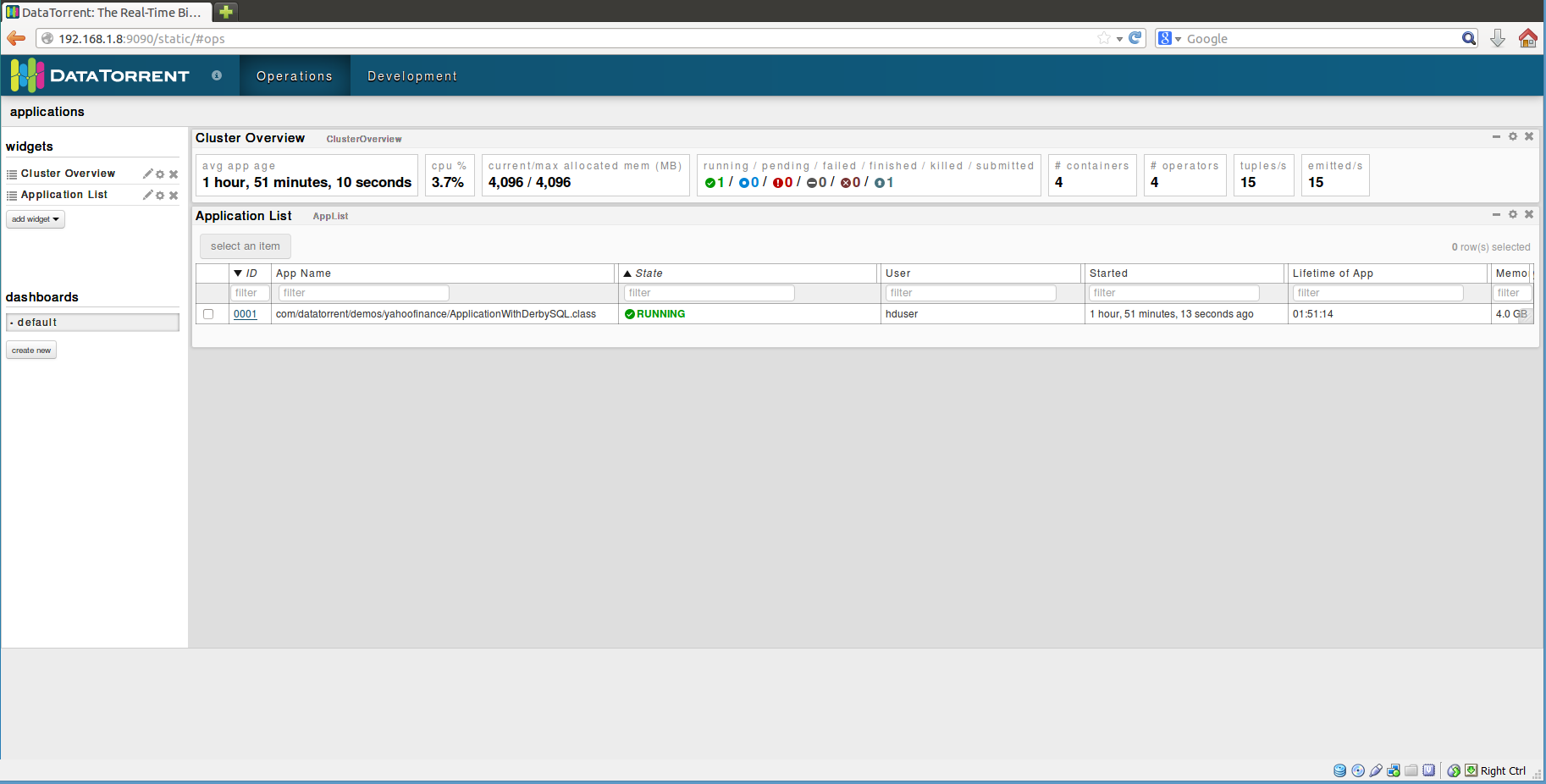


1. Similarly the user can use other dtcli to monitor/manage the application.

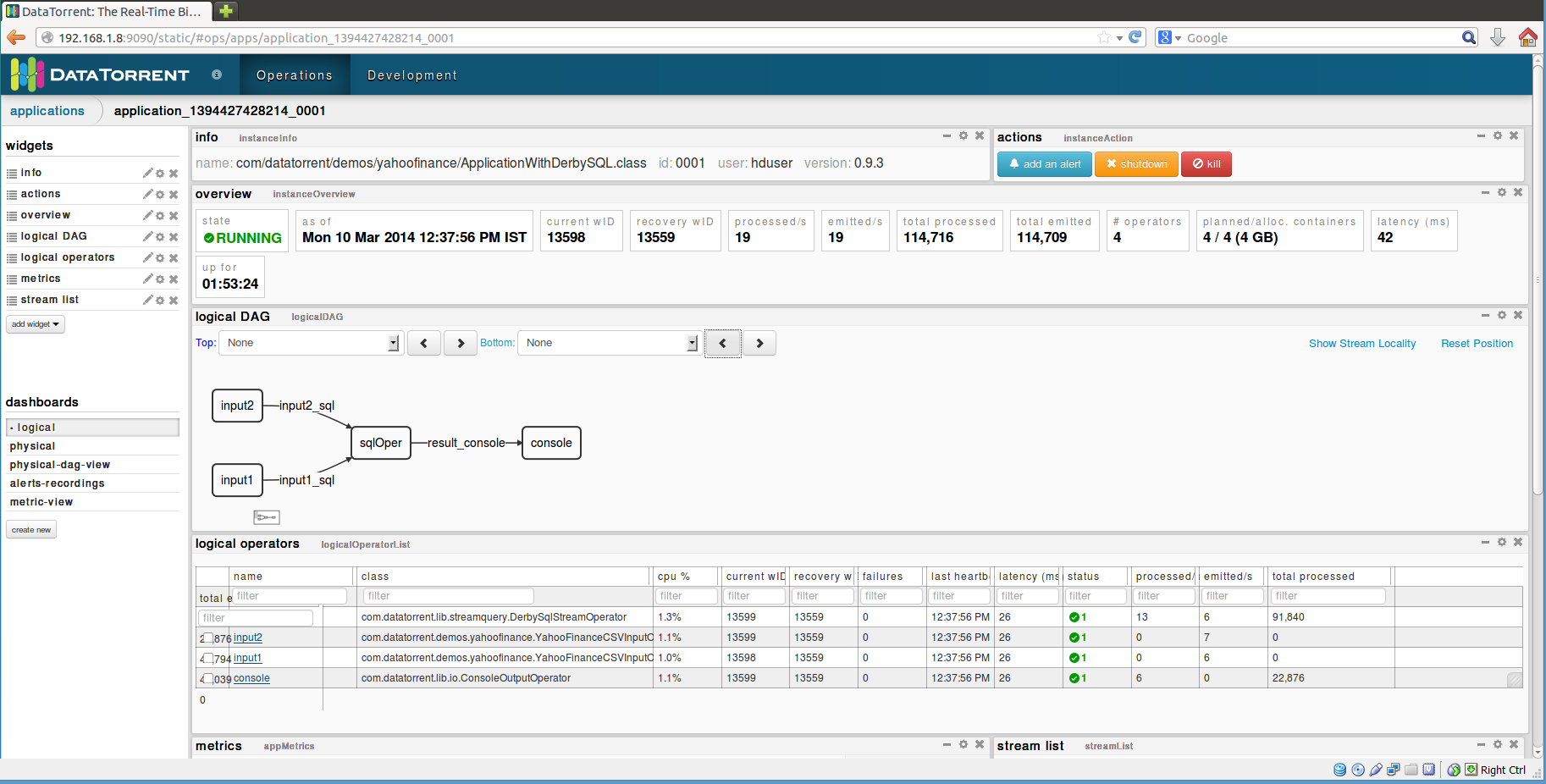
#### Using ‘Gateway Console’

# 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 –

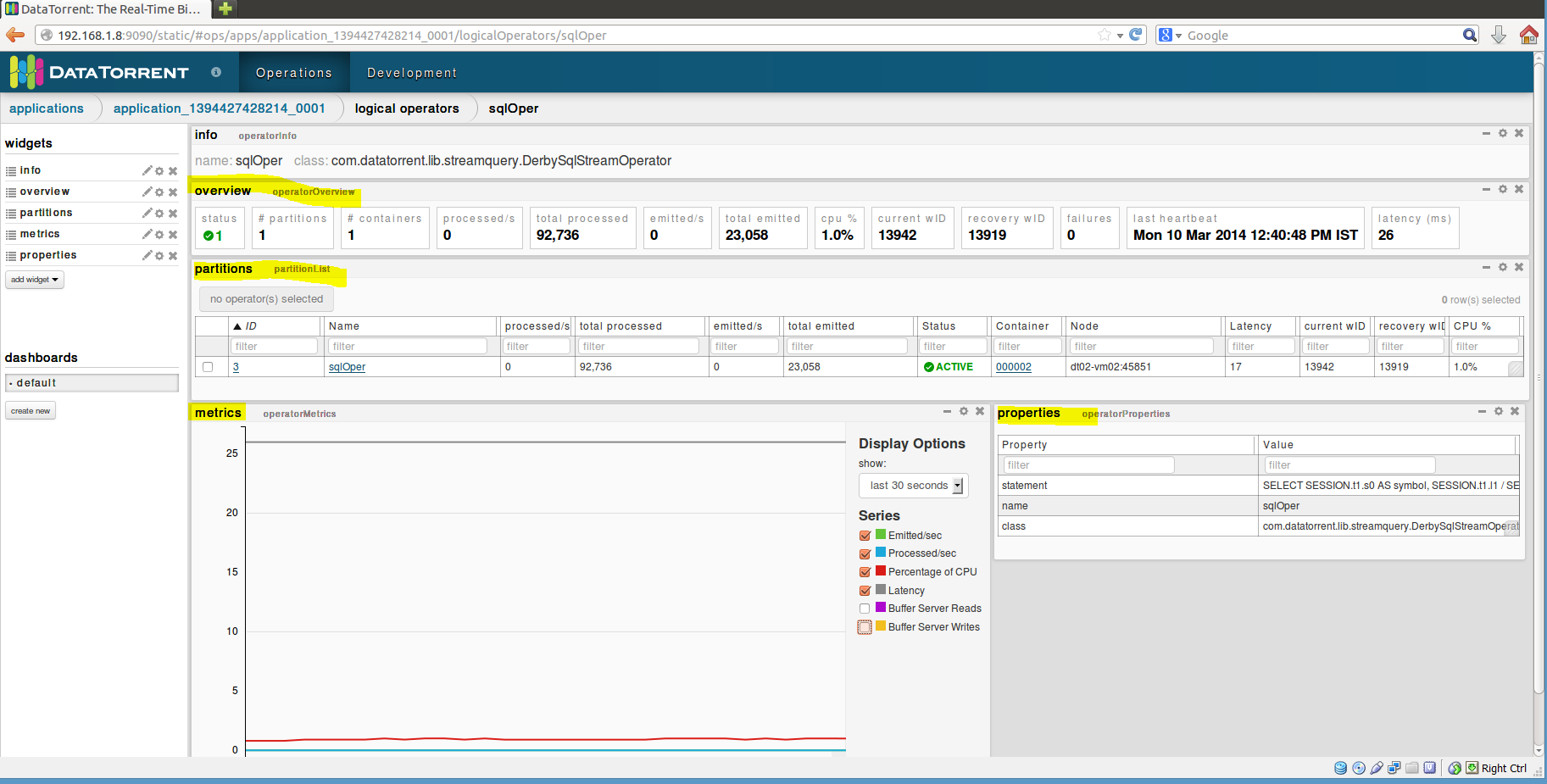


As you can see, it shows you a lot of details about the application, such as –

* 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.

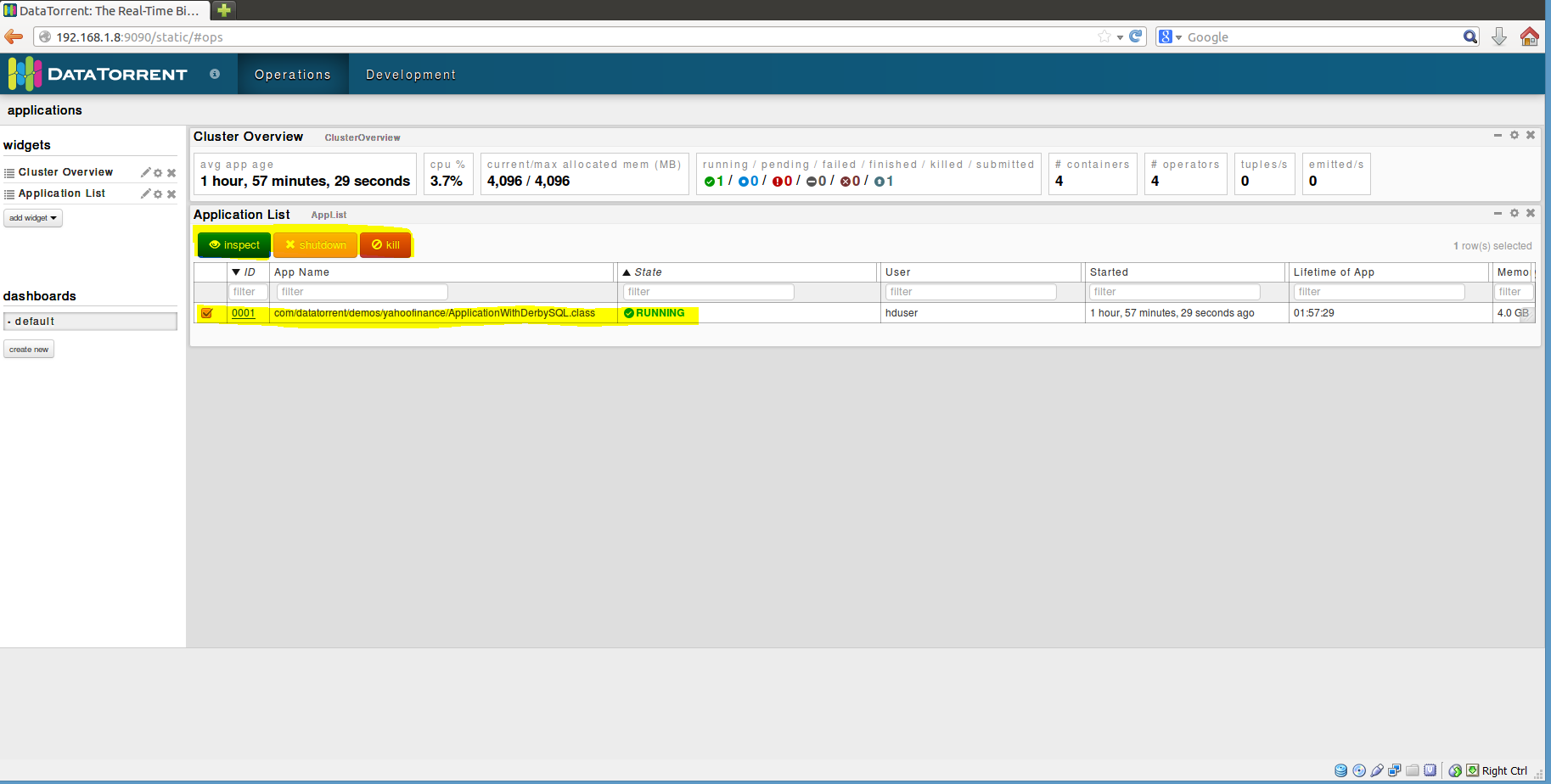


You can see detailed information about –

* 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

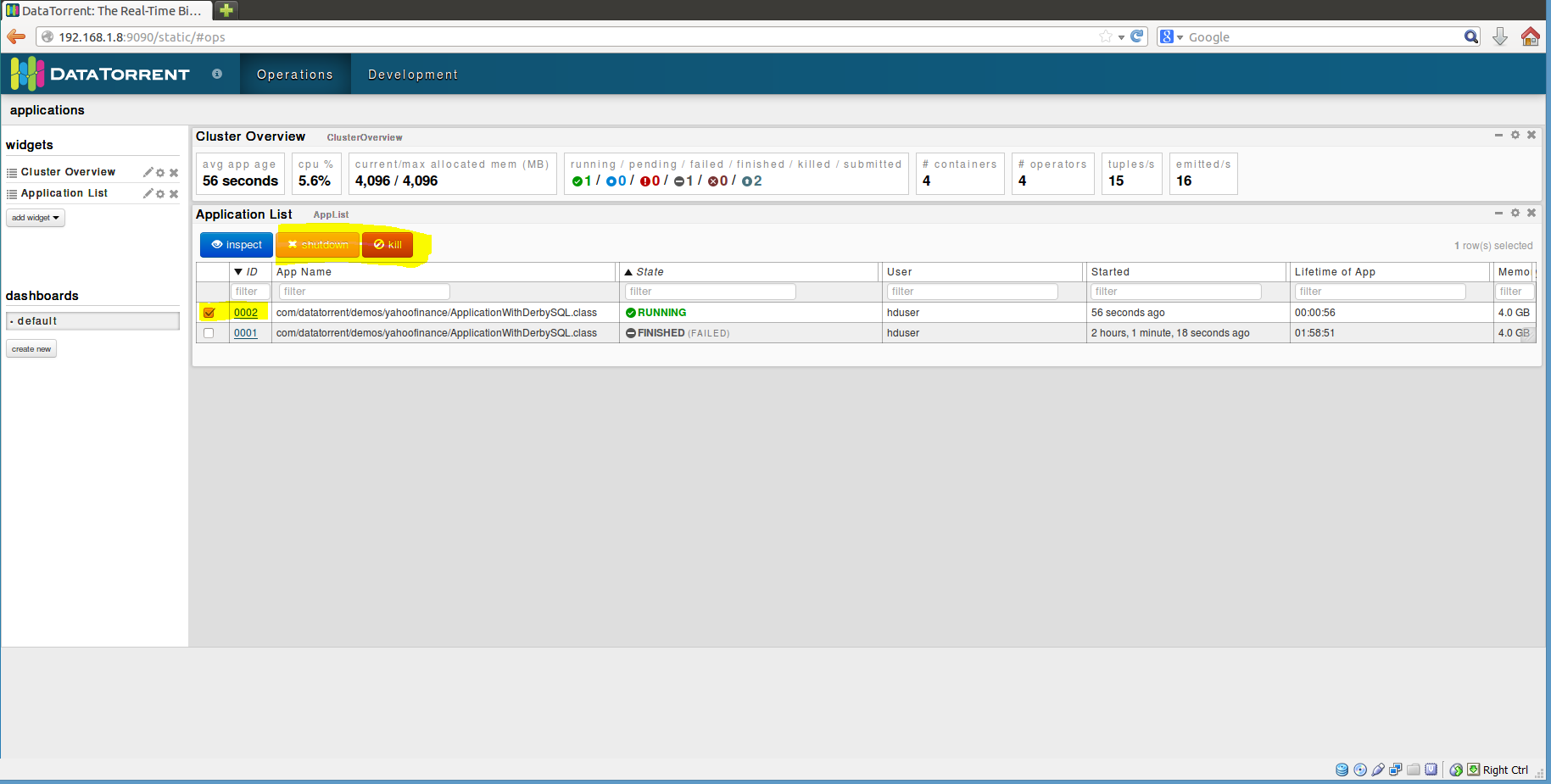
The user can use the ‘shutdown-app’ command to shut down the application from the dtcli.

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 perform massive number of varied computations in real time.

The output is displayed on the console as shown in the screenshot below –

