About the Tutorial

**Kibana** is an open source browser based visualization tool mainly used to analyze large volume of logs in the form of line graph, bar graph, pie charts, heat maps, region maps, coordinate maps, gauge, goals, timelion etc. The visualization makes it easy to predict or to see the changes in trends of errors or other significant events of the input source.

Kibana works in sync with Elasticsearch and Logstash which together forms the so called **ELK** stack.

Audience

This tutorial is designed for any technical or non-technical users interested in analyzing large volume of data i.e. log analysis, data analytics etc.. Kibana is browser based UI and very user friendly and any beginner can easily und in short easy for a new comer to grasp it.

Prerequisites

The installation for Kibana and Elasticsearch is straightforward and will be easy for the users to get it done quickly. To work with Kibana you need to have basic details of Elasticsearch.

Copyright & Disclaimer

@Copyright 2019 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com
# Table of Contents

- About the Tutorial
- Audience
- Prerequisites
- Copyright & Disclaimer
- Table of Contents

1. **KIBANA – OVERVIEW**
   - What is ELK Stack?
   - Features of Kibana
   - Advantages of Kibana
   - Disadvantages of Kibana

2. **KIBANA – ENVIRONMENT SETUP**
   - Elasticsearch Installation
   - Logstash Installation
   - Kibana Installation

3. **KIBANA — INTRODUCTION TO ELK STACK**
   - Load CSV data from Logstash to Elasticsearch
   - Use Data from Elasticsearch in Kibana

4. **KIBANA — LOADING SAMPLE DATA**
   - Using Logstash upload for data having fields in Elasticsearch
   - Using Dev Tools to Upload Bulk Data
   - Search for a Record in todo Index

5. **KIBANA — MANAGEMENT**
   - Create Index Pattern Without Time Filter field
   - Create Index Pattern with Time filter field
6. KIBANA — DISCOVER .............................................................. 32
   Index without date field ...................................................... 32
   Index with Date Field .......................................................... 43

7. KIBANA — AGGREGATIONS AND METRICS .................................. 49
   What is Kibana Aggregation? ................................................. 49
   Bucket Aggregation .............................................................. 49
   Date Histogram ..................................................................... 51
   Date Range ........................................................................... 52
   Filters ..................................................................................... 52
   Histogram .............................................................................. 53
   IPv4 Range ............................................................................. 54
   Range .................................................................................... 54
   Significant Terms ................................................................. 55
   Terms ..................................................................................... 55
   Metric Aggregation ............................................................... 56

8. KIBANA — CREATE VISUALIZATION ............................................. 59
   Create Visualization ............................................................. 59
   Bucket and Metric Aggregation ............................................. 61

9. KIBANA — WORKING WITH CHARTS .......................................... 68
   Horizontal Bar Chart ............................................................ 68
   Vertical Bar Chart ............................................................... 75
   Pie Chart .............................................................................. 77

10. KIBANA — WORKING WITH GRAPHS ......................................... 81
    Line Graph ........................................................................... 81
    Area Graph .......................................................................... 84

11. KIBANA — WORKING WITH HEAT MAP ..................................... 86
Getting Started with Heat Map ........................................................................................................86

12. KIBANA — WORKING WITH COORDINATE MAP .................................................................92
Create Index for Coordinate Map ..................................................................................................92
Getting Started with Coordinate Maps .............................................................................................96

13. KIBANA — WORKING WITH REGION MAP .........................................................................98
Create Index for Region Map ..........................................................................................................98
Getting Started with Region Maps ................................................................................................102
Vector Map and Join Field for Region Map ......................................................................................104
Self-hosted Vector Map and Join Field in Kibana .............................................................................107

14. KIBANA — WORKING WITH GUAGE AND GOAL ..............................................................109
Working with Gauge .......................................................................................................................109
Working with Goal ..........................................................................................................................114

15. KIBANA — WORKING WITH CANVAS ................................................................................118
Cloning an Existing Workpad in Canvas .........................................................................................122
Adding New Page to Workpad Inside Canvas ................................................................................127

16. KIBANA — CREATE DASHBOARD .......................................................................................129
Getting Started with Dashboard .......................................................................................................129
Add Visualization to Dashboard ......................................................................................................130
Save Dashboard ...............................................................................................................................133
Changing Time Range for Dashboard ..............................................................................................133
Using Search and Filter in Dashboard ............................................................................................135
Inspect and Fullscreen .....................................................................................................................141
Sharing Dashboard ..........................................................................................................................142

17. KIBANA — TIMELION ...........................................................................................................144
Where can we use Timelion? ............................................................................................................144
**1. Kibana – Overview**

**Kibana** is an open source browser based visualization tool mainly used to analyse large volume of logs in the form of line graph, bar graph, pie charts, heat maps, region maps, coordinate maps, gauge, goals, timelion etc. The visualization makes it easy to predict or to see the changes in trends of errors or other significant events of the input source. Kibana works in sync with Elasticsearch and Logstash which together forms the so called **ELK** stack.

**What is ELK Stack?**

**ELK** stands for Elasticsearch, Logstash, and Kibana. **ELK** is one of the popular log management platform used worldwide for log analysis. In the ELK stack, Logstash extracts the logging data or other events from different input sources. It processes the events and later stores them in Elasticsearch.

**Kibana** is a visualization tool, which accesses the logs from Elasticsearch and is able to display to the user in the form of line graph, bar graph, pie charts etc.

The basic flow of ELK Stack is shown in the image here:

![ELK Stack Diagram](image_url)

Logstash is responsible to collect the data from all the remote sources where the logs are filed and pushes the same to Elasticsearch.
Elasticsearch acts as a database where the data is collected and Kibana uses the data from Elasticsearch to represent the data to the user in the form of bargraphs, pie charts, heat maps as shown below:

It shows the data on real time basis, for example, day-wise or hourly to the user. Kibana UI is user friendly and very easy for a beginner to understand.

**Features of Kibana**

Kibana offers its users the following features:

**Visualization**

Kibana has a lot of ways to visualize data in an easy way. Some of the ones which are commonly used are vertical bar chart, horizontal bar chart, pie chart, line graph, heat map etc.

**Dashboard**

When we have the visualizations ready, all of them can be placed on one board – the Dashboard. Observing different sections together gives you a clear overall idea about what exactly is happening.

**Dev Tools**

You can work with your indexes using dev tools. Beginners can add dummy indexes from dev tools and also add, update, delete the data and use the indexes to create visualization.

**Reports**

All the data in the form of visualization and dashboard can be converted to reports (CSV format), embedded in the code or in the form of URLs to be shared with others.
Filters and Search query

You can make use of filters and search queries to get the required details for a particular input from a dashboard or visualization tool.

Plugins

You can add third party plugins to add some new visualization or also other UI addition in Kibana.

Coordinate and Region Maps

A coordinate and region map in Kibana helps to show the visualization on the geographical map giving a realistic view of the data.

Timelion

Timelion, also called as timeline is yet another visualization tool which is mainly used for time based data analysis. To work with timeline, we need to use simple expression language which helps us connect to the index and also perform calculations on the data to obtain the results we need. It helps more in comparison of data to the previous cycle in terms of week, month etc.

Canvas

Canvas is yet another powerful feature in Kibana. Using canvas visualization, you can represent your data in different colour combinations, shapes, texts, multiple pages basically called as workpad.

Advantages of Kibana

Kibana offers the following advantages to its users:

- Contains open source browser based visualization tool mainly used to analyse large volume of logs in the form of line graph, bar graph, pie charts, heat maps etc.
- Simple and easy for beginners to understand.
- Ease of conversion of visualization and dashboard into reports.
- Canvas visualization help to analyse complex data in an easy way.
- Timelion visualization in Kibana helps to compare data backwards to understand the performance better.

Disadvantages of Kibana

- Adding of plugins to Kibana can be very tedious if there is version mismatch.

- You tend to face issues when you want to upgrade from older version to a new one.
To start working with Kibana we need to install Logstash, Elasticsearch and Kibana. In this chapter, we will try to understand the installation of the ELK stack here.

We would discuss the following installations here:

- Elasticsearch Installation
- Logstash Installation
- Kibana Installation

**Elasticsearch Installation**

A detailed documentation on Elasticsearch exists in our library. You can check here for [elasticsearch installation](#). You will have to follow the steps mentioned in the tutorial to install Elasticsearch.

Once done with the installation, start the elasticsearch server as follows:

**Step 1**

For Windows

```
> cd kibanaproject/elasticsearch-6.5.4/elasticsearch-6.5.4/bin
> elasticsearch
```

Please note for windows user, the JAVA_HOME variable has to be set to the java jdk path.

For Linux

```
$ cd kibanaproject/elasticsearch-6.5.4/elasticsearch-6.5.4/bin
$ elasticsearch
```
The default port for elasticsearch is 9200. Once done, you can check the elasticsearch at port 9200 on localhost http://localhost:9200/as shown below:
Logstash Installation

For Logstash installation, follow this link which is already existing in our library.

Kibana Installation

Go to the official Kibana site: https://www.elastic.co/products/kibana
Click the *downloads* link on the top right corner and it will display screen as follows:

![Kibana Downloads](https://www.elastic.co/downloads)

Click the Download button for Kibana. Please note to work with Kibana we need 64 bit machine and it will not work with 32 bit.

![Download Kibana](https://www.elastic.co/downloads/kibana)

- **Version:** 6.5.4
- **Release date:** December 19, 2018
- **License:** Elastic License
- **Downloads:**
  - WINDOWS sha
  - LINUX 64-BIT sha
  - DEB 64-BIT sha
  - MAC sha
  - RPM 64-BIT sha
In this tutorial, we are going to use Kibana version 6. The download option is available for Windows, Mac and Linux. You can download as per your choice.

Create a folder and unpack the tar/zip downloads for kibana. We are going to work with sample data uploaded in elasticsearch. Thus, for now let us see how to start elasticsearch and kibana. For this, go to the folder where Kibana is unpacked.

**For Windows**

```bash
> cd kibanaproject/kibana-6.5.4/kibana-6.5.4/bin
> kibana
```

**For Linux**

```bash
$ cd kibanaproject/kibana-6.5.4/kibana-6.5.4/bin
$ kibana
```
Once Kibana starts, the user can see the following screen:
Once you see the ready signal in the console, you can open Kibana in browser using \texttt{http://localhost:5601/}. The default port on which kibana is available is 5601.

The user interface of Kibana is as shown here:

In our next chapter, we will learn how to use the UI of Kibana. To know the Kibana version on Kibana UI, go to Management Tab on left side and it will display you the Kibana version we are using currently.
Kibana is an open source visualization tool mainly used to analyze a large volume of logs in the form of line graph, bar graph, pie charts, heatmaps etc. Kibana works in sync with Elasticsearch and Logstash which together forms the so called **ELK** stack.

**ELK** stands for Elasticsearch, Logstash, and Kibana. **ELK** is one of the popular log management platform used worldwide for log analysis.

In the ELK stack:

- **Logstash** extracts the logging data or other events from different input sources. It processes the events and later stores it in Elasticsearch.

- **Kibana** is a visualization tool, which accesses the logs from Elasticsearch and is able to display to the user in the form of line graph, bar graph, pie charts etc.

In this tutorial, we will work closely with Kibana and Elasticsearch and visualize the data in different forms.

In this chapter, let us understand how to work with ELK stack together. Besides, you will also see how to:

- Load CSV data from Logstash to Elasticsearch
- Use indices from Elasticsearch in Kibana.

### Load CSV data from Logstash to Elasticsearch

We are going to use CSV data to upload data using Logstash to Elasticsearch. To work on data analysis, we can get data from kaggle.com website. Kaggle.com site has all types of data uploaded and users can use it to work on data analysis.

We have taken the **countries.csv** data from here: [https://www.kaggle.com/fernandol/countries-of-the-world](https://www.kaggle.com/fernandol/countries-of-the-world). You can download the csv file and use it.

The csv file which we are going to use has following details.

File name: **countriesdata.csv**

You can also create a dummy csv file and use it. We will be using logstash to dump this data from **countriesdata.csv** to elastic search.
Start the elasticsearch and Kibana in your terminal and keep it running. We have to create the config file for logstash which will have details about the columns of the CSV file and also other details as shown in the logstash-config file given below:

```json
input {
    file {
        path => "C:/kibanaproject/countriesdata.csv"
        start_position => "beginning"
        sincedb_path => "NUL"
    }
}
filter {
    csv {
        separator => ",,"
        columns => ["Country","Region","Population","Area"]
    }
    mutate {convert => ["Population", "integer"]}
    mutate {convert => ["Area", "integer"]}
}
output {
    elasticsearch {
        hosts => ["localhost:9200"]
        index => "countriesdata-%{+dd.MM.YYYY}"  
    }
    stdout {codec => json_lines } 
}
```

In the config file, we have created 3 components:

**Input**

We need to specify the path of the input file which in our case is a csv file. The path where the csv file is stored is given to the path field.

**Filter**

Will have the csv component with separator used which in our case is comma, and also the columns available for our csv file. As logstash considers all the data coming in as string, in-case we want any column to be used as integer, float the same has to be specified using mutate as shown above.
Output

For output, we need to specify where we need to put the data. Here, in our case we are using elasticsearch. The data required to be given to the elasticsearch is the hosts where it is running, we have mentioned it as localhost. The next field in is index which we have given the name as countries-currentdate. We have to use the same index in Kibana once the data is updated in Elasticsearch.

Save the above config file as logstash_countries.config. Note that we need to give the path of this config to logstash command in the next step.

To load the data from the csv file to elasticsearch, we need to start the elasticsearch server:

Now, run http://localhost:9200 in the browser to confirm if elasticsearch is running successfully.
We have Elasticsearch running. Now go to the path where Logstash is installed and run the following command to upload the data to Elasticsearch:

```
> logstash -f logstash_countries.conf
```

The above screen shows data loading from the CSV file to Elasticsearch. To know if we have the index created in Elasticsearch, we can check as follows:

We can see the `countriesdata-28.12.2018` index created as shown above.
The details of the index: countries-28.12.2018 is as follows:

```
{
"aliases":{},
"mappings":{
"properties":{
"@timestamp":{
"type":"date",
"@timestamp":{
"type":"text",
"fields":{
"keyword":{
"type":"keyword",
"ignore_above":256
}
}
"area":{
"type":"long",
"fields":{
"keyword":{
"type":"keyword",
"ignore_above":256
}
}
"country":{
"type":"text",
"fields":{
"keyword":{
"type":"keyword",
"ignore_above":256
}
}
"population":{
"type":"long",
"fields":{
"keyword":{
"type":"keyword",
"ignore_above":256
}
}
"region":{
"type":"text",
"fields":{
"keyword":{
"type":"keyword",
"ignore_above":256
}
}
"host":{
"type":"text",
"fields":{
"keyword":{
"type":"keyword",
"ignore_above":256
}
}
"message":{
"type":"text",
"fields":{
"keyword":{
"type":"keyword",
"ignore_above":256
}
}
"path":{
"type":"text",
"fields":{
"keyword":{
"type":"keyword",
"ignore_above":256
}
}
"settings":{
"index":{
"creation_date":154609995839,
"number_of_shards":5,
"number_of_replicas":1,
"uuid":"XzCelb0XaRyHfjRjXfV7K6Q",
"version":{
"created":6805499,
"provided_name":"countriesdata-28.12.2018"
}
"
```

Note that the mapping details with properties are created when data is uploaded from logstash to elasticsearch.

**Use Data from Elasticsearch in Kibana**

Currently, we have Kibana running on localhost, port 5601: [http://localhost:5601](http://localhost:5601). The UI of Kibana is shown here:

Note that we already have Kibana connected to Elasticsearch and we should be able to see `index :countries-28.12.2018` inside Kibana.
In the Kibana UI, click on Management Menu option on left side:

Now, click Index Management:

The indices present in Elasticsearch are displayed in index management. The index we are going to use in Kibana is countriesdata-28.12.2018.

Thus, as we already have the elasticsearch index in Kibana, next will understand how to use the index in Kibana to visualize data in the form of pie chart, bar graph, line chart etc.
We have seen how to upload data from logstash to elasticsearch. We will upload data using logstash and elasticsearch here. But about the data that has date, longitude and latitude fields which we need to use, we will learn in the upcoming chapters. We will also see how to upload data directly in Kibana, if we do not have a CSV file.

In this chapter, we will cover following topics:

- Using Logstash upload data having date, longitude and latitude fields in Elasticsearch
- Using Dev tools to upload bulk data

### Using Logstash upload for data having fields in Elasticsearch

We are going to use data in the form of CSV format and the same is taken from Kaggle.com which deals with data that you can use for an analysis.

The data [home medical visits](https://kaggle.com) to be used here is picked up from site Kaggle.com.

The following are the fields available for the CSV file:

```java
["Visit_Status","Time_Delay","City","City_id","Patient_Age","Zipcode","Latitude","Longitude","Pathology","Visiting_Date","Id_type","Id_personal","Number_Home_Visits","Is_Patient_Minor","Geo_point"]
```

The `Home_visits.csv` is as follows:

The following is the conf file to be used with logstash:

```json
input {
  file {
    path => "C:/kibanaproject/home_visits.csv"
    start_position => "beginning"
  }
}
```
filter {
    csv {
        separator => ",,"
        columns =>
            [
                "Visit_Status",
                "Time_Delay",
                "City",
                "City_id",
                "Patient_Age",
                "Zipcode",
                "Latitude",
                "Longitude",
                "Pathology",
                "Visiting_Date",
                "Id_type",
                "Id_personal",
                "Number_Home_Visits",
                "Is_Patient_Minor",
                "Geo_point"
            ]
    }
    date {
        match => 
            ["Visiting_Date", "dd-MM-YYYY HH:mm"]
        target => "Visiting_Date"
    }
    mutate {
        convert => ["Number_Home_Visits", "integer"]
    }
    mutate {
        convert => ["City_id", "integer"]
    }
    mutate {
        convert => ["Id_personal", "integer"]
    }
    mutate {
        convert => ["Id_type", "integer"]
    }
    mutate {
        convert => ["Zipcode", "integer"]
    }
    mutate {
        convert => ["Patient_Age", "integer"]
    }
    mutate {
        convert => {
            "Longitude" => "float"
        }
        convert => {
            "Latitude" => "float"
        }
    }
    mutate {
        rename => {
            "Longitude" => "[location][lon]"
            "Latitude" => "[location][lat]"
        }
    }
}
output {
    elasticsearch {
        hosts => ["localhost:9200"]
        index => "medicalvisits-%{+dd.MM.YYYY}"
    }
}
stdout {codec => json_lines }
}

By default, logstash considers everything to be uploaded in elasticsearch as string. Incase your CSV file has date field you need to do following to get the date format.

For date field:

date {
    match => ["Visiting_Date","dd-MM-YYYY HH:mm"]
    target => "Visiting_Date"
}

In-case of geo location, elasticsearch understands the same as:

"location": {
    "lat": 41.565505000000044,
    "lon": 2.2349995750000695
}

So we need to make sure we have Longitude and Latitude in the format elasticsearch needs it. So first we need to convert longitude and latitude to float and later rename it so that it is available as part of location json object with lat and lon. The code for the same is shown here:

```
mutate {
    convert => { "Longitude" => "float" }
    convert => { "Latitude" => "float" }
}
mutate {
    rename => {
        "Longitude" => "[location][lon]"
        "Latitude" => "[location][lat]"
    }
}
```

For converting fields to integers, use the following code:
Once the fields are taken care, run the following command to upload the data in elasticsearch:

- Go inside Logstash bin directory and run the following command.
  
  ```bash
  logstash -f logstash_homevisits.conf
  ```

- Once done you should see the index mentioned in logstash conf file in elasticsearch as shown below:

```
<table>
<thead>
<tr>
<th>Index Name</th>
<th>Date</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>logstash_homevisits</td>
<td>2019-01-26</td>
<td>1098</td>
</tr>
<tr>
<td>logstash_install</td>
<td>2019-01-25</td>
<td>1071</td>
</tr>
</tbody>
</table>
```

We can now create index pattern on above index uploaded and use it further for creating visualization.

## Using Dev Tools to Upload Bulk Data

We are going to use Dev Tools from Kibana UI. Dev Tools is helpful to upload data in Elasticsearch, without using Logstash. We can post, put, delete, search the data we want in Kibana using Dev Tools.

In this section, we will try to load sample data in Kibana itself. We can use it to practice with the sample data and play around with Kibana features to get a good understanding of Kibana.

Let us take the json data from the following [url](#) and upload the same in Kibana. Similarly, you can try any sample json data to be loaded inside Kibana.

Before we start to upload the sample data, we need to have the json data with indices to be used in elasticsearch. When we upload it using logstash, logstash takes care to add the indices and the user does not have to bother about the indices which are required by elasticsearch.

**Normal Json Data**
The json code to used with Kibana has to be with indexed as follows:

```json
{
"index":{"_index":"shakespeare","_id":0}
}
{
"index":{"_index":"shakespeare","_id":1}
}
{
"index":{"_index":"shakespeare","_id":2}
}
{
"type":"act","line_id":1,"play_name":"Henry IV","speech_number":null,"line_number":null,"speaker":null,"text_entry":"ACT I"
}
{
"type":"scene","line_id":2,"play_name":"Henry IV","speech_number":null,"line_number":null,"speaker":null,"text_entry":"SCENE I. London. The palace."
}
{
"type":"line","line_id":3,"play_name":"Henry IV","speech_number":null,"line_number":null,"speaker":null,"text_entry":"Enter KING HENRY, LORD JOHN OF LANCASTER, the EARL of WESTMORELAND, SIR WALTER BLUNT, and others"
}
```

Note that there is an additional data that goes in the jsonfile:

```json
{"index":{"_index":"nameofindex","_id":key}}
```

To convert any sample json file compatible with elasticsearch, here we have a small code in php which will output the json file given to the format which elasticsearch wants:

**PHP Code**

```php
<?php

$myfile = fopen("todo.json", "r") or die("Unable to open file!"); // your json file here

$alldata = fread($myfile,filesize("todo.json"));

fclose($myfile);

$farray = json_decode($alldata);

$afinalarray = [];

$index_name = "todo";
$i=0;
```
We have taken the todo json file from [https://jsonplaceholder.typicode.com/todos](https://jsonplaceholder.typicode.com/todos) and use php code to convert to the format we need to upload in Kibana.

To load the sample data, open the dev tools tab as shown below:

![Kibana Dev Tools](image)

We are now going to use the console as shown above. We will take the json data which we got after running it through php code.
The command to be used in dev tools to upload the json data is:

```
POST _bulk
```

Note that the name of the index we are creating is `todo`.

Once you click the green button the data is uploaded, you can check if the index is created or not in elasticsearch as follows:
You can check the same in dev tools itself as follows:

Command:

```
GET /_cat/indices
```

If you want to search something in your index:todo, you can do that as shown below:

Command in dev tool

```
GET /todo/_search
```

The output of the above search is as shown below:
It gives all the records present in the todo index. The total records we are getting is 200.

**Search for a Record in todo Index**

We can do that using the following command:

```json
GET /todo/_search
{
   "query":{
      "match":{
         "title": "delectus aut autem"
      }
   }
}
```
We are able to fetch the records which match with the title we have given.
The Management section in Kibana is used to manage the index patterns. In this chapter, we will discuss the following:

- Create Index Pattern without Time filter field
- Create Index Pattern with Time filter field

Create Index Pattern Without Time Filter field

To do this, go to Kibana UI and click Management:

To work with Kibana, we first have to create index which is populated from elasticsearch. You can get all the indices available from Elasticsearch -> Index Management as shown:
At present elasticsearch has the above indices. The Docs count tells us the no of records available in each of the index. If there is any index which is updated, the docs count will keep changing. Primary storage tells the size of each index uploaded.

To create New index in Kibana, we need to click on Index Patterns as shown below:

Once you click Index Patterns, we get the following screen:

Note that the Create Index Pattern button is used to create a new index. Recall that we already have countriesdata-28.12.2018 created at the very start of the tutorial.
Create Index Pattern with Time filter field

Click on Create Index Pattern to create a new index.

The indices from elasticsearch are displayed, select one to create a new index.

Now, click Next step.

The next step is to configure the setting, where you need to enter the following:

- Time filter field name is used to filter data based on time. The dropdown will display all time and date related fields from the index.

In the image shown below, we have Visiting_Date as a date field. Select Visiting_Date as the Time Filter field name.
Click **Create index pattern** button to create the index. Once done it will display all the fields present in your index medicalvisits-26.01.2019 as shown below:

We have following fields in the index medicalvisits-26.01.2019:

```plaintext
["Visit_Status","Time_Delay","City","City_id","Patient_Age","Zipcode","Latitude ","Longitude","Pathology","Visiting_Date","Id_type","Id_personal","Number_Home_Visits","Is_Patient_Minor","Geo_point"]
```

The index has all the data for home medical visits. There are some additional fields added by elasticsearch when inserted from logstash.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Format</th>
<th>Searchable</th>
<th>Aggregatable</th>
<th>Excluded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is_Patient_Minor</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Is_Patient_Minor.keyword</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number_Home_Visits</td>
<td>number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pathology</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pathology.keyword</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Patient_Age</td>
<td>number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time_Delay</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time_Delay.keyword</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visit_Status</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visit_Status.keyword</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visiting_Date</td>
<td>date</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zipcode</td>
<td>number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_id</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_index</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_score</td>
<td>number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_source</td>
<td>_source</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>_type</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>host</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>host.keyword</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>location.lat</td>
<td>number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>location.lon</td>
<td>number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>message</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>message.keyword</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>path</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>path.keyword</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This chapter discusses the Discover Tab in Kibana UI. We will learn in detail about the following concepts:

- Index without date field
- Index with date field

## Index without date field

Select Discover on the left side menu as shown below:

On the right side, it displays the details of the data available in **countriesdata-28.12.2018** index we created in previous chapter.

On the top left corner, it shows the total number of records available:

We can get the details of the data inside the index (**countriesdata-28.12.2018**) in this tab. On the top left corner in screen shown above, we can see Buttons like New, Save, Open, Share, Inspect and Auto-refresh.
If you click Auto-refresh, it will display the screen as shown below:

You can set the auto-refresh interval by clicking on the seconds, minutes or hour from above. Kibana will auto-refresh the screen and get fresh data after every interval timer you set.

The data from **index:countriesdata-28.12.2018** is displayed as shown below:

All the fields along with the data are shown row wise. Click the arrow to expand the row and it will give you details in Table format or JSON format.
### JSON Format

<table>
<thead>
<tr>
<th>Table</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
| 3 | "_type": "doc",
| 4 | "_id": "jdYj9wPcHgqUKuTwX",
| 5 | "_version": 1,
| 6 | "_score": 1,
| 7 | "_source": 
| 8 | 
| 9 | "path": "C:/kibana\project\countriesdata.csv",
| 10 | "version": "1",
| 11 | "timestamp": "2018-12-28T14:06:27.127Z",
| 12 | "Population": 1639833,
| 13 | "region": "SUB-SAHARAN AFRICA",
| 14 | "country": "Botswana",
| 15 | "host": "admin",
| 16 | "area": 600370
| 17 | },
| 18 | "fields": [
| 19 | "timestamp": {
| 20 | "2018-12-28T14:06:27.127Z"
| 21 | }
| 22 | ]
| 23 | }

There is a button on the left side called View single document.
If you click it, it will display the row or the data present in the row inside the page as shown below:

Though we are getting all the data details here, it is difficult to go through each of them.

Now let us try to get the data in tabular format. One way to expand one of the row and click the toggle column option available across each field is shown below:
Click on Toggle column in table option available for each and you will notice the data being shown in table format:

<table>
<thead>
<tr>
<th>Country</th>
<th>Area</th>
<th>Region</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bolivia</td>
<td>1,098,500</td>
<td>LATIN AMER. &amp; CARIB</td>
<td>8,989,046</td>
</tr>
<tr>
<td>Botswana</td>
<td>608,370</td>
<td>SUB-SAHARAN AFRICA</td>
<td>1,639,533</td>
</tr>
<tr>
<td>Burma</td>
<td>678,500</td>
<td>ASIA (EX. NEAR EAST)</td>
<td>47,382,633</td>
</tr>
<tr>
<td>China</td>
<td>9,596,968</td>
<td>ASIA (EX. NEAR EAST)</td>
<td>1,313,073,713</td>
</tr>
<tr>
<td>Costa Rica</td>
<td>51,120</td>
<td>LATIN AMER. &amp; CARIB</td>
<td>4,075,261</td>
</tr>
<tr>
<td>East Timor</td>
<td>15,007</td>
<td>ASIA (EX. NEAR EAST)</td>
<td>1,062,777</td>
</tr>
<tr>
<td>Equatorial Guinea</td>
<td>28,051</td>
<td>SUB-SAHARAN AFRICA</td>
<td>540,189</td>
</tr>
<tr>
<td>Gaza Strip</td>
<td>360</td>
<td>NEAR EAST</td>
<td>1,428,757</td>
</tr>
<tr>
<td>Gibraltar</td>
<td>7</td>
<td>WESTERN EUROPE</td>
<td>27,928</td>
</tr>
<tr>
<td>India</td>
<td>3,287,500</td>
<td>ASIA (EX. NEAR EAST)</td>
<td>1,095,351,995</td>
</tr>
<tr>
<td>Ireland</td>
<td>70,210</td>
<td>WESTERN EUROPE</td>
<td>4,062,235</td>
</tr>
<tr>
<td>Kazakhstan</td>
<td>2,717,300</td>
<td>C.W. OF IND. STATES</td>
<td>15,233,244</td>
</tr>
</tbody>
</table>

Here, we have selected fields Country, Area, Region and Population. Collapse the expanded row and you should see all the data in tabular format now.
The fields we selected are displayed on the left side of the screen as shown below:

Observe that there are 2 options: *Selected fields* and *Available fields*. The fields we have selected to show in tabular format are a part of selected fields. In case you want to remove any field you can do so by clicking the remove button which will be seen across the field name in selected field option.
Once removed, the field will be available inside the *Available fields* where you can add back by clicking the add button which will be shown across the field you want. You can also use this method to get your data in tabular format by choosing the required fields from *Available fields*.

We have a search option available in Discover, which we can use to search for data inside the index. Let us try examples related to search option here:

Suppose you want to search for country India, you can do as follows:

You can type your search details and click the Update button. If you want to search for countries starting with *Aus*, you can do so as follows:

Click Update to see the results

Here, we have two countries starting with *Aus*. The search field has a Options button as shown above. When a user clicks it, it displays a toggle button which when ON helps in writing the search query.
Turn on query features and type the field name in search, it will display the options available for that field.

For example, Country field is a string and it displays following options for the string field:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Country.keyword</code></td>
<td>Filter results that contain <code>Country.keyword</code></td>
</tr>
<tr>
<td><code>Country</code></td>
<td>Filter results that contain <code>Country</code></td>
</tr>
<tr>
<td><code>=</code></td>
<td><code>equals</code> some value</td>
</tr>
<tr>
<td><code>=*</code></td>
<td><code>exists</code> in any form</td>
</tr>
</tbody>
</table>
Similarly, Area is a Number field and it displays following options for Number field:

You can try out different combination and filter the data as per your choice in Discover field. The data inside the Discover tab can be saved using the Save button, so that you can use it for future purpose.

To save the data inside discover click on the save button on top right corner as shown below:
Give title to your search and click Confirm Save to save it. Once saved, next time you visit the Discover tab, you can click the Open button on the top right corner to get the saved titles as shown below:

You can also share the data with others using the Share button available on top right corner. If you click it, you can find sharing options as shown below:

You can share it using CSV Reports or in the form of Permalinks.

The option available onclick on CSV Reports are:

Click Generate CSV to get the report to be shared with others.
The option available onclick of Permalinks are as follows:

<table>
<thead>
<tr>
<th>Permalink</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Generate the link as</strong></td>
</tr>
<tr>
<td>📺 Snapshot</td>
</tr>
<tr>
<td>📸 Saved object</td>
</tr>
<tr>
<td>📄 Short URL</td>
</tr>
</tbody>
</table>

The Snapshot option will give a Kibana link which will display data available in the search currently.

The Saved object option will give a Kibana link which will display the recent data available in your search.

In general, Kibana gives longer links; there is option at the bottom to get Short URL. A sample short url for Snapshot and Saved object are shown here:

**Snapshot:** [http://localhost:5601/goto/309a983483fccd423950cfb708fabfa5](http://localhost:5601/goto/309a983483fccd423950cfb708fabfa5)

**Saved Object:** [http://localhost:5601/app/kibana#/discover/40bd89d0-10b1-11e9-9876-4f3d759b471e?_g=()](http://localhost:5601/app/kibana#/discover/40bd89d0-10b1-11e9-9876-4f3d759b471e?_g=())

You can work with Discover tab and search options available and the result obtained can be saved and shared with others.
Index with Date Field

Go to Discover tab and select index: **medicalvisits-26.01.2019**

It has displayed the message: “No results match your search criteria”, for the last 15 minutes on the index we have selected. The index has data for years 2015, 2016, 2017 and 2018.

Change the time range as shown below:
Click Absolute tab.

Select the date From: 1st Jan 2017 and To: 31st Dec 2017 as we will analyze data for year 2017.
Click the Go button to add the timerange. It will display you the data and bar chart as follows:

This is the monthly data for the year 2017:

Since we also have the time stored along with date, we can filter the data on hours and minutes too.

The figure shown above displays the hourly data for the year 2017.
Here the fields displayed from the index: medicalvisits-26.01.2019
We have the available fields on left side as shown below:

You can select the fields from available fields and convert the data into tabular format as shown below. Here we have selected the following fields:

- City
- Is_Patient_Minor
- Number_Home_Vis...
The tabular data for above fields is shown here:

<table>
<thead>
<tr>
<th>Time</th>
<th>City</th>
<th>Pathology</th>
<th>Is_Patient</th>
<th>Number_of_Home_Visits</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 9th 2017, 15:32:00.000</td>
<td>Cardedeu</td>
<td>APSODALITIS AGUDA</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>November 10th 2017, 20:51:00.000</td>
<td>Bellarte</td>
<td>APSODALITIS AGUDA</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>November 10th 2017, 20:41:00.000</td>
<td>Sant Cugat</td>
<td>APSODALITIS AGUDA</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>November 17th 2017, 17:21:00.000</td>
<td>Rubi</td>
<td>APSODALITIS AGUDA</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>November 17th 2017, 20:52:00.000</td>
<td>Sant Pau de Codines</td>
<td>APSODALITIS AGUDA</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>November 19th 2017, 06:16:00.000</td>
<td>Terrassa</td>
<td>APSODALITIS AGUDA</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>November 19th 2017, 20:03:00.000</td>
<td>Sant Joan Despi</td>
<td>APSODALITIS AGUDA</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>November 21st 2017, 07:55:00.000</td>
<td>Sant Cugat</td>
<td>APSODALITIS AGUDA</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>November 21st 2017, 17:34:00.000</td>
<td>Sant Cugat</td>
<td>APSODALITIS AGUDA</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>November 20th 2017, 22:46:00.000</td>
<td>Sant Cugat</td>
<td>APSODALITIS AGUDA</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>November 20th 2017, 09:10:00.000</td>
<td>Sant Cugat</td>
<td>APSODALITIS AGUDA</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>November 28th 2017, 19:51:00.000</td>
<td>Santa Eulalia de Roncana</td>
<td>APSODALITIS AGUDA</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>November 29th 2017, 03:20:00.000</td>
<td>Sonsetell</td>
<td>APSODALITIS AGUDA</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>November 29th 2017, 16:30:00.000</td>
<td>Sant Cugat</td>
<td>APSODALITIS AGUDA</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
The two terms that you come across frequently during your learning of Kibana are Bucket and Metrics Aggregation. This chapter discusses what role they play in Kibana and more details about them.

**What is Kibana Aggregation?**

Aggregation refers to the collection of documents or a set of documents obtained from a particular search query or filter. Aggregation forms the main concept to build the desired visualization in Kibana.

Whenever you perform any visualization, you need to decide the criteria, which means in which way you want to group the data to perform the metric on it.

In this section, we will discuss two types of Aggregation:

- Bucket Aggregation
- Metric Aggregation

**Bucket Aggregation**

A bucket mainly consists of a key and a document. When the aggregation is executed, the documents are placed in the respective bucket. So at the end you should have a list of buckets, each with a list of documents. The list of Bucket Aggregation you will see while creating visualization in Kibana is shown below:
Bucket Aggregation has the following list:

- Date Histogram
- Date Range
- Filters
- Histogram
- IPv4 Range
- Range
- Significant Terms
- Terms

While creating, you need to decide one of them for Bucket Aggregation i.e. to group the documents inside the buckets.

As an example, for analysis, consider the countries data that we have uploaded at the start of this tutorial. The fields available in the countries index is country name, area, population, region. In the countries data, we have name of the country along with its population, region and the area.

Let us assume that we want region wise data. Then, the countries available in each region becomes our search query, so in this case the region will form our buckets. The block diagram below shows that R1, R2,R3,R4,R5 and R6 are the buckets which we got and c1 , c2 ..c25 are the list of documents which are part of the buckets R1 to R6.
We can see that there are some circles in each of the bucket. They are set of documents based on the search criteria and considered to be falling in each of the bucket. In the bucket R1, we have documents c1, c8 and c15. These documents are the countries that falling in that region, same for others. So if we count the countries in Bucket R1 it is 3, 6 for R2, 6 for R3, 2 for R4, 5 for R5 and 4 for R6.

So through bucket aggregation, we can aggregate the document in buckets and have a list of documents in those buckets as shown above.

The list of Bucket Aggregation we have so far is:

- Date Histogram
- Date Range
- Filters
- Histogram
- IPv4 Range
- Range
- Significant Terms
- Terms

Let us now discuss how to form these buckets one by one in detail.

**Date Histogram**

Date Histogram aggregation is used on a date field. So the index that you use to visualize, if you have date field in that index than only this aggregation type can be used. This is a multi-bucket aggregation which means you can have some of the documents as a part of more than 1 bucket. There is an interval to be used for this aggregation and the details are as shown below:

![Date Histogram Aggregation](image)

When you Select Buckets Aggregation as Date Histogram, it will display the Field option which will give only the date related fields. Once you select your field, you need to select the Interval which has the following details:
So the documents from the index chosen and based on the field and interval chosen will categorize the documents in buckets. For example, if you chose the interval as monthly, the documents based on date will be converted into buckets and based on the month i.e, Jan-Dec the documents will be put in the buckets. Here Jan,Feb,..Dec will be the buckets.

**Date Range**

You need a date field to use this aggregation type. Here we will have a date range, that is from date and to date are to be given. The buckets will have its documents based on the form and to date given.
**Filters**

With Filters type aggregation, the buckets will be formed based on the filter. Here you will get a multi-bucket formed as based on the filter criteria one document can exists in one or more buckets.

Using filters, users can write their queries in the filter option as shown below:

You can add multiple filters of your choice by using Add Filter button.

**Histogram**

This type of aggregation is applied on a number field and it will group the documents in a bucket based on the interval applied. For example, 0-50, 50-100, 100-150 etc.
IPv4 Range

This type of aggregation is used and mainly used for IP addresses.

The index that we have that is the countriesdata-28.12.2018 does not have field of type IP so it displays a message as shown above. If you happen to have the IP field, you can specify the From and To values in it as shown above.

Range

This type of Aggregation needs fields to be of type number. You need to specify the range and the documents will be listed in the buckets falling in the range.

You can add more range if required by clicking on the Add Range button.
**Significant Terms**

This type of aggregation is mostly used on the string fields.

![Significant Terms Aggregation](image)

**Terms**

This type of aggregation is used on all the available fields namely number, string, date, boolean, IP address, timestamp etc. Note that this is the aggregation we are going to use in all our visualization that we are going to work on in this tutorial.

![Terms Aggregation](image)
We have an option order by which we will group the data based on the metric we select. The size refers to the number of buckets you want to display in the visualization.

Next, let us talk about Metric Aggregation.

**Metric Aggregation**

Metric Aggregation mainly refers to the maths calculation done on the documents present in the bucket. For example if you choose a number field the metric calculation you can do on it is COUNT, SUM, MIN, MAX, AVERAGE etc.

A list of metric aggregation that we shall discuss is given here:

![Metric Aggregation selection](image)

In this section, let us discuss the important ones which we are going to use often:

- Average
- Count
- Max
- Min
- Sum

The metric will be applied on the individual bucket aggregation that we have already discussed above.
Next, let us discuss the list of metrics aggregation here:

**Average**

This will give the average for the values of the documents present in the buckets. For example:

R1 to R6 are the buckets. In R1 we have c1, c8 and c15. Consider the value of c1 is 300, c8 is 500 and c15 is 700. Now to get the average value of R1 bucket:

\[
R1 = \frac{\text{value of } c1 + \text{value of } c8 + \text{value of } c15}{3} = \frac{300 + 500 + 700}{3} = 500.
\]

The average is 500 for bucket R1. Here the value of the document could be anything like if you consider the countries data it could be the area of the country in that region.

**Count**

This will give the count of documents present in the bucket. Suppose you want the count of the countries present in the region, it will be the total documents present in the buckets. For example, R1 it will be 3, R2 = 6, R3 = 5, R4 = 2, R5 = 5 and R6 = 4.
**Max**

This will give the max value of the document present in the bucket. Considering the above example if we have area wise countries data in the region bucket. The max for each region will be the country with the max area. So it will have one country from each region i.e. R1 to R6.

**in**

This will give the min value of the document present in the bucket. Considering above example if we have area wise countries data in the region bucket. The min for each region will be the country with the minimum area. So it will have one country from each region i.e. R1 to R6.

**Sum**

This will give the sum of the values of the document present in the bucket. For example if you consider the above example if we want the total area or countries in the region, it will be sum of the documents present in the region.

For example, to know the total countries in the region R1 it will be 3, R2 = 6, R3 = 5, R4 = 2, R5 = 5 and R6 = 4.

In case we have documents with area in the region than R1 to R6 will have the country wise area summed up for the region.
We can visualize the data we have in the form of bar charts, line graphs, pie charts etc. In this chapter, we will understand how to create visualization.

**Create Visualization**

Go to Kibana Visualization as shown below:

![Kibana Visualization](image)

We do not have any visualization created, so it shows blank and there is a button to create one.
Click the button **Create a visualization** as shown in the screen above and it will take you to the screen as shown below:

Here you can select the option which you need to visualize your data. We will understand each one of them in detail in the upcoming chapters. Right now will select pie chart to start with.
Once you select the visualization type, now you need to select the index on which you want to work on, and it will take you the screen as shown below:

Now we have a default pie chart. We will use the countriesdata-28.12.2018 to get the count of regions available in the countries data in pie chart format.

**Bucket and Metric Aggregation**

The left side has metrics, which we will select as count. In Buckets, there are 2 options Split slices and split chart. We will use the option Split slices.
Now, select Split Slices and it will display following options:

Now, select the Aggregation as Terms and it will display more options to be entered as follows:
The Fields dropdown will have all the field from the index: `countriesdata` chosen. We have chosen the Region field and Order By. Note that we have chosen, the metric Count for Order By. We will order it Descending and the size we have taken as 10. It means here, we will get the top 10 regions count from the countries index.

Now, click the analyse button as highlighted below and you should see the pie chart updated on right side.
**Pie chart display**

All the regions are listed at the right top corner with colours and the same colour is shown in the pie chart. If you mouse over the pie chart it will give the count of the region and also the name of the region as shown below:
So it tells us that 22.77% of region is occupied by Sub-Saharan Africa from the countries data we have uploaded.
Asia region covers 12.5% and the count is 28.

Now we can save the visualization by clicking on the save button on top right corner as shown below:

Now, save the visualization so that it can be used later.

We can also get the data as we want by using the search option as shown below:
We have filtered data for countries starting with Aus*. We will understand more on pie-chart and other visualization in the upcoming chapters.
Let us explore and understand the most commonly used charts in visualization.

- Horizontal Bar Chart
- Vertical Bar Chart
- Pie Chart

The following are the steps to be followed to create above visualization. Let us start with Horizontal Bar.

**Horizontal Bar Chart**

Open Kibana and click Visualize tab on left side as shown below:
Click the + button to create a new visualization:

Click the Horizontal Bar listed above. You will have to make a selection of the index you want to visualize.

From a New Search, Select Index

Filter... 1 of 1

Name ▲

countriesdata-28.12.2018
Select the **countriesdata-28.12.2018** index as shown above. On selecting the index, it displays a screen as shown below:

![Kibana interface showing a horizontal graph with Y-Axis Count and Buckets options](image)

It shows a default count. Now, let us plot a horizontal graph where we can see the data of top 10 country wise populations.

For this purpose, we need to select what we want on the Y and X axis. Hence, select the Bucket and Metric Aggregation:

![Bucket and Metric Aggregation options](image)
Now, if you click on Y-Axis, it will display the screen as shown below:

Now, select the Aggregation that you want from the options shown here:

Note that here we will select the Max aggregation as we want to display data as per the max population available.

Next we have to select the field whose max value is required. In the index countriesdata-28.12.2018, we have only 2 numbers field – area and population.
Since we want the max population, we select the Population field as shown below:

![Population Metric Selection]

By this, we are done with the Y-axis. The output that we get for Y-axis is as shown below:

![Population Chart]
Now let us select the X-axis as shown below:

If you select X-Axis, you will get the following output:

Choose Aggregation as Terms.
Choose the field from the dropdown. We want country wise population so select country field.

Order by we have following options:

- metric: Max Population
- Custom Metric
- Alphabetical

We are going to choose the order by as Max Population as want the country with highest population to be displayed first and so on. Once the data we want is added click on the apply changes button on top of the Metrics data as shown below:

Once you click apply changes, we have the horizontal graph wherein we can see that China is the country with highest population, followed by India, United States etc.
Similarly, you can plot different graphs by choosing the field you want. Next, we will save this visualization as max_population to be used later for Dashboard creation.

In the next section, we will create vertical bar chart.

**Vertical Bar Chart**

Click the Visualize tab and create a new visualization using vertical bar and index as `countriesdata-28.12.2018`.

In this vertical bar visualization, we will create bar graph with countries wise area, i.e. countries will be displayed with highest area.

So let us select the Y and X axes as shown below:

**Y-axis**

![Y-axis settings](image-url)
**X-axis**

When we apply the changes here, we can see the output as shown below:
From the graph, we can see that Russia is having the highest area, followed by Canada and United States. Please note this data is picked from the index countriesdata, and its dummy data, so figures might not be correct with live data.

Let us save this visualization as `countrywise_maxarea` to be used with dashboard later.

Next, let us work on Pie chart.

**Pie Chart**

So first create a visualization and select the pie chart with index as `countriesdata`. We are going to display the count of regions available in the countriesdata in pie chart format.

The left side has metrics which will give count. In Buckets, there are 2 options: Split slices and split chart. Now, we will use the option Split slices.

Now, if you select Split Slices, it will display the following options:
Select the Aggregation as Terms and it will display more options to be entered as follows:

The Fields dropdown will have all the fields from the index chosen. We have selected Region field and Order By that we have selected as Count. We will order it Descending and the size will take as 10. So here we will get the 10 regions count from the countries index.

Now, click the play button as highlighted below and you should see the pie chart updated on the right side.
Pie chart display

All the regions are listed at the right top corner with colours and the same colour is shown in the pie chart. If you mouse over the pie chart, it will give the count of the region and also the name of the region as shown below:
Thus, it tells us that 22.77% of region is occupied by Sub-Saharan Afri in the countries data we have uploaded.

From the pie chart, observe that the Asia region covers 12.5% and the count is 28.

Now we can save the visualization by clicking the save button on top right corner as shown below:

Now, save the visualization so that it can be used later in dashboard.
In this chapter, we will discuss the two types of graphs used in visualization:

- Line Graph
- Area

**Line Graph**

To start with, let us create a visualization, choosing a line graph to display the data and use `countriesdata` as the index. We need to create the Y-axis and X-axis and the details for the same are shown below:

**For Y-axis**

Observe that we have taken Max as the Aggregation. So here we are going to show data presentation in line graph. Now, we will plot graph that will show the max population country wise. The field we have taken is Population since we need maximum population country wise.
For X-axis:

**Buckets**

- **Aggregation**: Terms
- **Field**: Country.keyword
- **Order By**: metric: Max Population
- **Order**: Descend
  - **Size**: 5
- **Options**:
  - Group other values in separate bucket
  - Show missing values
On x-axis we have taken Terms as Aggregation, Country.keyword as Field and metric: Max Population for Order By, and order size is 5. So it will plot the 5 top countries with max population. After applying the changes, you can see the line graph as shown below:

So we have Max population in China, followed by India, United States, Indonesia and Brazil as the top 5 countries in population.

Now, let us save this line graph so that we can use in dashboard later.

Click Confirm Save and you can save the visualization.
Area Graph

Go to visualization and choose area with index as countriesdata. We need to select the Y-axis and X-axis. We will plot area graph for max area for country wise.

So here the X-axis and Y-axis will be as shown below:
After you click the apply changes button, the output that we can see is as shown below:

From the graph, we can observe that Russia has the highest area, followed by Canada, United States, China and Brazil. Save the visualization to use it later.
In this chapter we will understand how to work with heat map. Heat map will show the data presentation in different colours for the range selected in the data metrics.

**Getting Started with Heat Map**

To start with, we need to create visualization by clicking on the visualization tab on the left side as shown below:

Select visualization type as heat map as shown above. It will ask you to choose the index as shown below:
Select the index *countriesdata-28.12.2018* as shown above. Once the index is selected the we have the data to be selected as shown below:
Select the Metrics as shown below:

![Metrics Selection](image)

Select Max Aggregation from dropdown as shown below:

![Aggregation Selection](image)

We have select Max since we want to plot Max Area country wise.
Now will select the values for Buckets as shown below:

![Buckets Image]

Now, let us select the X-Axis as shown below:

![X-Axis Image]
We have used Aggregation as Terms, Field as Country and Order By Max Area. Click on Apply Changes as shown below:
If you click Apply Changes, the heat map looks as shown below:

The heat map is shown with different colours and the range of areas are displayed at the right side. You can change the colour by click on the small circles next to the area range as shown below:
Coordinate maps in Kibana will show you the geographic area and mark the area with circles based on aggregation you specify.

**Create Index for Coordinate Map**

The Bucket aggregation used for coordinate map is geohash aggregation. For this type of aggregation, your index which you are going to use should have a field of type geo point. The geo point is combination of latitude and longitude.

We will create an index using Kibana dev tools and add bulk data to it. We will add mapping and add the geo_point type that we need.

The data that we are going to use is shown here:

```json
{"index":{"_id":1}}
{"location": "2.089330000000046,41.47367000000008", "city": "SantCugat"}
{"index":{"_id":2}}
{"location": "2.2947825000000677,41.601800991000076", "city": "Granollers"}
{"index":{"_id":3}}
{"location": "2.110597495300474,41.5496295760424", "city": "Sabadell"}
{"index":{"_id":4}}
{"location": "2.13260567808395,41.5370461908878", "city": "Barbera"}
{"index":{"_id":5}}
{"location": "2.151270020052683,41.49777991834515", "city": "Cerdanyola"}
{"index":{"_id":6}}
{"location": "2.1364609496220606,41.371303520399344", "city": "Barcelona"}
{"index":{"_id":7}}
{"location": "2.0819450306711165,41.385491966414705", "city": "Sant Just Desvern"}
{"index":{"_id":8}}
{"location": "2.00532082278266,41.542294286427385", "city": "Rubi"}
{"index":{"_id":9}}
{"location": "1.9560805366930398,41.56142635214226", "city": "Viladecavalls"}
{"index":{"_id":10}}
{"location": "2.09205348251486,41.39327140161001", "city": "Esplugas de Llobregat"}
```
Now, run the following commands in Kibana Dev Tools as shown below:

```
PUT /cities
{
   "mappings": {
      "_doc": {
         "properties": {
            "location": {
               "type": "geo_point"
            }
         }
      }
   }
}

POST /cities/_city/_bulk?refresh
{"index":{"_id":1}}
{"location": "2.089330000000046,41.47367000000008", "city": "SantCugat"}
{"index":{"_id":2}}
{"location": "2.2947825000000677,41.601800991000076", "city": "Granollers"}
{"index":{"_id":3}}
{"location": "2.1105957495300474,41.5496295760424", "city": "Sabadell"}
{"index":{"_id":4}}
{"location": "2.132605678083895,41.5370461908878", "city": "Barbera"}
{"index":{"_id":5}}
{"location": "2.151270020052683,41.497779918345415", "city": "Cerdanyola"}
{"index":{"_id":6}}
{"location": "2.1364609496220606,41.371303520399344", "city": "Barcelona"}
{"index":{"_id":7}}
{"location": "2.081945030671165,41.385491966414705", "city": "Sant Just Desvern"}
{"index":{"_id":8}}
{"location": "2.00532082278266,41.542294286427385", "city": "Rubi"}
{"index":{"_id":9}}
{"location": "1.9560805366930398,41.561426352142226", "city": "Viladecavalls"}
{"index":{"_id":10}}
```
Now, run the above commands in Kibana dev tools:

```
PUT /cities
{
  "mappings": {
    "doc": {
      "properties": {
        "location": {
          "type": "geo_point"
        }
      }
    }
  }
}
```

The above will create index name cities of type _doc and the field location is of type geo_point.

Now let's add data to the index: cities:

```
POST /cities/doc/_bulkrefresh
{
  "errors": false,
  "index": {
    "_index": "cities",
    "_type": "doc",
    "_id": "1",
    "created": true,
    "shards": {
      "total": 2,
      "successful": 1,
      "failed": 0
    },
    "_seq_no": 86,
    "primary_term": 1,
    "status": 201
  }
}
```

"location": "2.09205348251486,41.39327140161001", "city": "Esplugas de Llobregat"
We are done creating index name cites with data. Now let us Create index pattern for cities using Management tab.

The details of fields inside cities index are shown here:

We can see that location is of type geo_point. We can now use it to create visualization.
Getting Started with Coordinate Maps

Go to Visualization and select coordinate maps.

Select the index pattern cities and configure the Aggregation metric and bucket as shown below:
If you click on Analyze button, you can see the following screen:

Based on the longitude and latitude, the circles are plotted on the map as shown above.
With this visualization, you see the data represented on the geographical world map. In this chapter, let us see this in detail.

**Create Index for Region Map**

We will create a new index to work with region map visualization. The data that we are going to upload is shown here:

```json
{"index":{"_id":1}}
{"country": "China", "population": "1313973713"}
{"index":{"_id":2}}
{"country": "India", "population": "1095351995"}
{"index":{"_id":3}}
{"country": "United States", "population": "298444215"}
{"index":{"_id":4}}
{"country": "Indonesia", "population": "245452739"}
{"index":{"_id":5}}
{"country": "Brazil", "population": "188078227"}
{"index":{"_id":6}}
{"country": "Pakistan", "population": "165803560"}
{"index":{"_id":7}}
{"country": "Bangladesh", "population": "147365352"}
{"index":{"_id":8}}
{"country": "Russia", "population": "142893540"}
{"index":{"_id":9}}
{"country": "Nigeria", "population": "131859731"}
{"index":{"_id":10}}
{"country": "Japan", "population": "127463611"}
```

Note that we will use _bulk upload in dev tools to upload the data.
Now, go to Kibana Dev Tools and execute following queries:

```
PUT /allcountries
{
   "mappings": {
      "_doc": {
         "properties": {
            "country": {"type": "keyword"},
            "population": {"type": "integer"}
         }
      }
   }
}

POST /allcountries/_doc/_bulk?refresh
{"index":{"_id":1}}
{"country": "China", "population": "1313973713"}
{"index":{"_id":2}}
{"country": "India", "population": "1095351995"}
{"index":{"_id":3}}
{"country": "United States", "population": "298444215"}
{"index":{"_id":4}}
{"country": "Indonesia", "population": "245452739"}
{"index":{"_id":5}}
{"country": "Brazil", "population": "188078227"}
{"index":{"_id":6}}
{"country": "Pakistan", "population": "165803560"}
{"index":{"_id":7}}
{"country": "Bangladesh", "population": "147365352"}
{"index":{"_id":8}}
{"country": "Russia", "population": "142893540"}
{"index":{"_id":9}}
{"country": "Nigeria", "population": "131859731"}
{"index":{"_id":10}}
{"country": "Japan", "population": "127463611"}
```
Next, let us create index allcountries. We have specified the country field type as **keyword**:

```json
PUT /allcountries
{
   "mappings": {
      "_doc": {
         "properties": {
            "country": {"type": "keyword"},
            "population": {"type": "integer"}
         }
      }
   }
}
```

**Note**: To work with region maps we need to specify the field type to be used with aggregation as type as keyword.
Once done, upload the data using _bulk command.

We will now create index pattern. Go to Kibana Management tab and select create index pattern.
Here are the fields displayed from allcountries index.

![Field Table]

**Getting Started with Region Maps**

We will now create the visualization using Region Maps. Go to Visualization and select Region Maps.
Once done select index as *allcountries* and proceed.

Select Aggregation Metrics and Bucket Metrics as shown below:

Here we have selected field as *country*, as I want to show the same on the world map.
Vector Map and Join Field for Region Map

For region maps we need to also select Option tabs as shown below:

The options tab has Layer Settings configuration which are required to plot the data on the world map.
A Vector Map has the following options:

Here we will select world countries as I have countries data.

The Join Field has following details:

In our index we have the country name, so we will select country name.
In Style settings you can choose the color to be displayed for the countries:

We will select Reds. We will not touch the rest of the details.

Now, click on Analyze button to see the details of the countries plotted on the world map as shown below:
**Self-hosted Vector Map and Join Field in Kibana**

You can also add your own Kibana settings for vector map and join field. To do that go to kibana.yml from the kibana config folder and add the following details:

```yaml
regionmap:
  includeElasticMapsService: false
layers:
  - name: "Countries Data"
    url: "http://localhost/kibana/worldcountries.geojson"
    attribution: "INRAP"
    fields:
      - name: "Country"
        description: "country names"
```

The vector map from options tab will have the above data populated instead of the default one. Please note the URL given has to be CORS enabled so that Kibana can download the same. The json file used should be in such a way that the coordinates are in continuation. For example:

https://vector.maps.elastic.co/blob/5659313586569216?elastic_tile_service_tos=agree

The options tab when region-map vector map details are self-hosted is shown below:
A gauge visualization tells how your metric considered on the data falls in the predefined range.
A goal visualization tells about your goal and how your metric on your data progresses towards the goal.

**Working with Gauge**

To start using Gauge, go to visualization and select Visualize tab from Kibana UI.
Click on Gauge and select the index you want to use.

We are going to work on *medicalvisits-26.01.2019* index.
Select the time range of February 2017

Now you can select the metric and bucket aggregation.
We have selected the metric aggregation as Count.

The bucket aggregation we have selected Terms and the field selected is Number_Home_Visits.

From Data options Tab, the options selected are shown below:
Gauge Type can be in the form of circle or arc. We have selected as arc and rest all others as the default values.

The predefined range we have added is shown here:

The colour selected is Green To Red.

Now, click on Analyze Button to see the visualization in the form of Gauge as shown below:
**Working with Goal**

Go to Visualize Tab and select Goal as shown below:

Select Goal and select the index.
Use *medicalvisits-26.01.2019* as the index.

Select the metric aggregation and bucket aggregation. **Metric Aggregation:**

We have selected Count as the metric aggregation.

---

**From a New Search, Select Index**

- countriesdata-28.12.2018
- test-28.12.2018
- kibana_sample_data_logs
- homemedicalvisits-26.01.2019
- cities
- museums
- mdata
- cdata
- filebeat-
- medicalvisits-26.01.2019
- allcountries
Bucket Aggregation

We have selected Terms as the bucket aggregation and field is Number_Home_Visits.

The options selected are as follows:
The Range selected is as follows:

Click on Analyze and you see the goal displayed as follows:
Canvas is yet another powerful feature in Kibana. Using canvas visualization, you can represent your data in different color combination, shapes, text, multipage setup etc.

We need data to show in the canvas. Now, let us load some sample data already available in Kibana.

**Loading Sample Data for Canvas Creation**

To get the sample data go to Kibana home page and click on Add sample data as shown below:
Click on Load a data set and a Kibana dashboard. It will take you to the screen as shown below:

Click on Add button for Sample eCommerce orders. It will take some time to load the sample data. Once done you will get an alert message showing “Sample eCommerce data loaded.”
Getting Started with Canvas Visualization

Now go to Canvas Visualization as shown below:

![Canvas Visualization](image)

Click on Canvas and it will display screen as shown below:

![Canvas Workpads](image)
We have eCommerce and Web Traffic sample data added. We can create new workpad or use the existing one. Here, we will select the existing one. Select eCommerce Revenue Tracking Workpad Name and it will display the screen as shown below:
Cloning an Existing Workpad in Canvas

We will clone the workpad so that we can make changes to it. To clone an existing workpad, click on the name of the workpad shown at the bottom left:

Click on the name and select clone option as shown below:
Click on the clone button and it will create a copy of the eCommerce Revenue Tracking workpad. You can find it as shown below:

In this section, let us understand how to use the workpad. If you see above workpad, there are 2 pages for it. So in canvas we can represent the data in multiple pages.
The page 2 display is as shown below:

Select Page 1 and click on the Total sales displayed on left side as shown below:
On the right side, you will get the data related to it:

Right now the default style used is green colour. We can change the colour here and check the display of same.
We have also changed the font and size for text settings as shown below:
Adding New Page to Workpad Inside Canvas

To add new page to the workpad, do as shown below:

Once the page is created as shown below:
Click on Add element and it will display all possible visualization as shown below:

We have added two elements Data table and Area Chart as shown below:

You can add more data elements to the same page or add more pages too.
In our previous chapters, we have seen how to create visualization in the form of vertical bar, horizontal bar, pie chart etc. In this chapter, let us learn how to combine them together in the form of Dashboard. A dashboard is collection of your visualizations created, so that you can take a look at it all together at a time.

**Getting Started with Dashboard**

To create Dashboard in Kibana, click on the Dashboard option available as shown below:
Now, click on Create new dashboard button as shown above. It will take us to the screen as shown below:

Observe that we do not have any dashboard created so far. There are options at the top where we can Save, Cancel, Add, Options, Share, Auto-refresh and also change the time to get the data on our dashboard. We will create a new dashboard, by clicking on the Add button shown above.

**Add Visualization to Dashboard**

When we click the Add button (top left corner), it displays us the visualization we created as shown below:
Select the visualization you want to add to your dashboard. We will select the first three visualizations as shown below:

This is how it is seen on the screen together:
Thus, as a user you are able to get the overall details about the data we have uploaded – country wise with fields country-name, regionname, area and population.

So now we know all the regions available, the max population country wise in descending order, the max area etc.

This is just the sample data visualization we uploaded, but in real world it becomes very easy to track the details of your business like for example you have a website which gets millions of hits monthly or daily, you want to keep a track on the sales done every day, hour, minute, seconds and if you have your ELK stack in place Kibana can show you your sales visualization right in front of your eyes every hour, minute, seconds as you want to see. It displays the real time data as it is happening in the real world.

Kibana, on the whole, plays a very important role in extracting the accurate details about your business transaction day wise, hourly or every minute, so the company knows how the progress is going on.
Save Dashboard

You can save your dashboard by using the save button at the top.

There is a title and description where you can enter the name of the dashboard and a short description which tells what the dashboard does. Now, click on Confirm Save to save the dashboard.

Changing Time Range for Dashboard

At present you can see the data shown is of **Last 15 minutes**. Please note this is a static data without any time field so the data displayed will not change. When you have the data connected to real time system changing the time, will also show the data reflecting.
By default, you will see Last 15 minutes as shown below:

Click on the Last 15 minutes and it will display you the time range which you can select as per your choice.

Observe that there are Quick, Relative, Absolute and Recent options. The following screenshot shows the details for Quick option:

Now, click on Relative to see the option available:

Here you can specify the From and To date in minutes, hours, seconds, months, years ago.
The Absolute option has the following details:

You can see the calendar option and can select a date range.

The recent option will give back the Last 15 minutes option and also other option which you have selected recently. Choosing the time range will update the data coming within that time range.

**Using Search and Filter in Dashboard**

We can also use search and filter on the dashboard. In search suppose if we want to get the details of a particular region, we can add a search as shown below:

In the above search, we have used the field Region and want to display the details of region: OCEANIA.
We get following results:

Looking at the above data we can say that in OCEANIA region, Australia has the max population and Area.
Similarly, we can add a filter as shown below:

![Add filter](image1)

Next, click on Add a filter button and it will display the details of the field available in your index as shown below:

![Add filter](image2)
Choose the field you want to filter on. I will use Region field to get the details of ASIA region as shown below:

Save the filter and you should see the filter as follows:

The data will now be shown as per the filter added:

You can also add more filters as shown below:
You can disable the filter by clicking on the disable checkbox as shown below.

![Image showing how to disable a filter]

You can activate the filter by clicking on the same checkbox to activate it. Observe that there is a delete button to delete the filter. Edit button to edit the filter or change the filter options.

For the visualization displayed, you will notice three dots as shown below:
Click on it and it will display options as shown below:
**Inspect and Fullscreen**

Click on Inspect and it gives the details of the region in tabular format as shown below:

<table>
<thead>
<tr>
<th>Region.keyword: Descending</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUB-SAHARAN AFRICA</td>
<td>51</td>
</tr>
<tr>
<td>LATIN AMER. &amp; CARIB</td>
<td>45</td>
</tr>
<tr>
<td>ASIA (EX. NEAR EAST)</td>
<td>28</td>
</tr>
<tr>
<td>WESTERN EUROPE</td>
<td>28</td>
</tr>
<tr>
<td>OCEANIA</td>
<td>21</td>
</tr>
<tr>
<td>NEAR EAST</td>
<td>16</td>
</tr>
<tr>
<td>C.W. OF IND. STATES</td>
<td>12</td>
</tr>
<tr>
<td>EASTERN EUROPE</td>
<td>12</td>
</tr>
<tr>
<td>NORTHERN AFRICA</td>
<td>6</td>
</tr>
<tr>
<td>NORTHERN AMERICA</td>
<td>5</td>
</tr>
</tbody>
</table>

There is an option to download the visualization in CSV format in-case you want to see it in excel sheet.
The next option full screen will get the visualization in a full screen mode as shown below:

![Full Screen Visualization](image)

You can use the same button to exit the full screen mode.

**Sharing Dashboard**

We can share the dashboard using the share button. On click of share button, you will get display as follows:

![Sharing Options](image)
You can also use embed code to show the dashboard on your site or use permalinks which will be a link to share with others.

The url will be as follows:

```
http://localhost:5601/goto/519c1a088d5d0f8703937d754923b84b
```
Timelion, also called as timeline is yet another visualization tool which is mainly used for time based data analysis. To work with timeline, we need to use simple expression language which will help us connect to the index and also perform calculations on the data to get the results we need.

**Where can we use Timelion?**

Timelion is used when you want to compare time related data. For example, you have a site, and you get your views daily. You want to analyse the data wherein you want to compare the current week data with previous week, i.e. Monday-Monday, Tuesday - Tuesday and so on how the views are differing and also the traffic.

**Getting Started with Timelion**

To start working with Timelion, click on Timelion as shown below:
Timelion by default shows the timeline of all indexes as shown below:

![Timelion Default Timeline](image)

Timelion works with expression syntax.

**Note:** `es(*) =>` means all indexes.

To get the details of function available to be used with Timelion, simply click on the textarea as shown below:

![Function Reference](image)

It gives you the list of function to be used with the expression syntax.

Once you start with Timelion, it displays a welcome message as shown below. The highlighted section i.e. Jump to the function reference, gives the details of all the functions available to be used with timelion.
Timelion Welcome Message

The Timelion welcome message is as shown below:

![Welcome to Timelion!](image)

Click on the next button and it will walk you through its basic functionality and usage. Now when you click Next, you can see the following details:

![Good news, Elasticsearch is configured correctly!](image)
Querying the Elasticsearch datasource

Now that we've validated that you have a working Elasticsearch datasource, you can start submitting queries. For starters, enter `.es()` in the input bar and hit enter.

This says _hey Elasticsearch, find everything in my default index_. If you want to find a subset, you could enter something like `.es(html)` to count events that match `html` or `.es(user=bob AND bytes>128)` to find events that contain `bob` in the `user` field and have a `bytes` field that is greater than 100. Note that this query is enclosed in single quotes—that's because it contains spaces. You can enter any _Lucene query string_ as the first argument to the `.es()` function.

Passing arguments

Timelion has a number of shortcuts that make it easy to do common things. One is that for simple arguments that don't contain spaces or special characters, you don't need to use quotes. Many functions also have defaults. For example, `.es()` and `.ag()` do the same thing. Arguments also have names, so you don't have to specify them in a specific order. For example, you can enter `.es(index="logstash-\*", q="\*\")` to tell the Elasticsearch datasource _use * as the q (query) for the logstash-* index_.

Beyond count

Counting events is all well and good, but the Elasticsearch datasource also supports any _Elasticsearch metric aggregation_ that returns a single value. Some of the most useful are _min max avg sum and cardinality_. Let's say you want a unique count of the _src_ip_ field. Simply use the _cardinality_ metric:.es(\*, metric=cardinality:src_ip). To get the average of the _bytes_ field, you can use the _avg_ metric:.es(metric=avg:bytes).

Expressing yourself with expressions

Every expression starts with a datasource function. From there, you can append new functions to the datasource to transform and augment it.

By the way, from here on out you probably know more about your data than we do. Feel free to replace the sample queries with something more meaningful!

We're going to experiment, so click _Add_ in the Kibana toolbar to add another chart or three. Then, select a chart, copy one of the following expressions, paste it into the input bar, and hit enter. Rinse, repeat to try out the other expressions.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Double the fun. Two expressions on the same chart.</th>
<th>Custom styling. Colorizes the first series red and uses 1 pixel wide bars for the second series.</th>
<th>Named arguments. Forget trying to remember what order you need to specify arguments in; use named arguments to make the expression easier to read and write.</th>
<th>Grouped expressions. You can also chain groups of expressions to functions. Here, both series are shown as points instead of lines.</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>.es(\*)</code></td>
<td><code>.es(\*)</code>.color(#e66), <code>.es(\*)</code>.bars(1)</td>
<td><code>.es(\*)</code>.color(#f66).lines(#f1113), <code>.es(\*)</code>.bars(1).points(radius=3, weight=1)</td>
<td><code>.es(\*), .es(\*)</code>.points()</td>
<td></td>
</tr>
</tbody>
</table>

Timelion provides _additional view transformation functions_ you can use to customize the appearance of your charts. For the complete list, see the _Function reference_.

---

147
Transforming your data: the real fun begins!

Now that you've mastered the basics, it's time to unleash the power of Timelion. Let's figure out what percentage some subset of our data represents of the whole, over time. For example, what percentage of our web traffic comes from the US?

First, we need to find all events that contain US: `.es('US')`.

Next, we want to calculate the ratio of US events to the whole. To divide `US` by everything, we can use the `divide` function: `.es('US').divide(es())`.

Not bad, but this gives us a number between 0 and 1. To convert it to a percentage, simply multiply by 100: `.es('US').divide(es()).multiply(100)`.

Now we know what percentage of our traffic comes from the US, and can see how it has changed over time! Timelion has a number of built-in arithmetic functions, such as add, subtract, multiply, and divide. Many of these can take a series or a number. There are also other useful data transformation functions, such as moving average, abs, and derivative.

Now that you're familiar with the syntax, refer to the Function reference to see how to use all of the available Timelion functions. You can view the reference at any time by clicking Docs in the Kibana toolbar. To get back to this tutorial, click the Tutorial link at the top of the reference.

Timelion Function Reference

Click on Help button to get the details of the function reference available for Timelion:
Timelion Configuration

The settings for timelion is done in Kibana Management -> Advanced Settings.

Click on Advanced Settings and select Timelion from Category.
Once Timelion is selected it will display all the necessary fields required for timelion configuration.

<table>
<thead>
<tr>
<th>Default columns</th>
<th>timelion.default_columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of columns on a timelion sheet by default</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Default rows</th>
<th>timelion.default_rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of rows on a timelion sheet by default</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Default Index</th>
<th>timelion.es.default_index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default elasticsearch index to search with .es()</td>
<td>_all</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time field</th>
<th>timelion.es.timefield</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default field containing a timestamp when using .es()</td>
<td>@timestamp</td>
</tr>
</tbody>
</table>

Graphite URL
(experimental) The URL of your graphite host

```plaintext
https://www.hostedgraphite.com/UID/ACCE
```

In the following fields you can change the default index and the timefield to be used on the index:

<table>
<thead>
<tr>
<th>Default Index</th>
<th>timelion.es.default_index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default elasticsearch index to search with .es()</td>
<td>_all</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time field</th>
<th>timelion.es.timefield</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default field containing a timestamp when using .es()</td>
<td>@timestamp</td>
</tr>
</tbody>
</table>

The default one is _all and timefield is @timestamp. We would leave it as it is and change the index and timefield in the timelion itself.
Using Timelion to Visualize Data

We are going to use index: `medicalvisits-26.01.2019`. The following is the data displayed from timelion for 1st Jan 2017 to 31st Dec 2017:

The expression used for above visualization is as follows:

```
.es(index=medicalvisits-26.01.2019,timefield=Visiting_Date).bars()
```

We have used the index `medicalvisits-26.01.2019` and timefield on that index is `Visiting_Date` and used bars function.

In the following we have analyzed 2 cities for the month of Jan 2017, day wise.
The expression used is:

\[ \text{es(index=medicalvisits-26.01.2019, timefield=Visiting\_Date, q=City:Sabadell).label(Sabadell), es(index=medicalvisits-26.01.2019, timefield=Visiting\_Date, q=City:Terrassa).label(Terrassa)} \]

The timeline comparison for 2 days is shown here:

Expression:

\[ \text{es(index=medicalvisits-26.01.2019, timefield=Visiting\_Date).label("August 2nd 2018"), es(index=medicalvisits-26.01.2019, timefield=Visiting\_Date, offset=-1d).label("August 1st 2018")} \]
Here we have used offset and given a difference of 1 day. We have selected the current date as 2nd August 2018. So it gives data difference for 2nd Aug 2018 and 1st Aug 2018.

The list of top 5 cities data for the month of Jan 2017 is shown below. The expression that we have used here is given below:

```
.es(index=medicalvisits-26.01.2019,timefield=Visiting_Date,split=City.keyword:5)
```

We have used split and given the field name as city and the since we need top five cities from the index we have given it as `split=City.keyword:5`

It gives the count of each city and lists their names as shown in the graph plotted.
We can use Dev Tools to upload data in Elasticsearch, without using Logstash. We can post, put, delete, search the data we want in Kibana using Dev Tools.

To create new index in Kibana we can use following command in dev tools:

**Create Index USING PUT**

The command to create index is as shown here:

```
PUT /usersdata?pretty
```

Once you execute this, an empty index userdata is created.

We are done with the index creation. Now will add the data in the index:

**Add Data to Index Using PUT**

You can add data to an index as follows:

```
PUT /usersdata/user/1
{
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "createdOn": "2018-01-10"
}
```
We will add one more record in usersdata index:

```
PUT /usersdata?pretty

GET /_cat/indices

PUT /usersdata/user/1
{
  "name": "Leanne Graham",
  "username": "Bret",
  "email": "Sincere@april.biz",
  "createdOn": "2018-01-10"
}
```

```
1 - {
  "_index": "usersdata",
  "_type": "user",
  "_id": "1",
  "_version": 1,
  "result": "created",
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  },
  "_seq_no": 0,
  "_primary_term": 1
}
```

```
PUT /usersdata/user/2
{
  "name": "Ervin Howell",
  "username": "Antonette",
  "email": "Shanna@melissa.tv",
  "createdOn": "2018-01-15"
}
```

```
1 - {
  "_index": "usersdata",
  "_type": "user",
  "_id": "2",
  "_version": 1,
  "result": "created",
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  },
  "_seq_no": 0,
  "_primary_term": 1
}
```

So we have 2 records in usersdata index.
Fetch Data from Index Using GET

We can get the details of record 1 as follows:

```
PUT /usersdata/pretty
GET /_cat/indices

PUT /usersdata/user/1
{
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "createdOn": "2018-01-10"
}

PUT /usersdata/user/2
{
    "name": "Ervin Howell",
    "username": "Antonette",
    "email": "Shanna@melissa.tv",
    "createdOn": "2018-01-15"
}

GET /usersdata/user/1
```

You can get all records as follows:

```
PUT /usersdata/pretty
GET /_cat/indices

PUT /usersdata/user/1
{
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "createdOn": "2018-01-10"
}

PUT /usersdata/user/2
{
    "name": "Ervin Howell",
    "username": "Antonette",
    "email": "Shanna@melissa.tv",
    "createdOn": "2018-01-15"
}

GET /usersdata/_search
```

Thus, we can get all the records from usersdata as shown above.
Update data in Index using PUT

To update the record, you can do as follows:

We have changed the name from “Ervin Howell” to “Clementine Bauch”. Now we can get all records from the index and see the updated record as follows:
Delete data from index using DELETE

You can delete the record as shown here:

```
PUT /usersdata/user/2
{
"name": "Ervin Howell",
"username": "Antonette",
"email": "Shanna@melissa.tv",
"createdOn": "2018-01-15"
}

GET /usersdata/user/1

GET /usersdata/_search

PUT /usersdata/user/2
{
"name": "Clementine Bauch",
"username": "Bret",
"email": "Sincere@april.biz",
"createdOn": "2018-01-10"
}

DELETE /usersdata/user/2
```

Now if you see the total records we will have only one record:

We can delete the index created as follows:
Now if you check the indices available we will not have `usersdata` index in it as deleted the index.
Kibana Monitoring gives the details about the performance of ELK stack. We can get the details of memory used, response time etc.

**Monitoring Details**

To get monitoring details in Kibana, click on the monitoring tab as shown below:

Since we are using the monitoring for the first time, we need to keep it ON. For this, click the button **Turn on monitoring** as shown above. Here are the details displayed for Elasticsearch:
It gives the version of elasticsearch, disk available, indices added to elasticsearch, disk usage etc.

The monitoring details for Kibana are shown here:

It gives the Requests and max response time for the request and also the instances running and memory usage.
Reports can be easily created by using the Share button available in Kibana UI.

Reports in Kibana are available in the following two forms:

- Permalinks
- CSV Report

**Report as Permalinks**

When performing visualization, you can share the same as follows:

Use the share button to share the visualization with others as Embed Code or Permalinks.
In case of Embed code you get the following options:

- Snapshot
- Saved object
- Short URL

You can generate the iframe code as short url or long url for snapshot or saved object. Snapshot will not give the recent data and user will be able to see the data saved when the link was shared. Any changes done later will not be reflected.

In case of saved object, you will get the recent changes done to that visualization.

Snapshot IFrame code for long url:

```html
<iframe src="http://localhost:5601/app/kibana#/visualize/edit/87afcb60-165f-11e9-aaf1-3524d1f04792?embed=true&g=()&a=()&filters:=[],linked:!f,query:(language:.lucene,query:''),uiState:(),vis:(aggs:![enabled:t,id:'1',params:(field:Area),schema:metric,type:max],enabled:t,id:'2',params:(field:Country.keyword,missingBucket:f,missingBucketLabel:Missing,order:desc,orderBy:'1',otherBucketLabel:Other,size:10),schema:segment,type:terms)),params:(addLegend:!t,addTimeMarker:!t,categoryAxes:![id:CategoryAxis-1,labels:show:t,truncate:100],position:bottom,scale:(type:linear),show:t,style:()],title:(),type:category),grid:(categoryLines:!f,style:(color:##eee)),legendPosition:right,seriesParams:![data:(id:'1',label:'Max+Area'),drawLinesBetweenPoints:!t,mode:stacked,show:true,showCircles:t,type:histogram,valueAxis:ValueAxis-1)),times:!t,type:histogram,valueAxes:![id:ValueAxis-1,labels:(filter:!f,rotate:0,show:t,truncate:100)],name:LeftAxis-1,position:left,scale:(mode:normal,type:linear),show:t,style:(),title:(text:'Max+Area'),type:Value),title:'countrywise_maxarea+',type:histogram)"
height="600" width="800"></iframe>
```

Snapshot Iframe code for short url:

```html
<iframe src="http://localhost:5601/goto/f0a6c852daedcb6b4fa7ace8c2ff6c4?embed=true" height="600" width="800"></iframe>
```
You can embed this code on your site or whenever required. You can also get the embed code for saved object by selecting the saved object radio button.

For Permalinks you get following options:

- **Snapshot**
- **Saved object**
- **Short URL**

As snapshot and shot url.

**With Short url:**

```
http://localhost:5601/goto/f0a6c852daedcb6b4fa74ccee82ff6c4
```

**With Short url off, the link looks as below:**

```
http://localhost:5601/app/kibana#/visualize/edit/87afcb60-165f-11e9-aaf1-3524d1f04792?_g=()&a=(filters:(),linked:1,query:(language:lucene,query:''),uiState:()),vis:(aggs:((enabled:1,id:'1',params:(field:Area),schema:metric,type:max),(enabled:1,id:'2',params:(field:Country.keyword,missingBucket:!f,missingBucketLabel:Missing,order:desc,orderBy:'1',otherBucket:!f,otherBucketLabel:Other,size:10),schema:segment,type:terms)),params:(addLegend:1,addTimeMarker:!f,addTooltip:1,categoryAxes:((id:CategoryAxis-1,labels:(show:1,truncate:100),position:bottom,scale:(type:linear),show:1,style:(),title:(),type:category)),grid:(categoryLines:1,style:(color:%23eee)),legendPosition:right,seriesParams:((data:1,label:'Max%20Area'),drawlinesBetweenPoints:1,mode:stacked,show:1,showCircles:1,type:histogram,valueAxis:ValueAxis-1)),times:1,type:histogram,valueAxes:((id:ValueAxis-1,labels:(filter:!f,rotate:0,show:1,truncate:100),name:LeftAxis-1,position:left,scale:(mode:normal,type:linear),show:1,style:(),title:(text:'Max%20Area'),type:value)),title:'countrywise_maxarea%',type:histogram))
```

When you hit the above link in the browser, you will get the same visualization as shown above. The above links are hosted locally, so it will not work when used outside the local environment.
You can get CSV Report in Kibana where there is data, which is mostly in the Discover tab.

Go to Discover tab and take any index you want the data for. Here we have taken the index: `countriesdata-26.12.2018`. Here is the data displayed from the index:

You can create tabular data from above data as shown below:
We have selected the fields from Available fields and the data seen earlier is converted into tabular format.

You can get above data in CSV report as shown below:

The share button has option for CSV report and permalinks. You can click on CSV Report and download the same.

Please note to get the CSV Reports you need to save your data.
Confirm Save and click on Share button and CSV Reports. You will get following display:

Click on Generate CSV to get your report. Once done, it will instruct you to go the management tab.

Go to Management Tab -> Reporting

It displays the report name, created at, status and actions. You can click on the download button as highlighted above and get your csv report.
The CSV file we just downloaded is as shown here:

<table>
<thead>
<tr>
<th>Country</th>
<th>Area</th>
<th>Region</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bolivia</td>
<td>10.98,380</td>
<td>LATIN AMER. &amp; CARIB</td>
<td>89,89,046</td>
</tr>
<tr>
<td>Botswana</td>
<td>6,00,370</td>
<td>SUB-SAHARAN AFRICA</td>
<td>16,35,933</td>
</tr>
<tr>
<td>Burma</td>
<td>6,76,500</td>
<td>ASIA (EX. NEAR EAST)</td>
<td>473,82,633</td>
</tr>
<tr>
<td>China</td>
<td>95,96,900</td>
<td>ASIA (EX. NEAR EAST)</td>
<td>1,313,973,713</td>
</tr>
<tr>
<td>Costa Rica</td>
<td>51,100</td>
<td>LATIN AMER. &amp; CARIB</td>
<td>40,75,282</td>
</tr>
<tr>
<td>East Timor</td>
<td>15,007</td>
<td>ASIA (EX. NEAR EAST)</td>
<td>10,62,777</td>
</tr>
<tr>
<td>Equatorial Guinea</td>
<td>28,051</td>
<td>SUB-SAHARAN AFRICA</td>
<td>5,40,109</td>
</tr>
<tr>
<td>Gaza Strip</td>
<td>800</td>
<td>NEAR EAST</td>
<td>14,78,757</td>
</tr>
<tr>
<td>Gibraltar</td>
<td>7</td>
<td>WESTERN EUROPE</td>
<td>27,928</td>
</tr>
<tr>
<td>India</td>
<td>32,87,990</td>
<td>ASIA (EX. NEAR EAST)</td>
<td>1,055,351,965</td>
</tr>
<tr>
<td>Ireland</td>
<td>70,280</td>
<td>WESTERN EUROPE</td>
<td>40,62,235</td>
</tr>
<tr>
<td>Kazakhstan</td>
<td>27,17,800</td>
<td>C.W. OF IND. STATES</td>
<td>152,18,244</td>
</tr>
<tr>
<td>Kiribati</td>
<td>811</td>
<td>OCEANIA</td>
<td>1,05,412</td>
</tr>
<tr>
<td>Kyrgyzstan</td>
<td>1,98,800</td>
<td>C.W. OF IND. STATES</td>
<td>59,18,208</td>
</tr>
<tr>
<td>Latvia</td>
<td>64,589</td>
<td>BALTICS</td>
<td>22,74,735</td>
</tr>
<tr>
<td>Lithuania</td>
<td>65,200</td>
<td>BALTICS</td>
<td>35,85,906</td>
</tr>
<tr>
<td>Mali</td>
<td>12,40,000</td>
<td>SUB-SAHARAN AFRICA</td>
<td>117,10,829</td>
</tr>
<tr>
<td>Netherlands</td>
<td>41,526</td>
<td>WESTERN EUROPE</td>
<td>164,51,461</td>
</tr>
<tr>
<td>Sao Tome &amp; Principe</td>
<td>1,001</td>
<td>SUB-SAHARAN AFRICA</td>
<td>1,93,413</td>
</tr>
<tr>
<td>Seychelles</td>
<td>455</td>
<td>SUB-SAHARAN AFRICA</td>
<td>81,541</td>
</tr>
<tr>
<td>Switzerland</td>
<td>41,290</td>
<td>WESTERN EUROPE</td>
<td>75,23,894</td>
</tr>
<tr>
<td>Taiwan</td>
<td>35,980</td>
<td>ASIA (EX. NEAR EAST)</td>
<td>210,16,087</td>
</tr>
<tr>
<td>United Arab Emirates</td>
<td>82,880</td>
<td>NEAR EAST</td>
<td>26,02,713</td>
</tr>
<tr>
<td>Zimbabwe</td>
<td>3,90,380</td>
<td>SUB-SAHARAN AFRICA</td>
<td>122,36,805</td>
</tr>
</tbody>
</table>