Lab: Storm with Kafka and HBase

Ingest Data in Real-time using Storm

About this Lab

Objective:	Use Storm to ingest/process data from Kafka and write into HBase
File locations:	`/labs/rtlabs/projects/storm/workshop`
Successful outcome:	Have successfully written data into HBase using Storm after data from Kafka
Before you begin	Complete the Word Count Topology lab
Related lesson:	Streaming Workshop Use Case

Setup Kafka Topic and Log File Generator

In the sandbox host, create a new topic to hold the input data for the streaming application.

```
[maria_dev@sandbox ~]$ /usr/hdp/current/kafka-broker/bin/kafka-topics.sh --
create --zookeeper sandbox.hortonworks.com:2181 --replication-factor 1 --
partitions 1 --topic logs
Created topic "logs".
[maria_dev@sandbox ~]$ /usr/hdp/current/kafka-broker/bin/kafka-topics.sh --
list --zookeeper sandbox.hortonworks.com:2181
ATLAS_HOOK
__consumer_offsets
logs
mariaTopic - marked for deletion
sentences
[maria_dev@sandbox ~]$
```

Tail this new topic which will initially be empty.

```
[maria_dev@sandbox ~]$ /usr/hdp/current/kafka-broker/bin/kafka-console-
consumer.sh --bootstrap-server sandbox.hortonworks.com:6667 --topic logs --
from-beginning
```

Start a new Terminal and remain as the `root` user once logged into the original (outer) host.

Explore the usage of the log generator script.

```
[root@ip-172-30-0-164 ~]# cd /root/rtlabs/log-gen
[root@ip-172-30-0-164 log-gen]# python logSim.py help
```

```
Incorrect usage of simulator
Please pass an integer messages/sec between 1 and 10, followed by a topic
Pass help to see more information: logSim.py help
[root@ip-172-30-0-164 log-gen]#
```

Kick off the generator instructing it to run as slow as it can and produce records on your previously created Kafka topic.

```
[root@ip-172-30-0-164 log-gen]# python logSim.py 1 logs
Starting Kafka Broker
Printing  1  messages per second.  Press control-c to stop
^CTraceback (most recent call last):
  File "logSim.py", line 24, in <module>
    sleep(x)
KeyboardInterrupt
[root@ip-172-30-0-164 log-gen]#
```

After a few seconds, halt the script and notice that your other terminal window has outputted the newly created records.

```
[maria_dev@sandbox ~]$ /usr/hdp/current/kafka-broker/bin/kafka-console-
consumer.sh --bootstrap-server sandbox.hortonworks.com:6667 --topic logs --
from-beginning
2017-06-04  15:51:38.71 48.257.926.101    INFO  Generic INFO # 18
2017-06-04  15:51:39.76 32.234.242.261    DEBUG Worthless DEBUG #1
2017-06-04  15:51:40.81 31.160.9.90 WARN  Generic WARN # 6
2017-06-04  15:51:41.87 57.118.219.201    INFO  Generic INFO # 14
2017-06-04  15:51:42.92 35.220.141.248    ERROR BAD ERROR # 2
2017-06-04  15:51:43.97 45.226.670.309    WARN  Generic WARN # 7
```

HINT: If only INFO message are present, run the generator a few more seconds until at least one WARN or ERROR message is created.

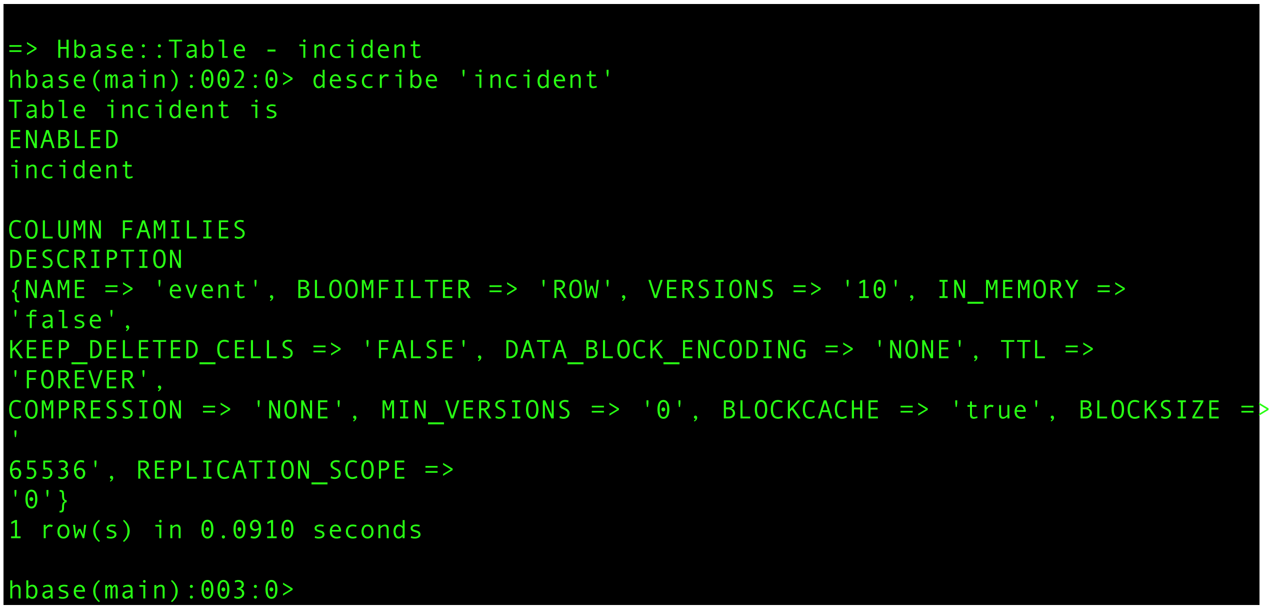
You can leave this window running if you would like to leverage it later when visualizing the entire implementation.

Setup HBase Table

Via the HBase Shell, create a table named `incident` that has a column family called `event` which will keep up to 10 versions.

```
[root@ip-172-30-0-164 ~]# ssh -p 2222 root@127.0.0.1
root@127.0.0.1's password:
Last login: Sun Jun  4 15:57:44 2017 from 172.17.0.1
[root@sandbox ~]# su - maria_dev
[maria_dev@sandbox ~]$ hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.1.2.2.6.0.3-8, r3307790b5a22cf93100cad0951760718dee5dec7, Sat
Apr  1 21:41:47 UTC 2017

hbase(main):001:0> create 'incident', {NAME => 'event', VERSIONS => 10}
0 row(s) in 2.4320 seconds
```

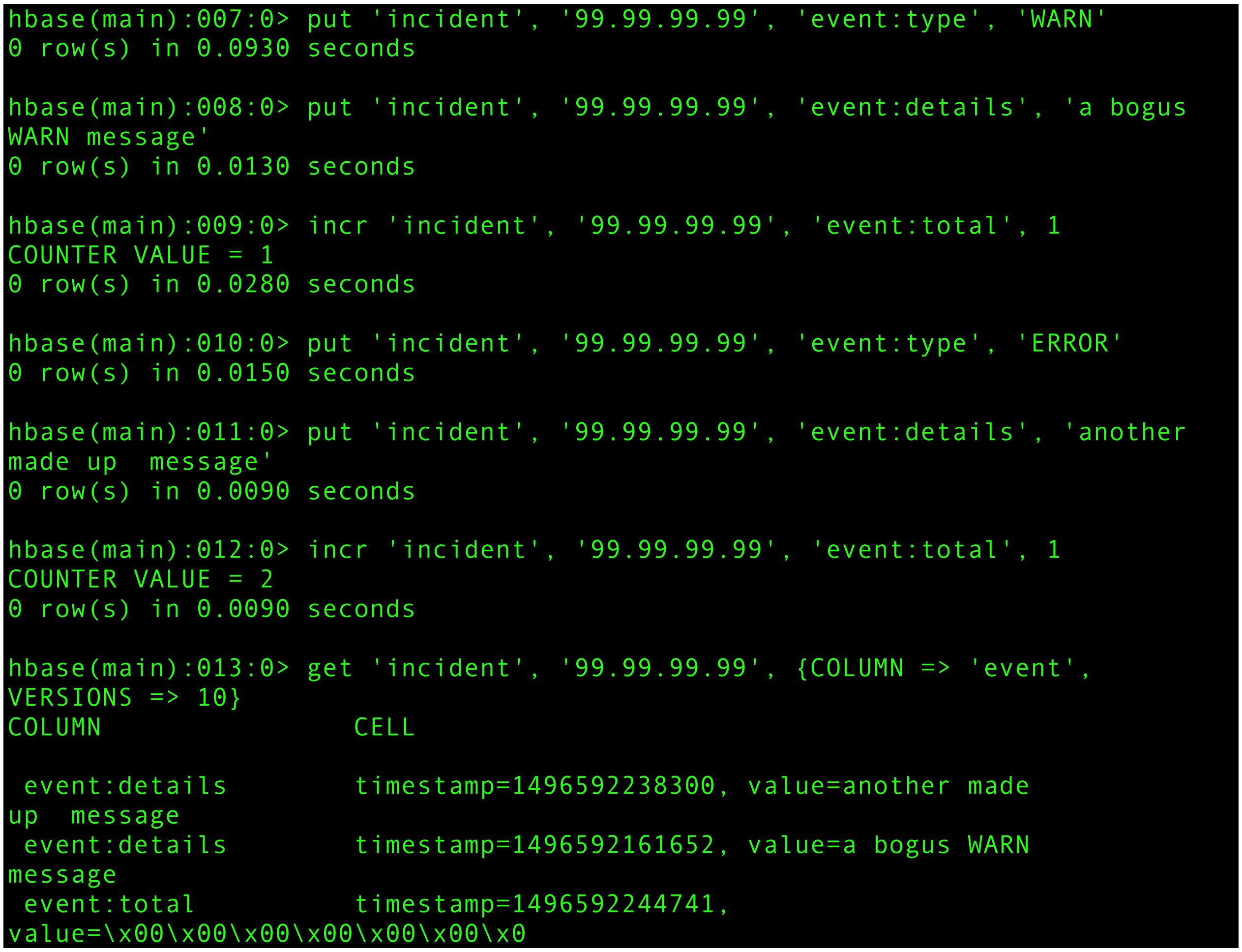


```
=> Hbase::Table - incident
hbase(main):002:0> describe 'incident'
Table incident is
ENABLED
incident

COLUMN FAMILIES
DESCRIPTION
{NAME => 'event', BLOOMFILTER => 'ROW', VERSIONS => '10', IN_MEMORY =>
'false',
KEEP_DELETED_CELLS => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL =>
'FOREVER',
COMPRESSION => 'NONE', MIN_VERSIONS => '0', BLOCKCACHE => 'true', BLOCKSIZE =>
'
65536', REPLICATION_SCOPE =>
'0'}
1 row(s) in 0.0910 seconds

hbase(main):003:0>
```

Add some example data and verify multiple versions can be retrieved.



```
hbase(main):007:0> put 'incident', '99.99.99.99', 'event:type', 'WARN'
0 row(s) in 0.0930 seconds

hbase(main):008:0> put 'incident', '99.99.99.99', 'event:details', 'a bogus
WARN message'
0 row(s) in 0.0130 seconds

hbase(main):009:0> incr 'incident', '99.99.99.99', 'event:total', 1
COUNTER VALUE = 1
0 row(s) in 0.0280 seconds

hbase(main):010:0> put 'incident', '99.99.99.99', 'event:type', 'ERROR'
0 row(s) in 0.0150 seconds

hbase(main):011:0> put 'incident', '99.99.99.99', 'event:details', 'another
made up  message'
0 row(s) in 0.0090 seconds

hbase(main):012:0> incr 'incident', '99.99.99.99', 'event:total', 1
COUNTER VALUE = 2
0 row(s) in 0.0090 seconds

hbase(main):013:0> get 'incident', '99.99.99.99', {COLUMN => 'event',
VERSIONS => 10}
COLUMN                CELL

 event:details        timestamp=1496592238300, value=another made
up  message
 event:details        timestamp=1496592161652, value=a bogus WARN
message
 event:total          timestamp=1496592244741,
value=\x00\x00\x00\x00\x00\x00\x0
```

```
                      0\x02

 event:total          timestamp=1496592178787,
value=\x00\x00\x00\x00\x00\x00\x0
                      0\x01

 event:type           timestamp=1496592210599,
value=ERROR
 event:type           timestamp=1496592136822,
value=WARN
6 row(s) in 0.0350 seconds

hbase(main):014:0>
```

Validate Skelton Project Maven Build

Open the skeleton project within IDEA.

HINT: File > Open > `/root/rtlabs/projects/storm/workshop` *> OK*

NOTE: If you open the project in a "new window" instead of "this window", you will be able to easily reference the prior project which will be helpful in building this topology.

Ensure the Maven tooling is accessible.

HINT: View > Tool Windows > Maven Projects

Build the Maven project and verify the jar file was created.

```
[student20@ip-172-30-0-199 target]$ pwd
/root/rtlabs/projects/storm/workshop/target
[student20@ip-172-30-0-199 target]$ ls -l
total 100260
drwxr-xr-x. 3 root root        82 Mar 11 05:58 classes
drwxr-xr-x. 2 root root        27 Mar 11 05:58 maven-archiver
drwxr-xr-x. 3 root root        34 Mar 11 05:58 maven-status
-rw-r--r--. 1 root root      6601 Mar 11 05:58 original-storm-kafka-hbase-
1.0-SNAPSHOT.jar
-rw-r--r--. 1 root root 102655598 Mar 11 05:58 storm-kafka-hbase-1.0-
SNAPSHOT.jar
[student20@ip-172-30-0-199 target]$
```

Create Kafka Spout

Modify the `LogAnalyzerTopology` class to leverage a new `KafkaSpout` instance that will read from the `logs` topic created in an earlier step.

HINT: Review the `KafkaWordCountTopology` *class in earlier lab.*

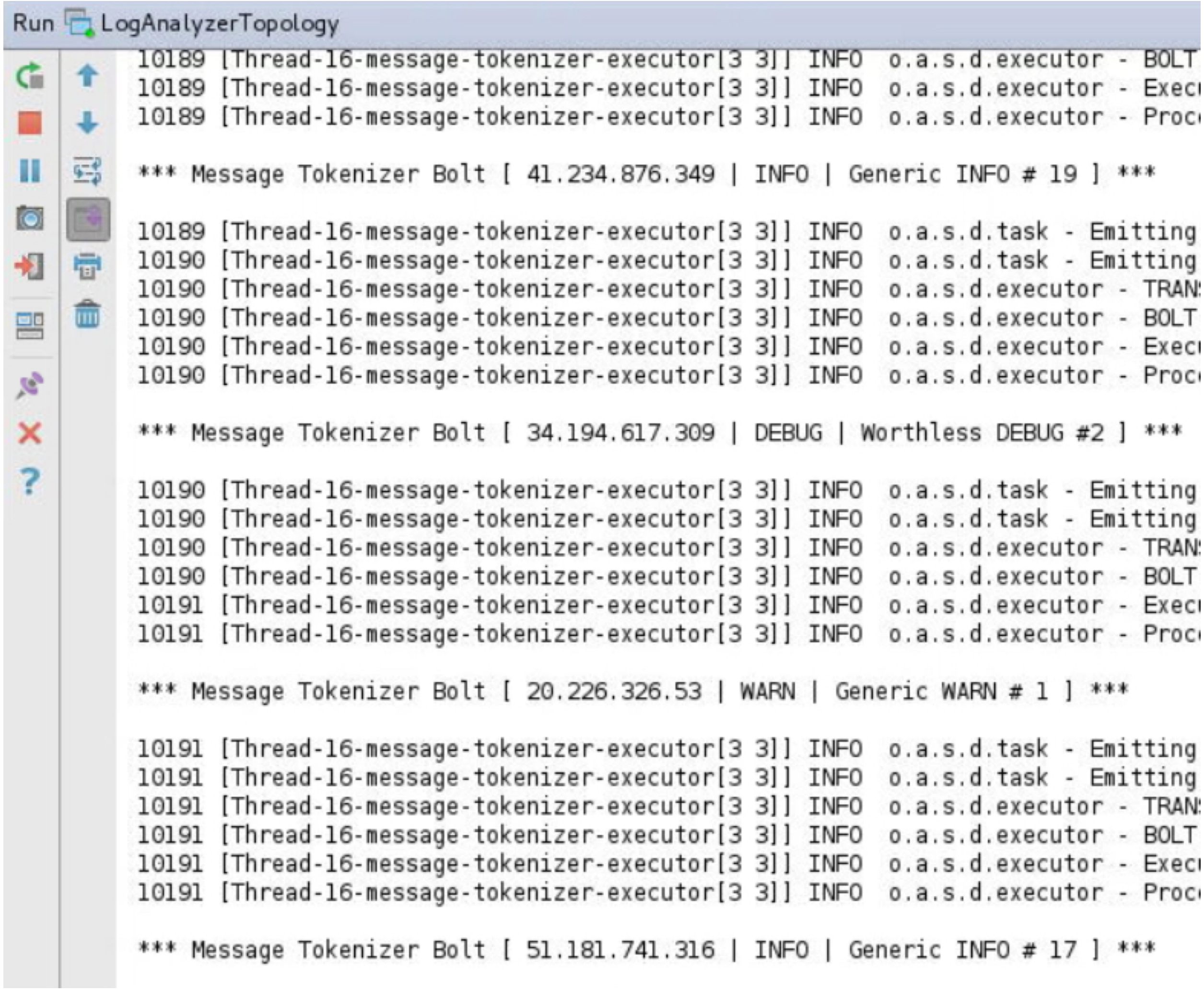
Ensure the project can be built.

Create a Log Entry Parser Bolt

Create a new subclass of `BaseBasicBolt` that will get the 1-tuple String emitted from the `KafkaSpout` instance. This string contains five values (date, time, ip address, event type, and event details) that are separated by tabs. Ignore the date and time fields for simplicity and emit the other three values.

HINT: The `StringUtil.split()` method from the earlier lab's `SplitSentenceBolt` could be modified to take a second parameter, `'\t'`, to tokenize the full String and the resulting array's third through fifth elements are what we are looking for.

Validate that you are properly raising the needed 3-tuple from this new bolt once you chain it to the `KafkaSpout`.



```
10189 [Thread-16-message-tokenizer-executor[3 3]] INFO  o.a.s.d.executor - BOLT
10189 [Thread-16-message-tokenizer-executor[3 3]] INFO  o.a.s.d.executor - Exec
10189 [Thread-16-message-tokenizer-executor[3 3]] INFO  o.a.s.d.executor - Proc

*** Message Tokenizer Bolt [ 41.234.876.349 | INFO | Generic INFO # 19 ] ***

10189 [Thread-16-message-tokenizer-executor[3 3]] INFO  o.a.s.d.task - Emitting
10190 [Thread-16-message-tokenizer-executor[3 3]] INFO  o.a.s.d.task - Emitting
10190 [Thread-16-message-tokenizer-executor[3 3]] INFO  o.a.s.d.executor - TRAN
10190 [Thread-16-message-tokenizer-executor[3 3]] INFO  o.a.s.d.executor - BOLT
10190 [Thread-16-message-tokenizer-executor[3 3]] INFO  o.a.s.d.executor - Exec
10190 [Thread-16-message-tokenizer-executor[3 3]] INFO  o.a.s.d.executor - Proc

*** Message Tokenizer Bolt [ 34.194.617.309 | DEBUG | Worthless DEBUG #2 ] ***

10190 [Thread-16-message-tokenizer-executor[3 3]] INFO  o.a.s.d.task - Emitting
10190 [Thread-16-message-tokenizer-executor[3 3]] INFO  o.a.s.d.task - Emitting
10190 [Thread-16-message-tokenizer-executor[3 3]] INFO  o.a.s.d.executor - TRAN
10190 [Thread-16-message-tokenizer-executor[3 3]] INFO  o.a.s.d.executor - BOLT
10191 [Thread-16-message-tokenizer-executor[3 3]] INFO  o.a.s.d.executor - Exec
10191 [Thread-16-message-tokenizer-executor[3 3]] INFO  o.a.s.d.executor - Proc

*** Message Tokenizer Bolt [ 20.226.326.53 | WARN | Generic WARN # 1 ] ***

10191 [Thread-16-message-tokenizer-executor[3 3]] INFO  o.a.s.d.task - Emitting
10191 [Thread-16-message-tokenizer-executor[3 3]] INFO  o.a.s.d.task - Emitting
10191 [Thread-16-message-tokenizer-executor[3 3]] INFO  o.a.s.d.executor - TRAN
10191 [Thread-16-message-tokenizer-executor[3 3]] INFO  o.a.s.d.executor - BOLT
10191 [Thread-16-message-tokenizer-executor[3 3]] INFO  o.a.s.d.executor - Exec
10191 [Thread-16-message-tokenizer-executor[3 3]] INFO  o.a.s.d.executor - Proc

*** Message Tokenizer Bolt [ 51.181.741.316 | INFO | Generic INFO # 17 ] ***
```

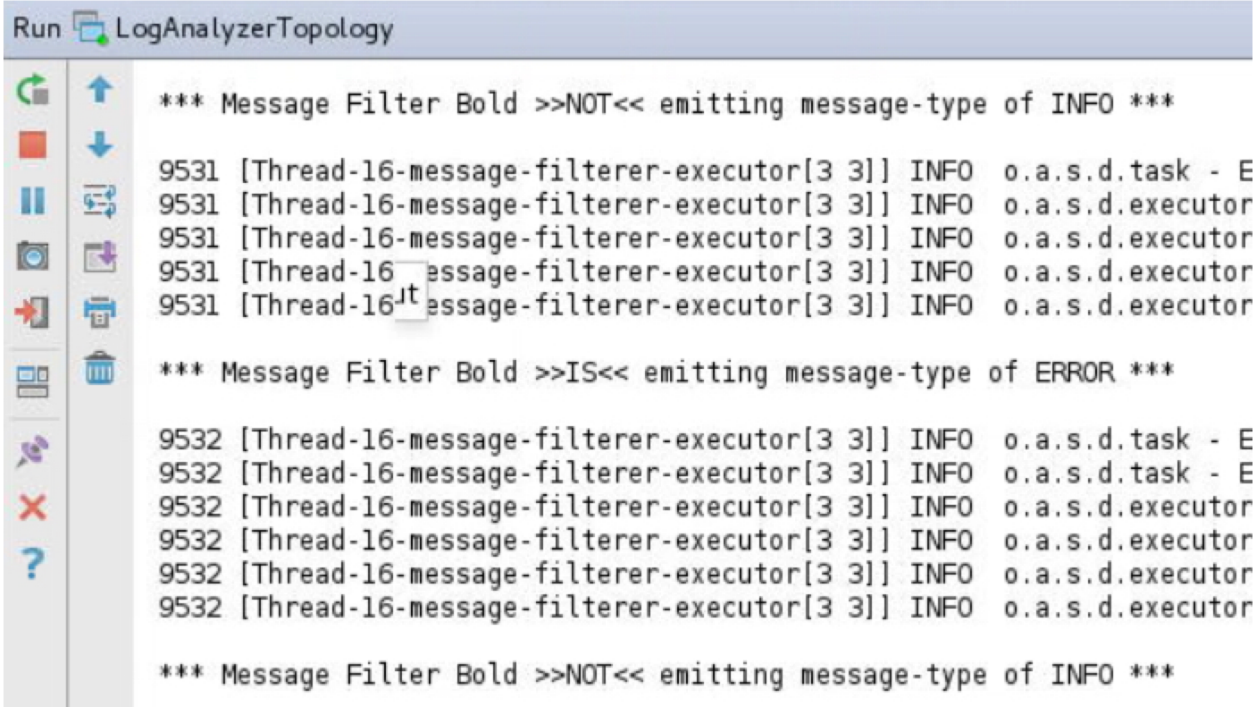
Create an Event Filtering Bolt

For this use case, we are only concerned with log messages of type WARN or ERROR to ultimately be persisted into HBase. Create, and wire into the topology, a bolt that will process the tokenized log messages from the bolt constructed in the last step. This filtering bolt just needs to emit the WARN and ERROR log messages and do nothing for any other messages.

To assist in the next step, be sure to use the following method body is used to declare the field names leaving this bolt. The first three are self-explanatory, but the “total” field should be populated with a primitive int value of 1 when this 4-tuple is emitted.

```
        outputFieldsDeclarer.declare(new Fields(
                "ip-address", "type", "details", "total"));
```

Ensure the bolt is only surfacing those log messages with the needed type.



```
*** Message Filter Bold >>NOT<< emitting message-type of INFO ***

9531 [Thread-16-message-filterer-executor[3 3]] INFO  o.a.s.d.task - E
9531 [Thread-16-message-filterer-executor[3 3]] INFO  o.a.s.d.executor
9531 [Thread-16-message-filterer-executor[3 3]] INFO  o.a.s.d.executor
9531 [Thread-16  essage-filterer-executor[3 3]] INFO  o.a.s.d.executor
9531 [Thread-16  essage-filterer-executor[3 3]] INFO  o.a.s.d.executor

*** Message Filter Bold >>IS<< emitting message-type of ERROR ***

9532 [Thread-16-message-filterer-executor[3 3]] INFO  o.a.s.d.task - E
9532 [Thread-16-message-filterer-executor[3 3]] INFO  o.a.s.d.task - E
9532 [Thread-16-message-filterer-executor[3 3]] INFO  o.a.s.d.executor
9532 [Thread-16-message-filterer-executor[3 3]] INFO  o.a.s.d.executor
9532 [Thread-16-message-filterer-executor[3 3]] INFO  o.a.s.d.executor
9532 [Thread-16-message-filterer-executor[3 3]] INFO  o.a.s.d.executor

*** Message Filter Bold >>NOT<< emitting message-type of INFO ***
```

Configure an HBase Bolt

Review the information at https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.6.0/bk_storm-component-guide/content/storm-write-to-hbase.html concerning the `SimpleHBaseMapper` class that is used to identify how inbound tuple fields should be used when writing data to HBase. Create an instance of this class that aligns with the data being created from the log file generator presented at the beginning of this lab.

NOTE: The hard-code int value of 1 being emitted from the prior bolt will be used to increment an HBase counter. The event column family will only keep the 10 most recent versions by the row key of ip address, but this counter will allow a running total of all events that were ever stored for a given server.

```
        SimpleHBaseMapper mapper = new SimpleHBaseMapper()
                .withRowKeyField("ip-address")
                .withColumnFields(new Fields("type", "details"))
                .withCounterFields(new Fields("total"))
                .withColumnFamily("event");
```

Instantiate an `HBaseBolt` and identify the name of the table where the data from the mapping exercise will be stored.

```
        HBaseBolt hbase = new HBaseBolt("incident", mapper)
```

Lastly, wire up this new bolt to the event filtering one.

```
        builder.setBolt("hbase-bolt", hbase, 1)
                .shuffleGrouping("message-filterer");
```

Ensure the project can be built.

Check to see how many WARN and ERROR log messages you earlier created.

NOTE: If not at least two, then run the log generator one, or more, short times until you do.

```
[maria_dev@sandbox ~]$ /usr/hdp/current/kafka-broker/bin/kafka-console-
consumer.sh --bootstrap-server sandbox.hortonworks.com:6667 --topic logs --
from-beginning
2017-06-04  15:51:38.71 48.257.926.101    INFO  Generic INFO # 18
2017-06-04  15:51:39.76 32.234.242.261    DEBUG Worthless DEBUG #1
2017-06-04  15:51:40.81 31.160.9.90 WARN  Generic WARN # 6
2017-06-04  15:51:41.87 57.118.219.201    INFO  Generic INFO # 14
2017-06-04  15:51:42.92 35.220.141.248    ERROR BAD ERROR # 2
2017-06-04  15:51:43.97 45.226.670.309    WARN  Generic WARN # 7
```

Execute the topology in your IDE and then verify that the number of applicable messages from the Kafka topic were stored into your HBase table as well as the appropriate row keys.

```
hbase(main):003:0> count 'incident'
3 row(s) in 0.3340 seconds

=> 3
hbase(main):004:0> scan 'incident'
ROW                   COLUMN+CELL

 31.160.9.90          column=event:details, timestamp=1496603855549,
value=Gener
```

```
                      ic WARN #
6
 31.160.9.90          column=event:total, timestamp=1496603855553,
value=\x00\x0
                      0\x00\x00\x00\x00\x00\x01

 31.160.9.90          column=event:type, timestamp=1496603855549,
value=WARN
 35.220.141.248       column=event:details, timestamp=1496603855570,
value=BAD E
                      RROR #
2
 35.220.141.248       column=event:total, timestamp=1496603855573,
value=\x00\x0
                      0\x00\x00\x00\x00\x00\x01

 35.220.141.248       column=event:type, timestamp=1496603855570,
value=ERROR
 45.226.670.309       column=event:details, timestamp=1496603855576,
value=Gener
                      ic WARN #
7
 45.226.670.309       column=event:total, timestamp=1496603855577,
value=\x00\x0
                      0\x00\x00\x00\x00\x00\x01

 45.226.670.309       column=event:type, timestamp=1496603855576,
value=WARN
3 row(s) in 0.0610 seconds

hbase(main):005:0>
```

Run the topology again and notice that the count stayed the same despite the data being processed twice.

```
hbase(main):005:0> count 'incident'
3 row(s) in 0.0080 seconds

=> 3
hbase(main):006:0>
```

Despite the fact that the data is duplicated, notice there are two distinct versions of all the column identifiers; even the counter.

```
hbase(main):006:0> get 'incident', '45.226.670.309', {COLUMN => 'event',
VERSIONS => 10}
COLUMN                CELL

 event:details        timestamp=1496603985605, value=Generic WARN #
7
 event:details        timestamp=1496603855576, value=Generic WARN #
7
 event:total          timestamp=1496603985608,
value=\x00\x00\x00\x00\x00\x00\x0
                      0\x02
 event:total          timestamp=1496603855577,
value=\x00\x00\x00\x00\x00\x00\x0
                      0\x01

 event:type           timestamp=1496603985605,
value=WARN
 event:type           timestamp=1496603855576,
value=WARN
6 row(s) in 0.0850 seconds

hbase(main):007:0>
```

This will be more interesting once multiple events for a given server are stored in HBase. Again, we will maintain the most recent 10 version of the type and details column identifiers and the most recent version of `event:total` will be the running total of all rows added for the particular server identified by the row key.

Ensure you have a `kafka-console-consumer.sh` script running.

```
[maria_dev@sandbox ~]$ /usr/hdp/current/kafka-broker/bin/kafka-console-
consumer.sh --bootstrap-server sandbox.hortonworks.com:6667 --topic logs --
from-beginning
2017-06-04  15:51:38.71 48.257.926.101    INFO  Generic INFO # 18
2017-06-04  15:51:39.76 32.234.242.261    DEBUG Worthless DEBUG #1
2017-06-04  15:51:40.81 31.160.9.90 WARN  Generic WARN # 6
2017-06-04  15:51:41.87 57.118.219.201    INFO  Generic INFO # 14
2017-06-04  15:51:42.92 35.220.141.248    ERROR BAD ERROR # 2
2017-06-04  15:51:43.97 45.226.670.309    WARN  Generic WARN # 7
```

Back in the Terminal window on the host where the desktop UI runs, kick off the log generator again at the rate of five new messages per second and notice the records being added to the topic.

```
[root@ip-172-30-0-164 log-gen]# pwd
/root/rtlabs/log-gen
[root@ip-172-30-0-164 log-gen]# ls
logData.txt  logSim.py  logSim.py.orig
[root@ip-172-30-0-164 log-gen]# python logSim.py 5 logs
Starting Kafka Broker
Printing  5  messages per second.  Press control-c to stop
```

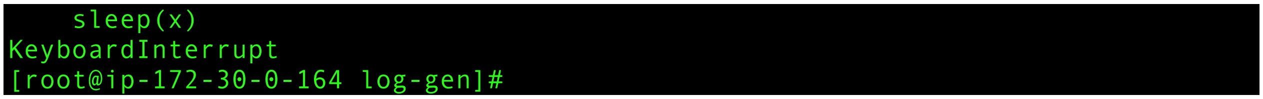
Verify that the number of rows in the HBase table are growing, too.

```
hbase(main):031:0> count 'incident'
162 row(s) in 0.0280 seconds

=> 162
hbase(main):032:0>
```

After you stop the topology from running within IntelliJ, be sure to halt the log generator as well.

```
[root@ip-172-30-0-164 log-gen]# python logSim.py 5 logs
Starting Kafka Broker
Printing  5  messages per second.  Press control-c to stop
^CTraceback (most recent call last):
  File "logSim.py", line 24, in <module>
```



```
    sleep(x)
KeyboardInterrupt
[root@ip-172-30-0-164 log-gen]#
```

“Extra Credit”

Add a `KafkaBolt` to your topology and publish the tuples emitted from the Log Filtering Bolt created earlier.

HINT: Visit “Apache Kafka Integration” page on the http://storm.apache.org Documentaiton page as well as visiting http://docs.hortonworks.com and drilling down into the “Streaming Data into Kafka” section of the “Apache Storm Component Guide”.

Summary

You have successful built, and tested, a Storm topology in the IDE that consumes data from Kafka and persists it in HBase.

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