APACHE PIG AND HADOOP WITH ELASTICSEARCH: THE ELASTICSEARCH-HADOOP CONNECTOR

Here we show how to retrieve data from ElasticSearch using Apache Pig. The reason for doing that is Pig is much easier to use that Java, Scala, and other tools for doing data extraction and transformation ElasticSearch. (You can read our introduction to Apache Pig here.) Also you can construct complex queries and sets using Pig that you could not with ES alone.

If you look on the internet, most of the examples you see, including those from ElasticSearch, explain how to **write** data to ElasticSearch (ES). For those who understand what ES does, that does not make much sense. ES is usually used together with Kibana and Logstash to store log data from applications. ES is a distributed database that stores documents in JSON format. But Apache Spark would be better suited to that.

(This article is part of our ElasticSearch Guide. Use the right-hand menu to navigate.)

The real power of the Hadoop-ElasticSearch plugin is to read data from logs for cybersecurity and operations purposes. It is common for companies to gather data in ELK for that purpose. But you cannot write complex queries there. But you can do complex queries with Pig and save the data in Hadoop, Spark, or ES and then apply analytics to that.

We won't explain how to install Hadoop and ELK here. You can get instructions for those from Hadoop and ElasticSearch. This article assumes some basic knowledge of ELK.

Instead we are doing to load some data in ElasticSearch and then use Apache Pig to query it.

Download the entirety of Shakespeare's plays from here. Granted these are not logs, but they are a good example for sample data and the same that many other tutorials use.

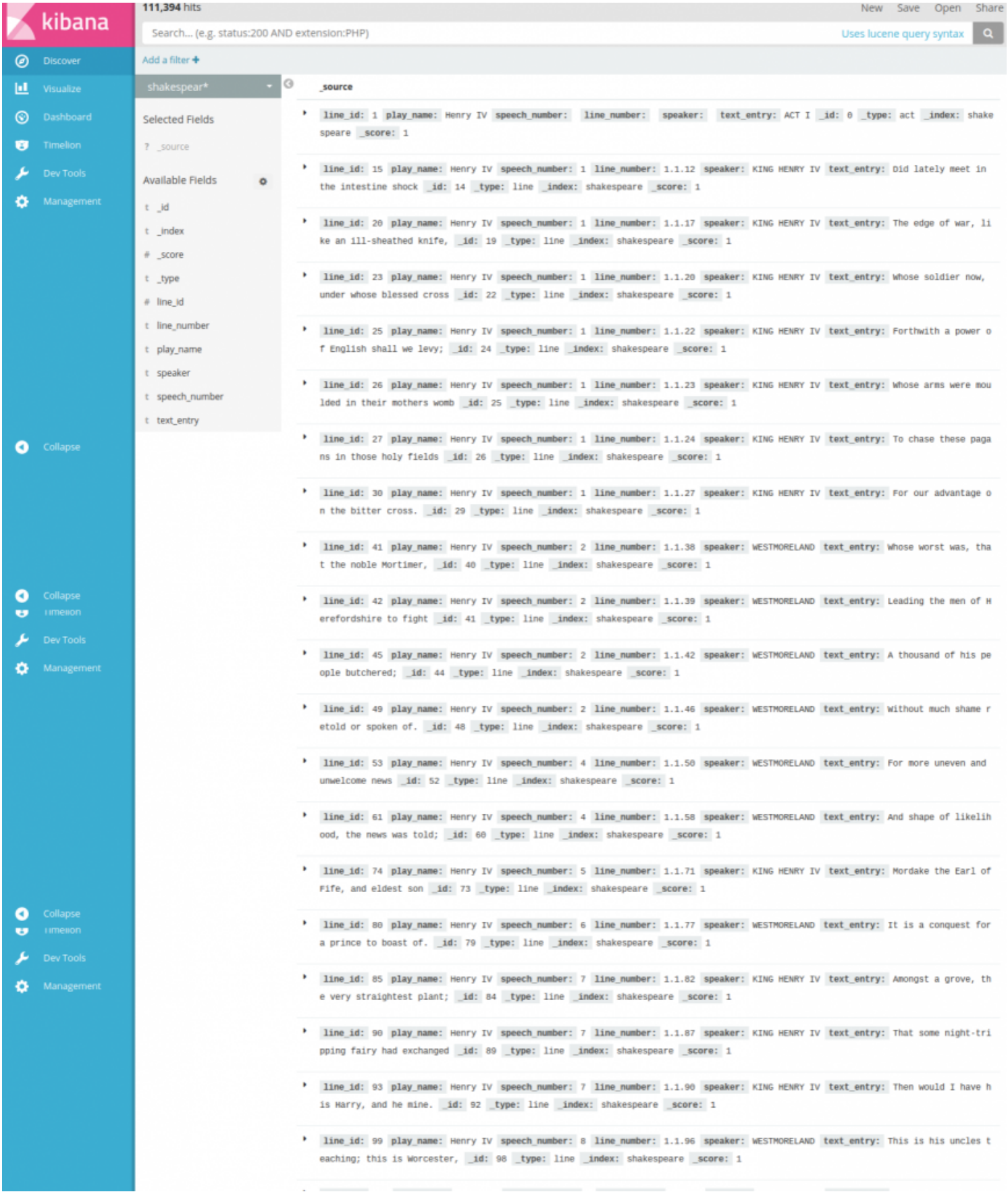
Each line looks like this:

```
{"index":{"_index":"shakespeare","_type":"line","_id":11}}
{"line_id":12,"play_name":"Henry
IV","speech_number":1,"line_number":"1.1.9","speaker":"KING HENRY
IV","text_entry":"Of hostile paces: those opposed ey
es,"}
```

Load that data into ES like this:

```
curl -XPUT localhost:9200/_bulk --data-binary @shakespeare.json
```

Then when you open Kibana you should see the data like this, under the **shakespeare** index.



Now download the last files elasticsearch-hadoop-5.5.2.jar and commons-httpclient-3.1.jar from Maven.

Then start Pig in local mode (or cluster if that is what you have). (You can make life easier if you run everything as root. Note that you cannot run ElasticSearch as root.)

```
pig -x local
```

This will open the Pig shell. So that those jars come into scope, enter these two commands into the shell:

```
REGISTER /home/walker/Documents/jars/elasticsearch-hadoop-5.5.2.jar
REGISTER /home/hadoop/Documents/jars/commons-httpclient-3.1.jar
```

Now, define a shortcut for ES storage like this:

```
DEFINE EsStorage org.elasticsearch.hadoop.pig.EsStorage();
```

There are lots of options you could pass to that like:

```
DEFINE EsStorage org.elasticsearch.hadoop.pig.EsStorage (
'es.http.timeout= 5m',
'es.index.auto.create = true',
'es.mapping.pig.tuple.use.field.names = true',
'es.mapping.id = id'
);
```

Now load (some of) the data into Pig from ElasticSearch.

```
a = LOAD 'shakespeare' USING
org.elasticsearch.hadoop.pig.EsStorage('es.query=?q=wine');
```

What we have done is use the Lucene (very simple, natural-language) query ability of ES to load every line in the play that has the word **wine** in it. (If you've read much Shakespeare you know they also call it **slack**.)

The result we get is a series of tuples.

ES has no schema since its storage format is JSON. Neither does a tuple.

```
(47371,Julius Caesar,32,2.2.134,CAESAR,Good friends, go in, and taste some
wine with me;)
(64337,Merry Wives of Windsor,83,1.1.165,PAGE,Nay, daughter, carry the wine
in; well drink within.)
(65573,Merry Wives of Windsor,32,3.2.79,FORD, I think I shall drink in pipe
wine first)
```

Now we can load the data with a schema like this:

```
b = LOAD 'shakespeare' USING
org.elasticsearch.hadoop.pig.EsStorage('es.query=?q=wine') as
(line_id:string:chararray, play_name:chararray, speech_number:int,
line_number:chararray, speaker:chararray, text_entry:chararray);
```

Then we ask Pig to show us the schema:

```
describe b
b: {line_id: chararray,play_name: chararray,speech_number: int,line_number:
chararray,speaker: chararray,text_entry: chararray}
```

When you are done with your dataset running queries and transformations you could load save it into Pig (meaning Hadoop) as when you close the Pig shell you would lose it.