

Java-Success.com

Prepare to fast-track, choose & go places with 800+ Java & Big Data Q&As with lots of code & diagrams.

[Home](#) [Why? ▾](#) [300+ Java FAQs ▾](#) [300+ Big Data FAQs ▾](#) [Courses ▾](#)[👤 Membership ▾](#) [Your Career ▾](#)[Home](#) › [bigdata-success.com](#) › [Tutorials - Big Data](#) › [TUT - Apache Storm](#) › 02: Simple

Apache Storm application running inside Eclipse in a local cluster

02: Simple Apache Storm application running inside Eclipse in a local cluster

 Posted on [January 14, 2018](#)**Step 1:** Create a Java project with Maven

```
1
2 bash-4.1$ cd projects
3 bash-4.1$ mvn archetype:generate -DgroupId=com.myt
4
```

Step 2: Open eclipse and import it as an existing maven project by clicking File → Import → Existing Maven Projects →

300+ Java Interview FAQs

300+ Java FAQs



16+ Java Key Areas Q&As



150+ Java Architect FAQs



80+ Java Code Quality Q&As



150+ Java Coding Q&As



300+ Big Data Interview FAQs

300+ Big Data FAQs



Tutorials - Big Data

TUT -  Starting Big Data

TUT - Starting Spark & Scala

“/home/cloudera/projects/simple-storm”, which has the pom.xml file.

Step 3: Add Apache Storm project to pom.xml and then select the project and right mouse click and do “Maven → Update Project”.

```

1
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
4     <modelVersion>4.0.0</modelVersion>
5     <groupId>com.mytutorial</groupId>
6     <artifactId>simple-storm</artifactId>
7     <packaging>jar</packaging>
8     <version>1.0-SNAPSHOT</version>
9     <name>simple-storm</name>
10    <url>http://maven.apache.org</url>
11    <dependencies>
12      <dependency>
13        <groupId>junit</groupId>
14        <artifactId>junit</artifactId>
15        <version>3.8.1</version>
16        <scope>test</scope>
17      </dependency>
18
19      <dependency>
20        <groupId>org.apache.storm</groupId>
21        <artifactId>storm-core</artifactId>
22        <version>1.1.1</version>
23        <scope>compile</scope> <!-- for cluster
24      </dependency>
25    </dependencies>
26
27    <build>
28      <plugins>
29        <plugin>
30          <artifactId>maven-assembly-plugin</artifactId>
31          <configuration>
32            <archive>
33              <manifest>
34                <mainClass>com.mytutorial
35              </manifest>
36            </archive>
37            <descriptorRefs>
38              <descriptorRef>jar-with-dependencies</descriptorRef>
39            </descriptorRefs>
40          </configuration>
41        </plugin>
42      </plugins>
43    </build>

```

TUT - Starting with Python

TUT - Kafka

TUT - Pig

TUT - Apache Storm

TUT - Spark Scala on Zeppelin

TUT - Cloudera

TUT - Cloudera on Docker

TUT - File Formats

TUT - Spark on Docker

TUT - Flume

TUT - Hadoop (HDFS)

TUT - HBase (NoSQL)

TUT - Hive (SQL)

TUT - Hadoop & Spark

TUT - MapReduce

TUT - Spark and Scala

TUT - Spark & Java

TUT - PySpark on Databricks

TUT - Zookeeper

800+ Java Interview Q&As

300+ Core Java Q&As



300+ Enterprise Java Q&As



150+ Java Frameworks Q&As



120+ Companion Tech Q&As



Tutorials - Enterprise Java



```
44 </project>
45
46
```

Add a Spout, Bolt & a Topology

Step 4: A **Spout** that produces incremental numbers starting from 1. A Spout is a source that passes the data to a **Bolt**.

```
1 package com.mytutorial;
2
3 import java.util.Map;
4
5 import org.apache.storm.spout.SpoutOutputCollector;
6 import org.apache.storm.task.TopologyContext;
7 import org.apache.storm.topology.OutputFieldsDeclarer;
8 import org.apache.storm.topology.base.BaseRichSpout;
9 import org.apache.storm.tuple.Fields;
10 import org.apache.storm.tuple.Values;
11
12 public class SimpleCounterSpout extends BaseRichSpout {
13
14     private static final long serialVersionUID = 1L;
15
16     private static int currentCount = 1;
17
18     private SpoutOutputCollector collector;
19
20     public void nextTuple() {
21         collector.emit(new Values(currentCount++));
22         System.out.println("Emitting " + currentCount);
23     }
24
25     public void open(Map arg0, TopologyContext arg1,
26                     SpoutOutputCollector arg2) {
27         this.collector = arg2;
28     }
29
30     public void declareOutputFields(OutputFieldsDeclarer declarer) {
31         declarer.declare(new Fields("number"));
32     }
33
34     public void ack(Object id) {}
35
36     public void fail(Object id) {}
37 }
38
39
40
```

Step 5: A Bolt is a computational component. The Bolt shown below prints only the odd numbers it receives from the Spout.

```
1 package com.mytutorial;
2
3 import java.util.Map;
4
5 import org.apache.storm.task.OutputCollector;
6 import org.apache.storm.task.TopologyContext;
7 import org.apache.storm.topology.OutputFieldsDeclarer;
8 import org.apache.storm.topology.base.BaseRichBolt;
9 import org.apache.storm.tuple.Fields;
10 import org.apache.storm.tuple.Tuple;
11
12 public class SimpleIsOddBolt extends BaseRichBolt {
13
14     private static final long serialVersionUID = 1L;
15
16     private OutputCollector collector;
17
18     public void execute(Tuple arg0) {
19         int number = arg0.getInteger(0);
20
21         if(number % 2 != 0 ) {
22             System.out.println("Processing Odd number: " + number);
23         }
24
25         collector.ack(arg0);
26     }
27
28     public void prepare(Map arg0, TopologyContext arg1, OutputCollector arg2) {
29         this.collector = arg2;
30     }
31
32     public void declareOutputFields(OutputFieldsDeclarer arg0) {
33         arg0.declare(new Fields("number"));
34     }
35 }
36
37
38
```

Step 6: A Topology that runs binds the Spout and the Bolt, and runs the code in a local cluster mode.

```
1 package com.mytutorial;
2
```

```
3 import org.apache.storm.Config;
4 import org.apache.storm.LocalCluster;
5 import org.apache.storm.generated.AlreadyAliveException;
6 import org.apache.storm.generated.AuthorizationException;
7 import org.apache.storm.generated.InvalidTopologyException;
8 import org.apache.storm.topology.TopologyBuilder;
9 import org.apache.storm.utils.Utils;
10
11 public class SimpleTopology {
12
13     public static void main(String[] args) throws
14         TopologyBuilder builder = new TopologyBuilder();
15         builder.setSpout("simple-spout", new SimpleSpout(), 1);
16         builder.setBolt("isOdd", new SimpleIsOddBolt(), 1);
17
18         Config conf = new Config();
19
20         /*StormSubmitter.submitTopology(args[0], conf, builder);
21
22         LocalCluster cluster = new LocalCluster();
23         cluster.submitTopology("simple-storm-tutorial", conf, builder);
24         Utils.sleep(5000);
25         cluster.killTopology("simple-storm-tutorial");
26         cluster.shutdown();
27     }
28 }
29
```

Run “SimpleTopology.java” inside Eclipse as a Java application by selectng it and right-mouse click and “Run As -> Java Application”.

Output

```
1
2 .....
3 Emitting 2
4 Emitting 3
5 Emitting 4
6 Emitting 5
7 Emitting 6
8 Emitting 7
9 Emitting 8
10 Emitting 9
11 Emitting 10
12 Emitting 11
13 Emitting 12
14 Emitting 13
15 Emitting 14
16 Emitting 15
```

```
17 Emitting 16
18 Emitting 17
19 Emitting 18
20 Emitting 19
21 Emitting 20
22 Emitting 21
23 Emitting 22
24 Emitting 23
25 Emitting 24
26 Emitting 25
27 Emitting 26
28 Emitting 27
29 Emitting 28
30 Emitting 29
31 Emitting 30
32 Emitting 31
33 Emitting 32
34 Emitting 33
35 19174 [Thread-14-__acker] INFO b.s.d.executor -
36 Emitting 34
37 Emitting 35
38 Emitting 36
39 Emitting 37
40 Emitting 38
41 Emitting 39
42 19177 [Thread-10-isOdd] INFO b.s.d.executor - P
43 Emitting 40
44 Emitting 41
45 19178 [Thread-14-__acker] INFO b.s.d.executor -
46 19178 [Thread-10-isOdd] INFO b.s.d.executor - P
47 Emitting 42
48 Emitting 43
49 Processing Odd number = 1
50 Emitting 44
51 Processing Odd number = 3
52 Emitting 45
53 Emitting 46
54 Emitting 47
55 Emitting 48
56 Emitting 49
57 Emitting 50
58 Emitting 51
59 Emitting 52
60 Emitting 53
61 Emitting 54
62 Emitting 55
63 Emitting 56
64 Emitting 57
65 Emitting 58
66 Emitting 59
67 Emitting 60
68 Emitting 61
69 Emitting 62
70 Emitting 63
71 Emitting 64
```

```
72 Emitting 65
73 Processing Odd number = 5
74 Processing Odd number = 7
75 Emitting 66
76 Processing Odd number = 9
77 Processing Odd number = 11
78 Processing Odd number = 13
79 Emitting 67
80 Processing Odd number = 15
81 Processing Odd number = 17
82 Processing Odd number = 19
83 Emitting 68
84 Emitting 69
85 Emitting 70
86 Emitting 71
87 Emitting 72
88 Processing Odd number = 21
89 Processing Odd number = 23
90 Processing Odd number = 25
91 Emitting 73
92 Processing Odd number = 27
93 Emitting 74
94 Emitting 75
95 Emitting 76
96 Emitting 77
97 Emitting 78
98 Emitting 79
99 Emitting 80
100 Emitting 81
101 Emitting 82
102 Processing Odd number = 29
103 Emitting 83
104 Processing Odd number = 31
105 Processing Odd number = 33
106 Emitting 84
107 Processing Odd number = 35
108 Processing Odd number = 37
109 Emitting 85
110 Processing Odd number = 39
111 Processing Odd number = 41
112 Emitting 86
113 Processing Odd number = 43
114 Processing Odd number = 45
115 Emitting 87
116 Processing Odd number = 47
117 Processing Odd number = 49
118 .....
119
```

◀ 01: Installing & getting started with Apache Storm on Cloudera quickstart

02: Simple Apache Storm application running on a single node local
cluster ▶

Disclaimer

The contents in this Java-Success are copyrighted and from EmpoweringTech pty ltd. The EmpoweringTech pty ltd has the right to correct or enhance the current content without any prior notice. These are general advice only, and one needs to take his/her own circumstances into consideration. The EmpoweringTech pty ltd will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. Links to external sites do not imply endorsement of the linked-to sites. [Privacy Policy](#).

© 2022 [java-success.com](https://www.java-success.com)