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

```

bash-4.1$ cd projects
bash-4.1$ mvn archetype:generate -DgroupId=com.mytu

```

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 righ mous click and do "Maven –> Update Project".

```

<project xmlns="http://maven.apache.org/POM/4.0.0'
    xsi:schemaLocation="http://maven.apache.org/P(
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mytutorial</groupId>
    <artifactId>simple-storm</artifactId>
    <packaging>jar</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>simple-storm</name>
    <url>http://maven.apache.org</url>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>org.apache.storm</groupId>
            <artifactId>storm-core</artifactId>
            <version>1.1.1</version>
            <scope>compile</scope> <!-- for clust
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <artifactId>maven-assembly-plugin
                <configuration>
                    <archive>
                        <manifest>
                            <mainClass>com.mytutor
                        </manifest>
                    </archive>
                    <descriptorRefs>
                        <descriptorRef>jar-with-d
                    </descriptorRefs>
                </configuration>
            </plugin>
        </plugins>
    </build>
```

```
</project>


```

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.

```
package com.mytutorial;

import java.util.Map;

import org.apache.storm.spout.SpoutOutputCollecto
import org.apache.storm.task.TopologyContext;
import org.apache.storm.topology.OutputFieldsDecl
import org.apache.storm.topology.base.BaseRichSpo
import org.apache.storm.tuple.Fields;
import org.apache.storm.tuple.Values;

public class SimpleCounterSpout extends BaseRichS

    private static final long serialVersionUID = 

    private static int currentCount = 1;

    private SpoutOutputCollector collector;

    public void nextTuple() {
        collector.emit(new Values(currentCount++)
        System.out.println("Emitting " + currentC
    }

    public void open(Map arg0, TopologyContext ar
        this.collector = arg2;
    }

    public void declareOutputFields(OutputFieldsD
        arg0.declare(new Fields("number"));
    }

    public void ack(Object id) {}

    public void fail(Object id) {}

}



```

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

```
package com.mytutorial;

import java.util.Map;

import org.apache.storm.task.OutputCollector;
import org.apache.storm.task.TopologyContext;
import org.apache.storm.topology.OutputFieldsDecl
import org.apache.storm.topology.base.BaseRichBol
import org.apache.storm.tuple.Fields;
import org.apache.storm.tuple.Tuple;

public class SimpleIsOddBolt extends BaseRichBolt

    private static final long serialVersionUID = 

    private OutputCollector collector;

    public void execute(Tuple arg0) {
        int number = arg0.getInteger(0);

        if(number % 2 != 0 ) {
            System.out.println("Processing Odd nu
        }

        collector.ack(arg0);

    }

    public void prepare(Map arg0, TopologyContext
        this.collector = arg2;
    }

    public void declareOutputFields(OutputFieldsD
        arg0.declare(new Fields("number"));
    }
}


```

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

```
package com.mytutorial;

```

```
import org.apache.storm.Config;
import org.apache.storm.LocalCluster;
import org.apache.storm.generated.AlreadyAliveExc
import org.apache.storm.generated.AuthorizationEx
import org.apache.storm.generated.InvalidTopology
import org.apache.storm.topology.TopologyBuilder;
import org.apache.storm.utils.Utils;

public class SimpleTopology {

    public static void main(String[] args) throws
        TopologyBuilder builder = new TopologyBui
        builder.setSpout("simple-spout", new Simp
        builder.setBolt("isOdd", new SimpleIsOddB

        Config conf = new Config();

        /*StormSubmitter.submitTopology(args[0],

        LocalCluster cluster = new LocalCluster()
        cluster.submitTopology("simple-storm-tuto
        Utils.sleep(5000);
        cluster.killTopology("simple-storm-tutori
        cluster.shutdown();
    }
}

```

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

Output

```

..........
Emitting 2
Emitting 3
Emitting 4
Emitting 5
Emitting 6
Emitting 7
Emitting 8
Emitting 9
Emitting 10
Emitting 11
Emitting 12
Emitting 13
Emitting 14
Emitting 15
Emitting 16
Emitting 17
Emitting 18
Emitting 19
Emitting 20
Emitting 21
Emitting 22
Emitting 23
Emitting 24
Emitting 25
Emitting 26
Emitting 27
Emitting 28
Emitting 29
Emitting 30
Emitting 31
Emitting 32
Emitting 33
19174 [Thread-14-__acker] INFO  b.s.d.executor -
Emitting 34
Emitting 35
Emitting 36
Emitting 37
Emitting 38
Emitting 39
19177 [Thread-10-isOdd] INFO  b.s.d.executor - P
Emitting 40
Emitting 41
19178 [Thread-14-__acker] INFO  b.s.d.executor -
19178 [Thread-10-isOdd] INFO  b.s.d.executor - P
Emitting 42
Emitting 43
Processing Odd number = 1
Emitting 44
Processing Odd number = 3
Emitting 45
Emitting 46
Emitting 47
Emitting 48
Emitting 49
Emitting 50
Emitting 51
Emitting 52
Emitting 53
Emitting 54
Emitting 55
Emitting 56
Emitting 57
Emitting 58
Emitting 59
Emitting 60
Emitting 61
Emitting 62
Emitting 63
Emitting 64
Emitting 65
Processing Odd number = 5
Processing Odd number = 7
Emitting 66
Processing Odd number = 9
Processing Odd number = 11
Processing Odd number = 13
Emitting 67
Processing Odd number = 15
Processing Odd number = 17
Processing Odd number = 19
Emitting 68
Emitting 69
Emitting 70
Emitting 71
Emitting 72
Processing Odd number = 21
Processing Odd number = 23
Processing Odd number = 25
Emitting 73
Processing Odd number = 27
Emitting 74
Emitting 75
Emitting 76
Emitting 77
Emitting 78
Emitting 79
Emitting 80
Emitting 81
Emitting 82
Processing Odd number = 29
Emitting 83
Processing Odd number = 31
Processing Odd number = 33
Emitting 84
Processing Odd number = 35
Processing Odd number = 37
Emitting 85
Processing Odd number = 39
Processing Odd number = 41
Emitting 86
Processing Odd number = 43
Processing Odd number = 45
Emitting 87
Processing Odd number = 47
Processing Odd number = 49
.........

```

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

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