Recommender Systems

An Elastic Training Course

0.0.1

elastic.co/training

6.x.x

Recommender Systems

Course: Data Science Specialization - Recommender Systems

Version 0.0.1

elastic

Lab Prep Checklist

- Visit training.elastic.co and create an account
  - **follow email instructions**
- Go to "My Account" and click on today's training
- Download the PDF file (this contains all the slides)
- Click on "Virtual Link (Strigo)" to access the Lab Environment
  - You will need an access token, which your instructor will provide

1. Eco-system
2. Elastic Graph
3. Spark Collaborative Filtering

Agenda and Introductions

elastic

Course Agenda

1. Apache Eco-system & Review
2. Elastic Graph
3. Spark Collaborative Filtering

Introductions

- Name
- Company
- What do you do?
- What are you using Elasticsearch for?
- What do you hope to get out of this training?

Logistics

- Facilities
- Emergency Exits
- Restrooms
- Breaks/Lunch

1 Eco-system

2 Elastic Graph

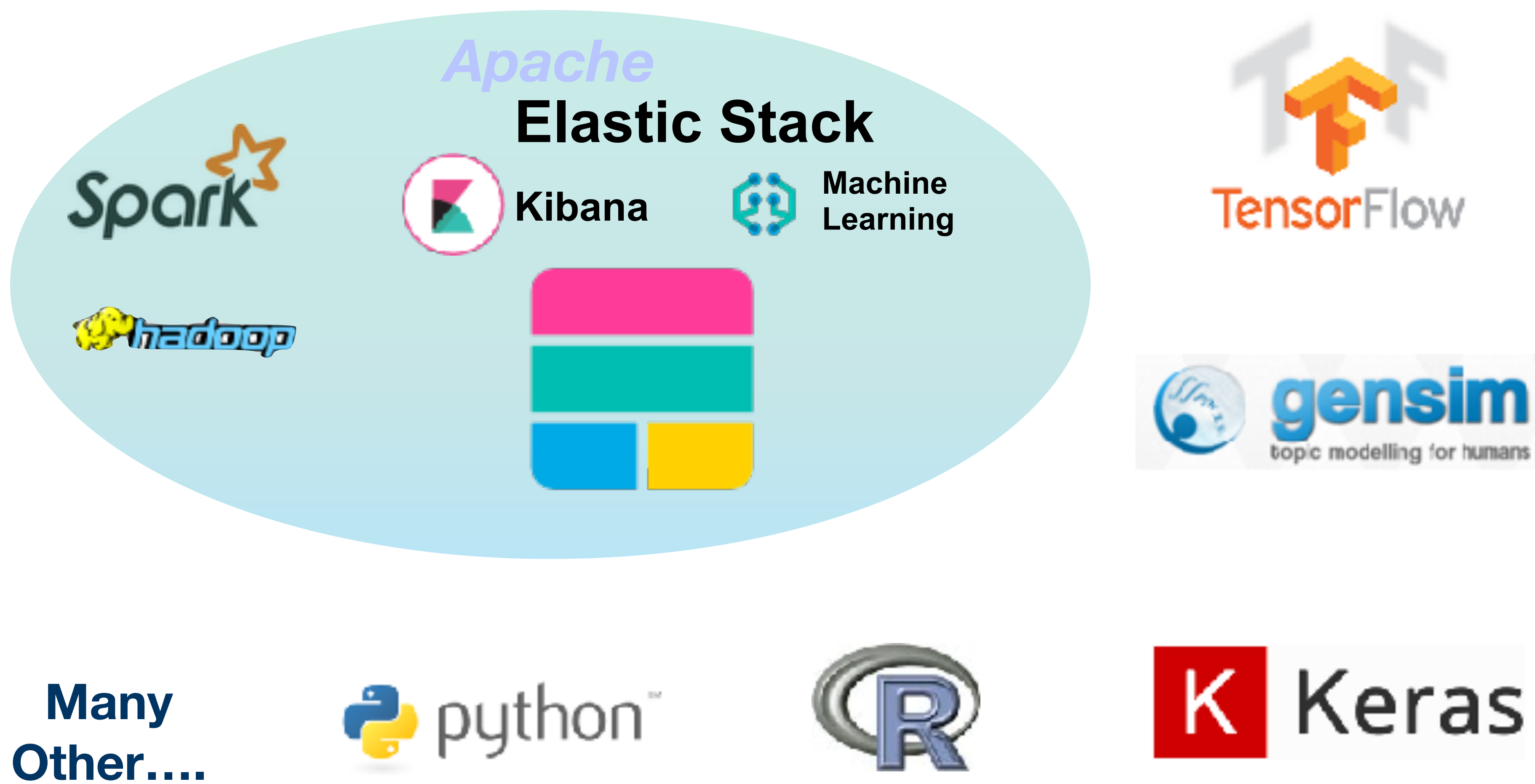
3 Spark Collaborative Filtering

Chapter 1

Apache Eco-system & Review

elastic

Elastic Stack Data Science Eco-System



Elasticsearch Hadoop Connector

ES-Hadoop

Elasticsearch for Hadoop

Two way connector	Index Hadoop data in Elasticsearch	Enable real-time search capabilities
Visualize data in Kibana	Read/Write directly to/ from Kafka	Support for Spark, Storm, MapReduce, and more

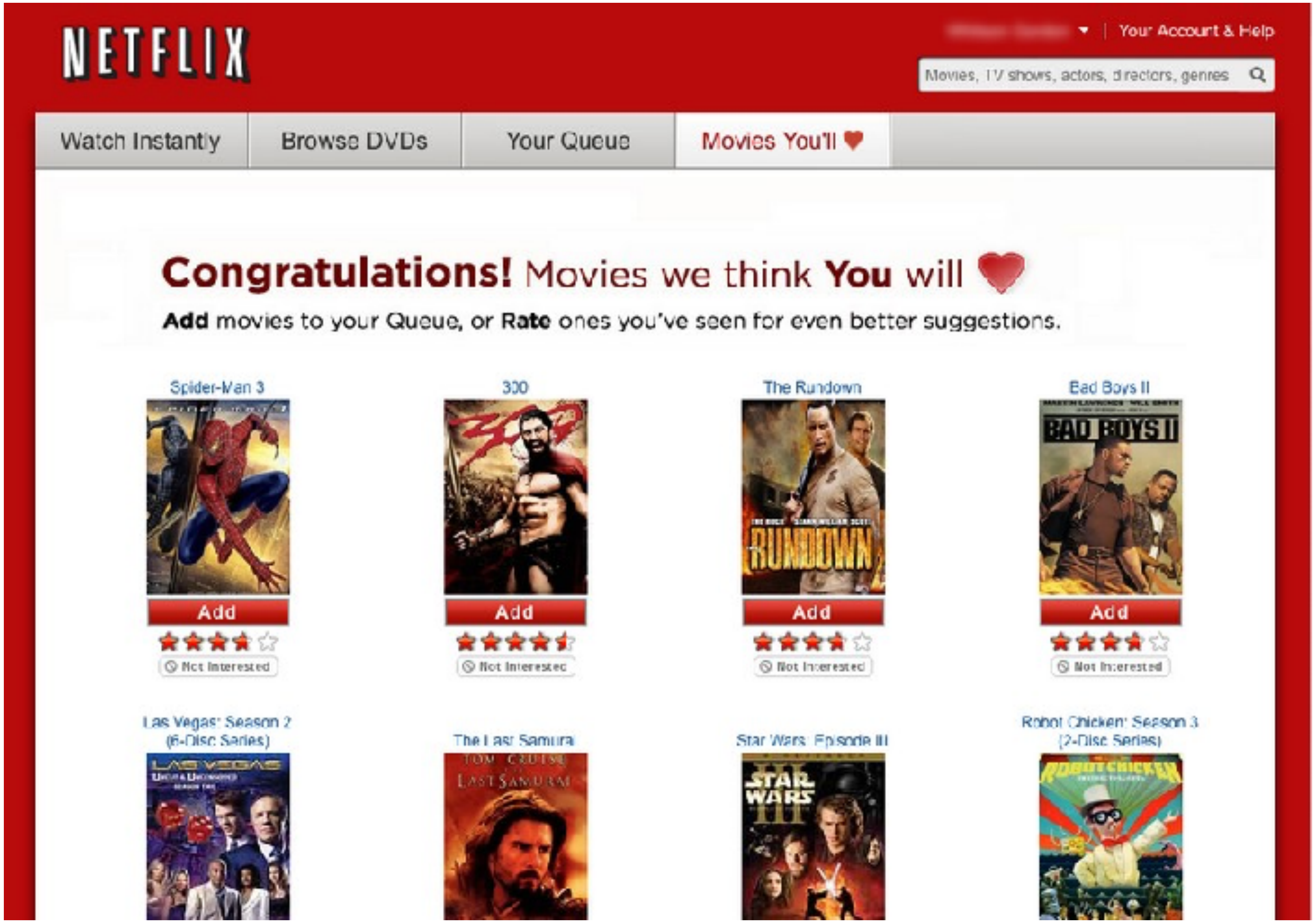
Machine Learning Categories

- Unsupervised Learning
  - Clustering
  - Outlier/Anomaly Detection
  - Affinity/Market Basket Analysis
  - **Recommendation Systems**

- Supervised Learning
  - Classification
  - Regression
  - **Recommendation Systems**

Depending on the source, recommendation engines are categorized under supervised, unsupervised, or neither

Example - Netflix Movies



Movielens Dataset

- University of Minnesota Grouplens research project
  - Several datasets of varying size
  - ml-20m dataset (largest) used here
- Six csv files: genome-scores.csv, genome-tags.csv, tags.csv, **links.csv**, **movies.csv**, **ratings.csv**
- Movies viewed from January 09, 1995 to March 31, 2015
- 5-star ratings
- Contains:
  - 20,000,263 ratings
  - 27,278 movies
  - 138,493 users

Citation/Attribution

```
F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and
Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19
(December 2015), 19 pages. DOI=<http://dx.doi.org/10.1145/2827872>
```

ratings.csv

```
userId,movieId,rating,timestamp
1,2,3.5,1112486027
1,29,3.5,1112484676
1,32,3.5,1112484819
1,47,3.5,1112484727
1,50,3.5,1112484580
1,112,3.5,1094785740
1,151,4.0,1094785734
1,223,4.0,1112485573
```

movies.csv

```
movieId,title,genres
1,Toy Story (1995),Adventure|Animation|Children|Comedy|Fantasy
2,Jumanji (1995),Adventure|Children|Fantasy
3,Grumpier Old Men (1995),Comedy|Romance
4,Waiting to Exhale (1995),Comedy|Drama|Romance
5,Father of the Bride Part II (1995),Comedy
6,Heat (1995),Action|Crime|Thriller
7,Sabrina (1995),Comedy|Romance
8,Tom and Huck (1995),Adventure|Children
```

Quick review

- Significant Terms vs. Terms Aggregations
- Fielddata and Sampler Aggregations

Significant Terms Agg

```
GET nutrition/_search
{
  "size":0,
  "aggs" : {
    "my_total_fat_histogram":{
      "histogram": {
        "field": "details.total_fat",
        "interval": 5,
        "min_doc_count": 5
      },
      "aggregations":{
        "my_top_words" : {
          "significant_terms" : {
            "field" : "ingredients.keyword",
            "size":5
          }
        }
      }
    }
  }
}
```

Let's look for the "uncommonly common" ingredients in relation to total fat.

The Results of significant_terms:

```
"key": 15,
"doc_count": 31,
"my_top_words": {
  "doc_count": 31,
  "bg_count": 499,
  "buckets": [
    {
      "key": "brazil nuts",
      "doc_count": 3,
      "score": 1.460978147762747,
      "bg_count": 3
    },
    {
      "key": "palm oil",
      "doc_count": 3,
      "score": 1.460978147762747,
      "bg_count": 3
    },
    {
      "key": "almonds",
      "doc_count": 3,
      "score": 1.460978147762747,
      "bg_count": 3
    },
    {
      "key": "apples",
      "doc_count": 3,
      "score": 1.0715400624349638,
      "bg_count": 4
    },
    {
      "key": "flour",
      "doc_count": 3,
      "score": 0.8378772112382935,
      "bg_count": 5
    }
  ]
```

The same 31 products with **total_fat** between 15 and 20

The top 5 significant ingredients are "**brazil nuts**", "**palm oil**", "**almonds**", "**apples**" and "**flour**"

Fielddata and Sampler Agg

Enabling Fielddata

- To run terms or significant_terms agg on **text** data types
  - enable **fielddata** for that field
  - In our example, we are going to perform both aggregations on the **ingredients** field of our **nutrition** index:

```
PUT nutrition/_mapping/_doc
{
  "properties" : {
    "ingredients" : {
      "type" : "text",
      "fielddata" : true
    }
  }
}
```

Update the existing mapping

Enable **fielddata** for **ingredients**

Alternative: Run aggs against **keyword** data types

Summary

- Spark has machine learning libraries to run recommendations and other techniques
- Spark/Hadoop and Elastic Stack can couple with the ES-Hadoop connector
- Terms aggregations find the most popular categories
- Significant terms aggregation finds more interesting categories
- Fielddata places our text data type field in memory (dynamically) so we can aggregate text

Quiz

1. **True or False**: Elastic Stack is self consistent and Elasticsearch only works with other Elastic systems (e.g. Kibana, Logstash, Beats, Machine Learning)?
2. What size dataset will we be using from Movielens?
3. **True or False**: Everyone agrees that recommender systems are considered supervised techniques, because you have to train the model?
4. What tool is the best way to transfer data between Elasticsearch and Spark?
5. What aggregation is built into Graph by default to find more relevant results?

Lab 1

Start up environment

How to Launch Strigo

- ✤ Go to '*training.elastic.co*' (you need your Elastic login and password - find in your emails)
- ✤ Login, find this class, scroll down and find a link to Strigo (and a pdf and a zip)
- ✤ In Strigo, click on the "cloud" icon on the left

  A command line (or "Loading") will appear. In the upper left is a small box with an even smaller gear-shaped icon (sometimes the gear is shaped like a little white box).
- ✤ Click on the gear -> pull down menu -> "Connect from Local"

  A pop-up box appears with an Amazon address labeled as "IP."
- ✤ Copy that and paste it into a new browser tab.

  A webpage called "Welcome to Recommendation Systems" appears with several links including one for Lab Instructions.

Chapter 2

Elastic Graph

elastic

Topics covered:

- Graph Overview
- Co-occurence
- Relevance
- API

Graph Overview

elastic

Graph

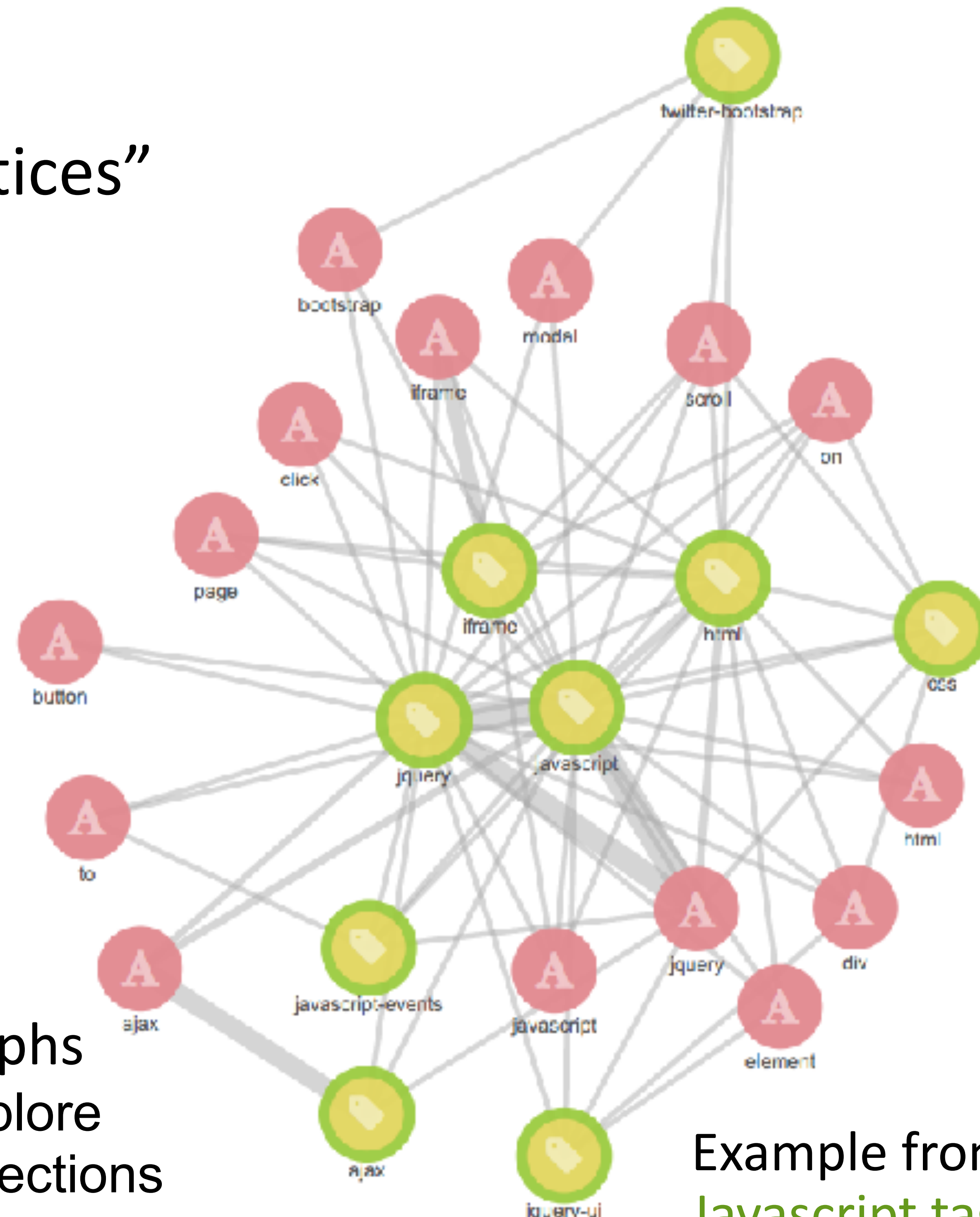
Graphs are a set of "vertices" and the "connections" between them

Combines graph algorithms and search

- Explore data relevancy
- Recommendation engine

Kibana UI to interact with graphs

- Select relationships to explore
- Find new or existing connections in the data
- Visualize results



Example from stackoverflow Javascript tag and related tags and connected title words

Why Graph?

- Uses existing Elasticsearch indexes
  - No need to reindex
  - No need to change data models
- Simple architecture
  - No need to deploy a special graph data system
  - Scales with Elasticsearch cluster
- Combine the power of search, relevancy and graph databases
  - Find significant relationships between data

Graphs

- An indexed value is a potential vertex
  - single value fields such as an email address which can link to other fields
  - array fields such as "liked movies" which can link to itself and other fields
- A relationship forms a connection
  - Not persisted in Elasticsearch
  - Uses aggregation framework to search and connect the vertices
- Graph Traversal
  - Graph traversal algorithms prioritize finding meaningful connections in the data…

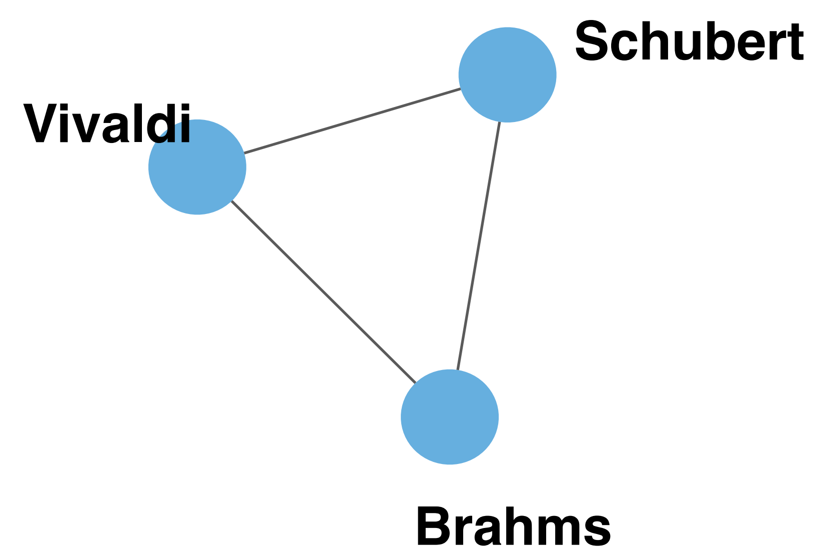
Graph Co-occurence

Example: Inferring relationships from co-occurrence

Music Recommendation

```
{  "user_id" :  "1",
   "liked"   :  { vivaldi, brahms, schubert}
}
```

Users that liked Vivaldi also liked ???



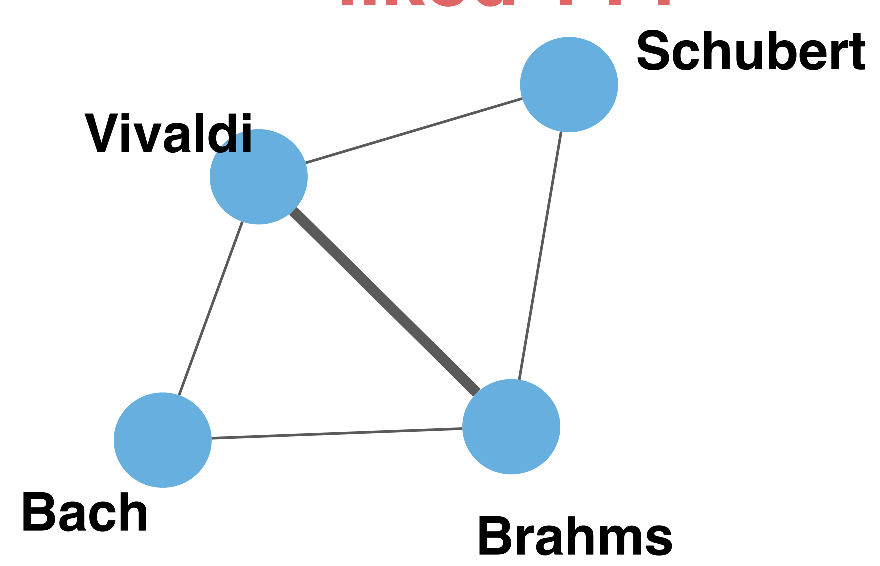
Example: Inferring relationships from co-occurrence

Music Recommendation

```
{   "user_id" :  "1",
    "liked"   :  { vivaldi, brahms, schubert}
}
```

```
{   "user_id" :  "2",
    "liked"   :  { vivaldi, brahms, bach }
}
```

Users that liked Vivaldi also liked ???



Example: Inferring relationships from co-occurrence

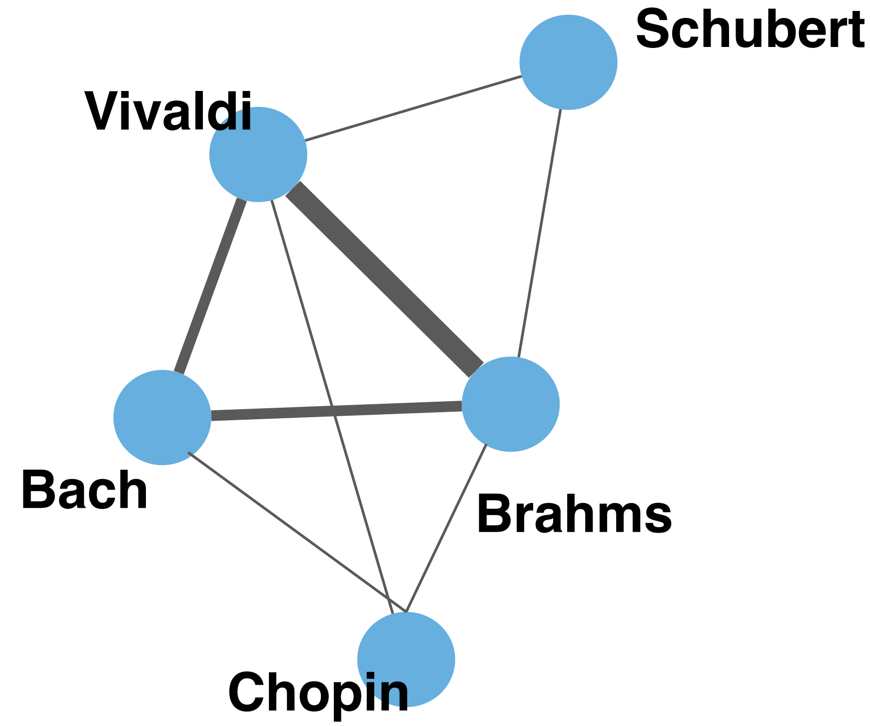
Music Recommendation

```
{   "user_id" :  "1",
    "liked"   :  { vivaldi, brahms, schubert}
}
```

```
{   "user_id" :  "2",
    "liked"   :  { vivaldi, brahms, bach }
}
```

```
{  "user_id" : "3",
   "liked"   : { vivaldi, brahms, bach, chopin}
}
```

Users that liked Vivaldi also liked ???



Graph and Relevance

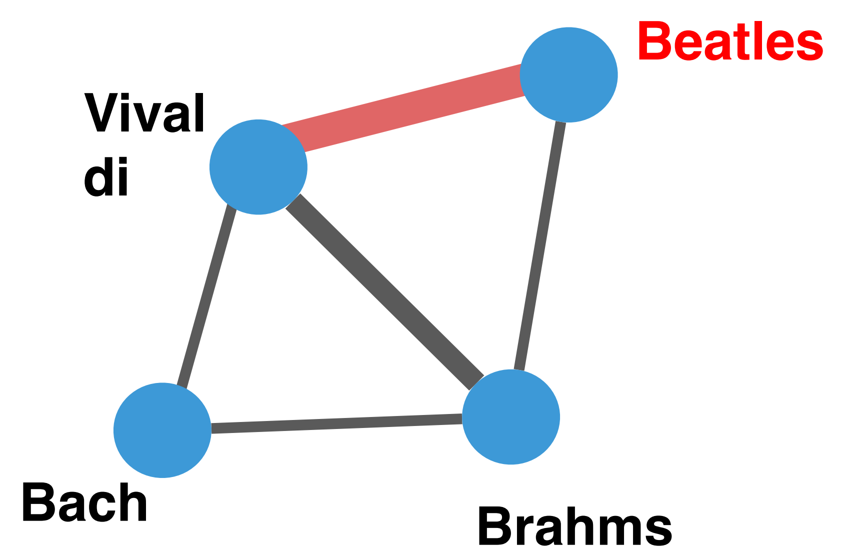
elastic

Example: Inferring relationships from co-occurrence

Music Recommendation

```
{  "user_id" :  "200000",
   "liked"   :  { vivaldi, brahms, beatles}
}
```

Users that liked Vivaldi also liked ???

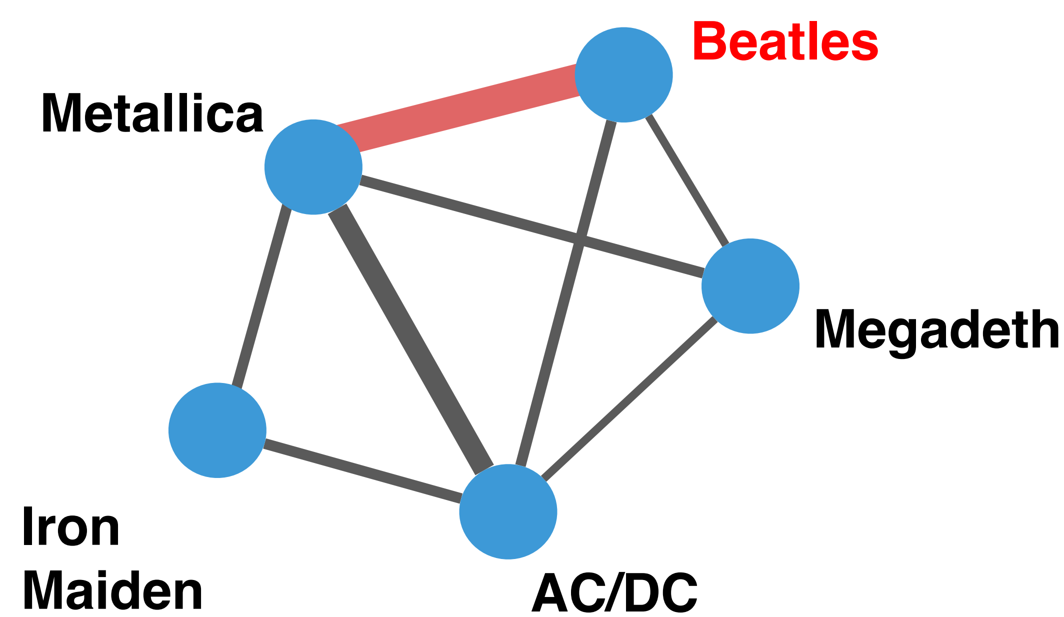


Example: Inferring relationships from co-occurence

Music Recommendation

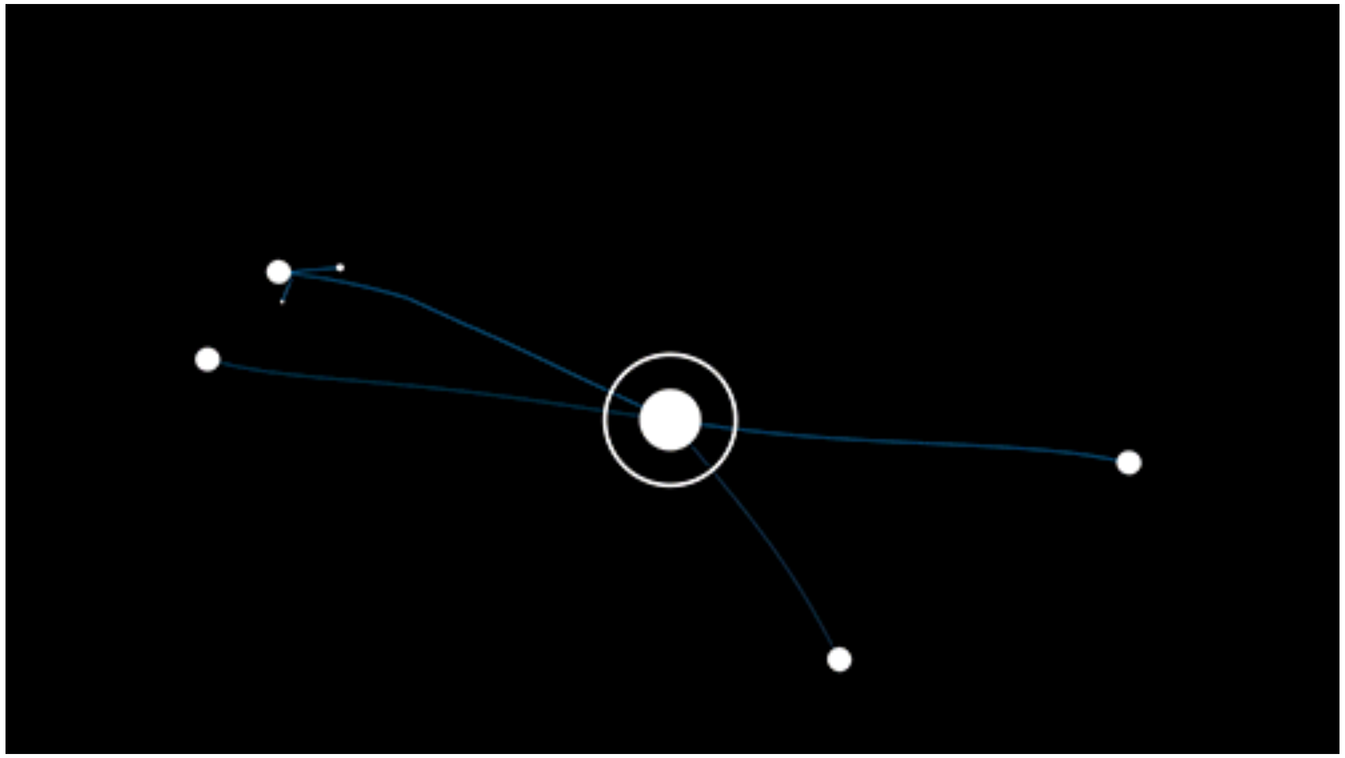
```
{  "user_id" :  "200005",
   "liked"   :  { metallica, AC/DC, beatles}
}
```

Users that liked Metallica also liked ???



What does meaningful mean?

- Super nodes (Super Connectors)
  - Traditional graphs get distorted by "super nodes"
  - They frequently include these heavily connected vertices during exploration which can distort finding relevant connections
  - When storing connections instead of computing them on-the-fly this becomes a major issue



- Wisdom of crowds
  - Use sampling and diversity settings to choose which signals we want to summarize
  - Provide a more personalized form of recommendation…

Personalized recommendations

Many approaches store edges so they can retrieve an answer for questions like:

"People who searched for X tend to click on product Y."

- Provides only a single interpretation of event

- Elastic Graph can find the answer to these questions by searching:
  - "People who searched for X (and ideally were females in London with an age range of 25-40 with interests in product Z) tend to click on product Y"
  - This is computed using aggregations and the search on top of proven graph algorithms

Graph API

Graph API - programmatic results

- REST interface accepts user graph-exploration criteria as JSON:

```
POST clicklogs/_graph/explore
{
    "query": {
        "match": {
            "query.raw": "midi"
        }
    },
    "vertices": [
        {
            "field": "product"
        }
    ],
    "connections": {
        "vertices": [
            {
                "field": "query.raw"
            }
        ]
    }
}
```

- Find product codes that are significantly associated with searches for "midi" and further, show other queries that led people to these products
- Internally a number of searches with aggregations are then made to build the graph

Basic use (1 of 2)

- Potential JSON response:

```
{
  "vertices": [
      {
         "field": "query.raw",
         "term": "midi cable",
         "weight": 0.08745858139552132,
         "depth": 1
      },
      {
         "field": "product",
         "term": "8567446",
         "weight": 0.13247784285434397,
         "depth": 0
      },
      {
         "field": "product",
         "term": "1112375",
         "weight": 0.018600718471158982,
         "depth": 0
      },
      {.
         "field": "query.raw",
         "term": "midi keyboard",
         "weight": 0.04802242866755111,
         "depth": 1
      }
   ],
   "connections": [
      {
        …
      }
   ]
}
```

Basic use (2 of 2)

- Potential JSON response:

```
{
  "vertices": [
      …
   ],
   "connections": [
      {
         "source": 0,
         "target": 1,
         "weight": 0.04802242866755111,
         "doc_count": 13
      },
      {
         "source": 2,
         "target": 3,
         "weight": 0.08120623870976627,
         "doc_count": 23
      }
   ]
}
```

Query Controls

- Configure controls to the query to tune the graph query results and performance
- Visual and Programmatic

Control	Description
use_significance	Connected terms are only those that are significantly associated with our query. Default: true
sample_size	Each connection considers a sample of the best-matching documents on each shard. Using samples has the dual benefit of keeping exploration focused on meaningfully connected terms and improving the speed of execution. Default: 100 document
timeout	Time in milliseconds exploration will be halted and results gathered so far are returned
sample_diversity	To avoid the top-matching documents sample being dominated by a single source of results sometimes it is useful to request diversity in the sample

Programmatic controls

- You can control each vertices settings too:

Control	Description
size	Number of vertex terms returned for each field, defaults to 5
min_doc_count	Acts as a certainty threshold - just how many documents have to contain a pair of terms before we consider this to be a useful connection? (default is 3)
shard_min_doc_count	Advanced setting - just how many documents on a shard have to contain a pair of terms before we return this for global consideration? (default is 2)

Summary

- Graph models data as a web of connections between vertexes
- Use graph to find similarities via co-occurrences between users
- Graph has beautiful, feature rich visual interactive analytics
- Graph has a powerful API for programmatic analysis
- Graph, as a recommender engine, uses the power of query, aggregation, and graph math to create compelling recommendations

Quiz

1. **True or False**: By default Graph recommends the most popular items?
2. Graph loads indexes from Elasticsearch into a special Graph data store called what?
3. **True or False**: Super nodes are a problem for most graphing tools, but not for Elastic Graph.
4. The visual Graph analyzer returns how many results at a time?
5. **True or False**: The API allows you to change the number of items returned?
6. The default min_doc_count is how many?
7. Is it better to learn Graph by reading these slides or doing the lab?

Lab 2

Elastic Graph

Chapter 3

Spark Collaborative Filtering

elastic

Topics covered:

- Spark Hadoop environment
- Spark architecture
- RDD and Dataframe
- Collaborative Filtering
- ES-Hadoop connector

Elastic Stack Data Science Eco-System

Apache

Elastic Stack

Spark

hadoop

Kibana

Machine Learning

TensorFlow

gensim
topic modelling for humans

Many Other….

python

R

Keras

Hadoop and Spark

Yet Another Resource Negotiator

Resourcemanager

Nodemanager1

Nodemanager2

Hadoop Distributed File System

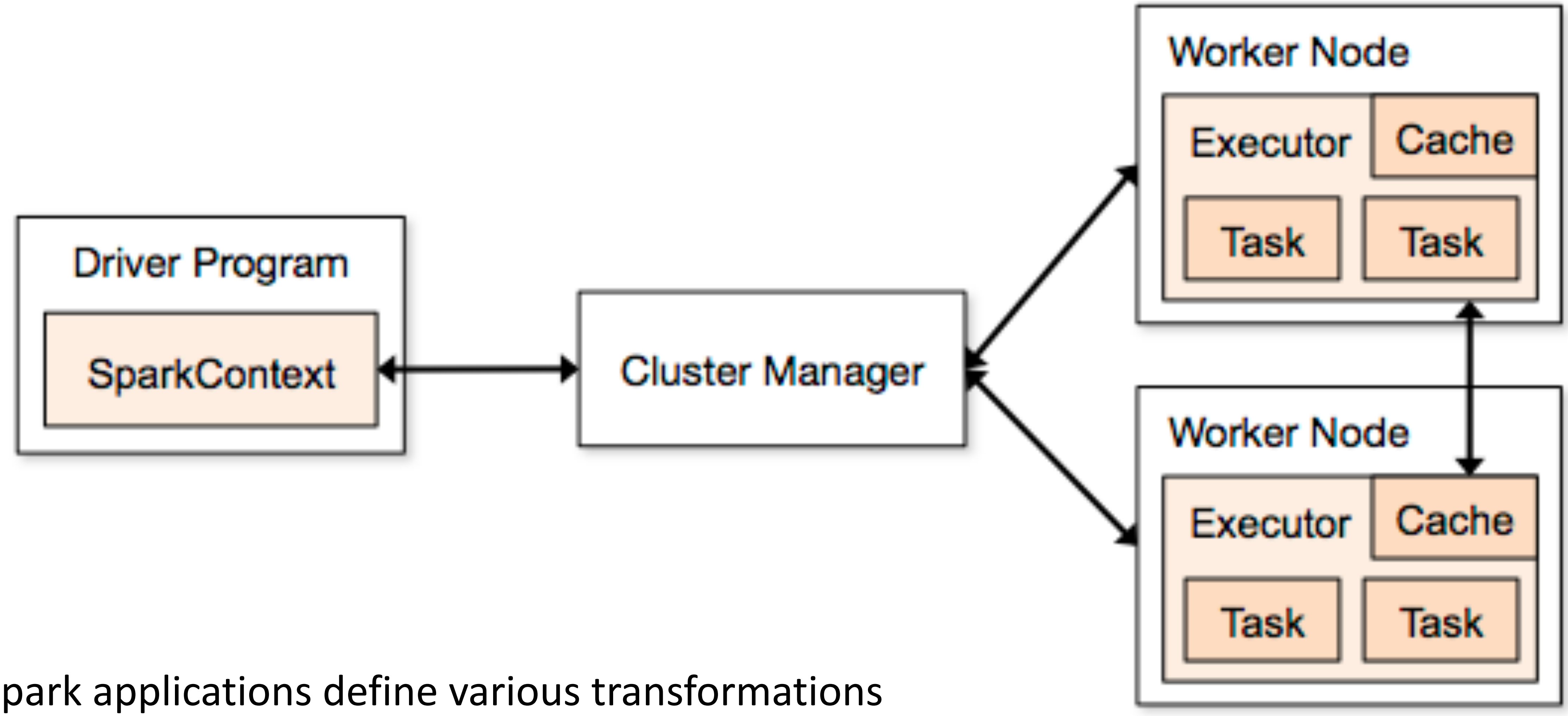
Namenode

Datanode1

Datanode2

HDFS = storage, YARN = cluster OS running apps including Spark, HBase, Storm, Hive, Pig, etc.

Understanding Spark Applications



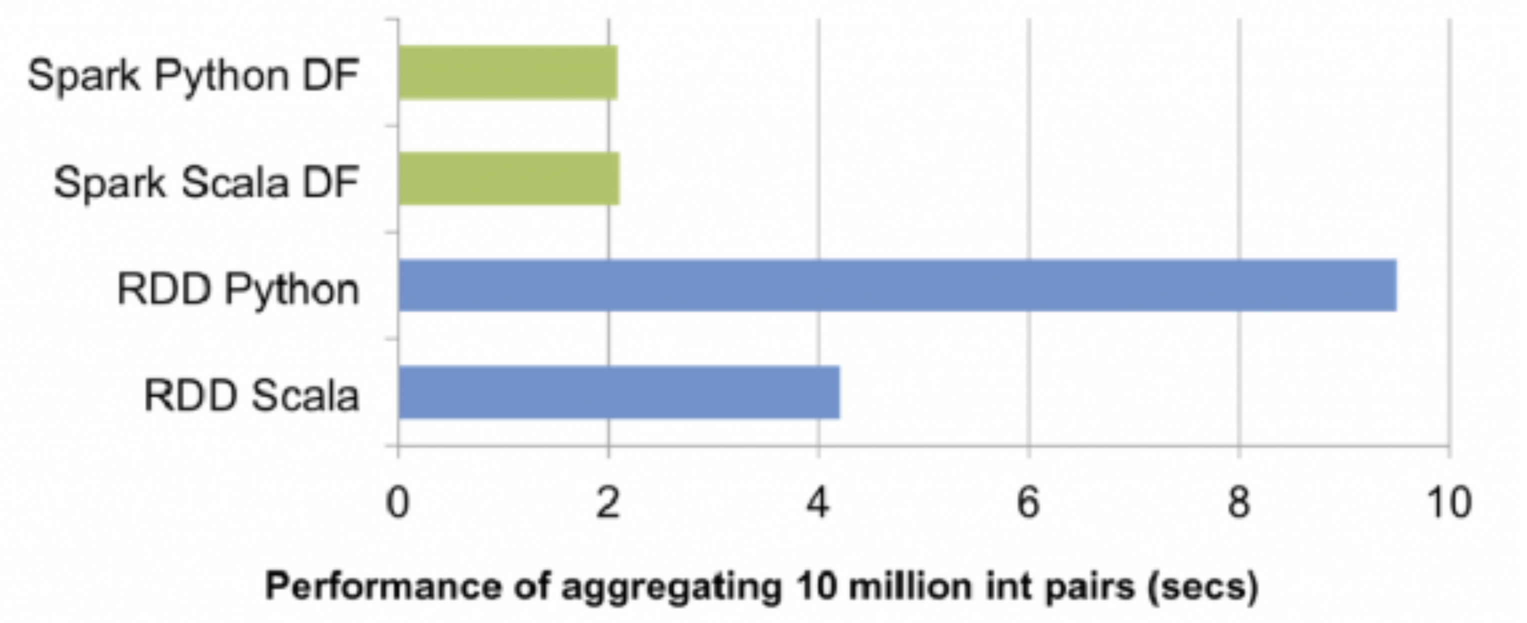
Spark applications define various transformations and actions

but the actual steps of the script are not executed until an output is requested (lazy)

Source code is written in Scala

Spark RDD and Dataframe

- ***RDD*** (***resilient distributed dataset***) is a read-only, fault-tolerant, lineage-driven collection of elements that can be operated on in parallel



- A dataframe is inspired by R (DataFrame) and Python (Pandas) but stored using RDDs underneath
    - A dataframe is organized into named rows and columns - i.e. a table!
    - Dataframes can run sql and execute joins (for ETL of our dataset)
    - API is available in Scala, Java, Python, and R

    https://spark.apache.org/docs/latest/api.html

Preference/Utility Matrix

	Harry potter	X-Men	Hobbit	Argo	Pirates
User1	5	2	4	?	?
User2	?	?	5	2	?
User3	1	2	?	?	3
.					
.					
.					

Harry potter	X-Men	Hobbit	Argo	Pirates
5	2	4	1	3
4	1	5	2	3
1	2	4	1	3

Rows = Users   Columns = Products   Values = Preference

Goal - Predict/Calculate Values for the Question Marks

Collaborative filtering

Harry potter	X-Men	Hobbit	Argo	Pirates
5	2	4	?	?
?	?	5	2	?
1	2	?	?	3

Harry potter	X-Men	Hobbit	Argo	Pirates
5	2	4	1	3
4	1	5	2	3
1	2	4	1	3

- Can be calculated in a couple of ways
- In SparkML there is only ALS - Alternating Least Squares
  - uses matrix decomposition and linear algebra to calculate the unknowns

Elasticsearch Hadoop Connector

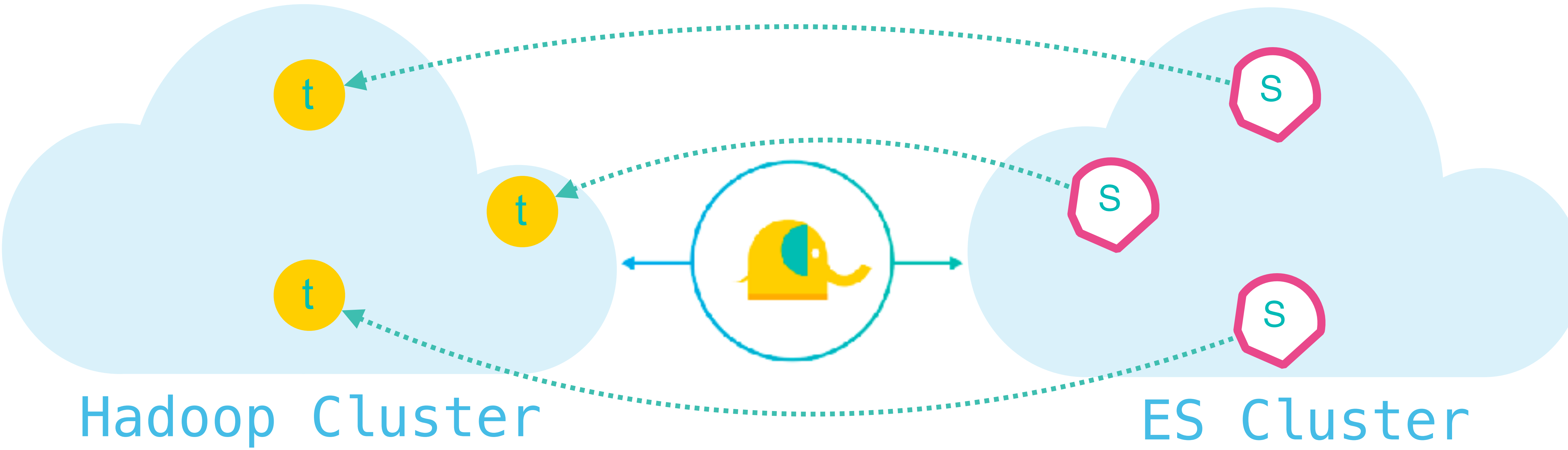
ES-Hadoop

Elasticsearch for Hadoop

Two way connector	Index Hadoop data in Elasticsearch	Enable real-time search capabilities
Visualize data in Kibana	Read/Write directly to/ from Kafka	Support for Spark, Storm, MapReduce, and more

elastic

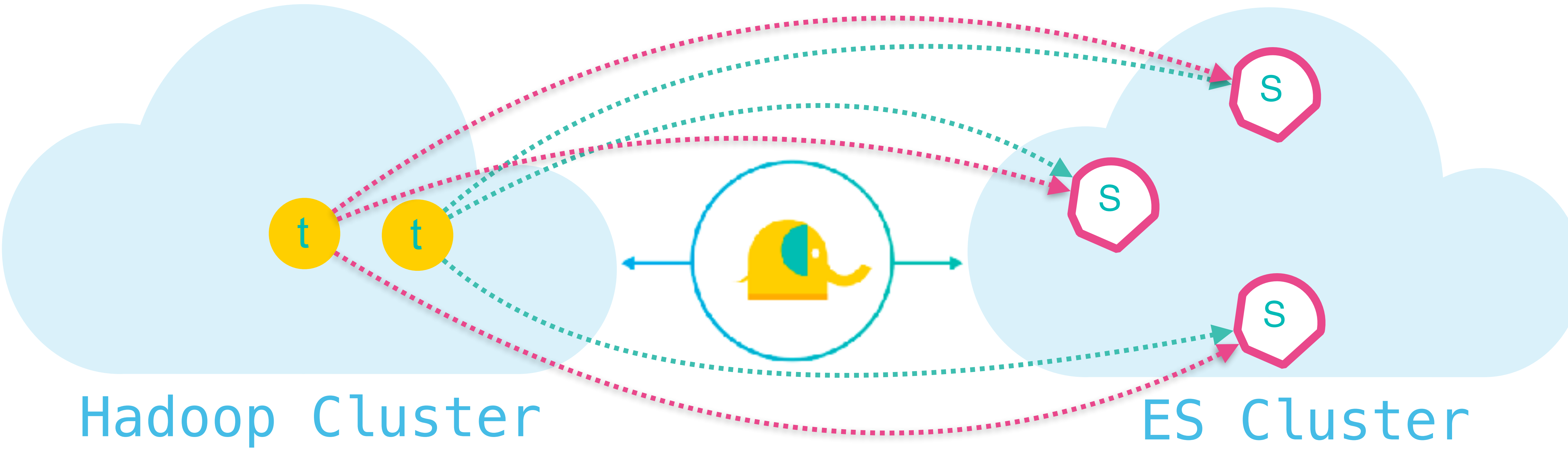
Reading from Elasticsearch



Reading from Elasticsearch

- ES-Hadoop will:
  - detect the number of shards (primary & replica) in query index
  - create one task (Hadoop split / spark partition) per shard

Writing to Elasticsearch



Writing to Elasticsearch

- ES-Hadoop will :
  - detect number of primary shards for write index
  - distribute writes between them
  - more splits = more parallel writes

Summary

- Spark distributes work across machines like Elasticsearch. Jobs are run on executors and a driver
- Spark dataframe represents the data in tables that can be joined
- Spark has machine learning libraries to run a variety of data science functions and tasks. For recommendation its algorithm is called Collaborative Filtering.
- The ES-Hadoop connector works in

Quiz

1. What process manages a Spark job?
2. What Spark processes run the distributed work on nodes?
3. What is faster: an RDD or a dataframe? why?
4. What data structure is need for collaborative filtering (on Spark)?
5. **True or False**: The ES-Hadoop connector works in both directions?
6. **True or False**: When sending data from Spark to Elasticsearch, the ES-Hadoop connector send all the data to the master node, then it gets pushed to shards?

Lab 3

Spark CF

Conclusions

elastic

Elastic Certification

- Elastic is launching its first ever ***professional certification:***

Elasticsearch Engineer I

Elasticsearch Engineer II

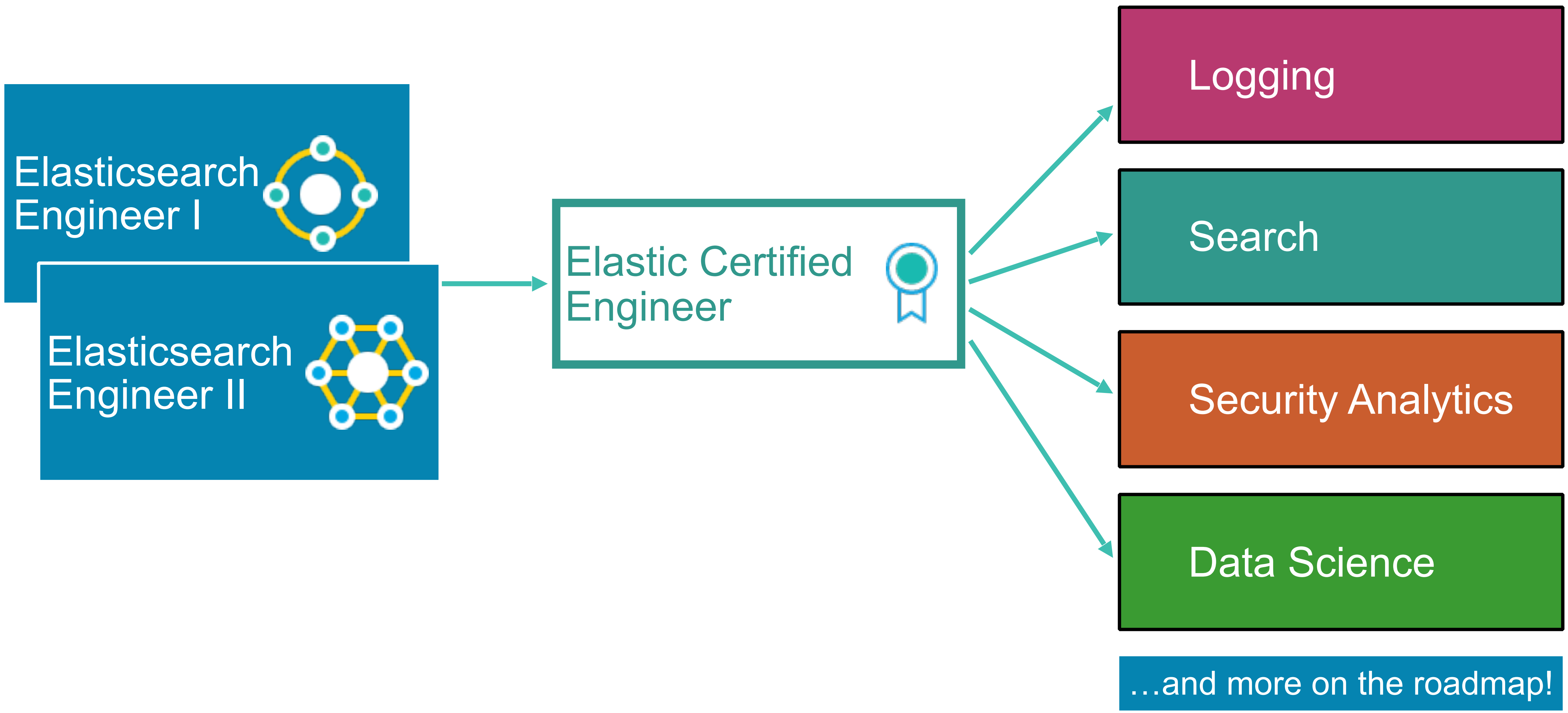
Elastic Certified Engineer

Two courses

One exam

Specialization Training

- We are also developing new ***specialization trainings*** focused on specific Elastic Stack use cases
  - including a plan to add exams for ***Elastic Certified Specialists***



Thank you!

Please complete the online survey

elastic

Quiz Answers

Chapter 1 Quiz Answers

1. True
2.

Chapter 2 Quiz Answers

1. **elasticsearch.yml**, **jvm.option**s, and **log4j2.properties**
2.

Chapter 3 Quiz Answers

1. Multiple **match** terms use “**and**” or “**or**”, while multiple terms in **mat**
2.

Elastic Specialization - Data Science

Course: Recommendation System

Version x.x.1

Course: Recommender Systems

Version 6.3.0-pilot