Computer Science & Information Systems

**Real Time Analytics / Stream Processing & Analytics**

**Apache Storm Lab Sheet 4**

**Storm – Kafka Integration**

1. Objective:

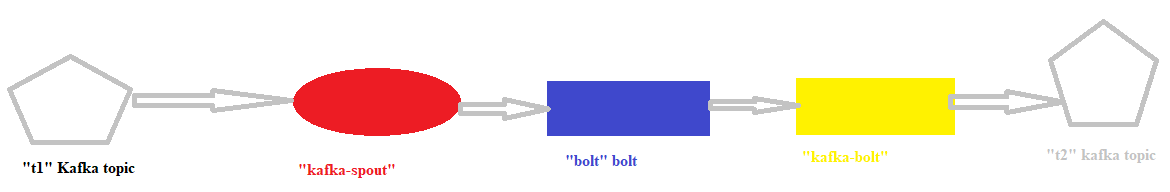
Students should be able to

1. Get hands-on experience of Storm – Kafka integration using the Java programs

Kafka and Storm naturally complement each other, and their powerful cooperation enables real-time streaming analytics for fast-moving big data. Kafka and Storm integration is to make easier for developers to ingest and publish data streams from Storm topologies.

A spout is a source of streams. For example, a spout may read tuples off a Kafka Topic and emit them as a stream. A bolt consumes input streams, process and possibly emits new streams. Bolts can do anything from running functions, filtering tuples, do streaming aggregations, streaming joins, talk to databases, and more. Each node in a Storm topology executes in parallel. A topology runs indefinitely until you terminate it. Storm will automatically reassign any failed tasks. Additionally, Storm guarantees that there will be no data loss, even if the machines go down and messages are dropped.

Let us go through the Kafka-Storm integration API’s in detail. This lab sheet will introduce students with usage of Storm Topology with Java. The application that will be taken as example is word processing application. The topology will consist of a Kafka spout and two bolts – one of them being Kafka bolt. The spout will receive the words from the Kafka Topic. Its responsibility is to provide a word received from Kafka topic to the next bolt in topology. The first bolt is the topology will be responsible for enriching the words by adding some extra characters in it and forwarding the words to the next Kafka bolt. The Kafka bolt will output the enriched word to the Kafka topic. The topology of this Kafka – Storm integration application can be visualized as shown below.

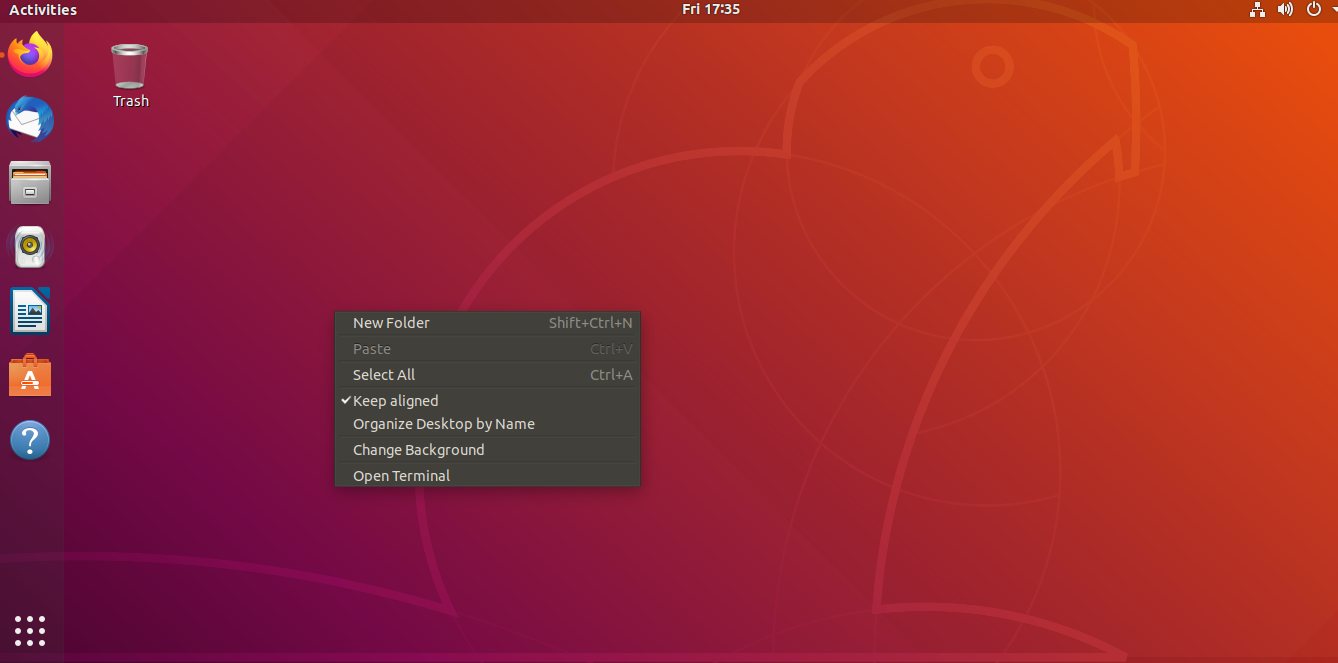


1. Steps to be performed:

Note - It’s assumed that student has made a slot reservation using the slot booking interface where Apache Storm framework was selected. The details of the Apache Strom systems to be used is received through an email. If not, please contact the administrators for the same.

Also it’s assumed that students are aware of the process of logging into these virtual machines. If not, then get access to the user manual maintained for the usage of remote lab setup.

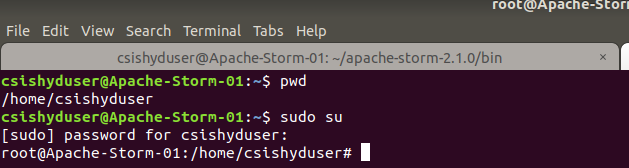
1. Open the terminal by right clicking on the desktop of the virtual machine.



1. Login as sudo user.

>>> sudo su

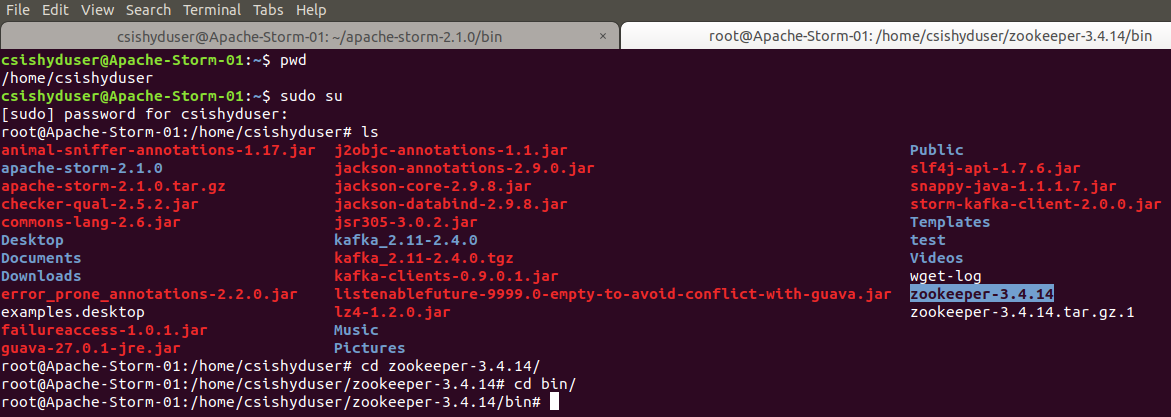
Provide the password provided in the email received from BITS remote lab team.



1. Look at the current working directory using the “pwd” command. Then change the directory to the Zookeepers bin directory.

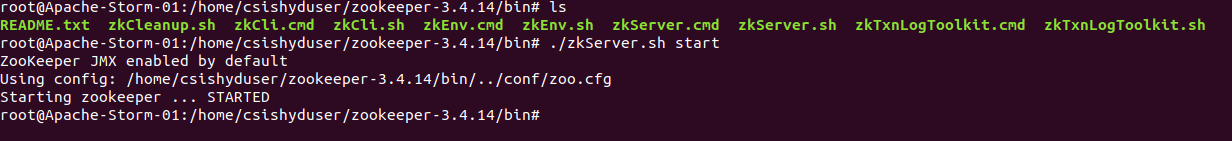
>>> pwd

>>> cd zookeeper-3.4.14/bin/



1. Start the zookeeper.

>>> ./zkServer.sh start



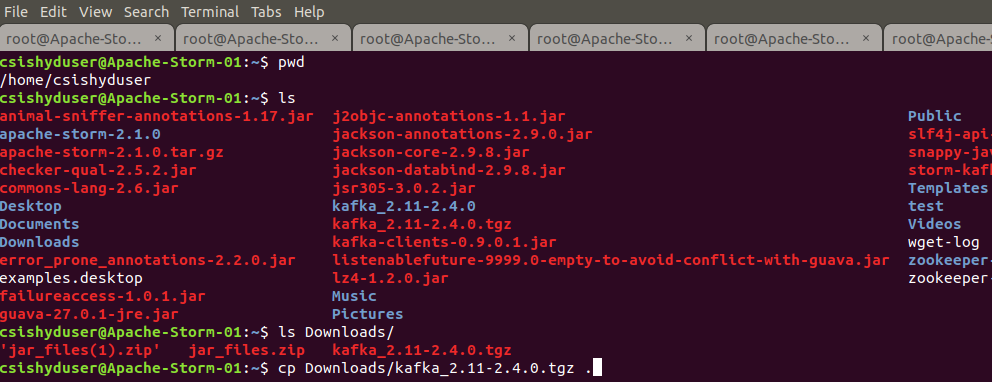
1. Open Firefox or any other browser application. Visit the following link in it to download the Apache Kafka tarball.

<https://kafka.apache.org/downloads>



1. The tarball will get download into “Downloads” directory. Copy it into the home directory or the directory into which you want to do its installation.

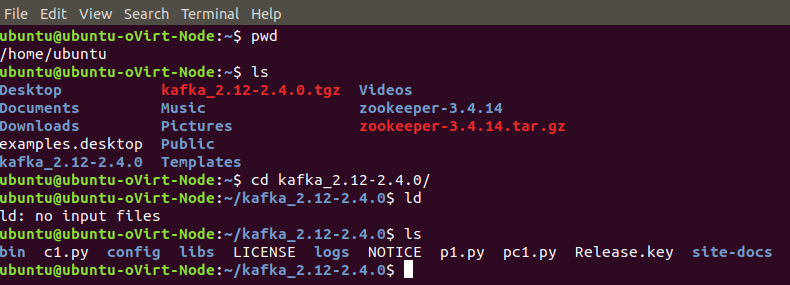
>>> cp Downloads/kafka\_2.11-2.4.0.tgz .



1. Unzip the tarball in the current directory.

>>> tar -xf kafka\_2.11-2.4.0.tgz

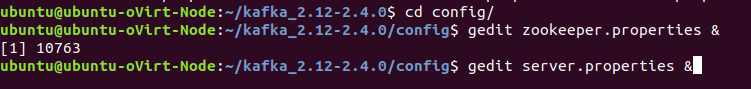
1. Change to the Kafka installation directory using the command and have a look at the files present in the directory.



1. Change to the “config” directory present in the Kafka installation directory and have a look at the Kafka server properties files in it.

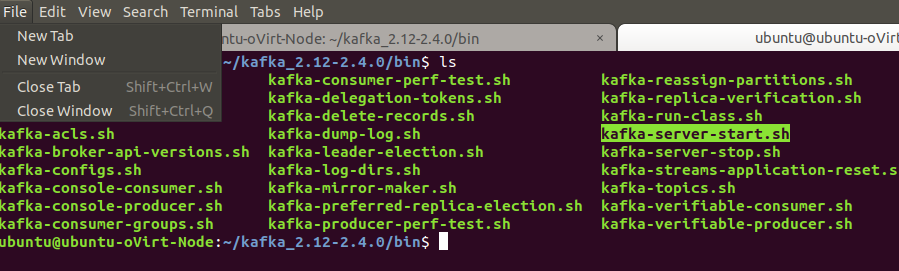
>>> cd config

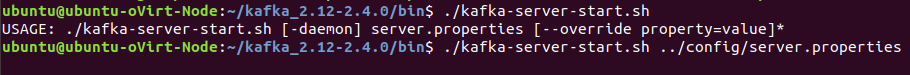
>>>gedit server.properties&



1. Open up another terminal and Change to the “bin” directory present in the Kafka installation directory and have a look at the script files present in it. Will be using the following scripts to start and stop the Kafka server

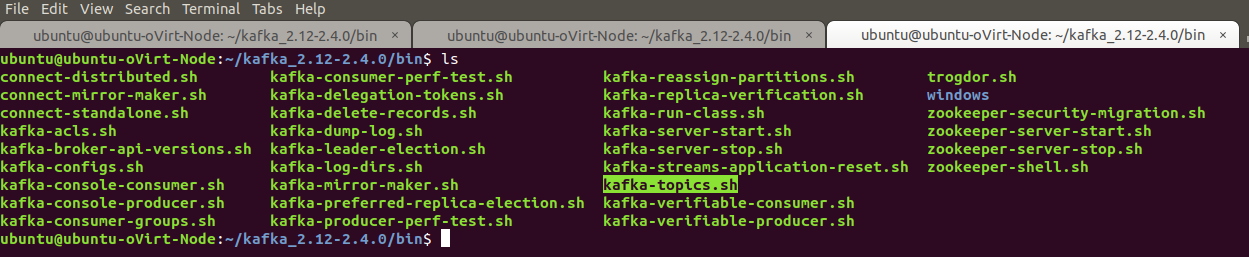
* kafka-server-start.sh
* kafka-server-stop.sh





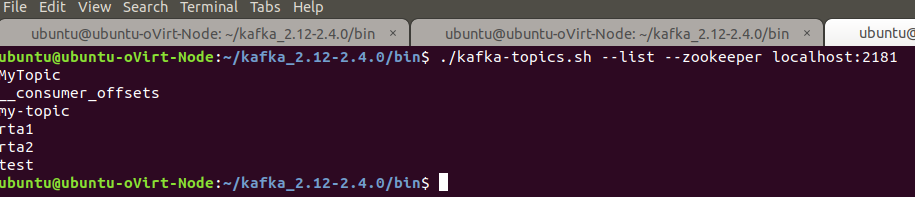
1. Open up another terminal and Change to the “bin” directory present in the Kafka installation directory and have a look at the script files present in it. Will be using the following script to deal with the Kafka topics.

* kafka-topics.sh



1. Use the –list option in it to list the topics present in the Kafka setup. Note – it will be empty for you as you don’t have created any topics as such but if the Kafka setup is shared you may see the topics created in earlier usages of the system.

* ./kafka-topics.sh --list --zookeeper localhost:2181



1. Let’s try to create a Kafka topic named “t1” and “t2” using the –create option.

* ./kafka-topics.sh --zookeeper localhost:2181 --create --topic t1 --replication-factor 1 --partitions 3
* ./kafka-topics.sh --zookeeper localhost:2181 --create --topic t2 --replication-factor 1 --partitions 3

1. List the Kafka topics again. Now the “t1” and “t2” should appear in the topics list.

* ./kafka-topics.sh --list --zookeeper localhost:2181

1. The Kafka distribution provides a command utility to send messages from the command line. It start up a terminal window where everything you type is sent to the Kafka topic. Kafka provides the utility kafka-console-producer.sh to send messages to a topic on the command line.

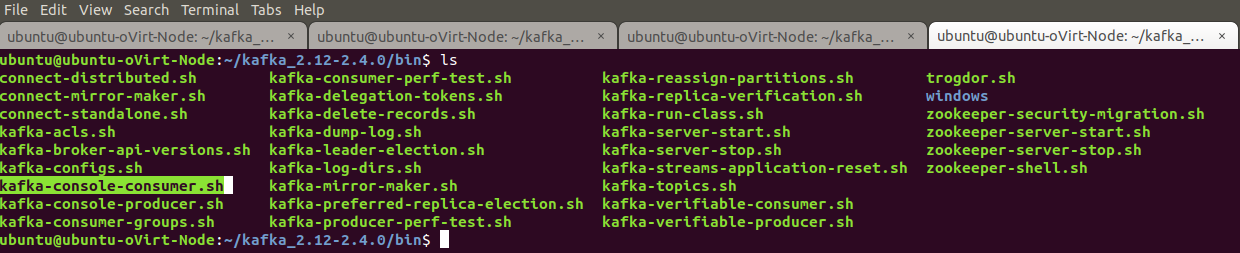
* ./kafka-console-producer.sh --topic t1 --broker-list localhost:9092



1. Try inserting some message in the command prompt provided by producer utility.

>>> ./kafka-console-producer.sh --broker-list localhost:9092 --topic t1

1. Open up another terminal and Change to the “bin” directory present in the Kafka installation directory and have a look at the script files present in it.



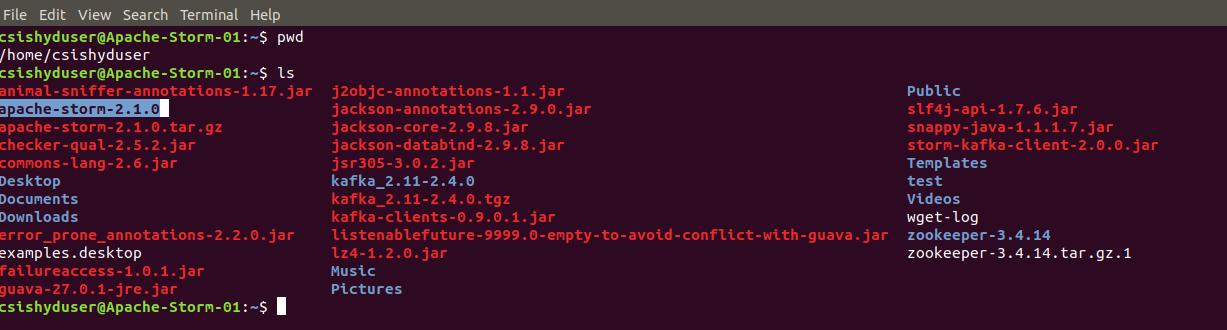
1. The Kafka distribution provides a command utility to see messages from the command line. It displays the messages in various modes. Kafka provides the utility kafka-console-consumer.sh which is located at /bin/kafka-console-producer.sh to receive messages from a topic on the command line.

* ./kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic t2

1. Open another terminal. Look at the current working directory using the “pwd” command. Then change the directory to the Storms directory.

>>> pwd

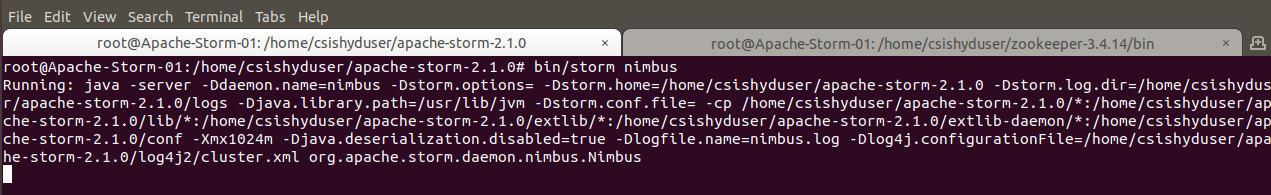
>>> cd apache-storm-2.1.0/



1. Start the nimbus node using the storm command.

>>> bin/strom nimbus

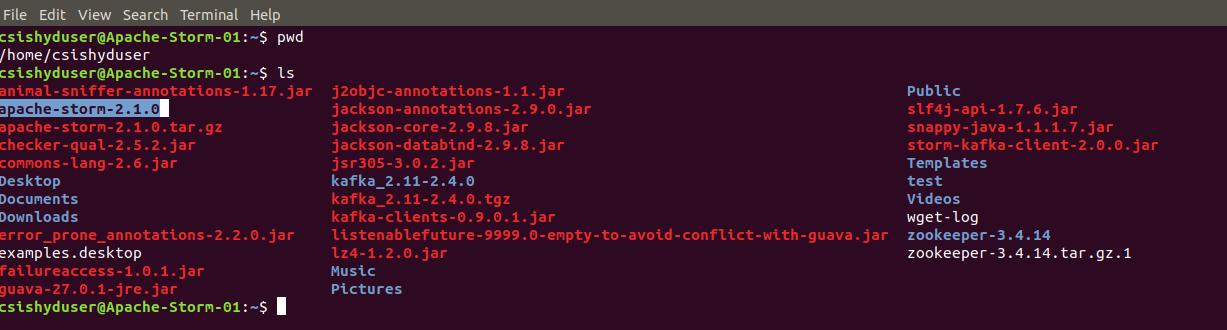
Note – if the command fails, then login as sudo su and then try again.



1. Open another terminal. Look at the current working directory using the “pwd” command. Then change the directory to the Storms directory.

>>> pwd

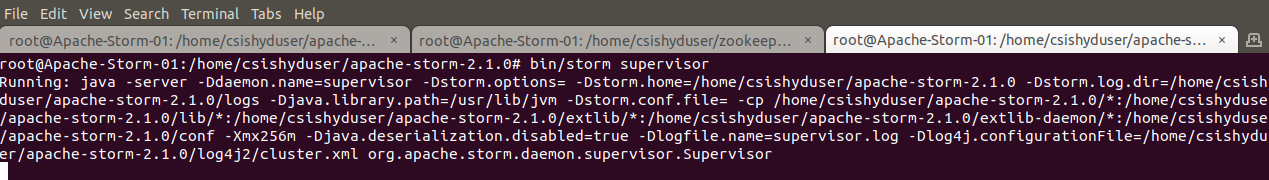
>>> cd apache-storm-2.1.0/



1. Start the supervisor node using the storm command.

>>> bin/strom suporvisor

Note – if the command fails, then login as sudo su and then try again.



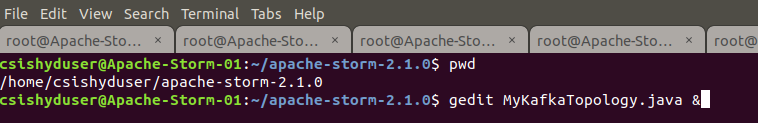
1. Open another terminal. Look at the current working directory using the “pwd” command. Then change the directory to the Storms directory.

>>> pwd

>>> cd apache-storm-2.1.0/

1. Open up gedit editor for writing the Java code.

>>> gedit MyKafkaTopology.java&



1. Copy paste the content of attached MyKafkaTopology.java file into the file opened in the geditor.
2. Repeat the step X and Y for two other java files namely

* MyKafkaBolt.java

1. Compile the MyKafkaTopology.java class which has the topology definition.

>>> javac -cp .:lib/\* MyKafkaTopology.java

Note – Make sure following jars are present inside the “lib” directory of your storm installation. (Attached with this lab).

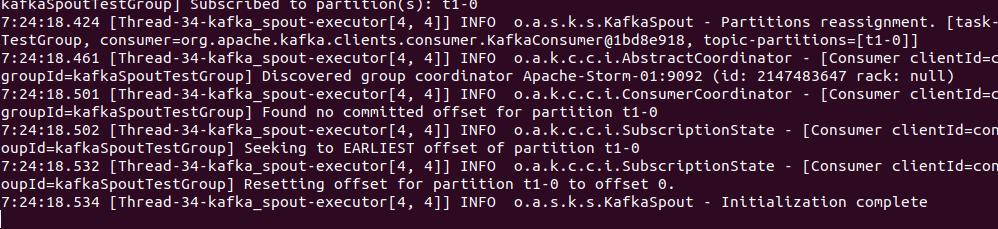
* kafka\_2.10-0.10.2.2.jar
* kafka-clients-1.1.0.jar
* storm-kafka-1.0.1.jar
* storm-kafka-client-2.0.0.jar

1. Run the MyKafkaTopology Storm application and observe the output.

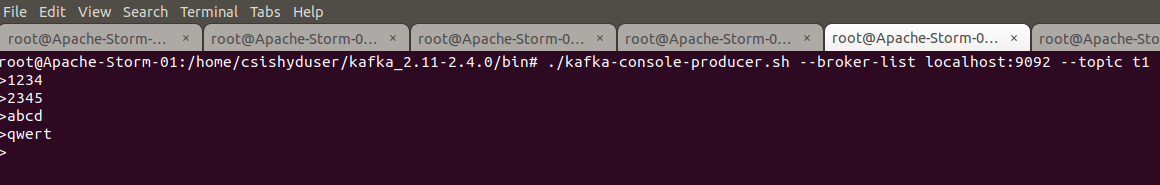
>>> java -cp .:lib/\*:/home/csishyduser/kafka\_2.11-2.4.0/libs/\* MyKafkaTopology

Note – The classpath also needs to include the “libs” directory of your kafka-installation.

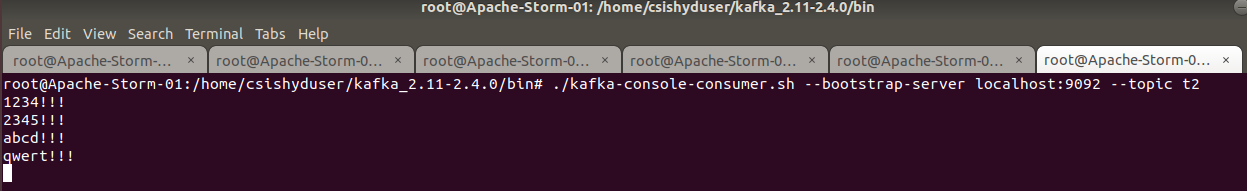
1. In the output you must be seeing the lines shown below which shows that program is executing.



1. Now enter the messages into the console producer screen for topic “t1”



1. You can see the enriched messages coming in the console consumer screen opened for topic “t2”.



1. Outputs/Results:

Students should be able to write a Storm application

* To read the tuples / records from the Kafka topic through Spout
* To do data processing on the tuples through the Bolt logic
* To execute the Storm Topology on the local cluster
* To write enriched output back to Kafka Topic

1. Observations:

Students carefully needs to observe the steps followed for

* Setting up Kafka Cluster and starting it
* Creating the Kafka topic and writing messages into it
* Building the topology with Spout and Bolt and executing it on cluster
* Consuming the enriched outcomes from Kafka Topics

1. References:
2. Storm Documentation
3. [Kafka Documentation](https://kafka.apache.org/documentation/)
4. [Data-flair training tutorial](https://data-flair.training/blogs/storm-kafka-integration/)