Elasticsearch Primer

Introduction to NoSQL Databases and Elasticsearch

Delivered by: Noureddin Sadawi, PhD

Pearson

About the Speaker

Name: Dr Noureddin Sadawi

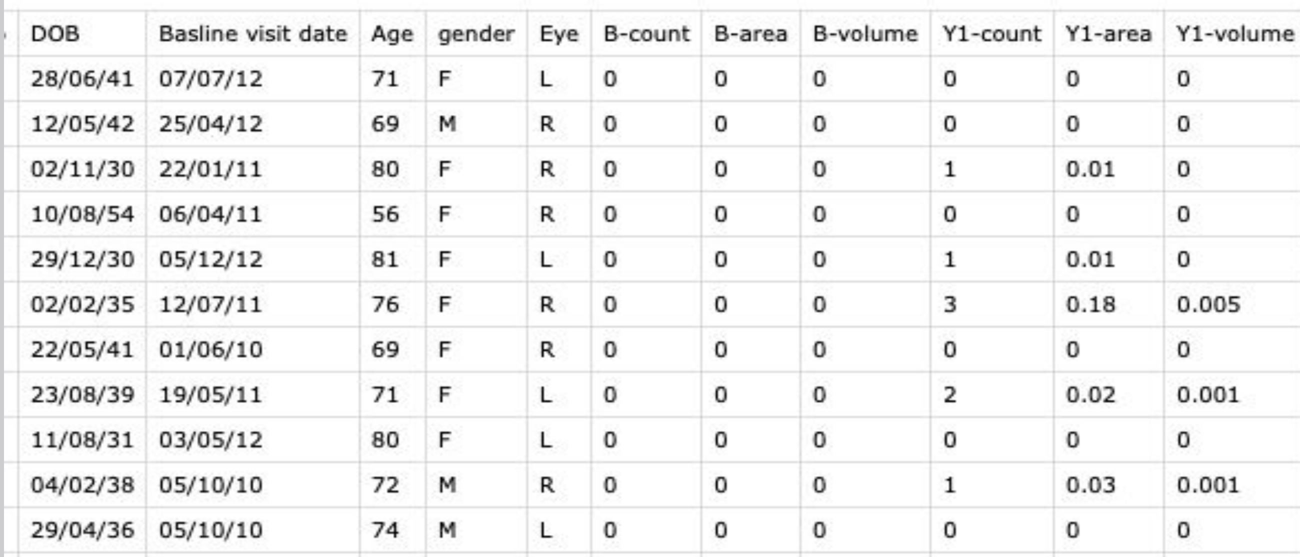
Qualification: PhD Computer Science

Experience: Several areas including but not limited to Docker, Machine Learning and Data Science, Python and much more

SQL Database 1/2

- Relational databases accessed by SQL (Structured Query Language)
- SQL is used to interact with relational databases where data is stored in tables (fixed rows and columns)
- SQL databases became popular in the early 1970s
- Back then, storage was highly costly
- Therefore software engineers worked on DB normalisation to minimise data duplication
- The *waterfall software development model* was commonly followed by engineers in the 1970s

SQL Database Table Example



DOB	Basline visit date	Age	gender	Eye	B-count	B-area	B-volume	Y1-count	Y1-area	Y1-volume
28/06/41	07/07/12	71	F	L	0	0	0	0	0	0
12/05/42	25/04/12	69	M	R	0	0	0	0	0	0
02/11/30	22/01/11	80	F	R	0	0	0	1	0.01	0
10/08/54	06/04/11	56	F	R	0	0	0	0	0	0
29/12/30	05/12/12	81	F	L	0	0	0	1	0.01	0
02/02/35	12/07/11	76	F	R	0	0	0	3	0.18	0.005
22/05/41	01/06/10	69	F	R	0	0	0	0	0	0
23/08/39	19/05/11	71	F	L	0	0	0	2	0.02	0.001
11/08/31	03/05/12	80	F	L	0	0	0	0	0	0
04/02/38	05/10/10	72	M	R	0	0	0	1	0.03	0.001
29/04/36	05/10/10	74	M	L	0	0	0	0	0	0

These values are not real

SQL Database 2/2

- This means detailed planning of projects was performed before development began
- Complex entity-relationship (E-R) diagrams were often created to make sure all data that needed storage was carefully thought about (this is really time-consuming)
- If project requirements change, adaptation becomes a nightmare
- Thus many projects were not successful in fulfilling user needs
- Also, it was easy for projects to go over budget and/or exceeded deadlines

NoSQL

- The term “NoSQL database” is used to refer to any non-relational database
- “NoSQL” can stand for “non SQL” or “not only SQL”
- NoSQL databases are databases that store data in a format other than relational tables
- Some think that NoSQL databases do not store relationship data well
- relationship data can be stored in NoSQL databases — it is just stored differently when compared with relational databases
- Some Differences between SQL and NoSQL: https://www.mongodb.com/nosql-explained/nosql-vs-sql

NoSQL

- NoSQL data models allow related data to be nested within a single data structure
- As storage costs rapidly decreased, the amount of data applications needed to store and query increased.
- Gone were the days of needing to create a complex, difficult-to-manage data model simply for the purposes of reducing data duplication
- This data came in all shapes and sizes—structured, semistructured, and polymorphic—and defining the schema in advance became nearly impossible
- NoSQL databases allow developers to store huge amounts of unstructured data, giving them a lot of flexibility

NoSQL

- The need to rapidly adapt to changing requirements
- The ability to iterate quickly and make changes throughout their software stack—all the way down to the database model. NoSQL databases gave them this flexibility

Types of NoSQL Databases

- **Document databases:** data is stored in documents similar to JSON (JavaScript Object Notation) objects. Each document contains pairs of fields and values (Examples: Elasticsearch, MongoDB)
- **Key-value databases:** each item contains keys and values. The key is used as a reference to retrieve a value (Examples: Redis, DynamoDB)
- **Wide-column stores:** data is stored in tables, rows, and dynamic columns (each row is not required to have the same columns). Examples: HBase, Cassandra
- **Graph databases:** data is stored in nodes and edges. Information about people, places, and things are usually stored in nodes. Information about the relationships between the nodes is stored in edges (Examples: Neo4j and JenusGraph)

JSON (JavaScript Object Notation)

- A lightweight data-interchange format
  - It is easy for humans to read and write
  - It is easy for machines to parse and generate

https://www.json.org/json-en.html

JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array
- An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence

JSON (JavaScript Object Notation)

```
{
        “First Name”: “Noureddin”,
        “Last Name”: “Sadawi”,
        “Weight in Kg”: 88.1,
        “Hobbies”: [“Reading”, “Running”, ”Sleeping”]
}
```

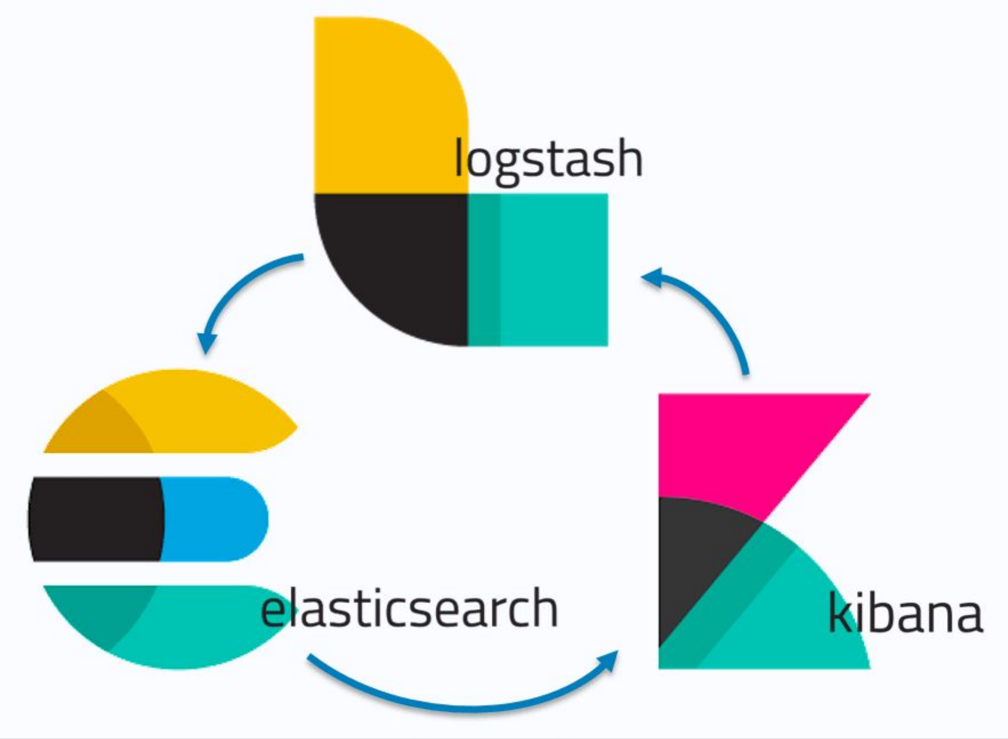
What is Elasticsearch

- Lucene:
  - A search engine library entirely written in Java
  - Developed in 1999 by Doug Cutting
  - Suitable for any application that requires full text indexing and searching capability
- But:
  - Challenging to use
  - Not originally designed for scaling
- Elasticsearch:
  - Built on top of Lucene
  - Provides scaling
  - Language independent

What is the ELK Stack?

- ElasticSearch:
    - The main datastore
    - Provides distributed search capabilities
    - RESTful API
- Logstash:
    - Parse & transform data for ingestion
    - Ingests from multiple of sources simultaneously
- Kibana:
    - An analytics and visualization platform
    - Search, visualize & interact with Elasticsearch data

The ELK Stack



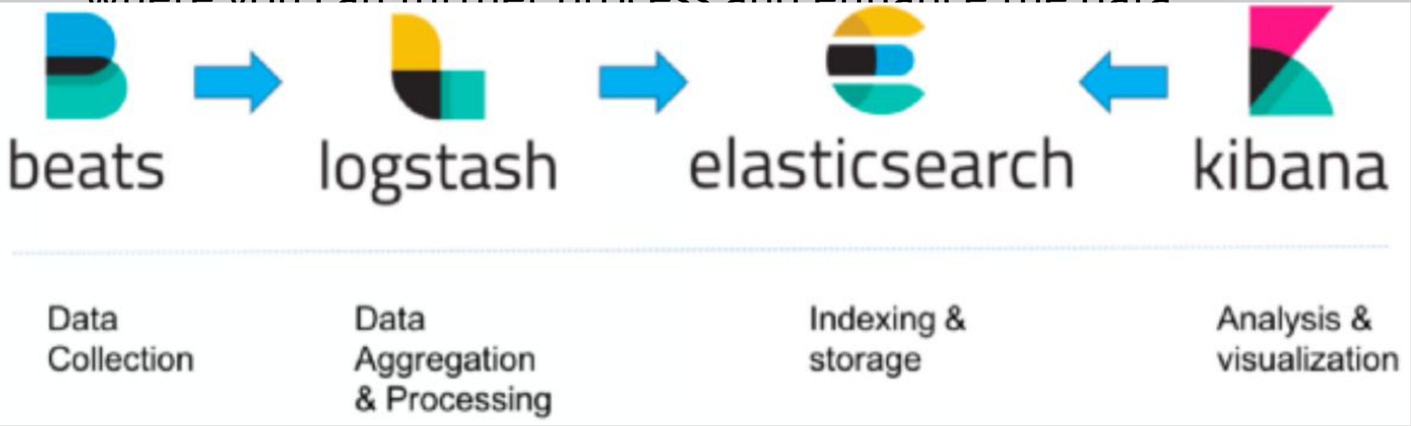
logstash

elasticsearch

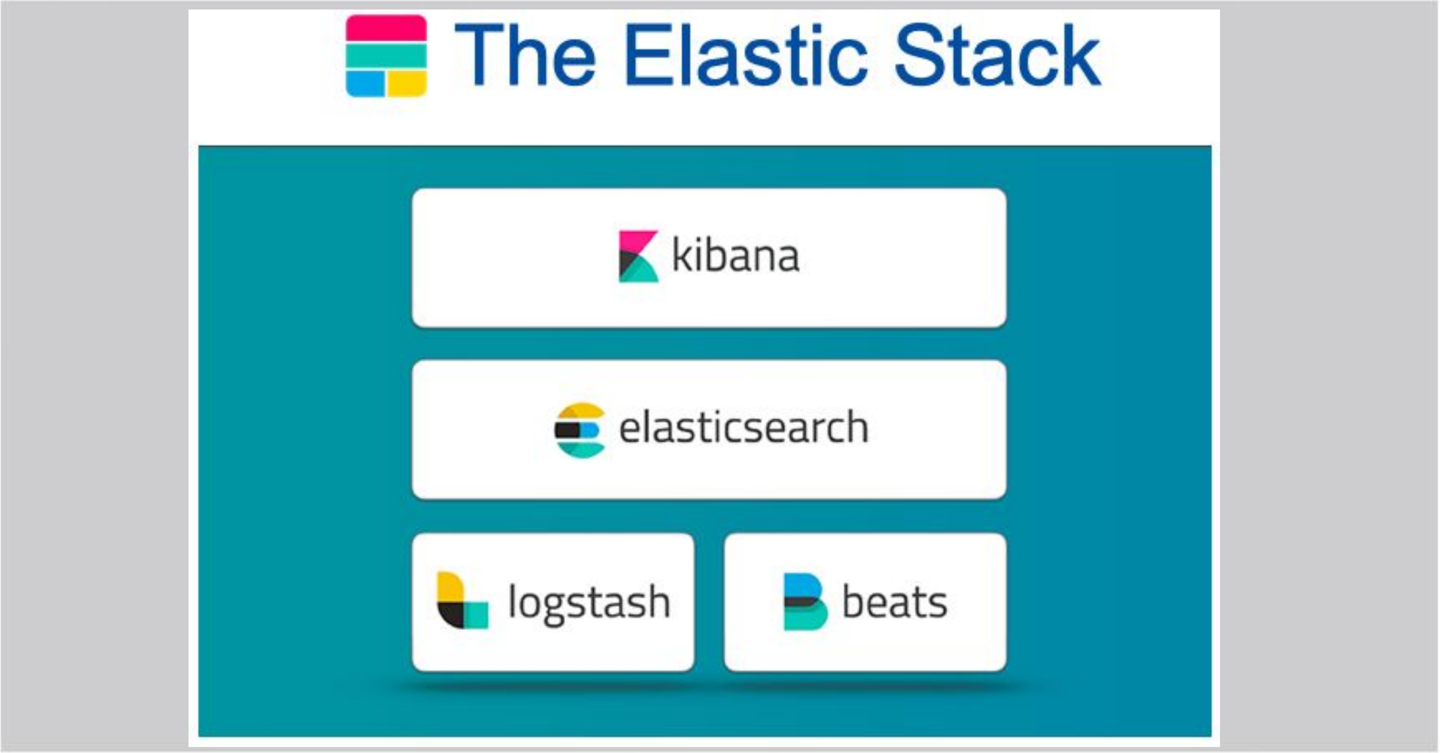
kibana

Beats

- The Beats are open source data shippers that you install as agents on your servers to send operational data to Elasticsearch
- Beats can send data directly to Elasticsearch or via Logstash, where you can further process and enhance the data



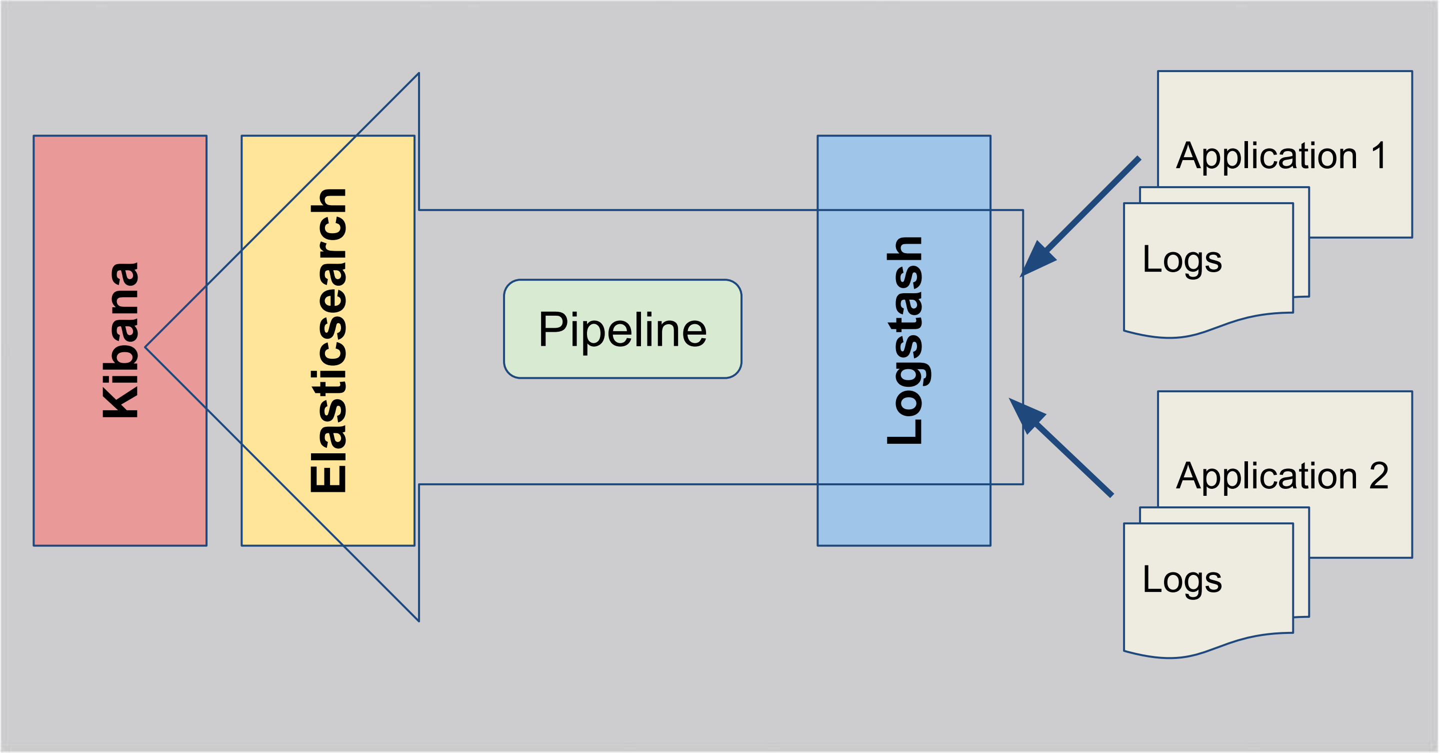
The ELK Stack



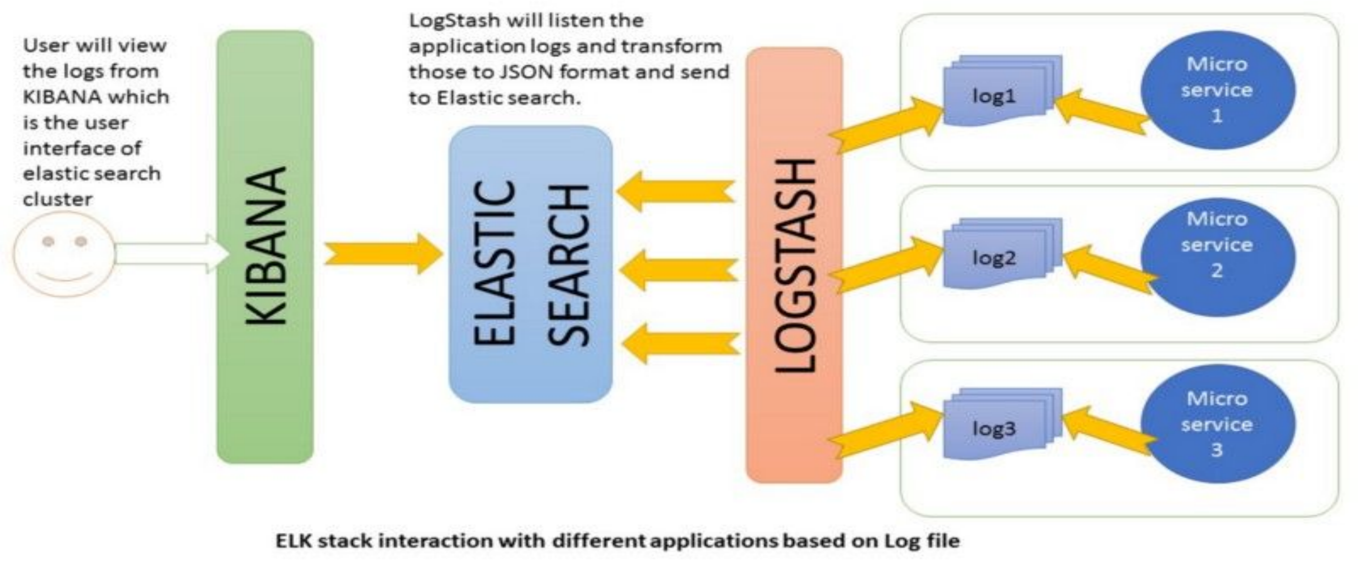
Why ELK Stack?

- A major challenge in distributed systems is to understand what is going on .. more importantly, what is wrong? where? and why?
- ELS stack is great for Log aggregation and analytics
- NoSQL DB, distributed search and analytics engine
- Easy to Install and use
- Powerful search technology
- Logstash is a transportation pipeline used to populate ES with data (log aggregator)

The Flow


Kibana
Elasticsearch
Pipeline
Logstash
Application 1
Logs
Application 2
Logs

The Flow


User will view the logs from KIBANA which is the user interface of elastic search cluster
KIBANA
LogStash will listen the application logs and transform those to JSON format and send to Elastic search.
ELASTIC SEARCH
LOGSTASH
log1
log2
log3
Micro service 1
Micro service 2
Micro service 3

ELK stack interaction with different applications based on Log file

https://www.freecodecamp.org/news/how-to-use-elasticsearch-logstash-and-kibana-to-visualise-logs-in-python-in-realtime-acaab281c9de/

ELK Stack Installation

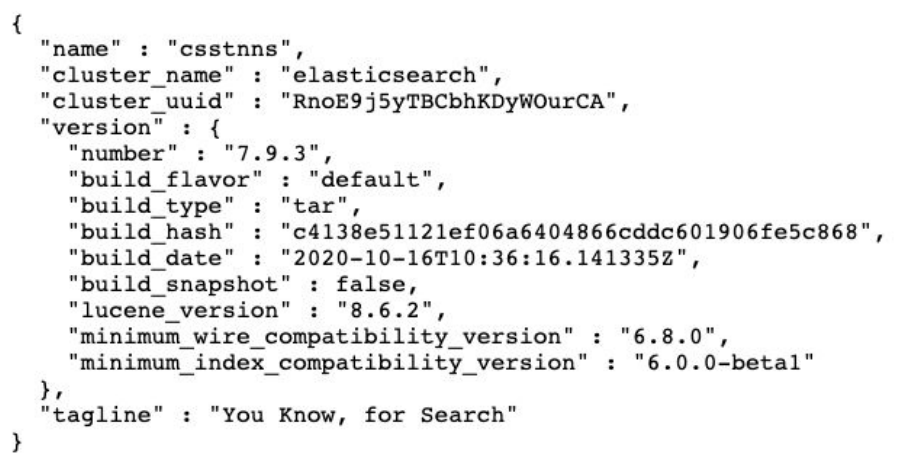
- ES: https://www.elastic.co/downloads/elasticsearch
- Kibana: https://www.elastic.co/downloads/kibana
- Logstash: https://www.elastic.co/downloads/logstash
- Beats: https://www.elastic.co/downloads/beats
- Filebeat: https://www.elastic.co/downloads/beats/filebeat

Java Installation and JAVA_HOME

- ElasticSearch requires JRE (JavaSE runtime environment) or JDK (Java Development Kit)
- On Ubuntu Linux: https://www.digitalocean.com/community/tutorials/how-to-install-java-with-apt-on-ubuntu-18-04
- On Windows 10: https://www.jackrutorial.com/2018/10/how-to-install-java-jdk-11-on-windows-10.html

Start ES and Test it

- cd into ES folder and then `bin/elasticsearch`
- `nohup elasticsearch-7.10.0/bin/elasticsearch > elastic.out 2>&1 &`
- Using the curl tool: `curl -X GET "localhost:9200/"`



```
{
  "name" : "csstnns",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "RnoE9j5yTBCbhKDyWOurCA",
  "version" : {
    "number" : "7.9.3",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "c4138e51121ef06a6404866cddc601906fe5c868",
    "build_date" : "2020-10-16T10:36:16.141335Z",
    "build_snapshot" : false,
    "lucene_version" : "8.6.2",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

ES Port Numbers

By default, Elasticsearch uses two ports to listen to external TCP traffic:

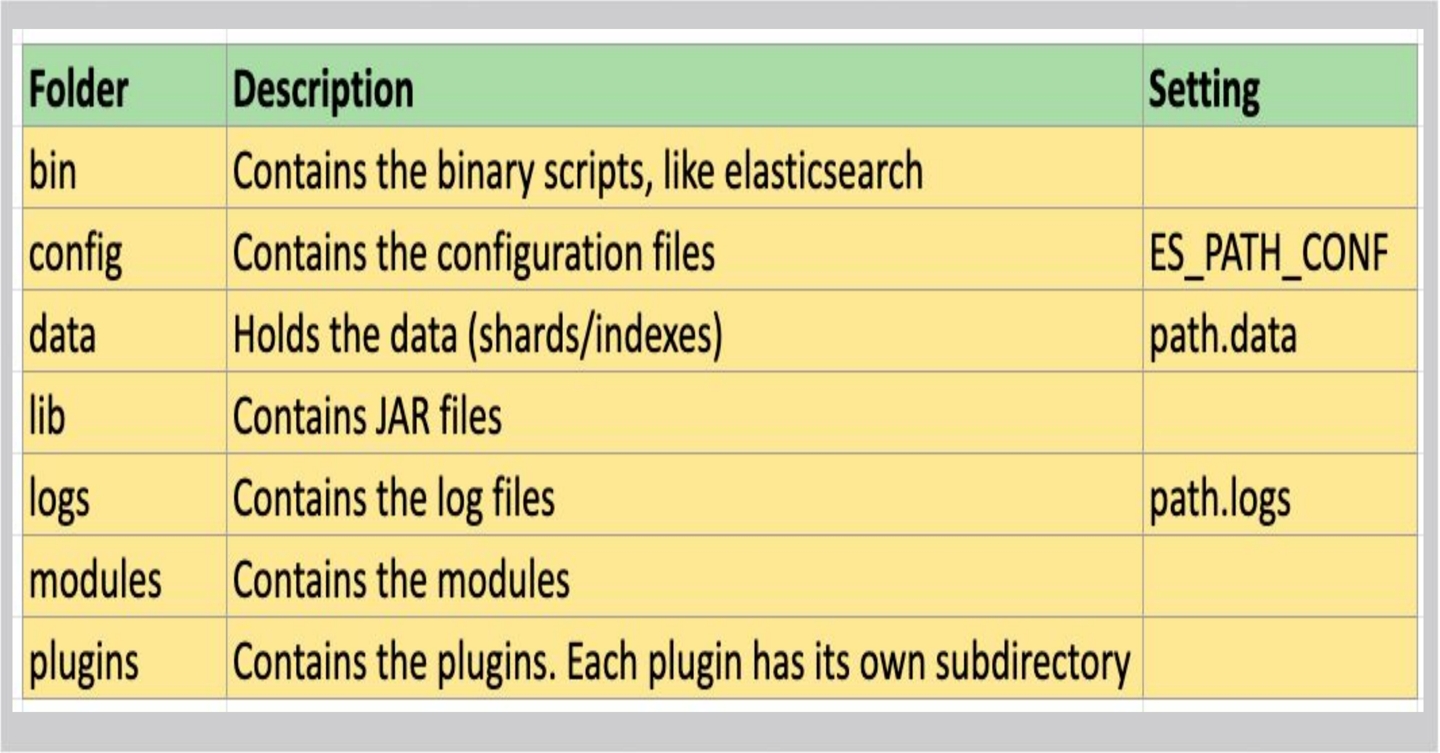
- Port 9200 is used for all API calls over HTTP
  a. This includes search and aggregations, monitoring and anything else that uses a HTTP request
  b. All client libraries will use this port to talk to Elasticsearch
- Port 9300 is a custom binary protocol used for communications between nodes in a cluster
  a. For things like cluster updates, master elections, nodes joining/leaving, shard allocation

Start ES .. Common Error

- If you get this error:
- `flood stage disk watermark [95%] exceeded on`
- It means you are running low on disk space

- ES will mark all indices on that node **read-only**
- You need to manually remove the read only mode
- See here: https://www.elastic.co/guide/en/elasticsearch/reference/7.0/disk-allocator.html

ES Directories



Folder	Description	Setting
bin	Contains the binary scripts, like elasticsearch	
config	Contains the configuration files	ES_PATH_CONF
data	Holds the data (shards/indexes)	path.data
lib	Contains JAR files	
logs	Contains the log files	path.logs
modules	Contains the modules	
plugins	Contains the plugins. Each plugin has its own subdirectory	

Configuration files

elasticsearch.yml

- The primary way of configuring a node
- It is a template which lists the most important settings for a production cluster

jvm.options

- JVM related options

log4j2.properties

- Elasticsearch uses Log4j 2 for logging

Variables can be set either:

- Using the configuration file: jvm.options: -Xms512mb
- or, using command line ES_JAVA_OPTS="-Xms512m" ./bin/elasticsearch

Elasticsearch.yml

node.name

- Every node should have a unique node.name
- Set it to something meaningful

cluster.name

- A cluster is a set of nodes sharing the same cluster.name
- Set it to something meaningful (production, qa, staging)

path.data

- Path to directory where to store the data (accepts multiple locations)

path.logs

- Path to log files

Make Elasticsearch Public

Default is Local

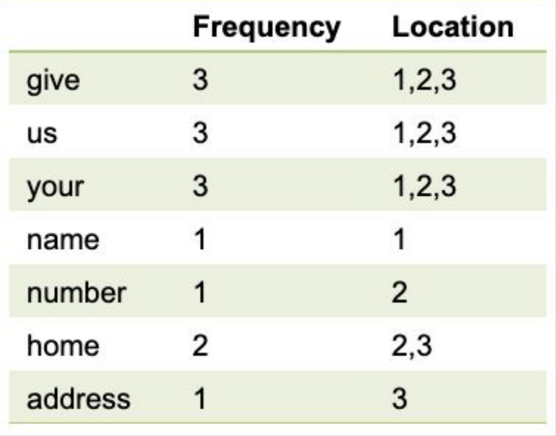
- By default ES is only accessible from within the localhost (i.e. the machine it is installed on)
- You can make it publicly accessible by making the following edits in the elasticsearch.yml file (restart ES if it is running):

```
transport.host: localhost

transport.tcp.port: 9300

http.port: 9200

network.host: 0.0.0.0
```

Lucene

- Lucene uses a data structure called **Inverted Index**
- An Inverted Index, inverts a page-centric data structure (page->words) to a keyword-centric data structure (word->pages)
- Allow fast **full text searches**, at a cost of increased processing when a document is added to the DB

1) Give us your name
2) Give us your home number
3) Give us your home address



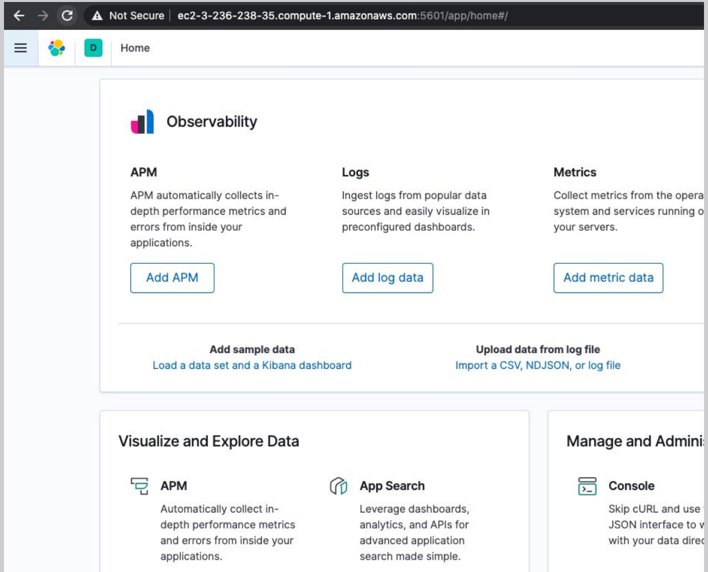
	Frequency	Location
give	3	1,2,3
us	3	1,2,3
your	3	1,2,3
name	1	1
number	1	2
home	2	2,3
address	1	3

Lucene - Key Terms

- A **Document** is the unit of search and index
- A Document consists of one or more **Fields**. A Field is simply a name-value pair
- An **index** consists of one or more **Documents**
- **Indexing:** involves adding Documents to an Index
- **Searching:**
  - Involves retrieving Documents from an index
  - Searching requires an index to have already been built
  - Returns a list of Hits

Start Kibana and Test it

- cd into Kibana folder and then `bin/kibana`
- Access Kibana from web-browser: http://localhost:5601



Not Secure | ec2-3-236-238-35.compute-1.amazonaws.com:5601/app/home#/

Home

Observability

APM

APM automatically collects in-depth performance metrics and errors from inside your applications.

Add APM

Logs

Ingest logs from popular data sources and easily visualize in preconfigured dashboards.

Add log data

Metrics

Collect metrics from the opera system and services running o your servers.

Add metric data

Add sample data
Load a data set and a Kibana dashboard

Upload data from log file
Import a CSV, NDJSON, or log file

Visualize and Explore Data

APM

Automatically collect in-depth performance metrics and errors from inside your applications.

App Search

Leverage dashboards, analytics, and APIs for advanced application search made simple.

Manage and Admini

Console

Skip cURL and use JSON interface to w with your data direc

Kibana Port Number and Public Access

- Kibana uses port 5601 by default (you can change it)
- Kibana by default binds to the local host
- This means it is not accessible from outside the machine it is installed on
- In order to make it publicly accessible, make sure you change the following setting in `config/kibana.yml`
- `server.port: 5601`
- `server.host: "localhost"`
- `elasticsearch.hosts: ["http://localhost:9200"]`
- set `server.host` to “`0.0.0.0`” and restart Kibana
- `nohup kibana-7.10.0-linux-x86_64/bin/kibana > kibana.out 2>&1  &`

Kibana Index Patterns

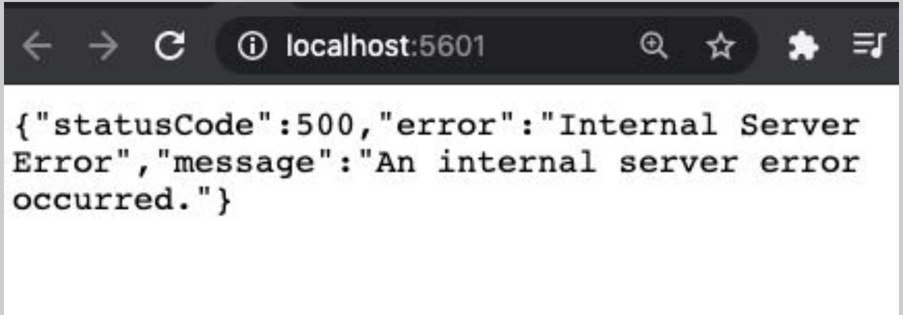
- https://www.elastic.co/guide/en/kibana/current/index-patterns.html
- Some nice Logstash pipelines and data for Kibana: https://github.com/PacktPublishing/Kibana-7-Quick-Start-Guide

Start Kibana .. Common Error

- If you see this error on the web-browser
- Or any error in the command line that says:

  ```
  TOO_MANY_REQUESTS/12/disk usage exceeded
  flood-stage watermark, index has
  read-only-allow-delete block
  ```

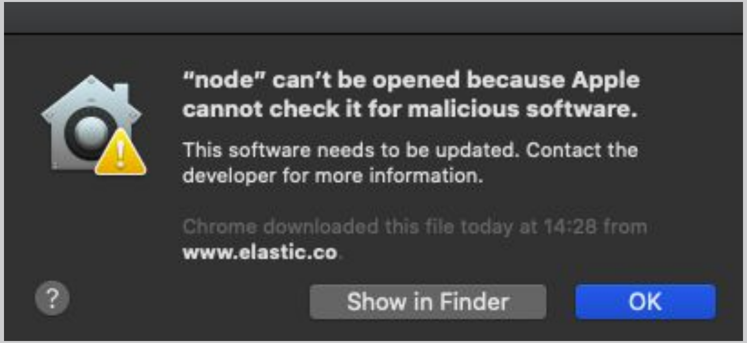
- Then refer the ES installation error because it is an ES issue



```
{"statusCode":500,"error":"Internal Server
Error","message":"An internal server error
occurred."}
```

Start Kibana .. MAC Users

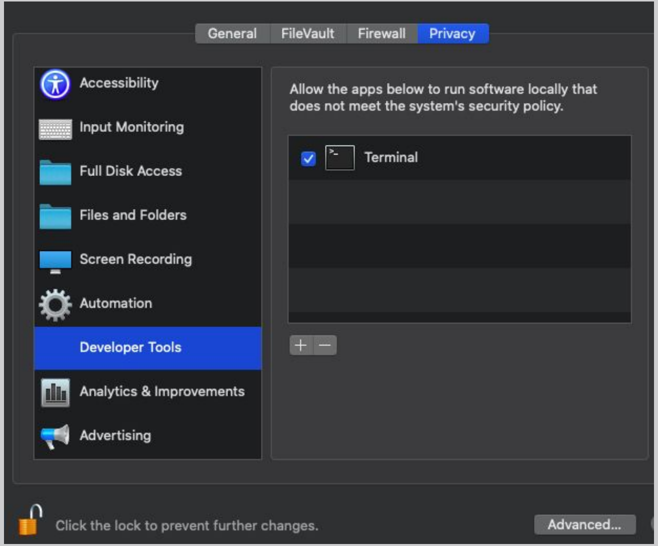
- If this pops up
- System preferences > security and privacy > developer tools ... edit to allow terminal to run software that doesn't meet the system security preferences.



"node" can't be opened because Apple cannot check it for malicious software.

This software needs to be updated. Contact the developer for more information.

Chrome downloaded this file today at 14:28 from **www.elastic.co**



Allow the apps below to run software locally that does not meet the system's security policy.

☑ Terminal

Click the lock to prevent further changes.

Start Logstash and Test it

- cd into Logstash folder and then `bin/logstash`
- Access Kibana from web-browser: http://localhost:9600
- LS uses port 9600 by default (you can change it)
- LS by default binds to the local host
- This means it is not accessible from outside the machine it is installed on
- In order to make it publicly accessible, make sure you change the following setting in `config/logstash.yml`

  ```
  http.host: 0.0.0.0
  ```

  The default value is `127.0.0.1`

Logstash Pipeline

- Download Apache access log (assume you are running apache)
- Download Logstash pipeline for Apache logs from: https://logz.io/blog/logstash-tutorial/
- Let’s go through the pipeline
- Make sure ES and Kibana are up and running
- Start LS by telling it to use the pipeline (so that it parses the log file and pushes the data unto ES)
- `bin/logstash -f path/to/pipeline`

Logstash Pipelines

- https://www.elastic.co/guide/en/logstash/current/configuration.html
- https://logz.io/blog/logstash-tutorial/
- https://logz.io/blog/logstash-pipelines/
- Grok tutorial: https://logz.io/blog/logstash-grok/
- Logstash with mutate filter: https://coralogix.com/log-analytics-blog/logstash-csv-import-parse-your-data-hands-on-examples/

End of Part 1

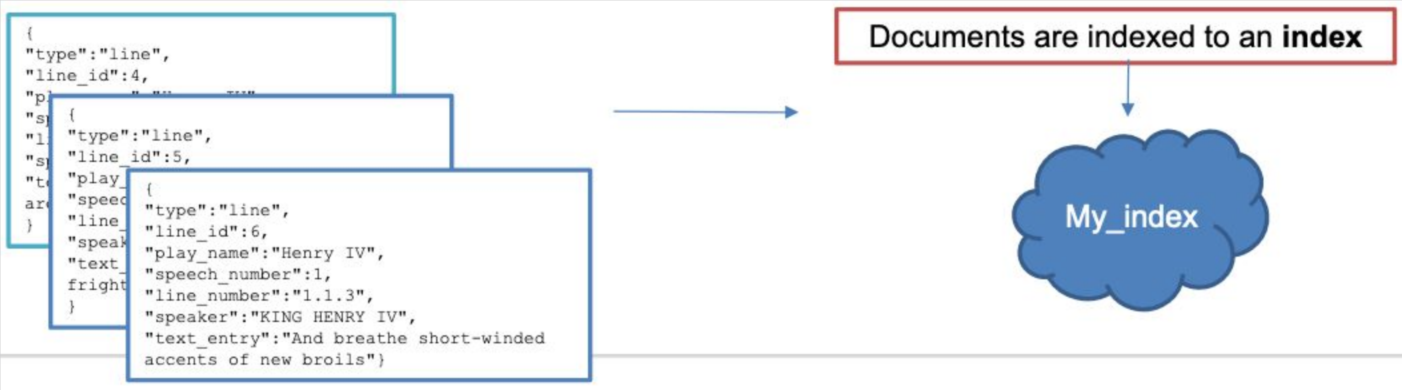
Working with Data - Index

- An index in Elasticsearch is a logical way of grouping data:
  - an index has a **mapping** that defines the fields in the index
  - an index is a **logical namespace** that maps to where its contents are stored in the cluster

- There are two different concepts in this definition:
  - an index has some type of data schema mechanism
  - an index has some type of mechanism to distribute data across a cluster

Working with Data - Index

- In the Elasticsearch world, index is used as a:
  - **Noun:** a document is put into an index in Elasticsearch
  - **Verb:** to index a document is to put the document into an index in Elasticsearch



```
{
"type":"line",
"line_id":4,
"p
"s
"l
"s
"t
ar
}
```

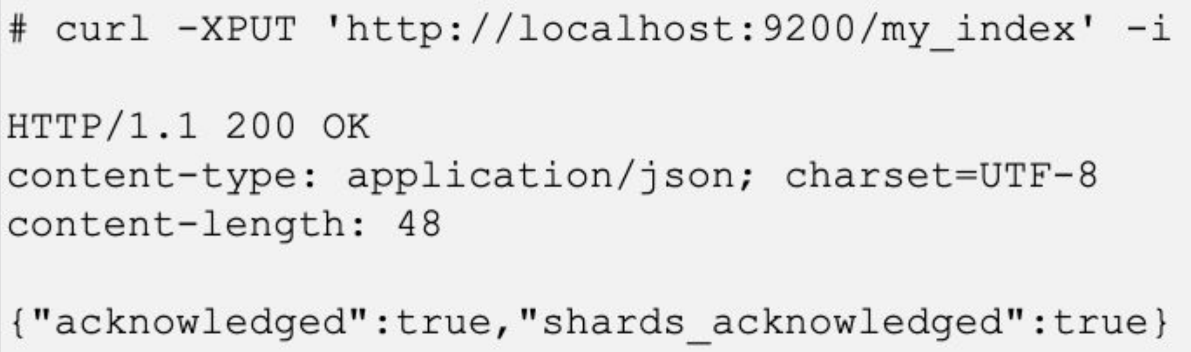
```
{
"type":"line",
"line_id":5,
"play_
"speec
"line_
"speak
"text_
fright
}
```

```
{
"type":"line",
"line_id":6,
"play_name":"Henry IV",
"speech_number":1,
"line_number":"1.1.3",
"speaker":"KING HENRY IV",
"text_entry":"And breathe short-winded
accents of new broils"}
```

Documents are indexed to an **index**

Define an Index

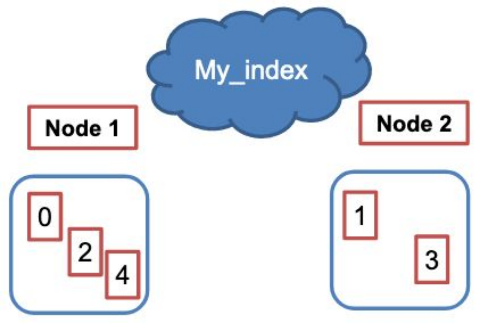
- Clients communicate with a cluster using Elasticsearch’s REST APIs
- An index is defined using the Create Index API, which can be accomplished with a simple PUT command
- https://www.elastic.co/guide/en/elasticsearch/reference/current/indices-create-index.html



```
# curl -XPUT 'http://localhost:9200/my_index' -i

HTTP/1.1 200 OK
content-type: application/json; charset=UTF-8
content-length: 48

{"acknowledged":true,"shards_acknowledged":true}
```

Shard

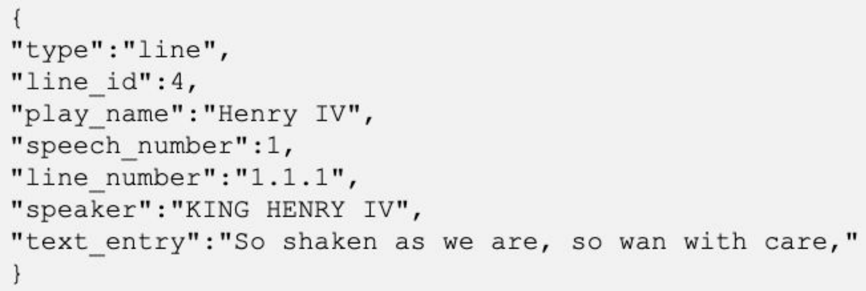
- A shard is a single piece of an Elasticsearch index
  - Indexes are partitioned into shards so they can be distributed across multiple nodes
- Each shard is a standalone Lucene index
  - The default number of shards for an index is 5. Number of shards can be changed at index creation time.



Working with Data - Document

Documents must be JSON objects

- A document can be any text or numeric data you want to search and/or analyze
    - Specifically, a **document** is a top-level object that is serialized into JSON and stored in ES
- Every document has a **unique ID**
    - Which either you provide, or ES generates one for you



```
{
"type":"line",
"line_id":4,
"play_name":"Henry IV",
"speech_number":1,
"line_number":"1.1.1",
"speaker":"KING HENRY IV",
"text_entry":"So shaken as we are, so wan with care,"
}
```

Working with Data - Document

- Data is stored in ES as documents
- A document is a JSON object
- Every document in ES is stored within an index
- An index groups documents together logically and provides config options related to availability, scalability and more
- An index therefore is a collection of documents that have similar characteristics and are logically related
- In other words, indices group documents together

Working with Data - Document

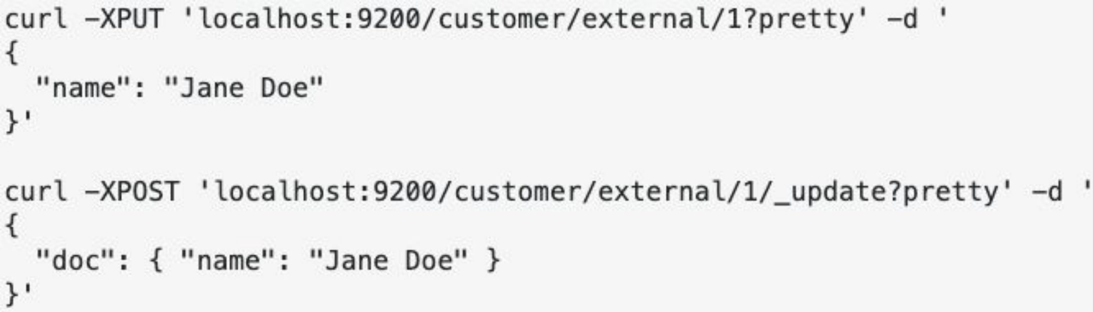
- ES is document oriented (stores objects are documents)
- We do not take our objects into several parts and fit them into several tables as we do in relational DBs for the sake of normalisation
- It indexes documents so that their content is searchable
- <u>The power of full text search</u>

How to Communicate with ES

- You communicate with ES via REST API
- ES uses JSON format
- You send your JSON document over HTTP to ES and you receive the response in JSON format
- Kibana is your friend
- From command-line? any HTTP client is fine .. `curl` is a good one

cURL HTTP Client

- `curl` is a tool to transfer data from or to a server, using one of the supported protocols [How To Use]
- [HTTP Methods for RESTful Services]
- PUT or POST?
- PUT will **update a full document**, not only the field you're sending
- POST will do a **partial update** and only update the fields you're sending, and not touch the ones already present in the document



```
curl -XPUT 'localhost:9200/customer/external/1?pretty' -d '
{
  "name": "Jane Doe"
}'

curl -XPOST 'localhost:9200/customer/external/1/_update?pretty' -d '
{
  "doc": { "name": "Jane Doe" }
}'
```

ES Terminology

- In RDBMs we have:
  - Database
  - Table
  - Tuple (Row)
- In ES we have:
  - Index
  - Type
  - Document

- Indexing a document means inserting/updating a document (noun and verb “index” have different meanings)
- A node is a running instance of ES
- One node per server
- When a node is started, it attempts to find and join a cluster
- A cluster is a group of nodes

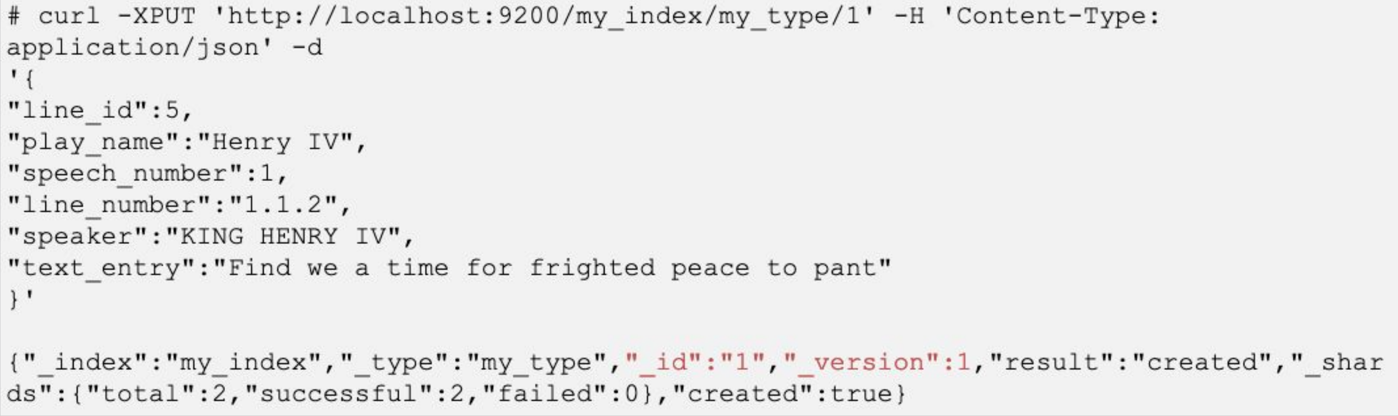
ES Terminology

- Primary shard is the first place where your document is stored when you index it
- Replicas of the your primary shard will get their copy too
- Replica shard is a copy of the primary shard

- Several copies of data because back-up is needed in case primary shard goes down
- Replica shards can enhance the performance of ES
- ES does not put primary shard and its replica on the same node

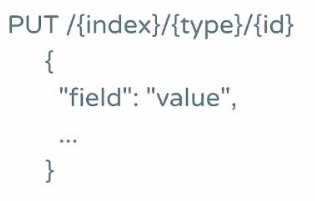
Index a Document

- The **Index API** is used to index a document
- use a **PUT** or a **POST** and add the document in the body request
- notice we specify the **index**, the **type** and an **ID**
- if no ID is provided, elasticsearch will generate one

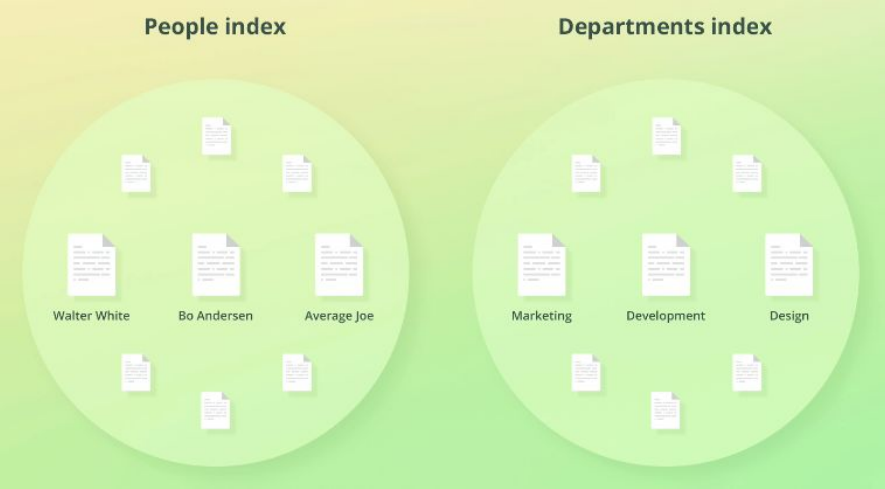


```
# curl -XPUT 'http://localhost:9200/my_index/my_type/1' -H 'Content-Type:
application/json' -d
'{
"line_id":5,
"play_name":"Henry IV",
"speech_number":1,
"line_number":"1.1.2",
"speaker":"KING HENRY IV",
"text_entry":"Find we a time for frighted peace to pant"
}'

{"_index":"my_index","_type":"my_type","_id":"1","_version":1,"result":"created","_shar
ds":{"total":2,"successful":2,"failed":0},"created":true}
```

Index a Document



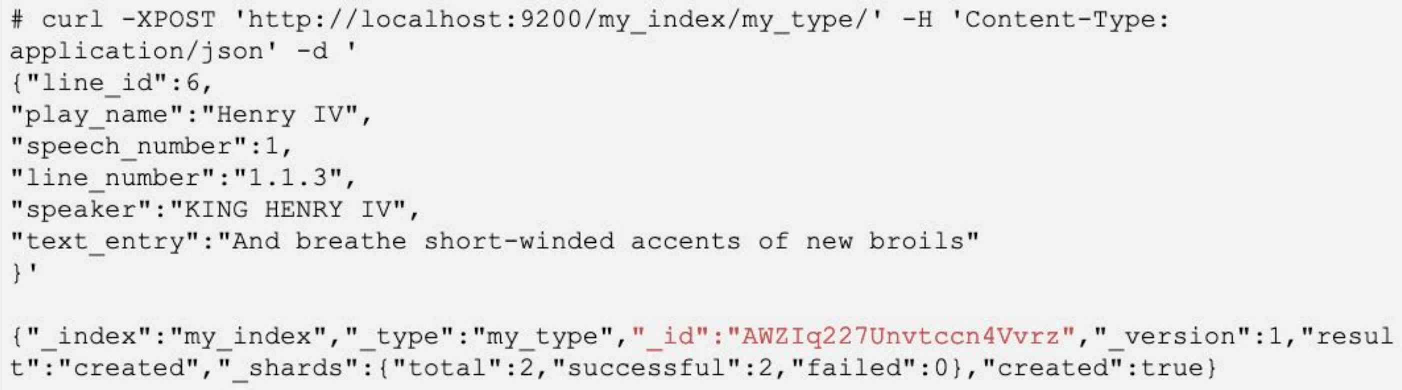
```
PUT /{index}/{type}/{id}
    {
      "field": "value",
      ...
    }
```



Index without specifying an ID

- You can leave off the id and let Elasticsearch generate one for you:
    - But notice that only works with POST, not PUT
    - The generated id comes back in the response

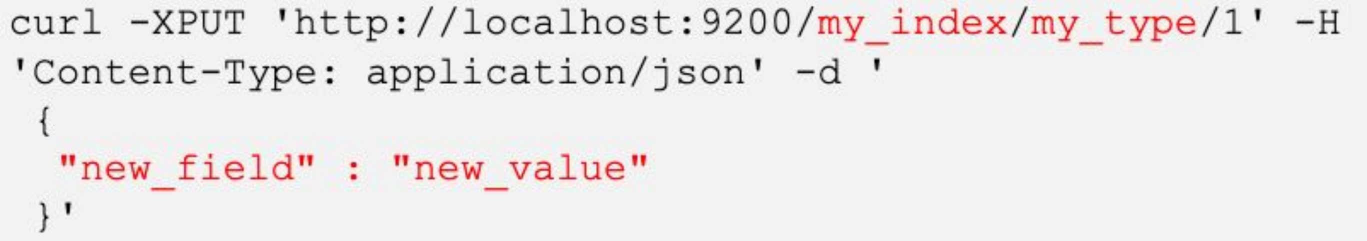
https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started-index.html



```
# curl -XPOST 'http://localhost:9200/my_index/my_type/' -H 'Content-Type:
application/json' -d '
{"line_id":6,
"play_name":"Henry IV",
"speech_number":1,
"line_number":"1.1.3",
"speaker":"KING HENRY IV",
"text_entry":"And breathe short-winded accents of new broils"
}'

{"_index":"my_index","_type":"my_type","_id":"AWZIq227Unvtccn4Vvrz","_version":1,"resul
t":"created","_shards":{"total":2,"successful":2,"failed":0},"created":true}
```

Reindex a Document

- What happens if we add another document with an existing ID?

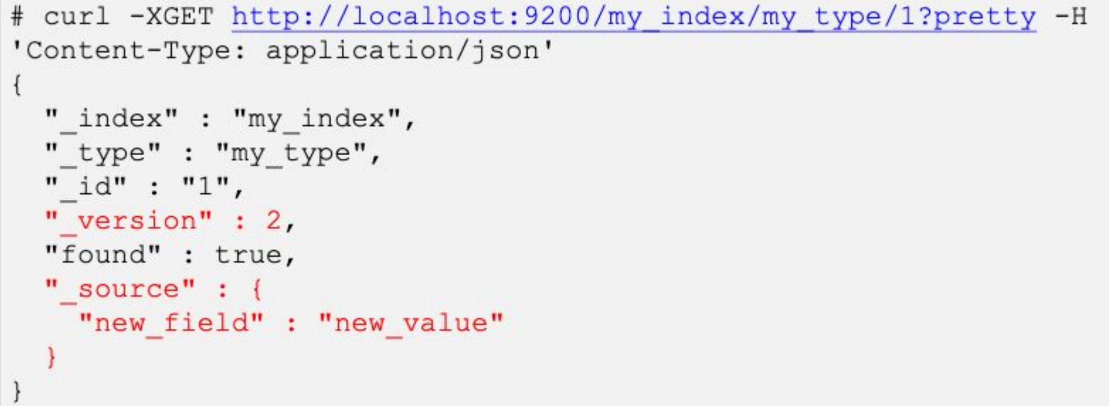


```
curl -XPUT 'http://localhost:9200/my_index/my_type/1' -H
'Content-Type: application/json' -d '
 {
  "new_field" : "new_value"
 }'
```

https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-reindex.html

Document is Overwritten

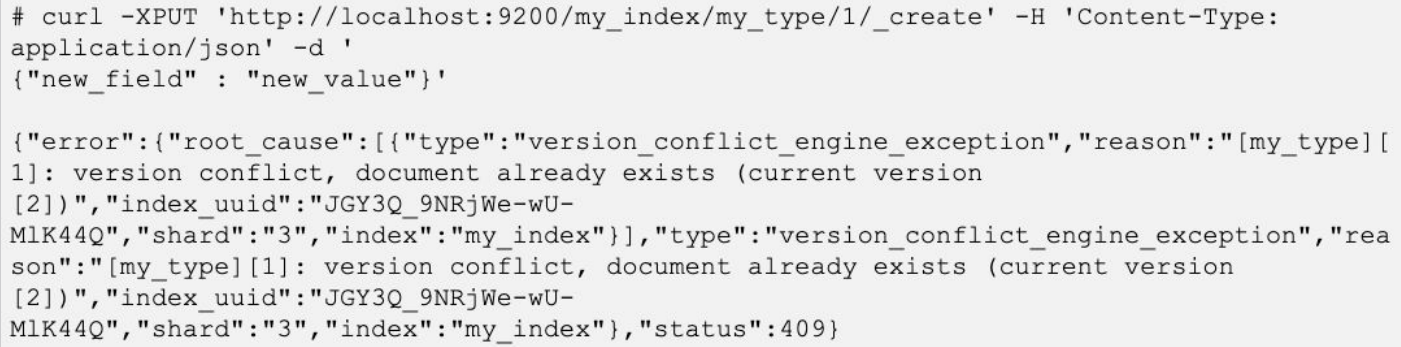
- The old field/value pairs of the document are gone
- the old document is deleted, and the new one gets indexed
- Notice every document has a _version that is incremented whenever the document is changed



```
# curl -XGET http://localhost:9200/my_index/my_type/1?pretty -H
'Content-Type: application/json'
{
  "_index" : "my_index",
  "_type" : "my_type",
  "_id" : "1",
  "_version" : 2,
  "found" : true,
  "_source" : {
    "new_field" : "new_value"
  }
}
```

The _create Endpoint

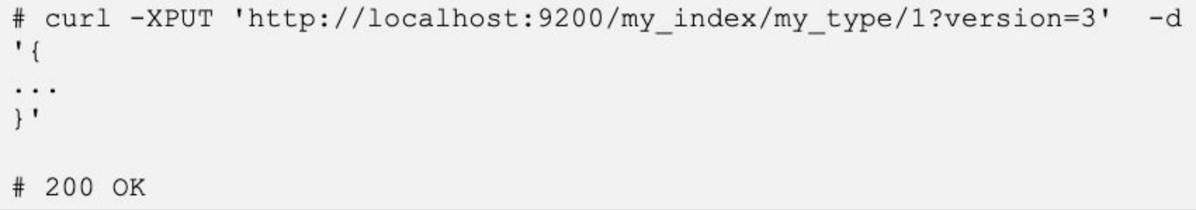
- If you do not want a document to be overwritten if it already exists, use the _create endpoint
- No indexing occurs and returns a 409 error message:



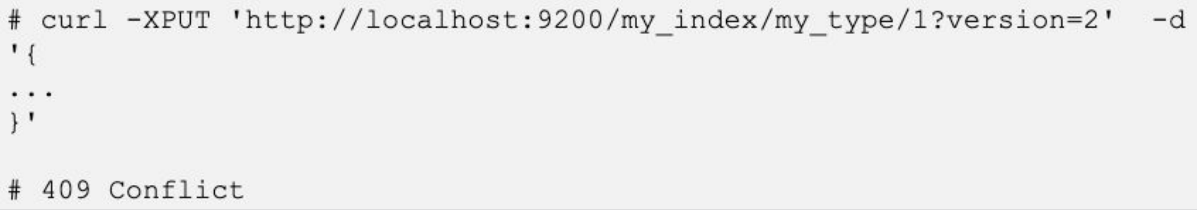
```
# curl -XPUT 'http://localhost:9200/my_index/my_type/1/_create' -H 'Content-Type:
application/json' -d '
{"new_field" : "new_value"}'

{"error":{"root_cause":[{"type":"version_conflict_engine_exception","reason":"[my_type][
1]: version conflict, document already exists (current version
[2])","index_uuid":"JGY3Q_9NRjWe-wU-
MlK44Q","shard":"3","index":"my_index"}],"type":"version_conflict_engine_exception","rea
son":"[my_type][1]: version conflict, document already exists (current version
[2])","index_uuid":"JGY3Q_9NRjWe-wU-
MlK44Q","shard":"3","index":"my_index"},"status":409}
```

Locking

- Every indexed document has a version number
- Elasticsearch uses Optimistic concurrency control **without** locking



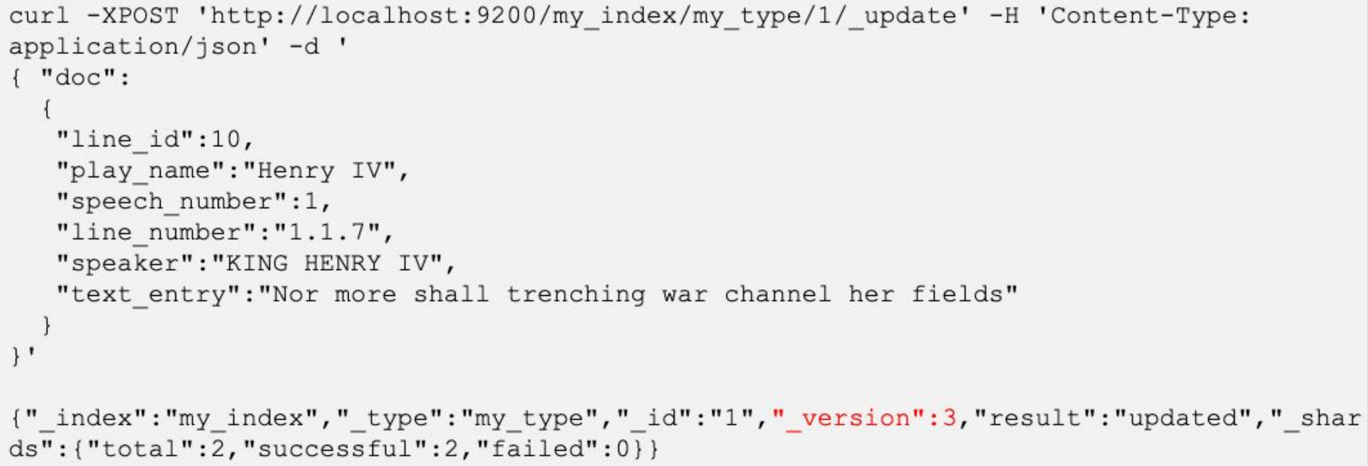
```
# curl -XPUT 'http://localhost:9200/my_index/my_type/1?version=3'  -d
'{
...
}'

# 200 OK
```



```
# curl -XPUT 'http://localhost:9200/my_index/my_type/1?version=2'  -d
'{
...
}'

# 409 Conflict
```

The _update Endpoint

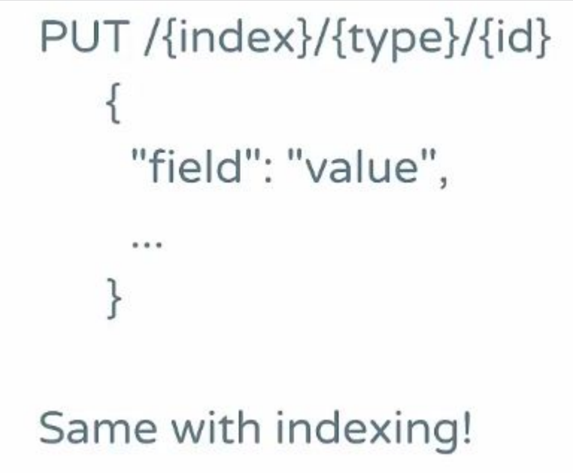
- To update fields in a document use the _update endpoint
  - Make sure to add the “doc” context



```
curl -XPOST 'http://localhost:9200/my_index/my_type/1/_update' -H 'Content-Type:
application/json' -d '
{ "doc":
  {
   "line_id":10,
   "play_name":"Henry IV",
   "speech_number":1,
   "line_number":"1.1.7",
   "speaker":"KING HENRY IV",
   "text_entry":"Nor more shall trenching war channel her fields"
  }
}'

{"_index":"my_index","_type":"my_type","_id":"1","_version":3,"result":"updated","_shar
ds":{"total":2,"successful":2,"failed":0}}
```

Update a Document

- Full update using the ID
- Same as indexing
- Try Examples on Kibana and observe response

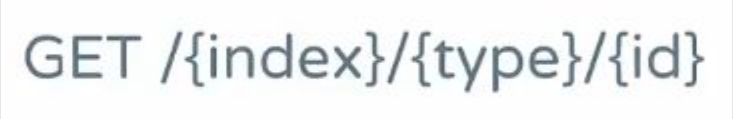


```
PUT /{index}/{type}/{id}
    {
      "field": "value",
      ...
    }
```

Same with indexing!

Retrieve a Document

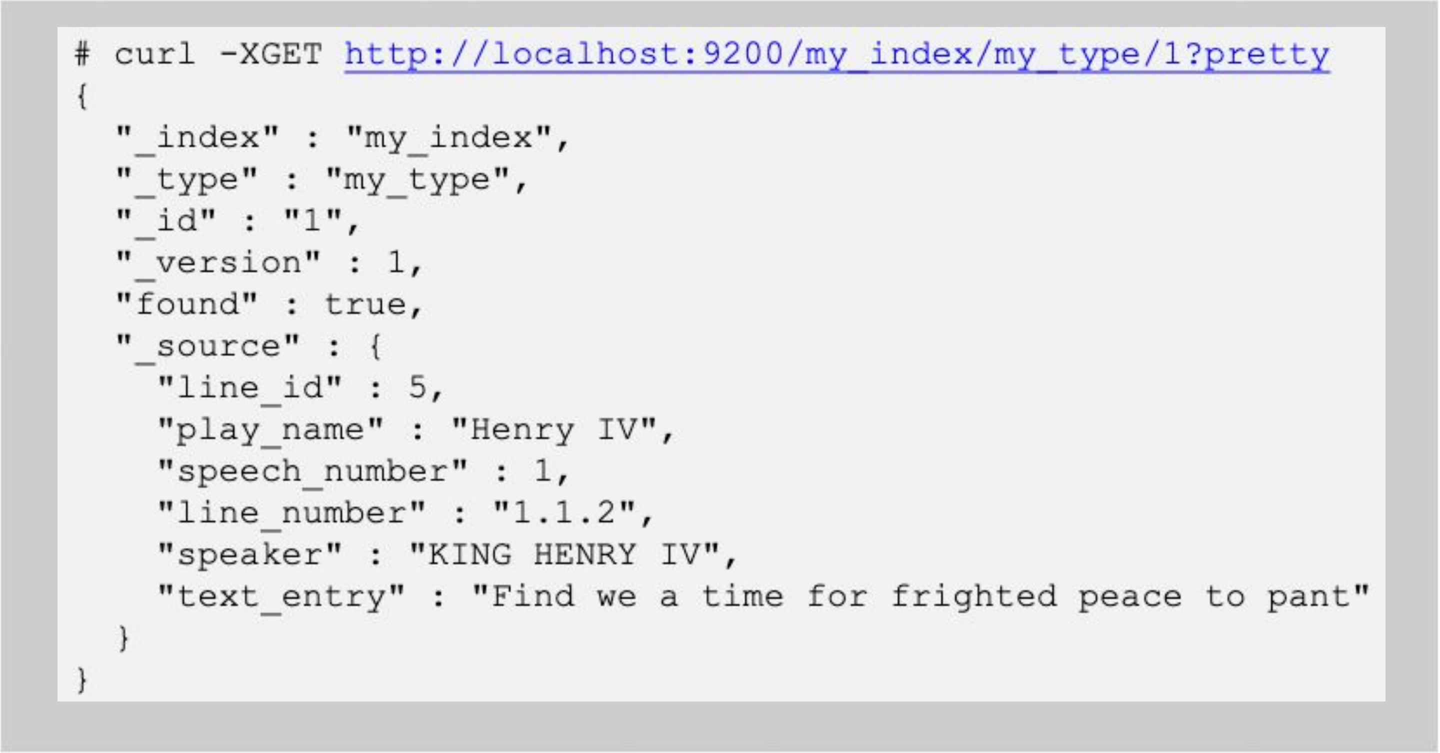
- Use GET to retrieve an indexed document
- Notice we specify the **index**, the **type** and an **ID**
- Returns a 200 code if document found or a 404 error if the document is not found



```
GET /{index}/{type}/{id}
```

```
GET quote_index/quote/2
GET quote_index/quote/2?_source=name,year
```

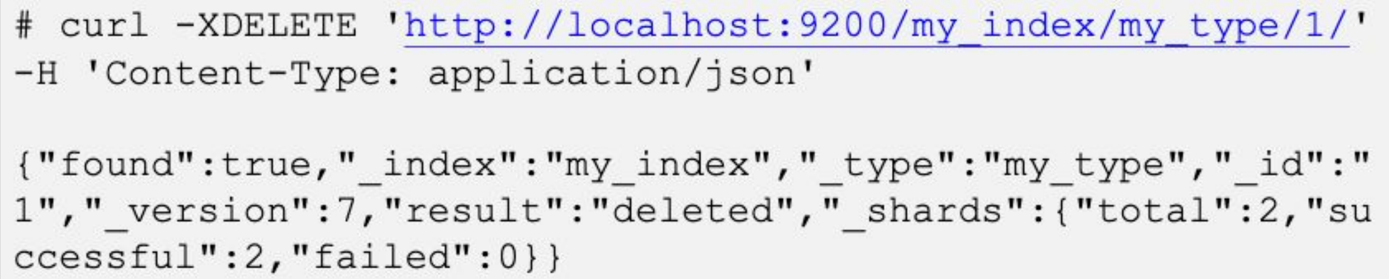
Retrieve a Document



```
# curl -XGET http://localhost:9200/my_index/my_type/1?pretty
{
  "_index" : "my_index",
  "_type" : "my_type",
  "_id" : "1",
  "_version" : 1,
  "found" : true,
  "_source" : {
    "line_id" : 5,
    "play_name" : "Henry IV",
    "speech_number" : 1,
    "line_number" : "1.1.2",
    "speaker" : "KING HENRY IV",
    "text_entry" : "Find we a time for frighted peace to pant"
  }
}
```

Delete a Document

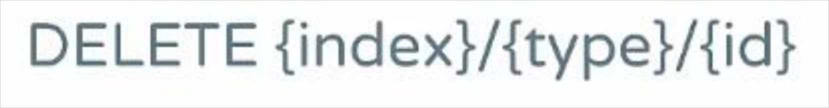
- Use DELETE to delete an indexed document
    - response code is **200** if the document is found, **404** if not



```
# curl -XDELETE 'http://localhost:9200/my_index/my_type/1/'
-H 'Content-Type: application/json'

{"found":true,"_index":"my_index","_type":"my_type","_id":"
1","_version":7,"result":"deleted","_shards":{"total":2,"su
ccessful":2,"failed":0}}
```

Delete a Document

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



```
DELETE {index}/{type}/{id}
```

Batch Processing

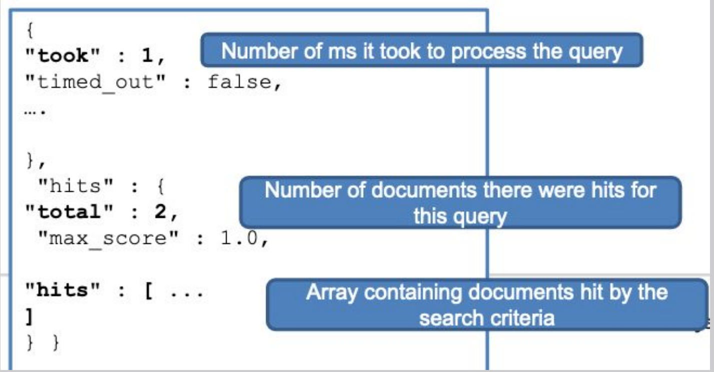
- For batch processing (i.e. performing actions on many documents) with a single query
- You use the bulk API
- The bulk API expects data using the NDJSON specification
- https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-bulk.html
- If you have a JSON file, all lines must end with ‘\n’ .. even the last line!
-
  ```
  curl -XPOST -uelastic:B6ZHn55SupJz0z5Wo2kE
  "http://34.236.187.15:9200/products/_bulk" -H
  'Content-Type: application/x-ndjson'
  --data-binary "@products-bulk.json"
  ```

A Simple Search

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

Search for all docs in my_index

```
curl -s -XGET 'http://localhost:9200/my_index/my_type/_search'
-H 'Content-Type: application/json'
```



```
{
"took" : 1,
"timed_out" : false,
….

},
 "hits" : {
"total" : 2,
 "max_score" : 1.0,

"hits" : [ ...
]
} }
```

Number of ms it took to process the query

Number of documents there were hits for this query

Array containing documents hit by the search criteria

Search Examples

- Query string query: Returns documents based on a provided query string, using a parser with a strict syntax
- https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-query-string-query.html
- https://coralogix.com/log-analytics-blog/42-elasticsearch-query-examples-hands-on-tutorial/
- https://www.tutorialspoint.com/elasticsearch/elasticsearch_query_dsl.htm

Python and R for ES

- Python Elasticsearch Client: https://elasticsearch-py.readthedocs.io/en/7.10.0/

- elasticsearchr: a Lightweight Elasticsearch Client for R: https://cran.r-project.org/web/packages/elasticsearchr/vignettes/quick_start.html

End of Part 2

More Hands-on with the ELK Stack

- Installing and configuring nginx to work as a reverse proxy so Kibana can be accessed on the internet
- Using Logstash to collect static Apache logs and analyzing them using Kibana
- Using Logstash to collect static .CSV file and analyzing its data using Kibana
- Collecting real-time web-logs, configuring Beats to upload them to Elasticsearch and analyzing them using Kibana
- Monitoring the performance of the Elastic Stack

Install and Configure Nginx on Ubuntu

```
sudo apt-get install -y nginx
sudo systemctl enable nginx
```

- Configure `nginx` as a reverse proxy for kibana
- See file logstash_conf/nginx-kibana.txt on the Github Repo

Filebeat (ELK for Nginx)

- List FB modules: `filebeat modules list`
- Enable FB modules: `filebeat modules enable nginx`
- In `nginx.yml` add:

  ```
  var.paths: ["/var/log/nginx/access.log*"]
  var.paths: ["/var/log/nginx/error.log*"]
  ```

- Enable FB modules: `filebeat modules enable system`
- in `system.yml` add:

  ```
  var.paths: ["/var/log/syslog*"]
  var.paths: ["/var/log/auth.log*"]
  ```

- Start filebeat
- To load dashboards: `filebeat setup -e`

Bonus

X-Pack

- X-Pack is an Elastic Stack extension that provides security, alerting, monitoring, reporting, machine learning, and many other capabilities
- By default, when you install Elasticsearch, X-Pack is installed
- If you want to try all of the X-Pack features, you can start a 30-day trial
- At the end of the trial period, you can purchase a subscription to keep using the full functionality of the X-Pack components

https://www.elastic.co/guide/en/elasticsearch/reference/current/setup-xpack.html

ELK Stack Security

- Add `xpack.security.enabled: true` to elasticsearch.yml
- Run `bin/elasticsearch-setup-passwords auto` to set passwords for built-in users
- Built-in users: https://www.elastic.co/guide/en/elasticsearch/reference/current/built-in-users.html
- In kibana.yml update these two fields accordingly:
- `elasticsearch.username: "kibana_system"`
- `elasticsearch.password: "password"`

ELK Stack Security

For LS to push data into ES, you need to add the following fields to the LS pipeline (in the output section):

```
user => "elastic"
password => "B6ZHn55SupJz0z5Wo2kE"
```

For filebeat .. update the `filebeat.yml` file (you can create user with appropriate permissions):

```
output.elasticsearch:
hosts: ["https://myEShost:9200"]
username: "elastic"
password: "password"
```

рахмат
danke
謝謝
ngiyabonga
spas
teşekkür ederim
Баярлалаа
mersi
kia ora
barka
welalin
tack
mahalo
спасибо
faafetai lava
vinaka
dank je
misaotra
paldies
grazzi
tapadh leat
спасибі
blagodaram
matondo
kiitos
dankie
thank you
gracias
хвала
nanni
nandri
dhanyavad
asante
manana
mauruuru
aciu
obrigada
tenki
bedankt
bayarlalaa
gràçie
hvala
köszönöm
dankon
murakoze
akun
djiere dieuf
tau
mochchakkeram
enkosi
дякую
mamnun
dziękuję
chnorakaloutioun
gratias ago
gràcies
sulpáy
chokrane
go raibh maith agat
sobodi
dēkuji
taiku
sagolun
sukriya
kop khun krap
obrigado
takk
arigatô
dakujem
trugarez
mèsi
ありがとう
didi madloba
najis tuke
tanemirt
grazie
terima kasih
rahmet
shukriya
mercê
мерси
kam sah hamnida
rahmat
dhanyavadagalu
diolch
তোমাকে ধন্যবাদ
감사합니다
xièxie
ευχαριστώ
merci