Elasticsearch & Lucene for Apache Spark and MLlib

Costin Leau (@costinl)

elastic

SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO

Mirror, mirror on the wall, what's the happiest team of us all ?

Briita Weber
- Rough translation from German by yours truly -

Purpose of the talk

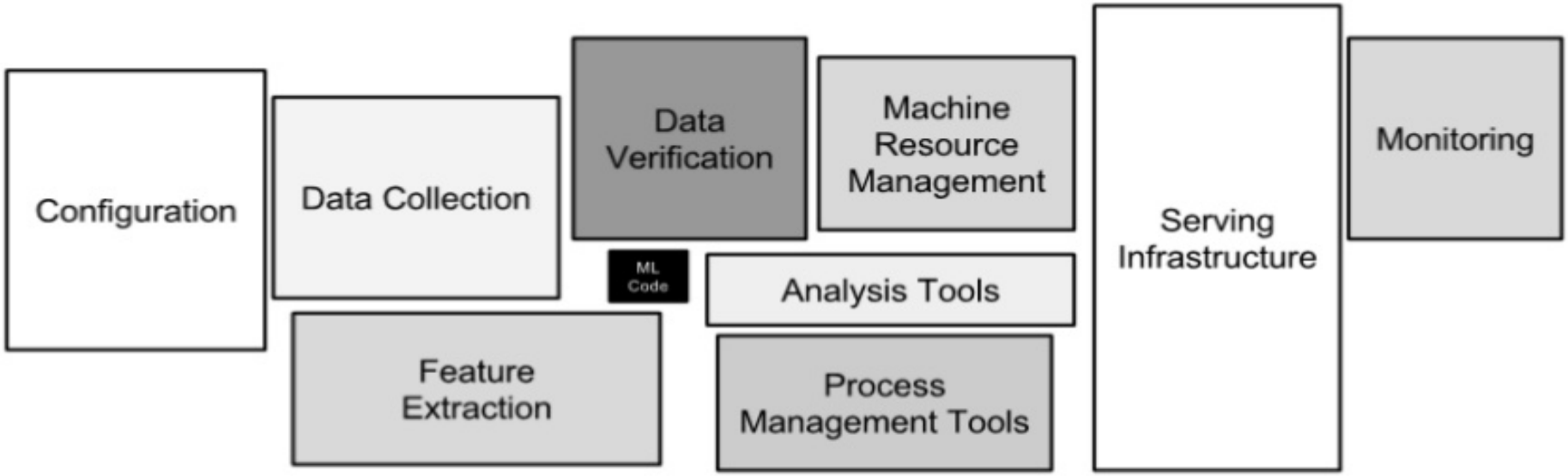
Improve ML pipelines through IR

Text processing

- Analysis
- Featurize/Vectorize *

* In research / poc / WIP / Experimental phase

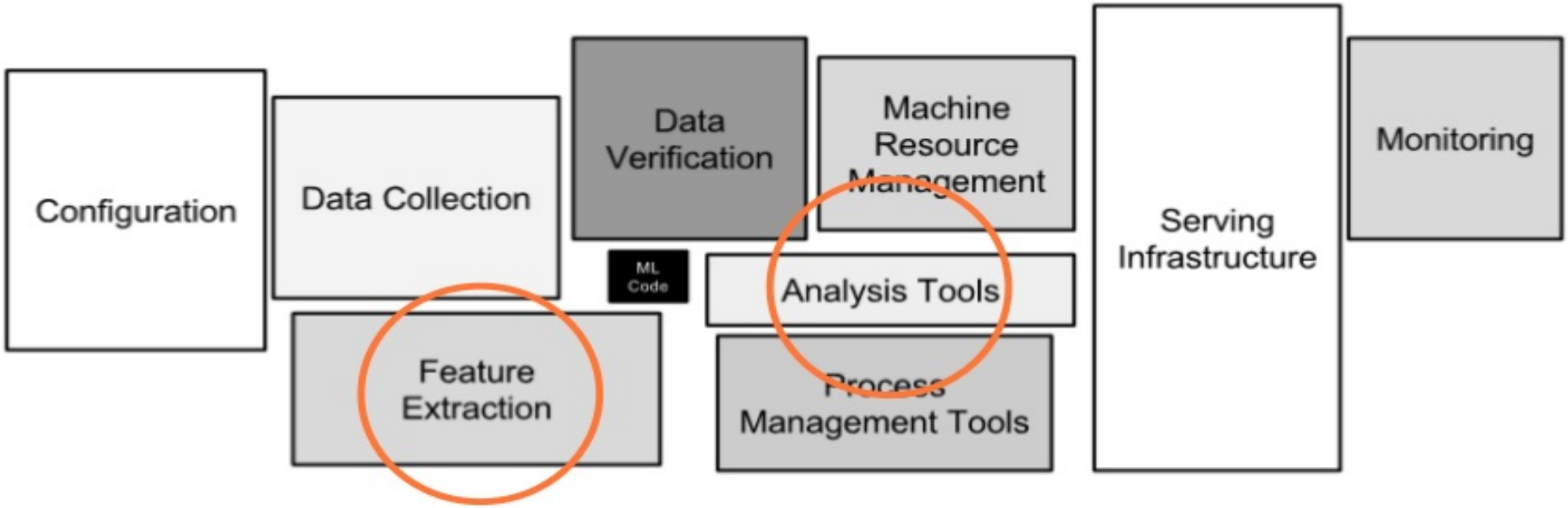
Technical Debt



Machine Learning: The High Interest Credit Card of Technical Debt", Sculley et al

http://research.google.com/pubs/pub43146.html

Technical Debt



Machine Learning: The High Interest Credit Card of Technical Debt", Sculley et al

http://research.google.com/pubs/pub43146.html

Challenge

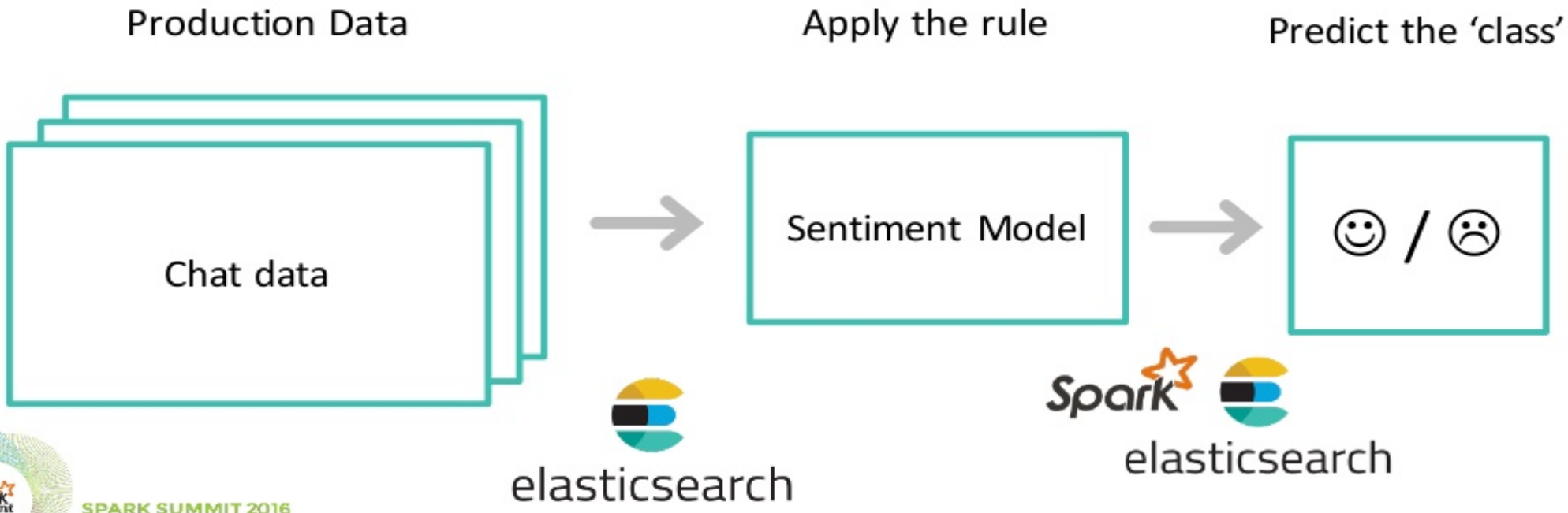
Challenge: What team at Elastic is most happy?

Data: Hipchat messages

Training / Test data: http://www.sentiment140.com

Result: Kibana dashboard

ML Pipeline


Production Data
Apply the rule
Predict the 'class'
Chat data
Sentiment Model
☺ / ☹
elasticsearch
Spark
elasticsearch

Data is King

Example: Word2Vec

Input snippet

```
it was introduced into mathematics in the book
disquisitiones arithmeticae by carl friedrich gauss in
one eight zero one ever since however modulo has gained
many meanings some exact and some imprecise
```

http://spark.apache.org/docs/latest/mllib-feature-extraction.html#example

Real data is *messy*

originally looked like this:

```
It was introduced into <a
href="https://en.wikipedia.org/wiki/Mathematics"
title="Mathematics">mathematics</a> in the book <i><a
href="https://en.wikipedia.org/wiki/Disquisitiones_Arithmeticae"
title="Disquisitiones Arithmeticae">Disquisitiones Arithmeticae</a></i>
by <a href="https://en.wikipedia.org/wiki/Carl_Friedrich_Gauss"
title="Carl Friedrich Gauss">Carl Friedrich Gauss</a> in 1801. Ever
since, however, "modulo" has gained many meanings, some exact and some
imprecise.
```

https://en.wikipedia.org/wiki/Modulo_(jargon)

~~Feature extraction~~ Cleaning up data

```
"huuuuuuunnnnnnngrrryyy",
"aaaaaamaaazingggggg",
"aaaaaamazing",
"aaaaaammm",
"aaaaaammmazzzingggg",
"aaaaaamy",
"aaaaaan",
"aaaaaand",
"aaaaaannnnnnddd",
"aaaaaanyways"
```

Does it help to clean that up?
see "Twitter Sentiment Classification using Distant Supervision", Go et al.
http://www-cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf

Language matters

读书须用意，一字值千金

Lucene to the rescue!

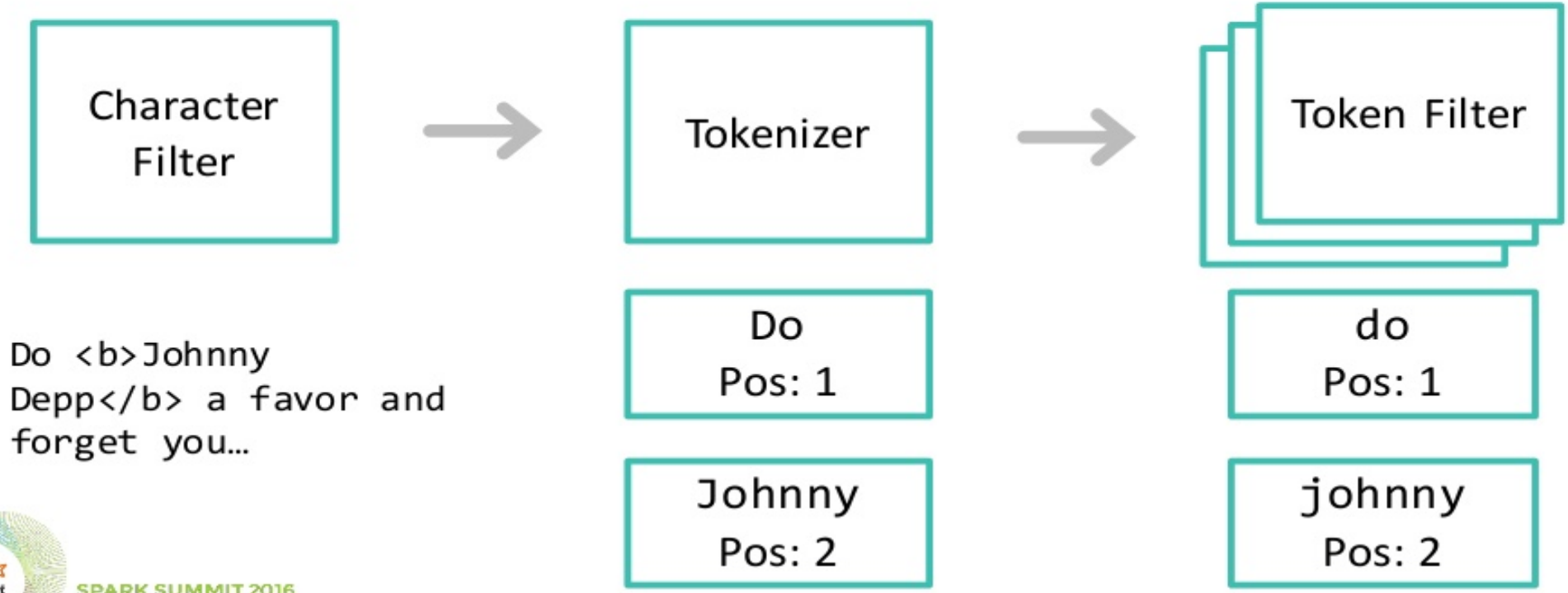
High-performance, full-featured text search library

15 years of experience

Widely recognized for its utility

- It's a primary test bed for new JVM versions

Text processing



Lucene for text analysis

state of the art text processing

many extensions available for different languages, use cases,…

however…

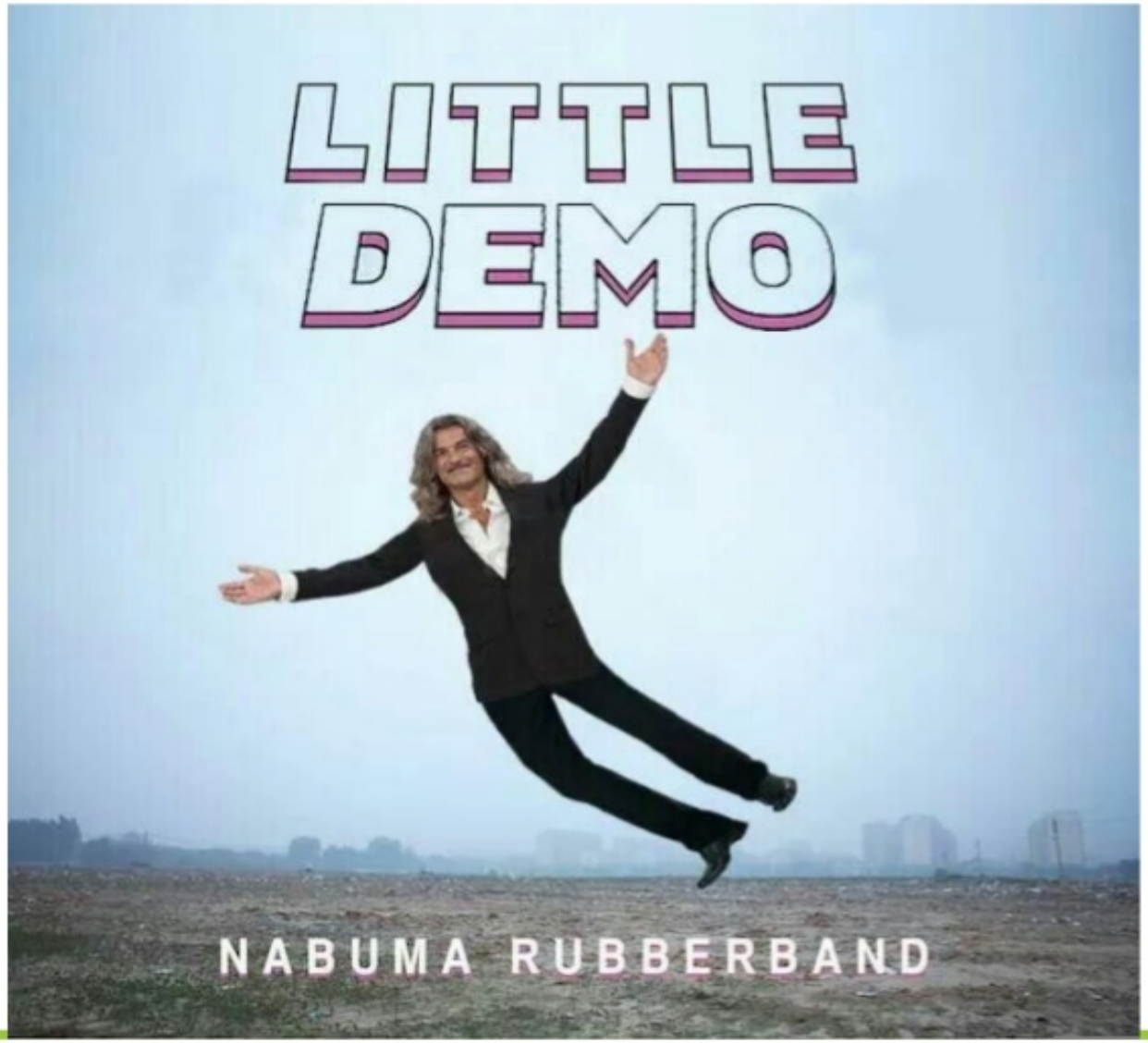
```
…
import org.apache.lucene.analysis…
…

Analyzer a = new Analyzer() {
  @Override
  protected TokenStreamComponents createComponents(String fieldName) {
    Tokenizer tokenizer = new StandardTokenizer();
    return new TokenStreamComponents(tokenizer, tokenizer);
  }

  @Override
  protected Reader initReader(String fieldName, Reader reader) {
    return new HTMLStripCharFilter(reader);
  }
};

TokenStream stream = a.tokenStream(null, "<a href=...>some text</a>");
CharTermAttribute term = stream.addAttribute(CharTermAttribute.class);
PositionIncrementAttribute posIncrement = stream.addAttribute(PositionIncrementAttribute.class);
stream.reset();
int pos = 0;
while (stream.incrementToken()) {
  pos += posIncrement.getPositionIncrement();
  System.out.println(term.toString() + " " + pos);
}
```

```
> some 1
> text 2
```

```
…
import org.apache.lucene.analysis…
…

Analyzer a = new Analyzer() {
  @Override
  protected TokenStreamComponents createComponents(String fieldName) {
    Tokenizer tokenizer = new StandardTokenizer();
    return new TokenStreamComponents(tokenizer, tokenizer);
  }

  @Override
  protected Reader initReader(String fieldName, Reader reader) {
    return new HTMLStripCharFilter(reader);
  }
};

TokenStream stream = a.tokenStream(null, "<a href=...>some text</a>");
CharTermAttribute term = stream.addAttribute(CharTermAttribute.class);
PositionIncrementAttribute posIncrement = stream.addAttribute(PositionIncrementAttribute.class);
stream.reset();
int pos = 0;
while (stream.incrementToken()) {
  pos += posIncrement.getPositionIncrement();
  System.out.println(term.toString() + " " + pos);
}
```

```
> some 1
> text 2
```

How about a declarative approach?


LITTLE
DEMO
NABUMA RUBBERBAND

Very quick intro to Elasticsearch

SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO

Elasticsearch in ~~5~~ 3’

Scalable, real-time search and analytics engine

Data distribution, cluster management

REST APIs

JVM based, uses Apache Lucene internally

Open-source (on Github, Apache 2 License)

Elasticsearch in 3’

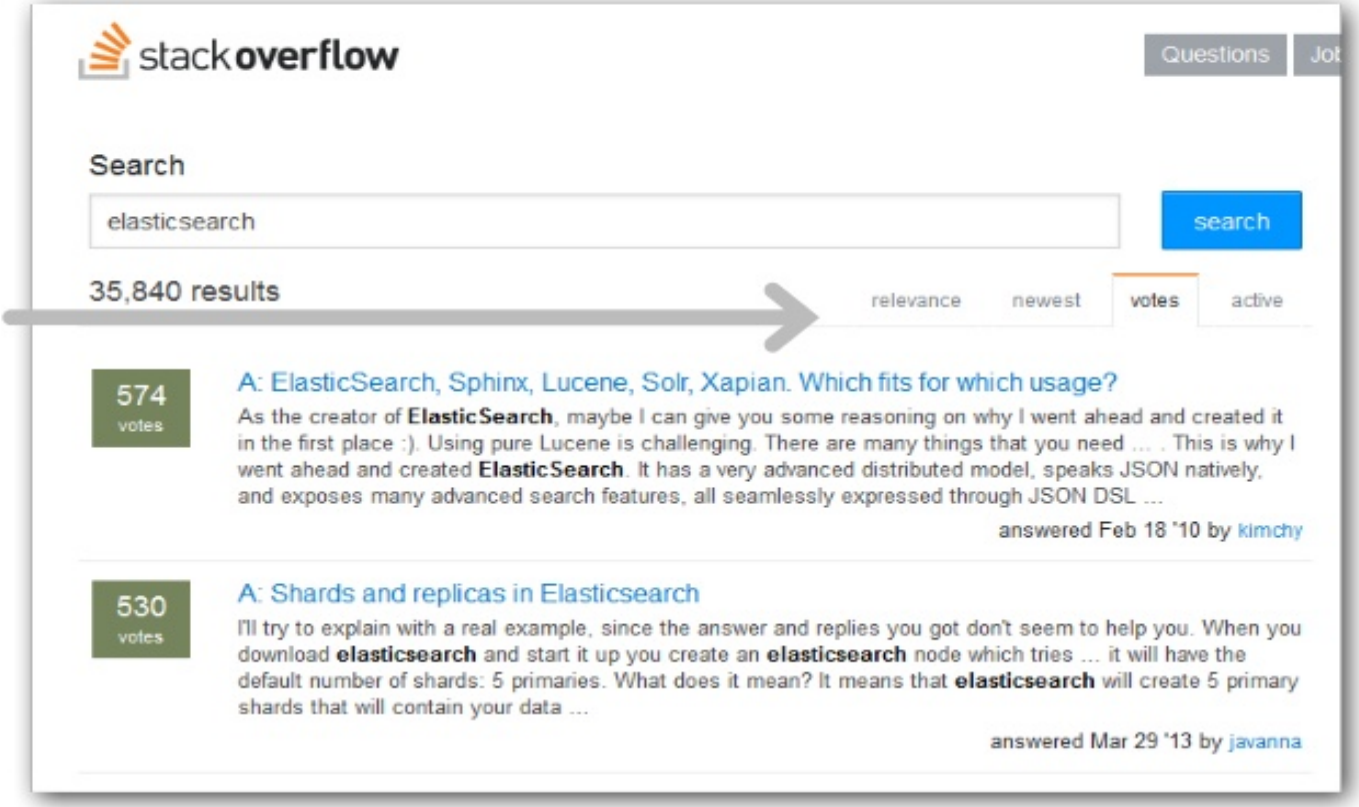
elasticsearch

Unstructured search

Elasticsearch in 3’

elasticsearch

Sorting / Scoring



Elasticsearch in 3’

elasticsearch

Pagination

stackoverflow

Questions Jo

Search

elasticsearch

search

35,840 results

relevance newest votes active

574 votes

A: ElasticSearch, Sphinx, Lucene, Solr, Xapian. Which fits for which usage?

As the creator of **ElasticSearch**, maybe I can give you some reasoning on why I went ahead and created it in the first place :). Using pure Lucene is challenging. There are many things that you need ... . This is why I went ahead and created **ElasticSearch**. It has a very advanced distributed model, speaks JSON natively, and exposes many advanced search features, all seamlessly expressed through JSON DSL ...

answered Feb 18 '10 by kimchy

530 votes

A: Shards and replicas in Elasticsearch

I'll try to explain with a real example, since the answer and replies you got don't seem to help you. When you download **elasticsearch** and start it up you create an **elasticsearch** node which tries ... it will have the default number of shards: 5 primaries. What does it mean? It means that **elasticsearch** will create 5 primary shards that will contain your data ...

answered Mar 29 '13 by javanna

1 2 3 4 5 ... 2390 next

15 30 50 per page

Elasticsearch in 3’

Enrichment

Search

elasticsearch

Repositories 8,033
Code 798,062
Issues 57,377
16

Languages

Java	1,349
JavaScript	1,015
Shell	828
Python	804
Ruby	783
PHP	405
Go	187
C#	174
Scala	165
HTML	99

We’ve found 8,033 repository results

elastic/ elasticsearch
Open Source, Distributed, RESTful Search Engine
Updated 8 hours ago

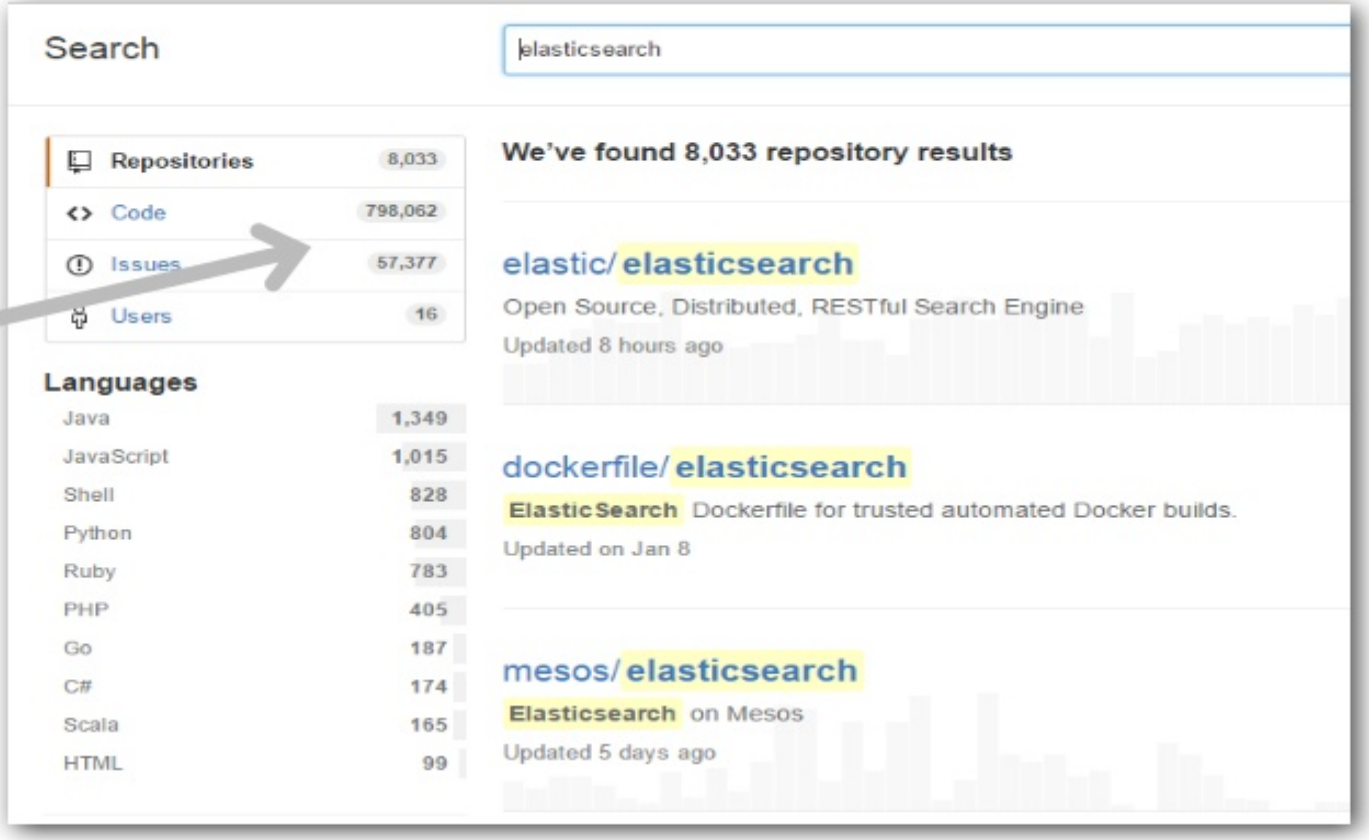
dockerfile/ elasticsearch
ElasticSearch Dockerfile for trusted automated Docker builds.
Updated on Jan 8

mesos/ elasticsearch
Elasticsearch on Mesos
Updated 5 days ago

Elasticsearch in 3’

elasticsearch

Structured search



Elasticsearch in 3’

elasticsearch

https://www.elastic.co/elasticon/2015/sf/unlocking-interplanetary-datasets-with-real-time-search

Machine Learning and Elasticsearch

SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO

Machine Learning and Elasticsearch

Machine Learning and Elasticsearch

Term Analysis (tf, idf, bm25)

Graph Analysis

Co-occurrence of Terms (significant terms)

- ChiSquare

Pearson correlation (#16817)

Regression (#17154)

What about classification/clustering/ etc… ?

It’s not the matching data, but the meta that lead to it

elastic

“

How to use Elasticsearch from Spark ?

Somebody on Stackoverflow

Elasticsearch for Apache Hadoop ™

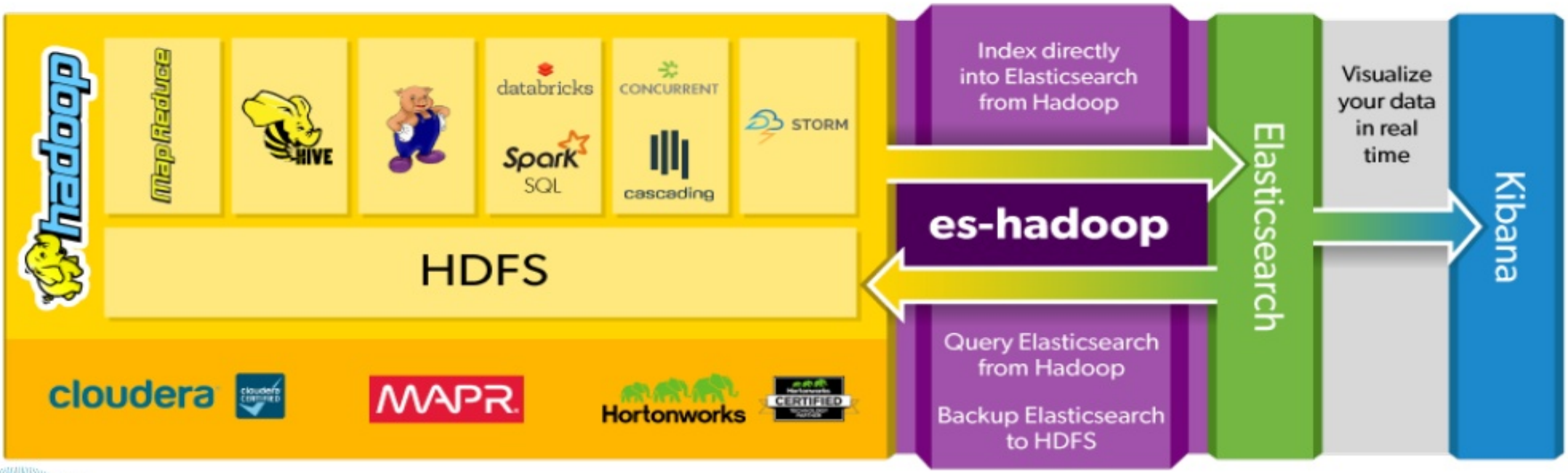
elasticsearch-hadoop

Java ★ 670 362

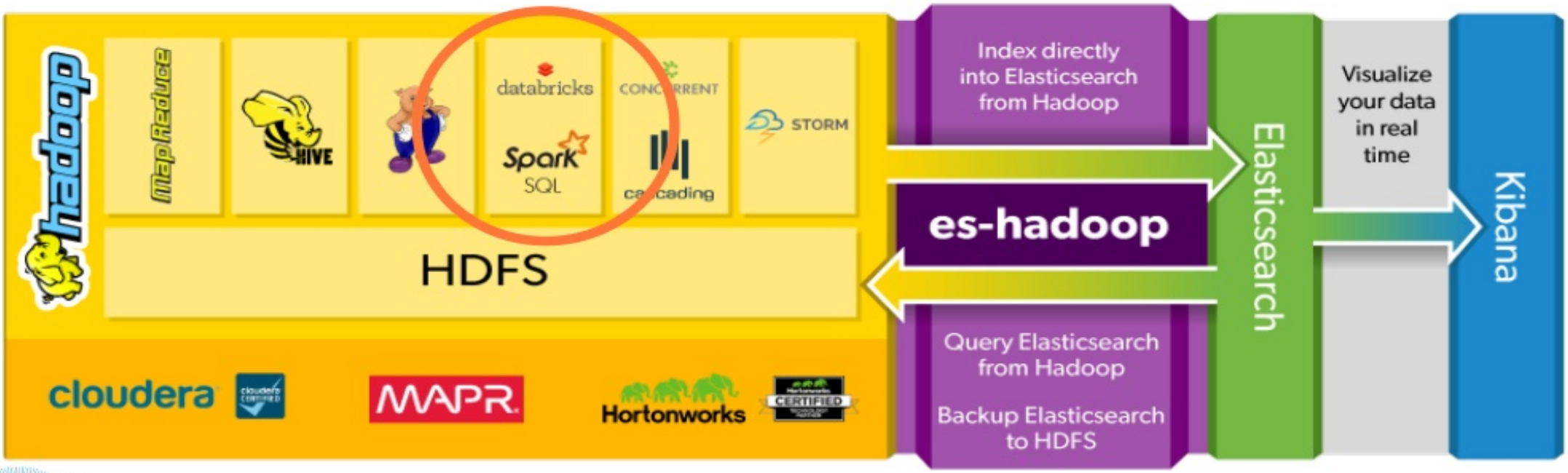
Elasticsearch real-time search and analytics natively integrated with Hadoop

Updated 3 hours ago

Elasticsearch for Apache Hadoop ™



Elasticsearch for Apache Hadoop ™



Elasticsearch Spark – Native integration

Spark Packages

Scala & Java API

Understands Scala & Java types

- Case classes
- Java Beans

Available as Spark package

Supports Spark Core & SQL

all 1.x version (1.0-1.6)

Available for Scala 2.10 and 2.11

elasticsearch-hadoop (homepage)

Official integration between Apache Spark and Elasticsearch real-time search and analytics

@elastic / (3)

Native Java/Scala API for Elasticsearch in Spark. Read/write RDDs and DataFrames from/to Elasticsearch.
Reference documentation available at https://www.elastic.co/guide/en/elasticsearch/hadoop/current/spark.html
Note that artifacts for Scala 2.11 are also available - simply use the _2.11 suffix instead in the artifact id.

Your rating

Tags

1 sql | 1 elasticsearch | 1 search | 1 analytics | 1 realtime | 1 core | 1 data source

How to [+]

Include this package in your Spark Applications using:

spark-shell, pyspark, or spark-submit

```
> $SPARK_HOME/bin/spark-shell --packages org.elasticsearch:elasticsearch-spark_2.10:5.0.0-alpha3
```

Releases

Version: 5.0.0-alpha3 (5e5a7d | zip | jar) / Date: 2016-06-01 / License: Apache-2.0 / Scala version: 2.10

Version: 2.3.2 (cae8b7 | zip | jar) / Date: 2016-05-22 / License: Apache-2.0 / Scala version: 2.10

Elasticsearch as RDD / Dataset*

```
import org.elasticsearch.spark._

val sc = new SparkContext(new SparkConf())
val rdd = sc.esRDD("buckethead/albums", "?q=pikes")
```

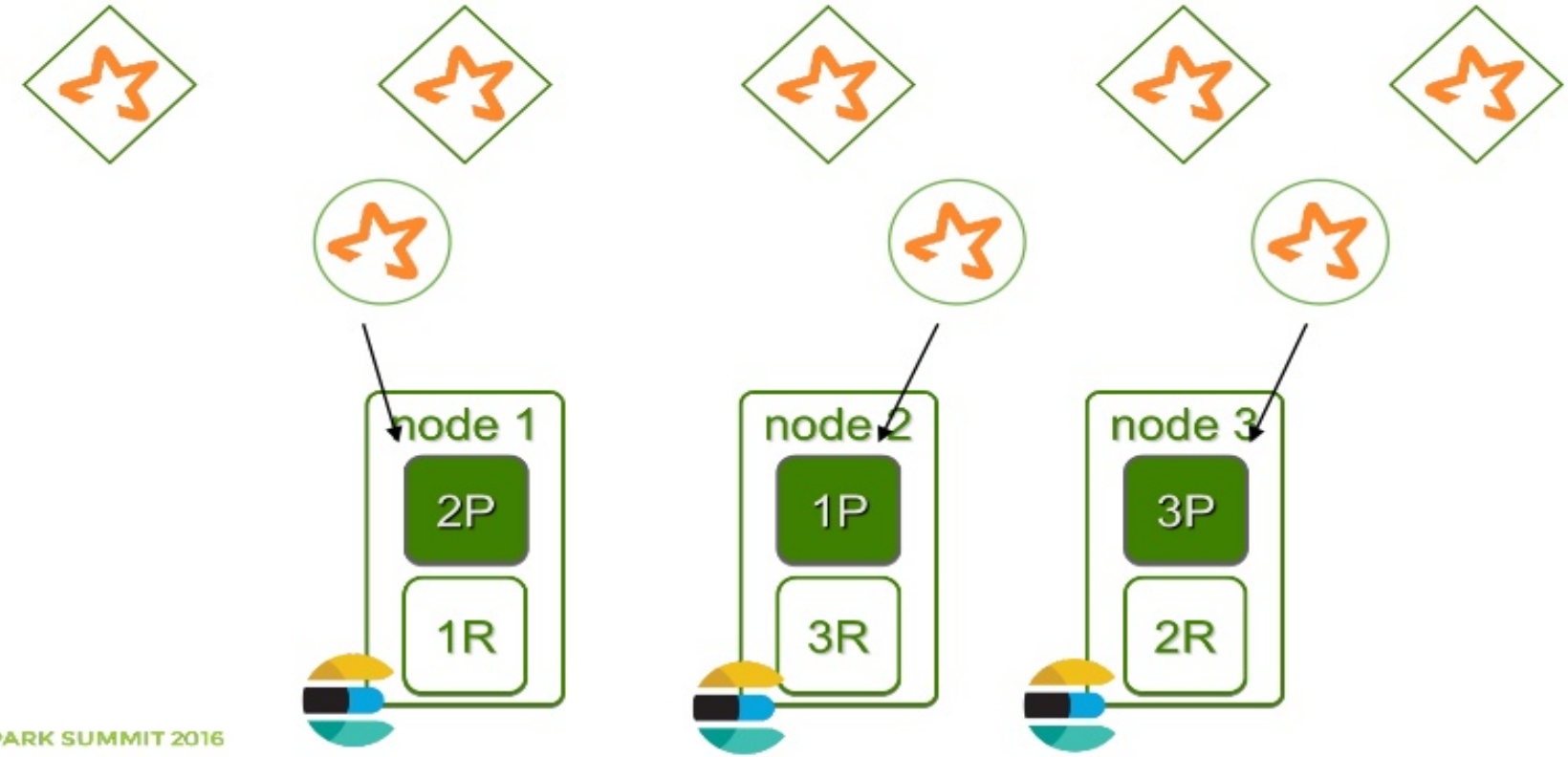
```
import org.elasticsearch.spark._

case class Artist(name: String, albums: Int)

val u2 = Artist("U2", 13)
val bh = Map("name"->"Buckethead","albums" -> 255, "age" -> 46)

sc.makeRDD(Seq(u2, bh)).saveToEs("radio/artists")
```

Elasticsearch as a DataFrame

```
val df = sql.read.format("es").load("buckethead/albums")

df.filter(df("category").equalTo("pikes").and(df("year").geq(2015)))
```

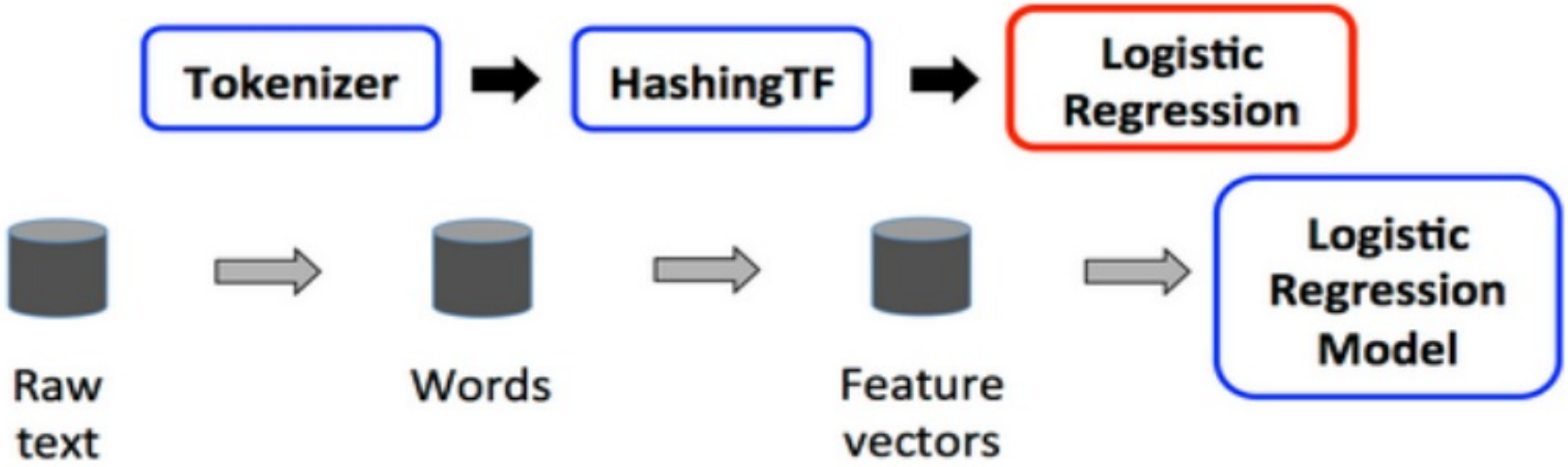
```
{ "query" :
   { "bool" : { "must" : [
      "match" : { "category" : "pikes" }
    ],
    "filter" : [
      { "range" : { "year" : {"gte" : "2015" }}}
    ]
}}}
```

Partition to Partition Architecture

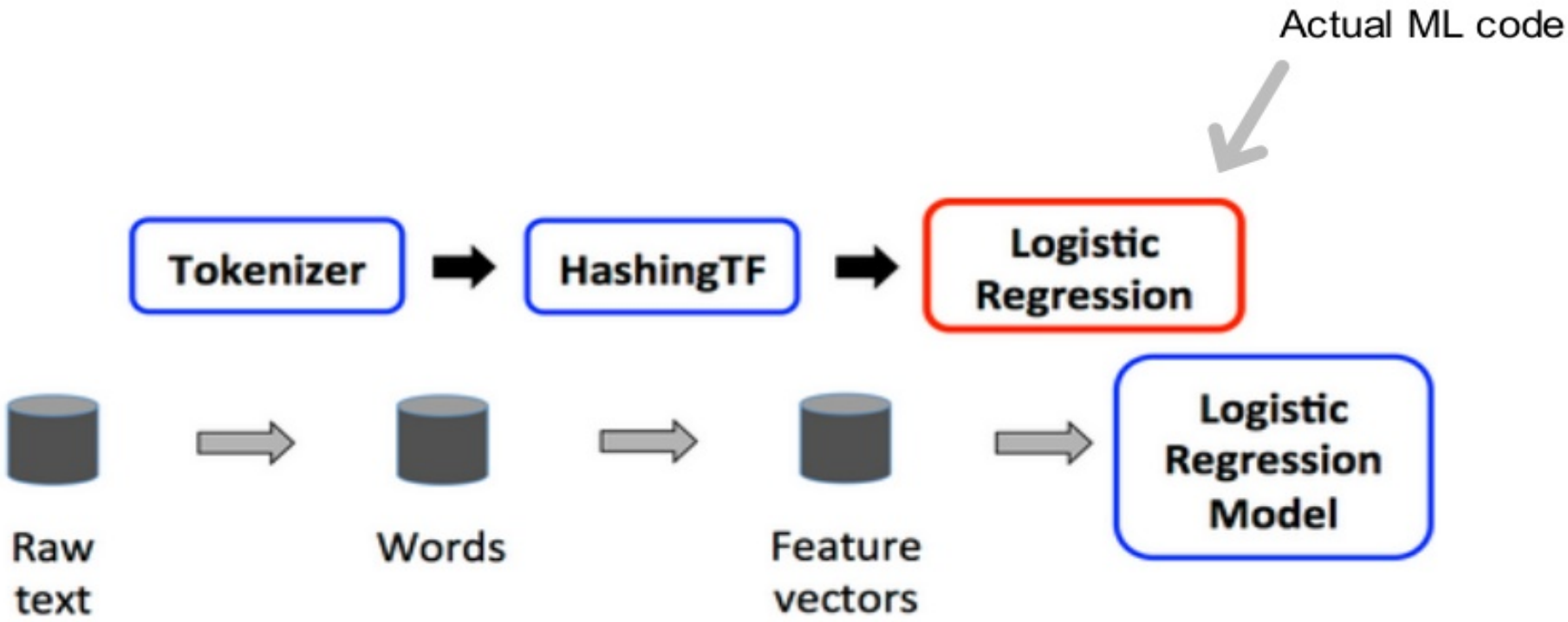


Putting the pieces together

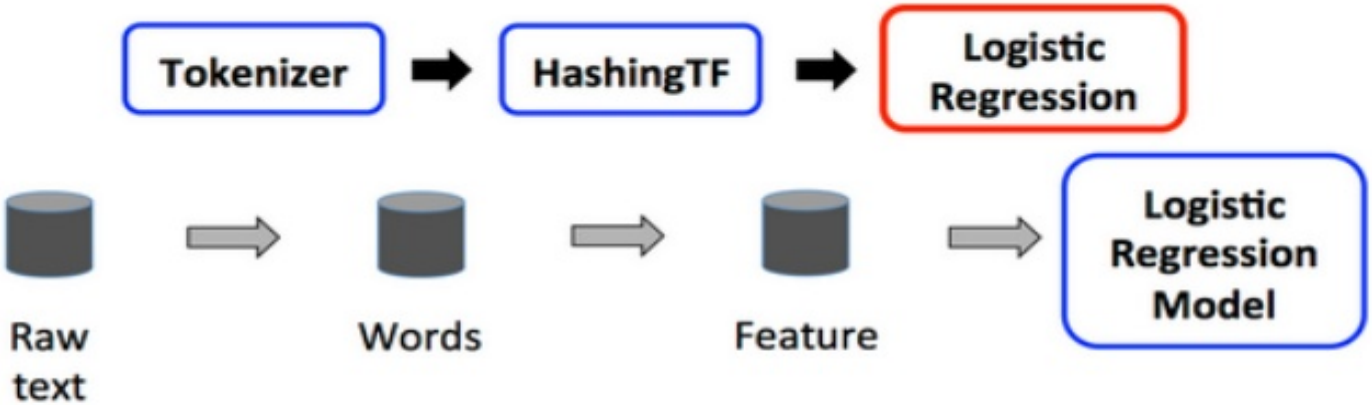
Typical ML pipeline for text

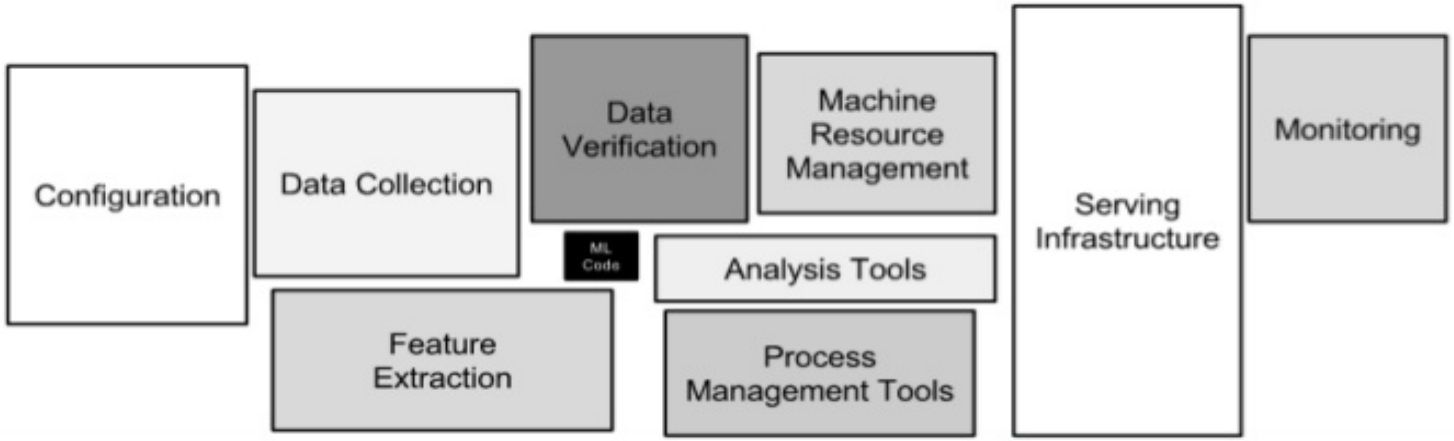


Typical ML pipeline for text



Typical ML pipeline for text





Pure Spark MLlib

```
val training = movieReviewsDataTrainingData

val tokenizer = new Tokenizer()
    .setInputCol("text")
    .setOutputCol("words")
val hashingTF = new HashingTF()
    .setNumFeatures(1000)
    .setInputCol(tokenizer.getOutputCol)
    .setOutputCol("features")
val lr = new LogisticRegression()
    .setMaxIter(10)
    .setRegParam(0.001)
val pipeline = new Pipeline()
    .setStages(Array(tokenizer, hashingTF, lr))

val model = pipeline.fit(training)
```

Pure Spark MLlib

```
val tokenizer = new Tokenizer()
    .setInputCol("text")
    .setOutputCol("words")
val hashingTF = new HashingTF()
    .setNumFeatures(1000)
    .setInputCol(tokenizer.getOutputCol)
    .setOutputCol("features")
val lr = new LogisticRegression()
    .setMaxIter(10)
    .setRegParam(0.001)
```

Pure Spark MLlib

```
val tokenizer = new Tokenizer()
    .setInputCol("text")
    .setOutputCol("words")
val hashingTF = new HashingTF()
    .setNumFeatures(1000)
    .setInputCol(tokenizer.getOutputCol)
    .setOutputCol("features")
val lr = new LogisticRegression()
    .setMaxIter(10)
    .setRegParam(0.001)
```

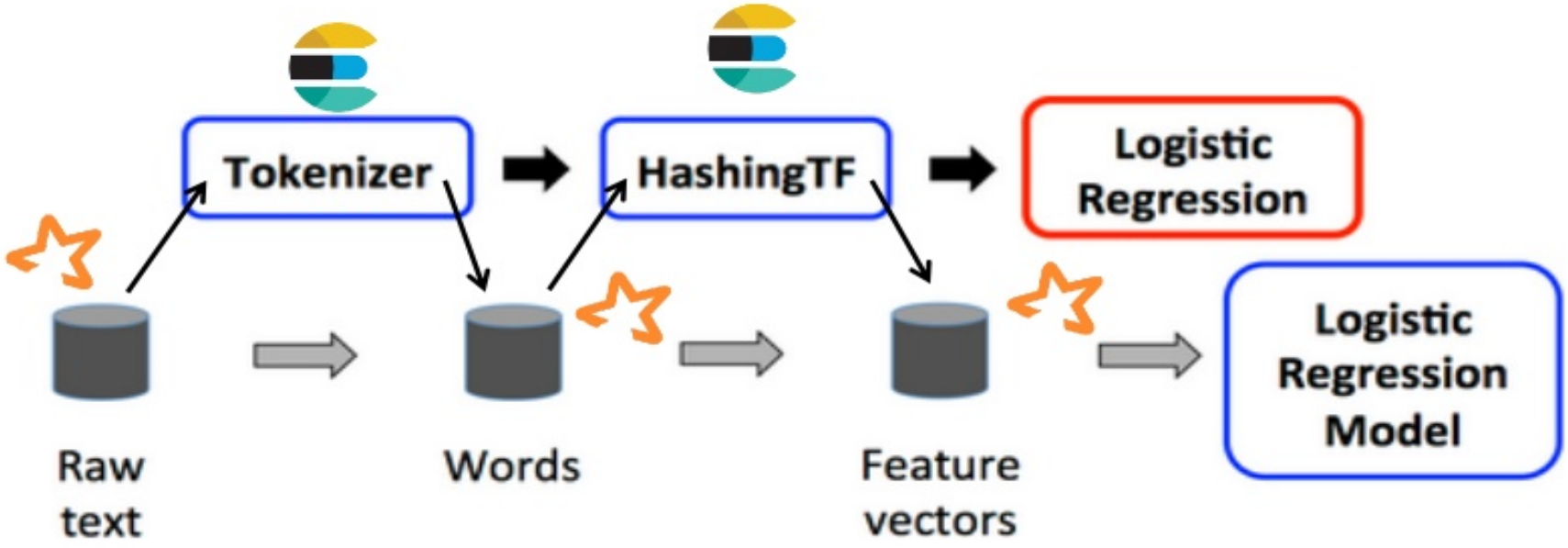
Pure Spark MLlib

```
val analyzer = new ESAnalyzer()
    .setInputCol("text")
    .setOutputCol("words")
val hashingTF = new HashingTF()
    .setNumFeatures(1000)
    .setInputCol(tokenizer.getOutputCol)
    .setOutputCol("features")
val lr = new LogisticRegression()
    .setMaxIter(10)
    .setRegParam(0.001)
```

Pure Spark MLlib

```
val analyzer = new ESAnalyzer()
    .setInputCol("text")
    .setOutputCol("words")
val hashingTF = new HashingTF()
    .setNumFeatures(1000)
    .setInputCol(tokenizer.getOutputCol)
    .setOutputCol("features")
val lr = new LogisticRegression()
    .setMaxIter(10)
    .setRegParam(0.001)
```

Data movement



Work once – reuse multiple times

```
// index / analyze the data

training.saveToEs("movies/reviews")
```

Work once – reuse multiple times

```
// prepare the spec for vectorize – fast and lightweight

val spec = s"""{ "features" : [{
        | "field": "text",
        | "type" : "string",
        | "tokens" : "all_terms",
        | "number" : "occurrence",
        | "min_doc_freq" : 2000
        | }],
        | "sparse" : "true"}""".stripMargin

ML.prepareSpec(spec, “my-spec”)
```

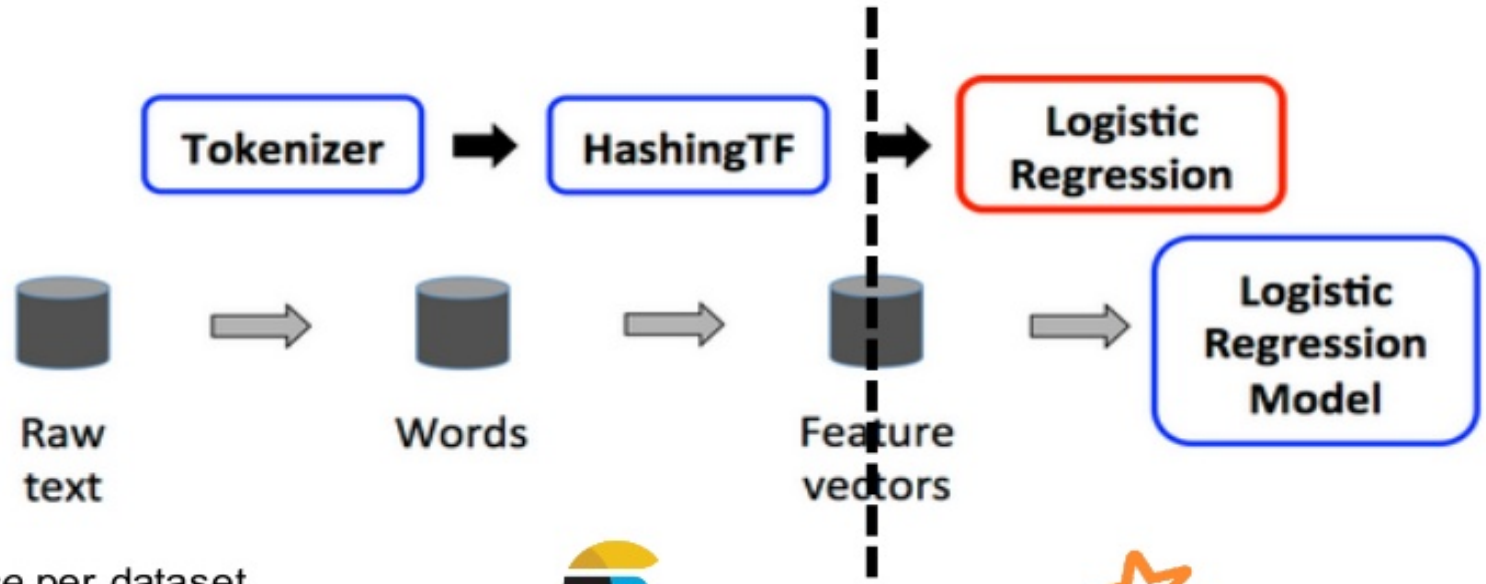
Access the vector directly

```
// get the features – just another query

val payload = s"""{"script_fields" : { "vector" :
  | { "script" : { "id" : “my-spec","lang" : “doc_to_vector" } }
  | }}""".stripMargin

// index the data
vectorRDD = sparkCtx.esRDD("ml/data", payload)

// feed the vector to the pipeline
val vectorized = vectorRDD.map ( x =>
  // get indices, the vector and length
  (if (x._1 == "negative") 0.0d else 1.0d, ML.getVectorFrom(x._2))
).toDF("label", "features")
```

Revised ML pipeline

```
val vectorized = vectorRDD.map...

val lr = new LogisticRegression()
      .setMaxIter(10)
      .setRegParam(0.001)

val model = lr.fit(vectorized)
```

Simplify ML pipeline

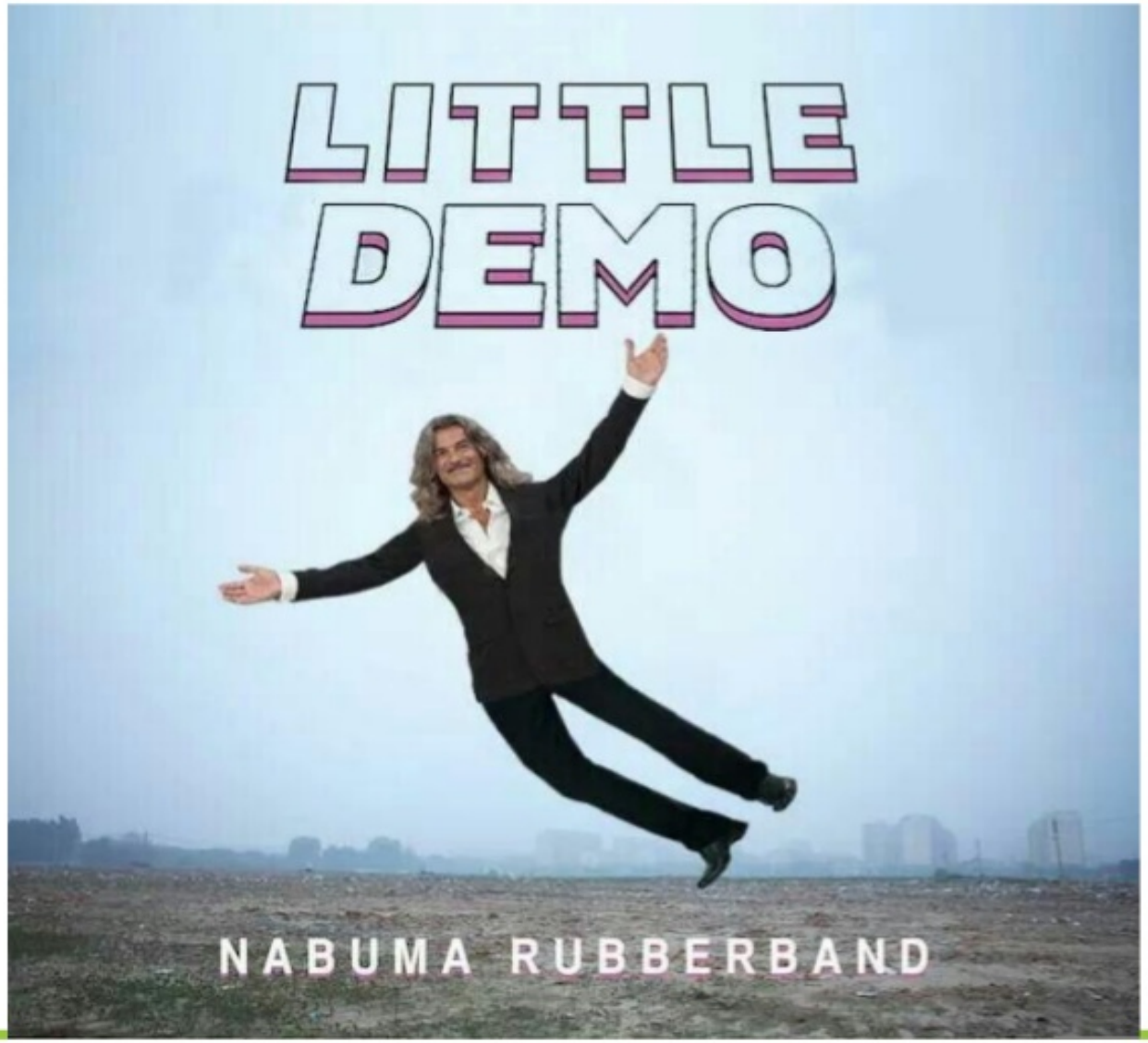


Once per dataset, regardless of # of pipelines

Raw data is not required any more

Need to adjust the model? Change the spec

```
val spec = s"""{ "features" : [{
        | "field": "text",
        | "type" : "string",
        | "tokens" : "given",
        | "number" : "tf",
        | "terms": ["term1", "term2", ...]
        | }],
        | "sparse" : "true"}""".stripMargin

ML.prepareSpec(spec)
```


LITTLE
DEMO
NABUMA RUBBERBAND

All this is WIP

Not all features available (currently dictionary, vectors)

Works with data outside or inside Elasticsearch (latter is **much** faster)

Bind vectors to queries

Other topics WIP:

Focused on document / text classification – numeric support is next

Model importing / exporting – Spark 2.0 ML persistence

Feedback highly sought - Is this useful?

THANK YOU.

j.mp/spark-summit-west-16
elastic.co/hadoop
github.com/elastic | costin | brwe
discuss.elastic.co
@costinl