Lab: Kafka Word Count

Storm Word Count with Kafka

About this Lab

Objective:	Enhance the existing Word Count application with integration to Kafka
File locations:	`/labs/rtlabs/projects/storm/word-count`
Successful outcome:	Have successfully leveraged Storm's Kafka spout to source the Word Count streaming application
Before you begin	N/A
Related lesson:	Storm Architecture

Leverage LocalCluster Submission

Enhance the developer experience by leveraging Storm's LocalCluster to run, and debug, Storm topologies within IDEA.

Replace the existing `submitTopology()` method with the following code.

```
        /*
        StormSubmitter.submitTopologyWithProgressBar(
                "word-count", conf,
                builder.createTopology());
        */

        LocalCluster cluster = new LocalCluster();
        cluster.submitTopology("word-count-local",
                conf, builder.createTopology());
```

To simplify things even further, reduce the parallelism hint down to 1 for the counter bolt.

HINT: The counter bolt is currently set to 2

Adapt the `pom.xml` to work with in `LocalCluster` configuration. First, uncomment the dependencies within the following XML comments.

```
        <!-- Logging dependencies needed for local mode -->

        <!-- End of Logging -->
```

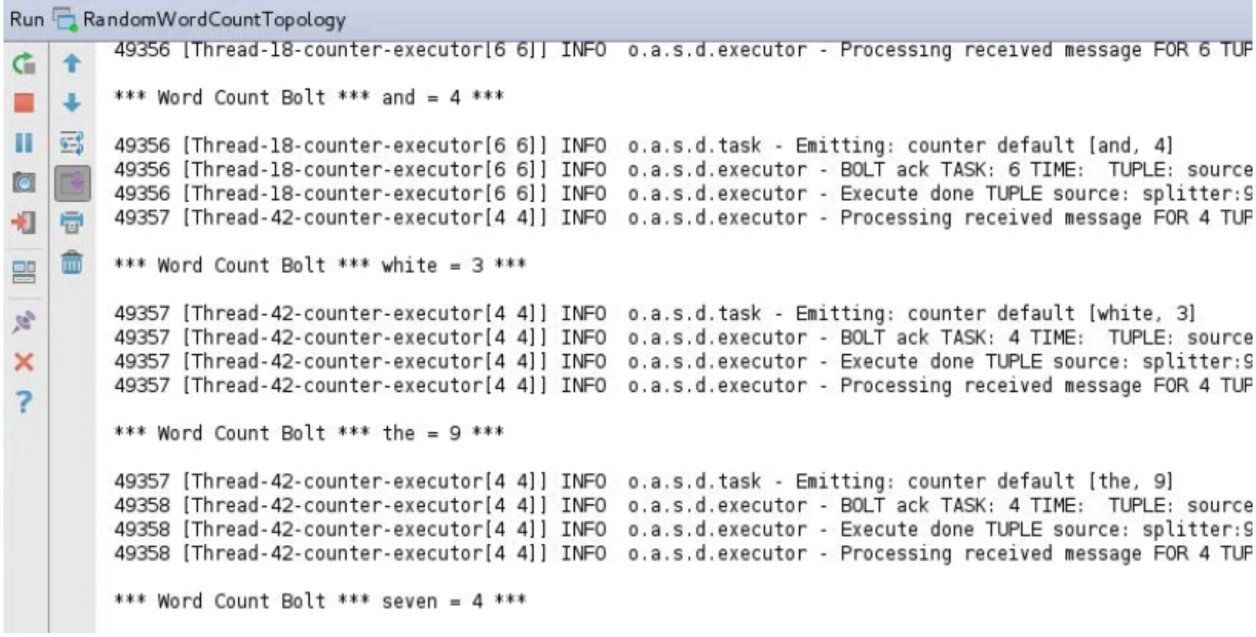
Comment out the `scope` provided tag for the `storm-core` artifact.

```
            <!-- <scope>provided</scope>  avail in cluster mode, but tag
needs to be commented out for local -->
```

For that same artifact, uncomment the `exclusions` block.

Ensure the project can be built.

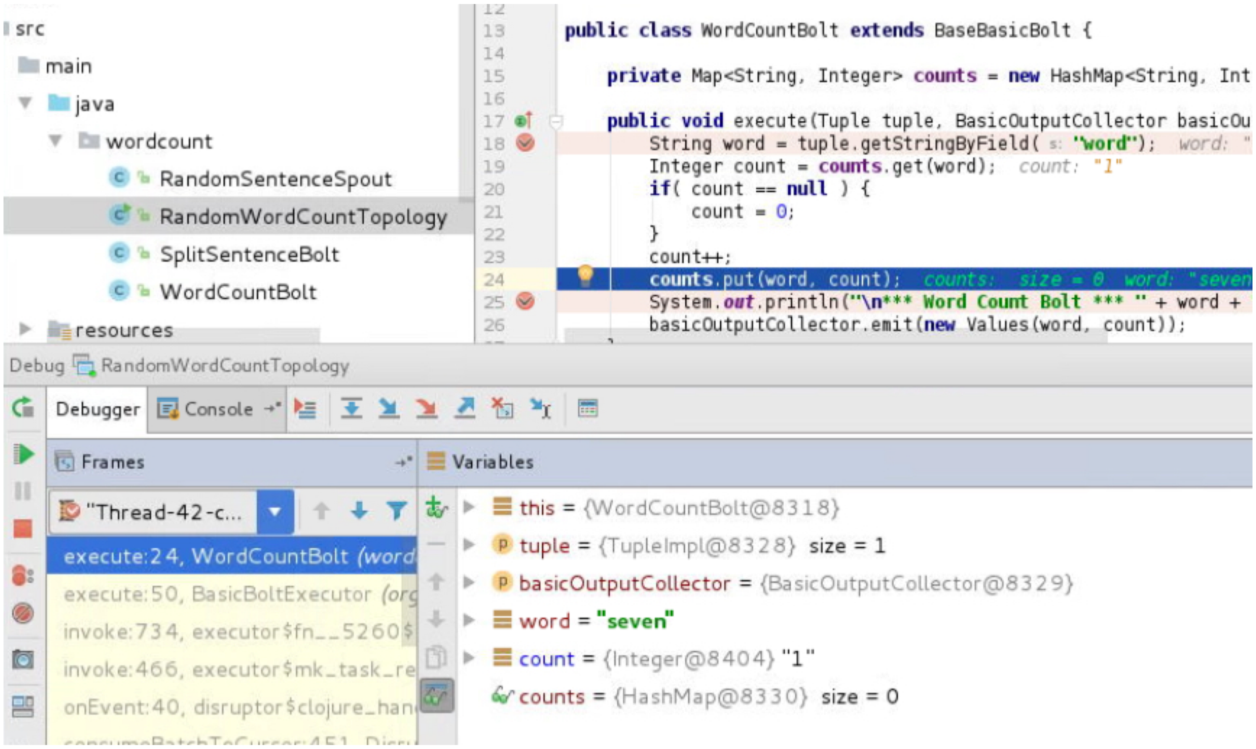
Run the project within IDEA by right-clicking on the `RandomWordCountTopology` class and selecting “Run … as main()” and examine the live (and consolidated) output.



```
49356 [Thread-18-counter-executor[6 6]] INFO  o.a.s.d.executor - Processing received message FOR 6 TUP

*** Word Count Bolt *** and = 4 ***

49356 [Thread-18-counter-executor[6 6]] INFO  o.a.s.d.task - Emitting: counter default [and, 4]
49356 [Thread-18-counter-executor[6 6]] INFO  o.a.s.d.executor - BOLT ack TASK: 6 TIME:  TUPLE: source
49356 [Thread-18-counter-executor[6 6]] INFO  o.a.s.d.executor - Execute done TUPLE source: splitter:
49357 [Thread-42-counter-executor[4 4]] INFO  o.a.s.d.executor - Processing received message FOR 4 TUP

*** Word Count Bolt *** white = 3 ***

49357 [Thread-42-counter-executor[4 4]] INFO  o.a.s.d.task - Emitting: counter default [white, 3]
49357 [Thread-42-counter-executor[4 4]] INFO  o.a.s.d.executor - BOLT ack TASK: 4 TIME:  TUPLE: source
49357 [Thread-42-counter-executor[4 4]] INFO  o.a.s.d.executor - Execute done TUPLE source: splitter:
49357 [Thread-42-counter-executor[4 4]] INFO  o.a.s.d.executor - Processing received message FOR 4 TUP

*** Word Count Bolt *** the = 9 ***

49357 [Thread-42-counter-executor[4 4]] INFO  o.a.s.d.task - Emitting: counter default [the, 9]
49358 [Thread-42-counter-executor[4 4]] INFO  o.a.s.d.executor - BOLT ack TASK: 4 TIME:  TUPLE: source
49358 [Thread-42-counter-executor[4 4]] INFO  o.a.s.d.executor - Execute done TUPLE source: splitter:
49358 [Thread-42-counter-executor[4 4]] INFO  o.a.s.d.executor - Processing received message FOR 4 TUP

*** Word Count Bolt *** seven = 4 ***
```

To stop, just use the red square in the upper portion of the left-nav toolbar.

Enhance the experience by placing one, or more, checkpoints in the source code and initiating the debugger by right-clicking the same class and choosing “Debug … as main()” option.



```
public class WordCountBolt extends BaseBasicBolt {

    private Map<String, Integer> counts = new HashMap<String, Int

    public void execute(Tuple tuple, BasicOutputCollector basicOu
        String word = tuple.getStringByField( s: "word");   word: "
        Integer count = counts.get(word);   count: "1"
        if( count == null ) {
            count = 0;
        }
        count++;
        counts.put(word, count);   counts:  size = 0  word: "seven
        System.out.println("\n*** Word Count Bolt *** " + word +
        basicOutputCollector.emit(new Values(word, count));
```

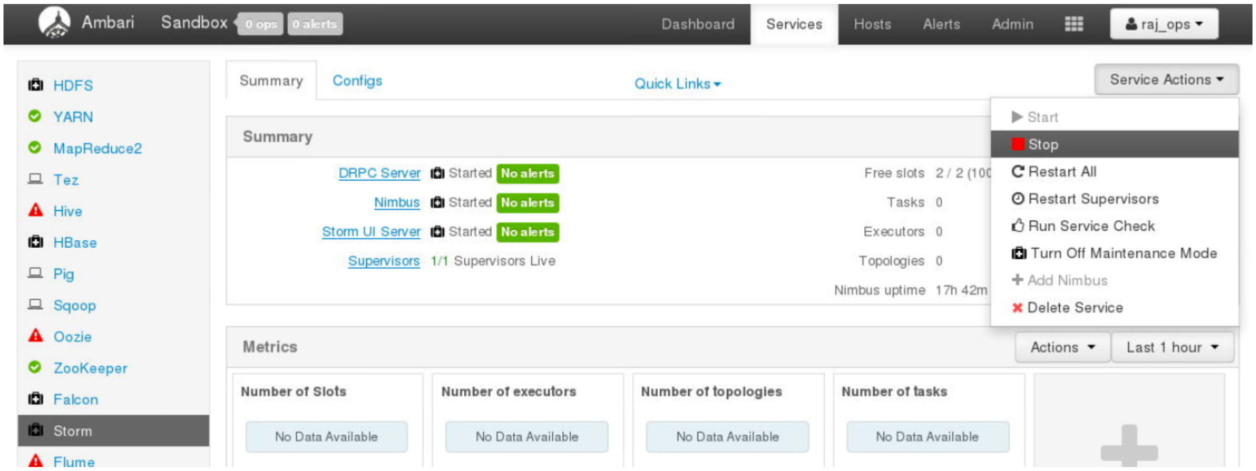
```
execute:24, WordCountBolt (word
execute:50, BasicBoltExecutor (org
invoke:734, executor$fn__5260$
invoke:466, executor$mk_task_re
onEvent:40, disruptor$clojure_han
```

```
this = {WordCountBolt@8318}
tuple = {TupleImpl@8328} size = 1
basicOutputCollector = {BasicOutputCollector@8329}
word = "seven"
count = {Integer@8404} "1"
counts = {HashMap@8330} size = 0
```

As before, click on the red square in the left-nav toolbar to stop the debugging session.

Stop the Storm Service

As we have transitioned to running topologies in local mode, we can shut down the Storm service by logging into Ambari at http://127.0.0.1:8081 as `raj_ops` (password also `raj_ops`) and selecting the service on the left-hand nav and "Stop" in the "Service Actions" pulldown.



Summary

DRPC Server Started No alerts
Nimbus Started No alerts
Storm UI Server Started No alerts
Supervisors 1/1 Supervisors Live

Free slots 2 / 2 (100
Tasks 0
Executors 0
Topologies 0
Nimbus uptime 17h 42m

Setup Kafka Sentence Topic

As explained in the Kafka Topics lab, create a topic named `sentences` with a single partition and a single replica.

NOTE: This topic may already exist from a prior lab exercise.

HINT: Use the --create switch on the kafka-topics.sh script.

Verify your new topic was created.

```
[maria_dev@sandbox ~]$ /usr/hdp/current/kafka-broker/bin/kafka-topics.sh --
list --zookeeper sandbox.hortonworks.com:2181
ATLAS_HOOK
__consumer_offsets
mariaTopic - marked for deletion
sentences
[maria_dev@sandbox ~]$
```

As explained earlier, establish a console-based producer to this new topic and add a few sentences. **NOTE: Do not close this window.**

HINT: Don't forget you need to use the --broker-list swich on the kafka-console-producer.sh script.

In another terminal window or tab, establish a console-based consumer to verify that your ad-hoc sentences are consumable from a Kafka client.

```
[maria_dev@sandbox ~]$ /usr/hdp/current/kafka-broker/bin/kafka-console-
consumer.sh --bootstrap-server sandbox.hortonworks.com:6667 --topic sentences
--from-beginning
Now is the time for all good men to come to the aid of their country
Here is another sentence
And yet another one
```

Create a Kafka Spout

Make a copy of the existing `RandomWordCountTopology` class and call it `KafkaWordCountTopology`.

Create an object to keep track of the Zookeeper quorum.

```
        BrokerHosts hosts = new ZkHosts("sandbox.hortonworks.com:2181");
```

Provide the necessary plumbing to instantiate a `KafkaSpout` referencing the topic we will be consuming.

```
        SpoutConfig sc = new SpoutConfig(hosts,
                "sentences", "/sentences",
                UUID.randomUUID().toString());
        sc.scheme = new SchemeAsMultiScheme(new StringScheme());
        KafkaSpout spout = new KafkaSpout(sc);
```

Change the existing `setSpout()` method to reference the spout object reference and rename is moniker to “kafka-spout”. Then reconfig the “splitter” bolt to be wired up to receive tuples from this new bolt.

HINT: The existing spot is referred to as “generator”.

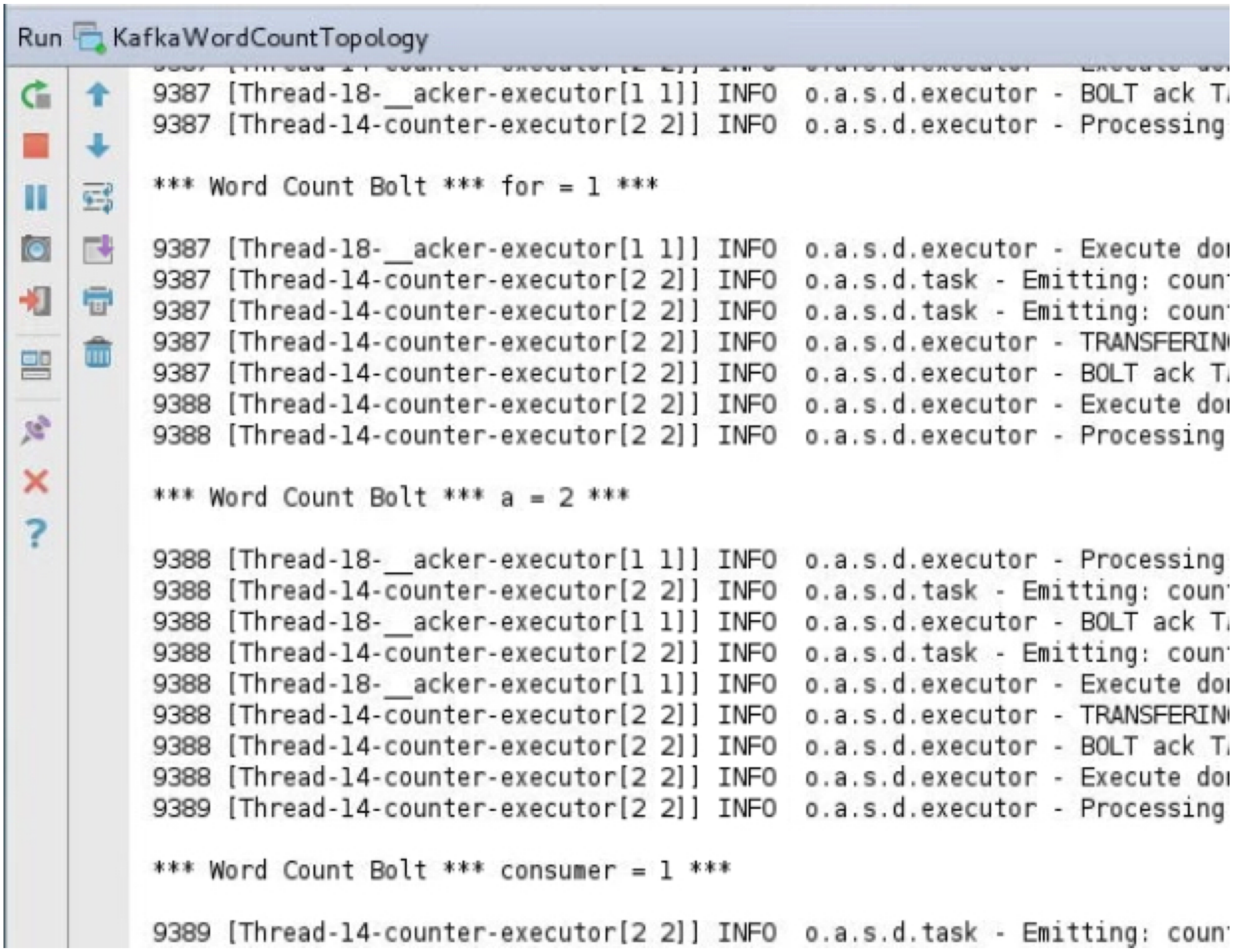
As we did not implement the `declareOutputFields()` method of this new bolt with a friendly name, update the `SplitSentenceBolt` to get the sentence by its numeric offset from the tuple instead of a named field.

HINT: Use `getString(0)` instead.

Ensure the project can be built.

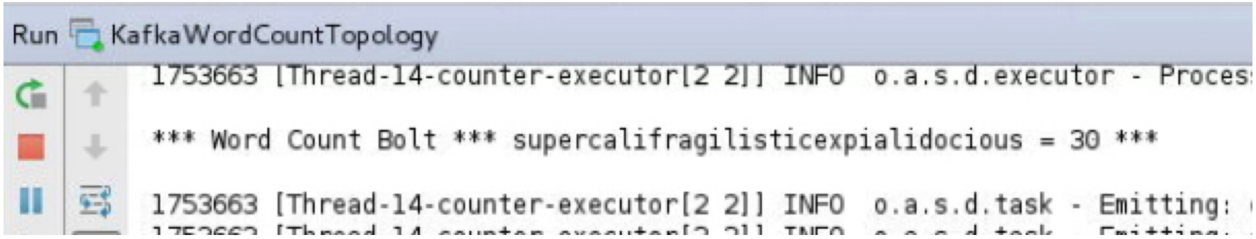
Execute the Updated Topology

Run and/or debug in IntelliJ as performed earlier and verify the correct word counts are occurring.



```
9387 [Thread-18-__acker-executor[1 1]] INFO  o.a.s.d.executor - BOLT ack T
9387 [Thread-14-counter-executor[2 2]] INFO  o.a.s.d.executor - Processing

*** Word Count Bolt *** for = 1 ***

9387 [Thread-18-__acker-executor[1 1]] INFO  o.a.s.d.executor - Execute do
9387 [Thread-14-counter-executor[2 2]] INFO  o.a.s.d.task - Emitting: coun
9387 [Thread-14-counter-executor[2 2]] INFO  o.a.s.d.task - Emitting: coun
9387 [Thread-14-counter-executor[2 2]] INFO  o.a.s.d.executor - TRANSFERIN
9387 [Thread-14-counter-executor[2 2]] INFO  o.a.s.d.executor - BOLT ack T
9388 [Thread-14-counter-executor[2 2]] INFO  o.a.s.d.executor - Execute do
9388 [Thread-14-counter-executor[2 2]] INFO  o.a.s.d.executor - Processing

*** Word Count Bolt *** a = 2 ***

9388 [Thread-18-__acker-executor[1 1]] INFO  o.a.s.d.executor - Processing
9388 [Thread-14-counter-executor[2 2]] INFO  o.a.s.d.task - Emitting: coun
9388 [Thread-18-__acker-executor[1 1]] INFO  o.a.s.d.executor - BOLT ack T
9388 [Thread-14-counter-executor[2 2]] INFO  o.a.s.d.task - Emitting: coun
9388 [Thread-18-__acker-executor[1 1]] INFO  o.a.s.d.executor - Execute do
9388 [Thread-14-counter-executor[2 2]] INFO  o.a.s.d.executor - TRANSFERIN
9388 [Thread-14-counter-executor[2 2]] INFO  o.a.s.d.executor - BOLT ack T
9388 [Thread-14-counter-executor[2 2]] INFO  o.a.s.d.executor - Execute do
9389 [Thread-14-counter-executor[2 2]] INFO  o.a.s.d.executor - Processing

*** Word Count Bolt *** consumer = 1 ***

9389 [Thread-14-counter-executor[2 2]] INFO  o.a.s.d.task - Emitting: coun
```

Via the Kafka console producer add two sentences repeating a single word multiple times and verify that this word and the appropriate count is displayed in the console output.



```
1753663 [Thread-14-counter-executor[2 2]] INFO  o.a.s.d.executor - Proces

*** Word Count Bolt *** supercalifragilisticexpialidocious = 30 ***

1753663 [Thread-14-counter-executor[2 2]] INFO  o.a.s.d.task - Emitting:
```

Summary

You have successful built a complete Storm topology, submitted it to the Storm cluster, and monitored this real-time application.

A complete solution to this lab can be found in `/root/rtlabs/proj-solns/storm/kafka-world-count`.