Twitter Rolling Top Words

Demo Application Guide

Version 1.0

March-2014

[Introduction 3](#_Toc382225872)

[Pre-Requisites 3](#_Toc382225873)

[Hadoop Services 3](#_Toc382225874)

[dtgateway Service 3](#_Toc382225875)

[Assumptions 4](#_Toc382225876)

[DataTorrent Install Location 4](#_Toc382225877)

[Location of Packaged Demo Applications 4](#_Toc382225878)

[Launching the Demo Applications (demos.jar) using “dtcli” 4](#_Toc382225879)

[Terms Used 4](#_Toc382225880)

[Twitter Rolling Top Words Application 5](#_Toc382225881)

[Twitter Application Functionality 5](#_Toc382225882)

[Application Directed Acyclic Graph (DAG) 6](#_Toc382225883)

[Operators 6](#_Toc382225884)

[Streams 6](#_Toc382225885)

[Application Source Code 6](#_Toc382225886)

[Twitter4J and Twitter Streaming API 6](#_Toc382225887)

[Twitter Application OAuth Access 7](#_Toc382225888)

[How to get Twitter OAuth Access Tokens 7](#_Toc382225889)

[How to use the Twitter OAuth Access Tokens inside Twitter Demo App 7](#_Toc382225890)

[Updating ~/.dt/dt-site.xml file 7](#_Toc382225891)

[How to Launch Twitter Demo application (dtcli) 8](#_Toc382225892)

[How to monitor the Application 9](#_Toc382225893)

[Monitoring the application using dtcli 9](#_Toc382225894)

[Monitoring the application using DataTorrent Gateway Console 9](#_Toc382225895)

[How to view Twitter Rolling Top Word Counts 10](#_Toc382225896)

[Looking at the Physical Dashboard 10](#_Toc382225897)

[Looking at Container details 11](#_Toc382225898)

[Looking at the generated Log File for displayed results 11](#_Toc382225899)

[Glossary of Terms used 13](#_Toc382225900)

# Introduction

Twitter Rolling Top Words application demonstrates one of the very important platform features which is to allow real-time computations over a sliding window.

Real-time stream processing applications require performing computations over a certain time period. There could be business requirement where application needs to compute the results over a given time period or window. That time window could be last few seconds, minutes, hours or even last entire day. DataTorrent’s real-time stream processing platform provides built-in support to handle window-based computations. DataTorrent Platform abstracts out the windowing operations and makes the task of writing such applications very easy for application developer. The application developer can only focus on the business logic.

Twitter Demo application samples the live tweets from Twitter website and generates the counts of unique words that were used most inside the tweets. This application samples all the tweets received from Twitter and generates the counts for top 10 words used inside the tweets over last 5 minutes time window. The 5 minute computation window is a sliding window and gets moved by duration of 500 milliseconds every time.

Twitter application uses Java based APIs to get live Twitter feed from Twitter Streaming/Firehose API.

# Pre-Requisites

The document assumes that following Hadoop and DataTorrent services are running before using this document,

## Hadoop Services

In order to be able to launch Demo Applications on the DataTorrent platform, Hadoop 2.2 (Yarn) grid needs to be running. The Hadoop install can be from any of the distributors or from Apache (<http://hadoop.apache.org/>) as long as it is Hadoop 2.2.

## dtgateway Service

Please make sure that “dtgateway” service is running. This can be checked by executing following command on the command shell,

$ sudo service dtgateway status

# Assumptions

This document assumes the following about the location where the Demo applications are placed.

## Location of Packaged Demo Applications

DataTorrent platform ships with a number of Demo applications. You will find a Jar file (demos.jar) containing Demo applications under DataTorrent installation directory. The Jar file demos.jar contains multiple pre-built demo applications that user can launch on a Hadoop cluster. The DataTorrent command line interface (dtcli) also present in the DataTorrent installation folder can be used to launch these Demo applications on a Hadoop cluster.

## Launching the Demo Applications (demos.jar) using “dtcli”

Please refer to Getting-Started-Guide to find out more about how to use “dtcli” and also how to use it to launch the demo applications (demos.jar) that are shipped with DataTorrent platform.

# Terms Used

To find out more about the terms used this document, please refer to section [Glossary](#_Glossary_of_Terms) in this document.

# Twitter Rolling Top Words Application

This section talks about following,

1. [Twitter Application Functionality](#_Twitter_Application_Functionality)
2. [How to get Twitter OAuth Access Tokens](#_How_to_get)
3. [How to use Twitter OAuth Access Tokens inside Twitter Demo App](#_How_to_use)
4. [How to launch the application](#_How_to_Launch)
5. [How to monitor the application](#_How_to_monitor)
6. [How to view Twitter Rolling Top Word Counts](#_How_to_view)

## Twitter Application Functionality

Application functionality is explained as below,

1. Application receives live Twitter feeds from Twitter’s Streaming API
2. Application extracts
3. Starts a 5 minute sliding window time interval every ½ a second
4. Computes aggregate count for individual unique words in the given sliding window
5. And finally, generates the counts for top 10 unique words for last sliding window, at end of every 500ms or ½ a second duration.

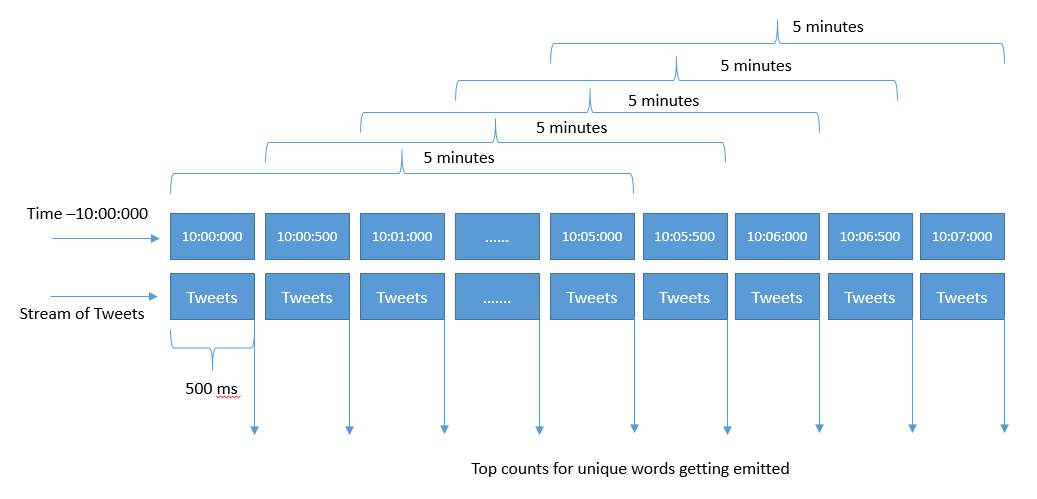


Figure Twitter Application Data Stream Flow

As shown in the above diagram, stream of Tweets gets processed within each 5 minute aggregation window. At the end of each 5 minute window, the top unique word counts get computed and the application outputs the counts. This computation window slides by ½ a second each time and repeats the computation of top unique word counts.

### Application Directed Acyclic Graph (DAG)

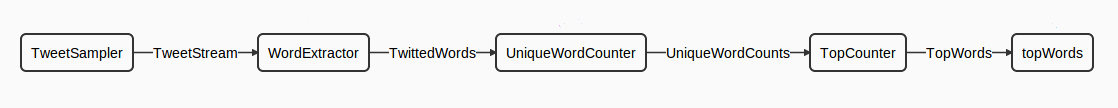


Figure Application DAG

#### Operators

Twitter application has following Operators as shown in the above DAG,

|  |  |
| --- | --- |
| **Operator** | **Description** |
| TweetSampler | Twitter Input Feed Sampler. This is an input operator. It uses Twitter4J API and samples the Tweets as received from Twitter Firehose/Streaming API. |
| WordExtractor | Uses Twitter4J APIs to extracts the words from the received tweets. |
| UniqueWordCounter | Generates the aggregated counts for the unique URLs as found inside the tweets. |
| TopCounter | Operator that outputs the counts for the top unique words for a sliding window of 5 minutes interval. |
| topWords | This is an Output operator. This operator is responsible for publishing the top unique word counts to standard output console. |

#### Streams

|  |  |
| --- | --- |
| **Stream** | **Description** |
| TweetStream | Stream of Tweets as created by TweetSampler operator. |
| TwittedWords | Stream of Tweets |
| UniqueWordCounts | Stream of Counts along with respective unique words. |
| TopWords | Stream of Top 10 Counts with respective unique words. |

### Application Source Code

Source code for the Operators used in the application can be viewed from links given below –

|  |  |  |
| --- | --- | --- |
| **Operator** | **Source** | **Library** |
| TweetSampler | [TwitterSampleInput.java](https://github.com/DataTorrent/Malhar/blob/master/contrib/src/main/java/com/datatorrent/contrib/twitter/TwitterSampleInput.java) | Custom |
| WordExtractor | [TwitterStatusWordExtractor.java](https://github.com/DataTorrent/Malhar/blob/master/demos/src/main/java/com/datatorrent/demos/rollingtopwords/TwitterStatusWordExtractor.java) | Custom |
| UniqueWordCounter | [UniqueCounter.java](https://github.com/DataTorrent/Malhar/blob/master/library/src/main/java/com/datatorrent/lib/algo/UniqueCounter.java) | Custom |
| TopCounter | [WindowedTopCounter.java](https://github.com/DataTorrent/Malhar/blob/master/demos/src/main/java/com/datatorrent/demos/rollingtopwords/WindowedTopCounter.java) | Custom |
| topWords | [ConsoleOutputOperator.java](https://github.com/DataTorrent/Malhar/blob/master/library/src/main/java/com/datatorrent/lib/io/ConsoleOutputOperator.java) | Malhar Library |

Source code for the wrapper application class can be viewed here –

* [Application.java](https://github.com/DataTorrent/Malhar/blob/master/demos/src/main/java/com/datatorrent/demos/rollingtopwords/Application.java)

### Twitter4J and Twitter Streaming API

Further details about Twitter4J and Twitter’s Streaming API can be found at,

* Twitter4J - <http://twitter4j.org/en/index.html>
* Twitter Streaming API - <https://dev.twitter.com/docs/api/streaming>

The number of tweets used for the computation are randomly sampled from Twitter Firehose and are 1% of total tweets (provided for the default access level).

## Twitter Application OAuth Access

For the Twitter Demo application to work, you need to get a Twitter API account and get the OAuth access tokens. Following sections talk about how to get the access tokens.

### How to get Twitter OAuth Access Tokens

Twitter account is needed to get the OAuth Access Tokens from Twitter.

You need to create a Twitter Application in your Twitter Account in order to get Access Tokens (API Keys). The created Access Token then can be used with Twitter Demo Application.

For further details please refer to Twitter’s official developer documentation for,

1. OAuth Access Details (<https://dev.twitter.com/docs/auth/oauth>)
2. Obtaining Access Tokens (<https://dev.twitter.com/docs/auth/obtaining-access-tokens>)

### How to use the Twitter OAuth Access Tokens inside Twitter Demo App

Once you have the OAuth Access Tokens, you should get following from your Twitter Account for the newly created application,

1. API Key
2. API Secret
3. Access Token
4. Access Token Secret

All above values need to be specified as configuration for Twitter Demo app to function, as explained in next section.

#### Updating ~/.dt/dt-site.xml file

1. Please open DataTorrent configuration file ~/.dt/dt-site.xml in any editor.
2. Identify following XML snippet,

<!-- Twitter demo -->

<property>

<name>dt.operator.TweetSampler.consumerKey</name>

<value>my API Key</value>

</property>

<property>

<name>dt.operator.TweetSampler.consumerSecret</name>

<value>my API Key Secret</value>

</property>

<property>

<name>dt.operator.TweetSampler.accessToken</name>

<value>my Access Token </value>

</property>

<property>

<name>dt.operator.TweetSampler.accessTokenSecret</name>

<value>my Access Token Secret</value>

</property>

1. In the above XML snippet replace values for keys as follows,

API Key - Value for dt.operator.TweetSampler.consumerKey

API Secret - Value for dt.operator.TweetSampler.consumerSecret

Access Token - Value for dt.operator.TweetSampler.accessToken

Access Token Secret - Value for dt.operator.TweetSampler.accessTokenSecret

## 

## How to Launch Twitter Demo application (dtcli)

The demo applications jar contains the application. On launching the jar from “dtcli” the list of applications packaged in the jar appears as shown in the screenshot below. The entry that is shown for each application is the path of the application class unless an alias was specified for the class in “dt-site.xml” in which case the alias is shown. An integer Id is also shown for each application. The Twitter Application Id appears to be Id “6” in this case as can be seen in screenshot corresponding to the application class. Please specify this Id “6” on the “dtcli” ” command shell to launch the application, as can be seen towards the end in the screenshot.

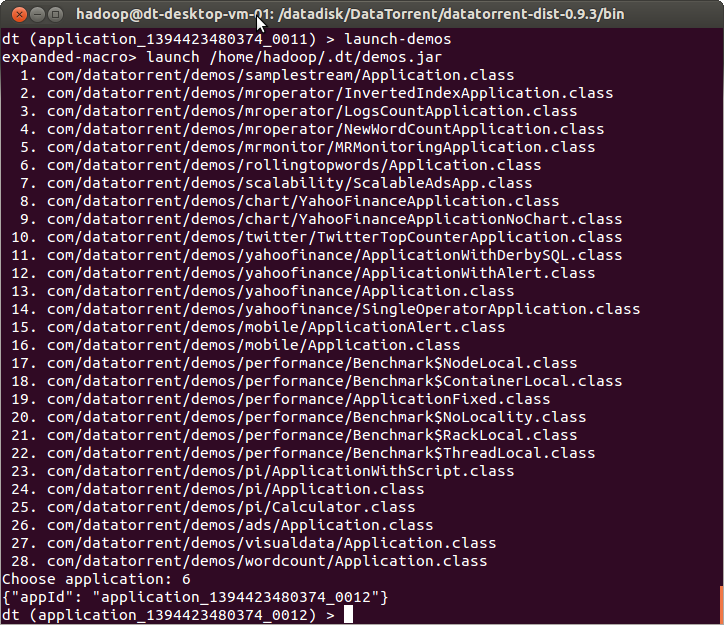


Figure Specify Application Id for Twitter Demo on dtcli

## How to monitor the Application

Application launched through “dtcli” can be monitored through two mechanisms,

1. Using “dtcli” command shell
2. Using DataTorrent Gateway Console

### Monitoring the application using dtcli

Twitter Demo application can be monitored through the “dtcli” command shell. Following are some of the useful commands that can be used to monitor as well as control the state of the application,

1. get-app-info – Print details for the selected application, details such as – total allocated memory, number of allocated containers, total numbers of tuples processed etc.
2. show-logical-plan – Print details about the logical plan for the selected application
3. show-physical-plan - Print details about the physical plan for the selected application
4. kill-app – Kill this application

For further details about various available commands please refer to [Operations and Installation Guide](https://www.datatorrent.com/docs/guides/OperationandInstallationGuide.html).

### Monitoring the application using DataTorrent Gateway Console

Twitter Demo application can now be monitored through DataTorrent Gateway Console. The launched application gets listed in the “*Application List*” widget as shown below,

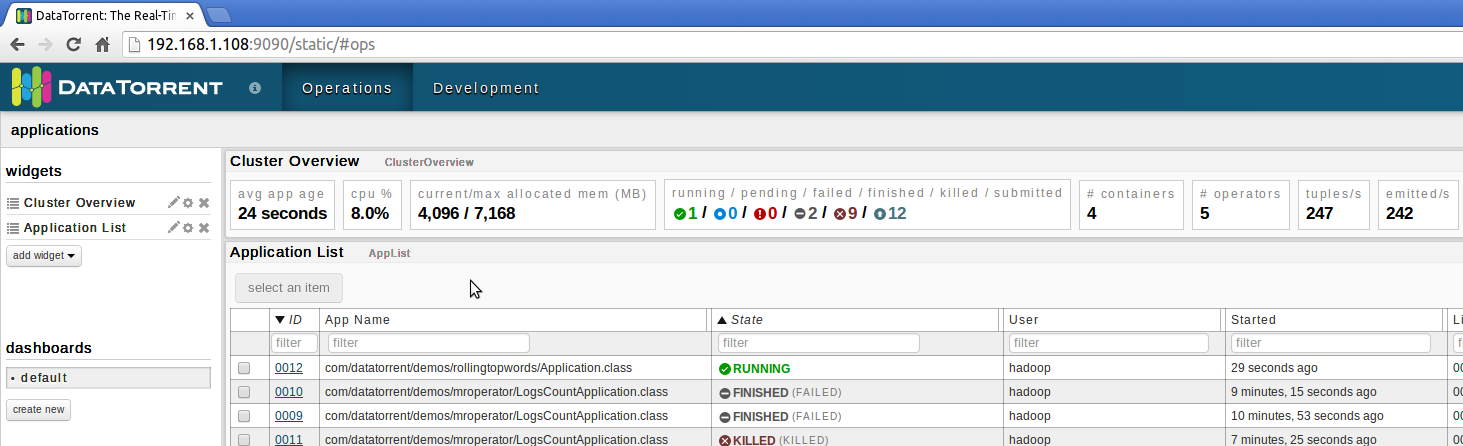


Figure Monitoring Application

On clicking the Application Id link (under the Id column), you can see the application state and further details such as -

1. Application Id and Version information
2. Application State (Running/Finished etc.)
3. Total number of Tuples Processed
4. Total number of Tuples Emitted
5. Latency (milliseconds)
6. Logical Directed Acyclic Graph (DAG)

Please refer to screenshot below that displays the above details,

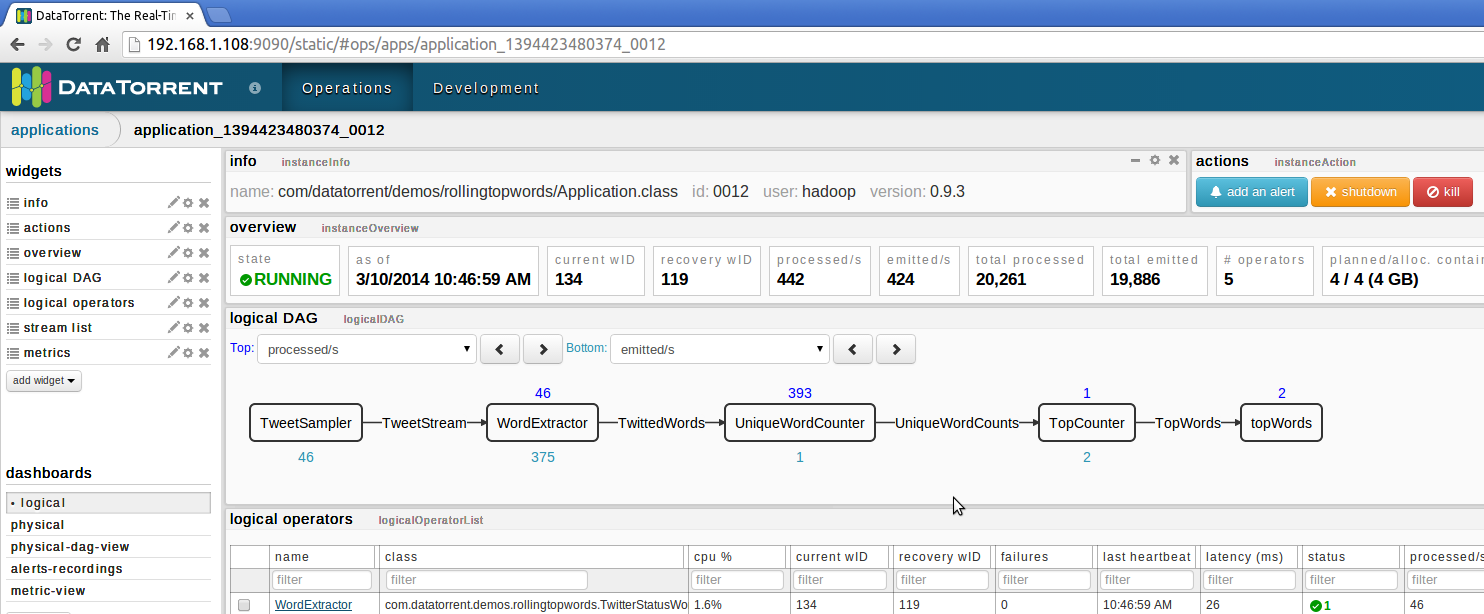


Figure Twitter Demo Application Details

## How to view Twitter Rolling Top Word Counts

As the Rolling Top Words application is running, it generates the counts for unique words. It prints these counts on the stdout console through “console” Operator. These results get redirected to log files that are generated for each running application. Following sections explain how to view these generated log files can be viewed.

### Looking at the Physical Dashboard

Click on the “physical” dashboard, under “dashboards” pane in the left hand bottom pane as seen in the Figure 5 above. This will display the screen that contains a list of “physical operators” (Figure 6 below). This list displays all the operator instances along with Container Ids and the Hadoop nodes where the container process is running.

This list contains the “console” operator which is responsible for displaying the application results. The “console” operator prints the results into the log file that gets generated on the same physical Hadoop node where the “container” process is running.

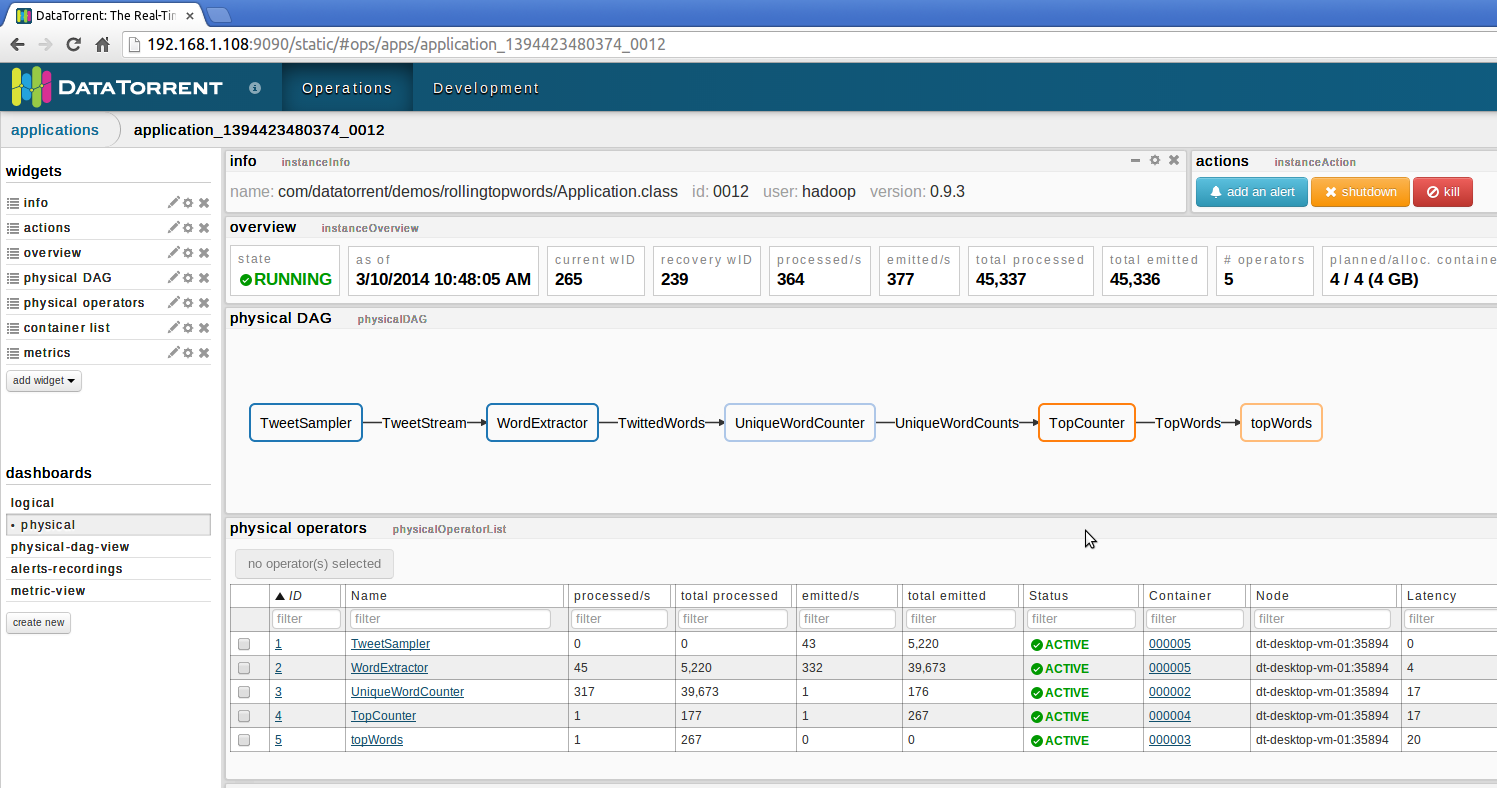


Figure 6 Physical Dashboard - Physical Operators List

### Looking at Container details

In order to get to the log files, we need to get to “container” details by clicking on the container Id (in this case “000003”) for the “console” operator in the “physical operators” list in above Figure 6. That will bring up the following screen as shown in Figure 7 below,

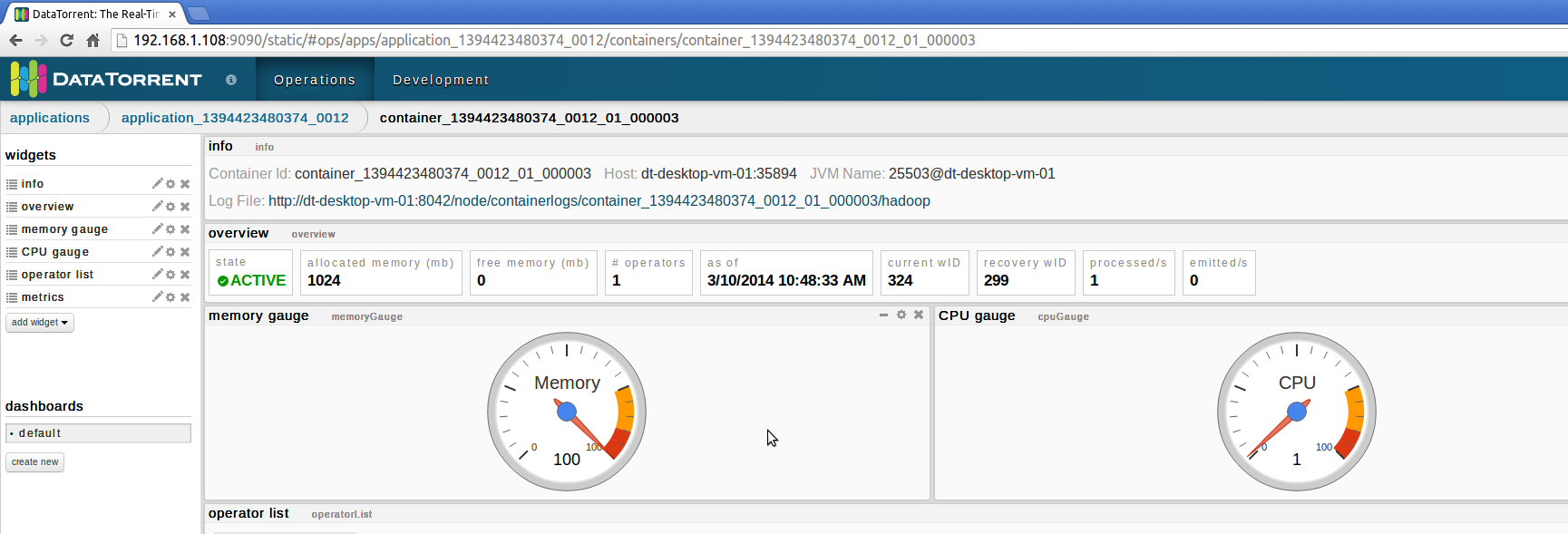


Figure 7 Container Overview

### Looking at the generated Log File for displayed results

Click on the “Log File” link in the “info” widget on the screen displayed in Figure 7 above. That would display the screen that displays names of three log files,

1. DT.log
2. stderr
3. stdout

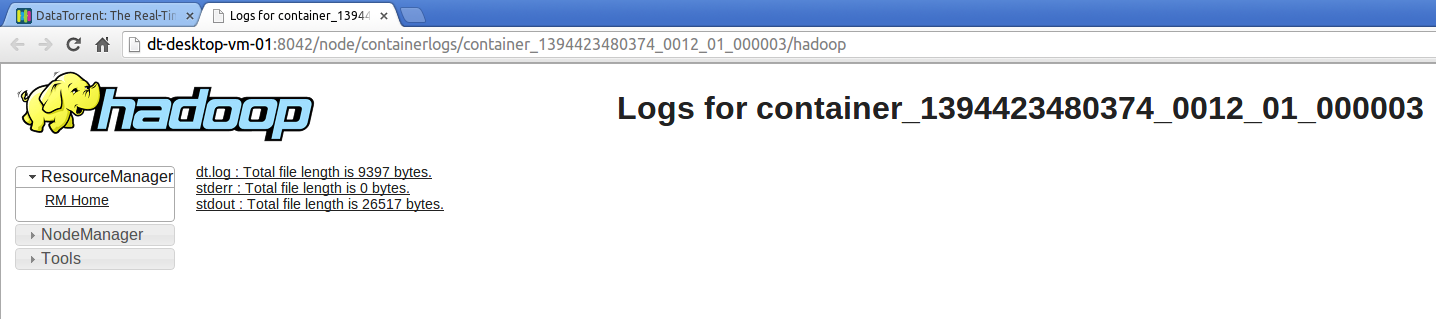


Figure 8 Container Log Files

Click on the “stdout” link as seen in the above Figure 8. That should bring a screen that looks like following screenshot,

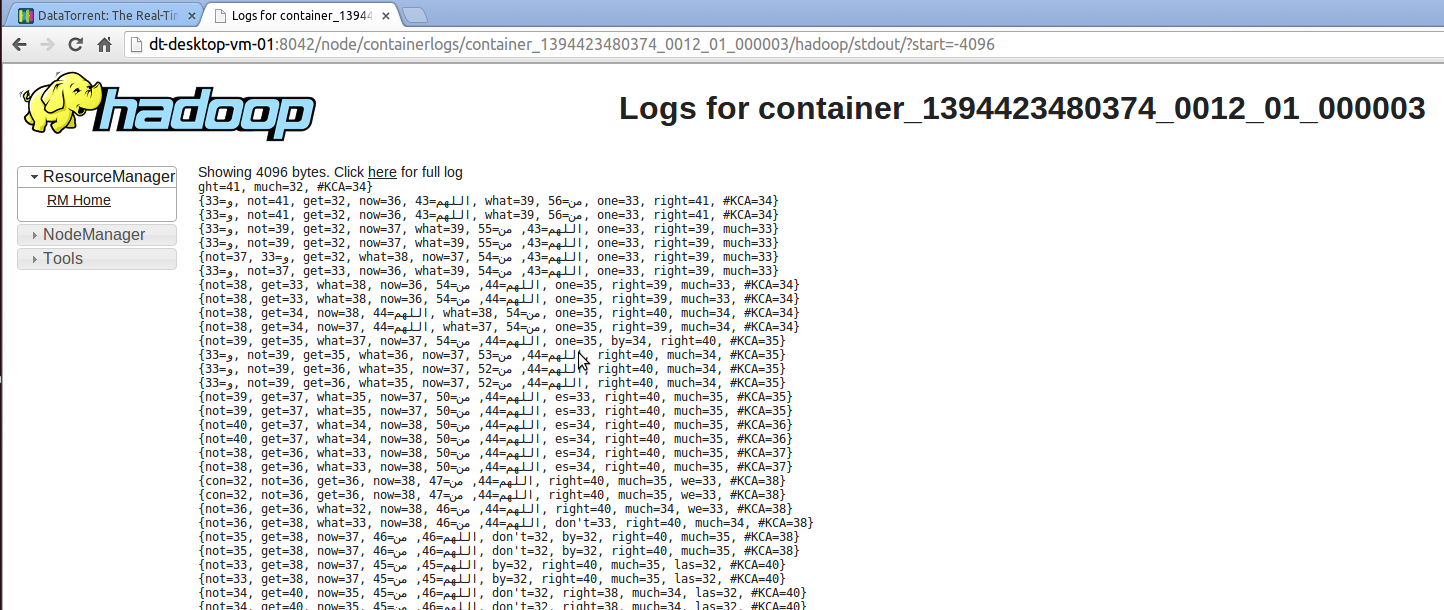


Figure 9 Container stdout log file

You can click on the “here” link to view the entire log file and see the generated results.

# Glossary of Terms used

###### Apache Hadoop

Apache Hadoop is the de-facto distributed operating system, details of which can be found on the web at - <http://apache.hadoop.org/>

###### Demo Application

The real-time stream processing application packaged with platform binary. The demo application can be launched in the standalone mode or on the Hadoop cluster to understand more about DataTorrent platform and its features.

###### Streaming Window

A streaming window is a slice of time that encapsulates a set of tuples. The collection of these tuples constitutes a window data set, which is also called as an atomic micro-batch.

###### Sliding Application Window

Sliding window is computation that requires previous N streaming windows. After each streaming window the Nth past window is dropped and the new window is added to the computation.

###### Directed Acyclic Graph (DAG)

It is the logical representation of real-time stream processing application. The computational units within DAG are called as operators and the data-flow edges are called as data streams.

###### Operator

An entity that holds a computational logic to process the data tuples. It is part of a real-time stream processing application. The Operator computational logic gets executed inside a container.

###### Port

Each operator can have ports on which it can receive input data tuples and also output processed data tuples.

###### Stream

A stream consists of data tuples that flow from one port of an operator to another.

###### Logical Plan or DAG

Logical Plan is the DOM (data object model) created as operators and streams are added to the DAG. It is identical to “Directed Acyclic Graph”.

###### Data Tuples Processed

Number of data objects processed by a real-time stream processing applications

###### Data Tuples Emitted

Number of data objects emitted after the processing is complete by a real-time stream processing applications