IR in Practice

(a.k.a. Elastic 4 IR)

https://github.com/ielab/afirm2019

Dr. Guido Zuccon

University of Queensland
g.zuccon@uq.edu.au

THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

ielab

www.ielab.io

Plan of the Session

- The architecture of a typical IR system — and that of Elasticsearch
- Intro to Elasticsearch: functionalities, installation and basic interaction (Activity 0)
- IR in Practice: Hands on with Elastic Search (most in Python/some in Java). Activities:

    1. Basic Indexing and Search in Elasticsearch
    2. Boolean retrieval
    3. Produce a TREC Run
    4. Access a Term Vector
    5. Implement a New Retrieval Model
    6. Document Priors and Boosting (e.g. Link Analysis)
    7. Text Snippetting for Search Results

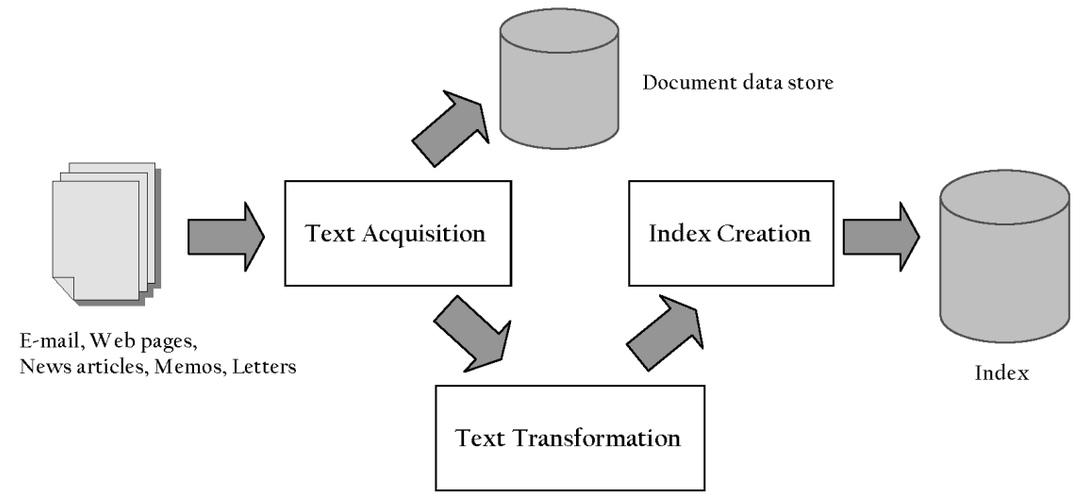
Practical activities

- Material is in GitHub: https://github.com/ielab/afirm2019
    - Including these slides, links to data, practical instructions, code
- The folder `hands-on` contains a subfolder for each of the activities
- Usually an activity has a README.md file with explanation, instructions and exercises. Most activities come with code
- Code is often in the form of a Python notebook (jupyter). One activity relies on Java code.
- When necessary, links to download data are provided

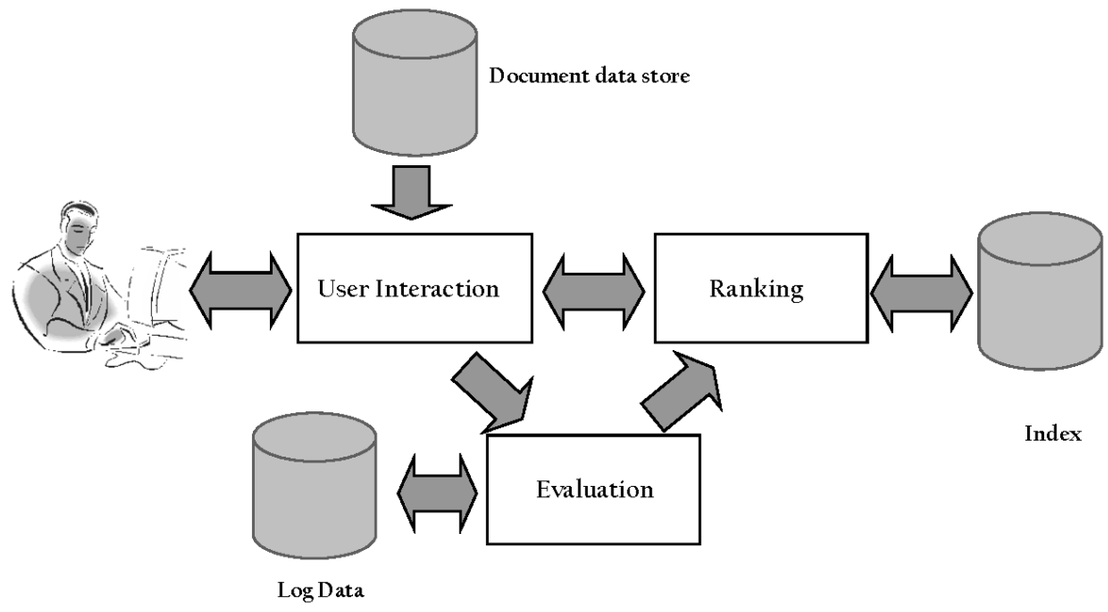
Search Engine Architecture

Basic building blocks:

- The indexing process

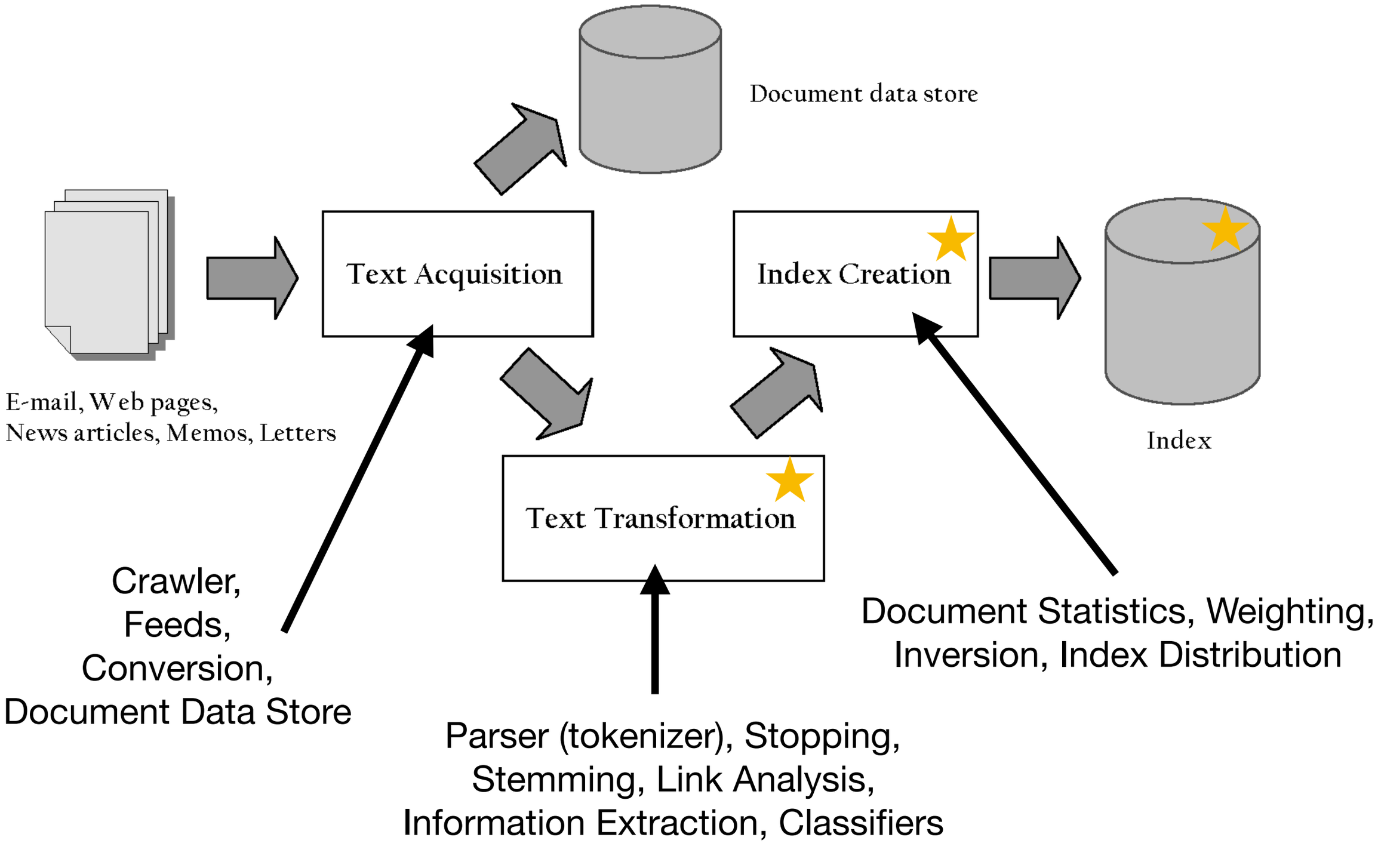


- The querying process

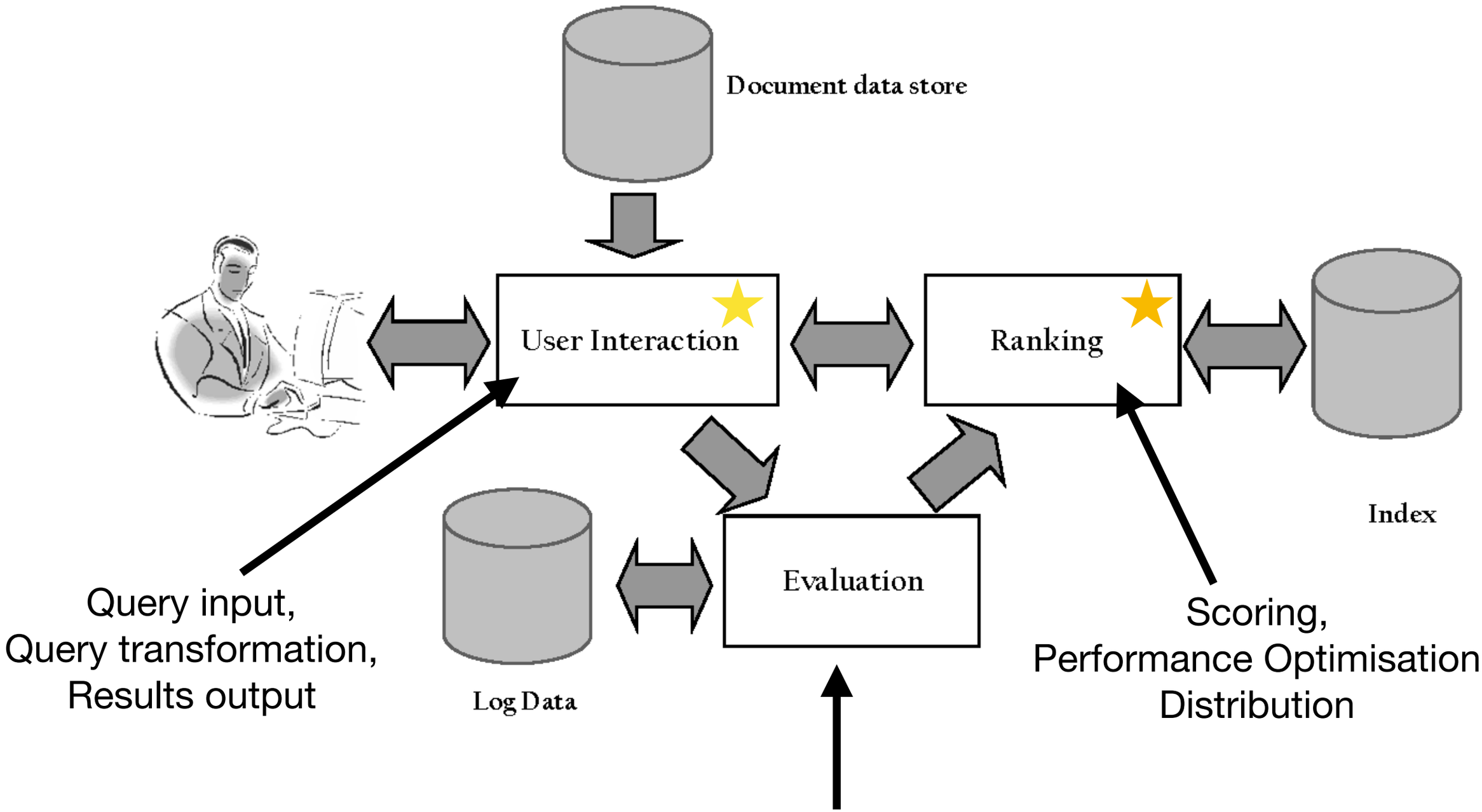


* Figures from Croft, Metzler, Strohman, “Search Engines: Information Retrieval in Practice” Free download at:

The indexing process

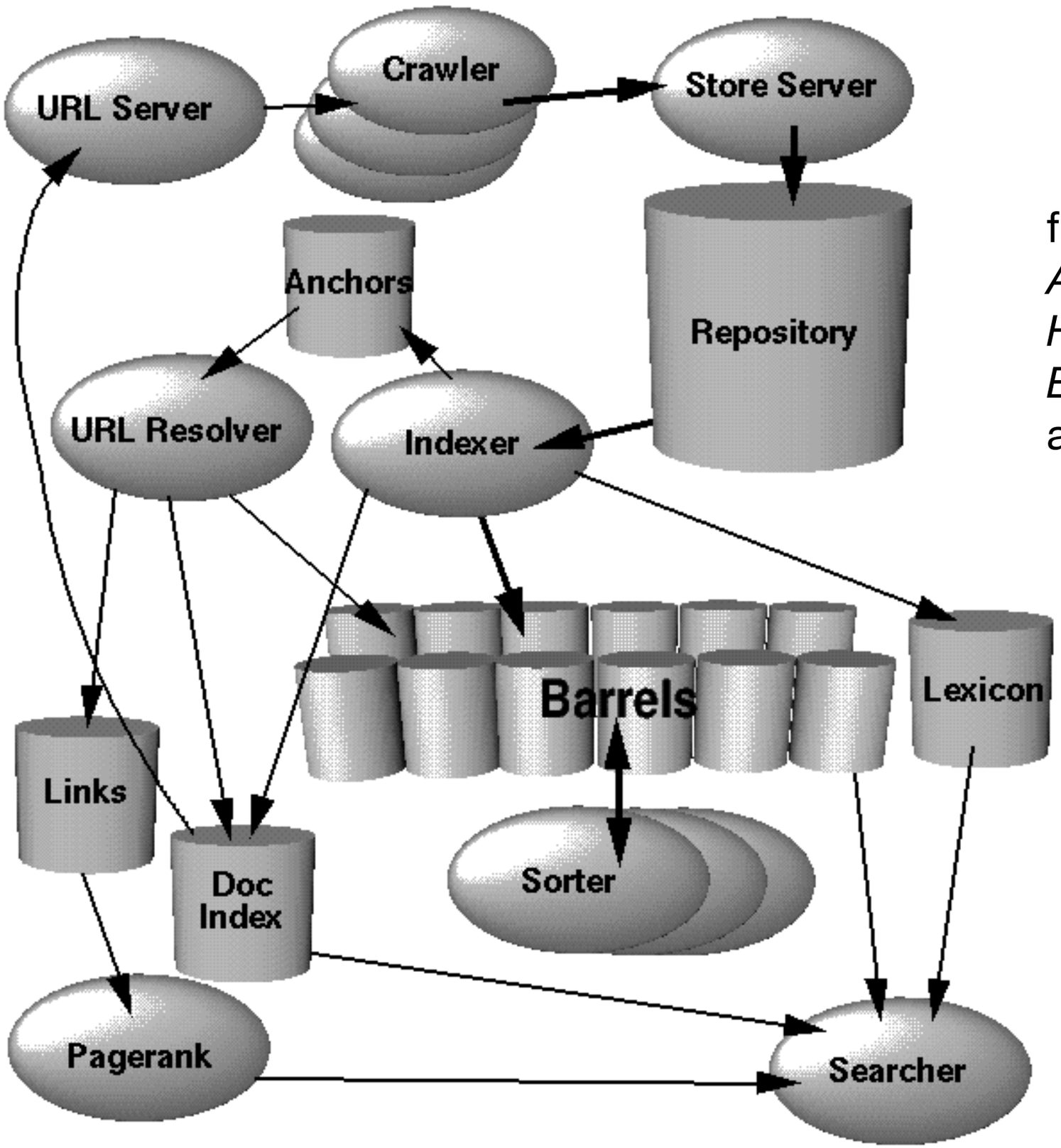


The querying process



The Architecture of Google

(in the early days)



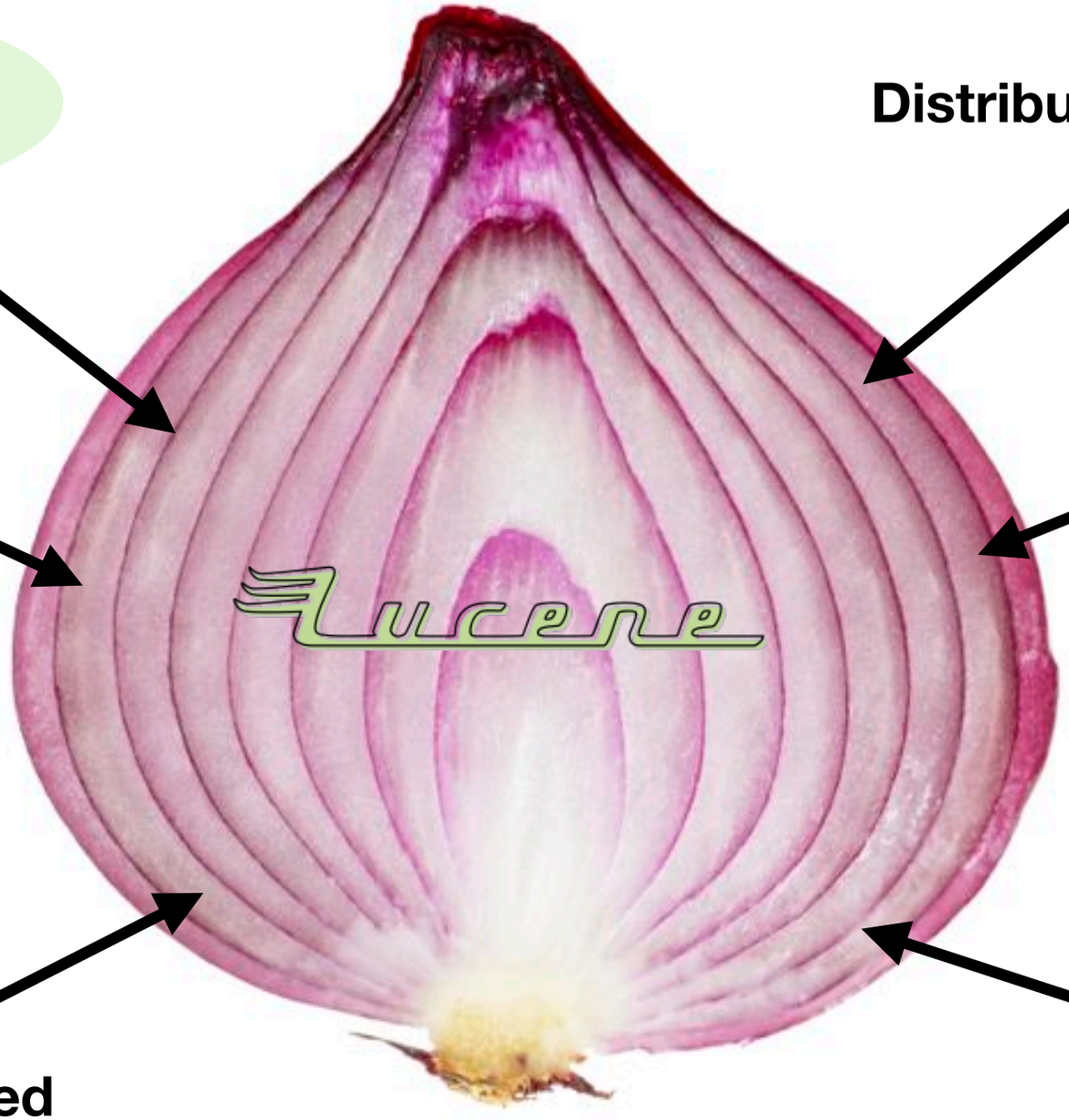
from: S Brin, L Page, *The Anatomy of a Large-Scale Hypertextual Web Search Engine*, Computer networks and ISDN systems, 1998

Many IR Toolkits out there

(a non-exhaustive list)

- Apache Lucene / Sorl / Elasticsearch / Anserini: http://lucene.apache.org/, http://lucene.apache.org/solr/, https://www.elastic.co/, http://anserini.io/
- Terrier: http://terrier.org/
- Lemur / Indri / Galago: https://www.lemurproject.org/ (and derivatives/wrappers e.g. Pyindri)
- ATIRE & JASS: http://atire.org, https://codedocs.xyz/andrewtrotman/JASSv2/
- Some are less popular/maintained:
    - MG4J: http://mg4j.di.unimi.it/
    - Zettair: http://www.seg.rmit.edu.au/zettair/
    - Etc

What is Elasticsearch?



RESTful API

Distributed indexing/querying

Large, powerful query syntax

Transaction logs

Multi-tenancy enabled

Horizontal scalable

A bit of vocabulary

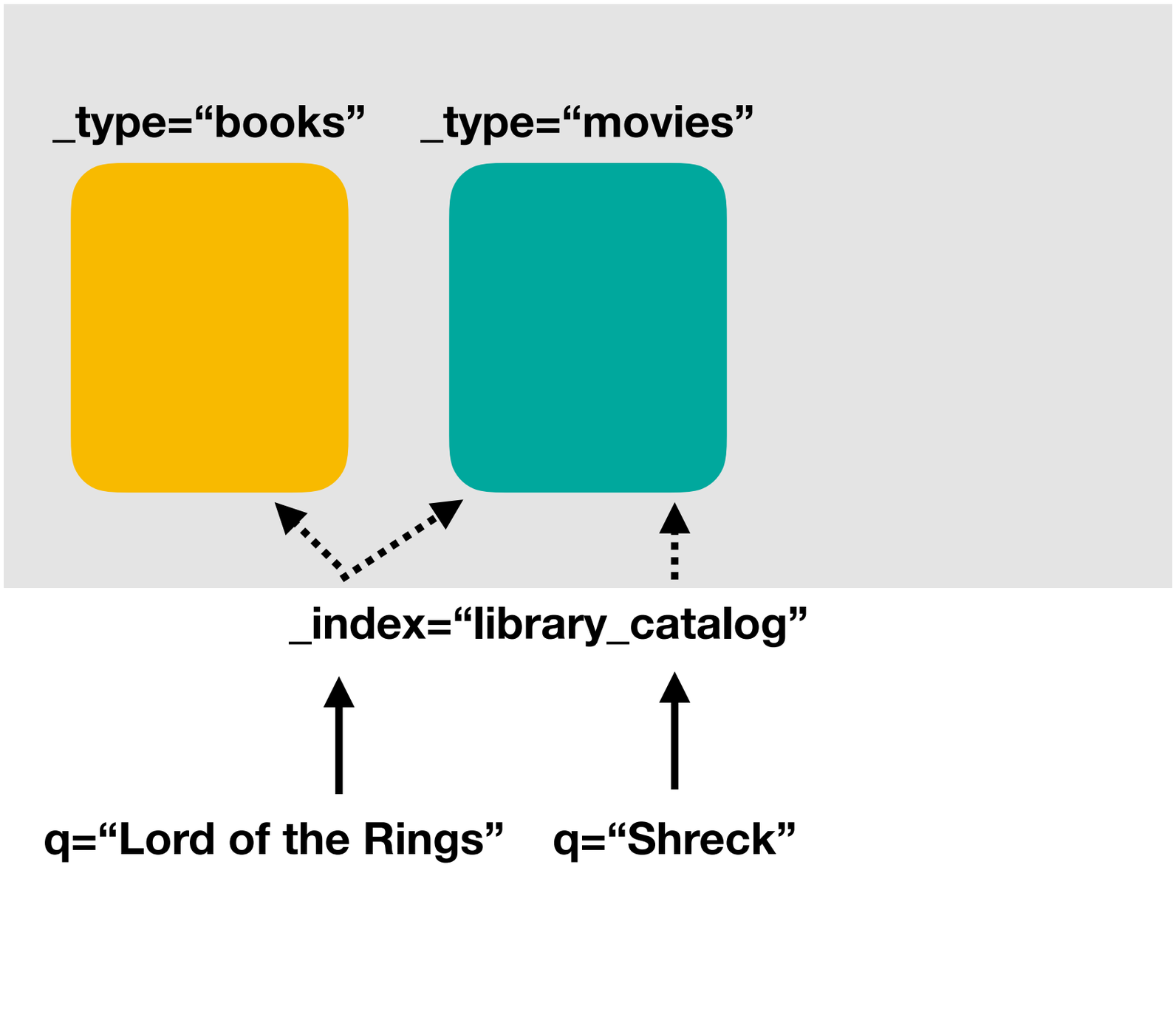
- **Node:** a JVM process executing Elasticsearch. Typically one node per machine

- **Index**: a Lucene Index, which contains documents

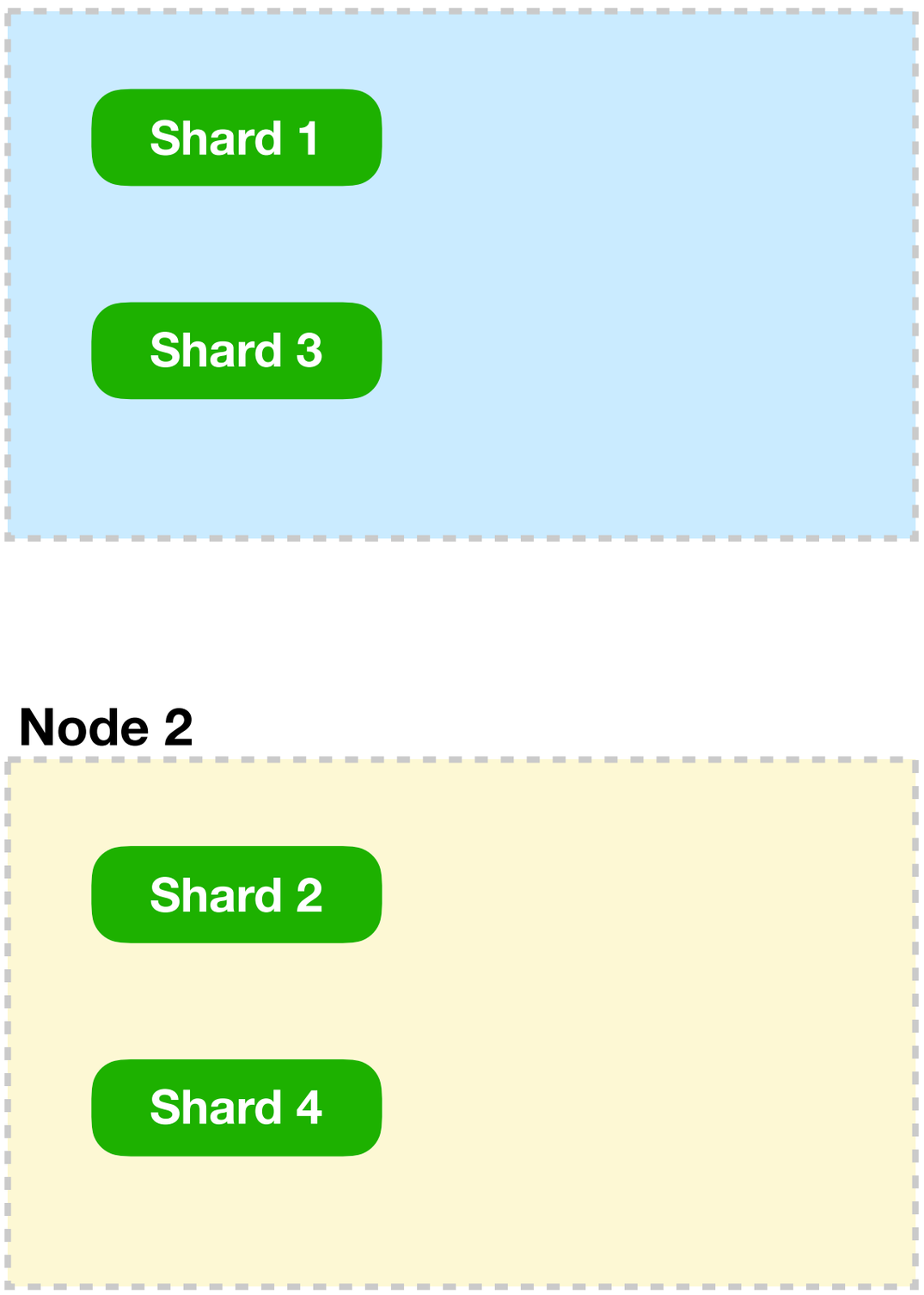
- **Document:** a JSON object

- **Type:** each document has a “_type” field used for filtering when searching on a specific type: this is used to include in an index data that is related, but of different nature (this has been phased out recently)

_type

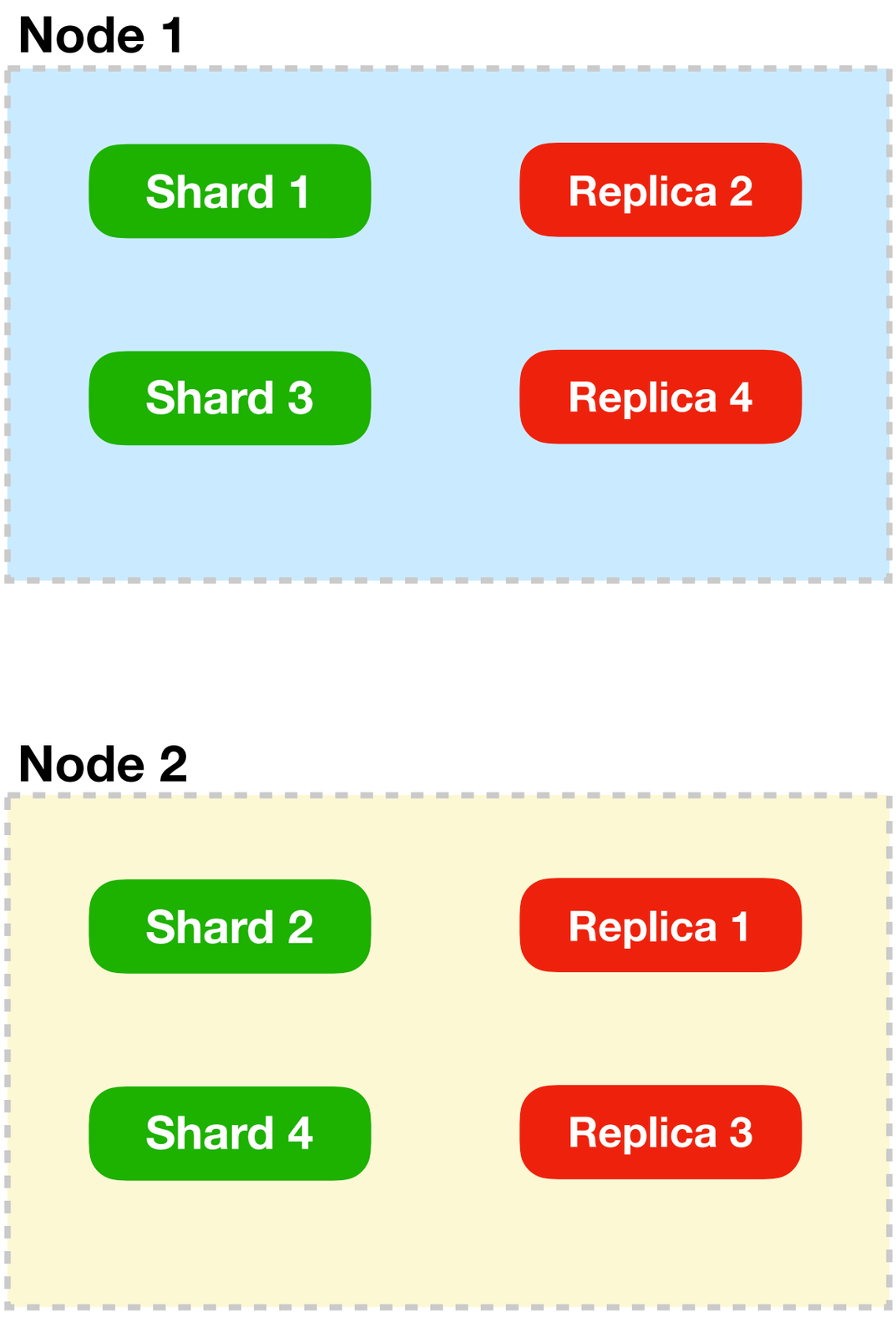

_type="books"
_type="movies"
_index="library_catalog"
q="Lord of the Rings"
q="Shreck"

Sharding



- sharding allows to address the hardware limits of each node, by splitting a data index across multiple nodes.
- each node may contain multiple shards
- sharing also allows for parallalisation of operations (also within the same node): multiple machines (or cores in one machine) can work on the same query at the same time.
- number of shards specified at index creation (default is 5).

Replication



- shards are copied across nodes to create replica shards
- replication delivers high reliability and increased performance for search queries,
- searches can be performed on the replicas in parallel
- number of replicas is defined at indexing (default 1)

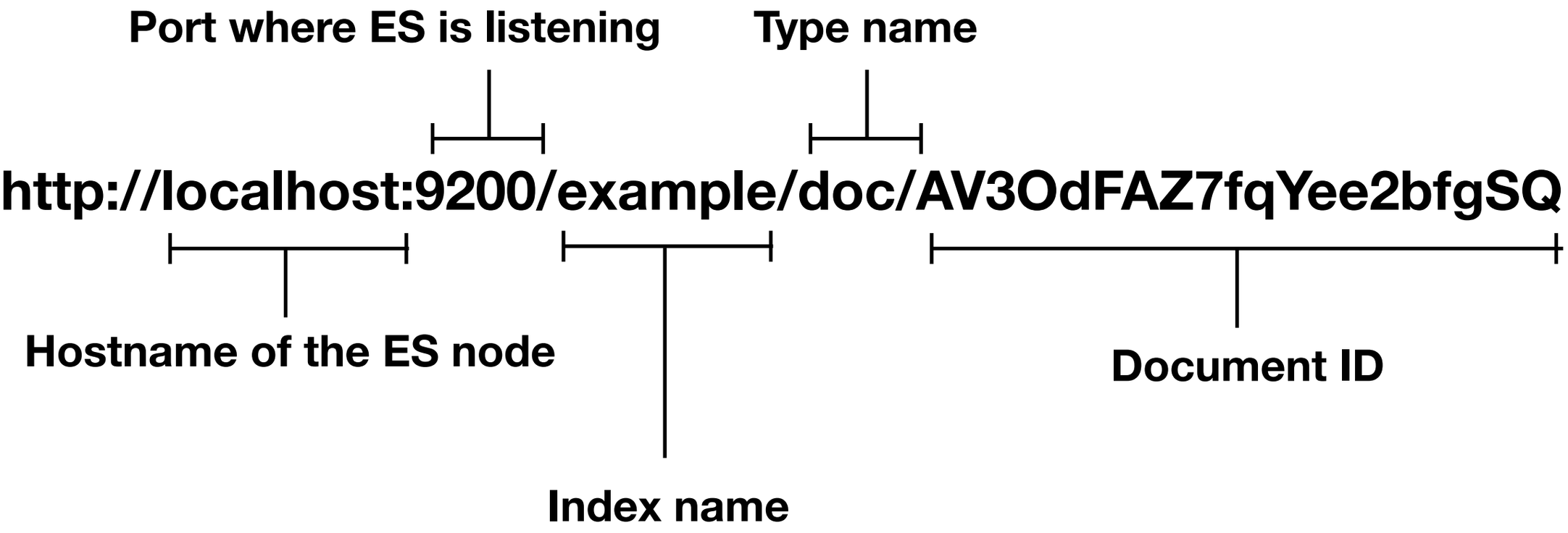
Interacting with Elasticsearch

- Interaction occur as HTTP requests to the RESTful API
- Can use any RESTful client: curl, Postman, Kibana's Dev Tool Console (from Elastic developers)
- Commands are in the form:

```
<REST verb> /<indexname>/<API>
```

- For example: GET /myindex/_search
- (In *curl*: curl -XGET “http://localhost:9200/my_index/_search")

Structure of ES URL



APIs

- **Indeces** API: Create, Delete, Get, Open / Close, Shrink, etc.
    - `PUT test?wait_for_active_shards=2`
- **Document** API: Index, Get, Delete, Update (also variants for multi-document)
    - `POST twitter/tweet/ {"user" : “guidozuc"}`
- **Search** API: execute a search query and get back search hits that match the query. Can pass complex queries
    - `GET /twitter/_search?q=user:guidozuc`
- **Cat** API: get information about the cluster in human readable format
    - `GET /_cat/indeces?v`
- **Explain** API: score explanation for a query and a specific document
- **Cluster** API: node specifications

Hands-on Activity 0: Installation and Basic Interaction

- All material is at https://github.com/ielab/afirm2019
- Activities are in folder hands-on: https://github.com/ielab/afirm2019/tree/master/hands-on
- Visualise Activity 0 readme

Take home from Activity 0

Hands-on Activity 1:
Basic Indexing and Search in Elasticsearch

- What we will learn:
    - How to create an index, add documents
    - How to perform searches
    - How to index a TREC collection (example with ClueWeb12)
- Activity at https://github.com/ielab/afirm2019/hands-on/activity-1/

Take home from Activity 1

Hands-on Activity 2: Boolean Retrieval

- What we will learn:
    - How to perform searches according to the Boolean model

- Activity at https://github.com/ielab/afirm2019/hands-on/activity-2/

Take home from Activity 2

Hands-on Activity 3: Produce a TREC Run

- What we will learn:
    - How to produce a valid TREC formatted run, using the default retrieval model

- Activity at https://github.com/ielab/afirm2019/hands-on/activity-3/

Take home from Activity 3

Hands-on Activity 4: Access a Term Vector

- What we will learn:
    - How to access the term vector of a document. This can be used e.g. to extend retrieval models

- Activity at https://github.com/ielab/afirm2019/hands-on/activity-4/

Take home from Activity 4

Hands-on Activity 5: Implement a New Retrieval Model

- What we will learn:
    - How to access implement a new retrieval model and run searches with it via Elasticsearch
        - There are two version of this: one for Elasticsearch 5.x.x (uses Java) and one for 6.x.x (uses Python/ script similarity). We shall see the one for Elasticsearch 6.x.x

- Activity at https://github.com/ielab/afirm2019/hands-on/activity-5/

Take home from Activity 5

Hands-on Activity 6: Document Priors and Boosting

- What we will learn:
    - How to add document priors to an Elasticsearch index
    - How to boost document scores by including document priors
- Activity at https://github.com/ielab/afirm2019/hands-on/activity-6/

Take home from Activity 6

Hands-on Activity 7:
Text Snippeting for Search Results

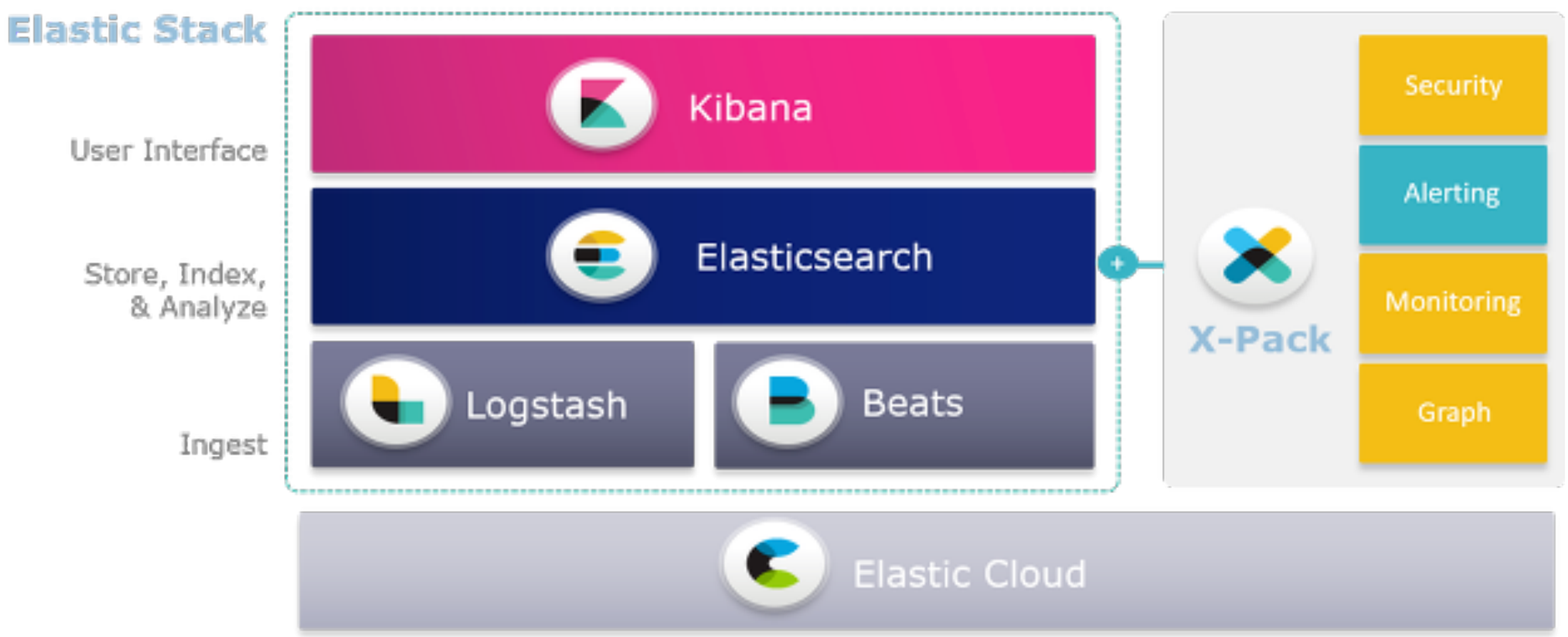
- What we will learn:
    - How to make Elasticsearch produce SERP snippets, that you can use in a search engine GUI
      (e.g. for a user-based experiment)

- Activity at https://github.com/ielab/afirm2019/hands-on/activity-7/

Take home from Activity 7

Beyond Elasticsearch: The ELK Stack

- Elasticsearch is one of the components within a larger stack for ingesting, search analyse and visualise data from any source, in any format, and in real time.



Questions?

If you have questions or follow ups from this practical session, you can **contact me at g.zuccon@uq.edu.au**

Thanks to the ielab team for developing parts of the activities we have seen today; in particular Harrisen Scells, Jimmy, Anton van der Vegt

THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

ielab

www.ielab.io