Elasticsearch
B0M33BDT

PROFINIT

Valerii Ulitin

15/12/2021

About me

PROFINIT

Ing. Valerii Ulitin

Consultant

valerii.ulitin@profinit.eu

linkedin.com/in/valerii-ulitin-005b7615a

Content

PROFINIT

1. General overview
2. Infrastructure
    1. clusters
    2. nodes
    3. indices
    4. shards
3. CRUD operations
4. Search, queries and aggregations
5. Mapping

General overview

Once upon a time…

PROFINIT

- As any good story begins: “Once upon a time…”
  - more precisely: in 1999, Doug Cutting created an open-source project called **Lucene**
- Lucene is:
  - a **search engine library** entirely written in Java
  - a top-level Apache project, as of 2005
  - great for full-text search
- But, Lucene is also:
  - a library that you have to incorporate into your application
  - challenging to use
  - not originally designed for scaling

The Birth of Elasticsearch

PROFINIT

- In 2004, Shay Banon developed a product called **Compass**
  - built on top of Lucene, Shay's goal was to have search integrated into Java applications as simply as possible
- The need for **scalability** became a top priority
- In 2010, Shay completely rewrote Compass with two main objectives:
  - **distributed from the ground up in its design**
  - **easily used by any other programming language**
- He called it Elasticsearch
  - …and we all lived happily ever after!
- Today Elasticsearch is the most popular enterprise search engine

What is Elasticsearch?

PROFINIT

- Elasticsearch is a **distributed** search and analytics engine for all types of data, including textual, numerical, geospatial, structured, and unstructured
- **Key features** of Elasticsearch:
  - Distributed
  - Scalable
  - Fast
  - Shipped with simple yet powerful REST API
  - Easily used by any other programming languages
  - Supports 34 text languages and provides analyzers for each

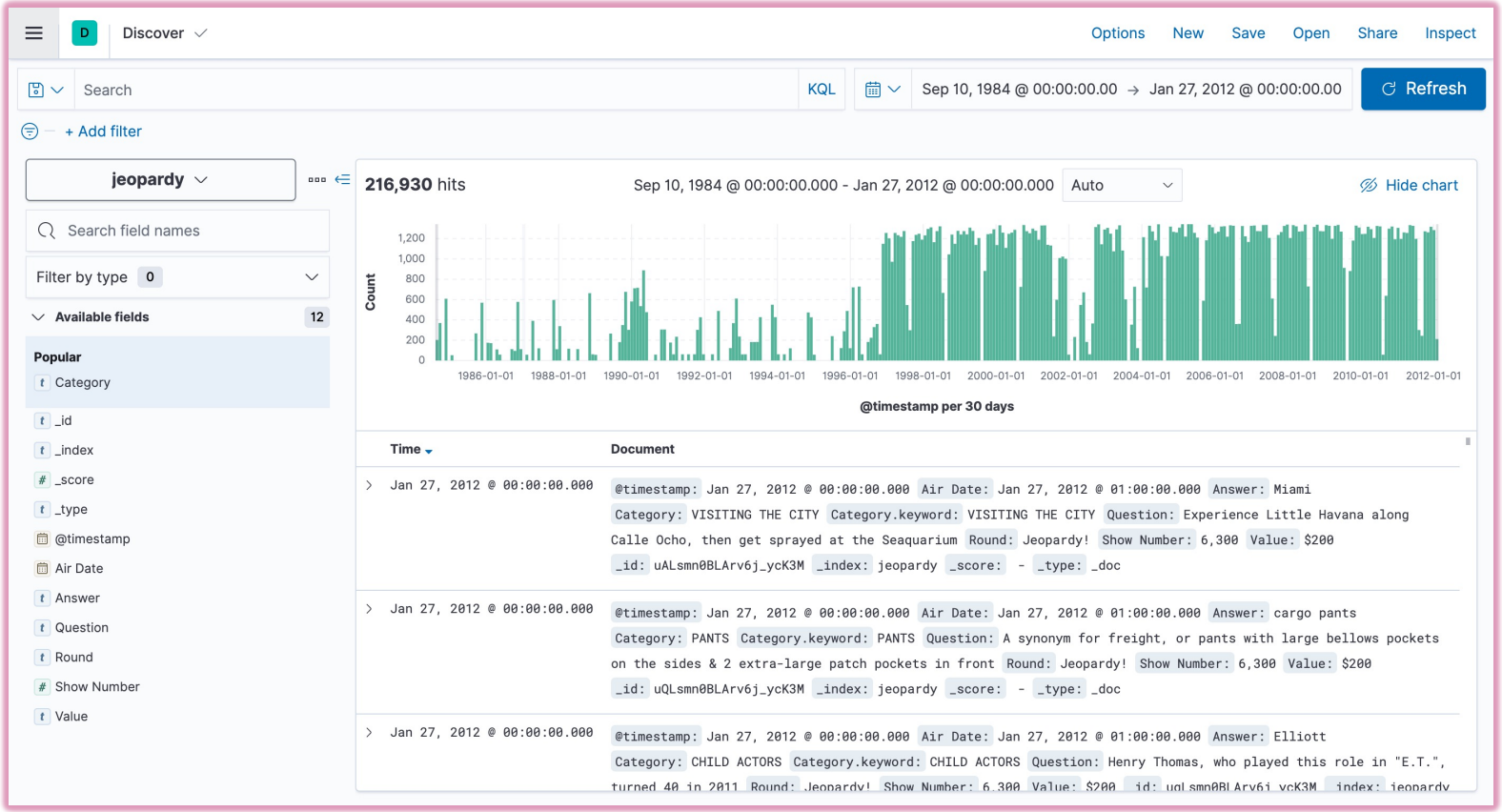
Are there others like Elasticsearch?

PROFINIT

- The closest one is the Amazon OpenSearch Service
- **Formerly** known as Amazon Elasticsearch Service
- Isn’t it just Elasticsearch in disguise? Well… yes and no:
  - it is a fork of older version of Elasticsearch (7.10.2), when it still was under an open source license (Apache 2.0)
  - because it is a fork it shares base functionality with Elasticsearch but with every update it diverges further away from it
- You may consider Amazon **OpenSearch** service if:
  - you already have other services running in AWS
  - you are true open source sympathizer and would like to contribute one day

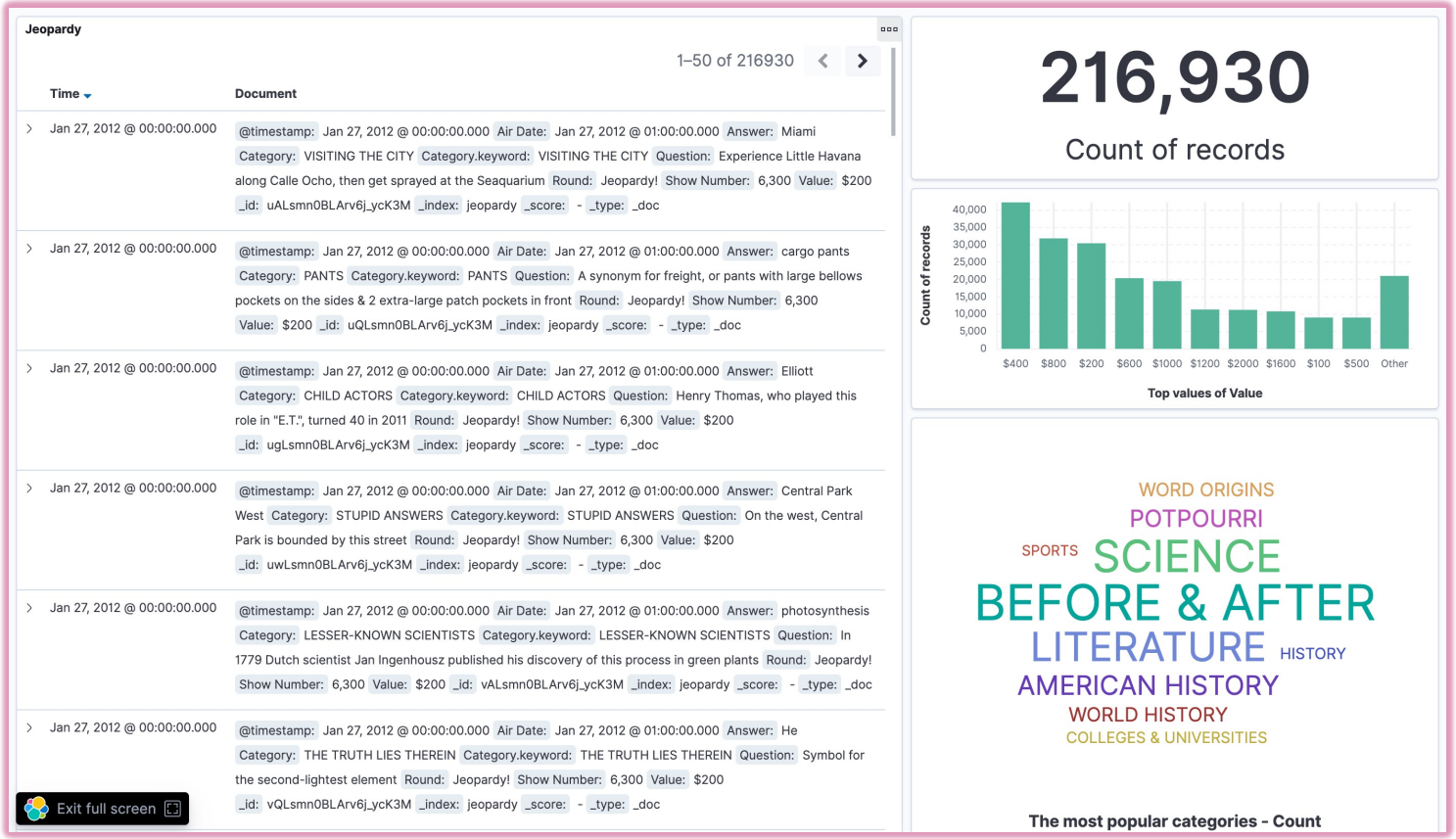
How Elasticsearch looks like

PROFINIT



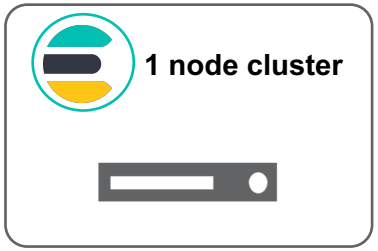
How Elasticsearch looks like #2

PROFINIT



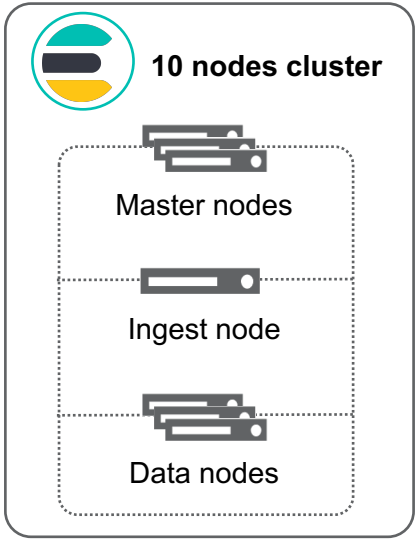
Distributed search

PROFINIT

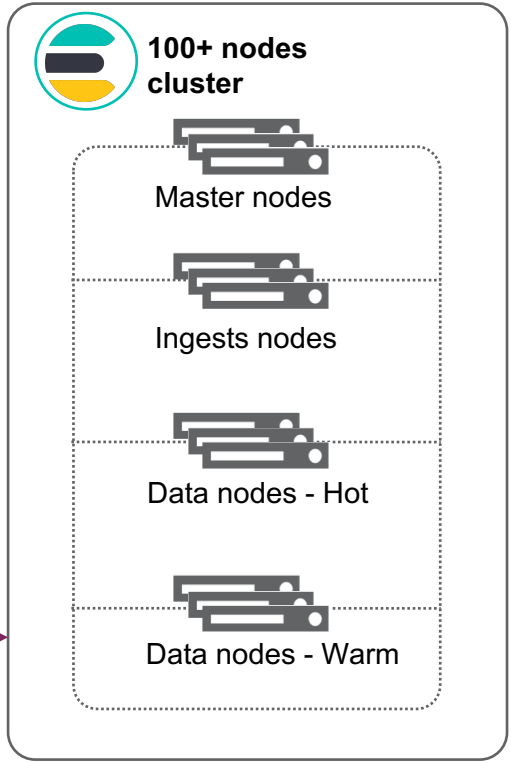


A **node** is an instance of Elasticsearch

A **cluster** is a collection of Elasticsearch nodes



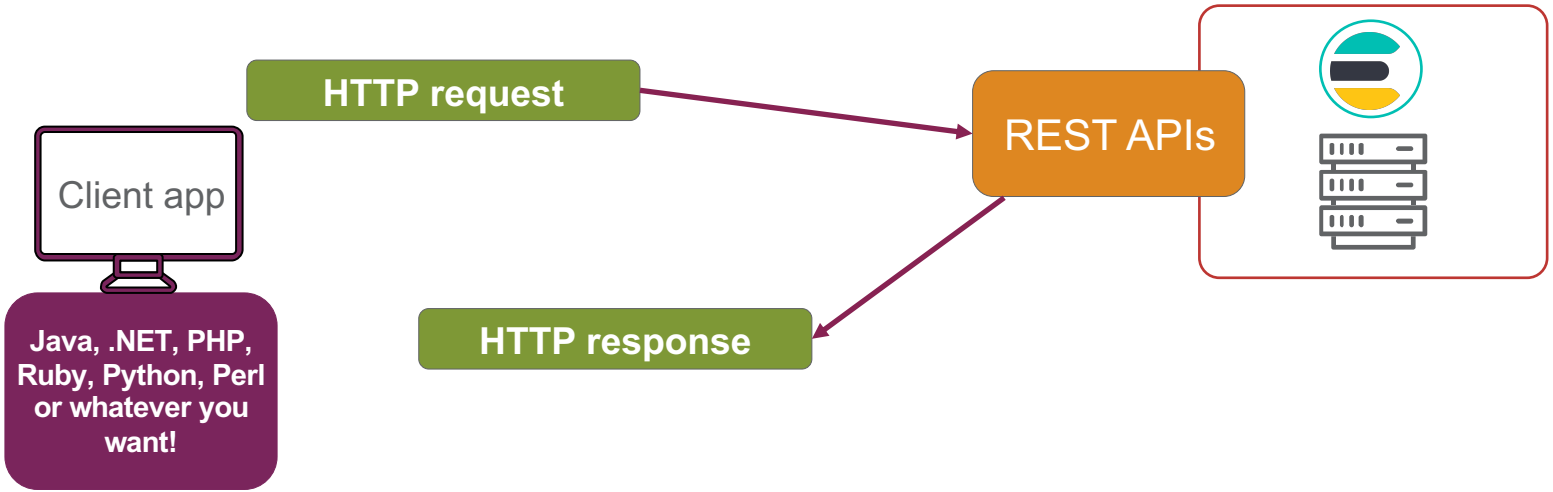
Your cluster can grow as your needs grow



Easily used by other programming languages

PROFINIT

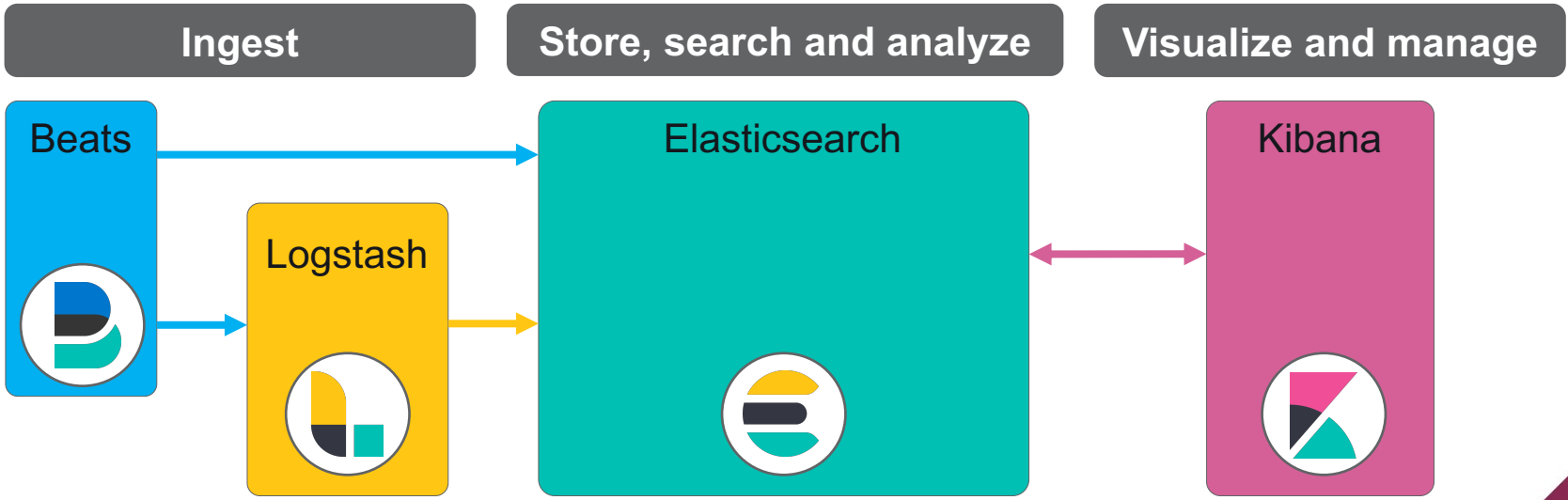
- Elasticsearch provides REST APIs for communicating with a cluster over HTTP(S)
  - allows client applications to be written in any language!



Elastic Stack (ELK)

PROFINIT

› Elasticsearch is a **distributed** search and analytics engine for all types of data, including textual, numerical, geospatial, structured, and unstructured
  – reliably and securely take data from any source, in any format, as well as search, analyze, and visualize it in real time



Standalone (self managed) Elastic Cloud Enterprise (self managed) Elastic Cloud (SaaS)

Common use cases

PROFINIT

- Application search
- Website search
- Enterprise search
- Logging and log analytics
- Infrastructure metrics and container monitoring
- Application performance monitoring
- Geospatial data analysis and visualization
- Security analytics
- Business analytics

Infrastructure

Zoo

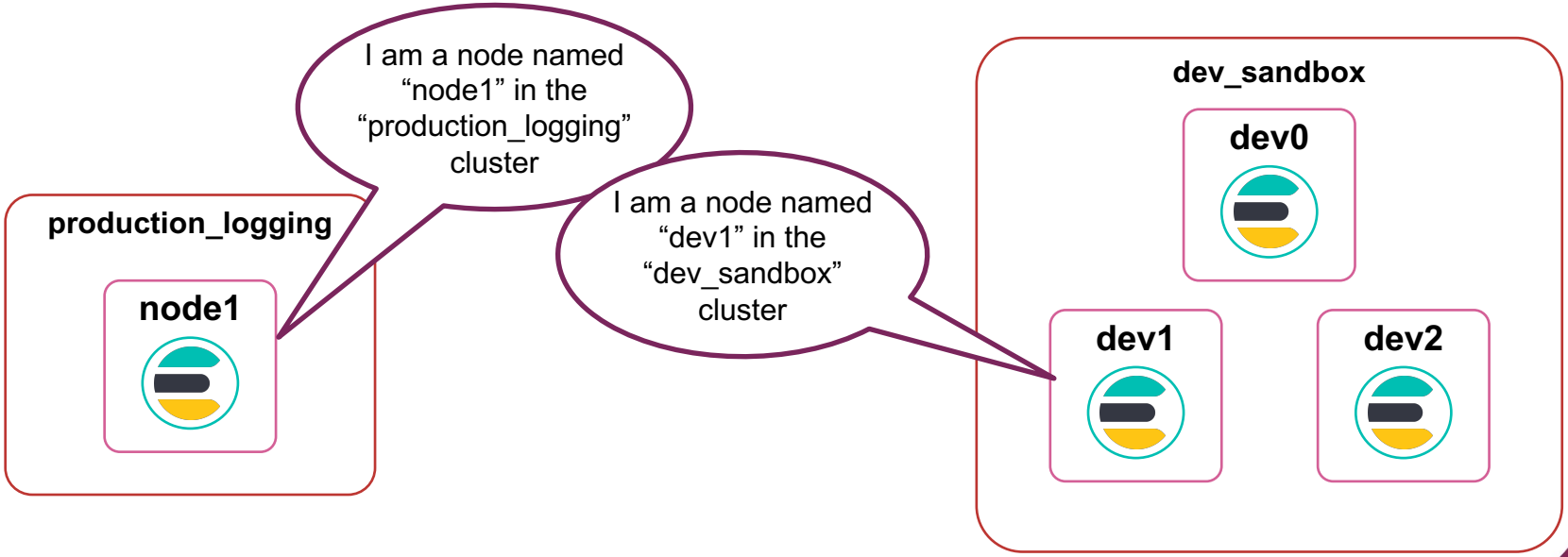
PROFINIT

- A **cluster** is a collection of Elasticsearch nodes
- A **node** is an instance of Elasticsearch
- An Elasticsearch **index** is a collection of documents that are related to each other
  - an index is a **virtual namespace** that points to a number of shards
- A **shard** is a worker unit that holds data and can be assigned to nodes
  - **primary** shards: the original shards of an index
  - **replica** shards: copies of the primary shard

Cluster

PROFINIT

- Every node belongs to a single **cluster**
- A cluster is one or multiple nodes working together in a distributed manner



Node

PROFINIT

- A **node** is an instance of Elasticsearch
  - a Java process that runs in a JVM
- A node is typically deployed 1-to-1 to a host

Nodes and their roles

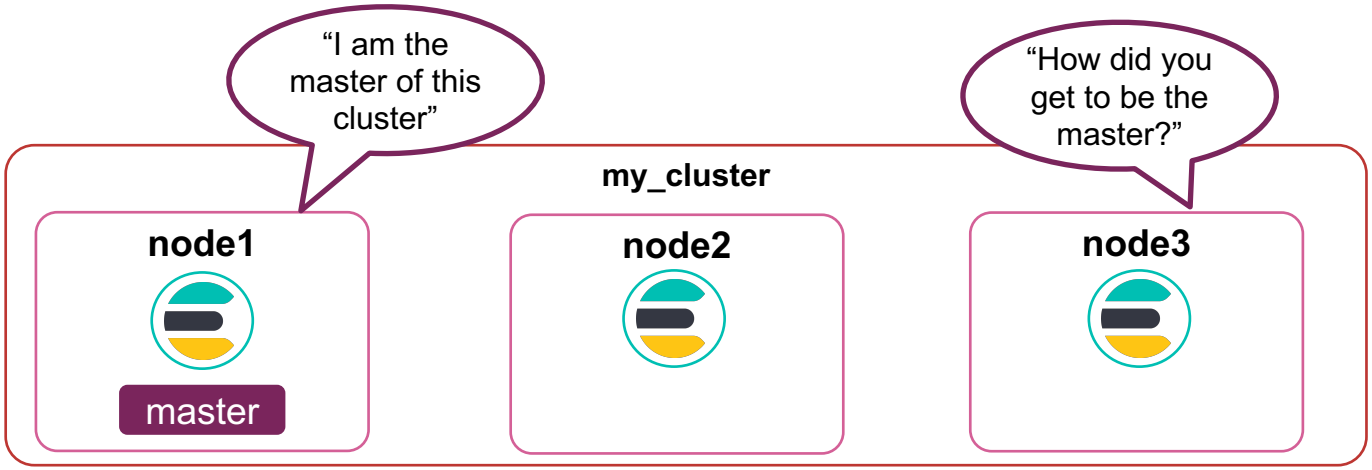
PROFINIT

- There are several **roles** a node can have:
  - **master/master-eligible**
  - **data**
  - **ingest**
  - **machine learning**
- Nodes can take on multiple roles at the same time
  - Or they can be **dedicated** nodes that only take on a single role

The Master Node

PROFINIT

- Every cluster has one node designated as the master
- The master node is in charge of cluster-wide settings and changes, like:
  - creating, updating or deleting indices (incl. mappings and settings)
  - adding or removing nodes
  - allocating shards to nodes

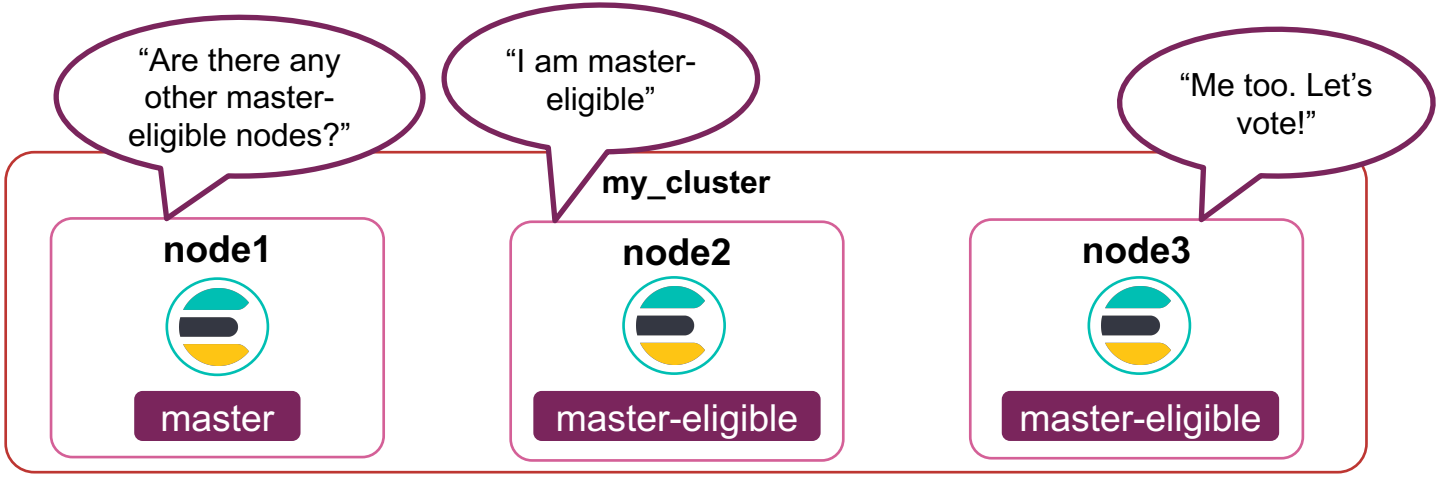


The Master-eligible Node

PROFINIT

- The master node is **elected** from the **master-eligible** nodes in the cluster
  - a node is master-eligible if **node.master*** is set to true (the default value)
  - only master-eligible nodes can **vote**

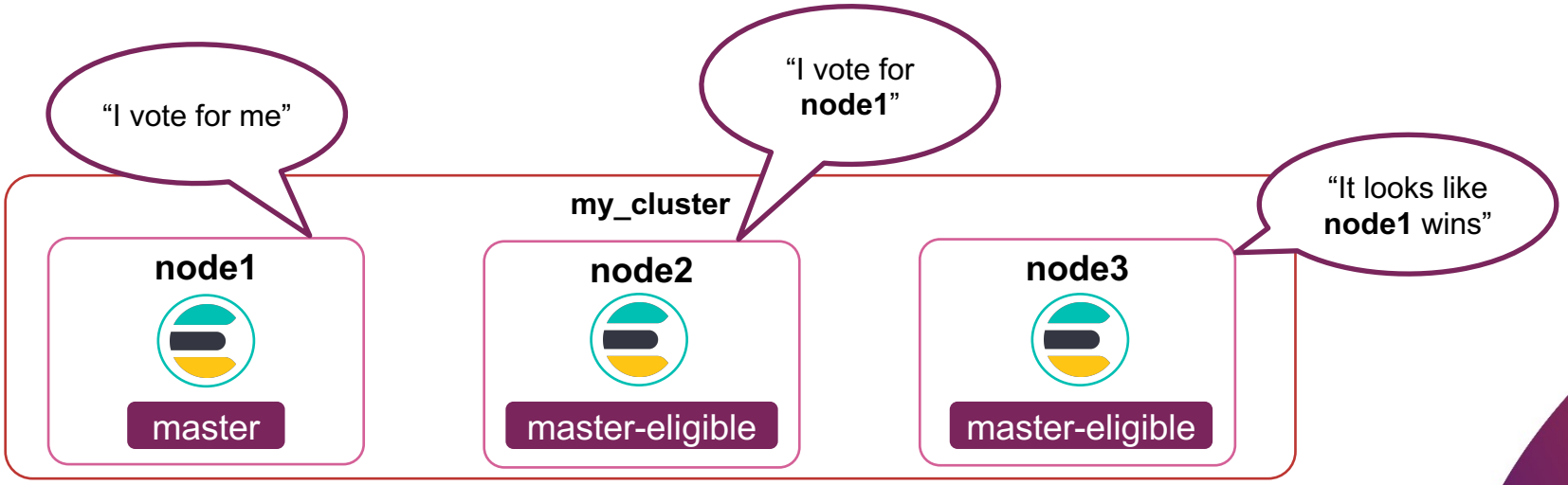
*the parameter is set in the **elasticsearch.yml**



Master Elections

PROFINIT

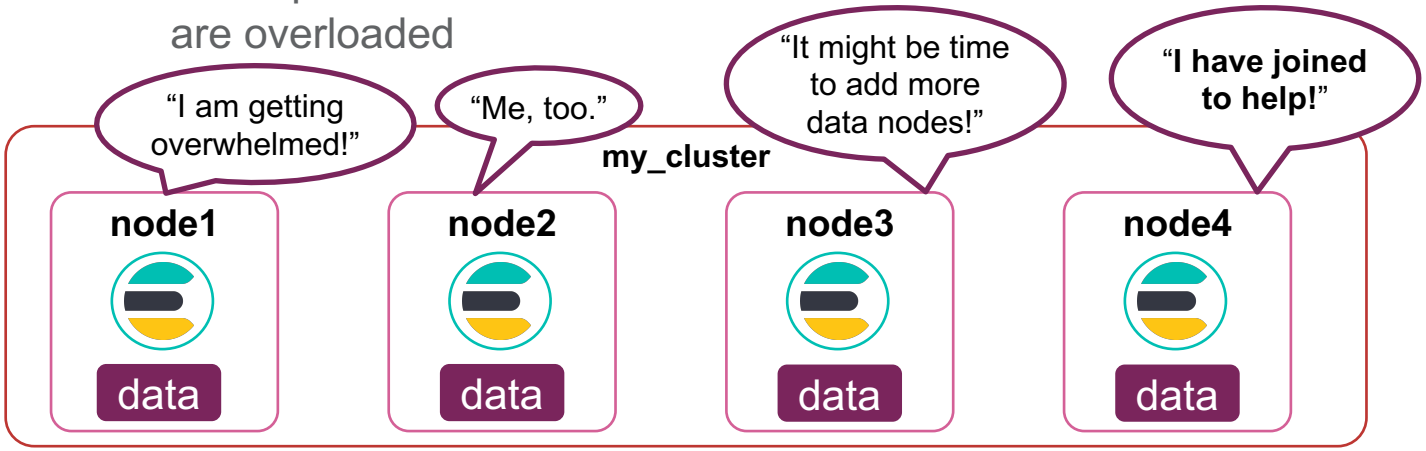
- The number of votes to win the election is automatically handled by Elasticsearch to ensure a **quorum**
  - which is **[N/2 + 1]**, where N is the number of master-eligible nodes
- It is important to have a **quorum** to avoid a "split brain"



Data Nodes

PROFINIT

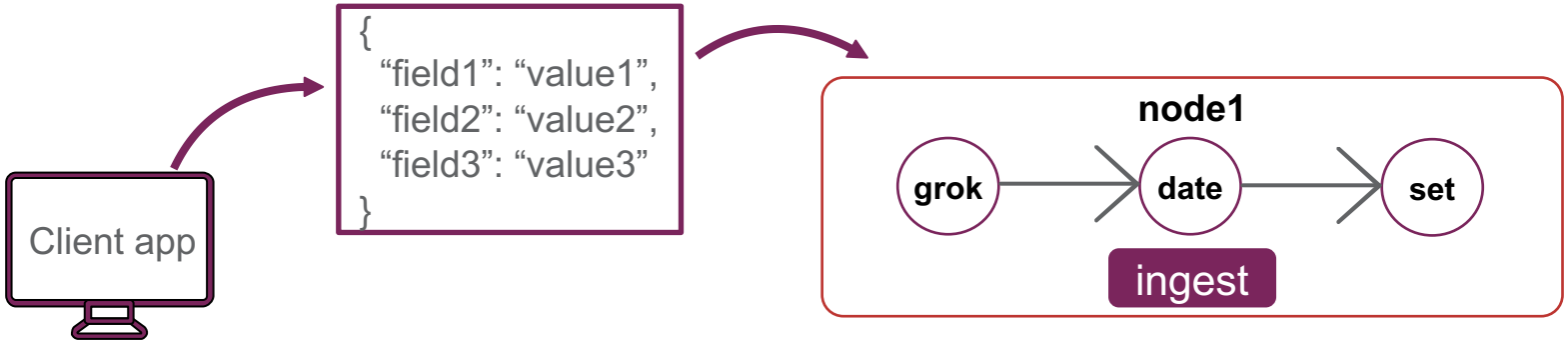
- Data nodes have two main features:
  - they hold the shards that contain the documents you have indexed
  - they execute data related operations like CRUD, search and aggregations
- All nodes are data nodes by default
- Data nodes are I/O, CPU, and memory-intensive
  - it is important to monitor these resources and add more data nodes if they are overloaded



Ingest Nodes

PROFINIT

- **Ingest nodes** provide the ability to
  - pre-process a document right before it gets indexed
- All nodes are ingest nodes by default

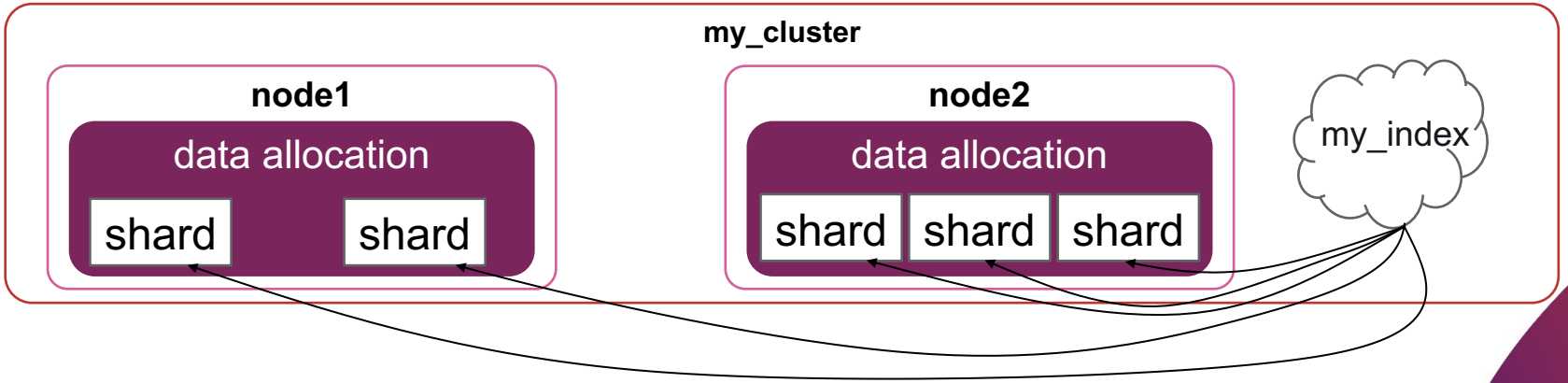


When indexing a doc, you can specify a **pipeline**

Shards and index relationship

PROFINIT

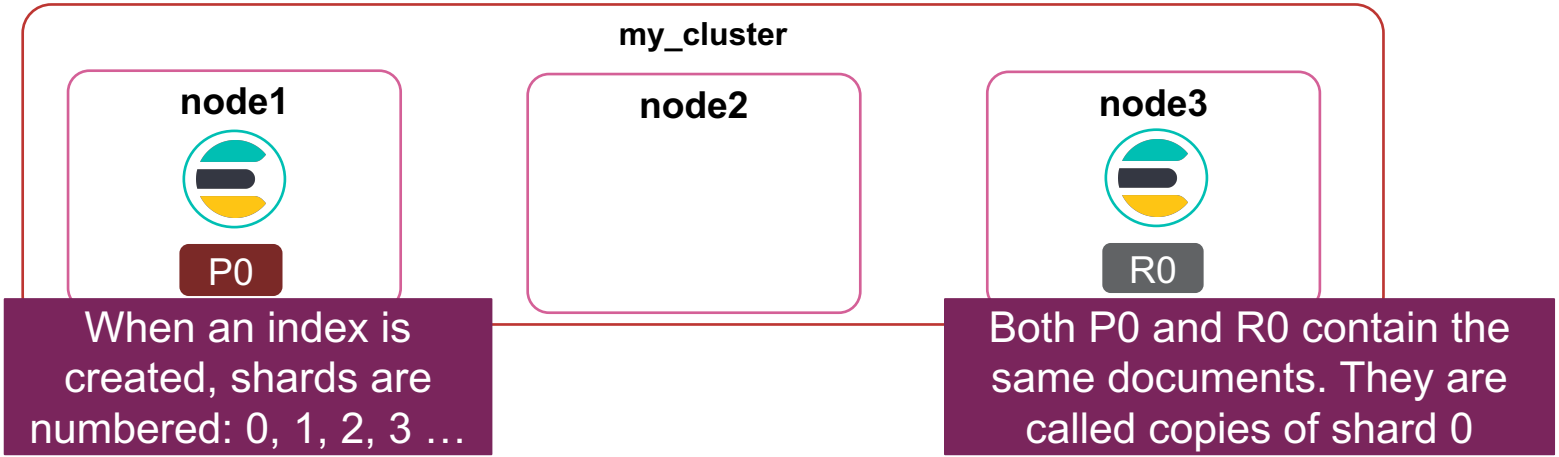
- A **shard** is a worker unit that holds data and can be assigned to nodes
- An index is a **virtual namespace** which points to a number of shards
  - an index is “split” into **shards** before any documents are indexed



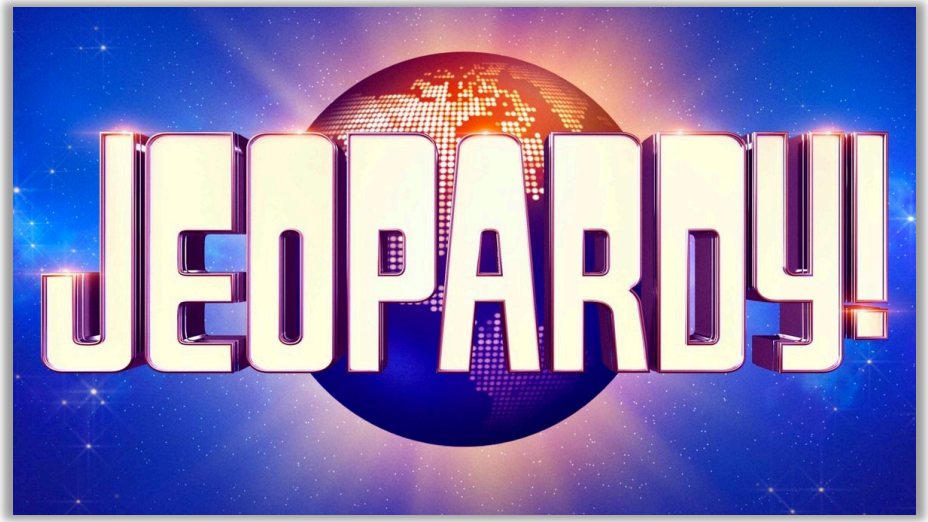
Primary vs. Replica

PROFINIT

- There are two types of shards
  - **primary** shards: the original shards of an index
  - **replica** shards: copies of the primary shard
- Documents are replicated between a primary and its replicas
  - a primary and all replicas are guaranteed to be on different nodes



CRUD operations



PROFINIT

dataset link: https://drive.google.com/file/d/0BwT5wj_P7BKXUl9tOUJWYzVvUjA/view?usp=sharing&resourcekey=0-uFrn8bQkUfSCvJImtKGCdQ

source:
https://www.jeopardy.com/jbuzz/behind-scenes/what-are-some-questions-about-jeopardy
https://g.cz/porad-riskuj-legendarni-televizni-soutez-ve-ktere-jste-mohli-vyhrat-pracku-i-felicii-v-kombiku/

Documents must be JSON Objects

PROFINIT

› Imagine records that are currently in a database table:

Show Number	Category	Air Date	Question	Answer	Value	Round
6298	20th CENTURY WORDS & PHRASES	2012-01-25	This word for a large self-service store that sells household goods as well as groceries hit the shelves in 1933	a supermarket	$600	Jeopardy!
3746	SECOND-MOST POPULOUS CITIES	2000-12-11	Brno	Czech Republic	$1000	Double Jeopardy!

› Each record needs to be converted to a **JSON** object:

JSON consists of ***fields...***

```
{
   "Show Number": 3746,
   "Category": "SECOND-MOST POPULOUS CITIES",
   "Air Date": "2000-12-11",
   "Question": "Brno",
   "Answer": "Czech Republic",
   "Value": "$1000",
   "Round": "Double Jeopardy!"
}
```

…and ***values***

Document Store

PROFINIT

- Elasticsearch is a distributed ***document store***
  - it can store and return complex data structures that are represented as JSON objects
- A ***document*** is a serialized JSON object that is stored in Elasticsearch under a unique ID

A JSON object…

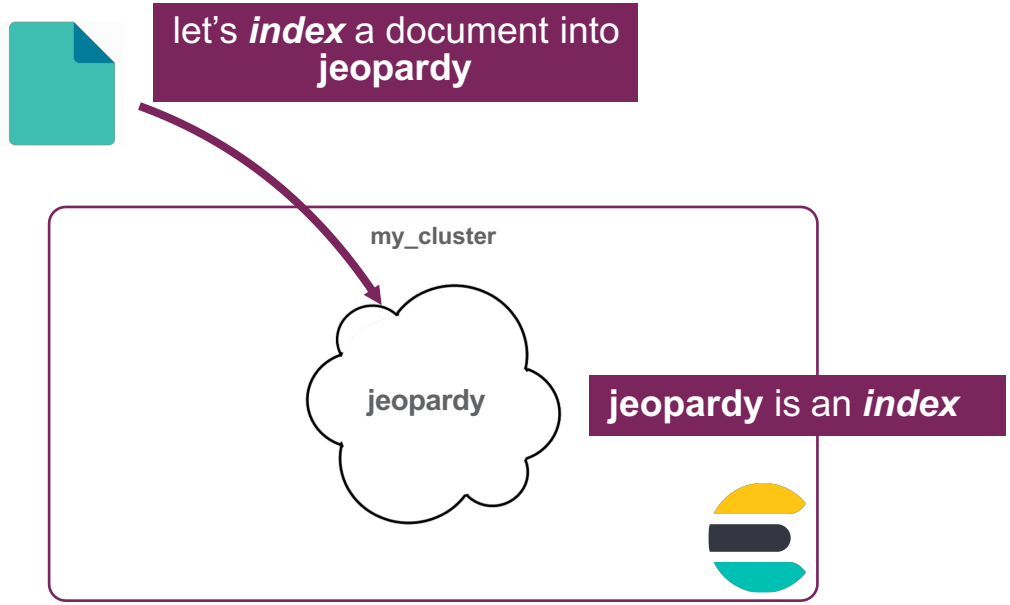
```
{
    "Show Number": 3746,
    "Category": "SECOND-MOST POPULOUS CITIES",
    "Air Date": "2000-12-11",
    "Question": "Brno",
    "Answer": "Czech Republic",
    "Value": "$1000",
    "Round": "Double Jeopardy!"
}
```

…is stored in Elasticsearch as a document

Documents are indexed into an index

PROFINIT

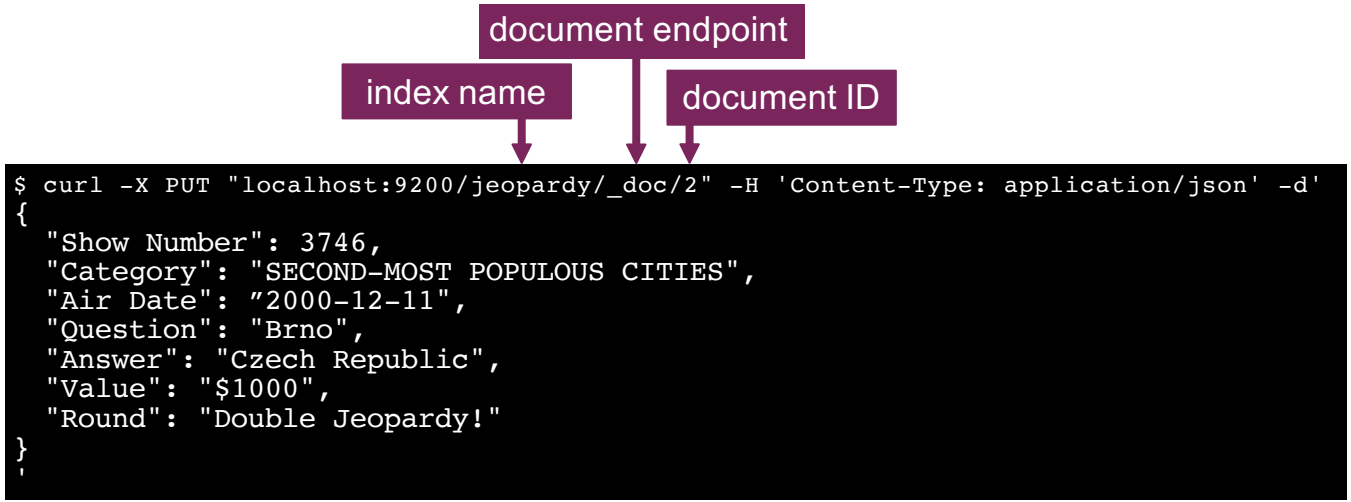
- In Elasticsearch, a document is ***indexed*** into an ***index***
  - yes, **index** is uded as both a noun and a verb



Index a document

PROFINIT

- The ***Index API*** is used to index a document
  - use a **PUT** or a **POST** and add the document in the body request
  - notice that the **index** and an ***ID*** *were* specified
  - if no ***ID*** is provided, Elasticsearch generates one for you



Kibana Dev Tools

PROFINIT

- Using curl all the time can be a bit tedious and convoluted
- **Kibana** has a developer tool named **Console** for creating and submitting Elasticsearch requests in a simpler fashion

The **Console** syntax will be used throughout this presentation in examples

The response

PROFINIT

201 response if successful

```
{
  "_index" : "jeopardy",
  "_type" : "_doc",
  "_id" : "2",
  "_version" : 1,
  "result" : "created",
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "failed" : 0
  },
  "_seq_no" : 216934,
  "_primary_term" : 1
}
```

The ID is stored in the **_id** field

Every document has a **_version**

What if the document ID already exists?

PROFINIT

- The document gets ***reindexed***
  - the new document ***overwrites*** the existing one

```
PUT jeopardy/_doc/2
{
  "Show Number": 3746,
  "Category": "SECOND-MOST POPULOUS
    CITIES",
  "Air Date": "2000-12-11",
  "Question": "Brno",
  "Answer": "Czech Republic",
  "Value": "$1000",
  "Round": "Double Jeopardy!"
}
```

```
{
  "_index" : "jeopardy",
  "_type" : "_doc",
  "_id" : "2",
  "_version" : 2,
  "result" : "updated",
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "failed" : 0
  },
  "_seq_no" : 216935,
  "_primary_term" : 1
}
```

200 response (instead of **201**)

_version is incremented "**updated**" instead of "**created**"

The _create endpoint

PROFINIT

- If you do ***not*** want a document to be overwritten if it already exists, use the **_create** endpoint
  - no indexing occurs and returns a **409** error message

```
PUT jeopardy/_create/2
{
  "Show Number": 3746,
  "Category": "SECOND-MOST POPULOUS
    CITIES",
  "Air Date": "2000-12-11",
  "Question": "Brno",
  "Answer": "Czech Republic",
  "Value": "$1000",
  "Round": "Double Jeopardy!"
}
```

```
{
  "error" : {
    "root_cause" : [
      {
        "type" : "version_conflict_engine_exception",
        "reason" : "[2]: version conflict, document already exists (current version [2])",
        "index_uuid" : "NZ8_38tASnOp3tY80UQZAQ",
        "shard" : "0",
        "index" : "jeopardy"
      }
    ],
    "type" : "version_conflict_engine_exception",
    "reason" : "[2]: version conflict, document already exists (current version [2])",
    "index_uuid" : "NZ8_38tASnOp3tY80UQZAQ",
    "shard" : "0",
    "index" : "jeopardy"
  },
  "status" : 409
}
```

Fails if a document with **_id=2** is already indexed

Retrieving a document

PROFINIT

- Use **GET** to retrieve an indexed document
  - note that both the **index** and **ID** are specified
  - response code is **200** if the document is found, **404** if not

"I am looking for the record with id=2"

```
GET jeopardy/_doc/2
```


```
{
  "_index" : "jeopardy",
  "_type" : "_doc",
  "_id" : "2",
  "_version" : 2,
  "_seq_no" : 216935,
  "_primary_term" : 1,
  "found" : true,
  "_source" : {
    "Show Number" : 3746,
    "Category" : "SECOND-MOST POPULOUS CITIES",
    "Air Date" : "2000-12-11",
    "Question" : "Brno",
    "Answer" : "Czech Republic",
    "Value" : "$1000",
    "Round" : "Double Jeopardy!"
  }
}
```


The original document is returned in the **_source** field

The _update endpoint

PROFINIT

- If you want to update fields in a document, use the **_update** endpoint
  - make sure to add the “**doc**” context

```
POST jeopardy/_update/2
{
  "doc": {
    "Air Date": "2000-12-10"
  }
}
```

update the “Air Date” field

```
{
  "_index" : "jeopardy",
  "_type" : "_doc",
  "_id" : "2",
  "_version" : 3,
  "result" : "updated",
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "failed" : 0
  },
  "_seq_no" : 216936,
  "_primary_term" : 1
}
```

_version is incremented

Deleting a document

PROFINIT

- Use **DELETE** to delete an indexed document
  - response code is **200** if the document is found, **404** if not

```
DELETE jeopardy/_doc/2
```

```
{
  "_index" : "jeopardy",
  "_type" : "_doc",
  "_id" : "2",
  "_version" : 4,
  "result" : "deleted",
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "failed" : 0
  },
  "_seq_no" : 216937,
  "_primary_term" : 1
```

CRUD operations overiew

PROFINIT

Index

```
POST    faq/_doc
{
    ”question" : "Who writes Wikipedia?",
    "answer": "Wikipedia is written and edited by volunteers from …"
}
```

```
PUT     faq/_doc/4
{
    "question" : "Who writes Wikipedia?",
    "answer": "Wikipedia is written and edited by volunteers from …"
}
```

Create

```
PUT     faq/_create/4
{
    "question" : "Who writes Wikipedia?",
    "answer": "Wikipedia is written and edited by volunteers from …"
}
```

Read

```
GET     faq/_doc/4
```

Update

```
POST    faq/_update/4
{
    "doc" : {
        "answer" : "Wikipedia is written and edited by people"
    }
}
```

Delete

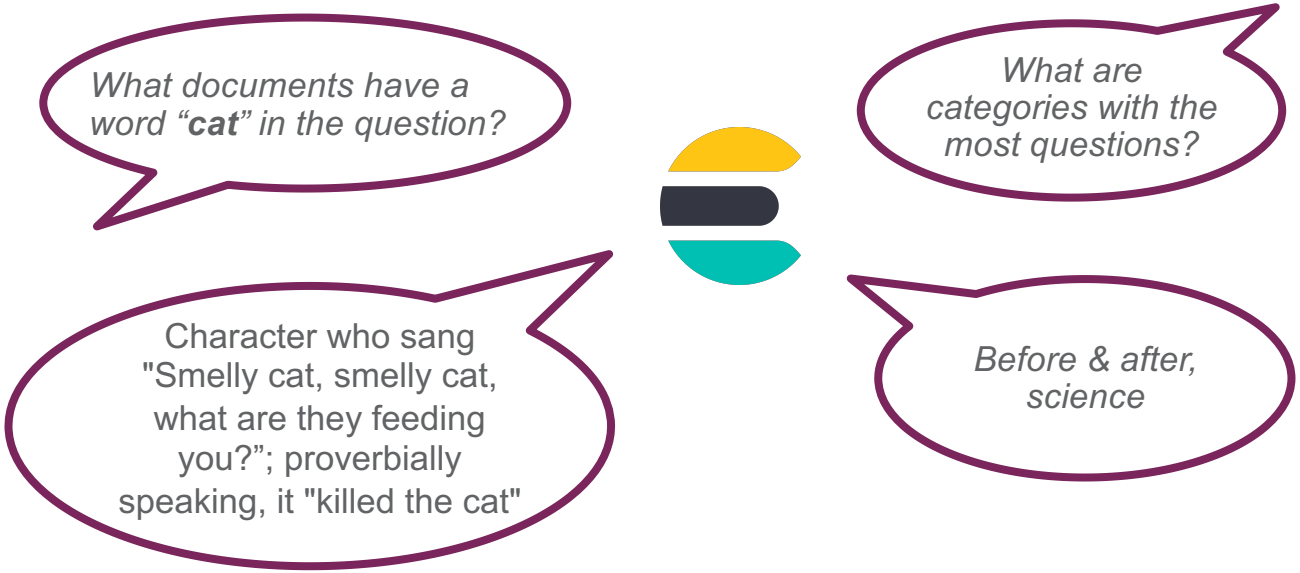
```
DELETE  faq/_doc/4
```

Search, queries and aggregations

Search

PROFINIT

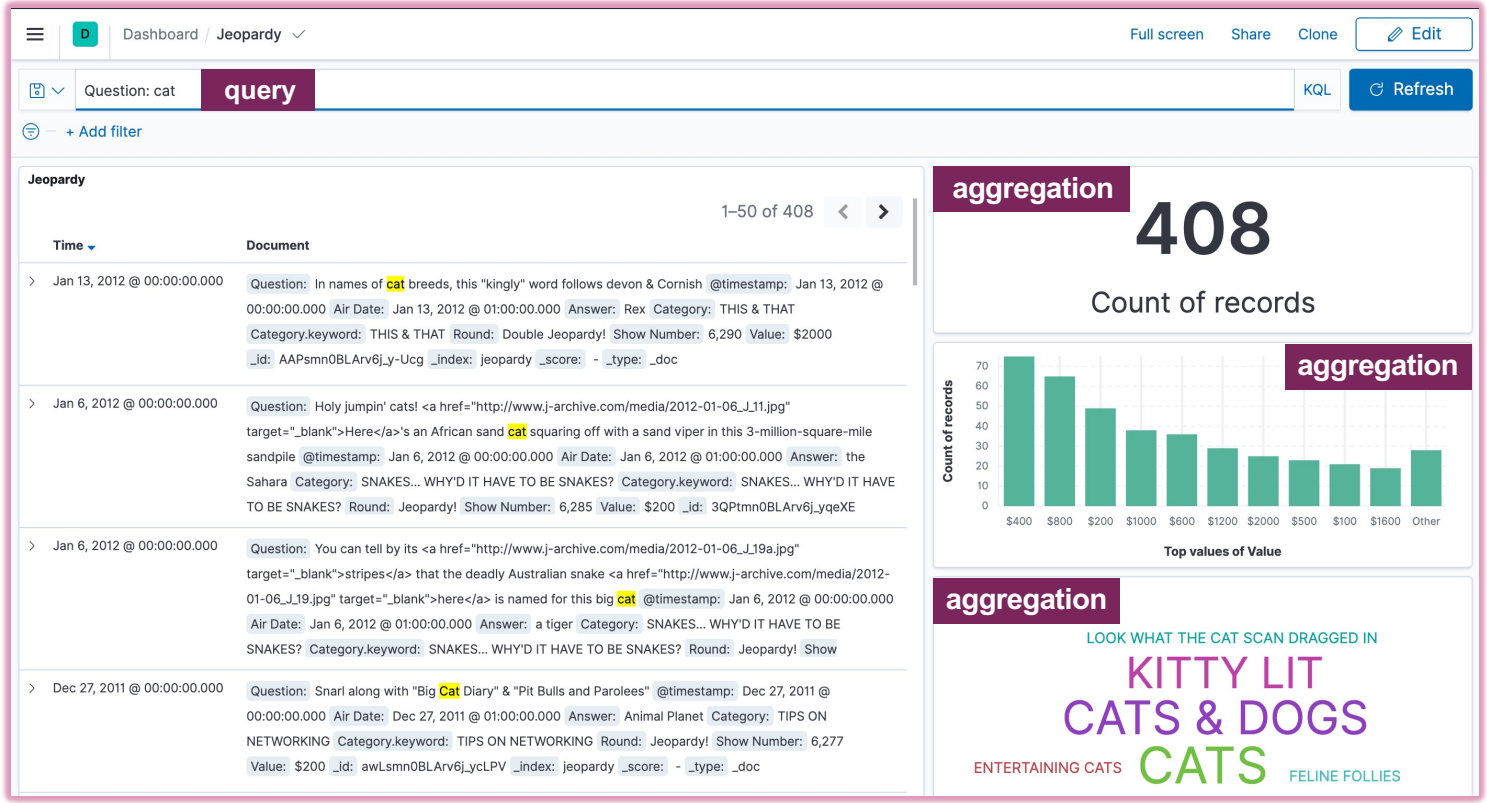
- Search is asking questions and getting answers
- There are two main ways to search:
  - **queries**
  - **aggregations**



Queries and aggregations

PROFINIT

› What is the difference between queries and aggregations?



Queries and aggregations request

PROFINIT

- How to create an aggregation request?
  - simply send a **GET** request using the **_search** endpoint

```
GET jeopardy/_search
{
  "query": {
    "match": {
      "Question": "beer"
    }
  },
  "aggs": {
    "num_questions_per_value": {
      "terms": {
        "field": "Category.keyword"
      }
    }
  }
}
```

_search = the search endpoint

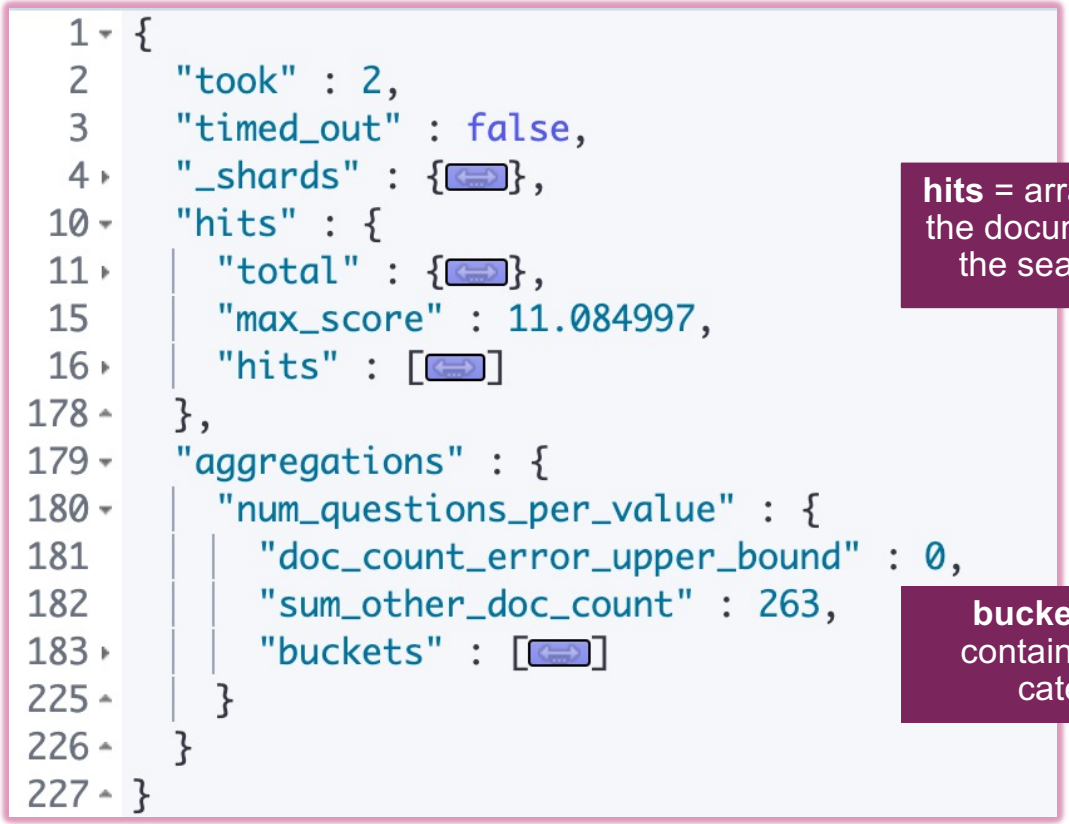
query = the query clauses to match documents

aggs = the aggregation clauses to summarize the data

Don't worry about the "keyword" here. We'll cover it a bit later.

Queries and aggregations response

PROFINIT



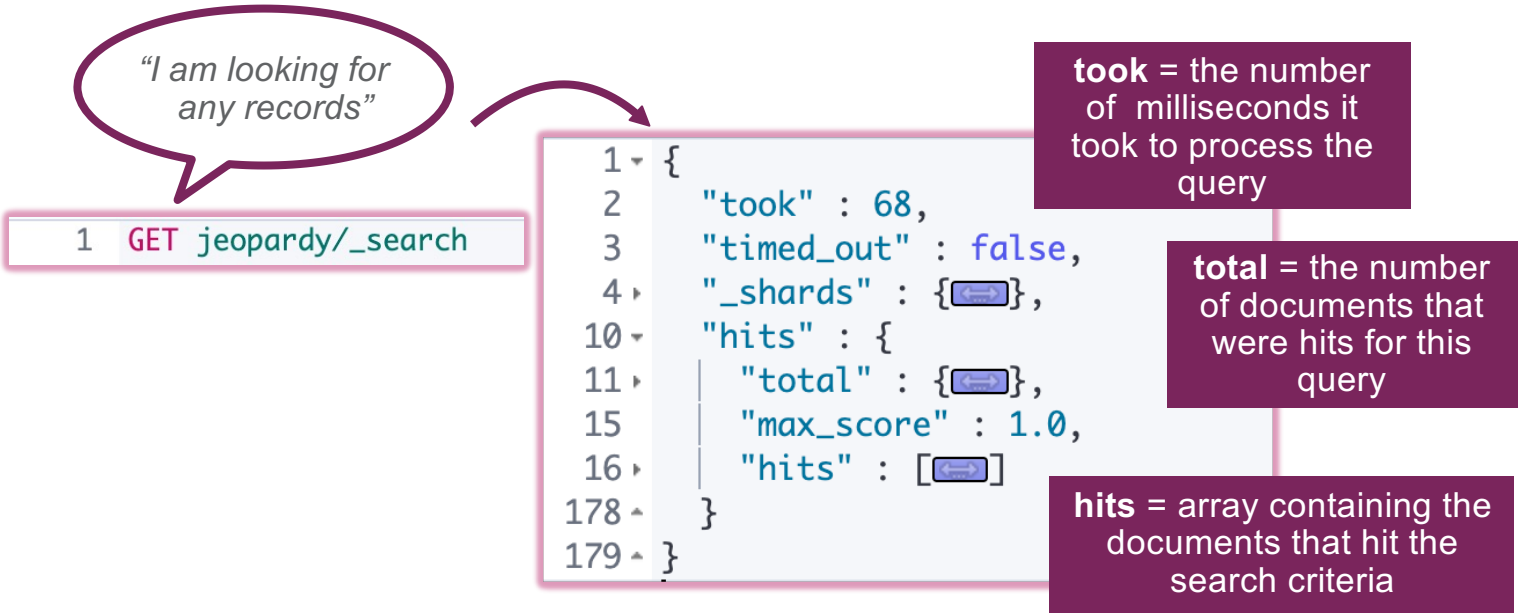
hits = array containing the documents that hit the search criteria

buckets = array containing the top categories

A simple search

PROFINIT

- Use a **GET** request sent to the **_search** endpoint
  - every document is a ***hit*** for this search
  - by default, Elasticsearch returns 10 hits



Search examples

PROFINIT

```
GET log_server1,log_server2/_search
```

```
GET log_server*/_search
```

```
GET jeopardy/_search
{
  "query": {
    "match": {
      "Question": "math"
    }
  }
}
```

```
GET jeopardy/_search
{
  "query": {
    "range": {
      "Air Date": {
        "gte": "2012-01-01"
      }
    }
  }
}
```

```
GET jeopardy/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "match": {
            "Question": "math"
          }
        }
      ],
      "filter": [
        {
          "range": {
            "Air Date": {
              "gte": "2012-01-01"
            }
          }
        }
      ]
    }
  }
}
```

Query types

Types

PROFINIT

- There are plenty of different query types for the **_search** endpoint
- We will cover just some of them:
  - **match** query
  - **match_phrase** query
  - **multi_match** query
  - **bool** query
- Although, other types can be found in the official documentation
  - https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html

Match query

PROFINIT

- Suppose we are interested in all the questions that mentioned
  - “**Czech republic**”
- Let’s search for it in the “**Question**” field
- What do you think is required of a document to be a hit?

```
GET jeopardy/_search
{
  "query": {
    "match": {
      "Question": "Czech republic"
    }
  }
}
```

https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-match-query.html

Match query

PROFINIT

- Suppose we are interested in all the questions that mentioned
  - "**Czech republic**"
- Let's search for it in the "**Question**" field
- By default, the **match** query uses "***or***" logic if multiple terms appear in the search query
  - any document with the term "Czech republic", "Czech" **or** "*republic*" in the "**Question**" field will be a hit

```
GET jeopardy/_search
{
  "query": {
    "match": {
      "Question": "Czech republic"
    }
  }
}
```

The match_phrase query

PROFINIT

- The ***match_phrase*** query is for searching text when you want to find terms that are near each other
  - all the terms in the phrase must be in the document
  - the position of the terms must be in the same relative order

```
GET jeopardy/_search
{
  "query": {
    "match_phrase": {
      "FIELD": "PHRASE"
    }
  }
}
```

The field you want to search

The phrase you are searching for

https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-match-query-phrase.html

Example of match_phrase

PROFINIT

- Let's try the "**Czech republic**" search using **match_phrase** instead of **match**:
  - only 31 hits (instead of 448 with **match**)
  - be careful, relevant questions may be ommited, e.g. "*The Romantic nationalist composer Bedrich Smetana was born in 1824 in what's now this republic*"
  - we got improved precision but recall is much worse now

```
GET jeopardy/_search
{
  "query": {
    "match_phrase": {
      "Question": "Czech republic"
    }
  }
}
```

Two things must happen for "**Czech republic**" to cause a hit:

1. "**Czech**" and "**republic**" must appear in the "**Question**" field
2. The terms must appear in that order and next to each other

The multi_match query

PROFINIT

- The **multi_match** query provides a convenient shorthand for running a **match** query against multiple fields
  - by default, Elasticsearch only considers the best scoring field when calculating the _score (**best_fields**)

```
GET jeopardy/_search
{
  "query": {
    "multi_match": {
      "query": "Czech republic",
      "fields": ["Question", "Answer"],
      "type": "best_fields"
    }
  }
}
```

https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-multi-match-query.html

Combining searches

PROFINIT

- Suppose we want to write the following query:
  - **find questions about “beer” in the ”FOOD & DRINK” category**
- This search is actually a combination of two queries:
  - we need “*beer*” in the **Question** field,
  - and “FOOD & DRINK” in the **Category** field
- How can we combine these two queries?
  - by using Boolean logic and the **bool** query…

Bool query

PROFINIT

- Each of the following clauses is possible (but optional) in a **bool** query
  - and they can appear in any order

```
GET jeopardy/_search
{
  "query": {
    "bool": {
      "must": [
        {}
      ],
      "must_not": [
        {}
      ],
      "should": [
        {}
      ],
      "filter": [
        {}
      ]
    }
  }
}
```

Notice the JSON array syntax. You can specify multiple queries in each clause if desired.

https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-bool-query.html

The must clause

PROFINIT

```
GET jeopardy/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "match": {
            "Question": "beer"
          }
        },
        {
          "match": {
            "Category.keyword": "FOOD &
              DRINK"
          }
        }
      ]
    }
  }
}
```

- The clause (query) must appear in matching documents and will contribute to the score
- Let's go back to our search
  - we are looking for questions with "**bear**" that reside in the "**FOOD & DRINK**" category

Other clauses

PROFINIT

- “**must_not**”
  - the clause (query) must not appear in the matching documents
  - scoring is ignored, a score of 0 for all documents is returned
- “**should**”
  - the clause (query) should appear in the matching document
- “**filter**”
  - the clause (query) must appear in matching documents. However unlike **must** the score of the query will be ignored.
- For more information reach out to the official Elastic documentation about the **bool** query
  - https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-bool-query.html

Aggregations

Aggregation types

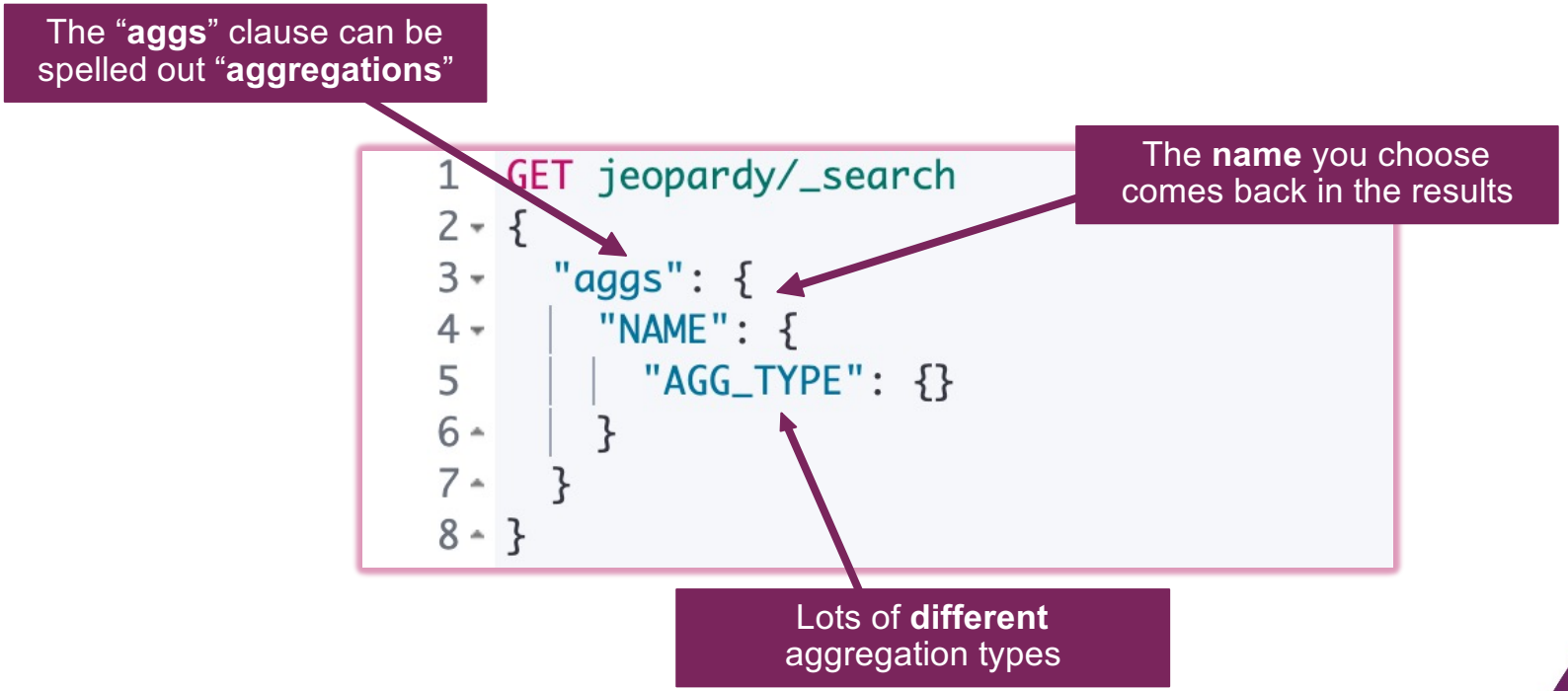
PROFINIT

- Metrics aggregations
  - What is the total amount of prize money in the “Science” category?
- Bucket aggregations
  - What are the top 5 most popular categories?
- Combining aggregations
  - What is the total amount of prize money per each category?
- More about aggregation in the official documentation
  - https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations.html

Aggregation syntax

PROFINIT

- An aggregation request is a part of the Search API
  - with or without a "**query**" clause



Aggregation example. Terms

PROFINIT

› What is the total number of documents per each **Value**?

The “**terms**” aggregation put all the documents into buckets defined by a field value

```
GET jeopardy/_search
{
  "aggs": {
    "num_questions_per_value": {
      "terms": {
        "field": "Value"
      }
    }
  }
}
```

In this agg the number of documents are collected per each unique **Value**

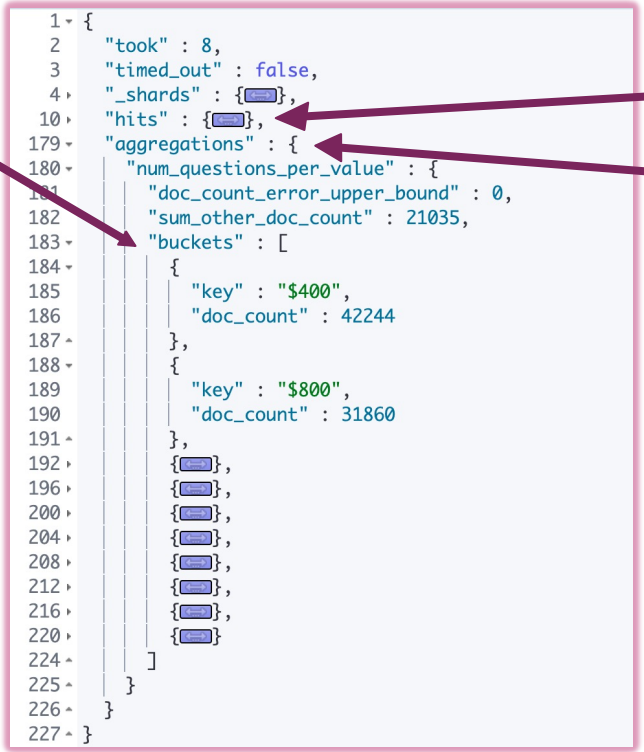
https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-bucket-terms-aggregation.html

Aggregation results

PROFINIT

› The response has an “**aggregations**” section that contains the results of all the “**aggs**” in the search request

A list of top 10 buckets. One bucket corresponds to a one unique **Value** with the total number of documents in it



We get the top 10 hits …

… and an “aggregations” section in the response

The scope of an aggregation

PROFINIT

- You can add a **query** clause to an aggregation to limit the scope

What are the top 3 categories with questions about the Czech republic

```
GET jeopardy/_search
{
  "size": 0,
  "query": {
    "match": {
      "Question": "Czech republic"
    }
  },
  "aggs": {
    "top_categories": {
      "terms": {
        "field": "Category.keyword",
        "size": 3
      }
    }
  }
}
```


```
{
  "took" : 1,
  "timed_out" : false,
  "_shards" : {...},
  "hits" : {...},
  "aggregations" : {
    "top_categories" : {
      "doc_count_error_upper_bound" : 0,
      "sum_other_doc_count" : 462,
      "buckets" : [
        {
          "key" : "WORLD CAPITALS",
          "doc_count" : 7
        },
        {
          "key" : "\"C\"OUNTRIES",
          "doc_count" : 4
        },
        {
          "key" : "CANCELED CZECHS",
          "doc_count" : 4
        }
      ]
    }
  }
}
```


buckets field consists of category names and number of documents in it that satisfy the query

Mapping

What is Mapping?

PROFINIT

- Elasticsearch will happily index any document without knowing its details (number of fields, their data types, etc.)
  - however, behind-the-scenes Elasticsearch assigns data types to your fields in a ***mapping***
- A **mapping** is a **schema definition** that contains:
  - name of fields
  - data types of fields
  - how the fields should be indexed and stored by Lucene
- Mappings map complex JSON documents into the simple flat documents that Lucene expects

Remember the odd “Category.keyword”?

PROFINIT

- The one on the slide number 44
- Let’s have a closer look where does it come from

```
    "num_questions_per_value": {
      "terms": {
        "field": "Category.keyword"
      }
    }
  }
}
```

```
GET jeopardy/_mapping
```


```
{
  "jeopardy" : {
    "mappings" : {
      "_meta" : {},
      "properties" : {
        "@timestamp" : {},
        "Air Date" : {
          "type" : "date",
          "format" : "iso8601"
        },
        "Answer" : {
          "type" : "text"
        },
        "Category" : {
          "type" : "text",
          "fields" : {
            "keyword" : {
              "type" : "keyword",
              "ignore_above" : 256
            }
          }
        },
        "Question" : {
          "type" : "text"
        },
        "Round" : {},
        "Show Number" : {},
        "Value" : {}
      }
    }
  }
}
```

Define a mapping

PROFINIT

- In many use cases, you will need to define your own mappings
- Mappings are defined in the “**mappings**” section of an index:
  - you can define mappings at index creation:

```
PUT jeopardy
{
  "mappings": {
    define your mapping here
  }
}
```

  - or, add to a mapping of an existing index:

```
PUT jeopardy/_mapping
{
  additional mapping here
}
```

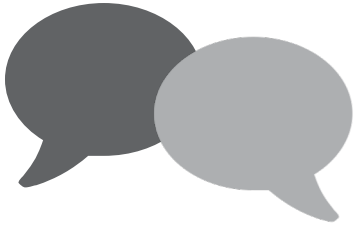
- More about mapping can be found in the documentation
  - https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping.html

Links

PROFINIT

- Elasticsearch documentation
  - https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html
- Kibana documentation
  - https://www.elastic.co/guide/en/kibana/current/index.html
- Information provided on slides was inspired by materials from Elasticsearch Engineer training
  - https://www.elastic.co/training/elasticsearch-engineer

PROFINIT



Questions

Thank you

PROFINIT

Profinit EU, s.r.o., Tychonova 2, 160 00 Praha 6
Tel.: + 420 224 316 016, web: www.profinit.eu

LinkedIn
linkedin.com/company/profinit

Twitter
twitter.com/Profinit_EU

Facebook
facebook.com/Profinit.EU

Youtube
Profinit EU