elasticsearch

María A. García Sopo

Schedule

2 Days

09:00h - 18:00h

Coffee time: 15’ mins

11:00h - 16:00h

Lunch time - 13:00h

1. Introducing to Elastic Stack

1. What is Elasticsearch, and why use it?
2. Elastic Stack components
3. Use cases of Elastic Stack

1. Introducing to Elastic Stack

1. What is Elasticsearch, and why use it?

Elasticsearch is a real-time, distributed search and analytics engine that is horizontally scalable and capable of solving a wide variety of use cases. At the heart of the Elastic Stack, it centrally stores your data so you can discover the expected and uncover the unexpected.

1. Introducing to Elastic Stack

1. What is Elasticsearch, and why use it?

    1.Schemaless, document-oriented


```
{
 "name": "John Smith",
 "address": "121 John Street, NY, 10010",
 "age": 40
}
```

```
{
 "name": "John Doe",
 "age": 38,
 "email": "john.doe@company.org"
}
```

1. Introducing to Elastic Stack

1. What is Elasticsearch, and why use it?

    1.Schemaless, document-oriented

    2.Searching

1. Introducing to Elastic Stack

1. What is Elasticsearch, and why use it?

    1.Schemaless, document-oriented

    2.Searching

    3.Analytics

1. Introducing to Elastic Stack

1. What is Elasticsearch, and why use it?

    1.Schemaless, document-oriented

    2.Searching

    3.Analytics

    4.Rich client library support and the REST API

    Java, C#, Python, JavaScript, PHP, Perl, Ruby

1. Introducing to Elastic Stack

1. What is Elasticsearch, and why use it?

    1.Schemaless, document-oriented

    2.Searching

    3.Analytics

    4.Rich client library support and the REST API

    5.Easy to operate and easy to scale

1. Introducing to Elastic Stack

1. What is Elasticsearch, and why use it?

    1.Schemaless, document-oriented

    2.Searching

    3.Analytics

    4.Rich client library support and the REST API

    5.Easy to operate and easy to scale

    6.Near real-time

1. Introducing to Elastic Stack

1. What is Elasticsearch, and why use it?

   7.Lightning-fast

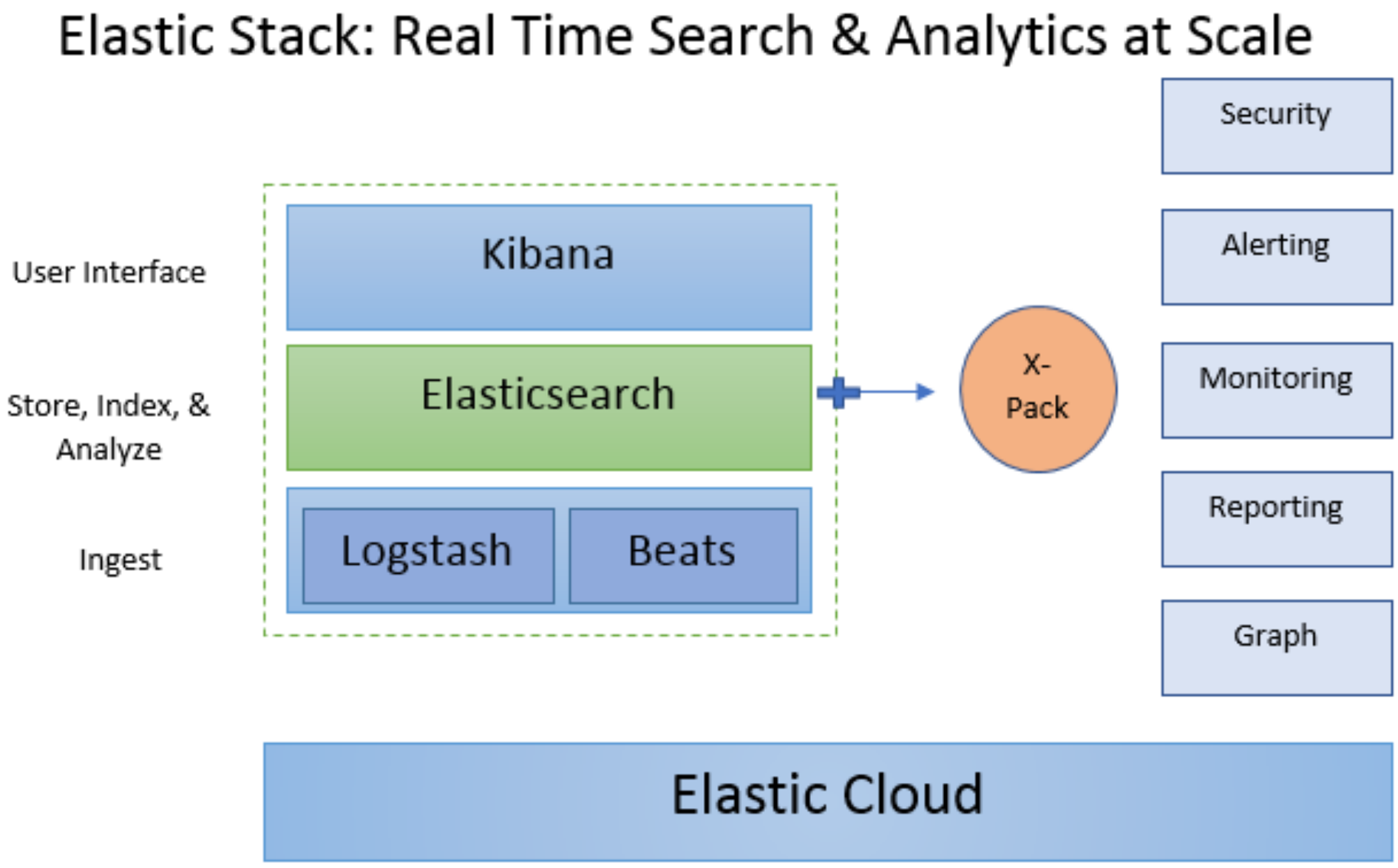
1. Introducing to Elastic Stack

1. What is Elasticsearch, and why use it?

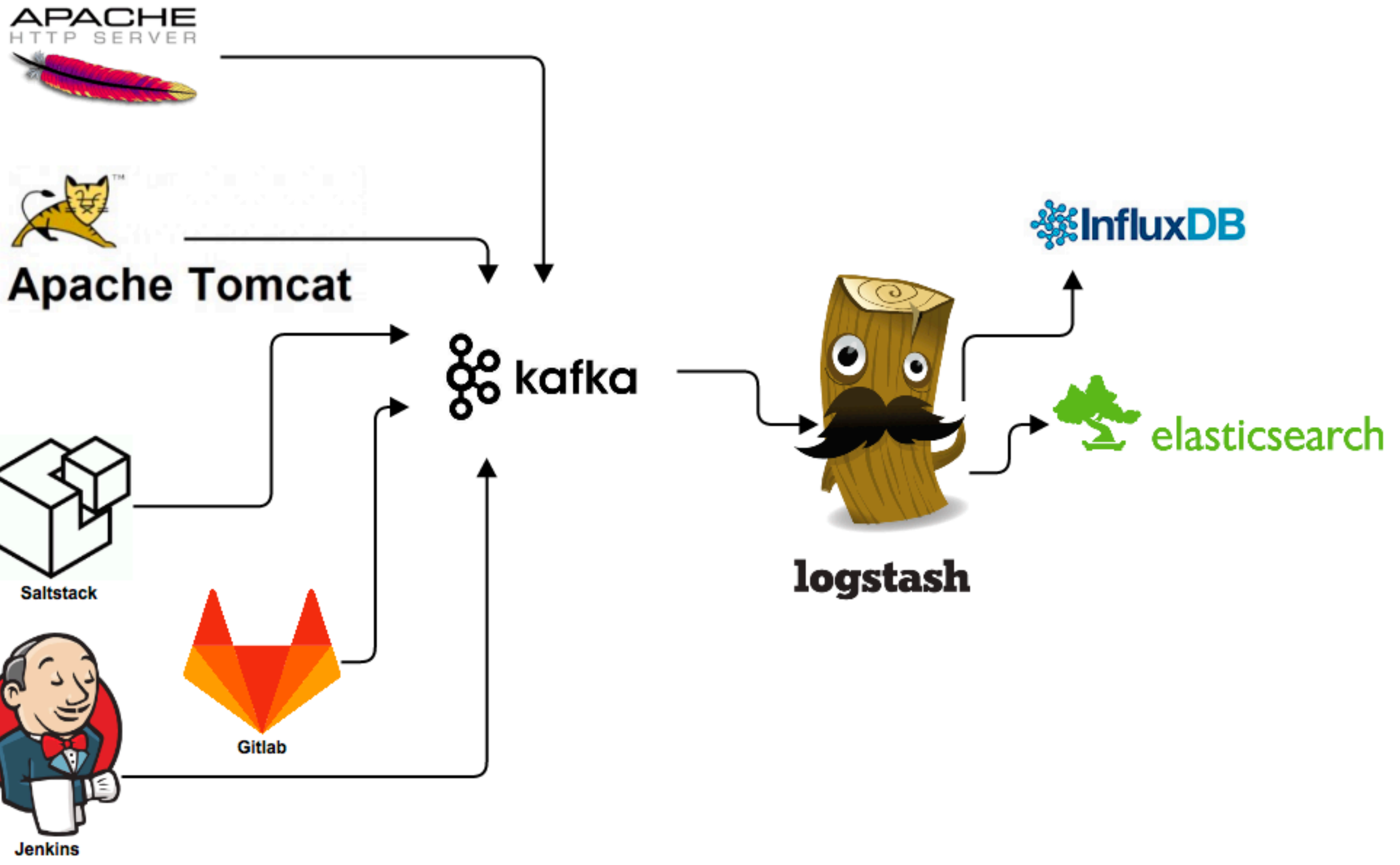
    7.Lightning-fast

    8.Fault-tolerant

1. Introducing to Elastic Stack



1. Introducing to Elastic Stack



1. Introducing to Elastic Stack

The Beats family

All kinds of shippers for all kinds of data.

Filebeat

Metricbeat

Packetbeat

Winlogbeat

Auditbeat

Heartbeat

Functionbeat

1. Introducing to Elastic Stack

The Beats family

All kinds of shippers for all kinds of data.

Filebeat

Metricbeat

Packetbeat

Winlogbeat

Auditbeat

Heartbeat

Functionbeat

Aggregate, “ tail -f ” & search

After you start Filebeat, open the Logs UI and watch your files being tailed right in Kibana. Use the search bar to filter by service, app, host, datacenter, or other criteria to track down curious behavior across your aggregated logs.

1. Introducing to Elastic Stack

The Beats family

All kinds of shippers for all kinds of data.

Filebeat

Metricbeat

Packetbeat

Winlogbeat

Auditbeat

Heartbeat

Functionbeat

System-level monitoring, simplified

Deploy Metricbeat on all your Linux, Windows, and Mac hosts, connect it to Elasticsearch and voila: you get system-level CPU usage, memory, file system, disk IO, and network IO statistics, as well as top-like statistics for every process running on your systems. **Explore the live demo**.

1. Introducing to Elastic Stack

The Beats family

All kinds of shippers for all kinds of data.

Filebeat

Metricbeat

Packetbeat

Winlogbeat

Auditbeat

Heartbeat

Functionbeat

Monitor services and applications

Network protocols like HTTP let you keep a pulse on application latency and errors, response times, SLA performance, user access patterns and trends, and more.

Packetbeat enables you to access this data to understand how traffic is flowing through your network. It's totally passive, has zero latency overhead, and doesn't interfere with your infrastructure. **Explore the live demo.**

1. Introducing to Elastic Stack

The Beats family

All kinds of shippers for all kinds of data.

Filebeat

Metricbeat

Packetbeat

Winlogbeat

Auditbeat

Heartbeat

Functionbeat

Read from any Windows event log channel

There's a lot to learn from your Windows event logs. Interested in security events like logon successes (4624) and failures (4625)? How about when a storage device is attached (4663) or a new service is installed (4798)? Winlogbeat can be configured to read from any event log channel, giving you access to the Windows data you need most.

1. Introducing to Elastic Stack

The Beats family

All kinds of shippers for all kinds of data.

Filebeat

Metricbeat

Packetbeat

Winlogbeat

Auditbeat

Heartbeat

Functionbeat

Keep an eye on your linux systems

Monitor user activity and processes, and analyze your event data in the Elastic Stack without touching auditd. Auditbeat communicates directly with the Linux audit framework, collects the same data as auditd, and sends the events to the Elastic Stack in real time. If you're feeling nostalgic, you can run auditd alongside Auditbeat (in newer kernels).

1. Introducing to Elastic Stack

The Beats family

All kinds of shippers for all kinds of data.

Filebeat

Metricbeat

Packetbeat

Winlogbeat

Auditbeat

Heartbeat

Functionbeat

Easy to use. Easy to configure.

Whether you're testing a service from the same host or across the open web, Heartbeat makes it easy to generate uptime and response time data. Create your own visualizations in Kibana to track availability, or jump into Elastic Uptime (powered by Heartbeat) to monitor your apps and services via our turnkey solution.

The Beats family

All kinds of shippers for all kinds of data.

Filebeat

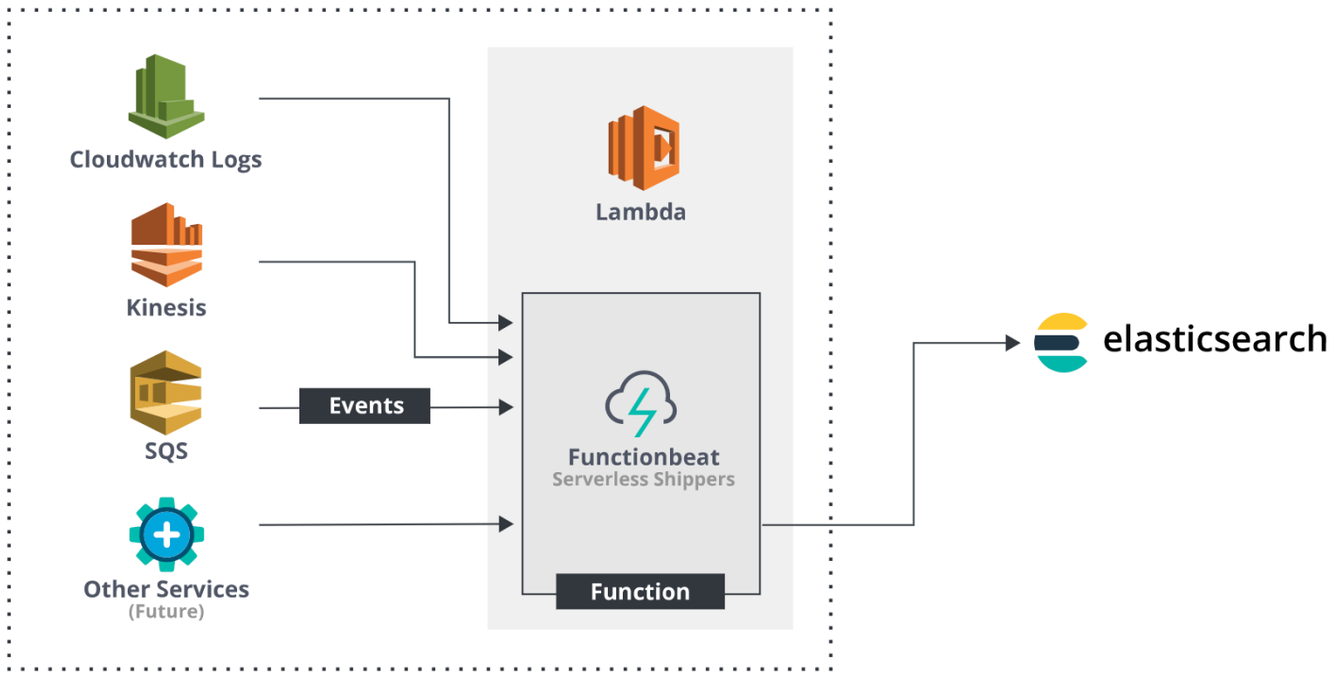
Metricbeat

Packetbeat

Winlogbeat

Auditbeat

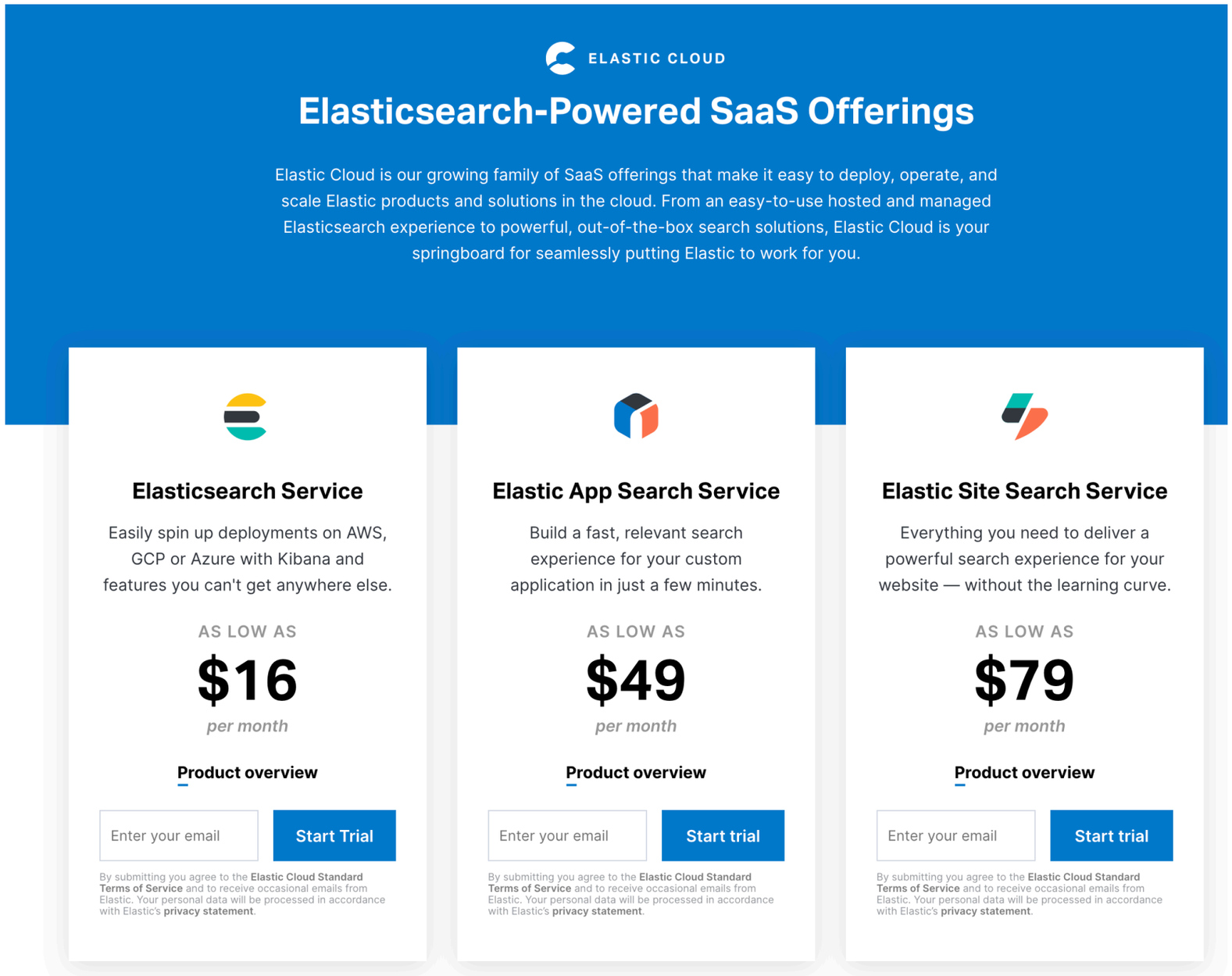
Functionbeat



One command to deploy

Serverless architectures let you deploy code, without needing to spin up and manage extra underlying software and hardware. Functionbeat brings that same simplicity to monitoring your cloud infrastructure. Deploy it as a function in your serverless framework, like AWS Lambda, to collect data from your cloud services — and bring them to the Elastic Stack for centralized analytics.

1. Introducing to Elastic Stack



https://www.elastic.co/cloud/

https://demo.elastic.co/

https://www.elastic.co/aws-elasticsearch-service

1. Introducing to Elastic Stack

3. Use cases of Elastic Stack

    1.Log and security analytics

    2.Product search

    3.Metrics analytics

    4.Web searches and website searches

1. Introducing to Elastic Stack

3. Installing ElasticSearch

    1.Start your Centos VM

    2.Install Elasticsearch
    https://www.elastic.co/downloads/elasticsearch

    3.Install Kibana
    https://www.elastic.co/downloads/kibana

    4.Install Postman - Optional

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

3.CRUD operations

4.Creating indexes and taking control of mapping REST API overview

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

- Indexes
- Types
- Documents
- Clusters
- Nodes
- Shards and replicas
- Mappings and types
- Inverted indexes

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

- Indexes
- Types
- Documents
- Clusters
- Nodes
- Shards and replicas
- Mappings and types
- Inverted indexes

```
PUT /catalog/_doc/1
  {
    "sku": "SP000001",
    "title": "Elasticsearch for Hadoop",
    "description": "Elasticsearch for Hadoop",
    "author": "Vishal Shukla",
    "ISBN": "1785288997",
    "price": 26.99
}
```

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

- Indexes
- Types
- Documents
- Clusters
- Nodes

```
PUT /catalog/_doc/1
  {
    "sku": "SP000001",
    "title": "Elasticsearch for Hadoop",
    "description": "Elasticsearch for Hadoop",
    "author": "Vishal Shukla",
    "ISBN": "1785288997",
    "price": 26.99
}
```

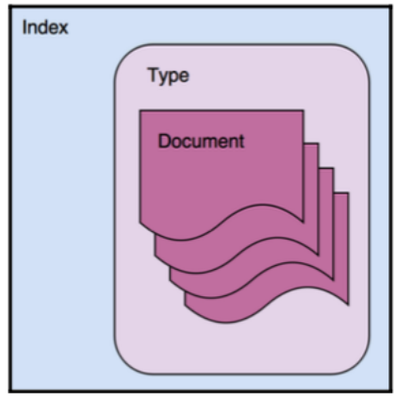


Figure 2.3: Organization of Index, Type, and Document

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

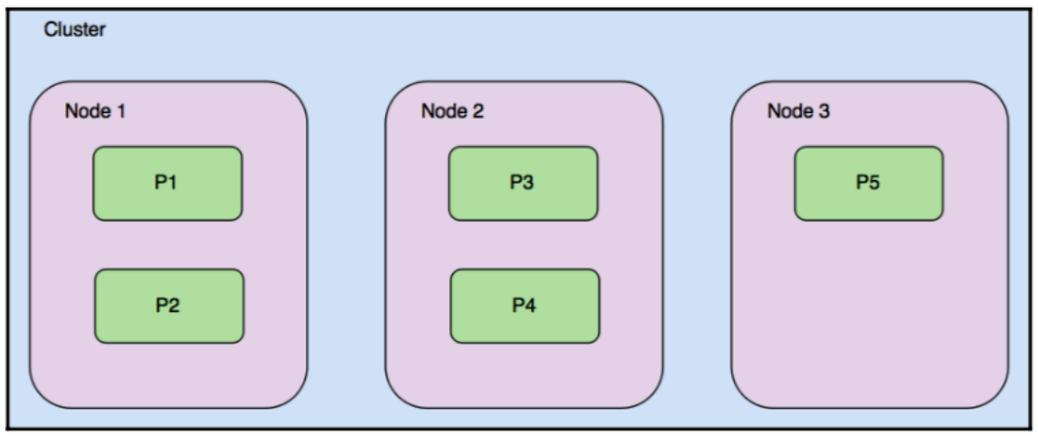
- Shards and replicas
- Mappings and datatypes
- Inverted indexes

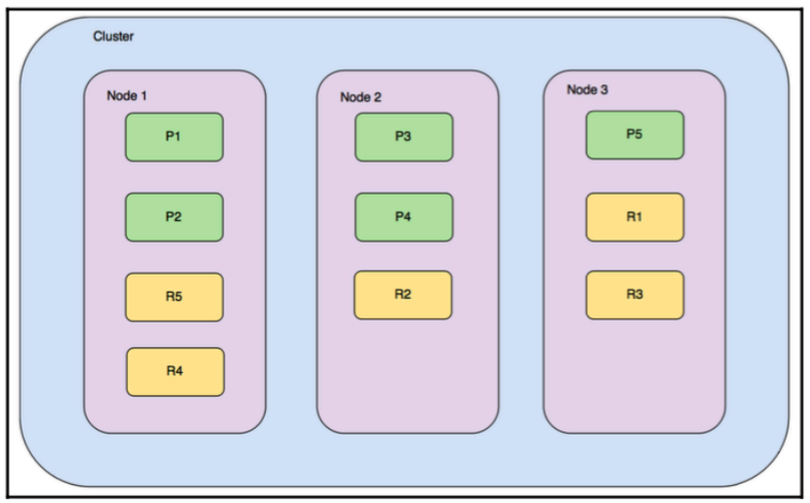
2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

- Shards and replicas
- Mappings and datatypes
- Inverted indexes





2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

- Shards and replicas
- Mappings and datatypes
- Inverted indexes

2. Getting Started with ElasticSearch

- **String datatypes**:
    - `text`: The `text` datatype is useful for supporting full-text search for fields that contain a description or lengthy text values. These fields are analyzed before indexing to support full-text search.
    - `keyword`: The `keyword` type enables analytics on string fields. Fields of this type support sorting, filtering, and aggregations.
- **Numeric datatypes**:
    - `byte`, `short`, `integer`, and `long`: Signed integers with 8-bit, 16-bit, 32-bit, and 64-bit precision, respectively
    - `float` and `double`: IEEE 754 floating-point numbers with single-precision 32-bit and double-precision 64-bit representations
    - `half_float`: IEEE 754 floating-point number with half-precision 16-bit representation
    - `scaled_float`: Floating-point number backed by a long and fixed scaling factor
- **Date datatype**:
    - `date`: Date with an optional timestamp component that's capable of storing precision timestamps down to the millisecond
- **Boolean datatype**:
    - `boolean`: The `boolean` datatype that is common in all programming languages
- **Binary datatype**:
    - `binary`: Allows you to store arbitrary binary values after performing Base64 encoding
- **Range datatypes**:
    - `integer_range`, `float_range`, `long_range`, `double_range`, and `date_range`: Defines ranges of integers, floats, longs, and more

2. Getting Started with ElasticSearch

- **String datatypes**:
    - `text`: The `text` datatype is useful for supporting full-text sea for fields that contain a description or lengthy text values. The fields are analyzed before indexing to support full-text search
    - `keyword`: The `keyword` type enables analytics on string fields Fields of this type support sorting, filtering, and aggregations
- **Numeric datatypes**:
    - `byte`, `short`, `integer`, and `long`: Signed integers with 8-bit, bit, 32-bit, and 64-bit precision, respectively
    - `float` and `double`: IEEE 754 floating-point numbers with sin precision 32-bit and double-precision 64-bit representations
    - `half_float`: IEEE 754 floating-point number with half-precision 16-bit representation
    - `scaled_float`: Floating-point number backed by a long and fixed scaling factor
- **Date datatype**:
    - `date`: Date with an optional timestamp component that's capable of storing precision timestamps down to the millisecond
- **Boolean datatype**:
    - `boolean`: The `boolean` datatype that is common in all programming languages
- **Binary datatype**:
    - `binary`: Allows you to store arbitrary binary values after performing Base64 encoding
- **Range datatypes**:
    - `integer_range`, `float_range`, `long_range`, `double_range`, and `date_range`: Defines ranges of integers, floats, longs, and more

Complex datatypes

The complex datatypes supported by Elasticsearch are as follows:

- **Array datatype**: Arrays of the same types of instances. For example, arrays of strings, integers, and more. Doesn't allow for the mixing of datatypes in arrays.
- **Object datatype**: Allows inner objects within JSON documents.
- **Nested datatype**: Useful for supporting arrays of inner objects, where each inner object needs to be independently queriable.

2. Getting Started with ElasticSearch

- **String datatypes**:
    - `text`: The `text` datatype is useful for supporting full-text sea for fields that contain a description or lengthy text values. The fields are analyzed before indexing to support full-text search
    - `keyword`: The `keyword` type enables analytics on string fields Fields of this type support sorting, filtering, and aggregations
- **Numeric datatypes**:
    - `byte`, `short`, `integer`, and `long`: Signed integers with 8-bit, bit, 32-bit, and 64-bit precision, respectively
    - `float` and `double`: IEEE 754 floating-point numbers with sin precision 32-bit and double-precision 64-bit representations
    - `half_float`: IEEE 754 floating-point number with half-precision 16-bit representation
    - `scaled_float`: Floating-point number backed by a long and fixed scaling factor
- **Date datatype**:
    - `date`: Date with an optional timestamp component that's cap of storing precision timestamps down to the millisecond
- **Boolean datatype**:
    - `boolean`: The `boolean` datatype that is common in all programming languages
- **Binary datatype**:
    - `binary`: Allows you to store arbitrary binary values after performing Base64 encoding
- **Range datatypes**:
    - `integer_range`, `float_range`, `long_range`, `double_rang` and `date_range`: Defines ranges of integers, floats, longs, an more

Complex datatypes

The complex datatypes supported by Elasticsearch are as follows:

- **Array datatype**: Arrays of the same types of instances. For example, arrays of strings, integers, and more. Doesn't allow for the mixing of datatypes in arrays.
- **Object datatype**: Allows inner objects within JSON documents.
- **Nested datatype**: Useful for supporting arrays of inner objects, where each inner object needs to be independently queriable.

Other datatypes

The other datatypes supported by Elasticsearch are as follows:

- **Geo-point datatype**: Allows the storing of geo-points as longitude and latitude. The geo-point datatype enables queries such as searching across all ATMs within a distance of 2 km from a point.
- **Geo-shape datatype**: Allows you to store geometric shapes such as polygons, maps, and more. Geo-shape enables queries such as searching for all items within a shape.
- **IP datatype**: Allows you to store IPv4 and IPv6 addresses.

2. Getting Started with ElasticSearch

```
GET /catalog/_mapping
```


```
{
  "catalog" : {
    "mappings" : {
      "properties" : {
        "ISBN" : {
          "type" : "text",
          "fields" : {
            "keyword" : {
              "type" : "keyword",
              "ignore_above" : 256
            }
          }
        },
        "author" : {
          "type" : "text",
          "fields" : {
            "keyword" : {
              "type" : "keyword",
              "ignore_above" : 256
            }
          }
        },
        "description" : {
          "type" : "text",
          "fields" : {
            "keyword" : {
              "type" : "keyword",
              "ignore_above" : 256
            }
          }
        },
        "os" : {
          "type" : "text",
          "fields" : {
            "keyword" : {
              "type" : "keyword",
              "ignore_above" : 256
            }
          }
        },
        "price" : {
          "type" : "float"
        },
        "resolution" : {
          "type" : "text",
          "fields" : {
            "keyword" : {
              "type" : "keyword",
              "ignore_above" : 256
            }
          }
        },
        "sku" : {
          "type" : "text",
          "fields" : {
            "keyword" : {
              "type" : "keyword",
              "ignore_above" : 256
            }
          }
        },
        "title" : {
          "type" : "text",
          "fields" : {
            "keyword" : {
              "type" : "keyword",
              "ignore_above" : 256
            }
          }
        }
      }
    }
  }
}
```

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

- Shards and replicas
- Mappings and datatypes
- Inverted indexes

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

- Shards and replicas
- Mappings and datatypes
- Inverted indexes

Document ID	Document
1	It is Sunday tomorrow.
2	Sunday is the last day of the week.
3	The choice is yours.

Elasticsearch builds a data structure from the three documents that have been indexed. The following data structure is called an **inverted index**:

Term	Frequency	Documents (postings)
choice	1	3
day	1	2
is	3	1, 2, 3

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

3.CRUD operations

- Index API
- Get API
- Update API
- Delete API

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

3.CRUD operations

- Index API
- Get API
- Update API
- Delete API

Indexing a document with ID

```
PUT /catalog/_doc/1
 {
    "sku": "SP000001",
    "title": "Elasticsearch for Hadoop",
    "description": "Elasticsearch for Hadoop",
    "author": "Vishal Shukla",
    "ISBN": "1785288997",
    "price": 26.99
}
```

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

3.CRUD operations

- Index API
- Get API
- Update API
- Delete API

Indexing a document with ID

```
PUT /catalog/_doc/1
 {
    "sku": "SP000001",
    "title": "Elasticsearch for Hadoop",
    "description": "Elasticsearch for Hadoop",
    "author": "Vishal Shukla",
    "ISBN": "1785288997",
    "price": 26.99
}
```

Indexing a document without ID

```
PUT /catalog/_doc
 {
    "_id" : "1ZFMpmoBa_wgE5i2FfWV",
    "sku": "SP000003",
    "title": “Mastering Elasticsearch",
    "description": “Mastering Elasticsearch",
    "author": “Bharvi Dixit",
    "price": 54.99
}
```

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

3.CRUD operations

- Index API
- Get API

```
GET /catalog/_doc/1ZFMpmoBa_wgE5i2FfWV
```

- Update API
- Delete API

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

3.CRUD operations

- Index API
- Get API
- Update API
- Delete API

```
POST /catalog/_update/1
{
  "doc": {
    "price": "28.99"
} }
```

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

3.CRUD operations

- Index API
- Get API
- Update API
- Delete API

```
POST /catalog/_update/1
{
  "doc": {
    "price": "28.99"
} }
```

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

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

3.CRUD operations

- Index API
- Get API
- Update API
- Delete API

```
DELETE /catalog/_doc/1ZFMpmoBa_wgE5i2FfWV
```

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

3.CRUD operations

4.Creating indexes and taking control of mapping REST API overview

- Creating an index
- Creating a mapping
- Updating a mapping

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

3.CRUD operations

4.Creating indexes and taking control of mapping REST API overview

- Creating an index

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

- Creating a mapping

- Updating a mapping

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

3.CRUD operations

4.Creating indexes and taking control of mapping REST API overview

- Creating an index

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

- Creating a mapping

- Updating a mapping

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

3.CRUD operations

4.Creating indexes and taking control of mapping REST API overview

- Creating an index

```
PUT /catalog1
  {
    "settings": {
      "index": {
        "number_of_shards": 5,
        "number_of_replicas": 2
      }
    },
    "mappings": {
      "properties": {
        "f1": {
          "type": "text"
        },
        "f2": {
          "type": "keyword"
} }

} }
```

- Creating a mapping

```
PUT /catalog/_mapping
  {
    "properties": {
      "name": {
        "type": "text"
      }
} }
```

- Updating a mapping

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

3.CRUD operations

4.Creating indexes and taking control of mapping REST API overview

- Creating an index
- Creating a mapping
- Updating a mapping

```
PUT /catalog/_mapping
  {
    "properties": {
     "name": {
       "type": "text"
     }
} }
```

```
POST /catalog/_doc
 {
  "name": "music",
  "description": "On-demand streaming music"
 }
```

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

3.CRUD operations

4.Creating indexes and taking control of mapping REST API overview

- Creating an index
- Creating a mapping
- Updating a mapping

```
PUT /catalog/_mapping
  {
    "properties": {
     "name": {
       "type": "text"
     }
} }
```

```
POST /catalog/_doc
 {
  "name": "music",
  "description": "On-demand streaming music"
 }
```

```
{
    "catalog" : {
     "mappings" : {
      "properties" : {
       "description" : {
         "type" : "text",
         "fields" : {
          "keyword" : {
           "type" : "keyword",
           "ignore_above" : 256
} }

},
"name" : {

          "type" : "text"
        }
} }

} }
```

2. Getting Started with ElasticSearch

1.Using the Kibana Console UI

2.Core concepts of Elasticsearch

3.CRUD operations

4.Creating indexes and taking control of mapping REST API overview

- Creating an index
- Creating a mapping
- Updating a mapping

```
PUT /catalog/_mapping
  {
    "properties": {
     "code": {
       "type": "keyword"
     }
} }
```

```
{
    "catalog" : {
     "mappings" : {
      "properties" : {
       "code" : {
         "type" : "keyword"
       },
       "description" : {
        "type" : "text",
        "fields" : {
          "keyword" : {
           "type" : "keyword",
           "ignore_above" : 256
} }

},
"name" : {

          "type" : "text"
        }
} }

} }
```

2. Getting Started with ElasticSearch

1.Search

2. Getting Started with ElasticSearch

1.Search

```
GET /_search
```

2. Getting Started with ElasticSearch

1.Search

```
GET /_search
```

Searching all documents in one index

```
GET /catalog/_search
```

```
GET /catalog/_doc/_search
```

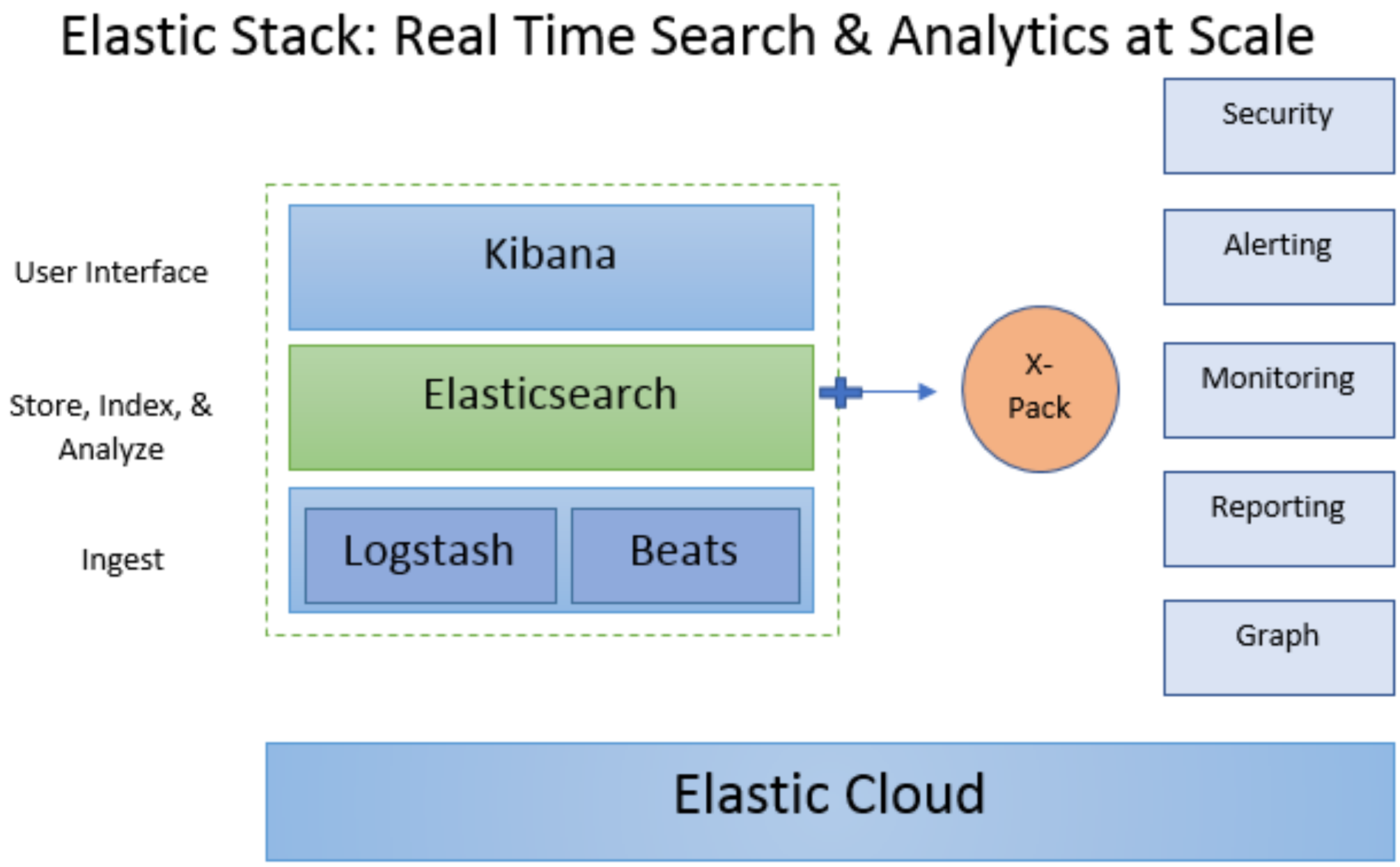
Searching all documents in multiple indexes

```
GET /catalog,my_index/_search
```

Searching all the documents of a particular type in all indexes

```
GET /_all/_doc/_search
```

Recap


Elastic Stack: Real Time Search & Analytics at Scale
User Interface
Store, Index, & Analyze
Ingest
Kibana
Elasticsearch
Logstash
Beats
X-Pack
Security
Alerting
Monitoring
Reporting
Graph
Elastic Cloud

Recap

3. Basic Concepts

- *Near Realtime (NTR)*
  Slight latency (normally one second) from the time you index a document until the time it becomes searchable.

- *Cluster*
  A cluster is a collection of one or more nodes (servers). Cluster is identified by a unique name. Do not reuse cluster names!

- *Node*
  Stores your data, and participates in the cluster's indexing and search capabilities. No limits!.

- *Index*
  Collection of documents that have somewhat similar characteristics. All indexes in lowercase. No limits!

- *Type: **DEPRECATED in 6.0.0***
  Allow you to store different types of documents in the same index.

- *Document*
  Basic unit of information that can be indexed. Json format.

- *Shards*
  Slices of one Index. Each shard is in itself a fully-functional and independent "index" that can be hosted on any node in the cluster. You are asked when create an index. Advantages:
  1 - Horizontal Scale
  2 - Parallelism in operations

- *Replicas*
  Provide HA in case shard/node fails.

Recap

3. Basic Concepts

Relational DB	*Databases*	*Tables*	*Rows*	*Columns*
Elasticsearch	*Indices*	*Types*	*Documents*	*Fields*

2. Installation & ELS distributed

1. Installation Cluster

2.Configuration

3. Run ELS

4. REST APIs

5. ELS Distributed

2. Installation & ELS distributed

1. Elastic Search Cluster
   - Check compatibility

	CentOS/RHEL 6.x/7.x	Oracle Enterprise Linux 6/7 with RHEL Kernel only	Ubuntu 14.04	Ubuntu 16.04	SLES 11 SP4**	SLES 12	openSUSE Leap 42	Windows Server 2012/R2	Windows Server 2016	Debian 7	Debian 8	Debian 9	Solaris/ SmartOS	Amazon Linux*
Elasticsearch 2.4.x	✔	✔	✔	✖	✔	✔	✔	✔	✖	✔	✔	✖	✖	✔
Elasticsearch 5.0.x	✔	✔	✔	✔	✔	✔	✔	✔	✖	✔	✔	✖	✖	✔
Elasticsearch 5.1.x	✔	✔	✔	✔	✔	✔	✔	✔	✖	✔	✔	✖	✖	✔
Elasticsearch 5.2.x	✔	✔	✔	✔	✔	✔	✔	✔	✖	✔	✔	✖	✖	✔
Elasticsearch 5.3.x	✔	✔	✔	✔	✔	✔	✔	✔	✖	✔	✔	✖	✖	✔
Elasticsearch 5.4.x	✔	✔	✔	✔	✔	✔	✔	✔	✖	✔	✔	✖	✖	✔
Elasticsearch 5.5.x	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✖	✔
Elasticsearch 5.6.x	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✖	✔
Elasticsearch 6.0.x	✔	✔	✔	✔	✖	✔	✔	✔	✔	✖	✔	✔	✖	✔
Elasticsearch 6.1.x	✔	✔	✔	✔	✖	✔	✔	✔	✔	✖	✔	✔	✖	✔
Elasticsearch 6.2.x	✔	✔	✔	✔	✖	✔	✔	✔	✔	✖	✔	✔	✖	✔

2. Installation & ELS distributed

1. Elastic Search Cluster

1- Setting up **Centos 7**:

- VirtualBox
- Elasticsearch.zip

2 - Install requisites

- java 1.8.0_131 or later: java -version

3 - Install ELS

1 - Import PGP key

```
rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch
```

2 - Import Repository:

```
[elasticsearch-7.x]
name=Elasticsearch repository for 7.x packages
baseurl=https://artifacts.elastic.co/packages/7.x/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=1
autorefresh=1
type=rpm-md
```

3 - Install ElasticSearch

```
sudo yum install elasticsearch
```

2. Installation & ELS distributed

2. Configuration

1- Configuration files

Type	Description	Location Debian/Ubuntu	Location RHEL/CentOS
home	Home of elasticsearch installation.	`/usr/share/elasticsearch`	`/usr/share/elasticsearch`
bin	Binary scripts including `elasticsearch` to start a node.	`/usr/share/elasticsearch/bin`	`/usr/share/elasticsearch/bin`
conf	Configuration files `elasticsearch.yml` and `logging.yml`.	`/etc/elasticsearch`	`/etc/elasticsearch`
conf	Environment variables including heap size, file descriptors.	`/etc/default/elasticsearch`	`/etc/sysconfig/elasticsearch`
data	The location of the data files of each index / shard allocated on the node.	`/var/lib/elasticsearch/data`	`/var/lib/elasticsearch`
logs	Log files location	`/var/log/elasticsearch`	`/var/log/elasticsearch`
plugins	Plugin files location. Each plugin will be contained in a subdirectory.	`/usr/share/elasticsearch/plugins`	`/usr/share/elasticsearch/plugins`

2. Installation & ELS distributed

2. Configuration

1- Configuration files

- /etc/elasticsearch/elasticsearch.yml
  Configuring Elasticsearch

- /etc/elasticsearch/jvm.options
  JVM settings

- /etc/elasticsearch/log4j2.properties
  L*ogging*

2. Installation & ELS distributed

2. Configuration

1- Configuration files: /etc/elasticsearch/elasticsearch.yml

```
# por defecto elasticsearch
cluster.name: my-application
# por defecto nombre aleatorio de marvel
node.name: node-1
# Conviene habilitarla, bloquea la dirección del proceso a la RAM, evita swap
bootstrap.memory_lock: true
# Permite ejecutar scripts dentro de las búsquedas, actualizaciones
script.inline: true
# Permite indexar ( guardar ) los scripts que nos creemos
script.stored: true
# Permite el tráfico proveniente en exclusiva de localhost
network.host: 127.0.0.1
```

2. Installation & ELS distributed

2. Configuration

1- Configuration files:

- minimun limit open files: 3200
  /etc/security/limits.conf

  (Hard limits are enforced "here and now", i.e. process can never access a resource if it would cause violation of a hard limit. A process can exceed soft limit for a period of time)

- Limit Memory: -Xms and -Xmx (half RAM and minimum 2gb)
  /etc/elasticsearch/jvm.options

2. Installation & ELS distributed

3. Run ELS

systemctl start elasticsearch

systemctl enable elasticsearch

systemctl status elasticsearch

Check logs!

2. Installation & ELS distributed

4. REST APIs

1 - Get Postman collection:

2 - Configure the access to your ELS

3 - Import Collection

4 - Run queries:

** Check health check status*

** Detalles del Cluster*

** Get master*

...

netstat -putan | grep -i listen

ifconfig | grep inet

3. Indexes

Exercise

1. *Create a new index 'Country' and their properties*
2. *type 'europe'*
3. *Populate the index by (Batch):*

Name	España	Portugal	Alemania
Capital	Madrid	Lisboa	Berlin
Language	Español	Portugues	Aleman
Currency	Euro	Euro	Euro
Habitants	46,354,321	10,379,573	152.000.000
	Italia	Francia	
	Roma	Paris	
	Italiano	Francia	
	Euro	Euro	
	60,589,445	67,201,000	

4. Update currency in all countries with the dollar value except Italy.
5. Create a new document (Colombia)
6. Show all documents
7. Add a new property to each document with the same location to include the latitude and longitude.
8. Delete the document Italy

3. Searching - What is Relevant

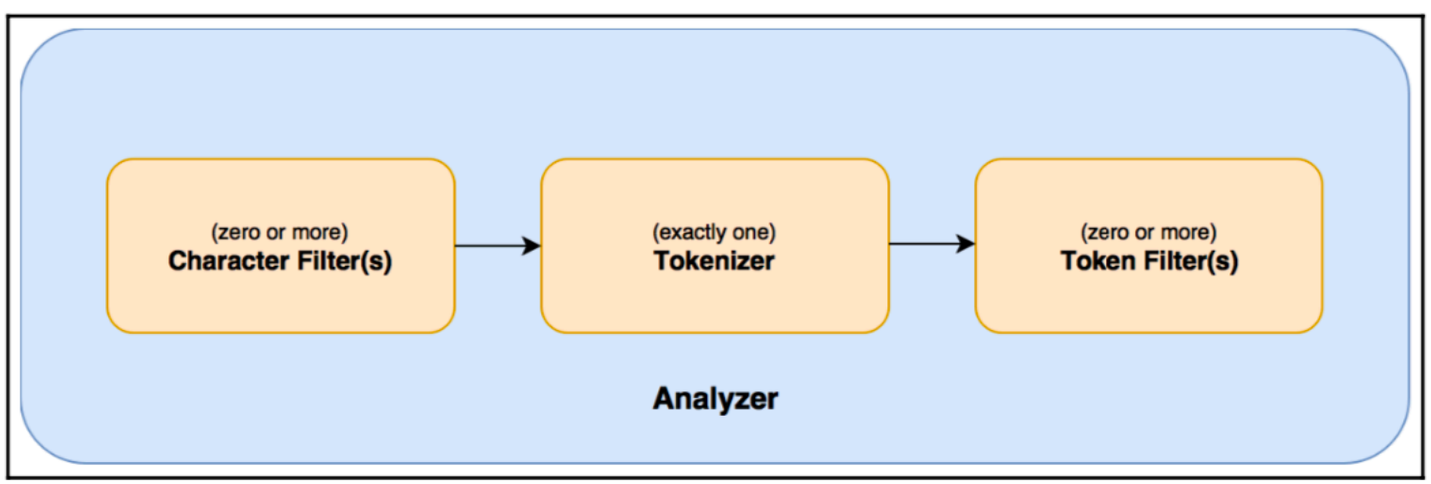
- The basics of text analysis
- Searching from structured data
- Searching from full-text
- Writing compound queries
- Modeling relationships

3. Searching - What is Relevant

- The basics of text analysis
  - Understanding Elasticsearch analyzers
  - Using built-in analyzers
  - Implementing autocomplete with a custom analyzer

3. Searching - What is Relevant

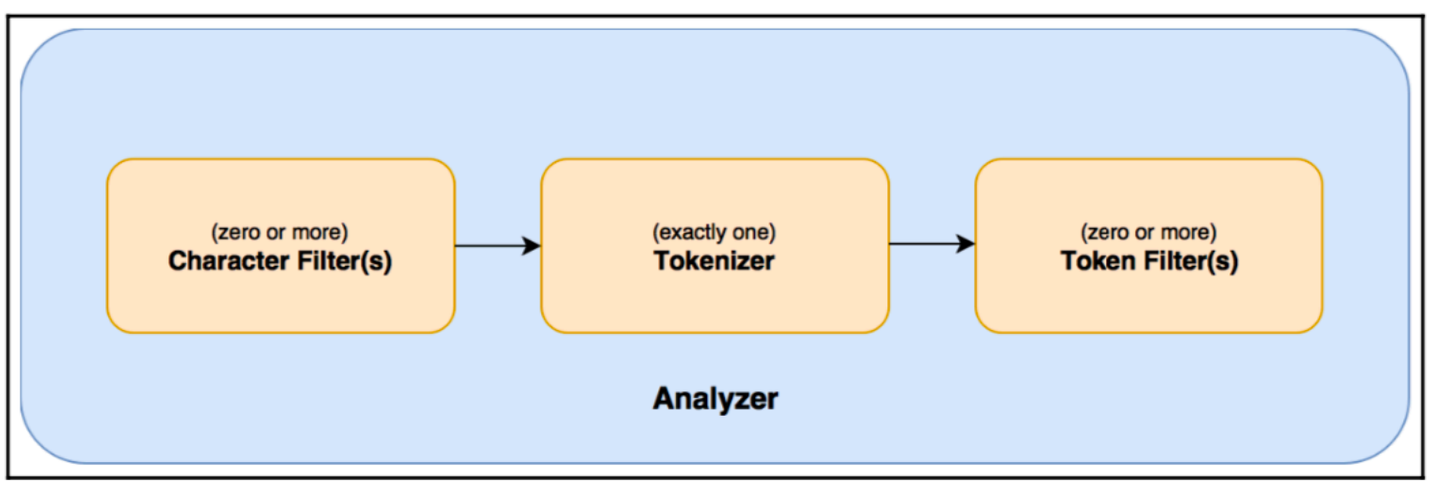
- The basics of text analysis
  - Understanding Elasticsearch analyzers
    - At the time of indexing
    - At the time of searching



3. Searching - What is Relevant

- The basics of text analysis
  - Understanding Elasticsearch analyzers

- At the time of indexing
- At the time of searching

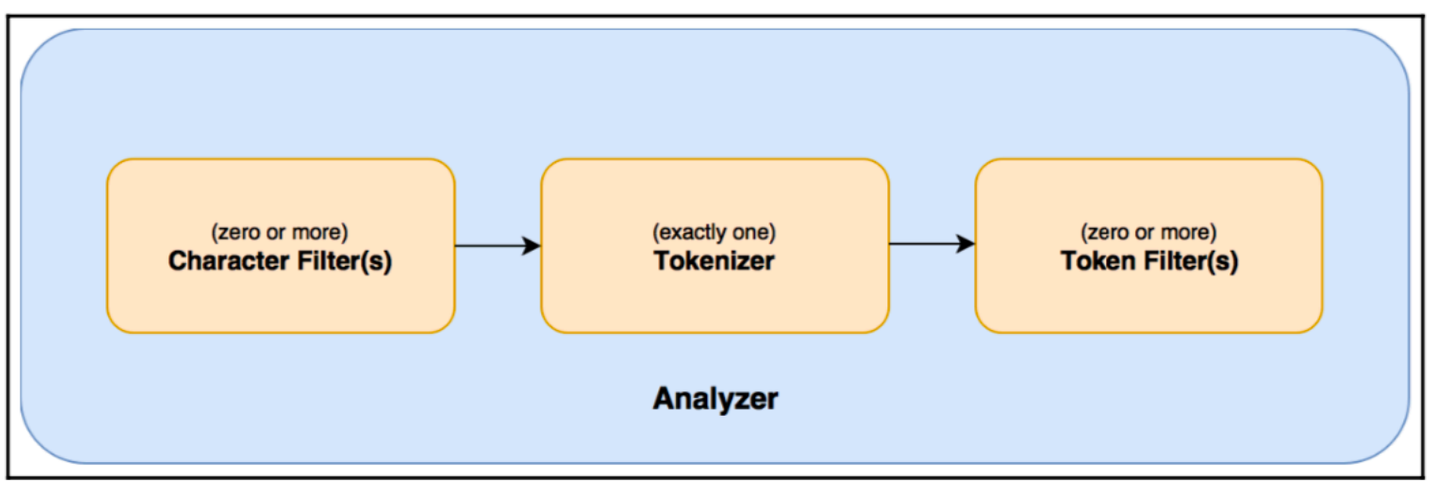



```
"char_filter": {
        "my_char_filter": {
          "type": "mapping",
          "mappings": [
            ":) => _smile_",
            ":( => _sad_",
            ":D => _laugh_"
] }

}
```

3. Searching - What is Relevant

- The basics of text analysis
  - Understanding Elasticsearch analyzers

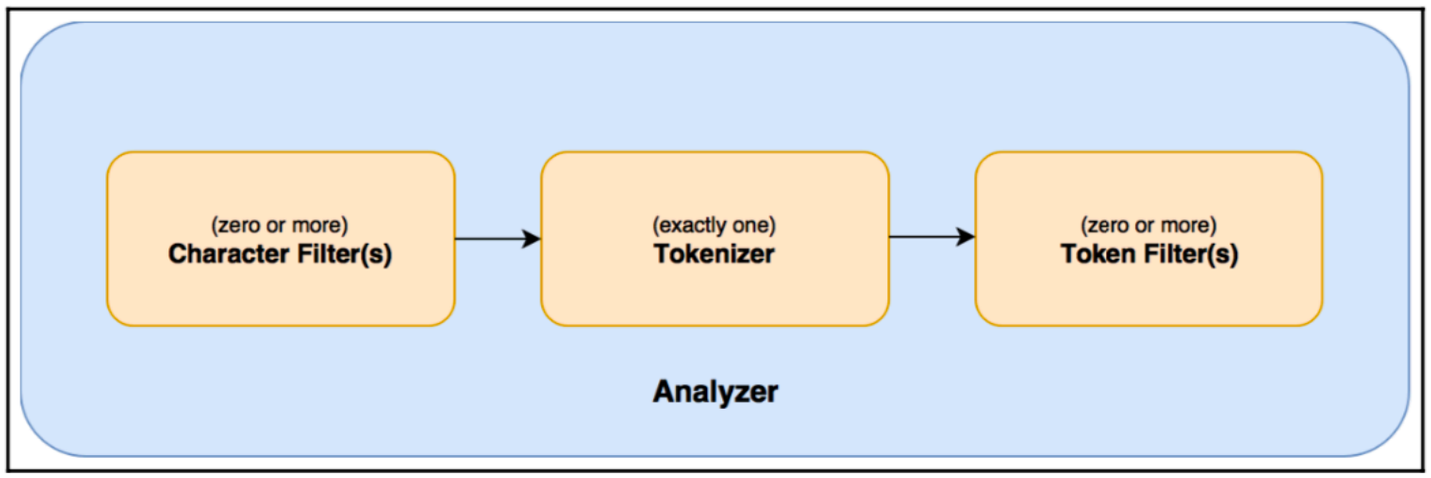
- At the time of indexing
- At the time of searching



https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-tokenizers.html

3. Searching - What is Relevant

- The basics of text analysis
- Understanding Elasticsearch analyzers
  - At the time of indexing
  - At the time of searching



https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-tokenizers.html

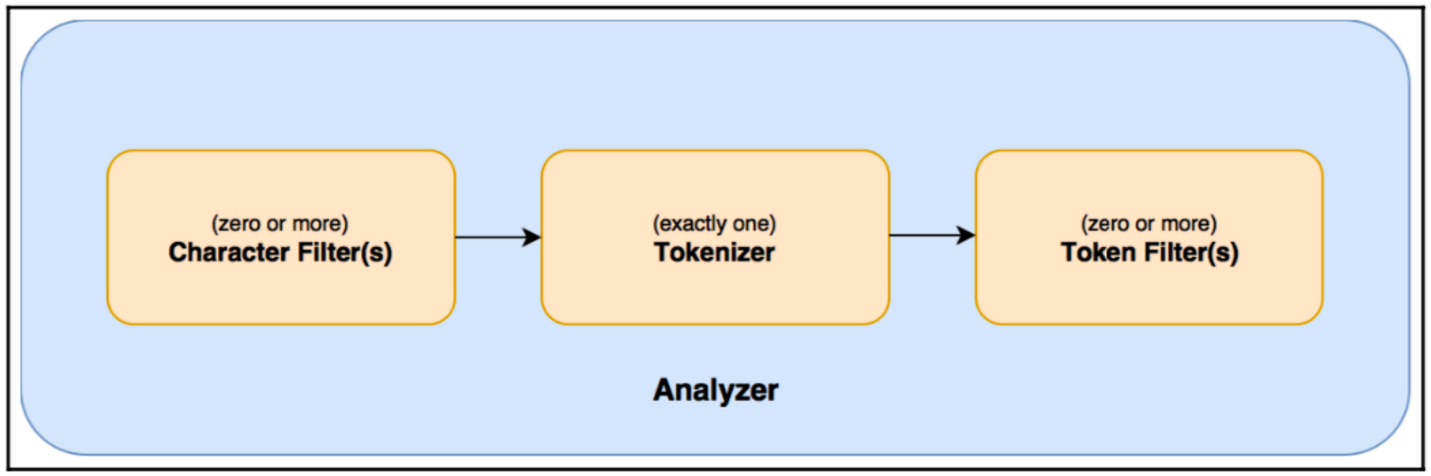
```
POST _analyze
 {
  "tokenizer": "standard",
  "text": "Tokenizer breaks characters into tokens!"
 }
```

```
{
"tokens": [
    {
     "token": "Tokenizer",
     "start_offset": 0,
     "end_offset": 9,
     "type": "<ALPHANUM>",
     "position": 0
}, {
     "token": "breaks",
     "start_offset": 10,
     "end_offset": 16,
     "type": "<ALPHANUM>",
     "position": 1
}, {
     "token": "characters",
     "start_offset": 17,
     "end_offset": 27,
     "type": "<ALPHANUM>",
     "position": 2
}, {
     "token": "into",
     "start_offset": 28,
     "end_offset": 32,
      "type": "<ALPHANUM>",
        "position": 3
      },
      {
        "token": "tokens",
        "start_offset": 33,
        "end_offset": 39,
        "type": "<ALPHANUM>",
        "position": 4
  } ]

  }
```

3. Searching - What is Relevant

- The basics of text analysis
  - Understanding Elasticsearch analyzers

- At the time of indexing
- At the time of searching



- **Lowercase token filter**: Replaces all tokens in the input with their lowercase versions.
- **Stop token filter**: Removes stopwords, that is, words that do not add more meaning to the context. For example, in English sentences, words like *is*, *a*, *an*, and *the*, do not add extra meaning to a sentence. For many text search problems, it makes sense to remove such words, as they don't add any extra meaning or context to the content.

3. Searching - What is Relevant

- The basics of text analysis
  - Understanding Elasticsearch analyzers
  - Using built-in analyzers

    - **Standard analyzer**: This is the default analyzer in Elasticsearch. If not overridden by any other field-level, type-level, or index-level analyzer, all fields are analyzed using this analyzer.
    - **Language analyzers**: Different languages have different grammatical rules. There are differences between some languages as to how a stream of characters is tokenized into words or tokens. Additionally, each language has its own set of stopwords, which can be configured while configuring language analyzers.
    - **Whitespace analyzer**: The whitespace analyzer breaks down input into tokens wherever it finds a whitespace token such as a space, a tab, a new line, or a carriage return.

3. Searching - What is Relevant

- The basics of text analysis
  - Understanding Elasticsearch analyzers
  - Using built-in analyzers
  - Implementing autocomplete with a custom analyzer

3. Searching - What is Relevant

- The basics of text analysis
  - Understanding Elasticsearch analyzers
  - Using built-in analyzers
  - Implementing autocomplete with a custom analyzer

```
GET /_analyze
{
  "text": "Learning Elastic Stack 7",
  "analyzer": "standard"
}
```

```
"mappings": {
    "properties": {
      "product": {
        "type": "text",
        "analyzer": "custom_analyzer",
        "search_analyzer": "standard"
}}
```

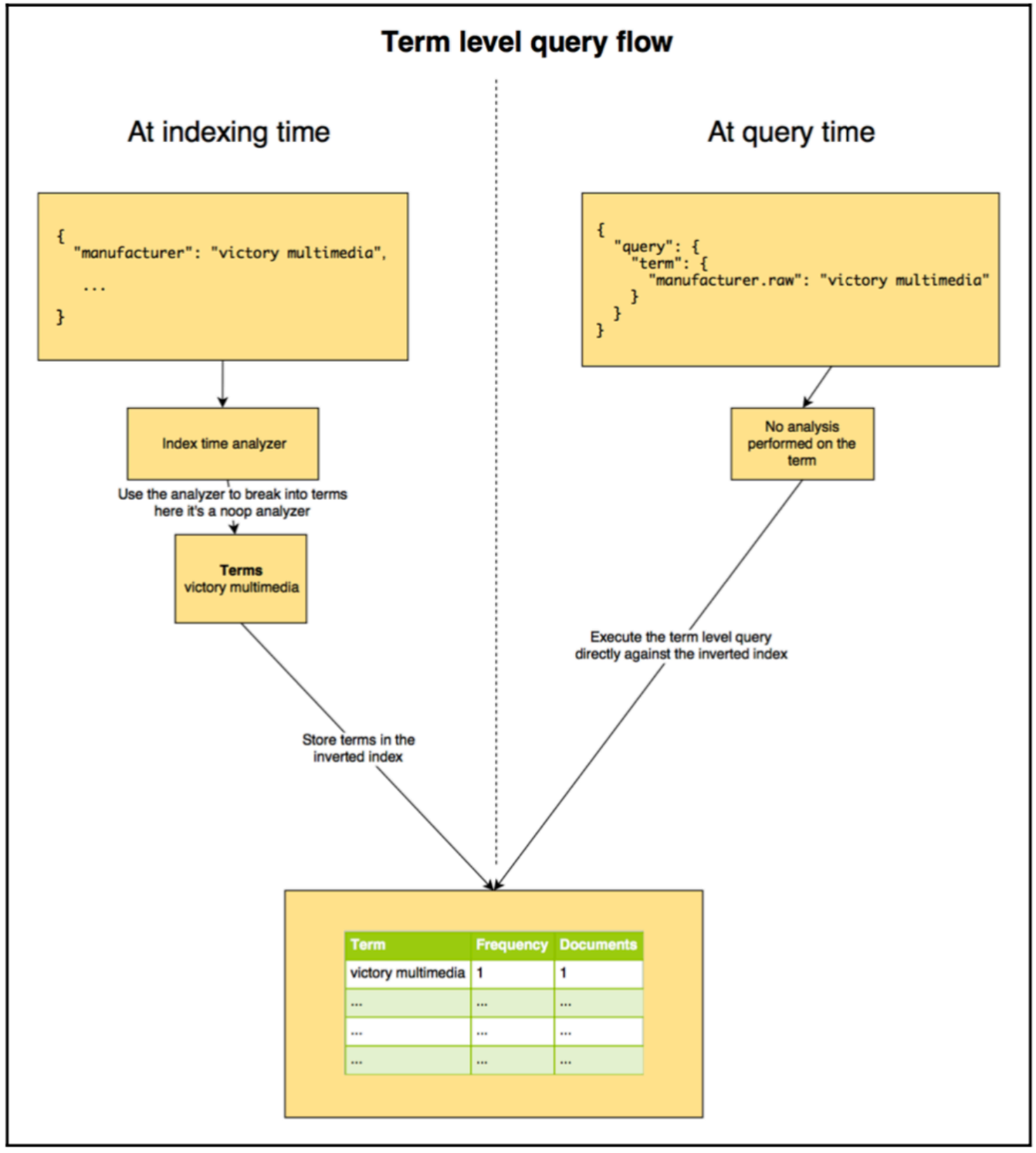
```
PUT /custom_analyzer_index
 {
  "settings": {
   "index": {
    "analysis": {
     "analyzer": {
      "custom_analyzer": {
       "type": “custom",
      "tokenizer": "standard",
       "filter": [
        "lowercase",
        "custom_edge_ngram"
       ]
} },
      "filter": {
       "custom_edge_ngram": {
        "type": "edge_ngram",
        "min_gram": 2,
        "max_gram": 10
} }

} }
```

3. Searching - What is Relevant

- The basics of text analysis
- Searching from structured data

3. Searchin

- The basics of text analysis
- Searching from structurec



3. Searching - What is Relevant

- The basics of text analysis
- Searching from structured data
    - Range query
    - Exists query
    - Term query

3. Searching - What is Relevant

- The basics of text analysis
- Searching from structured data
  - Range query
    - On numeric types
    - With score boosting
    - On dates

```
GET /amazon_products/_search
  {
    "query": {
     "range": {
       "price": {
         "gte": 10,
         "lte": 20
} }

} }
```

3. Searching - What is Relevant

- The basics of text analysis
- Searching from structured data
  - Range query
    - On numeric types
    - With score boosting
    - On dates

```
GET /amazon_products/_search
  {
    "from": 0,
    "size": 10,
    "query": {
      "range": {
        "price": {
          "gte": 10,
          "lte": 20,
          "boost": 2.2
        } }
    } }
```

3. Searching - What is Relevant

- The basics of text analysis
- Searching from structured data
  - Range query
    - On numeric types
    - With score boosting
    - On dates

```
GET /orders/_search
 {
    "query": {
       "range" : {
          "orderDate" : {
             "gte": "01/09/2017",
             "lte": "30/09/2017",
             "format": "dd/MM/yyyy"
} }
```

3. Searching - What is Relevant

- The basics of text analysis
- Searching from structured data
  - Range query
  - Exists query

```
GET /amazon_products/_search
{
  "query": {
    "exists": {
      "field": "description"
    }
} }
```

3. Searching - What is Relevant

- The basics of text analysis
- Searching from structured data
    - Range query
    - Exists query
    - Term query

3. Searching - What is Relevant

- The basics of text analysis
- Searching from structured data
    - Range query
    - Exists query
    - Term query

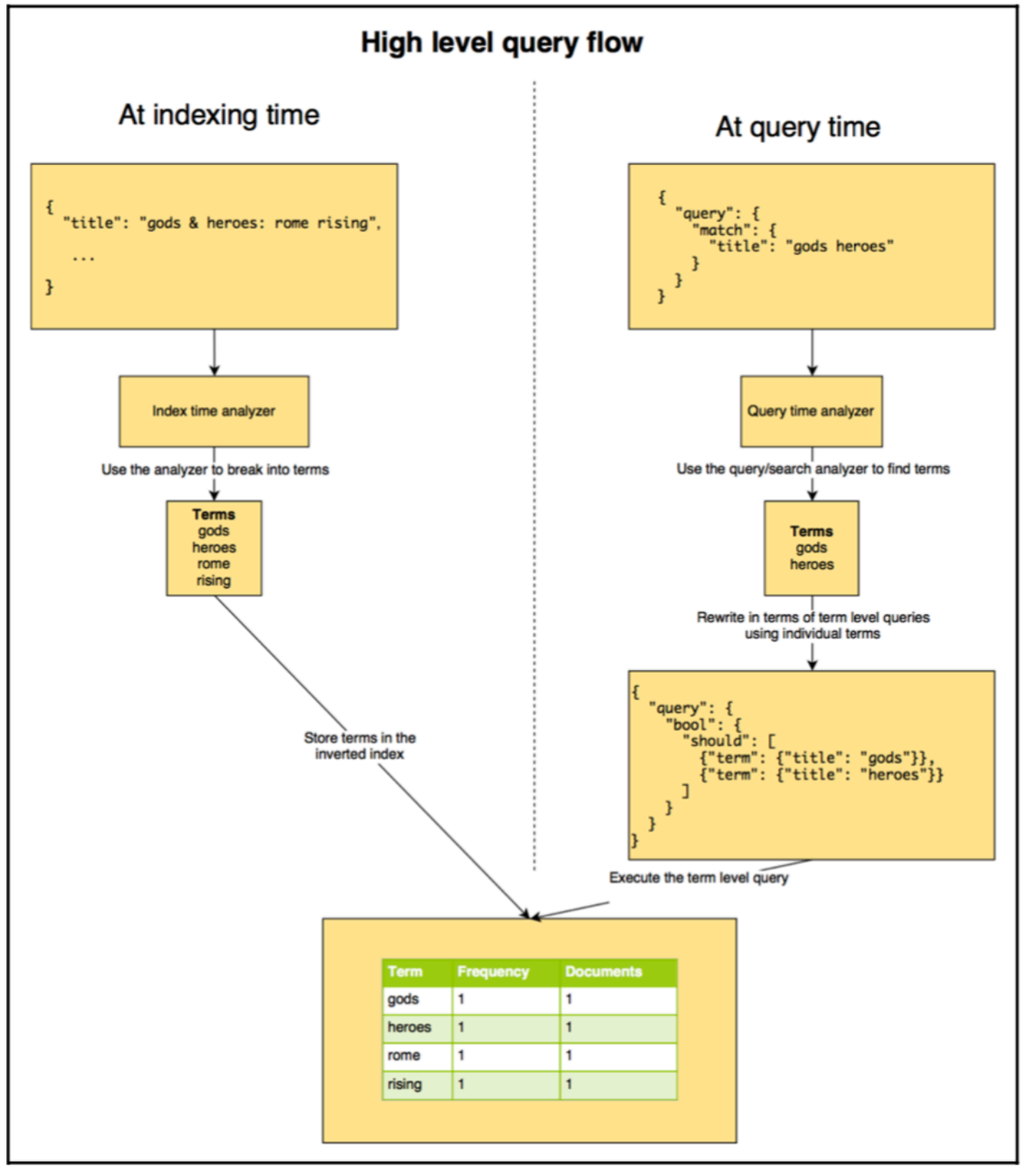
```
GET /amazon_products/_search
{
  "query": {
    "term": {
      "manufacturer.raw": "victory multimedia"
    }
} }
```

3. Searching - What is Relevant

- The basics of text analysis
- Searching from structured data
- Searching from full-text

3. Searching - What is Relevant

- The basics of text analysis
- Searching from structured da
- Searching from full-text



3. Searching - What is Relevant

- The basics of text analysis
- Searching from structured data
- Searching from full-text
    - Match query
    - Match phrase query
    - Multi match query

3. Searching - What is Relevant

- The basics of text analysis
- Searching from structured data
- Searching from full-text
  - Match query

```
GET /amazon_products/_search
 {
   "query": {
    "match": {
      "manufacturer.raw": "victory multimedia"
    }
} }
```

```
GET /amazon_products/_search
  {
    "query": {
     "match": {
       "manufacturer": "victory multimedia"
     }
} }
```

3. Searching - What is Relevant

- The basics of text analysis
- Searching from structured data
- Searching from full-text
    - Match query

- Search for the terms `victory` and `multimedia` across all documents within the `manufacturer` field.
- Find the best matching documents sorted by score in descending order.
- If both terms appear in the same order, right next to each other in a document, the document should get a higher score than other documents that have both terms but not in the same order, or not next to each other.
- Include documents that have either `victory` or `multimedia` in the results, but give them a lower score.

3. Searching - What is Relevant

- The basics of text analysis
- Searching from structured data
- Searching from full-text
  - Match query

```
GET /amazon_products/_search
  {
    "query": {
      "match": {
        "manufacturer": {
        "query": "victory multimedia",
        "operator": "and"
} }

} }
```

```
GET /amazon_products/_search
  {
    "query": {
     "match": {
       "manufacturer": {
       "query": "victory multimedia",
         "minimum_should_match": 2
       }
} }
```

3. Searching - What is Relevant

- The basics of text analysis
- Searching from structured data
- Searching from full-text
  - Match query

```
GET /amazon_products/_search
{
  "query": {
    "match": {
      "manufacturer": {
        "query": "victor multimedia",
        "fuzziness": 1
      }
    }
  }
}
```

3. Searching - What is Relevant

- The basics of text analysis
- Searching from structured data
- Searching from full-text
    - Match query
    - Match phrase query

```
GET /amazon_products/_search
{
  "query": {
    "match_phrase": {
      "description": {
        "query": "real video saltware aquarium'
      }
    }
```

```
{
  ...,
  "hits": {
    "total": 1,
    "max_score": 22.338196,
    "hits": [
      {
        "_index": "amazon_products",
        "_type": "products",
        "_id": "AV5rBfasNI_2eZGciIbg",
        "_score": 22.338196,
        "_source": {
          "price": "19.95",
          "description": "real video saltware aquarium on your
desktop!product information see real fish swimming on your desktop in full-
motion video! you'll find exotic saltwater fish such as sharks angelfish
and more! enjoy the beauty and serenity of a real aquarium at yourdeskt",
          "id": "b00004t2un",
          "title": "sales skills 2.0 ages 10+",
          "manufacturer": "victory multimedia",
          "tags": []
        }
      }
    ]
  }
}
```

3. Searching - What is Relevant

- The basics of text analysis
- Searching from structured data
- Searching from full-text
    - Match query
    - Match phrase query

```
GET /amazon_products/_search
{
  "query": {
    "multi_match": {
      "query": "monitor aquarium",
      "fields": ["title^3", "description"]
    }
  }
}
```

3. Searching - What is Relevant

- The basics of text analysis
- Searching from structured data
- Searching from full-text
    - Match query
    - Match phrase query
    - Multi match query

```
GET /amazon_products/_search
  {
"query": {

      "term": {
        "manufacturer.raw": "victory multimedia"
}

} }
```

3. Searching - What is Relevant

- The basics of text analysis
- Searching from structured data
- Searching from full-text
    - Match query
    - Match phrase query
    - Multi match query

```
GET /amazon_products/_search
{
  "query": {
    "constant_score": {
      "filter": {
        "term": {
          "manufacturer.raw": "victory multimedia"
        }
      }
    }
  }
}
```

```
{
  ...,
  "hits": {
    "total": 3,
    "max_score": 1,
    "hits": [
      {
        "_index": "amazon_products",
        "_type": "products",
        "_id": "AV5rBfasNI_2eZGciIbg",
        "_score": 1,
        "_source": {
          "price": "19.95",
          "description": ...
      }
    ...
}
```

3. Searching - What is Relevant

- The basics of text analysis
- Searching from structured data
- Searching from full-text
    - Match query
    - Match phrase query
    - Multi match query

Day 2

4. Backups & Recovery

http://{{dns-els}}:{{port}}/_cat/indices?v

1. Backup Data

- Snapshots: Cluster or specific indexes

- Compatibility:

A snapshot of an index created in 5.x can be restored to 6.x.
A snapshot of an index created in 2.x can be restored to 5.x.
A snapshot of an index created in 1.x can be restored to 2.x.

- Plugins for remote servers

Note: Recommendation one different snapshot per major version

4. Backups & Recovery

1. Backup Data

- Creating a backup

```
PUT /_snapshot/backups_<index>?wait_for_completion=true
{
  "type": "fs",
  "settings": {
    "location": "my_unverified_backup_location"
  }
}
```

path.repo: ["/mount/es_backups"

```
curl -XPOST
'http://localhost:9200/_snapshot/backups/backup_201905271530/_restore'
```

```
curl -XPUT
'http://localhost:9200/_snapshot/backups/backup_201905271530?pretty' -H
'Content-Type: application/json' -d'
{
  "indices": "b
  "ignore_unava
  "include_global_state": false
}
'
```

curl -XPOST
'http://localhost:9200/_snapshot/backups/backup_201905271530/_restore'

4. Backups & Recovery

1. Backup Data

- Creating a backup

`location`	Location of the snapshots. Mandatory.
`compress`	Turns on compression of the snapshot files. Compression is applied only to metadata files (index mapping and settings). Data files are not compressed. Defaults to `true`.
`chunk_size`	Big files can be broken down into chunks during snapshotting if needed. The chunk size can be specified in bytes or by using size value notation, i.e. 1g, 10m, 5k. Defaults to `null` (unlimited chunk size).
`max_restore_bytes_per_sec`	Throttles per node restore rate. Defaults to `40mb` per second.
`max_snapshot_bytes_per_sec`	Throttles per node snapshot rate. Defaults to `40mb` per second.
`readonly`	Makes repository read-only. Defaults to `false`.

4. Backups & Recovery

1. Backup Data

- Alternative. **Curator** https://github.com/elastic/curator.git

```
$ curator --help
Usage: curator [OPTIONS] ACTION_FILE

  Curator for Elasticsearch indices.

  See http://elastic.co/guide/en/elasticsearch/client/curator/current

Options:
  --config PATH  Path to configuration file. Default: ~/.curator/curator.yml
  --dry-run      Do not perform any changes.
  --version      Show the version and exit.
  --help         Show this message and exit.
```

```
curator --host <ip address> delete indices --time-unit days --older-than 45 --timestring '%Y%m%d'
```

```
curator --dry-run --host <ip address> delete indices --time-unit days --older-than 45 --timestring '%Y%m%d'
```

```
curator --host <ip address> snapshot --repository <repository name> indices --all-indices
```

```
curator --host <ip address> delete snapshots --repository <repository name> --older-than 10 --time-unit days
```

```
curator show indices - older-than 90 - time-unit days - timestring '%Y.%m.%d'
```

4. Backups & Recovery

1. Backup Data

- Alternative. **Curator** https://github.com/elastic/curator.git

```
actions:
  1:
    action: snapshot
    description: >-
      Snapshot log-production- prefixed indices older than 1 day (based on index
      creation_date) with the default snapshot name pattern of
      'curator-%Y%m%d%H%M%S'.  Wait for the snapshot to complete.  Do not skip
      the repository filesystem access check.  Use the other options to create
      the snapshot.
    options:
      repository: logs backup

    # Leaving name blank will result in the default 'curator-%Y%m%d%H%M%S'
      name: ProductionLogs-%Y%m%d%H%M%S
      ignore_unavailable: False
      include_global_state: True
      partial: False
      wait_for_completion: True
      skip_repo_fs_check: False
      disable_action: False
    filters:
    - filtertype: pattern
      kind: prefix
      value: log-production-
    - filtertype: age
      source: creation_date
      direction: older
      unit: days
      unit_count: 1
```

curator action_snapshot.yml

4. Backups & Recovery

2. Recovery Data

```
POST /_snapshot/my_backup/snapshot_1/_restore
```

```
GET index1,index2/_recovery?human
```

`id`	Shard ID
`type`	Recovery type: • store • snapshot • replica • relocating
`stage`	Recovery stage: • init: Recovery has not started • index: Reading index meta-data and copying bytes from source to destination • start: Starting the engine; opening the index for use • translog: Replaying transaction log • finalize: Cleanup • done: Complete
`primary`	True if shard is primary, false otherwise

4. Backups & Recovery

2. Recovery Data

`start_time`	Timestamp of recovery start
`stop_time`	Timestamp of recovery finish
`total_time_in_millis`	Total time to recover shard in milliseconds
`source`	Recovery source: • repository description if recovery is from a snapshot • description of source node otherwise
`target`	Destination node
`index`	Statistics about physical index recovery
`translog`	Statistics about translog recovery
`start`	Statistics about time to open and start the index

4. Backups & Recovery

2. Recovery Data

```yaml
actions:
  1:
    action: restore
    description: >-
      Restore all indices in the most recent curator-* snapshot with state
      SUCCESS.  Wait for the restore to complete before continuing.  Do not skip
      the repository filesystem access check.  Use the other options to define
      the index/shard settings for the restore.
    options:
      repository: logs_backup
      # If name is blank, the most recent snapshot by age will be selected
      name: ProductionLogs-20170803003417
      # If indices is blank, all indices in the snapshot will be restored
      indices:
      include_aliases: False
      ignore_unavailable: False
      include_global_state: False
      partial: False
      rename_pattern:
      rename_replacement:
      extra_settings:
      wait_for_completion: True
      skip_repo_fs_check: True
      disable_action: False
    filters:
    - filtertype: pattern
      kind: prefix
      value: ProductionLogs-
    - filtertype: state
      state: SUCCESS
```

curator action_snapshot_restore.yml

5. Analysing Log Data

- It is needed a process of
  - collecting logs,
  - extract the relevant information
  - publishing this to Elastic
- Logstash = Extract, Transform, Load
- Logs = Date + Data

5. Analysing Log Data

- Main logs use cases
    - Troubleshooting
    - Understand system/application behaviour
    - Auditing
    - Predictive analytics

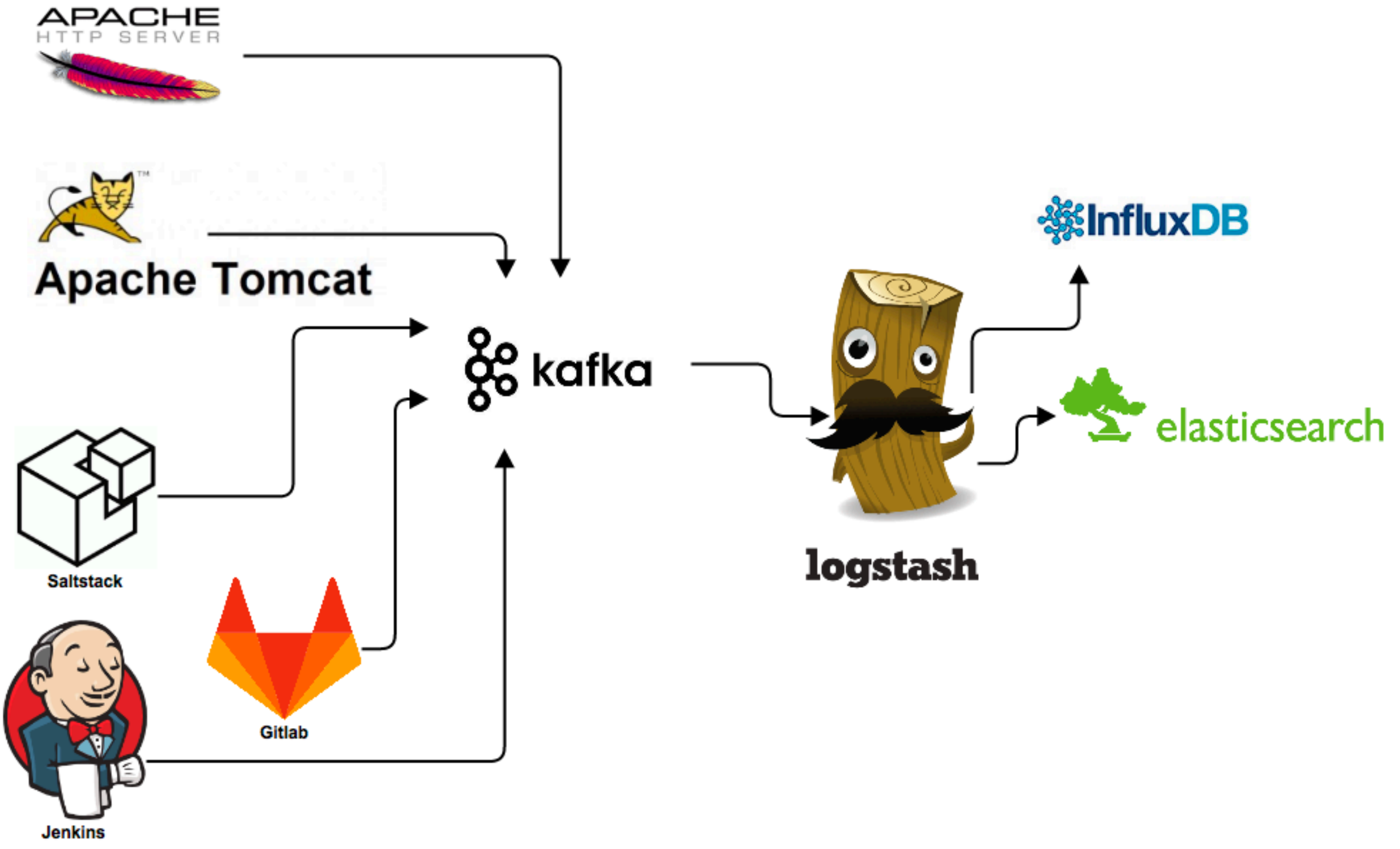
5. Analysing Log Data

- Logs Challenges
    - No common consistent format
    - Logs are decentralised
    - No consistent time format
    - Data is unstructured

5. Analysing Log Data

- **Logstash**
- Open source data collection engine
- real-data pipeline capabilities
- Collecting info from a variety of input sources
- Enrich, unify and store it

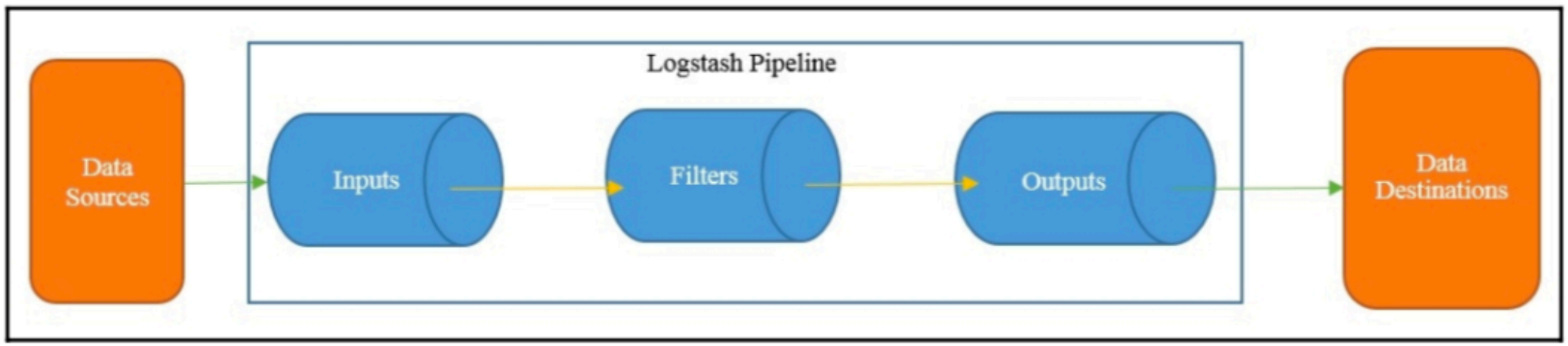
5. Analysing Log Data



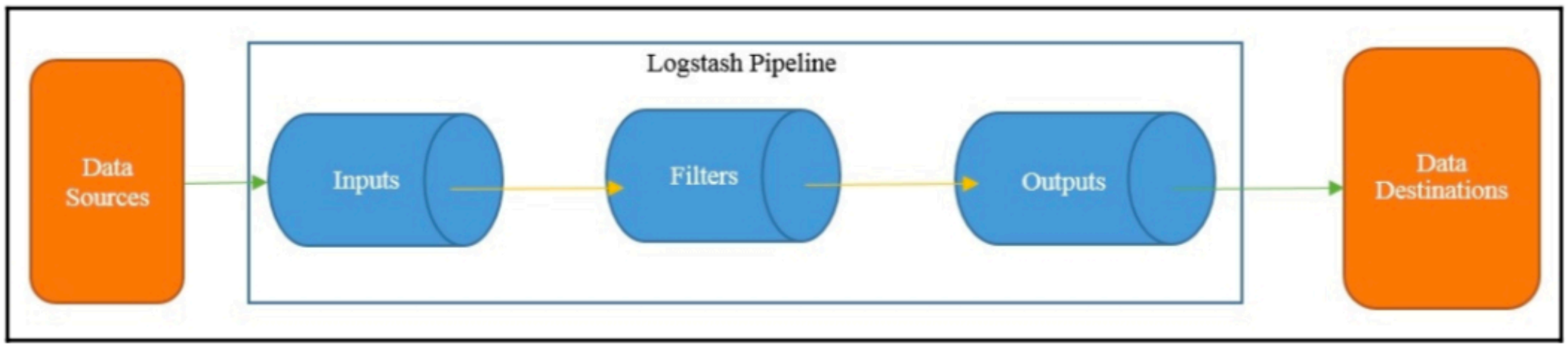
5. Analysing Log Data

- **Logstash Features**
- Pluggable data pipeline architecting:
- Contains over 200 plugins
- Extensibility:
- Logstash is written in JRuby
- Centralized data processing
- Variety and volume
- Synergy with ElasticSearch Beats

5. Analysing Log Data

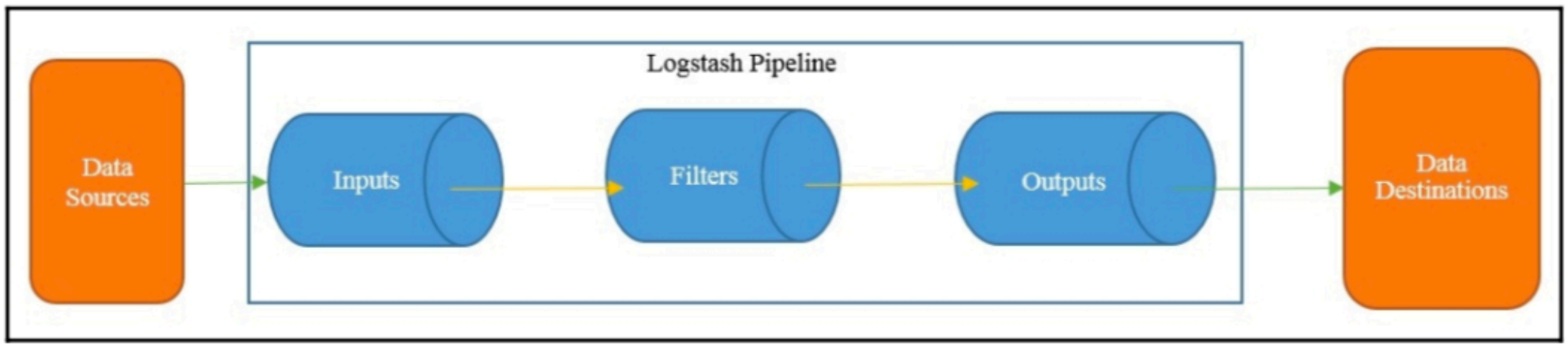


5. Analysing Log Data




```
input
{
}
filter
{
}
output
{
}
```

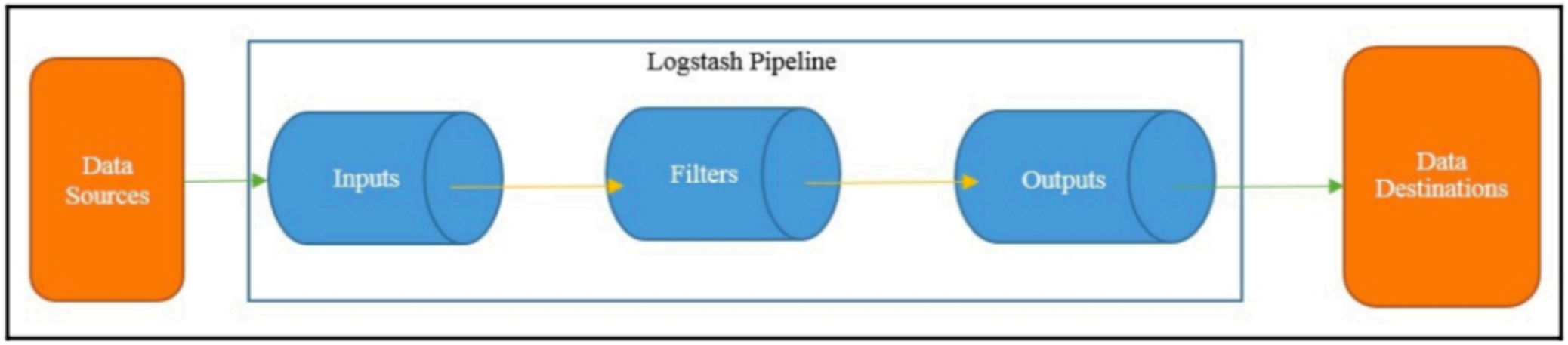
5. Analysing Log Data




```
input
{
}
filter
{
}
output
{
}
```

```
#simple.conf
#A simple logstash configuration

input {
  stdin { }


  }

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

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

5. Analysing Log Data

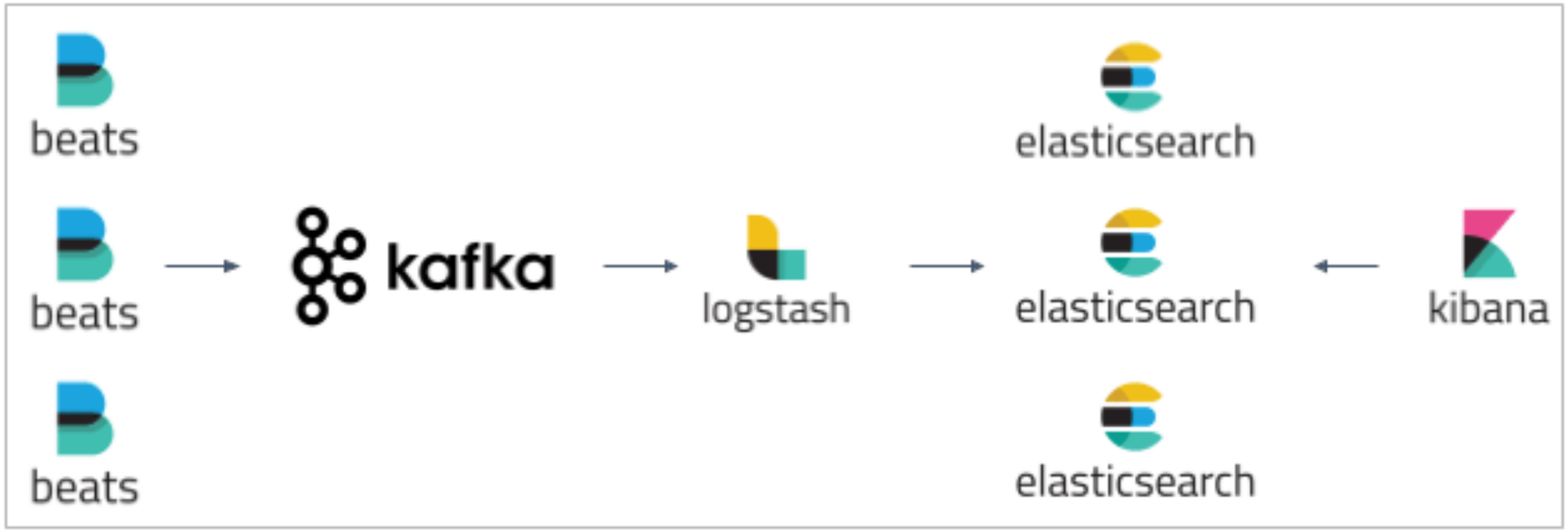



```
input
{
}
filter
{
}
output
{
}
```

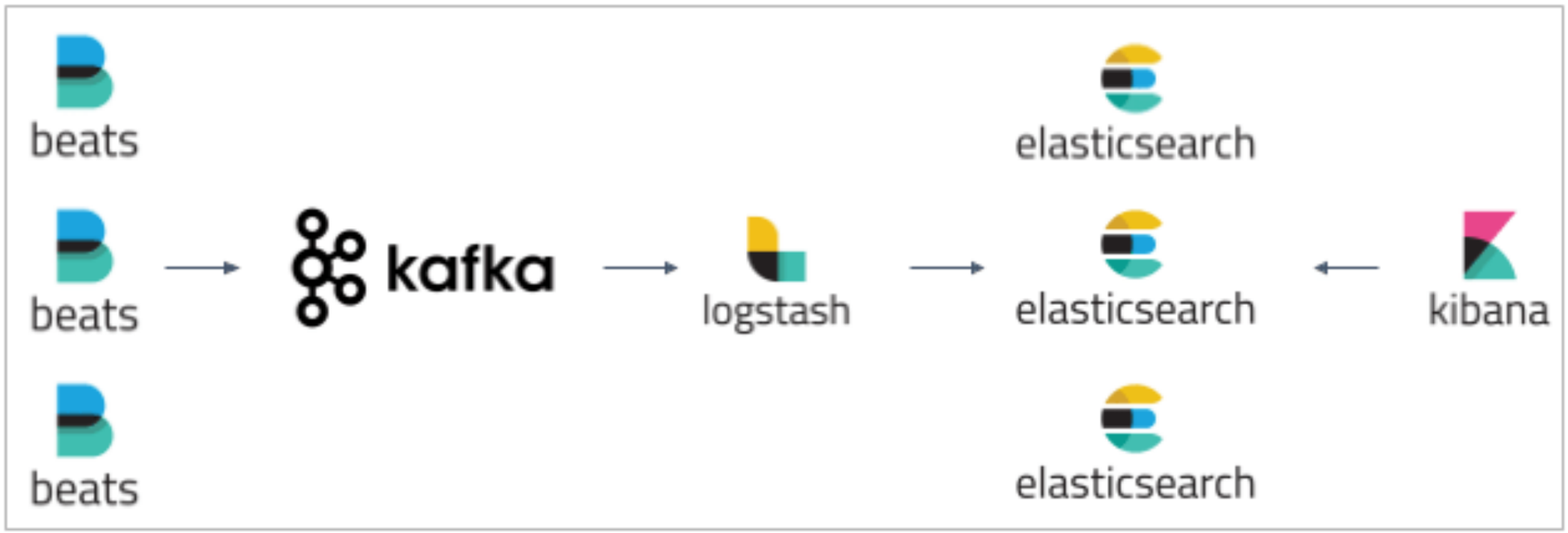
```
#simple.conf
input
{ file{
    path => "/usr/local/logfiles/*"
 }
}

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

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

Type: I am learning ELK!!

5. Analysing Log Data



5. Analysing Log Data



Persistent queues can be enabled by setting the `queue.type: persisted` property in the `logstash.yml` file, which can be found under the `LOGSTASH_HOME/config` folder. `logstash.yml` is a configuration file that contains settings related to Logstash. By default, the files are stored in `LOGSTASH_HOME/data/queue`. You can override this by setting the `path.queue` property in `logstash.yml`.

By default, Logstash starts with a heap size of 1 GB. This can be overridden by setting the `Xms` and `Xmx` properties in the `jvm.options` file, which is found under the `LOGSTASH_HOME/config` folder.

5. Analysing Log Data

- **Input Plugins**

bin/logstash-plugin install *logstash-output-email*

logstash-input-beats	logstash-input-kafka	logstash-input-elasticsearch	logstash-input-ganglia
logstash-input-heartbeat	logstash-input-unix	logstash-input-syslog	logstash-input-stdin
logstash-input-udp	logstash-input-twitter	logstash-input-tcp	logstash-input-sqs
logstash-input-snmptrap	logstash-input-redis	logstash-input-pipe	logstash-input-graphite
logstash-input-s3	logstash-input-rabbitmq	logstash-input-lumberjack	logstash-input-http_poller
logstash-input-exec	logstash-input-file	logstash-input-http	logstash-input-imap
logstash-input-gelf	logstash-input-jdbc	logstash-input-azure_event_hubs	logstash-input-generator

https://www.elastic.co/guide/en/logstash/7.0/input-plugins.html

5. Analysing Log Data

- **Output Plugins**

logstash-output-cloudwatch	logstash-output-csv	logstash-output-udp	logstash-output-webhdfs
logstash-output-elastic_app_search	logstash-output-elasticsearch	logstash-output-email	logstash-output-file
logstash-output-null	logstash-output-lumberjack	logstash-output-http	logstash-output-graphite
logstash-output-nagios	logstash-output-pagerduty	logstash-output-pipe	logstash-output-rabbitmq
logstash-output-redis	logstash-output-s3	logstash-output-sns	logstash-output-sqs
logstash-output-stdout	logstash-output-tcp		

https://www.elastic.co/guide/en/logstash/7.0/output-plugins.html

5. Analysing Log Data

- **Filter Plugins**

logstash-filter-de_dot	logstash-filter-dissect	logstash-filter-dns	logstash-filter-drop
logstash-filter-elasticsearch	logstash-filter-fingerprint	logstash-filter-geoip	logstash-filter-grok
logstash-filter-http	logstash-filter-jdbc_static	logstash-filter-jdbc_streaming	logstash-filter-json
logstash-filter-mutate	logstash-filter-metrics	logstash-filter-memcached	logstash-filter-kv
logstash-filter-ruby	logstash-filter-sleep	logstash-filter-split	logstash-filter-syslog_pri
logstash-filter-throttle	logstash-filter-translate	logstash-filter-urldecode	logstash-filter-truncate
logstash-filter-aggregate	logstash-filter-anonymize	logstash-filter-xml	logstash-filter-useragent
logstash-filter-date	logstash-filter-csv	logstash-filter-clone	logstash-filter-cidr
logstash-filter-anonymize	logstash-filter-aggregate		

https://www.elastic.co/guide/en/logstash/7.0/filter-plugins.html

5. Analysing Log Data

- **Codec Plugins**

logstash-codec-cef	logstash-codec-es_bulk	logstash-codec-json	logstash-codec-multiline
logstash-codec-collectd	logstash-codec-edn_lines	logstash-codec-json_lines	logstash-codec-netflow
logstash-codec-dots	logstash-codec-fluent	logstash-codec-line	logstash-codec-plain
logstash-codec-edn	logstash-codec-graphite	logstash-codec-msgpack	logstash-codec-rubydebug

https://www.elastic.co/guide/en/logstash/7.0/codec-plugins.html

5. Analysing Log Data

- **Try the plugins input file:**
- Path /var/log/kibana
- Start to read at the beginning of the file
- Exclude any other file different than the actual log
- Reach out every 30s
- Add a new field: type = applogs

5. Analysing Log Data

- **Similar solution**

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

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

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

5. Analysing Log Data

- **Other input sources**

```
#email_log.conf
input {
    imap {
      host => "imap.packt.com"
      password => "secertpassword"
      user => "user1@pact.com"
      port => 993
      check_interval => 10
      folder => "Inbox"

    }
}
```

```
output {
    stdout {
        codec => rubydebug
    }
    elasticsearch {
      index => "emails"
      document_type => "email"
      hosts => "localhost:9200"
    }
}
```

5. Analysing Log Data

- **Beats**

The Beats family

All kinds of shippers for all kinds of data.

Filebeat

Metricbeat

Packetbeat

Winlogbeat

Auditbeat

Heartbeat

Functionbeat

5. Analysing Log Data

- **Beats**

  - **Start listen port**

```
#beats.conf

input {
  beats {
    host => "192.168.10.229"
    port => 1234
  }
  beats {
    host => "192.168.10.229"
    port => 5065
  }

}

output {
  elasticsearch {
  }
}
```

https://plani.ishtar.io/survey/1012201901

Netstat -putan | grep -i listen

5. Analysing Log Data

- **Ingest Node**

The ingest node can be used to preprocess documents before the actual indexing is performed on the document.
This preprocessing is performed via an ingest node that intercepts bulk and index requests, applies the transformations to the data, and then passes the documents back to the index or bulk APIs

  - Ingest APIs
    - Define, simulate, remove or find information

5. Analysing Log Data

- **Ingest Node**

```
{
  "description" : "...",
  "processors" : [ ... ]
}
```

5. Analysing Log Data

- **Ingest Node**

```
{
  "description" : "...",
  "processors" : [ ... ]
}
```

- **Put Pipeline API**

```
curl -X PUT http://localhost:9200/_ingest/pipeline/firstpipeline -H
'content-type: application/json'
  -d '{
  "description" : "uppercase the incoming value in the message field",
  "processors" : [
    {
      "uppercase" : {
        "field": "message"
      }
    }
  ]
}'
```

5. Analysing Log Data

- **Ingest Node**

- **Using a Pipeline API**

```
curl -X PUT 'http://localhost:9200/myindex/mytpe/1?pipeline=firstpipeline' -H 'content-type: application/json' -d '{

    "message":"elk is awesome"

}'
```

5. Analysing Log Data

- **Ingest Node**

- **Using a Pipeline API**

```
curl -X PUT 'http://localhost:9200/myindex/mytpe/1?pipeline=firstpipeline' -H 'content-type: application/json' -d '{

    "message":"elk is awesome"

}'
```

- **Delete a Pipeline API**

```
curl -X DELETE http://localhost:9200/_ingest/pipeline/firstpipeline  -H
'content-type: application/json'
```

5. Analysing Log Data

- **Ingest Node**

  - **Simulate a Pipeline API**

```
curl -X POST
http://localhost:9200/_ingest/pipeline/firstpipeline/_simulate -H 'content-
type: application/json' -d '{
  "docs" : [
    { "_source": {"message":"first document"} },
    { "_source": {"message":"second document"} }
  ]
}'
```

5. Analysing Log Data

- **Exercise**

  Create a pipeline where the result is:

  Input: elk is awesome

  Output:

```
"_source": {
    "label": "testlabel",
    "data": "ELK IS AWESOME"
}
```

6. Building Data Pipelines with Logstash

- Parsing and enriching logs using Logstash
- The Elastic Beats platform
- Installing and configuring Filebeats for shipping logs

6. Building Data Pipelines with Logstash

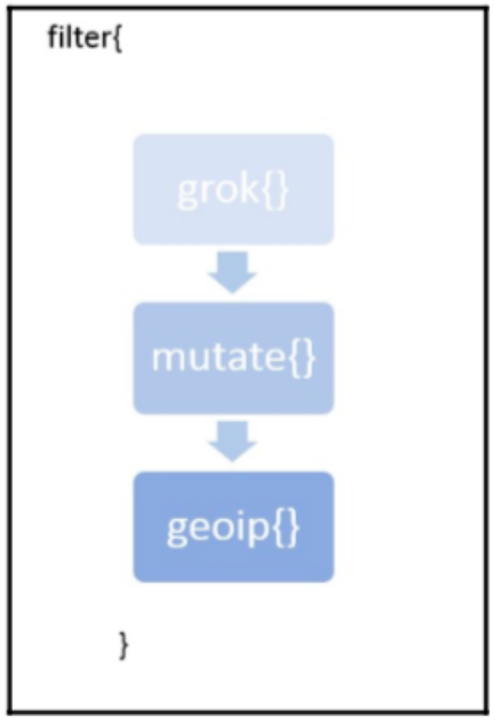
- Parsing and enriching logs using Logstash

```
logdata = timestamp + data
```

6. Building Data Pipelines with Logstash

- Parsing and enriching logs using Logstash

```
logdata = timestamp + data
```



6. Building Data Pipelines with Logstash

- Parsing and enriching logs using Logstash:

- CSV Filter

users.csv:

```
FName,LName,Age,Salary,EmailId,Gender
John,Thomas,25,50000,John.Thomas,m
Raj, Kumar,30,5000,Raj.Kumar,f
Rita,Tony,27,60000,Rita.Tony,m
```

- Añadir en Elastic estos datos a través de log stash

- Indice: users

- Cambiar datatypes: Age: Integer, Salary: float, Genero en uppercase

- Renombrar: FName = First Name, LName = Last Name

- Eliminar espacios en blanco de los campos FName y LName

6. Building Data Pipelines with Logstash

- Parsing and enriching logs using Logstash:

- CSV Filter

```
#csv_file.conf
input {
   file{
      path => "D:\es\logs\users.csv"
      start_position => "beginning"
      }
}

filter {
     csv{
    autodetect_column_names => true
        }
}

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

6. Building Data Pipelines with Logstash

- Parsing and enriching logs using Logstash:

- CSV Filter

```
#csv_file_mutuate.conf
input {
   file{
      path => "D:\es\logs\users.csv"
      start_position => "beginning"
      sincedb_path => "NULL"
   }

}
```

```
filter {
  csv{
    autodetect_column_names => true
  }
  mutate {
    convert => {
        "Age" => "integer"
        "Salary" => "float"
      }
    rename => { "FName" => "Firstname"
                "LName" => "Lastname" }
    gsub => [
       "EmailId", "\.", "_"
      ]
    strip => ["Firstname", "Lastname"]
    uppercase => [ "Gender" ]
  }
}

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

6. Building Data Pipelines with Logstash

- Parsing and enriching logs using Logstash:

- Grok Filter

https://github.com/logstash-plugins/logstash-patterns-core/blob/master/patterns/grok-patterns

6. Building Data Pipelines with Logstash

- Parsing and enriching logs using Logstash:

- Grok Filter

https://github.com/logstash-plugins/logstash-patterns-core/blob/master/patterns/grok-patterns

- Add custom patterns

6. Building Data Pipelines with Logstash

- Parsing and enriching logs using Logstash:

- Grok Filter

https://github.com/logstash-plugins/logstash-patterns-core/blob/master/patterns/grok-patterns

  - Add custom patterns
  - Test: https://grokdebug.herokuapp.com/

6. Building Data Pipelines with Logstash

- Parsing and enriching logs using Logstash:

- Grok Filter

https://github.com/logstash-plugins/logstash-patterns-core/blob/master/patterns/grok-patterns

- Add custom patterns

- Test: https://grokdebug.herokuapp.com/

%{LOGDATE:log_date};%{WORD:uuid};%{LOGLEVEL:loglevel};(%{IP:ip})?;%{VERSION:version};%{WORD:endpoint}?;%{UNIXPATH:resource};

6. Building Data Pipelines with Logstash

- Parsing and enriching logs using Logstash:
- Date Filter:
- Logstash add a timestamp metadata: @timestamp

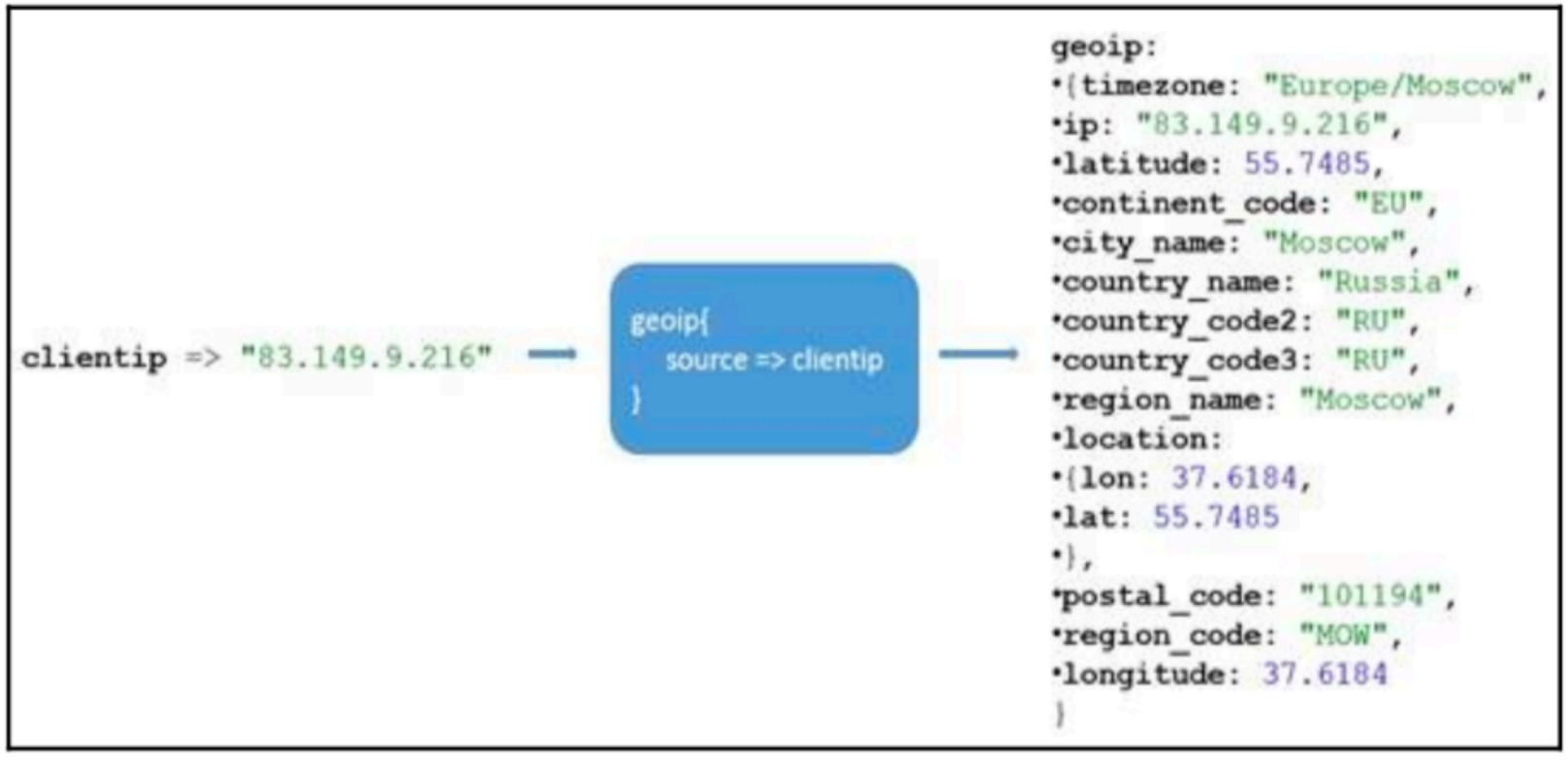
```
filter {
    date {
    match => [ "timestamp", "dd/MMM/YYYY:HH:mm:ss Z" ]
    target => "event_timestamp"
         }
}
```

6. Building Data Pipelines with Logstash

- Parsing and enriching logs using Logstash:
- Geoip Filter:
- GeoLite2 City Database

6. Building Data Pipelines with Logstash

- Parsing and enriching logs using Logstash:
- Geoip Filter:
- GeoLite2 City Database



6. Building Data Pipelines with Logstash

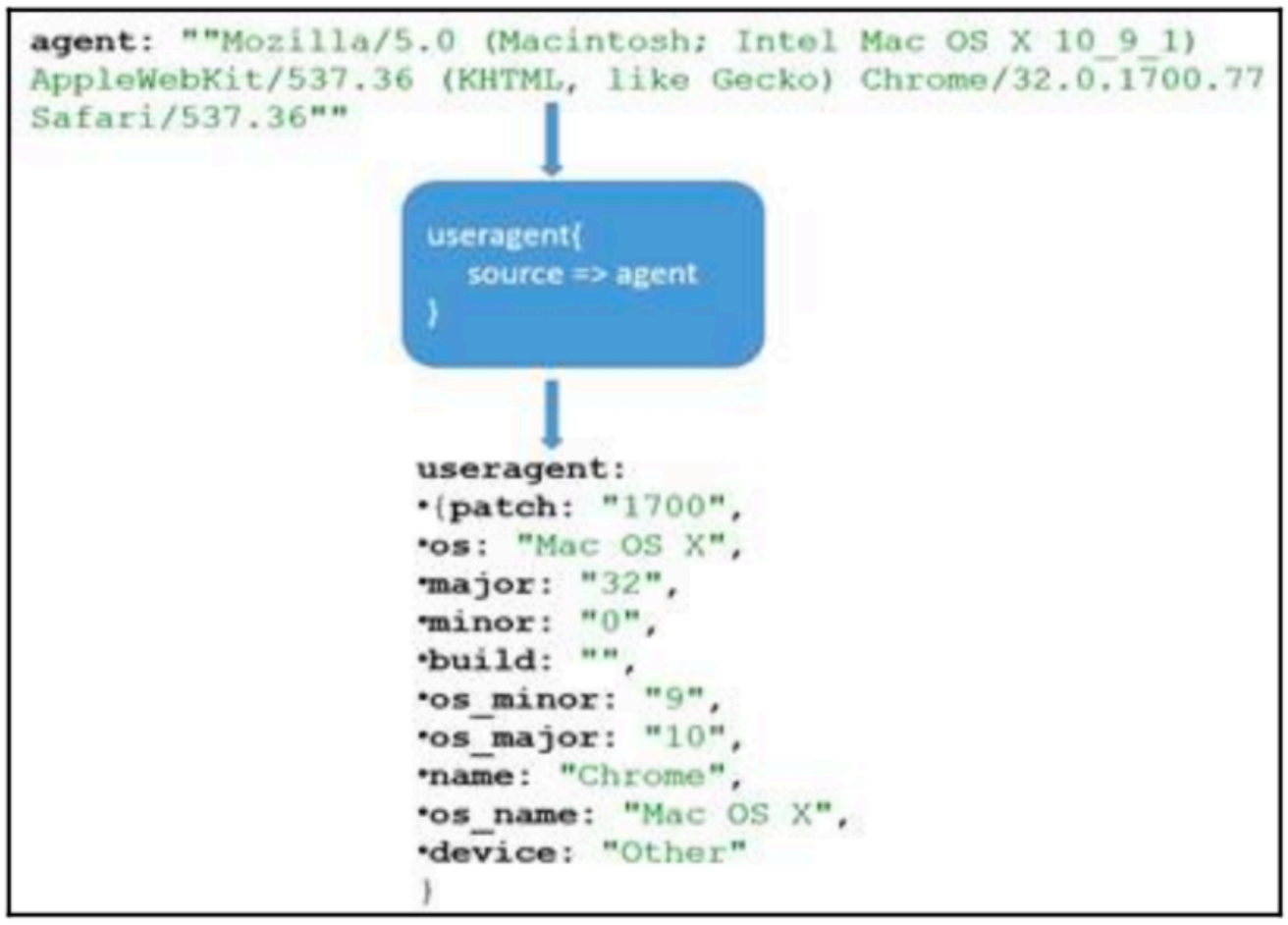
- Parsing and enriching logs using Logstash:

- UserAgent Filter:

6. Building Data Pipelines with Logstash

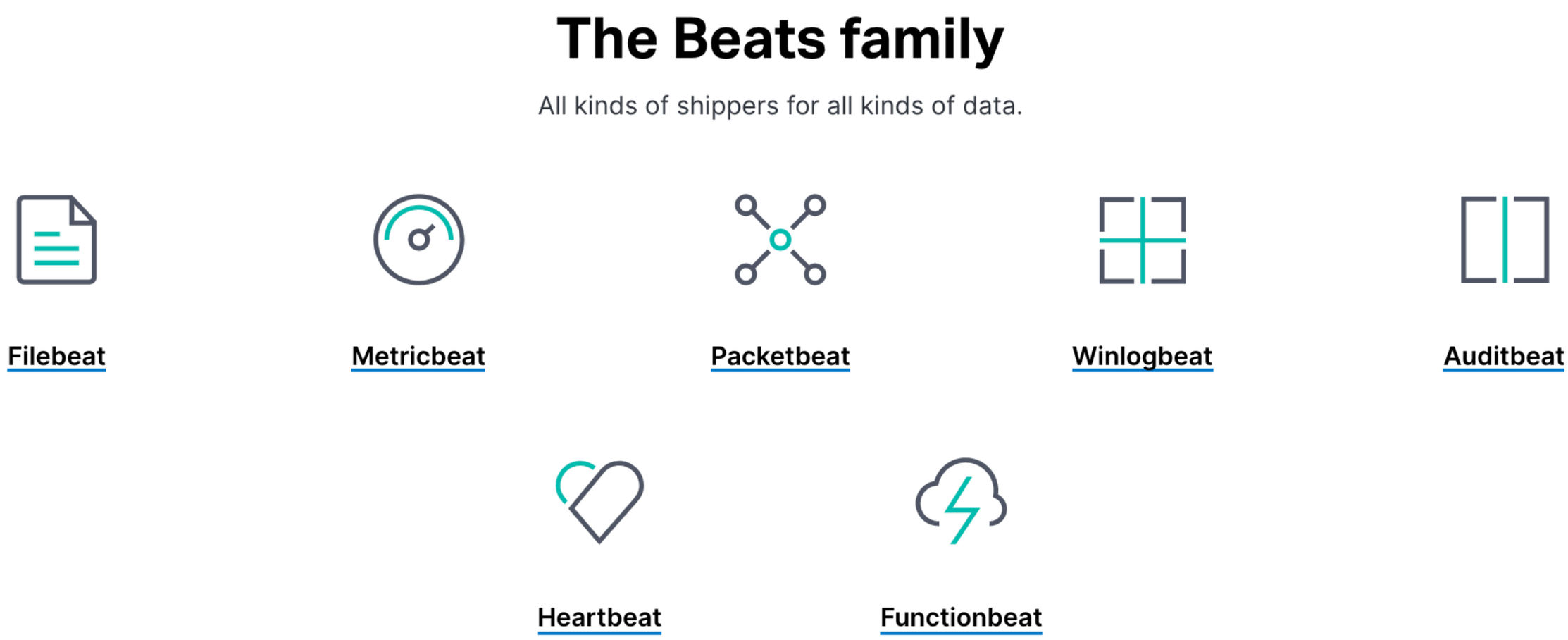
- Parsing and enriching logs using Logstash:

- UserAgent Filter:



6. Building Data Pipelines with Logstash

- Parsing and enriching logs using Logstash
- The Elastic Beats platform



https://www.elastic.co/guide/en/beats/devguide/current/community-beats.html

6. Building Data Pipelines with Logstash

- Parsing and enriching logs using Logstash
- The Elastic Beats platform
- Installing and configuring Filebeats for shipping logs

7. Visualising Data with Kibana

- Downloading and installing Kibana
- Preparing data
- Kibana UI
- Timelion
- Using plugins

7. Visualising Data with Kibana

- Downloading and installing Kibana
    - Node.js
    - Checking status

7. Visualising Data with Kibana

- Downloading and installing Kibana
    - Node.js
    - Checking status

`server.port`	This setting specifies the port Kibana will be serving requests on. It defaults to `5601`.
`server.host`	This specifies the address to which the Kibana server will bind. IP addresses and hostnames are both valid values. It defaults to `localhost`.
`elasticsearch.url`	This is the URL of the Elasticsearch instance to use for all your queries. It defaults to `http://localhost:9200`. If your Elasticsearch is running on a different host/port, make sure you update this property.
`elasticsearch.username` `elasticsearch.password`	If Elasticsearch is secured, specify the username/password details that have access to Elasticsearch here. In the next chapter (`Chapter 8`, *Elastic X-pack*), we will be exploring how to secure Elasticsearch.
`server.name`	A human-readable display name that identifies this Kibana instance. Defaults to hostname.
`kibana.index`	Kibana uses an index in Elasticsearch to store saved searches, visualizations, and dashboards. Kibana creates a new index if the index doesn't already exist. Defaults to `.kibana`.

7. Visualising Data with Kibana

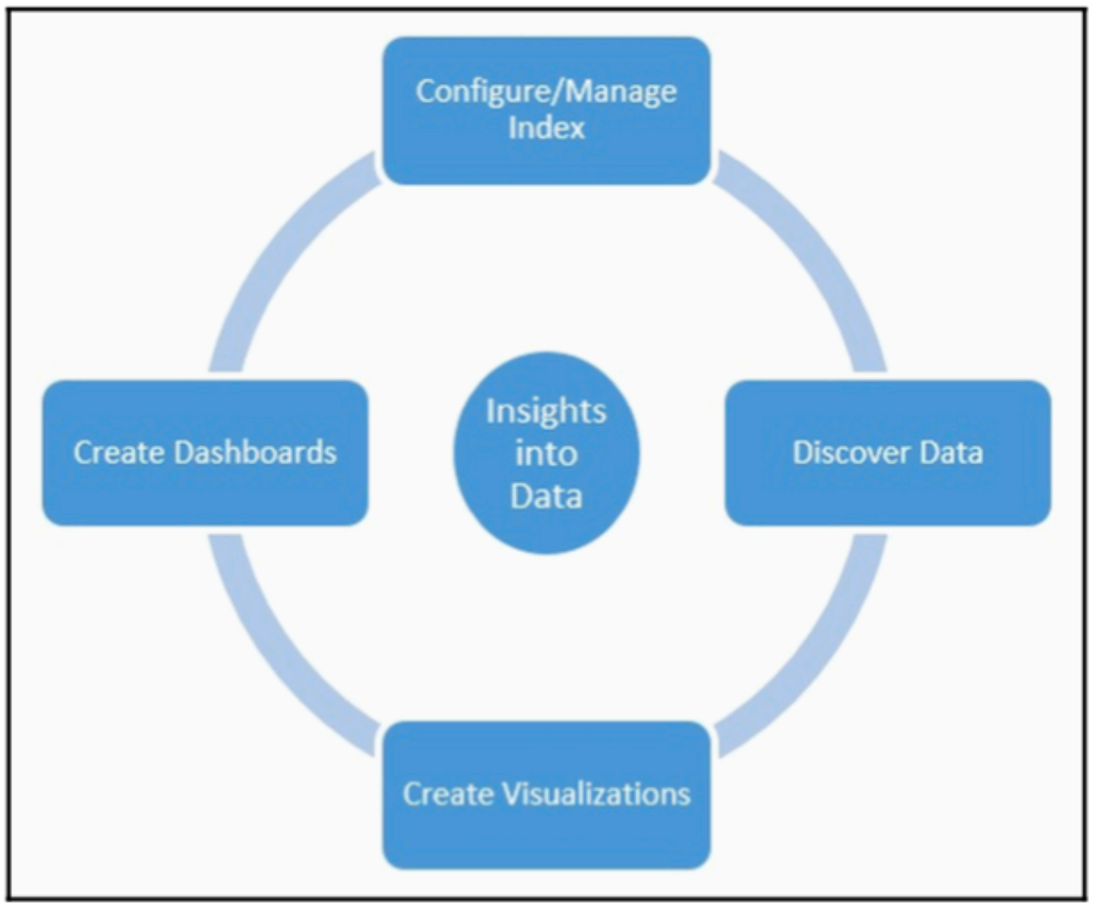
- Downloading and installing Kibana
- Preparing data:
  - Load example data
  - Add Logs from Apache

7. Visualising Data with Kibana

- Downloading and installing Kibana
- Preparing data
- Kibana UI

7. Visualising Data with Kibana

- Downloading and installing Kibana
- Preparing data
- Kibana UI



7. Visualising Data with Kibana

- Downloading and installing Kibana
- Preparing data
- Kibana UI
- Exploring Kibana

● **RED:** Damnit. Some or all of (primary) shards are not ready.

● **YELLOW:** Elasticsearch has allocated all of the primary shards, but some/all of the replicas have not been allocated.

● **GREEN:** Great. Your cluster is fully operational. Elasticsearch is able to allocate all shards and replicas to machines within the cluster.

7. Visualising Data with Kibana

- Downloading and installing Kibana
- Preparing data
- Kibana UI
- Configure index pattern

7. Visualising Data with Kibana

- Downloading and installing Kibana
- Preparing data
- Kibana UI
- Discover: JSON, Lucene, DSL , KQL

7. Visualising Data with Kibana

- Downloading and installing Kibana
- Preparing data
- Kibana UI
- Discover

Buscar todos los documentos que presenten las palabras files logstash

7. Visualising Data with Kibana

- Downloading and installing Kibana
- Preparing data
- Kibana UI
- Discover

Buscar todos los documentos que tengan la ciudad de Amsterdam

7. Visualising Data with Kibana

- Downloading and installing Kibana
- Preparing data
- Kibana UI
- Discover

Buscar todos los documentos que tengan la ciudad de Fairfield y respuesta Diferente a 200

7. Visualising Data with Kibana

- Downloading and installing Kibana
- Preparing data
- Kibana UI
- Discover

Operadores booleanas: AND, OR Y - (must not)

Buscar todos los documentos que tengan la ciudad de Fairfield y respuesta Diferente a 200

7. Visualising Data with Kibana

- Downloading and installing Kibana
- Preparing data
- Kibana UI
- Discover

Buscar todos los documentos que tengan el rango de respuesta entre 300 y 500

7. Visualising Data with Kibana

- Downloading and installing Kibana
- Preparing data
- Kibana UI
- Discover

Se aceptan regex en las queries: ?, *

7. Visualising Data with Kibana

- Downloading and installing Kibana
- Preparing data
- Kibana UI
- Visualizations
  - Bucket aggregations
  - Metric aggregations

7. Visualising Data with Kibana

- **Histogram**: This type of aggregation works only on numeric fields and, given the value of the numeric field and the interval, it works by distributing them into fixed-size interval buckets. For example, a histogram can be used to find the number of products per price range, with an interval of 100.
- **Date Histogram**: This is a type of histogram aggregation that works only on date fields. It works by distributing them into fixed-size date interval buckets. It supports date/time-oriented intervals such as 2 hours, days, weeks, and so on. Kibana provides various intervals including auto, millisecond, second, minute, hour, day, week, month, year, and custom, for ease of use. Using the `Custom` option, date/time-oriented intervals such as 2 hours, days, weeks, and so on, can be supplied. This histogram is ideal for analyzing time-series data—for example, finding the total number of incoming web requests per week/day.
- **Range**: This is similar to histogram aggregations; however, rather than fixed intervals, ranges can be specified. Also, it not only works on numeric fields, but it can work on dates and IP addresses. Multiple ranges can be specified using `from` and `to` values—for example, finding the number of employees falling in the age ranges 0-25, 25-35, 35-50, and 50 and above.

This type of aggregation includes the `from` value and excludes the `to` value for each range.

- **Terms**: This type of aggregation works by grouping documents based on each unique term in the field. This aggregation is ideal for finding the top *n* values for a field—for example, finding the top five countries based on the number of incoming web requests.

This aggregation works on `keyword` fields only.

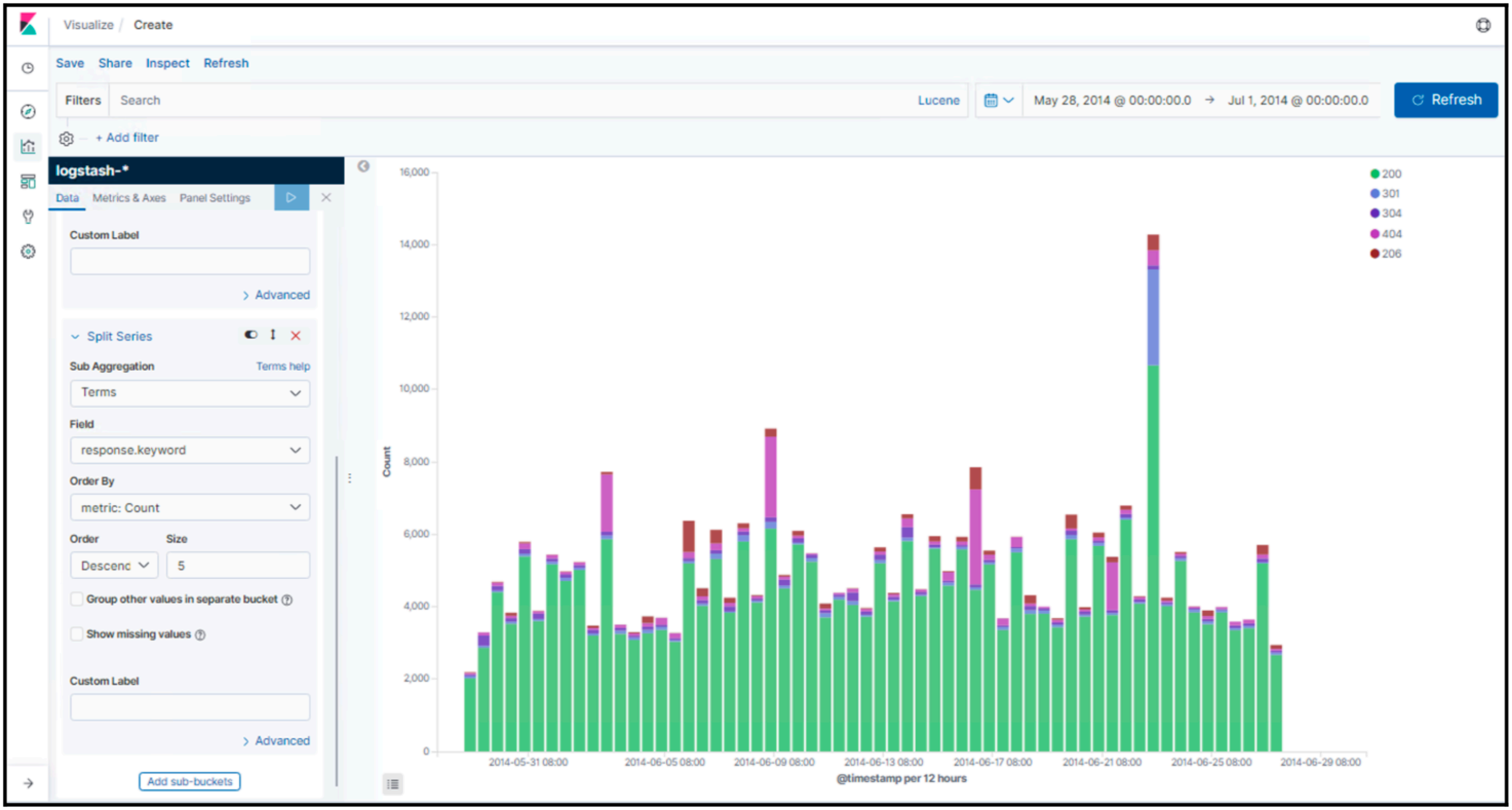
- **Filters**: This aggregation is used to create buckets based on a filter condition. This aggregation allows for the comparison of specific values. For example, finding the average number of web requests in India compared to the US.

- **GeoHash Grid**: This aggregation works with fields containing `geo_point` values. This aggregation is used for plotting `geo_points` on a map by grouping them into buckets—for example, visualizing web request traffic over different geographies.

- **Count**: The default metric in Kibana visualizations; returns the count of documents
- **Average**: Used to compute the average value (for a field) of all the documents in the bucket
- **Sum**: Used to compute the sum value (for a field) of all the documents in the bucket
- **Median**: Used to compute the median value (for a field) of all the documents in the bucket
- **Min**: Used to compute the minimum value (for a field) of all the documents in the bucket
- **Max**: Used to compute the maximum value (for a field) of all the documents in the bucket
- **Standard deviation**: Used to compute the standard deviation (for a field) of all the documents in the bucket
- **Percentiles**: Used to compute the number of percentile values
- **Percentile ranks**: For a set of percentiles, this is used to compute the corresponding values

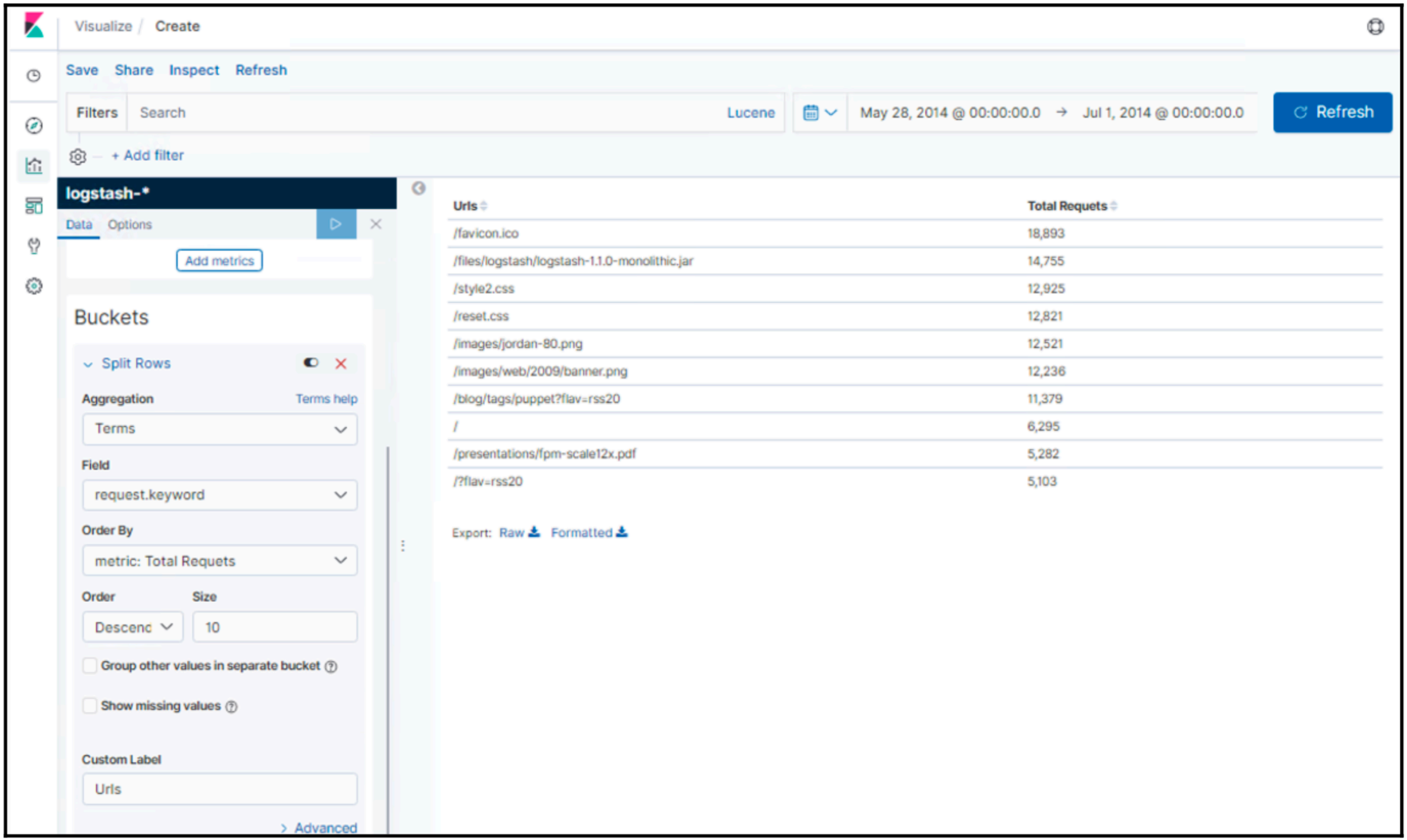
7. Visualising Data with Kibana

Response codes over time



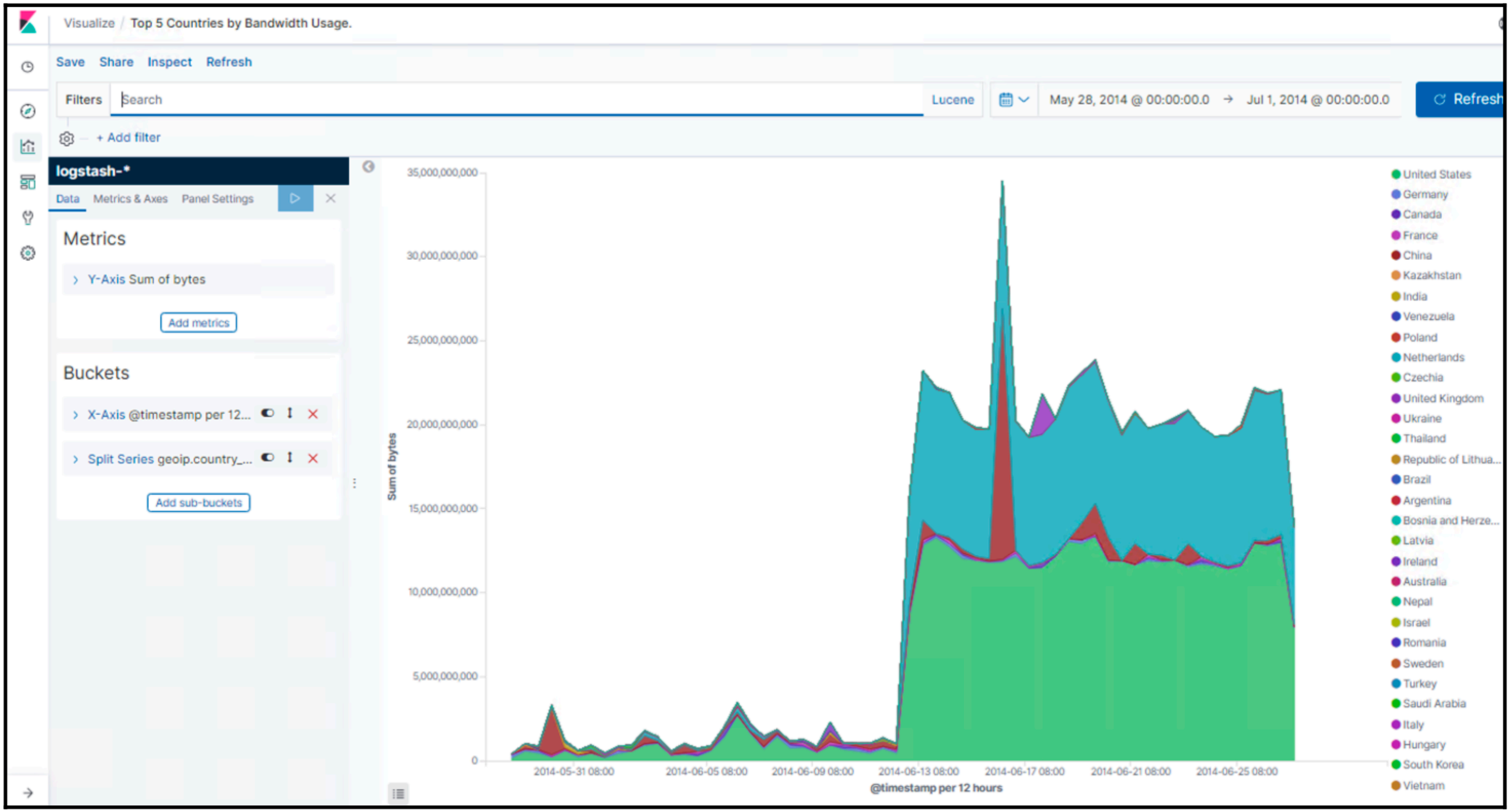
7. Visualising Data with Kibana

Top ten requested URLs



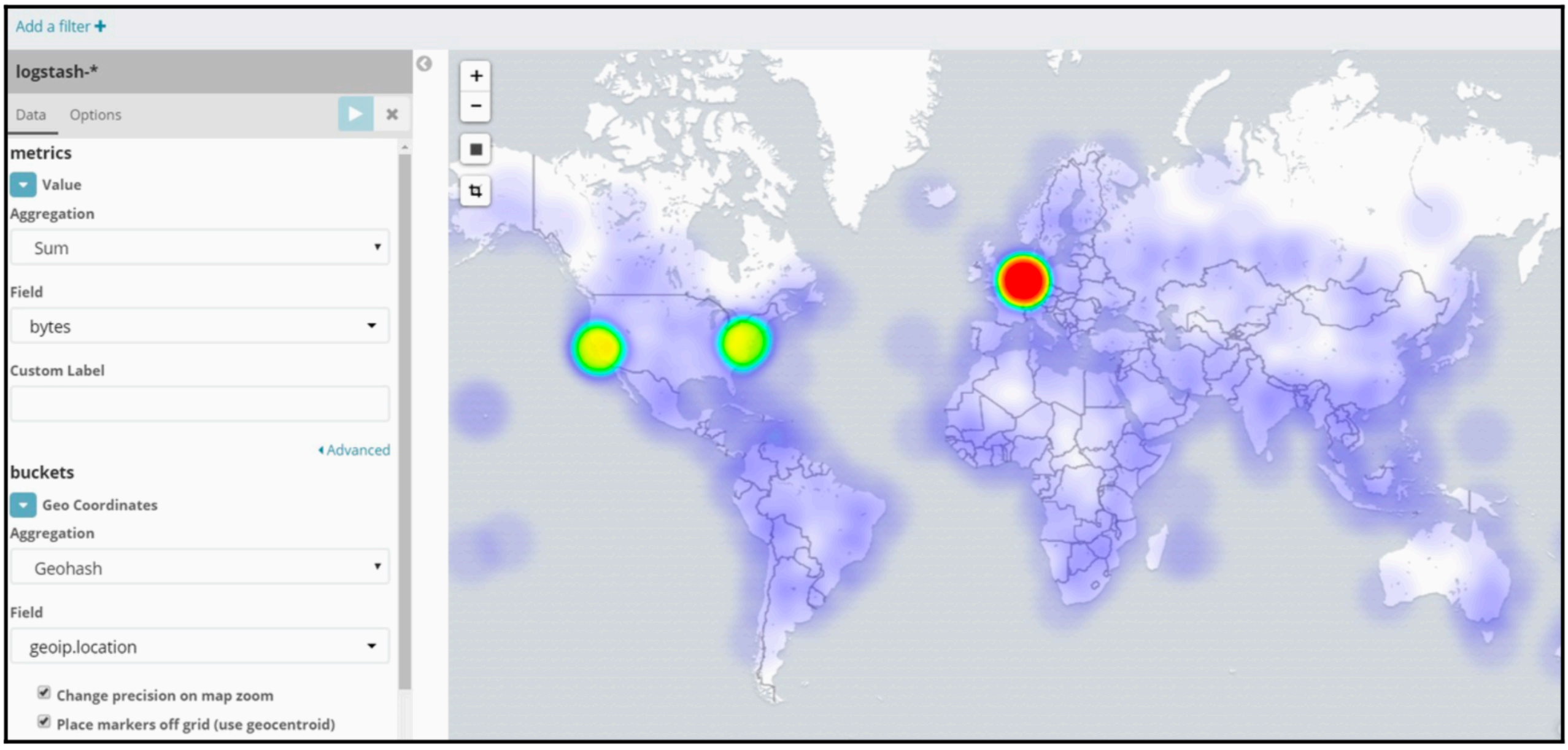
7. Visualising Data with Kibana

Bandwidth usage of the top five countries



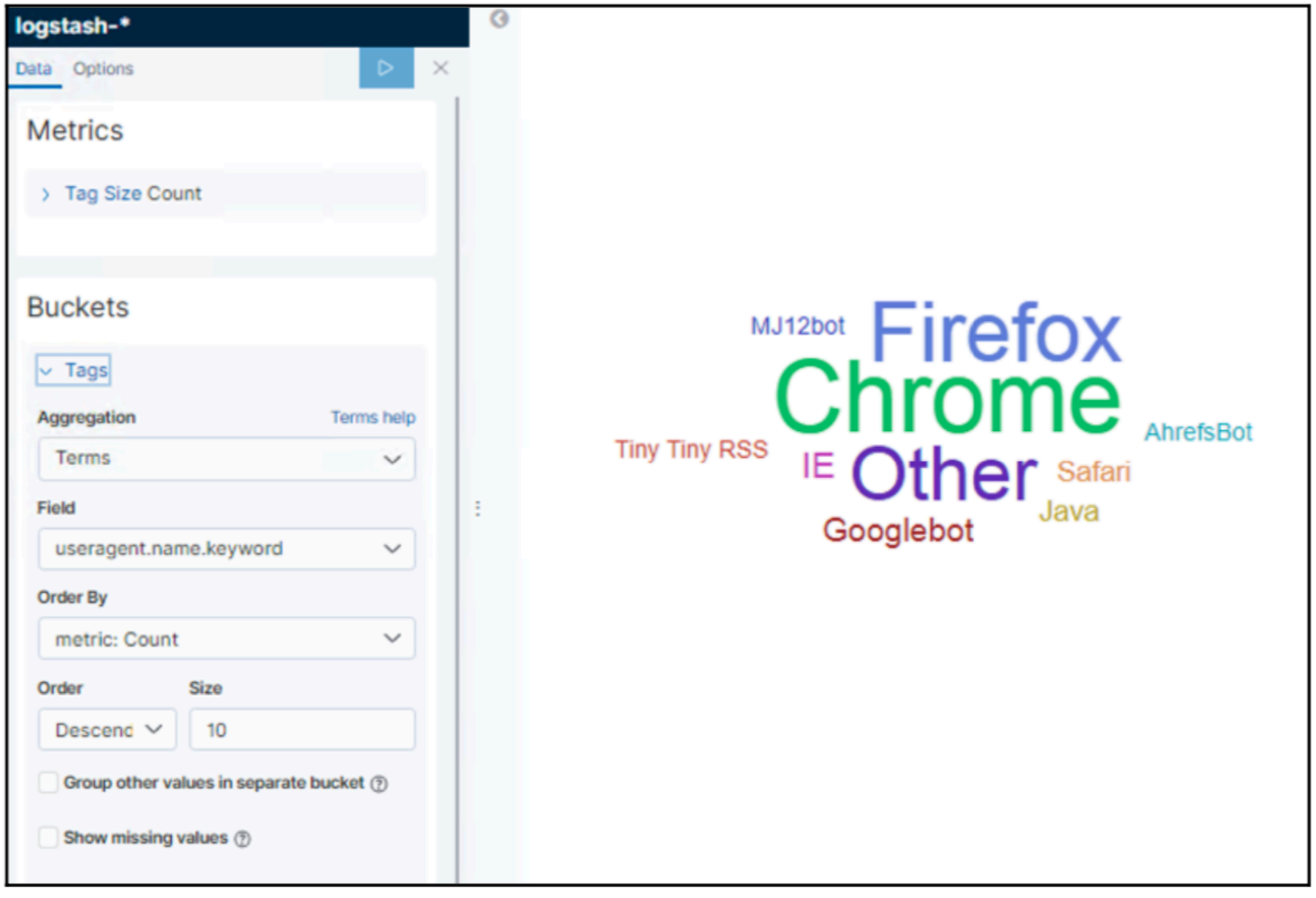
7. Visualising Data with Kibana

Traffic by country



7. Visualising Data with Kibana

Most used user agent



7. Visualising Data with Kibana

- Downloading and installing Kibana
- Preparing data
- Kibana UI
- Creating a dashboard
- Saving a dashboard
- Cloning a dashboard
- Sharing a dashboard

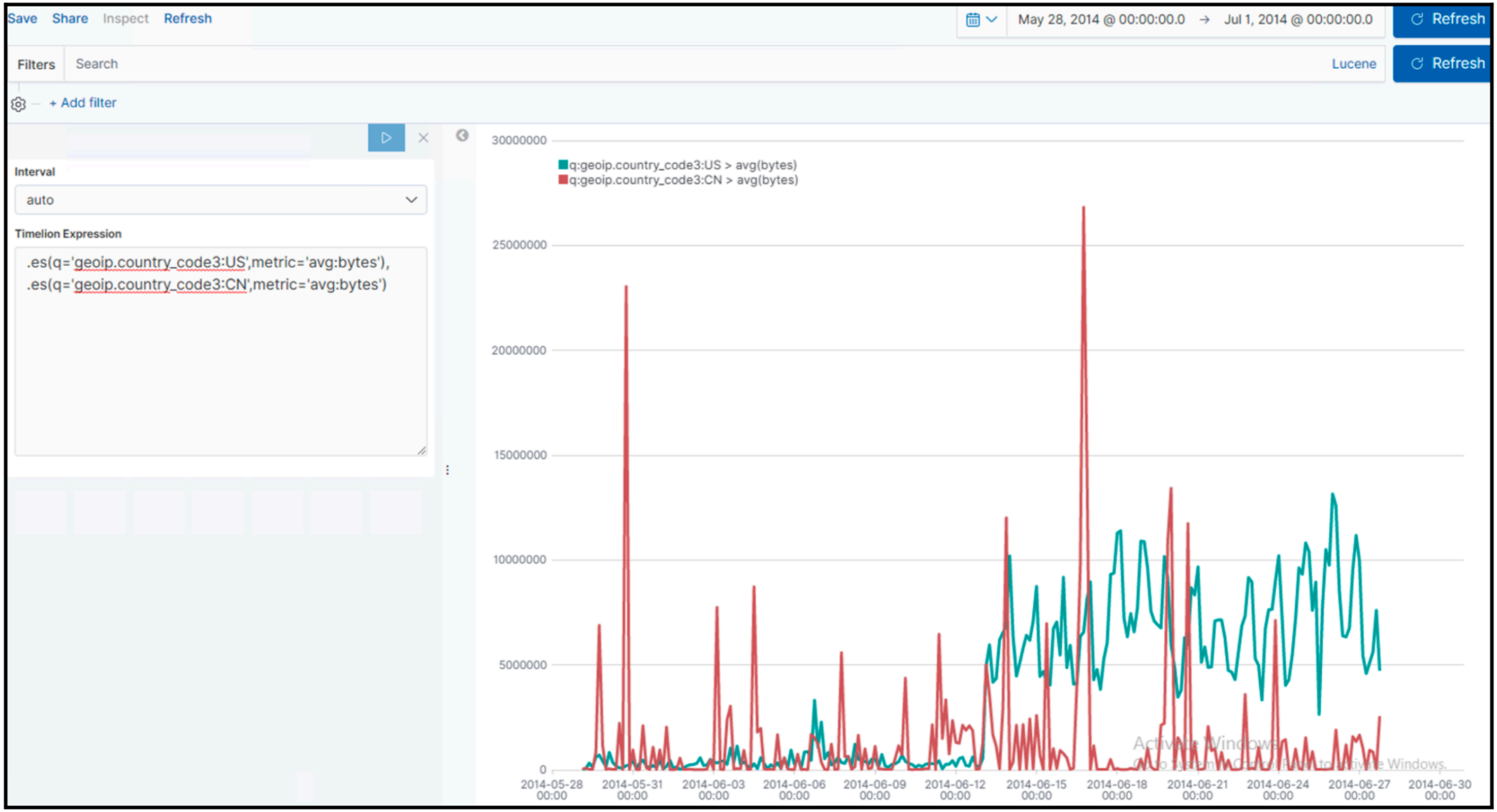
7. Visualising Data with Kibana

- Downloading and installing Kibana
- Preparing data
- Kibana UI
- Timelion

https://github.com/elastic/timelion/blob/master/FUNCTIONS.md

7. Visualising Data with Kibana

Bytes usage over time for the US and China



7. Visualising Data with Kibana

- Downloading and installing Kibana
- Preparing data
- Kibana UI
- Timelion
- Using plugins

bin/kibana-plugin install

bin/kibana-plugin remove