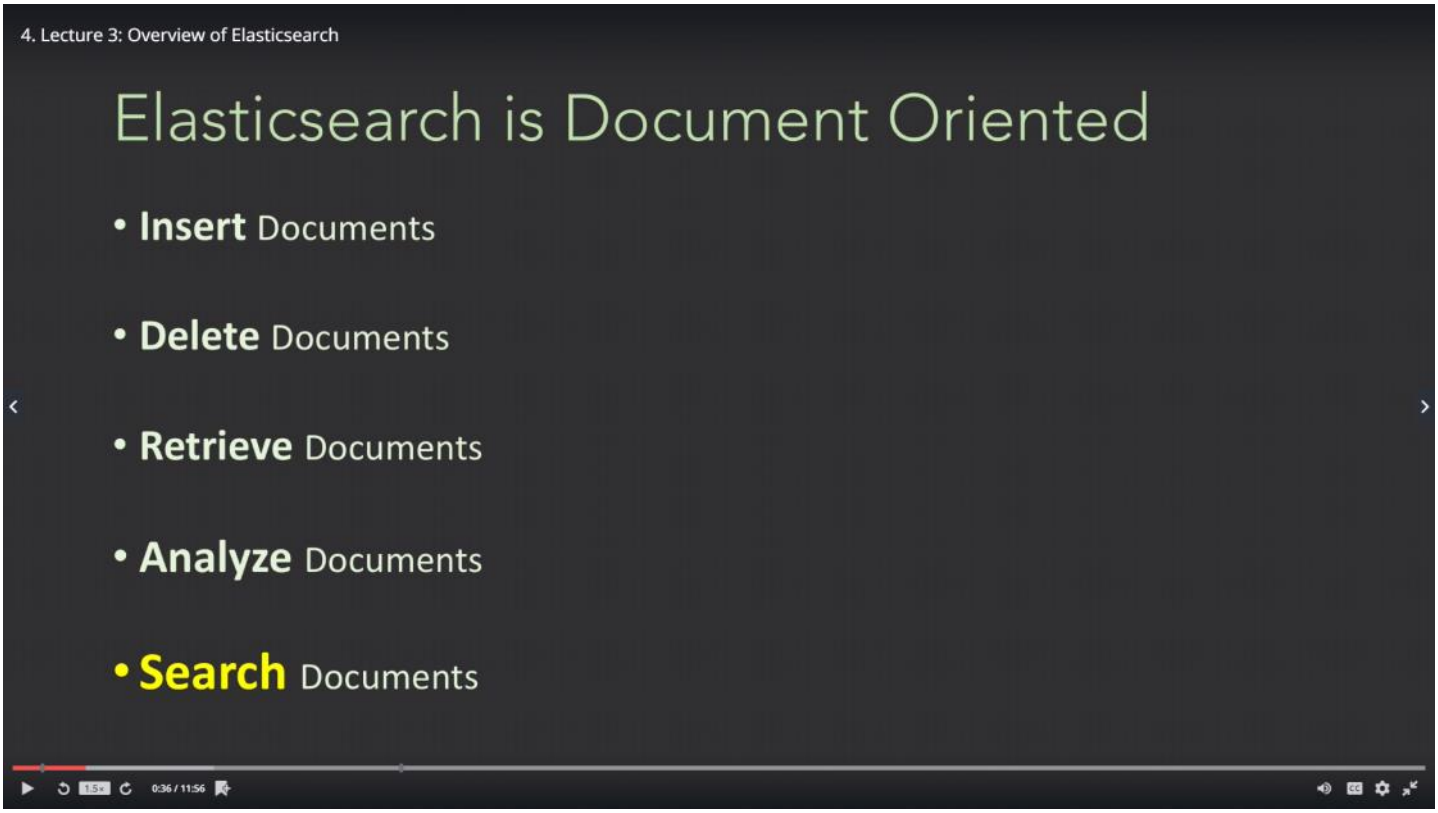
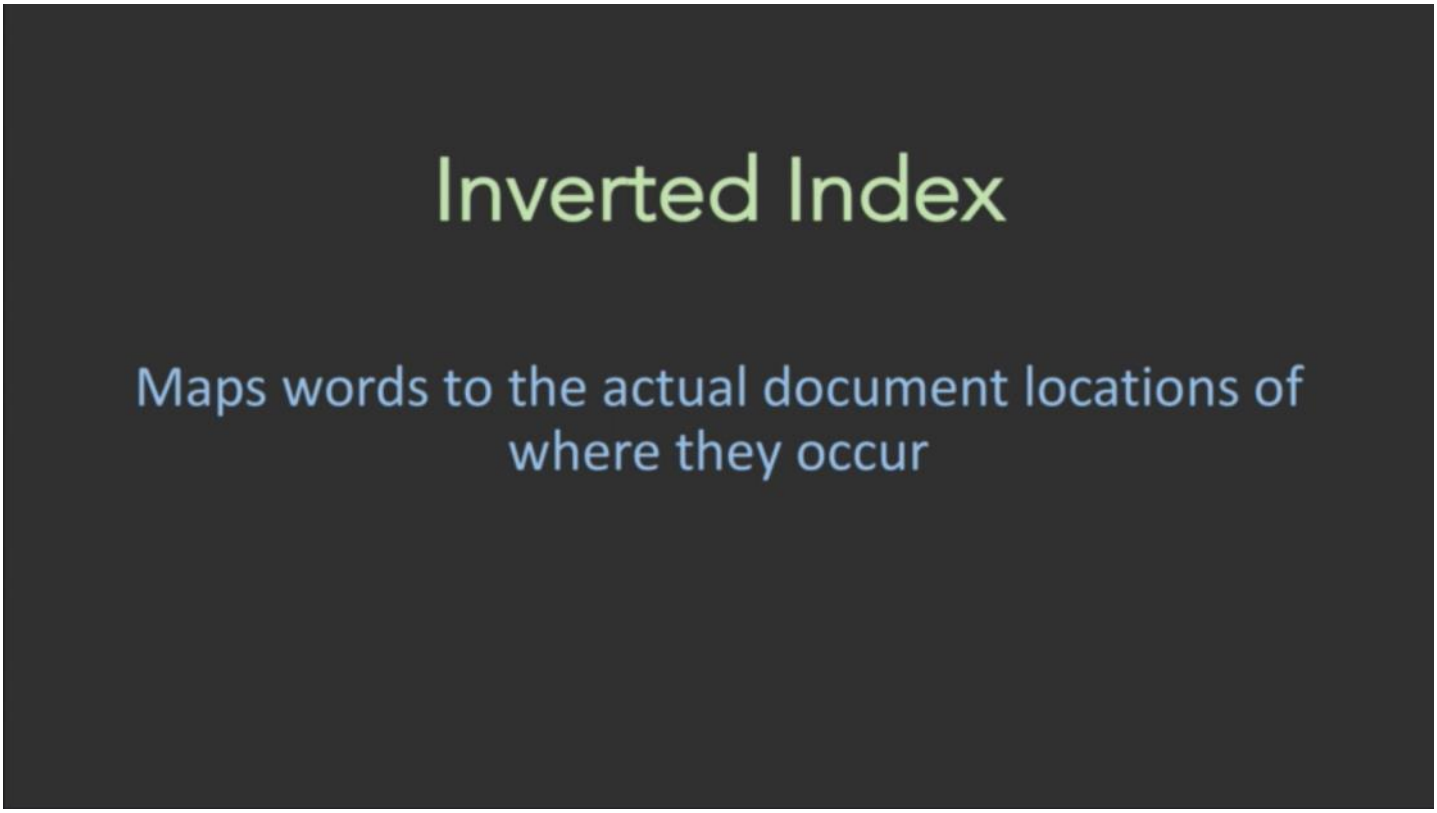
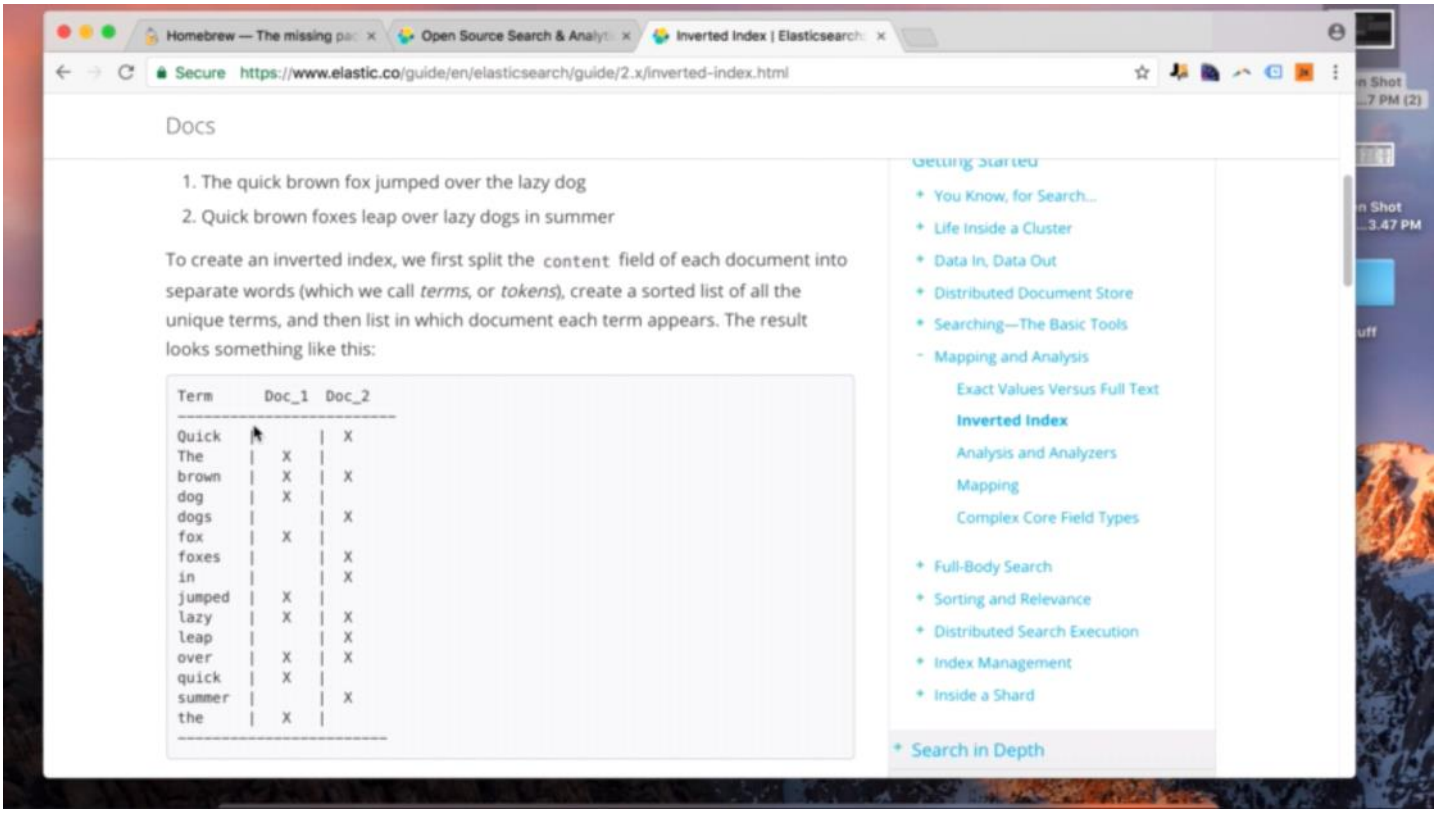
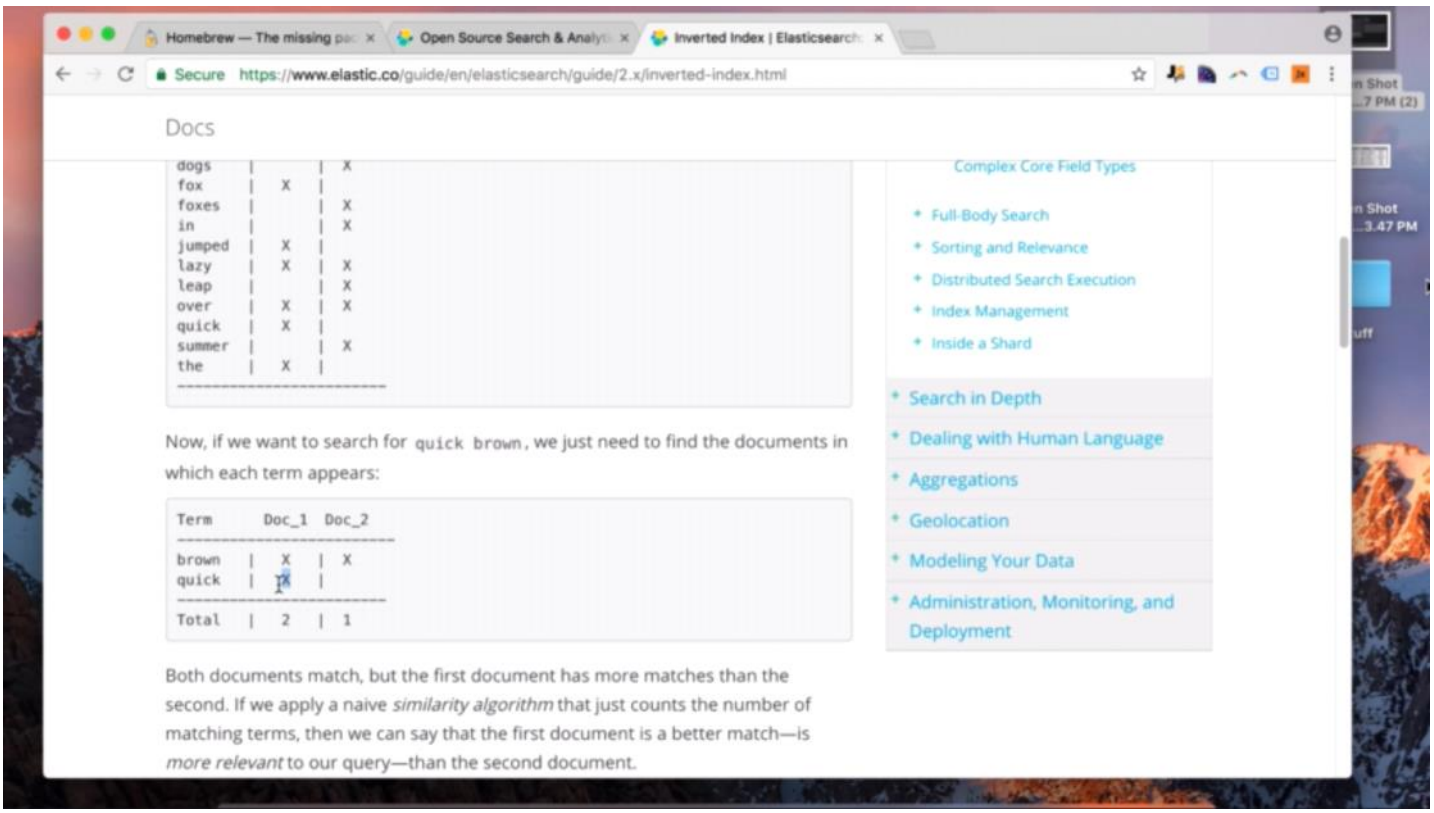
Lecture 3: Overview of Elasticsearch

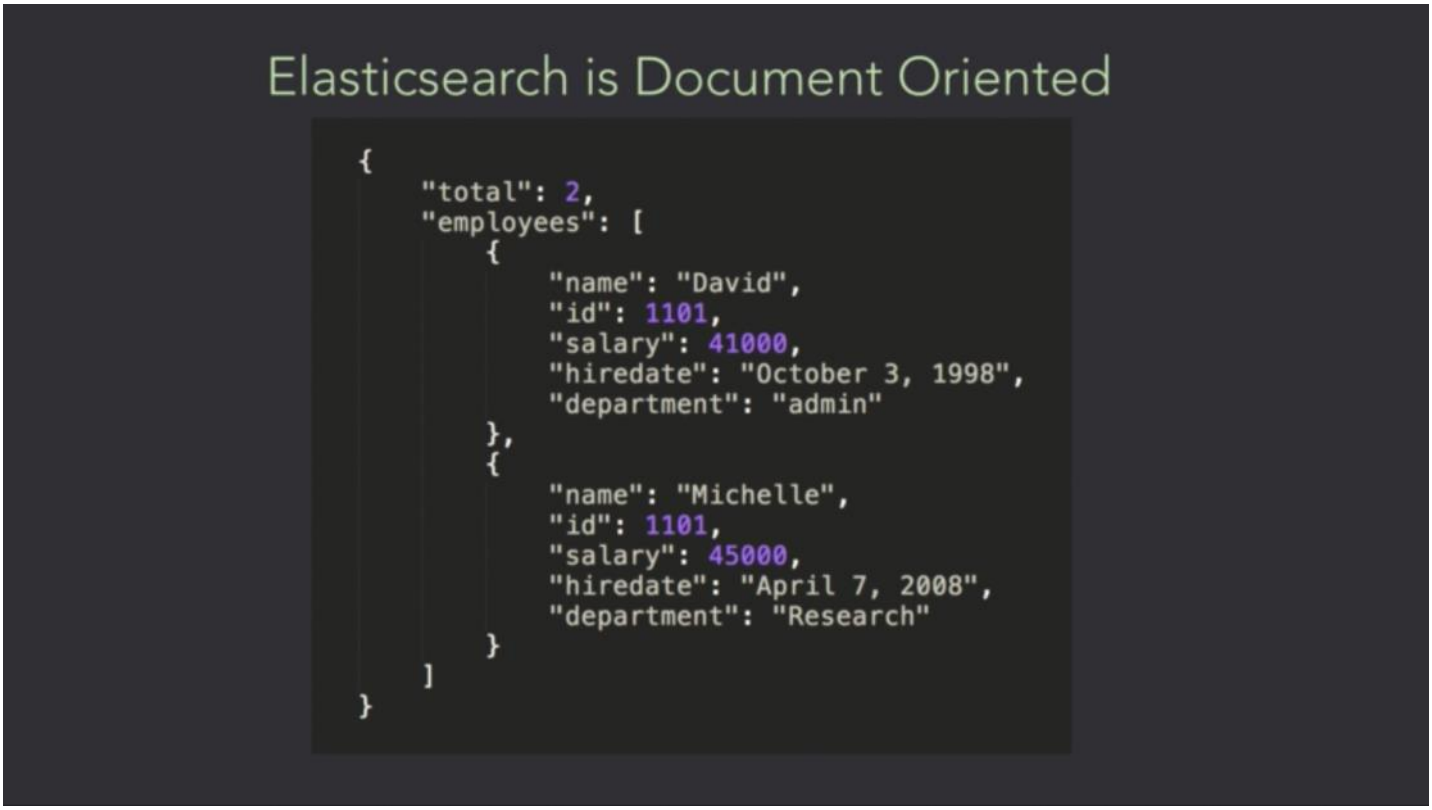
Friday, June 7, 2019 11:43 AM

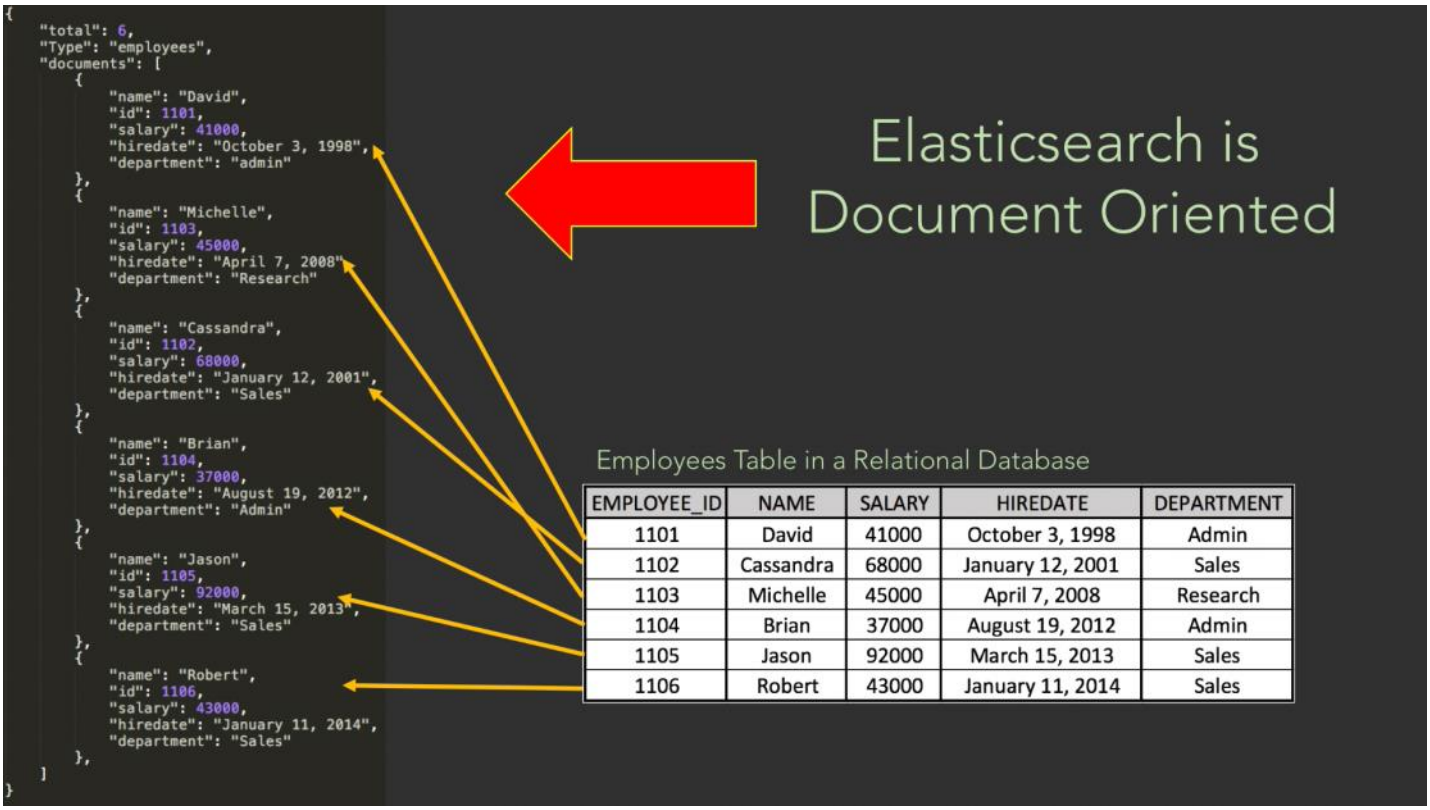





Homebrew — The missing pac
Open Source Search & Analyt
Inverted Index | Elasticsearch
Secure https://www.elastic.co/guide/en/elasticsearch/guide/2.x/inverted-index.html
Docs
1. The quick brown fox jumped over the lazy dog
2. Quick brown foxes leap over lazy dogs in summer
To create an inverted index, we first split the content field of each document into separate words (which we call terms, or tokens), create a sorted list of all the unique terms, and then list in which document each term appears. The result looks something like this:
Term Doc_1 Doc_2
Quick | | X
The | X | |
brown | X | X
dog | X | |
dogs | | X
fox | X | |
foxes | | X
in | | X
jumped | X | |
lazy | X | X
leap | | X
over | X | X
quick | X | |
summer | | X
the | X | |
Getting Started
You Know, for Search...
Life Inside a Cluster
Data In, Data Out
Distributed Document Store
Searching—The Basic Tools
Mapping and Analysis
Exact Values Versus Full Text
Inverted Index
Analysis and Analyzers
Mapping
Complex Core Field Types
Full-Body Search
Sorting and Relevance
Distributed Search Execution
Index Management
Inside a Shard
Search in Depth


Homebrew — The missing pac
Open Source Search & Analyt
Inverted Index | Elasticsearch
Secure https://www.elastic.co/guide/en/elasticsearch/guide/2.x/inverted-index.html
Docs
dogs | | X
fox | X | |
foxes | | X
in | | X
jumped | X | |
lazy | X | X
leap | | X
over | X | X
quick | X | |
summer | | X
the | X | |
Now, if we want to search for quick brown, we just need to find the documents in which each term appears:
Term Doc_1 Doc_2
brown | X | X
quick | X | |
Total | 2 | 1
Both documents match, but the first document has more matches than the second. If we apply a naive similarity algorithm that just counts the number of matching terms, then we can say that the first document is a better match—is more relevant to our query—than the second document.
Complex Core Field Types
Full-Body Search
Sorting and Relevance
Distributed Search Execution
Index Management
Inside a Shard
Search in Depth
Dealing with Human Language
Aggregations
Geolocation
Modeling Your Data
Administration, Monitoring, and Deployment

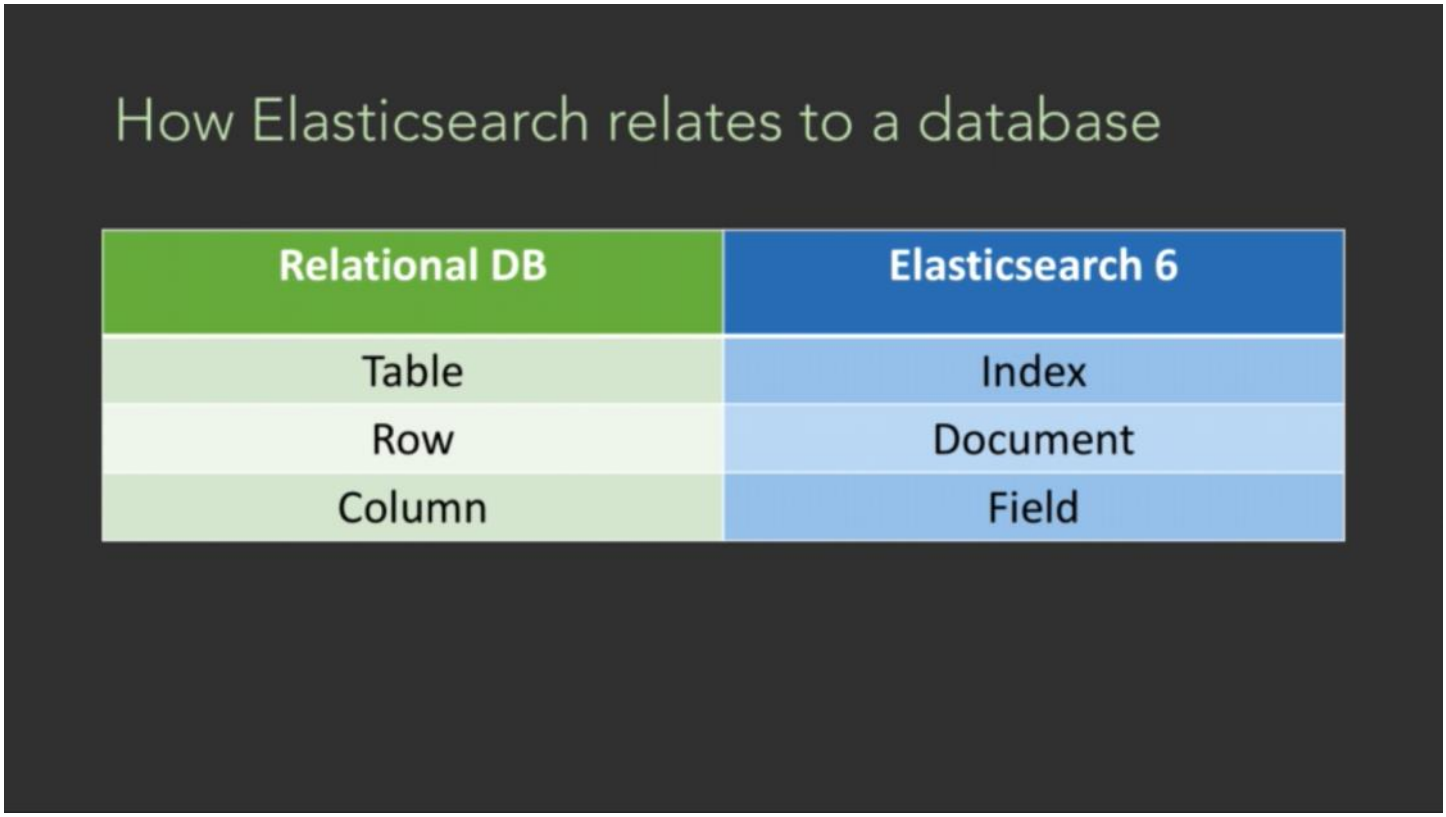


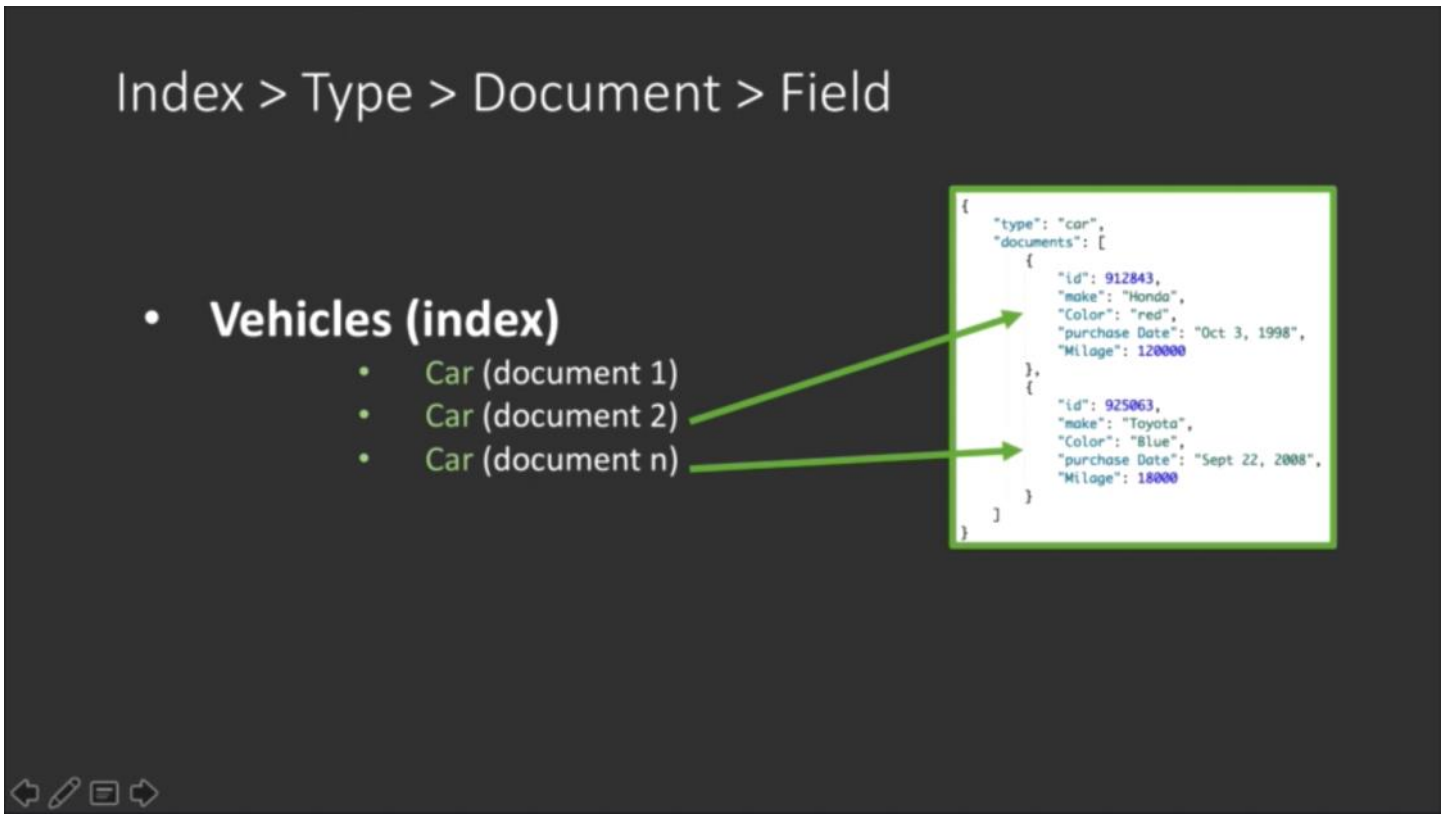


- https://www.elastic.co/guide/en/elasticsearch/guide/current/document.html

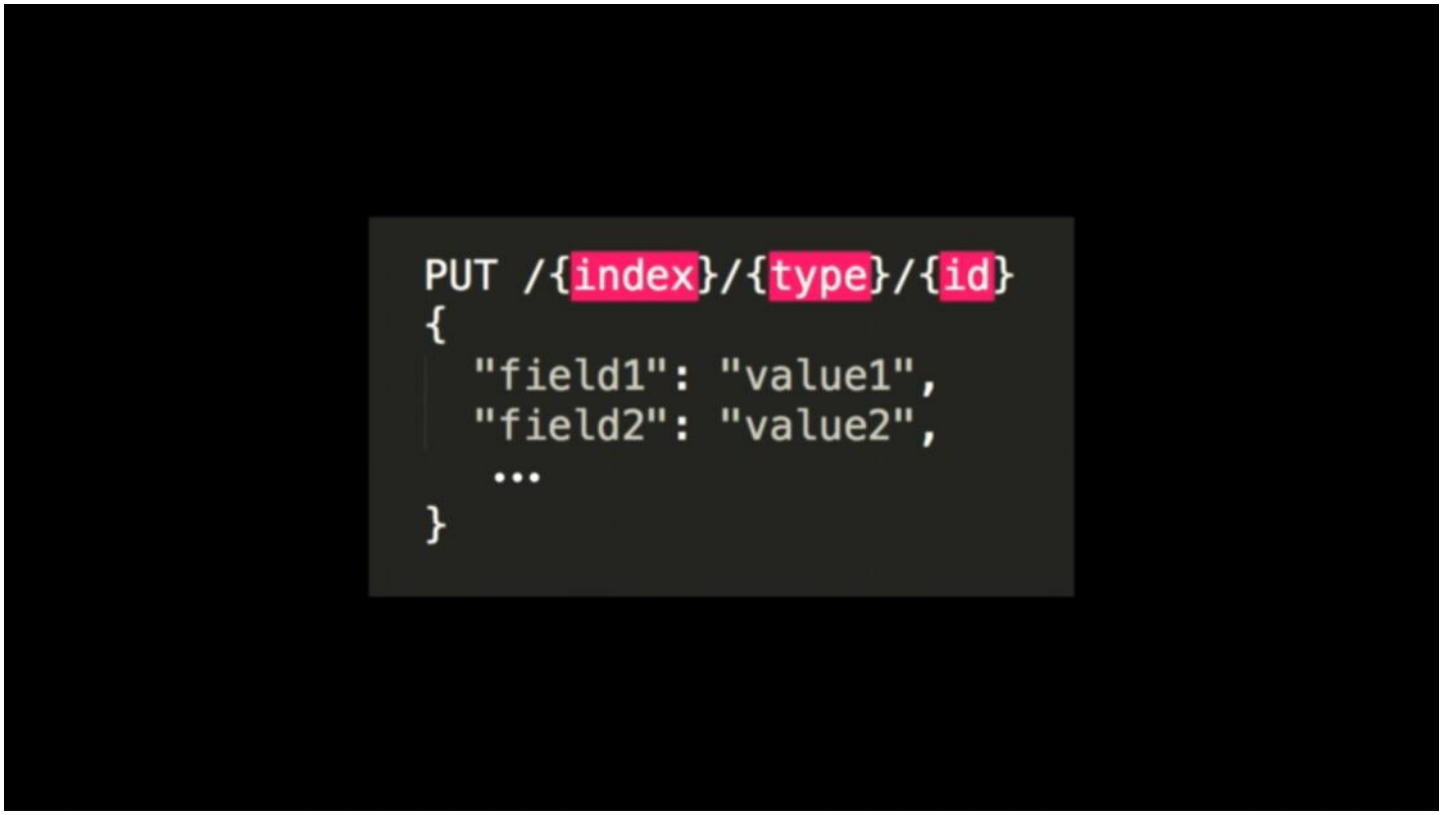
Lecture 4: Indexing, Retrieving, and Deleting Documents

Friday, June 7, 2019 2:04 PM

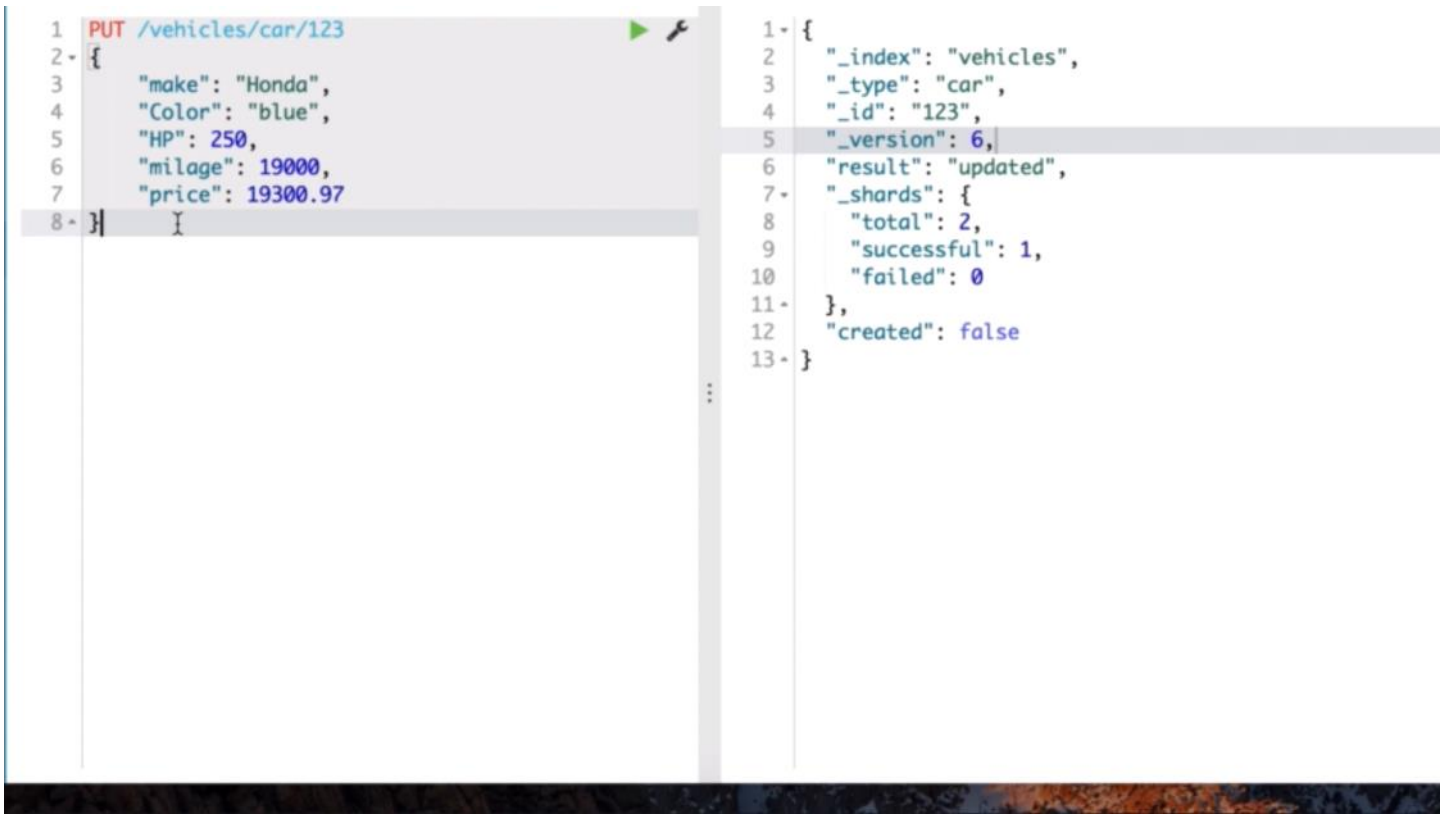




- **Inserting = Indexing**



- **_version is incremented via overwriting the existing document**
- **anything with an "_" is considered a meta element**
- **frequent updates, additions, and deprecations**
- **documents are immutable**
- **PUT, POST, GET, HEAD, DELETE**



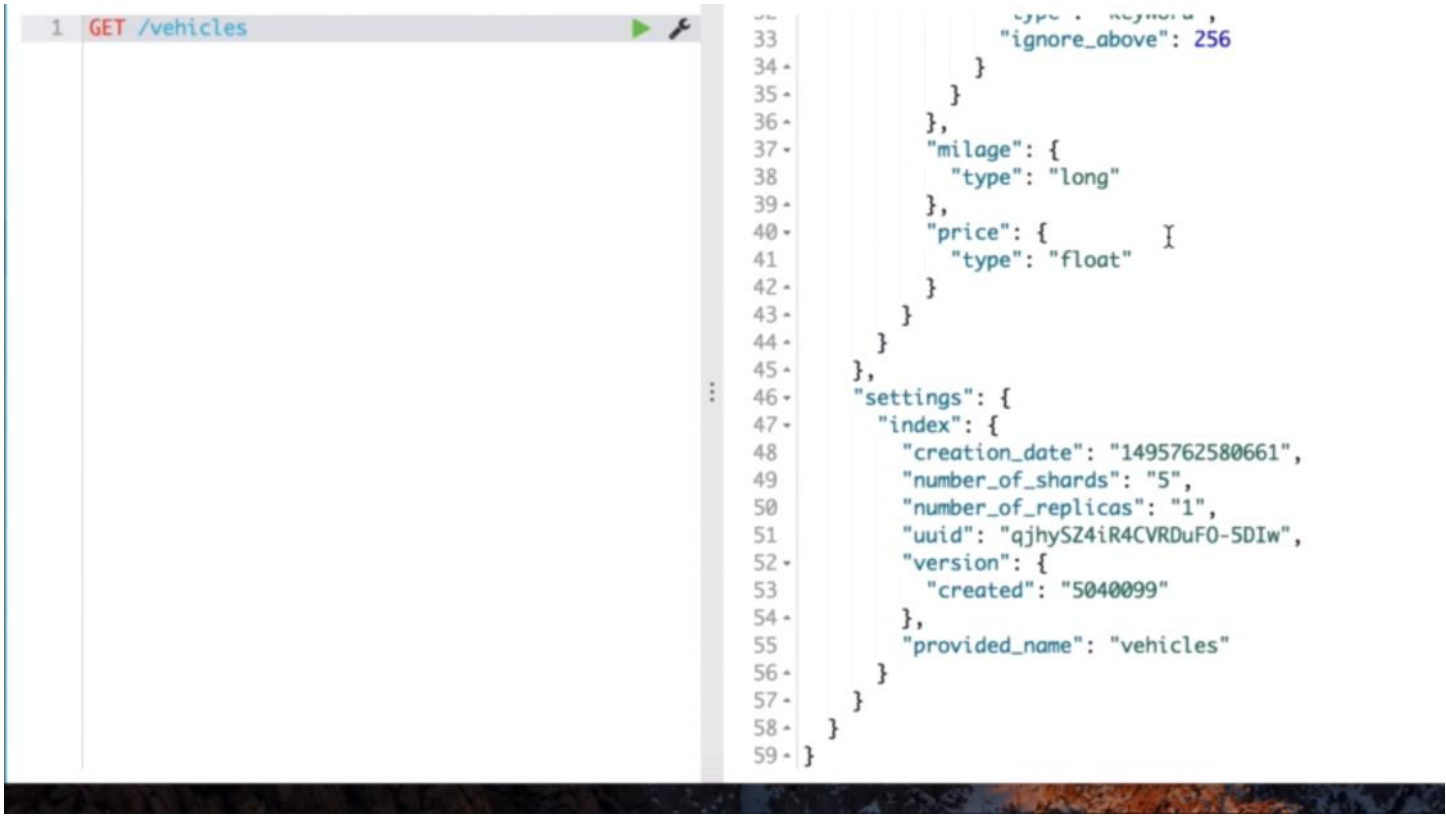
```
POST /vehicles/car/123/_update
{
  "doc": {
    "driver": "Tom"
  }
}
```

```
{
  "_index": "vehicles",
  "_type": "car",
  "_id": "123",
  "_version": 8,
  "result": "updated",
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  }
}
```

```
DELETE /vehicles/car/123
```

```
{
  "found": true,
  "_index": "vehicles",
  "_type": "car",
  "_id": "123",
  "_version": 9,
  "result": "deleted",
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  }
}
```

- **when you DELETE a document, you can no longer access it via GET, but it still exists**
  - **Elasticsearch marks those documents as DELETED, and every so often behind the scenes, it collects all documents marked as DELETED and wipes them off the disk**

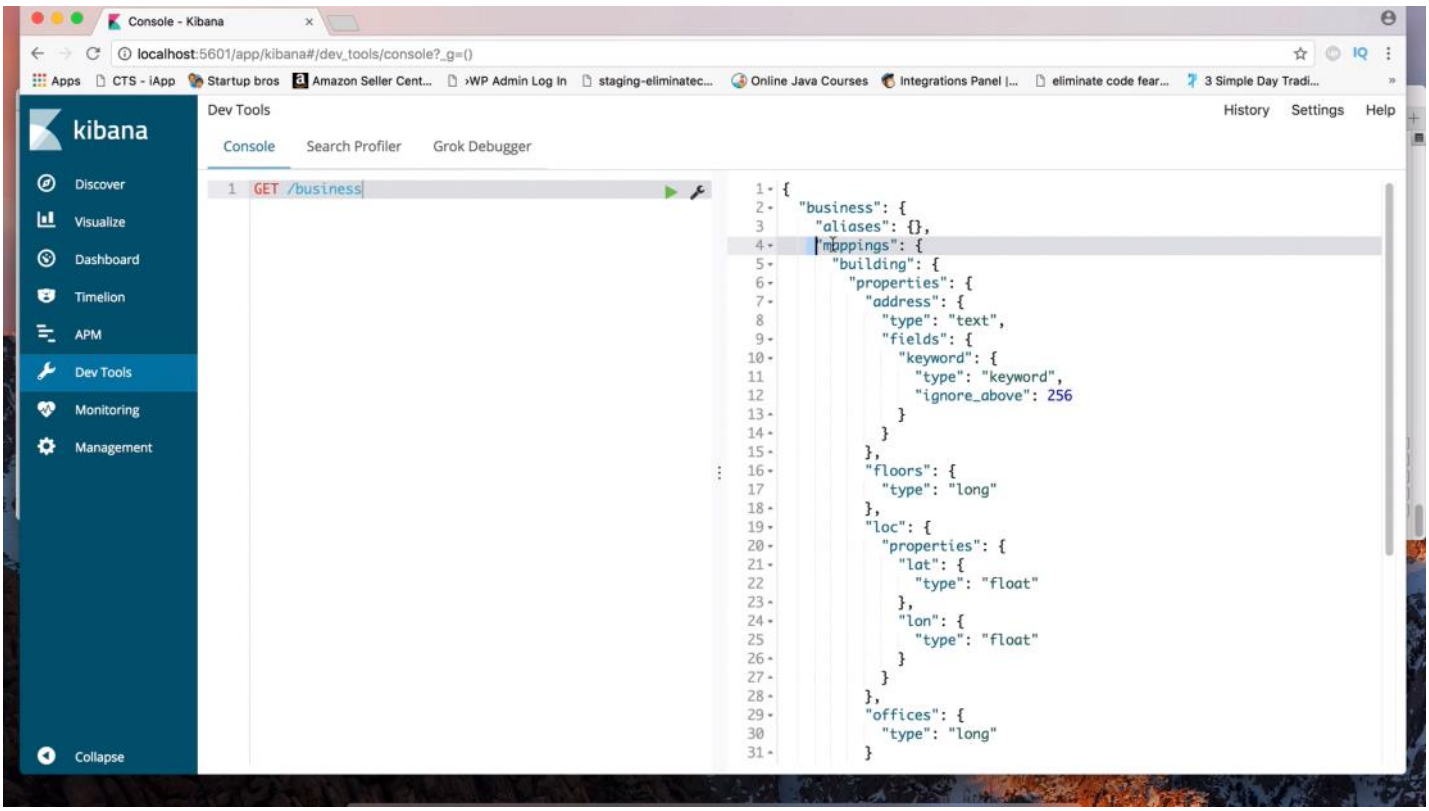


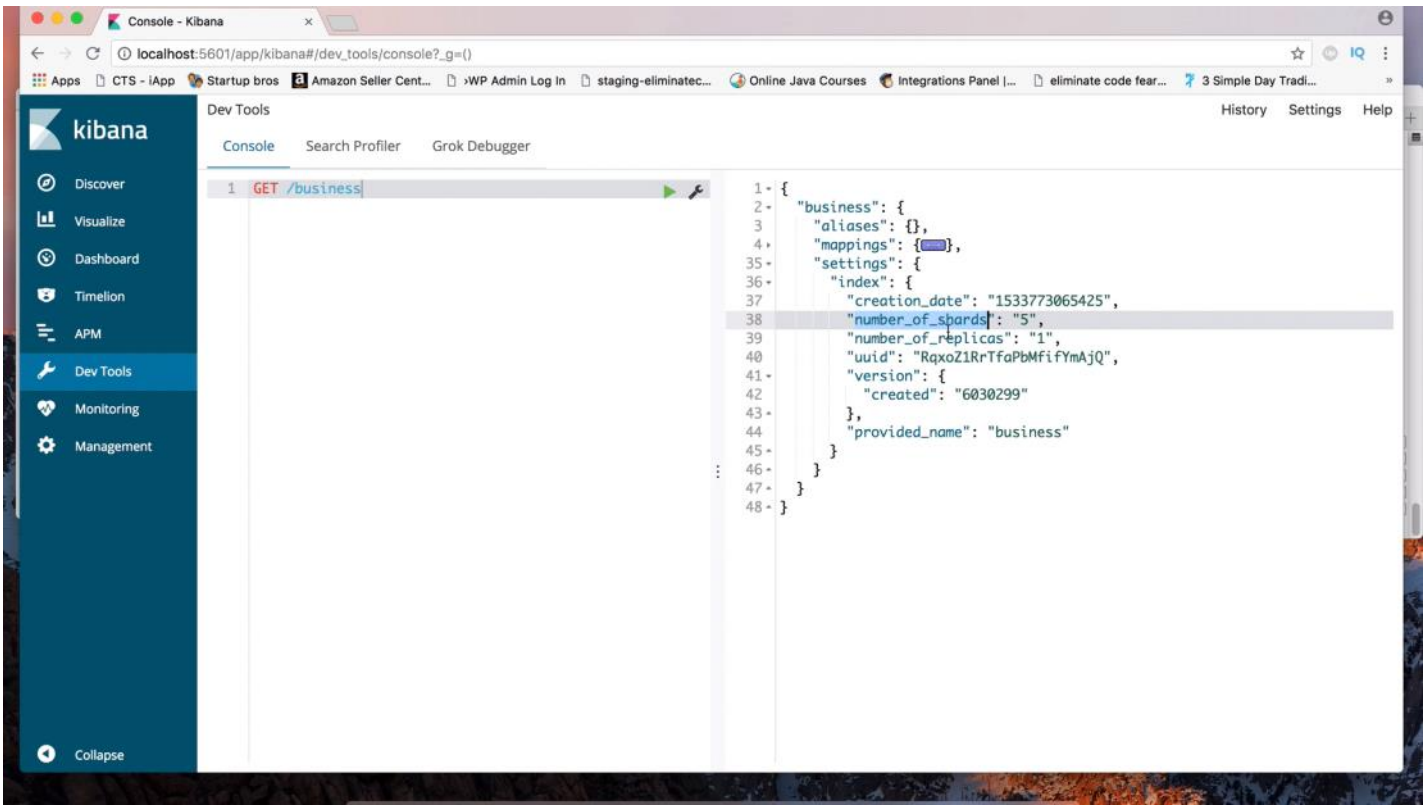
- **doing a GET command on an index will provide the structure of that index, auto-generated via Elasticsearch**
- ***make sure to look out for ES/Kibana updates in interfacing!***

Lecture 5: Components of an Index

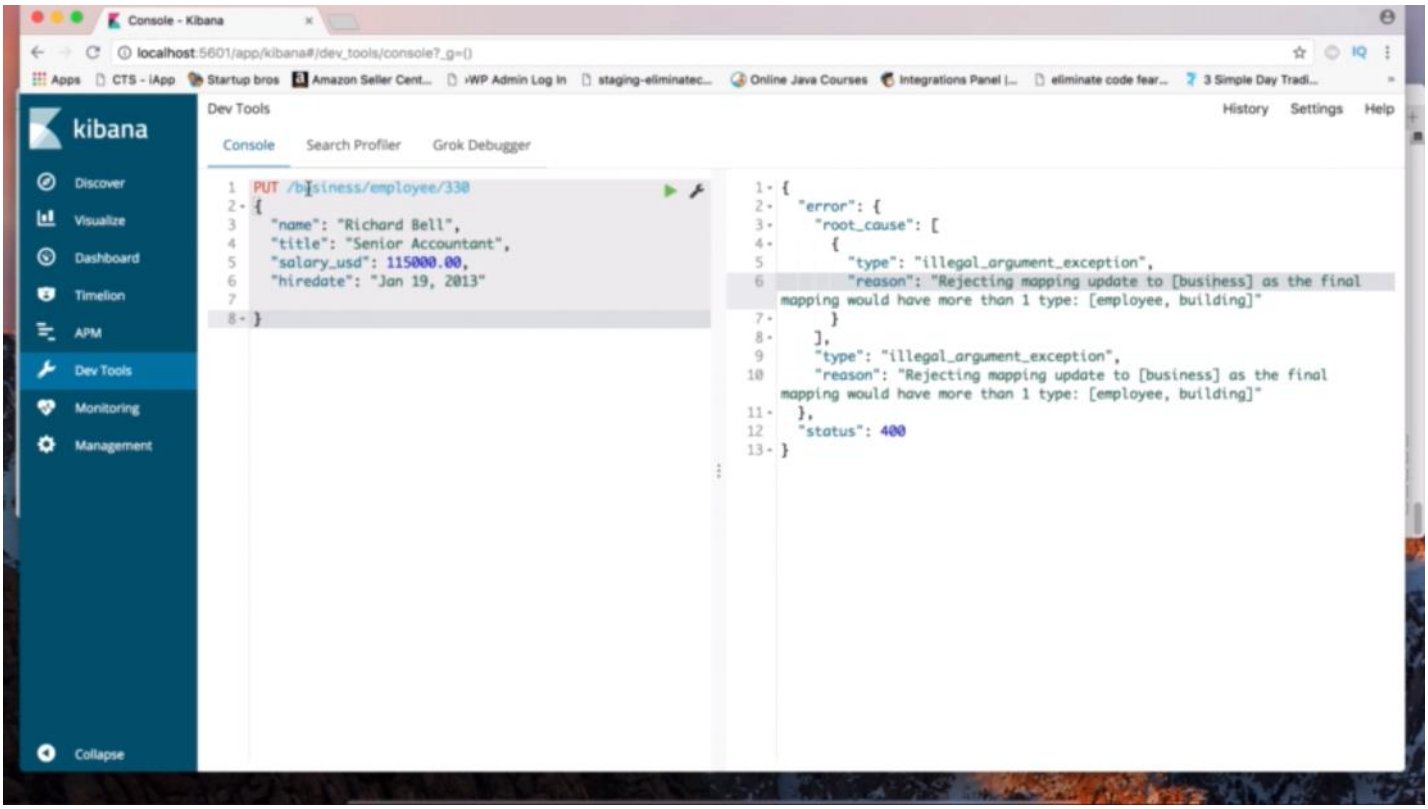
Friday, June 7, 2019 3:52 PM

- **Index name, aliases, mappings (structure of fields w/properties [auto-generated]), settings (shards, replicas, IDs, etc.)**

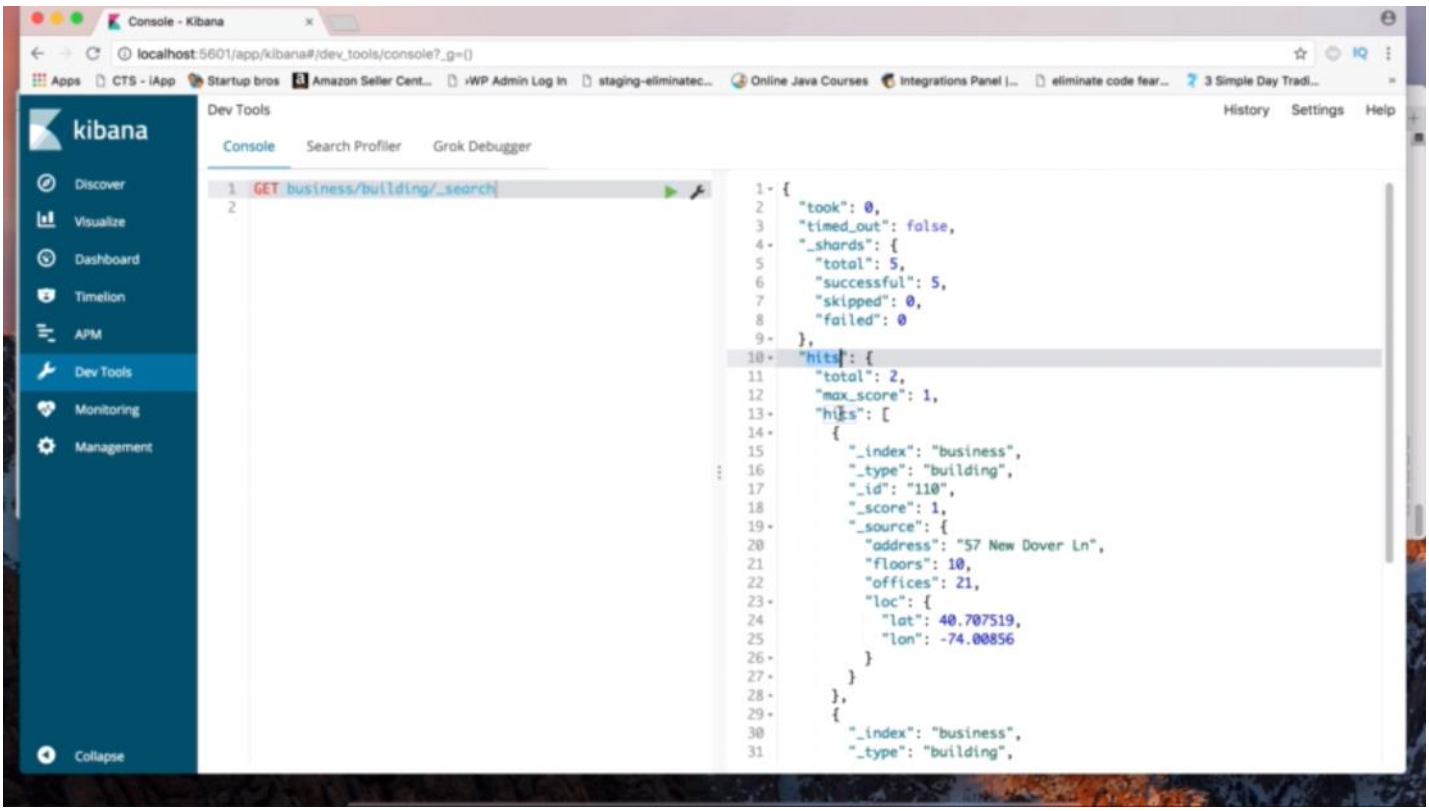




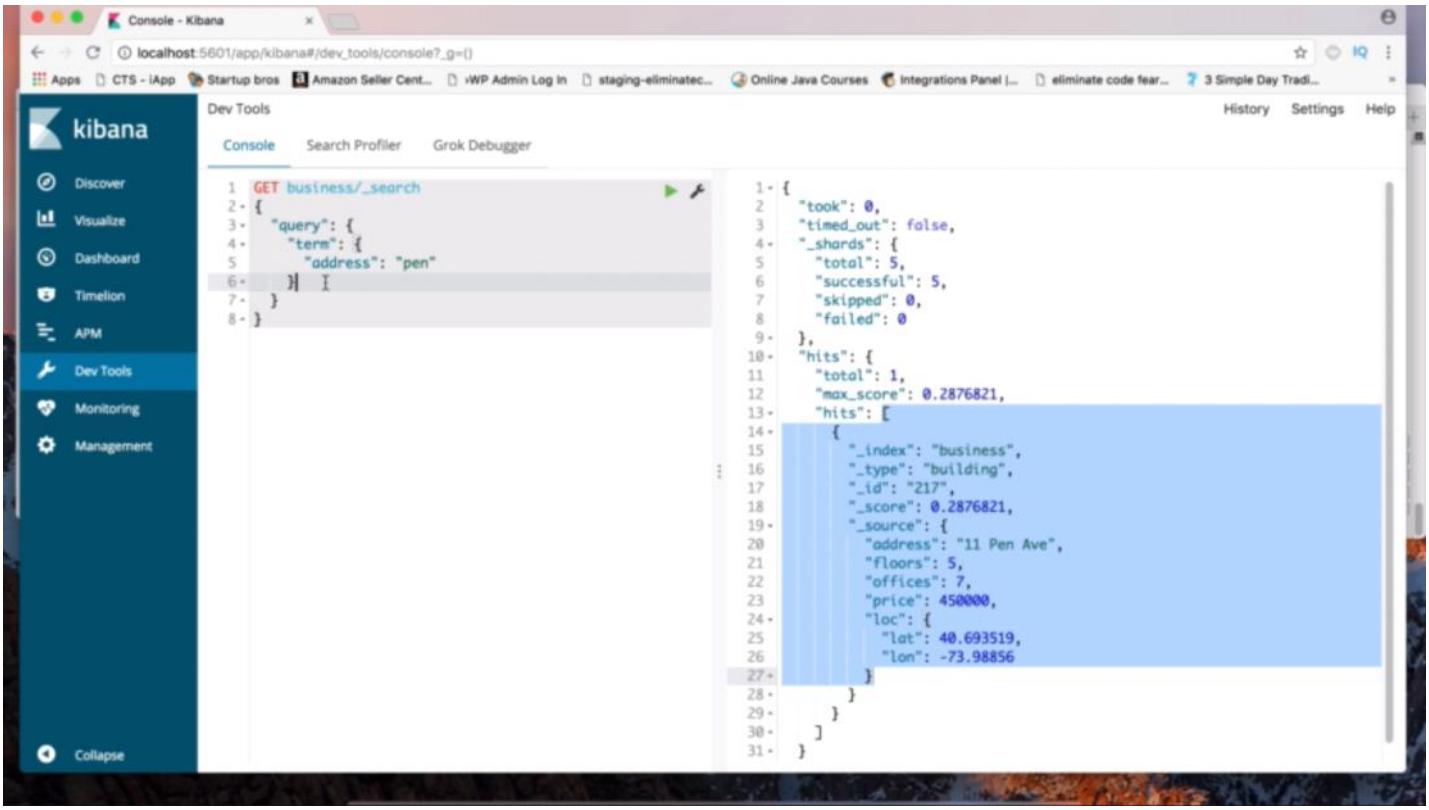
- **ES able to dynamically change index structure to accommodate new fields**



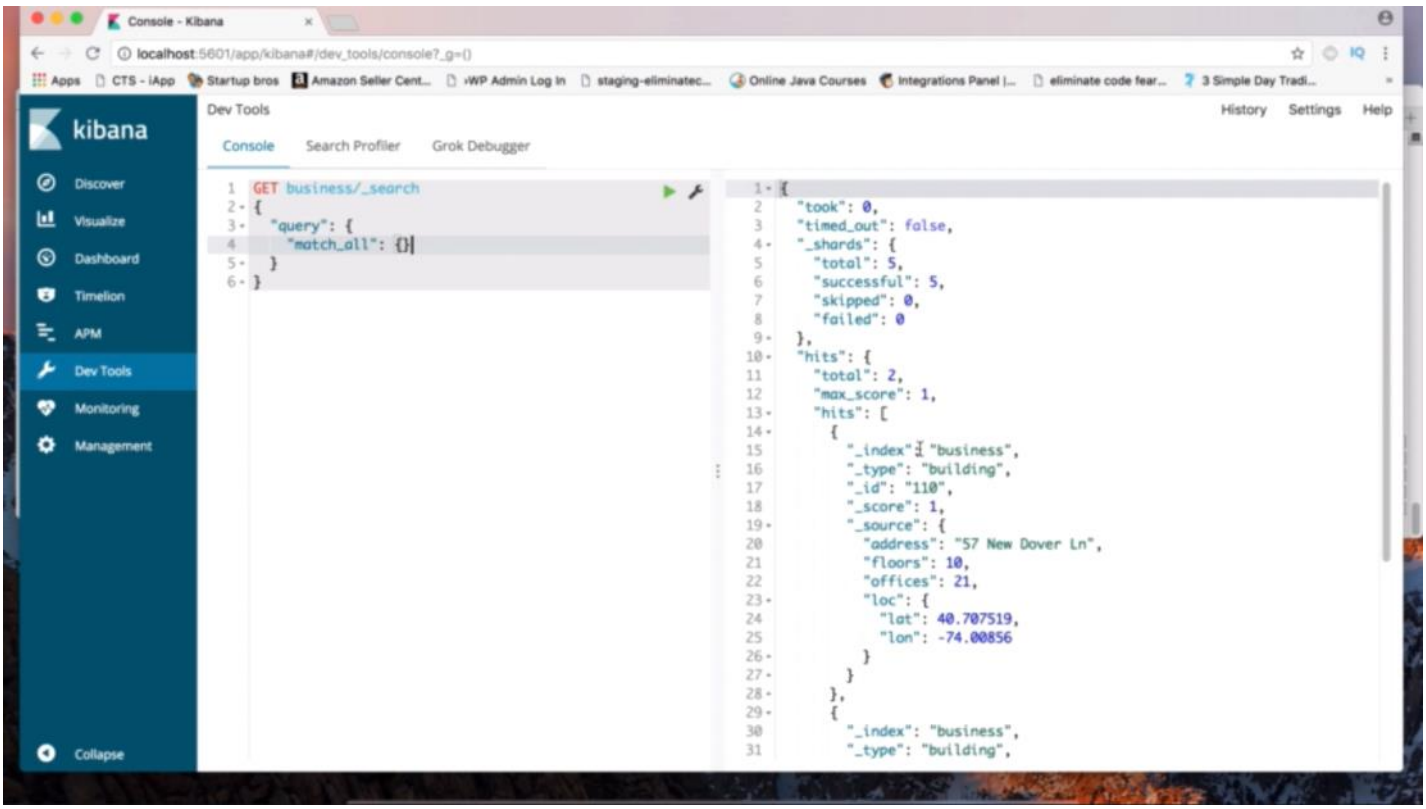
- **For ES6, multi-mapping to a specific index is NOT allowed**
- **Must make a new index to map a new type of object**



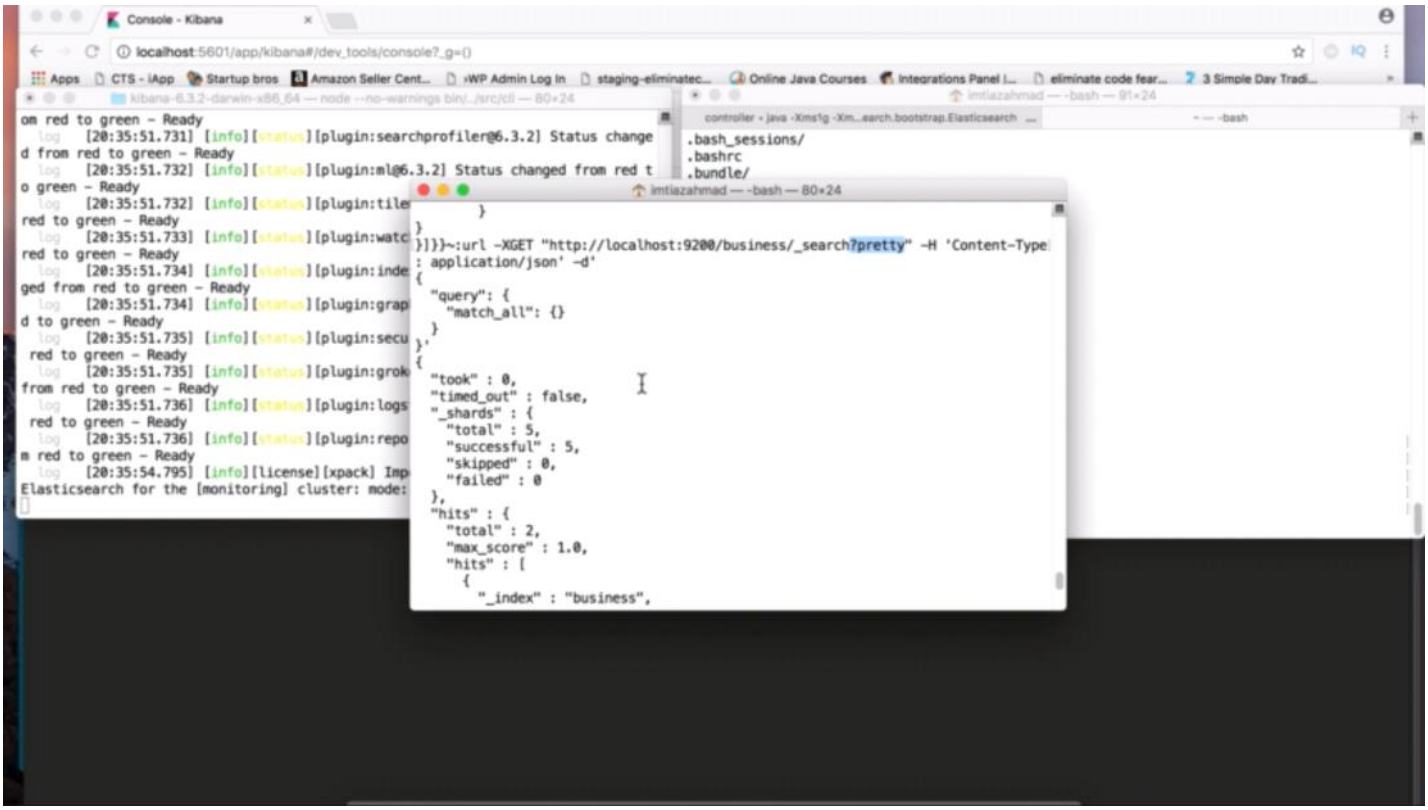
- **HITS array contains all documents that are indexed**



- **The ABOVE code is known as a <u>term query</u>**
- **Here, the query specifies a certain term/property to search for, and matches the string provided with the ones in the indices**



- **The ABOVE is known as a <u>match_all query</u>, which provides all the documents in the index (similar to just searching)**



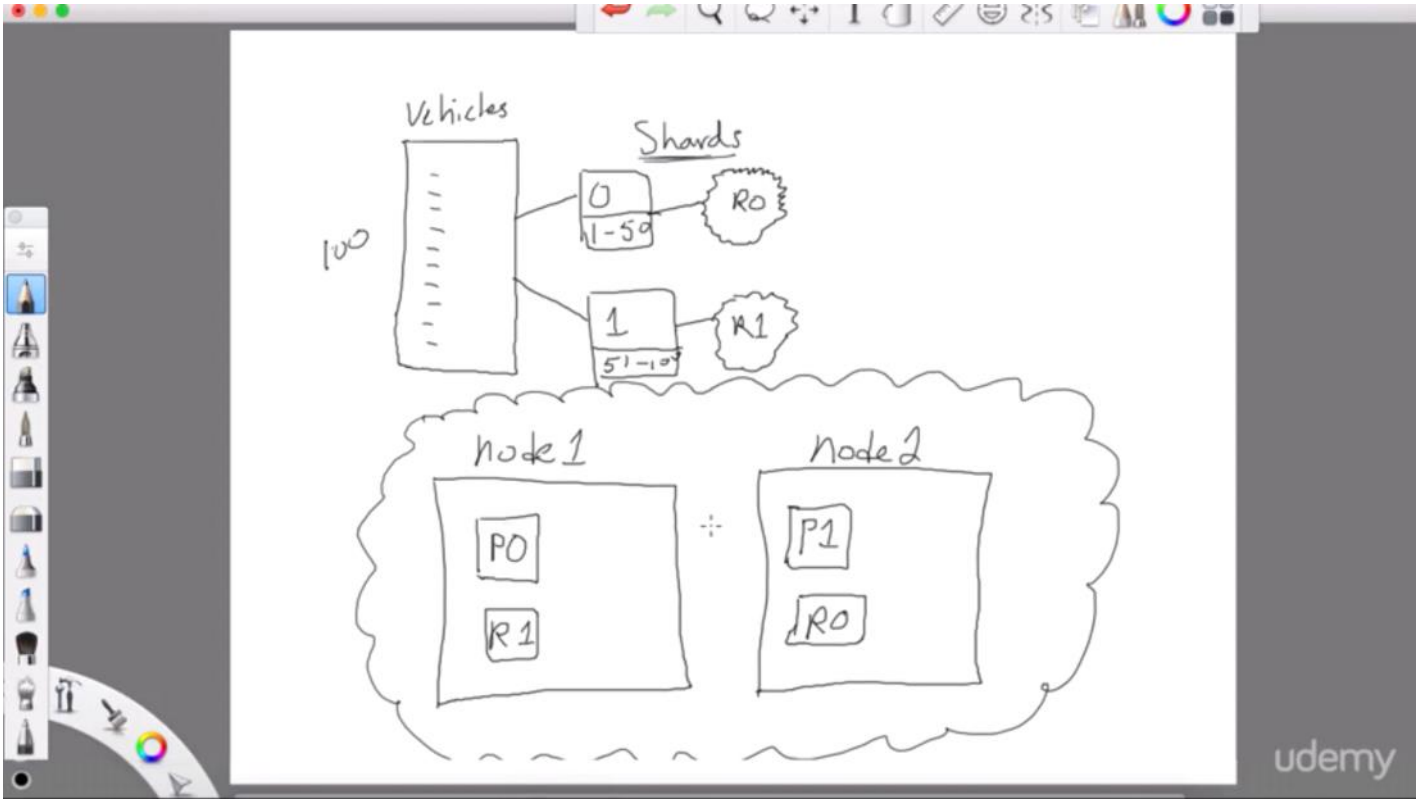
- **Go to wrench icon, click "Copy as cURL"**
- **Paste in terminal to retrieve same index and documentation**
- **Add "?pretty" to the end of the request to format the response**

Lecture 6: Distributed Execution of Requests

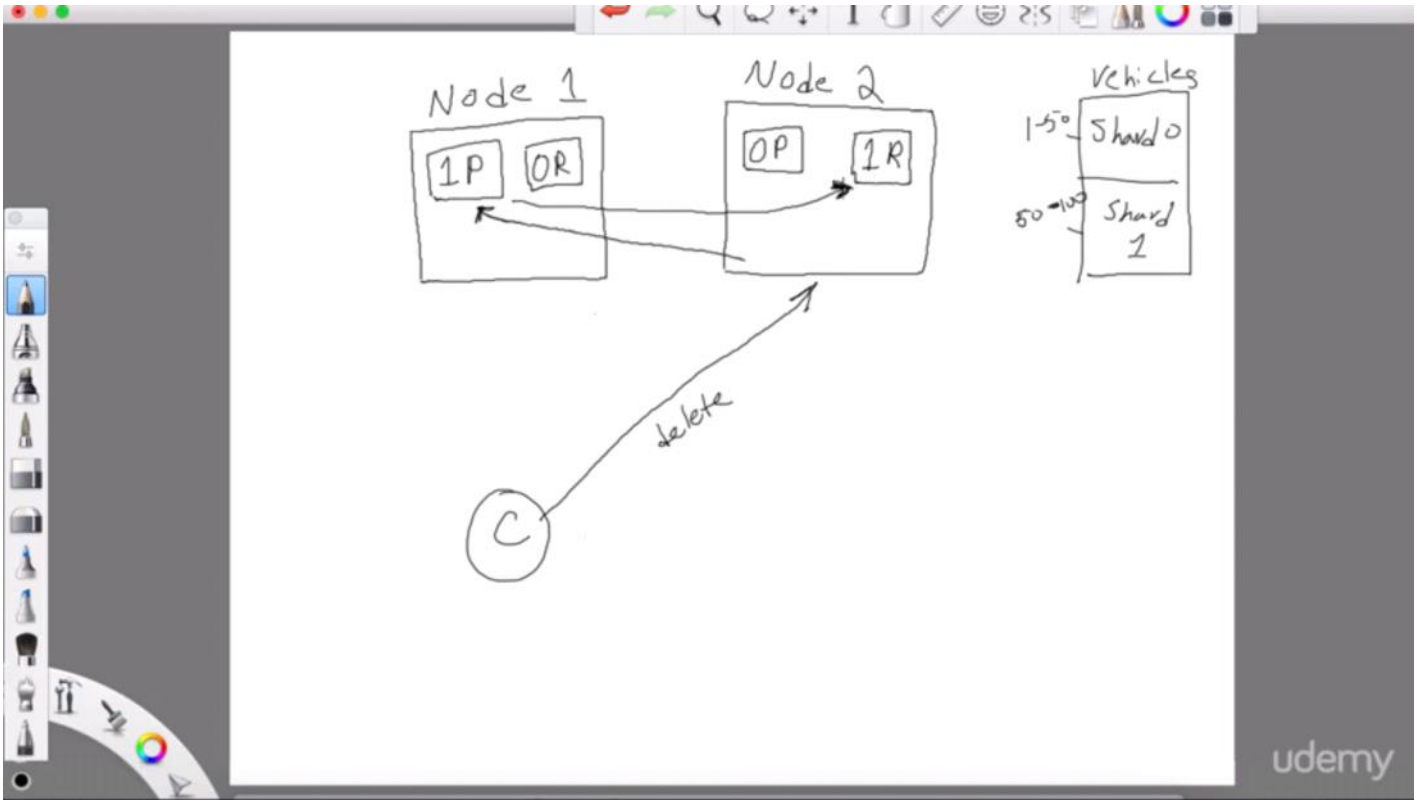
Tuesday, June 11, 2019 11:21 AM

- **ES is a distributed technology, meaning that devices on the same network running ES can fluidly communicate with each other, resulting in a cluster**
- **A cluster can efficiently distribute large search problems to various nodes, reducing runtime and increasing throughput**

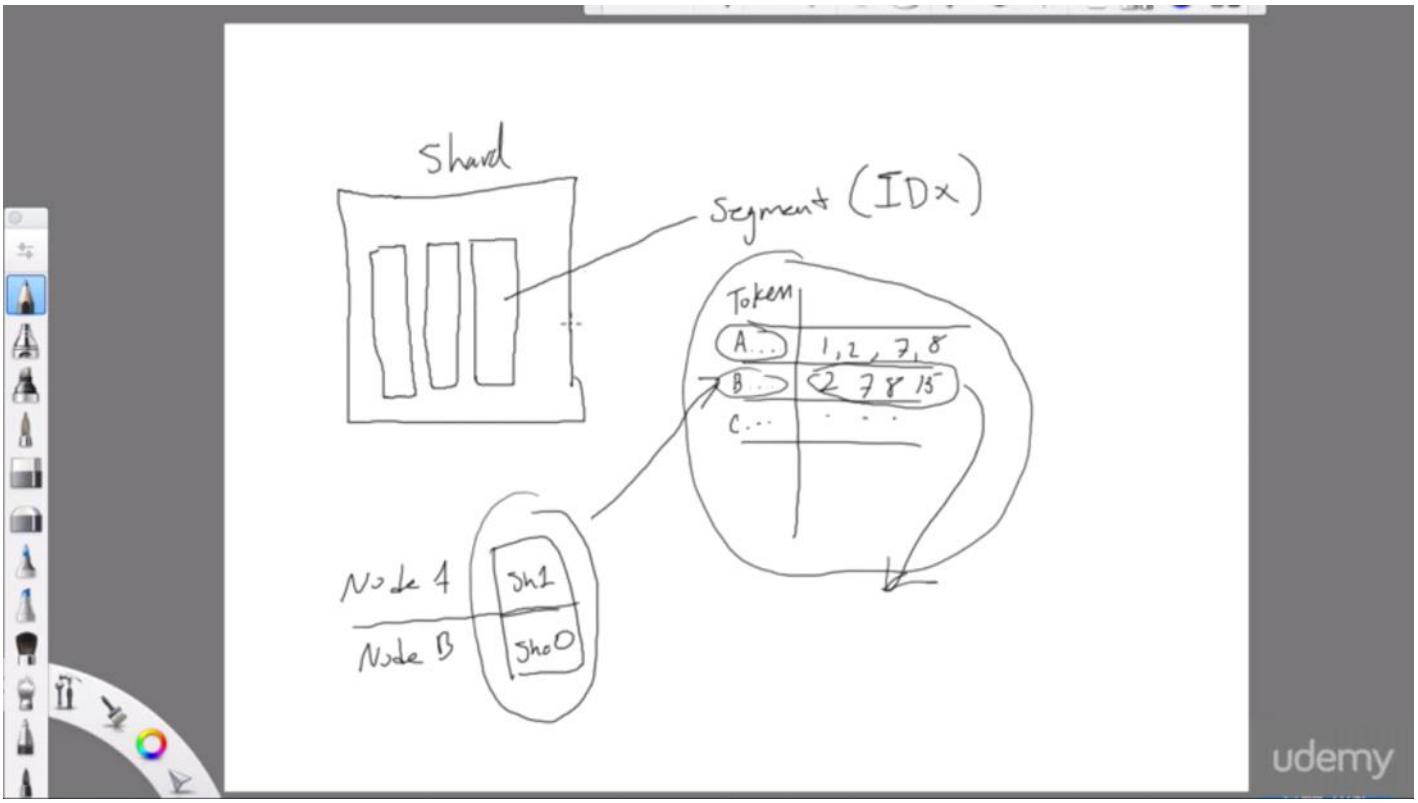
- **Physical representation of an index is a shard**
- **Shards contain data from multiple documents (via docIDs)**
- **"P" represents a primary shard (P0, P1)**
- **Also have replica shards, which contains same exact documents as the ones in the shards (R0, R1)**
- **Replicas used for data distribution, w/o replica shards, would create isolated bubbles**



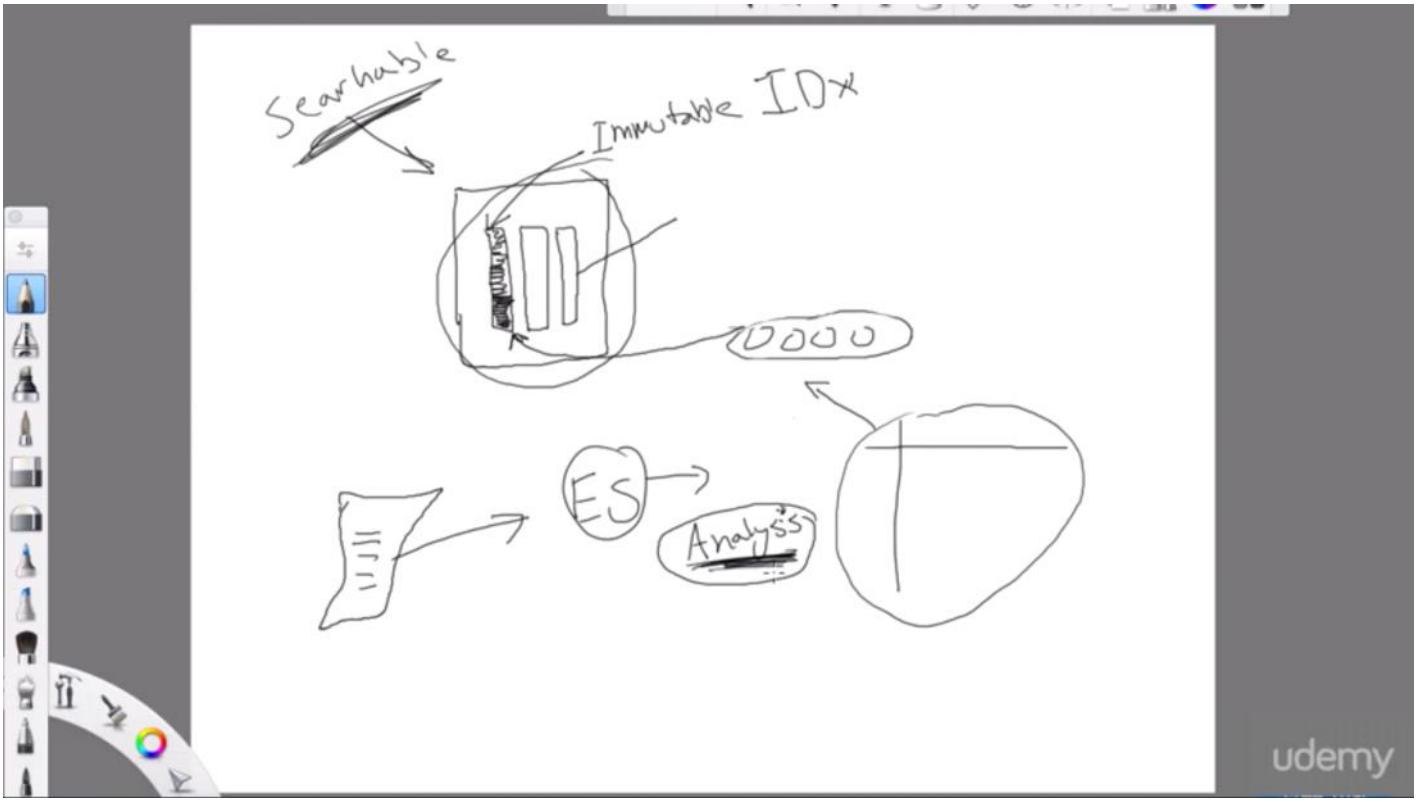
- **Given a request, the nodes use the ID of the document requested to first <u>find the proper primary shard</u>, and then <u>updates each of its replica shards</u> in turn (indexing, deleting, getting, etc.)**
- **This is a form of load-balancing using an algorithm call <u>Round-robin scheduling (RR)</u>, where the process bounces amongst nodes instead of simply hammering on one node.**



- **A shard is a lucene index**
- **A shard is a physical container; it can have multiple segments (inverted indices)**



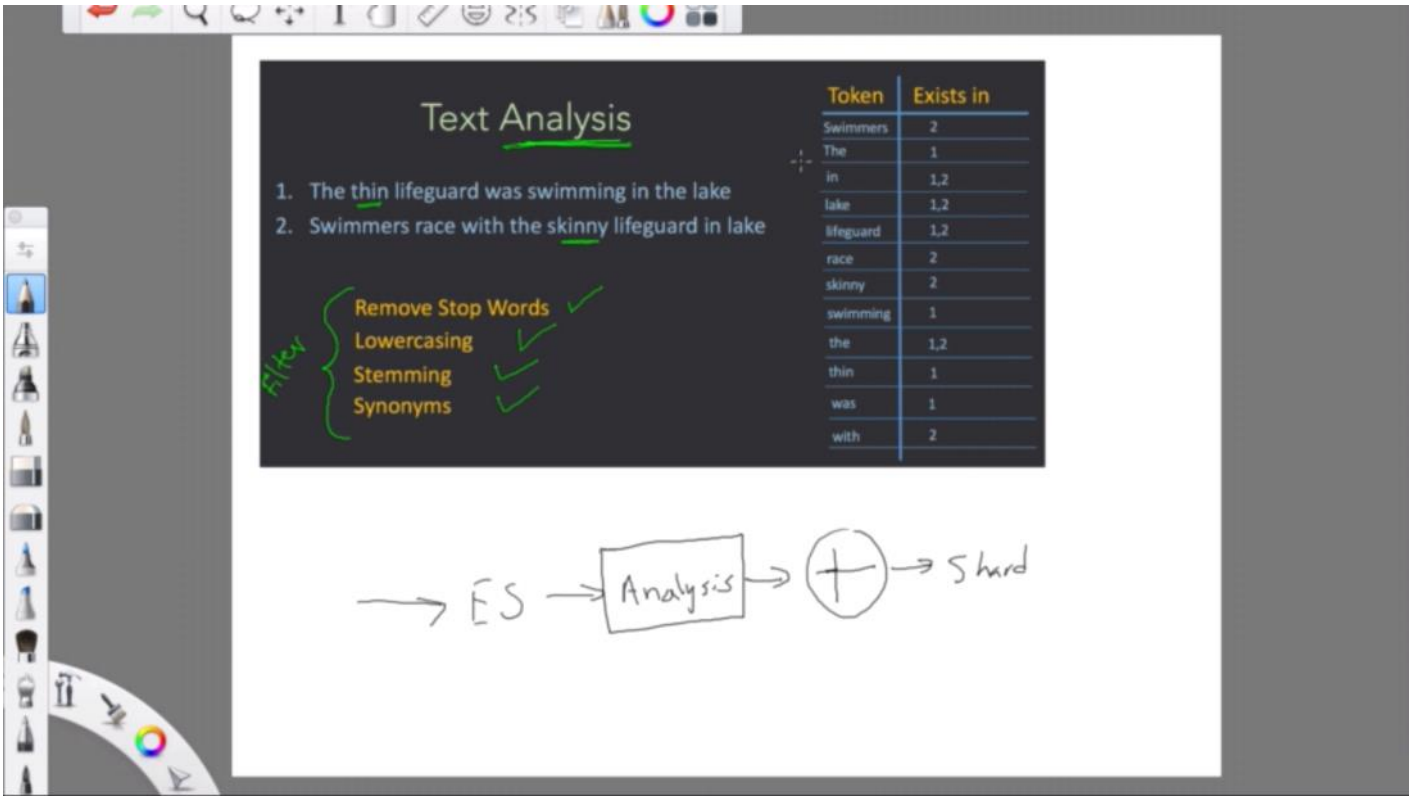
- **ES formulates an inverted index when it's given documents**
- **ES is made of shards, which are made of segments**
- **Given a JSON document (data) sent to ES, ES will deconstruct the data via a step known as text analysis to convert text to tokens to feed the inverted index**
- **Once it forms an inverted index, it gets sent to a memory buffer, which gets populated with each document that passes through**
- **The buffer gets committed to a specific segment, which is how those segments get confirmed**
- **For each committed buffer, a new one rises up to continue processing incoming documents**
- **Once a segment gets completed, the segment becomes an *immutable inverted index* (can create more segments, but can't overwrite existing segments)**


Searchable
Immutable IDx
ES
Analysis
udemy

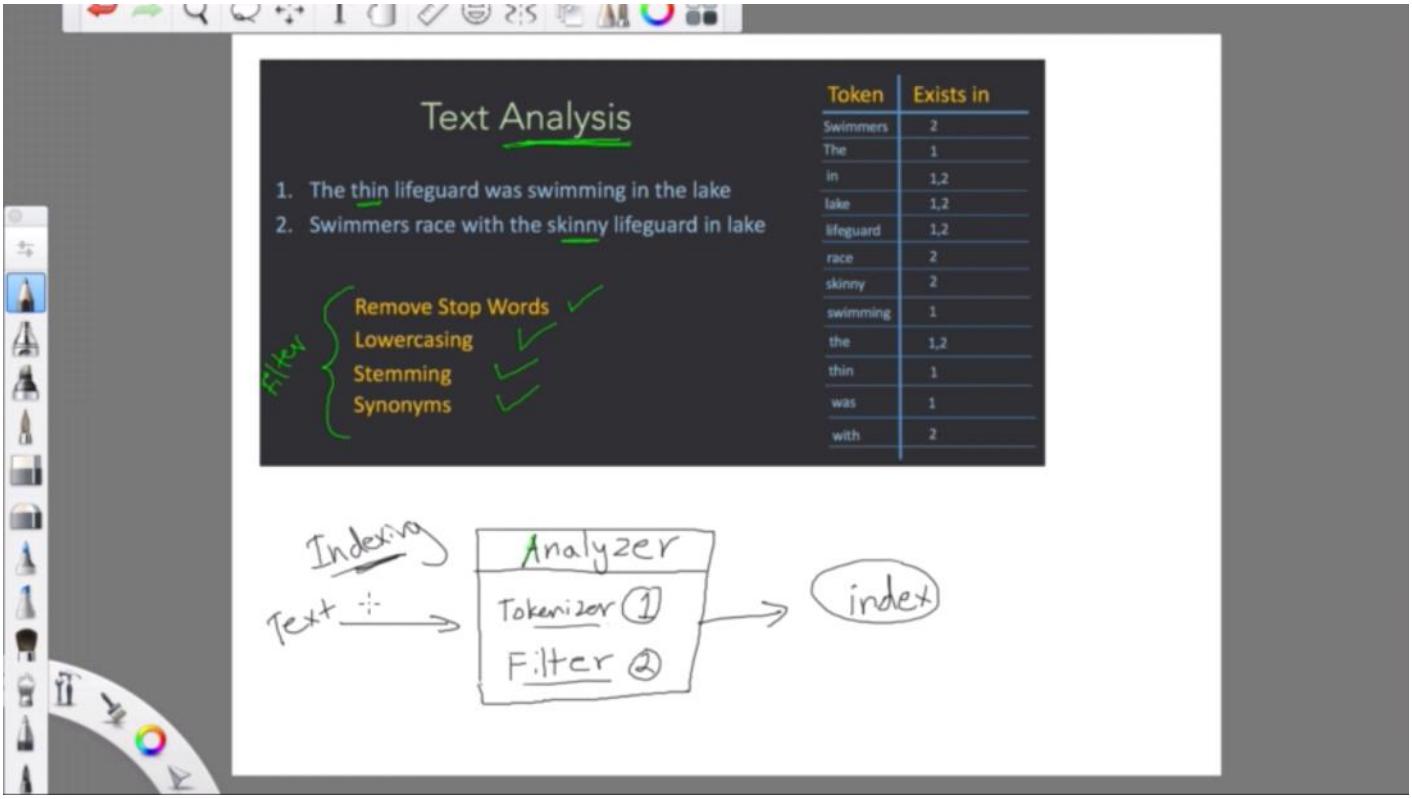
Lecture 7: Text Analysis for Indexing and Searching

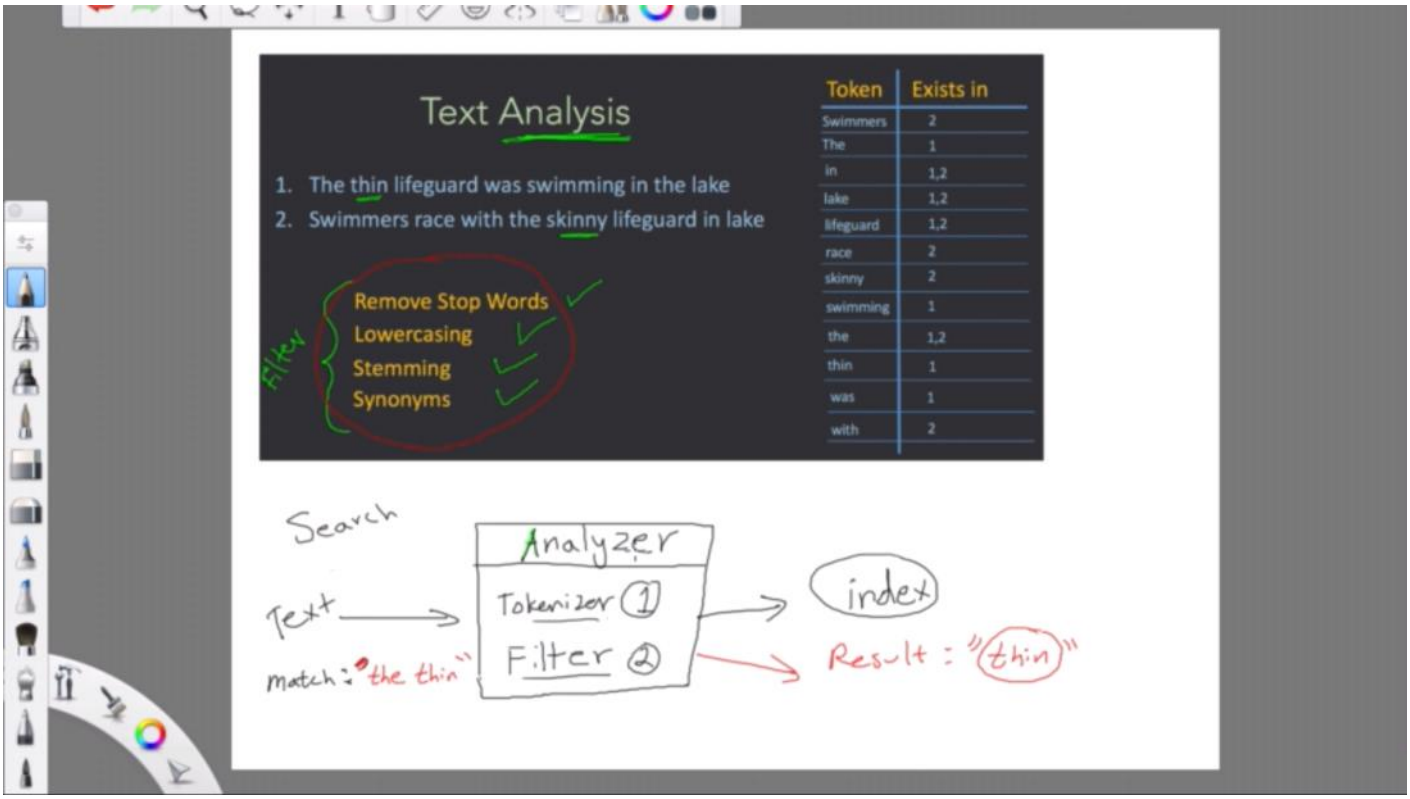
Tuesday, June 11, 2019 2:59 PM

- **ES --> Analysis --> Inverted Index (IDx) --> Shard (Segment)**
- **This process is key for indexing, retrieving, etc.**
- **Text Analysis (Filters):**
  - **Remove Stop Words ("useless" words)**
  - **Lowercasing**
  - **Stemming (root words prioritized)**
  - **Synonyms**



- **Analyzer (for both Indexing and Querying)**
  - **Tokenizer**
  - **Filter**


Text Analysis
1. The thin lifeguard was swimming in the lake
2. Swimmers race with the skinny lifeguard in lake
Filter
Remove Stop Words
Lowercasing
Stemming
Synonyms
Token | Exists in
Swimmers 2
The 1
in 1,2
lake 1,2
lifeguard 1,2
race 2
skinny 2
swimming 1
the 1,2
thin 1
was 1
with 2
Indexing
Text
Analyzer
Tokenizer 1
Filter 2
index


Text Analysis
1. The thin lifeguard was swimming in the lake
2. Swimmers race with the skinny lifeguard in lake
Filter
Remove Stop Words
Lowercasing
Stemming
Synonyms
Token | Exists in
Swimmers 2
The 1
in 1,2
lake 1,2
lifeguard 1,2
race 2
skinny 2
swimming 1
the 1,2
thin 1
was 1
with 2
Search
Text
match: "the thin"
Analyzer
Tokenizer 1
Filter 2
index
Result: "thin"

Configure the Index Structure

```
{
    "settings": {
    ...
    },

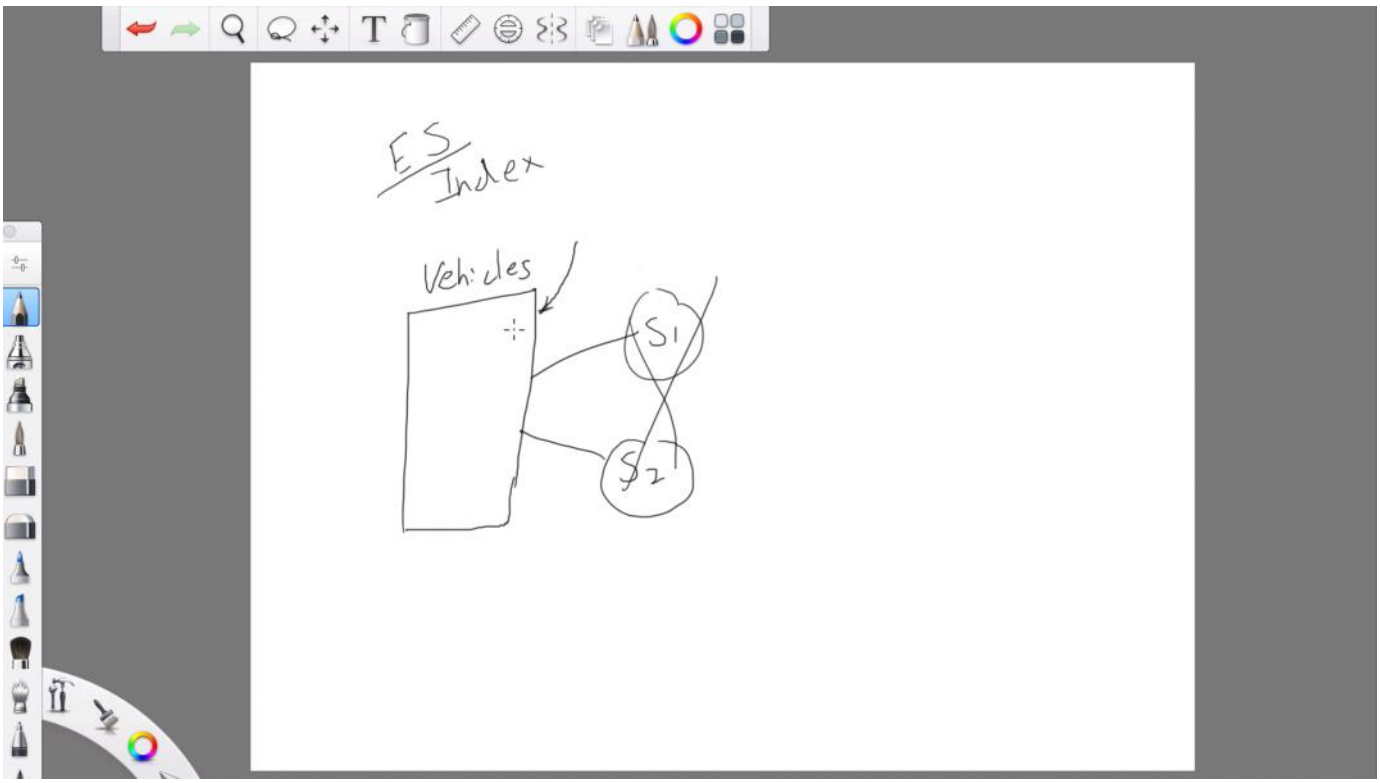
    "mappings": {
    ...
    }

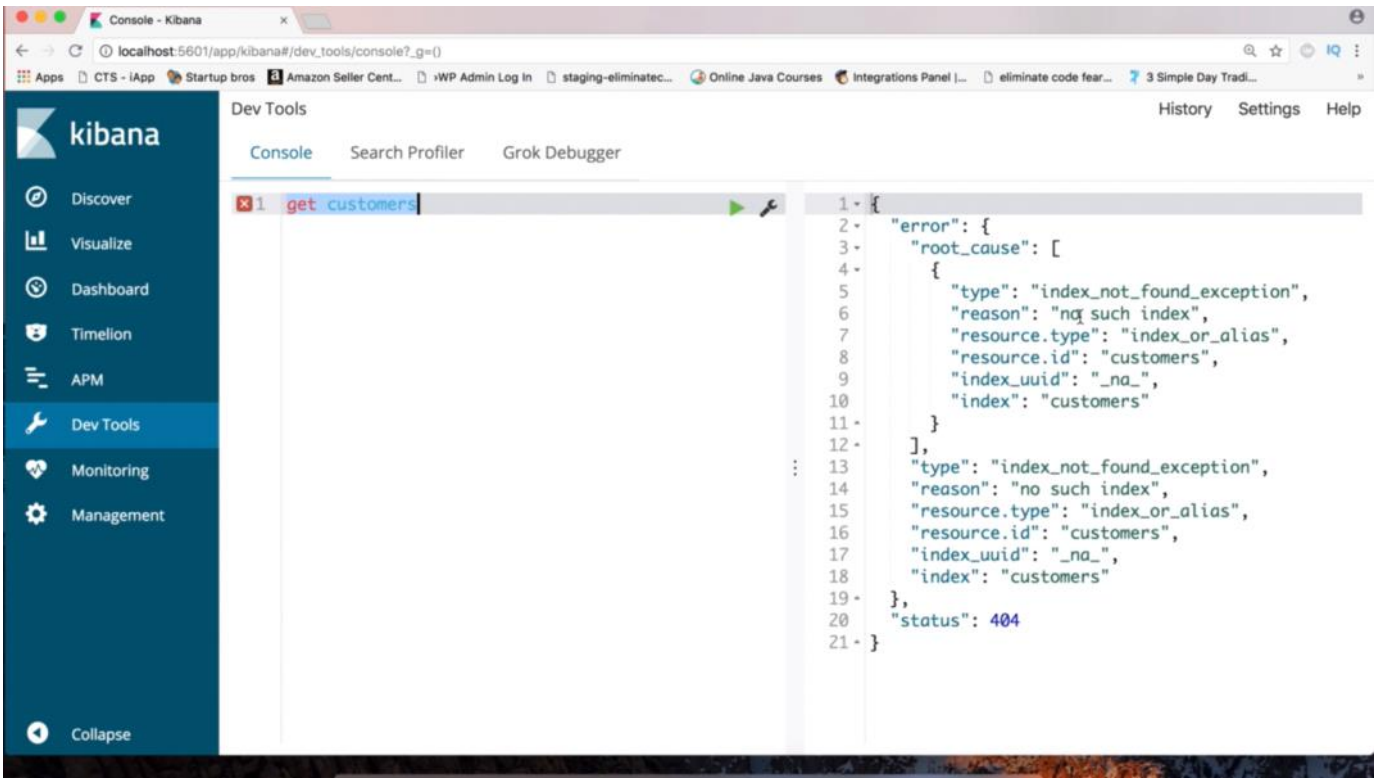
}
```

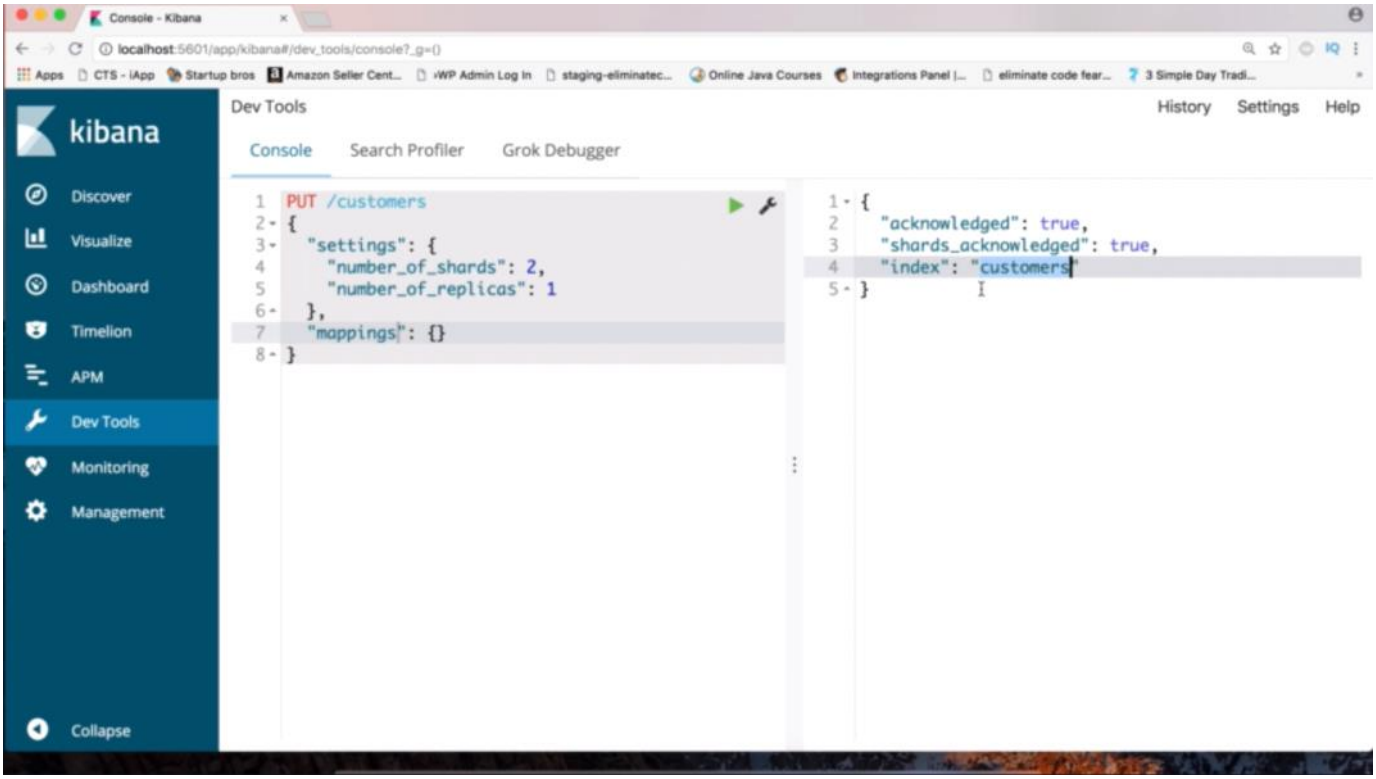
Lecture 8: Index Settings and Mappings (Part 1)

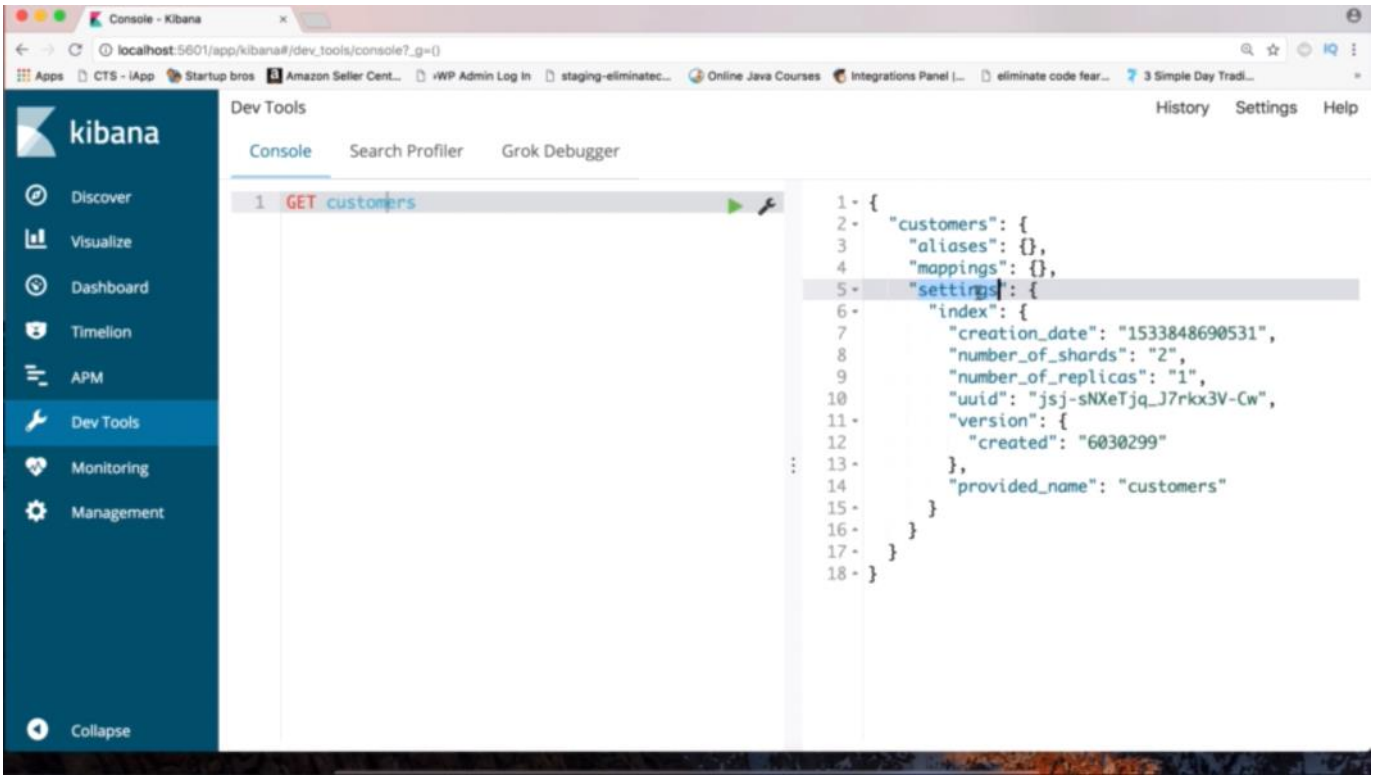
Tuesday, July 9, 2019 3:22 PM

- **Not as much of a focus on shards, more concerned with the logical representation of said data and how to load data into each index --> higher level usability standpoint**

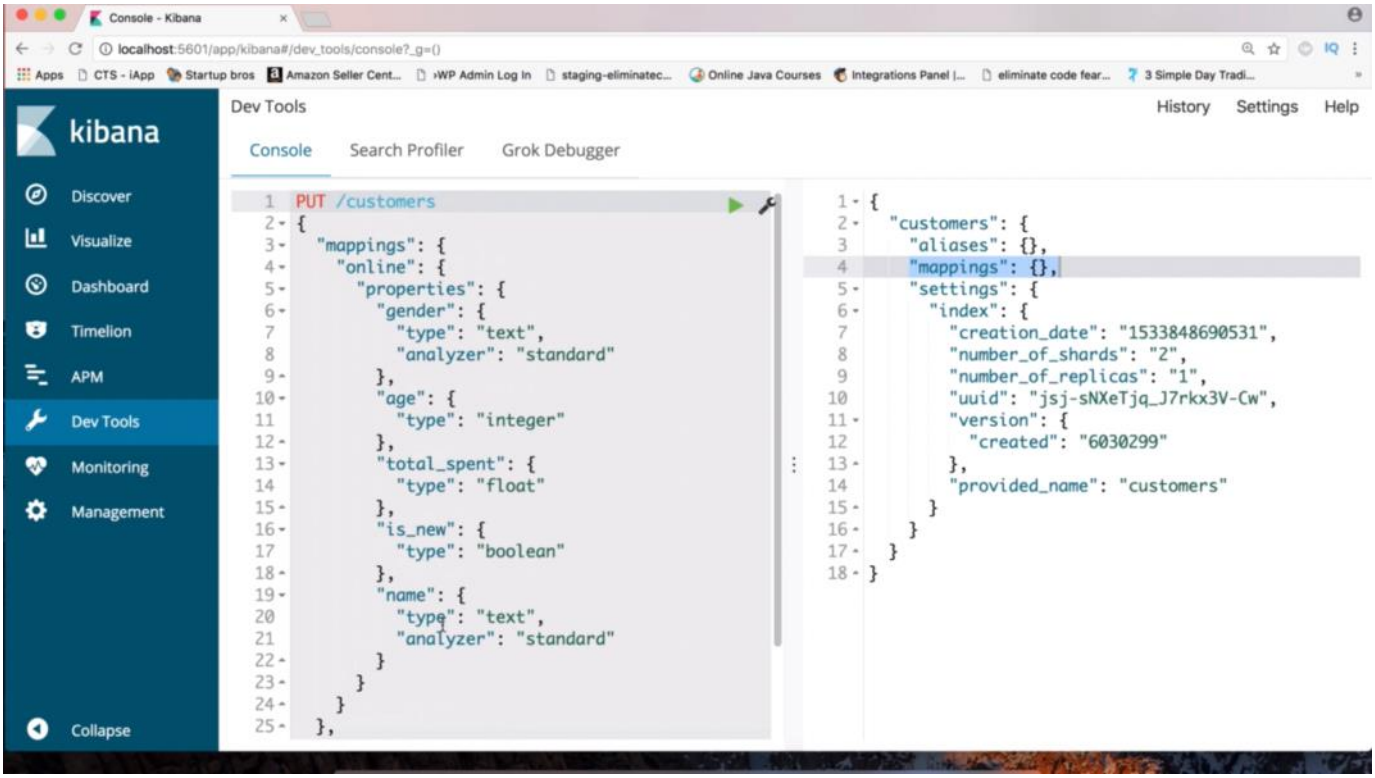


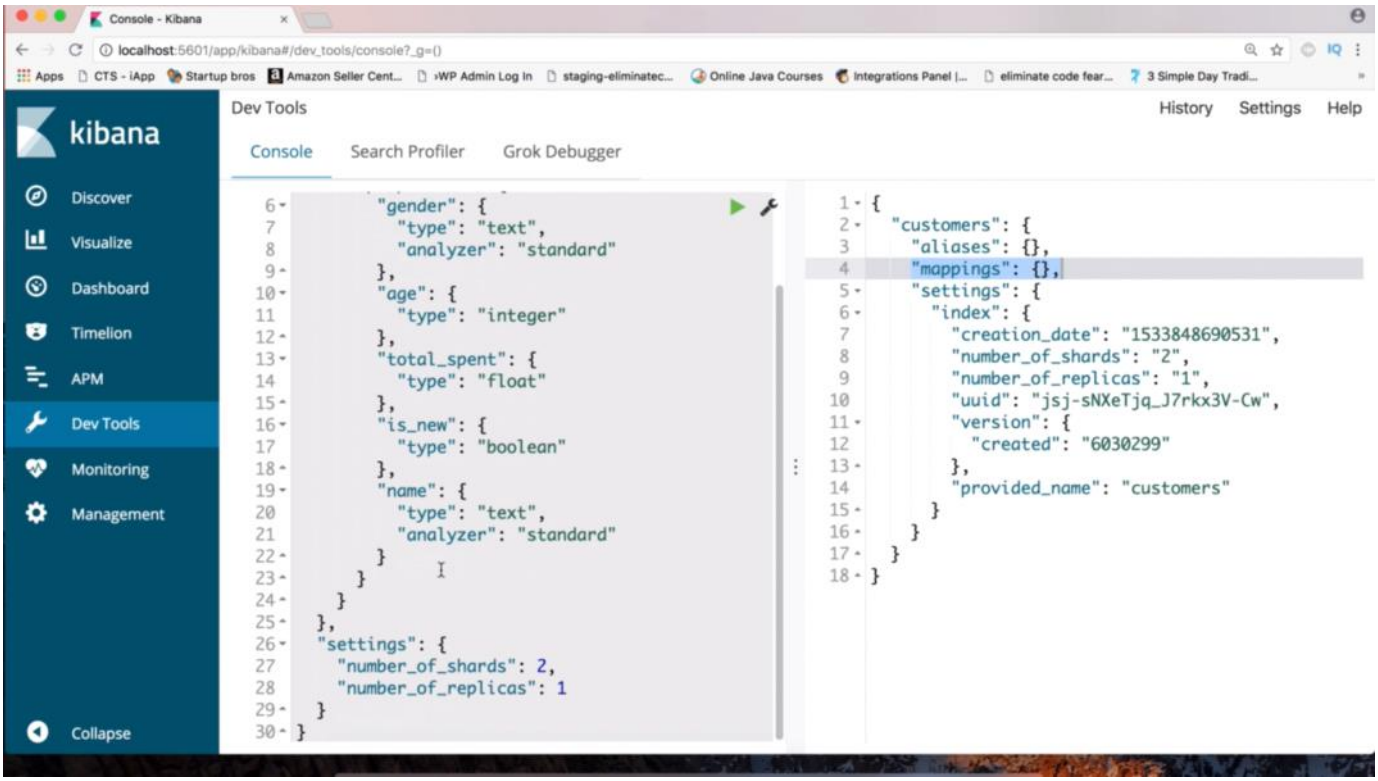






- **<u>Analyzers</u> are used for splitting text to be stored in the inverted index**





- **PUT /customers**
  {
  "mappings": {
  "online": {
  "properties": {
  "gender": {
  "type": "text",
  "analyzer": "standard"

```
      },
      "age": {
        "type": "integer"
      },
      "total_spent": {
        "type": "float"
      },
      "is_new": {
        "type": "boolean"
      },
      "name": {
        "type": "text",
        "analyzer": "standard"
      }
    }
  }
 },
 "settings": {
  "number_of_shards": 2,
  "number_of_replicas": 1
 }
}
```

- ```
  {
   "acknowledged": true,
   "shards_acknowledged": true,
   "index": "customers"
  }
  ```
  ```

Lecture 9: Index Settings and Mappings (Part 2)

Tuesday, June 11, 2019 3:22 PM

```
PUT customers/online/124
{
    "name": "Mary Cranford",
    "address": "310 Clark Ave",
    "gender": "female",
    "age": 34,
    "total_spent": 550.75,
    "is_new": false
}
```

```
{
  "_index": "customers",
  "_type": "online",
  "_id": "124",
  "_version": 1,
  "result": "created",
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  },
  "created": true
}
```

- **Elastic will take additional fields and auto-edit the schema/mapping in a dynamic manner**
- **Can configure Elasticsearch to multiple levels of dynamic mapping/strictness**

```
GET customers
```

```
"customers": {
  "aliases": {},
  "mappings": {
    "online": {
      "properties": {
        "address": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        },
        "age": {
          "type": "integer"
        },
        "gender": {
          "type": "text"
          "analyzer": "standard"
        },
        "is_new": {
          "type": "boolean"
        },
        "name": {
          "type": "text",
          "analyzer": "standard"
        },
        "total_spent": {
          "type": "float"
```

dynamic :

If set to false - indexing field will be Ignored

If set to strict - indexing field will throw error

```
GET customers/_mapping/online
{
  "dynamic": false
}
```

```
{
  "acknowledged": true
}
```

```
PUT /customers/online/124
{
    "name": "Mary Cranford",
    "address": "310 Clark Ave",
    "gender": "female",
    "age": 34,
    "total_spent": 550.75,
    "is_new": false,
    "retired":true
}
```

```
{
  "error": {
    "root_cause": [
      {
        "type": "strict_dynamic_mapping_exception",
        "reason": "mapping set to strict, dynamic introduction of [retired] within [online] is not allowed"
      }
    ],
    "type": "strict_dynamic_mapping_exception",
    "reason": "mapping set to strict, dynamic introduction of [retired] within [online] is not allowed"
  },
  "status": 400
}
```

- https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-analyzers.html
- **Can test how analyzers work**

```
POST _analyze
{
  "analyzer": "whitespace",
  "text":     "The quick brown fox."
}
```

```
{
  "tokens": [
    {
      "token": "The",
      "start_offset": 0,
      "end_offset": 3,
      "type": "word",
      "position": 0
    },
    {
      "token": "quick",
      "start_offset": 4,
      "end_offset": 9,
      "type": "word",
      "position": 1
    },
    {
      "token": "brown",
      "start_offset": 10,
      "end_offset": 15,
      "type": "word",
      "position": 2
    },
    {
      "token": "fox.",
      "start_offset": 16,
      "end_offset": 20,
      "type": "word",
      "position": 3
    }
```

```
POST _analyze
{
  "analyzer": "standard",
  "text":     "The quick brown fox."
}
```

```
    {
      "token": "the",
      "start_offset": 0,
      "end_offset": 3,
      "type": "<ALPHANUM>",
      "position": 0
    },
    {
      "token": "quick",
      "start_offset": 4,
      "end_offset": 9,
      "type": "<ALPHANUM>",
      "position": 1
    },
    {
      "token": "brown",
      "start_offset": 10,
      "end_offset": 15,
      "type": "<ALPHANUM>",
      "position": 2
    },
    {
      "token": "fox",
      "start_offset": 16,
      "end_offset": 19,
      "type": "<ALPHANUM>",
      "position": 3
    }
  ]
}
```

- **Where standard analyzer considers consecutive alphanumeric strings as one object, simple analyzer splits on any non-alphabetic character (i.e. numbers and punctuation)**

```
POST _analyze
{
  "analyzer": "simple",
  "text":     "The quick984 brown746 fox."
}
```

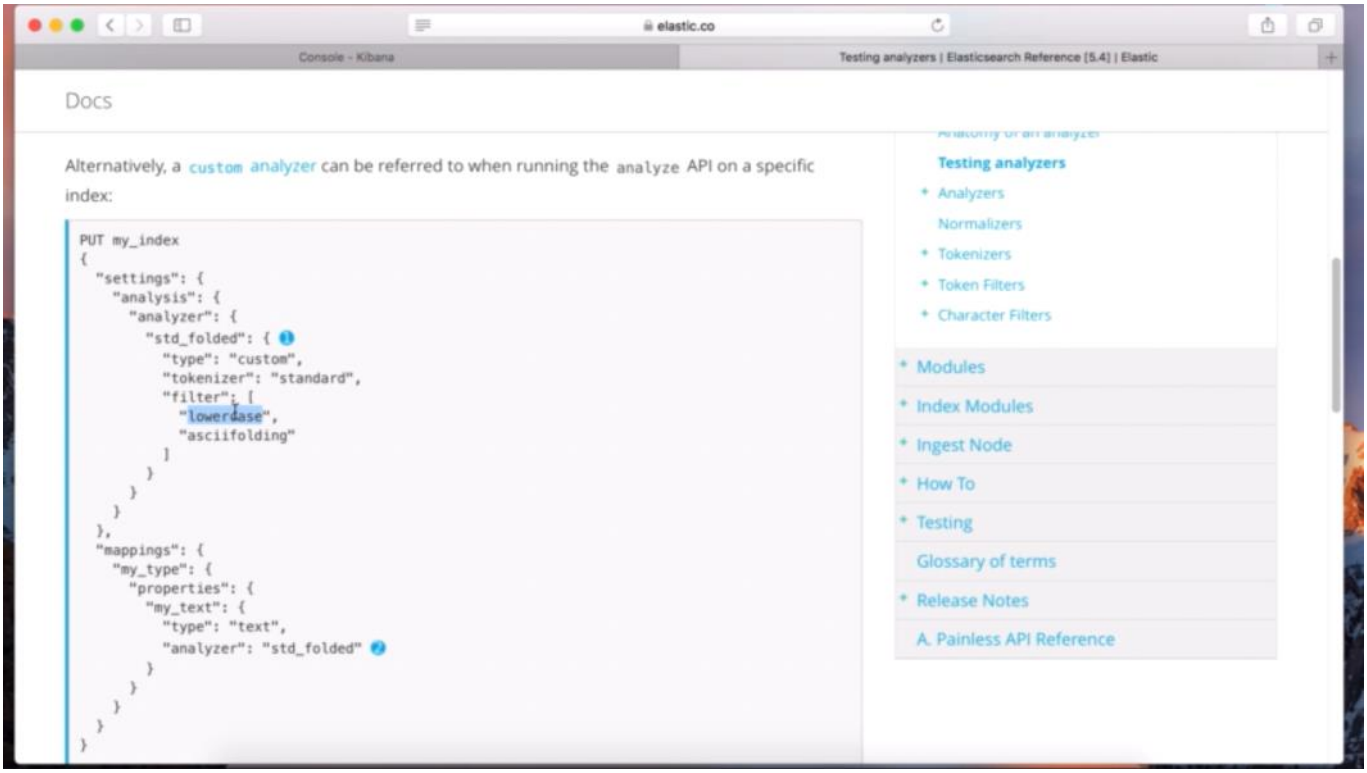
```
{
  "tokens": [
    {
      "token": "the",
      "start_offset": 0,
      "end_offset": 3,
      "type": "word",
      "position": 0
    },
    {
      "token": "quick",
      "start_offset": 4,
      "end_offset": 9,
      "type": "word",
      "position": 1
    },
    {
      "token": "brown",
      "start_offset": 13,
      "end_offset": 18,
      "type": "word",
      "position": 2
    },
    {
      "token": "fox",
      "start_offset": 22,
      "end_offset": 25,
      "type": "word",
      "position": 3
    }
```

- **The english analyzer splits on punctuation and can split root words from gerunds and other forms**

```
POST _analyze
{
  "analyzer": "english",
  "text":     "The quick running fox."
}
```

```
{
  "tokens": [
    {
      "token": "quick",
      "start_offset": 4,
      "end_offset": 9,
      "type": "<ALPHANUM>",
      "position": 1
    },
    {
      "token": "run",
      "start_offset": 10,
      "end_offset": 17,
      "type": "<ALPHANUM>",
      "position": 2
    },
    {
      "token": "fox",
      "start_offset": 18,
      "end_offset": 21,
      "type": "<ALPHANUM>",
      "position": 3
    }
  ]
}
```

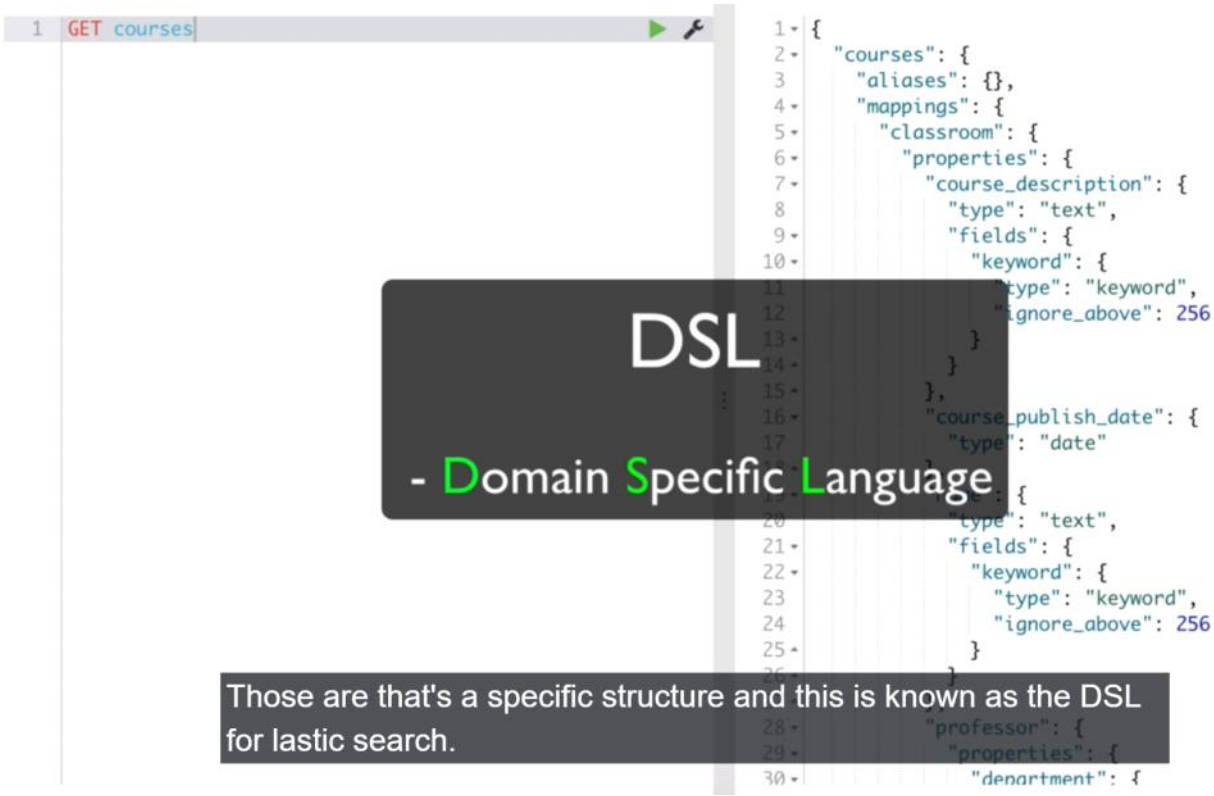
- **Can define custom analyzer, which depends on tokenizer, filter, mappings, properties, etc.**


elastic.co
Console - Kibana
Testing analyzers | Elasticsearch Reference [5.4] | Elastic
Docs
Alternatively, a custom analyzer can be referred to when running the analyze API on a specific index:
PUT my_index
{
"settings": {
"analysis": {
"analyzer": {
"std_folded": {
"type": "custom",
"tokenizer": "standard",
"filter": [
"lowercase",
"asciifolding"
]
}
}
}
},
"mappings": {
"my_type": {
"properties": {
"my_text": {
"type": "text",
"analyzer": "std_folded"
}
}
}
}
}
Testing analyzers
Analyzers
Normalizers
Tokenizers
Token Filters
Character Filters
Modules
Index Modules
Ingest Node
How To
Testing
Glossary of terms
Release Notes
A. Painless API Reference

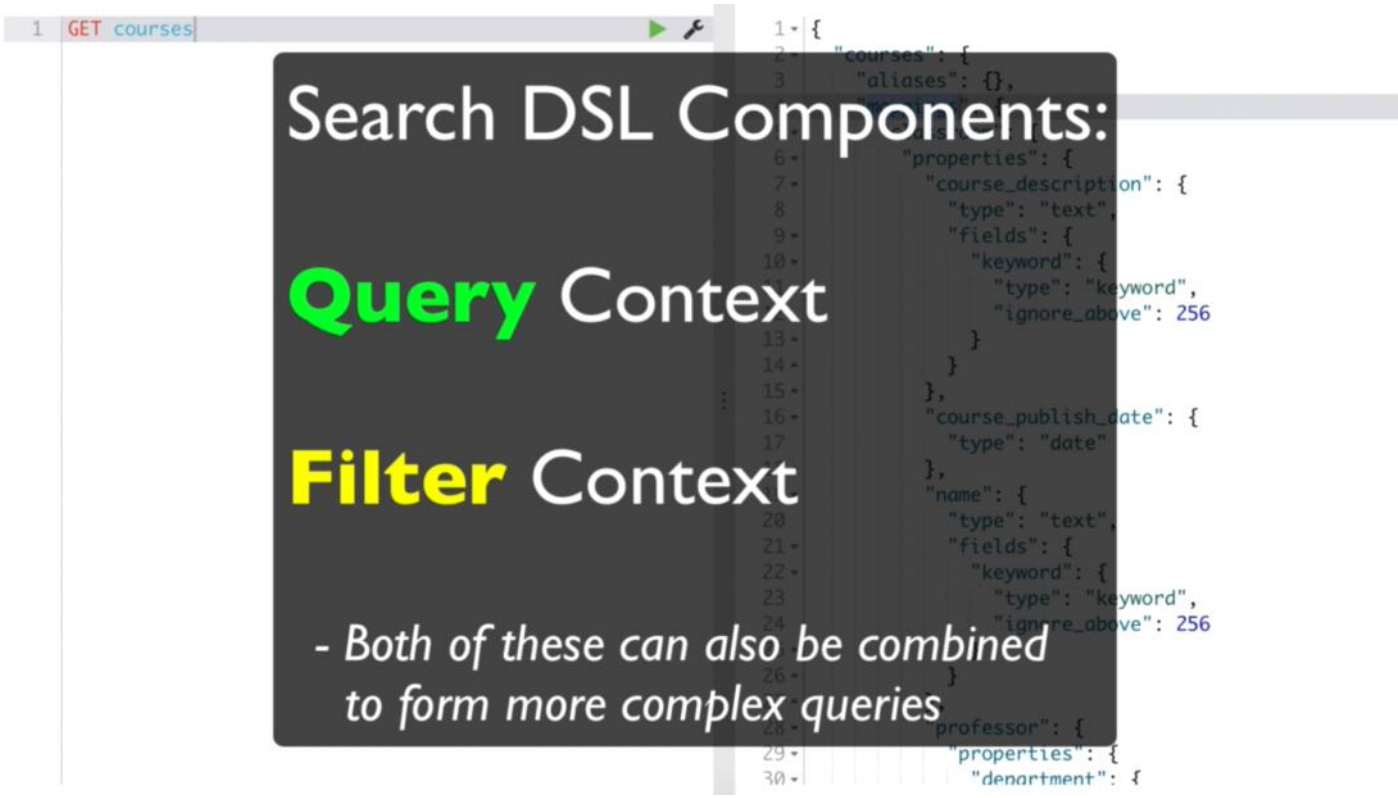
Lecture 10: Search DSL Query Context (Part 1)

Tuesday, June 11, 2019 3:23 PM

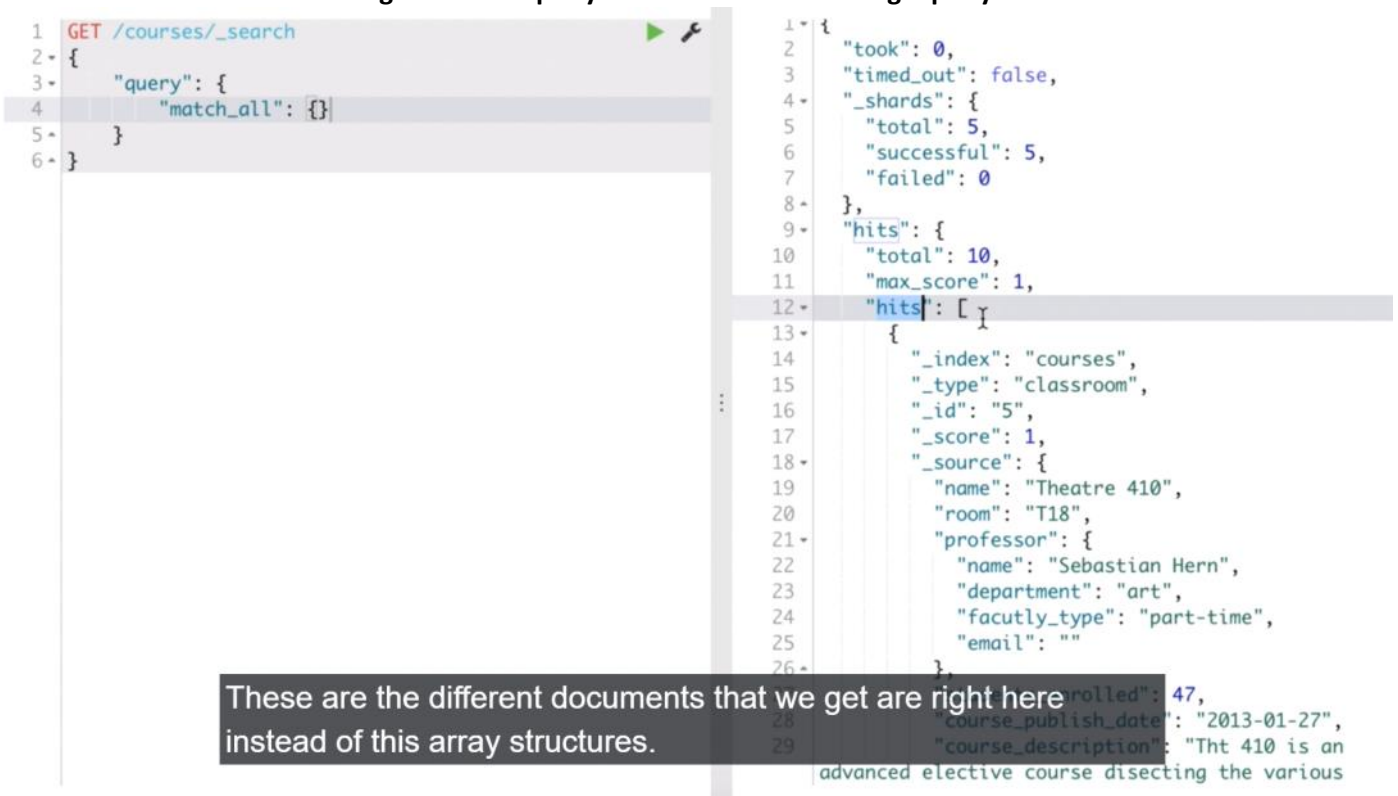
- **Elasticsearch is HTTP and JSON**
- **We provide the HTTP URL endpoint for indexing and for querying**
- **The body of the request contains that JSON to send as part of the request**



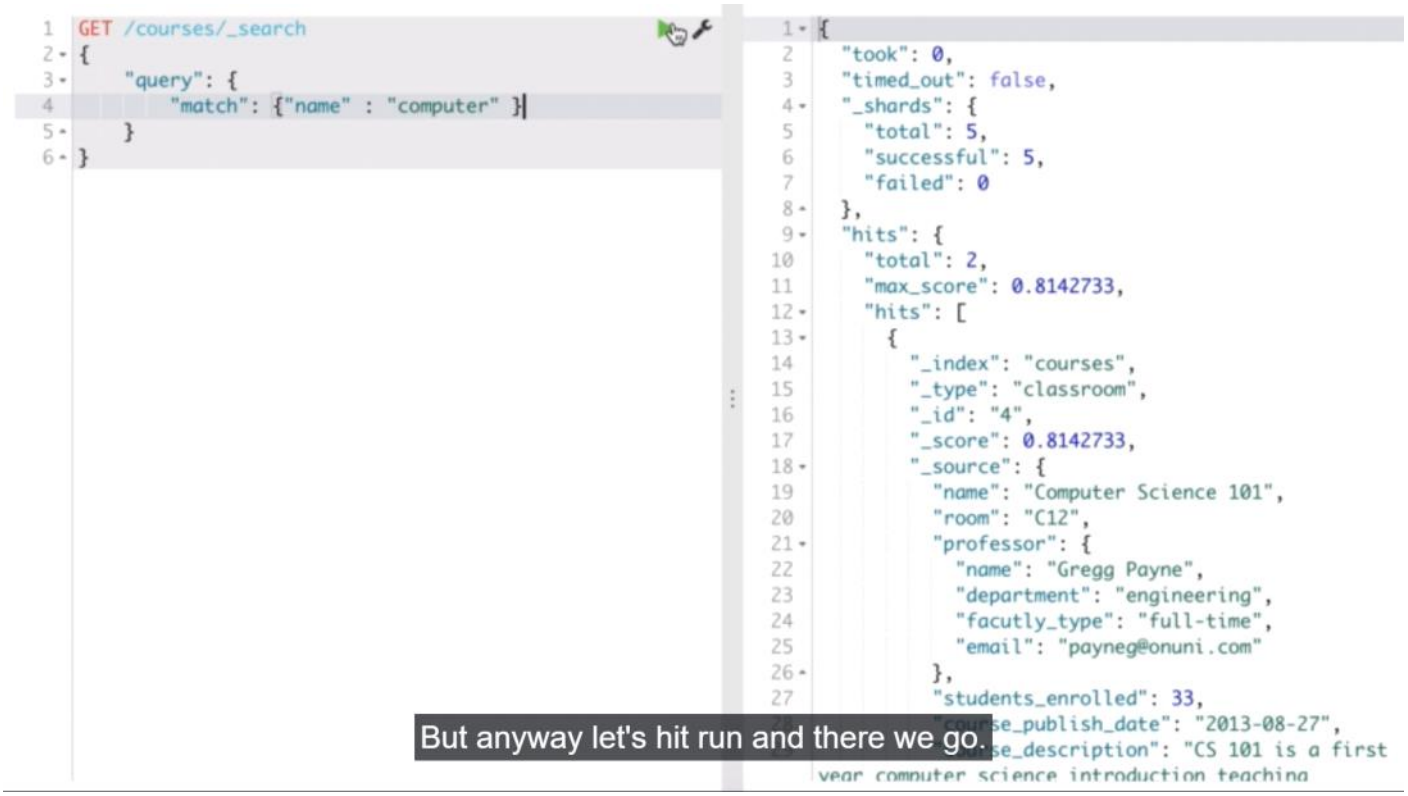
- **One objective: match your search criteria and provide some kind of relevancy score**
  - **Ex. Searching for the term "technology" in documents where documents that contain more occurrences of the word are considered more relevant**
- **<u>Query context</u> is used for full text searches**
  - **Match documents that contain the search criteria as well as specify how well that document matched the particular search criteria by providing that relevancy score**



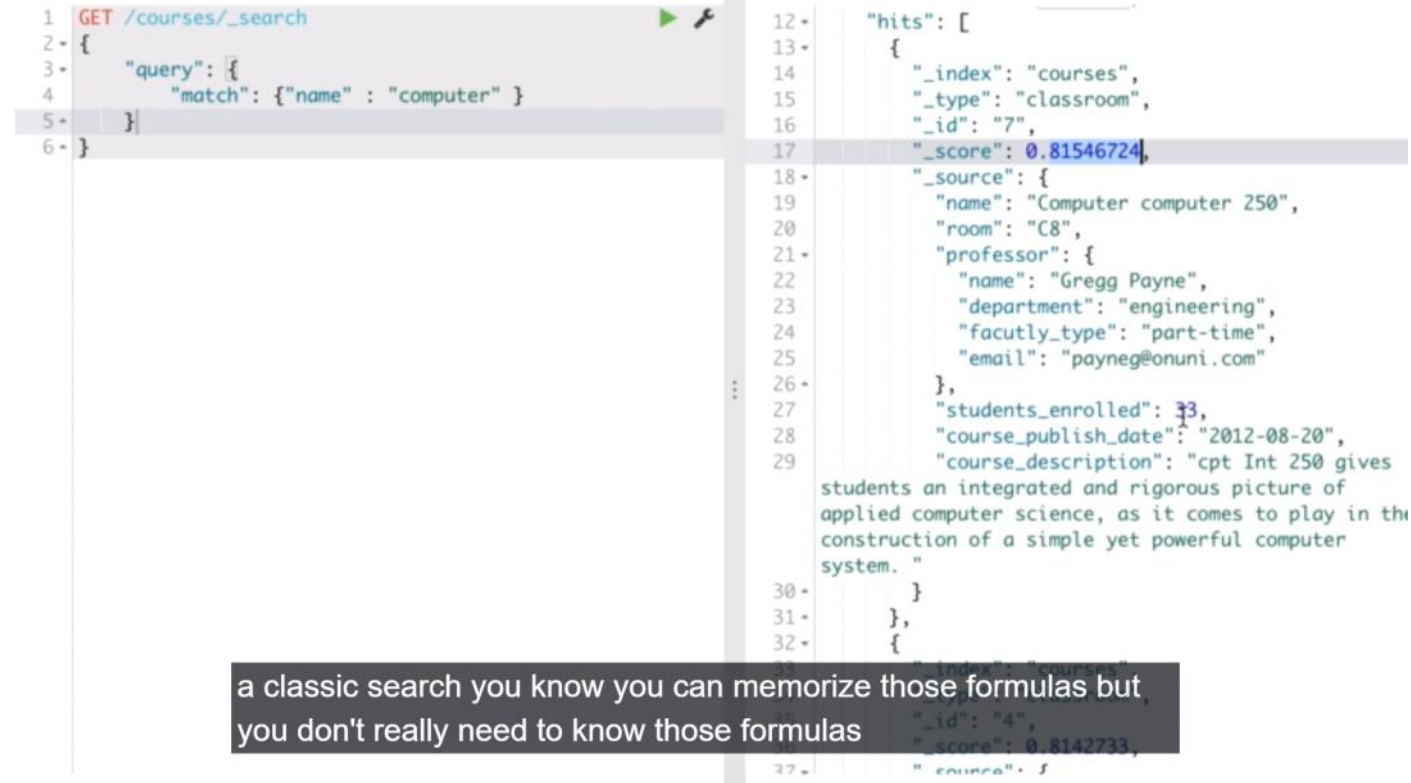
- **Filter context**
  - **Can be combined together with query context to form one large query**



- **All the documents inside the array structure are called hits**
- **"_score" field holds the relevancy score**



- **Score based off of "significance" within the field specified**
  - **Ex. A term's score within a field could be based off of # chars**



- **For multiple criteria searches, you <u>must have a "must" structure</u> -- this allows for multiple variable confirmations simultaneously**
- **You also require an <u>enclosing "bool" statement</u>, without which the query will not run properly**
- **GET /courses/_search**
  {

```
 "query": {
  "bool": {
     "must":[
      {"match": {"name": "computer"}},
      {"match": {"room": "c8"}}
      ]
  }
 }
}
```

Lecture 11: Search DSL Query Context (Part 2)

Tuesday, June 11, 2019 3:23 PM

- **With must comes must not**

```
GET /courses/_search
{
    "query": {
      "bool": {
        "must":[
          {"match": {"name": "accounting"}},
          {"match": {"room": "e3"}}
          ],

         "must_not":[
            {"match": {"professor.name": "bill"}}
          ],

          "should" :[
            {"match": {"name": "computer"}}
            ]
      }

    }
}
```

```
  },
  "hits": {
    "total": 1,
    "max_score": 1.379555,
    "hits": [
      {
        "_index": "courses",
        "_type": "classroom",
        "_id": "1",
        "_score": 1.379555,
        "_source": {
          "name": "Accounting 101",
          "room": "E3",
          "professor": {
            "name": "Thomas Baszo",
            "department": "finance",
            "facutly_type": "part-time",
            "email": "baszot@onuni.com"
          },
          "students_enrolled": 27,
          "course_publish_date": "2015-01-19",
          "course_description": "Act 101 is a course
from the business school on the introduction to
accounting that teaches students how to read and
compose basic financial statements"
        }
      ]
    }
}
```

It's pretty much almost always ignored by classic search unless we specify a special field.

- **With must not comes should which is ^^**
- **"minimum_should_match" refers to a query where the should statement must be true in at least one hit**

```
GET /courses/_search
{
    "query": {
      "bool": {
        "must":[
          {"match": {"name": "accounting"}}
          ],

         "must_not":[
            {"match": {"professor.name": "bill"}}
          ],

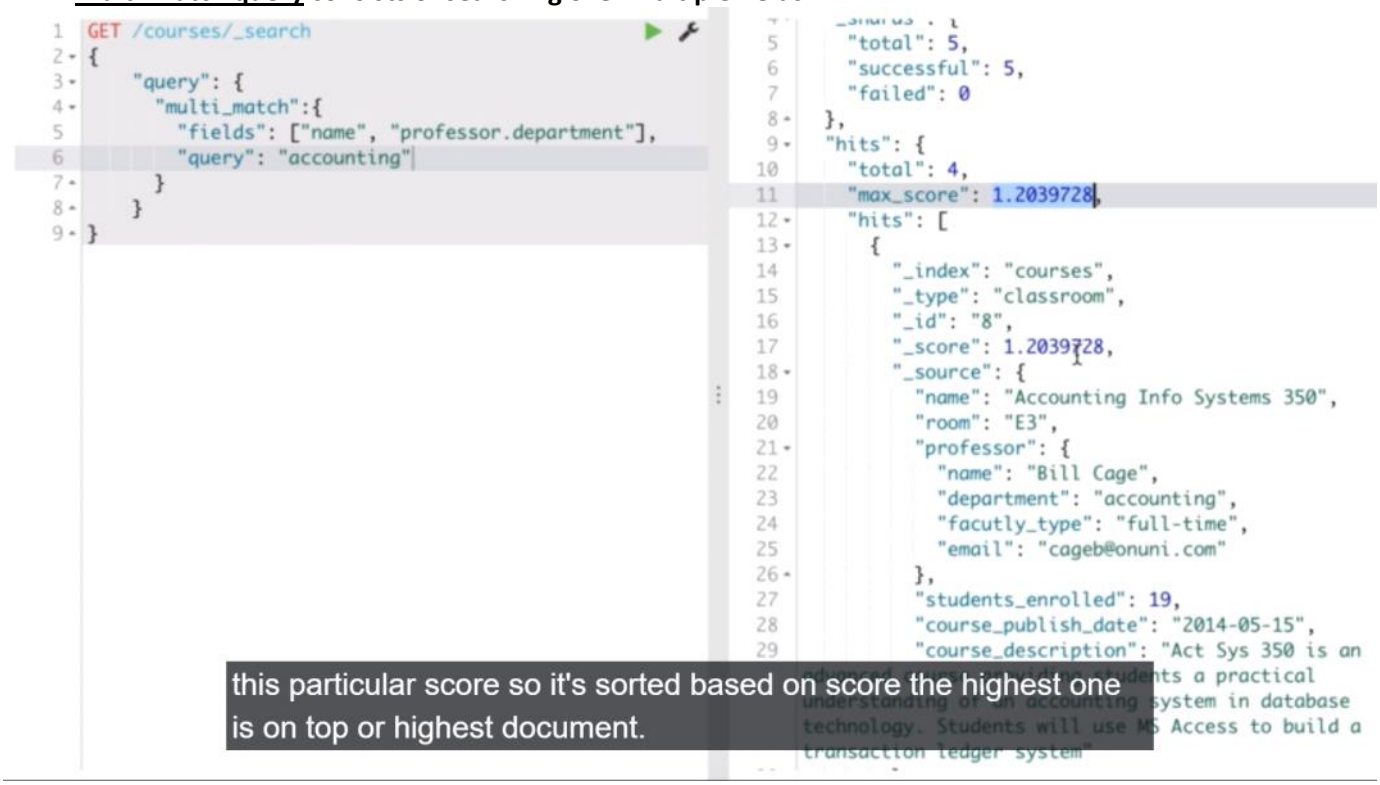
          "should" :[
            {"match": {"room": "e7"}}
            ],
            "minimum_should_match": 1
      }


    }
}
```

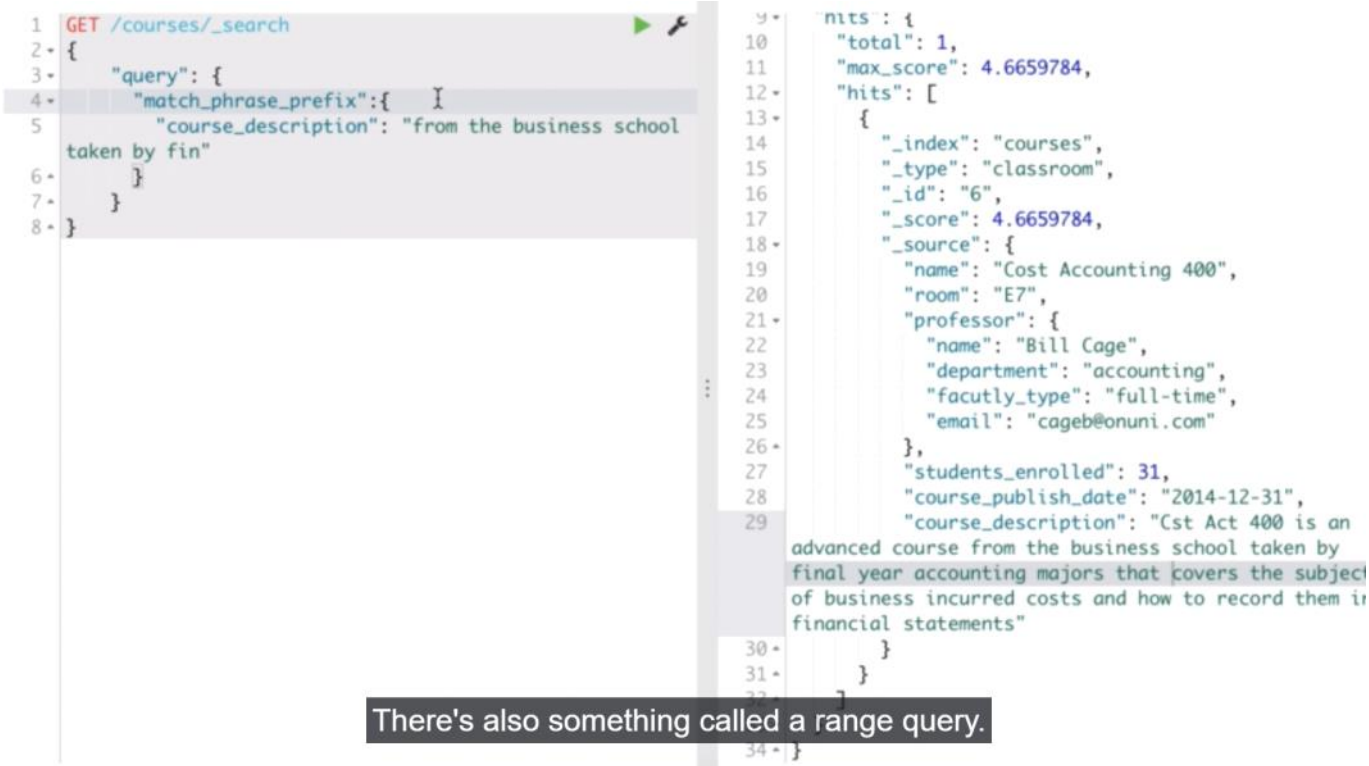
```
          "course_description": "Act 101 is a course
from the business school on the introduction to
accounting that teaches students how to read and
compose basic financial statements"
        }
      },
      {
        "_index": "courses",
        "_type": "classroom",
        "_id": "9",
        "_score": 0.6099695,
        "_source": {
          "name": "Tax Accounting 200",
          "room": "E7",
          "professor": {
            "name": "Thomas Baszo",
            "department": "finance",
            "facutly_type": "part-time",
            "email": "baszot@onuni.com"
          },
          "students_enrolled": 17,
          "course_publish_date": "2016-06-15",
          "course_description": "Tax Act 200 is an
intermediate course covering various aspects of tax
law"
        }
      ]
    }
}
```

And if I at a minimum should match one that's going to only return this document.

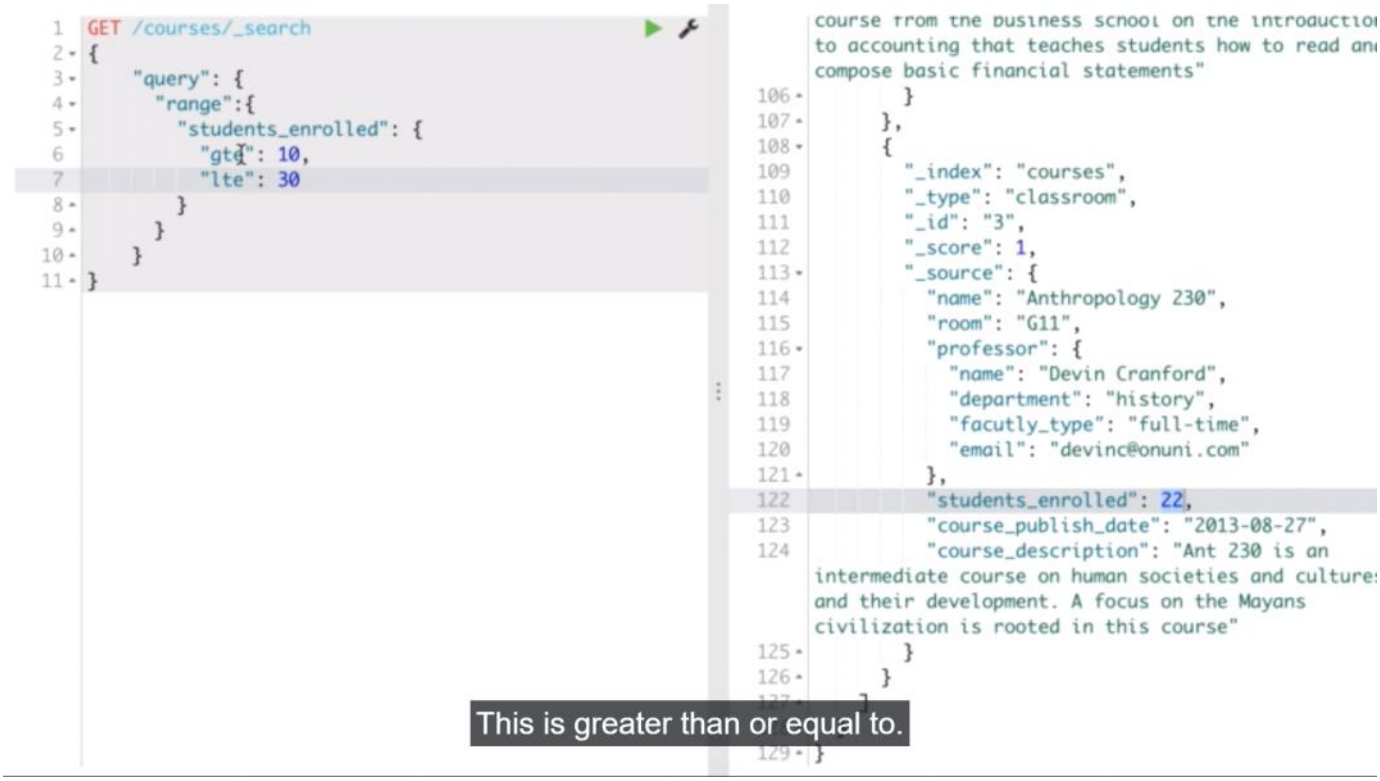
- **Multi-match query consists of searching over multiple fields**



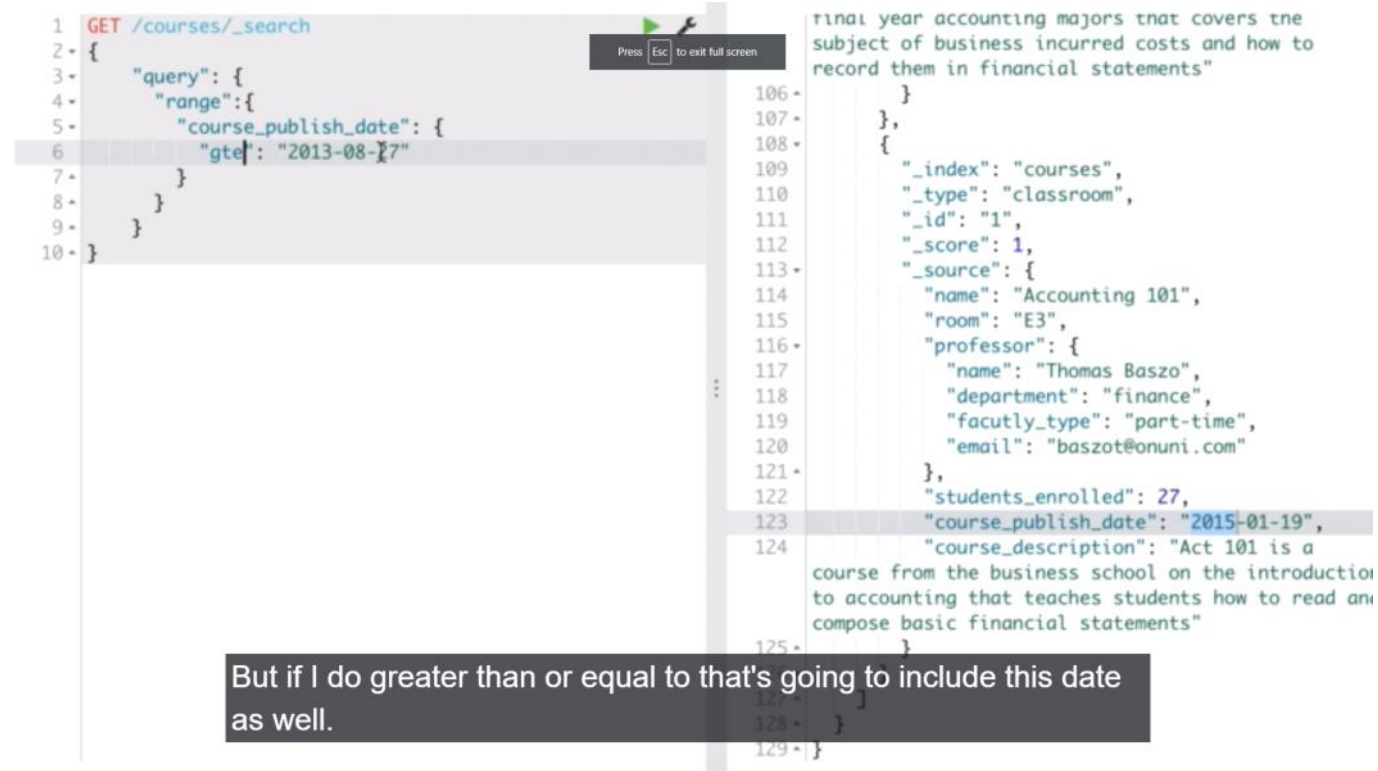
- **"match_phrase" queries for a specific phrase contained in a specific field (requires complete tokens & complete words)**
- **"match_phrase_prefix" can be used to figure out partial tokens**

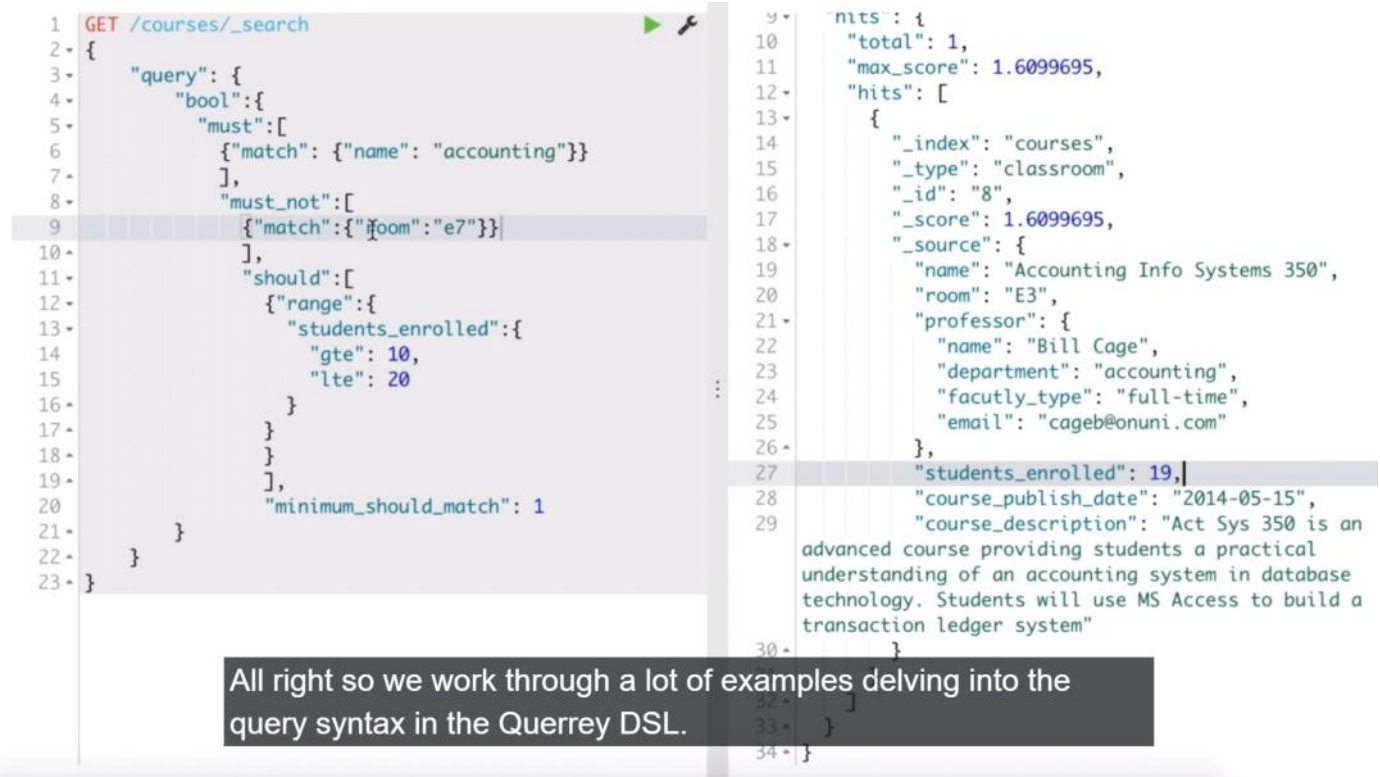


- **A range query is used to search for documents that exist within a given range**



- **"gte" - greater than or equal to, "gt" - greater than, "lte" - less than or equal to, "lt" - less than**




GET /courses/_search
{
"query": {
"bool":{
"must":[
{"match": {"name": "accounting"}}
],
"must_not":[
{"match":{"room":"e7"}}
],
"should":[
{"range":{
"students_enrolled":{
"gte": 10,
"lte": 20
}
}
}
],
"minimum_should_match": 1
}
}
}
"hits" : {
"total": 1,
"max_score": 1.6099695,
"hits": [
{
"_index": "courses",
"_type": "classroom",
"_id": "8",
"_score": 1.6099695,
"_source": {
"name": "Accounting Info Systems 350",
"room": "E3",
"professor": {
"name": "Bill Cage",
"department": "accounting",
"facutly_type": "full-time",
"email": "cageb@onuni.com"
},
"students_enrolled": 19,
"course_publish_date": "2014-05-15",
"course_description": "Act Sys 350 is an advanced course providing students a practical understanding of an accounting system in database technology. Students will use MS Access to build a transaction ledger system"
}
]
}
}
All right so we work through a lot of examples delving into the query syntax in the Querrey DSL.

Lecture 12: Search DSL Filter Context

Tuesday, June 11, 2019 3:23 PM

- **Filter query used for filtering documents that we need**
- **Filter queries are cached, so Elasticsearch will cache previously run queries**

```
GET /courses/_search
{
  "query": {
    "bool": {
      "filter":{
        "bool":{
          "must":[
              { "match":{"professor.name": "bill"}},
              {"match":{"name": "accounting"}}
          ]

        }
      }
    }
  }

}
```

```
  },
  "hits": {
    "total": 2,
    "max_score": 0,
    "hits": [
      {
        "_index": "courses",
        "_type": "classroom",
        "_id": "8",
        "_score": 0,
        "_source": {
          "name": "Accounting Info Systems 350",
          "room": "E3",
          "professor": {
            "name": "Bill Cage",
            "department": "accounting",
            "facutly_type": "full-time",
            "email": "cageb@onuni.com"
          },
          "students_enrolled": 19,
          "course_publish_date": "2014-05-15",
          "course_description": "Act Sys 350 is an
advanced course providing students a practical
understanding of an accounting system in database
technology. Students will use MS Access to build a
transaction ledger system"
        }
      }
    ]
  }
}
```

And again look at the max score here.

- **Aspects within a filter query will NOT impact the relevancy score**
- **Within a multi-match, you can use field boosting to alter the guidelines of the relevancy score**

```
                  "gte": 12
                }
              }
            }
          ]

      }
    },

    "should":[
        {"match":{"room": "e3"}},
        {
          "range": {
              "students_enrolled":{
                "gte": 13,
                "lte": 14
              }
            }
          },
          {
    "multi_match": {
        "query":  "market",
        "fields": [ "name", "course_description^2" ]
    }
}
      ]
    }
  }

}
```

```
    "total": 5,
    "successful": 5,
    "failed": 0
  },
  "hits": {
    "total": 10,
    "max_score": 1.8784901,
    "hits": [
      {
        "_index": "courses",
        "_type": "classroom",
        "_id": "2",
        "_score": 1.8784901,
        "_source": {
          "name": "Marketing 101",
          "room": "E4",
          "professor": {
            "name": "William Smith",
            "department": "finance",
            "facutly_type": "part-time",
            "email": "wills@onuni.com"
          },
          "students_enrolled": 18,
          "course_publish_date": "2015-06-21",
          "course_description": "Mkt 101 is a
course from the business school on the introductio
to marketing that teaches students the fundamental
market analysis, customer retention and online
advertisements"
        }
```

with a factor of two.

- **You can alter relevancy via the "^" symbol**
- **Queries are gonna be slower than filters**
- **Syntax wise the filter is very similar to the query where the same components can apply to both**
- **For EXACT matches with no care for relevancy, use FILTER**
- **For RELEVANCY SCORING and for FULL TEXT SEARCH, use a QUERY**

Lecture 13: Aggregations DSL (Part 1)

Tuesday, June 11, 2019 3:24 PM

- ***https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-bulk.html***
- **BULK API can be used to send multiple documents as part of the same request**

```
POST /vehicles/cars/_bulk
{ "index": {}}
{ "price" : 10000, "color" : "white", "make" : "honda", "sold" : "2016-10-28", "condition": "okay"}
{ "index": {}}
{ "price" : 20000, "color" : "white", "make" : "honda", "sold" : "2016-11-05", "condition": "new" }
{ "index": {}}
{ "price" : 30000, "color" : "green", "make" : "ford", "sold" : "2016-05-18", "condition": "new" }
{ "index": {}}
{ "price" : 15000, "color" : "blue", "make" : "toyota", "sold" : "2016-07-02", "condition": "good" }
{ "index": {}}
{ "price" : 12000, "color" : "green", "make" : "toyota", "sold" : "2016-08-19" , "condition": "good"}
{ "index": {}}
{ "price" : 18000, "color" : "red", "make" : "dodge", "sold" : "2016-11-05", "condition": "good" }
{ "index": {}}
{ "price" : 80000, "color" : "red", "make" : "bmw", "sold" : "2016-01-01", "condition": "new" }
{ "index": {}}
{ "price" : 25000, "color" : "blue", "make" : "ford", "sold" : "2016-08-22", "condition": "new" }
{ "index": {}}
{ "price" : 10000, "color" : "gray", "make" : "dodge", "sold" : "2016-02-12", "condition": "okay" }
{ "index": {}}
{ "price" : 19000, "color" : "red", "make" : "dodge", "sold" : "2016-02-12", "condition": "good" }
{ "index": {}}
{ "price" : 20000, "color" : "red", "make" : "chevrolet", "sold" : "2016-08-15", "condition": "good" }
{ "index": {}}
{ "price" : 13000, "color" : "gray", "make" : "chevrolet", "sold" : "2016-11-20", "condition": "okay" }
{ "index": {}}
{ "price" : 12500, "color" : "gray", "make" : "dodge", "sold" : "2016-03-09", "condition": "okay" }
{ "index": {}}
{ "price" : 35000, "color" : "red", "make" : "dodge", "sold" : "2016-04-10", "condition": "new" }
{ "index": {}}
{ "price" : 28000, "color" : "blue", "make" : "chevrolet", "sold" : "2016-08-15", "condition": "new" }
{ "index": {}}
{ "price" : 30000, "color" : "gray", "make" : "bmw", "sold" : "2016-11-20", "condition": "good" }
```

```
GET /vehicles/cars/_search
{
  "from": 0,
  "size": 5,

  "query": {
    "match_all":{}
  },

  "sort": [
    {"price": {"order": "desc"}}
    ]
}
```

```
  "hits": [
    {
      "_index": "vehicles",
      "_type": "cars",
      "_id": "AVyLItUrq0M1E3BufUu6",
      "_score": null,
      "_source": {
        "price": 80000,
        "color": "red",
        "make": "bmw",
        "sold": "2016-01-01",
        "condition": "new"
      },
      "sort": [
        80000
      ]
    },
    {
      "_index": "vehicles",
      "_type": "cars",
      "_id": "AVyLItUrq0M1E3BufUvB",
      "_score": null,
      "_source": {
        "price": 35000,
        "color": "red",
        "make": "dodge",
        "condition": "new"
      },
      "sort": [
```

Well there's a there's a mechanism for queries It's called the query and fetch.

- **How does the paginate and sorting process work?**
  - **Mechanism called the "query and fetch"**
- **In this order:**
  - **"query" is the query portion**
  - **"sort" sorts the displayed entries**
  - **"from" and "size" configure the amount and form of displayed hits**
- **Aggregations are about gaining insights and analytics from your data**
  - **This is achieved through summarization**

```
GET /vehicles/cars/_count
{
  "query": {
    "match":{"make": "dodge"}
  }

}
```

```
{
  "count": 5,
  "_shards": {
    "total": 5,
    "successful": 5,
    "failed": 0
  }
}
```

OK so five cars are manufactured by Dodge.

- **More powerful syntax for aggregations:**
  - **"aggs" (short for aggregation query)**

```
GET /vehicles/cars/_search
{

  "aggs": {
    "popular_cars": {
      "terms":{
        "field": "make.keyword"
      }

    }
  }

}
```

```
  "failed": 0
},
"hits": {...},
"aggregations": {
  "popular_cars": {
    "doc_count_error_upper_bound": 0,
    "sum_other_doc_count": 0,
    "buckets": [
      {
        "key": "dodge",
        "doc_count": 5
      },
      {
        "key": "chevrolet",
        "doc_count": 3
      },
      {
        "key": "bmw",
        "doc_count": 2
      },
      {
        "key": "ford",
        "doc_count": 2
      },
      {
        "key": "honda",
        "doc_count": 2
      },
      {
        "key": "toyota",
```

the inverted index so we can have sentences you know entire emails or blog posts into the text field.

- **We use "make.keyword" to ensure that Elasticsearch treats whatever text is within the text field as one word instead of multiple terms (which would be the case w/ just "make")**

```
GET /vehicles/cars/_search
{

  "aggs": {
    "popular_cars": {
      "terms":{
        "field": "make.keyword"
      },
      "aggs": {
        "avg_price": {
          "avg":{
            "field": "price"
          }
        },
        "max_price":{
            "max":{
            "field": "price"
          }
        }
      }

    }
  }

}
```

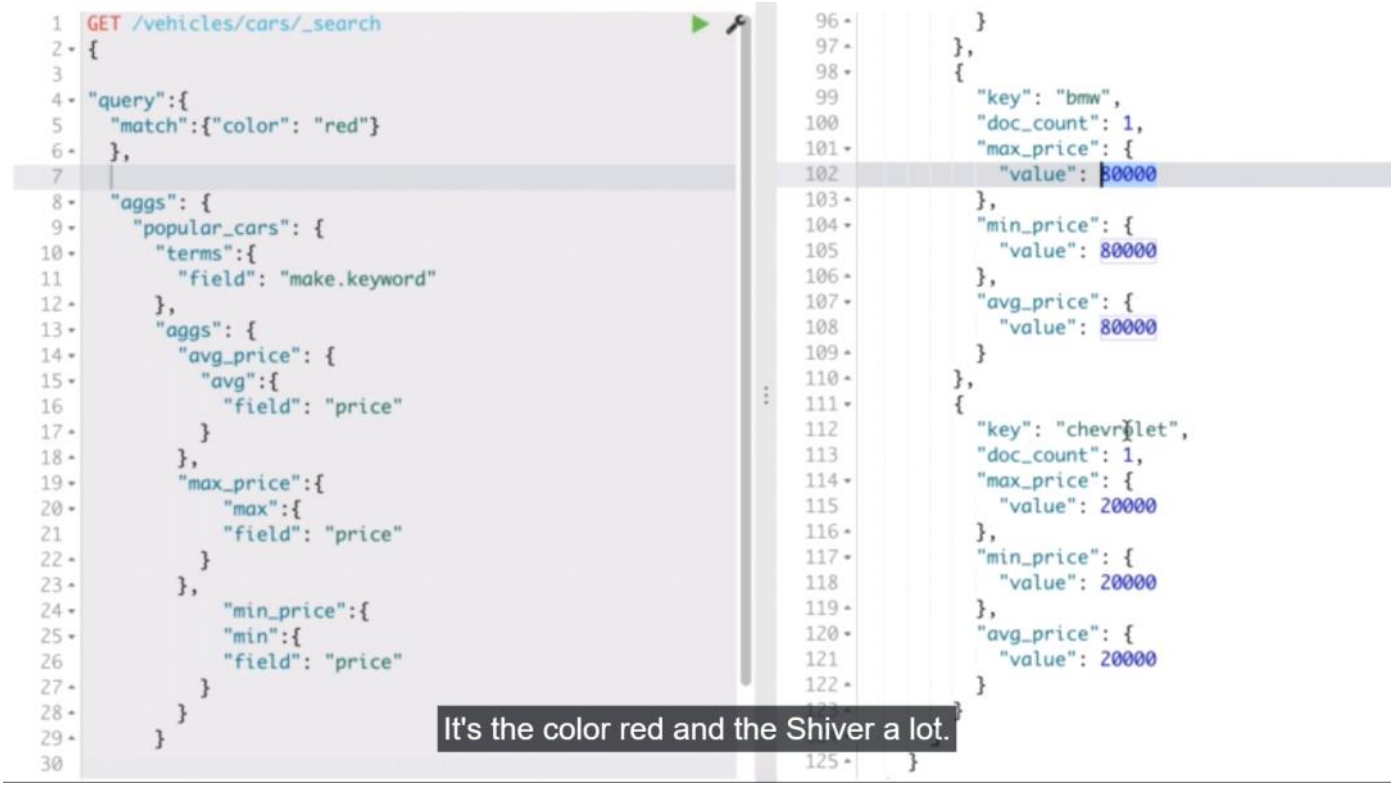
```
"popular_cars" : {
  "doc_count_error_upper_bound": 0,
  "sum_other_doc_count": 0,
  "buckets": [
    {
      "key": "dodge",
      "doc_count": 5,
      "max_price": {
        "value": 35000
      },
      "avg_price": {
        "value": 18900
      }
    },
    {
      "key": "chevrolet",
      "doc_count": 3,
      "max_price": {
        "value": 28000
      },
      "avg_price": {
        "value": 20333.333333333332
      }
    },
    {
      "key": "bmw",
      "max_price" : {
        "value": 80000
      },
```

That's the most expensive dodge that we have the most expensive Chevrolet we have is 28000 the most

- **You can use queries to limit the range of the aggregation and simplify data analysis**

```
GET /vehicles/cars/_search
{
```

```
    }
  },
```

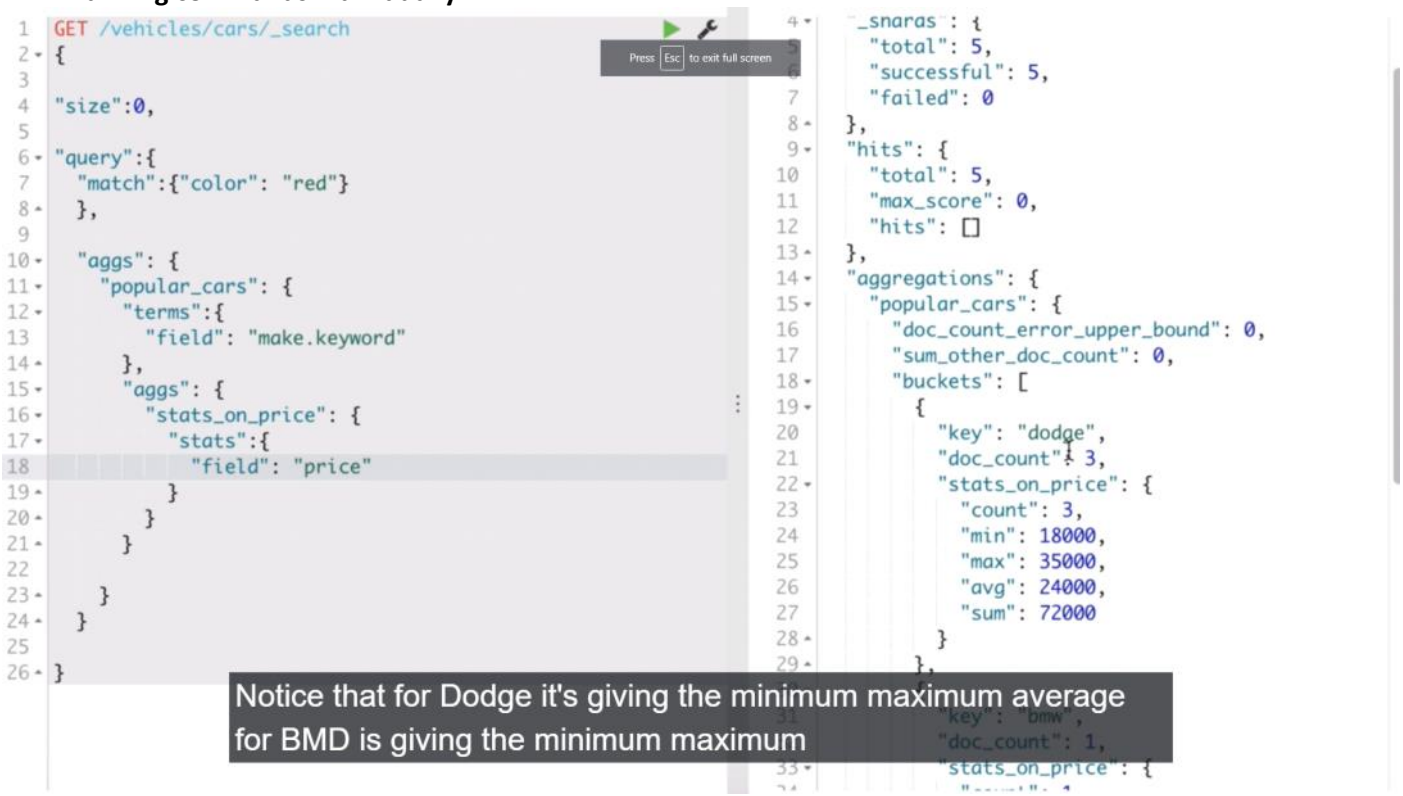


- **Running the aggregation will also show the documents in case you want to access the raw data via a query, filter, or for some other purpose**
- **To NOT display the individual documents/hits, either minimize them in the console OR set ["size": 0] to show zero docs**

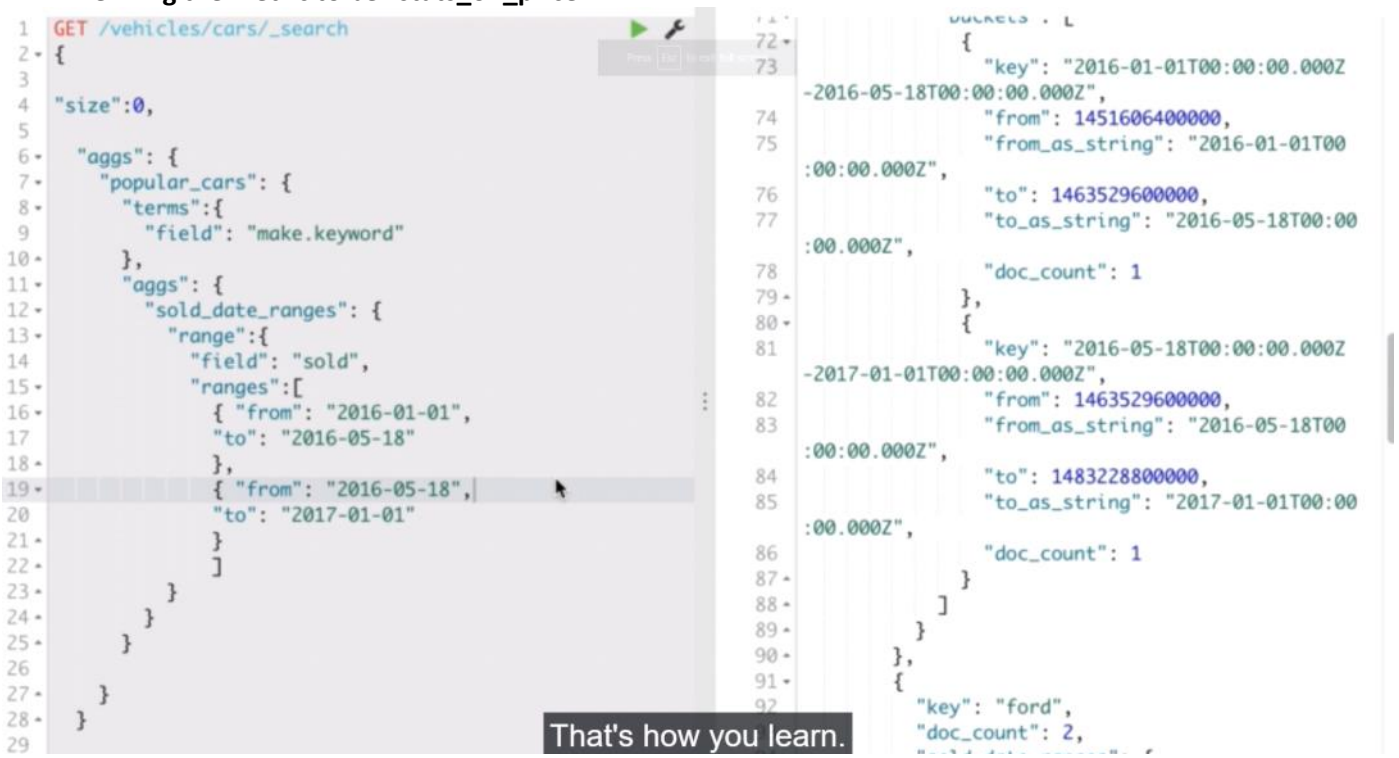
Lecture 14: Aggregations DSL (Part 2)

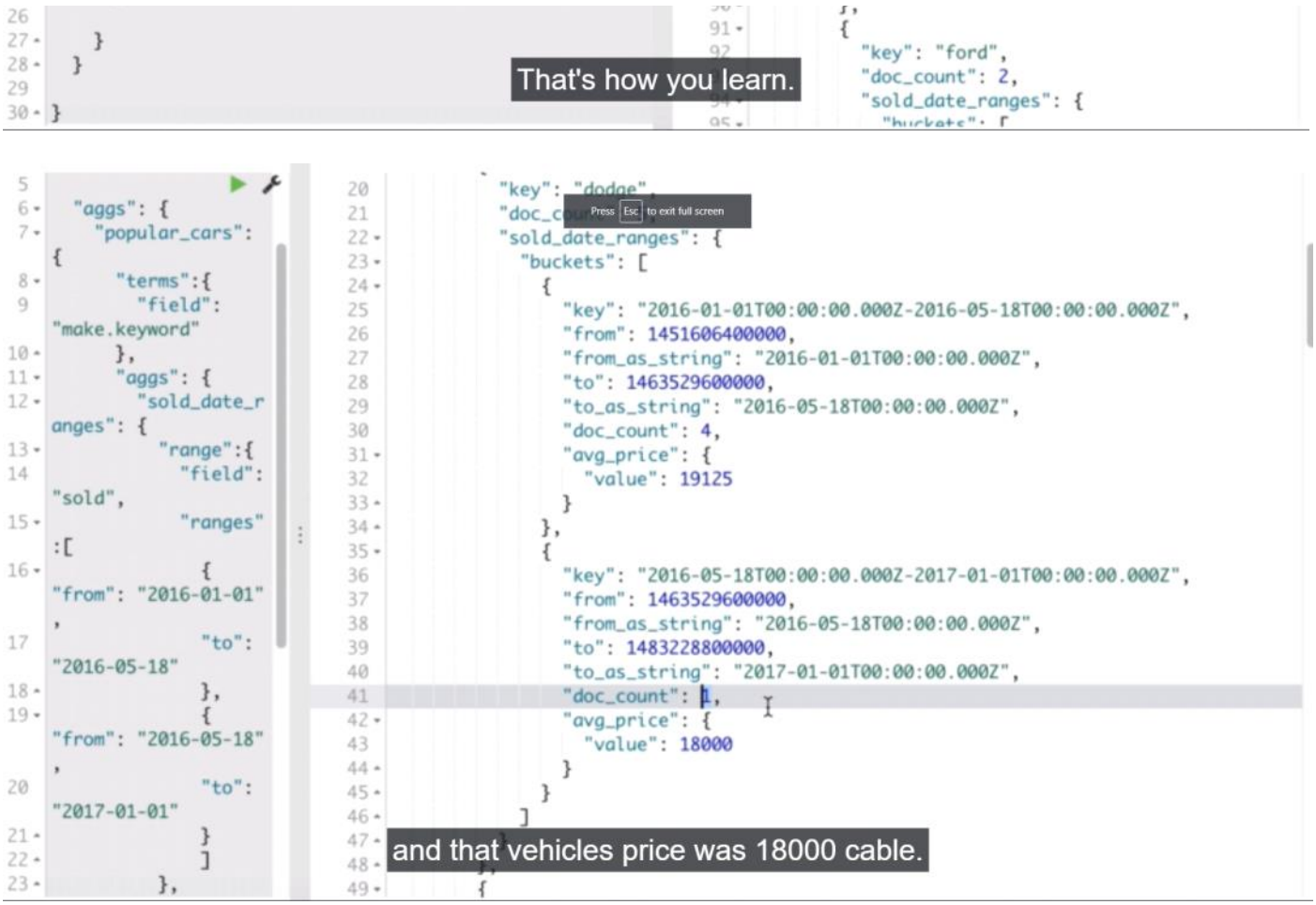
Tuesday, June 11, 2019 3:24 PM

- **For getting all of the statistics such as avg, min, max, sum, etc., we can use stats instead of running commands individually**

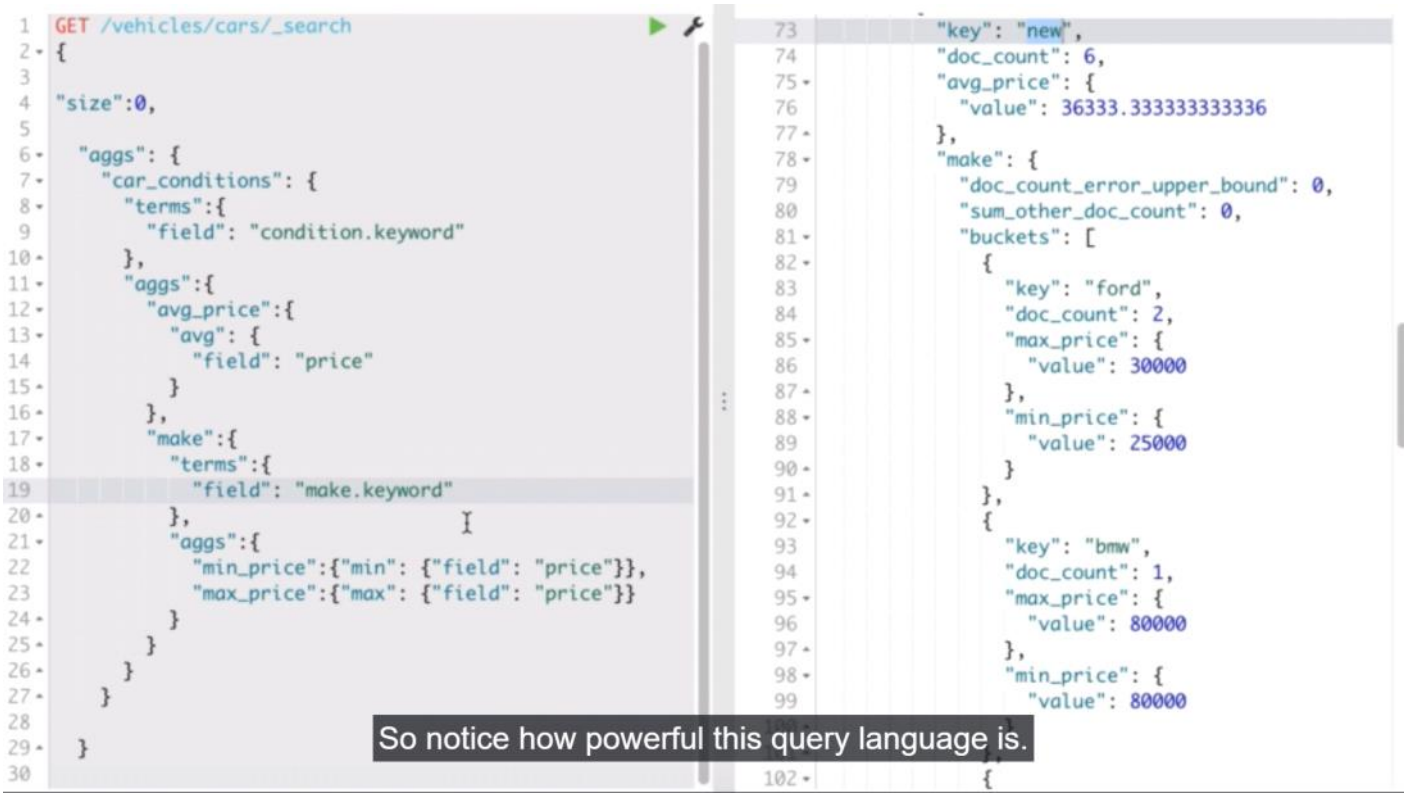


- **Structure known as buckets and metrics**
- **Creating a bucket called "popular_cars"**
- **Defining the metric to be "stats_on_price"**




That's how you learn.
and that vehicles price was 18000 cable.


GET /vehicles/cars/_search
We've got six new vehicles the average price is thirty six thousand three hundred thirty three.

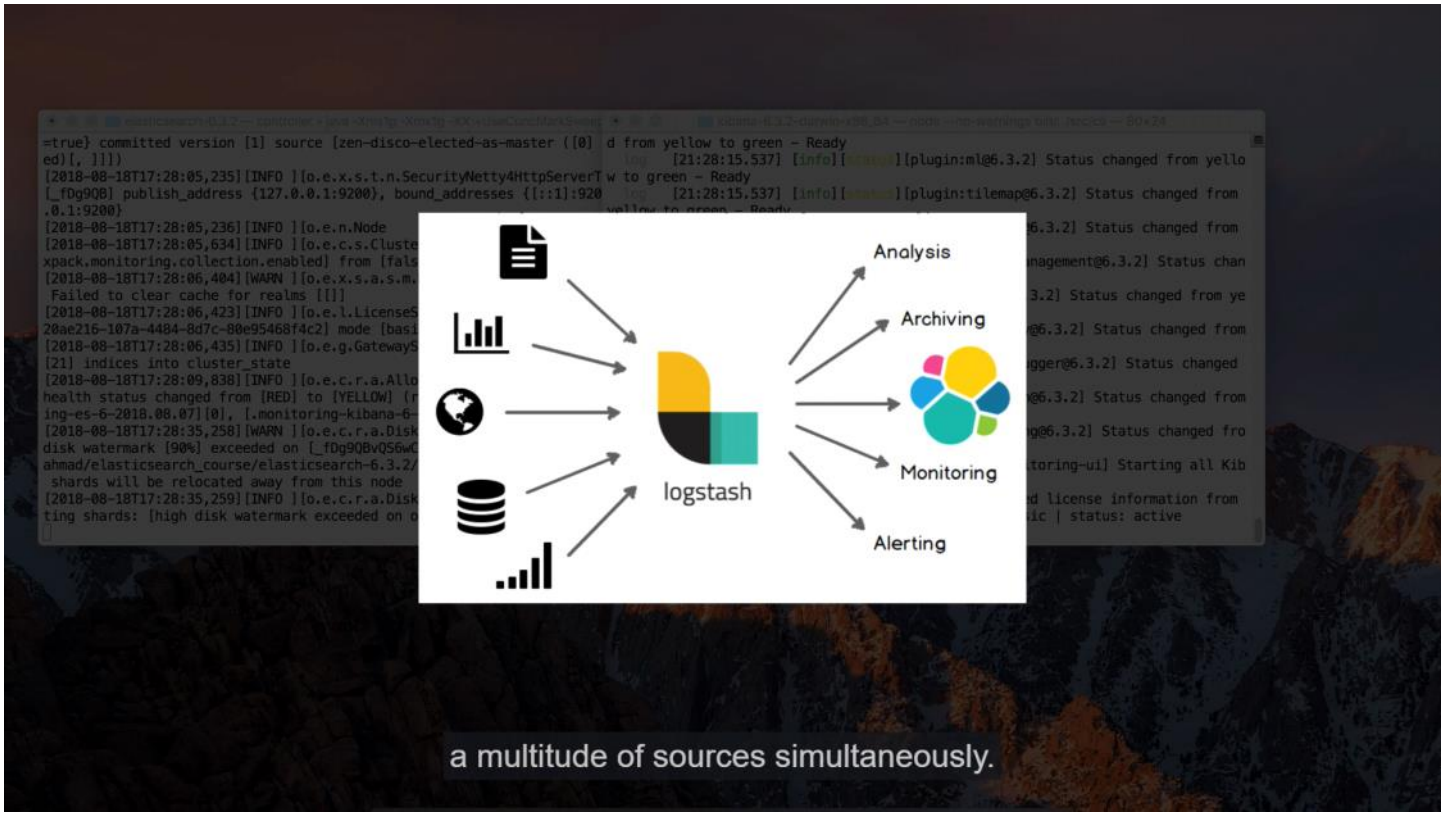


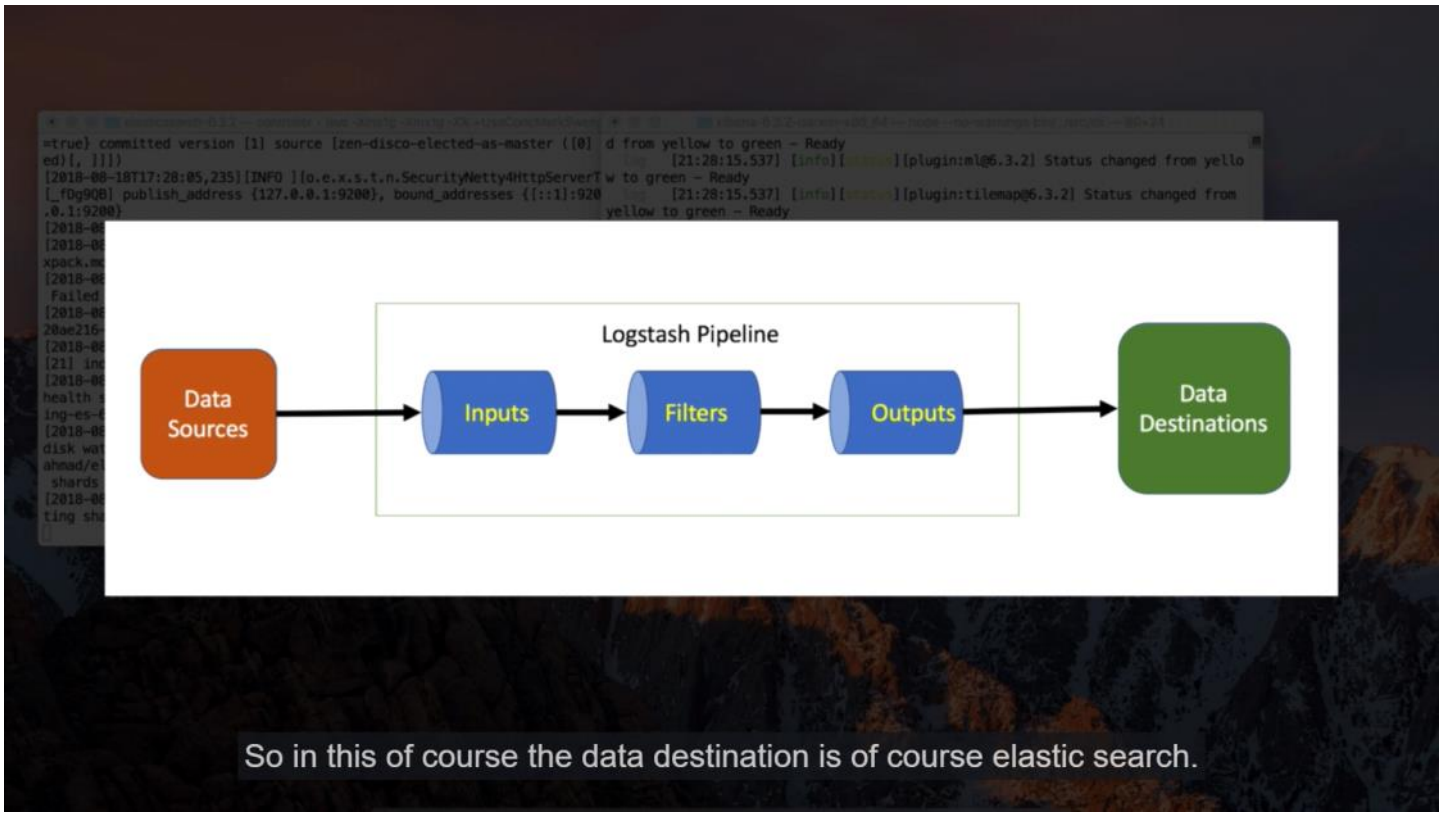
- **Concurrently running a query and an aggregation based off that query is known as <u>aggregation scoping</u> and can be done in real time (much faster than current relational databases)**

Lecture 15: Download and Configure Logstash

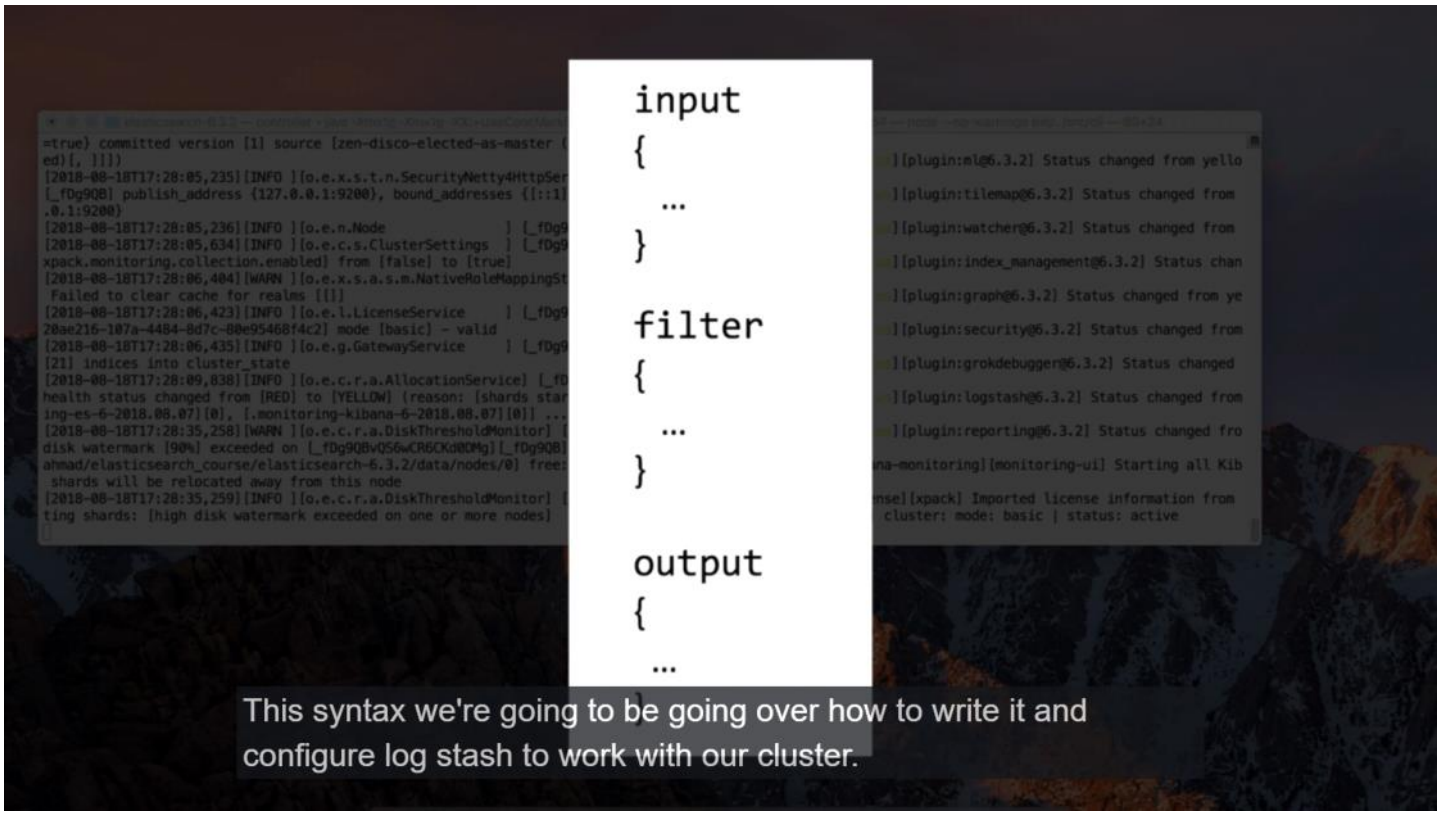
Wednesday, July 10, 2019 8:15 AM

- **Logstash is a very powerful open source data processing pipeline engine and it's used to ingest data from a multitude of sources simultaneously**

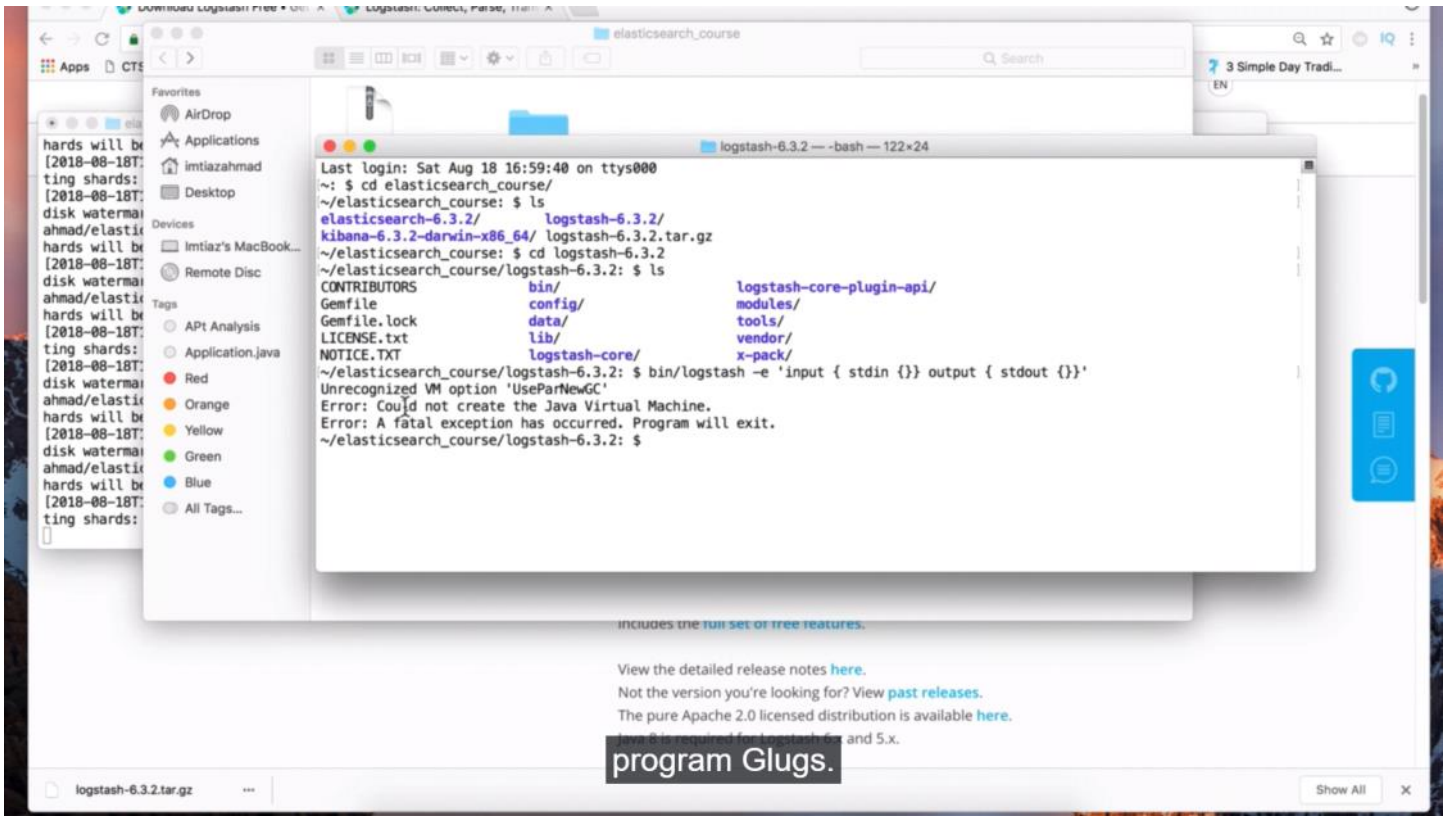




- **Configured in the JRuby language**

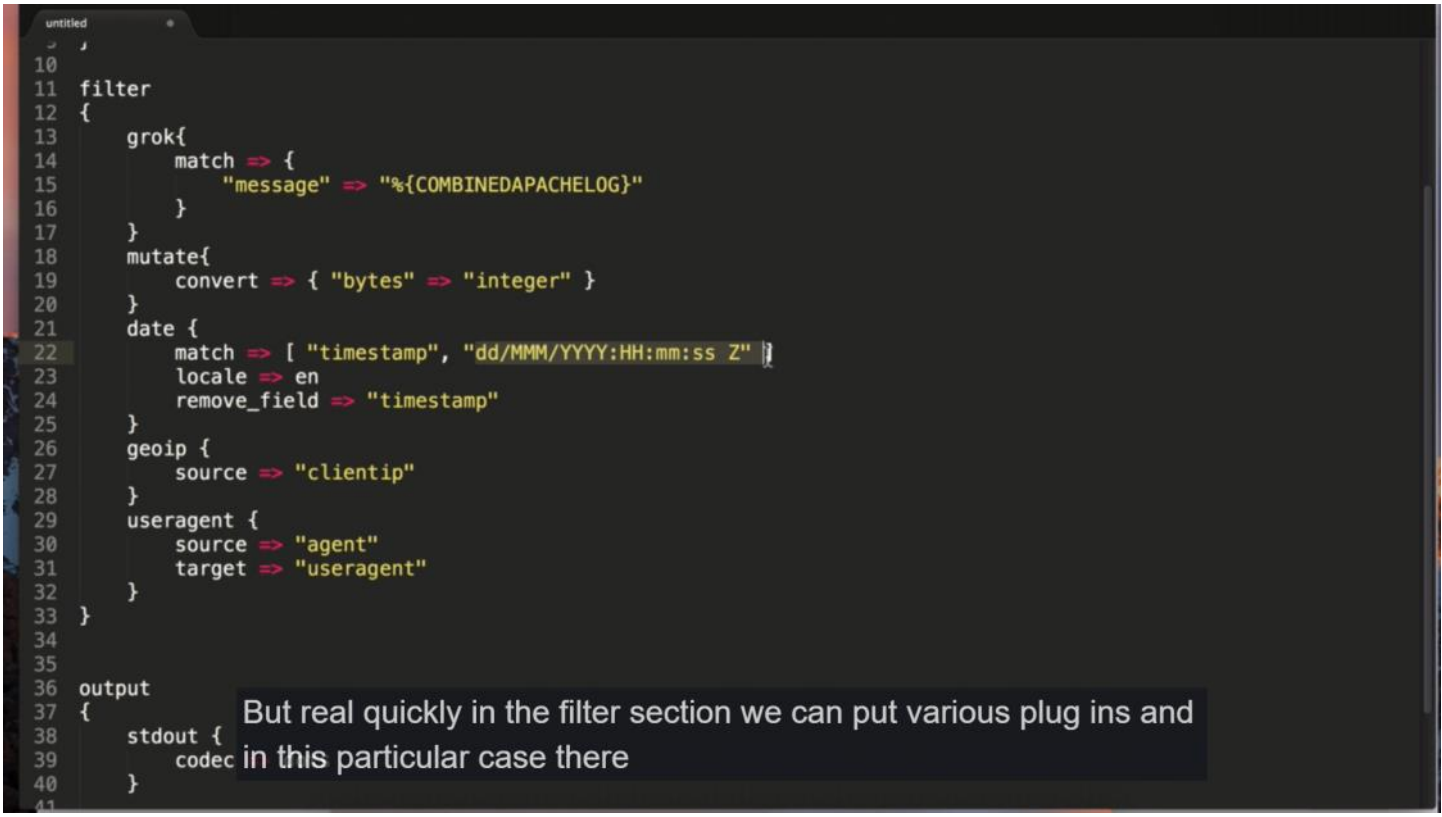


- **To test logstash:**

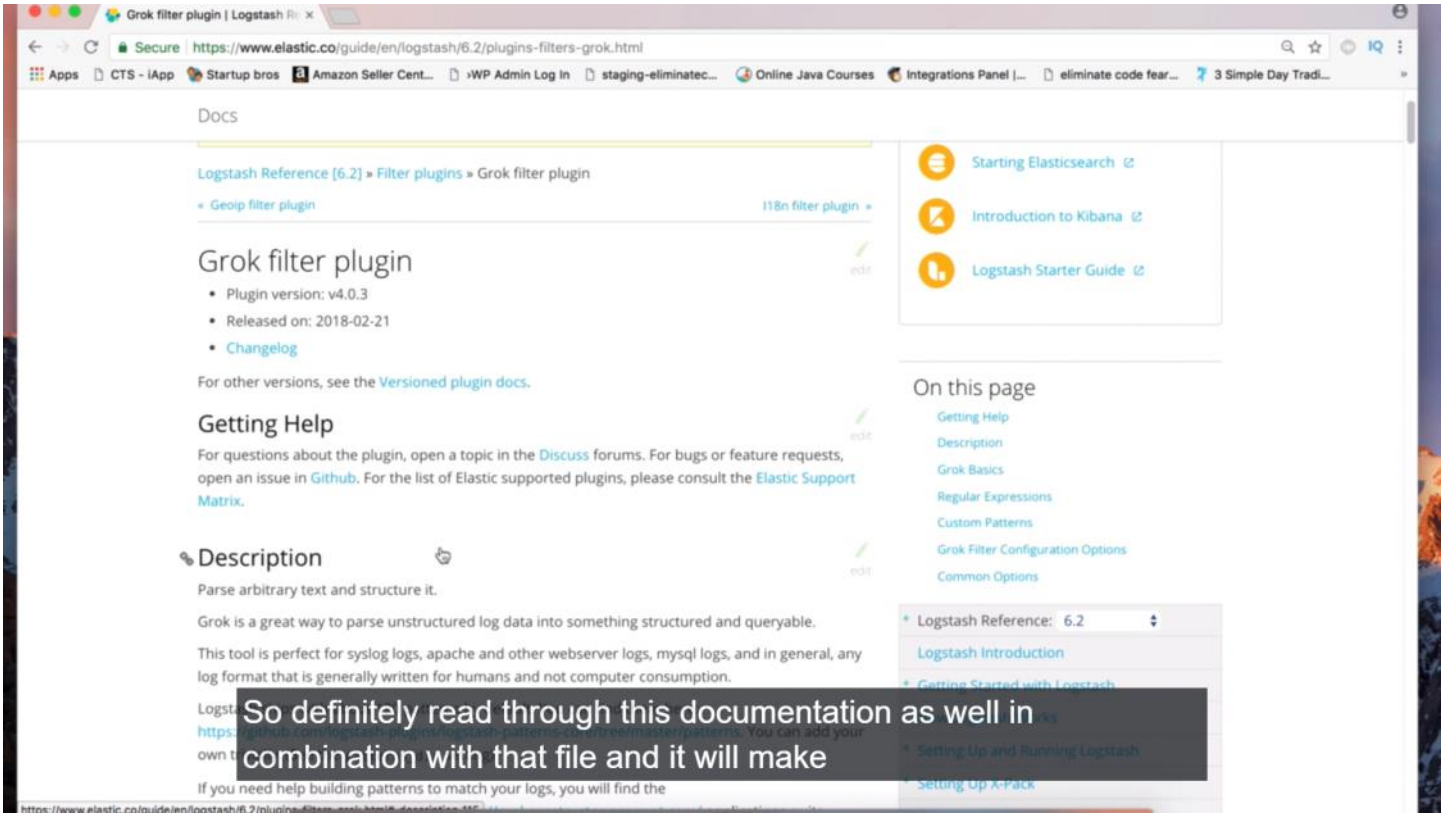


Lecture 16: Overview of Logstash & Indexing Apache Logs

Wednesday, July 10, 2019 8:59 AM



- **In configuration file, filter section, we can instantiate multiple plugins**
  - **grok, mutate, date, geoip, useragent**
  - **Grok - regex matcher (built-in vars that are predefined (APACHELOG))**
    - **https://github.com/elastic/logstash/blob/v1.4.2/patterns/grok-patterns**
  - **Mutate - convert data types**
  - **Date - formats dates from the log into a certain format specified**
  - **Geoip - finds IP address from the log**
  - **Useragent - stores whether the user is using a phone or tablet or PC**

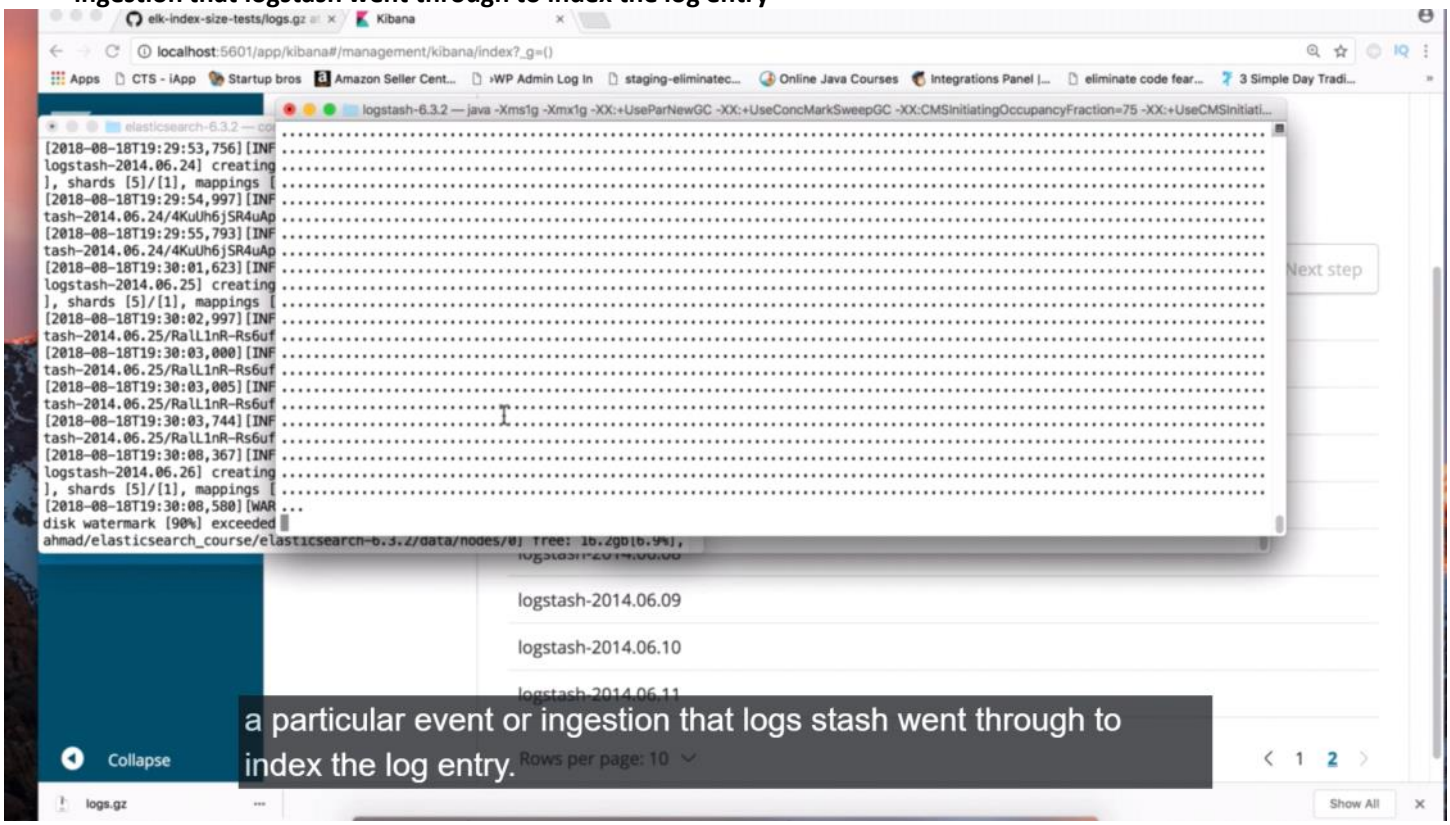


- **Change path to "\Users\A647655\Downloads\data\logs\logs"**

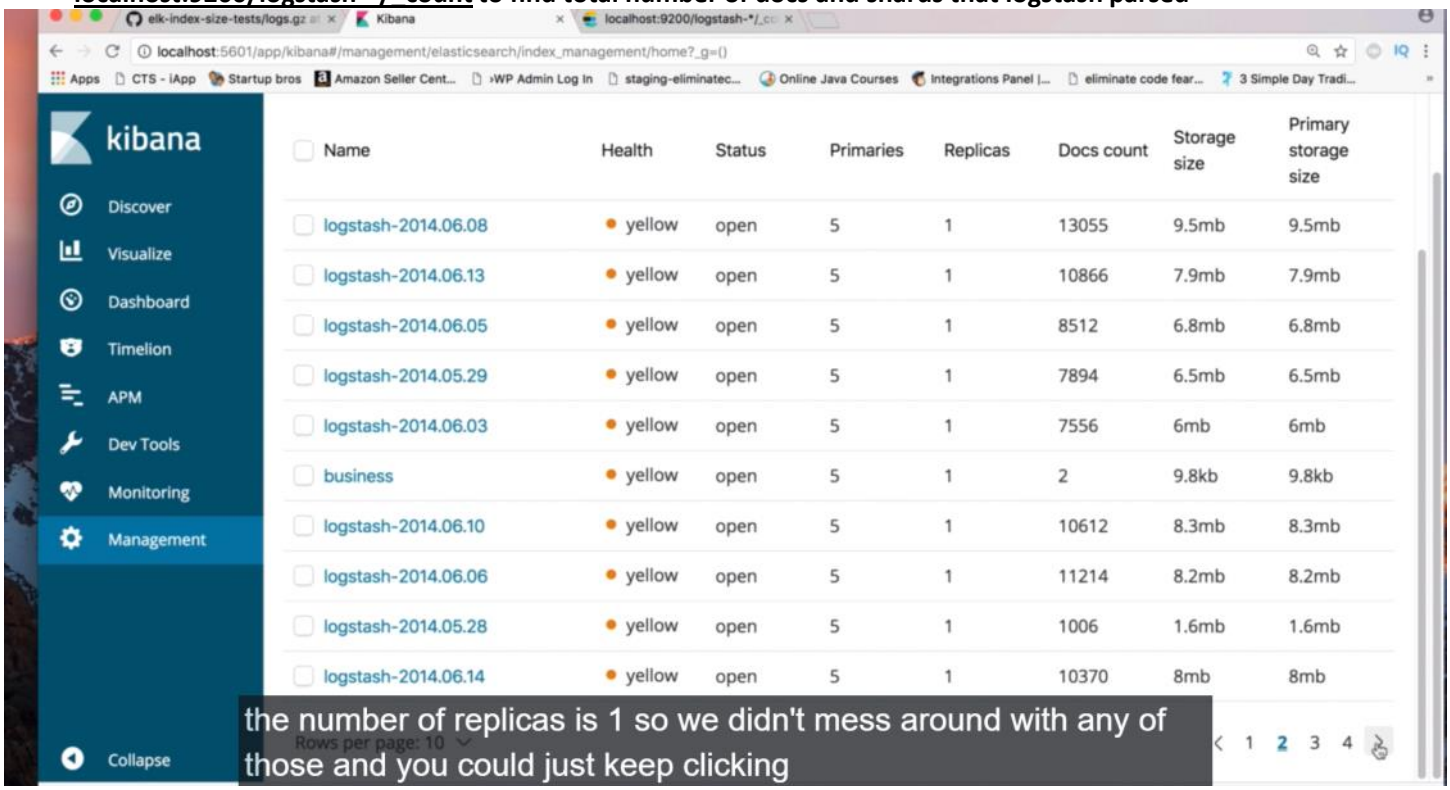
Lecture 17: Kibana Visualizations and Dashboards Overview

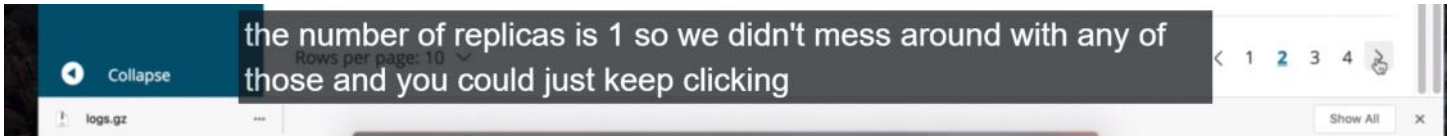
Wednesday, July 10, 2019 9:52 AM

- **After running the conf file, each dot that appears after basically represents a particular event or ingestion that logstash went through to index the log entry**

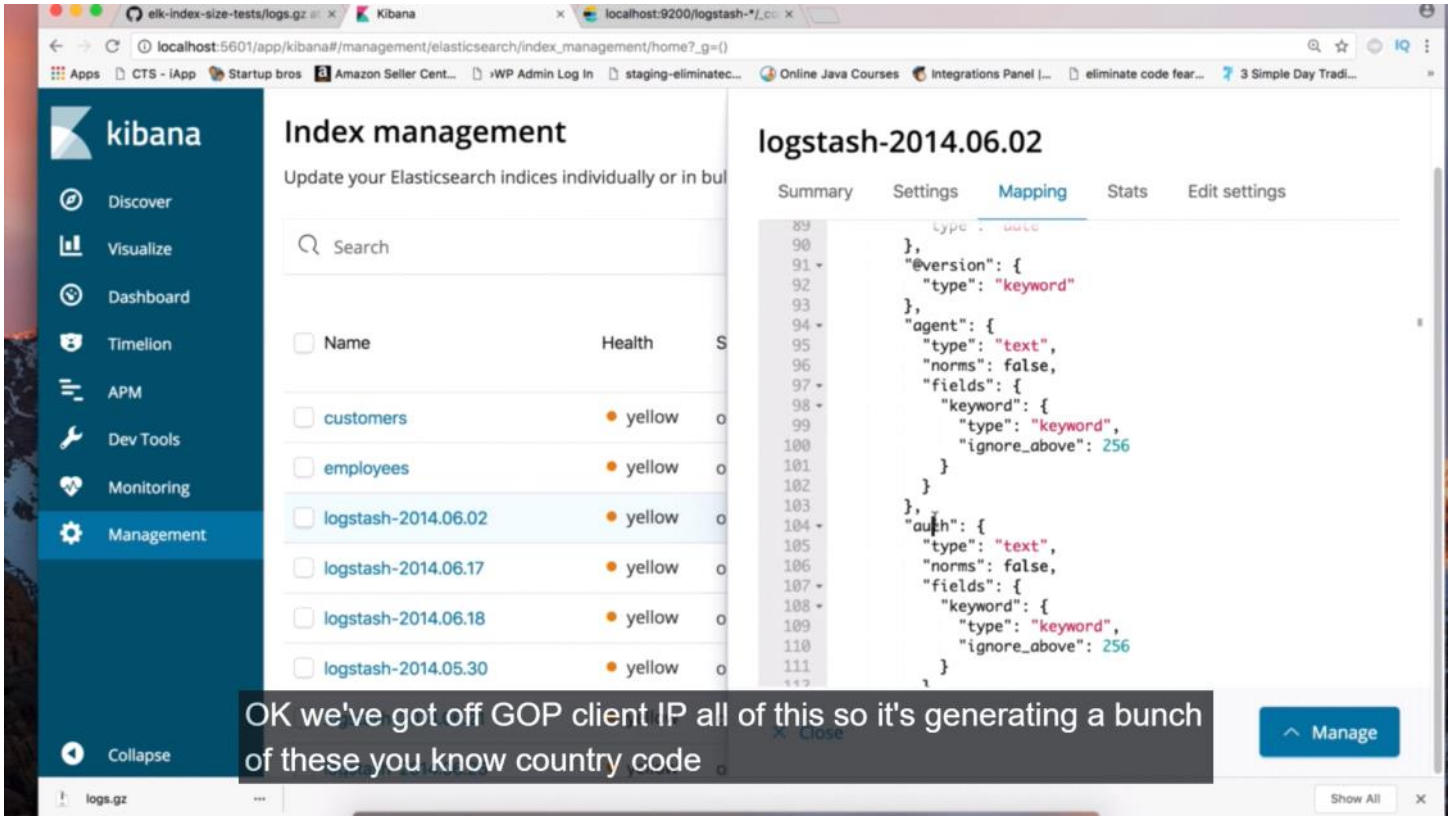


- **localhost:9200/logstash-*/_count to find total number of docs and shards that logstash parsed**

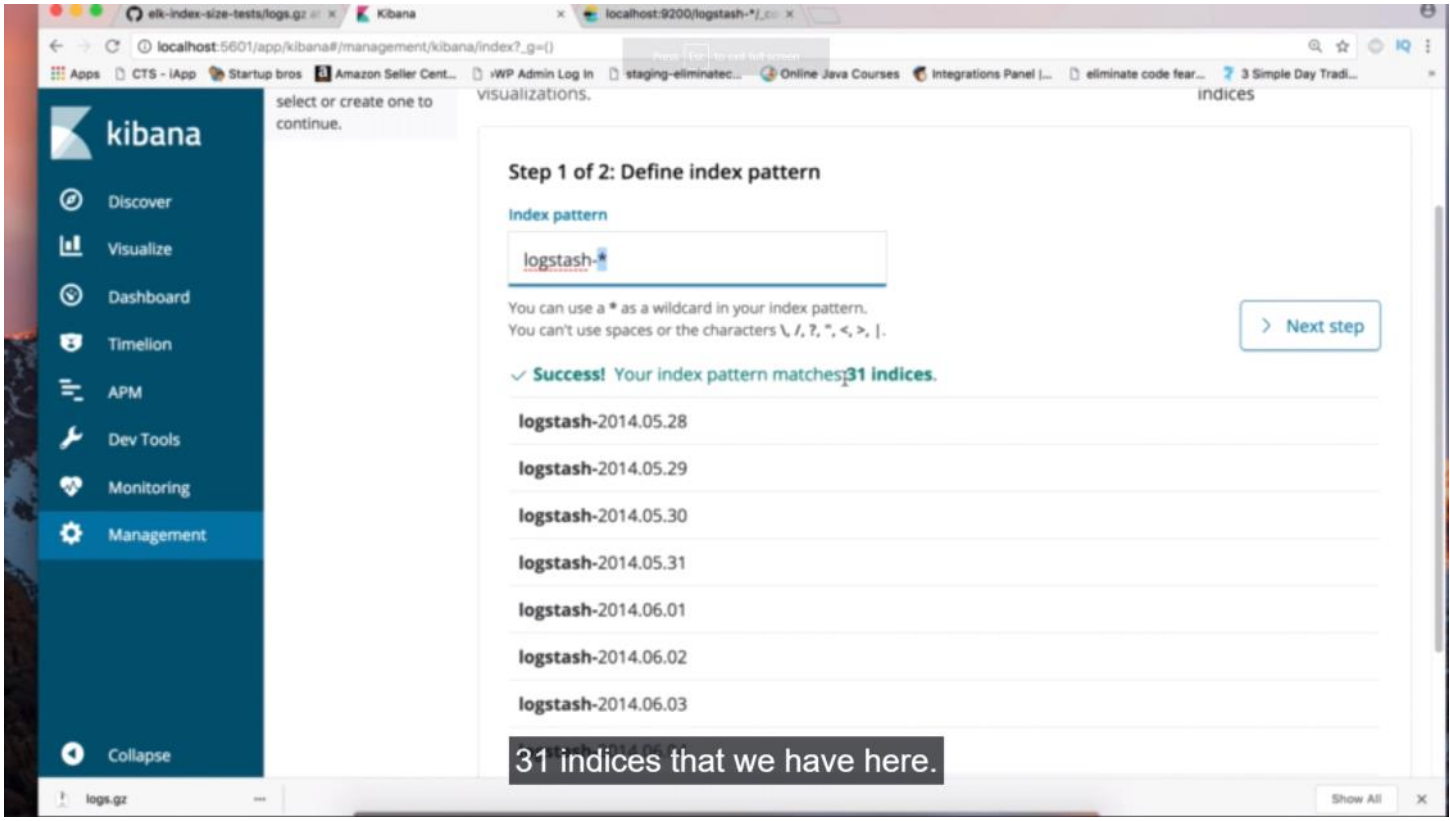


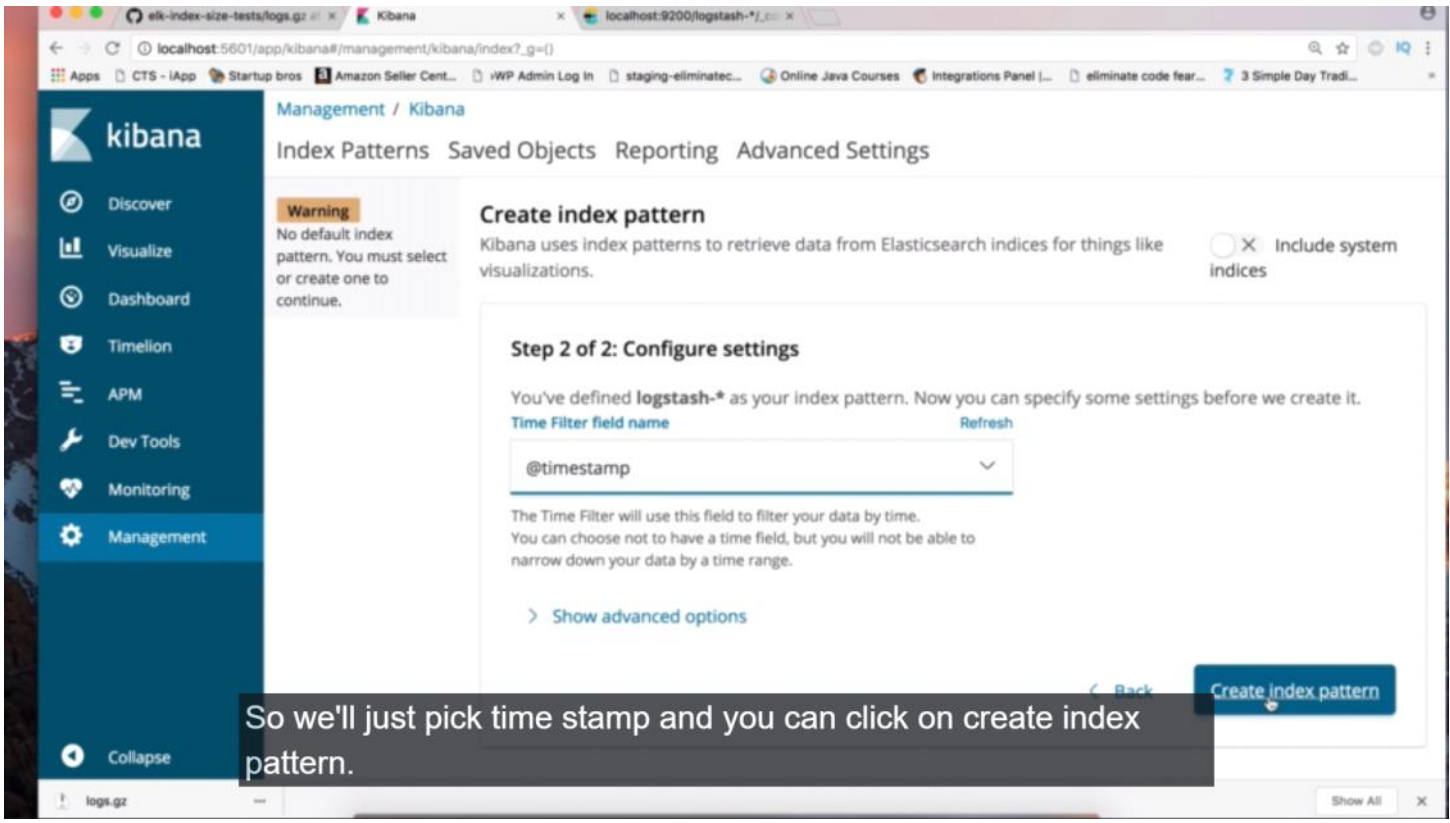


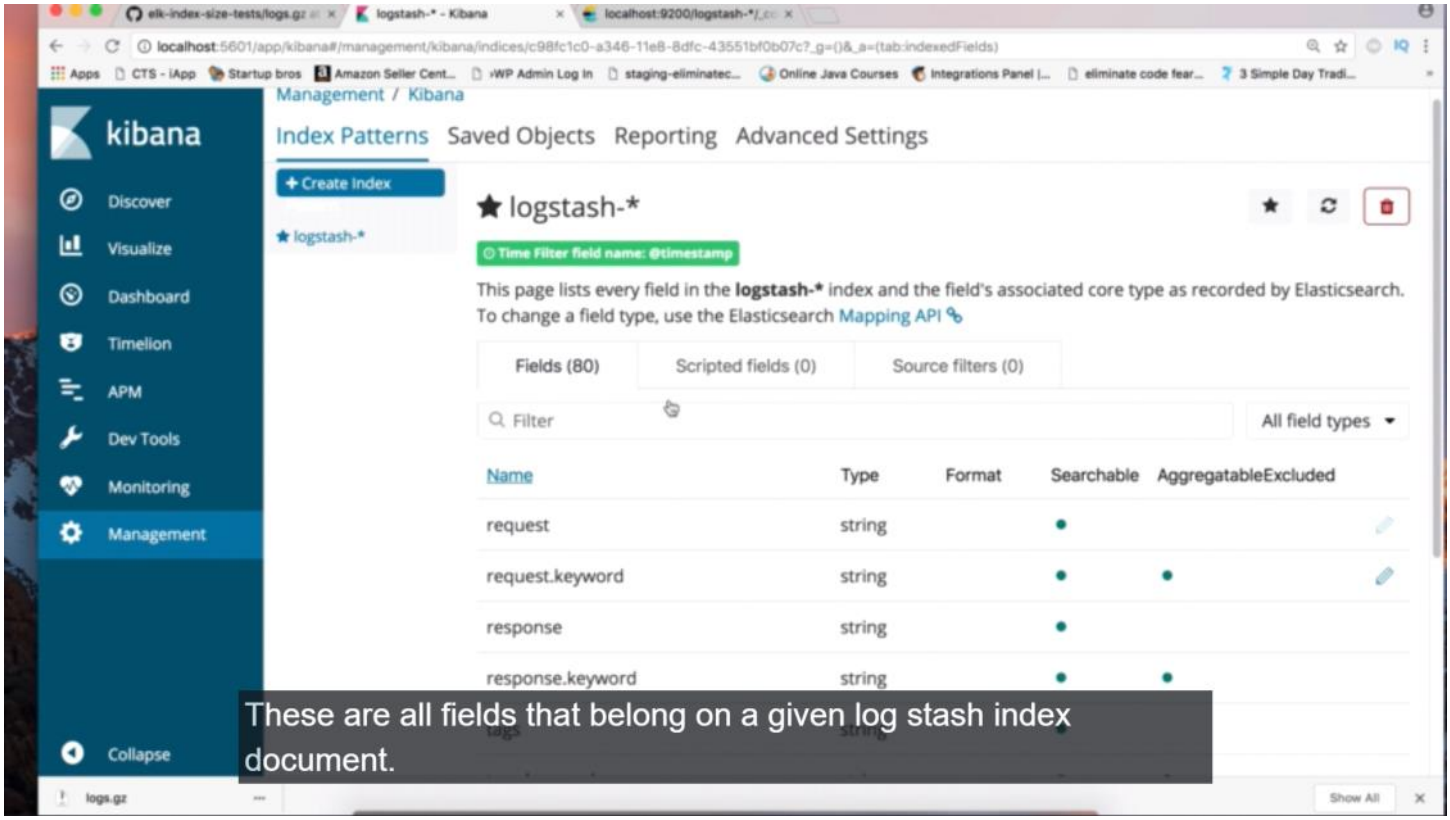
- **By default the number of primaries is 5 and the number of replicas is 1**

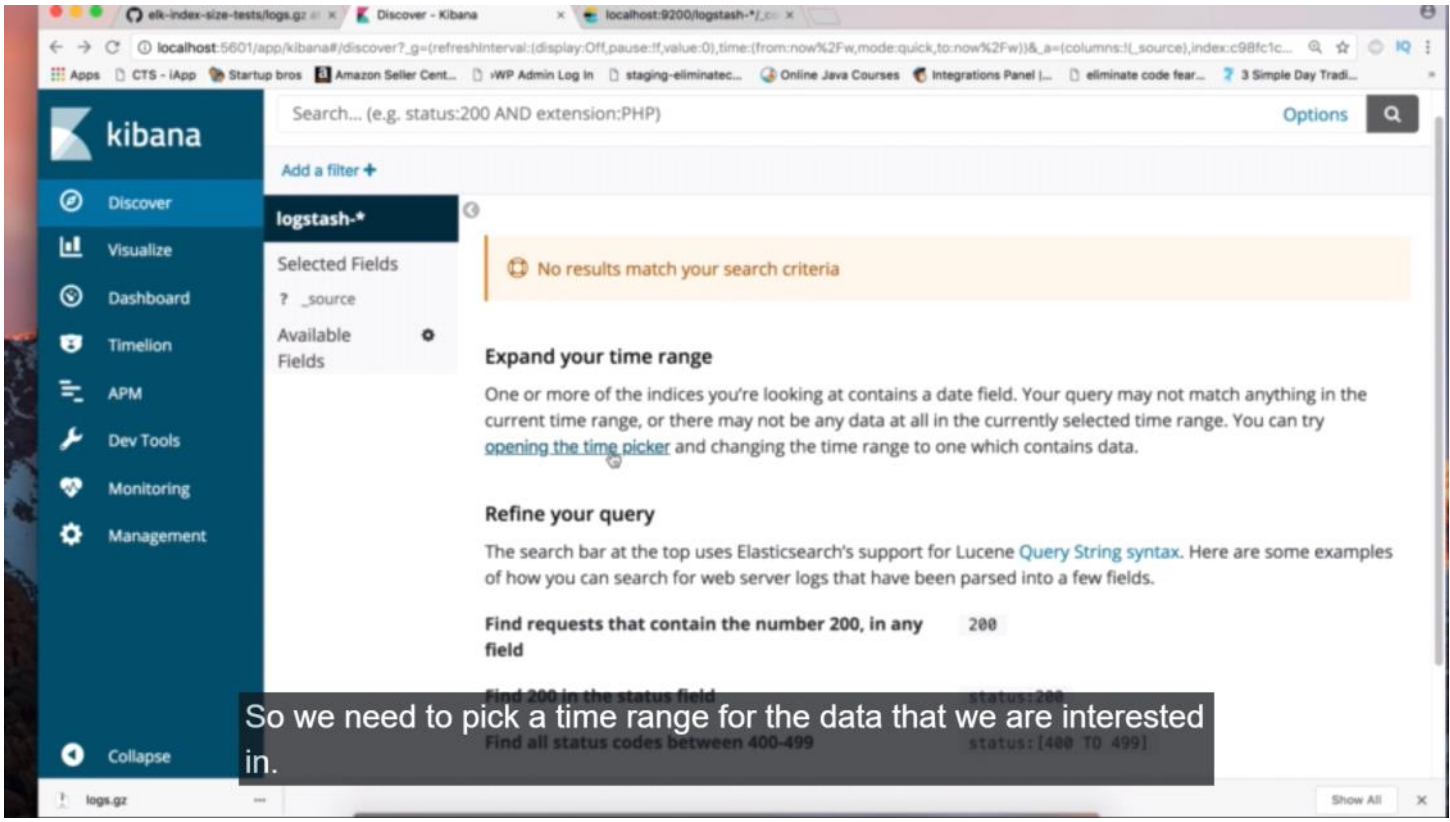


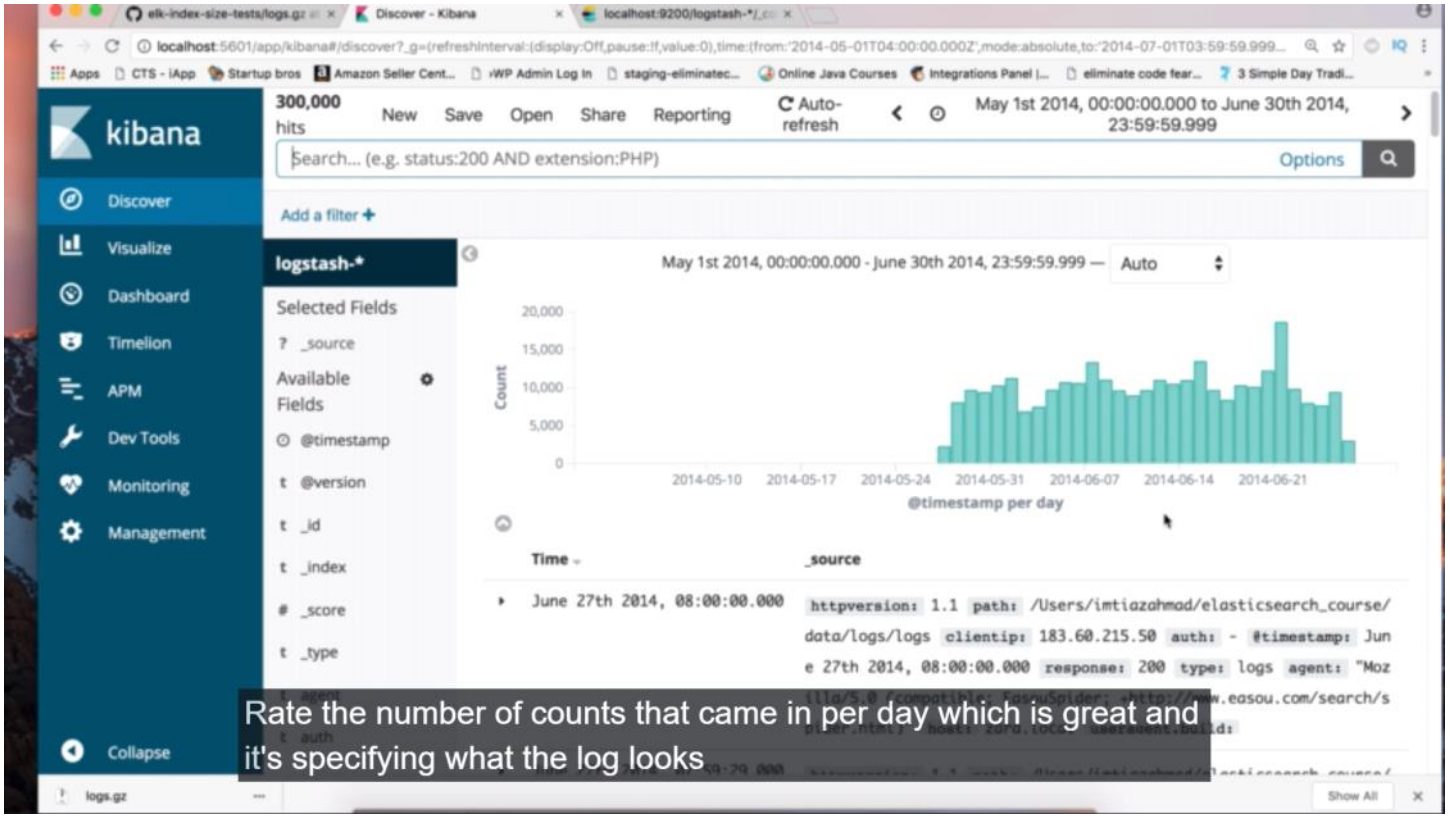
- **You must define an <u>index pattern</u> and choose an option for <u>time filter</u> in order to create your index pattern for visualizations and dashboard generation**

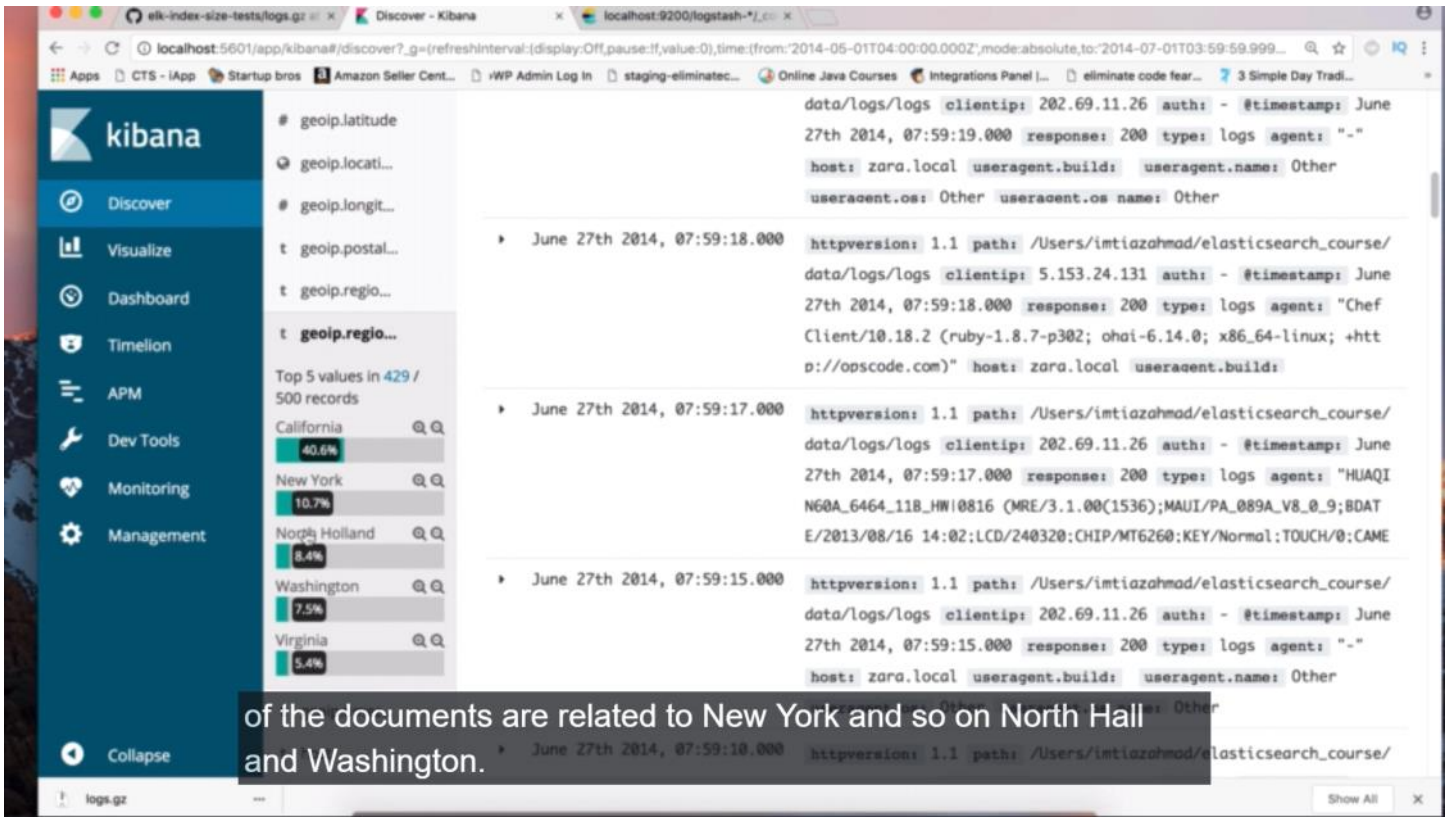

kibana
Discover
Visualize
Dashboard
Timelion
APM
Dev Tools
Monitoring
Management
Collapse
Step 1 of 2: Define index pattern
Index pattern
logstash-*
You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, ", <, >, |.
Next step
Success! Your index pattern matches 31 indices.
logstash-2014.05.28
logstash-2014.05.29
logstash-2014.05.30
logstash-2014.05.31
logstash-2014.06.01
logstash-2014.06.02
logstash-2014.06.03
31 indices that we have here.


kibana
Management / Kibana
Index Patterns
Saved Objects
Reporting
Advanced Settings
Warning
No default index pattern. You must select or create one to continue.
Create index pattern
Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.
Include system indices
Step 2 of 2: Configure settings
You've defined logstash-* as your index pattern. Now you can specify some settings before we create it.
Time Filter field name
Refresh
@timestamp
The Time Filter will use this field to filter your data by time.
You can choose not to have a time field, but you will not be able to narrow down your data by a time range.
Show advanced options
Back
Create index pattern
So we'll just pick time stamp and you can click on create index pattern.

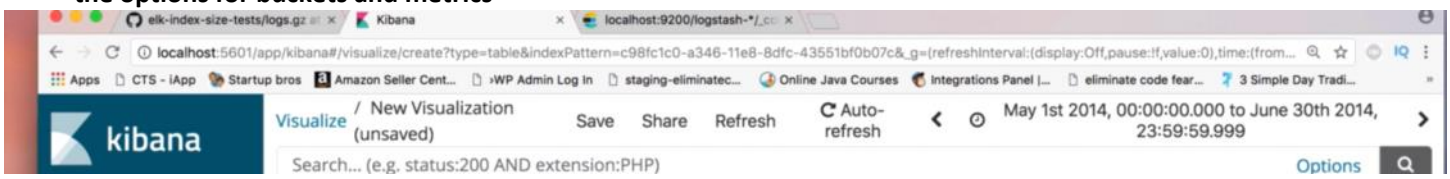

kibana
Discover
Visualize
Dashboard
Timelion
APM
Dev Tools
Monitoring
Management
Collapse
Management / Kibana
Index Patterns
Saved Objects
Reporting
Advanced Settings
+ Create Index
logstash-*
Time Filter field name: @timestamp
This page lists every field in the logstash-* index and the field's associated core type as recorded by Elasticsearch.
To change a field type, use the Elasticsearch Mapping API
Fields (80)
Scripted fields (0)
Source filters (0)
Filter
All field types
Name
Type
Format
Searchable
AggregatableExcluded
request
string
request.keyword
string
response
string
response.keyword
string
These are all fields that belong on a given log stash index document.

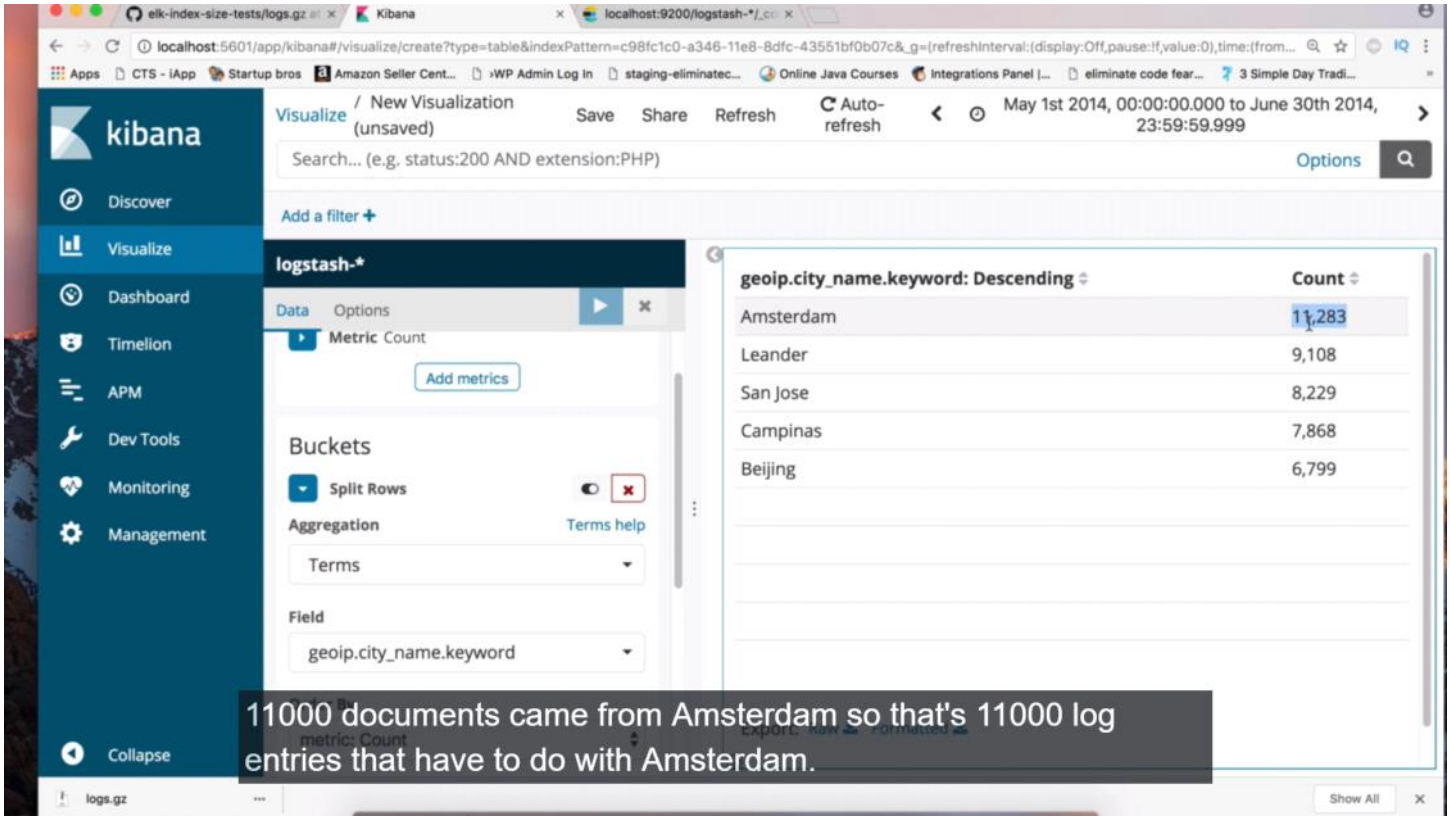

kibana
Discover
Visualize
Dashboard
Timelion
APM
Dev Tools
Monitoring
Management
Collapse
Search... (e.g. status:200 AND extension:PHP)
Options
Add a filter +
logstash-*
Selected Fields
_source
Available Fields
No results match your search criteria
Expand your time range
One or more of the indices you're looking at contains a date field. Your query may not match anything in the current time range, or there may not be any data at all in the currently selected time range. You can try opening the time picker and changing the time range to one which contains data.
Refine your query
The search bar at the top uses Elasticsearch's support for Lucene Query String syntax. Here are some examples of how you can search for web server logs that have been parsed into a few fields.
Find requests that contain the number 200, in any field
200
Find all status codes between 400-499
status:[400 TO 499]
So we need to pick a time range for the data that we are interested in.

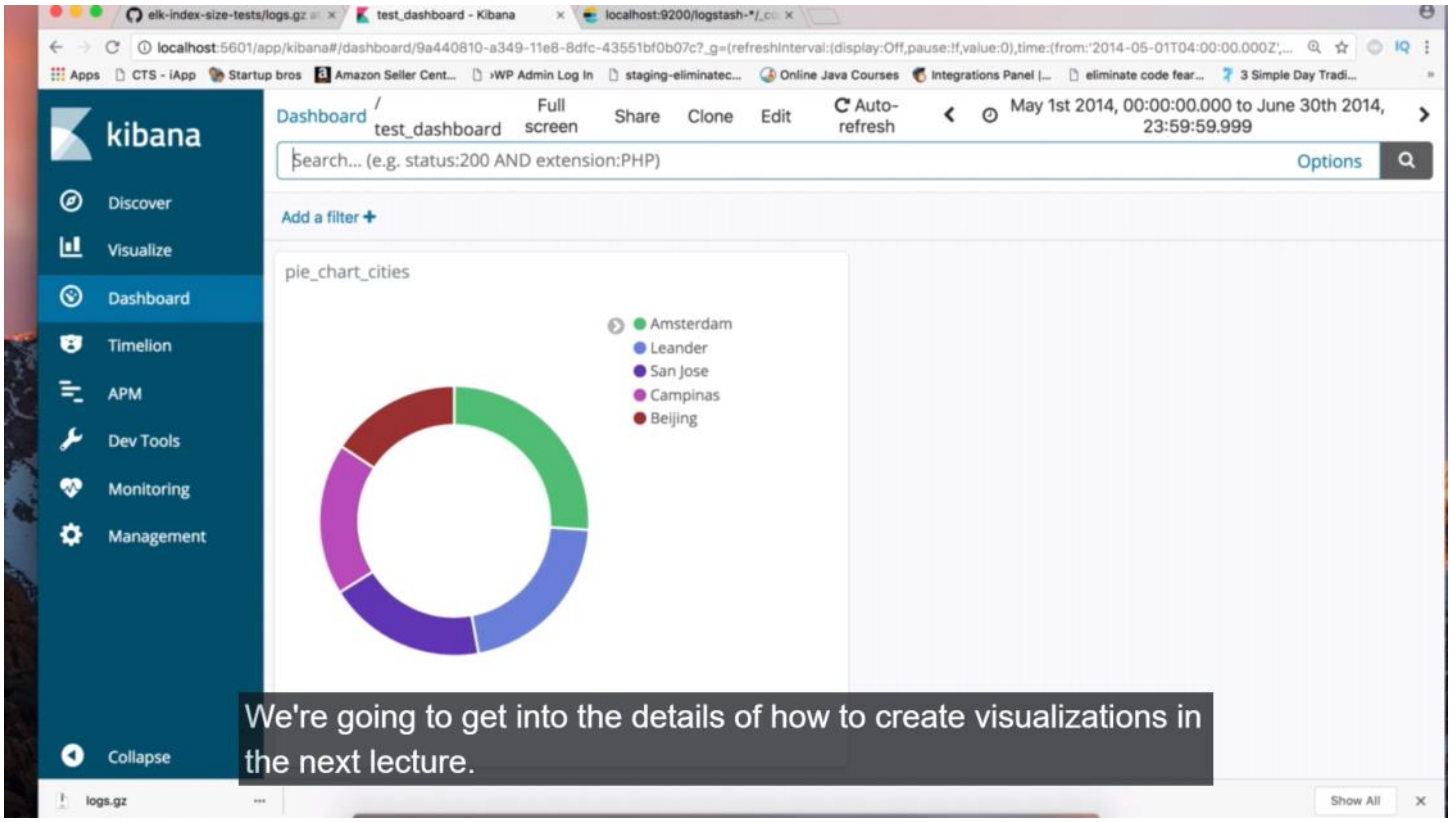




- **Creating a <u>data table</u> is as easy as clicking "Visualize", choosing our index pattern, and filling in the options for buckets and metrics**





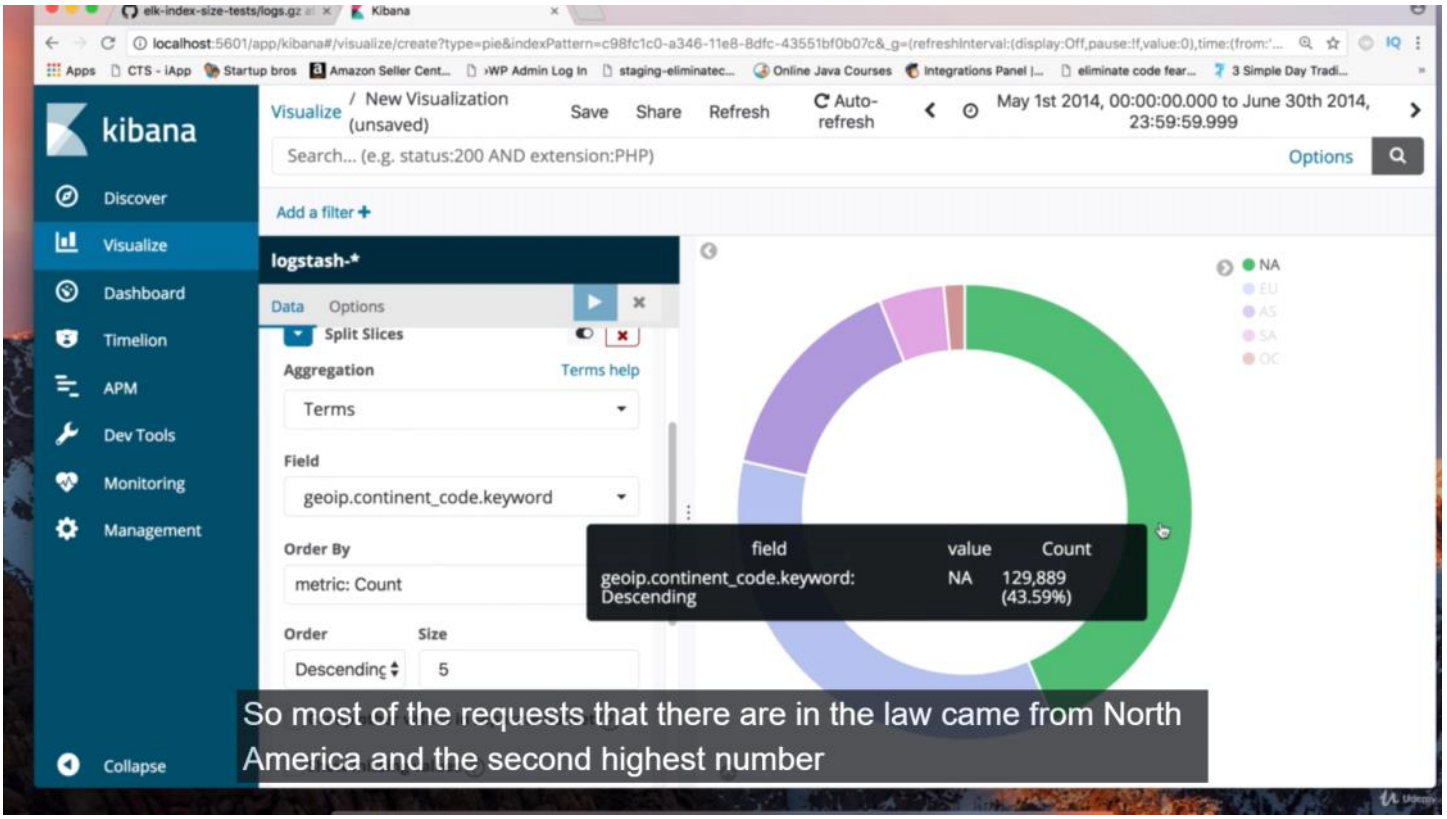


- **Visualizations, dashboards, and more!**

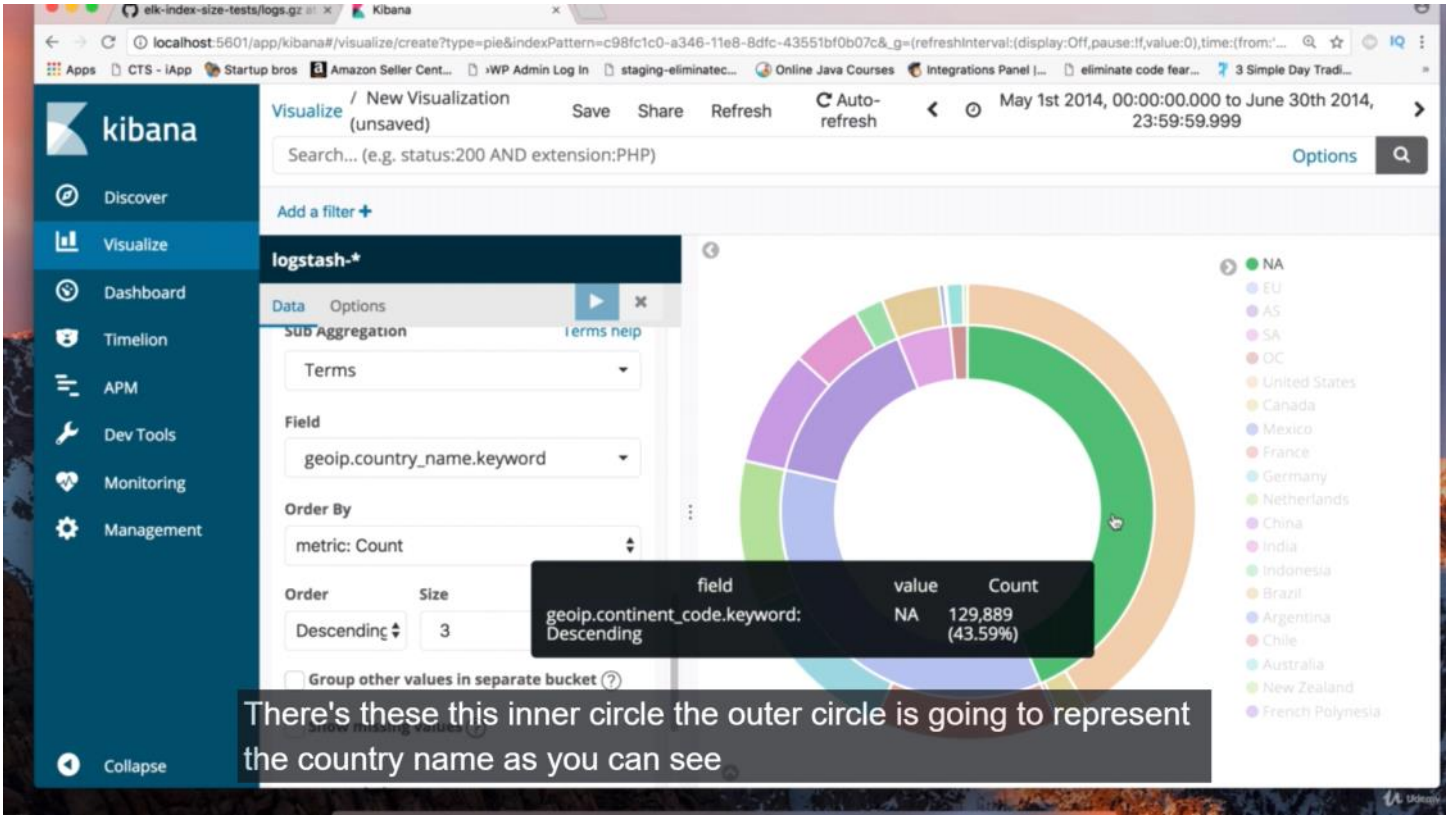
Lecture 18: More on Visualizations and Dashboards

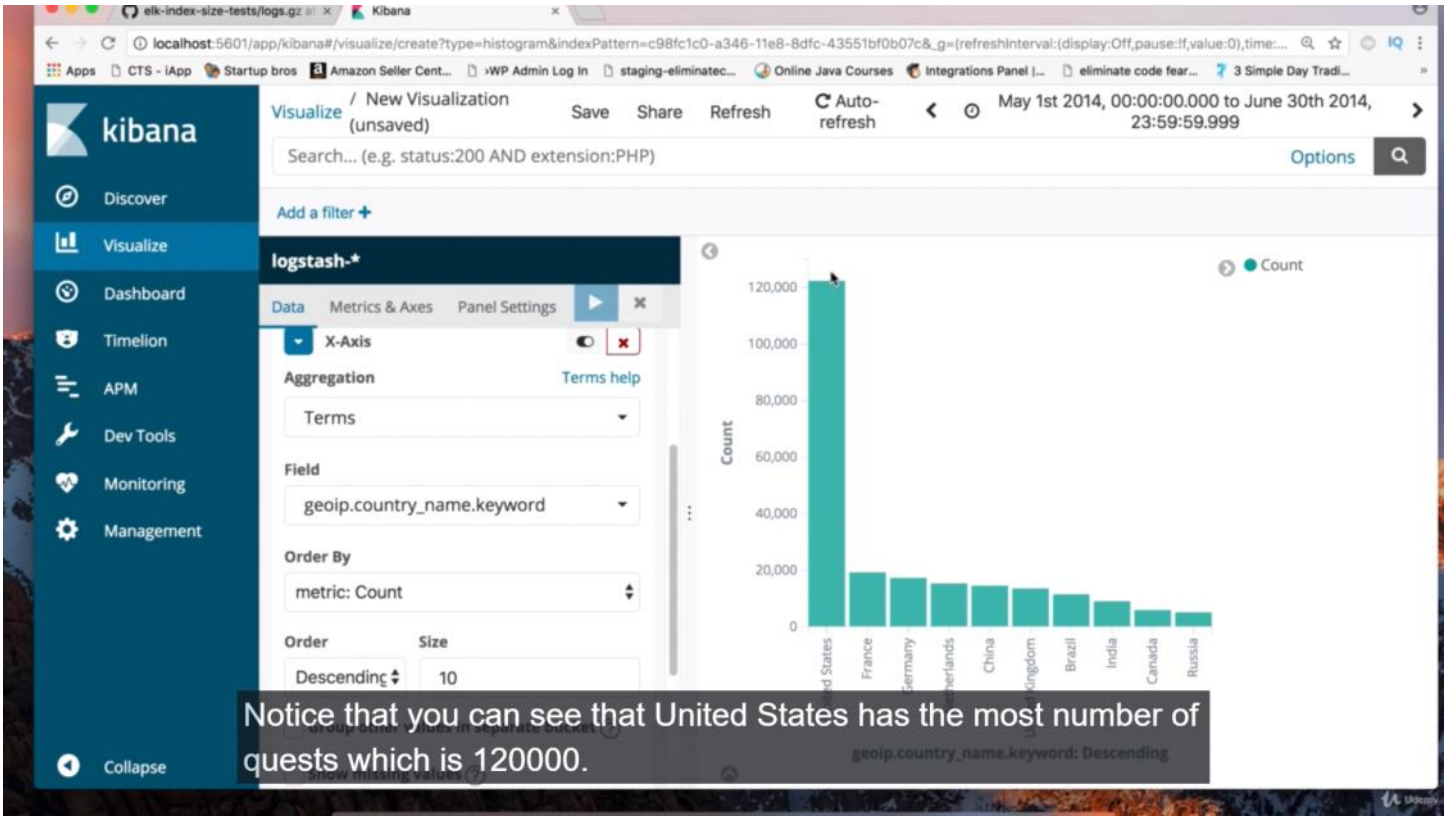
Wednesday, July 10, 2019 10:35 AM

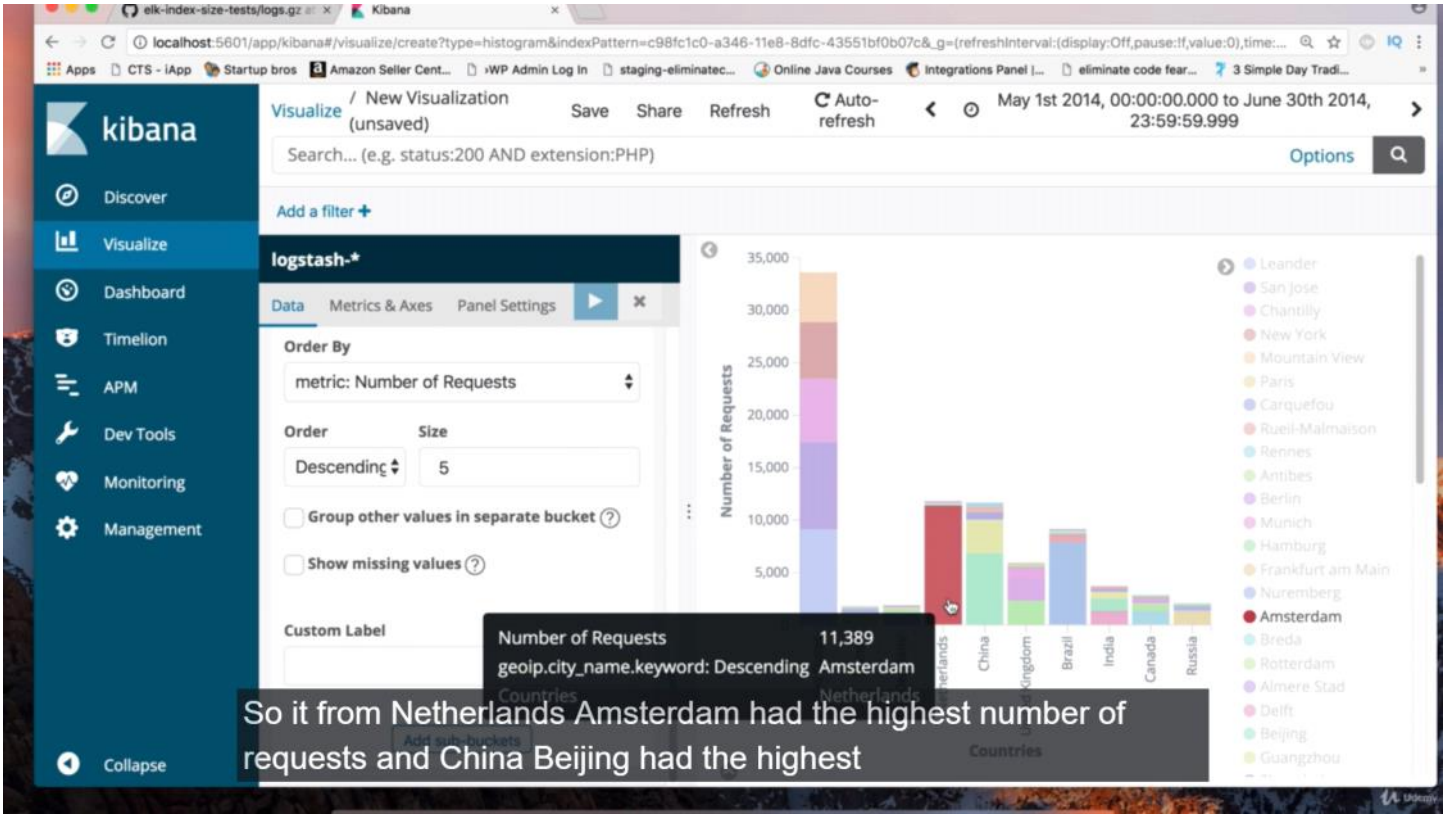
- **Once the indexing is complete, the indexes need to be basically identified by an index pattern**

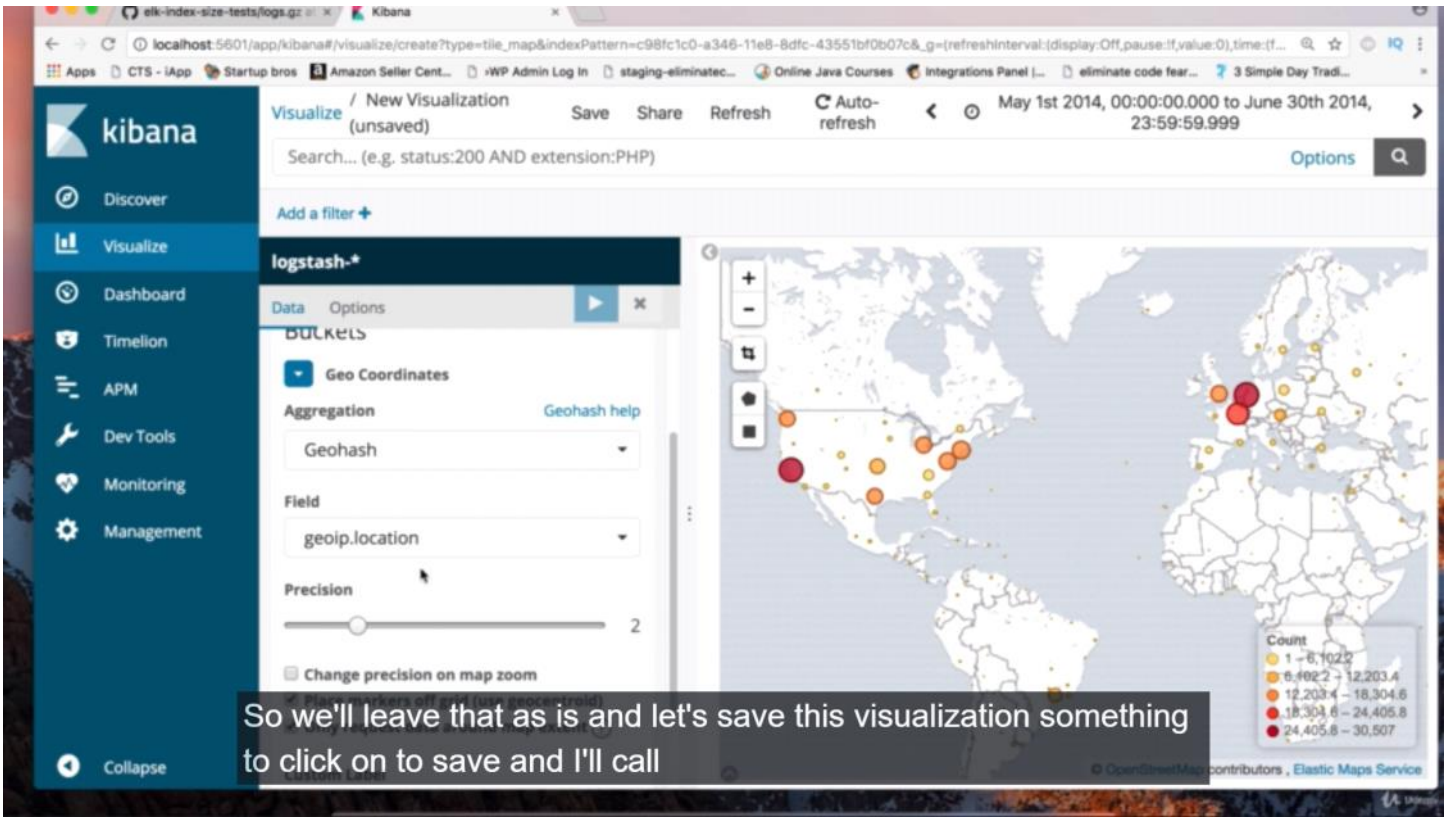


- **You can use sub-buckets to add multiple levels of visualization**


kibana
Visualize / New Visualization (unsaved)
Save Share Refresh Auto-refresh
May 1st 2014, 00:00:00.000 to June 30th 2014, 23:59:59.999
Search... (e.g. status:200 AND extension:PHP)
Options
Add a filter
Discover
Visualize
Dashboard
Timelion
APM
Dev Tools
Monitoring
Management
Collapse
logstash-*
Data Options
Terms
Field
geoip.country_name.keyword
Order By
metric: Count
Order
Descending
Size
3
Group other values in separate bucket
field value Count
geoip.continent_code.keyword: Descending NA 129,889 (43.59%)
NA
EU
AS
SA
OC
United States
Canada
Mexico
France
Germany
Netherlands
China
India
Indonesia
Brazil
Argentina
Chile
Australia
New Zealand
French Polynesia
There's these this inner circle the outer circle is going to represent the country name as you can see


kibana
Visualize / New Visualization (unsaved)
Save Share Refresh Auto-refresh
May 1st 2014, 00:00:00.000 to June 30th 2014, 23:59:59.999
Search... (e.g. status:200 AND extension:PHP)
Options
Add a filter
Discover
Visualize
Dashboard
Timelion
APM
Dev Tools
Monitoring
Management
Collapse
logstash-*
Data Metrics & Axes Panel Settings
X-Axis
Aggregation
Terms help
Terms
Field
geoip.country_name.keyword
Order By
metric: Count
Order
Descending
Size
10
Count
120,000
100,000
80,000
60,000
40,000
20,000
0
France
China
India
Canada
Russia
Brazil
geoip.country_name.keyword: Descending
Notice that you can see that United States has the most number of quests which is 120000.

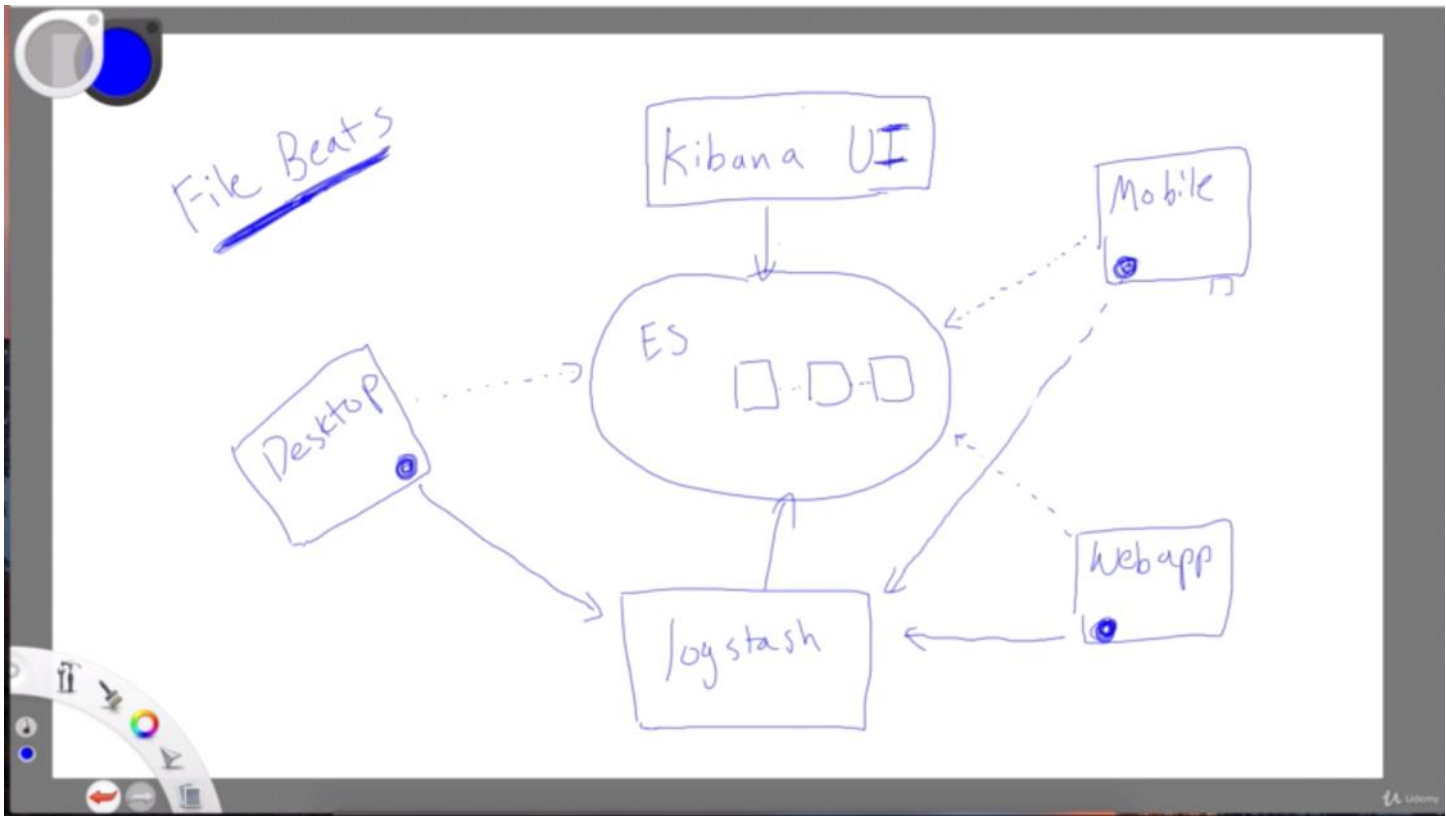

kibana
Visualize / New Visualization (unsaved)
Save
Share
Refresh
Auto-refresh
May 1st 2014, 00:00:00.000 to June 30th 2014, 23:59:59.999
Search... (e.g. status:200 AND extension:PHP)
Options
Add a filter
Discover
Visualize
Dashboard
Timelion
APM
Dev Tools
Monitoring
Management
logstash-*
Data
Metrics & Axes
Panel Settings
Order By
metric: Number of Requests
Order
Descending
Size
5
Group other values in separate bucket
Show missing values
Custom Label
Number of Requests 11,389
geoip.city_name.keyword: Descending Amsterdam
Number of Requests
Countries
Collapse
So it from Netherlands Amsterdam had the highest number of requests and China Beijing had the highest

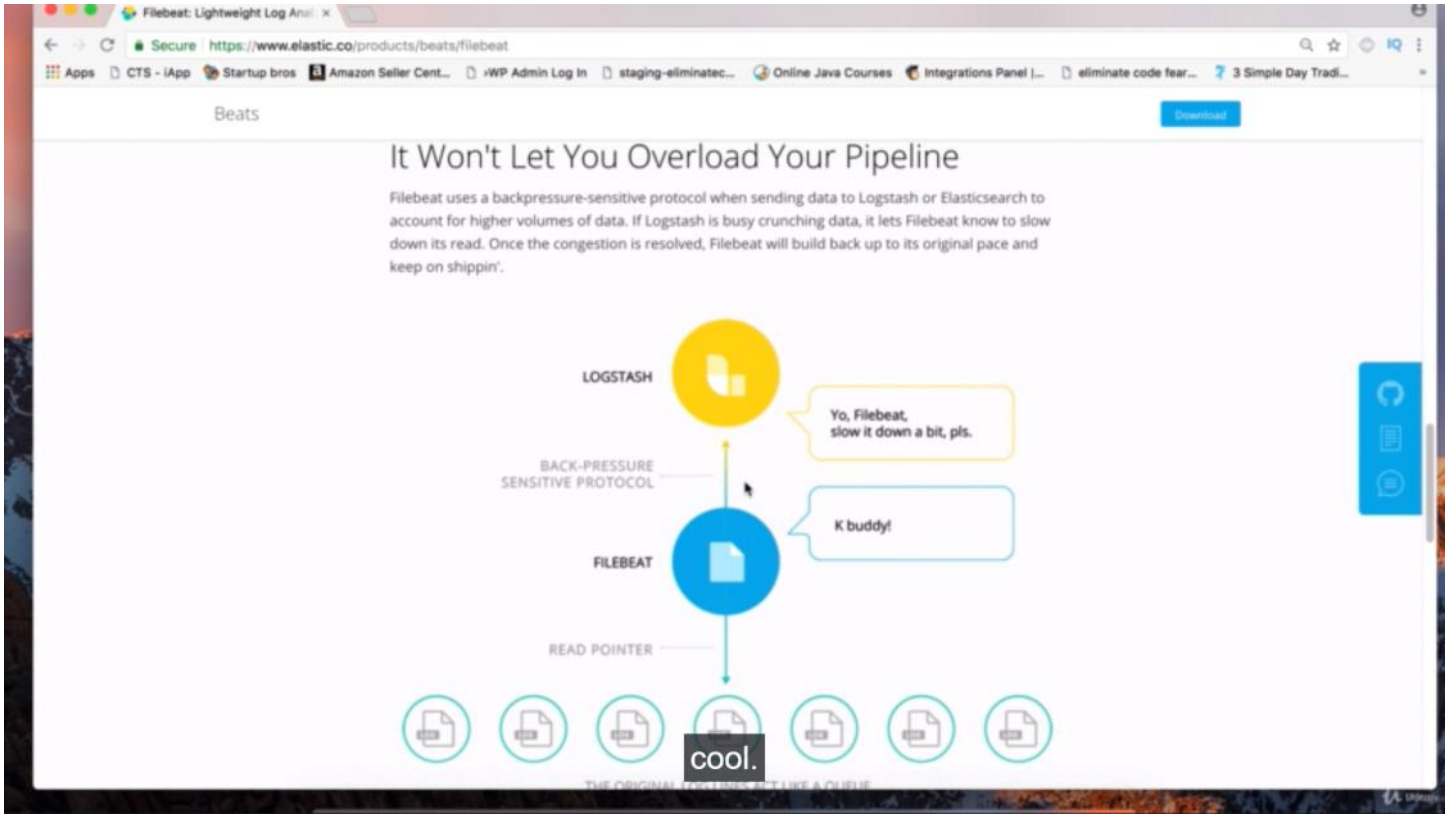

kibana
Visualize / New Visualization (unsaved)
Save
Share
Refresh
Auto-refresh
May 1st 2014, 00:00:00.000 to June 30th 2014, 23:59:59.999
Search... (e.g. status:200 AND extension:PHP)
Options
Add a filter
Discover
Visualize
Dashboard
Timelion
APM
Dev Tools
Monitoring
Management
logstash-*
Data
Options
Buckets
Geo Coordinates
Aggregation
Geohash help
Geohash
Field
geoip.location
Precision
2
Change precision on map zoom
Count
Collapse
So we'll leave that as is and let's save this visualization something to click on to save and I'll call

Lecture 19: Architecting the Elastic Stack

Wednesday, July 10, 2019 11:11 AM

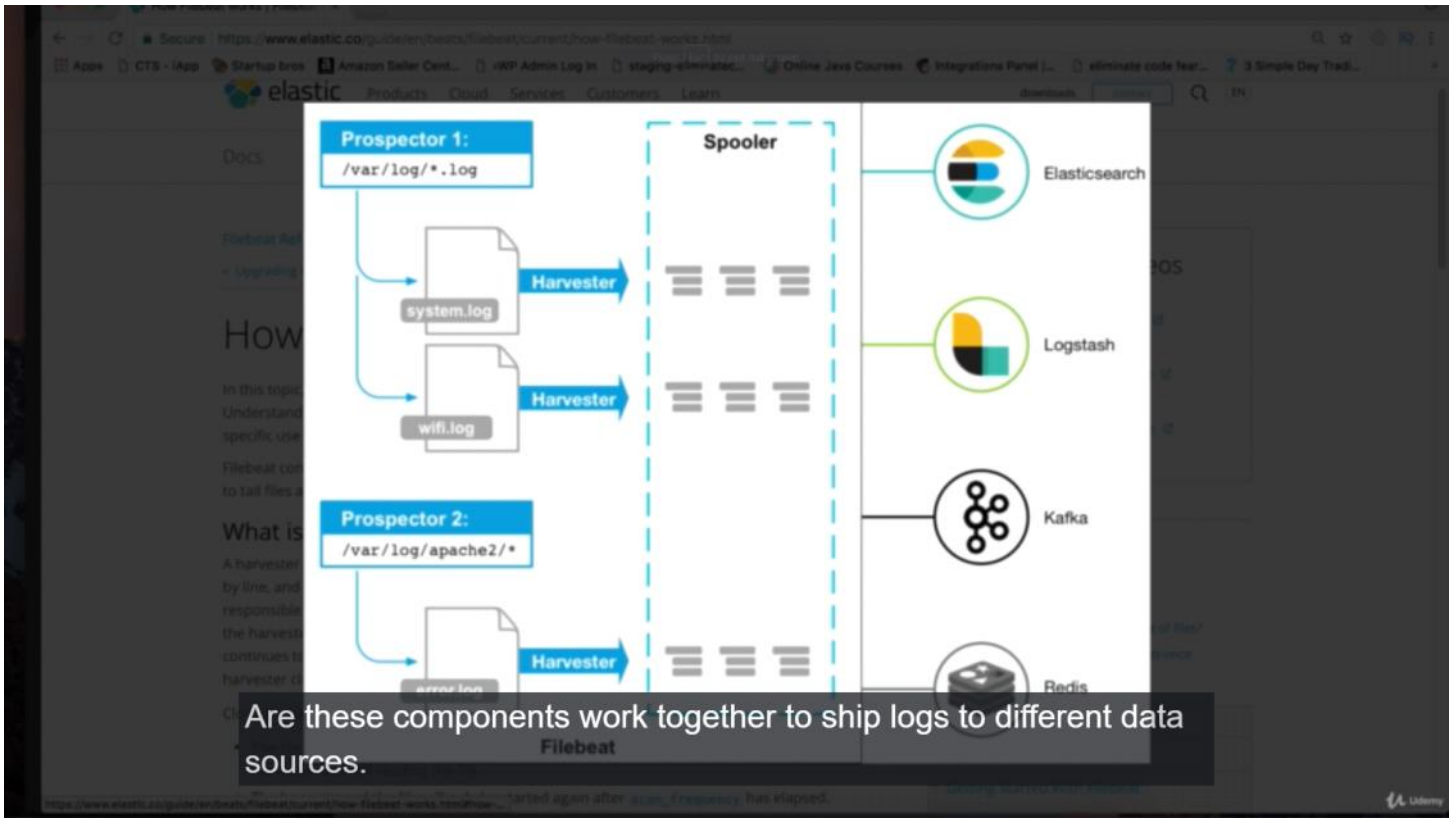
- **ES cluster used to interact w/ Kibana UI for query requests, index requests, etc.**
- **Logstash is used to parse logs and send them to ES**
  - **Creates a pipeline for parsing and indexing events or logs**
  - **Enhances data via plugins**
- **Ex. e-commerce business w/ mobile and web apps**
  - **Products for sale online**
  - **Applications have to interact w/ ELK stack for data analytics, insights, etc.**
- **DO NOT: install Logstash into outward-facing apps (bad architecture b/c resource-intensive)**
  - **INSTEAD: have a lightweight process that can ship application logs to the Logstash server so that the server is able to parse those logs and index them into ES**
- **Can utilize SCP servers or rsync to push data over, OR use:**
  - **BEATS, which are lightweight data servers that can be deployed on your edge apps**
- **Filebeat allow for shipping logs to Logstash or other tools such as Kafka, MongoDB, MySQL, etc.**
- **Logstash (parsing, indexing) is meant to reside on a central location where logs are stashed and parsed and then sent for indexing into ES (dedicated node)**
- **Metricbeat is a lightweight shipper that periodically collects metrics from the OS.**
- **Packetbeat is used for networking**
- **Heartbeat logs the performance of the application and its operations**
- **Winlogbeat, Auditbeat, Mongobeat, Amazonbeat all fall within the Beats auxiliary tool ecosystem**
- **https://www.elastic.co/products/beats/filebeat**




Filebeat: Lightweight Log Anal
Secure https://www.elastic.co/products/beats/filebeat
Apps
CTS - iApp
Startup bros
Amazon Seller Cent...
WP Admin Log In
staging-eliminatec...
Online Java Courses
Integrations Panel |...
eliminate code fear...
3 Simple Day Tradi...
Beats
Download
It Won't Let You Overload Your Pipeline
Filebeat uses a backpressure-sensitive protocol when sending data to Logstash or Elasticsearch to account for higher volumes of data. If Logstash is busy crunching data, it lets Filebeat know to slow down its read. Once the congestion is resolved, Filebeat will build back up to its original pace and keep on shippin'.
LOGSTASH
Yo, Filebeat, slow it down a bit, pls.
BACK-PRESSURE SENSITIVE PROTOCOL
K buddy!
FILEBEAT
READ POINTER
cool.

Lecture 20: Filebeat Installation and Overview

Wednesday, July 10, 2019 12:57 PM



- **These components work together to tail files and send event data to the output that you specify**
- **Summary of how Filebeat software works^**

Lecture 21: Using Filebeat with Logstash

Wednesday, July 10, 2019 1:23 PM

- **Ship logs over to Logstash and let Logstash do the indexing**
- **https://www.elastic.co/guide/en/beats/filebeat/current/logstash-output.html**
- **https://notepad.pw/sw891zjs**