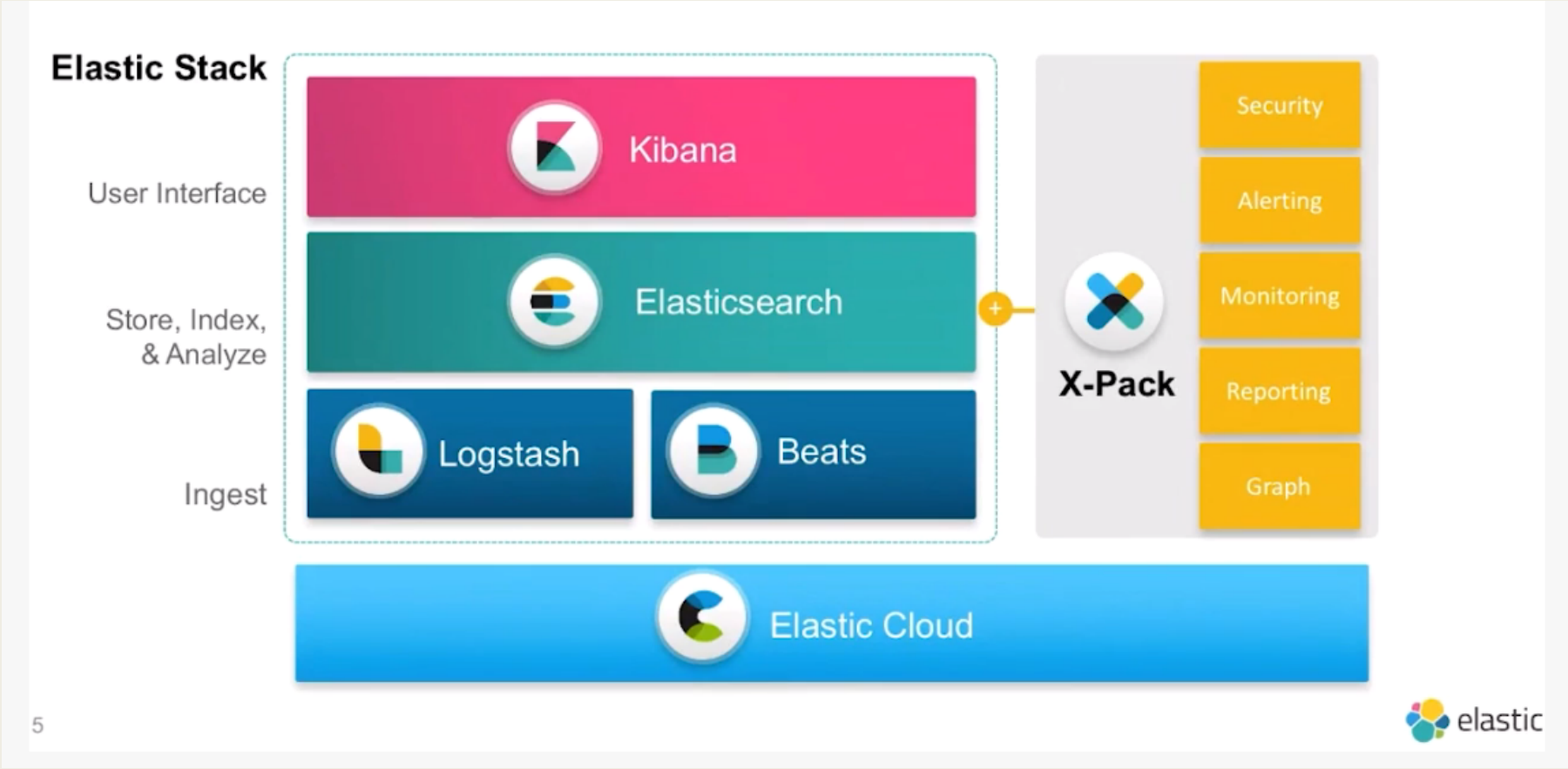
ELASTIC STACK

A Brief Introduction

What is ELK Stack?

Elastic Stack



Elasticsearch

- Elasticsearch is a highly scalable open-source full-text search and analytics engine. It allows you to store, search, and analyze big volumes of data quickly and in near real time. It is generally used as the underlying engine/technology that powers applications that have complex search features and requirements.

Logstash

- Logstash is an open source data collection engine with real-time pipelining capabilities. Logstash can dynamically unify data from disparate sources and normalize the data into destinations of your choice. Cleanse and democratize all your data for diverse advanced downstream analytics and visualization use cases.

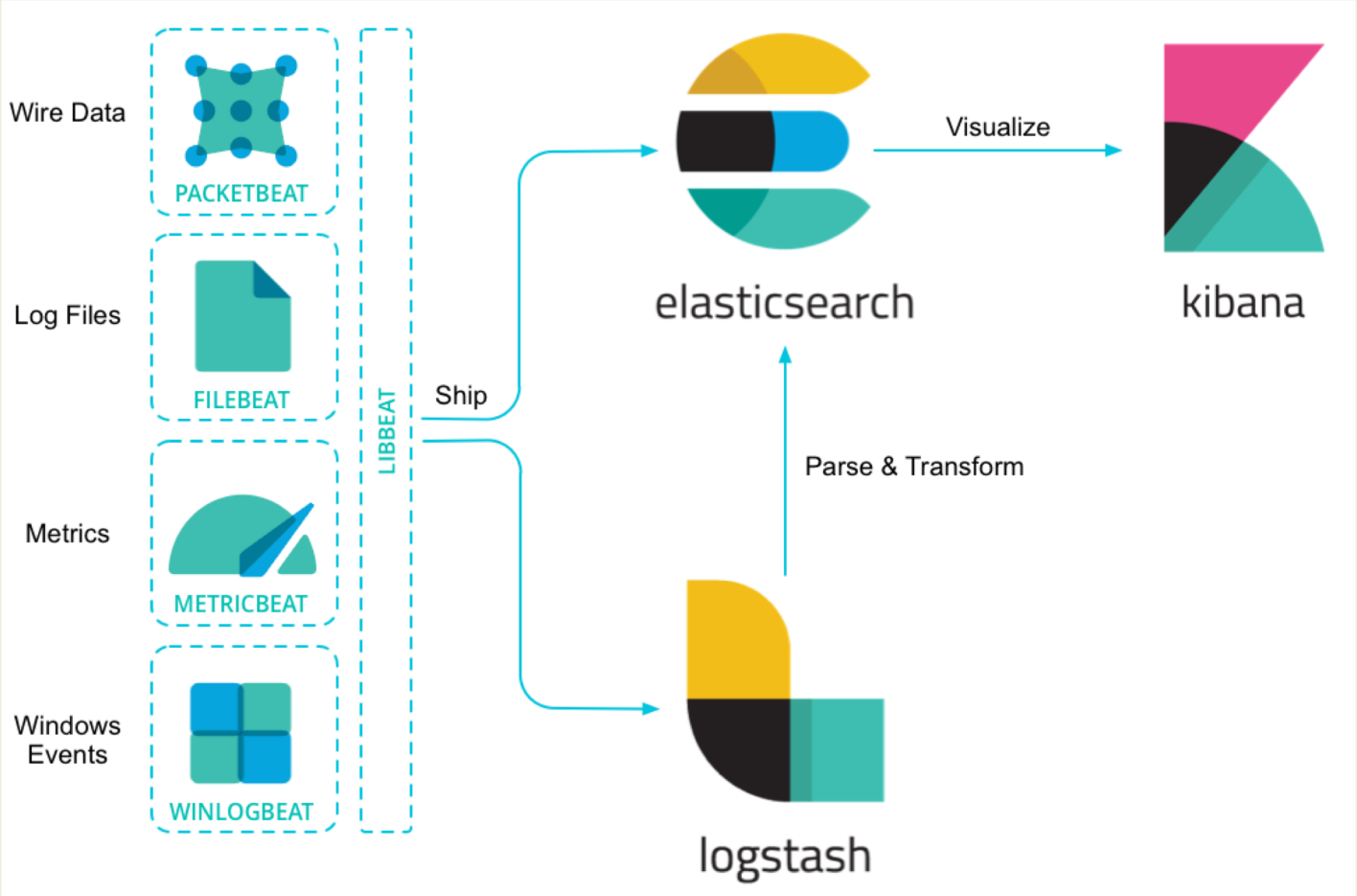
Kibana

- Kibana is an open source analytics and visualization platform designed to work with Elasticsearch. You use Kibana to search, view, and interact with data stored in Elasticsearch indices. You can easily perform advanced data analysis and visualize your data in a variety of charts, tables, and maps.

Beats & X-Pack

- The *Beats* are open source data shippers that you install as *agents* on your servers to send different types of operational data to Elasticsearch. Beats can send data directly to Elasticsearch or send it to Elasticsearch via Logstash, which you can use to parse and transform the data.
- X-Pack is an Elastic Stack extension that bundles security, alerting, monitoring, reporting, and graph capabilities into one easy-to-install package. While the X-Pack components are designed to work together seamlessly, you can easily enable or disable the features you want to use.

Data flow of Elastic Stack



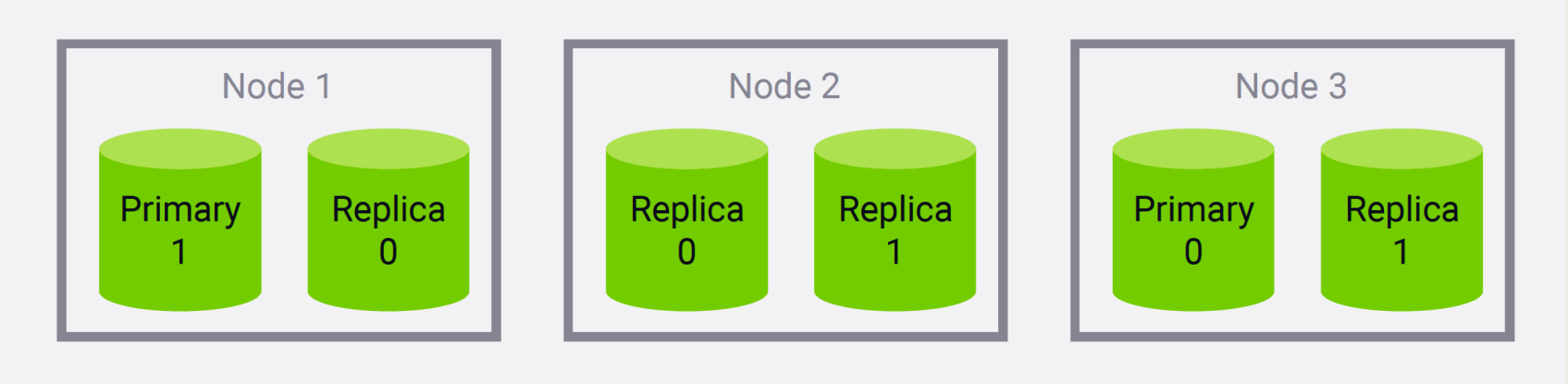
ELASTICSEARCH

Terms - data

- Index
- Type
- Document
- Field
- Mapping

Terms - architecture

- Shard (default 5)
- Replica (default 1)
- Node
- Cluster



RESTful API

- Elasticsearch provides a handful of RESTful API
- CRUD
- POST, GET, PUT, DELETE
- Basic form (using curl)

```
curl –H 'Content-Type: application/json' -XGET '127.0.0.1:9200/movies/movie/_search' -d '
{
    "query": {
        "match": {
            "year" : 2014
        }
    }
}'
```

Mapping

- Define the schema of the documents (type)
  - *Field*
  - *Field type*
    - text, keyword, byte, short, integer, long, float, double, boolean, date, binary
    - https://www.elastic.co/guide/en/elasticsearch/reference/6.2/mapping-types.html
  - *Field analyzer*

Create a mapping

```
curl -XPUT '127.0.0.1:9200/movies?pretty' -d '
{
  "mappings": {
    "movie": {
      "properties" : {
        "year" : {"type": "date"}
      }
    }
  }
}'
```

Get the mapping

```
curl -XGET '127.0.0.1:9200/movies/movie/_mapping'
```

Insert a single document

```
curl -XPUT '127.0.0.1:9200/movies/movie/112556' -d '
{
  "genre": ["Drama", "Thriller"],
  "title": "Gone Girl",
  "year": 2014
}'
```

Retrieve a single document

```
curl -XGET '127.0.0.1:9200/movies/movie/112556'
```

Search the entire index

```
curl -XGET '127.0.0.1:9200/movies/_search?size=20'
```

Update a document

```
curl -XPOST '127.0.0.1:9200/movies/movie/112556/_update' -d '
{
  "doc": {
    "title": "Gone boy"
  }
}'
```

Update a mapping

```
curl -XPUT '127.0.0.1:9200/movies/_mapping/movie' -d '
{
  "properties": {
    "rating": {
      "type": "float"
    }
  }
}'
```

Delete a document

```
curl -XDELETE '127.0.0.1:9200/movies/movie/112556'
```

Delete an index

```
curl -XDELETE '127.0.0.1:9200/movies'
```

List all indices

```
curl -XGET '127.0.0.1:9200/_cat/indices?v'
```

Insert a bulk of documents

```
curl -XPUT '127.0.0.1:9200/_bulk' -d '

{ "create" : { "_index" : "movies", "_type" : "movie", "_id" : "135569" } }
{ "id": "135569", "title" : "Star Trek Beyond", "year":2016 , "genre":["Action", "Adventure", "Sci-Fi"] }
{ "create" : { "_index" : "movies", "_type" : "movie", "_id" : "122886" } }
{ "id": "122886", "title" : "Star Wars: Episode VII -The Force Awakens", "year":2015 , "genre":["Action",
"Adventure", "Fantasy", "Sci-Fi", "IMAX"] }
{ "create" : { "_index" : "movies", "_type" : "movie", "_id" : "109487" } }
{ "id": "109487", "title" : "Interstellar", "year":2014 , "genre":["Sci-Fi", "IMAX"] }
{ "create" : { "_index" : "movies", "_type" : "movie", "_id" : "58559" } }
{ "id": "58559", "title" : "Dark Knight, The", "year":2008 , "genre":["Action", "Crime", "Drama", "IMAX"] }
{ "create" : { "_index" : "movies", "_type" : "movie", "_id" : "1924" } }
{ "id": "1924", "title" : "Plan 9 from Outer Space", "year":1959 , "genre":["Horror", "Sci-Fi"] }'
```

- Import documents from file

```
curl -XPUT '127.0.0.1:9200/_bulk' --data-binary @movie.json
```

Exercise

- Import csv file into Elasticsearch using Python scripts
  - *Deal with JSON format directly*
  - *Use Python elasticsearch package*

SEARCH

Analyzer

- Character filter
  - *HTML strip, mapping, pattern replace*
- Tokenizer
  - *https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-tokenizers.html*
- Token filter
  - *Standard, ASCII folding*

What does a standard analyzer do?

- It provides grammar based tokenization (based on the Unicode Text Segmentation algorithm, as specified in Unicode Standard Annex #29) and works well for most languages.

```
{
  "analyzer": "standard",
  "text": "The 2 QUICK Brown-Foxes jumped over the lazy dog's bone."
}
```

```
[ the, 2, quick, brown, foxes, jumped, over, the, lazy, dog's, bone ]
```

Search the documents

- Query line search

```
curl -XGET '127.0.0.1:9200/movies/_search?q=star'
```

- Request body search

```
curl -XGET '127.0.0.1:9200/movies/movie/_search' -d '
{
  "query": {
    "match": {
      "title": "star"
    }
  }
}'
```

Query level

- Term level - Yes or no?
  - *Range*
  - *Exists*
  - *missing*
  - *Term*
  - *Terms*
- High level - Which is more relevant?
  - *Match*
  - *Match phrase*
  - *Multi match*

Term level query

```
curl -XGET '127.0.0.1:9200/movies/movie/_search' -d '
{
  "query": {
    "range": {
      "year": {
        "gte": 2015,
        "lte": 2017
      }
    }
  }
}'
```

Term level query

```
curl -XGET '127.0.0.1:9200/movies/movie/_search' -d '
{
  "query": {
    "term": {
      "title": "star"
    }
  }
}'
```

High level query

```
curl -XGET '127.0.0.1:9200/movies/movie/_search' -d '
{
  "query": {
    "match": {
      "title": "star dark"
    }
  }
}'
```

High level query

```
curl -XGET '127.0.0.1:9200/movies/movie/_search' -d '
{
  "query": {
    "match_phrase": {
      "title": "star trek"
    }
  }
}'
```

Bool query

```
curl -XGET '127.0.0.1:9200/movies/movie/_search' -d '
{
  "query": {
    "bool": {
      "must": [...],
      "should": [...],
      "must_not": {...},
      "filter": {...}
    }
  }
}'
```

Examples

```
curl -XGET '127.0.0.1:9200/movies/_search' -d'
{
  "query": {
    "bool": {
      "must": {"match": {"genre": "Sci-Fi"}},
      "must_not": {"match": {"title": "trek"}},
      "filter": {"range": {"year": {"gte": 2010, "lt": 2015}}}
    }
  }
}'
```

Other usage

- Slop
- Fuzziness
- Sorting
- Pagination
- Partial match (wildcard, prefix)

AGGREGATIONS

Types of aggregations

- Bucket aggregations
  - *group count, histogram, ranges, etc.*
- Metric aggregations
  - *sum, average, min, max, etc.*
- Pipeline aggregations
- Matrix aggregations

Bucket aggregations

```
curl -XGET '127.0.0.1:9200/products/product/_search' -d '
{
  "aggs": {
    "byCategory": {
      "terms": {
        "field": "category"
      }
    }
  },
  "size": 0
}'
```

Bucket aggregations

```
curl -XGET '127.0.0.1:9200/products/product/_search' -d '
{
  "aggs": {
    "byPrice": {
      "histogram": {
        "field": "price",
        "interval": 100
      }
    }
  },
  "size": 0
}'
```

Bucket aggregations

```
curl -XGET '127.0.0.1:9200/products/product/_search?size=0' -d '
{
  "aggs": {
    "byPrice": {
      "range": {
        "field": "price",
        "ranges": [
          { "to": 100 },
          { "from": 100, "to": 1000 },
          { "from": 1000 }
      }
    }
  }
}'
```

Metric aggregations

```
curl -XGET '127.0.0.1:9200/products/product/_search?size=0' -d '
{
  "aggs": {
    "revenueSum": {
      "sum": {
        "field": "revenue"
      }
    }
  }
}'
```

* replace "sum" with "avg", "min", "max", "stats", "extended_stats", etc.

Pipeline aggregations

```
curl -XGET '127.0.0.1:9200/products/product/_search?size=0' -d '
{
  "aggs": {
    "byProducer": {
      "terms": {
        "field": "producer"
      }
    },
    "aggs": {
      "avgPrice": {
        "avg": {
          "filed": "price"
        }
      }
    }
  }
}'
```

LOGSTASH

Logstash

- Use a config file to define
  - *Where to collect data (input)*
  - *How to process data (filter)*
  - *Where to send data (output)*
- There are various plugins to meet your needs
  - *https://www.elastic.co/guide/en/logstash/current/input-plugins.html*
  - *https://www.elastic.co/guide/en/logstash/current/filter-plugins.html*
  - *https://www.elastic.co/guide/en/logstash/current/output-plugins.html*

Configuration file template

```
input {
    stdin { }
}

filter {
    mutate {
        uppercase => [ "message" ]
    }
}

output {
    stdout {
        codec => rubydebug
    }
}
```

Input sample - file plugin

```
input {
  file {
    path => [
      "D:\es\app\*",
      "D:\es\logs\*.txt"
    ]
    sincedb_path => "/dev/null"
    start_position => "beginning"
    exclude => ["*.csv]
    discover_interval => "10s"
    type => "applogs"
  }
}
```

Output sample - elasticsearch plugin

```
output {
  elasticsearch {
    index => "company"
    document_type => "employee"
    hosts => "localhost:9200"
  }
}
```

Filter sample - grok plugin

```
filter {
  grok {
    match => {
      "message" => "%{IP:client} %{WORD:method} %{URIPATHPARAM:request}
%{NUMBER:bytes} %{NUMBER:duration}"
    }
  }
}
```

- https://github.com/elastic/logstash/blob/v1.4.2/patterns/grok-patterns

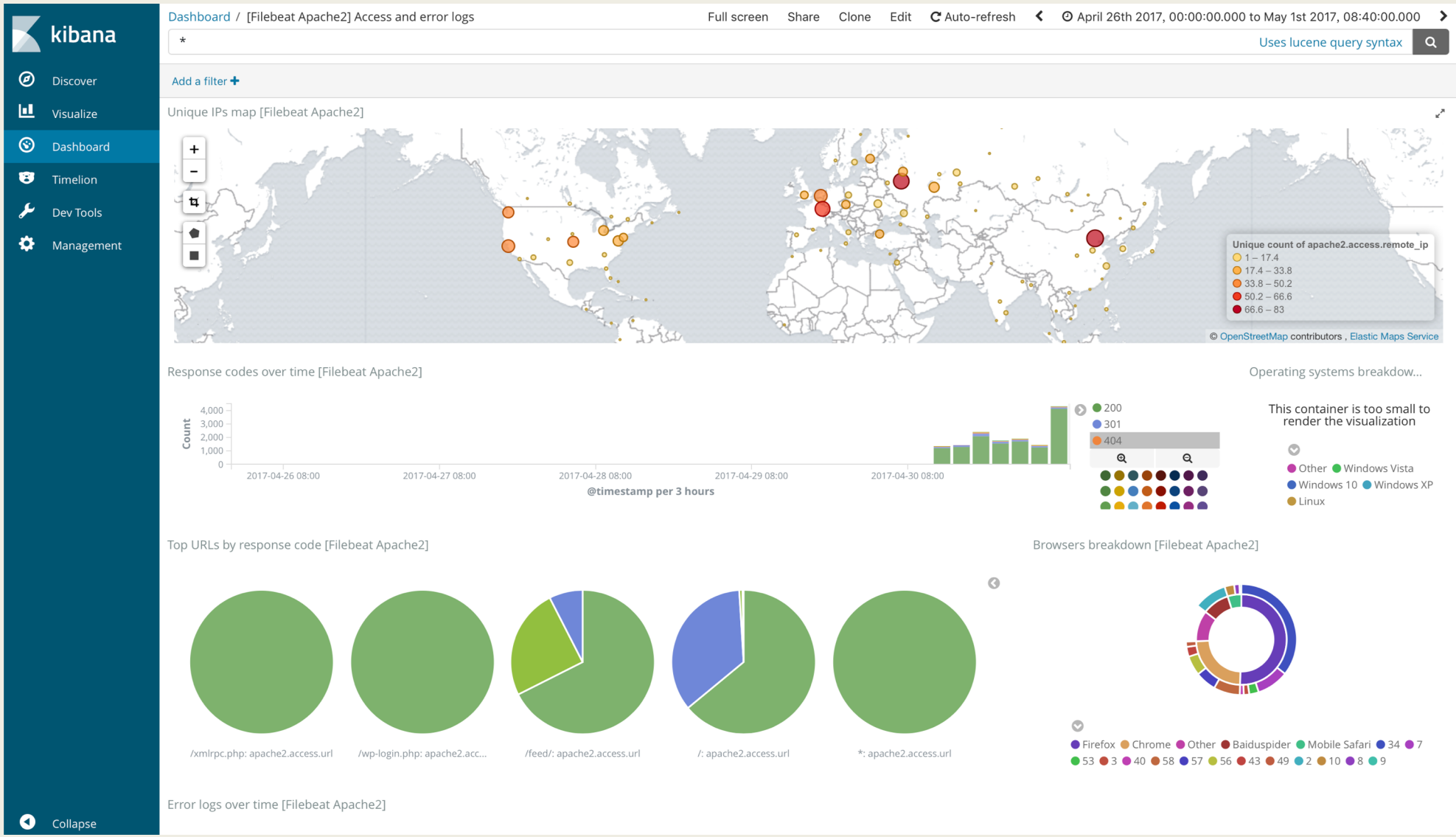
Exercise

- Import Apache access log to Elasticsearch
  - *Use Logstash*
  - *Use Filebeat*

KIBANA

How to use Kibana

- Create an index pattern
- Go Discover to perform a query search
- Go Visualize for data visualization
- Go Dev Tools to execute RESTful API


kibana
Discover
Visualize
Dashboard
Timelion
Dev Tools
Management
Collapse
Dashboard / [Filebeat Apache2] Access and error logs
Full screen
Share
Clone
Edit
Auto-refresh
April 26th 2017, 00:00:00.000 to May 1st 2017, 08:40:00.000
*
Uses lucene query syntax
Add a filter
Unique IPs map [Filebeat Apache2]
Unique count of apache2.access.remote_ip
1 – 17.4
17.4 – 33.8
33.8 – 50.2
50.2 – 66.6
66.6 – 83
© OpenStreetMap contributors , Elastic Maps Service
Response codes over time [Filebeat Apache2]
Count
4,000
3,000
2,000
1,000
0
2017-04-26 08:00
2017-04-27 08:00
2017-04-28 08:00
2017-04-29 08:00
2017-04-30 08:00
@timestamp per 3 hours
200
301
404
Operating systems breakdow...
This container is too small to render the visualization
Other
Windows Vista
Windows 10
Windows XP
Linux
Top URLs by response code [Filebeat Apache2]
/xmlrpc.php: apache2.access.url
/wp-login.php: apache2.acc...
/feed/: apache2.access.url
/: apache2.access.url
*: apache2.access.url
Browsers breakdown [Filebeat Apache2]
Firefox
Chrome
Other
Baiduspider
Mobile Safari
34
7
53
3
40
58
57
56
43
49
2
10
8
9
Error logs over time [Filebeat Apache2]

THANK YOU

Reference

- https://www.elastic.co
- Udemy Course "Elasticsearch 6 and Elastic Stack - In Depth and Hands On!" by Frank Kane
  https://www.udemy.com/elasticsearch-6-and-elastic-stack-in-depth-and-hands-on/
  (Sample codes were taken from this course)
- "Learning Elastic Stack 6.0" By Pranav Shukla, Sharath Kumar M N (Packt)