

Computer Science & Information Systems

Real Time Analytics / Stream Processing & Analytics

Apache Storm Lab Sheet 2

Simple Storm Application

1. Objective:

Students should be able to

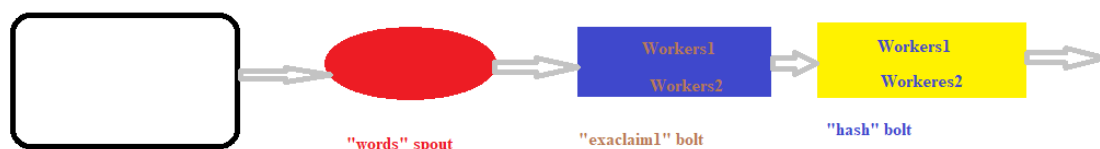
- A. Get familiarity with the working of Storm Application
- B. Get hands-on experience writing Java program for Streams processing using Storm Topology consisting of Spout and Bolts

Apache Storm is a free and open source distributed real-time computation system. Apache Storm makes it easy to reliably process unbounded streams of data, doing for real-time processing what Hadoop did for batch processing. Apache Storm is simple, can be used with any programming language, and is a lot of fun to use! Apache Storm has many use cases: real-time analytics, online machine learning, continuous computation, distributed RPC, ETL, and more. Apache Storm is fast: a benchmark clocked it at over a million tuples processed per second per node. It is scalable, fault-tolerant, guarantees your data will be processed, and is easy to set up and operate. Apache Storm integrates with the queueing and database technologies you already use. An Apache Storm topology consumes streams of data and processes those streams in arbitrarily complex ways, repartitioning the streams between each stage of the computation however needed.

The logic for a real-time application is packaged into a Storm topology. A Storm topology is analogous to a MapReduce job. One key difference is that a MapReduce job eventually finishes, whereas a topology runs forever (or until you kill it, of course). A topology is a graph of spouts and bolts that are connected with stream groupings. The stream is the core abstraction in Storm. A stream is an unbounded sequence of tuples that is processed and created in parallel in a distributed fashion. Streams are defined with a schema that names the fields in the stream's tuples. By default, tuples can contain integers, longs, shorts, bytes, strings, doubles, floats, booleans, and byte arrays. You can also define your own serializers so that custom types can be used natively within tuples.

A spout is a source of streams in a topology. Generally spouts will read tuples from an external source and emit them into the topology (e.g. a Kestrel queue or the Twitter API). Spouts can either be reliable or unreliable. A reliable spout is capable of replaying a tuple if it failed to be processed by Storm, whereas an unreliable spout forgets about the tuple as soon as it is emitted. All processing in topologies is done in bolts. Bolts can do anything from filtering, functions, aggregations, joins, talking to databases, and more. Bolts can do simple stream transformations. Doing complex stream transformations often requires multiple steps and thus multiple bolts. Part of defining a topology is specifying for each bolt which streams it should receive as input. A stream grouping defines how that stream should be partitioned among the bolt's tasks.

This lab sheet will introduce students with usage of Storm Topology API with Java. We are considering a very simple topology that will just enhance the incoming tuples. The topology will consist of a spout and two bolts. The spout will serve as abstraction for the real world. Its responsibility is to provide a word at random from the given list of words. The assumption made here is that the words are the tuples received from the real world. Spout is helping us to accept them from external world and making it available for further processing. The first bolt is the topology will be responsible for enhancing the word by adding three “!” mark and forwarding the processed tuple to the next bolt. The next/last bolt will also repeat the same operation on the incoming tuple. As this bolt is another instance of the bolt used in the previous step, the incoming tuple will again be enhancing the word by adding three “!” mark and the enhanced tuple will be emitted on the console. The topology of this application can be visualized as shown below.

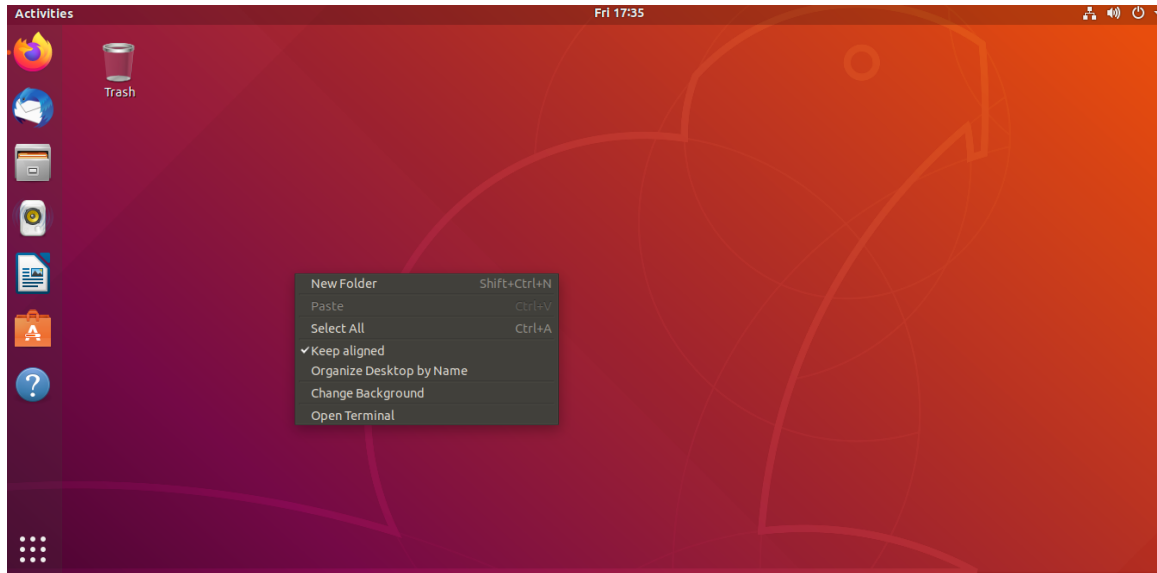


2. Steps to be performed:

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 Storm 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.

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



- B. Login as sudo user.

```
>>> sudo su
```

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

```
root@Apache-Storm-01:~$  
File Edit View Search Terminal Tabs Help  
csishyduser@Apache-Storm-01: ~/apache-storm-2.1.0/bin  
csishyduser@Apache-Storm-01:~$ pwd  
/home/csishyduser  
csishyduser@Apache-Storm-01:~$ sudo su  
[sudo] password for csishyduser:  
root@Apache-Storm-01:/home/csishyduser#
```

- C. 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/
```

```
File Edit View Search Terminal Tabs Help
csishyduser@Apache-Storm-01: ~/apache-storm-2.1.0/bin x root@Apache-Storm-01: /home/csishyduser/zookeeper-3.4.14/bin
csishyduser@Apache-Storm-01:~$ pwd
/home/csishyduser
csishyduser@Apache-Storm-01:~$ sudo su
[sudo] password for csishyduser:
root@Apache-Storm-01:/home/csishyduser# ls
animal-sniffer-annotations-1.17.jar  j2objc-annotations-1.1.jar  Public
apache-storm-2.1.0                    jackson-annotations-2.9.0.jar  slf4j-api-1.7.6.jar
apache-storm-2.1.0.tar.gz             jackson-core-2.9.8.jar       snappy-java-1.1.1.7.jar
checker-qual-2.5.2.jar               jackson-databind-2.9.8.jar   storm-kafka-client-2.0.0.jar
commons-lang-2.6.jar                 jsr305-3.0.2.jar            Templates
Desktop                               kafka_2.11-2.4.0             test
Documents                             kafka_2.11-2.4.0.tgz         Videos
Downloads                             kafka-clients-0.9.0.1.jar    wget-log
error_prone_annotations-2.2.0.jar     listenablefuture-9999.0-empty-to-avoid-conflict-with-guava.jar  zookeeper-3.4.14
examples.desktop                     lz4-1.2.0.jar               zookeeper-3.4.14.tar.gz.1
failureaccess-1.0.1.jar              Music
guava-27.0.1-jre.jar                 Pictures
root@Apache-Storm-01:/home/csishyduser# cd zookeeper-3.4.14/
root@Apache-Storm-01:/home/csishyduser/zookeeper-3.4.14# cd bin/
root@Apache-Storm-01:/home/csishyduser/zookeeper-3.4.14/bin#
```

D. Start the zookeeper.

>>> ./zkServer.sh start

```
root@Apache-Storm-01:/home/csishyduser/zookeeper-3.4.14/bin# ls
README.txt zkCleanup.sh zkCli.cmd zkEnv.cmd zkEnv.sh zkServer.cmd zkServer.sh zkTxnLogToolkit.cmd zkTxnLogToolkit.sh
root@Apache-Storm-01:/home/csishyduser/zookeeper-3.4.14/bin# ./zkServer.sh start
ZooKeeper JMX enabled by default
Using config: /home/csishyduser/zookeeper-3.4.14/bin/./conf/zoo.cfg
Starting zookeeper ... STARTED
root@Apache-Storm-01:/home/csishyduser/zookeeper-3.4.14/bin#
```

E. 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/

```
File Edit View Search Terminal Help
csishyduser@Apache-Storm-01:~$ pwd
/home/csishyduser
csishyduser@Apache-Storm-01:~$ ls
animal-sniffer-annotations-1.17.jar  j2objc-annotations-1.1.jar  Public
apache-storm-2.1.0                    jackson-annotations-2.9.0.jar  slf4j-api-1.7.6.jar
apache-storm-2.1.0.tar.gz             jackson-core-2.9.8.jar       snappy-java-1.1.1.7.jar
checker-qual-2.5.2.jar               jackson-databind-2.9.8.jar   storm-kafka-client-2.0.0.jar
commons-lang-2.6.jar                 jsr305-3.0.2.jar            Templates
Desktop                               kafka_2.11-2.4.0             test
Documents                             kafka_2.11-2.4.0.tgz         Videos
Downloads                             kafka-clients-0.9.0.1.jar    wget-log
error_prone_annotations-2.2.0.jar     listenablefuture-9999.0-empty-to-avoid-conflict-with-guava.jar  zookeeper-3.4.14
examples.desktop                     lz4-1.2.0.jar               zookeeper-3.4.14.tar.gz.1
failureaccess-1.0.1.jar              Music
guava-27.0.1-jre.jar                 Pictures
csishyduser@Apache-Storm-01:~$
```

F. Start the nimbus node using the storm command.

>>> bin/storm nimbus

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

```
File Edit View Search Terminal Tabs Help
root@Apache-Storm-01:/home/csishyduser/apache-storm-2.1.0
root@Apache-Storm-01:/home/csishyduser/zookeeper-3.4.14/bin
root@Apache-Storm-01:/home/csishyduser/apache-storm-2.1.0# bin/storm nimbus
Running: java -server -Ddaemon.name=nimbus -Dstorm.options= -Dstorm.home=/home/csishyduser/apache-storm-2.1.0 -Dstorm.log.dir=/home/csishyduser/apache-storm-2.1.0/logs -Djava.library.path=/usr/lib/jvm -Dstorm.conf.file= -cp /home/csishyduser/apache-storm-2.1.0/lib/*:/home/csishyduser/apache-storm-2.1.0/extlib/*:/home/csishyduser/apache-storm-2.1.0/extlib-daemon/*:/home/csishyduser/apache-storm-2.1.0/conf -Xmx1024m -Djava.deserialization.disabled=true -Dlogfile.name=nimbus.log -Dlog4j.configurationFile=/home/csishyduser/apache-storm-2.1.0/log4j2/ccluster.xml org.apache.storm.daemon.nimbus.Nimbus
```

G. 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/
```

```
File Edit View Search Terminal Help
csishyduser@Apache-Storm-01:~$ pwd
/home/csishyduser
csishyduser@Apache-Storm-01:~$ ls
animal-sniffer-annotations-1.17.jar  jackson-annotations-2.9.0.jar  Public
apache-storm-2.1.0  jackson-core-2.9.8.jar  slf4j-api-1.7.6.jar
apache-storm-2.1.0.tar.gz  jackson-databind-2.9.8.jar  snappy-java-1.1.1.7.jar
checker-qual-2.5.2.jar  jsr305-3.0.2.jar  storm-kafka-client-2.0.0.jar
commons-lang-2.6.jar  kafka_2.11-2.4.0  Templates
Desktop  kafka_2.11-2.4.0.tgz  test
Documents  kafka-clients-0.9.0.1.jar  Videos
Downloads  listenablefuture-9999.0-empty-to-avoid-conflict-with-guava.jar  wget-log
error_prone_annotations-2.2.0.jar  lz4-1.2.0.jar  zookeeper-3.4.14
examples.desktop  Music  zookeeper-3.4.14.tar.gz.1
failureaccess-1.0.1.jar  Pictures
guava-27.0.1-jre.jar
csishyduser@Apache-Storm-01:~$
```

H. Start the supervisor node using the storm command.

```
>>> bin/storm supervisor
```

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

```
File Edit View Search Terminal Tabs Help
root@Apache-Storm-01:/home/csishyduser/apache-storm-2.1.0# bin/storm supervisor
Running: java -server -Ddaemon.name=supervisor -Dstorm.options= -Dstorm.home=/home/csishyduser/apache-storm-2.1.0 -Dstorm.log.dir=/home/csishyduser/apache-storm-2.1.0/logs -Djava.library.path=/usr/lib/jvm -Dstorm.conf.file= -cp /home/csishyduser/apache-storm-2.1.0/lib/*:/home/csishyduser/apache-storm-2.1.0/extlib/*:/home/csishyduser/apache-storm-2.1.0/extlib-daemon/*:/home/csishyduser/apache-storm-2.1.0/conf -Xmx256m -Djava.deserialization.disabled=true -Dlogfile.name=supervisor.log -Dlog4j.configurationFile=/home/csishyduser/apache-storm-2.1.0/log4j2/ccluster.xml org.apache.storm.daemon.supervisor.Supervisor
```

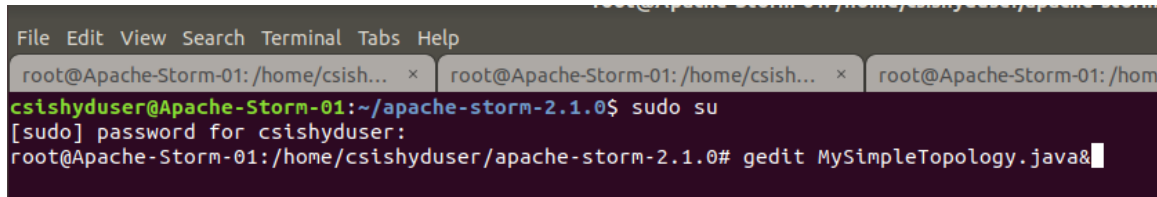
I. Open another terminal. Look at the current working directory using the “pwd” command. Then change the directory to the Storms directory.

```
>>> pwd
```

```
>>> cd apache-storm-2.1.0/
```

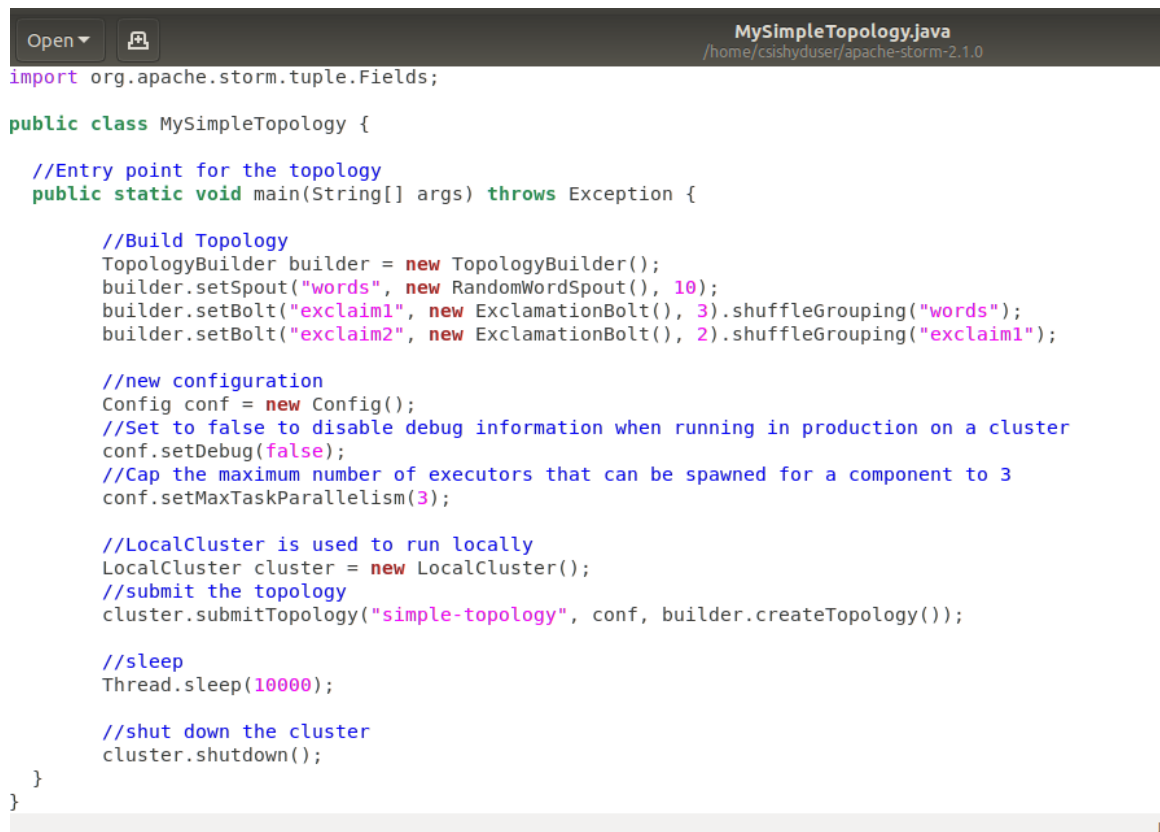
J. Open up gedit editor for writing the Java code.

>>> gedit MySimpleTopology.java&



```
File Edit View Search Terminal Tabs Help
root@Apache-Storm-01: /home/csish... x root@Apache-Storm-01: /home/csish... x root@Apache-Storm-01: /home
csishyduser@Apache-Storm-01:~/apache-storm-2.1.0$ sudo su
[sudo] password for csishyduser:
root@Apache-Storm-01: /home/csishyduser/apache-storm-2.1.0# gedit MySimpleTopology.java&
```

K. Copy paste the content of attached MySimpleTopology.java file into the file opened in the geditor.



```
Open [icon] MySimpleTopology.java
/home/csishyduser/apache-storm-2.1.0

import org.apache.storm.tuple.Fields;

public class MySimpleTopology {

    //Entry point for the topology
    public static void main(String[] args) throws Exception {

        //Build Topology
        TopologyBuilder builder = new TopologyBuilder();
        builder.setSpout("words", new RandomWordSpout(), 10);
        builder.setBolt("exclaim1", new ExclamationBolt(), 3).shuffleGrouping("words");
        builder.setBolt("exclaim2", new ExclamationBolt(), 2).shuffleGrouping("exclaim1");

        //new configuration
        Config conf = new Config();
        //Set to false to disable debug information when running in production on a cluster
        conf.setDebug(false);
        //Cap the maximum number of executors that can be spawned for a component to 3
        conf.setMaxTaskParallelism(3);

        //LocalCluster is used to run locally
        LocalCluster cluster = new LocalCluster();
        //submit the topology
        cluster.submitTopology("simple-topology", conf, builder.createTopology());

        //sleep
        Thread.sleep(10000);

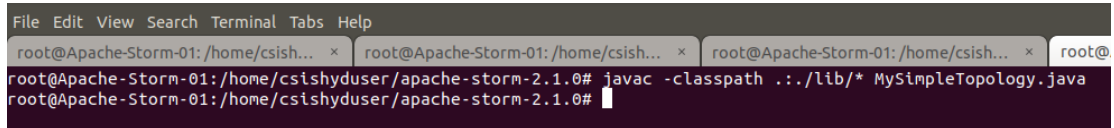
        //shut down the cluster
        cluster.shutdown();
    }
}
```

L. Repeat the step J and K for two other java files namely

- RandomWordSpout.java
- ExclamationBolt.java

M. Compile the MySimpleTopology.java class which has the topology definition.

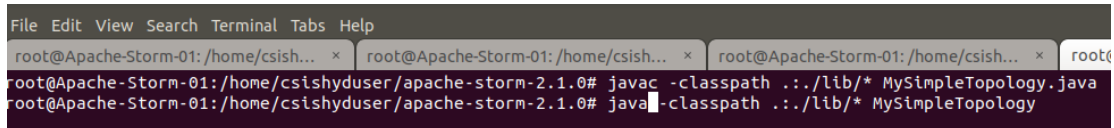
```
>>> javac -classpath ../lib/* MySimpleTopology.java
```



```
File Edit View Search Terminal Tabs Help
root@Apache-Storm-01: /home/csish... x root@Apache-Storm-01: /home/csish... x root@Apache-Storm-01: /home/csish... x root@
root@Apache-Storm-01: /home/csishyduser/apache-storm-2.1.0# javac -classpath ../lib/* MySimpleTopology.java
root@Apache-Storm-01: /home/csishyduser/apache-storm-2.1.0#
```

N. Run the MySimpleTopology Storm application and observe the output.

```
>>> java -classpath ../lib/* MySimpleTopology
```



```
File Edit View Search Terminal Tabs Help
root@Apache-Storm-01: /home/csish... x root@Apache-Storm-01: /home/csish... x root@Apache-Storm-01: /home/csish... x root@
root@Apache-Storm-01: /home/csishyduser/apache-storm-2.1.0# java -classpath ../lib/* MySimpleTopology
root@Apache-Storm-01: /home/csishyduser/apache-storm-2.1.0#
```

1. In the output you must be seeing the lines shown below which shows that the random word is generated in the Spout and getting processed with the Bolt defined.

```
*****Emitting a enriched word ---->Hyderabad!!!
```

```
*****Emitting a enriched word ---->Hyderabad!!!!!!
```

```
*****Emitting a enriched word ---->Pilani!!!
```

```
*****Emitting a enriched word ---->Pilani!!!!!!
```

```
*****Emitting a enriched word ---->Goa!!!
```

```

8:08:21.044 [main] INFO o.a.s.s.o.a.z.ZooKeeper - Session: 0x100760894df000c closed
8:08:21.044 [main] INFO o.a.s.d.s.ReadClusterState - Setting Thread[SLOT_1027,5,main] assignment to null
8:08:21.045 [main] INFO o.a.s.d.s.ReadClusterState - Setting Thread[SLOT_1028,5,main] assignment to null
8:08:21.045 [main] INFO o.a.s.d.s.ReadClusterState - Setting Thread[SLOT_1029,5,main] assignment to null
8:08:21.045 [main] INFO o.a.s.d.s.ReadClusterState - Waiting for Thread[SLOT_1027,5,main] to be EMPTY, currently running
*****Emitting a enriched word ---->Hyderabad!!!
*****Emitting a enriched word ---->Hyderabad!!!!!!
*****Emitting a enriched word ---->Pilani!!!
*****Emitting a enriched word ---->Pilani!!!!!!
*****Emitting a enriched word ---->Goa!!!
*****Emitting a enriched word ---->Goa!!!!!!
*****Emitting a enriched word ---->Hyderabad!!!
*****Emitting a enriched word ---->Hyderabad!!!!!!
*****Emitting a enriched word ---->Dubai!!!
*****Emitting a enriched word ---->Dubai!!!!!!
*****Emitting a enriched word ---->Goa!!!
*****Emitting a enriched word ---->Goa!!!!!!
*****Emitting a enriched word ---->Goa!!!!!!
*****Emitting a enriched word ---->Goa!!!!!!

```

O. In order to run the code continuously comment out the following line in the MySimpleTopology.java file, recompile and execute the program to see the continuous output.

- cluster.shutdown();

3. Outputs/Results:

Students should be able to write a Storm application

- To read the tuples / records from the external sources through Spout
- To do some basic processing on the tuples through the Bolt logic
- To execute the Storm Topology on the local cluster

4. Observations:

Students carefully needs to observe the code written for the usage of Storm API for

- Building the Spout and data handling with it
- Writing the Bolt and process the data with it
- Building the topology with Spout and Bolt and executing it on cluster

5. References:

- a. [Storm Documentation](#)



- b. [Strom Tutorial](#)