[Overview 2](#_Toc381975154)

[Pre-requisites 2](#_Toc381975155)

[Configuration 2](#_Toc381975156)

[Algorithm 2](#_Toc381975157)

[Design 2](#_Toc381975158)

[DAG 3](#_Toc381975159)

[Operators 3](#_Toc381975160)

[StockTickInput 3](#_Toc381975161)

[SumKeyVal 4](#_Toc381975162)

[ConsolidatorKeyVal 5](#_Toc381975163)

[RangeKeyVal 5](#_Toc381975164)

[SimpleMovingAverage 5](#_Toc381975165)

[ConsoleOutputOperator 5](#_Toc381975166)

[Functionality 6](#_Toc381975167)

[Launching the application 6](#_Toc381975168)

[Monitoring the application 7](#_Toc381975169)

[Operations 12](#_Toc381975170)

[Stopping the application 15](#_Toc381975171)

[Conclusions 16](#_Toc381975172)

Application ‘Yahoo Finance’

# Overview

The ‘Yahoo finance’ demo application is one of the simplest applications to understand massive computational ability of the DataTorrent platform. This application gets Yahoo finance feed and calculates minute price range, minute volume and simple moving average of over a period of 5 minutes. Computing this data is useful for charting a ticker on an intraday basis

The application samples yahoo finance ticker every 200ms. All data points in one second are streamed from the input adapter. 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. This application analyzes and provides the user the following information in real time -

* Consolidated volume
* The high and low trading prices over every minute
* Simple moving average over every 5 minute interval

All the above is done for each ticker symbol and in real time.

This application also demonstrates the statefulness of the application. An outage on a node would cause the averages to be lost. This application can be tested to see the averages retained without any loss of data on the node that computes these averages

# Pre-requisites

There are no special pre-requisites needed to run the ‘Yahoo finance’ demo application on the DataTorrent platform.

# Configuration

There is no particular configuration that needs to be done to run the ‘Yahoo finance’ demo application on the DataTorrent platform.

# Algorithm

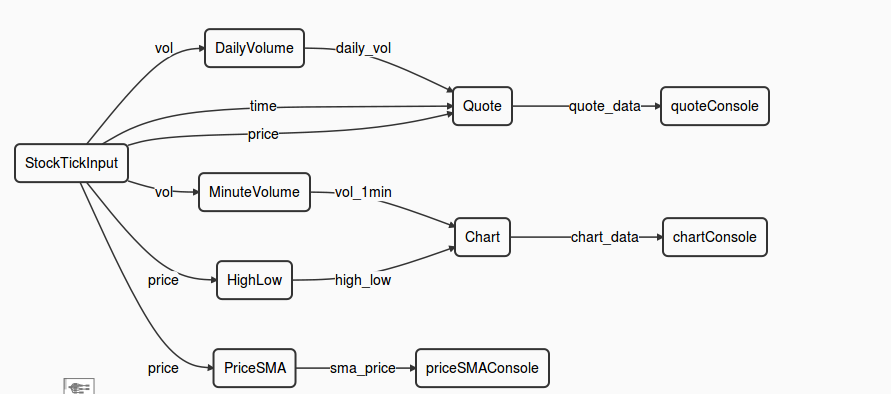
The ‘Yahoo finance’ application written for DataTorrent platform uses this algorithm:

1. Get the price, volume and timestamp for the specified tickers and emit the incremental volume, timestamp and the total volume
2. Calculate incremental volume over the past minute
3. Identify the high and low price over the past one minute
4. Calculate the simple moving average every second over a period of past 5 minutes.

# 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’ 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** |
| [StockTickInput](#StockTickInput) | tick | Custom operator |
| [SumKeyVal](#SumKeyVal) | dailyVolume | lib/math/SumKeyVal |
| [ConsolidatorKeyVal](#ConsolidatorKeyVal) | quoteOperator | lib/stream/ConsolidatorKeyVal |
| [RangeKeyVal](#RangeKeyVal) | highlow | lib/math/RangeKeyVal |
| [SumKeyVal](#SumKeyVal) | minuteVolume | lib/math/SumKeyVal |
| [ConsolidatorKeyVal](#ConsolidatorKeyVal) | chartOperator | lib/stream/ConsolidatorKeyVal |
| [SImpleMovingAverage](#SimpleMOvingAverage) | priceSMA | lib/multiwindow/SimpleMovingAverage |
| [ConsoleOutputOperator](#ConsoleOutputOperator) | quoteConsole | lib/io/ConsoleOutputOperator |
| [ConsoleOutputOperator](#ConsoleOutputOperator) | chartConsole | lib/io/ConsoleOutputOperator |
| [ConsoleOutputOperator](#ConsoleOutputOperator) | priceSMAConsole | lib/io/ConsoleOutputOperator |

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

This is a custom operator which has been written to interacts with ‘Yahoo finance’. This operator gets information about the stock prices and volume for the specified ticker symbols. By default, it fetches the information for IBM, Google, Apple and Yahoo. It fetches this data from “Yahoo finance”. Following information for each of the ticker symbols is received –

* Price
* Volume
* Time

On receiving this information, this operator calculates the change in volume. The values for the price, incremental volume, last traded time and total volume for each ticker symbol specified are sent as output on the respective output ports for further processing.

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

This operator takes a key-value pair and sums the values for a given key. It then outputs a key-value pair where the value is the summation of all the individual values received for that particular key.

The user has an option to filter the keys, in which case, these keys will not be sent out.

The user also has an option to retain the sum of values calculated over a streaming window. In such a case the sum of values in the next window will be done over the sum in the current window i.e. in other words the sum would be cumulative. By default the summation is done for a streaming window and is cleared at the end of the window.

*Default behavior –*

* If ‘k1’ is the key and v11, v12, v13 …. are the values received as input, then this operator will sum all the values received for the key ‘k1’ i.e.
* v1 = v11 + v12 + v13 + …..
* It will out put the key-value pair (k1, v1) and then clear v1.
* For the next window, v2 will be calculated and will be –
* v2 = v21 + v22 + v23 + …
* The operator will out put the key-value pair (k1, v2) and then clear v2.

*If specified so, this default behavior can be changed by setting the attribute ‘cumulative’ to ‘true’ so that –*

* The first streaming window will still out put the key-value pair (k1, v1) and not clear the value v1.
* The second streaming window will now out put the key-value pair (k1, v2) where, v2 is calculated as –

**v2** = **v1** + v21 + v22 + v23 + …

* Similarly, for the third streaming window, the output would be (k1, v3) where v3 will be calculated as –

**v3** = **v2** + v31 + v32 + v33 + …

Thus we get a consolidated sum of the values across streaming windows for a given key ‘k1’.

For the instance ‘dailyVolume’ of the ‘SumKeyVal’ operator, ‘cumulative’ is set to ‘true’ since we want to calculate the consolidated volume for the given ticker.

For the instance ‘minuteVolume’, the default value of ‘cumulative’ is retained (i.e. it is ‘false’)

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

This operator consolidates the information for a given ticker symbol. There are two separate instances of this operator in the DAG viz. ‘quoteOperator’ and ‘chartOperator’.

The quoteOperator consolidates the current price, current time and the dailyVolume till the current time. This consolidated information is then sent to the console operator.

The ‘chartOperator’ consolidates the highest and the lowest price for a given ticker with the volume during that application window, and then sends it to the console operator.

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

This operator is used to get the high and low value for a given key in the specific application window. It takes as input, a key-value pair. For each key in the incoming input data, it will find the highest and the lowest of the ‘value’. At the end of the application window, it will output another value pair which has the value as the highest and the lowest data value over the application window for the given key.

For example,

For a given key ‘k1’, the incoming stream has the values as ‘v1’, ‘v2’, …, ‘vmin’, ….’vmax’…. And so on up to ‘vn’

Assume that ‘vmin’ turns out to be the minimum value and ‘vmax’ turns out to be the maximum value out of the above range of values. In this case, the output from this operator would be a key value pair where the ‘value’ would be an object made up of ‘vmin’ and ‘vmax’. So, the output will be –



In this case, the instance ‘highlow’ of the ‘RangeKeyVal’ operator will give us the highest and the lowest price for a given ticker during the application window.

### [SimpleMovingAverage](https://github.com/DataTorrent/Malhar/blob/src/main/java/com/datatorrent/lib/multiwindow/SimpleMovingAverage.java)

This operator calculates the simple moving average of the incoming data over a specified interval. In case of this application, the calculation of simple moving average is done over a period of 5 minutes.

This operator calculates the average price for a given ticker over a 5 minute sliding window. As the window moves forward, the oldest value is dropped and the newest value is taken in and the average is calculated again. This happens every second i.e. the average over the past 5 minutes is calculated every second.

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

The ‘ConsoleOutputOperator’ is used to display various outputs on the console. In the case of the current application under discussion, we display the following information using various instances of the ConsoleOutputOperator –

1. Ticker symbol, current time, quote at current time and the daily volume
2. Ticker symbol, highest and lowest price and volume – during the past one minute interval
3. Ticker symbol and the simple moving average for the stock over a 5 min. window

#### Streams

|  |  |
| --- | --- |
| **Stream** | **Description** |
| price | Carries information about the last traded price for a ticker symbol |
| vol | Carries information about the incremental volume |
| time | Carries information about the time when the last trade occurred for a ticker symbol |
| daily\_vol | Carries the cumulative sum of the volume |
| quote\_data | Carries the consolidated information about the cumulative volume at the specified time, the time and the price at that time to the consoleoutput operator |
| high\_low | Carries the highest and the lowest price for a given ticker during the one minute application window |
| vol\_lmin | Carries the value of volume for a given ticker in a one minute application window |
| chart\_data | Carries the consolidated information about the highest prce, lowest price and the volume for a given ticker in a one minute application window |
| sma\_price | Carries the value of simple moving average value to 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.

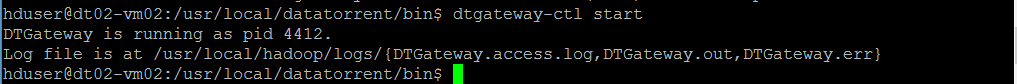
|  |  |  |
| --- | --- | --- |
| **Type of Operator** | **Statefull** | **Partitionable** |
| [StockTickInput](#StockTickInput) | Y | Y |
| [SumKeyVal](#SumKeyVal) | Y | Y |
| [ConsolidatorKeyVal](#ConsolidatorKeyVal) | Y | Y |
| [RangeKeyVal](#RangeKeyVal) | Y | Y |
| [SImpleMovingAverage](#SimpleMOvingAverage) | Y | Y |

# Launching the application

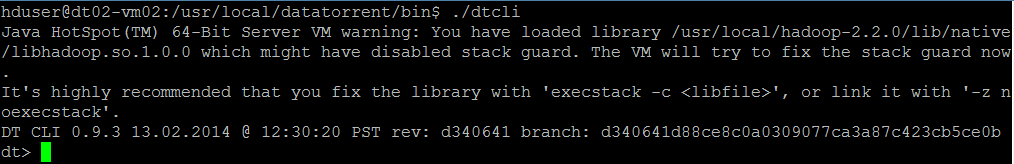
In order to be able to launch sample Demo Applications, please make sure you have completed following steps.

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

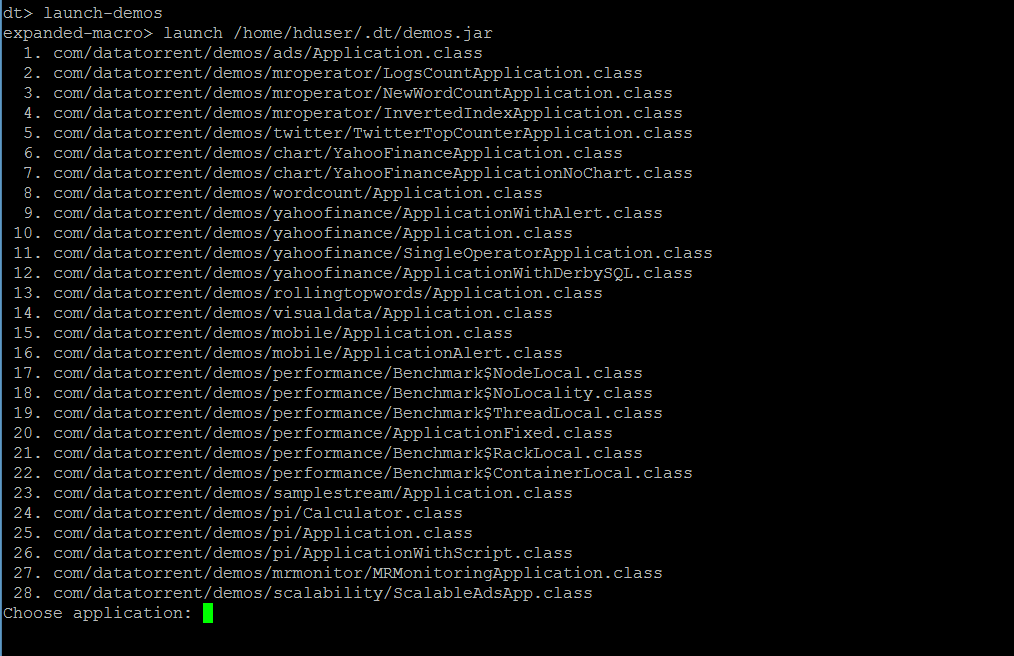
1. Start the DataTorrent Gateway process. This can be done from the command line as –



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

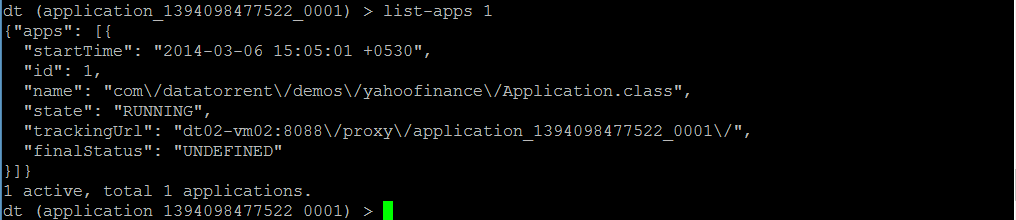
Please refer to DataTorrent Quick Start Guide for additional details.

# Monitoring the application

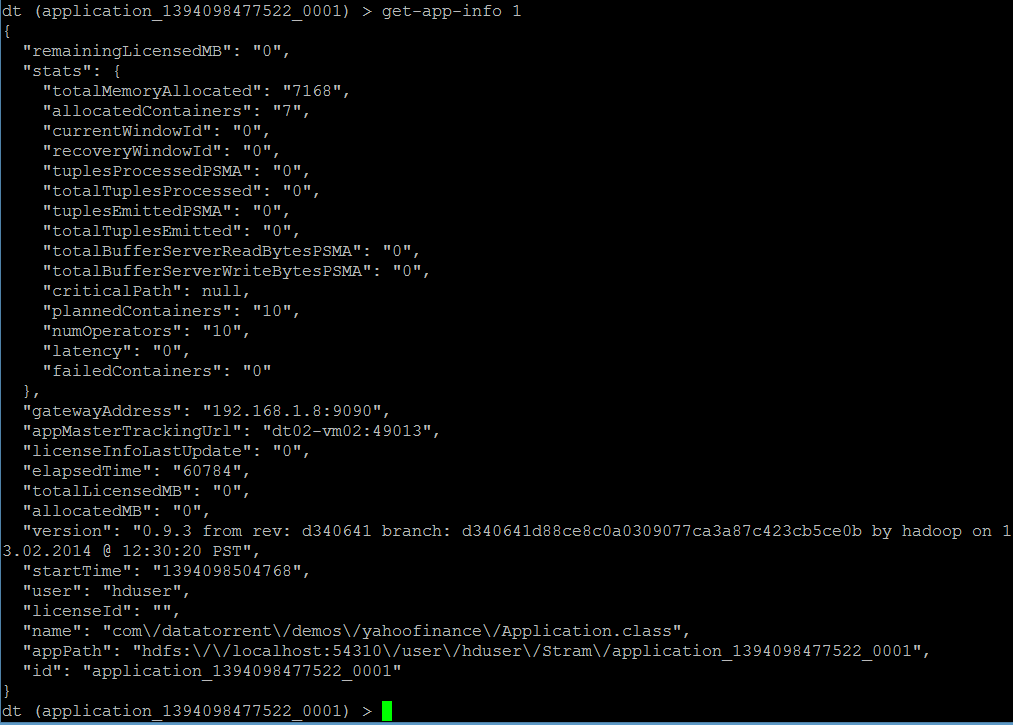
Once launched, the application can be monitored/managed from either the command line viz. ‘dtcli’ or the GUI viz. DataTorrent Gateway Console. The console and cli get data from same webservices layer. Following are some of the sample operations that are typically performed while monitoring the application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

1. List the application -

#### Using ‘dtcli’



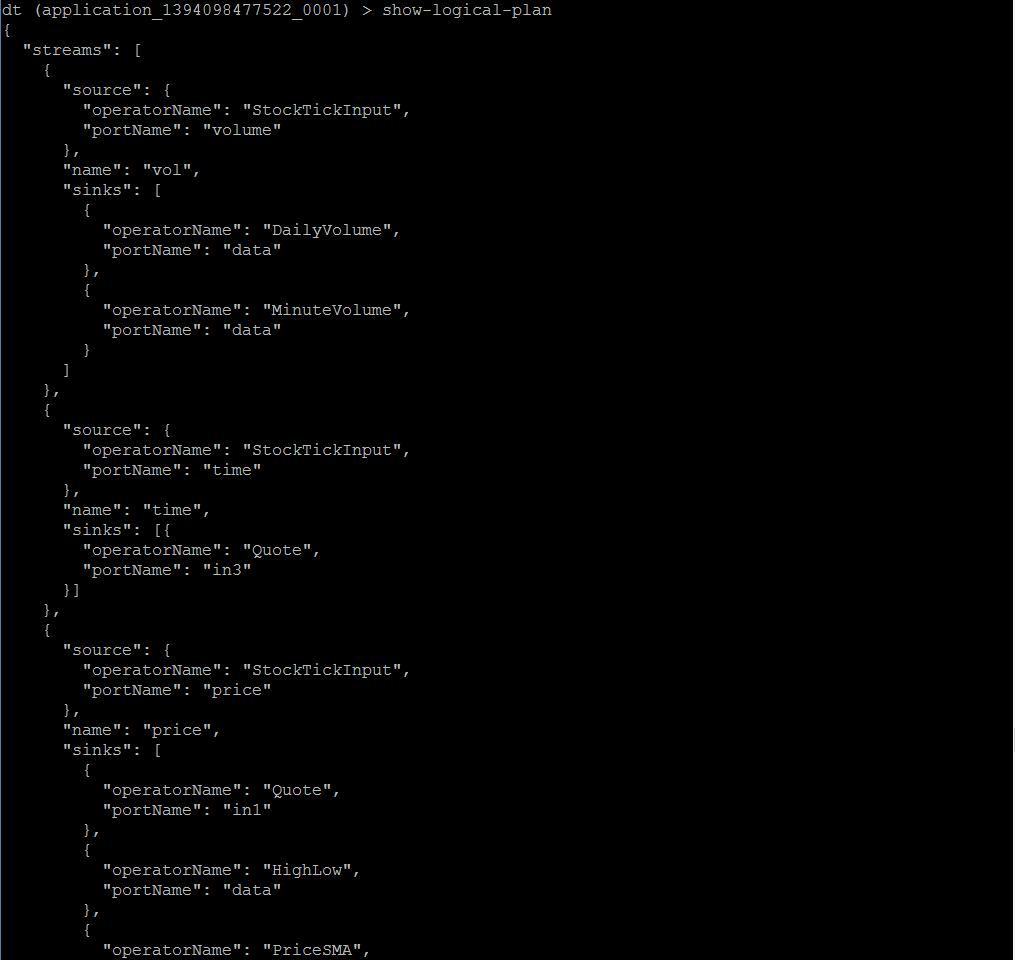
1. Get detailed information about a given application –



1. Connect to the application –



1. Show logical plan



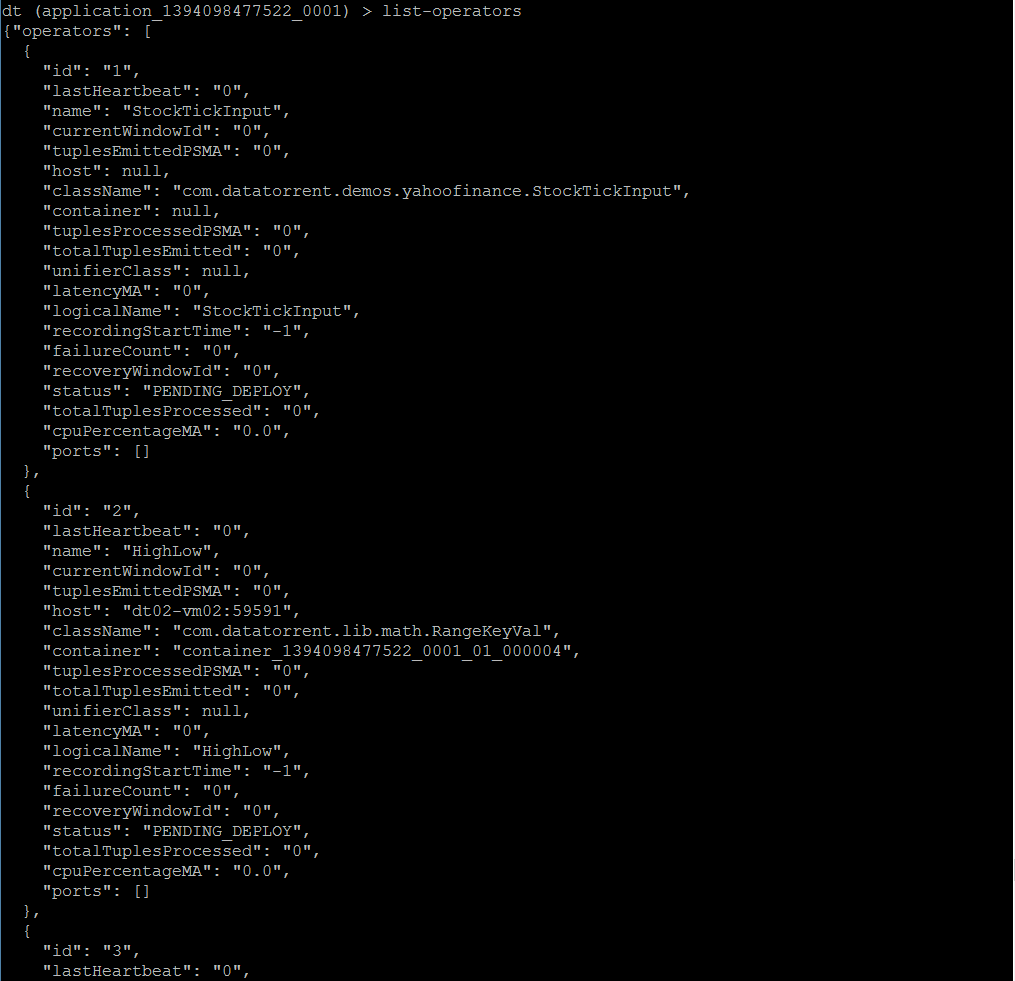
1. List the containers :

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

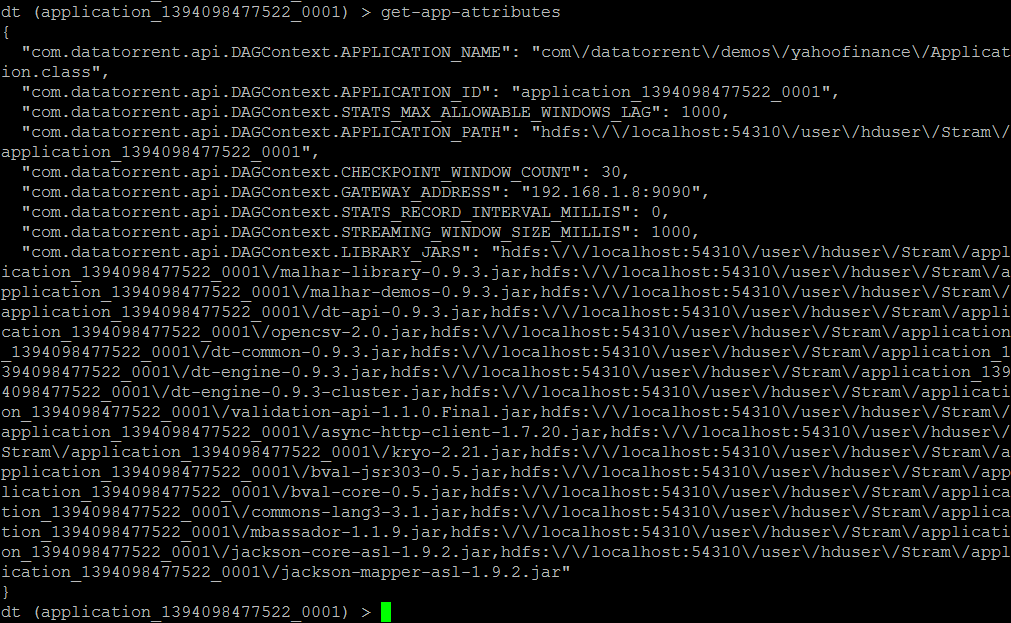


1. List all the operators –

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



1. Get application attributes :

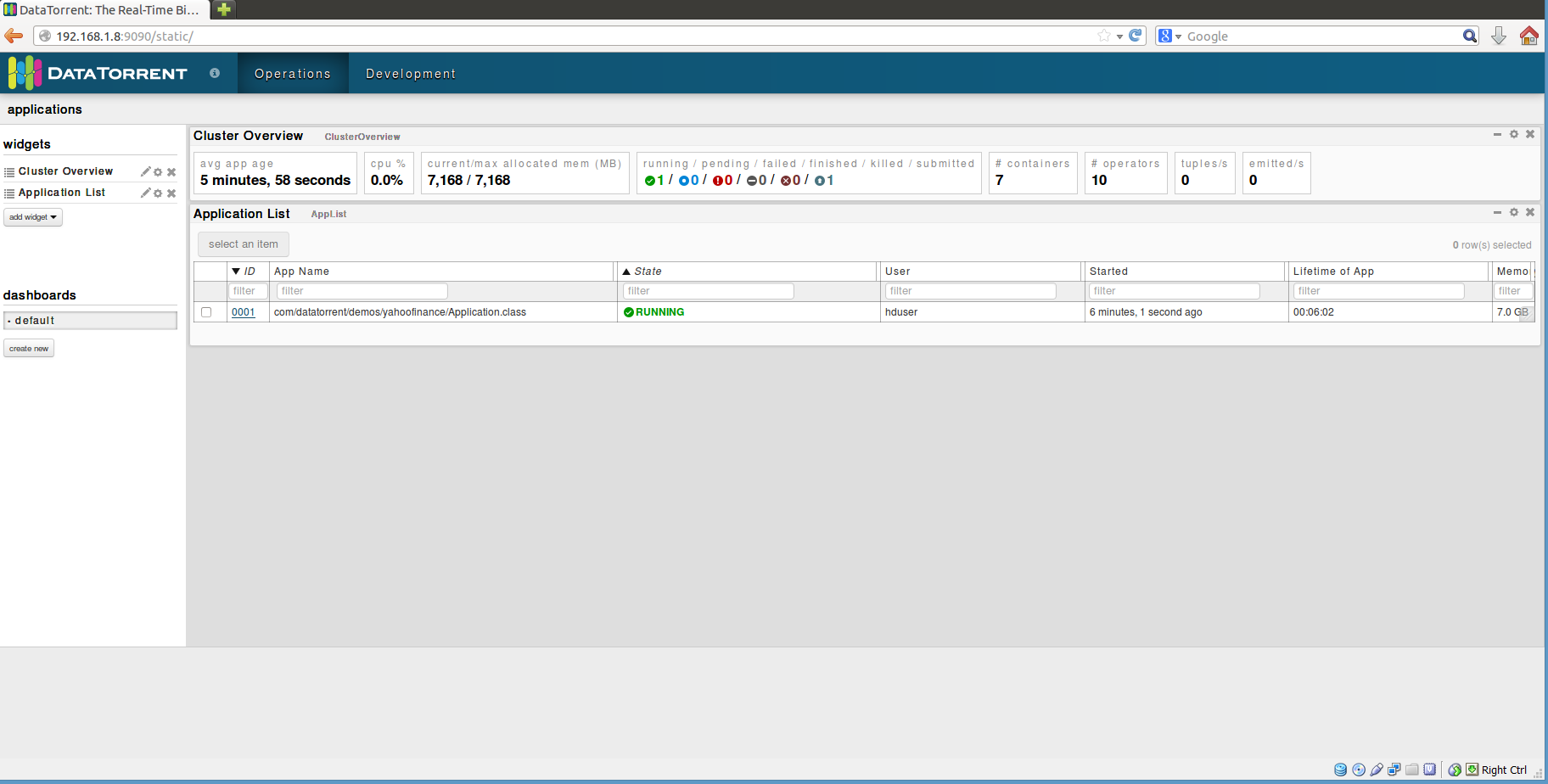


1. Similarly the user can use other dtcli to monitor/manage the application.

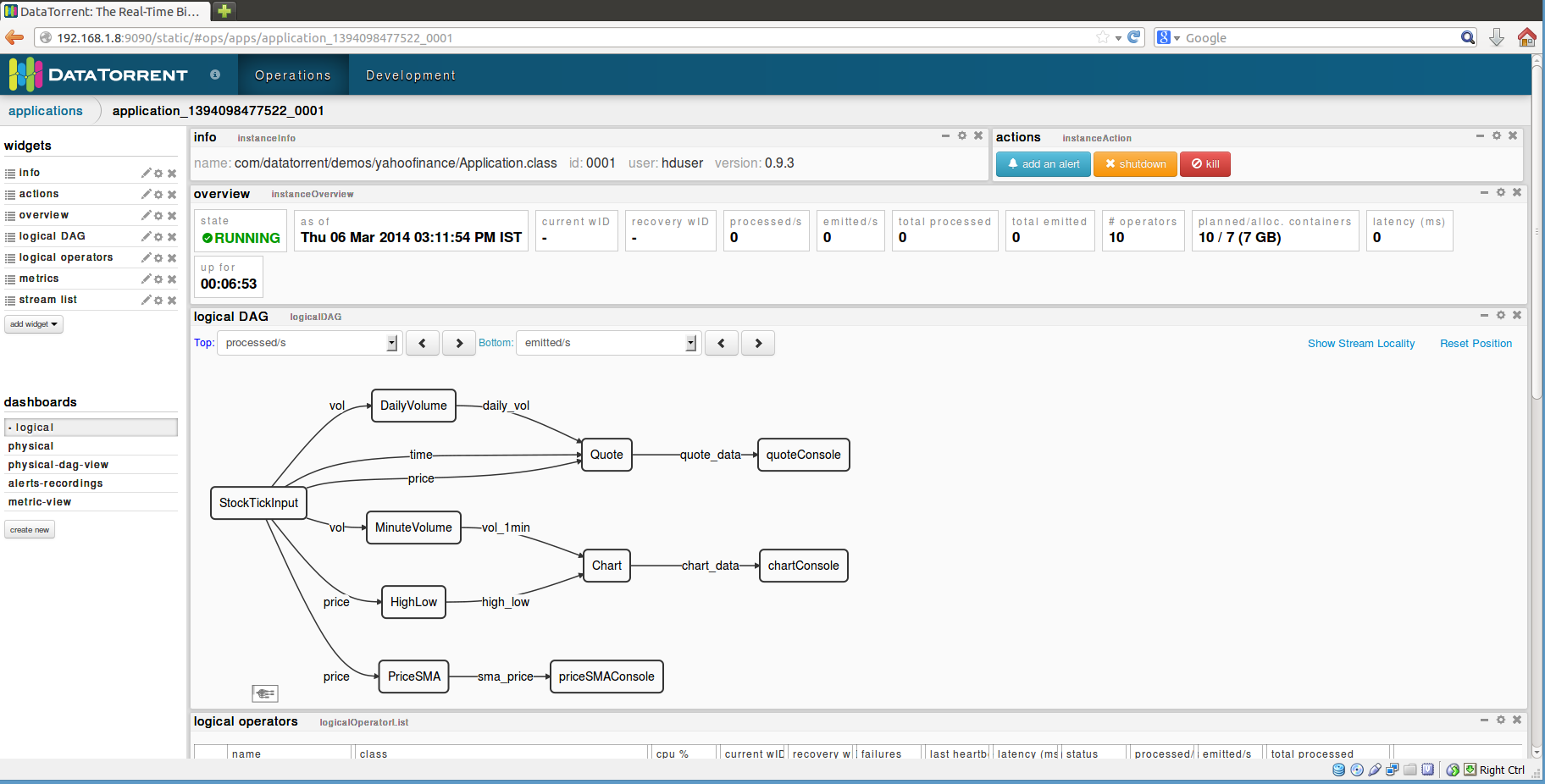
#### Using ‘Gateway Console’

# 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 –

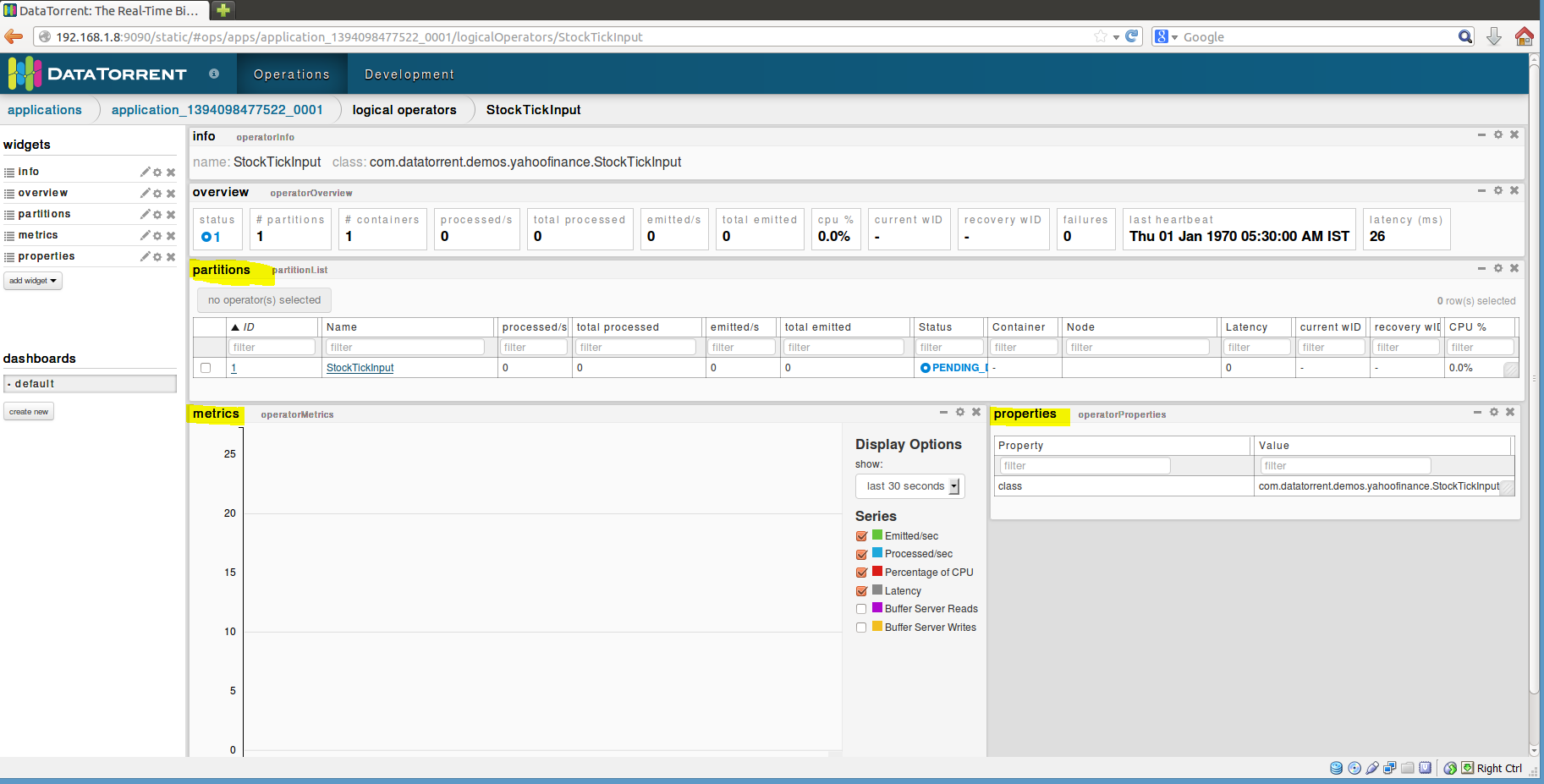


As you can see, it shows you a lot of details about the application, such as –

* 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.

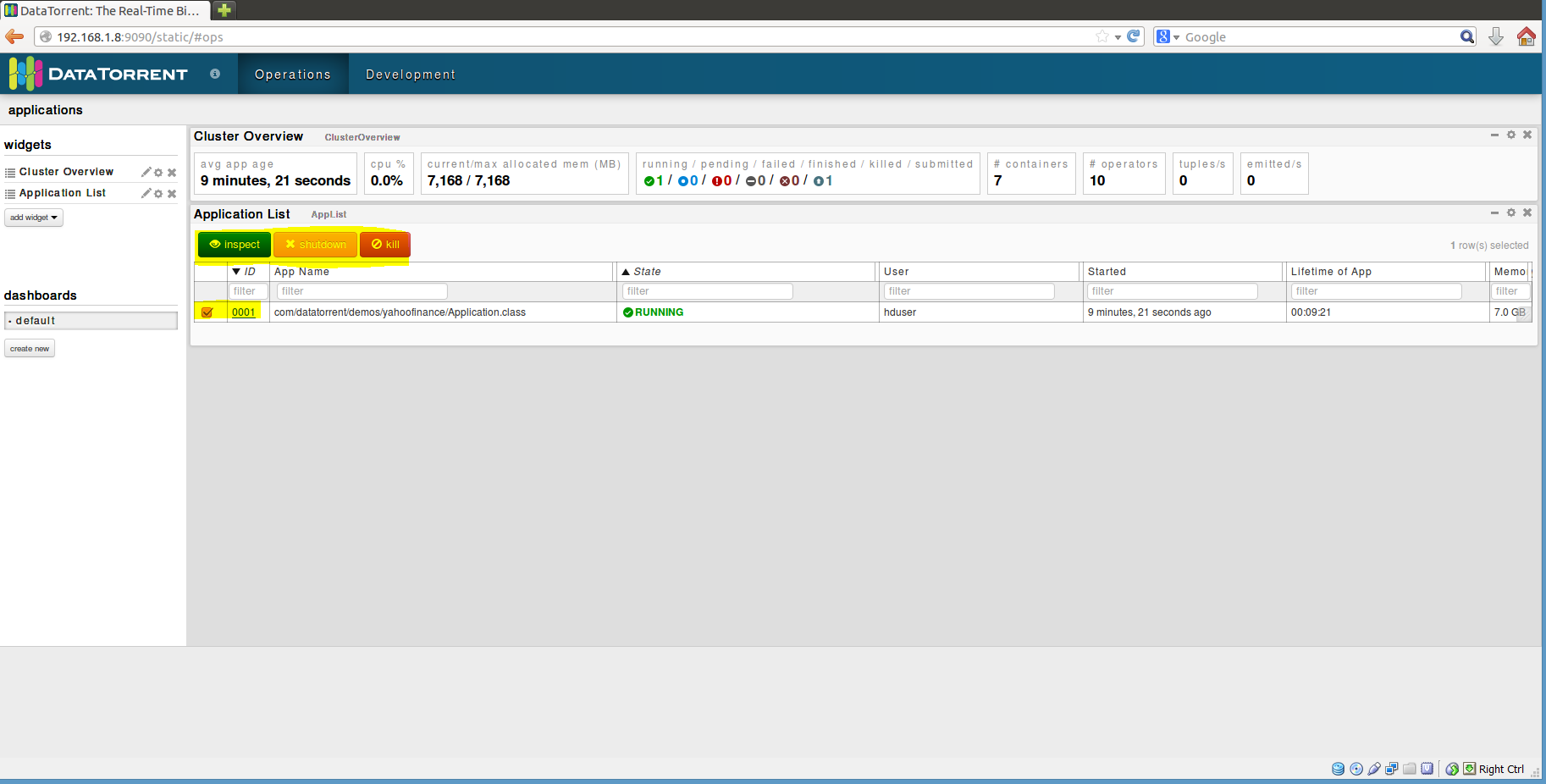


You can see detailed information about –

* 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

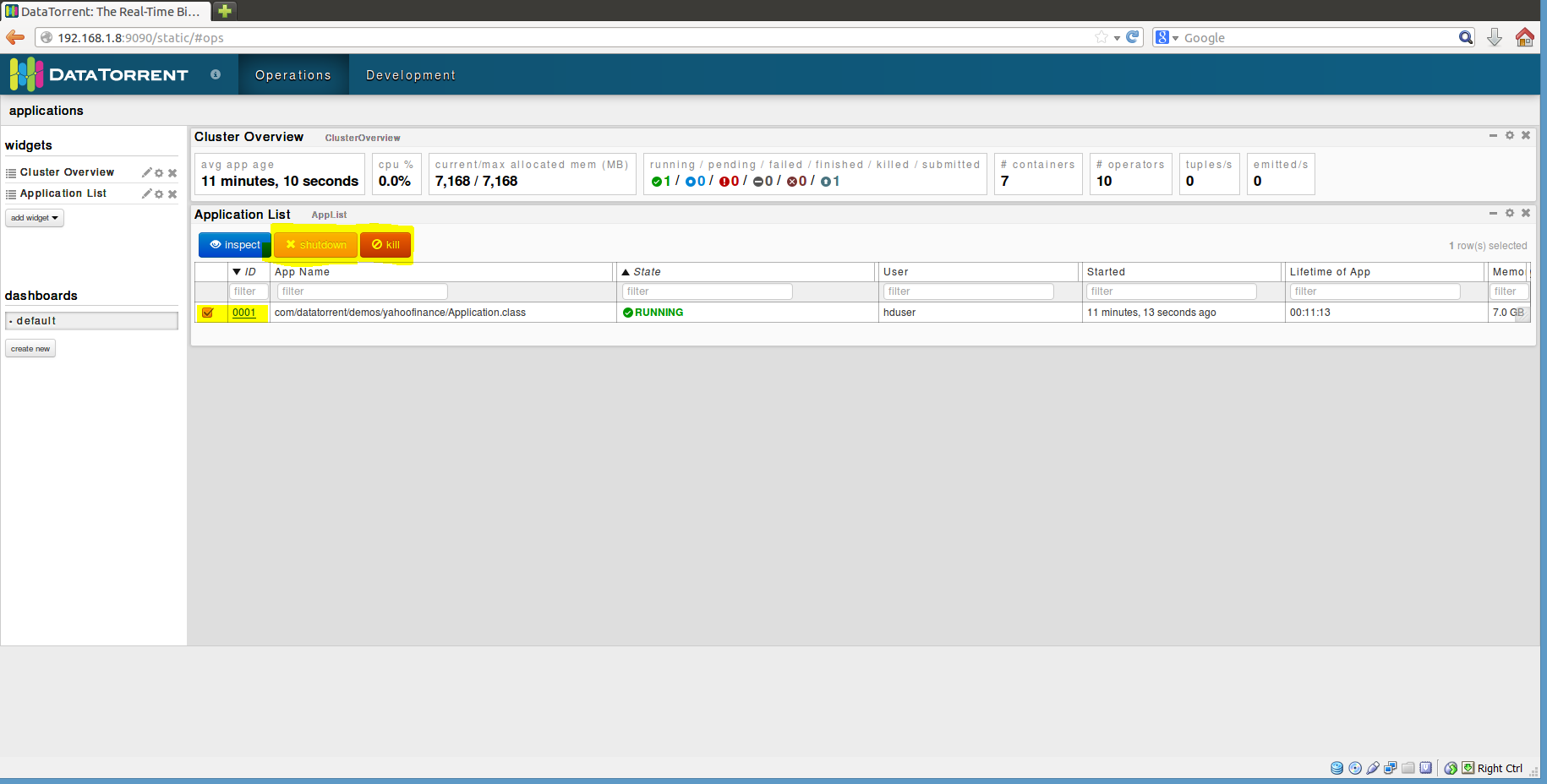
The user can use the ‘shutdown-app’ command to shut down the application from the dtcli.

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.