

Elastic Stack 7.0

Table of Contents

1. Introducing Elastic Stack: 3
2. Getting Started with Elasticsearch: 34
3. Searching - What is Relevant: 102
4. Analytics with Elasticsearch: 189
5. Analyzing Log Data: 271
6. Building Data Pipelines with Logstash: 338
7. Visualizing Data with Kibana: 398
8. Elastic X-Pack: 496 
9. Running Elastic Stack in Production: 588
10. Building a Sensor Data Analytics Application: 632
Monitoring Server Infrastructure: 680



1. Introducing Elastic Stack



Introducing Elastic Stack

In this lesson, we will cover the following topics:

- What is Elasticsearch, and why use it?
- A brief history of Elasticsearch and Apache Lucene
- Elastic Stack components
- Use cases of Elastic Stack



What is Elasticsearch, and why use it?

Let's look at the key benefits of using Elasticsearch as your data store:

- Schemaless, document-oriented
- Searching
- Analytics
- Rich client library support and the REST API
- Easy to operate and easy to scale
- Near real-time
- Lightning-fast
- Fault-tolerant



Schemaless and document-oriented

- JSON documents naturally support this type of data.
For example, take a look at the following document:

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

Schemaless and document-oriented

- This document may represent a customer's record. Here the record has the name, address, and age fields of the customer.
- Another record may look like the following:

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

Searching capability

- The core strength of Elasticsearch lies in its text-processing capabilities.
- Elasticsearch is great at searching, especially full-text searches, Let's understand what a full-text search is:
- Full-text search means searching through all the terms of all the documents available in the database.
- This requires the entire contents of all documents to be parsed and stored beforehand. When you hear full-text search, think of Google Search.

Analytics

- Apart from searching, the second most important functional strength of Elasticsearch is analytics.
- Yes, what was originally known as just a full-text search engine is now used as an analytics engine in a variety of use cases.
- Many organizations are running analytics solutions powered by Elasticsearch in production. 

Rich client library support and the REST API

- Elasticsearch has very rich client library support to make it accessible to many programming languages.
- There are client libraries available for Java, C#, Python, JavaScript, PHP, Perl, Ruby, and many more.
- Apart from the official client libraries, there are community-driven libraries for 20 plus programming languages.

Easy to operate and easy to scale

- Elasticsearch can run on a single node and easily scale out to hundreds of nodes.
- It is very easy to start a single node instance of Elasticsearch; it works out of the box without any configuration changes and scales to hundreds of nodes.



Near real-time capable

- Typically, data is available for queries within a second after being indexed (saved).
- Not all big data storage systems are real-time capable.
- Elasticsearch allows you to index thousands to hundreds of thousands of documents per second and makes them available for searching almost immediately.

Lightning-fast

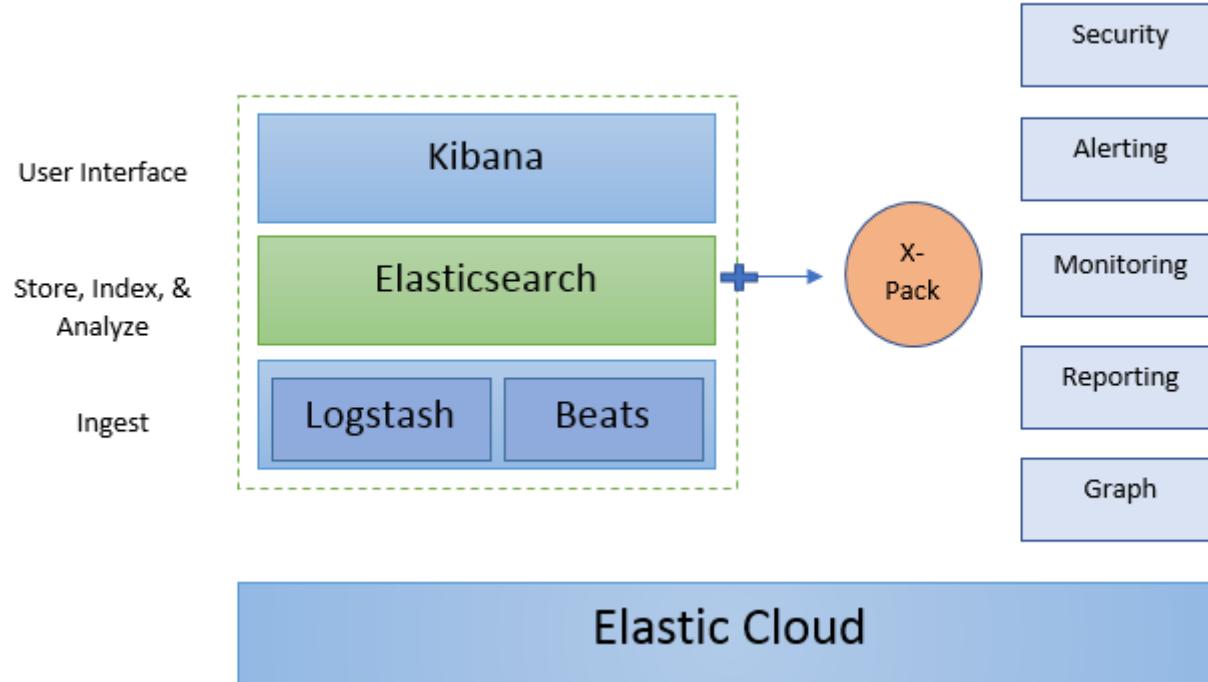
- Elasticsearch uses Apache Lucene as its underlying technology. By default, Elasticsearch indexes all the fields of your documents.
- This is extremely invaluable as you can query or search by any field in your records.
- You will never be in a situation in which you think, If only I had chosen to create an index on this field.

Fault-tolerant

- Elasticsearch clusters can keep running even when there are hardware failures such as node failure and network failure.
- In the case of node failure, it replicates all the data on the failed node to another node in the cluster.
- In the case of network failure, Elasticsearch seamlessly elects master replicas to keep the cluster running.

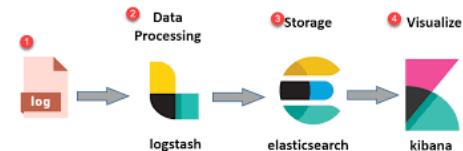
Exploring the components of the Elastic Stack

Elastic Stack: Real Time Search & Analytics at Scale



Elasticsearch

- Elasticsearch is at the heart of the Elastic Stack.
- It stores all your data and provides search and analytic capabilities in a scalable way.
- We have already looked at the strengths of Elasticsearch and why you would want to use it.
- Elasticsearch can be used without using any other components to power your application in terms of search and analytics.



Logstash

- Logstash helps centralize event data such as logs,metrics, or any other data in any format.
- It can perform a number of transformations before sending it to a stash of your choice.
- It is a key component of the Elastic Stack, used to centralize the collection and transformation processes in your data pipeline.

Beats

- Beats is a platform of open-source lightweight data shippers. Its role is complementary to Logstash.
- Logstash is a server-side component, whereas Beats has a role on the client side.
- Beats consists of a core library, libbeat, which provides an API for shipping data from the source, configuring the input options, and implementing logging.



Kibana

- Kibana is the visualization tool for the Elastic Stack, and can help you gain powerful insights about your data in Elasticsearch.
- It is often called a window into the Elastic Stack.
- It offers many visualizations including histograms, maps, line charts, time series, and more.
- You can build visualizations with just a few clicks and interactively explore data.

X-Pack

- X-Pack adds essential features to make the Elastic Stack production-ready.
- It adds security, monitoring, alerting, reporting, graph, and machine learning capabilities to the Elastic Stack.



Elastic Cloud

- Elastic Cloud is the cloud-based, hosted, and managed setup of the Elastic Stack components.
- The service is provided by Elastic (<https://www.elastic.co/>), which is behind the development of Elasticsearch and other Elastic Stack components.
- All Elastic Stack components are open source except X-Pack (and Elastic Cloud).

Use cases of Elastic Stack

The following list of example use cases is by no means exhaustive, but highlights some of the most common ones:

- Log and security analytics
- Product search
- Metrics analytics
- Web searches and website searches



Log and security analytics

Application support teams face a great challenge in administering and managing large numbers of applications deployed across tens or hundreds of servers. The application infrastructure could have the following components:

- Web servers
- Application servers
- Database servers
- Message brokers



Log and security analytics

- Typically, enterprise applications have all, or most, of the types of servers described earlier, and there are multiple instances of each server.
- In the event of an error or production issue, the support team has to log in to individual servers and look at the errors.
- It is quite inefficient to log in to individual servers and look at the raw log files.
- The Elastic Stack provides a complete toolset to collect, centralize, analyze, visualize, alert, and report errors as they occur.

Product search

- A product search involves searching for the most relevant product from thousands or tens of thousands of products and presenting the most relevant products at the top of the list before other, less relevant, products.
- You can directly relate this problem to e-commerce websites, which sell huge numbers of products sold by many vendors or resellers.

Metrics analytics

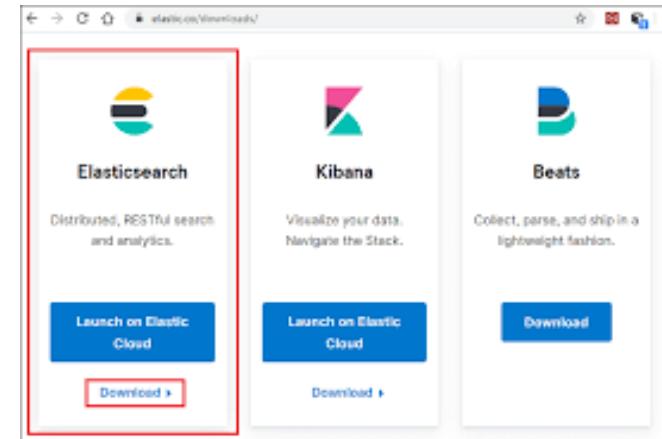
- Elastic Stack has excellent analytics capabilities, thanks to the rich Aggregations API in Elasticsearch.
- This makes it a perfect tool for analyzing data with lots of metrics.
- Metric data consists of numeric values as opposed to unstructured text such as documents and web pages.
- Some examples are data generated by sensors, Internet of Things (IoT) devices, metrics generated by mobile devices, servers, virtual machines, network routers, switches, and so on. The list is endless.

Web search and website search

- Elasticsearch can serve as a search engine for your website and perform a Google-like search across the entire content of your site.
- GitHub, Wikipedia, and many other platforms power their searches using Elasticsearch.
- Elasticsearch can be leveraged to build content aggregation platforms.

Downloading and installing

- Now that we have enough motivation and reasons to learn about Elasticsearch and the Elastic Stack, let's start by downloading and installing the key components.
- Firstly, we will download and install Elasticsearch and Kibana.
- We will install the other components as we need them on the course of our journey.



Installing Elasticsearch

We will use the ZIP format as it is the least intrusive and the easiest for development purposes:

- Go to <https://www.elastic.co/downloads/elasticsearch> and download the ZIP distribution. You can also download an older version if you are looking for an exact version.
- Extract the file and change your directory to the top-level extracted folder. Run bin/elasticsearch or bin/elasticsearch.bat.
- Run curl <http://localhost:9200> or open the URL in your favorite browser.

You should see an output like this

```
$ curl http://localhost:9200?pretty
{
  "name" : "Pranav-MBP.local",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "bbmcs19iS4uKr_7qo5cC9Q",
  "version" : {
    "number" : "7.0.1",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "e4efcb5",
    "build_date" : "2019-04-29T12:56:03.145736Z",
    "build_snapshot" : false,
    "lucene_version" : "8.0.0",
    "minimum_wire_compatibility_version" : "6.7.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
$
```

Installing Kibana

Kibana is also available in a variety of packaging formats such as ZIP, TAR.GZ, RMP, and DEB for 32-bit and 64-bit architecture machines:

- Go to <https://www.elastic.co/downloads/kibana> and download the ZIP or TAR.GZ distribution for the platform that you are on.
- Extract the file and change your directory to the top-level extracted folder. Run bin/kibana or bin/kibana.bat. Open <http://localhost:5601> in your favorite browser.

Summary

- In this lesson, we started by understanding the motivations of various search and analytics technologies other than relational databases and NoSQL stores.
- We looked at the strengths of Elasticsearch, which is at the heart of the Elastic Stack.
- We then looked at the rest of the components of the Elastic Stack and how they fit into the ecosystem.



COMPLETE LAB 1

2. Getting Started with Elasticsearch



Getting Started with Elasticsearch

We will cover the following topics in this lesson:

- Using the Kibana Console UI
- Core concepts of Elasticsearch
- CRUD operations
- Creating indexes and taking control of mapping
- REST API overview



Using the Kibana Console UI

- Before we start writing our first queries to interact with Elasticsearch, we should familiarize ourselves with a very important tool: Kibana Console.
- This is important because Elasticsearch has a very rich REST API, allowing you to do all sorts of operations with Elasticsearch.
- Kibana Console has an editor that is very capable and aware of the REST API.
- It allows for auto completion, and for the formatting of queries as you write them.

Console - Kibana

localhost:5601/app/kibana#/dev_tools/console?_g=()

Apps iGoogle Elasticsearch Addons Webservices - RE... Hadoop Spark IoT Startup Scala Big Data Utilities Log Analytics

Dev Tools

History Settings Help

Console Search Profiler Grok Debugger

1 GET /

```
1+ {  
2   "name" : "Pranav-MBP.local",  
3   "cluster_name" : "elasticsearch",  
4   "cluster_uuid" : "bbmcs19iS4uKr_7qo5cC9Q",  
5+   "version" : {  
6     "number" : "7.0.1",  
7     "build_flavor" : "default",  
8     "build_type" : "tar",  
9     "build_hash" : "e4efcb5",  
10    "build_date" : "2019-04-29T12:56:03.145736Z",  
11    "build_snapshot" : false,  
12    "lucene_version" : "8.0.0",  
13    "minimum_wire_compatibility_version" : "6.7.0",  
14    "minimum_index_compatibility_version" : "6.0.0-beta1"  
15+   },  
16   "tagline" : "You Know, for Search"  
17+ }  
18
```

Console - Kibana

localhost:5601/app/kibana#/dev_tools/console?_g=()

Apps iGoogle Elasticsearch Addons Webservices - RE... Hadoop Spark IoT Startup Scala Big Data Utilities Log Analytics

Dev Tools

History Settings Help

Console Search Profiler Grok Debugger

```
1 GET /
2
3
4
5 GET /my_index/_m
  _mapping endpoint
  _mapping/field endpoint
  _mget endpoint
  _migration/deprecations endpoint
  _msearch endpoint
  _msearch/template endpoint
  _mtermvectors endpoint
  _ilm/explain endpoint
```

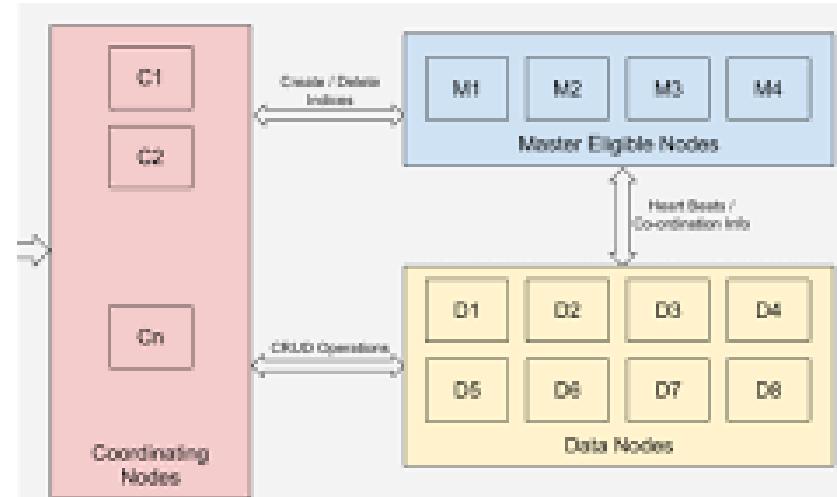
```
1 {
2   "name" : "Pranav-MBP.local",
3   "cluster_name" : "elasticsearch",
4   "cluster_uuid" : "bbmcs19iS4uKr_7qo5cC9Q",
5   "version" : {
6     "number" : "7.0.1",
7     "build_flavor" : "default",
8     "build_type" : "tar",
9     "build_hash" : "e4efcb5",
10    "build_date" : "2019-04-29T12:56:03.145736Z",
11    "build_snapshot" : false,
12    "lucene_version" : "8.0.0",
13    "minimum_wire_compatibility_version" : "6.7.0",
14    "minimum_index_compatibility_version" : "6.0.0
15    -beta1"
16  },
17  "tagline" : "You Know, for Search"
18 }
```

localhost:5601/app/kibana#/dev_tools/searchprofiler

Core concepts of Elasticsearch

We will look at the following core abstractions of Elasticsearch:

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



Core concepts of Elasticsearch

Let's start learning about these with an example:

```
PUT /catalog/_doc/1
```

```
{
```

```
  "sku": "SP000001",
```

```
  "title": "Elasticsearch for Hadoop",
```

```
  "description": "Elasticsearch for Hadoop",
```

```
  "author": "Vishal Shukla",
```

```
  "ISBN": "1785288997",
```

```
  "price": 26.99
```

```
}
```

Core concepts of Elasticsearch

- All of the examples that are written for the Kibana Console UI can be very easily converted into curl commands that can be executed from the command line.
- The following is the curl version of the previous Kibana Console UI command:

```
curl -XPUT http://localhost:9200/catalog/_doc/1 -d '{ "sku":  
"SP000001", "title": "Elasticsearch for Hadoop",  
"description": "Elasticsearch for Hadoop", "author": "Vishal  
Shukla", "ISBN": "1785288997", "price": 26.99}'
```

Indexes

Index

Type

Document

Types

- In our example of a product catalog, the document that was indexed was of the product type.
- Each document stored in the product type represents one product.
- Since the same index cannot have other types, such as customers, orders, and order line items, and more, types help in logically grouping or organizing the same kind of documents within an index.

Types

- The following code is for the index for customers:

```
PUT /customers/_doc/1
{
  "firstName": "John",
  "lastName": "Smith",
  "contact": {
    "mobile": "212-xxx-yyyy"
  },
  ...
}
```

Types

- The following code is for the index for products:

```
PUT /products/_doc/1
{
  "title": "Apple iPhone Xs (Gold, 4GB RAM, 64GB Storage, 12 MP Dual Camera, 458 PPI Display)",
  "price": 999.99,
  ...
}
```

Documents

- Documents contain multiple fields. Each field in the JSON document is of a particular type.
- In the product catalog example that we saw earlier, these fields were sku, title, description, and price.
- Each field and its value can be seen as a key-value pair in the document, where key is the field name and value is the field value.
- The field name is similar to a column name in a relational database.

Documents



These fields are as follows:

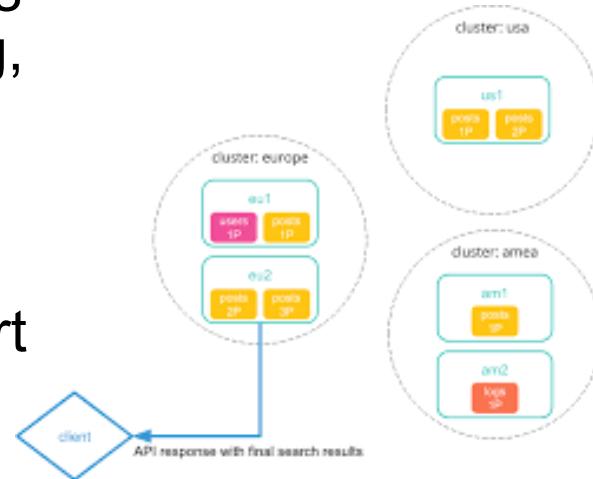
- _id: This is the unique identifier of the document within the type, just like a primary key in a database table. It can be autogenerated or specified by the user.
- _type: This field contains the type of the document.
- _index: This field contains the index name of the document.

Nodes

- An Elasticsearch node is a single server of Elasticsearch, which may be part of a larger cluster of nodes.
- It participates in indexing, searching, and performing other operations that are supported by Elasticsearch.
- Every Elasticsearch node is assigned a unique ID and name when it is started.
- A node can also be assigned a static name via the `node.name` parameter in the Elasticsearch configuration file, `config/elasticsearch.yml`.

Clusters

- A cluster hosts one or more indexes and is responsible for providing operations such as searching, indexing, and aggregations.
- A cluster is formed by one or more nodes.
- Every Elasticsearch node is always part of a cluster, even if it is just a single node cluster.



Shards and replicas

- First, let's understand what a shard is, An index contains documents of one or more types.
- Shards help in distributing an index over the cluster.
- Shards help in dividing the documents of a single index over multiple nodes.
- There is a limit to the amount of data that can be stored on a single node, and that limit is dictated by the storage, memory, and processing capacities of that node.

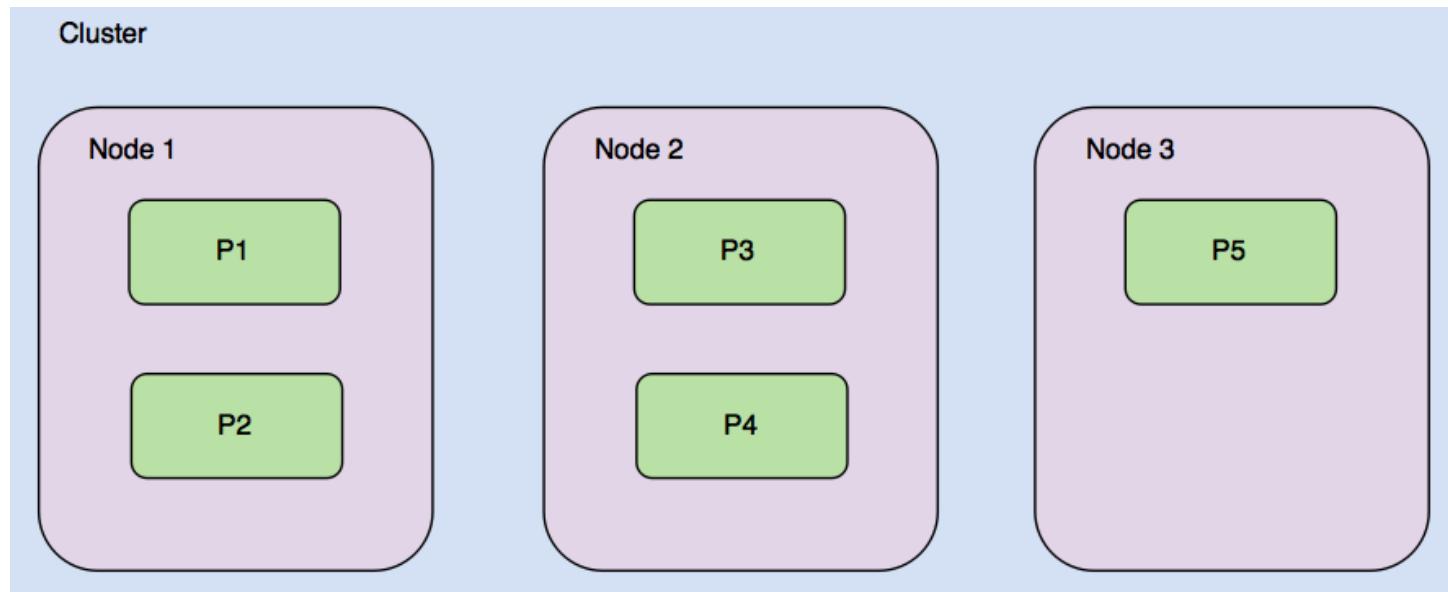
Documents

The process of dividing the data among shards is called sharding. Sharding is inherent in Elasticsearch and is a way of scaling and parallelizing, as follows:

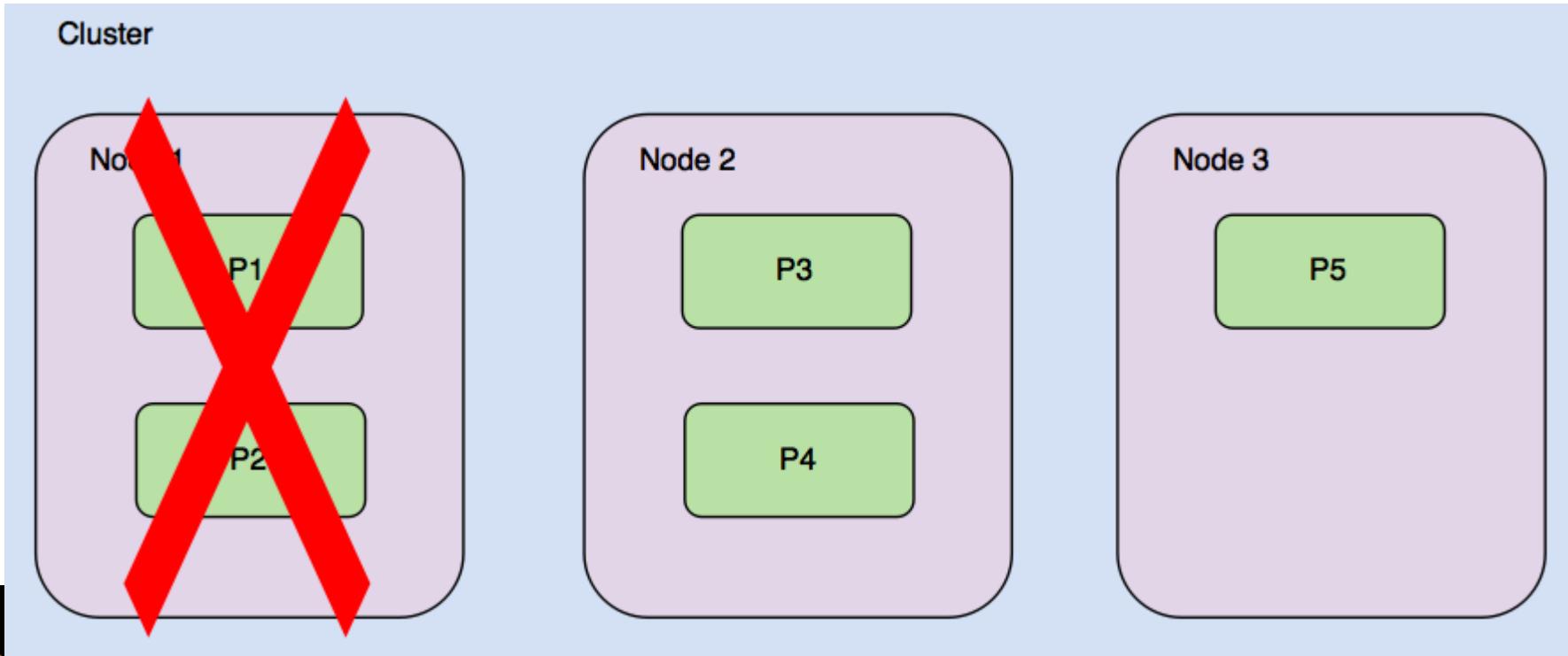
- It helps in utilizing storage across different nodes of the cluster
- It helps in utilizing the processing power of different nodes of the cluster

Documents

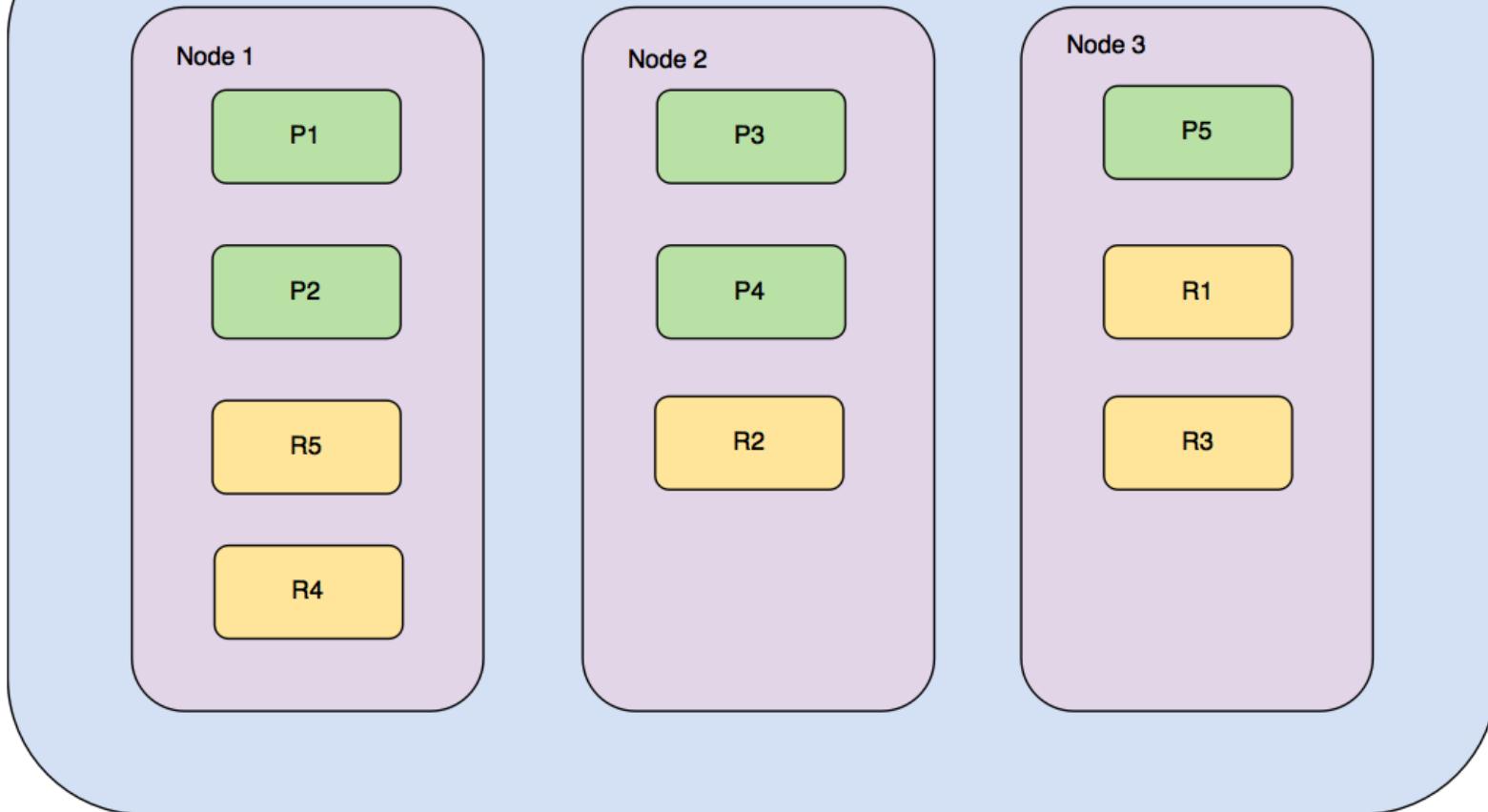
- The following diagram illustrates how five shards of one index may be distributed on a three-node cluster:



- Now, imagine that one of the nodes (Node 1) goes down.
- With Node 1, we also lose the share of data, which was stored in shards P1 and P2:



Cluster



Mappings and datatypes

- Elasticsearch is schemaless, meaning that you can store documents with any number of fields and types of fields.
- In a real-world scenario, data is never completely schemaless or unstructured.
- There are always some sets of fields that are common across all documents in a type.
- In fact, types within the indexes should be created based on common fields.



Datatypes

- Elasticsearch supports a wide variety of datatypes for different scenarios where you want to store text data, numbers, booleans, binary objects, arrays, objects, nested types, geo-points, geo-shapes, and many other specialized datatypes, such as IPv4 and IPv6 addresses.
- In a document, each field has a datatype associated with it.

Mappings

```
PUT /catalog/_doc/2
{
  "sku": "SP000002",
  "title": "Google Pixel Phone 32GB - 5 inch display",
  "description": "Google Pixel Phone 32GB - 5 inch display  
(Factory Unlocked US Version)",
  "price": 400.00,
  "resolution": "1440 x 2560 pixels",
  "os": "Android 7.1"
}
```

Mappings

Remember, unlike relational databases, we didn't have to define the fields that would be part of each document. In fact, we didn't even have to create an index with the name catalog. When the first document about the product type was indexed in the index catalog, the following tasks were performed by Elasticsearch:

- Creating an index with the name catalog
- Defining the mappings for the type of documents that will be stored in the index's default type – `_doc`

Creating an index with the name catalog

- The first step involves creating an index, because the index doesn't exist already.
- The index is created using the default number of shards.
- We will look at a concept called index templates – you can create templates for any new indexes.
- Sometimes, an index needs to be created on the fly, just like in this case, where the insertion of the first document triggers the creation of a new index.

Defining the mappings for the type of product

- The second step involves defining the mappings for the type of product.
- This step is executed because the type catalog did not exist before the first document was indexed.
- Remember the analogy of type with a relational database table.
- The table needs to exist before any row can be inserted.

Defining the mappings for the type of product

- To see the mappings of the product type in the catalog index, execute the following command in the Kibana Console UI:

GET /catalog/_mapping



Defining the mappings for the type of product

- The response should look like the following:

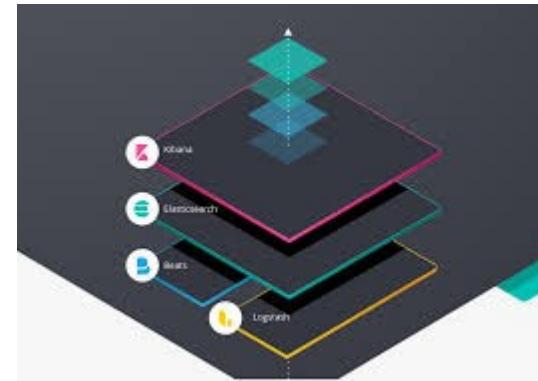
https://github.com/fenago/elasticsearch/blob/master/snippets/2_1.txt



Defining the mappings for the type of product

- Each text datatype field is mapped as follows:

```
"field_name": {  
    "type": "text",  
    "fields": {  
        "keyword": {  
            "type": "keyword",  
            "ignore_above": 256  
        }  
    }  
}
```



Inverted indexes

- An inverted index is the core data structure of Elasticsearch and any other system supporting full-text search.
- An inverted index is similar to the index that you see at the end of any course.
- It maps the terms that appear in the documents to the documents.

Inverted indexes

- For example, you may build an inverted index from the following strings:

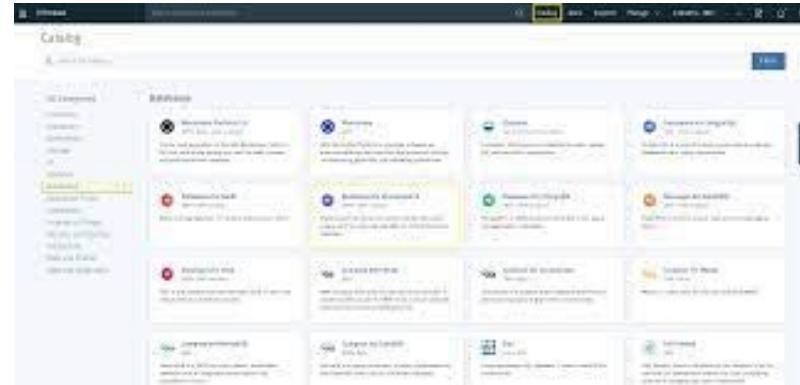
Document ID	Document
1	It is Sunday tomorrow.
2	Sunday is the last day of the week.
3	The choice is yours.

Term	Frequency	Documents (postings)
choice	1	3
day	1	2
is	3	1, 2, 3
it	1	1
last	1	2
of	1	2
sunday	2	1, 2
the	3	2, 3
tomorrow	1	1
week	1	2
yours	1	3

CRUD operations

To understand how to perform CRUD operations, we will cover the following APIs. These APIs fall under the category of document APIs, which deal with documents:

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



Index API

- In Elasticsearch terminology, adding (or creating) a document to a type within an index of Elasticsearch is called an indexing operation.
- There are two ways we can index a document:
 1. Indexing a document by providing an ID
 2. Indexing a document without providing an ID

Indexing a document by providing an ID

- The format of this request is PUT /<index>/<type>/<id>, with the JSON document as the body of the request:

```
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 providing an ID

- The format of this request is POST /<index>/<type>, with the JSON document as the body of the request:

POST /catalog/_doc

{

```
"sku": "SP000003",
"title": "Mastering Elasticsearch",
"description": "Mastering Elasticsearch",
"author": "Bharvi Dixit",
"price": 54.99
```

}

- The ID, in this case, will be generated by Elasticsearch. It is a hash string, as highlighted in the response:

```
{  
  "_index": "catalog",  
  "_type": "_doc",  
  "_id": "1ZFMpmoBa_wgE5i2FfWV",  
  "_version": 1,  
  "result": "created",  
  "_shards": {  
    "total": 2,  
    "successful": 1,  
    "failed": 0  
  },  
  "_seq_no": 4,  
  "_primary_term": 1  
}
```

Get API

- The get API is useful for retrieving a document when you already know the ID of the document.
- It is essentially a get by primary key operation, as follows:

GET /catalog/_doc/1ZFMpmoBa_wgE5i2FfWV

- The format of this request is GET /<index>/<type>/<id>.
- The response would be as expected:

```
{  
  "_index": "catalog",  
  "_type": "_doc",  
  "_id": "1ZFMpmoBa_wgE5i2FfWV",  
  "_version": 1,  
  "_seq_no": 4,  
  "_primary_term": 1,  
  "found": true,  
  "_source": {  
    "sku": "SP000003",  
    "title": "Mastering Elasticsearch",  
    "description": "Mastering Elasticsearch",  
    "author": "Bharvi Dixit",  
    "price": 54.99  
  }  
}
```

Update API

- The update API is useful for updating the existing document by ID.
- The format of an update request is POST <index>/<type>/<id>/_update, with a JSON request as the body:

```
POST /catalog/_update/1
```

```
{  
  "doc": {  
    "price": "28.99"  
  }  
}
```

- The response of the update request is as follows:

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

- The following example uses `doc_as_upsert` to merge into the document with an ID of 3 or insert a new document if it doesn't exist:

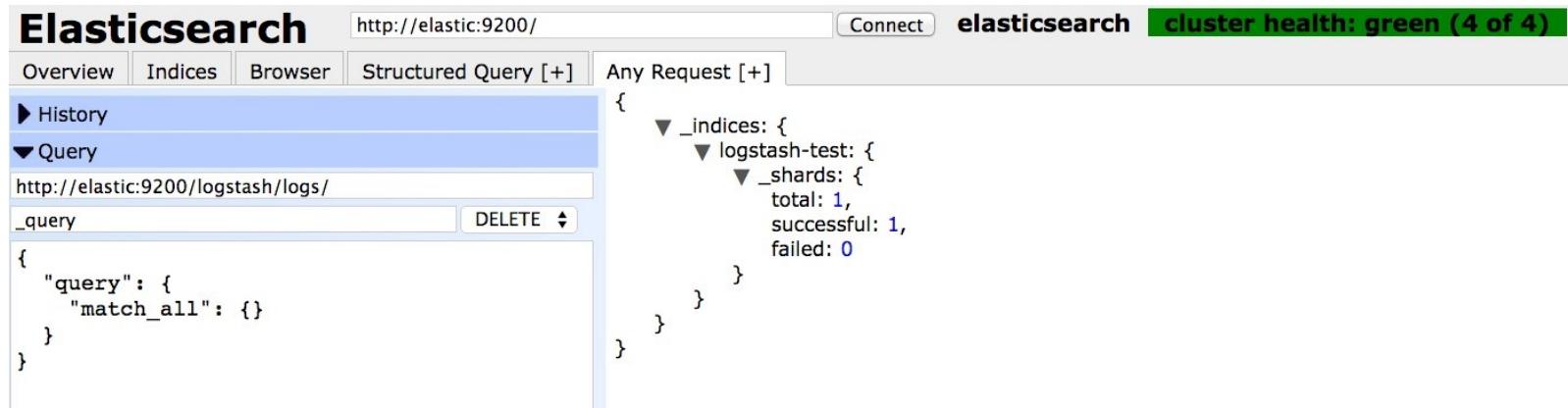
```
POST /catalog/_update/3
{
  "doc": {
    "author": "Albert Paro",
    "title": "Elasticsearch 5.0 Cookbook",
    "description": "Elasticsearch 5.0 Cookbook Third Edition",
    "price": "54.99"
  },
  "doc_as_upsert": true
}
```

- We can update the value of a field based on the existing value of that field or another field in the document.
- The following update uses an inline script to increase the price by two for a specific product:

```
POST /catalog/_update/1ZFMpmoBa_wgE5i2FfWV
{
  "script": {
    "source": "ctx._source.price += params.increment",
    "lang": "painless",
    "params": {
      "increment": 2
    }
  }
}
```

Delete API

- The delete API lets you delete a document by ID:
DELETE /catalog/_doc/1ZFMpmoBa_wgE5i2FfWV



The screenshot shows the Elasticsearch Query interface. The URL is `http://elastic:9200/`. The left sidebar has tabs for Overview, Indices, Browser, and Structured Query [+]. The main area shows a query history entry for `http://elastic:9200/logstash/logs/`. Below it, a query is being constructed in a structured query editor:

```
{  
  "query": {  
    "match_all": {}  
  }  
}
```

To the right, the results of a previous request are displayed:

```
{  
  "_indices": {  
    "logstash-test": {  
      "_shards": {  
        "total: 1,  
        "successful: 1,  
        "failed: 0  
      }  
    }  
  }  
}
```

The status bar at the bottom right indicates "cluster health: green (4 of 4)".

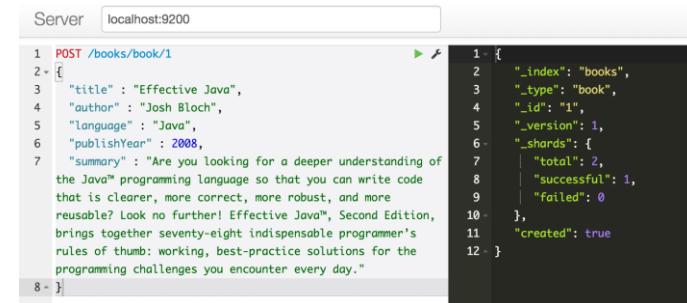
- The response of the delete operation is as follows:

```
{  
  "_index": "catalog",  
  "_type": "_doc",  
  "_id": "1ZFMpmoBa_wgE5i2FfWV",  
  "_version": 4,  
  "result": "deleted",  
  "_shards": {  
    "total": 2,  
    "successful": 1,  
    "failed": 0  
  },  
  "_seq_no": 9,  
  "_primary_term": 1  
}
```

Creating indexes and taking control of mapping

Usually, you wouldn't want to let things happen automatically, as you would want to control how indexes are created and also how mapping is created. We will see how you can take control of this process in this section, and we will look at the following:

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



```
POST /books/book/1
{
  "title": "Effective Java",
  "author": "Josh Bloch",
  "language": "Java",
  "publishYear": 2008,
  "summary": "Are you looking for a deeper understanding of the Java programming language so that you can write code that is clearer, more correct, more robust, and more reusable? Look no further! Effective Java", Second Edition, brings together seventy-eight indispensable programmer's rules of thumb: working, best-practice solutions for the programming challenges you encounter every day."
}
{
  "_index": "books",
  "_type": "book",
  "_id": "1",
  "_version": 1,
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  },
  "created": true
}
```

Creating an index

- You can create an index and specify the number of shards and replicas to create:

```
PUT /catalog
```

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

Creating an index

- It is possible to specify a mapping for a type at the time of index creation.
- The following command will create an index called catalog, with five shards and two replicas.
- Additionally, it also defines a type called my_type with two fields, one of the text type and another of the keyword type:

https://github.com/fenago/elasticsearch/blob/master/snippets/2_2.txt

Creating type mapping in an existing index

- A type can be added within an index after the index is created using the following code.
- The mappings for the type can be specified as follows:

```
PUT /catalog/_mapping
```

```
{
```

```
  "properties": {
```

```
    "name": {
```

```
      "type": "text"
```

```
    }
```

```
}
```

```
}
```

Creating type mapping in an existing index

- Let's add a couple of documents after creating the new type:

```
POST /catalog/_doc
```

```
{  
  "name": "books"  
}
```

```
POST /catalog/_doc
```

```
{  
  "name": "phones"  
}
```

Creating type mapping in an existing index

- Elasticsearch will assign a type automatically based on the value that you insert for the new field.
- It only takes into consideration the first value that it sees to guess the type of that field:

```
POST /catalog/_doc
```

```
{
```

```
  "name": "music",
```

```
  "description": "On-demand streaming music"
```

```
}
```

Creating type mapping in an existing index

- When the new document is indexed with fields, the field is assigned a datatype based on its value in the initial document.
- Let's look at the mapping after this document is indexed:

https://github.com/fenago/elasticsearch/blob/master/snippets/2_3.txt

Updating a mapping

- Let's add a code field, which is of the keyword type, but with no analysis:

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

Updating a mapping

- This mapping is merged into the existing mappings of the `_doc` type.
- The mapping looks like the following after it is merged:

https://github.com/fenago/elasticsearch/blob/master/snippets/2_4.txt

Updating a mapping

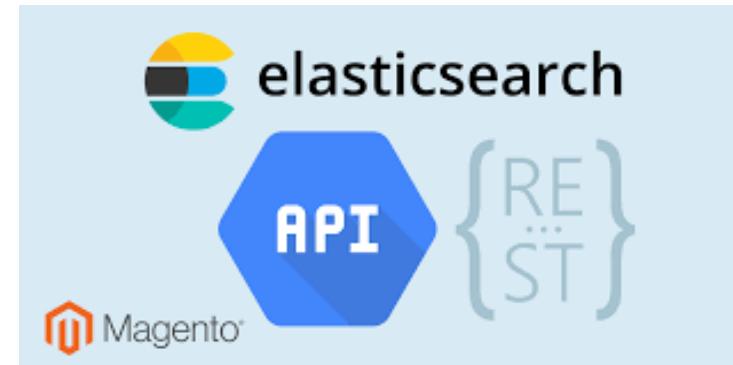
- Any subsequent documents that are indexed with the code field are assigned the right datatype:

```
POST /catalog/_doc
{
  "name": "sports",
  "code": "C004",
  "description": "Sports equipment"
}
```

REST API overview

The APIs that deal with Elasticsearch are categorized into the following types of APIs:

- Document APIs
- Search APIs
- Aggregation APIs
- Indexes APIs
- Cluster APIs
- cat APIs



Common API conventions

All Elasticsearch REST APIs share some common features. They can be used across almost all APIs. In this section, we will cover the following features:

- Formatting the JSON response
- Dealing with multiple indexes

Formatting the JSON response

- By default, the response of all the requests is not formatted.
- It returns an unformatted JSON string in a single line:

```
curl -XGET http://localhost:9200/catalog/_doc/1
```

Formatting the JSON response

- The following response is not formatted:

```
{"_index":"catalog","_type":"product","_id":"1","_version":3,"fo  
und":true,"_source":{  
    "sku": "SP000001",  
    "title": "Elasticsearch for Hadoop",  
    "description": "Elasticsearch for Hadoop",  
    "author": "Vishal Shukla",  
    "ISBN": "1785288997",  
    "price": 26.99  
}}
```

- Passing pretty=true formats the response:

```
curl -XGET http://localhost:9200/catalog/_doc/1?pretty=true
```

```
{  
  "_index" : "catalog",  
  "_type" : "product",  
  "_id" : "1",  
  "_version" : 3,  
  "found" : true,  
  "_source" : {  
    "sku" : "SP000001",  
    "title" : "Elasticsearch for Hadoop",  
    "description" : "Elasticsearch for Hadoop",  
    "author" : "Vishal Shukla",  
    "ISBN" : "1785288997",  
    "price" : 26.99  
  }  
}
```

Dealing with multiple indexes

We cover the following scenarios when dealing with multiple indexes within a cluster:

- Searching all documents in all indexes
- Searching all documents in one index
- Searching all documents of one type in an index
- Searching all documents in multiple indexes
- Searching all documents of a particular type in all indexes



Dealing with multiple indexes

- This will return all the documents from all the indexes of the cluster.
- The response looks like the following, and it is truncated to remove the unnecessary repetition of documents:

https://github.com/fenago/elasticsearch/blob/master/snippets/2_5.txt

Searching all documents in one index

- The following code will search for all documents, but only within the catalog index:

GET /catalog/_search

- You can also be more specific and include the type in addition to the index name, like so:

GET /catalog/_doc/_search

Searching all documents in multiple indexes

- The following will search for all the documents within the catalog index and an index named my_index:

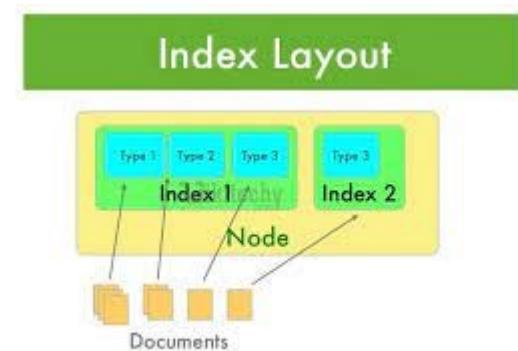
`GET /catalog,my_index/_search`



Searching all the documents of a particular type in all indexes

- The following will search all the indexes in the cluster, but only documents of the product type will be searched:

GET /_all/_doc/_search



Summary

- In this lesson, we learned about the essential Kibana Console UI and curl commands that we can use to interact with Elasticsearch with the REST API.
- Then, we looked at the core concepts of Elasticsearch.
- We performed customary CRUD operations, which are required as support for any data store.
- We took a closer look at how to create indexes, and how to create and manage mappings.



COMPLETE LAB 2

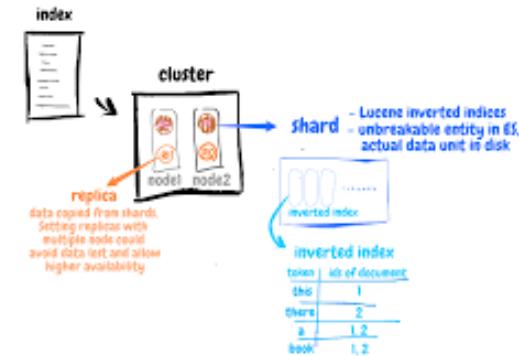
3. Searching - What is Relevant



Searching - What is Relevant

We will cover the following topics in this lesson:

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



The basics of text analysis

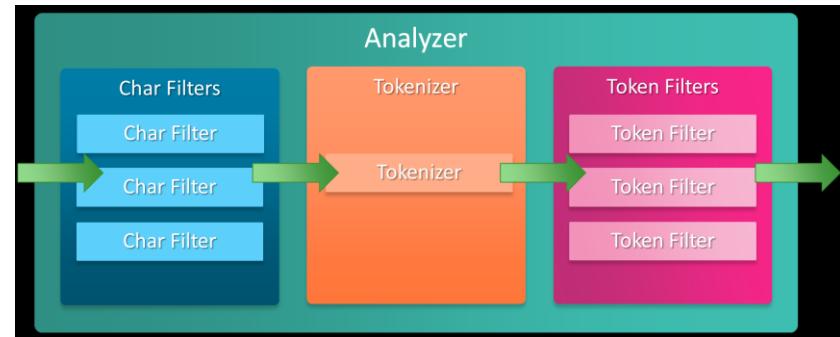
In the following sections, we will cover the following topics:

- Understanding Elasticsearch analyzers
- Using built-in analyzers
- Implementing autocomplete with a custom analyzer

Understanding Elasticsearch analyzers

The analyzer performs this process of breaking up input character streams into terms. This happens twice:

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



Understanding Elasticsearch analyzers

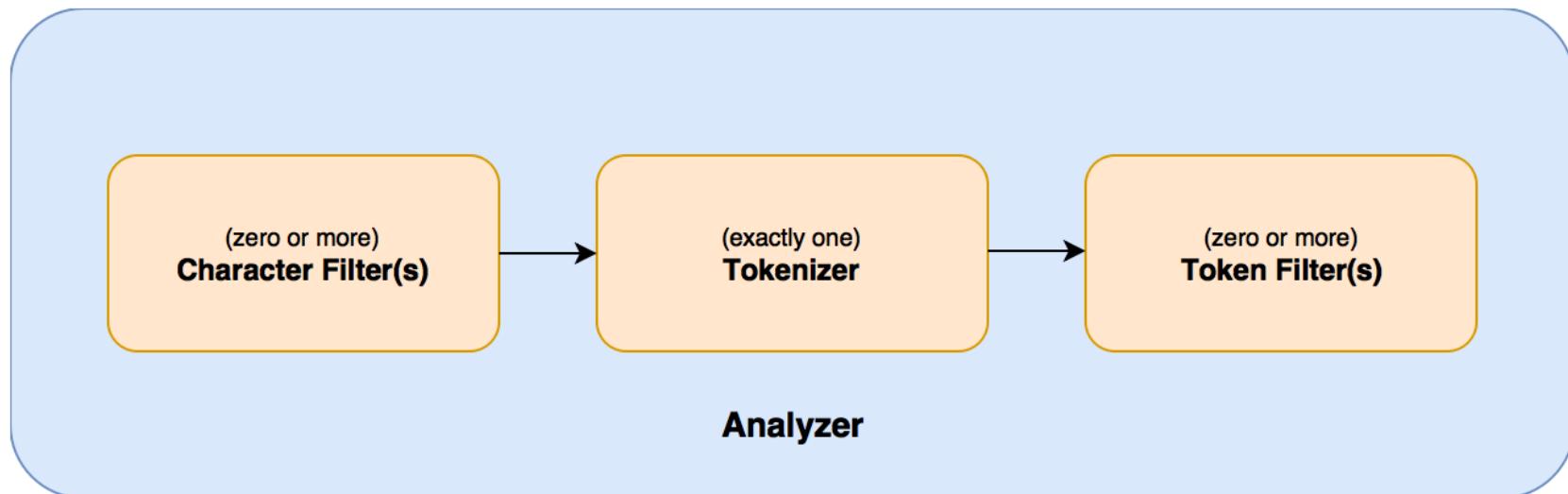
Elasticsearch uses analyzers to analyze text data. An analyzer has the following components:

- Character filters: Zero or more
- Tokenizer: Exactly one
- Token filters: Zero or more



Understanding Elasticsearch analyzers

- The following diagram depicts the components of an analyzer:



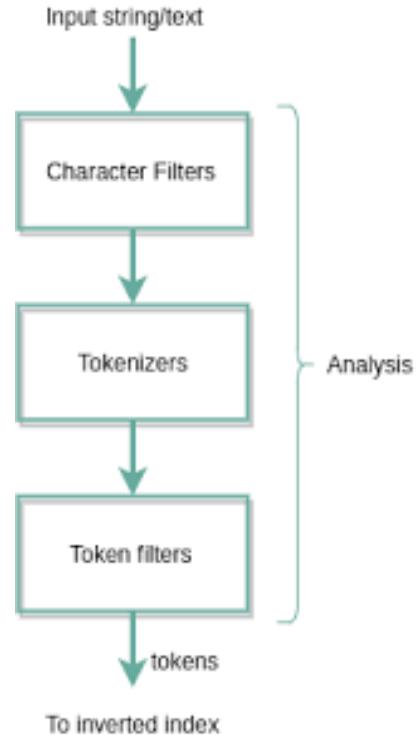
Character filters

- When composing an analyzer, we can configure zero or more-character filters.
- A character filter works on a stream of characters from the input field; each character filter can add, remove, or change the characters in the input field.

Character filters

For example, you may want to transform emoticons into some text that represents those emoticons:

- :) should be translated to _smile_
- :(should be translated to _sad_
- :D should be translated to _laugh_



Character filters

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

Tokenizer

- An analyzer has exactly one tokenizer, The responsibility of a tokenizer is to receive a stream of characters and generate a stream of tokens & These tokens are used to build an inverted index.
- A token is roughly equivalent to a word & In addition to breaking down characters into words or tokens, it also produces, in its output, the start and end offset of each token in the input stream.

Standard tokenizer

- Loosely speaking, the standard tokenizer breaks down a stream of characters by separating them with whitespace characters and punctuation.
- The following example shows how the standard tokenizer breaks a character stream into tokens:

```
POST _analyze
```

```
{
```

```
  "tokenizer": "standard",
```

```
  "text": "Tokenizer breaks characters into tokens!"
```

```
}
```

Standard tokenizer

- The preceding command produces the following output; notice the start_offset, end_offset, and positions in the output:

https://github.com/fenago/elasticsearch/blob/master/snippets/3_1.txt

Token filters

- There can be zero or more token filters in an analyzer.
- Every token filter can add, remove, or change tokens in the input token stream that it receives.
- Since it is possible to have multiple token filters in an analyzer, the output of each token filter is sent to the next one until all token filters are considered.
- Elasticsearch comes with several token filters, and they can be used to compose your own custom analyzers.

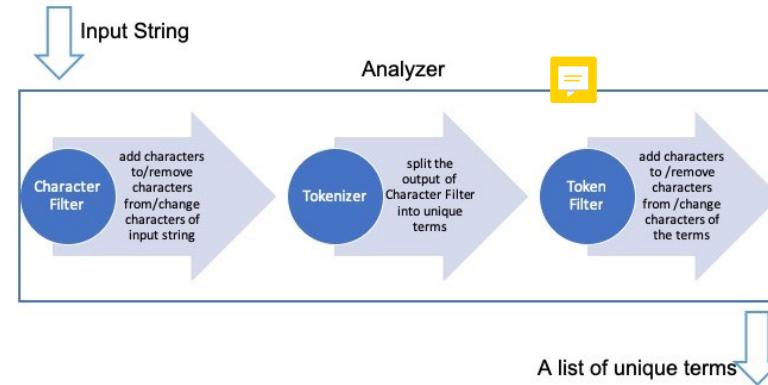
Using built-in analyzers

- Elasticsearch comes with several built-in analyzers that can be used directly.
- Almost all these analyzers work without any need for additional configuration, but they provide the flexibility of configuring some parameters.

Standard analyzer

Standard Analyzer is suitable for many languages and situations. It can also be customized for the underlying language or situation. Standard analyzer comprises of the following components:

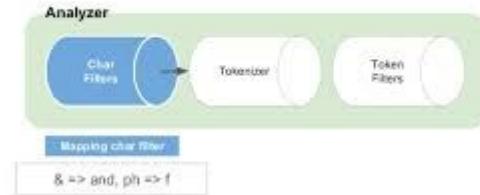
- Tokenizer
- Token filters



Standard analyzer

- Let's see how Standard analyzer works by default with an example:

https://github.com/fenago/elasticsearch/blob/master/snippets/3_2.txt



Standard analyzer

- Let's check how Elasticsearch will do the analysis for the my_text field whenever any document is indexed in this index.
- We can do this test using the _analyze API, as we saw earlier:

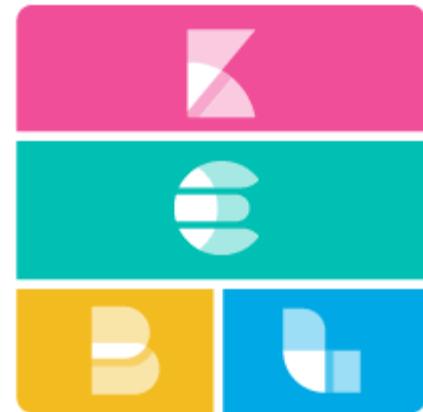
POST index_standard_analyzer/_analyze

```
{  
  "field": "my_text",  
  "text": "The Standard Analyzer works this way."  
}
```

Standard analyzer

- The output of previous command shows the following tokens:

https://github.com/fenago/elasticsearch/blob/master/snippets/3_3.txt



Standard analyzer

- Even though the Standard Analyzer has a stop token filter, none of the tokens are filtered out. Therefore the `_analyze` output has all words as tokens.
- Let's create another index that uses English language stop words:

https://github.com/fenago/elasticsearch/blob/master/snippets/3_4.txt

Standard analyzer

- When you try the `_analyze` API on the new index, you will see that it removes the stopwords, such as the and this:

POST

```
index_standard_analyzer_english_stopwords/_analyze
{
  "field": "my_text",
  "text": "The Standard Analyzer works this way."
}
```

Standard analyzer

- Previous code returns a response like the following:

https://github.com/fenago/elasticsearch/blob/master/snippets/3_5.txt



elastic

Implementing autocomplete with a custom analyzer

- In certain situations, you may want to create your own custom analyzer by composing character filters, tokenizers, and token filters of your choice.
- Please remember that most requirements can be fulfilled by one of the built-in analyzers with some configuration.
- Let's create an analyzer that can help when implementing autocomplete functionality.

Implementing autocomplete with a custom analyzer

- If we were to use Standard Analyzer at indexing time, the following terms would be generated for the field with the Learning Elastic Stack 7 value:

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

Implementing autocomplete with a custom analyzer

- For example, if the user has typed elas, it should still recommend Learning Elastic Stack 7 as a product.
- Let's compose an analyzer that can generate terms such as el, ela, elas, elast, elasti, elastic, le, lea, and so on:

https://github.com/fenago/elasticsearch/blob/master/snippets/3_6.txt

Implementing autocomplete with a custom analyzer

- Given that the following two products are indexed, and the user has typed Ela so far, the search should return both products:

https://github.com/fenago/elasticsearch/blob/master/snippets/3_7.txt

Implementing autocomplete with a custom analyzer

- We are going to use product catalog data taken from the popular e-commerce site www.amazon.com.
- The data is downloadable from <http://dbs.uni-leipzig.de/file/Amazon-GoogleProducts.zip>.
- Before we start with the queries, let's create the required index and import some data:

https://github.com/fenago/elasticsearch/blob/master/snippets/3_8.txt

Implementing autocomplete with a custom analyzer

- After you have imported the data, verify that it is imported

with the following query:

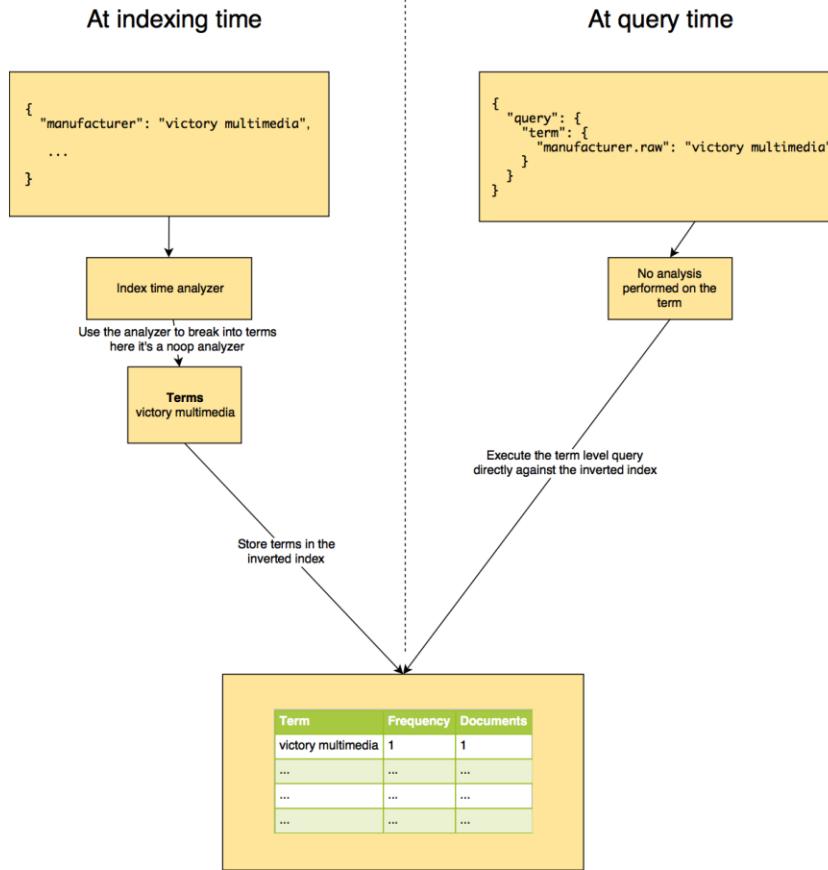
GET /amazon_products/_search

```
{  
  "query": {  
    "match_all": {}  
  }  
}
```

Searching from structured data

- In certain situations, we may want to find out whether a given document should be included or not; that is, a simple binary answer.
- On the other hand, there are other types of queries that are relevance-based.
- Such relevance-based queries also return a score against each document to say how well that document fits the query.

Term level query flow

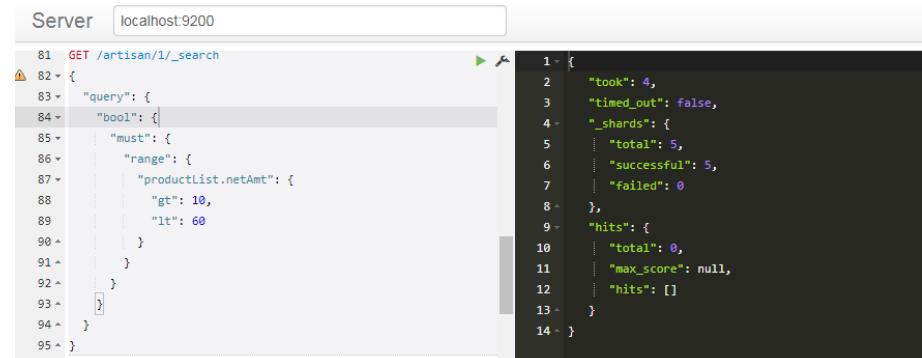


Range query

Range queries can be applied to fields with datatypes that have natural ordering. We will look at how to apply range queries in the following ways:

- On numeric types
- With score boosting
- On dates

Let's look at the most typical range query on a numeric field.



The screenshot shows a browser developer tools Network tab with a request to `localhost:9200/_search`. The request body contains a JSON search query with a range filter on the `productList.netAmt` field, specifying values greater than 10 and less than 60. The response body is a JSON object containing the search results, including the total number of hits (5), successful operations (5), failed operations (0), and the list of hits.

```
81 GET /artisan/_search
82 {
83   "query": {
84     "bool": {
85       "must": [
86         {
87           "range": {
88             "productList.netAmt": {
89               "gt": 10,
90               "lt": 60
91             }
92           }
93         }
94       ]
95     }
96   }
97 }
```

```
1 {
2   "took": 4,
3   "timed_out": false,
4   "_shards": {
5     "total": 5,
6     "successful": 5,
7     "failed": 0
8   },
9   "hits": {
10    "total": 0,
11    "max_score": null,
12    "hits": []
13  }
14 }
```

- Suppose we are storing products with their prices in an Elasticsearch index and we want to get all products within a range.
- The following is the query to get products in the range of \$10 to \$20:

```
GET /amazon_products/_search
```

```
{  
  "query": {  
    "range": {  
      "price": {  
        "gte": 10,  
        "lte": 20  
      }  
    }  
  }  
}
```

Range query on
numeric types

Range query

- The response of previous query looks like the following:

https://github.com/fenago/elasticsearch/blob/master/snippets/3_9.txt

- The range query allows you to provide a boost parameter to enhance its score relative to other query/queries that it is combined with:

GET /amazon_products/_search

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

Range query with
score boosting

Range query on dates

- A range query can also be applied to date fields since dates are also inherently ordered. You can specify the date format while querying a date range:

GET /orders/_search

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

Range query on dates

- Elasticsearch allows us to use dates with or without the time in its queries.
- It also supports the use of special terms, including nowto denote the current time.
- For example, the following query queries data from the last 7 days up until now, that is, data from exactly 24 x 7 hours ago till now with a precision of milliseconds:

GET /orders/_search

```
{"query": {"range": {"orderDate": {"gte": "now-7d", "lte": "now"}}}}
```

Exists query

- Sometimes it is useful to obtain only records that have non-null and non-empty values in a certain field.
- For example, getting all products that have description fields defined:

```
GET /amazon_products/_search
```

```
{  
  "query": {  
    "exists": {  
      "field": "description"  
    }  
  }  
}
```

Term query

- When we defined the manufacturer field, we stored it as both text and keyword fields.
- When doing an exact match, we have to use the field with the keyword type:

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

- The response looks like the following (only the partial response is included):

```
{  
...  
"hits": {  
  "total" : {  
    "value" : 3,  
    "relation" : "eq"  
  },  
  "max_score": 5.965414,  
  "hits": [  
    {  
      "_index": "amazon_products",  
      "_type": "products",  
      "_id": "AV5rBfPNNI_2eZGciHC",  
      "_score": 5.965414,  
      ...  
    }  
  ]  
}
```

Term query

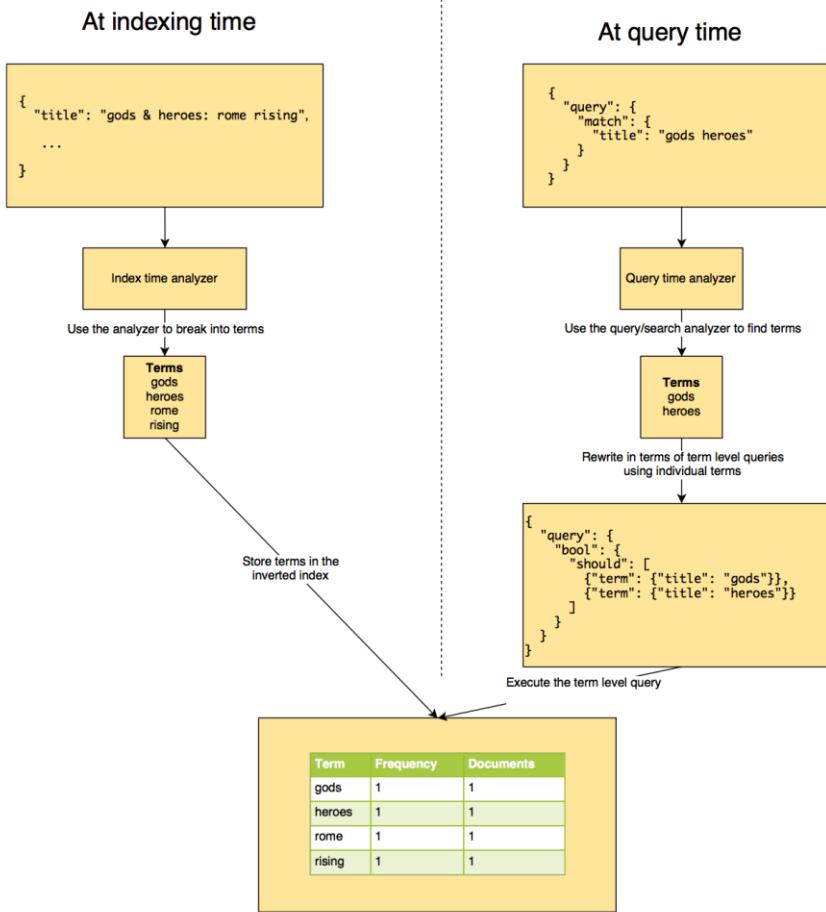
- It needs to be wrapped inside a constant_score filter:

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

Searching from the full text

- Full-text queries can work on unstructured text fields, These queries are aware of the analysis process.
- Full-text queries apply the analyzer on the search terms before performing the actual search operation.
- That determines the right analyzer to be applied by first checking whether a field-level search_analyzer is defined, and then by checking whether a field-level analyzer is defined.

High level query flow



Match query

- A match query is the default query for most full-text search requirements.
- It is one of the high-level queries that is aware of the analyzer used for the underlying field.
- Let's get an understanding of what this means under the hood.

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

Match query

- In fact, in this particular case, the match query gets converted into a term query, such as the following:

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

Match query

- Let's see what happens if you execute a match query against a text field, which is a real use case for a full-text query:

```
GET /amazon_products/_search
```

```
{  
  "query": {  
    "match": {  
      "manufacturer": "victory multimedia"  
    }  
  }  
}
```

Operator

- By default, if the search term specified results in multiple terms after applying the analyzer, we need a way to combine the results from individual terms.
- As we saw in the preceding example, the default behavior of the match query is to combine the results using the or operator, that is, one of the terms has to be present in the document's field.

Operator

- It can be changed to use the and operator using the following query:

https://github.com/fenago/elasticsearch/blob/master/snippets/3_10.txt

Minimum should match

- Instead of applying the and operator, we can keep the or operator and specify at least how many terms should match in each document for it to be included in the result.
- This allows for finer-grained control:

https://github.com/fenago/elasticsearch/blob/master/snippets/3_11.txt

Fuzziness

- With the fuzziness parameter, we can turn thematch query into a fuzzy query.
- This fuzziness is based on the Levenshtein edit distance, to turn one term into another by making several edits to the original text.
- Edits can be insertions, deletions, substitutions, or the transposition of characters in the original term.

Fuzziness

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

- If we wanted to still allow more room for errors to be correctable, the fuzziness should be increased to 2.
- For example, a fuzziness of 2 will even match victer. Victory is two edits away from victer:

```
GET /amazon_products/_search
```

```
{  
  "query": {  
    "match": {  
      "manufacturer": {  
        "query": "victer multimedia",  
        "fuzziness": 2  
      }  
    }  
  }  
}
```

Match phrase query

- When you want to match a sequence of words, as opposed to separate terms in a document, the `match_phrase` query can be useful.
- For example, the following text is present as part of the description for one of the products:

real video saltware aquarium on your desktop!

- The match query can include all those documents that have any of the terms, even when they are out of order within the document:

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

Match phrase query

- The response will look like the following:

https://github.com/fenago/elasticsearch/blob/master/snippets/3_12.txt

- For example, a slop value of 1 would allow one missing word in the search text but would still match the document:

```
GET /amazon_products/_search
{
  "query": {
    "match_phrase": {
      "description": {
        "query": "real video aquarium",
        "slop": 1
      }
    }
  }
}
```

Multi match query

The multi_match query can be used with different options.
We will look at the following options:

- Querying multiple fields with defaults
- Boosting one or more fields
- With types of multi_match queries
- Let's look at each option, one by one.

Querying multiple fields with defaults

- We want to provide a product search functionality in our web application.
- When the end user searches for some terms, we want to query both the title and description fields.
- This can be done using the multi_match query.

Querying multiple fields with defaults

- The following query will find all of the documents that have the terms monitor or aquarium in the title or the description fields:

```
GET /amazon_products/_search
```

```
{  
  "query": {  
    "multi_match": {  
      "query": "monitor aquarium",  
      "fields": ["title", "description"]  
    }  
  }  
}
```

Boosting one or more fields

- In an e-commerce type of web application, the user intends to search for an item, and they might search for some keywords.
- What if we want the title field to be more important than the description? If one or more of the search terms appears in the title, it is definitely a more relevant product than the ones that have those values only in the description.

Boosting one or more fields

- Let's make the title field three times more important than the description field.
- This can be done by using the following syntax:

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

Writing compound queries

- This class of queries can be used to combine one or more queries to come up with a more complex query.
- Some compound queries convert scoring queries into non-scoring queries and combine multiple scoring and non-scoring queries.
- We will look at the following compound queries:
 1. Constant score query
 2. Bool query

Constant score query

- Elasticsearch supports querying both structured data and full text.
- While full-text queries need scoring mechanisms to find the best matching documents, structured searches don't need scoring.
- The constant score query allows us to convert a scoring query that normally runs in a query context to a non-scoring filter context.

- For example, a term query is normally run in a query context.
- This means that when Elasticsearch executes a term query, it not only filters documents but also scores all of them:

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

- The response contains the score for every document. Please see the following partial response:

```
{  
...,  
"hits": {  
  "total": 3,  
  "max_score": 5.966147,  
  "hits": [  
    {  
      "_index": "amazon_products",  
      "_type": "products",  
      "_id": "AV5rBfasNI_2eZGcilbg",  
      "_score": 5.966147,  
      "_source": {  
        "price": "19.95",  
...  
    }  
  ]  
}
```

- The original query can be converted to run in a filter context using the following constant_score query:

```
GET /amazon_products/_search
```

```
{  
  "query": {  
    "constant_score": {  
      "filter": {  
        "term": {  
          "manufacturer.raw": "victory multimedia"  
        }  
      }  
    }  
  }  
}
```

- It assigns a neutral score of 1 to each document by default.
- Please note the partial response in the following code:

```
{  
...,  
"hits": {  
  "total": 3,  
  "max_score": 1,  
  "hits": [  
    {  
      "_index": "amazon_products",  
      "_type": "products",  
      "_id": "AV5rBfasNI_2eZGcilbg",  
      "_score": 1,  
      "_source": {  
        "price": "19.95",  
        "description": ...  
      }  
    }  
...  
}
```

- It is possible to specify a boost parameter, which will assign that score instead of the neutral score of 1:

```
GET /amazon_products/_search
{
  "query": {
    "constant_score": {
      "filter": {
        "term": {
          "manufacturer.raw": "victory multimedia"
        }
      },
      "boost": 1.2
    }
  }
}
```

Bool query

- The bool query in Elasticsearch is your Swiss Army knife.
- It can help you write many types of complex queries.
- If you are come from an SQL background, you already know how to filter based on multiple AND and OR conditions in the WHERE clause.
- The bool query allows you to combine multiple scoring and non-scoring queries.

Bool query

- A bool query has the following sections:

```
GET /amazon_products/_search
{
  "query": {
    "bool": {
      "must": [...],      scoring queries executed in query context
      "should": [...],   scoring queries executed in query context
      "filter": {},      non-scoring queries executed in filter context
      "must_not": [...]  non-scoring queries executed in filter context
    }
  }
}
```

Combining OR conditions

- To find all of the products in the price range 10 to 13, OR manufactured by valuesoft:

https://github.com/fenago/elasticsearch/blob/master/snippets/3_13.txt

Combining AND and OR conditions

- Find all products in the price range 10 to 13, AND manufactured by valuesoft or pinnacle:

https://github.com/fenago/elasticsearch/blob/master/snippets/3_14.txt

Adding NOT conditions

- It is possible to add NOT conditions, that is, specifically filtering out certain clauses using the `must_not` clause in the `bool` filter.
- For example, find all of the products in the price range 10 to 20, but they must not be manufactured by encore.
- The following query will do just that:

https://github.com/fenago/elasticsearch/blob/master/snippets/3_15.txt

- The bool query with the must_not element is useful for negate any query.
- To negate or apply a NOT filter to the query, it should be wrapped inside the bool with must_not, as follows:

```
GET /amazon_products/_search
{
  "query": {
    "bool": {
      "must_not": {
        .... original query to be negated ...
      }
    }
  }
}
```

Modeling relationships

- At the same time, products in the Automobile GPS Systems category may have features such as screen size, whether GPS can speak street names, or whether it has free lifetime map updates available.
- Because we may have tens of thousands of products in hundreds of product categories, we may have tens of thousands of features.
- One solution might be to create one field for each feature.

Title	Category	Screen Size	Processor Type	Clock Speed	Speaks Street Names	Map Updates
ThinkPad X1	Laptops	14 inches	core i5	3 GHz		
Acer Predator	Laptops	6 inches	core i7	6 GHz		
Trucker 600	GPS Navigation Systems	6 inches			Yes	Yes
RV Tablet 70	GPS Navigation Systems	7 inches			Yes	Yes

Modeling relationships

- The product table would be modeled as follows:

ProductID	Title	Category	Description	Other Product Columns...
c0001	ThinkPad X1	Laptops
c0002	Acer Predator	Laptops
c0003	Trucker 600	GPS Navigation Systems
c0004	RV Tablet 70	GPS Navigation Systems

Modeling relationships

- The features would be modeled as a separate table, where the ProductID and Feature may be a composite primary key:

ProductID	Feature	FeatureValue
c001	Screen Size	14 inches
c001	Processor Type	core i5
c001	Clock Speed	3 GHz
c002	Screen Size	6 inches
...

- The join datatype mapping that establishes the relationship is defined as follows:

```
PUT /amazon_products_with_features
```

```
{  
  ...  
  "mappings": {  
    "doc": {  
      "properties": {  
        ...  
        "product_or_feature": {  
          "type": "join",  
          "relations": {  
            "product": "feature"  
          }  
        },  
        ...  
      }  
    }  
  }  
}
```

Modeling relationships

- When indexing product records, we use the following syntax:

https://github.com/fenago/elasticsearch/blob/master/snippets/3_16.txt



- The value of product_or_feature, which is set to product suggests that this document is referring to a product.
- A feature record is indexed as follows:

PUT

amazon_products_with_features/doc/c0001_screen_size?routing=c0001

```
{  
  "product_or_feature": {  
    "name": "feature",  
    "parent": "c0001"  
  },  
  "feature_key": "screen_size",  
  "feature_value": "14 inches",  
  "feature": "Screen Size"
```

}

has_child query

- If you want to get products based on some condition on the features, you can use has_child queries.
- The outline of a has_child query is as follows:

```
GET <index>/_search
{
  "query": {
    "has_child": {
      "type": <the child type against which to run the following query>,
      "query": {
        <any elasticsearch query like term, bool, query_string to be run against the child
type>
      }
    }
  }
}
```

has_child query

- Let's learn about this through an example.
- We want to get all of the products where processor_series is Core i7 from the example dataset that we loaded:

https://github.com/fenago/elasticsearch/blob/master/snippets/3_17.txt

has_child query

- The result of this `has_child` query is that we get back all the products that satisfy the query mentioned under the `has_child` element executed against all the features.
- The response should look like the following:

https://github.com/fenago/elasticsearch/blob/master/snippets/3_18.txt

has_parent query

- We want to get all the features of a specific product that has the product id = c0003.
- We can use a has_parent query as follows:

https://github.com/fenago/elasticsearch/blob/master/snippets/3_19.txt

has_parent query

- The result is all the features of that product:

https://github.com/fenago/elasticsearch/blob/master/snippets/3_20.txt



parent_id query

- You guessed correctly; it is the parent_id query, where we obtain all children using the parent's ID:

```
GET /amazon_products_with_features/_search
```

```
{  
  "query": {  
    "parent_id": {  
      "type": "feature",  
      "id": "c0001"  
    }  
  }  
}
```



Summary

- In this lesson, we took a deep dive into the search capabilities of Elasticsearch.
- We looked at the role of analyzers and the anatomy of an analyzer, saw how to use some of the built-in analyzers that come with Elasticsearch, and saw how to create custom analyzers.
- Along with a solid background on analyzers, we learned about two main types of queries—term-level queries and full-text queries.

COMPLETE LAB 3

4. Analytics with Elasticsearch



Analytics with Elasticsearch

In this lesson, we will look at how Elasticsearch can serve as our analytics engine. We will cover the following topics:

- The basics of aggregations
- Preparing data for analysis
- Metric aggregations
- Bucket aggregations
- Pipeline aggregations



The basics of aggregations

- While learning about searching, we used the following API:

```
POST /<index_name>/_search
{
  "query": {
    ...
    ... type of query ...
  }
}
```

The basics of aggregations

- The aggregations, or aggs, element allows us to aggregate data.
- All aggregation requests take the following form:

```
POST /<index_name>/_search
{
  "aggs": {
    ... type of aggregation ...
  },
  "query": { ... type of query ... },           //optional query part
  "size": 0                                     //size typically set to 0
}
```

Bucket aggregations

- Bucket aggregations segment the data in question (defined by the query context) into various buckets that are identified by the buckets key.
- Bucket aggregation evaluates each document in the context by deciding which bucket it falls into.

Bucket aggregations

- For people who are coming from an SQL background, a query that has GROUP BY, such as the following query, does the following with bucket aggregations:

```
SELECT column1, count(*) FROM table1 GROUP BY  
column1;
```

Metric aggregations

- Metric aggregations work on numerical fields.
- They compute the aggregate value of a numerical field in the given context.
- For example, let's suppose that we have a table containing the results of a student's examination.
- Each record contains marks obtained by the student.

Metric aggregations

- In SQL terms, the following query gives a rough analogy of what a metric aggregation may do:

```
SELECT avg(score) FROM results;
```

Matrix aggregations

- Matrix aggregations were introduced with Elasticsearch version 5.0.
- Matrix aggregations work on multiple fields and compute matrices across all the documents within the query context.

The screenshot shows the Elasticsearch Dev Tools interface with the 'Console' tab selected. The request is a GET /nested_aggregation/_search with the following JSON body:

```
1 GET /nested_aggregation/_search
2 {
3   "aggs": {
4     "Nested_Aggregation": {
5       "nested": {
6         "path": "Employee"
7       },
8       "aggs": {
9         "Min_Salary": {
10          "min": {
11            "field": "Employee.salary"
12          }
13        }
14      }
15    }
16 }
```

The response shows a list of employees with their details and a summary aggregation:

```
42   {
43     "first" : "Maculum",
44     "last" : "Maculum",
45     "salary" : "58000"
46   },
47   {
48     "first" : "Vinod",
49     "last" : "Kambli",
50     "salary" : "63000"
51   },
52   {
53     "first" : "DJ",
54     "last" : "Bravo",
55     "salary" : "71000"
56   },
57   {
58     "first" : "Jaques",
59     "last" : "Kallis",
60     "salary" : "75000"
61   }
62 ]
63 ],
64 ],
65 ],
66 "aggregations": {
67   "Nested_Aggregation": {
68     "doc_count": 7,
69     "Min_Salary": {
70       "value": 58000.0
71     }
72   }
73 }
```

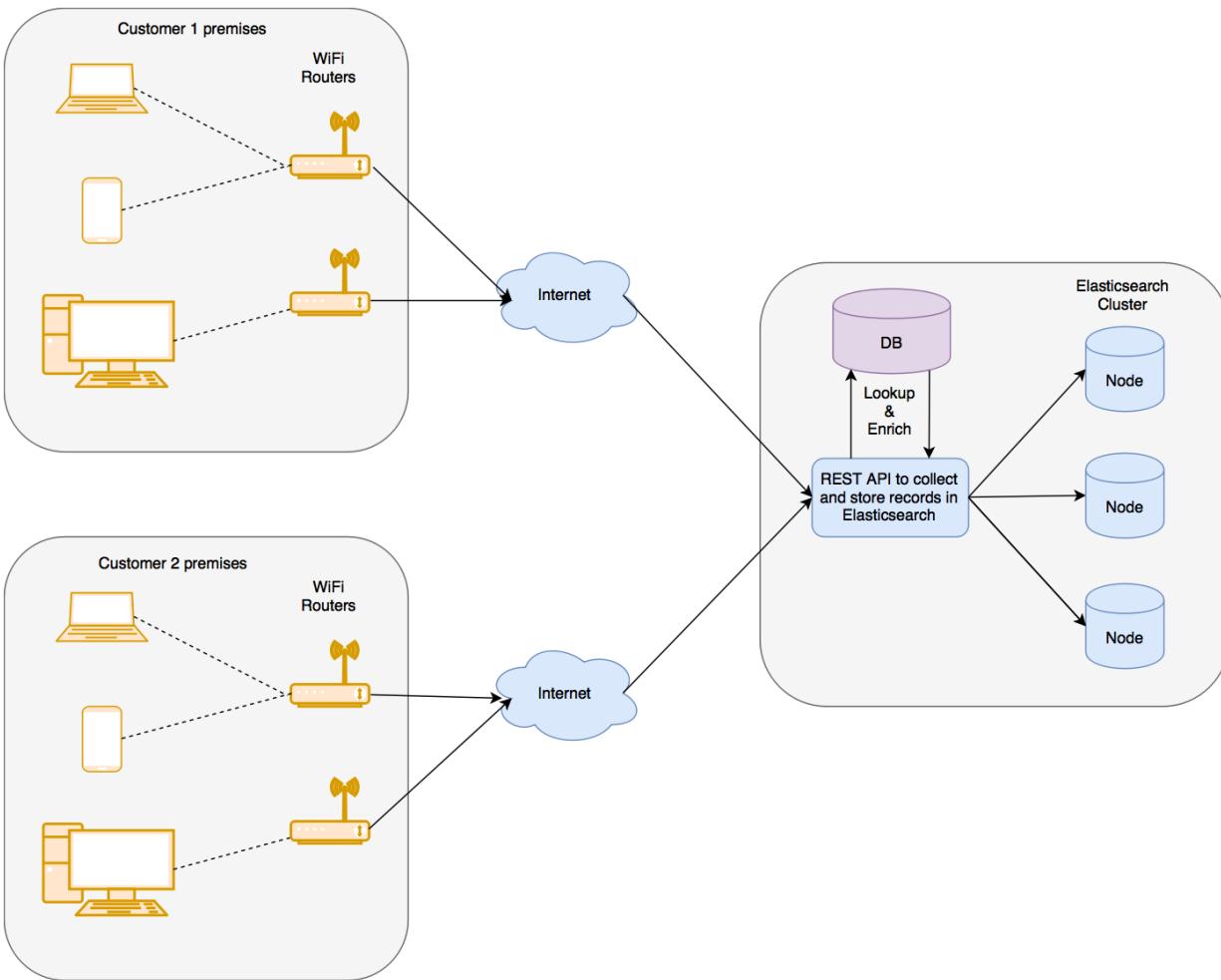
Pipeline aggregations

- Pipeline aggregations are higher order aggregations that can aggregate the output of other aggregations.
- These are useful for computing something, such as derivatives & We will look at some pipeline aggregations later in this lesson.
- This was an overview about the different types of aggregations supported by Elasticsearch at a high level.

Preparing data for analysis

We will cover the following topics while we prepare and load the data into the local Elasticsearch instance:

- Understanding the structure of the data
- Loading the data using Logstash



Understanding the structure of the data

- Finally, the enriched records are stored in Elasticsearch in a flat data structure.
- One record looks as follows:

https://github.com/fenago/elasticsearch/blob/master/snippets/4_1.txt

Loading the data using Logstash

- To import the data, please follow the instructions in this course's accompanying source code repository on GitHub, at <https://github.com/fenago/elasticsearch>. This can be found in the v7.0 branch.
- Please clone or download the repository from GitHub & The instructions for importing data are at the following path within the project.
- Once you have cloned the repository, check out the v7.0 branch.

Loading the data using Logstash

- Once you have imported the data, verify that your data has been imported with the following query:

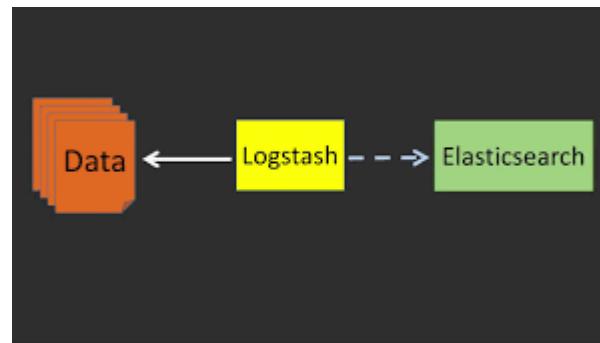
```
GET /bigginsight/_search
```

```
{  
  "query": {  
    "match_all": {}  
  },  
  "size": 1  
}
```

Loading the data using Logstash

- You should see a response similar to the following:

https://github.com/fenago/elasticsearch/blob/master/snippets/4_2.txt



Metric aggregations

In this section, we will go over the following metric aggregations:

- Sum, average, min, and max aggregations
- Stats and extended stats aggregations
- Cardinality aggregations

Sum, average, min, and max aggregations

- Finding the sum of a field, the minimum value for a field, the maximum value for a field, or an average, are very common operations.
- For people who are familiar with SQL, the query to find the sum is as follows:

```
SELECT sum(downloadTotal) FROM usageReport;
```

Sum aggregation

- Here is how to write a simple sum aggregation:

```
GET bigginsight/_search?track_total_hits=true
```

```
{  
  "aggregations": {  
    "download_sum": {  
      "sum": {  
        "field": "downloadTotal"  
      }  
    }  
  },  
  "size": 0  
}
```

- The response should look like the following:

```
{  
  "took": 92,  
  ...  
  "hits": {  
    "total" : {  
      "value" : 242836,          1  
      "relation" : "eq"  
    },          1  
    "max_score": 0,  
    "hits": []  
  },  
  "aggregations": {          2  
    "download_sum": {          3  
      "value": 2197438700       4  
    }  
  }  
}
```

- The average aggregation finds an average across all the documents in the querying context:

```
GET bigginsight/_search
{
  "aggregations": {
    "download_average": {
      "avg": {
        "field": "downloadTotal"
      }
    }
  },
  "size": 0
}
```

- The min aggregation is how we will find the minimum value of the downloadTotal field in the entire index/type:

```
GET bigginsight/_search
{
  "aggregations": {
    "download_min": {
      "min": {
        "field": "downloadTotal"
      }
    }
  },
  "size": 0
}
```

Max aggregation

- Here's how we will find the maximum value of the downloadTotal field in the entire index/type:

```
GET bigginsight/_search
{
  "aggregations": {
    "download_max": {
      "max": {
        "field": "downloadTotal"
      }
    }
  },
  "size": 0
}
```

Stats and extended stats aggregations

- These aggregations compute some common statistics in a single request, without having to issue multiple requests.
- This saves resources on the Elasticsearch side, as well, because the statistics are computed in a single pass, rather than being requested multiple times.
- The client code also becomes simpler if you are interested in more than one of these statistics.

- Stats aggregation computes the sum, average, min, max, and count of documents in a single pass:

```
GET bigginsight/_search
```

```
{  
  "aggregations": {  
    "download_stats": {  
      "stats": {  
        "field": "downloadTotal"  
      }  
    }  
  },  
  "size": 0  
}
```

- The response should look like the following:

```
{  
  "took": 4,  
  ...  
  "hits": {  
    "total" : {  
      "value" : 10000,  
      "relation" : "gte"  
    },  
    "max_score": 0,  
    "hits": []  
  },  
  "aggregations": {  
    "download_stats": {  
      "count": 242835,  
      "min": 0,  
      "max": 241213,  
      "avg": 9049.102065188297,  
      "sum": 2197438700  
    }  
  }  
}
```

- The extended stats aggregation returns a few more statistics in addition to the ones returned by the stats aggregation:

```
GET bigginsight/_search
```

```
{  
  "aggregations": {  
    "download_estats": {  
      "extended_stats": {  
        "field": "downloadTotal"  
      }  
    }  
  },  
  "size": 0  
}
```

Extended stats aggregation

- The response looks like the following:

https://github.com/fenago/elasticsearch/blob/master/snippets/4_3.txt



Cardinality aggregation

- Finding the count of unique elements can be done with the cardinality aggregation. It is similar to finding the result of a query such as the following:

```
select count(*) from (select distinct username from usageReport) u;
```

- Let's look at how we can find out the count of unique users for which we have network traffic data:

```
GET bigginsight/_search
```

```
{  
  "aggregations": {  
    "unique_visitors": {  
      "cardinality": {  
        "field": "username"  
      }  
    }  
  },  
  "size": 0  
}
```

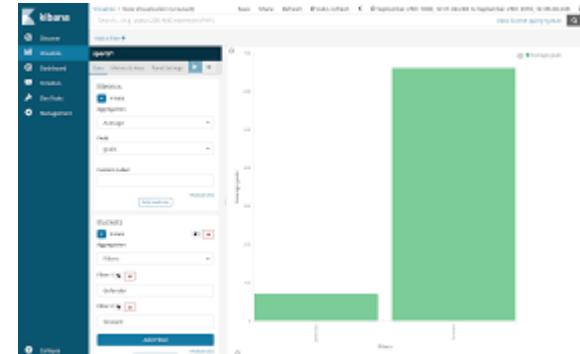
- The cardinality aggregation response is just like the other metric aggregations:

```
{  
  "took": 110,  
  ...  
  "hits": {  
    "total" : {  
      "value" : 10000,  
      "relation" : "gte"  
    },  
    "max_score": 0,  
    "hits": []  
  },  
  "aggregations": {  
    "unique_visitors": {  
      "value": 79  
    }  
  }  
}
```

Bucket aggregations

In this section, we will cover the following topics, keeping the network traffic data example at the center:

- Bucketing on string data
- Bucketing on numerical data
- Aggregating filtered data
- Nesting aggregations
- Bucketing on custom conditions
- Bucketing on date/time data
- Bucketing on geospatial data



Bucketing on string data

Some examples of scenarios in which you may want to segment the data by a string typed field are as follows:

- Segmenting the network traffic data per department
- Segmenting the network traffic data per user
- Segmenting the network traffic data per application, or per category

Terms aggregation

- Which are the top categories, that is, categories that are surfed the most by users?
- We are interested in the most surfed categories – not in terms of the bandwidth used, but just in terms of counts (record counts).
- In a relational database, we could write a query like the following:

```
SELECT category, count(*) FROM usageReport GROUP  
BY category ORDER BY count(*) DESC;
```

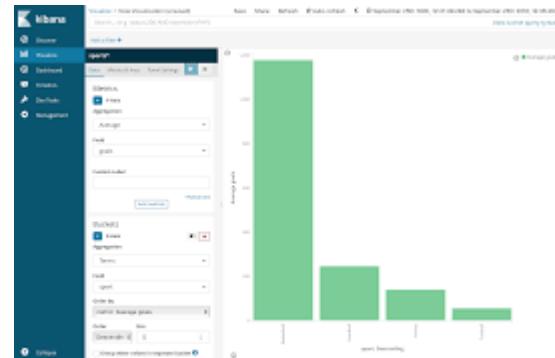
- The Elasticsearch aggregation query, which would do a similar job, can be written as follows:

```
GET /bigginsight/_search
{
  "aggs": {          1
    "byCategory": {  2
      "terms": {     3
        "field": "category" 4
      }
    }
  },
  "size": 0          5
}
```

Terms aggregation

- The response looks like the following:

https://github.com/fenago/elasticsearch/blob/master/snippets/4_4.txt



Terms aggregation

- Next, we want to find out the top applications in terms of the maximum number of records for each application:

```
GET /bigginsight/_search?size=0
```

```
{  
  "aggs": {  
    "byApplication": {  
      "terms": {  
        "field": "application"  
      }  
    }  
  }  
}
```

- Returns a response like the following:

```
{  
...,  
"aggregations": {  
    "byApplication": {  
        "doc_count_error_upper_bound": 6339,  
        "sum_other_doc_count": 129191,  
        "buckets": [  
            {  
                "key": "Skype",  
                "doc_count": 26115  
            },  
            ...  
        ]  
    }  
}
```

- To get the top n buckets instead of the default 10, we can use the size parameter inside the terms aggregation:

```
GET /bigginsight/_search?size=0
```

```
{  
  "aggs": {  
    "byApplication": {  
      "terms": {  
        "field": "application",  
        "size": 15  
      }  
    }  
  }  
}
```

Bucketing on numerical data

- Another common scenario is when we want to segment or slice the data into various buckets, based on a numerical field.
- For example, we may want to slice the product data by different price ranges, such as up to \$10, \$10 to \$50, \$50 to \$100, and so on.
- You may want to segment the data by age group, employee count, and so on.

- We have some records of network traffic usage data.
- The usage field tells us about the number of bytes that are used for uploading or downloading data.
- Let's try to divide or slice all the data based on the usage:

POST /bigginsight/_search?size=0

```
{  
  "aggs": {  
    "by_usage": {  
      "histogram": {  
        "field": "usage",  
        "interval": 1000  
      }  
    }  
  }  
}
```

Histogram aggregation

- The response should look like the following (truncated for brevity):

```
{  
...,  
  "aggregations": {  
    "by_usage": {  
      "buckets": [  
        {  
          "key": 0.0,  
          "doc_count": 30060  
        },  
        {  
          "key": 1000.0,  
          "doc_count": 42880  
        },  
        {  
          "key": 2000.0,  
          "doc_count": 42041  
        },  
        ...  
      ]  
    }  
  }  
}
```

- The `to` value is exclusive, and is not included in the current bucket's range:

```
POST /bigginsight/_search?size=0
```

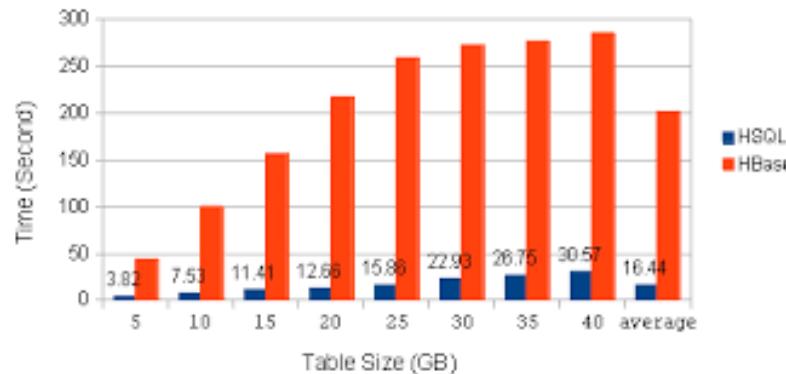
```
{  
  "aggs": {  
    "by_usage": {  
      "range": {  
        "field": "usage",  
        "ranges": [  
          { "to": 1024 },  
          { "from": 1024, "to": 102400 },  
          { "from": 102400 }  
        ]  
      }  
    }  
  }  
}
```

Range aggregation

Range aggregation

- The response looks like the following:

https://github.com/fenago/elasticsearch/blob/master/snippets/4_5.txt



- It is possible to specify custom key labels for the range buckets, as follows:

```
POST /bigginsight/_search?size=0
```

```
{  
  "aggs": {  
    "by_usage": {  
      "range": {  
        "field": "usage",  
        "ranges": [  
          { "key": "Upto 1 kb", "to": 1024 },  
          { "key": "1 kb to 100 kb", "from": 1024, "to": 102400 },  
          { "key": "100 kb and more", "from": 102400 }  
        ]  
      }  
    }  
  }  
}
```

Aggregations on filtered data

- In our quest to learn about different bucket aggregations, let's take a very short detour to understand how to apply aggregations on filtered data.
- So far, we have been applying all of our aggregations on all the data of the given index/type.
- In the real world, you will almost always need to apply some filters before applying aggregations (either metric or bucket aggregations).

- Now, what we want to do is find the top category for a specific customer, not for all of the customers:

```
GET /bigginsight/_search?size=0&track_total_hits=true
```

```
{  
  "query": {  
    "term": {  
      "customer": "Linkedin"  
    }  
  },  
  "aggs": {  
    "byCategory": {  
      "terms": {  
        "field": "category"  
      }  
    }  
  }  
}
```

Aggregations on filtered data

- Let's look at the response of this query to understand this better:

```
{  
  "took": 18,  
  ...,  
  "hits": {  
    "total" : {  
      "value" : 76607,  
      "relation" : "eq"  
    },  
    "max_score": 0,  
    "hits": []  
  },  
  ...  
}
```

```
GET /bigginsight/_search?size=0
{
  "query": {
    "bool": {
      "must": [
        {"term": {"customer": "Linkedin"}},
        {"range": {"time": {"gte": 1506277800000, "lte": 1506294200000}}}
      ]
    }
  },
  "aggs": {
    "byCategory": {
      "terms": {
        "field": "category"
      }
    }
  }
}
```

Nesting aggregations

- Bucket aggregations split the context into one or more buckets.
- We can restrict the context of the aggregation by specifying the query element, as we saw in the previous section.
- When a metric aggregation is nested inside a bucket aggregation, the metric aggregation is computed within each bucket.

Nesting aggregations

- The following query does exactly this.
- Please refer to the annotated numbers, which correspond to the three main objectives of the the following query:

https://github.com/fenago/elasticsearch/blob/master/snippets/4_6.txt

The screenshot shows a debugger interface with two panes. The left pane displays a C# code snippet for performing a search with nested aggregations. The right pane shows the resulting JSON response from Elasticsearch, which is annotated with numbers 1 through 9, corresponding to the objectives listed in the slide. A pink oval highlights the first two annotations in the JSON response.

```
    var results = query
        .Aggregations
        .Terms("employers")
        .Buckets
        .Select(e => new {
            e.Key,
            count = e.DocCount, /* total employees */
            genders = e
                .Terms("genders")
                .Buckets.Select(g => new {
                    gender = g.Key,
                    count = g.DocCount
                })
            .ToList()
        }).ToList();

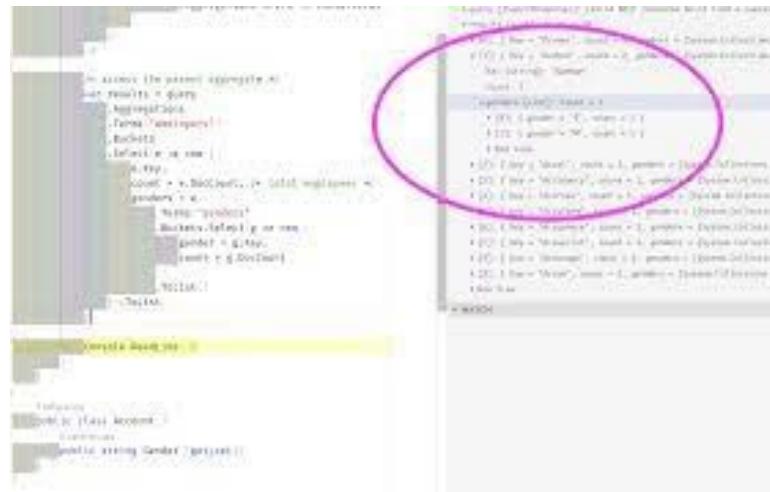
Console.ReadLine();
```

```
> query [ISearchResponse]: {Valid NEST response built from a successful
  > GET request on http://localhost:9200/_search
  > [0]: { Key = "Pyrami", count = 4, genders = {System.Collections.Generic.Dictionary`2[[String, Object]]} }
  > [1]: { Key = "Kurban", count = 2, genders = {System.Collections.Generic.Dictionary`2[[String, Object]]} }
  >   Key [string]: "Kurban"
  >   count: 2
  >   genders [List]: Count = 2
  >     [0]: { gender = "F", count = 1 }
  >     [1]: { gender = "M", count = 1 }
  >   Raw View
  > [2]: { Key = "Accel", count = 1, genders = {System.Collections.Generic.Dictionary`2[[String, Object]]} }
  > [3]: { Key = "Accidency", count = 1, genders = {System.Collections.Generic.Dictionary`2[[String, Object]]} }
  > [4]: { Key = "Accruex", count = 1, genders = {System.Collections.Generic.Dictionary`2[[String, Object]]} }
  > [5]: { Key = "Accufarm", count = 1, genders = {System.Collections.Generic.Dictionary`2[[String, Object]]} }
  > [6]: { Key = "Accupharm", count = 1, genders = {System.Collections.Generic.Dictionary`2[[String, Object]]} }
  > [7]: { Key = "Accuprint", count = 1, genders = {System.Collections.Generic.Dictionary`2[[String, Object]]} }
  > [8]: { Key = "Accusage", count = 1, genders = {System.Collections.Generic.Dictionary`2[[String, Object]]} }
  > [9]: { Key = "Acium", count = 1, genders = {System.Collections.Generic.Dictionary`2[[String, Object]]} }
  >   Raw View
  > WATCH
```

Nesting aggregations

- The response looks like the following:

https://github.com/fenago/elasticsearch/blob/master/snippets/4_7.txt



A screenshot of a web browser displaying the results of an Elasticsearch search query. The results are presented in a table format with columns for '_id', '_index', '_score', and '_type'. A purple oval highlights the '_score' column, which contains numerical values ranging from 0.000 to 0.004. The rest of the table shows document IDs, indices, types, and their respective scores.

_id	_index	_score	_type
1	0	0.000	Document
2	0	0.000	Document
3	0	0.000	Document
4	0	0.000	Document
5	0	0.000	Document
6	0	0.000	Document
7	0	0.000	Document
8	0	0.000	Document
9	0	0.000	Document
10	0	0.000	Document
11	0	0.000	Document
12	0	0.000	Document
13	0	0.000	Document
14	0	0.000	Document
15	0	0.000	Document
16	0	0.000	Document
17	0	0.000	Document
18	0	0.000	Document
19	0	0.000	Document
20	0	0.000	Document
21	0	0.000	Document
22	0	0.000	Document
23	0	0.000	Document
24	0	0.000	Document
25	0	0.000	Document
26	0	0.000	Document
27	0	0.000	Document
28	0	0.000	Document
29	0	0.000	Document
30	0	0.000	Document
31	0	0.000	Document
32	0	0.000	Document
33	0	0.000	Document
34	0	0.000	Document
35	0	0.000	Document
36	0	0.000	Document
37	0	0.000	Document
38	0	0.000	Document
39	0	0.000	Document
40	0	0.000	Document
41	0	0.000	Document
42	0	0.000	Document
43	0	0.000	Document
44	0	0.000	Document
45	0	0.000	Document
46	0	0.000	Document
47	0	0.000	Document
48	0	0.000	Document
49	0	0.000	Document
50	0	0.000	Document
51	0	0.000	Document
52	0	0.000	Document
53	0	0.000	Document
54	0	0.000	Document
55	0	0.000	Document
56	0	0.000	Document
57	0	0.000	Document
58	0	0.000	Document
59	0	0.000	Document
60	0	0.000	Document
61	0	0.000	Document
62	0	0.000	Document
63	0	0.000	Document
64	0	0.000	Document
65	0	0.000	Document
66	0	0.000	Document
67	0	0.000	Document
68	0	0.000	Document
69	0	0.000	Document
70	0	0.000	Document
71	0	0.000	Document
72	0	0.000	Document
73	0	0.000	Document
74	0	0.000	Document
75	0	0.000	Document
76	0	0.000	Document
77	0	0.000	Document
78	0	0.000	Document
79	0	0.000	Document
80	0	0.000	Document
81	0	0.000	Document
82	0	0.000	Document
83	0	0.000	Document
84	0	0.000	Document
85	0	0.000	Document
86	0	0.000	Document
87	0	0.000	Document
88	0	0.000	Document
89	0	0.000	Document
90	0	0.000	Document
91	0	0.000	Document
92	0	0.000	Document
93	0	0.000	Document
94	0	0.000	Document
95	0	0.000	Document
96	0	0.000	Document
97	0	0.000	Document
98	0	0.000	Document
99	0	0.000	Document
100	0	0.000	Document
101	0	0.000	Document
102	0	0.000	Document
103	0	0.000	Document
104	0	0.000	Document
105	0	0.000	Document
106	0	0.000	Document
107	0	0.000	Document
108	0	0.000	Document
109	0	0.000	Document
110	0	0.000	Document
111	0	0.000	Document
112	0	0.000	Document
113	0	0.000	Document
114	0	0.000	Document
115	0	0.000	Document
116	0	0.000	Document
117	0	0.000	Document
118	0	0.000	Document
119	0	0.000	Document
120	0	0.000	Document
121	0	0.000	Document
122	0	0.000	Document
123	0	0.000	Document
124	0	0.000	Document
125	0	0.000	Document
126	0	0.000	Document
127	0	0.000	Document
128	0	0.000	Document
129	0	0.000	Document
130	0	0.000	Document
131	0	0.000	Document
132	0	0.000	Document
133	0	0.000	Document
134	0	0.000	Document
135	0	0.000	Document
136	0	0.000	Document
137	0	0.000	Document
138	0	0.000	Document
139	0	0.000	Document
140	0	0.000	Document
141	0	0.000	Document
142	0	0.000	Document
143	0	0.000	Document
144	0	0.000	Document
145	0	0.000	Document
146	0	0.000	Document
147	0	0.000	Document
148	0	0.000	Document
149	0	0.000	Document
150	0	0.000	Document
151	0	0.000	Document
152	0	0.000	Document
153	0	0.000	Document
154	0	0.000	Document
155	0	0.000	Document
156	0	0.000	Document
157	0	0.000	Document
158	0	0.000	Document
159	0	0.000	Document
160	0	0.000	Document
161	0	0.000	Document
162	0	0.000	Document
163	0	0.000	Document
164	0	0.000	Document
165	0	0.000	Document
166	0	0.000	Document
167	0	0.000	Document
168	0	0.000	Document
169	0	0.000	Document
170	0	0.000	Document
171	0	0.000	Document
172	0	0.000	Document
173	0	0.000	Document
174	0	0.000	Document
175	0	0.000	Document
176	0	0.000	Document
177	0	0.000	Document
178	0	0.000	Document
179	0	0.000	Document
180	0	0.000	Document
181	0	0.000	Document
182	0	0.000	Document
183	0	0.000	Document
184	0	0.000	Document
185	0	0.000	Document
186	0	0.000	Document
187	0	0.000	Document
188	0	0.000	Document
189	0	0.000	Document
190	0	0.000	Document
191	0	0.000	Document
192	0	0.000	Document
193	0	0.000	Document
194	0	0.000	Document
195	0	0.000	Document
196	0	0.000	Document
197	0	0.000	Document
198	0	0.000	Document
199	0	0.000	Document
200	0	0.000	Document
201	0	0.000	Document
202	0	0.000	Document
203	0	0.000	Document
204	0	0.000	Document
205	0	0.000	Document
206	0	0.000	Document
207	0	0.000	Document
208	0	0.000	Document
209	0	0.000	Document
210	0	0.000	Document
211	0	0.000	Document
212	0	0.000	Document
213	0	0.000	Document
214	0	0.000	Document
215	0	0.000	Document
216	0	0.000	Document
217	0	0.000	Document
218	0	0.000	Document
219	0	0.000	Document
220	0	0.000	Document
221	0	0.000	Document
222	0	0.000	Document
223	0	0.000	Document
224	0	0.000	Document
225	0	0.000	Document
226	0	0.000	Document
227	0	0.000	Document
228	0	0.000	Document
229	0	0.000	Document
230	0	0.000	Document
231	0	0.000	Document
232	0	0.000	Document
233	0	0.000	Document
234	0	0.000	Document
235	0	0.000	Document
236	0	0.000	Document
237	0	0.000	Document
238	0	0.000	Document
239	0	0.000	Document
240	0	0.000	Document
241	0	0.000	Document
242	0	0.000	Document
243	0	0.000	Document
244	0	0.000	Document
245	0	0.000	Document
246	0	0.000	Document
247	0	0.000	Document
248	0	0.000	Document
249	0	0.000	Document
250	0	0.000	Document
251	0	0.000	Document
252	0	0.000	Document
253	0	0.000	Document
254	0	0.000	Document
255	0	0.000	Document
256	0	0.000	Document
257	0	0.000	Document
258	0	0.000	Document
259	0	0.000	Document
260	0	0.000	Document
261	0	0.000	Document
262	0	0.000	Document
263	0	0.000	Document
264	0	0.000	Document
265	0	0.000	Document
266	0	0.000	Document
267	0	0.000	Document
268	0	0.000	Document
269	0	0.000	Document
270	0	0.000	Document
271	0	0.000	Document
272	0	0.000	Document
273	0	0.000	Document
274	0	0.000	Document
275	0	0.000	Document
276	0	0.000	Document
277	0	0.000	Document
278	0	0.000	Document
279	0	0.000	Document
280	0	0.000	Document
281	0	0.000	Document
282	0	0.000	Document
283	0	0.000	Document
284	0	0.000	Document
285	0	0.000	Document
286	0	0.000	Document
287	0	0.000	Document
288	0	0.000	Document
289	0	0.000	Document
290	0	0.000	Document
291	0	0.000	Document
292	0	0.000	Document
293	0	0.000	Document
294	0	0.000	Document
295	0	0.000	Document
296	0	0.000	Document
297	0	0.000	Document
298	0	0.000	Document
299	0	0.000	Document
300	0	0.000	Document
301	0	0.000	Document
302	0	0.000	Document
303	0	0.000	Document
304	0	0.000	Document
305	0	0.000	Document
306	0	0.000	Document
307	0	0.000	Document
308	0	0.000	Document
309	0	0.000	Document
310	0	0.000	Document
311	0	0.000	Document
312	0	0.000	Document
313	0	0.000	Document
314	0	0.000	Document
315	0	0.000	Document
316	0	0.000	Document
317	0	0.000	Document
318	0	0.000	Document
319	0	0.000	Document
320	0	0.000	Document
321	0	0.000	Document
322	0	0.000	Document
323	0	0.000	Document
324	0	0.000	Document
325	0	0.000	Document
326	0	0.000	Document
327	0	0.000	Document
328	0	0.000	Document
329	0	0.000	Document
330	0	0.000	Document
331	0	0.000	Document
332	0	0.000	Document
333	0	0.000	Document
334	0	0.000	Document
335	0	0.000	Document
336	0	0.000	Document
337	0	0.000	Document
338	0	0.000	Document
339	0	0.000	Document
340	0	0.000	Document
341	0	0.000	Document
342	0	0.000	Document
343	0	0.000	Document
344	0	0.000	Document
345	0	0.000	Document
346	0	0.000	Document
347	0	0.000	Document
348	0	0.000	Document
349	0	0.000	Document
350	0	0.000	Document
351	0	0.000	Document
352	0	0.000	Document
353	0	0.000	Document
354	0	0.000	Document
355	0	0.000	Document
356	0	0.000	Document
357	0	0.000	Document

- Please see the following partial query, which has been modified to sort the buckets in descending order of the total_usage metric:

GET /bigginsight/usageReport/_search

```
{  
...,  
"aggs": {  
    "by_users": {  
        "terms": {  
            "field": "username",  
        "order": { "total_usage": "desc"}  
    },  
    "aggs": {  
        ...  
    }  
}
```

Nesting aggregations

- Who are the top two users in each department, given the total bandwidth consumed by each user?
- The following query will help us get that answer:

https://github.com/fenago/elasticsearch/blob/master/snippets/4_8.txt

Bucketing on custom conditions

- Sometimes, what we want is more control over how the buckets are created.
- The aggregations that we have looked at so far have dealt with a single type of field.
- If the given field that we want to slice data from is of the string type, we generally use the terms aggregation.
- If the field is of the numerical type, we have a few choices, including histogram, range aggregation, and others, to slice the data into different segments.

Filter aggregation

- We want to create a bucket of all records that have category = Chat:

```
POST /bigginsight/_search?size=0
```

```
{  
  "aggs": {  
    "chat": {  
      "filter": {  
        "term": {  
          "category": "Chat"  
        }  
      }  
    }  
  }  
}
```

- The response should look like the following:

```
{  
  "took": 4,  
  ...,  
  "hits": {  
    "total" : {  
      "value" : 10000,  
      "relation" : "gte"  
    },  
    "max_score": 0,  
    "hits": []  
  },  
  "aggregations": {  
    "chat": {  
      "doc_count": 52277  
    }  
  }  
}
```

Filters aggregation

- With filters aggregation, you can create multiple buckets, each with its own specified filter that will cause the documents satisfying that filter to fall into the related bucket. Let's look at an example.
- Suppose that we want to create multiple buckets to understand how much of the network traffic was caused by the Chat category.

- It allows us to write arbitrary filters to create buckets:

```
GET bigginsight/_search?size=0
{
  "aggs": {
    "messages": {
      "filters": {
        "filters": {
          "chat": { "match": { "category": "Chat" } },
          "skype": { "match": { "application": "Skype" } },
          "other_than_skype": {
            "bool": {
              "must": { "match": { "category": "Chat" } },
              "must_not": { "match": { "application": "Skype" } }
            }
          }
        }
      }
    }
  }
}
```

Bucketing on date/time data

In the context of the network traffic example that we are going through, the following questions can be answered through time series analysis of the data:

- How are the bandwidth requirements changing for my organization over a period of time?
- Which are the top applications, over a period of time, in terms of bandwidth usage?

Date Histogram aggregation

Using Date Histogram aggregation, we will see how we can create buckets on a date field. In the process, we will go through the following stages:

- Creating buckets across time periods
- Using a different time zone
- Computing other metrics within sliced time intervals
- Focusing on a specific day and changing intervals

Creating buckets across time periods

- The following query will create buckets on different values of time, grouped by one-day intervals:

```
GET /bigginsight/_search?size=0          1  
{  
  "aggs": {  
    "counts_over_time": {  
      "date_histogram": {  
        "field": "time",           2  
        "interval": "1d"          3  
      }  
    }  
  }  
}
```

Creating buckets across time periods

- The response looks like the following:

https://github.com/fenago/elasticsearch/blob/master/snippets/4_9.txt

Using a different time zone

```
GET /bigginsight/_search?size=0
{
  "aggs": {
    "counts_over_time": {
      "date_histogram": {
        "field": "time",
        "interval": "1d",
        "time_zone": "+05:30"
      }
    }
  }
}
```

Using a different time zone

- The response now looks like the following:

https://github.com/fenago/elasticsearch/blob/master/snippets/4_10.txt

Computing other metrics within sliced time intervals

- We have just sliced the data across time by using the Date Histogram to create the buckets on the time field.
- This gave us the document counts in each bucket. Next, we will try to answer the following question:
- What is the day-wise total bandwidth usage for a given customer?

```
GET /bigginsight/_search?size=0
{
  "query": { "term": {"customer": "Linkedin"} },
  "aggs": {
    "counts_over_time": {
      "date_histogram": {
        "field": "time",
        "interval": "1d",
        "time_zone": "+05:30"
      },
      "aggs": {
        "total_bandwidth": {
          "sum": { "field": "usage" }
        }
      }
    }
  }
}
```

- The following is the shortened response to the query:

```
{  
  ...  
  "aggregations": {  
    "counts_over_time": {  
      "buckets": [  
        {  
          "key_as_string": "2017-09-23T00:00:00.000+05:30",  
          "key": 1506105000000,  
          "doc_count": 18892,  
          "total_bandwidth": {  
            "value": 265574303  
          }  
        },  
        ...  
      ]  
    }  
  }  
}
```

Focusing on a specific day and changing intervals

- What we are doing is also called drilling down in the data, Often, the result of the previous query is displayed as a line chart, with time on the x axis and data used on the y axis.
- If we want to zoom in on a specific day from that line chart, the following query can be useful:

https://github.com/fenago/elasticsearch/blob/master/snippets/4_11.txt

Focusing on a specific day and changing intervals

- The shortened response would look like the following:

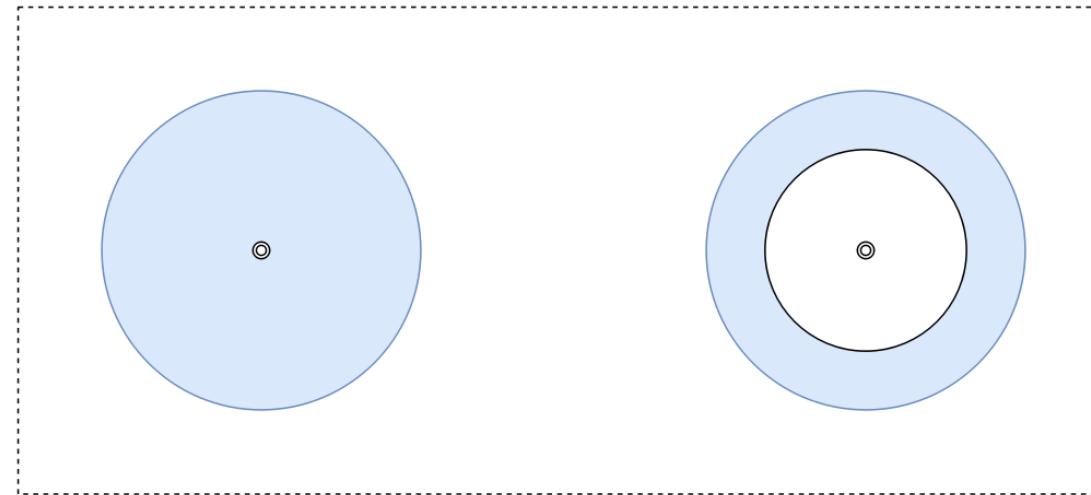
https://github.com/fenago/elasticsearch/blob/master/snippets/4_12.txt

Bucketing on geospatial data

- Another powerful feature of bucket aggregation is the ability to do geospatial analysis on the data.
- If your data contains fields of the geo-point datatype, where the coordinates are captured, you can perform some interesting analysis, which can be rendered on a map to give you better insight into the data.

Geodistance aggregation

- Geodistance aggregation helps in creating buckets of distances from a given geo-point.
- This can be better illustrated using a diagram:



- The shaded area is from the center up to the given radius, forming a circle:

```
GET bigginsight/_search?size=0
```

```
{  
  "aggs": {  
    "within_radius": {  
      "geo_distance": {  
        "field": "location",  
        "origin": {"lat": 23.102869,"lon": 72.595692},  
        "ranges": [{"to": 5}]  
      }  
    }  
  }  
}
```

- let's look at what happens if you specify both from and to in the geodistance aggregation.
- This will correspond to the right circle in the preceding diagram:

```
GET bigginsight/_search?size=0
```

```
{  
  "aggs": {  
    "within_radius": {  
      "geo_distance": {  
        "field": "location",  
        "origin": {"lat": 23.102869, "lon": 72.595692},  
        "ranges": [{"from": 5, "to": 10}]  
      }  
    }  
  }  
}
```

GeoHash grid aggregation

- Lower values of precision represent larger geographical areas, while higher values represent smaller, more precise geographical areas:

```
GET bigginsight/_search?size=0
```

```
{  
  "aggs": {  
    "geo_hash": {  
      "geohash_grid": {  
        "field": "location",  
        "precision": 7  
      }  
    }  
  }  
}
```

GeoHash grid aggregation

- The data that we have in our network traffic example is spread over a very small geographical area, so we have used a precision of 7.
- The supported values for precision are from 1 to 12.
 - Let's look at the response to this request:

https://github.com/fenago/elasticsearch/blob/master/snippets/4_13.txt

GeoHash grid aggregation

- When you try a precision value of 9, you will see the following response:

https://github.com/fenago/elasticsearch/blob/master/snippets/4_14.txt

Pipeline aggregations

Pipeline aggregations are a relatively new feature, and they are still experimental. At a high level, there are two types of pipeline aggregation:

- Parent pipeline aggregations have the pipeline aggregation nested inside other aggregations
- Sibling pipeline aggregations have the pipeline aggregation as the sibling of the original aggregation from which pipelining is done

Calculating the cumulative sum of usage over time

- Using cumulative sum aggregation, we can also compute the cumulative bandwidth usage at the end of every hour of the day.
- Let's look at the query and try to understand it:

https://github.com/fenago/elasticsearch/blob/master/snippets/4_15.txt

Calculating the cumulative sum of usage over time

- The response should look as follows.
- It has been truncated for brevity:

https://github.com/fenago/elasticsearch/blob/master/snippets/4_16.txt



Summary

- In this lesson, you learned how to use Elasticsearch to build powerful analytics applications.
- We covered how to slice and dice the data to get powerful insight.
- We started with metric aggregation and dealt with numerical datatypes.
- We then covered bucket aggregation in order to find out how to slice the data into buckets or segments, in order to drill down into specific segments.

COMPLETE LAB 4

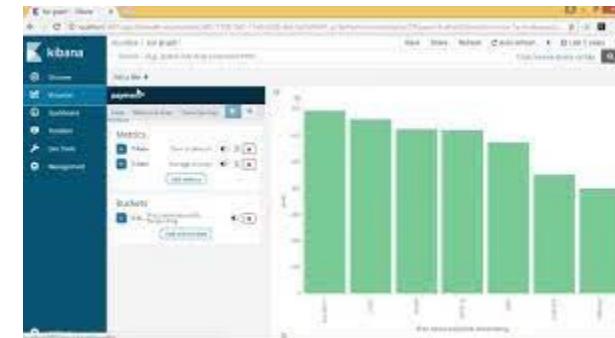
5. Analyzing Log Data



Analyzing Log Data

In this lesson, we will be exploring Logstash, another key component of the Elastic Stack that is mainly used as an ETL (Extract, Transform, and Load) engine. We will also be exploring the following topics:

- Log Analysis challenges
- Using Logstash
- The Logstash architecture
- Overview of Logstash plugins
- Ingest node



Log analysis challenges

- Logs are defined as records of incidents or observations.
- Logs are generated by a wide variety of resources, such as systems, applications, devices, humans, and so on.
- A log is typically made of two things; that is, a timestamp (the time the event was generated) and data (the information related to the event):

Log = Timestamp + Data

2011-05-20 20:06:06.46 Server This instance of SQL Server last reported using a process ID of 1760 at 5/20/2011 8:03:41 PM (C)
2011-05-20 20:06:06.46 Server Registry startup parameters:
-d C:\Program Files\Microsoft SQL Server\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\master.mdf
-e C:\Program Files\Microsoft SQL Server\MSSQL10_50.MSSQLSERVER\MSSQL\Log\ERRORLOG
-l C:\Program Files\Microsoft SQL Server\MSSQL10_50.MSSQLSERVER\MSSQL\DATA\mastlog.ldf

SQL Server Logs

2011-05-20 20:06:06.66 Server SQL Server is starting at normal priority base (=7). This is an informational message only. No

```
[2017-08-03 10:26:03,550]{WARN }{index.store} [model] {ho_index}[3] failed to build store metadata. checking segment info
integrity (with commit [no])
java.nio.file.NoSuchFileException: /u01/psft/pt/es2.3.2/data/E8Cluster/nodes/0/indices/ho_hr_job_data_hc92stp/3/index/segments_lg
at sun.nio.fs.UnixException.translateToIOException(UnixException.java:86)
at sun.nio.fs.UnixException.rethrowAsIOException(UnixException.java:102)
at sun.nio.fs.UnixException.rethrowAsIOException(UnixException.java:107)
at sun.nio.fs.UnixFileSystemProvider.newFileChannel(UnixFileSystemProvider.java:177)
at java.nio.channels.FileChannel.open(FileChannel.java:287)
at java.nio.channels.FileChannel.open(FileChannel.java:335)
at org.apache.lucene.store.NIOFSDirectory.openInput(NIOFSDirectory.java:81)
at org.apache.lucene.store.FileSwitchDirectory.openInput(FileSwitchDirectory.java:186)
at org.apache.lucene.store.FilterDirectory.openInput(FilterDirectory.java:89)
at org.apache.lucene.store.FilterDirectory.openInput(FilterDirectory.java:89)
at org.elasticsearch.index.store.Store$MetadataSnapshot.checksumFromLuceneFile(Store.java:930)
at org.elasticsearch.index.store.Store$MetadataSnapshot.loadMetadata(Store.java:840)
at org.elasticsearch.index.store.Store$MetadataSnapshot.<init>(Store.java:764)
at org.elasticsearch.index.store.Store.getMetadata(Store.java:233)
at org.elasticsearch.index.store.Store.getMetadataOrEmpty(Store.java:192)
at org.elasticsearch.indices.store.TransportNodesList$ShardStoreMetaData.listStoreMetaData(TransportNodesList$ShardStoreMetaData.java:161)
at org.elasticsearch.indices.store.TransportNodesList$ShardStoreMetaData.nodeOperation(TransportNodesList$ShardStoreMetaData.java:142)
at org.elasticsearch.indices.store.TransportNodesList$ShardStoreMetaData.nodeOperation(TransportNodesList$ShardStoreMetaData.java:67)
at org.elasticsearch.action.support.nodes.TransportNodesAction.nodeOperation(TransportNodesAction.java:92)
at org.elasticsearch.action.support.nodes.TransportNodesAction$NodeTransportHandler.messageReceived(TransportNodesAction.java:230)
at org.elasticsearch.action.support.nodes.TransportNodesAction$NodeTransportHandler.messageReceived(TransportNodesAction.java:226)
at org.elasticsearch.transport.RequestHandlerRegistry.processMessageReceived(RequestHandlerRegistry.java:75)
at org.elasticsearch.transport.netty.MessageChannelHandler$RequestHandler.doRun(MessageChannelHandler.java:300)
at org.elasticsearch.common.util.concurrent.AbstractRunnable.run(AbstractRunnable.java:37)
```

Elasticsearch Exceptions

93.180.71.3 -- [17/May/2015:08:05:23 +0000] "GET /downloads/product_1 HTTP/1.1" 304 0 "--" "Debian
APT-HTTP/1.3 (0.8.16~exp12ubuntu10.21)"
80.91.33.133 -- [17/May/2015:08:05:24 +0000] "GET /downloads/product_1 HTTP/1.1" 304 0 "--" "Debian
APT-HTTP/1.3 (0.8.16~exp12ubuntu10.17)"

GNIX logs

Log analysis challenges

- Correlating events occur across multiple systems at the same time.
- Some example time formats that can be seen in the logs are as follows:

Nov 14 22:20:10

[10/Oct/2000:13:55:36 -0700]

172720538

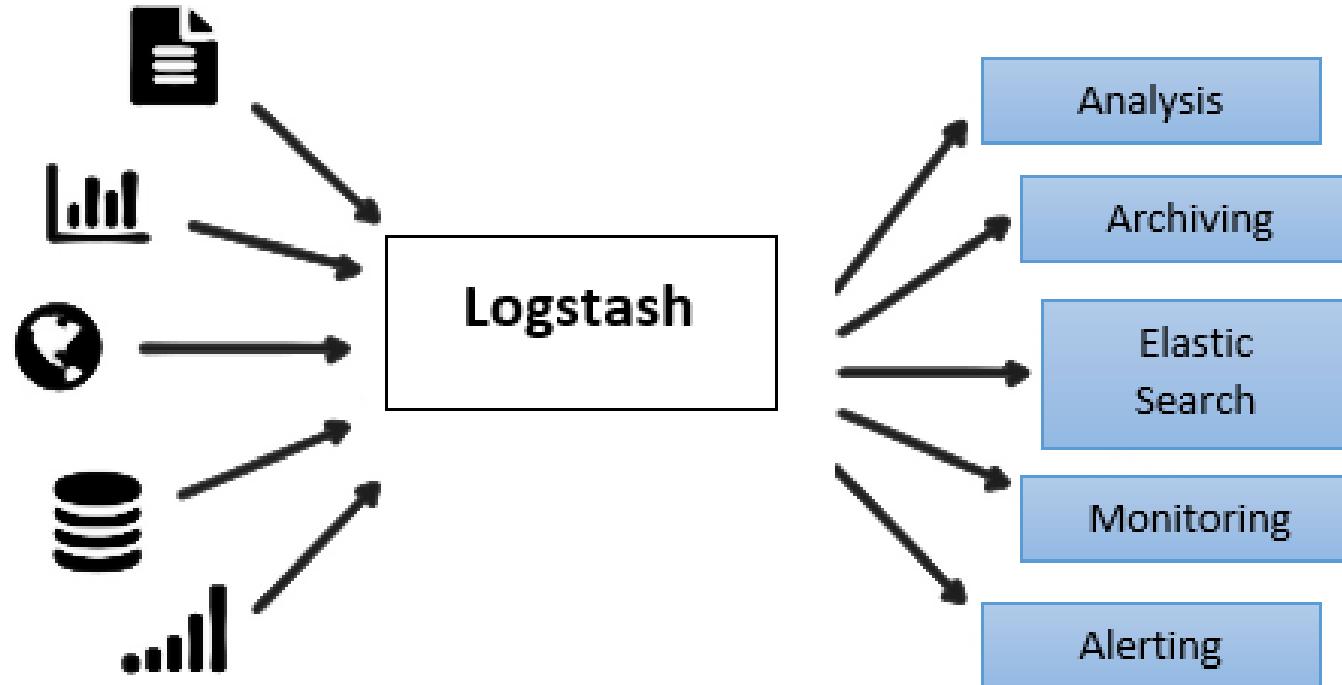
053005 05:45:21

1508832211657

Using Logstash

- Logstash is a popular open source data collection engine with real-time pipelining capabilities.
- Logstash allows us to easily build a pipeline that can help in collecting data from a wide variety of input sources, and parse, enrich, unify, and store it in a wide variety of destinations.

Using Logstash



Installation and configuration

Prerequisites

- Java runtime is required to run Logstash. Logstash requires Java 8.
- Make sure that JAVA_HOME is set as an environment variable, and to check your Java version, run the following command:

java -version

- You should see the following output:

java version "1.8.0_65"Java(TM) SE

RuntimeEnvironment(build 1.8.0_65-b17)JavaHotSpot(TM)64-BitServer VM (build 25.65-b01, mixed mode)

Downloading and installing Logstash

Downloads

Download Logstash - OSS Only

Want to upgrade? We'll give you a hand. [Migration Guide »](#)

Version: 7.0.0

Release date: April 10, 2019

License: [Apache 2.0](#)

Downloads: [↳ TAR.GZ sha](#)

[↳ ZIP sha](#)

[↳ DEB sha](#)

[↳ RPM sha](#)

Notes: This distribution only includes features licensed under the Apache 2.0 license. To get access to full [set of free features](#), use the [default distribution](#).

View the detailed release notes [here](#).

Not the version you're looking for? View [past releases](#).

Java 8 is required for Logstash 6.x and 5.x.

Installing on Windows

- Rename the downloaded file logstash-7.0.0.zip. Unzip the downloaded file.
- Once unzipped, navigate to the newly created folder, as shown in the following code snippet:

```
E:\>cd logstash-oss-7.0.0
```

Installing on Linux

- Unzip the tar.gz package and navigate to the newly created folder, as follows:

```
$>tar -xzf logstash-oss-7.0.0.tar.gz  
$>cd logstash-7.0.0/
```

Running Logstash

- Let's ensure that Logstash works fine after installation by running the following command with a simple configuration (the logstash pipeline) as a parameter:

```
E:\logstash-7.0.0\bin>logstash -e "input { stdin {} } output { stdout {} }"
```

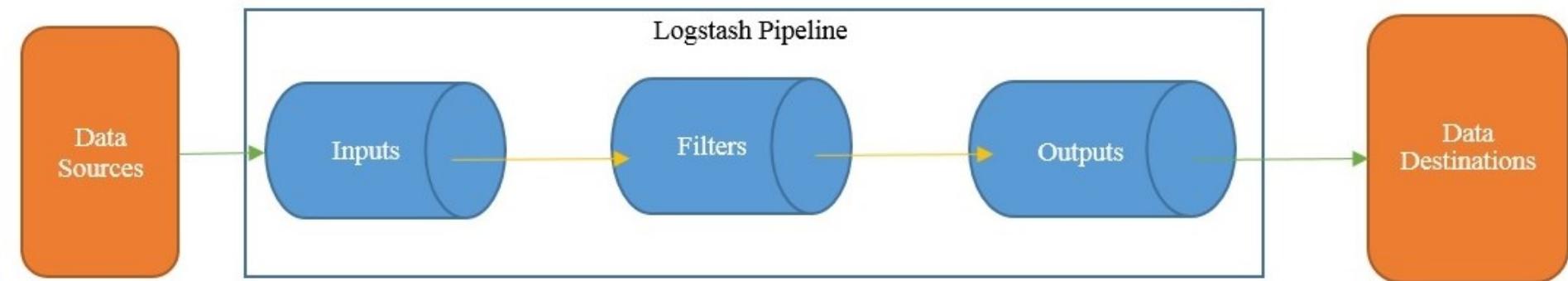
Running Logstash

- You should get the following logs:

https://github.com/fenago/elasticsearch/blob/master/snippets/5_1.txt

The Logstash architecture

- The Logstash event processing pipeline has three stages, that is, Inputs, Filters, and Outputs.
- A Logstash pipeline has two required elements, that is, input and output, and one option element known as filters:



The Logstash architecture

- The Logstash pipeline is stored in a configuration file that ends with a .conf extension.
- The three sections of the configuration file are as follows:

input

```
{  
}
```

filter

```
{  
}
```

output

```
{  
}
```

The Logstash architecture

- Let's use the same configuration that we used in the previous section, with some little modifications, and store it in a file:

https://github.com/fenago/elasticsearch/blob/master/snippets/5_2.txt

The Logstash architecture

- Let's run Logstash using this new pipeline/configuration that's stored in the simple.conf file, as follows:

```
E:\logstash-7.0.0\bin>logstash -f ..\conf\simple.conf
```

The Logstash architecture

- Once Logstash has started, enter any input, say, LOGSTASH IS AWESOME, and you should see the response, as follows:

```
{  
  "@version" => "1",  
  "host" => "SHMN-IN",  
  "@timestamp" => 2017-11-03T11:42:56.221Z,  
  "message" => "LOGSTASH IS AWESOME\r"  
}
```

Overview of Logstash plugins

- By default, as part of the Logstash distribution, many common plugins are available out of the box.
- You can verify the list of plugins that are part of the current installation by executing the following command:

```
E:\logstash-7.0.0\bin>logstash-plugin list
```

Overview of Logstash plugins

- Using the --group flag, followed by either input, filter, output, or codec, you can find the list of installed input, filters, output, codecs, and plugins, respectively.
- For example:

E:\logstash-7.0.0\bin>logstash-plugin list --group filter

- You can list all the plugins containing a name fragment by passing the name fragment to logstash-plugin list.
- For example:

E:\logstash-7.0.0\bin>logstash-plugin list 'pager'

Installing or updating plugins

- If the required plugin is not bundled by default, you can install it using the bin\logstash-plugin install command.
- For example, to install the logstash-output-email plugin, execute the following command:

```
E:\logstash-7.0.0\bin>logstash-plugin install logstash-output-email
```

Installing or updating plugins

- By using the bin\logstash-plugin update command and passing the plugin name as a parameter to the command, you can get the latest version of the plugin:

```
E:\logstash-oss-7.0.0\bin>logstash-plugin update  
logstash-output-s3
```

Input plugins

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

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		

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		

Codec plugins

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

Exploring input plugins

File

- The file plugin is used to stream events from file(s) line by line. It works in a similar fashion to the tail -Of Linux\Unix command.
- For each file, it keeps track of any changes in the file, and the last location from where the file was read, only sends the data since it was last read.
- It also automatically detects file rotation, This plugin also provides the option to read the file from the beginning of the file.

- Let's take some example configurations to understand this plugin better:

```
#sample configuration 1
#simple1.conf
input
{
  file{
    path => "/usr/local/logfiles/*"
  }
}
output
{
  stdout {
    codec => rubydebug
  }
}
```

Exploring input plugins

- The preceding configuration specifies the streaming of all the new entries (that is, tailing the files) to the files found under the /usr/local/logfiles/ location:

https://github.com/fenago/elasticsearch/blob/master/snippets/5_3.txt

Beats

- The Beats input plugin allows Logstash to receive events from the Elastic Beats framework.
- Beats are a collection of lightweight daemons that collect operational data from your servers and ship to configured outputs such as Logstash, Elasticsearch, Redis, and so on.
- There are several Beats available, including Metricbeat, Filebeat, Winlogbeat, and so on.

Beats

- By using the beats input plugin, we can make Logstash listen on desired ports for incoming Beats connections:

```
#beats.conf
input {
  beats {
    port => 1234
  }
}
output {
  elasticsearch {
  }
}
```

Beats

- port is the only required setting for this plugin, The preceding configuration makes Logstash listen for incoming Beats connections and index into Elasticsearch.
- When you start Logstash with the preceding configuration, you may notice Logstash starting an input listener on port 1234 in the logs, as follows:

https://github.com/fenago/elasticsearch/blob/master/snippets/5_4.txt

Beats

- You can start multiple listeners to listen for incoming Beats connections as follows:

https://github.com/fenago/elasticsearch/blob/master/snippets/5_5.txt

- This plugin is used to import data from a database to Logstash.
- Each row in the results set would become an event, and each column would get converted into fields in the event.
- Using this plugin, you can import all the data at once by running a query, or you can periodically schedule the import using cron syntax (using the schedule parameter/setting).

- Let's look at an example:

https://github.com/fenago/elasticsearch/blob/master/snippets/5_6.txt

IMAP

- This plugin is used to read emails from an IMAP server.
- This plugin can be used to read emails and, depending on the email context, the subject of the email, or specific senders, it can be conditionally processed in Logstash and used to raise JIRA tickets, pagerduty events, and so on.
- The required configurations are host, password, and user.

IMAP

- User and password are where you need to specify the user credentials to authenticate/connect to the IMAP server:

https://github.com/fenago/elasticsearch/blob/master/snippets/5_7.txt

Elasticsearch

- This plugin is used for transferring events from Logstash to Elasticsearch.
- This plugin is the recommended approach for pushing events/log data from Logstash to Elasticsearch.
- Once the data is in Elasticsearch, it can be easily visualized using Kibana.
- This plugin requires no mandatory parameters and it automatically tries to connect to Elasticsearch, which is hosted on localhost:9200.

Elasticsearch

- The simple configuration of this plugin would be as follows:

```
#elasticsearch1.conf
input {
  stdin{
  }
}
output {
  elasticsearch {
  }
}
```

Elasticsearch

- Often, Elasticsearch will be hosted on a different server that's usually secure, and we might want to store the incoming data in specific indexes.
- Let's look at an example of this:

https://github.com/fenago/elasticsearch/blob/master/snippets/5_8.txt

CSV

- This plugin is used for storing output in the CSV format.
- The required parameters for this plugin are the path parameter, which is used to specify the location of the output file, and fields, which specifies the field names from the event that should be written to the CSV file.
- If a field does not exist on the event, an empty string will be written.

CSV

- Let's look at an example. In the following configuration, Elasticsearch is queried against the apachelogs index for all documents matching statuscode:200, and the message, @timestamp, and host fields are written to a .csv file:

https://github.com/fenago/elasticsearch/blob/master/snippets/5_9.txt

Kafka

- This plugin is used to write events to a Kafka topic.
- It uses the Kafka Producer API to write messages to a topic on the broker.
- The only required configuration is the `topic_id`.

Kafka

- Let's look at a basic Kafka configuration:

```
#kafka.conf
input {
  stdin{
  }
}
output {
kafka {
    bootstrap_servers => "localhost:9092"
    topic_id => 'logstash'
}
}
```

PagerDuty

- This output plugin will send notifications based on preconfigured services and escalation policies.
- The only required parameter for this plugin is the `service_key` to specify the Service API Key.

PagerDuty

- Let's look at a simple example with basic pagerduty configuration.
- In the following configuration, Elasticsearch is queried against the nginxlogs index for all documents matching statuscode:404, and pagerduty events are raised for each document returned by Elasticsearch:

https://github.com/fenago/elasticsearch/blob/master/snippets/5_10.txt

JSON

- This codec is useful if the data consists of .json documents, and is used to encode (if used in output plugins) or decode (if used in input plugins) the data in the .json format.
- If the data being sent is a JSON array at its root, multiple events will be created (that is, one per element).

JSON

- The simple usage of a JSON codec plugin is as follows:

```
input{  
    stdin{  
        codec => "json"  
    }  
}
```

Rubydebug

- This codec will output your Logstash event data using the Ruby Awesome Print library.
- The simple usage of this codec plugin is as follows:

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

Multiline

- The sample usage of this codec plugin is shown in the following snippet:

```
input {  
    file {  
        path =>"/var/log/access.log"  
        codec => multiline {pattern =>"^\\s "  
            negate =>false  
            what =>"previous"}}  
}
```

Ingest node

- Prior to Elasticsearch 5.0, if we wanted to preprocess documents before indexing them to Elasticsearch, then the only way was to make use of Logstash or preprocess them programmatically/manually and then index them to Elasticsearch.
- Elasticsearch lacked the ability to preprocess/transform the documents, and it just indexed the documents as they were.

Ingest node

- If an Elasticsearch node is implemented with the default configuration, by default, it would be master, data, and ingest enabled (that is, it would act as a master node, data node, and ingest node).
- To disable ingest on a node, configure the following setting in the `elasticsearch.yml` file:

`node.ingest: false`

Ingest node

- To preprocess a document before indexing, we must define the pipeline (which contains sequences of steps known as processors for transforming an incoming document).
- To use a pipeline, we simply specify the pipeline parameter on an index or bulk request to tell the ingest node which pipeline to use:

```
POST my_index/my_type?pipeline=my_pipeline_id  
{"key": "value"}
```

Defining a pipeline

- A pipeline defines a series of processors.
- Each processor transforms the document in some way.
- Each processor is executed in the order in which it is defined in the pipeline.
- A pipeline consists of two main fields: a description and a list of processors.

Defining a pipeline

- The typical structure of a pipeline is as follows:

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

- Let's look at an example. As we can see in the following code, we have defined a new pipeline named firstpipeline, which converts the value present in the message field into upper case:

```
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"  
      }  
    }  
  ]  
}'
```

Put pipeline API

Defining a pipeline

- Let's look at an example for this. As we can see in the following code, we have created a new pipeline called secondpipeline that converts the uppercase value present in the message field and renames the message field to data.
- It creates a new field named label with the testlabel value:

https://github.com/fenago/elasticsearch/blob/master/snippets/5_11.txt

Defining a pipeline

- Let's make use of the second pipeline to index a sample document:

```
curl -X PUT  
'http://localhost:9200/myindex/mytype/1?pipeline=secondpi  
peline' -H 'content-type: application/json' -d '{  
    "message": "elk is awesome"  
}'
```

- Let's retrieve the same document and validate the transformation:

```
curl -X GET http://localhost:9200/myindex/mytpe/1 -H 'content-type: application/json'
```

Response:

```
{  
  "_index": "myindex",  
  "_type": "mytpe",  
  "_id": "1",  
  "_version": 1,  
  "found": true,  
  "_source": {  
    "label": "testlabel",  
    "data": "ELK IS AWESOME"  
  }  
}
```

Get pipeline API

- This API is used to retrieve the definition of an existing pipeline.
- Using this API, you can find the details of a single pipeline definition or find the definitions of all the pipelines.
- The command to find the definition of all the pipelines is as follows:

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

Get pipeline API

- To find the definition of an existing pipeline, pass the pipeline ID to the pipeline API.
- The following is an example of finding the definition of the pipeline named secondpipeline:

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

Delete pipeline API

- The delete pipeline API deletes pipelines by ID or wildcard match.
- The following is an example of deleting the pipeline named firstpipeline:

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

Simulate pipeline API

- This pipeline can be used to simulate the execution of a pipeline against the set of documents provided in the body of the request.
- You can either specify an existing pipeline to execute against the provided documents or supply a pipeline definition in the body of the request.

Simulate pipeline API

- The following is an example of simulating an existing pipeline:

```
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"} }
```

```
    ]
```

```
}
```

- The following is an example of a simulated request, with the pipeline definition in the body of the request itself:

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

Summary

- In this lesson, we laid out the foundations of Logstash. We walked you through the steps to install and configure Logstash to set up basic data pipelines, and studied Logstash's architecture.
- We also learned about the ingest node that was introduced in Elastic 5.x, which can be used instead of a dedicated Logstash setup.



COMPLETE LAB 5

6. Building Data Pipelines with Logstash



Building Data Pipelines with Logstash

In this lesson, we will be covering the following topics:

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

Parsing and enriching logs using Logstash

- The analysis of structured data is easier and helps us find meaningful/deeper analysis, rather than trying to perform analysis on unstructured data.
- Most analysis tools depend on structured data, Kibana, which we will be making use of for analysis and visualization, can be used effectively if the data in Elasticsearch is right

Parsing and enriching logs using Logstash

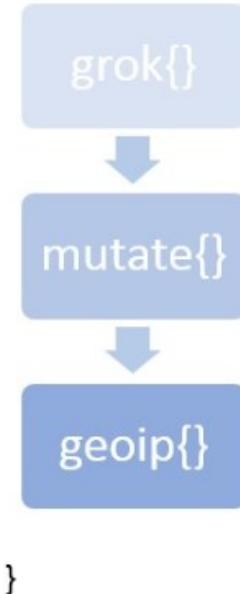
- Log data is typically made up of two parts, as follows:

logdata = timestamp + data

Filter plugins

- A filter plugin is used to perform transformations on data.
- It allows us to combine one or more plugins, and the order of the plugins defines the order in which the data is transformed.
- A sample filter section in a Logstash pipeline would look as follows:

```
filter{
```



CSV filter

- This filter is useful for parsing .csv files.
- This plugin takes an event containing CSV data, parses it, and stores it as individual fields.
- Let's take some sample data and use a CSV filter to parse data out of it.
- Store the following data in a file named 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

- In doing so, you can let the plugin know that it needs to detect column names automatically, as follows:

```
#csv_file.conf
input {
  file{
    path => "D:\es\logs\users.csv"
    start_position => "beginning"
  }
}
filter {
  csv{
    autodetect_column_names => true
  }
}
output {
  stdout {
    codec => rubydebug
  }
}
```

Mutate filter

- You can perform general mutations on fields using this filter.
- The fields in the event can be renamed, converted, stripped, and modified.
- Let's enhance the `csv_file.conf` file we created in the previous section with the mutate filter and understand its usage.
- The following code block shows the use of the mutate filter:

https://github.com/fenago/elasticsearch/blob/master/snippets/6_1.txt

Grok filter

- This is a powerful and often used plugin for parsing the unstructured data into structured data, thus making the data easily queryable/filterable.
- In simple terms, Grok is a way of matching a line against a pattern (which is based on a regular expression) and mapping specific parts of the line to dedicated fields.
- The general syntax of a grok pattern is as follows:

`%{PATTERN:FIELDNAME}`

Grok filter

- By default, groked fields are strings.
- To cast either to float or int values, you can use the following format:

`%{PATTERN:FIELDNAME:type}`

Grok filter

- Patterns consist of a label and a regex. For example:

USERNAME [a-zA-Z0-9._-]+

- Patterns can contain other patterns, too; for example:

HTTPDATE

%{MONTHDAY}/%{MONTH}/%{YEAR}: %{TIME} %{INT}

Grok filter

- If a pattern is not available, then you can use a regular expression by using the following format:

(?<field_name>regex)

Grok filter

- For example, regex (?<phone>\d\d\d-\d\d\d-\d\d\d) would match telephone numbers, such as 123-123-1234, and place the parsed value into the phone field.
- Let's look at some examples to understand grok better:

https://github.com/fenago/elasticsearch/blob/master/snippets/6_2.txt

Grok filter

- If the input line is of the "2017-10-11T21:50:10.000+00:00 tmi_19 001 this is a random message" format, then the output would be as follows:

https://github.com/fenago/elasticsearch/blob/master/snippets/6_3.txt

Date filter

- This plugin is used for parsing the dates from the fields, This plugin is very handy and useful when working with time series events.
- By default, Logstash adds a @timestamp field for each event, representing the time it processed the event.
- But the user might be interested in the actual timestamp of the generated event rather than the processed timestamp.

Date filter

- We can use the plugin like so:

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

Date filter

- The user can keep the event time processed by Logstash, too:

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

Date filter

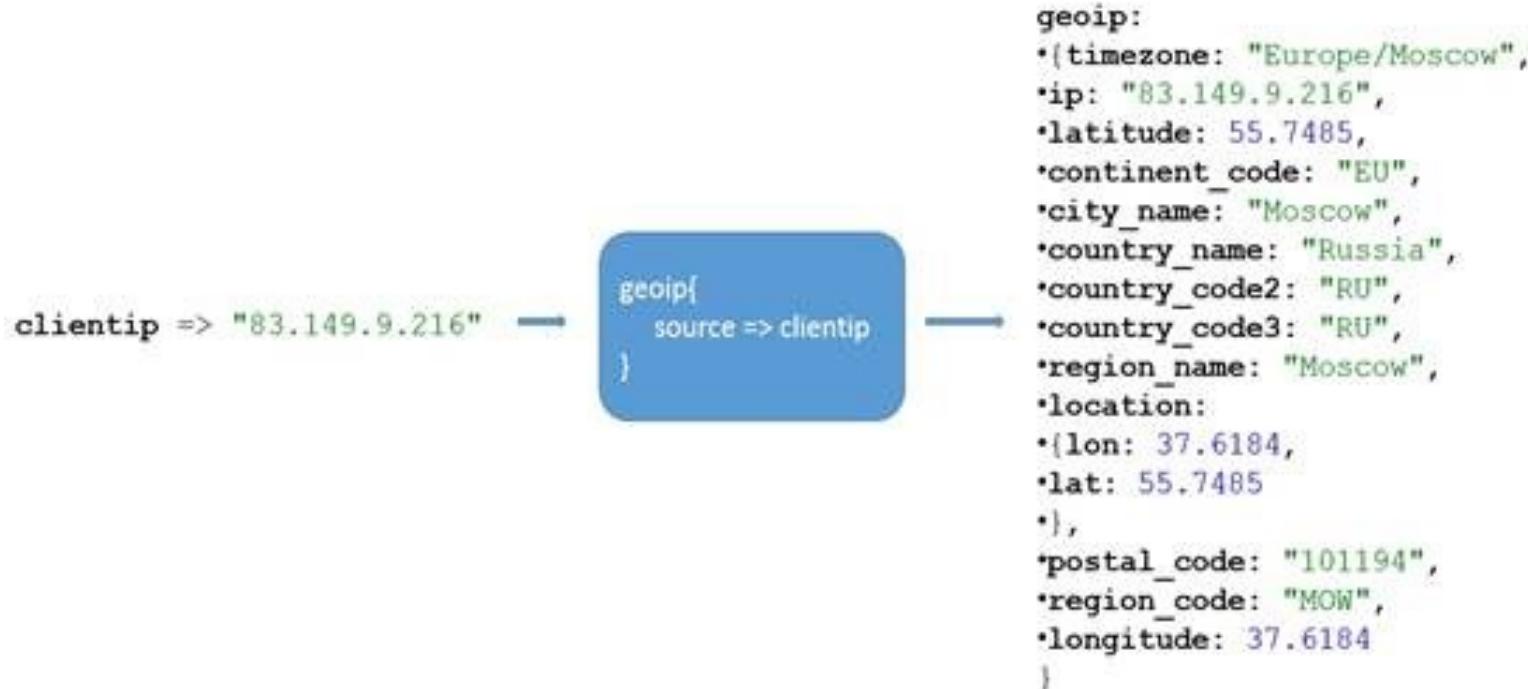
- If the time field has multiple possible time formats, then those can be specified as an array of values to the match parameter:

```
match => [ "eventdate", "dd/MMM/YYYY:HH:mm:ss Z",  
"MMM dd yyyy HH:mm:ss","MMM d yyyy HH:mm:ss",  
"ISO8601" ]
```

Geoip filter

- This plugin is used to enrich the log information.
- Given the IP address, it adds the geographical location of the IP address.
- It finds the geographical information by performing a lookup against the GeoLite2 City database for valid IP addresses and populates fields with results.
- The GeoLite2 City database is a product of the Maxmind organization and is available under the CCA- ShareAlike 4.0 license.

Geoip filter

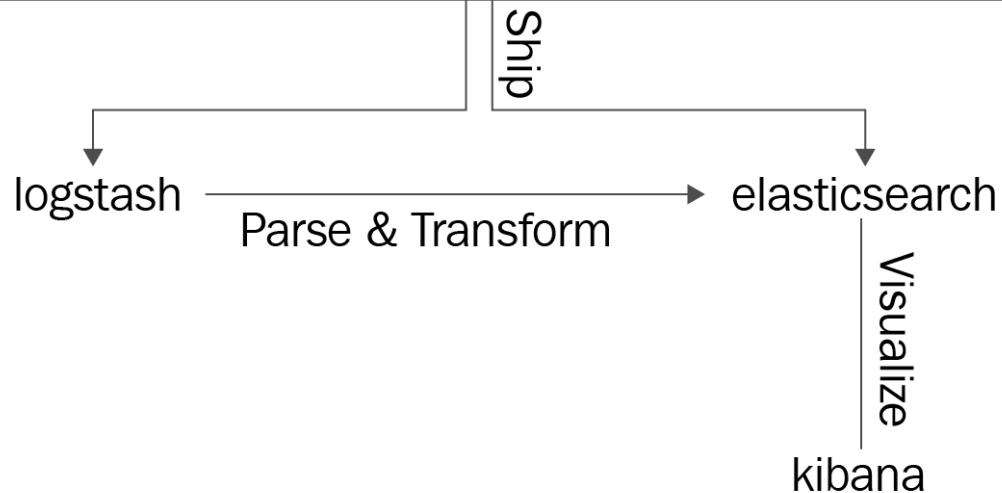
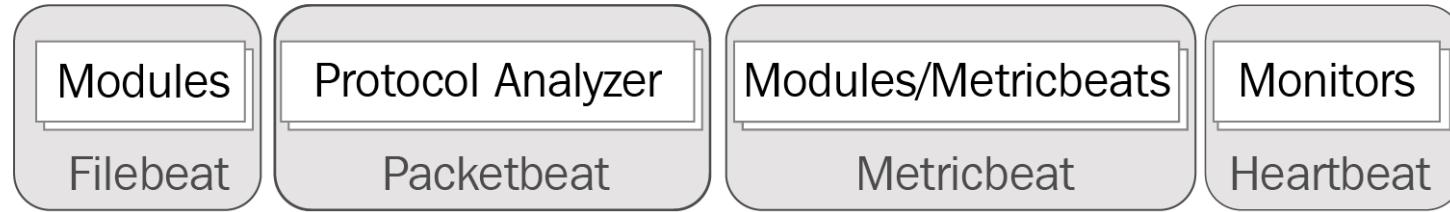


Useragent filter



Introducing Beats

- Beats are lightweight data shippers that are installed as agents on edge servers to ship operational data to Elasticsearch.
- Just like Elasticsearch, Logstash, Kibana, and Beats are open source products too.
- Depending on the use case, Beats can be configured to ship the data to Logstash to transform events prior to pushing the events to Elasticsearch.



Beats by Elastic.co

Let's take a look at some used Beats commonly used by Elastic.co.

- Filebeat
- Metricbeat
- Packetbeat
- Heartbeat
- Winlogbeat
- Auditbeat
- Journalbeat
- Functionbeat



Community Beats

Beat Name	Description
springbeat	Used to collect health and metrics data from Spring Boot applications that are running within the actuator module.
rsbeat	Ships Redis slow logs to Elasticsearch.
nginxbeat	Reads the status from Nginx.
mysqlbeat	Runs any query in MySQL and send the results to Elasticsearch.
mongobeat	It can be configured to send multiple document formats to Elasticsearch. It also monitors mongodb instances.
gabesbeat	Collects data from the Google Analytics Real Time Reporting API.
apachebeat	Reads the status from Apache HTTPD server-status.
dockbeat	Reads docker container statistics and pushes them to Elasticsearch.
kafkabeat	Reads data from kafkatopic.
amazonbeat	Reads data from a specified Amazon product.

Logstash versus Beats

- After reading through the Logstash and Beats introduction, you might be confused as to whether Beats is a replacement for Logstash, the difference between them, or when to use one over the other.
- Beats are lightweight agents and consume fewer resources, and hence are installed on the edge servers where the operational data needs to be collected and shipped.

Filebeat

- Filebeat is an open source, lightweight log shipping agent that is installed as an agent to ship logs from local files.
- It monitors log directories, tails the files, and sends them to Elasticsearch, Logstash, Redis, or Kafka.
- It allows us to send logs from different systems to a centralized server.
- The logs can then be parsed or processed from here.

Downloading and installing Filebeat

The screenshot shows a web browser displaying the Elastic website at <https://www.elastic.co/downloads/beats/filebeat-oss>. The page is titled "Download Filebeat - OSS Only". It features a "Migration Guide" button, version information (7.0.0, April 10, 2019), and a "License" section highlighting "Apache 2.0". Below this, there are download links for various operating systems: DEB 32-BIT sha, RPM 32-BIT sha, LINUX 32-BIT sha, MAC sha, DEB 64-BIT sha, RPM 64-BIT sha, LINUX 64-BIT sha, and WINDOWS 32-BIT sha.

Want to upgrade? We'll give you a hand. [Migration Guide »](#)

Version: 7.0.0

Release date: April 10, 2019

License: [Apache 2.0](#)

Downloads:

- DEB 32-BIT sha
- RPM 32-BIT sha
- LINUX 32-BIT sha
- MAC sha
- DEB 64-BIT sha
- RPM 64-BIT sha
- LINUX 64-BIT sha
- WINDOWS 32-BIT sha
- WINDOWS 64-BIT sha

Installing on Windows

- Unzip the downloaded file and navigate to the extracted location, as follows:

```
E:> cd E:\filebeat-7.0.0-windows-x86_64
```

Installing on Windows

- Run the following commands from the PowerShell prompt to install Filebeat as a Windows service:

```
PS >cd E:\filebeat-7.0.0-windows-x86_64
```

```
PS E:\filebeat-7.0.0-windows-x86_64>.\install-service-filebeat.ps1
```

- In the event script execution is disabled on your system, you will have to set the execution policy for the current session:

```
PowerShell.exe -ExecutionPolicy UnRestricted -File .\install-service-filebeat.ps1
```

Installing on Linux

- Unzip the tar.gz package and navigate to the newly created folder, as follows:

```
$> tar -xzf filebeat-7.0.0-linux-x86_64.tar.gz  
$> cd filebeat
```

- To install using dep or rpm, execute the appropriate commands in the Terminal:
- deb:

```
curl -L -O https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-7.0.0-amd64.deb  
sudo dpkg -i filebeat-7.0.0-amd64.deb
```

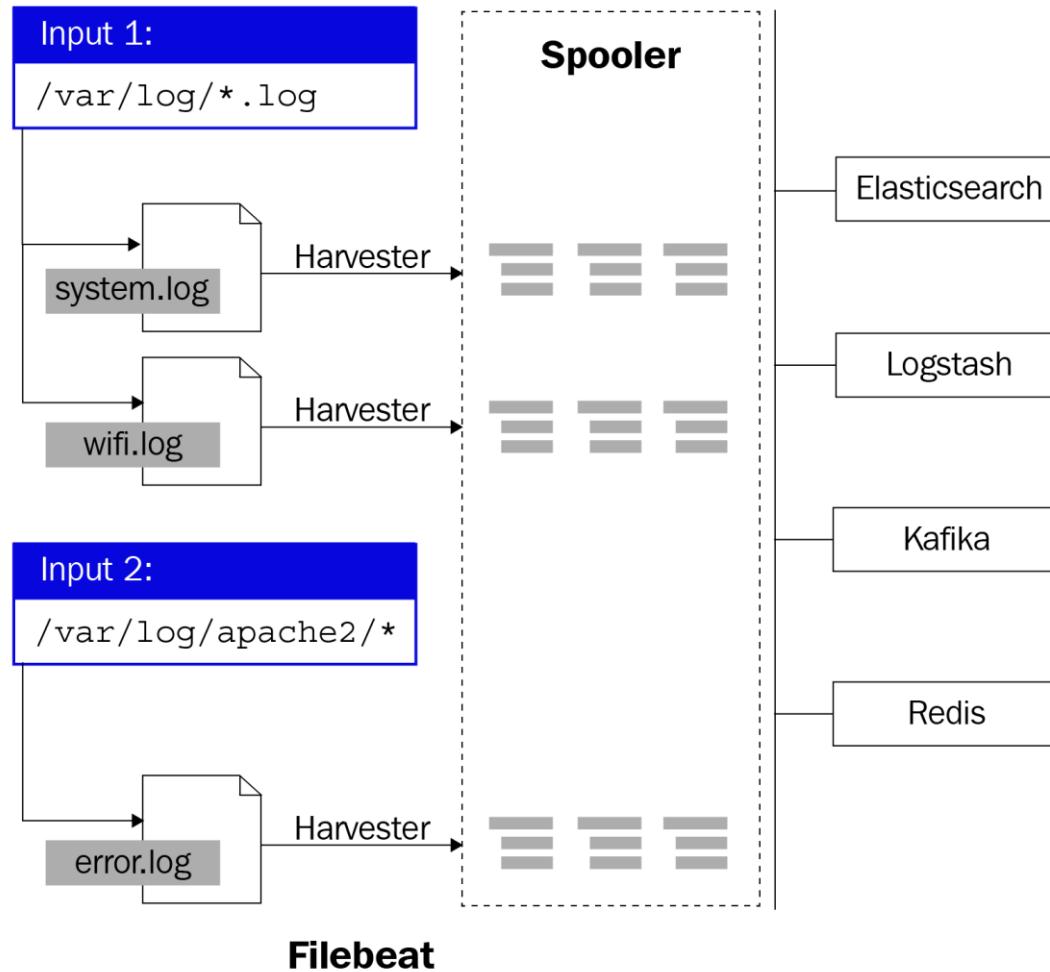
Installing on Linux

- rpm:

```
curl -L -O  
https://artifacts.elastic.co/downloads/beats/filebeat/filebeat  
-7.0.0-x86_64.rpm  
sudo rpm -vi filebeat-7.0.0-x86_64.rpm
```

Architecture

- Filebeat is made up of key components called inputs, harvesters, and spoolers.
- These components work in unison in order to tail files and allow you to send event data to the specified output.
- The input is responsible for identifying the list of files to read logs from.
- The input is configured with one or many file paths, from which it identifies the files to read logs from; it starts a harvester for each file.



Configuring Filebeat

The filebeat.yml file contains the following important sections:

- Filebeat inputs
- Filebeat modules
- Elasticsearch template settings
- Filebeat general/global options
- Kibana dashboard settings
- Output configuration
- Processors configuration
- Logging configuration

Configuring Filebeat

- let's see what a simple configuration would look like.
- As we can see in the following configuration, when Filebeat is started, it looks for files ending with the .log extension in the E:\fenago\logs\ path.
- It ships the log entries of each file to Elasticsearch, which is configured as the output, and is hosted at localhost:9200:

https://github.com/fenago/elasticsearch/blob/master/snippets/6_4.txt

Configuring Filebeat

- Place some log files in E:\fenago\logs\. To get Filebeat to ship the logs, execute the following command:

Windows:

E:\>filebeat-7.0.0-windows-x86_64>filebeat.exe

Linux:

[locationOfFilebeat]\$./filebeat

Configuring Filebeat

- To validate whether the logs were shipped to Elasticsearch, execute the following command:

https://github.com/fenago/elasticsearch/blob/master/snippets/6_5.txt

Filebeat inputs

- A sample configuration is as follows:

```
#----- Filebeat inputs -----  
  
filebeat.inputs:  
  
- type: log  
  # Enable or disable  
  enabled: true  
  paths:  
    - E:\packt\logs\*.log  
    - C:\programdata\elasticsearch\logs\*  
  
  exclude_lines: ['^DBG']  
  include_lines: ['^ERR', '^WARN']  
  exclude_files: ['.gz$']  
  fields:  
    level: error_warn_logs  
  tags: ['eslogs']  
  
  multiline.pattern: '^[[\s]]+'  
  multiline.negate: false  
  multiline.match: after  
  
- type: docker  
  enabled: true  
  containers.path: "/var/lib/docker/containers"  
  containers.stream: "all"  
  containers.ids: "*"  
  tags: ['dockerlogs']
```

Filebeat general/global options

- **registry_file:** It is used to specify the location of the registry file, which is used to maintain information about files, such as the last offset read and whether the read lines are acknowledged by the configured outputs or not.
- The default location of the registry is \${path.data}/registry:

`filebeat.registry_file: /etc/filebeat/registry`

Filebeat general/global options

- **shutdown_timeout:** This setting specifies how long Filebeat waits on shutdown for the publisher to finish.
- This ensures that if there is a sudden shutdown while filebeat is in the middle of sending events, it won't wait for the output to acknowledge all events before it shuts down.
- Hence, the filebeat waits for a certain time before it actually shuts down:

`filebeat.shutdown_timeout: 10s`

Filebeat general/global options

- **registry_flush:** This setting specifies the time interval when registry entries are to be flushed to the disk:
`filebeat.registry_flush: 5s`
- **name:** The name of the shipper that publishes the network data. By default, hostname is used for this field:
`name: "dc1-host1"`

Filebeat general/global options

- **tags:** The list of tags that will be included in the tags field of every event Filebeat ships.
- Tags make it easy to group servers by different logical properties and aids filtering of events in Kibana and Logstash:

`tags: ["staging", "web-tier", "dc1"]`

- **max_procs:** The maximum number of CPUs that can be executed simultaneously.
- The default is the number of logical CPUs available in the system:

`max_procs: 2`

Output configuration

- **Elasticsearch:** It is used to send the events directly to Elasticsearch.
- A sample Elasticsearch output configuration is as follows:

```
output.elasticsearch:  
  enabled: true  
  hosts: ["localhost:9200"]
```

Output configuration

- If Elasticsearch is secure, then the credentials can be passed using the username and password settings:

```
output.elasticsearch:  
  enabled: true  
  hosts: ["localhost:9200"]  
  username: "elasticuser"  
  password: "password"
```

Output configuration

- To ship an event to the Elasticsearch ingest node pipeline so that it can be preprocessed before it is stored in Elasticsearch, the pipeline information can be provided using the pipeline setting:

```
output.elasticsearch:  
  enabled: true  
  hosts: ["localhost:9200"]  
  pipeline: "apache_log_pipeline"
```

Output configuration

- **logstash:** This is used to send events to Logstash.
- A sample Logstash output configuration is as follows:

```
output.logstash:  
  enabled: true  
  hosts: ["localhost:5044"]
```

Output configuration

- To enable load balancing of events across the Logstash hosts, use the `loadbalance` flag, set to true:

```
output.logstash:  
hosts: ["localhost:5045", "localhost:5046"]  
loadbalance: true
```

Output configuration

- **console:** This is used to send the events to stdout. The events are written in JSON format. It is useful during debugging or testing.
- A sample console configuration is as follows:

```
output.console:  
  enabled: true  
  pretty: true
```

Logging

- A sample configuration is as follows:

```
logging.level: debug
logging.to_files: true
logging.files:
  path: C:\logs\filebeat
  name: metricbeat.log
  keepfiles: 10
```

Filebeat modules

A module is made up of one or more filesets. A fileset is made up of the following:

- Filebeat input configurations that contain the default paths needed to look out for logs. It also provides configuration for combining multiline events when needed.
- An Elasticsearch Ingest pipeline definition to parse and enrich logs.
- Elasticsearch templates, which define the field definitions so that appropriate mappings are set to the fields of the events.
- Sample Kibana dashboards, which can be used for visualizing logs.

Filebeat modules

- Since each module comes with the default configuration, make the appropriate changes in the module configuration file.
- The basic configuration for the redis module is as follows:

https://github.com/fenago/elasticsearch/blob/master/snippets/6_6.txt

Filebeat modules

- To enable modules, execute the modules enable command, passing one or more module names:

Windows:

```
E:\filebeat-7.0.0-windows-x86_64>filebeat.exe modules enable  
redis mysql
```

Linux:

```
[locationOfFileBeat]$./filebeat modules enable redis mysql
```

Filebeat modules

- To disable modules, execute the modules disable command, passing one or more module names to it. For example:

Windows:

```
E:\filebeat-7.0.0-windows-x86_64>filebeat.exe modules disable  
redis mysql
```

Linux:

```
[locationOfFileBeat]$./filebeat modules disable redis mysql
```

Filebeat modules

- Once the module is enabled, to load the recommended index template for writing to Elasticsearch, and to deploy sample dashboards for visualizing data in Kibana, execute the `setup` command, as follows:

Windows:

```
E:\filebeat-7.0.0-windows-x86_64>filebeat.exe -e setup
```

Linux:

```
[locationOfFileBeat]$./filebeat -e setup
```

Filebeat modules

- Rather than enabling the modules by passing them as command-line parameters, you can enable the modules in the filebeat.yml configuration file itself, and start Filebeat as usual:

```
filebeat.modules:  
-module: nginx  
-module: mysql
```

Filebeat modules

- For the configuration file, use the following code:

```
filebeat.modules:  
-module: nginx  
access:  
  var.paths: ["C:\nginx\access.log*"]
```

Filebeat modules

- For the command line, use the following code:

Windows:

```
E:\filebeat-7.0.0-windows-x86_64>filebeat.exe-e-
modules=nginx-
M"nginx.access.var.paths=[C:\nginx\access.log*]"
```

Linux:

```
[locationOfFileBeat]$./filebeat-e-modules=nginx-
M"nginx.access.var.paths=[\var\nginx\access.log*]"
```



Summary

- In this lesson, we covered the powerful filter section of Logstash, which can be used for parsing and enriching log events.
- We have also covered some commonly used filter plugins.
- Then, we covered the Beats framework and took an overview of various Beats, including Filebeat, Heartbeat, Packetbeat, and so on, covering Filebeat in detail.

COMPLETE LAB 6

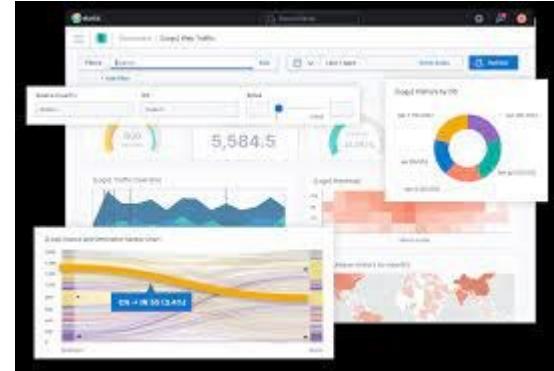
7. Visualizing Data with Kibana



Visualizing Data with Kibana

We will cover the following topics in this lesson:

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



Downloading and installing Kibana

- Navigate to <https://www.elastic.co/downloads/kibana-oss> and, depending on your operating system, download the ZIP/TAR file, as shown in the following screenshot:

The screenshot shows a web browser displaying the Elastic website at <https://www.elastic.co/downloads/kibana-oss>. The page title is "Download Kibana - OSS Only". It features a "Migration Guide" link, version information (7.0.0), release date (April 10, 2019), and a highlighted Apache 2.0 license. It also lists download links for Windows, Linux, and RPM packages, as well as Docker container support. A note at the bottom mentions the Apache 2.0 license and a link to detailed release notes.

Want to upgrade? We'll give you a hand. [Migration Guide »](#)

Version: 7.0.0

Release date: April 10, 2019

License: **Apache 2.0**

Downloads: [WINDOWS sha](#) [MAC sha](#)
[LINUX 64-BIT sha](#) [RPM 64-BIT sha](#)
[DEB 64-BIT sha](#)

Containers: Run with [Docker](#)

Notes: This distribution only includes features licensed under the Apache 2.0 license. To get access to full [set of free features](#), use the [default distribution](#).

[View the detailed release notes here.](#)

Installing on Windows

- Unzip the downloaded file. Once unzipped, navigate to the newly created folder, as shown in the following line of code:

```
E:\>cd kibana-7.0.0-windows-x86_64
```

- To start Kibana, navigate to the bin folder, type kibana.bat, and press Enter.

Installing on Linux

- Unzip the tar.gz package and navigate to the newly created folder, shown as follows:

```
$> tar -xzf kibana-oss-7.0.1-linux-x86_64.tar.gz  
$> cd kibana-7.0.1-linux-x86_64/
```

Downloading and installing Kibana

- To start Kibana, navigate to the bin folder, type ./kibana (in the case of Linux) or kibana.bat (in the case of Windows), and press Enter.
- You should get the following logs:

https://github.com/fenago/elasticsearch/blob/master/snippets/7_1.txt

localhost:5601/status#?_g=0

Server Status

Kibana status is Green

MADSH01-APM01

1.42 GB Heap total	71.10 MB Heap used	0.00, 0.00, 0.00 Load
241.33 ms Response time avg	1972.00 ms Response time max	1.80 Requests per second

Plugin status

BUILD 23117 COMMIT ee89fda8

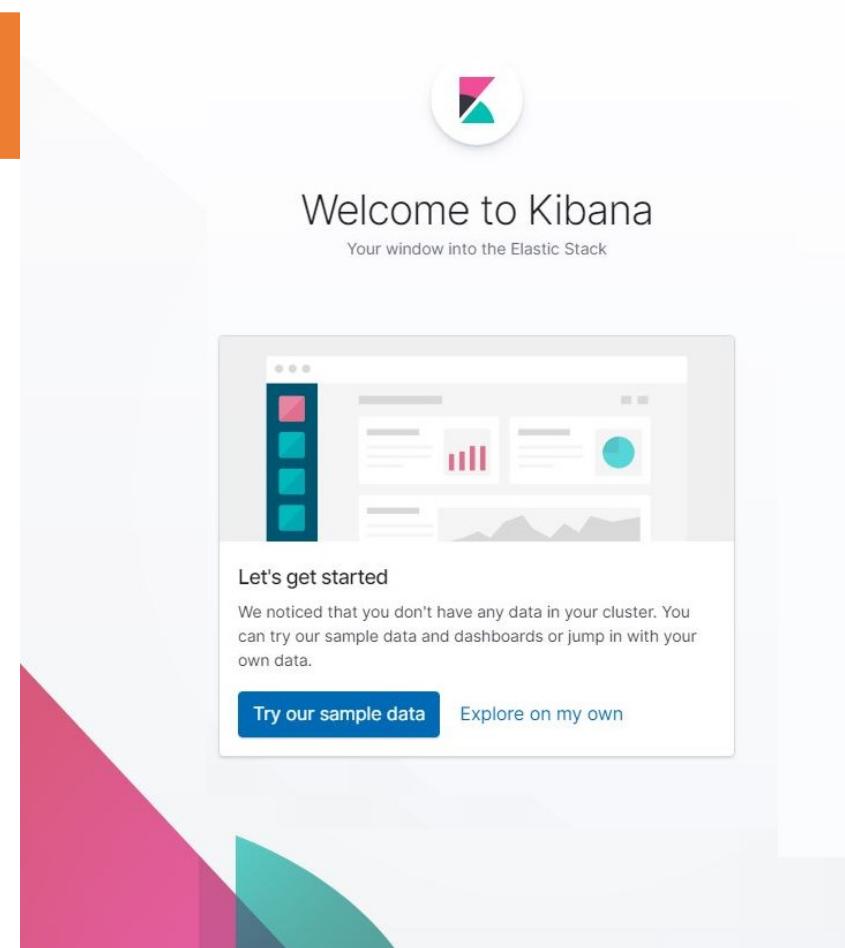
ID	Status
plugin:apm_oss@undefined	Ready
plugin:kibana@undefined	Ready
plugin:elasticsearch@undefined	Ready
plugin:interpreter@undefined	Ready
plugin:metrics@undefined	Ready
plugin:console@undefined	Ready
plugin:timelion@undefined	Ready
plugin:tile_map@undefined	Ready

Configuring Kibana

server.port	This setting specifies the port Kibana will be serving requests on. It defaults to <code>5601</code> .
server.host	This specifies the address to which the Kibana server will bind. IP addresses and hostnames are both valid values. It defaults to <code>localhost</code> .
elasticsearch.url	This is the URL of the Elasticsearch instance to use for all your queries. It defaults to <code>http://localhost:9200</code> . 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 <code>.kibana</code> .

Preparing data

- When you launch Kibana by accessing the <http://localhost:5601> link in your browser for the first time, you will see the following screen:



Preparing data

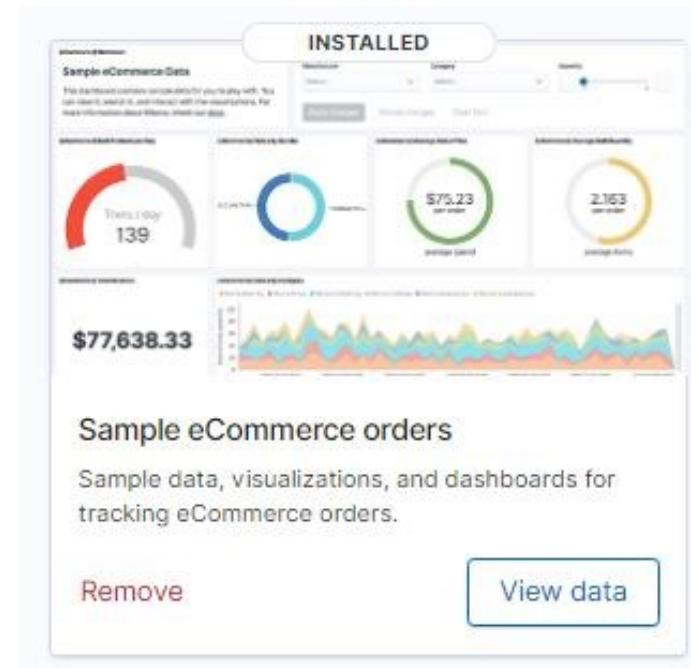
- Clicking on the Try our sample data button will take you to the following screen:

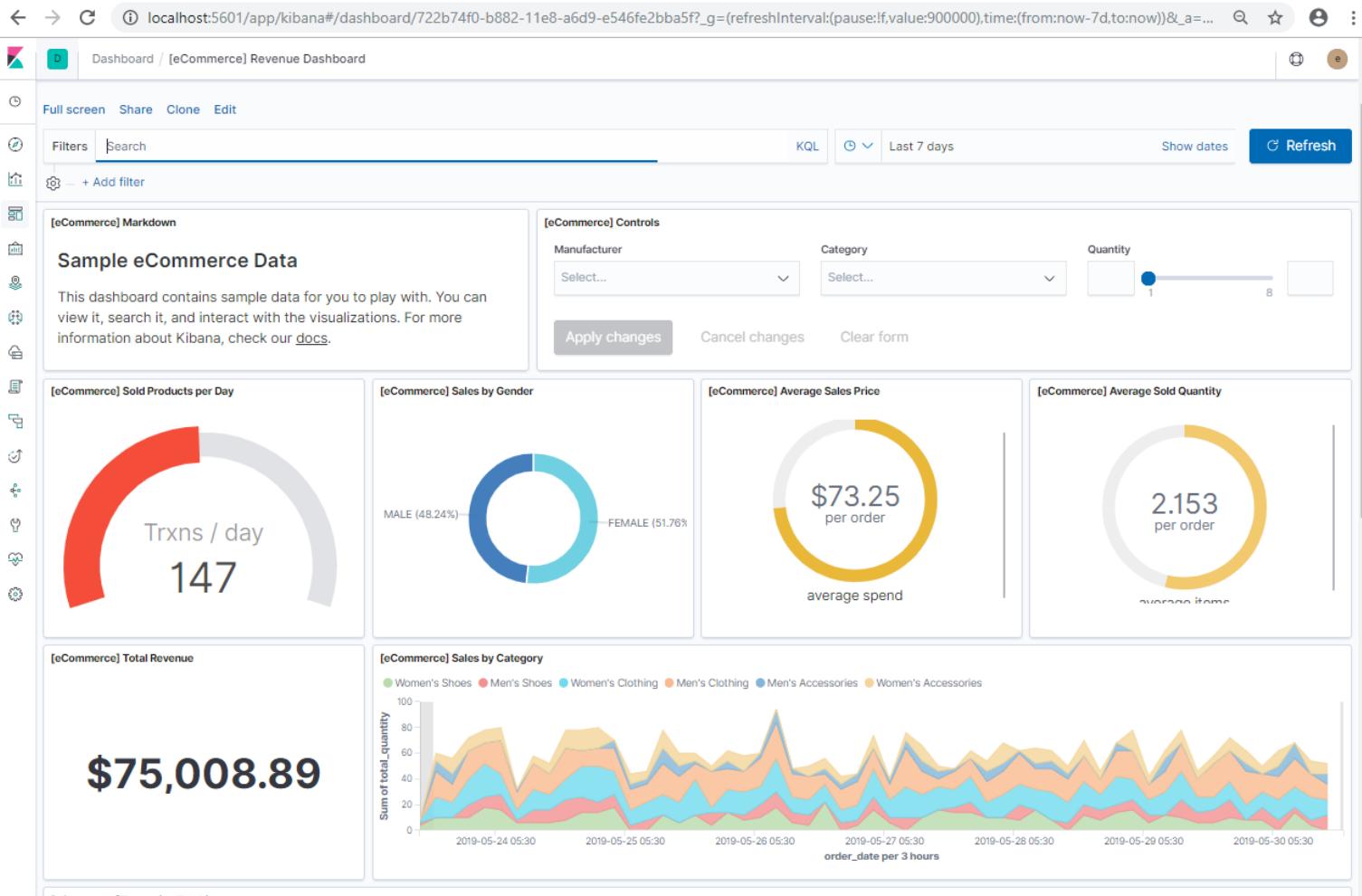
The screenshot shows the Kibana interface with the URL `localhost:5601/app/kibana#/home/tutorial_directory/sampleData?_g=()`. The left sidebar has icons for Home, Add data, Settings, and Help. The main title is "Add Data to Kibana". Below it are tabs: All, Logging, Metrics, Security analytics, and Sample data (which is selected). Three cards are displayed:

- Sample eCommerce orders**: Includes a summary of 139 transactions, a donut chart for payment method distribution, and a line chart for total sales (\$77,638.33). A button labeled "Add data" is at the bottom.
- Sample flight data**: Includes a summary of 313 flights, a donut chart for destination, and a line chart for total price (\$596). A button labeled "Add data" is at the bottom.
- Sample web logs**: Includes a summary of 801 log entries, a donut chart for status code distribution, and a line chart for error count. A button labeled "Add data" is at the bottom.

Preparing data

- Once ready, you can click on View data, which will take you to the eCommerce dashboard:

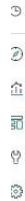




Preparing data

- The actual data that is powering these dashboards/visualizations can be verified in Elasticsearch by executing the following command.
- As seen, the sample data is loaded into the kibana_sample_data_ecommerce index, which has 4675 docs:

https://github.com/fenago/elasticsearch/blob/master/snippets/7_2.txt



Add Data to Kibana

Use these solutions to quickly turn your data into pre-built dashboards and monitoring systems.



APM

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



Logging

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



Metrics

Collect metrics from the operating system and services running on your servers.



Security analytics

Centralize security events for interactive investigation in ready-to-go visualizations.

[Add APM](#)[Add log data](#)

3

[Add metric data](#)[Add security events](#)[Add sample data](#)

1

Load a data set and a Kibana dashboard

[Use Elasticsearch data](#)

2

Connect to your Elasticsearch index

Visualize and Explore Data



Dashboard

Display and share a collection of visualizations and saved searches.



Visualize

Create visualizations and aggregate data stores in your Elasticsearch indices.



Discover

Interactively explore your data by querying and filtering raw documents.

Manage and Administer the Elastic Stack



Console

Skip cURL and use this JSON interface to work with your data directly.



Index Patterns

Manage the index patterns that help retrieve your data from Elasticsearch.



Saved Objects

Import, export, and manage your saved searches, visualizations, and dashboards.

Didn't find what you were looking for?

[View full directory of Kibana plugins](#)

Preparing data

- Navigate to <https://github.com/elastic/elk-index-size-tests/blob/master/logs.gz> and click the Download button. Unzip the logs.gz file and place it in a folder
- Make sure you have Logstash version 7.0 or above installed.
- Create a config file named apache.conf in the \$LOGSTASH_HOME\bin folder, as shown in the following code block:

https://github.com/fenago/elasticsearch/blob/master/snippets/7_3.txt

Preparing data

- Start Logstash, shown as follows, so that it can begin processing the logs, and index them to Elasticsearch.
- Logstash will take a while to start and then you should see a series of dots (a dot per processed log line):

```
$LOGSTASH_HOME\bin>logstash -f apache.conf
```

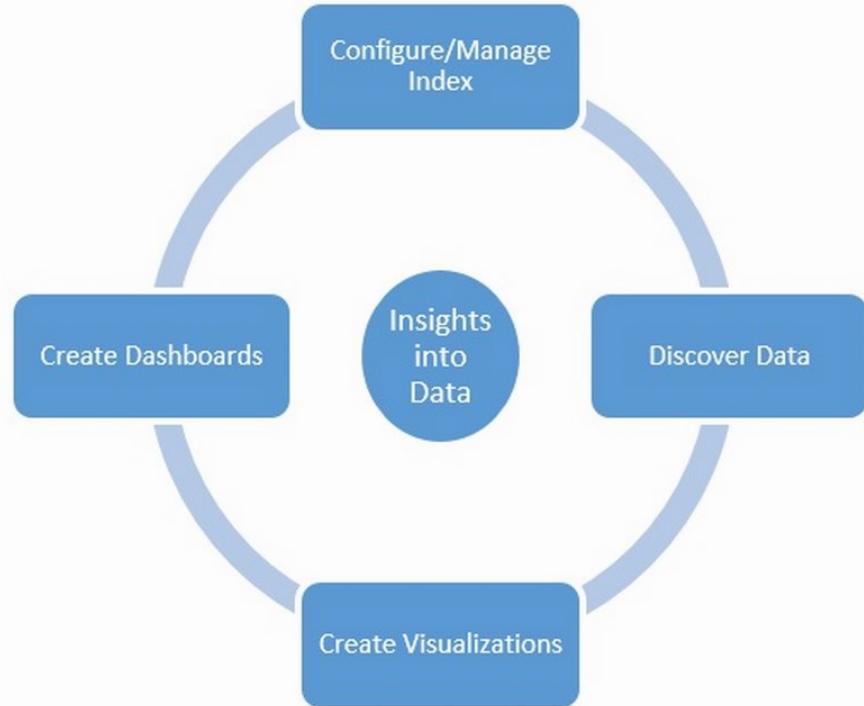
Preparing data

- Let's verify the total number of documents (log events) indexed into Elasticsearch:

```
curl -X GET http://localhost:9200/logstash-*/_count
```

User interaction

- Let's understand user interaction before diving into the core components of Kibana.
- A typical user interaction flow is as depicted in the following diagram:



Configuring the index pattern

An index pattern is a string with optional wildcards that can match multiple indices. Typically, two types of index exist within Elasticsearch:

- Time-series indexes
- Regular indexes

Create index pattern

No default index pattern. You must select or create one to continue.

Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

X Include system indices

Step 1 of 2: Define index pattern

Index pattern

logstash-*

You can use a * as a wildcard in your index pattern.

You can't use spaces or the characters \, /, ?, ., <, >, |.

✓ Success! Your index pattern matches 31 indices.

logstash-2014.05.28

logstash-2014.05.29

logstash-2014.05.30

logstash-2014.05.31

logstash-2014.06.01

logstash-2014.06.02

logstash-2014.06.03

logstash-2014.06.04

logstash-2014.06.05

logstash-2014.06.06

Rows per page: 10 ▾

> Next step

< 1 2 3 4 >

Configuring the index pattern

Create index pattern

No default index pattern. You must select or create one to continue.

Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

X Include system indices

Step 2 of 2: Configure settings

You've defined **logstash-*** as your index pattern. Now you can specify some settings before we create it.

Time Filter field name

Refresh

@timestamp

The Time Filter will use this field to filter your data by time.
You can choose not to have a time field, but you will not be able to
narrow down your data by a time range.

> Show advanced options

< Back

Create index pattern

[Create index pattern](#)

★ logstash-*

★ logstash-*

Time Filter field name: @timestamp



This page lists every field in the **logstash-*** index and the field's associated core type as recorded by Elasticsearch. To change a field type, use the Elasticsearch Mapping API

Fields (78)

Scripted fields (0)

Source filters (0)

Q Filter

All field types ▾

Name	Type	Format	Searchable	Aggregatable	Excluded
@timestamp	date		●	●	
@version	string		●	●	
_id	string		●	●	
_index	string		●	●	
_score	number				
_source	_source				
_type	string		●	●	
agent	string		●		
agent.keyword	string		●	●	
auth	string		●		

Rows per page: 10 ▾

< 1 2 3 4 5 ... 8 >

Discover

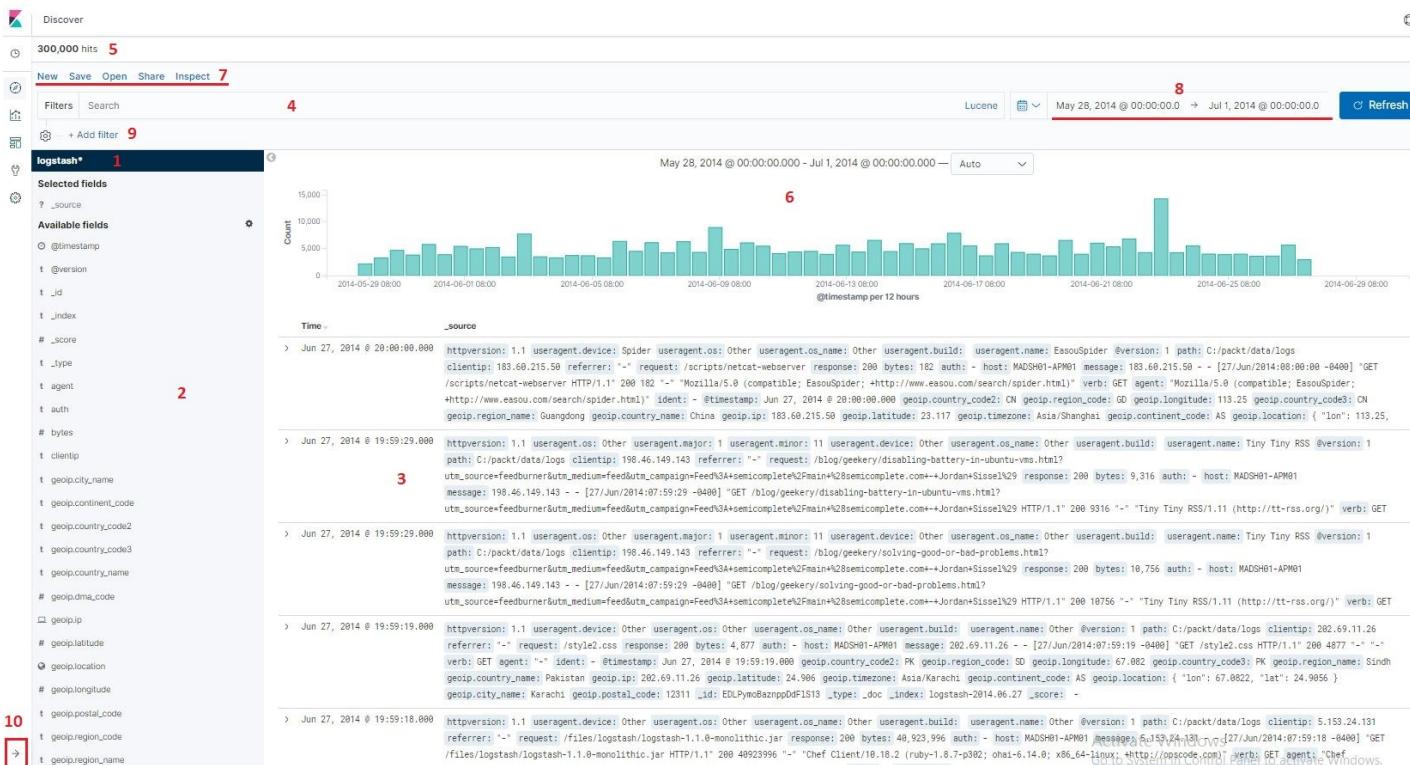
- The Discover page helps you to interactively explore data.
- It allows the user to interactively perform search queries, filter search results, and view document data. It also allows the user to save the search, or filter criteria so that it can be reused or used to create visualizations on top of the filtered results.
- Clicking on the third icon from the top-left takes you to the Discover page.

Discover

- Click Update, as shown in the following screenshot:

The screenshot shows the Elasticsearch Discover interface. At the top, there are navigation links: New, Save, Open, Share, and Inspect. Below these are buttons for Filters, us, and + Add filter. A sidebar on the left lists Selected fields (logstash-*), Available fields, and a _source link. The main search bar at the top has a placeholder "May 28, 2014 @ 00:00:00.000 → Jul 1, 2014 @ 00:00:00.000". To the right of the search bar is a green "Update" button. Below the search bar, a message says "No results match your search criteria". Underneath this message, there are two sections: "Expand your time range" and "Refine your query". The "Expand your time range" section contains the text: "One or more of the indices you're looking at contains a date field. Your query may not return any results because there may not be any data at all in the currently selected time range. You can try changing the time range or refining your query." The "Refine your query" section contains the text: "The search bar at the top uses Elasticsearch's support for Lucene Query String syntax." A date range picker is open, showing July 2014. The date 1 is highlighted in blue. The time dropdown shows "12:00 AM" with other options like 12:30 AM, 01:00 AM, etc. At the bottom of the date range picker, there is a red box highlighting the input field containing "2014-07-01 00:00:00.000".

- The Discover page contains the sections shown in the following screenshot:



Discover



Discover

Time	_source
Expand Button	
🕒 June 27th 2014, 17:43:20.000	<pre>request: /scripts/netcat-webserver agent: "Mozilla/5.0 (compatible; EasouSpider; +http://www.easou.com/search/spider.html)" geoip.ci geoip.timezone: Asia/Shanghai geoip.ip: 183.60.215.50 geoip.latitude: 23.117 geoip.country_name: China geoip.country_code2: CN geoip.co geoip.country_code3: CN geoip.region_name: Guangdong geoip.location: { "lon": 113.25, "lat": 23.1167 } geoip.region_code: 44 geoip.long ident: - verb: GET useragent.os: Other useragent.build: useragent.name: EasouSpider useragent.os_name: Other useragent.device: Spider -- [27/Jun/2014:08:00:00 -0400] "GET /scripts/netcat-webserver HTTP/1.1" 200 182 "-" "Mozilla/5.0 (compatible; EasouSpider; +http://</pre>
Table	JSON
🕒 @timestamp	⌚ ⓘ * June 27th 2014, 17:43:20.000
t @version	⌚ ⓘ * 1
t _id	⌚ ⓘ * AV4jH1xYxVeTbjX4rA1W
t _index	⌚ ⓘ * logstash-2014.06.27
# _score	⌚ ⓘ * -
t _type	⌚ ⓘ * logs
t agent	⌚ ⓘ * "Mozilla/5.0 (compatible; EasouSpider; +http://www.easou.com/search/spider.html)"
t auth	⌚ ⓘ * -
# bytes	⌚ ⓘ * 182
t clientip	⌚ ⓘ * 183.60.215.50
t geoip.city_name	⌚ ⓘ * Guangzhou
t geoip.continent_code	⌚ ⓘ * AS
t geoip.country_code2	⌚ ⓘ * CN

Discover

- In order to add fields to the document table, either hover over the field on the fields list and click its add button, or expand the document and click the field's Toggle column in table button:

The screenshot shows the Apache Solr interface. At the top, there is a search bar with the query "geoip.country_name" and a dropdown menu showing "China". Below the search bar, the results are displayed in a table. The first row, "geoip.country_name", is expanded, showing its value "China". The second row, "geoip.ip", is collapsed. A tooltip with the text "Toggle column in table" has an arrow pointing to the "geoip.ip" row.

t	geoip.country_name	Q	Q	□	*	China
□	geoip.ip	Toggle column in table				

Time ▾	geoip.city_name	response	request ✖
▶ June 27th 2014, 17:43:20.000	Guangzhou	200	Move column to the left ↗
▶ June 27th 2014, 17:42:49.000	Buffalo	200	/blog/geekery/solving-good-or-bad-problems.html? utm_source=feedburner&utm_medium=feed&utm_campaign=Fe
▶ June 27th 2014, 17:42:49.000	Buffalo	200	/blog/geekery/disabling-battery-in-ubuntu-vms.html? utm_source=feedburner&utm_medium=feed&utm_campaign=Fe
▶ June 27th 2014, 17:42:39.000	-	200	/style2.css
▶ June 27th 2014, 17:42:38.000	Amsterdam	200	/files/logstash/logstash-1.1.0-monolithic.jar
▶ June 27th 2014, 17:42:37.000	-	200	/images/jordan-80.png
▶ June 27th 2014, 17:42:35.000	-	200	/reset.css
▶ June 27th 2014, 17:42:30.000	-	200	/blog/tags/X11
▶ June 27th 2014, 17:42:12.000	-	200	/images/googledotcom.png

Elasticsearch query string/Lucene query

The screenshot shows a portion of the Kibana interface. At the top right, there is a small box labeled "Lucene" with a red border. Below it, the section title "SYNTAX OPTIONS" is centered. A text block explains that the Kibana Query Language (KQL) offers simplified query syntax and support for scripted fields. It notes that KQL also provides autocomplete if a Basic license or above is present. If KQL is turned off, Kibana uses Lucene. At the bottom, there is a toggle switch for "Kibana Query Language" which is currently set to "Off".

The [Kibana Query Language](#) (KQL) offers a simplified query syntax and support for scripted fields. KQL also provides autocomplete if you have a Basic license or above. If you turn off KQL, Kibana uses Lucene.

Kibana Query Language
 X Off

88,602 hits

New Save Open Share Inspect

Filters files logstash

Lucene



May 28, 2014 @ 00:00:00.000 → Jul 1, 2014 @ 00:00:00.000

Refresh

logstash-*

Selected fields

? _source

Available fields

@timestamp

t _version

t _id

t _index

_score

t _type

t agent

t auth

bytes

t clientip

t geoip.city_name

t geoip.continent_code

t geoip.country_code2

t geoip.country_code3

t geoip.country_name

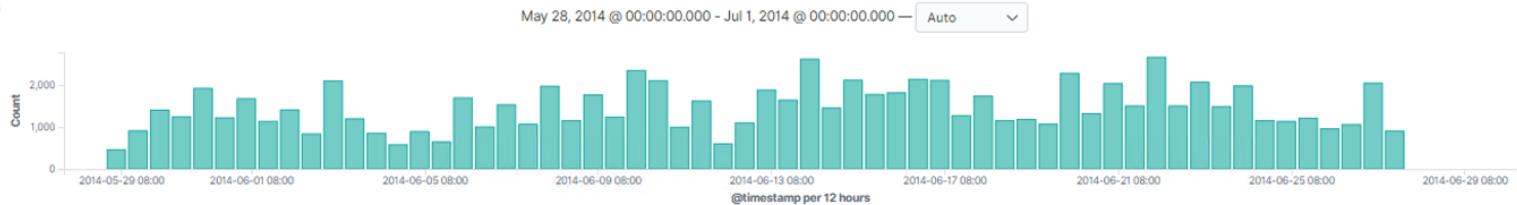
geoip.dma_code

□ geoip.ip

geoip.latitude

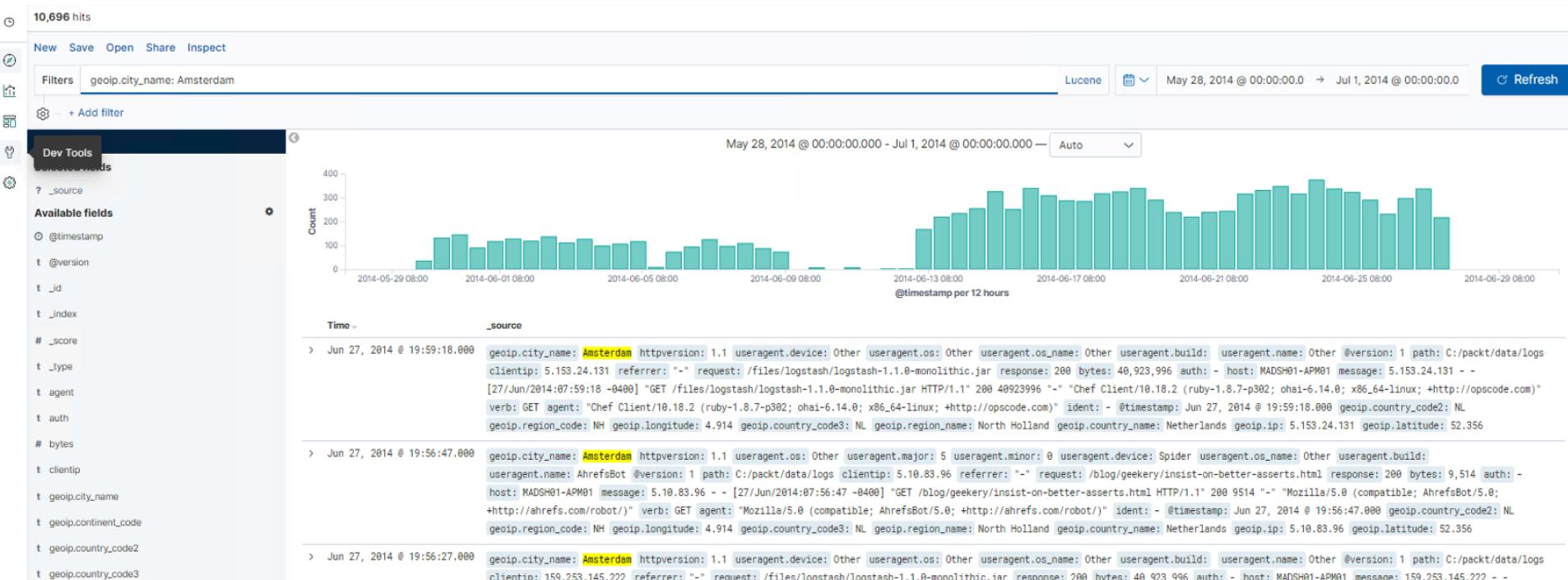
geoip.location

geoip.longitude

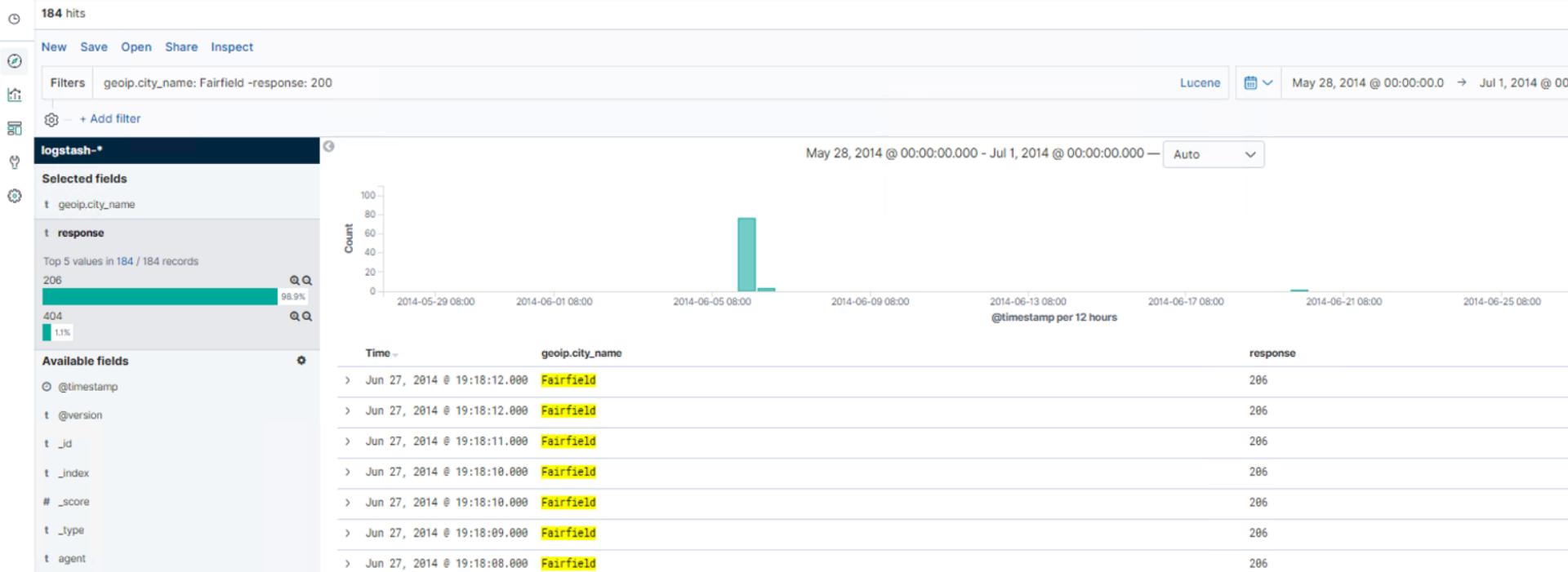


Time	_source
> Jun 27, 2014 @ 19:59:18.000	request: /files/logstash/logstash-1.1.0-monolithic.jar message: 5.153.24.131 - - [27/Jun/2014:07:59:18 -0400] "GET /files/logstash/logstash-1.1.0-monolithic.jar HTTP/1.1" 200 40923996 "-" "Chef Client/10.18.2 (ruby-1.8.7-p302; ohai-6.14.0; x86_64-linux; +http://opscode.com)" httpversion: 1.1 useragent.device: Other useragent.os: Other useragent.os_name: Other useragent.build: useragent.name: Other @version: 1 path: C:/packt/data/logs Clientip: 5.153.24.131 referer: "-" response: 200 bytes: 40,923,996 auth: - host: MADSH01-APM01 verb: GET agent: "Chef Client/10.18.2 (ruby-1.8.7-p302; ohai-6.14.0; x86_64-linux; +http://opscode.com)" ident: - @timestamp: Jun 27, 2014 @ 19:59:18.000 geoip.country_code2: NL geoip.region_code: NH geoip.longitude: 4.914 geoip.country_code3: NL geoip.region_name: North Holland geoip.country_ip: 5.153.24.131 geoip.latitude: 52.356 geoip.timezone: Europe/Amsterdam
> Jun 27, 2014 @ 19:58:16.000	request: /files/logstash/logstash-1.1.0-monolithic.jar message: 173.192.221.212 - - [27/Jun/2014:07:58:16 -0400] "GET /files/logstash/logstash-1.1.0-monolithic.jar HTTP/1.1" 200 40923996 "-" "Chef Client/10.18.2 (ruby-1.8.7-p302; ohai-6.14.0; x86_64-linux; +http://opscode.com)" httpversion: 1.1 useragent.device: Other useragent.os: Other useragent.os_name: Other useragent.build: useragent.name: Other @version: 1 path: C:/packt/data/logs Clientip: 173.192.221.212 referer: "-" response: 200 bytes: 40,923,996 auth: - host: MADSH01-APM01 verb: GET agent: "Chef Client/10.18.2 (ruby-1.8.7-p302; ohai-6.14.0; x86_64-linux; +http://opscode.com)" ident: - @timestamp: Jun 27, 2014 @ 19:58:16.000 geoip.country_code2: US geoip.region_code: VA geoip.longitude: -77.446 geoip.country_code3: US geoip.region_name: Virginia geoip.country_name: United States geoip.ip: 173.192.221.212 geoip.latitude: 38.887 geoip.timezone: America/New_York
> Jun 27, 2014 @ 19:56:27.000	request: /files/logstash/logstash-1.1.0-monolithic.jar message: 159.253.145.222 - - [27/Jun/2014:07:56:27 -0400] "GET /files/logstash/logstash-1.1.0-monolithic.jar HTTP/1.1" 200 40923996 "-" "Chef Client/10.18.2 (ruby-1.8.7-p302; ohai-6.14.0; x86_64-linux; +http://opscode.com)" httpversion: 1.1 useragent.device: Other useragent.os: Other useragent.os_name: Other useragent.build: useragent.name: Other @version: 1 path: C:/packt/data/logs Clientip: 159.253.145.222 referer: "-" response: 200 bytes: 40,923,996 auth: - host: MADSH01-APM01 verb: GET agent: "Chef Client/10.18.2 (ruby-1.8.7-p302; ohai-6.14.0; x86_64-linux; +http://opscode.com)" ident: - @timestamp: Jun 27, 2014 @ 19:56:27.000 geoip.country_code2: NL geoip.region_code: NH geoip.longitude: 4.914 geoip.country_code3: NL geoip.region_name: North Holland geoip.country_name: Netherlands geoip.ip: 159.253.145.222 geoip.latitude: 52.356
> Jun 27, 2014 @ 19:54:16.000	request: /files/xdotool/docs/man?C=N;O=D message: 5.10.83.95 - - [27/Jun/2014:07:54:16 -0400] "GET /files/xdotool/docs/man?C=N;O=D HTTP/1.1" 200 955 "-" "Mozilla/5.0 (compatible; AhrefsBot/5.0; +http://ahrefs.com/robot/)" httpversion: 1.1 useragent.os: Other useragent.major: 5 useragent.minor: 0 useragent.device: Spider useragent.os_name: Other useragent.build: useragent.name: AhrefsBot @version: 1 path: C:/packt/data/logs Clientip: 5.10.83.95 referer: "-" response: 200 bytes: 955 auth: - host: MADSH01-APM01 verb: GET agent: "Mozilla/5.0 (compatible; AhrefsBot/5.0; +http://ahrefs.com/robot/)" ident: - @timestamp: Jun 27, 2014 @ 19:54:16.000 geoip.country_code2: NL geoip.region_code: NH geoip.longitude: 4.914 geoip.country_code3: NL geoip.region_name: North Holland geoip.country_name: Netherlands geoip.ip: 5.10.83.95 geoip.latitude: 52.356 geoip.timezone: Europe/Amsterdam geoip.continent_code: EU

Elasticsearch query string/Lucene query

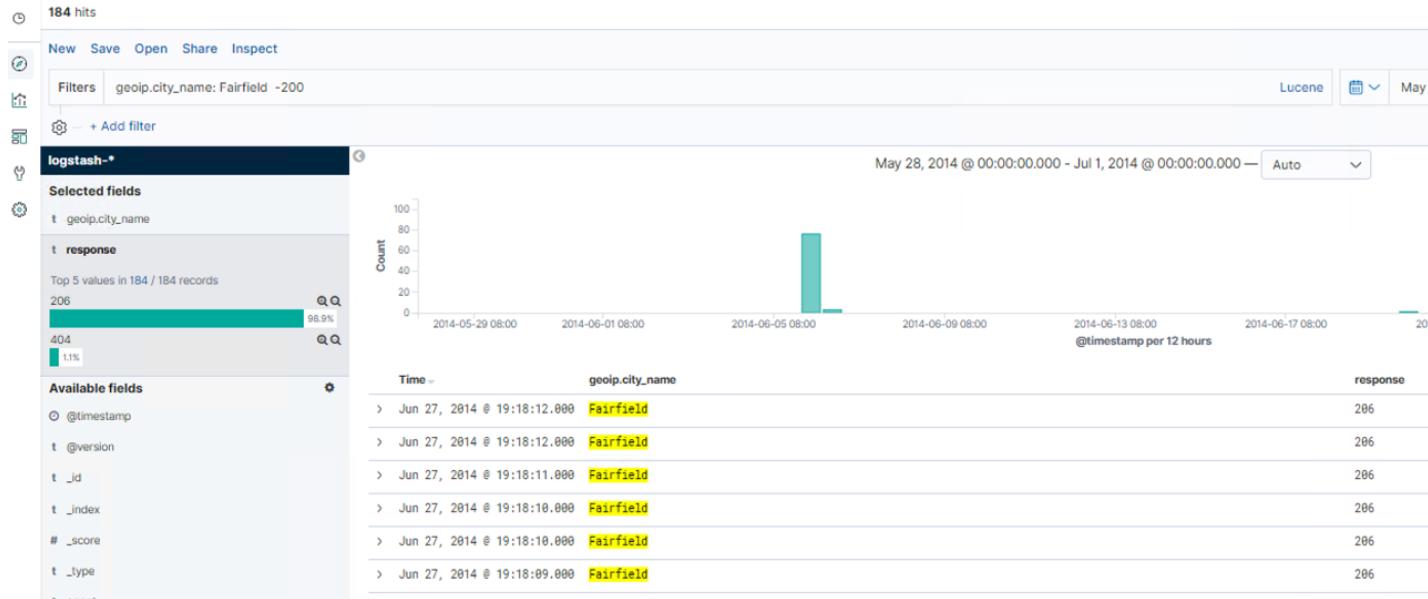


Elasticsearch query string/Lucene query



Elasticsearch query string/Lucene query

- The following is an example of a Must Not operator with free text:



Elasticsearch query string/Lucene query

184 hits

New Save Open Share Inspect

Filters geoip.city_name: Fairfield -200 Lucene May :

+ Add filter

Selected fields

t geoip.city_name

t response

Top 5 values in 184 / 184 records

206	98.9%
404	1.1%

Available fields

@timestamp @version _id _index # _score _type

May 28, 2014 @ 00:00:00.000 - Jul 1, 2014 @ 00:00:00.000 — Auto

Count

May 28, 2014 @ 00:00:00.000 - Jul 1, 2014 @ 00:00:00.000 — Auto

201

Time geoip.city_name response

Time	geoip.city_name	response
> Jun 27, 2014 @ 19:18:12.000	Fairfield	206
> Jun 27, 2014 @ 19:18:12.000	Fairfield	206
> Jun 27, 2014 @ 19:18:11.000	Fairfield	206
> Jun 27, 2014 @ 19:18:10.000	Fairfield	206
> Jun 27, 2014 @ 19:18:10.000	Fairfield	206
> Jun 27, 2014 @ 19:18:09.000	Fairfield	206

Filters

response:[301 TO 500]

+ Add filter

logstash-*

Selected fields

t geoip.city_name

t response

Available fields

Popular

t geoip.country_code2

@@ @timestamp

t @version

t _id

t _index

_score

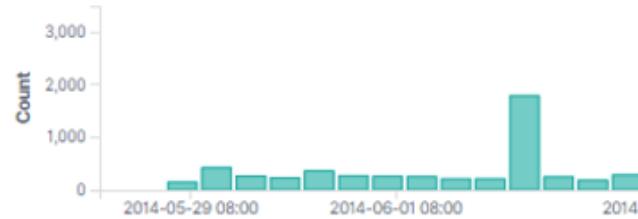
t _type

t agent

t auth

bytes

t clientip



Time	response
> Jun 27, 2014 @ 19:58:29.000	304
> Jun 27, 2014 @ 19:54:26.000	304
> Jun 27, 2014 @ 19:50:27.000	304
> Jun 27, 2014 @ 19:48:45.000	304
> Jun 27, 2014 @ 19:32:50.000	304
> Jun 27, 2014 @ 19:27:28.000	301
> Jun 27, 2014 @ 19:22:30.000	404
> Jun 27, 2014 @ 19:21:38.000	404

Filters

geoip.city_name: S?n*e

[+ Add filter](#)

logstash-*

Selected fields

t geoip.city_name

t response

Available fields

Popular

t geoip.country_code2

 @timestamp

t @version

t _id

t _index

_score

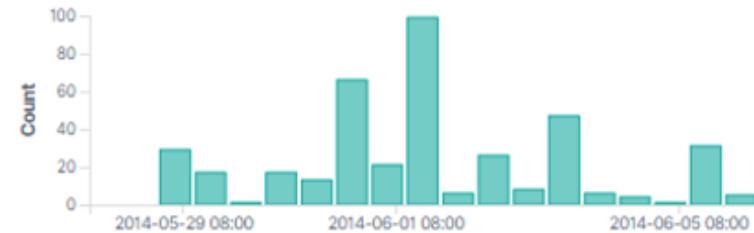
t _type

t agent

t auth

bytes

t clientip



Time

geoip.city_name

> Jun 27, 2014 @ 18:58:01.000 Sunnyvale

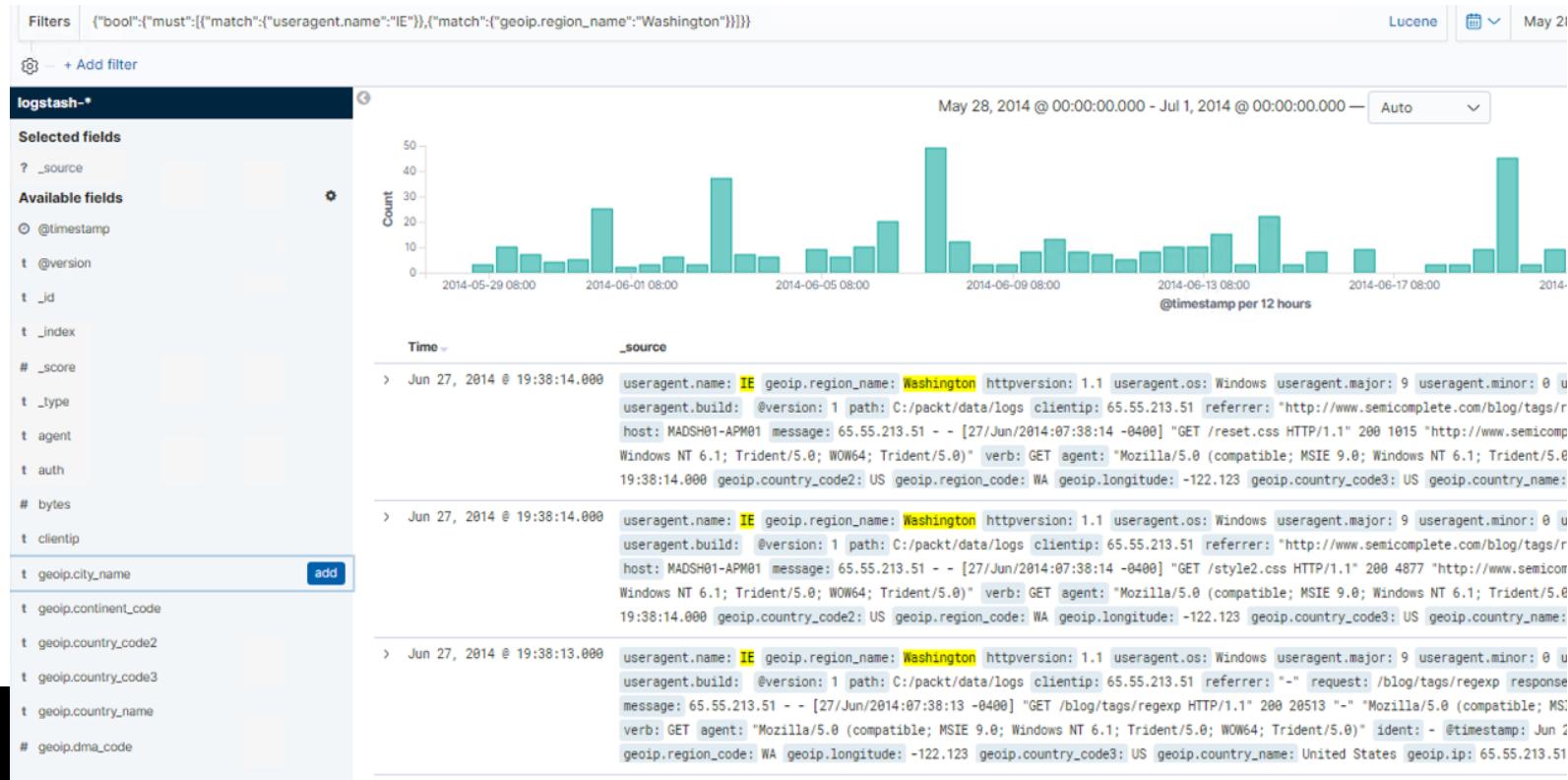
> Jun 27, 2014 @ 18:57:44.000 Singapore

> Jun 27, 2014 @ 18:57:42.000 Singapore

> Jun 27, 2014 @ 18:57:41.000 Singapore

> Jun 27, 2014 @ 11:33:24.000 Sunnyvale

Elasticsearch DSL query



KQL

- Kibana Query Language (KQL) is a query language specifically built for Kibana that is built to simplify query usage with easy-to-use syntax, support for querying on scripted fields, and ease of migration of queries as the product evolves.
- The query syntax is similar to the Lucene query syntax that was explained in the previous sections.

6 hits

New Save Open Share Inspect

Filters response:200 or geoip.city_name:Diedorf

KQL

May 28, 2014 @ 00:00:00.0 → Jul 1, 2014 @ 00:00:00.000

+ Add filter

logstash-*

Selected fields

t geoip.city_name

t response

Available fields

Popular

t geoip.country_code2

@timestamp

t @version

t _id

t _index

_score

t _type

t agent

t auth

bytes

t clientip



Time	geoip.city_name	response
> Jun 15, 2014 @ 06:02:02.000	Diedorf	200
> Jun 15, 2014 @ 06:02:02.000	Diedorf	200
> Jun 15, 2014 @ 06:02:01.000	Diedorf	200
> Jun 15, 2014 @ 06:02:01.000	Diedorf	200
> Jun 15, 2014 @ 06:02:01.000	Diedorf	200
> Jun 15, 2014 @ 06:02:01.000	Diedorf	200

KQL

59 hits

New Save Open Share Inspect

Filters (response:404 AND (geoip.city_name:Chicago or geoip.city_name: Nathon) and not useragent.name: BLEXBot)

KQL May 28, 2014 @ 00:00:00.0 → Jul 1, 2014 @ 00:00:00.0

logstash-*

Selected fields

- t geoip.city_name
- t response
- t useragent.name

Available fields

Popular

- t geoip.country_code2
- ⌚ @timestamp
- t @version
- t _id
- t _index
- # _score
- t _type
- t agent
- t auth
- # bytes
- t clientip
- t geoip.continent_code
- t geoip.country_code3

May 28, 2014 @ 00:00:00.000 - Jul 1, 2014 @ 00:00:00.000 — Auto

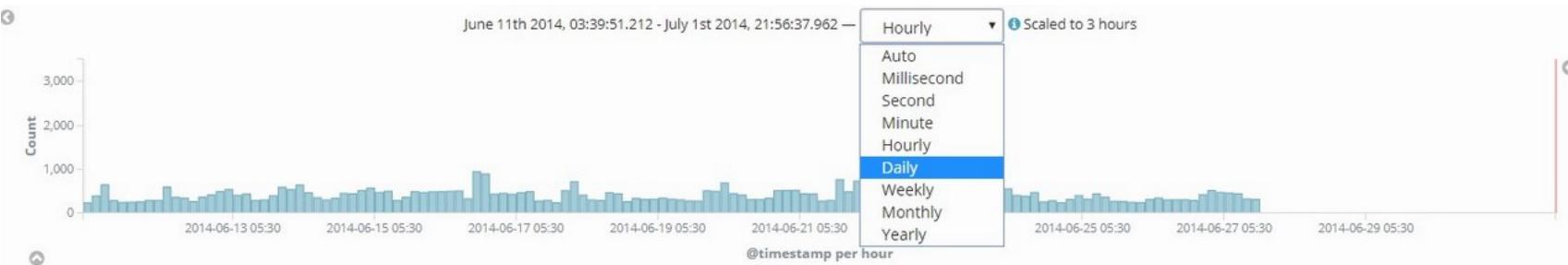
Count

Time geoip.city_name response useragent.name

Time	geoip.city_name	response	useragent.name
> Jun 27, 2014 @ 19:03:23.000	Chicago	404	Chrome
> Jun 27, 2014 @ 17:39:54.000	Nathon	404	Chrome
> Jun 27, 2014 @ 16:42:10.000	Chicago	404	Chrome
> Jun 27, 2014 @ 12:45:55.000	Chicago	404	Chrome
> Jun 26, 2014 @ 15:01:27.000	Chicago	404	Chrome
> Jun 26, 2014 @ 10:43:18.000	Chicago	404	Firefox
> Jun 25, 2014 @ 06:07:35.000	Chicago	404	Chrome
> Jun 23, 2014 @ 12:35:45.000	Chicago	404	Firefox
> Jun 22, 2014 @ 20:30:51.000	Chicago	404	Firefox
> Jun 22, 2014 @ 04:04:15.000	Chicago	404	Firefox

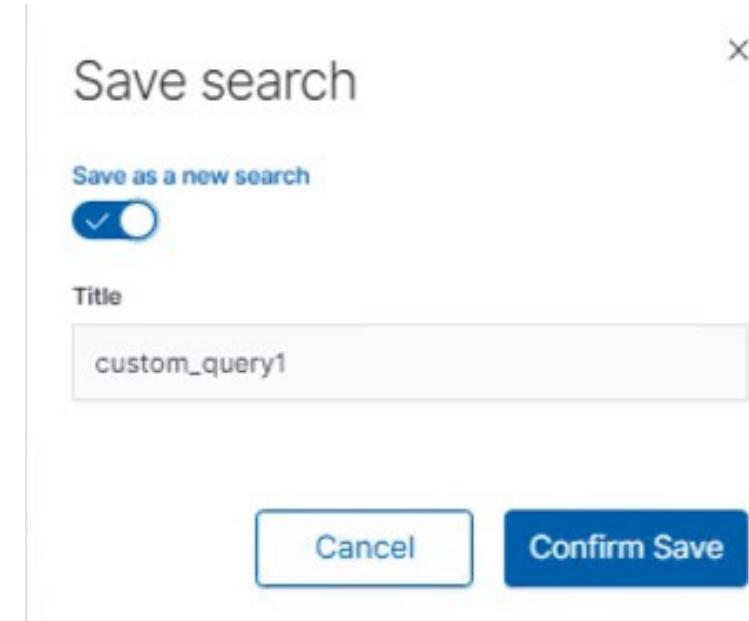
KQL

- The histogram interval can be changed by selecting the interval from the dropdown, as shown in the following screenshot:



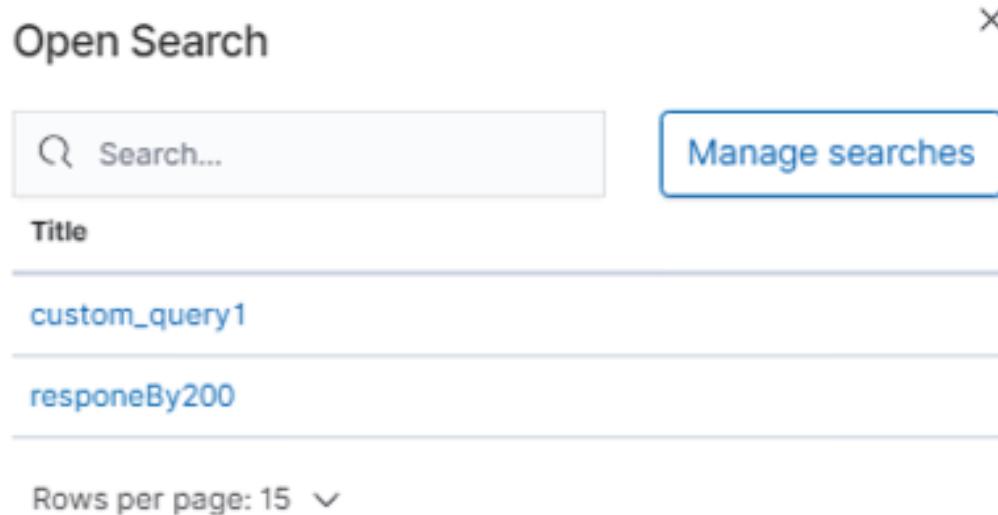
KQL

- The user can refer to existing stored searches later and modify the query, and they can either overwrite the existing search or save it as a new search, as follows:



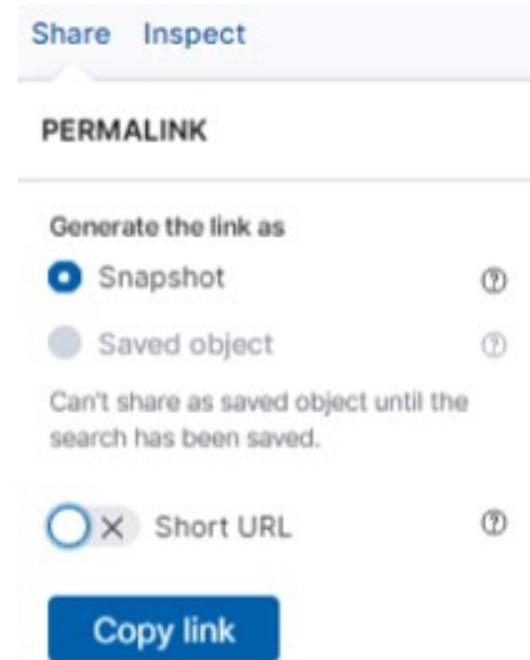
KQL

- Clicking the Open button displays the saved searches, as shown in the following screenshot:



KQL

- In Kibana, the state of the current page/UI is stored in the URL itself, thus allowing it to be easily shareable.
- Clicking the Share button allows you to share the Saved Search, as shown in the following screenshot:



I request was made
Request: Segment 0
This request queries Elasticsearch to fetch the data for the search.

Statistics Request Response

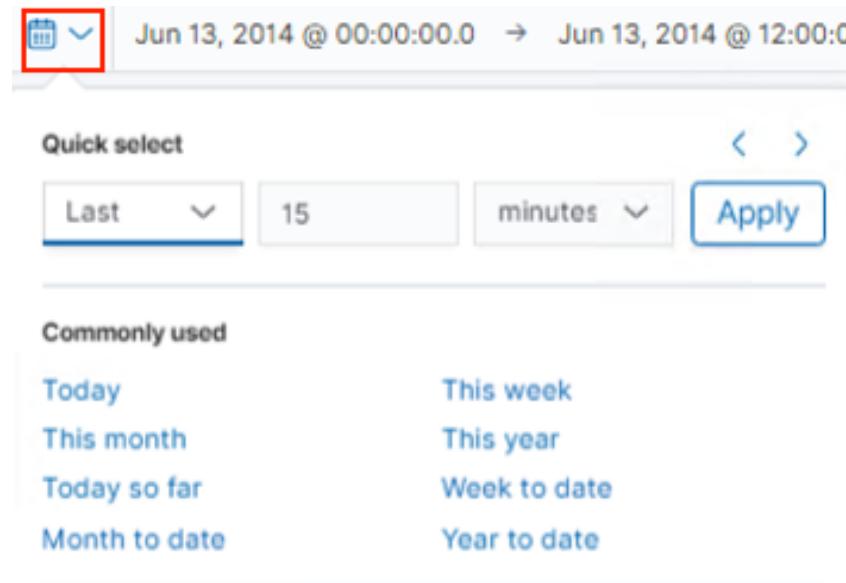
```
{  
  "version": true,  
  "size": 500,  
  "sort": [  
    {  
      "@timestamp": {  
        "order": "desc",  
        "unmapped_type": "boolean"  
      }  
    }  
  ],  
  "_source": {  
    "excludes": []  
  },  
  "aggs": {  
    "2": {  
      "date_histogram": {  
        "field": "@timestamp",  
        "interval": "10m",  
        "time_zone": "Asia/Singapore",  
        "min_doc_count": 1  
      }  
    }  
  },  
  "stored_fields": [  
    "*"  
  ],  
  "script_fields": {},  
  "docvalue_fields": [  
    {  
      "field": "@timestamp",  
      "format": "date_time"  
    }  
  ],  
  "query": {  
    "bool": {  
      "must": [  
        {  
          "bool": {  
            "must": [  
              {  
                "match": {  
                  "useragent.name": "IE"  
                }  
              },  
              {  
                "match": {  
                  "geoip.region_name": "Washington"  
                }  
              }  
            ]  
          }  
        ]  
      }  
    }  
  }  
}
```

KQL

- The Inspect button allows to view query statistics such as total hits, query time, the actual query fired against ES, and the actual response returned by ES.
- This would be useful to understand how the Lucene/KQL query we entered in the query bar translates to an actual ES query, as shown in the following screenshot:

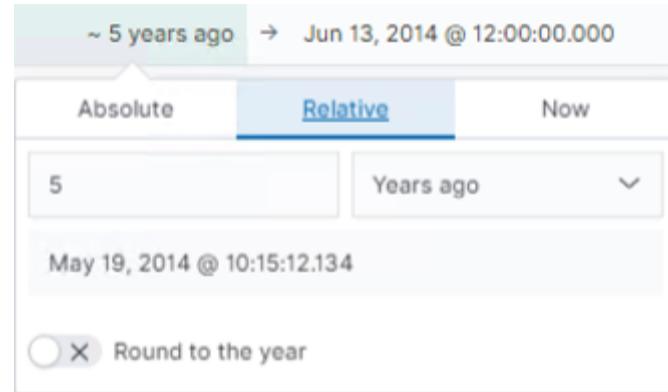
KQL

- Quick time filter: This helps you to filter quickly based on some already available time ranges:



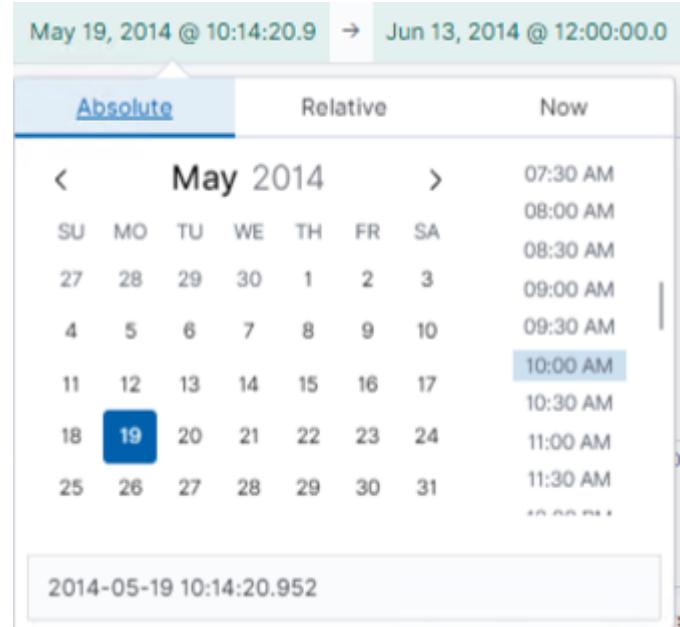
KQL

- Relative time filter: This helps you to filter based on the relative time with respect to the current time.
- Relative times can be in the past or the future, A checkbox is provided to round the time:



KQL

- Absolute time filter: This helps you to filter based on input start and end times:



 May 28, 2014 @ 00:00:00.00 → Jul 1, 2014 @ 00:00:00

Quick select



Last 

15

minutes 

Apply

Commonly used

Today

This week

This month

This year

Today so far

Week to date

Month to date

Year to date

Recently used date ranges

May 28, 2014 @ 00:00:00.000 to Jul 1, 2014 @ 00:00:00.000

May 28, 2016 @ 00:00:00.000 to Jul 1, 2016 @ 00:00:00.000

Last 15 minutes

Jun 13, 2014 @ 00:00:00.000 to Jun 13, 2014 @

Refresh every

10

seconds 

 Start

Available Fields

- @timestamp
- t @version
- t _id
- t _index
- # _score
- t _type
- t agent
- t auth
- # bytes
- t clientip
- t geoip.city_name

Top 5 values in 393 / 500 records

San Jose	33.3%
Amsterdam	9.2%
Seattle	6.6%
Moscow	4.6%
Chantilly	4.3%

negative filter

exists filter

positive filter

Time

Source

June 27th 2014, 17:30:00.000

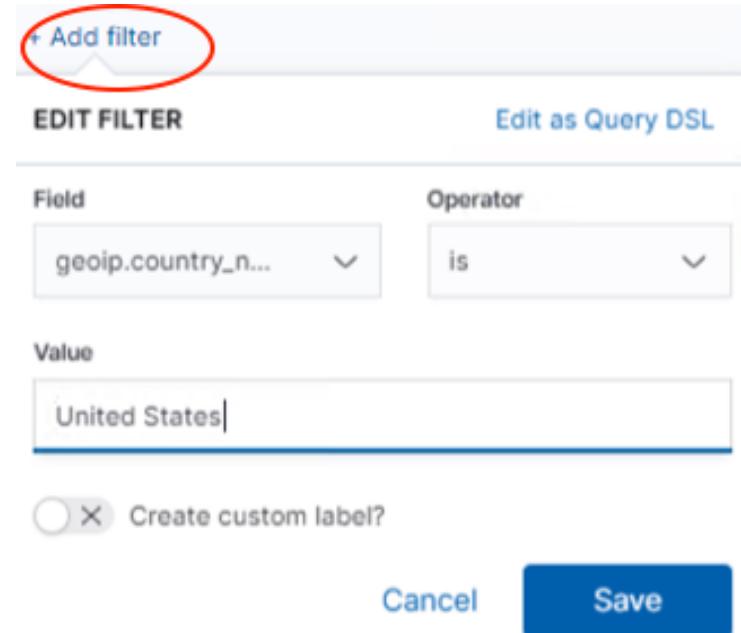
```
request: /scripts/netcat-webserver agent: "Mozilla/5.0 (compatible; EasouSpider; +http://www.easou.com/search/spider.html)" geoip.timezone: Asia/Shanghai geoip.ip: 183.60.215.50 geoip.latitude: 23.117 geoip.country_code3: CN geoip.region_name: Guangdong geoip.location: { "lon": 113.25, "lat": 23.117 } ident: - verb: GET useragent.name: EasouSpider useragent.os_name: Other useragent.os: 0 -- [27/Jun/2014:08:00:00 -0400] "GET /scripts/netcat-webserver HTTP/1.1" 200 182 "
```

Filter for field present

- @timestamp * June 27th 2014, 17:30:00.000
- t @version * obiXWGABy4LieyyOIZ4u
- t _id * logstash-2014.06.27
- # _score * -
- t _type * logs
- t agent * "Mozilla/5.0 (compatible; EasouSpider; +http://www.easou.com/search/spider.html)"
- t auth * -
- # bytes * 182
- t clientip * 183.60.215.50
- t geoip.city_name * Guangzhou
- t geoip.continent_code * AS
- t geoip.country_code2 * CN

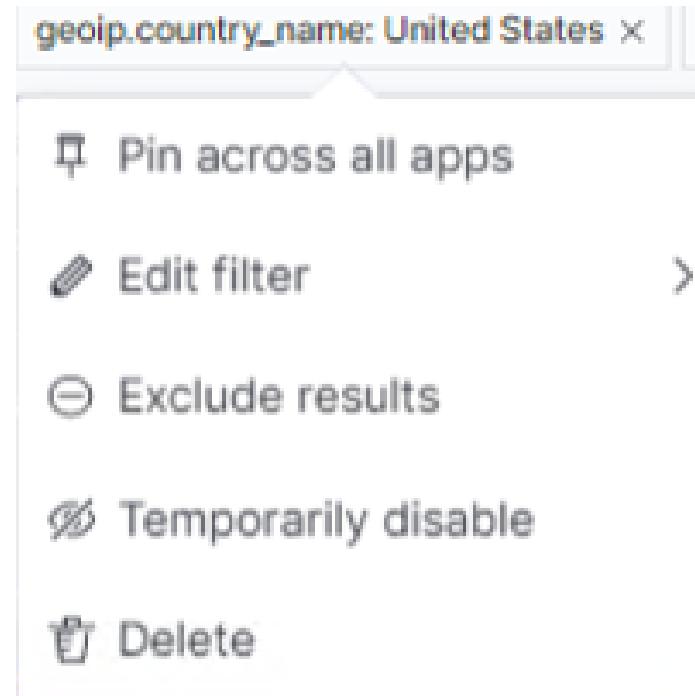
KQL

- You can also add filters manually by clicking the Add a Filter button found below the query bar.
- Clicking on the button will launch a popup in which filters can be specified and applied by clicking the Save button, as follows:



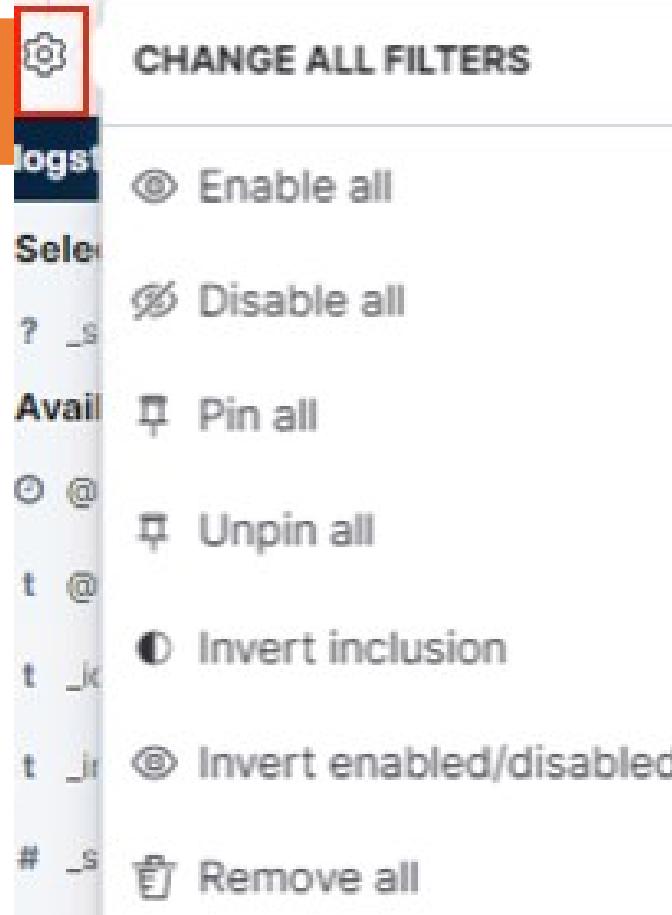
KQL

- The following screenshot displays the preceding actions that can be applied:



KQL

- You can perform the preceding actions across multiple filters at once rather than one at a time by clicking on the filter settings icon, as follows:



Visualize

For our Apache access log analysis use case, the user can easily find out answers to some of the typical questions raised in log analysis, such as the following:

- What's the traffic in different regions of the world?
- What are the top URLs requested?
- What are the top IP addresses making requests?
- How's the bandwidth usage over time?
- Is there any suspicious or malicious activity from any region/IP address?

X

New Visualization



Filter



Area



Controls



Coordinate Map



Data Table



Gauge



Goal



Heat Map



Horizontal Bar



Line



Markdown



Metric



Pie



Region Map



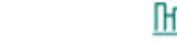
Tag Cloud



Timeline



Vega



Vertical Bar



Visual Builder

Select a visualization type

Start creating your visualization by selecting a type for that visualization.



Kibana aggregations

Kibana supports two types of aggregations:

- Bucket aggregations
- Metric aggregations

Kibana aggregations

Bucket aggregations can do the following:

- Give an employee index containing employee documents
- Find the number of employees based on their age group or location
- Give the Apache access logs index, and find the number of 404 responses by country

Kibana aggregations

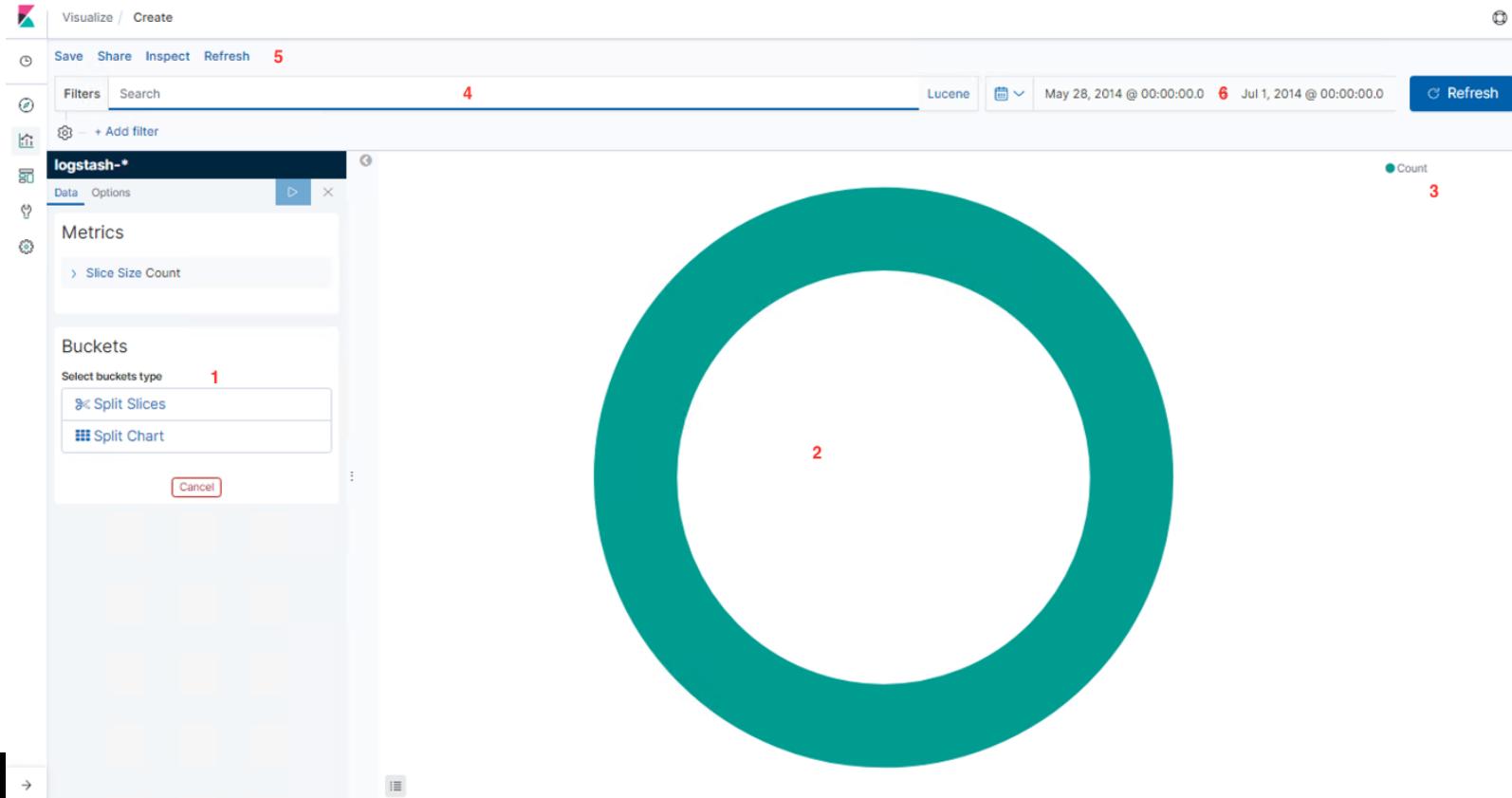
- Bucket aggregation supports sub aggregations, that is, given a bucket, all the documents present in the bucket can be further bucketed (grouped based on criteria); for example, finding the number of 404 responses by country and also by state.
- Depending on the type of bucket aggregation, some define a single bucket, some define a fixed number of multiple buckets, and others dynamically create buckets during the aggregation process.

Creating a visualization

The following are the steps to create visualizations:

- Navigate to the Visualize page and click the Create a new Visualization button or the + button
- Select a visualization type
- Select a data source
- Build the visualization

The Visualize Interface looks as follows:



Visualization types

Let's take a look at each visualization type.

- Line, area, and bar charts
- Data tables
- Markdown widgets
- Metrics
- Goals
- Gauges
- Pie charts etc.

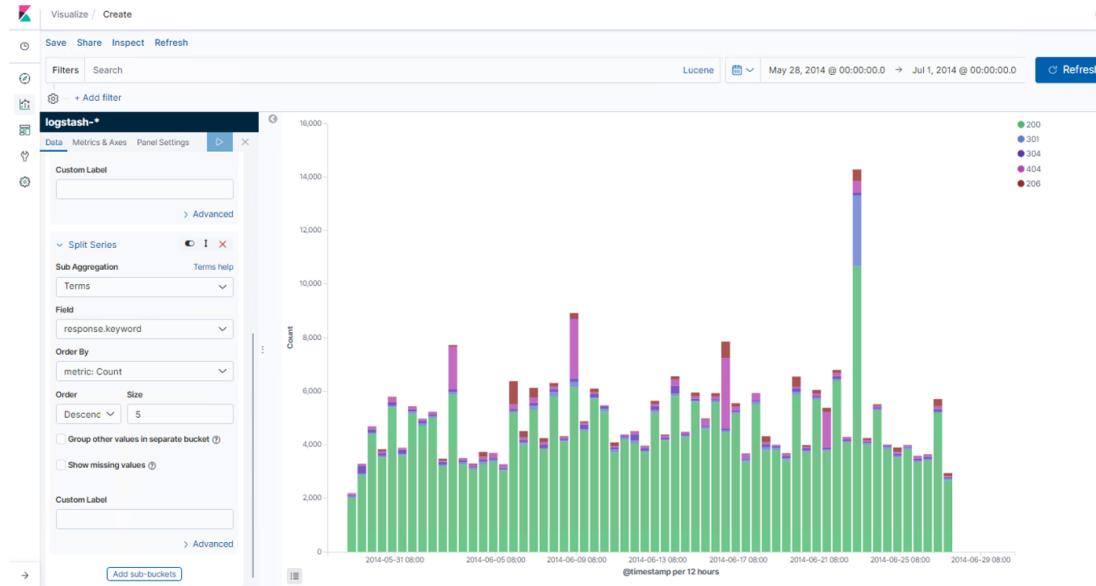
Visualizations in action

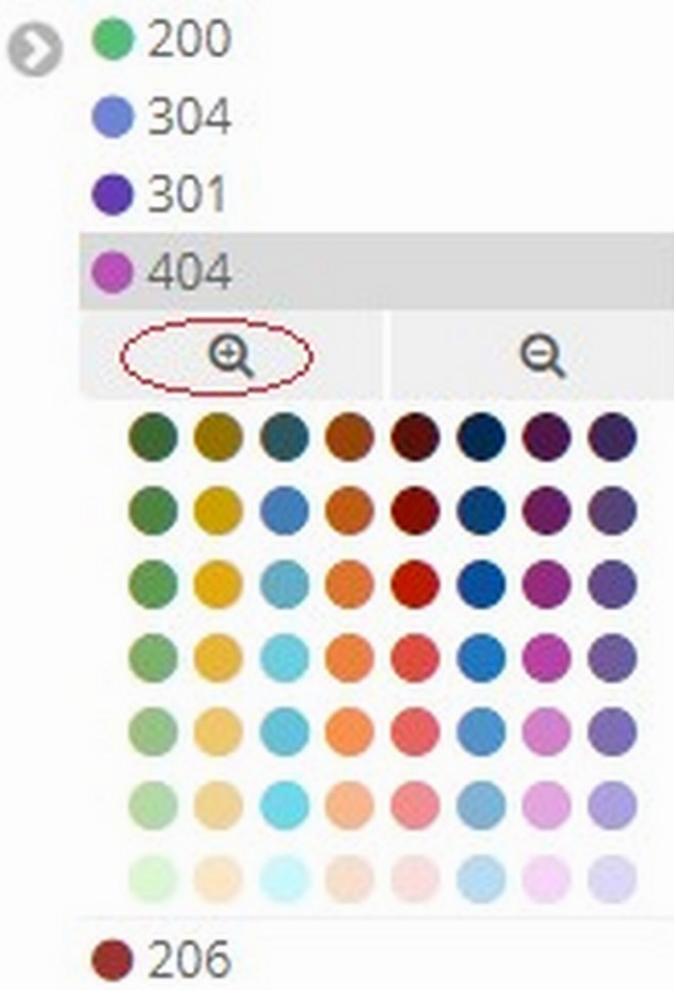
Let's see how different visualizations can help us in doing the following:

- Analyzing response codes over time
- Finding the top 10 requested URLs
- Analyzing the bandwidth usage of the top five countries over time
- Finding the most used user agent

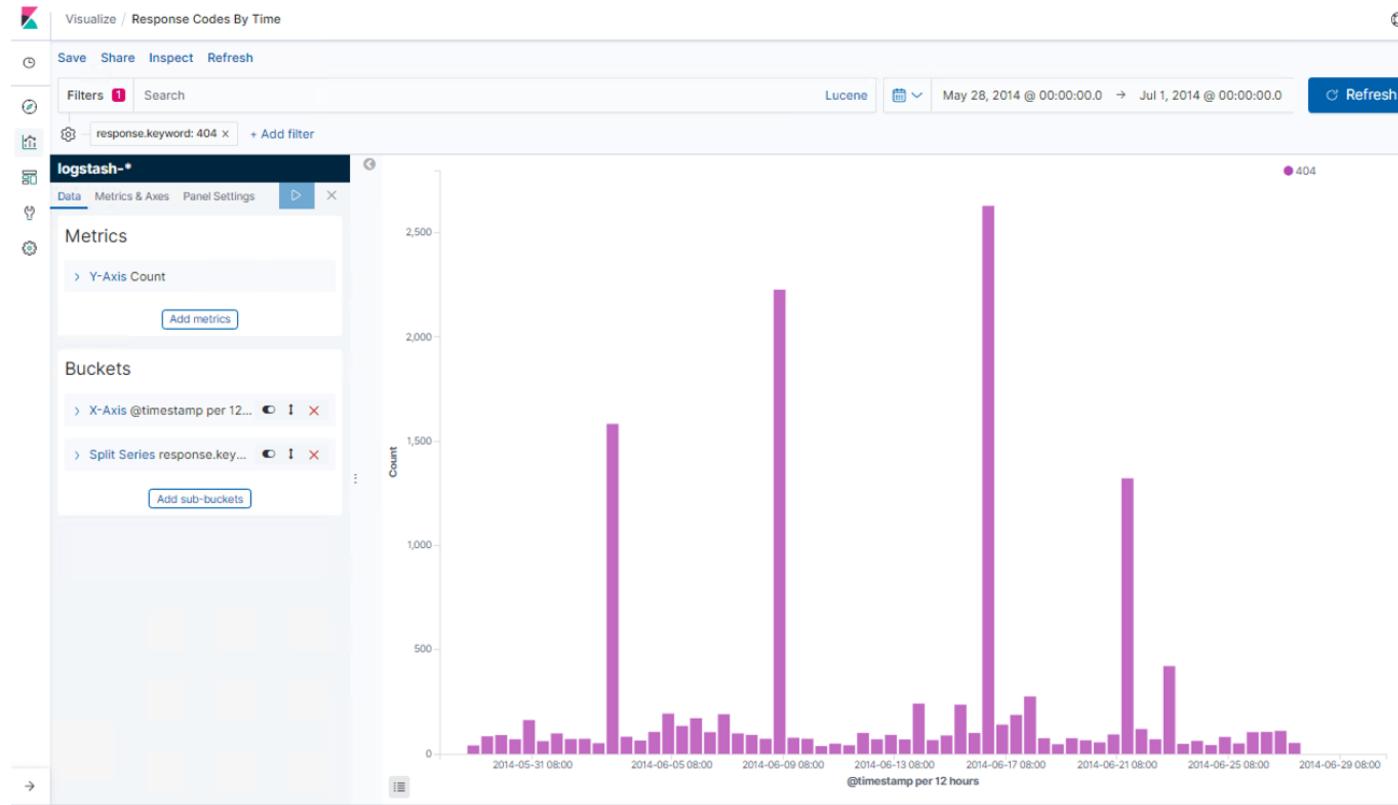
Response codes over time

- The following screenshot displays the steps to create a new visualization for response codes over time:



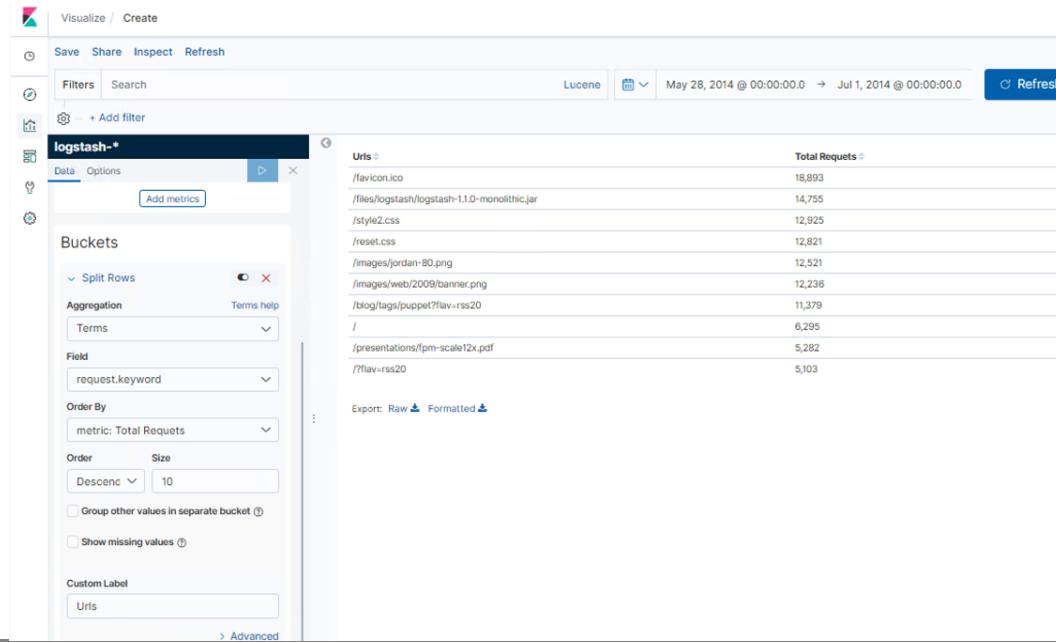


- The resulting graph is shown in the following screenshot:



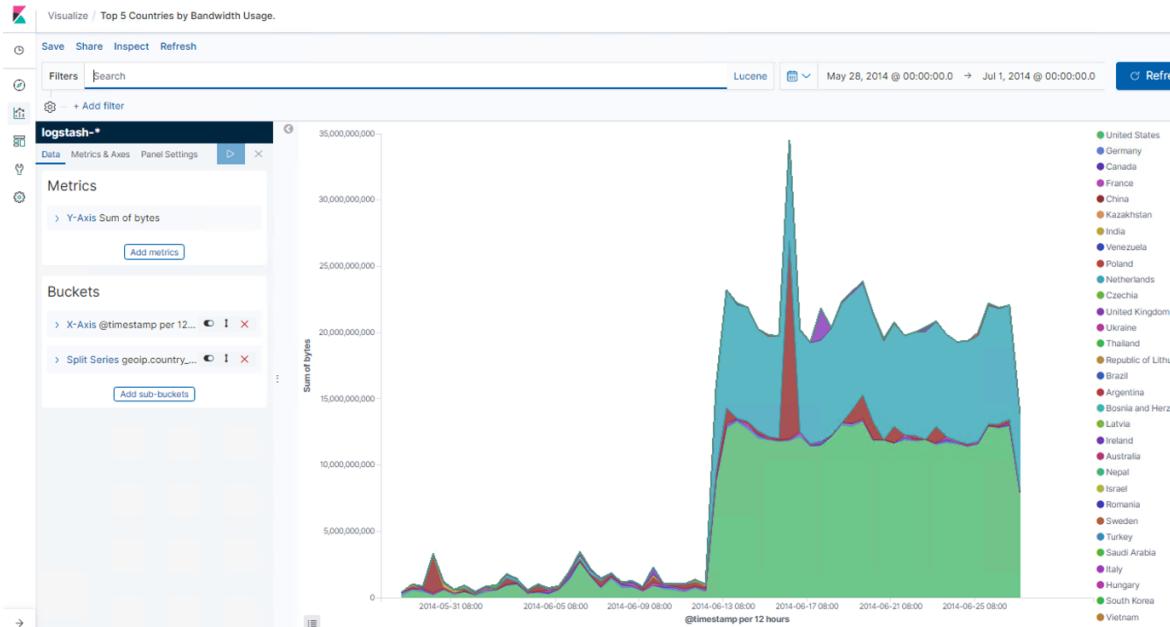
Top 10 requested URLs

- The following screenshot displays the steps to create a new visualization for the top 10 requested URLs:



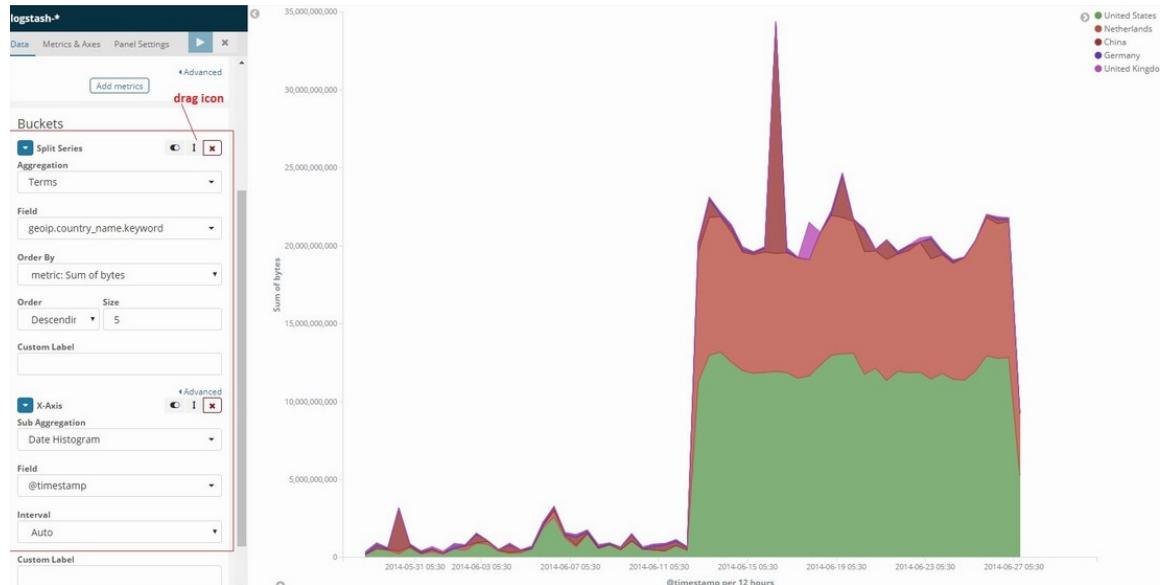
Bandwidth usage of the top five countries over time

- The following screenshot displays the steps to create a new visualization for the bandwidth usage of the top five countries over time:

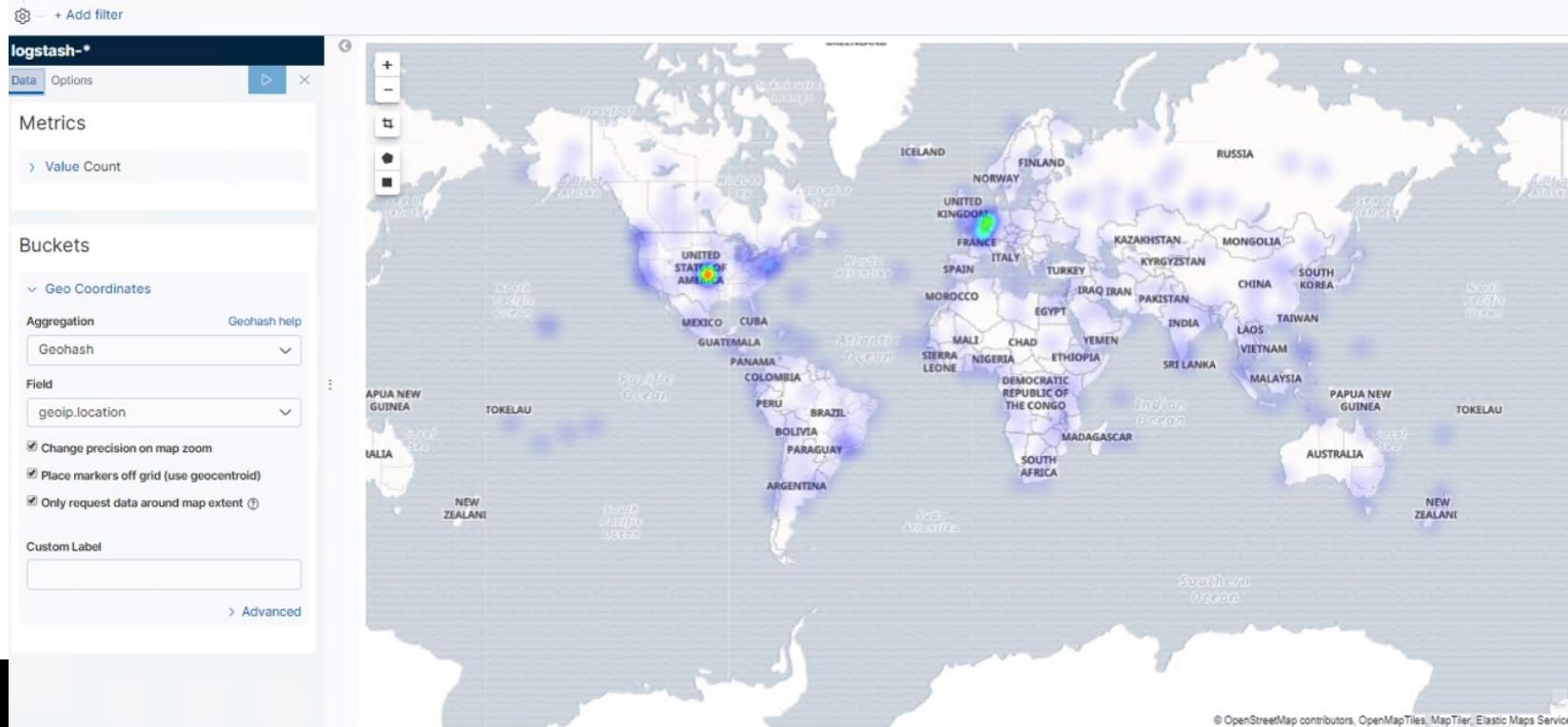


Top 10 requested URLs

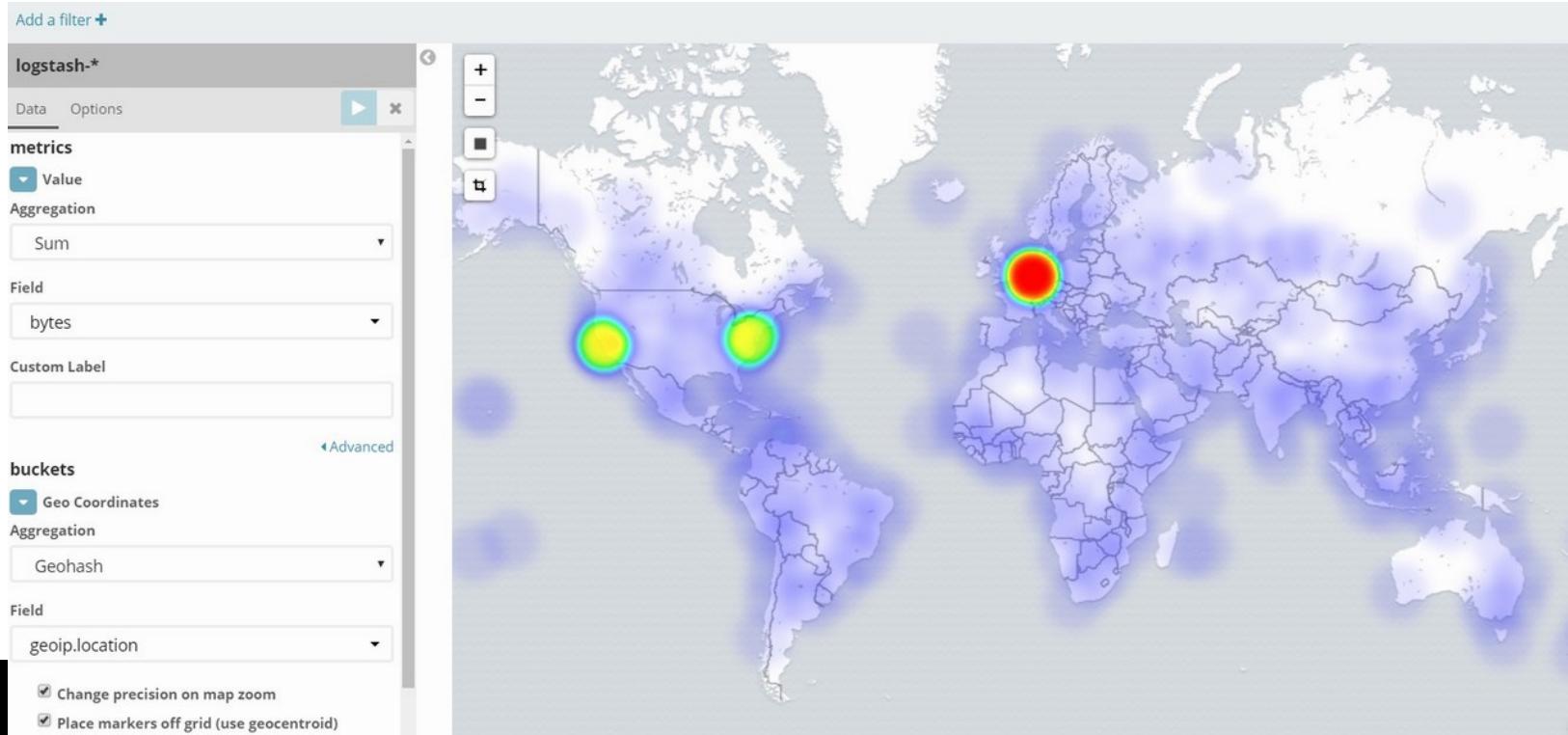
- What if we were not interested in finding only the top five countries? Rearrange the aggregation and click Play, as follows:



Web traffic originating from different countries



Web traffic originating from different countries



Most used user agent

logstash-*

Data Options

Metrics

> Tag Size Count

Buckets

Tags

Aggregation Terms help

Terms

Field useragent.name.keyword

Order By metric: Count

Order Descenc Size

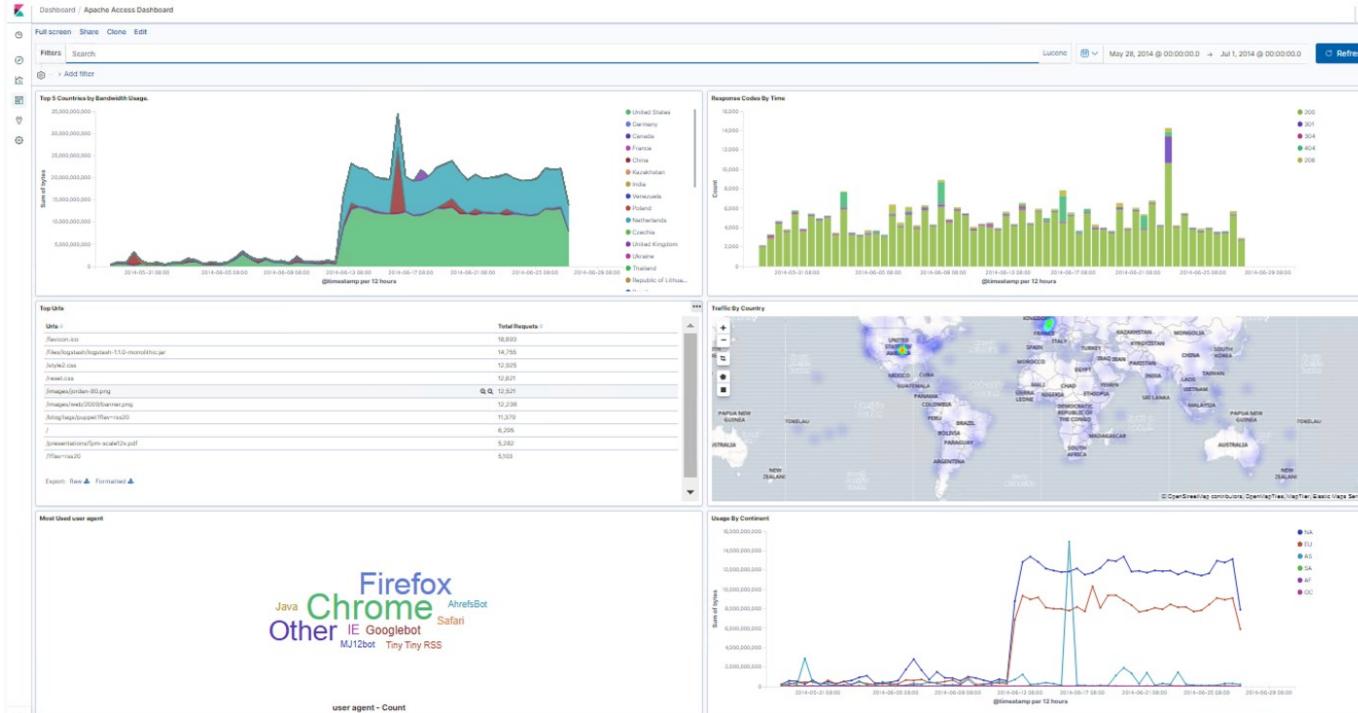
10

Group other values in separate bucket ⑦

Show missing values ⑦

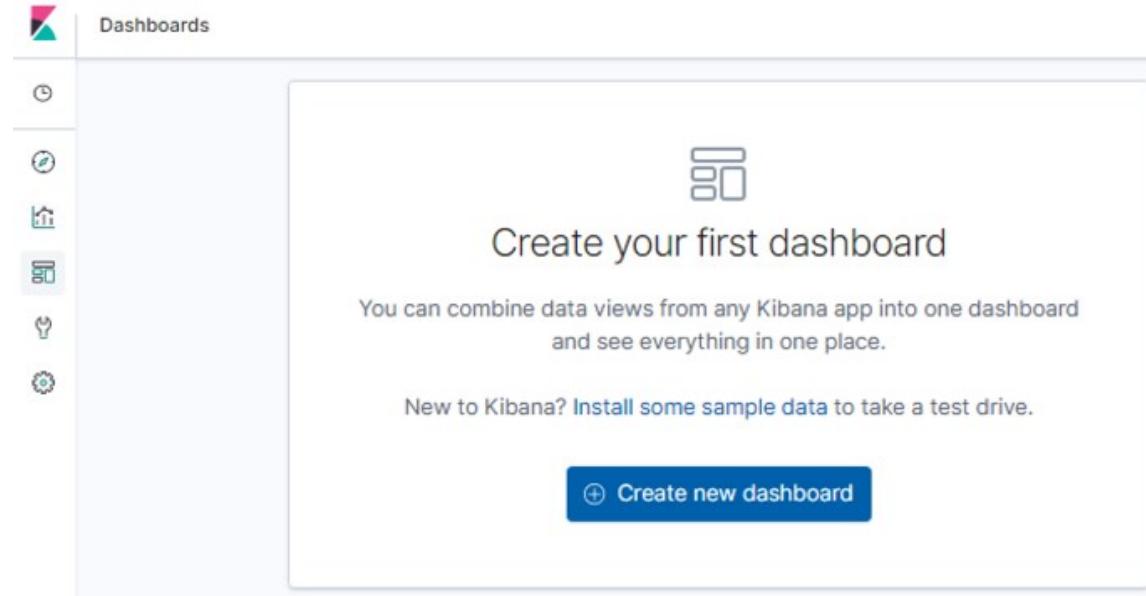


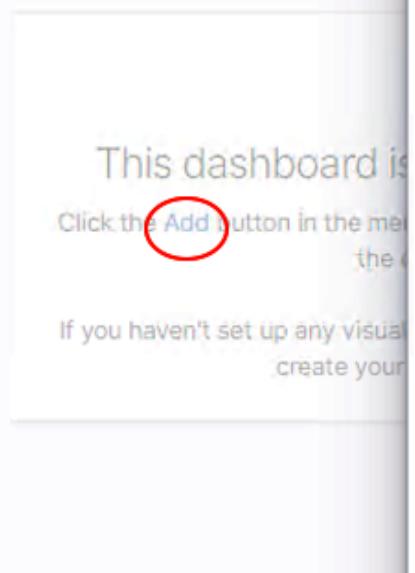
Dashboards



Creating a dashboard

- In order to create a new dashboard, navigate to the Dashboard page and click the Create a Dashboard button:



[Save](#) [Cancel](#) [Add](#) [Options](#) [Share](#)[Filters](#) [Search](#) [+ Add filter](#)

Add Panels

[Visualization](#)[Saved Search](#) [Search...](#)[Add new Visualization](#)**Title**[Top 5 Countries by Bandwidth Usage.](#)[Response Codes By Time](#)[Top URLs](#)[Traffic By Country](#)[Most Used User Agent](#)[Usage By Continent](#)

Rows per page: 15 ▾

Creating a dashboard

- The user can expand, edit, rearrange, or remove visualizations using the buttons available at the top corner of each visualization, as follows:





Save dashboard

Save as a new dashboard



Title

Apache Access Dashboard

Description

Store time with dashboard



This changes the time filter to the currently selected time each time this dashboard is loaded.

Cancel

Confirm Save

Cloning the dashboard

Clone Dashboard

X

Please enter a new name for your dashboard.

Cancel

Confirm Clone

Share Clone Edit

SHARE THIS DASHBOARD

> Embed code <

🔗 Permalinks < >

Share Clone Edit

< PERMALINK

Generate the link as

Snapshot ?

Saved object ?

Short URL ?

Copy link

Share Clone Edit

< EMBED CODE

Generate the link as

Snapshot ?

Saved object ?

Short URL ?

Copy iFrame code

Sharing the dashboard

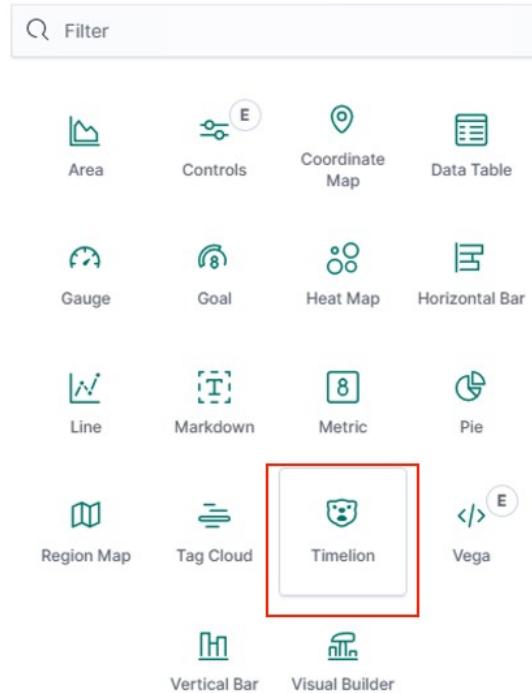
Timelion

- Timelion visualizations are special type of visualization for analyzing time-series data in Kibana.
- They enable you to combine totally independent data sources within the same visualization.

Timelion

- Timelion is available just like any other visualization in the New Visualization window, as follows:

New Visualization



Timelion

Build time-series using functional expressions

Timelion expressions

- The simplest Timelion expression used for generating graphs is as follows:

.es(*)

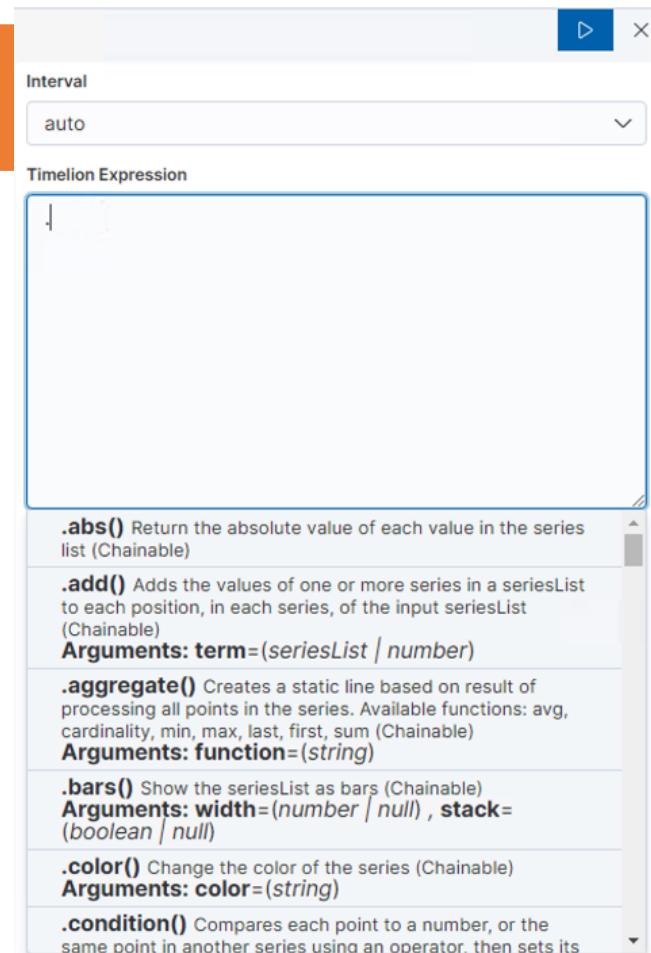
Timelion expressions

- If you'd like to restrict Timelion to data within a specific index (for example, logstash-*), you can specify the index within the function as follows:

```
.es(index=logstash-*)
```

Timelion expressions

- Timelion's helpful autocomplete feature will help you build the expression as you go along, as follows:



Timelion expressions

- Let's find the average bytes usage over time for the US. The expression for this would be as follows:

```
.es(q='geoip.country_code3:US',metric='avg:bytes')
```

- The output is displayed in the next slide.

Filters Search

Lucene

Refresh

+ Add filter



14000000

q:geoip.country_code3:US > avg(bytes)

12000000

10000000

8000000

6000000

4000000

2000000

0

Dev Tools

auto

Timelion Expression

```
.es(q='geoip.country_code3:US',metric='avg:bytes')
```



Timelion expressions

- Let's find the average bytes usage over time for the US and the average bytes usage over time for China. The expression for this would be as follows:

```
es(q='geoip.country_code3:US',metric='avg:bytes'),  
.es(q='geoip.country_code3:CN',metric='avg:bytes')
```

- The output is displayed in the next slide.

+ Add filter

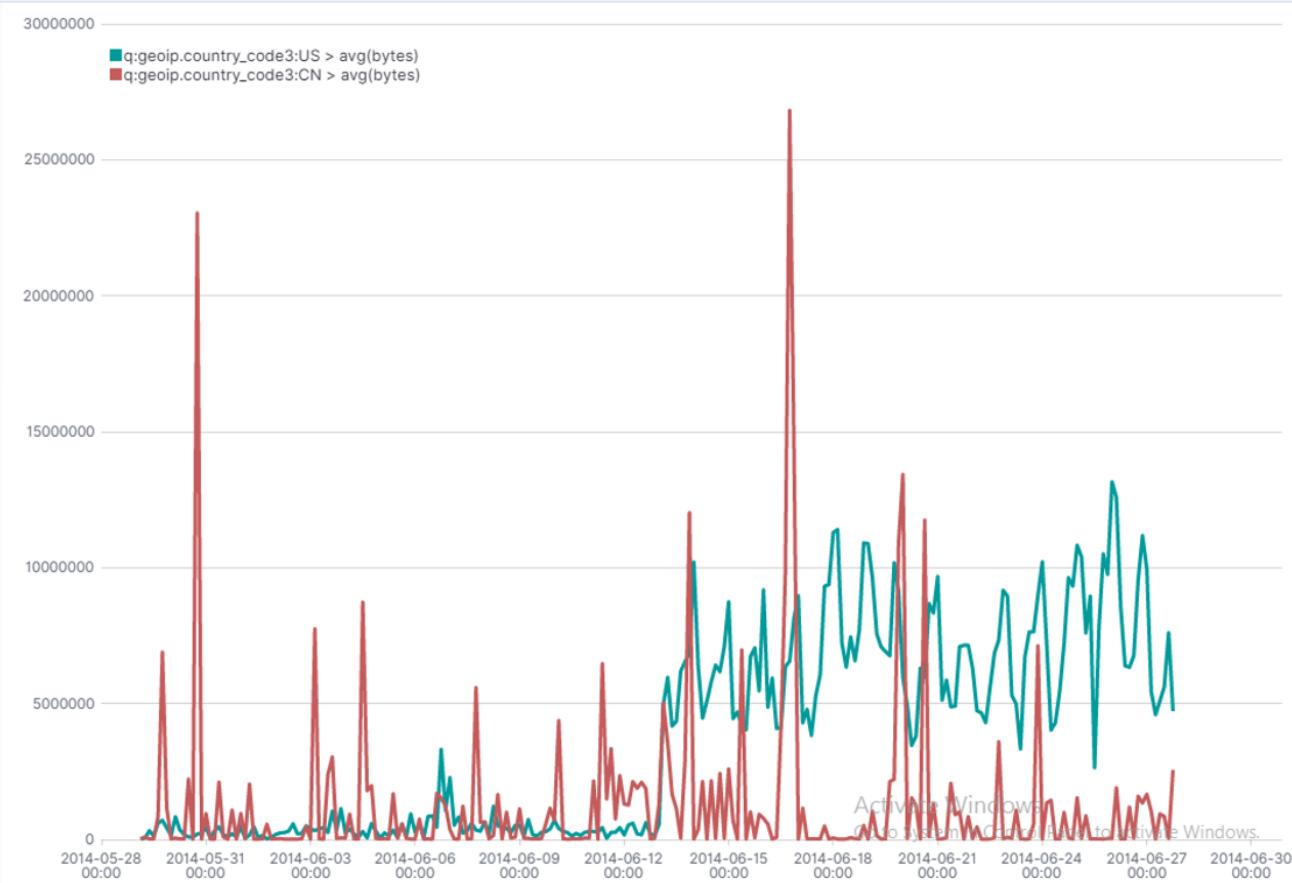


Interval

auto

Timelion Expression

```
.es(q='geoip.country_code3:US',metric='avg:bytes'),  
.es(q='geoip.country_code3:CN',metric='avg:bytes')
```



Timelion expressions

- Timelion also allows for the chaining of functions.
- Let's change the label and color of the preceding graphs.
- The expression for this would be as follows:

```
.es(q='geoip.country_code3:US',metric='avg:bytes').label('United States').color('yellow'),  
.es(q='geoip.country_code3:CN',metric='avg:bytes').label('China').color('red')
```

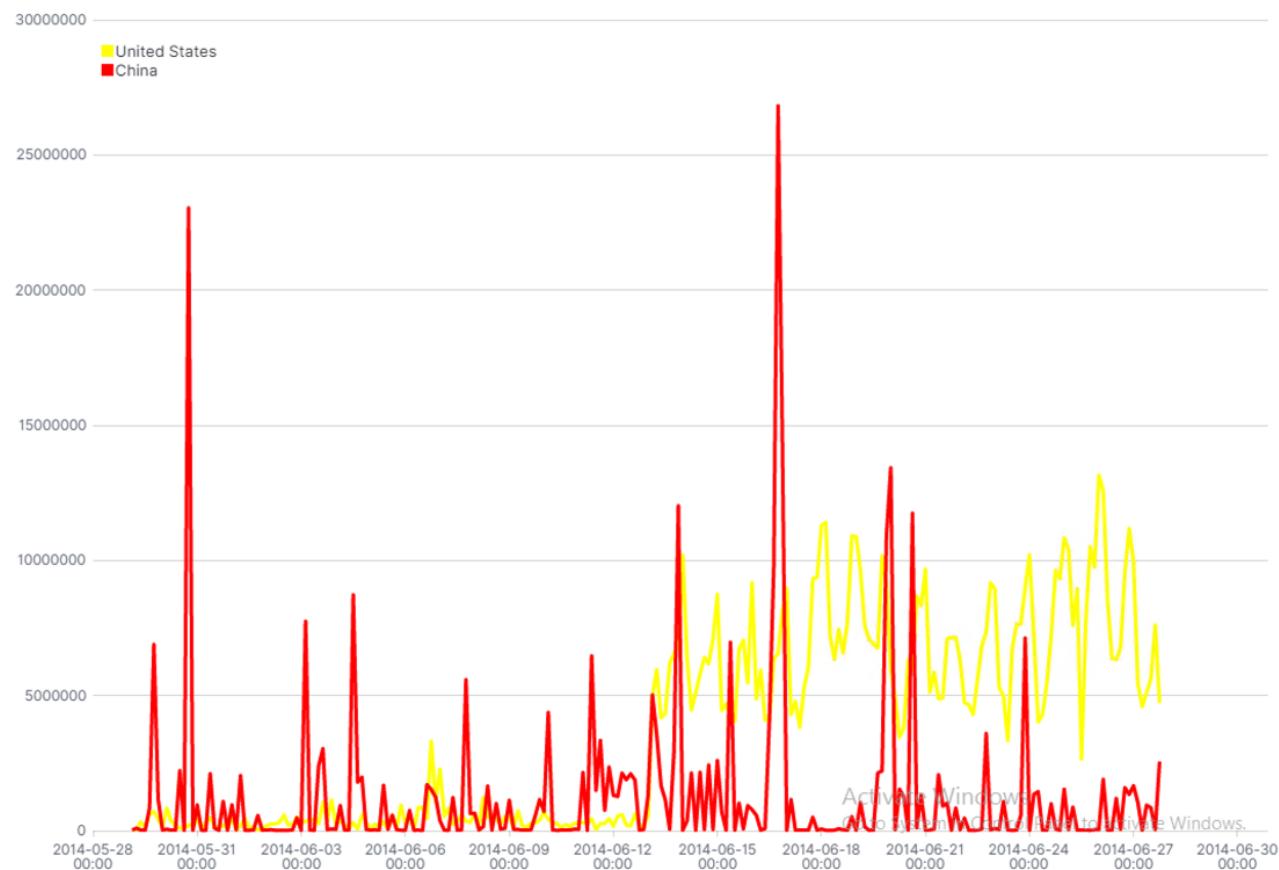
- The output is displayed in the next slide.

+ Add filter

Interval

Timeline Expression

```
.es(q='geoip.country_code3:US',metric='avg:bytes').label('United States').color('yellow'),  
.es(q='geoip.country_code3:CN',metric='avg:bytes').label('China').color('red')
```



Timelion expressions

- Let's compare the sum of bytes usage to the previous week for the US.

- The expression for this would be as follows:

```
.es(q='geoip.country_code3:US',metric='sum:bytes').label('Current Week'),  
.es(q='geoip.country_code3:US',metric='sum:bytes',  
offset=-1w).label('Previous Week')
```

- The output is displayed in the next slide.

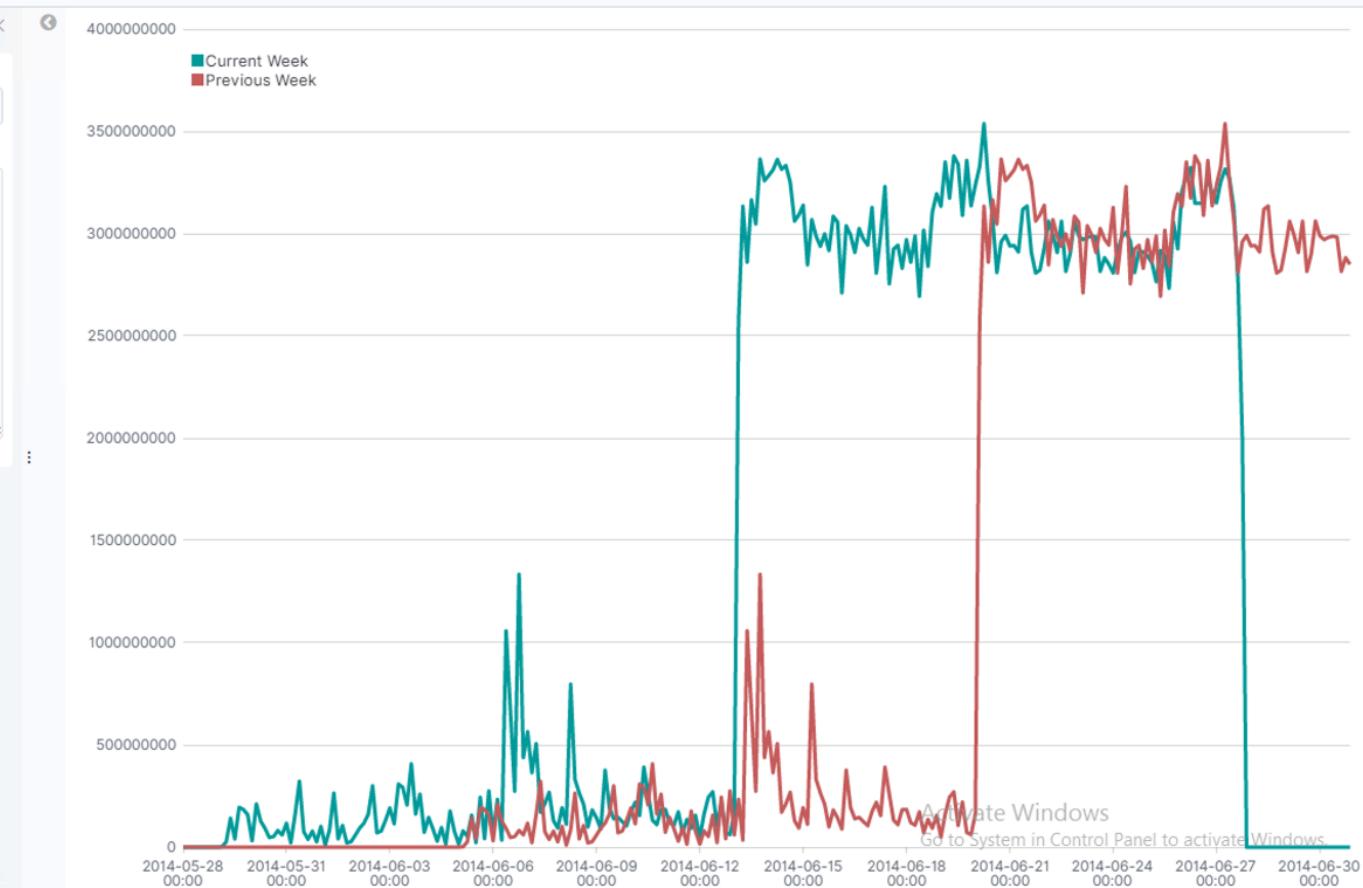
+ Add filter

Interval

auto

Timeline Expression

```
.es(q='geopip.country_code3:US',metric='sum:bytes').label('Current Week'),  
.es(q='geopip.country_code3:US',metric='sum:bytes',  
offset=-1w).label('Previous Week')
```



Using plugins

- Plugins are a way to enhance the functionality of Kibana.
- All the plugins that are installed will be placed in the `$KIBANA_HOME/plugins` folder.
- Elastic, the company behind Kibana, provides many plugins that can be installed, and there are quite a number of public plugins that are not maintained by Elastic that can be installed, too.

Installing plugins

- Navigate to KIBANA_HOME and execute the install command, as shown in the following code, to install any plugins.
- During installation, either the name of the plugin can be given (if it's hosted by Elastic itself), or the URL of the location where the plugin is hosted can be given:

```
$ KIBANA_HOME>bin/kibana-plugin install <package  
name or URL>
```

Installing plugins

- For example, to install x-pack, a plugin developed and maintained by Elastic, execute the following command:
\$ KIBANA_HOME>bin/kibana-plugin install x-pack
- To install a public plugin, for example, LogTrail (<https://github.com/sivasamyk/logtrail>), execute the following command:

```
$ KIBANA_HOME>bin/kibana-plugin install  
https://github.com/sivasamyk/logtrail/releases/download/v0.1.3  
1/logtrail-6.7.1-0.1.31.zip
```

Removing plugins

- To remove a plugin, navigate to KIBANA_HOME and execute the remove command followed by the plugin name:

```
$ KIBANA_HOME>bin/kibana-plugin remove x-pack
```



Summary

- In this lesson, we covered how to effectively use Kibana to build beautiful dashboards for effective storytelling about your data.
- We learned how to configure Kibana to visualize data from Elasticsearch.
- We also looked at how to add custom plugins to Kibana.

COMPLETE LAB 7

8. Elastic X-Pack



Elastic X-Pack

In this lesson, we will cover the following topics:

- Installing Elasticsearch and Kibana with X-Pack
- Configuring/activating X-Pack trial account
- Securing Elasticsearch and Kibana
- Monitoring Elasticsearch
- Alerting



Installing Elasticsearch and Kibana with X-Pack

Prior to Elastic 6.3, X-Pack was an extension of Elastic Stack that could have been installed on top of Elasticsearch or Kibana. Now, Elasticsearch and Kibana come in two flavors:

- OSS version, that is, Apache 2.0 License
- Elastic license

Installation

- In order to explore X-Pack and its features, we will need to download Elasticsearch and Kibana with Elastic license.
- You can download Elasticsearch from
<https://www.elastic.co/downloads/past-releases/elasticsearch-7-0-0>.
- You can download Kibana from
<https://www.elastic.co/downloads/past-releases/kibana-7-0-0>.

Installation

- When you start Elasticsearch, you should see X-Pack-related plugins getting loaded and new files getting created under the `ES_HOME/config` folder.
- The bootup logs should also indicate the license type; in this case, this will be basic:

https://github.com/fenago/elasticsearch/blob/master/snippets/8_1.txt

Installation

 **elasticsearch.keystore**

 **elasticsearch.yml**

 **jvm.options**

 **log4j2.properties**

 **role_mapping.yml**

 **roles.yml**

 **users**

 **users_roles**

Installation

- When you start Kibana, you should see a list of X-Pack-related plugins that are loaded, as shown in the following code:

https://github.com/fenago/elasticsearch/blob/master/snippets/8_2.txt





Recently viewed

- Discover
- Visualize
- Dashboard
- Canvas
- Maps
- Machine Learning
- Infrastructure
- Logs
- APM
- Uptime
- Dev Tools
- Stack Monitoring
- Management

Add Data to Kibana

Use these solutions to quickly turn your data into pre-built dashboards and monitoring systems.



APM

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

[Add APM](#)

Logging

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

[Add log data](#)

Metrics

Collect metrics from the operating system and services running on your servers.

[Add metric data](#)

Security analytics

Centralize security events for interactive investigation in ready-to-go visualizations.

[Add security events](#)[Add sample data](#)

Load a data set and a Kibana dashboard

[Upload data from log file](#)

Import a CSV, NDJSON, or log file

[Use Elasticsearch data](#)

Connect to your Elasticsearch index

Visualize and Explore Data



APM

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



Canvas

Showcase your data in a pixel-perfect way.



Dashboard

Display and share a collection of visualizations and saved searches.



Discover

Interactively explore your data by querying and filtering raw documents.



Graph

Surface and analyze relevant relationships in your Elasticsearch data.



Infrastructure

Explore infrastructure metrics and logs for common servers, containers, and services.



Logs

Stream logs in real time or scroll through historical



Machine Learning

Automatically model the normal behavior of your

Manage and Administer the Elastic Stack



Console

Skip cURL and use this JSON interface to work with your data directly.



Index Patterns

Manage the index patterns that help retrieve your data from Elasticsearch.



Monitoring

Track the real-time health and performance of your Elastic Stack.



Saved Objects

Import, export, and manage your saved searches, visualizations, and dashboards.



Spaces

Organize your dashboards



Rollups

Summarize and store historical data in a smaller index for future analysis.



Security Settings

Protect your data and easily manage who has access to what with users and roles.



Watcher

Detect changes in your

[Collapse](#)

Activating X-Pack trial account

The screenshot shows the Elasticsearch Management interface with the 'License management' tab selected. On the left, there's a sidebar with various icons and links. The 'Elasticsearch' section includes 'Index Management', 'Index Lifecycle Policies', 'Rollup Jobs', 'Cross Cluster Replication', 'Remote Clusters', 'License Management' (which is highlighted with a red box), and '8.0 Upgrade Assistant'. The 'Kibana' section includes 'Index Patterns', 'Saved Objects', 'Spaces', 'Reporting', and 'Advanced Settings'. At the bottom, there's a 'Management' link. The main content area displays a message: 'Your Basic license is inactive' with a green checkmark, followed by 'Your license will never expire.' Below this are two boxes: 'Update your license' (with a 'Update license' button) and 'Start a 30-day trial' (with a 'Start trial' button, also highlighted with a red box).

Management / License management

Elasticsearch

- Index Management
- Index Lifecycle Policies
- Rollup Jobs
- Cross Cluster Replication
- Remote Clusters
- License Management**
- 8.0 Upgrade Assistant

Kibana

- Index Patterns
- Saved Objects
- Spaces
- Reporting
- Advanced Settings

Management

Your Basic license is inactive

Your license will never expire.

Update your license

If you already have a new license, upload it now.

Update license

Start a 30-day trial

Experience what security, machine learning, and all our other Platinum features have to offer.

Start trial

Installation

- Click on the Start my Trial button in the resultant popup, as follows:

Start your free 30-day trial ×

This trial is for the full set of [Platinum features](#) of the Elastic Stack. You'll get immediate access to:

- Machine learning
- Alerting
- Graph capabilities
- JDBC and ODBC connectivity for SQL

Security features, such as authentication (native, AD/LDAP, SAML, PKI), role-based access control, and auditing, require configuration. See the [documentation](#) for instructions.

By starting this trial, you agree that it is subject to these [terms and conditions](#).



Send basic feature usage statistics to Elastic periodically. [Read more](#)

[Cancel](#)

[Start my trial](#)

Installation

The screenshot shows the Elasticsearch License Management interface. On the left, there's a sidebar with icons and a navigation bar. The main area displays a message about a trial license, followed by three options: Update your license, Extend your trial, and Revert to Basic license.

Management / License management

Elasticsearch

- Index Management
- Index Lifecycle Policies
- Rollup Jobs
- Cross Cluster Replication
- Remote Clusters
- Watcher
- License Management
- 8.0 Upgrade Assistant

Kibana

- Index Patterns
- Saved Objects

Your Trial license is active

Your license will expire on June 20, 2019 10:49 AM +08

Update your license

If you already have a new license, upload it now.

Update license

Extend your trial

If you'd like to continue using security, machine learning, and our other awesome Platinum features, request an extension now.

Extend trial

Revert to Basic license

You'll revert to our free features and lose access to security, machine learning and other Platinum features.

Revert to Basic

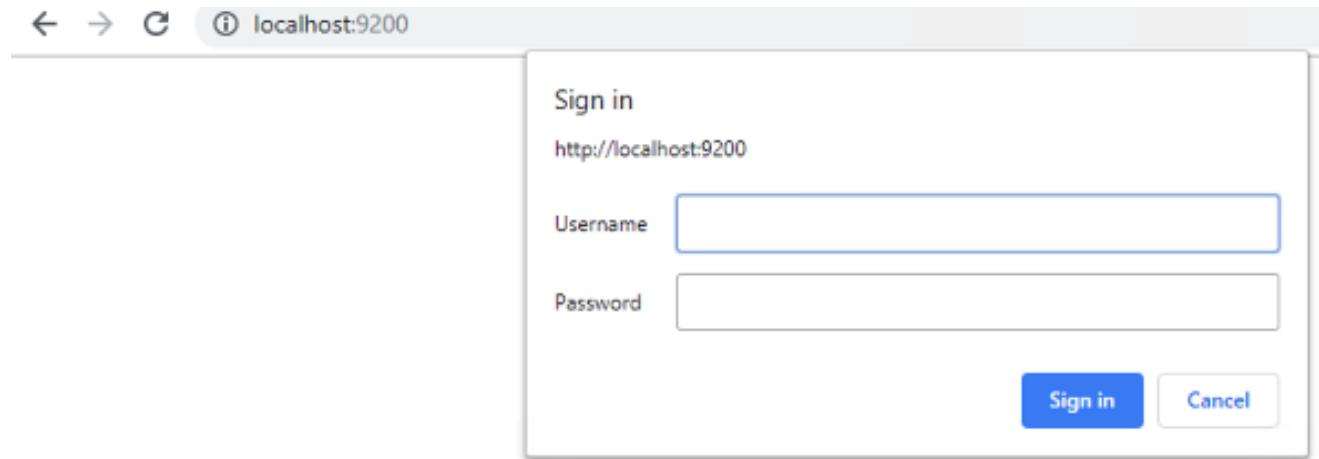
Installation

- Open elasticsearch.yml, which can be found under the \$ES_HOME/config folder, and add the following line at the end of the file to enable X-Pack and restart Elasticsearch and Kibana:

```
xpack.security.enabled: true
```

Installation

- Now, when you try to access Elasticsearch via `http://localhost:9200`, the user will be prompted for login credentials, as shown in the following screenshot:



Generating passwords for default users

- Let's generate the passwords for the reserved/default users—elastic, kibana, apm_system, remote_monitoring_user, beats_system, and logstash_system—by executing the following command:

```
$ ES_HOME>bin/elasticsearch-setup-passwords  
interactive
```

Generating passwords for default users

- You should get the following logs/prompts to enter the password for the reserved/default users:

https://github.com/fenago/elasticsearch/blob/master/snippets/8_3.txt

Generating passwords for default users

- To verify X-Pack's installation and enforcement of security, point your web browser to `http://localhost:9200/` to open Elasticsearch.
- You should be prompted to log in to Elasticsearch. To log in, you can use the built-in `elastic` user and `elastic` password.
- Upon logging in, you should see the following response:

https://github.com/fenago/elasticsearch/blob/master/snippets/8_4.txt

Generating passwords for default users

- Add the following credentials in the kibana.yml file, which can be found under \$KIBANA_HOME/config, and save it:

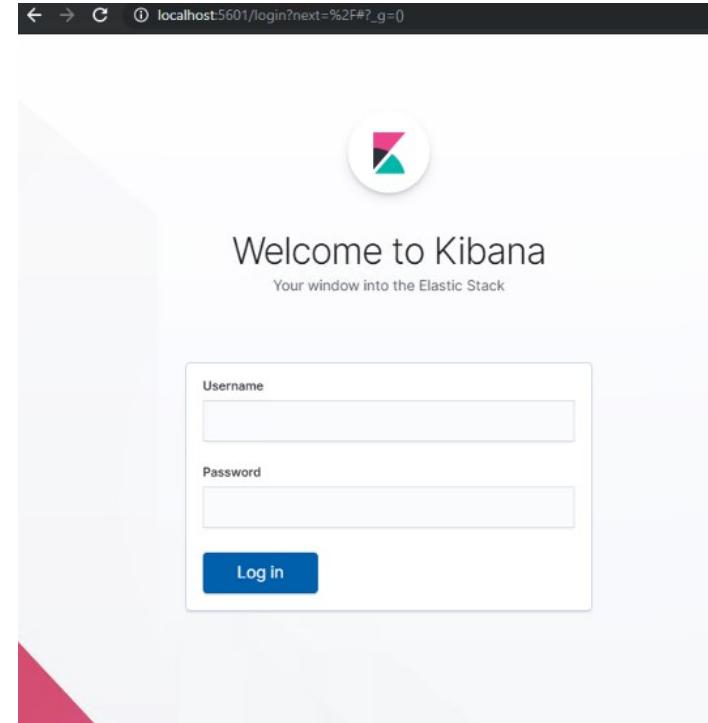
```
elasticsearch.username: "kibana"  
elasticsearch.password: "kibana"
```

Generating passwords for default users

Start Kibana:

\$KIBANA_HOME>bin/kibana

- To verify that the authentication is in place, go to <http://localhost:5601/> to open Kibana.
- You should be prompted to login to Kibana.
- To log in, you can use the built-in elastic user and the elastic password, as follows:



Configuring X-Pack

- Elasticsearch supports the following features and settings in the `elasticsearch.yml` file:

Feature	Setting	Description
Machine learning	<code>xpack.ml.enabled</code>	Set this to false to disable X-Pack machine learning features
Monitoring	<code>xpack.monitoring.enabled</code>	Set this to false to disable Elasticsearch's monitoring features
Security	<code>xpack.security.enabled</code>	Set this to false to disable X-Pack security features
Watcher	<code>xpack.watcher.enabled</code>	Set this to false to disable Watcher

Configuring X-Pack

- Kibana supports the following features and settings in the kibana.yml file:

Feature	Setting	Description
Machine learning	<code>xpack.ml.enabled</code>	Set this to false to disable X-Pack machine learning features
Monitoring	<code>xpack.monitoring.enabled</code>	Set this to false to disable Kibana's monitoring features
Security	<code>xpack.security.enabled</code>	Set this to false to disable X-Pack security features
Graph	<code>xpack.graph.enabled</code>	Set this to false to disable X-Pack graph features
Reporting	<code>xpack.reporting.enabled</code>	Set this to false to disable X-Pack reporting features

Securing Elasticsearch and Kibana

The X-Pack security module provides the following ways to secure Elastic Stack:

- User authentication and user authorization
- Node/Client authentication and channel encryption
- Auditing

User authentication

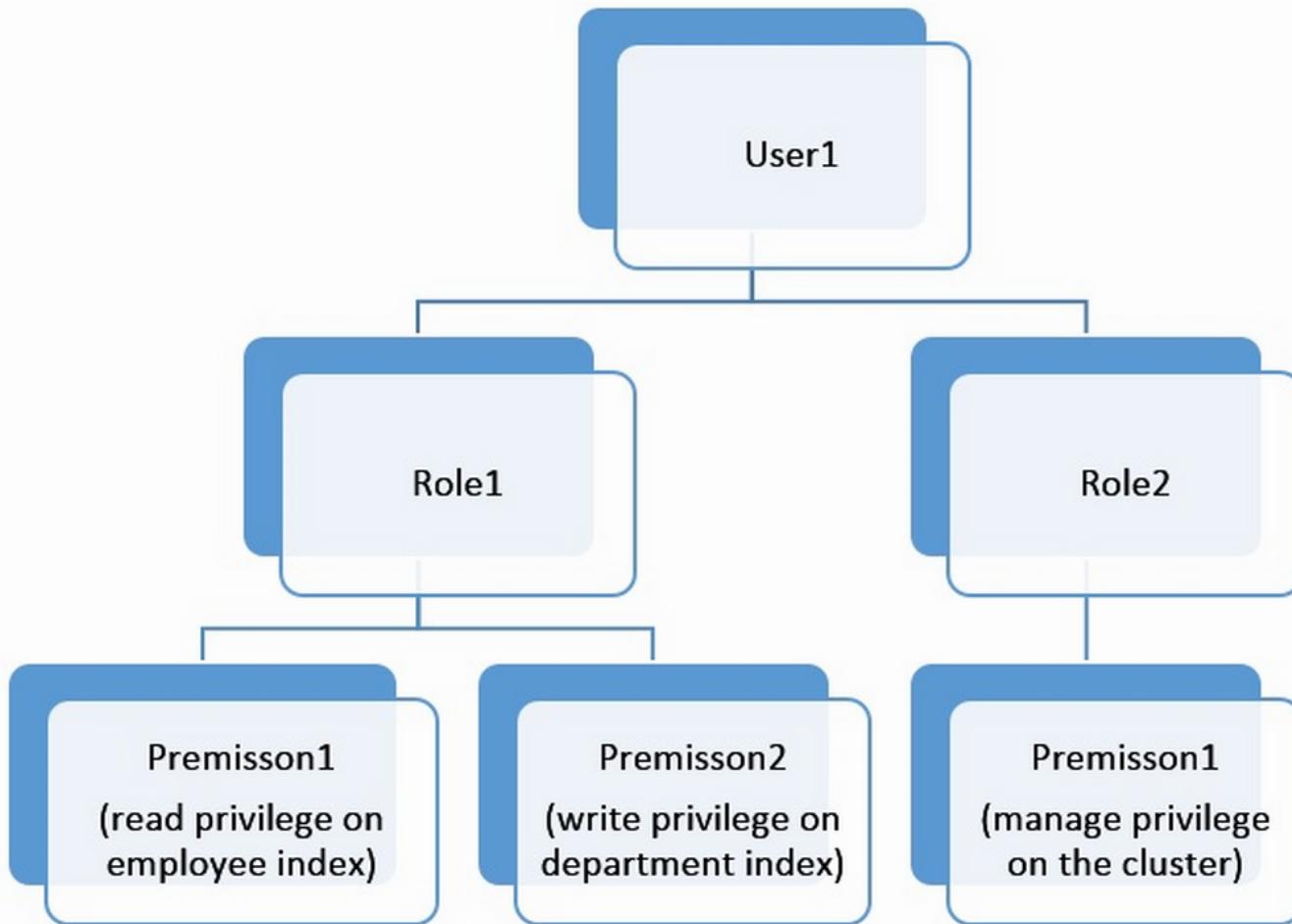
- User authentication is the process of validating the user and thus preventing unauthorized access to the Elastic Cluster.
- In the X-Pack security module, the authentication process is handled by one or more authentication services called realms.
- The Security module provides two types of realms, namely internal realms and external realms.

- For example, the following snippet shows the configuration for the realm chain containing native, file, and ldap:

```
xpack.security.authc:  
  realms:  
    native:  
      type: native  
      order: 0  
    file:  
      type: file  
      order: 1  
    ldap_server:  
      type: ldap  
      order: 2  
      url: 'url_to_ldap_server'
```

User authorization

- Once the user has been successfully authenticated, the authorization process kicks in.
- Authorization determines whether the user behind the request has enough permissions to execute a particular request.
- In X-Pack security, secured resources are the foundation of user-based security.



Security in action

- In this section, we will look into creating new users, creating new roles, and associating roles with users.
- Let's import some sample data and use it to understand how security works.
- Save the following data to a file named data.json:
https://github.com/fenago/elasticsearch/blob/master/snippets/8_5.txt

Security in action

- Navigate to the directory where the file is stored and execute the following command to import the data into Elasticsearch:

```
$ directoy_of_data_file> curl -s -H "Content-Type: application/json" -u elastic:elastic -XPOST http://localhost:9200/_bulk --data-binary @data.json
```

Security in action

- To check whether the import was successful, execute the following command and validate the count of documents:

```
curl -s -H "Content-Type: application/json" -u  
elastic:elastic -XGET  
http://localhost:9200/employee,department/_count  
{"count":6,"_shards":{"total":10,"successful":10,"skipped":  
0,"failed":0}}
```

Creating a new user

The screenshot shows the Kibana Management interface with the 'Users' section selected. The left sidebar includes links for Elasticsearch, Kibana, Logstash, Beats, and Security. The Security section has 'Users' and 'Roles' sub-links, both of which are highlighted with red boxes. The main content area displays a table of users with columns for Full Name, User Name, Email Address, Roles, and Reserved status. A 'Create new user' button is located in the top right corner of the table area.

Full Name ↑	User Name	Email Address	Roles	Reserved
	elastic		superuser	✓
	kibana		kibana_system	✓
	logstash_system		logstash_system	✓
	beats_system		beats_system	✓
	apm_system		apm_system	✓
	remote_monitoring_user		remote_monitoring_collector, remote_monitoring_agent	✓

Creating a new user

Users

Search...

Default Users

Full Name ↑	User Name	Email Address	Roles	Reserved
	elastic		superuser	✓
	kibana		kibana_system	✓
	logstash_system		logstash_system	✓
	beats_system		beats_system	✓
	apm_system		apm_system	✓
	remote_monitoring_user		remote_monitoring_collector, remote_monitoring_agent	✓

Rows per page: 20 ▾

[Create new user](#)

Creating a new user

- To create a new user, click on the Create new user button and enter the required details, as shown in the following screenshot.
- Then, click on Create user:

New user

Username

Password

Confirm password

Full name

Email address

Roles

Create user

Cancel

Creating a new user

- The user gets HTTP status code 403, stating that the user is not authorized to carry out the operation:

```
curl -s -H "Content-Type: application/json" -u user1:password -  
XGET http://localhost:9200
```

Response:

```
{"error": {"root_cause": [{"type": "security_exception", "reason": "action [cluster:monitor/main] is unauthorized for user [user1"]}], "type": "security_exception", "reason": "action [cluster:monitor/main] is unauthorized for user [user1"]}, "status": 403}
```

Deleting a user

Users

Create new user

Delete 1 user

Search...

<input type="checkbox"/> Full Name ↑	User Name	Email Address	Roles	Reserved
<input type="checkbox"/> user1	user1	user1@packt.com		
<input checked="" type="checkbox"/> user2	user2	user2@packt.com		
<input type="checkbox"/> elastic			superuser	✓
<input type="checkbox"/> kibana			kibana_system	✓
<input type="checkbox"/> logstash_system			logstash_system	✓
<input type="checkbox"/> beats_system			beats_system	✓
<input type="checkbox"/> apm_system			apm_system	✓
<input type="checkbox"/> remote_monitoring_user			remote_monitoring_collector, remote_monitoring_agent	✓

Rows per page: 20 ▾

Changing the password

Edit user1 user

Username

Username's cannot be changed after creation.

Full name

Email address

Roles

Password

[Change password](#)

[Update user](#) [Cancel](#) [Delete user](#)

Creating a new role

The screenshot shows the Elasticsearch Management interface with the 'Roles' page selected. The left sidebar lists various management sections: Elasticsearch (Index Management, Index Lifecycle Policies, Rollup Jobs, Cross Cluster Replication, Remote Clusters, Watcher, License Management, 8.0 Upgrade Assistant), Kibana (Index Patterns, Saved Objects, Spaces, Reporting, Advanced Settings), Logstash (Pipelines), Beats (Central Management), and Security (Users, Roles). The 'Roles' tab under Security is highlighted with a red box. The main area displays a table of roles with a 'Create role' button at the top right, also highlighted with a red box.

Role	Reserved
apm_system	✓
apm_user	✓
beats_admin	✓
beats_system	✓
code_admin	✓
code_user	✓
ingest_admin	✓
kibana_dashboard_only_user	✓
kibana_system	✓
kibana_user	✓
logstash_admin	✓
logstash_system	✓
machine_learning_admin	✓
machine_learning_user	✓
monitoring_user	✓
remote_monitoring_agent	✓
remote_monitoring_collector	✓
reporting_user	✓
rollup_admin	✓
rollup_user	✓

Management / Users / Create

Elasticsearch

- Index Management
- Index Lifecycle Policies
- Rollup Jobs
- Cross Cluster Replication
- Remote Clusters
- Watcher
- License Management
- 8.0 Upgrade Assistant

Kibana

- Index Patterns
- Saved Objects
- Spaces
- Reporting
- Advanced Settings

Logstash

- Pipelines

Beats

- Central Management

Security

- Users
- Roles**

Create role

Set privileges on your Elasticsearch data and control access to your Kibana spaces.

Role name
monitor_role

Elasticsearch hide

Cluster privileges
Manage the actions this role can perform against your cluster. Learn more

monitor

Run As privileges
Allow requests to be submitted on behalf of other users. Learn more

Add a user...

Index privileges
Control access to the data in your cluster. Learn more

Indices Privileges Granted fields (optional)

Grant read privileges to specific documents

Kibana hide

Minimum privileges for all spaces
Specify the minimum actions users can perform in your spaces.

none
No access to spaces

Higher privileges for individual spaces
Grant more privileges on a per space basis. For example, if the privileges are **read** for all spaces, you can set the privileges to **all** for an individual space.

View summary of spaces p

Create role

Creating a new user

Edit user1 user

Username

Username's cannot be changed after creation.

Full name

Email address

Roles

- mo
- monitor_role
- monitoring_user
- remote_monitoring_agent
- remote_monitoring_collector

Update user

Cancel

Delete user

Creating a new user

- Now, let's validate that user1 can access some cluster/node details APIs:

https://github.com/fenago/elasticsearch/blob/master/snippets/8_6.txt

```
curl -u user2:password "http://localhost:9200/_cluster/health?pretty"
{
  "error" : {
    "root_cause" : [
      {
        "type" : "security_exception",
        "reason" : "action [cluster:monitor/main] is unauthorized for user [user2]"
      }
    ],
    "type" : "security_exception",
    "reason" : "action [cluster:monitor/main] is unauthorized for user [user2]"
  },
  "status" : 403
}
```

Deleting or editing a role

Roles

Apply roles to groups of users and manage permissions across the stack

Role ↑	Reserved ⓘ
apm_system	✓
apm_user	✓
beats_admin	✓
beats_system	✓
code_admin	✓
code_user	✓
ingest_admin	✓
kibana_dashboard_only_user	✓
kibana_system	✓
kibana_user	✓
logstash_admin	✓
logstash_system	✓
machine_learning_admin	✓
machine_learning_user	✓
<input checked="" type="checkbox"/> monitor_role	

Deleting or editing a role

- You can also delete the role from this page:

The screenshot shows the 'Edit role' page for a role named 'monitor_role'. The 'Role name' field contains 'monitor_role'. Below it, a note states: 'A role's name cannot be changed once it has been created.'

Elasticsearch Cluster privileges: Manage the actions this role can perform against your cluster. Learn more. The 'monitor' and 'manage' checkboxes are selected and highlighted with a red box.

Run As privileges: Allow requests to be submitted on behalf of other users. Learn more. A dropdown menu labeled 'Add a user...' is shown.

Index privileges: Control access to the data in your cluster. Learn more. It includes sections for 'Indices', 'Privileges', and 'Granted fields (optional)'. A checkbox for 'Grant read privileges to specific documents' is checked. A blue button labeled '+ Add index privilege' is visible.

Kibana Minimum privileges for all spaces: Specify the minimum actions users can perform in your spaces. A dropdown menu shows 'none' and a note: 'No access to spaces'.

Kibana Higher privileges for individual spaces: Grant more privileges on a per space basis. A note: 'Grant more privileges on a per space basis. For example, if the privileges are read for all spaces, you can set the privileges to all for an individual space.' A blue button labeled '+ Add space privilege' is visible. A link 'View summary of spaces pri' is at the bottom right.

At the bottom, there are 'Update role' and 'Cancel' buttons, with 'Update role' also highlighted with a red box.

Document-level security or field-level security

Create role

Set privileges on your Elasticsearch data and control access to your Kibana spaces.

Role name
employee_read

Elasticsearch hide

Cluster privileges
Manage the actions this role can perform against your cluster. [Learn more](#)

Run As privileges
Allow requests to be submitted on behalf of other users. [Learn more](#)

Index privileges
Control access to the data in your cluster. [Learn more](#)

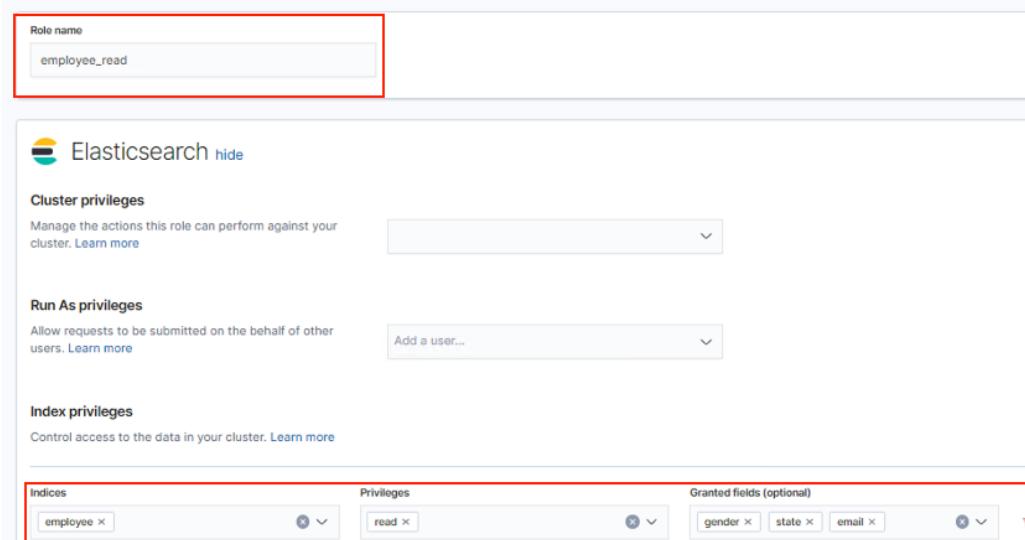
Indices
employee

Privileges
read

Granted fields (optional)
gender state email

Grant read privileges to specific documents

[Add index privilege](#)



- Assign the newly created role to user2:

Edit user2 user

Username

A red rectangular box highlights the "user2" input field.

Username's cannot be changed after creation.

Full name

Email address

Roles

A red rectangular box highlights the "employee_read" input field. To the right of the input field is a small icon consisting of a circular arrow and a checkmark.

Password

[Change password](#)

[Update user](#)

[Cancel](#)

[Delete user](#)

Document-level security or field-level security

- Now, let's search in the employee index and check what fields were returned in the response.
- As we can see in the following response, we have successfully restricted the user from accessing salary and address details:

https://github.com/fenago/elasticsearch/blob/master/snippets/8_7.txt

Create role

Set privileges on your Elasticsearch data and control access to your Kibana spaces.

Role name

department_IT_role



Elasticsearch [hide](#)

Cluster privileges

Manage the actions this role can perform against your cluster. [Learn more](#)

Run As privileges

Allow requests to be submitted on the behalf of other users. [Learn more](#)

 Add a user...

Index privileges

Control access to the data in your cluster. [Learn more](#)

Indices

department

Privileges

read

Granted fields (optional)

*



Grant read privileges to specific documents

Granted documents query

```
{"match":{"name": "IT"}}
```

Document-level security or field-level security

- Associate the newly created role with user1:

Edit user1 user

Username
user1

Username's cannot be changed after creation.

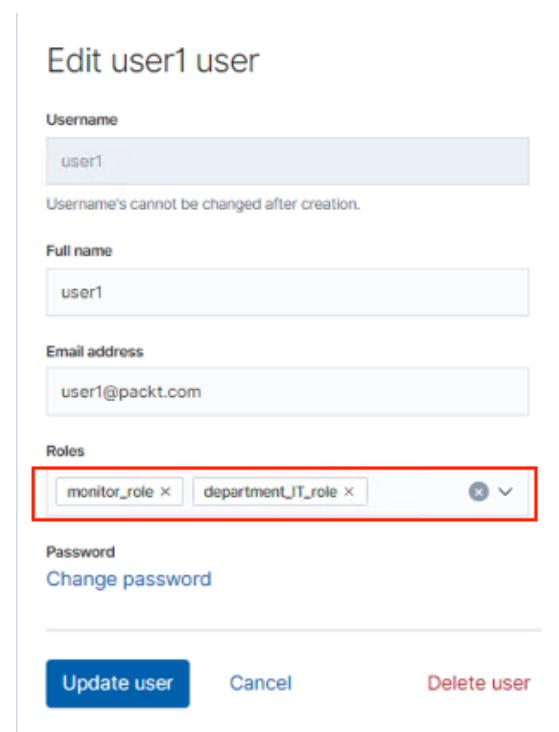
Full name
user1

Email address
user1@packt.com

Roles
monitor_role × department_IT_role ×

>Password
[Change password](#)

[Update user](#) [Cancel](#) [Delete user](#)

A screenshot of a user edit form titled "Edit user1 user". The form includes fields for Username (user1), Full name (user1), Email address (user1@packt.com), and Roles. The "Roles" field contains two items: "monitor_role" and "department_IT_role", which are highlighted with a red rectangular box. Below the roles is a "Password" field with a "Change password" link. At the bottom are buttons for "Update user", "Cancel", and "Delete user".

Document-level security or field-level security

- Let's verify that it is working as expected by executing a search against the department index using the user1 credentials:

https://github.com/fenago/elasticsearch/blob/master/snippets/8_8.txt

X-Pack security APIs

User Management APIs

- This provides a set of APIs to create, update, or delete users from the native realm.
- The following is a list of available APIs and how to use them:

https://github.com/fenago/elasticsearch/blob/master/snippets/8_9.txt

User Management APIs

- Example 1: Create a new user, user3, with monitor_role assigned to it:

https://github.com/fenago/elasticsearch/blob/master/snippets/8_10.txt

User Management APIs

- Example 2: Get the list of all users:

```
curl -u elastic:elastic -XGET
```

```
http://localhost:9200/_xpack/security/user?pretty
```

- Example 3: Delete user3:

```
curl -u elastic:elastic -XDELETE
```

```
http://localhost:9200/_xpack/security/user/user3
```

Response:

```
{"found":true}
```

User Management APIs

- Example 4: Change the password:

```
curl -u elastic:elastic -XPUT  
http://localhost:9200/_xpack/security/user/user2/_passwor  
d -H "content-type: application/json" -d "{\"password\":  
\"newpassword\"}"
```

Role Management APIs

- This provides a set of APIs to create, update, remove, and retrieve roles from the native realm.
- The list of available APIs under this section, as well as information on what they do, is as follows:

https://github.com/fenago/elasticsearch/blob/master/snippets/8_11.txt

Role Management APIs

Let's take a look at a few examples of managing different roles using APIs:

- Example 1: Create a new role with field-level security imposed on the employee index:

https://github.com/fenago/elasticsearch/blob/master/snippets/8_12.txt

Role Management APIs

- Example 2: Get the details of a specific role:

https://github.com/fenago/elasticsearch/blob/master/snippets/8_13.txt

Role Management APIs

- Example 3: Delete a role:

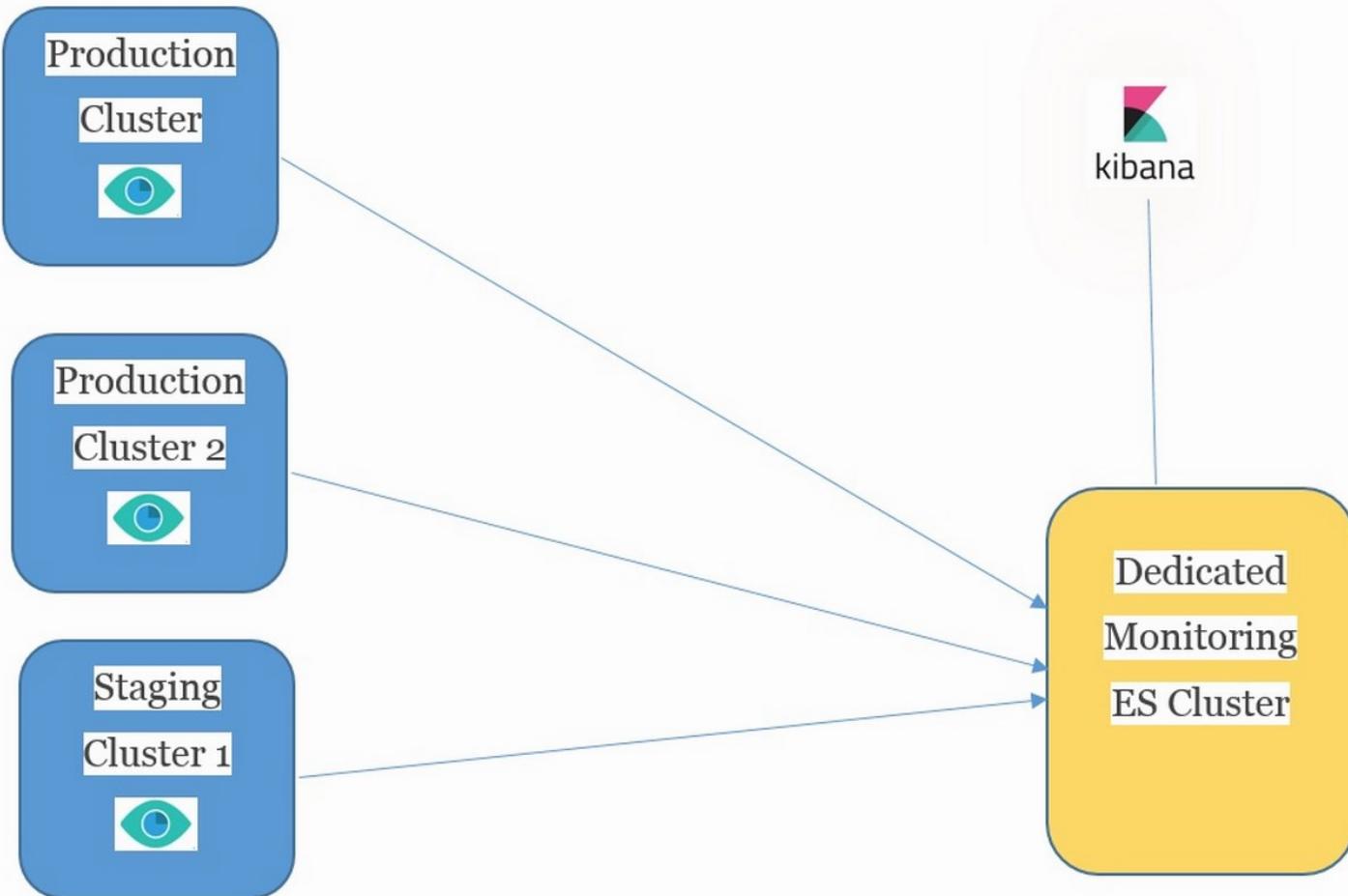
```
curl -u elastic:elastic -XDELETE  
http://localhost:9200/_xpack/security/role/employee_read
```

Response:

```
{"found":true}
```

Monitoring Elasticsearch

- Elasticsearch exposes a rich set of APIs, known as stats APIs, to monitor Elasticsearch at the cluster, node, and indices levels.
- Some of these APIs are `_cluster/stats`, `_nodes/stats`, and `myindex/stats`.
- These APIs provide state/monitoring information in real time, and the statistics that are presented in these APIs are point-in-time and in `.json` format.

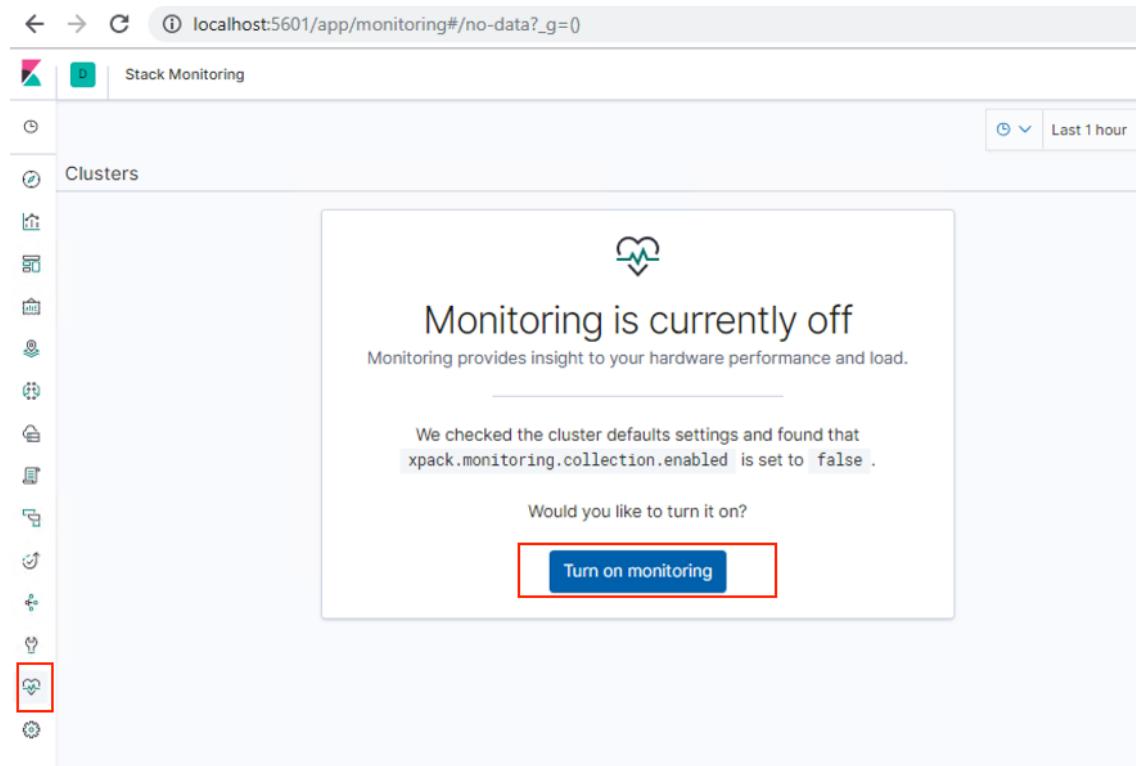


Monitoring Elasticsearch

- If a dedicated monitoring cluster is set up, then we need to configure where to send/ship the metrics to in the monitored instances.
- This can be configured in the elasticsearch.yml file of each node, as shown in the following code:

```
xpack.monitoring.exporters:  
  id1:  
    type: http  
    host: ["http://dedicated_monitoring_cluster:port"]
```

Monitoring UI



The screenshot shows the Elasticsearch Stack Monitoring interface at the URL `localhost:5601/app/monitoring#/no-data?_g=0`. The left sidebar has a 'Clusters' section with a red box around the monitoring icon. The main area displays a message: "Monitoring is currently off". It explains that monitoring provides insight into hardware performance and load, and that the setting `xpack.monitoring.collection.enabled` is set to `false`. A blue button labeled "Turn on monitoring" is present, with a red box highlighting it.

← → ⌛ ⓘ localhost:5601/app/monitoring#/no-data?_g=0

Kubernetes Stack Monitoring

Clusters

Last 1 hour

Monitoring is currently off

Monitoring provides insight to your hardware performance and load.

We checked the cluster defaults settings and found that `xpack.monitoring.collection.enabled` is set to `false`.

Would you like to turn it on?

Turn on monitoring

- Once you click on Turn on monitoring, the cluster settings will update, which can be verified by using the following API:

```
curl -u elastic:elastic -XGET http://localhost:9200/_cluster/settings?pretty
{
  "persistent" : {
    "xpack" : {
      "monitoring" : {
        "collection" : {
          "enabled" : "true"
        }
      }
    }
  },
  "transient" : { }
}
```

Monitoring UI

The screenshot displays the Elasticsearch Monitoring UI, showing the status and metrics for the elasticsearch cluster.

Top cluster alerts:

- A yellow alert box indicates a **Medium severity alert**: "Elasticsearch cluster status is yellow. Allocate missing replica shards." (Last checked May 23, 2019 4:16:36 PM (triggered 14 min ago)).

Elasticsearch: Health is yellow. Trial license will expire on June 20, 2019.

Overview	
Version	7.0.0
Uptime	2 days
Jobs	0

Nodes: 1	
Disk Available	87.06% 522.1 GB / 599.7 GB
JVM Heap	15.74% 155.8 MB / 989.9 MB

Indices: 11	
Documents	1,732
Disk Usage	1.8 MB
Primary Shards	11
Replica Shards	0

Kibana: Health is green.

Overview	
Requests	2
Max. Response Time	58 ms

Instances: 1	
Connections	6
Memory Usage	14.24% 207.3 MB / 1.4 GB

Elasticsearch metrics

- You can monitor the Elasticsearch performance data at a cluster level, node level, or index level.
- The Elasticsearch Monitoring UI provides three tabs, each displaying the metrics at the cluster, node, and index levels.
- The three tabs are Overview, Nodes, and Indices, respectively.

Overview tab

- Cluster-level metrics provide aggregated information across all the nodes and is the first place one should look when monitoring an Elasticsearch cluster.
- Cluster-level metrics are displayed in the Overview tab and can be navigated to by clicking on the Overview link under the Elasticsearch section found in the landing page of the Monitoring UI.

- The Overview tab provides key metrics that indicate the overall health of an Elasticsearch cluster:



Nodes tab

- Clicking on the Nodes tab displays the summary details of each node present in the cluster, as shown in the following screenshot:

The screenshot shows the Elasticsearch interface with the 'Clusters / elasticsearch / Elasticsearch' path selected. The 'Nodes' tab is active, displaying cluster statistics and a detailed view of a single node.

Cluster Statistics:

Status	Nodes	Indices	Memory	Total Shards	Unassigned Shards	Documents	Data
Yellow	1	42	474.6 MB / 989.9 MB	75	33	304,564	176.2 MB

Node Details:

Name	Status	CPU Usage	Load Average	JVM Memory	Disk Free Space	Shards
★ MADSH01-APM01 127.0.0.1:9300	Online	7% ↑ 10% max 0% min	N/A	N/A max N/A min	521.5 GB ↓ 522.1 GB max 521.5 GB min	42

Filter Nodes...
Rows per page: 20



★ Overview Advanced

Status

Online

Transport Address

127.0.0.1:9300

JVM Heap

45%

Free Disk Space

521.5 GB

Documents

305k

Data

176.5 MB

Indices

42

Shards

42

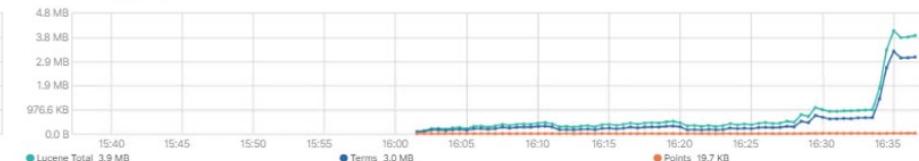
Type

Master Node

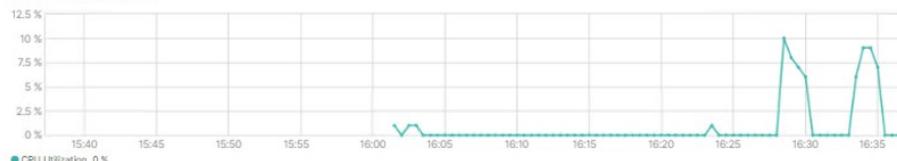
JVM Heap (MB) ⚡



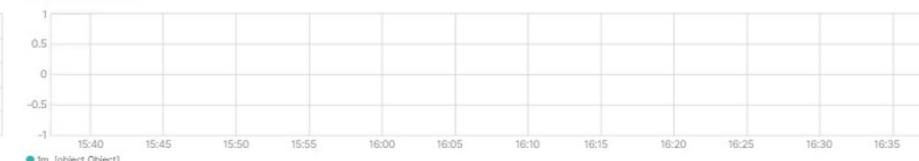
Index Memory (MB) ⚡



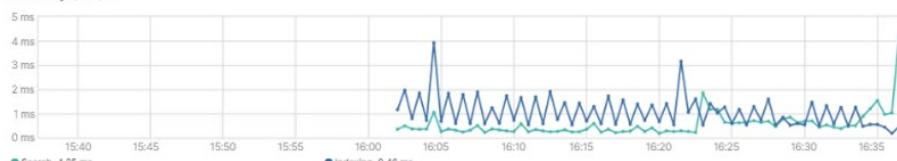
CPU Utilization (%) ⚡



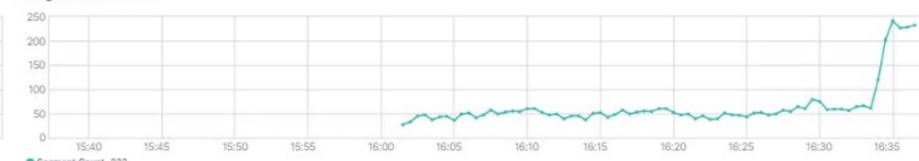
System Load ⚡



Latency (ms) ⚡



Segment Count ⚡



Shard Legend

Primary | Replica | Relocating | Initializing | Unassigned Primary | Unassigned Replica

Indices System indices



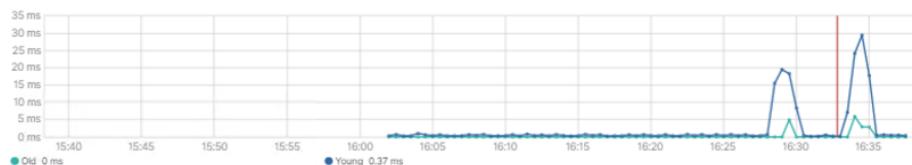
★ Overview Advanced

Status	Transport Address	JVM Heap	Free Disk Space	Documents	Data	Indices	Shards	Type
Online	127.0.0.1:9300	49%	521.5 GB	305.3k	177.2 MB	42	42	Master Node

GC Count



GC Duration (ms)



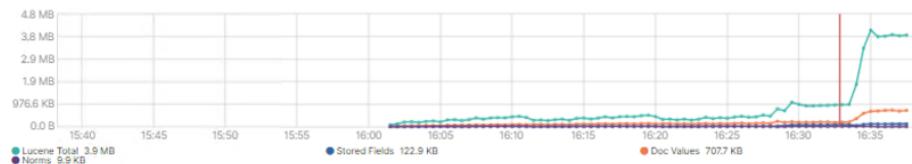
JVM Heap (MB)



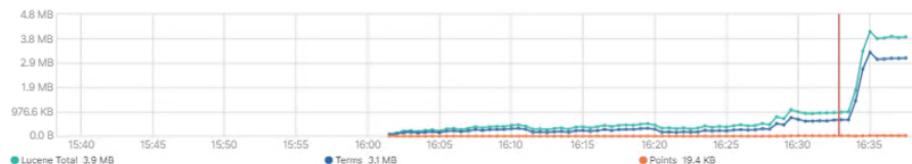
CPU Utilization (%)



Index Memory - Lucene 1 (MB)



Index Memory - Lucene 2 (MB)



Index Memory - Lucene 3 (MB)



Index Memory - Elasticsearch (KB)



The Indices tab

The screenshot shows the Elasticsearch Indices tab in the Kibana interface. At the top, there are cluster statistics: 1 node (Yellow), 42 indices, 471.6 MB / 989.9 MB memory usage, 75 total shards, 33 unassigned shards, 305,452 documents, and 177.0 MB data. Below this, a table lists 20 indices, each with its name, status (all yellow), document count, data size, index rate, search rate, and unassigned shards. The indices are logstash logs from May 28 to June 14, 2014. The bottom of the table shows pagination for 20 rows per page.

Name	Status	Document Count	Data	Index Rate	Search Rate	Unassigned Shards
department	Yellow	3	4.0 kB	0/s	0/s	1
employee	Yellow	3	7.5 kB	0/s	0/s	1
logstash-2014-05-28	Yellow	1k	1.1 MB	0/s	0/s	1
logstash-2014-05-29	Yellow	7.9k	5.1 MB	0/s	0/s	1
logstash-2014-05-30	Yellow	9.1k	5.6 MB	0.72/s	0/s	1
logstash-2014-05-31	Yellow	9.4k	5.9 MB	0/s	0/s	1
logstash-2014-06-01	Yellow	10.4k	6.1 MB	0/s	0/s	1
logstash-2014-06-02	Yellow	11k	6.3 MB	0/s	0/s	1
logstash-2014-06-03	Yellow	7.6k	4.4 MB	0.98/s	0/s	1
logstash-2014-06-04	Yellow	6.9k	4.1 MB	0/s	0/s	1
logstash-2014-06-05	Yellow	8.5k	5.2 MB	0/s	0/s	1
logstash-2014-06-06	Yellow	11.2k	6.6 MB	0/s	0/s	1
logstash-2014-06-07	Yellow	10.7k	6.0 MB	1.17/s	0/s	1
logstash-2014-06-08	Yellow	13.1k	6.9 MB	0/s	0/s	1
logstash-2014-06-09	Yellow	10.7k	5.9 MB	0/s	0/s	1
logstash-2014-06-10	Yellow	10.6k	6.6 MB	0.8/s	0/s	1
logstash-2014-06-11	Yellow	8.1k	4.4 MB	0/s	0/s	1
logstash-2014-06-12	Yellow	10.1k	5.9 MB	0/s	0/s	1
logstash-2014-06-13	Yellow	10.9k	6.4 MB	0/s	0/s	1
logstash-2014-06-14	Yellow	10.4k	5.9 MB	0.99/s	0/s	1



Last 1 hour

Show dates

Refresh

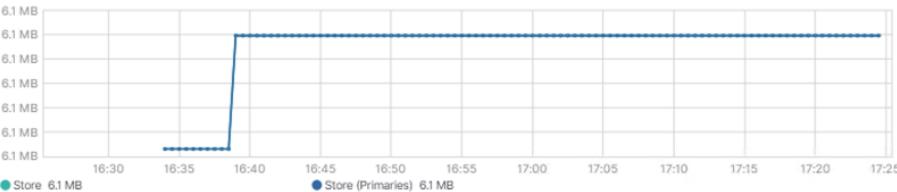
Overview Advanced

Status
Health: YellowTotal
6.1 MBPrimaries
6.1 MBDocuments
10.4kTotal Shards
2Unassigned Shards
1

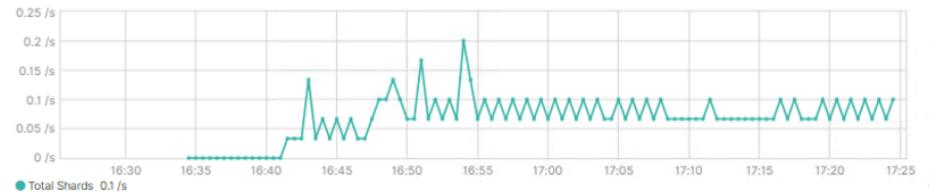
Index Memory (KB)



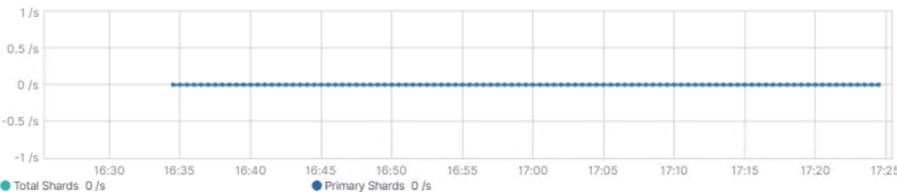
Disk (MB)



Search Rate (/s)



Indexing Rate (/s)



Segment Count



Document Count



Overview Advanced

Status
Health: Yellow

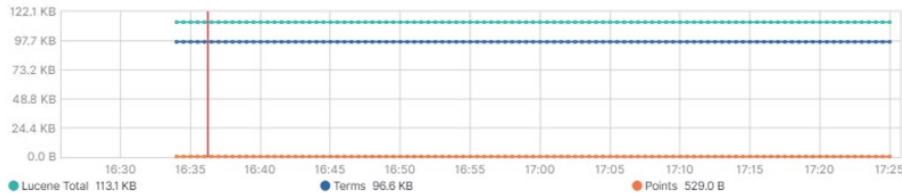
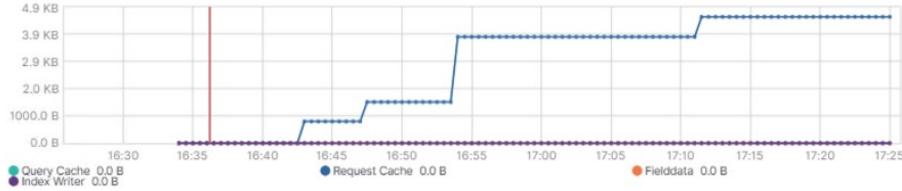
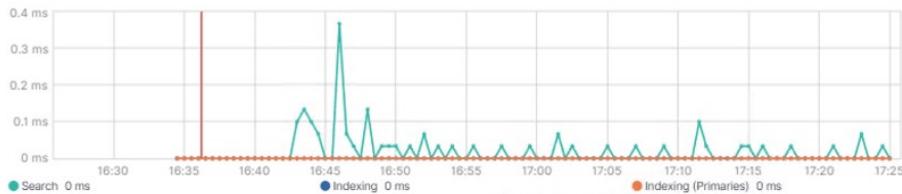
Total
6.1 MB

Primaries
6.1 MB

Documents
10.4k

Total Shards
2

Unassigned Shards
1

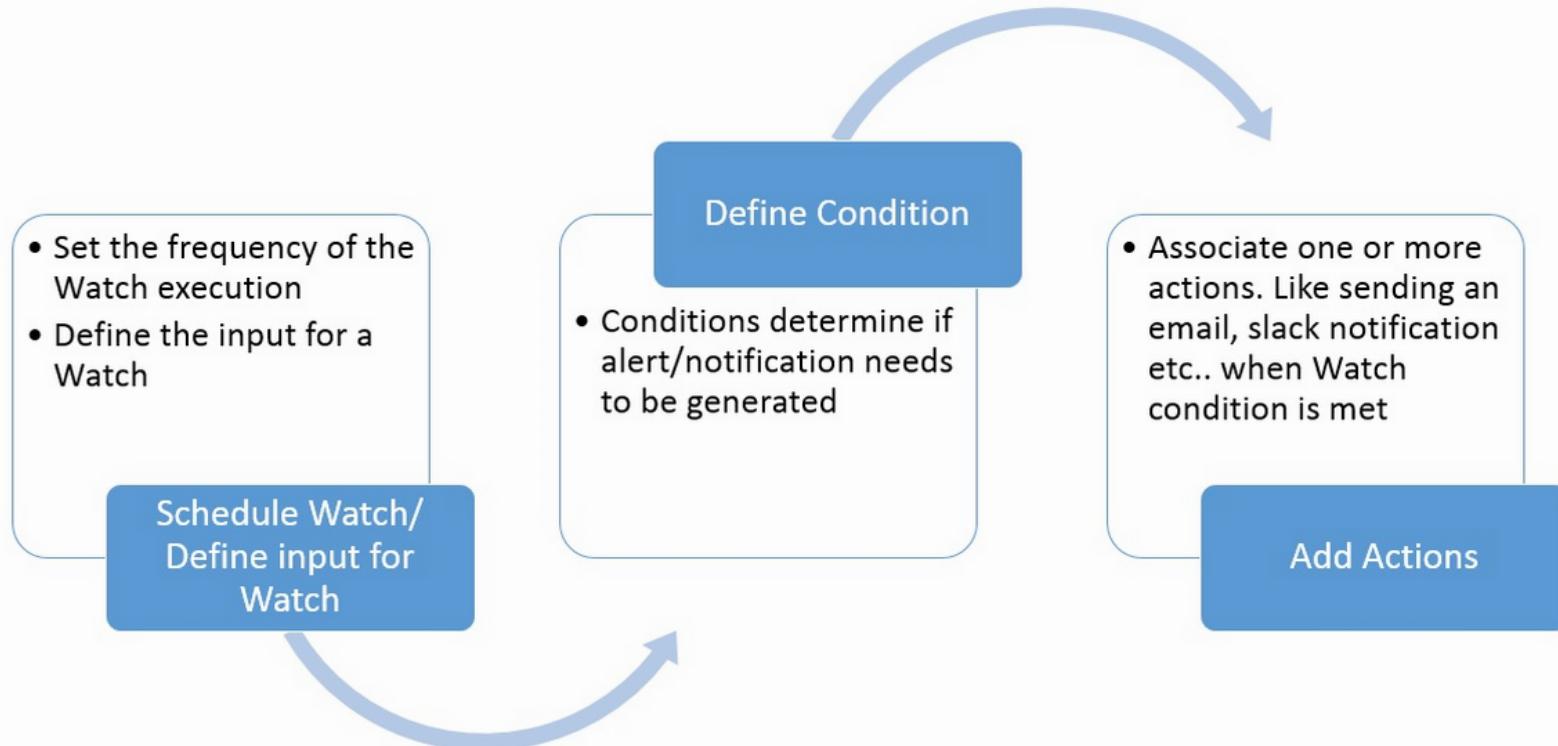
Index Memory - Lucene 1 (KB) ①Index Memory - Lucene 2 (KB) ①Index Memory - Lucene 3 (KB) ①Index Memory - Elasticsearch (KB) ①Request Rate ①Request Time (ms) ①

Alerting

It would be nice if the administrator gets notified when, for example, the following events occur:

- There is an outage in one of the servers being monitored
- An Elasticsearch Cluster turns red/yellow due to some nodes leaving the cluster
- Disk space/CPU utilization crosses a specific threshold
- There is an intrusion in the network
- There are errors reported in the logs

Anatomy of a watch



Anatomy of a watch

- Let's look into a sample watch and understand the building blocks of a watch in detail.
- The following code snippet creates a watch:

```
curl -u elastic:elastic -X POST
```

```
http://localhost:9200/_xpack/watcher/watch/logstash_error_watch -H  
'content-type: application/json' -d '{  
  "trigger": {"schedule": {"interval": "30s"}}, "input": {"search": {"request": {"in-  
  dices": ["logstash*"], "body": {"query": {"match": {"message": "error"} }}}}}, "con-  
  dition": {"compare": {"ctx.payload.hits.total": {"gt": 0}}}, "actions": {"log  
 _error": {"logging": {"text": "The number of errors in logs is  
 {{ctx.payload.hits.total}}"} }}}
```

Anatomy of a watch

- Example to specify hourly trigger: The following snippet shows how to specify an hourly trigger that triggers the watch every 45th minute of an hour:

```
{"trigger": {"schedule": {"hourly": {"minute": 45}}}}
```

Anatomy of a watch

- You can specify an array of minutes, too. The following snippet shows how to specify an hourly trigger that triggers the watch every 15th and 45th minute of an hour:

```
{"trigger": {"schedule": {"hourly": {"minute": [ 15,45]}}}}
```

- The following is an example of specifying that the watch should trigger daily at 8 PM:

```
{"trigger": {"schedule": {"daily": {"at": "20:00"}}}}
```

Anatomy of a watch

- The following is an example of specifying a watch to trigger weekly on Mondays at 10 AM and on Fridays at 8 PM:

```
{"trigger": {"schedule": {"weekly": [{"on": "monday", "at": "10:00"}, {"on": "friday", "at": "20:00"}]}}}
```

Anatomy of a watch

- The following is an example of specifying a schedule using cron syntax.
- The following snippet specifies a watch to be triggered hourly at the 45th minute:

```
{  
  "trigger" : {  
    "schedule" : {  
      "cron" : "0 45 * * * ?"  
    }  
  }  
}
```

Anatomy of a watch

- The payload can be accessed using the `ctx.payload.*` variable:

```
"input":{"search":{"request":{"indices":["logstash*"],"body":{  
"query":{"match":{"message":"error"}}}}}}
```

Anatomy of a watch

- condition: This section is used to specify a condition against the payload in order to determine whether an action needs to be executed or not:

```
"condition":{"compare":{"ctx.payload.hits.total":{"gt":0}}}
```

Anatomy of a watch

- actions: This section is used to specify one or more actions that need to be taken when the watch condition evaluates to true:

```
"actions": {"log_error": {"logging": {"text": "The number of errors in logs is {{ctx.payload.hits.total}}"}}}
```

Alerting in action

- Now that we know what a Watch is made up of, in this section, we will explore how to create, delete, and manage watches.
- You can create/delete/manage watches using the following software:
 1. Kibana Watcher UI
 2. X-Pack Watcher REST APIs

Creating a new alert

The screenshot shows the Elasticsearch Management interface with the 'Management / Watcher' tab selected. On the left, a sidebar lists various management features: Index Management, Index Lifecycle Policies, Rollup Jobs, Cross Cluster Replication, Remote Clusters, **Watcher** (which is highlighted with a red box), License Management, and 8.0 Upgrade Assistant. Below that, under 'Kibana', are Index Patterns, Saved Objects, Spaces, and Reporting.

The main content area is titled 'Create threshold alert' and includes a sub-instruction: 'Send out emails, slack messages and log events when specific parameters are hit'. Below this are two blue buttons: 'Create threshold alert' and 'Create advanced watch', which are also highlighted with a red box.

The main search area has a search bar labeled 'Search...', a 'Delete' button, and pagination controls showing '0-0 of 0' with arrows. The text 'No watches found.' is displayed in the center of this area. A second identical search area is visible below it, also showing '0-0 of 0'.

Threshold Alert

- Specify the name of the alert; choose the index to be used to query against, the time field, and the trigger frequency in the Threshold Alert UI:

Create a new threshold alert

Send an alert when a specific condition is met. This will run every 1 minute.

Name

Indices to query

Time field

Run watch every

 minutes

Use * to broaden your search query

Threshold Alert

Matching the following condition

WHEN count() OVER all documents IS ABOVE 100000 FOR THE LAST 5 minutes



Threshold Alert

- It provides three types of actions, that is, email, slack, and logging actions.
- One or more actions can be configured:

Will perform 1 action once met

Add new action

Logging

Log text

Number of logs : {{ctx.metadata.name}} has exceeded the threshold

Remove Logging Action

Log a sample message now

Creating a new alert

- Clicking on Save will save the watch in the watches index and can be validated using the following query:

```
curl -u elastic:elastic -XGET  
http://localhost:9200/.watches/_search?q=metadata.name  
:logs_watch
```

Advanced Watch

New watch

Save Cancel

Edit	Simulate
------	----------

ID
errored_logs_watch

Name
errored_logs_watch

Watch JSON (Syntax)

```
8+   "search": {  
9+     "request": {  
10+       "body": {  
11+         "size": 0,  
12+         "query": {  
13+           "match": {"message": "error"}  
14+         }  
15+       },  
16+       "indices": [  
17+         "logs"  
18+       ]  
19+     }  
20+   },  
21+   "condition": {  
22+     "compare": {  
23+       "ctx.payload.hits.total": {  
24+         "gte": 10  
25+       }  
26+     }  
27+   },  
28+   "actions": {  
29+     "my-logging-action": {  
30+       "logging": {  
31+         "text": "There are {{ctx.payload.hits.total}} documents in your index. Threshold is 10."  
32+       }  
33+     }  
34+   }  
35+ }  
36 ]
```

Advanced Watch

- Clicking on Save will save the watch in the watches index and can be validated using the following query:

```
curl -u elastic:elastic -XGET
```

```
http://localhost:9200/.watches/_search?q=metadata.name:errored_logs_watch
```

- Since we have configured logging as the action, when the alert is triggered, the same can be seen in elasticsearch.log:

```
[2019-05-23T17:43:07.809][INFO ][o.e.x.w.a.l.ExecutableLoggingAction] [MADSH01-APM01] The number of errors in logs is 39
[2019-05-23T17:43:37.379][INFO ][o.e.x.w.a.l.ExecutableLoggingAction] [MADSH01-APM01] The number of errors in logs is 39
[2019-05-23T17:44:07.343][INFO ][o.e.x.w.a.l.ExecutableLoggingAction] [MADSH01-APM01] The number of errors in logs is 39
[2019-05-23T17:44:37.380][INFO ][o.e.x.w.a.l.ExecutableLoggingAction] [MADSH01-APM01] The number of errors in logs is 39
[2019-05-23T17:45:07.373][INFO ][o.e.x.w.a.l.ExecutableLoggingAction] [MADSH01-APM01] The number of errors in logs is 39
[2019-05-23T17:45:37.393][INFO ][o.e.x.w.a.l.ExecutableLoggingAction] [MADSH01-APM01] The number of errors in logs is 39
[2019-05-23T17:46:04.394][INFO ][o.e.x.w.a.l.ExecutableLoggingAction] [MADSH01-APM01] There are 39 documents in your index. Threshold is 10.
[2019-05-23T17:46:07.383][INFO ][o.e.x.w.a.l.ExecutableLoggingAction] [MADSH01-APM01] The number of errors in logs is 39
[2019-05-23T17:46:37.428][INFO ][o.e.x.w.a.l.ExecutableLoggingAction] [MADSH01-APM01] The number of errors in logs is 39
```

Deleting/deactivating/editing a watch

Create threshold alert
Send out emails, slack messages and log events when specific parameters are hit

[Create threshold alert](#) [Create advanced watch](#)

ID ↑	Name	State	Comment	Last Fired	Last Triggered	
<input type="checkbox"/> 252f1c48-98f...	logs_watch	✓ OK				
<input type="checkbox"/> 988aff02-dc8...	logs_error_watch	✓ OK		21 minutes ago	a few seconds...	
<input checked="" type="checkbox"/> errored_logs_...	errored_logs_watch	✓ OK				
I2RVLSk2Rr6l...	X-Pack Monitoring: Cluster Status...	Firing		a minute ago	a minute ago	
I2RVLSk2Rr6l...	X-Pack Monitoring: Nodes Chang...	✓ OK				
I2RVLSk2Rr6l...	X-Pack Monitoring: Elasticsearch ...	✓ OK				
I2RVLSk2Rr6l...	X-Pack Monitoring: Kibana Versio...	✓ OK				
I2RVLSk2Rr6l...	X-Pack Monitoring: Logstash Vers...	✓ OK				
I2RVLSk2Rr6l...	X-Pack Monitoring: License Expira...	✓ OK				
<input type="checkbox"/> logstash_error...		Firing		a few seconds...	a few seconds...	

1 watch selected

1-10 of 10 < >

Current Status

Deactivate Delete

Action ↑	State
logging_1	✓ OK

Watch History

Last 1 hour ▾	State	Comment
May 23, 2019 @ 17:19:54.740	✓ OK	
May 23, 2019 @ 17:19:24.729	✓ OK	
May 23, 2019 @ 17:18:54.713	✓ OK	
May 23, 2019 @ 17:18:24.704	✓ OK	
May 23, 2019 @ 17:17:54.694	✓ OK	
May 23, 2019 @ 17:17:24.678	✓ OK	
May 23, 2019 @ 17:16:54.670	✓ OK	
May 23, 2019 @ 17:16:24.659	✓ OK	
May 23, 2019 @ 17:15:54.651	✓ OK	
May 23, 2019 @ 17:15:24.638	✓ OK	
May 23, 2019 @ 17:14:54.630	✓ OK	
May 23, 2019 @ 17:14:24.619	✓ OK	
May 23, 2019 @ 17:13:54.611	✓ OK	
May 23, 2019 @ 17:13:24.602	✓ OK	

1-20 of 66 < >

Summary

- In this lesson, we explored how to install and configure the X-Pack components in Elastic Stack and how to secure the Elastic cluster by creating users and roles.
- We also learned how to monitor the Elasticsearch server and alerting in order to generate notifications when there are changes or anomalies in the data.



COMPLETE LAB 8

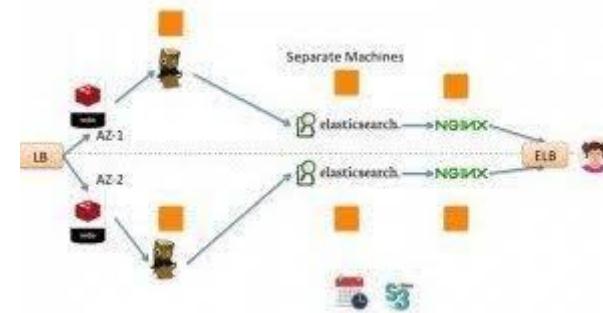
9. Running Elastic Stack in Production



Running Elastic Stack in Production

We will cover the following topics to help you take your next Elastic Stack project to production:

- Hosting Elastic Stack on a managed cloud
- Hosting Elastic Stack on your own, that is, self-hosting
- Backing up and restoring
- Setting up index aliases
- Setting up index templates
- Modeling time series data



Hosting Elastic Stack on a managed cloud

- Cloud providers make the process of setting up a production-ready cluster much easier.
- As a user, we don't have to do low-level configuration or the selection and management of hardware, an operating system, and many of the Elasticsearch and Kibana configuration parameters.

Getting up and running on Elastic Cloud

- Sign up for Elastic Cloud using <https://www.elastic.co/cloud/as-a-service/signup>, provide your email address, and verify your email.
- You will be asked to set your initial password.
- After your initial password is set, you can log in to the Elastic Cloud console at <https://cloud.elastic.co>

- Upon logging in, you can create a cluster from the following screen:

The screenshot shows the 'Create deployment' wizard interface. At the top right, there are user details: pranav.shukla@gmail.com, Trial started, and a help icon. The left sidebar includes links for Deployments, Custom plugins, Account, and Help.

Create deployment

- Name your deployment**
Give your deployment a name: test-cluster
- Select a cloud platform**
Pick your cloud and let us handle the rest. No additional accounts required.
Options: AWS (Amazon Web Services) and Google Cloud Platform.
- Select a region**
Options include US East (N. Virginia), US West (N. California), US West (Oregon), EU (Ireland), Asia Pacific (Singapore), Asia Pacific (Tokyo), South America (Sao Paulo), and Asia Pacific (Sydney). EU (Frankfurt) is also listed.
- Set up your deployment**
Stable versions: 7.1.0 (Recommended), 6.8.0, 5.6.16. Select a deployment to restore from its latest snapshot.
- Optimize your deployment**
Options include I/O Optimized (Recommended), Compute Optimized, Memory Optimized, and UNAVAILABLE Hot-Warm Architecture. Each option has a brief description and a 'Default specs' link.

Elastic Cloud supports many more options to cater to your specific use case such as hot-warm architecture optimized for logging, compute-focused setup optimized for analytics etc. [Learn more ...](#)

Deployment pricing
Free! As part of your 14-day trial, you can try it out without a credit card. If you want to unleash the full power of Elasticsearch Service now, you can enter your credit card details or contact sales@elastic.co.

Create deployment Customize deployment

Getting up and running on Elastic Cloud

The screenshot shows the Elastic Cloud interface. At the top right, there is a trial status message: "pranav.shukla@gmail.com Trial started". On the left, a sidebar menu includes sections for Deployments (with test-cluster selected), Custom plugins, Account, and Help. The main content area is titled "Activity" and shows a "Generated user" section. It displays credentials for Elasticsearch and Kibana, including a redacted password and a long redacted Cloud ID. Below this, there is a note about Beats and Logstash integration and an "APM Server secret token". At the bottom, there is an "Elasticsearch change history" section showing a recent change applied 4 minutes ago, with "Show details" and "Reapply" buttons.

pranav.shukla@gmail.com Trial started

Deployments

test-cluster

- Edit
- Elasticsearch
 - Logs
 - Snapshots
 - API Console
- Kibana
- APM
- Activity**
- Security
- Performance

Custom plugins

Account

Help

test-cluster

Activity

aws, US East (N. Virginia)

Elasticsearch Kibana APM

Generated user

You can use the credentials below to login to Elasticsearch or Kibana. Make sure to save the password somewhere as this is the only time we can show it to you.

Username	elastic
Password	[REDACTED]
Cloud ID	test-[REDACTED] [REDACTED]
APM Server secret token	[REDACTED]

Get started with Beats and Logstash quickly. The Cloud ID simplifies sending data to your cluster on Elastic Cloud. [Learn more ...](#)

Elasticsearch change history

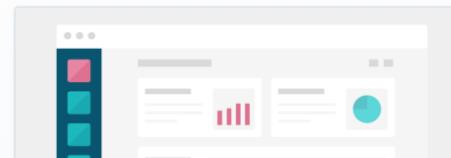
✓ #0 — Applied 4 minutes ago

Show details Reapply



Welcome to Kibana

Your window into the Elastic Stack



Let's get started

We noticed that you don't have any data in your cluster. You can try our sample data and dashboards or jump in with your own data.

[Try our sample data](#)

[Explore on my own](#)

Overriding configuration

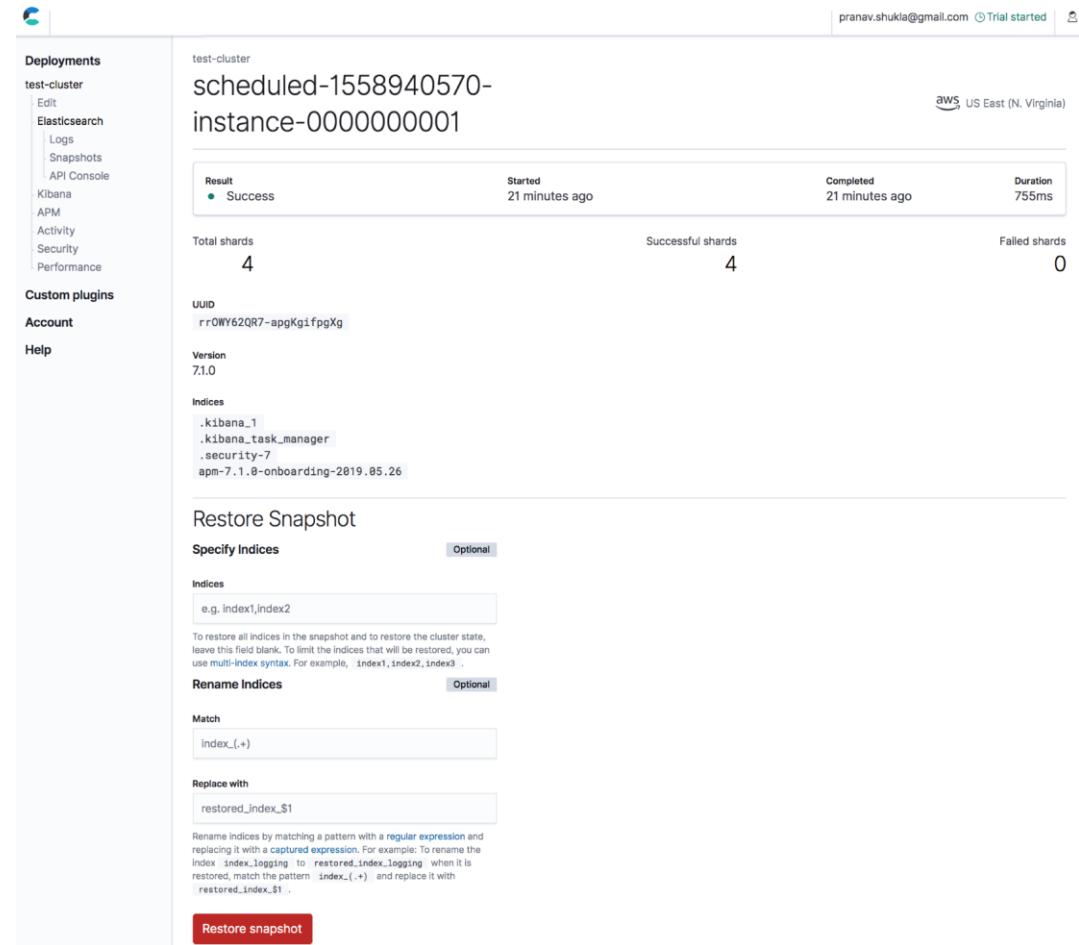
- It is possible to override the configuration of your Elasticsearch nodes via the Edit menu in the navigation panel on the left side under the Deployments.
- Elastic Cloud doesn't allow you to edit the `elasticsearch.yml` file directly.
- However, it provides a section called User Settings, which allows you to override a subset of the configuration parameters.

Recovering from a snapshot

The screenshot shows the Elasticsearch Snapshots interface. On the left, a sidebar menu includes Deployments, test-cluster (Edit, Elasticsearch, Logs, Snapshots), Kibana, APM, Activity, Security, Performance, Custom plugins, Account, and Help. The 'Solutions' section is collapsed. The main area is titled 'test-cluster / Elasticsearch Snapshots' and shows 'aws US East (N. Virginia)'. It displays the last successful snapshot ('19 minutes ago') and the next scheduled snapshot ('In 11 minutes'). The 'Snapshot frequency' is set to '30 minutes'. Below this, there are buttons for 'Take snapshot now' and 'Restore from another deployment'. A note says 'Choose a snapshot in another deployment to restore on this deployment'. The 'Solutions' section is expanded, showing a table of snapshots:

Snapshot	Status	Completed	Duration	Actions
scheduled-1558940570-instance-0000000001	Success	19 minutes ago	A few seconds	<button>Restore</button>
scheduled-1558938768-instance-0000000001	Success	An hour ago	A few seconds	<button>Restore</button>
scheduled-1558936966-instance-0000000001	Success	An hour ago	A few seconds	<button>Restore</button>
scheduled-1558935163-instance-0000000001	Success	2 hours ago	A few seconds	<button>Restore</button>
scheduled-1558933361-instance-0000000001	Success	2 hours ago	A few seconds	<button>Restore</button>
scheduled-1558931559-instance-0000000001	Success	3 hours ago	A few seconds	<button>Restore</button>
scheduled-1558929757-instance-0000000001	Success	3 hours ago	A few seconds	<button>Restore</button>
scheduled-1558927955-instance-0000000001	Success	4 hours ago	A few seconds	<button>Restore</button>
scheduled-1558926153-instance-0000000001	Success	4 hours ago	A few seconds	<button>Restore</button>
scheduled-1558924351-instance-0000000001	Success	5 hours ago	A few seconds	<button>Restore</button>

Pagination controls at the bottom show pages 1, 2, 3, >.



The screenshot shows the AWS Elasticsearch service console. On the left, there's a sidebar with navigation links: Deployments, test-cluster (selected), Edit, Elasticsearch (Logs, Snapshots, API Console), Kibana, APM, Activity, Security, Performance, Custom plugins, Account, and Help.

The main content area has a title "test-cluster scheduled-1558940570-instance-0000000001". It includes a summary table:

Result	Started	Completed	Duration
Success	21 minutes ago	21 minutes ago	755ms

Below the summary, it shows "Total shards: 4", "Successful shards: 4", and "Failed shards: 0".

Details section:

- UUID: rr0WY62QR7-apgKgfpqXg
- Version: 7.1.0
- Indices:
 - .kibana_1
 - .kibana_task_manager
 - .security-7
 - apm-7.1.0-onboarding-2019.05.26

Restore Snapshot section:

Specify Indices (Optional)

Indices: e.g. index1,index2

To restore all indices in the snapshot and to restore the cluster state, leave this field blank. To limit the indices that will be restored, you can use multi-index syntax. For example, `index1,index2,index3`.

Rename Indices (Optional)

Match: index_(.+)

Replace with: restored_index_\${1}

Rename indices by matching a pattern with a regular expression and replacing it with a captured expression. For example: To rename the index `index_logging` to `restored_index.logging` when it is restored, match the pattern `index_(.+)` and replace it with `restored_index_${1}`.

Restore snapshot

- You can choose the snapshot that you want to restore from, and you will be presented with the following screen:

Hosting Elastic Stack on your own

Hosting Elastic Stack on your own, that is, self-hosting Elastic Stack, requires you to install, configure, and manage Elasticsearch and your other Elastic Stack products. This can be done in one of two ways:

- Self-hosting on-premise
- Self-hosting on a cloud

Selecting hardware

- Elasticsearch primarily has memory-bound tasks which rely on the inverted index.
- The more data that it can fit in the RAM, the faster the performance will be.
- But this statement cannot always be generalized.

Selecting an operating system

- Linux is the preferred choice when deploying Elasticsearch and the Elastic Stack components.
- Your choice of operating system will mostly depend on the preferred technologies of your organization.
- Elastic Stack can also be deployed on Windows if your organization prefers the Microsoft stack.

Configuring Elasticsearch nodes

Elasticsearch, which is the heart of Elastic Stack, needs some configuration before starting it in production. Most of the configuration should work out of the box, but will require the following things to be reviewed at the OS level or JVM level.

- JVM heap size
- Disable swapping
- File descriptors
- Thread pools and garbage collector

Managing and monitoring Elasticsearch

- When you self-host Elasticsearch, the entire monitoring and management activities for the cluster are on you.
- It is necessary to monitor your Elasticsearch node process status, memory, and disk space on the node.
- If a node crashes for any reason, or becomes unavailable, it needs to be started back again.

Running in Docker containers

- Docker is a popular way of containerizing and shipping software.
- The advantage of Docker is that the software that is dockerized and runs inside a light-weight container that has a small overhead compared to a virtual machine.
- As a result of its reduced overhead and large pool of publicly available Docker images, Docker is a great way to run software in production in a predictable way without the need of much configuration.

Running in Docker containers

- The following simple commands will get your single-node Elasticsearch 7.0.1 up and running if you have Docker installed on your system:

```
docker pull
```

```
docker.elastic.co/elasticsearch/elasticsearch:7.0.1
```

```
docker run -p 9200:9200 -p 9300:9300 -e
```

```
"discovery.type=single-node"
```

```
docker.elastic.co/elasticsearch/elasticsearch:7.0.1
```

Special considerations while deploying to a cloud

Using a cloud provider as opposed to running on your own hardware comes with the following advantages:

- No upfront investment in hardware
- Ability to upgrade/downgrade servers
- Ability to add or remove servers as and when needed

Installing the S3 repository plugin

- It is important to back up your data in Elasticsearch regularly to restore the data if a catastrophic event occurs or if you want to revert to a last known healthy state.
- We will look at how to backup and restore using the snapshot/restore APIs of Elasticsearch in the next section.

Installing the S3 repository plugin

- The S3 repository plugin can be installed using the following command; it needs to be installed on every node of your Elasticsearch cluster:

```
sudo bin/elasticsearch-plugin install repository-s3
```

Setting up periodic snapshots

- Once you have a repository set up on S3, we need to ensure that actual snapshots are taken periodically. What this means is that we need a scheduled job that triggers the command to take a snapshot at regular intervals.
- The interval could be 15 minutes, 30 minutes, one hour, and so on, depending on the sensitivity of your data.

Backing up and restoring

- Taking regular backups of your data to recover in the event of catastrophic failures is absolutely critical.
- It is important that all of your data is saved periodically at fixed time intervals and a sufficient number of such backups are preserved

Setting up a repository for snapshots

The first step in setting up a regular backup process is setting up a repository for storing snapshots. There are different places where we could store snapshots:

- A shared filesystem
- Cloud or distributed filesystems (S3, Azure, GCS, or HDFS)

Shared filesystem

- When your cluster has a shared filesystem accessible from all the nodes of the cluster, you have to ensure that the shared filesystem is accessible on a common path.
- You should mount that shared folder on all nodes and add the path of the mounted directory.
- The shared, mounted filesystem's path should be added to each node's elasticsearch.yml as follows:

`path.repo: ["/mount/es_backups"]`

Shared filesystem

- The next step is to register a named repository under this registered folder.
- This is done using the following curl command, where we are registering a named repository with the name backups:

```
curl -XPUT 'http://localhost:9200/_snapshot/backups' -H 'Content-Type: application/json' -d '{  
  "type": "fs",  
  "settings": {  
    "location": "/mount/es_backups/backups",  
    "compress": true  
  }  
'
```

Cloud or distributed filesystems

- Elasticsearch has official plugins that allow you to store the snapshots in S3.
- All you need to do is install the repository—s3 plugin on all nodes of your cluster and set up the repository settings in a similar way to how we set up the shared filesystem repository:

https://github.com/fenago/elasticsearch/blob/master/snippets/9_1.txt

Taking snapshots

- Once the repository is set up, we can put named snapshots into a specific repository:

```
curl -XPUT
```

```
'http://localhost:9200/_snapshot/backups/backup_2019052715  
30?pretty' -H 'Content-Type: application/json' -d'
```

```
{
```

```
  "indices": "bigginsight,logstash-*",  
  "ignore_unavailable": true,  
  "include_global_state": false
```

```
}
```

```
,
```

Taking snapshots

- Snapshots are incremental by default.
- They don't store all the redundant data in all snapshots.
- Having taken the snapshots periodically, you would want to list all the snapshots that exist in a repository.
This can be done using the following command:

```
curl -XGET  
'http://localhost:9200/_snapshot/backups/_all?pretty'
```

Restoring a specific snapshot

- If the need arises, you can restore the state from a specific snapshot using the following command:

```
curl -XPOST  
'http://localhost:9200/_snapshot/backups/backup_201905  
271530/_restore'
```

Setting up index aliases

Index aliases let you create aliases for one or more indexes or index name patterns. We will cover the following topics in order to learn how index aliases work:

- Understanding index aliases
- How index aliases can help

Understanding index aliases

- An index alias just provides an extra name to refer to an index; it can be defined in the following way:

```
POST /_aliases
{
  "actions" : [
    { "add" : { "index" : "index1", "alias" : "current_index" } }
  ]
}
```

Understanding index aliases

- index1 can be referred to with the alias current_index.
- Similarly, the index alias can be removed with the remove action of the _aliases REST API:

```
POST /_aliases
{
  "actions" : [
    { "remove" : { "index" : "index1", "alias" : "current_index" }
  }
]
```

}

Understanding index aliases

- For example, the following call would be completely transparent to the client:

```
POST /_aliases
{
  "actions" : [
    { "remove" : { "index" : "index1", "alias" : "current_index"
    } },
    { "add" : { "index" : "index2", "alias" : "current_index" } }
  ]
}
```

How index aliases can help

- Once in production, it often happens that we need to reindex data from one index to another.
- We might have one or more applications developed in Java, Python, .NET, or other programming environments that may be referring to these indexes.
- In the event that the production index needs to be changed from index1 to index2, it will require a change in all client applications.

Setting up index templates

- One important step while setting up your index is defining the mapping for the types, number of shards, replica, and other configurations.
- Depending upon the complexity of the types within your index, this step can involve a substantial amount of configuration.

- We start by defining an index template:

```
PUT _template/readings_template          1
{
  "index_patterns": ["readings*"],        2
  "settings": {                          3
    "number_of_shards": 1
  },
  "mappings": {                         4
    "properties": {
      "sensorId": {
        "type": "keyword"
      },
      "timestamp": {
        "type": "date"
      },
      "reading": {
        "type": "double"
      }
    }
  }
}
```

Defining an index template

Creating indexes on the fly

- When any client tries to index the data for a particular sensor device, it should use the index name with the current day appended in yyyy-mm-dd format after readings.
- A call to index data for 2019-05-01 would look like the following:

```
POST /readings-2019-05-01/_doc
```

```
{
```

```
  "sensorId": "a11111",  
  "timestamp": 1483228800000,  
  "reading": 1.02
```

```
}
```

Modeling time series data

Often, we have a need to store time series data in Elasticsearch. Typically, one would create a single index to hold all documents. This typical approach of one big index to hold all documents has its own limitations, especially for the following reasons:

- Scaling the index with an unpredictable volume over time
- Changing the mapping over time
- Automatically deleting older documents

Scaling the index with unpredictable volume over time

- Let's understand how the number of shards becomes important in the following sub-sections:

Unit of parallelism in Elasticsearch:

1. The effect of the number of shards on the relevance score
2. The effect of the number of shards on the accuracy of aggregations

Changing the mapping over time

- Once an index is created and documents start getting stored, the requirements can change. There is only one thing that is constant, change.
- When the schema changes, the following types of change may happen with respect to the schema:
 - New fields get added
 - Existing fields get removed

Automatically deleting older documents

- No cluster has an infinite capacity to retain data forever.
- With the volume growing over a period of time, you may decide to only store necessary data in Elasticsearch.
- Typically, you may want to retain data for the past few weeks, months, or years in Elasticsearch, depending on your use case.

How index-per-timeframe solves these issues

- Instead of going with one big monolithic index, we now create one index per timeframe.
- The timeframe could be one day, one week, one month, or any arbitrary time duration.
- For example, in our example in the Index Template section, we chose index-per-day.
- The names of the index would reflect that—we had indexes such as readings-2019-05-01, readings-2019-05-02, and so on.



Summary

- In this lesson, we have seen essential techniques necessary to take your next Elastic Stack application to production.
- We have seen various deployment options, including cloud-based and on-premise.
- We have seen how to use a managed cloud service provider such as Elastic Cloud and have also covered how to self-host Elastic Stack.

COMPLETE LAB 9

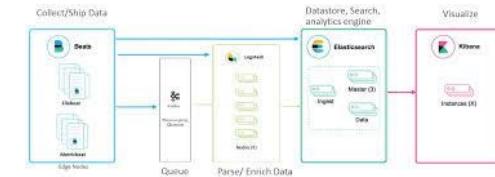
10. Building a Sensor Data Analytics Application



Building a Sensor Data Analytics Application

We will cover the following topics as we build a sensor-data analytics application:

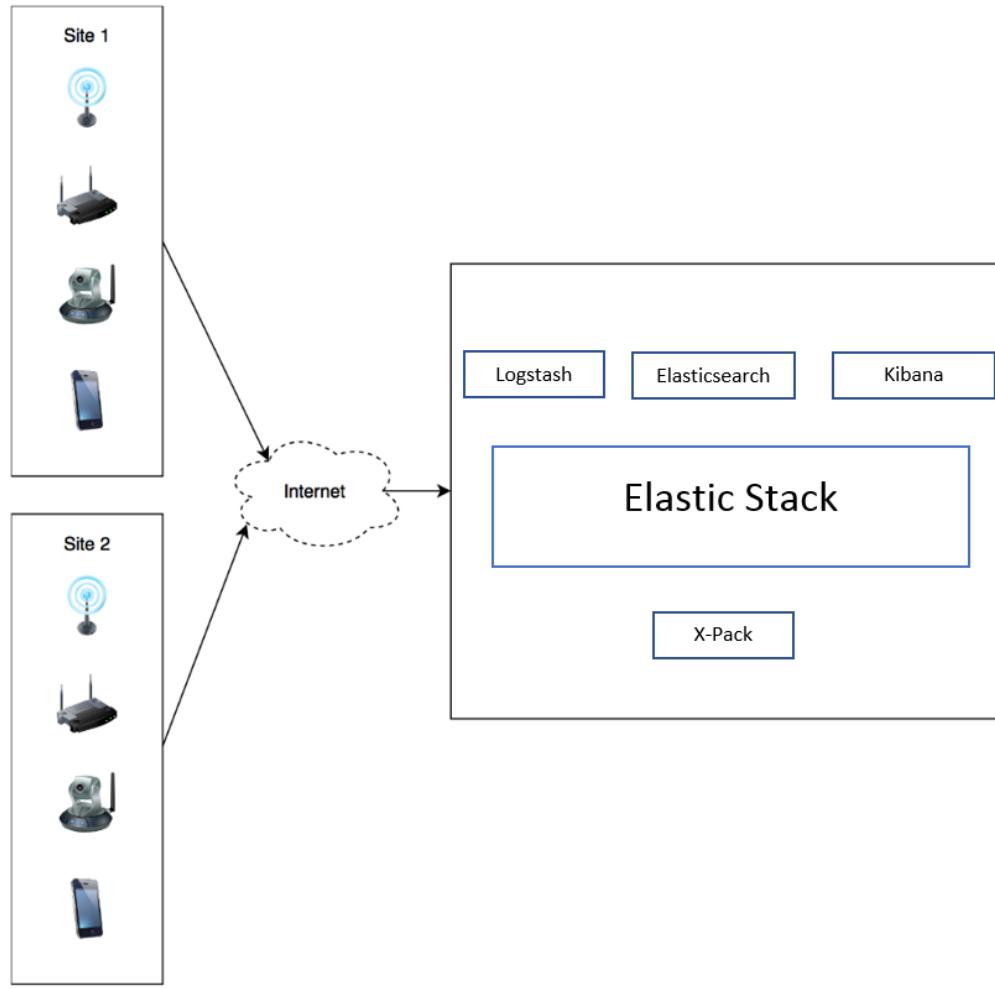
- Introduction to the application
- Modeling data in Elasticsearch
- Setting up the metadata database
- Building the Logstash data pipeline
- Sending data to Logstash over HTTP
- Visualizing the data in Kibana



Introduction to the application

The internet of things (IoT) has found a wide range of applications in modern times. IoT can be defined as follows:

- The Internet of things (IoT) is the collective web of connected smart devices that can sense and communicate with each other by exchanging data via the Internet.



Understanding the sensor-generated data

- What does the data look like when it is generated by the sensor? The sensor sends JSON-format data over the internet and each reading looks like the following:

```
{  
  "sensor_id": 1,  
  "time": 1511935948000,  
  "value": 21.89  
}
```

Understanding the sensor metadata

- The metadata about all the sensors across all locations is available to us in a relational database.
- In our example, we have stored it in MySQL.
- This type of metadata can be stored in any relational database other than MySQL.
- It can also be stored in Elasticsearch in an index.

Understanding the sensor metadata

- **sensor_type:** Defines various sensor types and their sensor_type_id:

sensor_type_id	sensor_type
1	Temperature
2	Humidity

Understanding the sensor metadata

- **location:** This defines locations with their latitude/longitude and address within a physical building:

location_id	customer	department	building_name	room	floor	location_on_floor	latitude	longitude
1	Abc Labs	R & D	222 Broadway	101	1	C-101	710936	-74.008500

Understanding the sensor metadata

- **sensors:** This maps sensor_id with sensor types and locations:

sensor_id	sensor_type_id	location_id
1	1	1
2	2	1

Understanding the sensor metadata

- Given this database design, it is possible to look up all of the metadata associated with the given `sensor_id` using the following SQL query:

https://github.com/fenago/elasticsearch/blob/master/snippets/10_1.txt

Understanding the sensor metadata

- The result of the previous query will look like this:

sensorType	customer	department	buildingName	room	floor	locationOnFloor	latitude	longitude
Temperature	Abc Labs	R & D	222 Broadway	101	Floor1	C-101	710936	-74.0085

Modeling data in Elasticsearch

We can apply some of the techniques mentioned in lesson 9, Running the Elastic Stack in Production, to model the data:

- Defining an index template
- Understanding the mapping
- Let's look at the index template that we will define.

Defining an index template

- Please create the index template mentioned in Step 1 of the README.md file or execute the following script in your Kibana Dev Tools Console:

https://github.com/fenago/elasticsearch/blob/master/snippets/10_2.txt

Understanding the mapping

The mapping that we defined in the index template contains all the fields that will be present in the denormalized record after lookup.

- A few things to notice in the index template mapping are as follows:
- All the fields that contain a text type of data are stored as the keyword type; additionally, they are stored as text in an analyzed field.
- For example, please have a look at the customer field.

Setting up the metadata databases

- We need to have a database that has metadata about the sensors.
- This database will hold the tables that we discussed in the Introduction to the application section.
- We are storing the data in a relational database MySQL, but you can use any other relational database equally well. Since we are using MySQL,

Setting up the metadata databases

- You can verify that the database was created and populated successfully by executing the following query:

https://github.com/fenago/elasticsearch/blob/master/snippets/10_3.txt

Setting up the metadata databases

- The result of the previous query will look like this:

sensorType	customer	department	buildingName	room	floor	locationOnFloor	latitude	longitude
Temperature	Abc Labs	R & D	222 Broadway	101	Floor1	C-101	710936	-74.0085

Building the Logstash data pipeline

Having set up the mechanism to automatically create the Elasticsearch index and the metadata database, we can now focus on building the data pipeline using Logstash. What should our data pipeline do? It should perform the following steps:

- Accept JSON requests over the web (over HTTP).
- Enrich the JSON with the metadata we have in the MySQL database.
- Store the resulting documents in Elasticsearch.

Accepting JSON requests over the web

- The relevant part from logstash_sensor_data_http.conf, which has the input filter, is as follows:

```
input {  
    http {  
        id => "sensor_data_http_input"  
    }  
}
```

Accepting JSON requests over the web

- We can configure user and password parameters to protect this endpoint with the desired username and password, as follows:

```
input {  
    http {  
        id => "sensor_data_http_input"  
        user => "sensor_data"  
        password => "sensor_data"  
    }  
}
```

Accepting JSON requests over the web

- When Logstash is started with this input plugin, it starts an HTTP server on port 8080, which is secured using basic authentication with the given username and password.
- We can send a request to this Logstash pipeline using a curl command, as follows:

```
curl -XPOST -u sensor_data:sensor_data --header "Content-Type: application/json" "http://localhost:8080/" -d '{"sensor_id":1,"time":1512102540000,"reading":16.24}'
```

Enriching the JSON with the metadata

- Logstash has a `jdbc_streaming` filter plugin that can be used to do lookups from any relational database and enrich the incoming JSON documents.
- Let's zoom into the filter plugin section in our Logstash configuration file:

https://github.com/fenago/elasticsearch/blob/master/snippets/10_4.txt

The jdbc_streaming plugin

- We essentially specify the whereabouts of the database that we want to connect to, the username/password, the JDBC driver .jar file, and the class.
- We already created the database in the Setting up the metadata database section.
- Download the latest MySQL JDBC Driver, also known as Connector/J, from
<https://dev.mysql.com/downloads/connector/j/>

The jdbc_streaming plugin

- The statement parameter has the same SQL query that we saw earlier.
- It looks up the metadata for the given sensor_id.
- A successful query will fetch all additional fields for that sensor_id.
- The result of the lookup query is stored in a new field, lookupResult, as specified by the target parameter.
- The resulting document, up to this point, should look like this:
https://github.com/fenago/elasticsearch/blob/master/snippets/10_5.txt

The mutate plugin

The output of the `jdbc_streaming` filter plugin has some undesired aspects. Our JSON payload needs the following modifications:

- Move the looked-up fields that are under `lookupResult` directly into the JSON file.
- Combine the `latitude` and `longitude` fields under `lookupResult` as a `location` field.
- Remove the unnecessary fields.

The mutate plugin

```
mutate {  
    rename => {"[lookupResult][0][sensorType]" => "sensorType"}  
    rename => {"[lookupResult][0][customer]" => "customer"}  
    rename => {"[lookupResult][0][department]" => "department"}  
    rename => {"[lookupResult][0][buildingName]" => "buildingName"}  
    rename => {"[lookupResult][0][room]" => "room"}  
    rename => {"[lookupResult][0][floor]" => "floor"}  
    rename => {"[lookupResult][0][locationOnFloor]" => "locationOnFloor"}  
    add_field => {  
        "location" => "%{lookupResult[0]latitude},%{lookupResult[0]longitude}"  
    }  
    remove_field => ["lookupResult", "headers", "host"]  
}
```

Moving the looked-up fields that are under lookupResult directly in JSON

- For example, the following operation renames the existing sensorType field directly under the JSON payload:

```
rename => {"[lookupResult][0][sensorType]" =>  
"sensorType"}
```

The mutate plugin

Combining the latitude and longitude fields under lookupResult as a location field

- The geo_point type accepts a value that is formatted as a string with latitude and longitude appended together, separated by a comma.
- This is achieved by using the add_field operation to construct the location field, as follows:

```
add_field => {
    "location" =>
    "%{[lookupResult][0][latitude]},%{[lookupResult][0][longitude]}"
}
```

Removing the unnecessary fields

- After moving all the elements from the `lookupResult` field directly in the JSON, we don't need that field anymore.
- Similarly, we don't want to store the headers or the host fields in the Elasticsearch index, so we remove them all at once using the following operation:

```
remove_field => ["lookupResult", "headers", "host"]
```

Store the resulting documents in Elasticsearch

- We use the Elasticsearch output plugin that comes with Logstash to send data to Elasticsearch.
- The usage is very simple; we just need to have `elasticsearch` under the `output` tag, as follows:

```
output {  
    elasticsearch {  
        hosts => ["localhost:9200"]  
        index => "sensor_data-%{+YYYY.MM.dd}"  
    }  
}
```

Store the resulting documents in Elasticsearch

- If you want to send events to a secured Elasticsearch cluster as we did when we used X-Pack in lesson 8, Elastic X-Pack, you can configure the user and password parameters as follows:

```
output {  
    elasticsearch {  
        hosts => ["localhost:9200"]  
        index => "sensor_data-%{+YYYY.MM.dd}"  
        user => "elastic"  
        password => "elastic"  
    }  
}
```

Sending data to Logstash over HTTP

- At this point, sensors can start sending their readings to the Logstash data pipeline that we have created in the previous section.
- They just need to send the data as follows:

```
curl -XPOST -u sensor_data:sensor_data --header  
"Content-Type: application/json" "http://localhost:8080/" -d  
'{"sensor_id":1,"time":1512102540000,"reading":16.24}'
```

Sending data to Logstash over HTTP

- Now, go to the lesson-10/data directory and execute load_sensor_data.sh:

```
$ pwd  
/Users/pranavshukla/workspace/learningelasticstack  
$ cd lesson-10/data  
$ ls  
load_sensor_data.sh sensor_data.json  
$ ./load_sensor_data.sh
```

Visualizing the data in Kibana

- Let's start by doing a sanity check to see if the data is loaded correctly.
- We can do so by going to Kibana Dev Tools and executing the following query:

```
GET /sensor_data-  
*/_search?size=0&track_total_hits=true  
{  
  "query": {"match_all": {}}  
}
```

Setting up an index pattern in Kibana

The screenshot shows the Kibana Home page at localhost:5601/app/kibana#/home?_g={}. The left sidebar contains various icons for different Elasticsearch modules like APM, Logging, Metrics, and Machine Learning. The main content area is divided into sections:

- Add Data to Kibana:** This section provides quick solutions to turn data into dashboards and monitoring systems. It includes four cards: APM, Logging, Metrics, and Security analytics, each with a description and a "Add [Module]" button.
- Visualize and Explore Data:** This section lists APM, Canvas, Dashboard, and Discover modules.
- Manage and Administer the Elastic Stack:** This section lists Console, Index Patterns, Monitoring, and Rollups modules. The "Index Patterns" card is highlighted with a red border.

Setting up an index pattern in Kibana

The screenshot shows the Kibana Management interface with the 'Create index pattern' page open. The left sidebar includes links for Elasticsearch (Index Management, Index Lifecycle Policies, Rollup Jobs, Cross Cluster Replication, Remote Clusters, License Management, 8.0 Upgrade Assistant) and Kibana (Index Patterns, Saved Objects, Spaces, Reporting, Advanced Settings). The main header says 'Management / Create index pattern'. A blue button labeled 'Create index pattern' is visible. A message states: 'No default index pattern. You must select or create one to continue.' Below this is a section titled 'Create index pattern' with the sub-instruction: 'Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.' A toggle switch labeled 'Include system indices' is shown. The central area is titled 'Step 1 of 2: Define index pattern' and contains an input field with the value 'sensor_data*'. A note explains: 'You can use a * as a wildcard in your index pattern. You can't use spaces or the characters \, /, ?, ", <, >, |.' A green success message says: '✓ Success! Your index pattern matches 1 index.' Below it, the matching index is listed as 'sensor_data-2019.05.26'. A dropdown menu shows 'Rows per page: 10'. A blue 'Next step' button is at the bottom right.

Setting up an index pattern in Kibana

The screenshot shows the Kibana Management interface with the 'Create index pattern' page open. The left sidebar has 'Management' selected. Under 'Elasticsearch', 'Index Management' is highlighted. The main area shows a 'Create index pattern' button and a note: 'No default index pattern. You must select or create one to continue.' A 'Create index pattern' section with a 'Step 2 of 2: Configure settings' header is displayed. It shows a dropdown for 'Time Filter field name' set to 'time'. A note explains: 'The Time Filter will use this field to filter your data by time. You can choose not to have a time field, but you will not be able to narrow down your data by a time range.' Advanced options are shown below with a 'Show advanced options' link. Navigation buttons 'Back' and 'Create index pattern' are at the bottom right.

Management / Create index pattern

Elasticsearch

- Index Management
- Index Lifecycle Policies
- Rollup Jobs
- Cross Cluster Replication
- Remote Clusters
- License Management
- 8.0 Upgrade Assistant

Kibana

Index Patterns

- Saved Objects
- Spaces
- Reporting
- Advanced Settings

Create index pattern

No default index pattern. You must select or create one to continue.

Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

Include system indices

Step 2 of 2: Configure settings

You've defined **sensor_data*** as your index pattern. Now you can specify some settings before we create it.

Time Filter field name Refresh

time

The Time Filter will use this field to filter your data by time.
You can choose not to have a time field, but you will not be able to narrow down your data by a time range.

› Show advanced options

Back Create index pattern

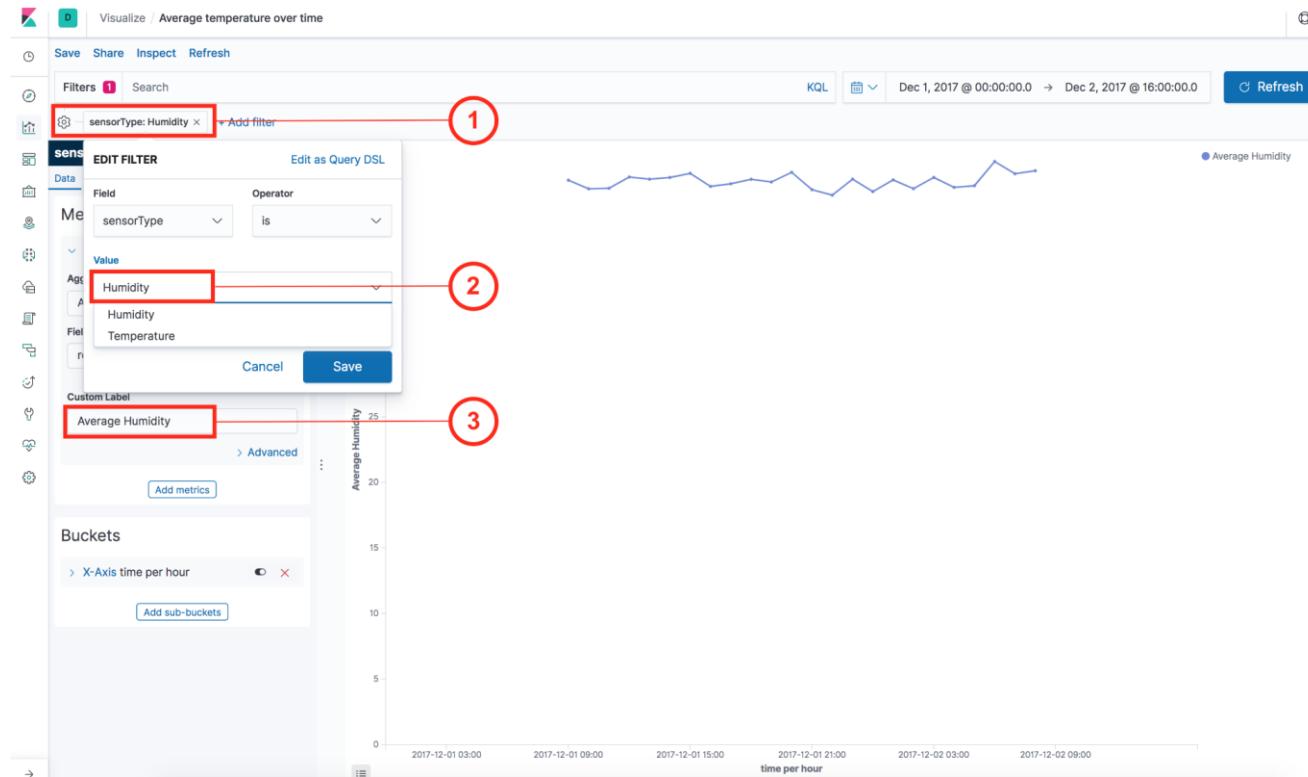
Building visualizations

- Before we embark on an analytics project, we often already have some questions that we want to get answered quickly from visualizations.
- These visualizations, which answer different questions, may be packaged as a dashboard or may be used as, and when, needed.

How does the average temperature change over time?



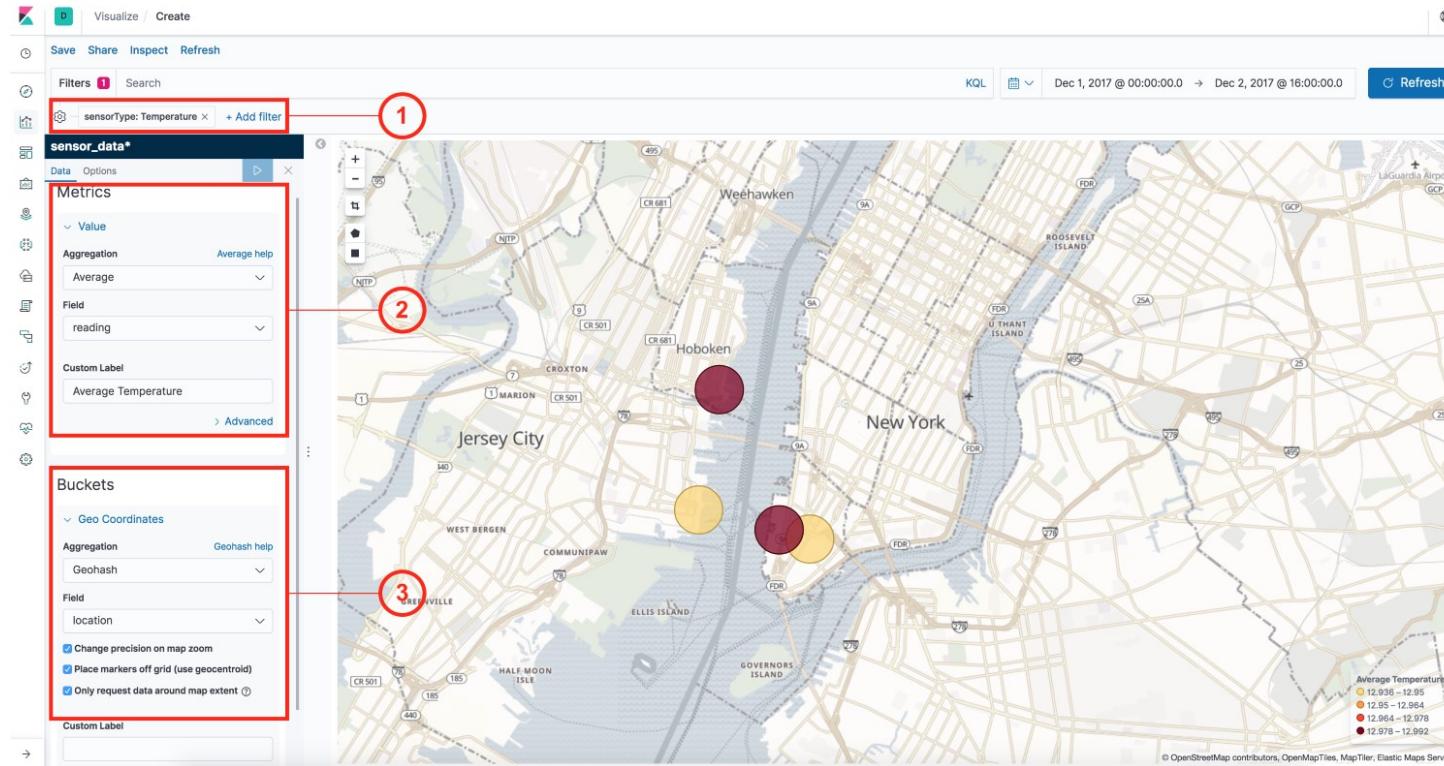
How does the average humidity change over time?



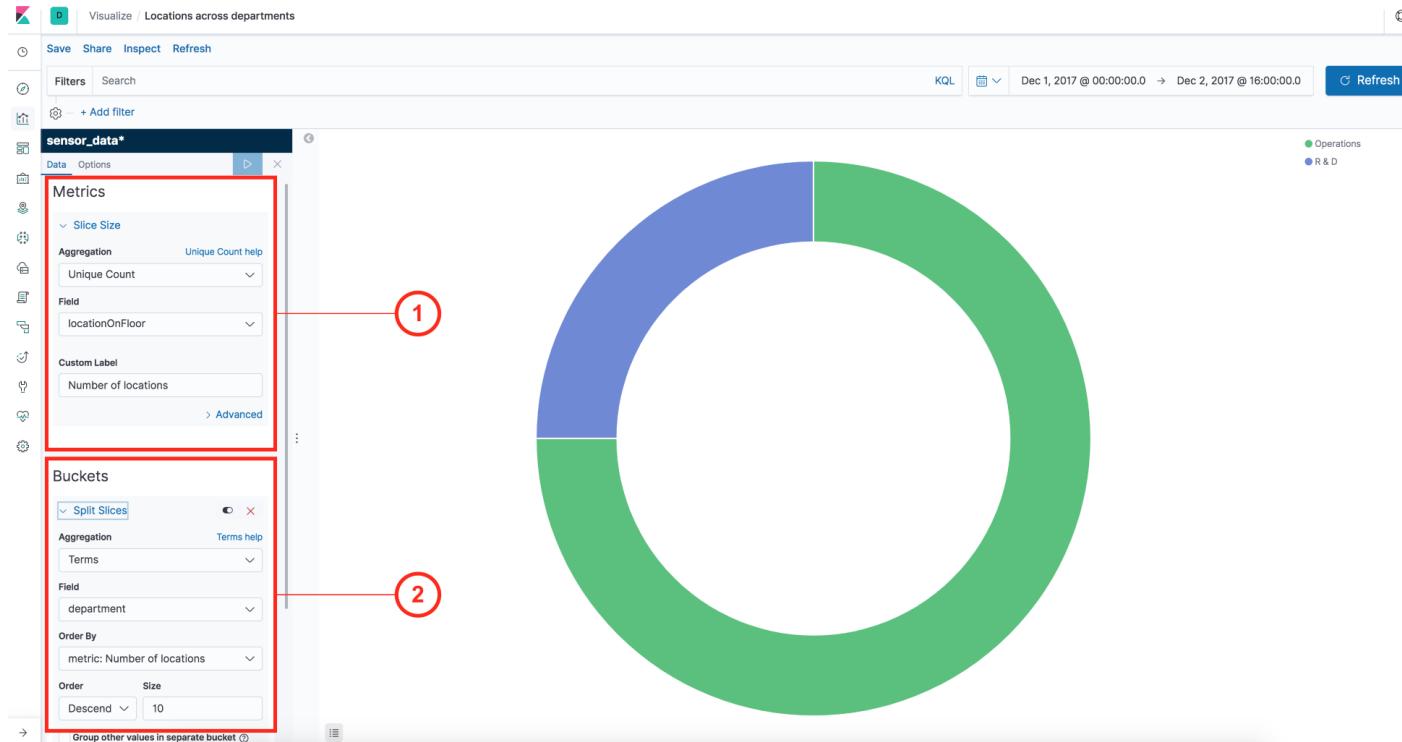
How do temperature and humidity change at each location over time?



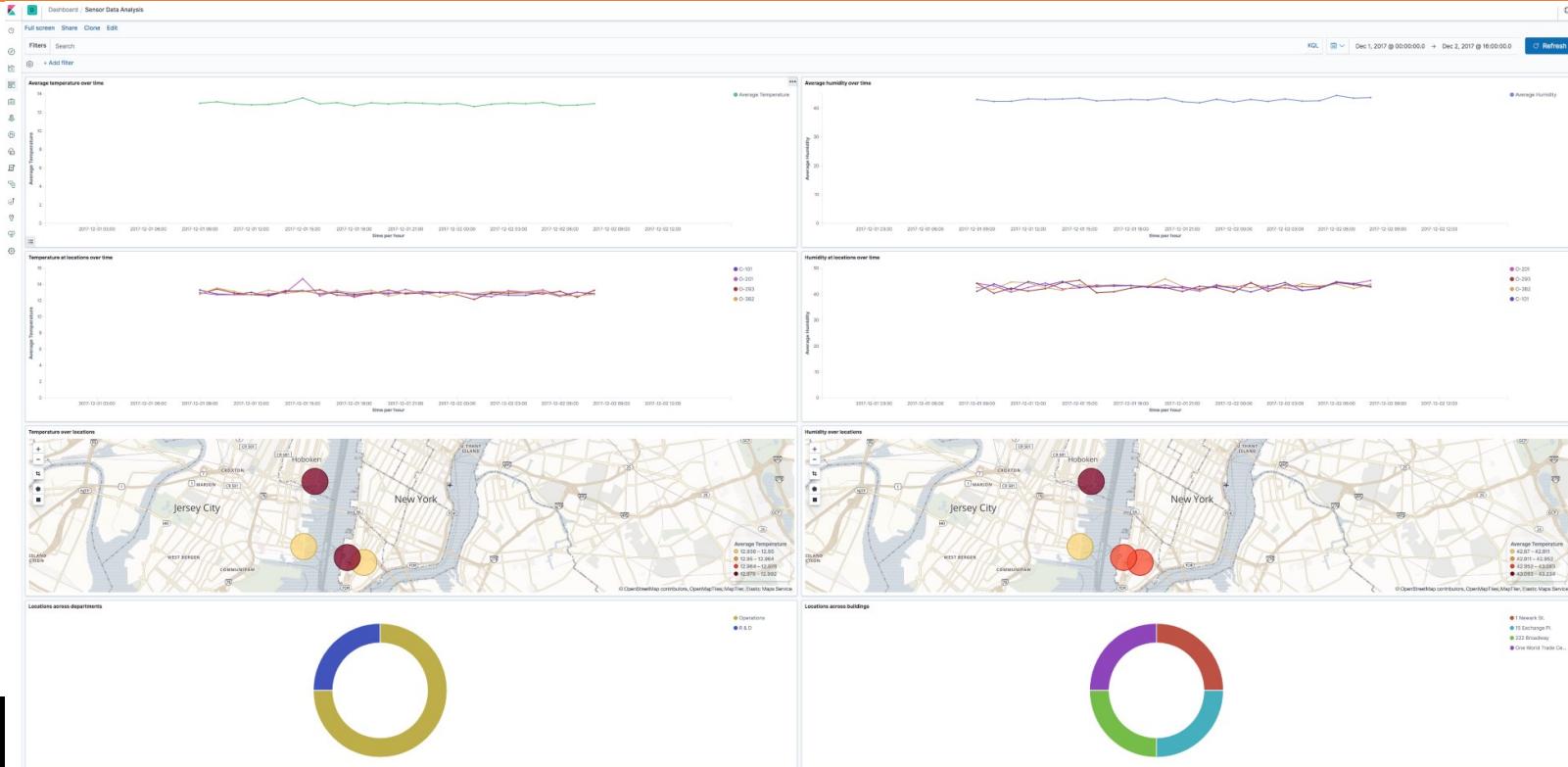
Can I visualize temperature and humidity over a map?



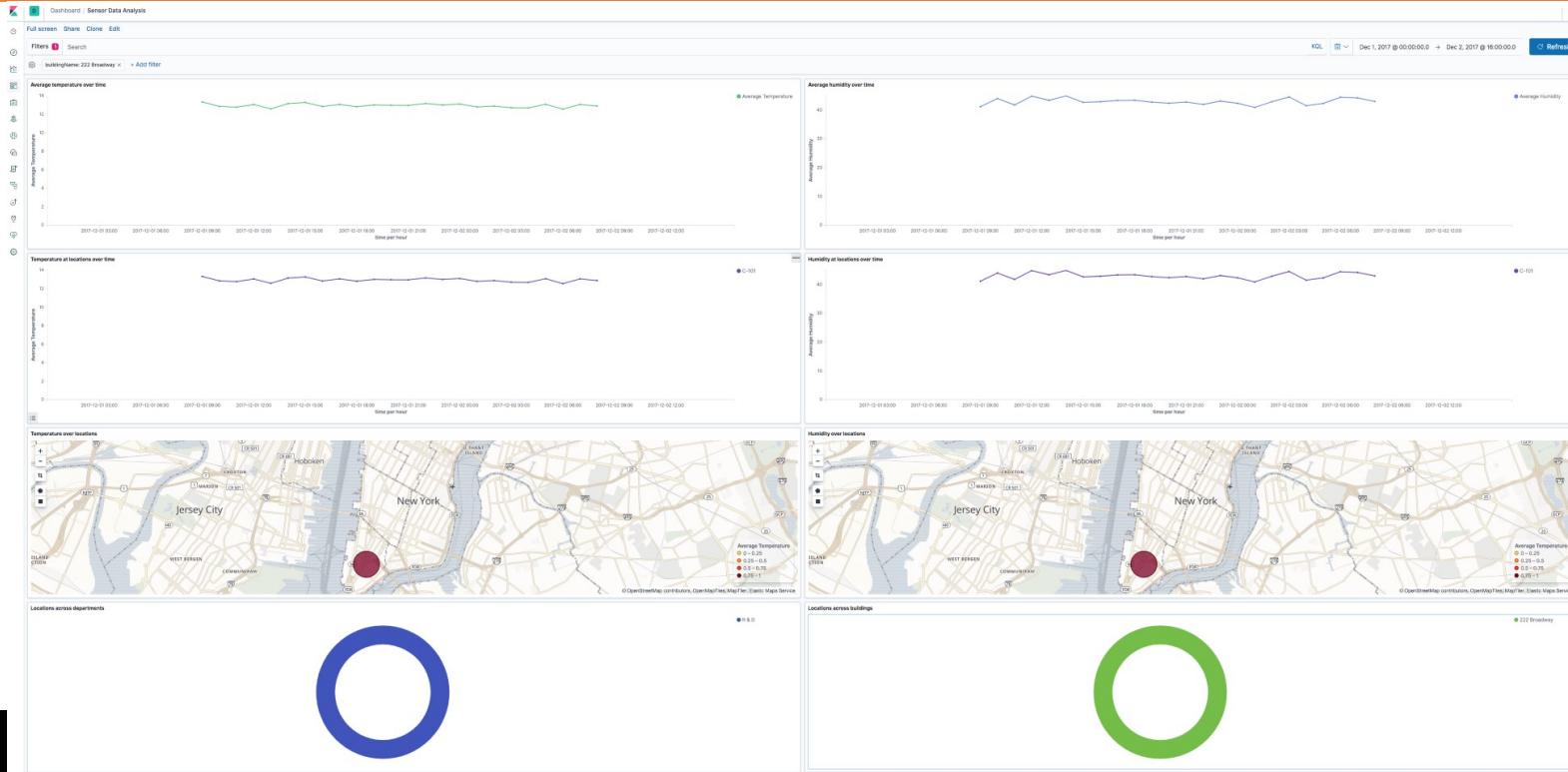
How are the sensors distributed across departments?



Creating a dashboard



Creating a dashboard



Creating a dashboard





Summary

- In this lesson, we built a sensor data analytics application that has a wide variety of applications, as it is related to the emerging field of IoT.
- We understood the problem domain and the data model, including metadata related to sensors.
- We wanted to build an analytics application using only the components of the Elastic Stack, without using any other tools and programming languages, to obtain a powerful tool that can handle large volumes of data.

COMPLETE LAB 10

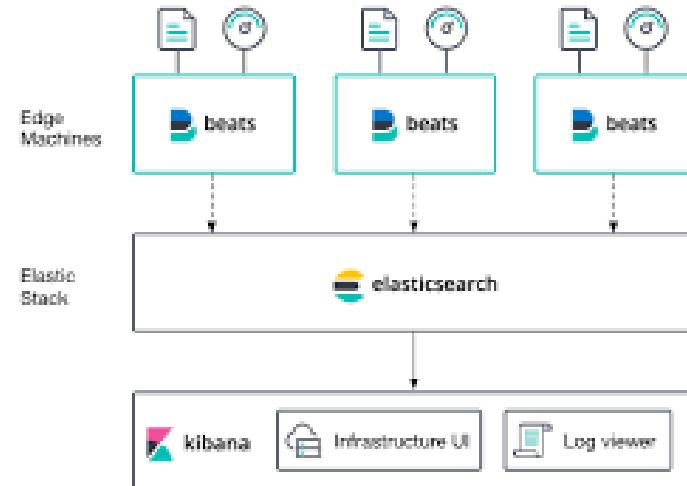
11. Monitoring Server Infrastructure



Monitoring Server Infrastructure

In this lesson, we will cover the following topics:

- Metricbeat
- Configuring Metricbeat
- Capturing system metrics
- Deployment architecture



Metricbeat

- Metricbeat is a lightweight shipper that periodically collects metrics from the operating system and from services running on the server.
- It helps you monitor servers by collecting metrics from the system and services such as Apache, MongoDB, Redis, and so on, that are running on the server.

Downloading and installing Metricbeat

← → ⌂ <https://www.elastic.co/downloads/beats/metricbeat-oss>

 elastic Products Cloud Services Customers Learn

downloads contact EN

Downloads

Download Metricbeat - OSS Only

Want to upgrade? We'll give you a hand. [Migration Guide »](#)

Version: 7.0.0

Release date: April 10, 2019

License: [Apache 2.0](#)

Downloads:

↳ DEB 32-BIT sha	↳ DEB 64-BIT sha
↳ RPM 32-BIT sha	↳ RPM 64-BIT sha
↳ LINUX 32-BIT sha	↳ LINUX 64-BIT sha
↳ MAC sha	↳ WINDOWS 32-BIT sha
↳ WINDOWS 64-BIT sha	

Containers:

Run with Docker	Run with Kubernetes
---------------------------------	-------------------------------------

Notes: This distribution only includes features licensed under the Apache 2.0 license. To get access to full [set of free features](#), use

Installing on Windows

- Unzip the downloaded file and navigate to the extracted location, as follows:

```
E:>cd E:\metricbeat-7.0.0-windows-x86_64
```

Installing on Windows

- Run the following commands to install Metricbeat as a Windows service from the PowerShell prompt as follows:

```
PS >cd E:\metricbeat-7.0.0-windows-x86_64
```

```
PS >E:\metricbeat-7.0.0-windows-x86_64>.\install-service-metricbeat.ps1
```

Installing on Linux

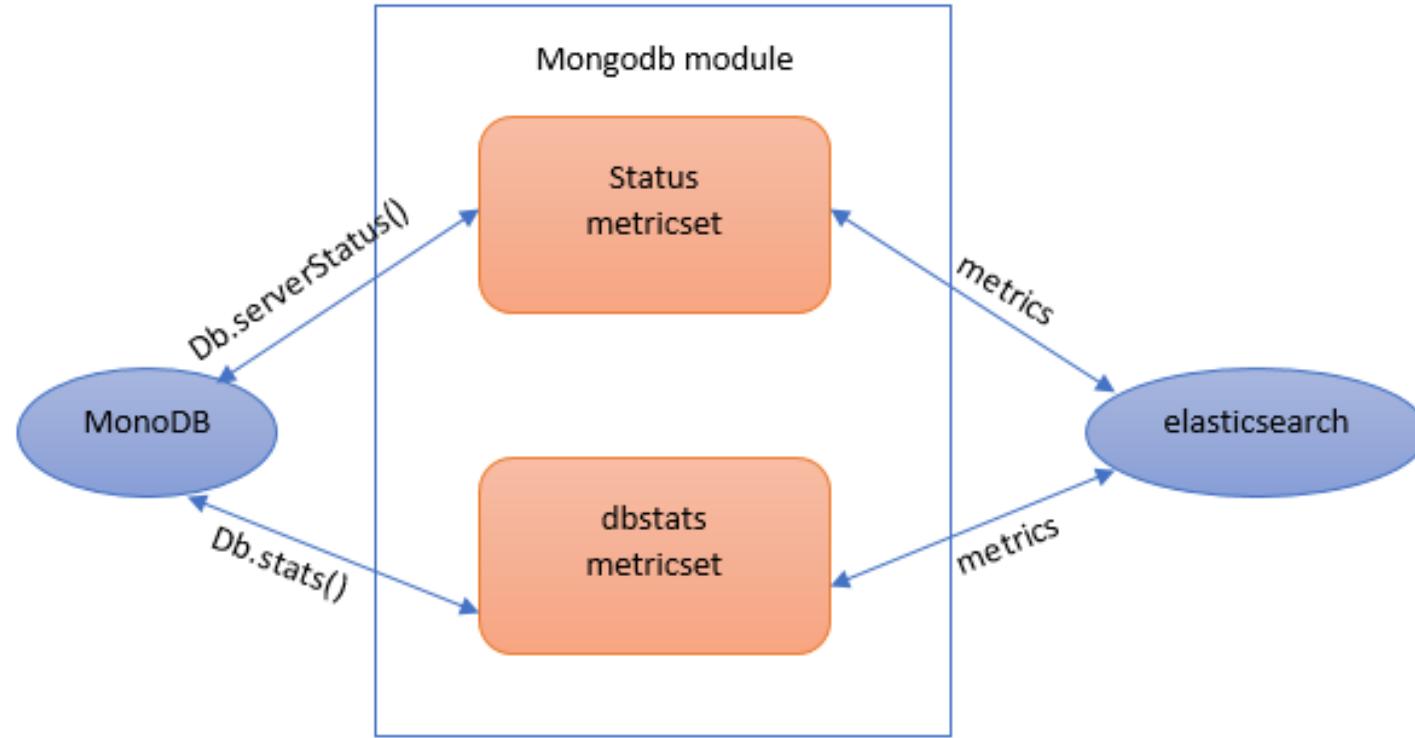
- Unzip the tar.gz package and navigate to the newly created folder, as shown in the following code snippet:

```
$>tar -xzf metricbeat-7.0.0-linux-x86_64.tar.gz  
$>cd metricbeat
```

Architecture

- Metricbeat is made up of two components: one is called modules and the other is called metricsets.
- A Metricbeat module defines the basic logic of collecting data from a specific service, such as MongoDB, Apache, and so on.
- The module specifies details about the service, including how to connect, how often to collect metrics, and which metrics to collect.

Architecture



Event structure

Metricbeat sends two types of event:

- Regular events containing the fetched metrics
- Error events when the service is down/unreachable

Event structure

- In the case of error events, an error field such as `error.message`, containing the error message, code, and type, will be appended to the event.
- An example of a regular event is as follows:

https://github.com/fenago/elasticsearch/blob/master/snippets/11_1.txt

Event structure

- An example of an error event when mongodb is not reachable is as follows:

https://github.com/fenago/elasticsearch/blob/master/snippets/11_2.txt

Configuring Metricbeat

The configurations related to Metricbeat are stored in a configuration file named metricbeat.yml, which uses YAML syntax.

The metricbeat.yml file contains the following:

- Module configuration
- General settings
- Output configuration etc.

Module configuration

- Metricbeat comes bundled with various modules to collect metrics from the system and applications, such as Apache, MongoDB, Redis, MySQL, and so on.
- Metricbeat provides two ways of enabling modules and metricsets as follows:

Enabling module configs in the modules.d directory
Enabling module configs in the metricbeat.yml file

Enabling module configs in the modules.d directory

- By default, except for the system module, all other modules are disabled.
- To list the modules that are available in Metricbeat, execute the following command:

Windows:

```
E:\metricbeat-7.0.0-windows-x86_64>metricbeat.exe  
modules list
```

Linux:

```
[locationOfMetricBeat]$./metricbeat modules list
```

Enabling module configs in the modules.d directory

- The basic configuration for the mongodb module will look as follows:
 - module: mongodb
 - metricsets: ["dbstats", "status"]
 - period: 10s
 - hosts: ["localhost:27017"]
 - username: user
 - password: pass

Enabling module configs in the modules.d directory

- To enable it, execute the modules enable command, passing one or more module names. For example:

Windows:

```
E:\metricbeat-7.0.0-windows-x86_64>metricbeat.exe  
modules enable redis mongodb
```

Linux:

```
[locationOfMetricBeat]$./metricbeat modules enable redis  
mongodb
```

Enabling module configs in the modules.d directory

- Similar to disabling modules, execute the modules disable command, passing one or more module names to it. For example:

Windows:

```
E:\metricbeat-7.0.0-windows-x86_64>metricbeat.exe modules  
disable redis mongodb
```

Linux:

```
[locationOfMetricBeat]$./metricbeat modules disable redis  
mongodb
```

Enabling module configs in the metricbeat.yml file

- If you're used to using earlier versions of Metricbeat, you can enable the appropriate modules and metricsets in the metricbeat.yml file directly by adding entries to the metricbeat.modules list.
- Each entry in the list begins with a dash (-) and is followed by the settings for that module. For example:
https://github.com/fenago/elasticsearch/blob/master/snippets/11_3.txt

General settings

- name: The name of the shipper that publishes the network data.
- By default, the hostname is used for this field, as follows:

name: "dc1-host1"

General settings

- **tags:** A list of tags that will be included in the tags field of every event Metricbeat ships.
- Tags make it easy to group servers by different logical properties and are useful when filtering events in Kibana and Logstash, as follows:

```
tags: ["staging", "web-tier", "dc1"]
```

- **max_procs:** The maximum number of CPUs that can be executing simultaneously & The default is the number of logical CPUs available in the system:

```
max_procs: 2
```

Output configuration

- **elasticsearch:** This is used to send events directly to Elasticsearch.
- A sample Elasticsearch output configuration is shown in the following code snippet:

```
output.elasticsearch:  
  enabled: true  
  hosts: ["localhost:9200"]
```

Output configuration

- If Elasticsearch is secure, then credentials can be passed using the username and password settings, as follows:

```
output.elasticsearch:  
  enabled: true  
  hosts: ["localhost:9200"]  
  username: "elasticuser"  
  password: "password"
```

Output configuration

- To ship events to the Elasticsearch ingest node pipeline so that they can be preprocessed before being stored in Elasticsearch, pipeline information can be provided using the pipeline setting, as follows:

```
output.elasticsearch:  
  enabled: true  
  hosts: ["localhost:9200"]  
  pipeline: "nginx_log_pipeline"
```

Output configuration

- You can override the index name or the pattern using the index setting. In the following configuration snippet, a new index is created for every month, as follows:

```
output.elasticsearch:  
hosts: ["http://localhost:9200"]  
index: "metricbeat-%{[beat.version]}-%{+yyyy.MM}"
```

Output configuration

- If the message contains neither of these strings, then those events will be pushed to the logs-%{+yyyy.MM.dd} index, as specified in the index parameter, as follows:

```
output.elasticsearch:  
  hosts: ["http://localhost:9200"]  
  index: "logs-%{+yyyy.MM.dd}"  
  indices:  
    -index: "debug-%{+yyyy.MM.dd}"  
      when.contains:  
        message: "DEBUG"  
      -index: "error-%{+yyyy.MM.dd}"  
        when.contains:  
          message: "ERR"
```

Output configuration

- A sample Logstash output configuration is as follows:

```
output.logstash:  
  enabled: true  
  hosts: ["localhost:5044"]
```

Output configuration

- To enable load-balancing events across the Logstash hosts, use the loadbalance flag, set to true, as follows:

```
output.logstash:  
hosts: ["localhost:5045", "localhost:5046"]  
loadbalance: true
```

Output configuration

- **console:** This is used to send events to stdout. These events are written in JSON format. This is useful during debugging or testing.
- A sample console configuration is as follows:

```
output.console:  
  enabled: true  
  pretty: true
```

Logging

- A sample configuration is as follows:

```
logging.level: debug
logging.to_files: true
logging.files:
  path: C:\logs\metricbeat
  name: metricbeat.log
  keepfiles: 10
```

Capturing system metrics

In order to monitor and capture metrics related to servers, Metricbeat provides the system module. The system module provides the following metricsets to capture server metrics, as follows:

- core: This metricset provides usage statistics for each CPU core.
- cpu: This metricset provides CPU statistics.



Running Metricbeat with the system module

- Replace the content of metricbeat.yml with the following configuration and save the file:

https://github.com/fenago/elasticsearch/blob/master/snippets/11_4.txt

Running Metricbeat with the system module

- By default, the system module is enabled. Make sure that it is enabled by executing the following command:

Windows:

```
E:\metricbeat-7.0.0-windows-x86_64>metricbeat.exe modules enable  
system
```

```
Module system is already enabled
```

Linux:

```
[locationOfMetricBeat]$./metricbeat modules enable system  
Module system is already enabled
```

Running Metricbeat with the system module

- You can verify the metricsets that are enabled for the system module by opening the system.yml file, which can be found under the modules.d directory, as follows:

https://github.com/fenago/elasticsearch/blob/master/snippets/11_5.txt

Running Metricbeat with the system module

- Start Metricbeat by executing the following command:

Windows:

```
E:\metricbeat-7.0.0-windows-x86_64>metricbeat.exe -e
```

Linux:

```
[locationOfMetricBeat]$./metricbeat -e
```

Running Metricbeat with the system module

- Once Metricbeat is started, it loads sample Kibana dashboards and starts shipping metrics to Elasticsearch.
- To validate this, execute the following command:

https://github.com/fenago/elasticsearch/blob/master/snippets/11_6.txt

Specifying aliases

- Elasticsearch allows the user to create an alias—a virtual index name that can be used to refer to an index or multiple indices.
- The Elasticsearch index API aliases an index with a name.
- This enables all the APIs to automatically convert their alias names into the actual index name.

- For example, as shown in the following code snippet, an alias called april_04_metrics is created for all the indexes of the metricbeat-7.0.0-2019.04.* pattern, that is, those Metricbeats indexes that are created on a daily basis in the month of April 2019:

```
curl -X POST http://localhost:9200/_aliases -H 'content-type: application/json' -d '  
{  
  "actions":  
  [  
    {"add":{ "index" : "metricbeat-7.0.0-2019.04.*", "alias": "april_04_metrics"} }  
  ]  
}'
```

Specifying aliases

- Now, using the april_04_metrics alias name, the query can be executed against all the indexes of the metricbeat-7.0.0-2019.04.* pattern as follows:

```
curl -X GET  
http://localhost:9200/april_04_metrics/_search
```

Specifying aliases

- In the future, if a new sales index gets created, then that index can also point to the sales index so that the end user/application can always make use of the sales endpoint to query all sales data, as follows:

```
curl -X POST  http://localhost:9200/_aliases -d '{  
  "actions": [  
    {"add": {"index": "it_sales", "alias": "sales"}},  
    {"add": {"index": "retail_sales", "alias": "sales"}}  
  ]}
```

Specifying aliases

- To remove an alias from an index, use the remove action of the aliases API, as follows:

```
curl -X POST http://localhost:9200/_aliases -d '  
{"actions": [{"remove": {"index": "retail_sales", "alias": "sales"} } ]}'
```

Visualizing system metrics using Kibana

The screenshot shows the Kibana interface with a search bar containing the text "syste". Three dashboard results are listed:

Title	Description	Actions
[Metricbeat System] Overview ECS	Overview of system metrics	Edit
[Metricbeat System] Containers overview ECS	Overview of container metrics	Edit
[Metricbeat System] Host overview ECS	Overview of host metrics	Edit

Rows per page: 20

Visualizing system metrics using Kibana

Dashboard / [Metricbeat System] Overview ECS

Full screen Share Clone Edit

Filters *| Lucene | Last 15 minutes Show dates Update

System Navigation [Metricbeat System] ECS

[System Overview](#) | [Host Overview](#) | [Containers overview](#)

Number of hosts [Metricbeat System... CPU Usage Gauge [Metricbeat Syst... Memory Usage Gauge [Metricbeat Sy... Disk used [Metricbeat System] ECS Inbound Traffic [Metricbeat System] ... Outbound Traffic [Metricbeat Syst...

1 CPU Usage 46.725% Memory Usage 66.9% Disk used 0%

Inbound Traffic **8.467KB/s** Total Transferred 2.679MB

Outbound Traffic **76.4KB/s** Total Transferred 11.027MB

Top Hosts By CPU (Realtime) [Metricbeat System] ECS

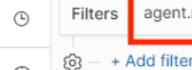
metricbeat_inst1 114.2%

Top Hosts By Memory (Realtime) [Metricbeat System] ECS

metricbeat_inst1 66.9%

LEARNING VOYAGE

1-722



Filters agent.name:metricbeat_inst1

Lucene



Last 15 minutes

Show dates

Refresh

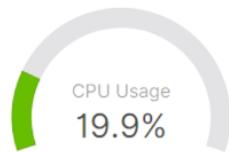
System Navigation [Metricbeat System] ECS

[System Overview](#) | [Host Overview](#) | [Containers overview](#)

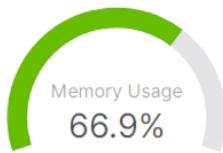
Tip [Metricbeat System] ECS

TIP: To select another host, go to the [System Overview](#) dashboard and double-click a host name.

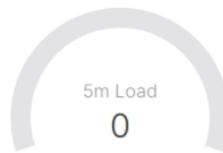
CPU Usage Gauge [Metricbeat System] ECS

CPU Usage
19.9%

Memory Usage Gauge [Metricbeat System] ECS

Memory Usage
66.9%

Load Gauge [Metricbeat System] ECS

5m Load
0

Inbound Traffic [Metricbeat System] ECS

Inbound Traffic
6.22KB/s
Total Transferred 3.577MB

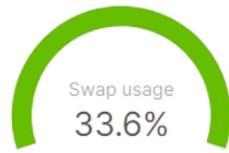
Outbound Traffic [Metricbeat System] ECS

Outbound Traffic
12.04KB/s
Total Transferred 14.912MB

Packetloss [Metricbeat System] ECS

In Packetloss
0
Out Packetloss 0

Swap usage [Metricbeat System] ECS

Swap usage
33.6%

Memory usage vs total ECS

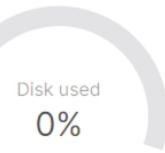
Memory usage
5.349GB
Total Memory 8GB

Number of processes [Metricbeat System] ECS

Number of processes [Metricbeat System] ECS

21
Processes

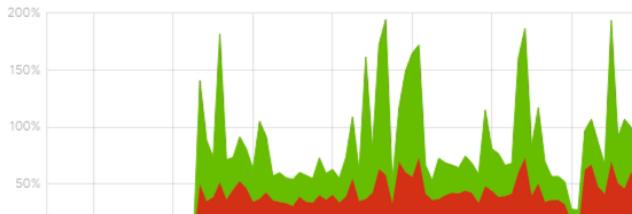
Disk used [Metricbeat System] ECS

Disk used
0%

Disk Usage [Metricbeat System] ECS

E:\ 0%
C:\ 0%

CPU Usage [Metricbeat System] ECS



user	37.6%
system	42%
nice	0%
irq	0%
softirq	0%
iowait	0%

System Load [Metricbeat System] ECS



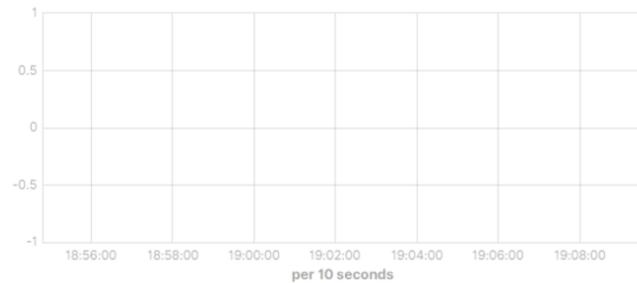
1m	0
5m	0
15m	0



CPU Usage [Metricbeat System] ECS

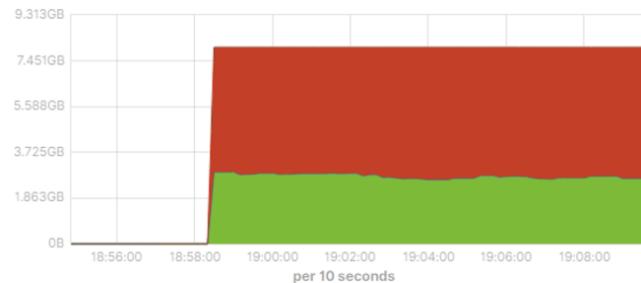


System Load [Metricbeat System] ECS



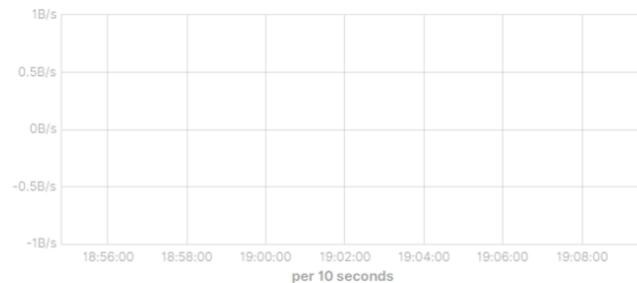
1m	0
5m	0
15m	0

Memory Usage [Metricbeat System] ECS



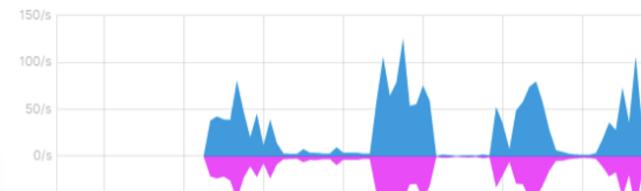
Used	5.349GB
Cache	0B
Free	2.651GB

Disk IO (Bytes) [Metricbeat System] ECS

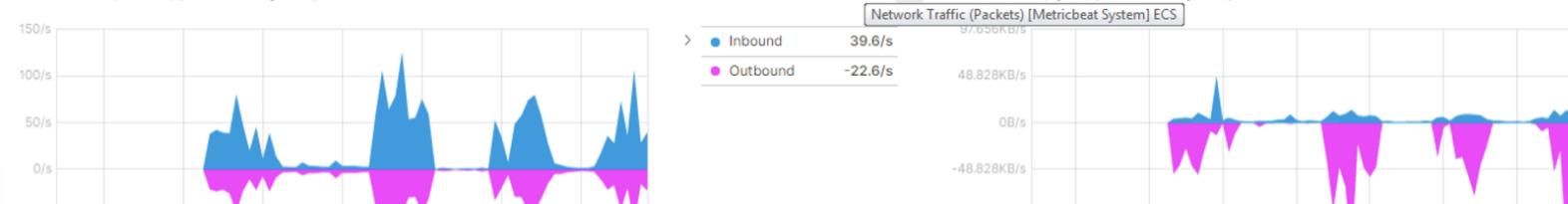


reads	0B/s
writes	0B/s

Network Traffic (Packets) [Metricbeat System] ECS



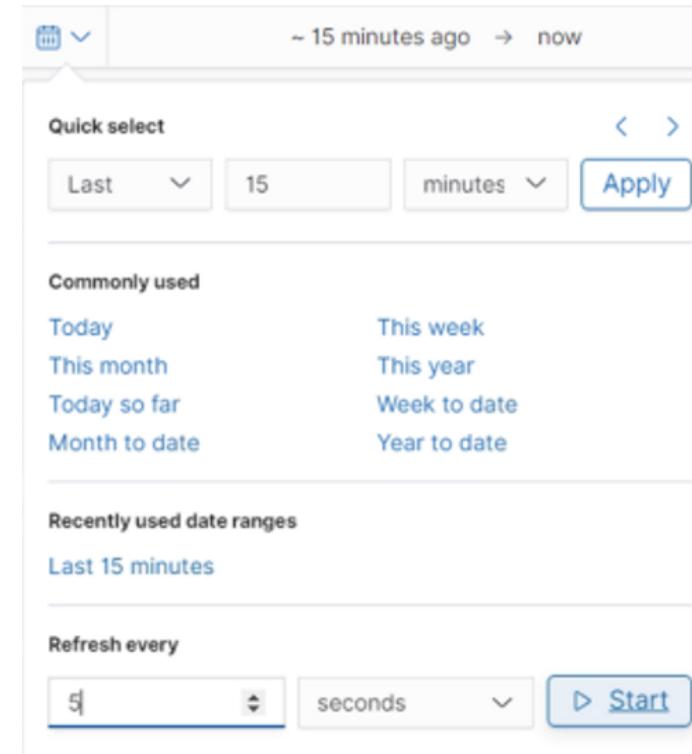
Network Traffic (Bytes) [Metricbeat System] ECS



Inbound	6.22KB/s
Outbound	-12.04KB/s

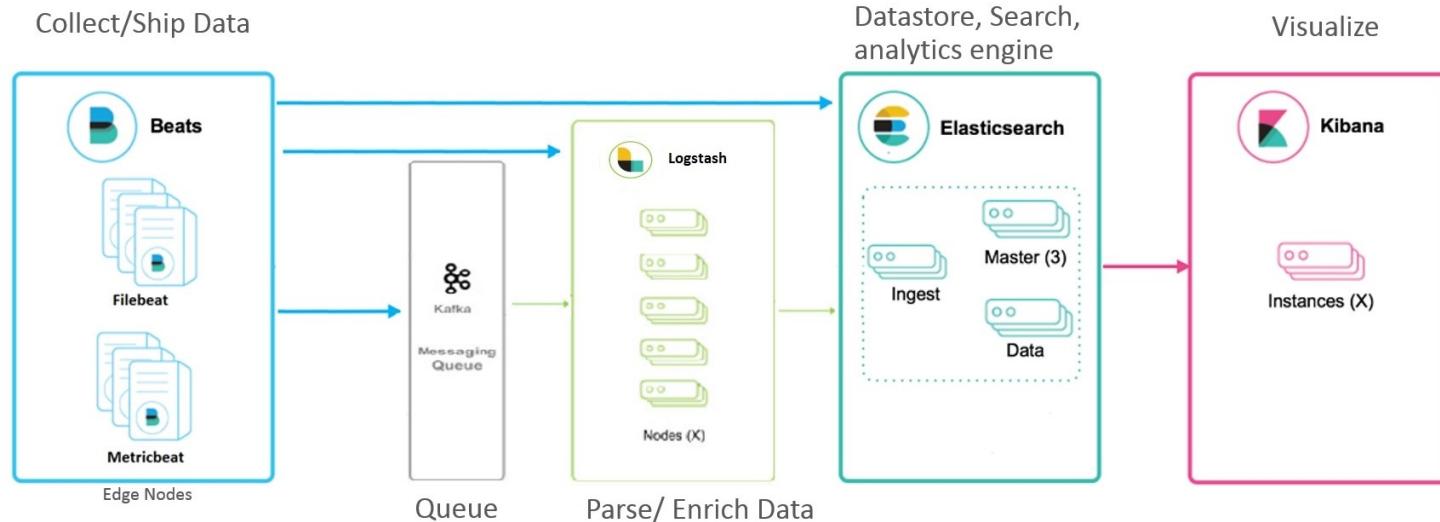
Visualizing system metrics using Kibana

- To see the dashboard refresh in real time, in the top right corner select the time and enter the appropriate refresh interval.
- Then, click the Start button as shown in the following screenshot:



Deployment architecture

- The following diagram depicts the commonly used Elastic Stack deployment architecture:





Summary

- In this lesson, we covered another Beat library called Metricbeat in detail.
- We covered how to install and configure Metricbeat so that it can send operational metrics to Elasticsearch.
- We also covered the various deployment architectures for building real-time monitoring solutions using Elasticsearch Stack in order to monitor servers and applications.

COMPLETE LAB 11