ELASTIC DELPHI

Using An Elasticsearch Document Store With Delphi

NIGEL TAVENDALE

- Programing Delphi since 1997
- Using Elasticsearch for 18 months
- Management of large datasets

nigel.tavendale@allthingssyslog.com

OBJECTIVES

- Demystify Elasticsearch
- Make documentation & code samples obvious
- Get enough information to do a proof of concept if someone asks
- Be able to answer questions about it

NIGEL TAVENDALE
www.allthingssyslog.com

DOWNLOAD ELASTICSEARCH

https://www.elastic.co/downloads/elasticsearch

CODE SAMPLES AVAILABLE AT:

https://github.com/ntavendale/ElasticDelphi

ELASTIC EXPLORER AVAILABLE FROM:

http://www.allthingssyslog.com/ElasticExplorer/

NIGEL TAVENDALE

www.allthingssyslog.com

ALL THINGS SYSLOG

STRUCTURED vs. UNSTRUCTURED DATA

<13>Jul 29 11:46:07 ServerMonitor MonitorProcess[2559]: EVID0039 DBServer4: SqlServer stopped with process identifier 3232

STRUCTURED HEADER (RFC-3164)

<13>Jul 29 11:46:07 ServerMonitor MonitorProcess[2559]:

Priority DateTime Sending Host Process [ProcessID]:

UNSTRUCTURED MESSAGE TEXT

EVID0039 DBServer4: SqlServer stopped with process identifier 3232

Is there a problem with DBServer4?

Or is there a problem with SQL Server?

STRUCTURED vs. UNSTRUCTURED DATA

In a relational database…..

MESSAGES

<13>Jul 29 11:46:07 ServerMonitor MonitorProcess[2559]: EVID0039 DBServer4: SqlServer stopped with process identifier 3232

CATAGORIES

1. SERVER PROBLEMS
2. SQL DB PROBLEMS

CATEGORY TO MESSAGE

Category, Message

Category, Message

INTRODUCING ELASTICSEARCH

Elasticsearch is a document store, not a database.

- Not ACID Compliant. Changes to individual documents are ACIDic, but not multiple.
- No such thing as stored procedures.
- Allows nested data and links, but does not do joins when retrieving data.
- Primarily intended for indexing data and making it searchable.
- Works best with data that is rarely updated (logs, reports, etc).

THE LIBRARY MODEL

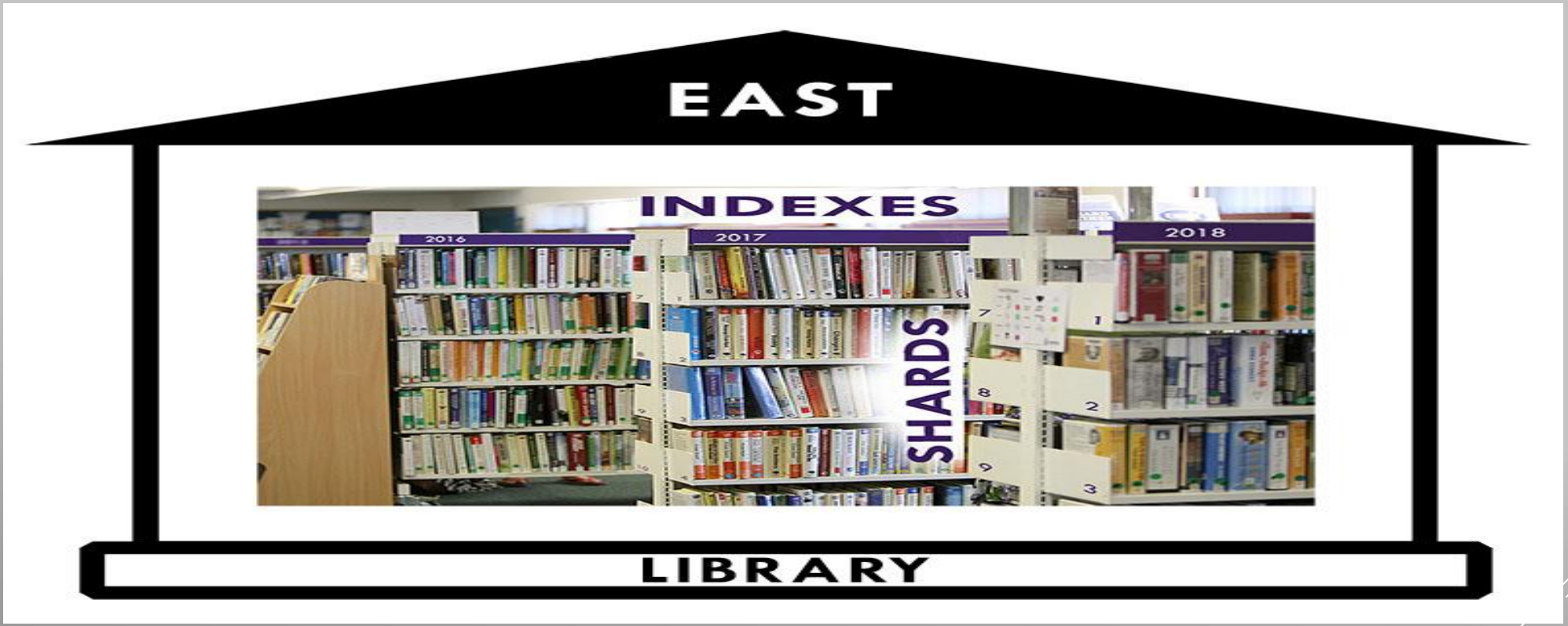
Elasticsearch can be thought of as a big library containing documents grouped into sections called indexes.

DATA COMMONLY GROUPED IN INDEXES BY DATE

Indexes are made up of multiple pieces called shards.

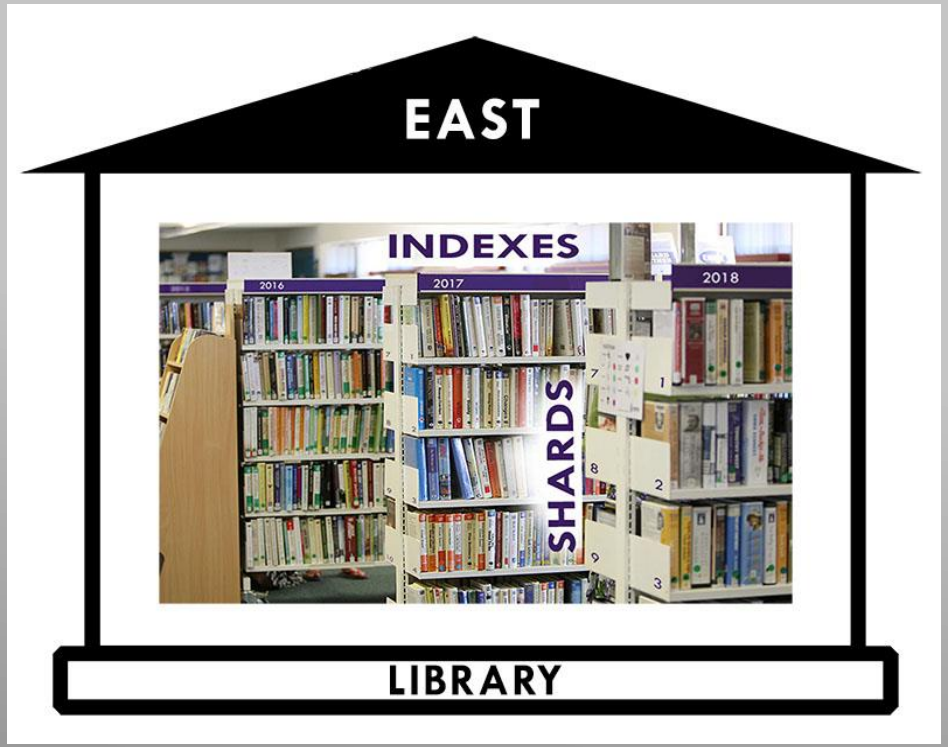
SHARDS CAN BE DISTRIBUTED

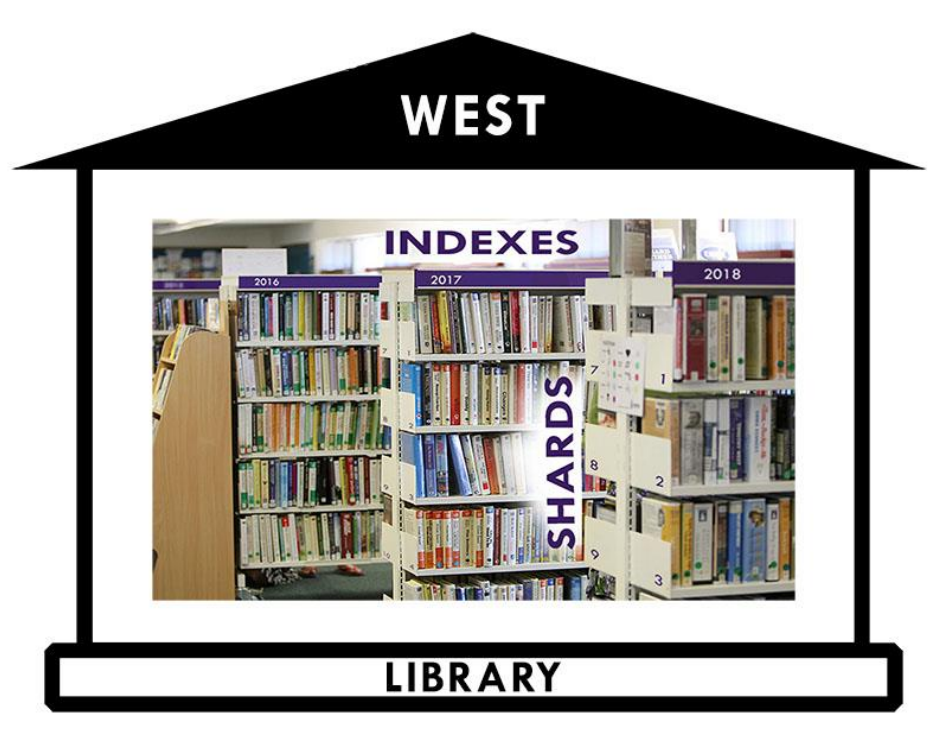
No need to make the library bigger...

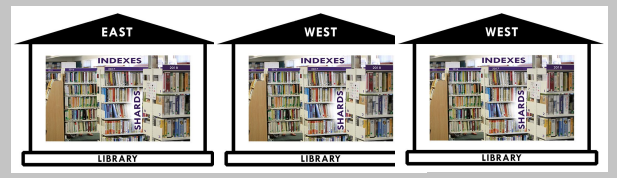


SHARDS CAN BE DISTRIBUTED

Just add another building and form a cluster!

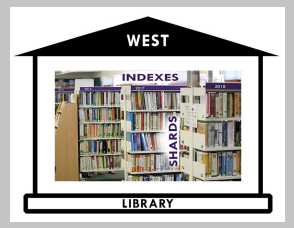






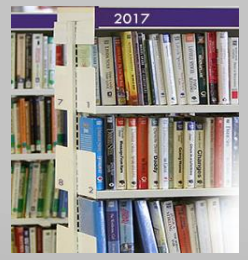
LIBRARY SYSTEM

CLUSTER



LIBRARY

NODE

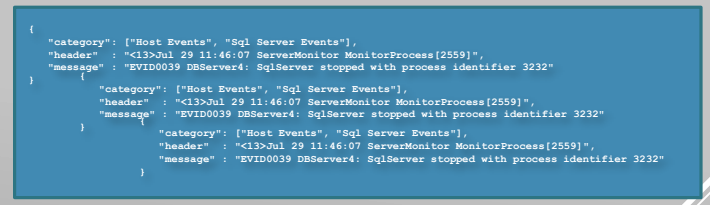


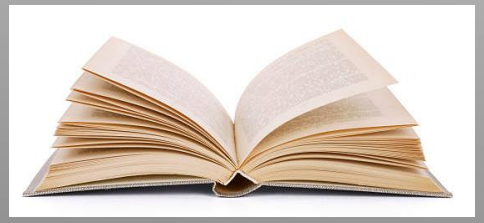
SECTION

INDEX

SHELF

SHARD





BOOK

DOCUMENT

```
{
   "category": ["Host Events", "Sql Server Events"],
   "header"  : "<13>Jul 29 11:46:07 ServerMonitor
MonitorProcess[2559]",
   "message" : "EVID0039 DBServer4: SqlServer stopped
with process identifier 3232"
}
```

DATA IS PARSED INTO DOCUMENTS

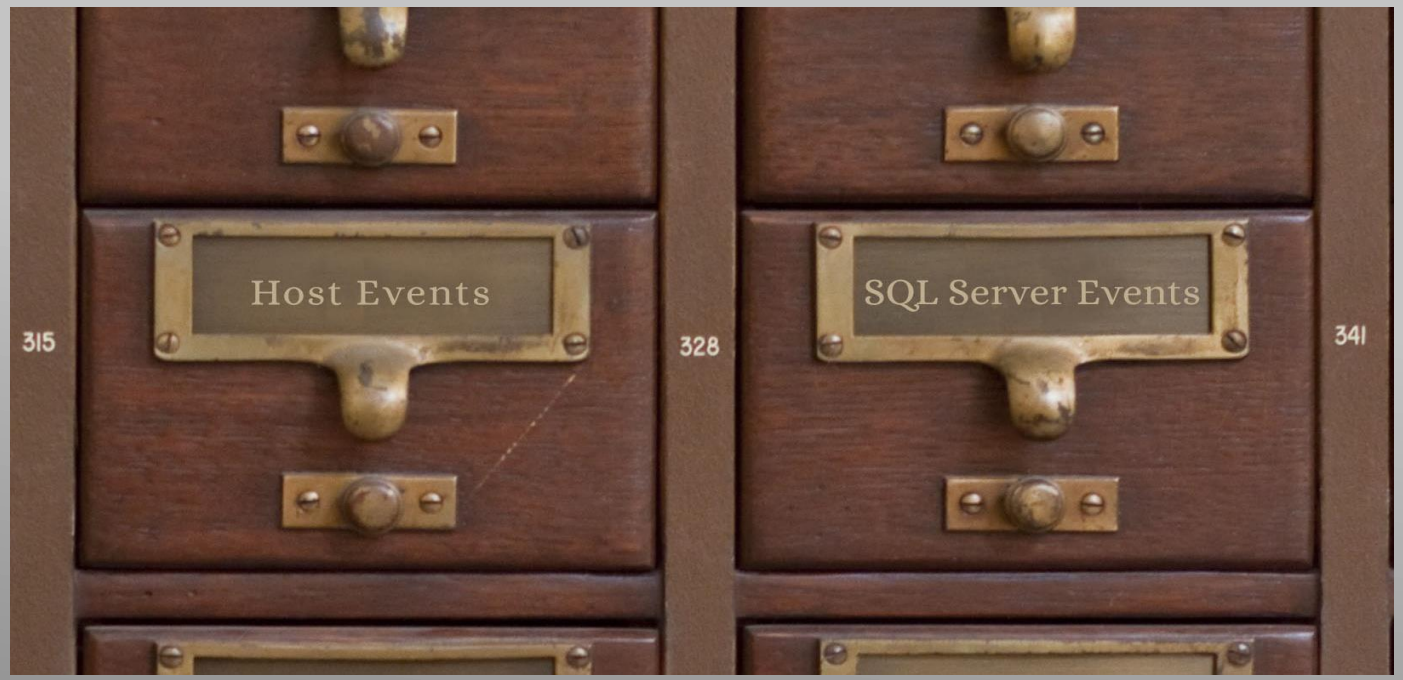
<13>Jul 29 11:46:07 ServerMonitor MonitorProcess[2559]: EVID0039 DBServer4: SqlServer stopped with process identifier 3232

```
{
   "category": ["Host Events", "Sql Server Events"],
   "header"  : "<13>Jul 29 11:46:07 ServerMonitor MonitorProcess[2559]",
   "message" : "EVID0039 DBServer4: SqlServer stopped with process identifier 3232"
}
```

Documents are then added to the cluster.

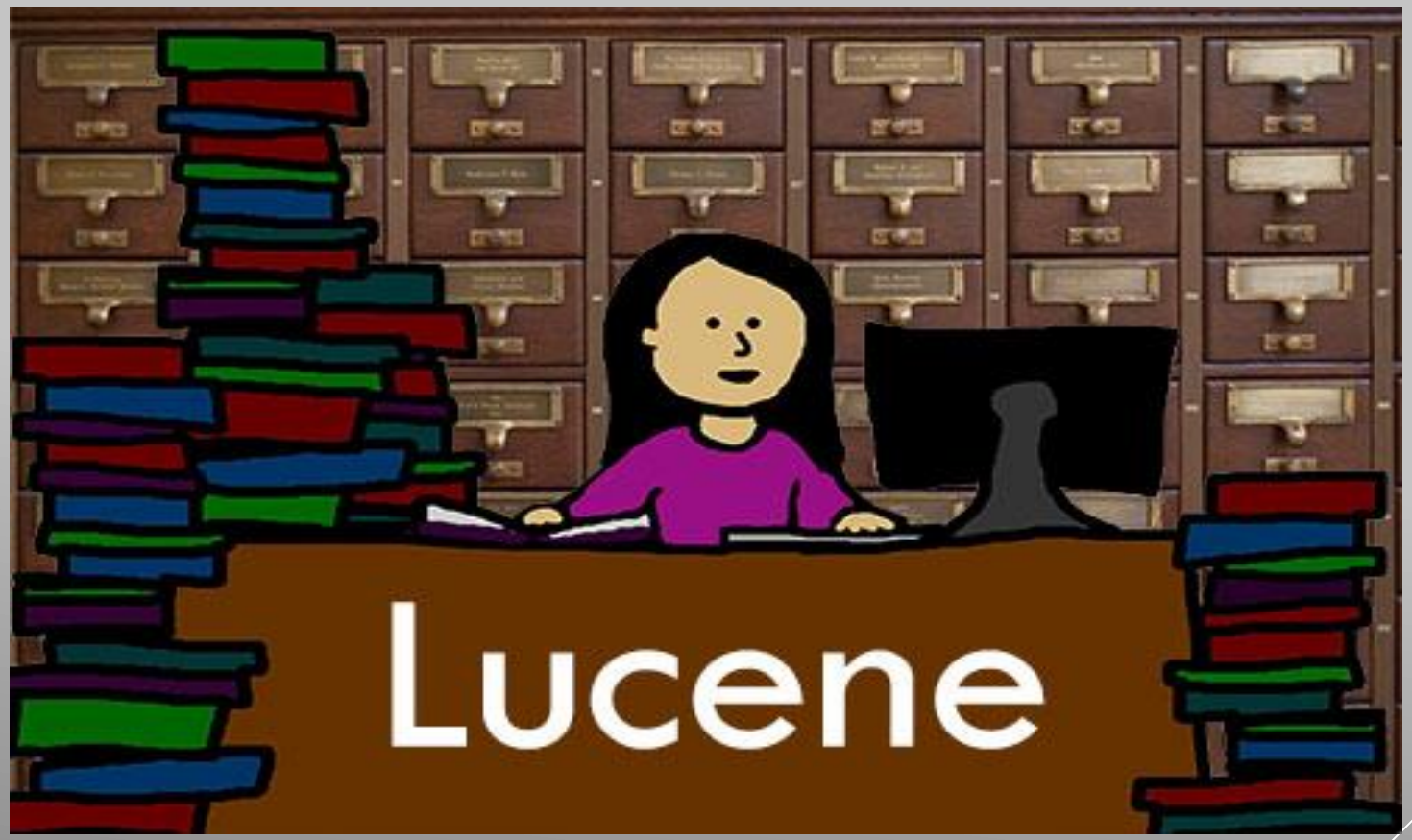
DOCUMENTS ARE INDEXED IN ELASTICSEARCH

```
{
   "category": ["Host Events", "Sql Server Events"],
   "header"  : "<13>Jul 29 11:46:07 ServerMonitor MonitorProcess[2559]",
   "message" : "EVID0039 DBServer4: SqlServer stopped with process identifier 3232"
}
```



INDEXES ARE MANAGED BY LUCENE

Your friendly Elasticsearch librarian!



APACHE LUCENE

- Indexing system originally written in Java.
- Ported to other languages. C#, Python, C++, even Object Pascal!
- Written by Doug Cutting. Named for his wife's middle name.
- Has it's own query syntax, different from elasticsearch.

USED IN OTHER PROJECTS

- Compass
- Solr
- CrateDB

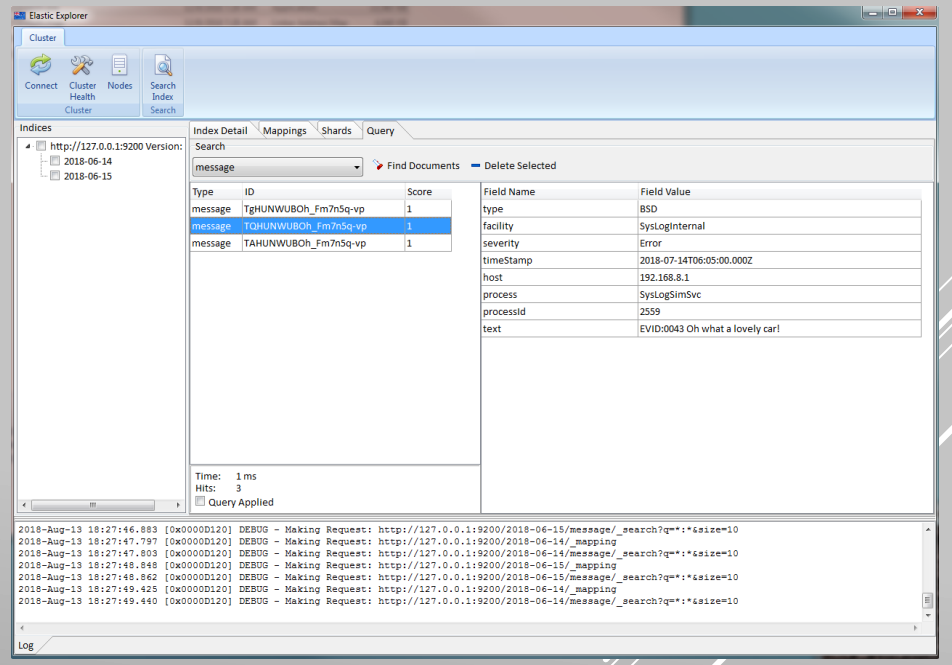
THE ELASTICSEARCH API

- REST API over HTTP
- Documents, meta data, settings almost all described using JSON (except _cat operations)
- No built in security out of the box
- HEAD, GET, PUT, POST, DELETE
- Most examples are using CURL, and I *hate* CURL
- Instead, I use Elastic Explorer.

ELASTIC EXPLORER

Or how I learned what I know about Elasticsearch

- Visual representation of cluster
- Check health of cluster
- Provide information on shards, nodes and disk usage
- Create mappings
- Run & test queries against a single index or the entire cluster
- Free to download and use.
- http://www.allthingssyslog.com/ElasticExplorer/



CHECK INDEX EXISTS

Use http HEAD command

- 200 – Index exists
- 404 – Index does not exist

CREATE AN INDEX

Use http PUT command. 200 or 201 equals success.

```
{
  "settings" : {
    "index" : {
       "number_of_shards" : 5,
       "number_of_replicas" : 2
    }
  }
}
```

Index name must be lower case, cannot start with "_", or contain commas.

ADDING OR UPDATING A DOCUMENT

```
{
  "type":"BSD",
  "facility":"SystemDaemon",
  "severity":"Emergency",
  "timeStamp":"2018-07-02T18:24:02.662Z",
  "host":"localhost",
  "process":"MyProcess",
  "processId":99,
  "text":"Oops! We have an emergency!"
}
```

If we supply the Document ID adding a document is done with a PUT

If we want Elasticsearch to auto generate a Document ID we must use a POST

When updating we have the Document ID and always use a PUT

THE BULK API (/_bulk)

Index multiple documents using a POST. Separated by newline.

Single Index (/20180614/message/_bulk):

```
{"index":{}}\n
{"type":"BSD","facility":"MailSystem","severity":"Critical","timeStamp":"2018-06-
14T06:00:00.000Z","host":"192.168.8.1","process":"SysLogSimSvc","processId":2559,
' + '"text":"EVID:0018 Reconnaissance activity detected 111.148.118.9:40083 ->
161.200.1.9:443 TCP"}\n
{"index":{}}\n
{"type":"BSD","facility":"SysLogInternal","severity":"Error","timeStamp":"2018-
06-
14T06:05:00.000Z","host":"192.168.8.1","process":"SysLogSimSvc","processId":2559,
"text":"EVID:0043 Host: 172.10.1.14 has a vulnerability on port: 80 protocol:
http"}\n
```


THE BULK API (/_bulk)

Index multiple documents using a POST. Separated by newline.

Multiple Index (/_bulk):

```
{"index":{"_index":"2018-06-14", "_type":"message"}}\n
{"type":"BSD","facility":"MailSystem","severity":"Critical","timeStamp":"2018-06-
14T06:00:00.000Z","host":"192.168.8.1","process":"SysLogSimSvc","processId":2559,
' + '"text":"EVID:0018 Reconnaissance activity detected 111.148.118.9:40083 ->
161.200.1.9:443 TCP"}\n
{"index":{"_index":"2018-06-16", "_type":"message"}}\n
{"type":"BSD","facility":"SysLogInternal","severity":"Error","timeStamp":"2018-
06-
16T06:05:00.000Z","host":"192.168.8.1","process":"SysLogSimSvc","processId":2559,
"text":"EVID:0043 Host: 172.10.1.14 has a vulnerability on port: 80 protocol:
http"}\n
```

QUERYING ELASTICSEARCH

Query DSL (Domain Specific Language) based on JSON.

- Query Context – How well does it match? *Results are scored.*
- Filter Context – Does it match yes/no? *No score.*

Filters highly dependent on initial mapping creation and analyzer selection.

Elasticsearch not schema-less. Examples seen here rely on ***dynamic*** schema creation.

Possible to define schema by adding mappings prior to creating documents.

A SIMPLE MATCH QUERY

Match breaks query up into tokens then searches the named field for the tokens. Default search type is OR. Very simple, doesn't support wildcards for example, so less likely to fail.

Message 1:

Search For "EVID what day"

Search In "EVID:0018 Oh what a lovely day!"

Message 2:

Search For "EVID what day"

Search In "EVID:0043 Oh what a lovely car!"

Message 3:

Search For "EVID what day"

Search In "EVID:0042 111.148.118.9 accessed url: http://Website001.com at UserPC5"

A SIMPLE MATCH QUERY

Since the default search type is OR. We don't need to specify it, but we can.

Simple Query Syntax (Default OR)

```
{
    "query": {
        "match":{ "text": "EVID what day"}
    }
}
```

Equivalent To The Following:

```
{
    "query": {
        "match" : {
            "text" : {
                "query":"EVID what day",
                "operator" : "or"
            }
        }
    }
}
```

A SIMPLE MATCH QUERY

We can also use a match query with an AND condition. In this case *all* of the tokens must be present and only documents with all present are returned.

```
{
    "query": {
        "match" : {
            "text" : {
                "query":"EVID what day",
                "operator" : "and"
            }
        }
    }
}
```


WHERE TO FROM HERE?

Elasticsearch

http://www.elastic.co/learn

Mappings

https://www.elastic.co/blog/found-elasticsearch-mapping-introduction

Analyzers

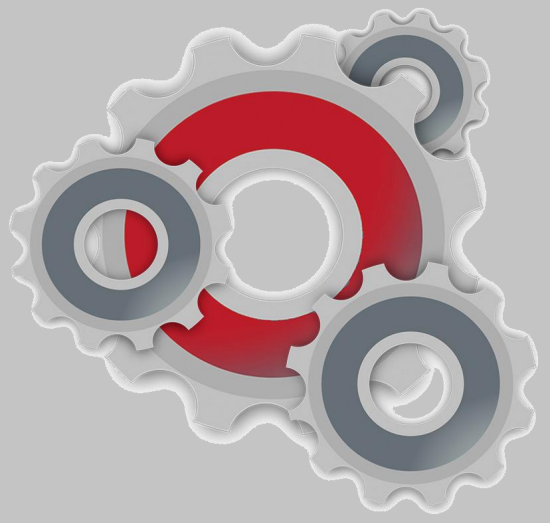
https://www.elastic.co/blog/found-text-analysis-part-1

https://www.elastic.co/blog/found-text-analysis-part-2

Queries

https://www.elastic.co/blog/found-beginner-troubleshooting

RESOURCES



All Things Syslog:

http://www.allthingssyslog.com/

Code samples available on GitHub:

https://github.com/ntavendale/Elastic.NET/

Elastic Explorer available from:

http://www.allthingssyslog.com/ElasticExplorer/

THANK YOU!