kibana

tutorialspoint

SIMPLY EASY LEARNING

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

Table of Contents

About the Tutorial ii

Audience ii

Prerequisites ii

Copyright & Disclaimer ii

Table of Contents iii

1. KIBANA – OVERVIEW 1

What is ELK Stack? 1

Features of Kibana 2

Advantages of Kibana 3

Disadvantages of Kibana 3

2. KIBANA – ENVIRONMENT SETUP 4

Elasticsearch Installation 4

Logstash Installation 6

Kibana Installation 6

3. KIBANA— INTRODUCTION TO ELK STACK 11

Load CSV data from Logstash to Elasticsearch 11

Use Data from Elasticsearch in Kibana 15

4. KIBANA — LOADING SAMPLE DATA 17

Using Logstash upload for data having fields in Elasticsearch 17

Using Dev Tools to Upload Bulk Data 20

Search for a Record in *todo* Index 25

5. KIBANA — MANAGEMENT 27

Create Index Pattern Without Time Filter field 27

Create Index Pattern with Time filter field 29

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 53

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**

Getting Started with Timelion 144

Timelion Configuration 149

Using Timelion to Visualize Data 151

18. KIBANA — DEV TOOLS 154

Create Index USING PUT 154

Add Data to Index Using PUT 154

Fetch Data from Index Using GET 156

Update data in Index using PUT 157

Delete data from index using DELETE 158

19. KIBANA — MONITORING 160

Monitoring Details 160

20. KIBANA — CREATE REPORTS USING KIBANA 162

Report as Permalinks 162

CSV Report 165

1. Kibana – Overview

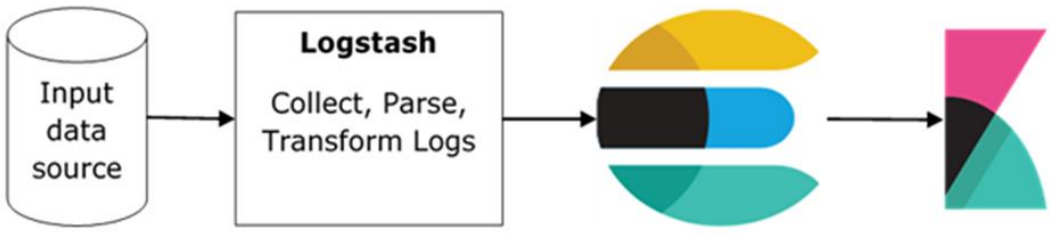
Kibana is an open source browser based visualization tool mainly used to analyse large volume of logs in the form of line graph, bar graph, pie charts , heat maps, region maps, coordinate maps, gauge, goals, timelion etc. The visualization makes it easy to predict or to see the changes in trends of errors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

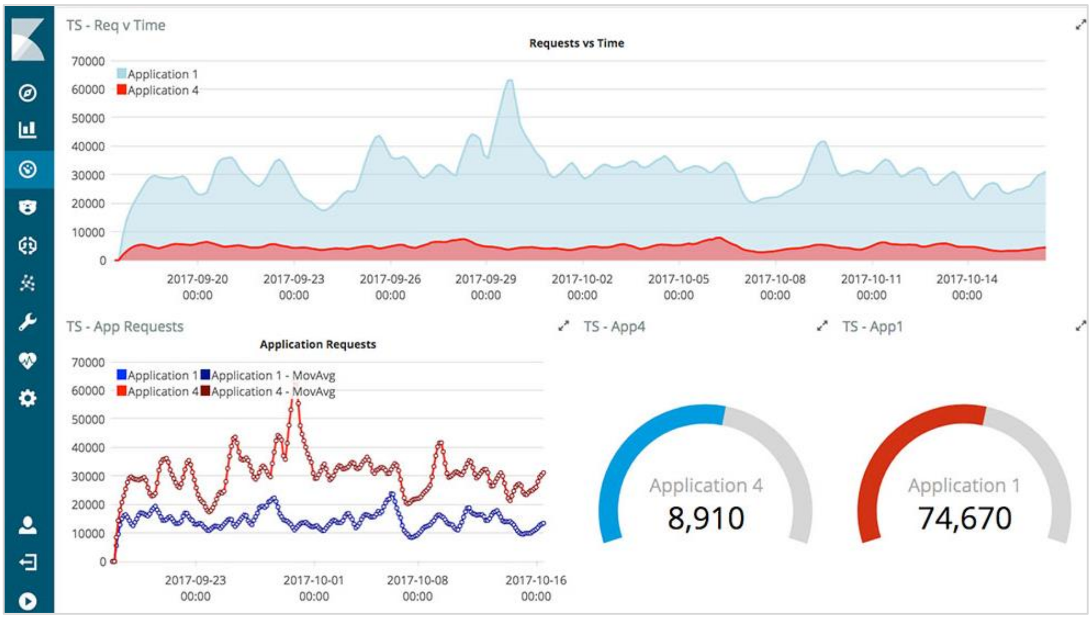
Kibana is a visualization tool, which accesses the logs from Elasticsearch and is able to display to the user in the form of line graph, bar graph, pie charts etc.

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



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.

2. Kibana – Environment Setup

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

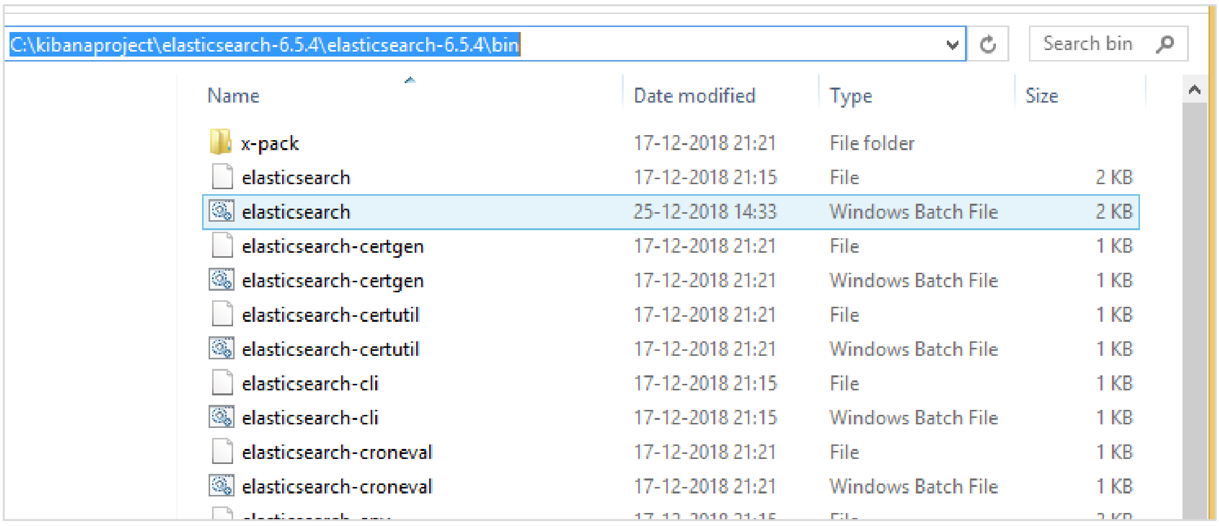
For Windows

```
> 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

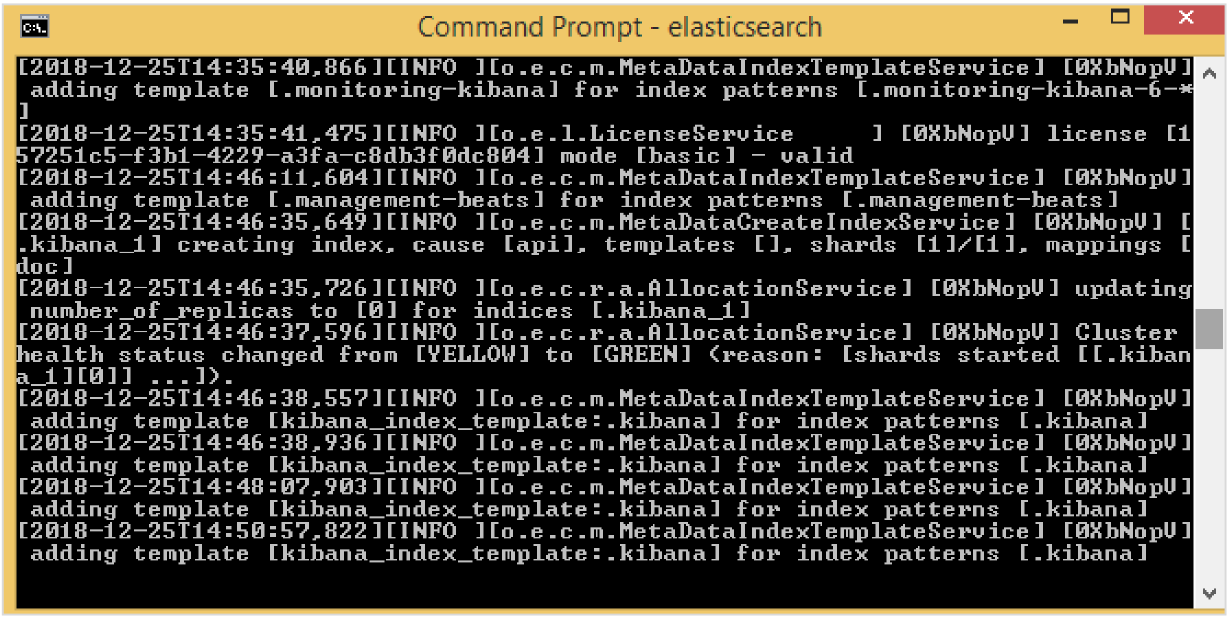
```
$ cd kibanaproject/elasticsearch-6.5.4/elasticsearch-6.5.4/bin

$ elasticsearch
```



C:\kibanaproject\elasticsearch-6.5.4\elasticsearch-6.5.4\bin

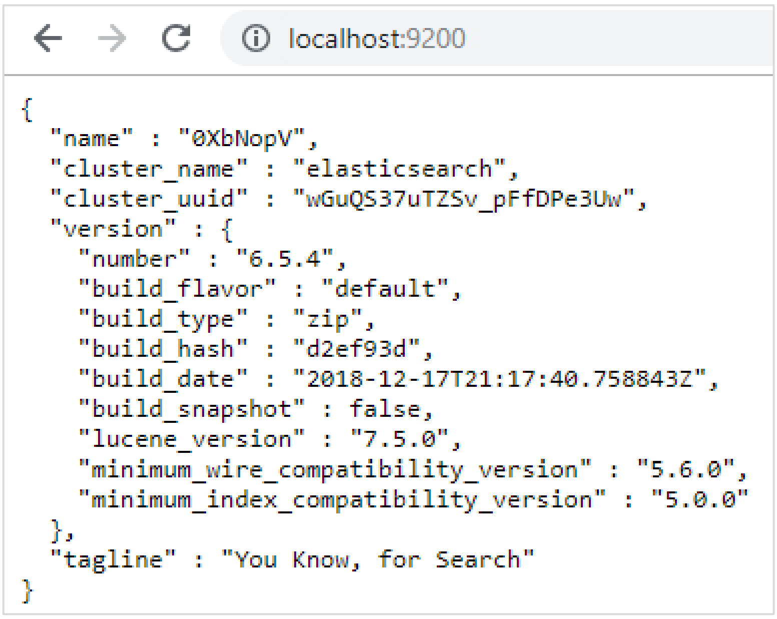
Name	Date modified	Type	Size
x-pack	17-12-2018 21:21	File folder	
elasticsearch	17-12-2018 21:15	File	2 KB
elasticsearch	25-12-2018 14:33	Windows Batch File	2 KB
elasticsearch-certgen	17-12-2018 21:21	File	1 KB
elasticsearch-certgen	17-12-2018 21:21	Windows Batch File	1 KB
elasticsearch-certutil	17-12-2018 21:21	File	1 KB
elasticsearch-certutil	17-12-2018 21:21	Windows Batch File	1 KB
elasticsearch-cli	17-12-2018 21:15	File	1 KB
elasticsearch-cli	17-12-2018 21:15	Windows Batch File	1 KB
elasticsearch-croneval	17-12-2018 21:21	File	1 KB
elasticsearch-croneval	17-12-2018 21:21	Windows Batch File	1 KB

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:



Command Prompt - elasticsearch

```
[2018-12-25T14:35:40,866][INFO ][o.e.c.m.MetaDataIndexTemplateService] [0XbNopV]
 adding template [.monitoring-kibana] for index patterns [.monitoring-kibana-6-*
]
[2018-12-25T14:35:41,475][INFO ][o.e.l.LicenseService      ] [0XbNopV] license [1
57251c5-f3b1-4229-a3fa-c8db3f0dc804] mode [basic] - valid
[2018-12-25T14:46:11,604][INFO ][o.e.c.m.MetaDataIndexTemplateService] [0XbNopV]
 adding template [.management-beats] for index patterns [.management-beats]
[2018-12-25T14:46:35,649][INFO ][o.e.c.m.MetaDataCreateIndexService] [0XbNopV] [
.kibana_1] creating index, cause [api], templates [], shards [1]/[1], mappings [
doc]
[2018-12-25T14:46:35,726][INFO ][o.e.c.r.a.AllocationService] [0XbNopV] updating
 number_of_replicas to [0] for indices [.kibana_1]
[2018-12-25T14:46:37,596][INFO ][o.e.c.r.a.AllocationService] [0XbNopV] Cluster
health status changed from [YELLOW] to [GREEN] (reason: [shards started [[.kiban
a_1][0]] ...]).
[2018-12-25T14:46:38,557][INFO ][o.e.c.m.MetaDataIndexTemplateService] [0XbNopV]
 adding template [kibana_index_template:.kibana] for index patterns [.kibana]
[2018-12-25T14:46:38,936][INFO ][o.e.c.m.MetaDataIndexTemplateService] [0XbNopV]
 adding template [kibana_index_template:.kibana] for index patterns [.kibana]
[2018-12-25T14:48:07,903][INFO ][o.e.c.m.MetaDataIndexTemplateService] [0XbNopV]
 adding template [kibana_index_template:.kibana] for index patterns [.kibana]
[2018-12-25T14:50:57,822][INFO ][o.e.c.m.MetaDataIndexTemplateService] [0XbNopV]
 adding template [kibana_index_template:.kibana] for index patterns [.kibana]
```



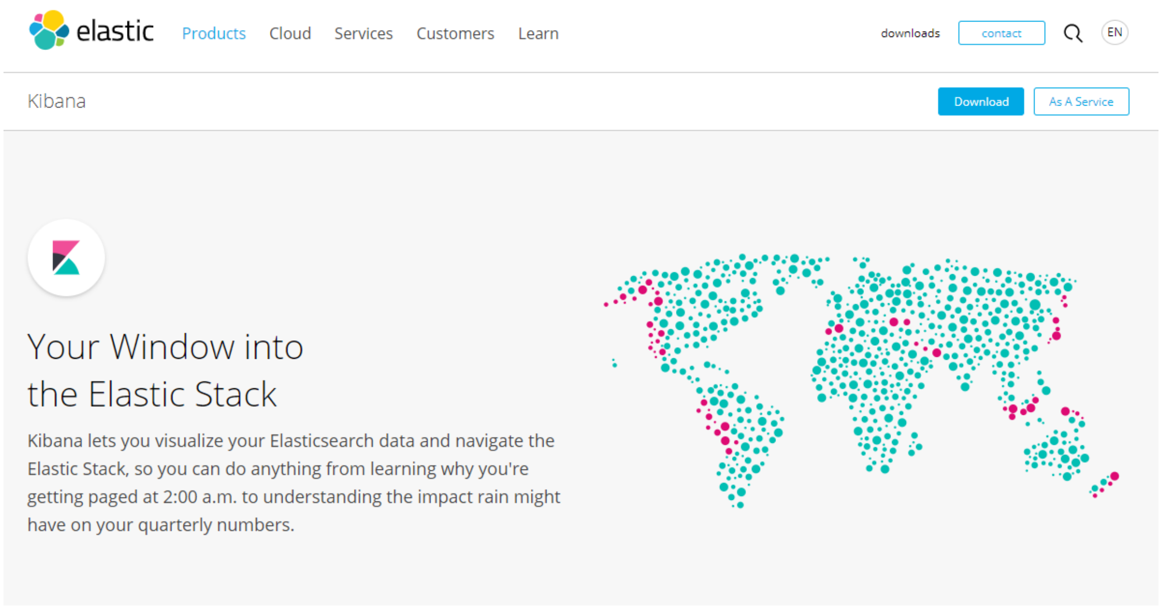
```
{
  "name" : "0XbNopV",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "wGuQS37uTZSv_pFfDPe3Uw",
  "version" : {
    "number" : "6.5.4",
    "build_flavor" : "default",
    "build_type" : "zip",
    "build_hash" : "d2ef93d",
    "build_date" : "2018-12-17T21:17:40.758843Z",
    "build_snapshot" : false,
    "lucene_version" : "7.5.0",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

Logstash Installation

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

Kibana Installation

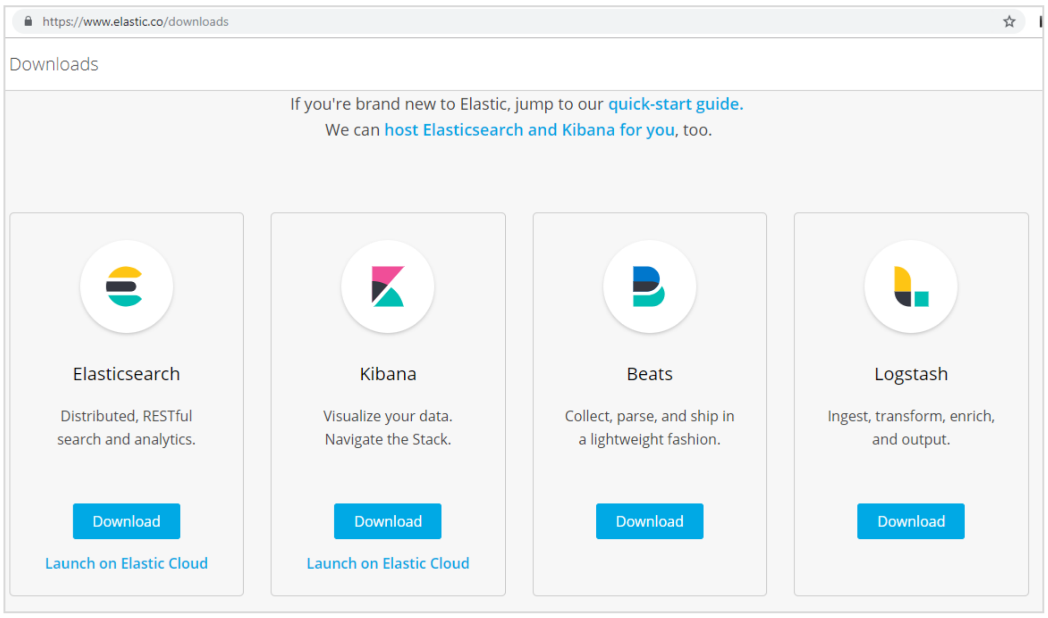
Go to the official Kibana site:https://www.elastic.co/products/kibana



Your Window into the Elastic Stack

Kibana lets you visualize your Elasticsearch data and navigate the Elastic Stack, so you can do anything from learning why you're getting paged at 2:00 a.m. to understanding the impact rain might have on your quarterly numbers.

Click the *downloads* link on the top right corner and it will display screen as follows:



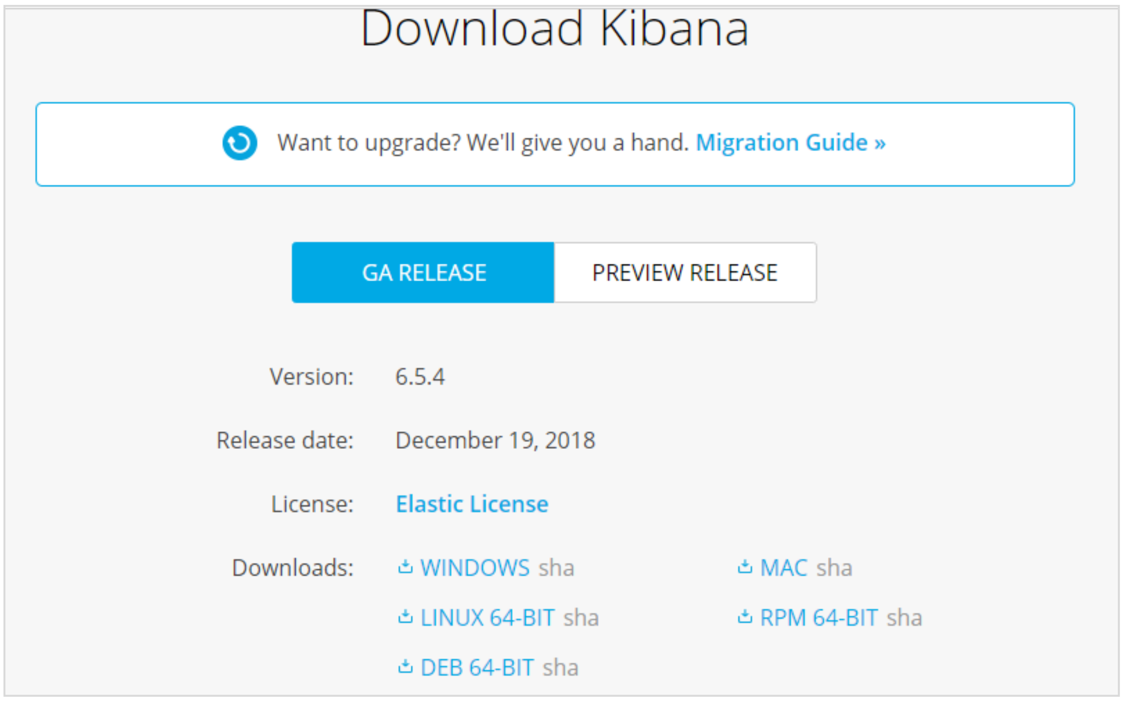
https://www.elastic.co/downloads

Downloads

If you're brand new to Elastic, jump to our quick-start guide.
We can host Elasticsearch and Kibana for you, too.

Elasticsearch
Distributed, RESTful search and analytics.
Download
Launch on Elastic Cloud
Kibana
Visualize your data. Navigate the Stack.
Download
Launch on Elastic Cloud
Beats
Collect, parse, and ship in a lightweight fashion.
Download
Logstash
Ingest, transform, enrich, and output.
Download

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

Want to upgrade? We'll give you a hand. Migration Guide »

Version: 6.5.4

Release date: December 19, 2018

License: Elastic License

Downloads: WINDOWS sha, MAC sha, LINUX 64-BIT sha, RPM 64-BIT sha, DEB 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.

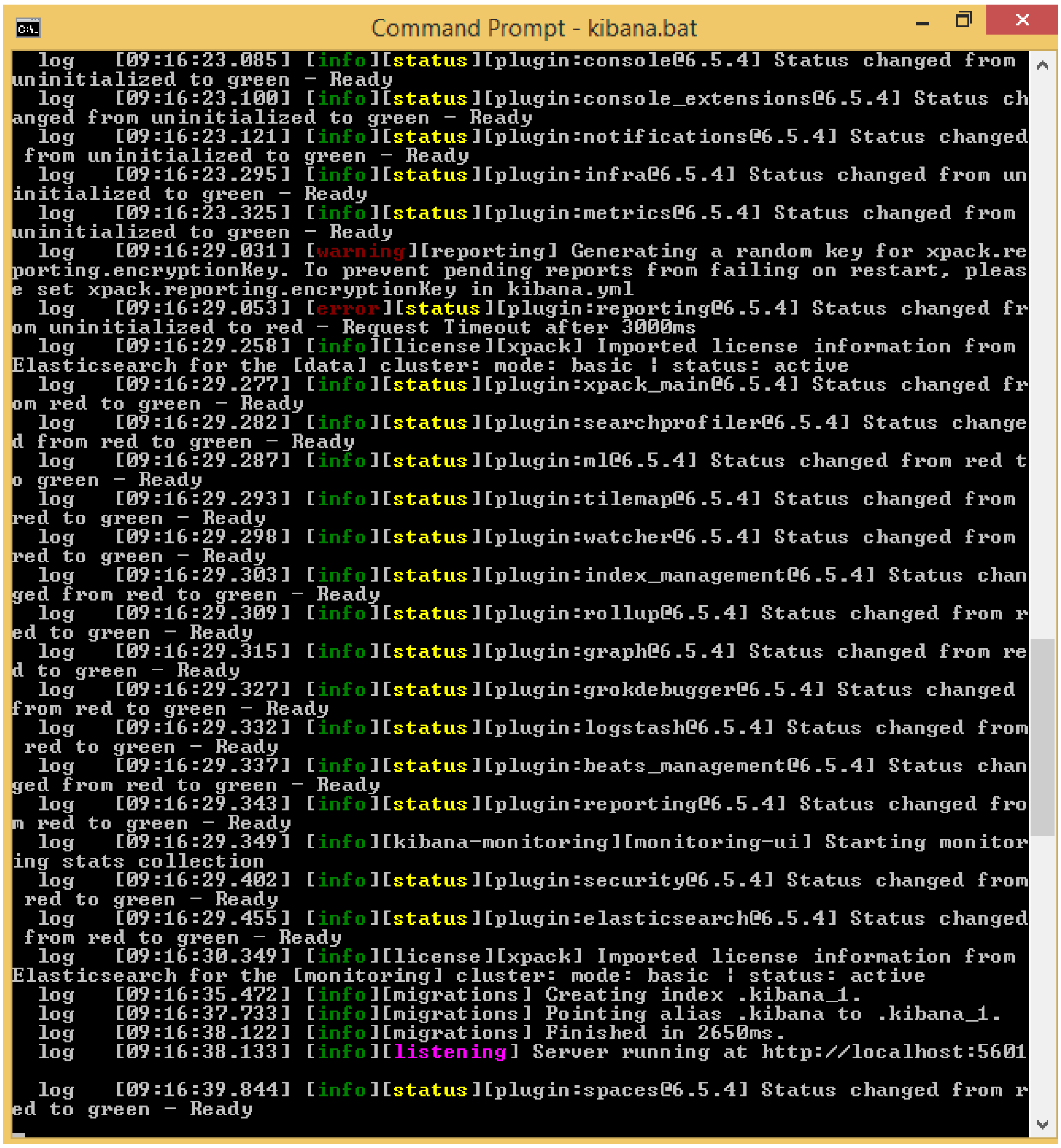
For Windows

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

> kibana
```

For Linux

```
$ cd kibanaproject/kibana-6.5.4/kibana-6.5.4/bin

$ kibana
```

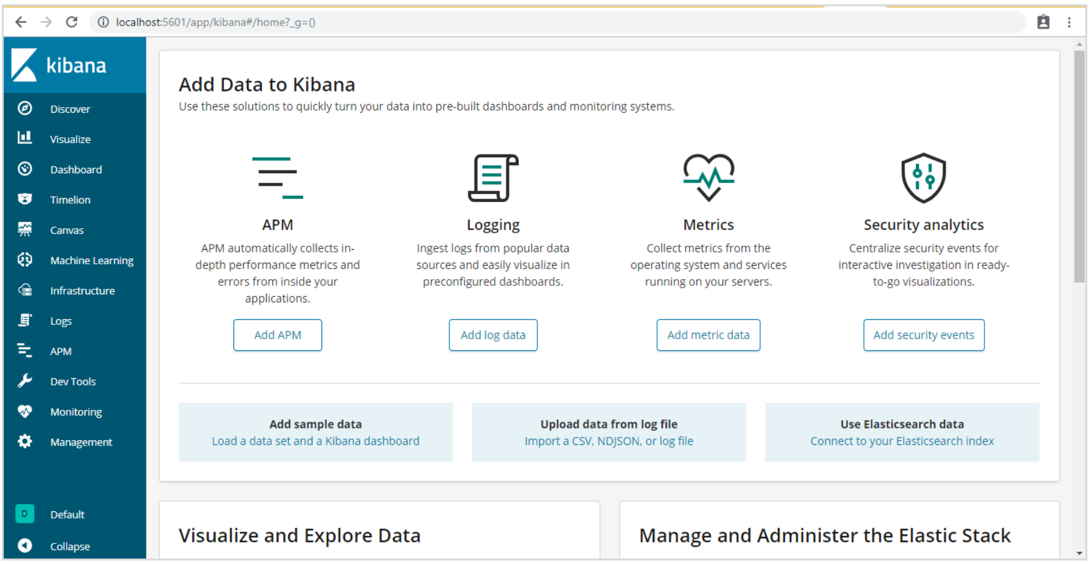
Once Kibana starts, the user can see the following screen:



```
  log   [09:16:23.085] [info][status][plugin:console@6.5.4] Status changed from
uninitialized to green - Ready
  log   [09:16:23.100] [info][status][plugin:console_extensions@6.5.4] Status ch
anged from uninitialized to green - Ready
  log   [09:16:23.121] [info][status][plugin:notifications@6.5.4] Status changed
 from uninitialized to green - Ready
  log   [09:16:23.295] [info][status][plugin:infra@6.5.4] Status changed from un
initialized to green - Ready
  log   [09:16:23.325] [info][status][plugin:metrics@6.5.4] Status changed from
uninitialized to green - Ready
  log   [09:16:29.031] [warning][reporting] Generating a random key for xpack.re
porting.encryptionKey. To prevent pending reports from failing on restart, pleas
e set xpack.reporting.encryptionKey in kibana.yml
  log   [09:16:29.053] [error][status][plugin:reporting@6.5.4] Status changed fr
om uninitialized to red - Request Timeout after 3000ms
  log   [09:16:29.258] [info][license][xpack] Imported license information from
Elasticsearch for the [data] cluster: mode: basic | status: active
  log   [09:16:29.277] [info][status][plugin:xpack_main@6.5.4] Status changed fr
om red to green - Ready
  log   [09:16:29.282] [info][status][plugin:searchprofiler@6.5.4] Status change
d from red to green - Ready
  log   [09:16:29.287] [info][status][plugin:ml@6.5.4] Status changed from red t
o green - Ready
  log   [09:16:29.293] [info][status][plugin:tilemap@6.5.4] Status changed from
red to green - Ready
  log   [09:16:29.298] [info][status][plugin:watcher@6.5.4] Status changed from
red to green - Ready
  log   [09:16:29.303] [info][status][plugin:index_management@6.5.4] Status chan
ged from red to green - Ready
  log   [09:16:29.309] [info][status][plugin:rollup@6.5.4] Status changed from r
ed to green - Ready
  log   [09:16:29.315] [info][status][plugin:graph@6.5.4] Status changed from re
d to green - Ready
  log   [09:16:29.327] [info][status][plugin:grokdebugger@6.5.4] Status changed
from red to green - Ready
  log   [09:16:29.332] [info][status][plugin:logstash@6.5.4] Status changed from
 red to green - Ready
  log   [09:16:29.337] [info][status][plugin:beats_management@6.5.4] Status chan
ged from red to green - Ready
  log   [09:16:29.343] [info][status][plugin:reporting@6.5.4] Status changed fro
m red to green - Ready
  log   [09:16:29.349] [info][kibana-monitoring][monitoring-ui] Starting monitor
ing stats collection
  log   [09:16:29.402] [info][status][plugin:security@6.5.4] Status changed from
 red to green - Ready
  log   [09:16:29.455] [info][status][plugin:elasticsearch@6.5.4] Status changed
 from red to green - Ready
  log   [09:16:30.349] [info][license][xpack] Imported license information from
Elasticsearch for the [monitoring] cluster: mode: basic | status: active
  log   [09:16:35.472] [info][migrations] Creating index .kibana_1.
  log   [09:16:37.733] [info][migrations] Pointing alias .kibana to .kibana_1.
  log   [09:16:38.122] [info][migrations] Finished in 2650ms.
  log   [09:16:38.133] [info][listening] Server running at http://localhost:5601

  log   [09:16:39.844] [info][status][plugin:spaces@6.5.4] Status changed from r
ed to green - Ready
```

Once you see the ready signal in the console, you can open Kibana in browser using http://localhost:5601/.The default port on which kibana is available is 5601.

The user interface of Kibana is as shown here:



Add Data to Kibana

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

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

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

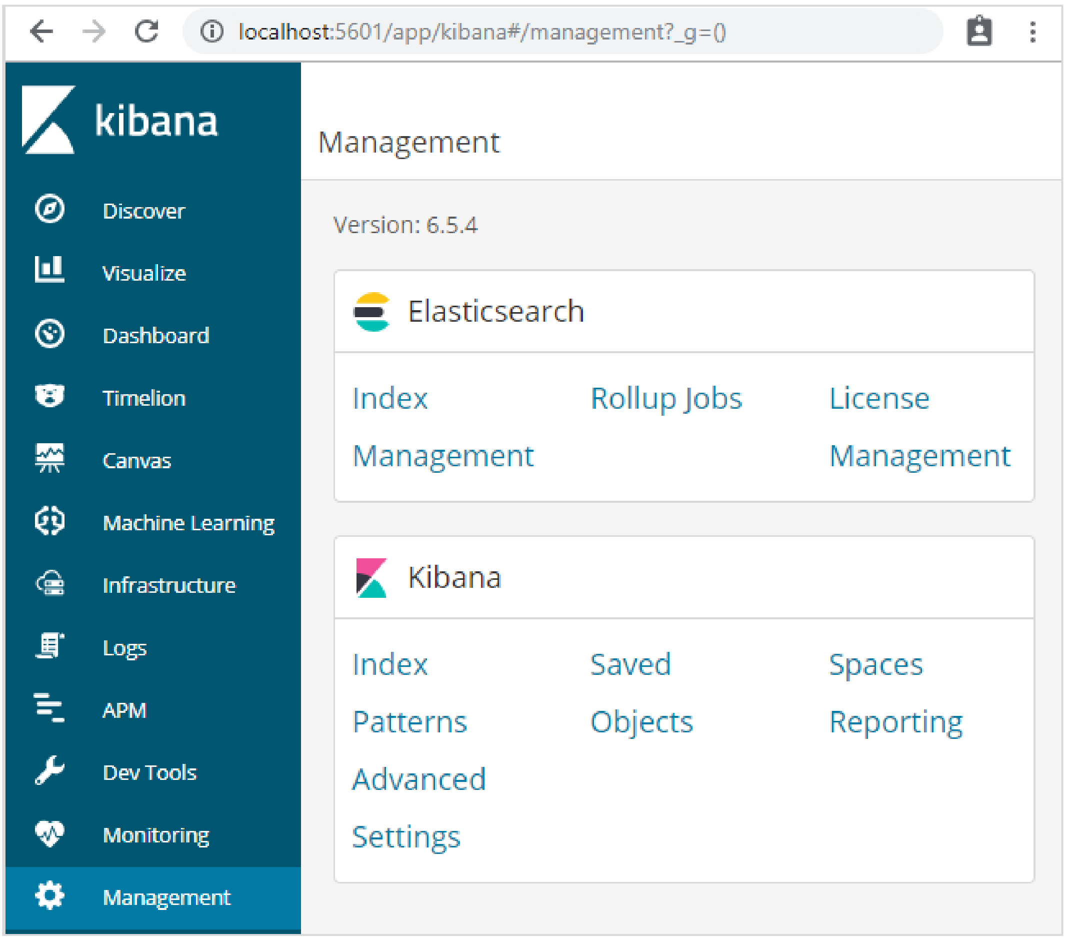
Collect metrics from the operating system and services running on your servers.

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

Visualize and Explore Data

Manage and Administer the Elastic Stack

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.



Management

Version: 6.5.4

3. Kibana— Introduction to ELK Stack

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. 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
Columns: *"Country","Region","Population","Area"*

You can also create a dummy csv file and use it. We will be using logstash to dump this data from *countriesdata.csv* to elasticsearch.

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:

```
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

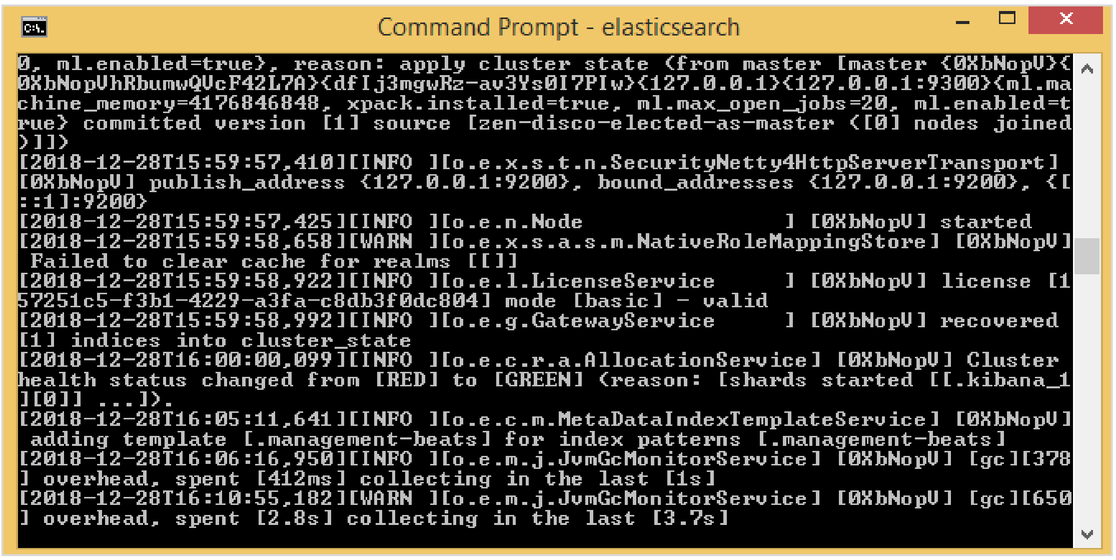
Will have the csv component with separator used which in our case is comma, and also the columns available for our csv file.As logstash considers all the data coming in as string , in-case we want any column to be used as integer , float the same has to be specified using mutate as shown above.

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.

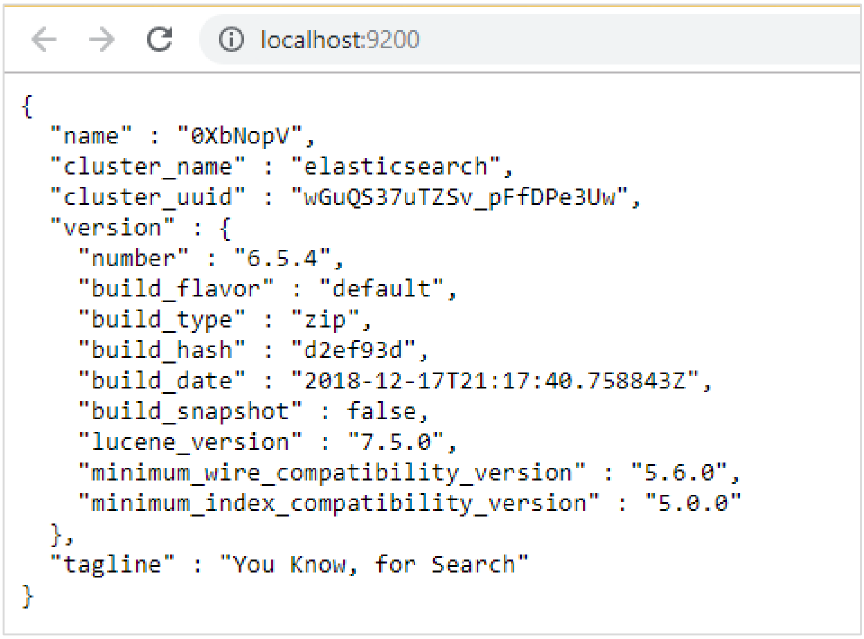
Save the above config file as *logstash_countries.config*. Note that we need to give the path of this config to logstash command in the next step.

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


Command Prompt - elasticsearch

```
0, ml.enabled=true}, reason: apply cluster state (from master [master {0XbNopV}{
0XbNopVhRbumwQVcF42L7A}{dfIj3mgwRz-av3Ys0I7PIw}{127.0.0.1}{127.0.0.1:9300}{ml.ma
chine_memory=4176846848, xpack.installed=true, ml.max_open_jobs=20, ml.enabled=t
rue} committed version [1] source [zen-disco-elected-as-master ([0] nodes joined
)]])
[2018-12-28T15:59:57,410][INFO ][o.e.x.s.t.n.SecurityNetty4HttpServerTransport]
[0XbNopV] publish_address {127.0.0.1:9200}, bound_addresses {127.0.0.1:9200}, {[
::1]:9200}
[2018-12-28T15:59:57,425][INFO ][o.e.n.Node               ] [0XbNopV] started
[2018-12-28T15:59:58,658][WARN ][o.e.x.s.a.s.m.NativeRoleMappingStore] [0XbNopV]
 Failed to clear cache for realms [[]]
[2018-12-28T15:59:58,922][INFO ][o.e.l.LicenseService     ] [0XbNopV] license [1
57251c5-f3b1-4229-a3fa-c8db3f0dc804] mode [basic] - valid
[2018-12-28T15:59:58,992][INFO ][o.e.g.GatewayService     ] [0XbNopV] recovered
[1] indices into cluster_state
[2018-12-28T16:00:00,099][INFO ][o.e.c.r.a.AllocationService] [0XbNopV] Cluster
health status changed from [RED] to [GREEN] (reason: [shards started [[.kibana_1
][0]] ...]).
[2018-12-28T16:05:11,641][INFO ][o.e.c.m.MetaDataIndexTemplateService] [0XbNopV]
 adding template [.management-beats] for index patterns [.management-beats]
[2018-12-28T16:06:16,950][INFO ][o.e.m.j.JvmGcMonitorService] [0XbNopV] [gc][378
] overhead, spent [412ms] collecting in the last [1s]
[2018-12-28T16:10:55,182][WARN ][o.e.m.j.JvmGcMonitorService] [0XbNopV] [gc][650
] overhead, spent [2.8s] collecting in the last [3.7s]
```

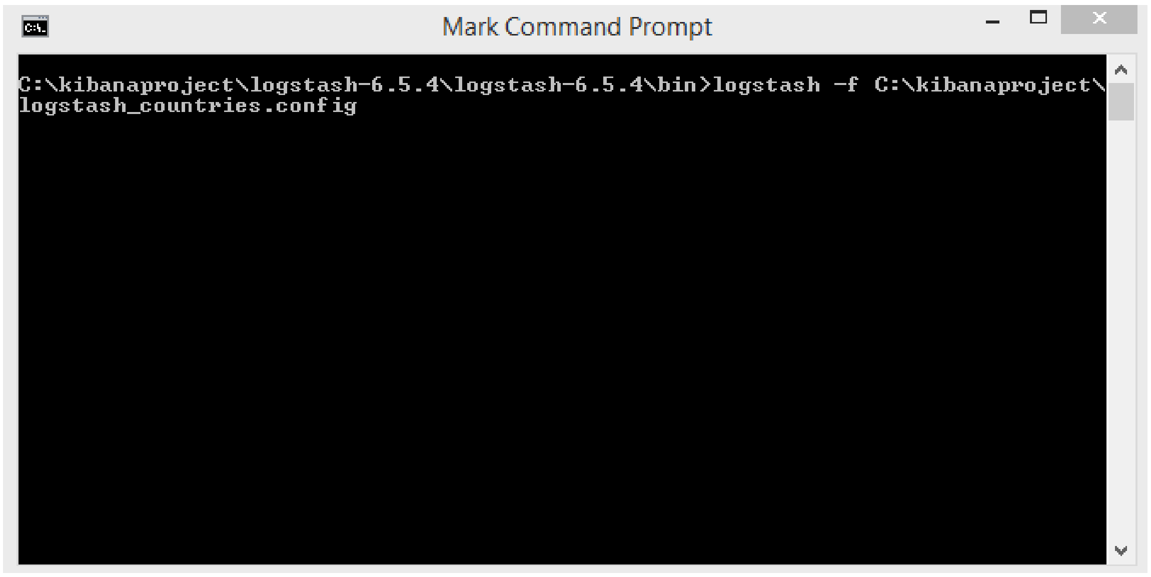
Now, run http://localhost:9200 in the browser to confirm if elasticsearch is running successfully.


localhost:9200

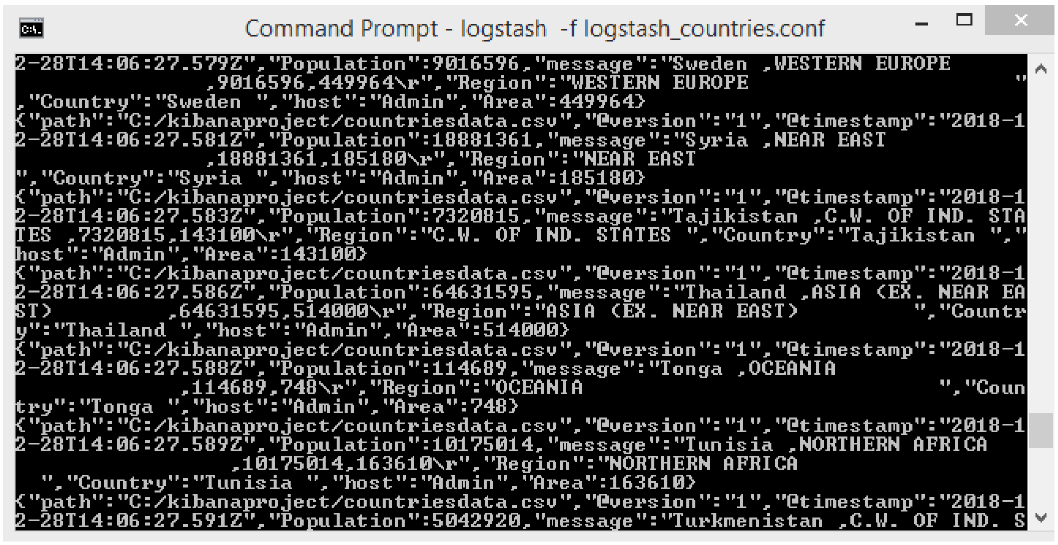
```
{
  "name" : "0XbNopV",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "wGuQS37uTZSv_pFfDPe3Uw",
  "version" : {
    "number" : "6.5.4",
    "build_flavor" : "default",
    "build_type" : "zip",
    "build_hash" : "d2ef93d",
    "build_date" : "2018-12-17T21:17:40.758843Z",
    "build_snapshot" : false,
    "lucene_version" : "7.5.0",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

We have elasticsearch running. Now go to the path where logstash is installed and run following command to upload the data to elasticsearch.

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

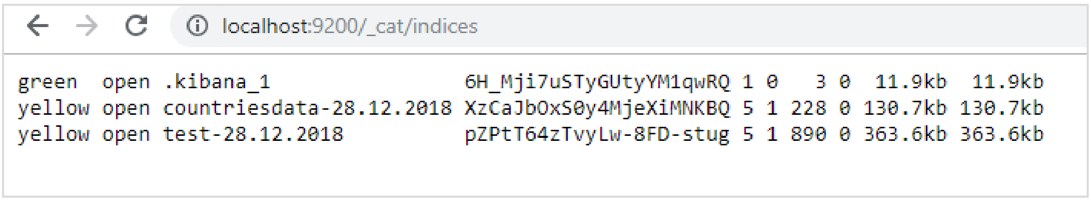


```
C:\kibanaproject\logstash-6.5.4\logstash-6.5.4\bin>logstash -f C:\kibanaproject\
logstash_countries.config
```



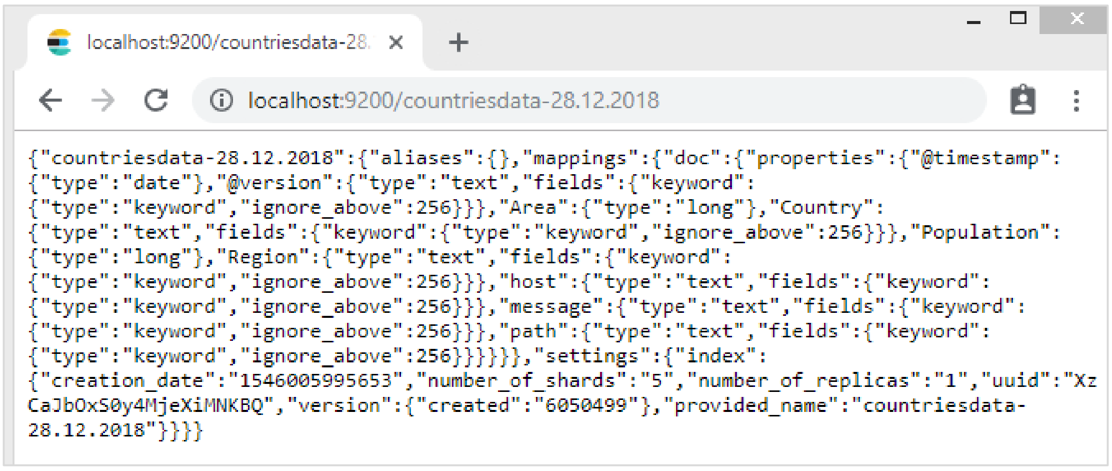
```
2-28T14:06:27.579Z","Population":9016596,"message":"Sweden ,WESTERN EUROPE
                 ,9016596,449964\r","Region":"WESTERN EUROPE                   "
,"Country":"Sweden ","host":"Admin","Area":449964}
{"path":"C:/kibanaproject/countriesdata.csv","@version":"1","@timestamp":"2018-1
2-28T14:06:27.581Z","Population":18881361,"message":"Syria ,NEAR EAST
                 ,18881361,185180\r","Region":"NEAR EAST
","Country":"Syria ","host":"Admin","Area":185180}
{"path":"C:/kibanaproject/countriesdata.csv","@version":"1","@timestamp":"2018-1
2-28T14:06:27.583Z","Population":7320815,"message":"Tajikistan ,C.W. OF IND. STA
TES ,7320815,143100\r","Region":"C.W. OF IND. STATES ","Country":"Tajikistan ","
host":"Admin","Area":143100}
{"path":"C:/kibanaproject/countriesdata.csv","@version":"1","@timestamp":"2018-1
2-28T14:06:27.586Z","Population":64631595,"message":"Thailand ,ASIA (EX. NEAR EA
ST)          ,64631595,514000\r","Region":"ASIA (EX. NEAR EAST)         ","Countr
y":"Thailand ","host":"Admin","Area":514000}
{"path":"C:/kibanaproject/countriesdata.csv","@version":"1","@timestamp":"2018-1
2-28T14:06:27.588Z","Population":114689,"message":"Tonga ,OCEANIA
                ,114689,748\r","Region":"OCEANIA                           ","Coun
try":"Tonga ","host":"Admin","Area":748}
{"path":"C:/kibanaproject/countriesdata.csv","@version":"1","@timestamp":"2018-1
2-28T14:06:27.589Z","Population":10175014,"message":"Tunisia ,NORTHERN AFRICA
                    ,10175014,163610\r","Region":"NORTHERN AFRICA
  ","Country":"Tunisia ","host":"Admin","Area":163610}
{"path":"C:/kibanaproject/countriesdata.csv","@version":"1","@timestamp":"2018-1
2-28T14:06:27.591Z","Population":5042920,"message":"Turkmenistan ,C.W. OF IND. S
```

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 same as follows:
We can see the countriesdata-28.12.2018 index created as shown above.



```
green  open .kibana_1                6H_Mji7uSTyGUtyYM1qwRQ 1 0   3 0  11.9kb  11.9kb
yellow open countriesdata-28.12.2018 XzCaJbOxS0y4MjeXiMNKBQ 5 1 228 0 130.7kb 130.7kb
yellow open test-28.12.2018          pZPtT64zTvyLw-8FD-stug 5 1 890 0 363.6kb 363.6kb
```

The details of the index: countries-28.12.2018 is as follows:

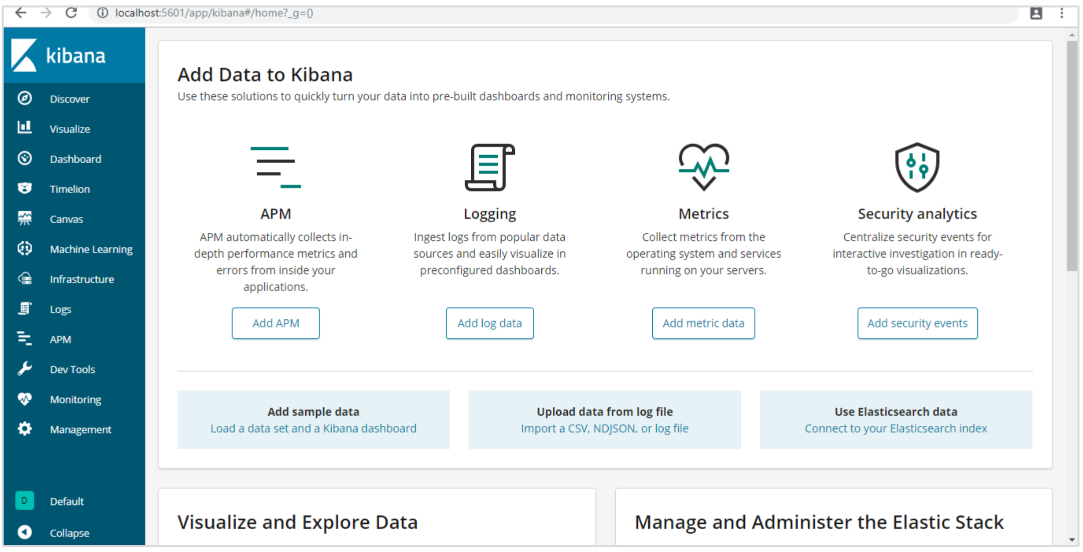

localhost:9200/countriesdata-28.
localhost:9200/countriesdata-28.12.2018

```
{"countriesdata-28.12.2018":{"aliases":{},"mappings":{"doc":{"properties":{"@timestamp":
{"type":"date"},"@version":{"type":"text","fields":{"keyword":
{"type":"keyword","ignore_above":256}}},"Area":{"type":"long"},"Country":
{"type":"text","fields":{"keyword":{"type":"keyword","ignore_above":256}}},"Population":
{"type":"long"},"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":"1546005995653","number_of_shards":"5","number_of_replicas":"1","uuid":"Xz
CaJbOxS0y4MjeXiMNKBQ","version":{"created":"6050499"},"provided_name":"countriesdata-
28.12.2018"}}}}
```

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

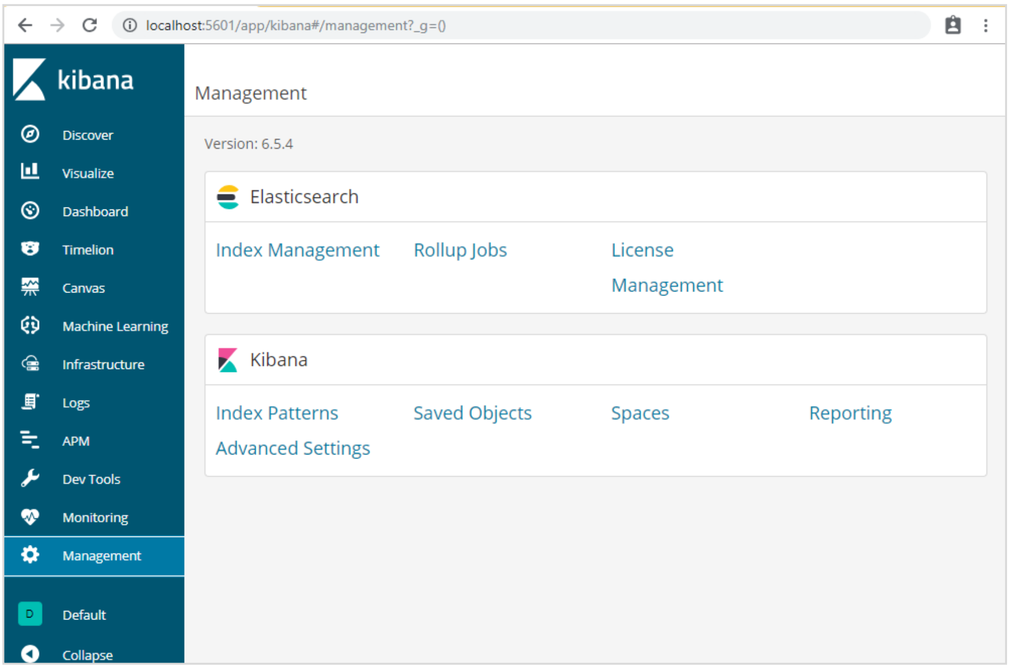
Use Data from Elasticsearch in Kibana

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

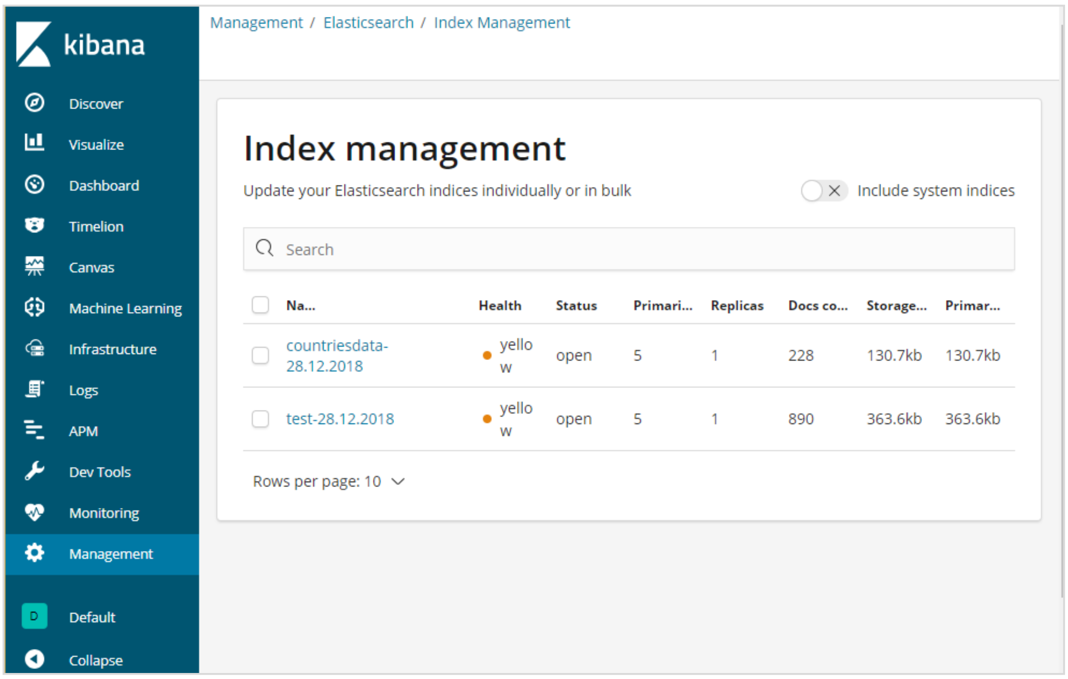

localhost:5601/app/kibana#/home?_g=()
kibana
Discover
Visualize
Dashboard
Timelion
Canvas
Machine Learning
Infrastructure
Logs
APM
Dev Tools
Monitoring
Management
Default
Collapse
Add Data to Kibana
Use these solutions to quickly turn your data into pre-built dashboards and monitoring systems.
APM
APM automatically collects in-depth performance metrics and errors from inside your applications.
Add APM
Logging
Ingest logs from popular data sources and easily visualize in preconfigured dashboards.
Add log data
Metrics
Collect metrics from the operating system and services running on your servers.
Add metric data
Security analytics
Centralize security events for interactive investigation in ready-to-go visualizations.
Add security events
Add sample data
Load a data set and a Kibana dashboard
Upload data from log file
Import a CSV, NDJSON, or log file
Use Elasticsearch data
Connect to your Elasticsearch index
Visualize and Explore Data
Manage and Administer the Elastic Stack

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:



Index management

Update your Elasticsearch indices individually or in bulk

Na...	Health	Status	Primari...	Replicas	Docs co...	Storage...	Primar...
countriesdata-28.12.2018	● yellow	open	5	1	228	130.7kb	130.7kb
test-28.12.2018	● yellow	open	5	1	890	363.6kb	363.6kb

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.

4. Kibana — Loading Sample Data

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

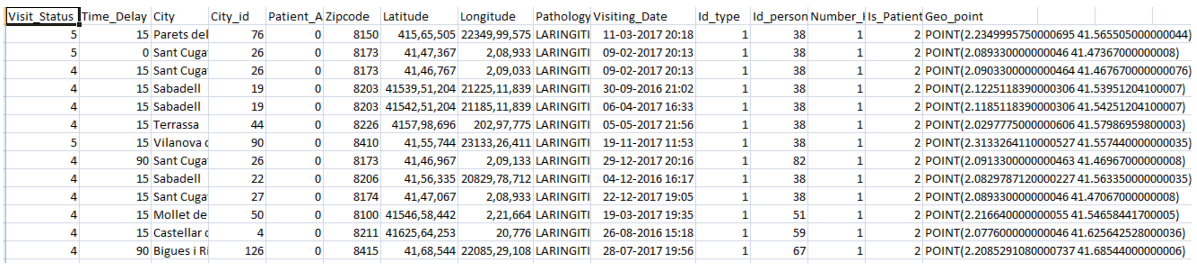
We are going to use data in the form of CSV format and the same is taken from Kaggle.com which deals with data that you can use for an analysis.

The data home medical visits to be used here is picked up from site Kaggle.com.

The following are the fields available for the CSV file:

```
["Visit_Status","Time_Delay","City","City_id","Patient_Age","Zipcode","Latitude
","Longitude",

"Pathology","Visiting_Date","Id_type","Id_personal","Number_Home_Visits","Is_Pa
tient_Minor","Geo_point"]
```

The Home_visits.csv is as follows:



Visit_Status	Time_Delay	City	City_id	Patient_A	Zipcode	Latitude	Longitude	Pathology	Visiting_Date	Id_type	Id_person	Number_I	Is_Patient	Geo_point
5	15	Parets del	76	0	8150	415,65,505	22349,99,575	LARINGITI	11-03-2017 20:18	1	38	1	2	POINT(2.2349995750000695 41.565505000000044)
5	0	Sant Cuga	26	0	8173	41,47,367	2,08,933	LARINGITI	09-02-2017 20:13	1	38	1	2	POINT(2.089330000000046 41.47367000000008)
4	15	Sant Cuga	26	0	8173	41,46,767	2,09,033	LARINGITI	09-02-2017 20:13	1	38	1	2	POINT(2.0903300000000464 41.467670000000076)
4	15	Sabadell	19	0	8203	41539,51,204	21225,11,839	LARINGITI	30-09-2016 21:02	1	38	1	2	POINT(2.1225118390000306 41.53951204100007)
4	15	Sabadell	19	0	8203	41542,51,204	21185,11,839	LARINGITI	06-04-2017 16:33	1	38	1	2	POINT(2.1185118390000306 41.54251204100007)
4	15	Terrassa	44	0	8226	4157,98,696	202,97,775	LARINGITI	05-05-2017 21:56	1	38	1	2	POINT(2.0297775000000606 41.57986959800003)
5	15	Vilanova c	90	0	8410	41,55,744	23133,26,411	LARINGITI	19-11-2017 11:53	1	38	1	2	POINT(2.3133264110000527 41.557440000000035)
4	90	Sant Cuga	26	0	8173	41,46,967	2,09,133	LARINGITI	29-12-2017 20:16	1	82	1	2	POINT(2.0913300000000463 41.46967000000008)
4	15	Sabadell	22	0	8206	41,56,335	20829,78,712	LARINGITI	04-12-2016 16:17	1	38	1	2	POINT(2.0829787120000227 41.563350000000035)
4	15	Sant Cuga	27	0	8174	41,47,067	2,08,933	LARINGITI	22-12-2017 19:05	1	38	1	2	POINT(2.089330000000046 41.47067000000008)
4	15	Mollet de	50	0	8100	41546,58,442	2,21,664	LARINGITI	19-03-2017 19:35	1	51	1	2	POINT(2.216640000000055 41.54658441700005)
4	15	Castellar c	4	0	8211	41625,64,253	20,776	LARINGITI	26-08-2016 15:18	1	59	1	2	POINT(2.077600000000046 41.62564252800036)
4	90	Bigues i Ri	126	0	8415	41,68,544	22085,29,108	LARINGITI	28-07-2017 19:56	1	67	1	2	POINT(2.2085291080000737 41.68544000000006)

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

```
input {
      file {
            path => "C:/kibanaproject/home_visits.csv"
            start_position => "beginning"
            sincedb_path => "NUL"
     }
}
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:

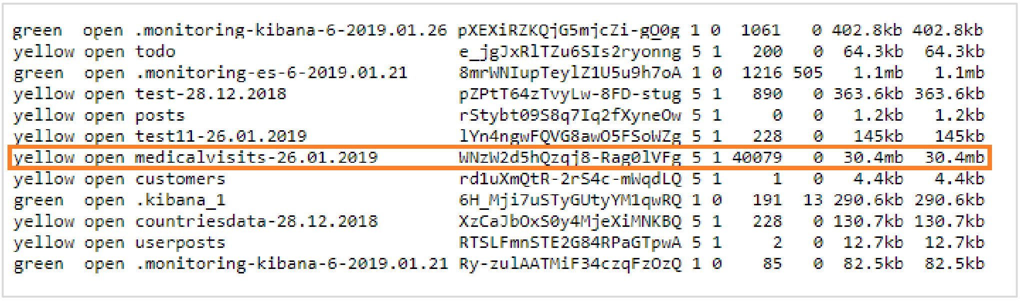
```
mutate {convert => ["Number_Home_Visits", "integer"]}
mutate {convert => ["City_id", "integer"]}
mutate {convert => ["Id_personal", "integer"]}
mutate {convert => ["Id_type", "integer"]}
mutate {convert => ["Zipcode", "integer"]}
mutate {convert => ["Patient_Age", "integer"]}
```

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.

```
logstash -f logstash_homevisists.conf
```

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



```
green  open .monitoring-kibana-6-2019.01.26 pXEXiRZKQjG5mjcZi-gQ0g 1 0  1061   0 402.8kb 402.8kb
yellow open todo                            e_jgJxRlTZu6SIs2ryonng 5 1   200   0  64.3kb  64.3kb
green  open .monitoring-es-6-2019.01.21     8mrWNIupTeylZ1U5u9h7oA 1 0  1216 505   1.1mb   1.1mb
yellow open test-28.12.2018                 pZPtT64zTvyLw-8FD-stug 5 1   890   0 363.6kb 363.6kb
yellow open posts                           rStybt09S8q7Iq2fXyneOw 5 1     0   0   1.2kb   1.2kb
yellow open test11-26.01.2019               lYn4ngwFQVG8awO5FSoWZg 5 1   228   0   145kb   145kb
yellow open medicalvisits-26.01.2019        WNzW2d5hQzqj8-Rag0lVFg 5 1 40079   0  30.4mb  30.4mb
yellow open customers                       rd1uXmQtR-2rS4c-mWqdLQ 5 1     1   0   4.4kb   4.4kb
green  open .kibana_1                       6H_Mji7uSTyGUtyYM1qwRQ 1 0   191  13 290.6kb 290.6kb
yellow open countriesdata-28.12.2018        XzCaJbOxS0y4MjeXiMNKBQ 5 1   228   0 130.7kb 130.7kb
yellow open userposts                       RTSLFmnSTE2G84RPaGTpwA 5 1     2   0  12.7kb  12.7kb
green  open .monitoring-kibana-6-2019.01.21 Ry-zulAATMiF34czqFzOzQ 1 0    85   0  82.5kb  82.5kb
```

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

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

The json code to used with Kibana has to be with indexed as follows:

```
{"index":{"_index":"shakespeare","_id":0}}
{"type":"act","line_id":1,"play_name":"Henry IV",
"speech_number":"","line_number":"","speaker":"","text_entry":"ACT I"}
{"index":{"_index":"shakespeare","_id":1}}
{"type":"scene","line_id":2,"play_name":"Henry
IV","speech_number":"","line_number":"","speaker":"","text_entry":"SCENE I.
London. The palace."}
{"index":{"_index":"shakespeare","_id":2}}
{"type":"line","line_id":3,"play_name":"Henry
IV","speech_number":"","line_number":"","speaker":"","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 :**{"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:

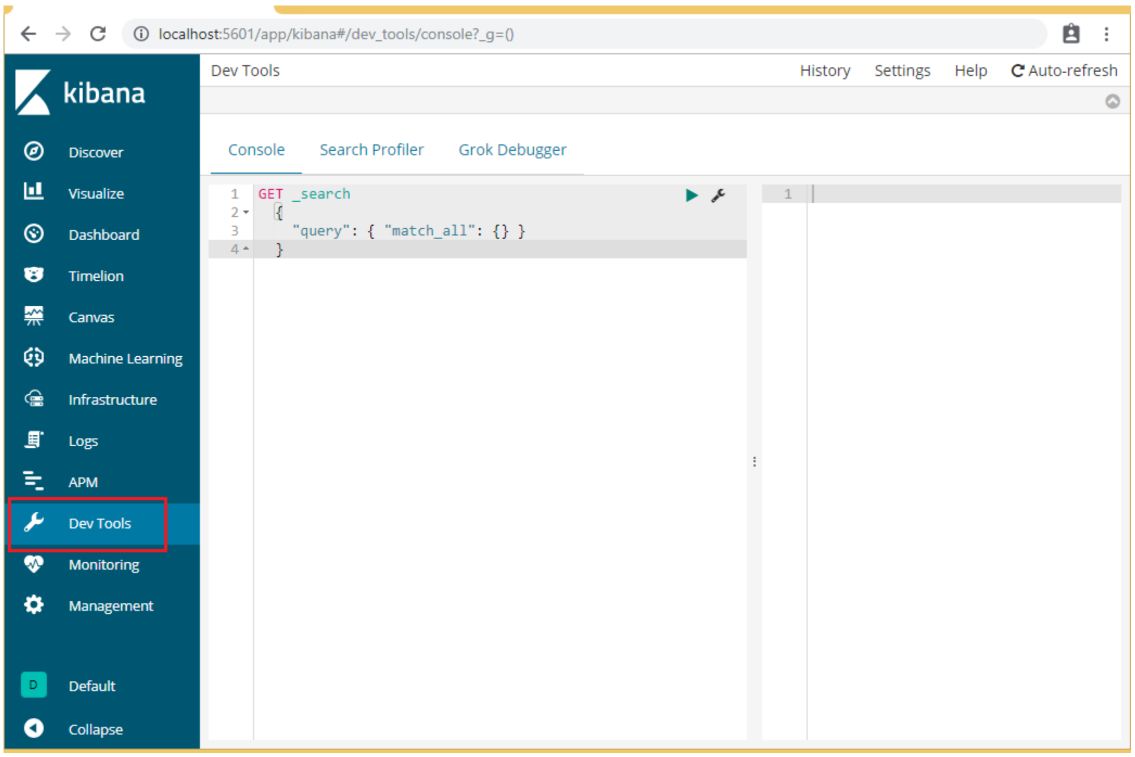
PHP Code

```
<?php
$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;
```

```
$myfile1 = fopen("todonewfile.json", "w") or die("Unable to open file!"); //
writes a new file to be used in kibana dev tool

foreach ($farray as $a => $value) {

     $_index = json_decode('{"index": {"_index": "'.$index_name.'", "_id":
"'.$i.'"}}');

     fwrite($myfile1, json_encode($_index));

     fwrite($myfile1, "\n");

     fwrite($myfile1, json_encode($value));

     fwrite($myfile1, "\n");

     $i++;

}

?>
```

We have taken the todo json file from 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:



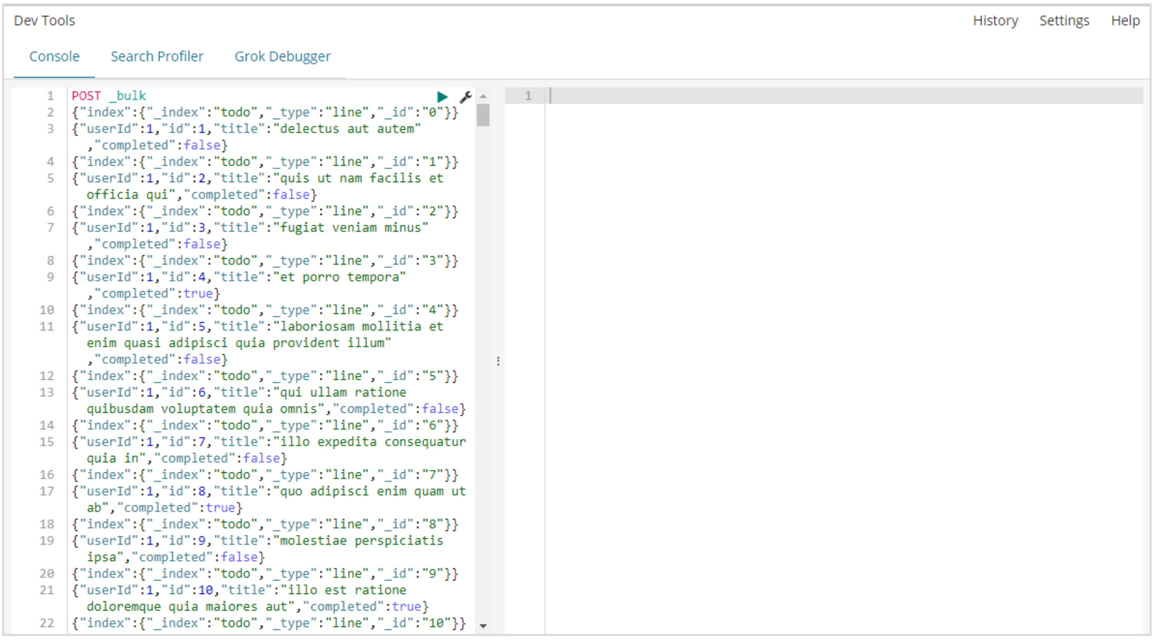
```
GET _search
{
  "query": { "match_all": {} }
}
```

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.

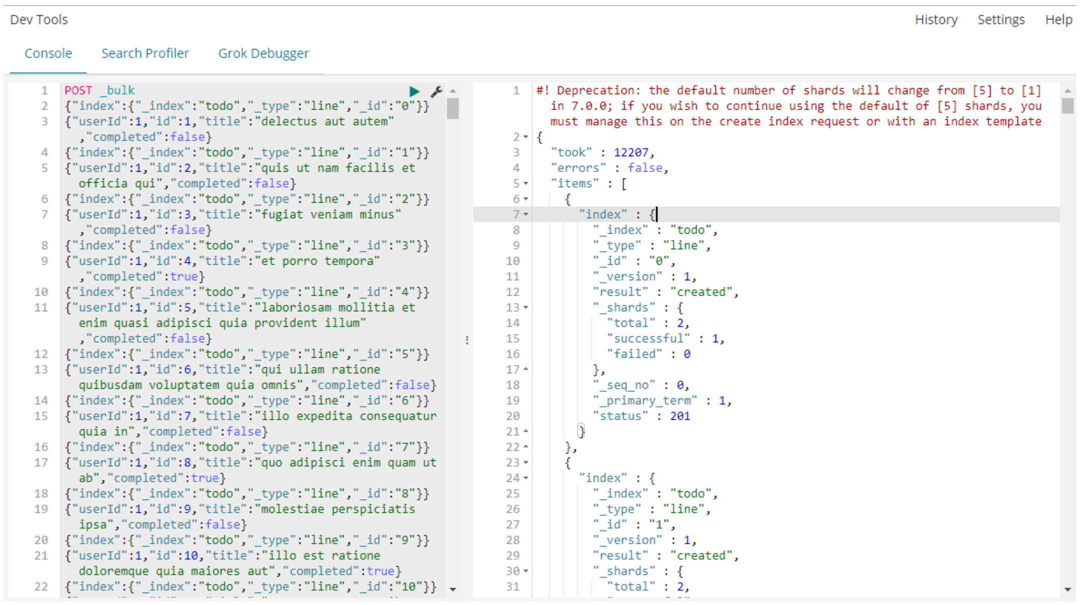
The command to be used in dev tools to upload the json data is:

```
POST _bulk
```

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



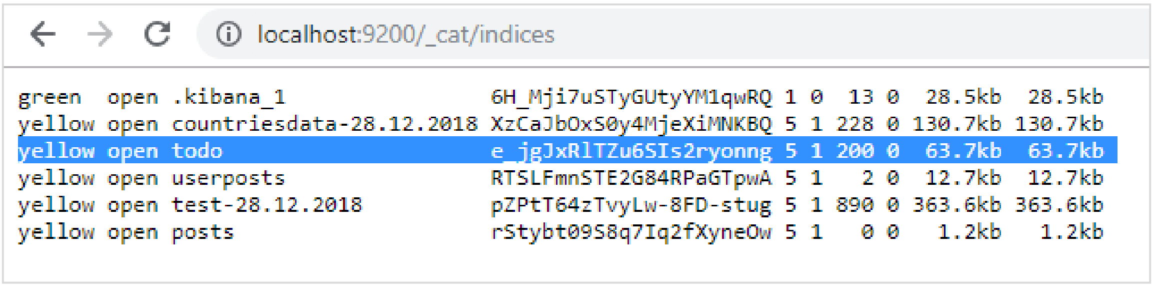
```
POST _bulk
{"index":{"_index":"todo","_type":"line","_id":"0"}}
{"userId":1,"id":1,"title":"delectus aut autem"
  ,"completed":false}
{"index":{"_index":"todo","_type":"line","_id":"1"}}
{"userId":1,"id":2,"title":"quis ut nam facilis et
  officia qui","completed":false}
{"index":{"_index":"todo","_type":"line","_id":"2"}}
{"userId":1,"id":3,"title":"fugiat veniam minus"
  ,"completed":false}
{"index":{"_index":"todo","_type":"line","_id":"3"}}
{"userId":1,"id":4,"title":"et porro tempora"
  ,"completed":true}
{"index":{"_index":"todo","_type":"line","_id":"4"}}
{"userId":1,"id":5,"title":"laboriosam mollitia et
  enim quasi adipisci quia provident illum"
  ,"completed":false}
{"index":{"_index":"todo","_type":"line","_id":"5"}}
{"userId":1,"id":6,"title":"qui ullam ratione
  quibusdam voluptatem quia omnis","completed":false}
{"index":{"_index":"todo","_type":"line","_id":"6"}}
{"userId":1,"id":7,"title":"illo expedita consequatur
  quia in","completed":false}
{"index":{"_index":"todo","_type":"line","_id":"7"}}
{"userId":1,"id":8,"title":"quo adipisci enim quam ut
  ab","completed":true}
{"index":{"_index":"todo","_type":"line","_id":"8"}}
{"userId":1,"id":9,"title":"molestiae perspiciatis
  ipsa","completed":false}
{"index":{"_index":"todo","_type":"line","_id":"9"}}
{"userId":1,"id":10,"title":"illo est ratione
  doloremque quia maiores aut","completed":true}
{"index":{"_index":"todo","_type":"line","_id":"10"}}
```



```
POST _bulk
{"index":{"_index":"todo","_type":"line","_id":"0"}}
{"userId":1,"id":1,"title":"delectus aut autem"
  ,"completed":false}
{"index":{"_index":"todo","_type":"line","_id":"1"}}
{"userId":1,"id":2,"title":"quis ut nam facilis et
  officia qui","completed":false}
{"index":{"_index":"todo","_type":"line","_id":"2"}}
{"userId":1,"id":3,"title":"fugiat veniam minus"
  ,"completed":false}
{"index":{"_index":"todo","_type":"line","_id":"3"}}
{"userId":1,"id":4,"title":"et porro tempora"
  ,"completed":true}
{"index":{"_index":"todo","_type":"line","_id":"4"}}
{"userId":1,"id":5,"title":"laboriosam mollitia et
  enim quasi adipisci quia provident illum"
  ,"completed":false}
{"index":{"_index":"todo","_type":"line","_id":"5"}}
{"userId":1,"id":6,"title":"qui ullam ratione
  quibusdam voluptatem quia omnis","completed":false}
{"index":{"_index":"todo","_type":"line","_id":"6"}}
{"userId":1,"id":7,"title":"illo expedita consequatur
  quia in","completed":false}
{"index":{"_index":"todo","_type":"line","_id":"7"}}
{"userId":1,"id":8,"title":"quo adipisci enim quam ut
  ab","completed":true}
{"index":{"_index":"todo","_type":"line","_id":"8"}}
{"userId":1,"id":9,"title":"molestiae perspiciatis
  ipsa","completed":false}
{"index":{"_index":"todo","_type":"line","_id":"9"}}
{"userId":1,"id":10,"title":"illo est ratione
  doloremque quia maiores aut","completed":true}
{"index":{"_index":"todo","_type":"line","_id":"10"}}
```

```
#! Deprecation: the default number of shards will change from [5] to [1]
  in 7.0.0; if you wish to continue using the default of [5] shards, you
  must manage this on the create index request or with an index template
{
  "took" : 12207,
  "errors" : false,
  "items" : [
    {
      "index" : {
        "_index" : "todo",
        "_type" : "line",
        "_id" : "0",
        "_version" : 1,
        "result" : "created",
        "_shards" : {
          "total" : 2,
          "successful" : 1,
          "failed" : 0
        },
        "_seq_no" : 0,
        "_primary_term" : 1,
        "status" : 201
      }
    },
    {
      "index" : {
        "_index" : "todo",
        "_type" : "line",
        "_id" : "1",
        "_version" : 1,
        "result" : "created",
        "_shards" : {
          "total" : 2,
```

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

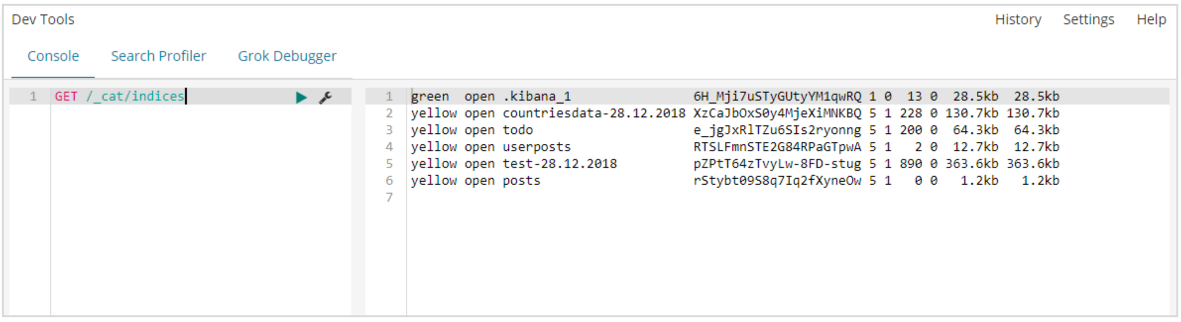


```
green  open .kibana_1                6H_Mji7uSTyGUtyYM1qwRQ 1 0   13 0  28.5kb  28.5kb
yellow open countriesdata-28.12.2018 XzCaJbOxS0y4MjeXiMNKBQ 5 1  228 0 130.7kb 130.7kb
yellow open todo                     e_jgJxRlTZu6SIs2ryonng 5 1  200 0  63.7kb  63.7kb
yellow open userposts                RTSLFmnSTE2G84RPaGTpwA 5 1    2 0  12.7kb  12.7kb
yellow open test-28.12.2018          pZPtT64zTvyLw-8FD-stug 5 1  890 0 363.6kb 363.6kb
yellow open posts                    rStybt09S8q7Iq2fXyneOw 5 1    0 0   1.2kb   1.2kb
```

You can check the same in dev tools itself as follows:

Command:

```
GET /_cat/indices
```



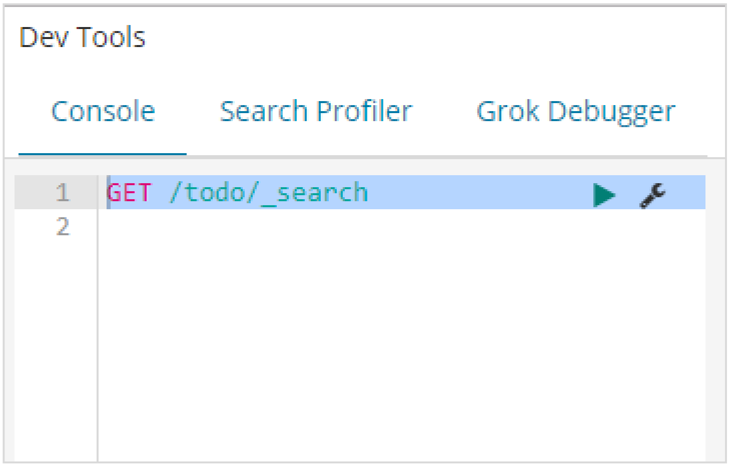
```
GET /_cat/indices
```

```
green  open .kibana_1                   6H_Mji7uSTyGUtyYM1qwRQ 1 0  13 0  28.5kb  28.5kb
yellow open countriesdata-28.12.2018 XzCaJbOxS0y4MjeXiMNKBQ 5 1 228 0 130.7kb 130.7kb
yellow open todo                     e_jgJxRlTZu6SIs2ryonng 5 1 200 0  64.3kb  64.3kb
yellow open userposts                RTSLFmnSTE2G84RPaGTpwA 5 1   2 0  12.7kb  12.7kb
yellow open test-28.12.2018          pZPtT64zTvyLw-8FD-stug 5 1 890 0 363.6kb 363.6kb
yellow open posts                    rStybt09S8q7Iq2fXyneOw 5 1   0 0   1.2kb   1.2kb
```

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

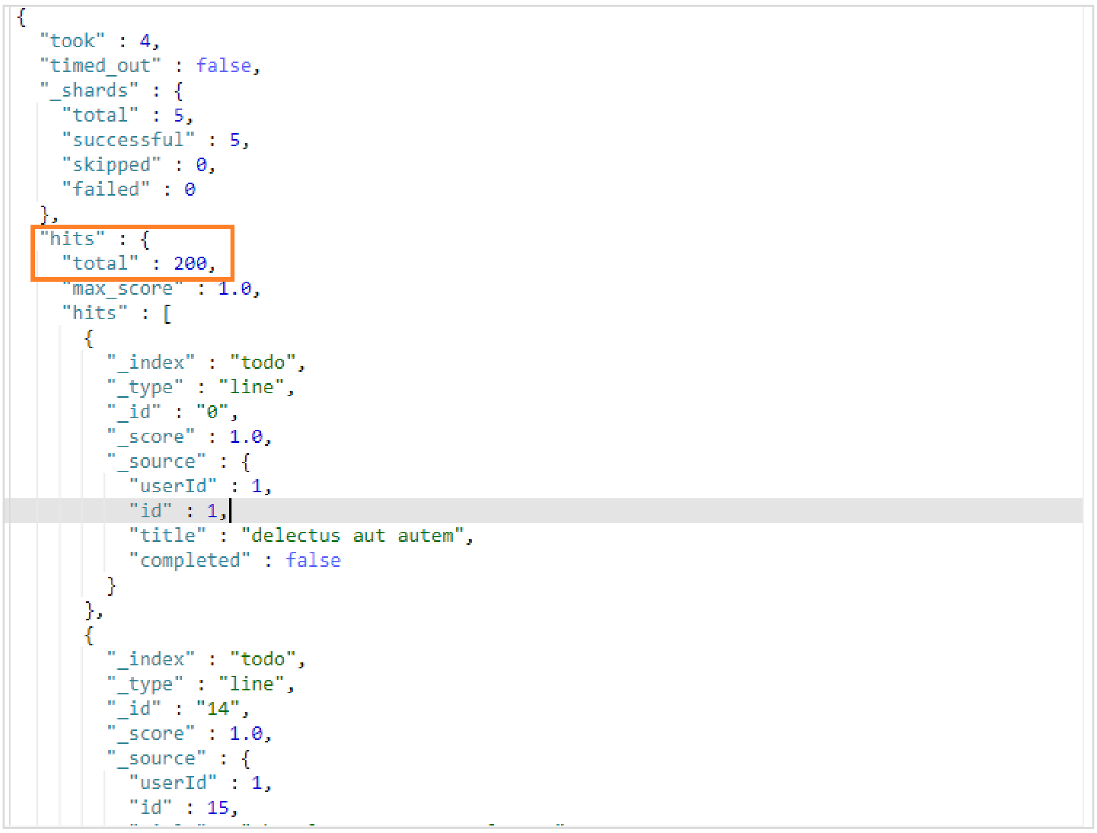
Command in dev tool

```
GET /todo/_search
```



```
GET /todo/_search
```

The output of the above search is as shown below:



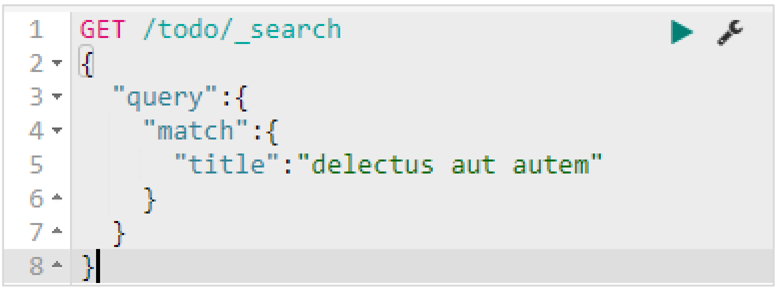
```
{
  "took" : 4,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 200,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "todo",
        "_type" : "line",
        "_id" : "0",
        "_score" : 1.0,
        "_source" : {
          "userId" : 1,
          "id" : 1,
          "title" : "delectus aut autem",
          "completed" : false
        }
      },
      {
        "_index" : "todo",
        "_type" : "line",
        "_id" : "14",
        "_score" : 1.0,
        "_source" : {
          "userId" : 1,
          "id" : 15,
```

It gives all the records present in the *todo*index. The total records we are getting is 200.

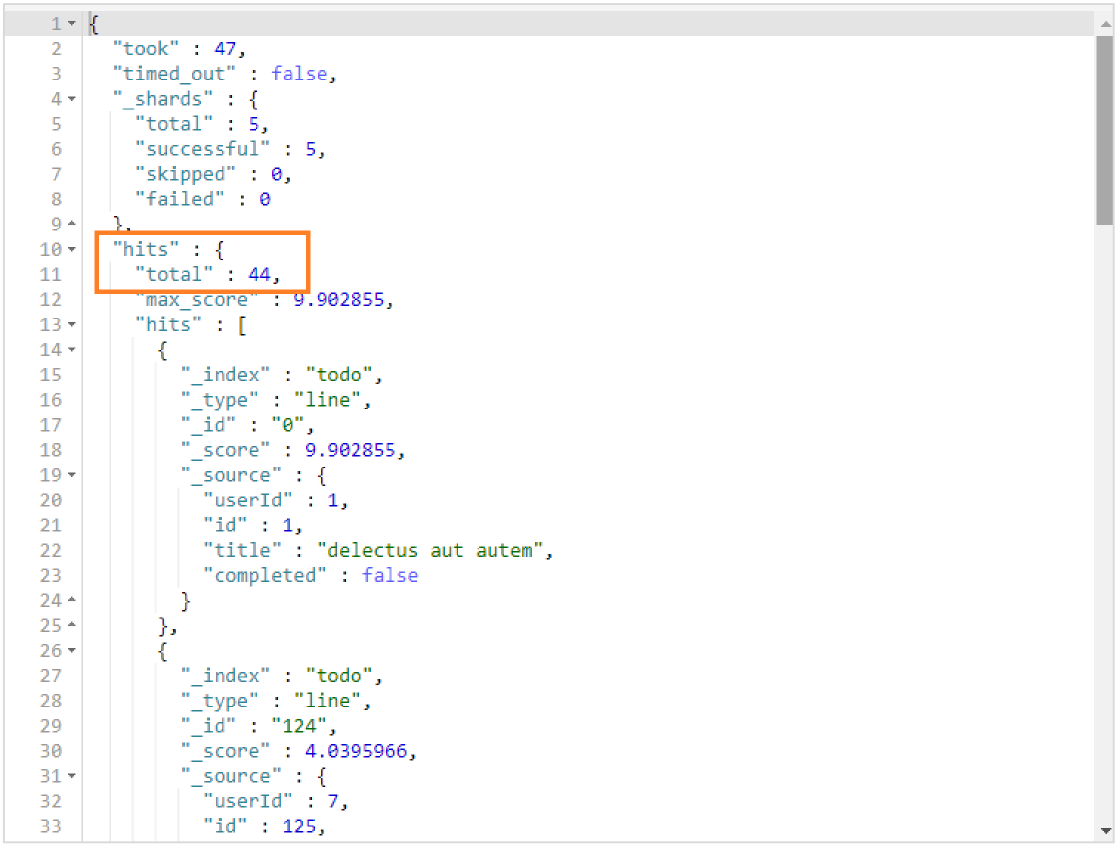
Search for a Record in *todo* Index

We can do that using the following command:

```
GET /todo/_search
{
      "query":{
             "match":{
                                         "title":"delectusautautem"
             }
      }
}
```



```
GET /todo/_search
{
  "query":{
    "match":{
      "title":"delectus aut autem"
    }
  }
}
```



```
{
  "took" : 47,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 44,
    "max_score" : 9.902855,
    "hits" : [
      {
        "_index" : "todo",
        "_type" : "line",
        "_id" : "0",
        "_score" : 9.902855,
        "_source" : {
          "userId" : 1,
          "id" : 1,
          "title" : "delectus aut autem",
          "completed" : false
        }
      },
      {
        "_index" : "todo",
        "_type" : "line",
        "_id" : "124",
        "_score" : 4.0395966,
        "_source" : {
          "userId" : 7,
          "id" : 125,
```

We are able to fetch the records which match with the title we have given.

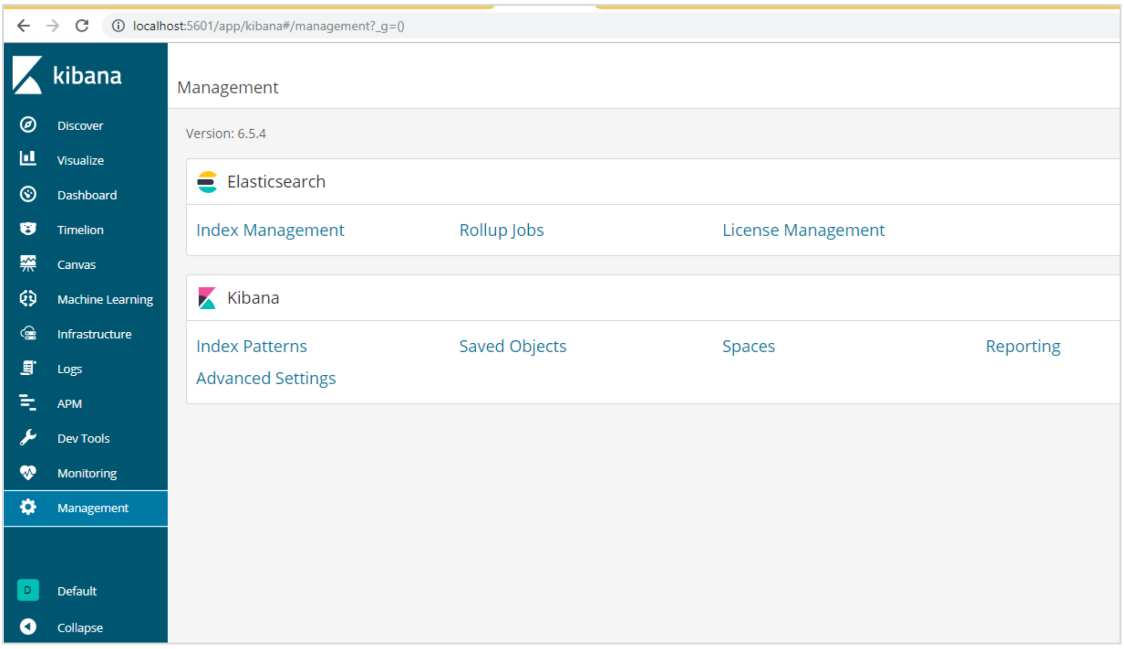
5. Kibana — Management

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:



Management

Version: 6.5.4

Elasticsearch

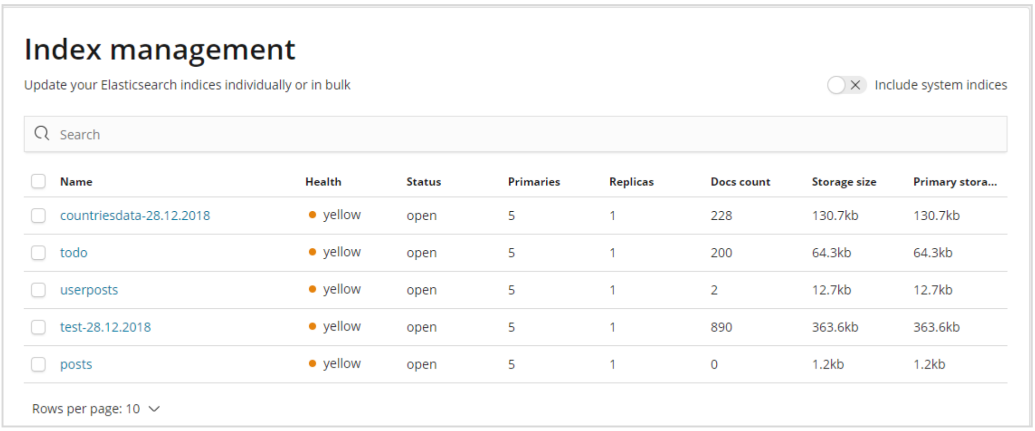
Index Management | Rollup Jobs | License Management

Kibana

Index Patterns | Saved Objects | Spaces | Reporting

Advanced Settings

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:



Index management

Update your Elasticsearch indices individually or in bulk

Include system indices

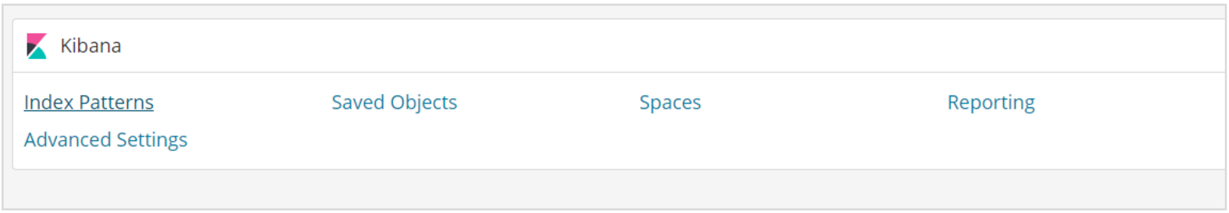
Search

Name	Health	Status	Primaries	Replicas	Docs count	Storage size	Primary stora...
countriesdata-28.12.2018	● yellow	open	5	1	228	130.7kb	130.7kb
todo	● yellow	open	5	1	200	64.3kb	64.3kb
userposts	● yellow	open	5	1	2	12.7kb	12.7kb
test-28.12.2018	● yellow	open	5	1	890	363.6kb	363.6kb
posts	● yellow	open	5	1	0	1.2kb	1.2kb

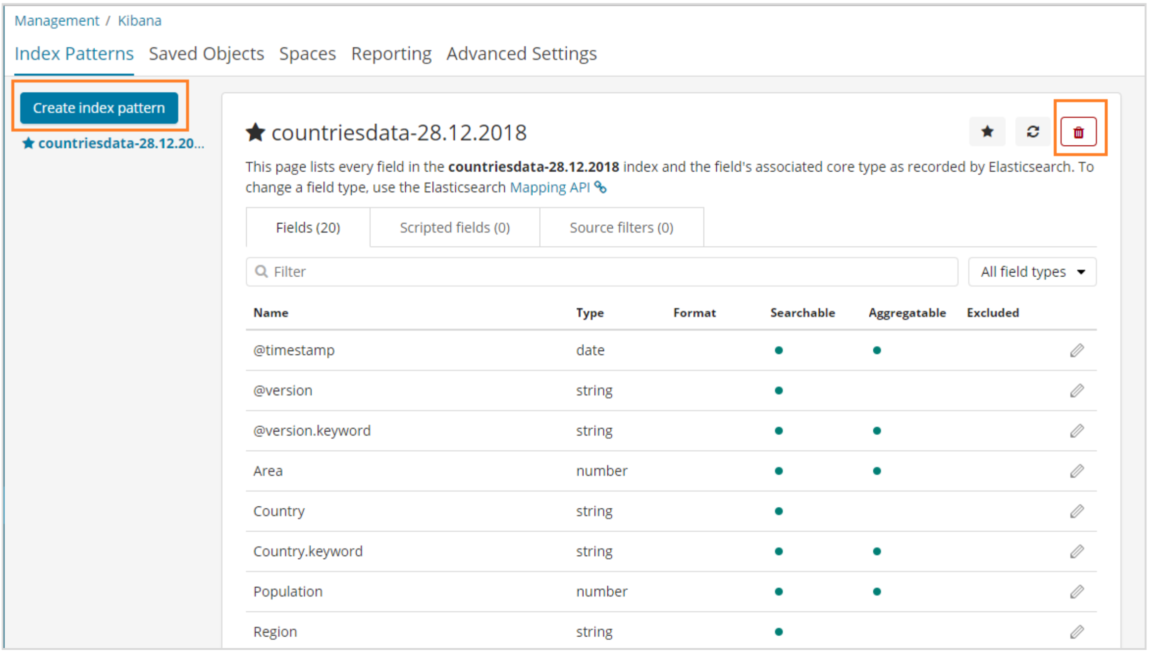
Rows per page: 10

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:



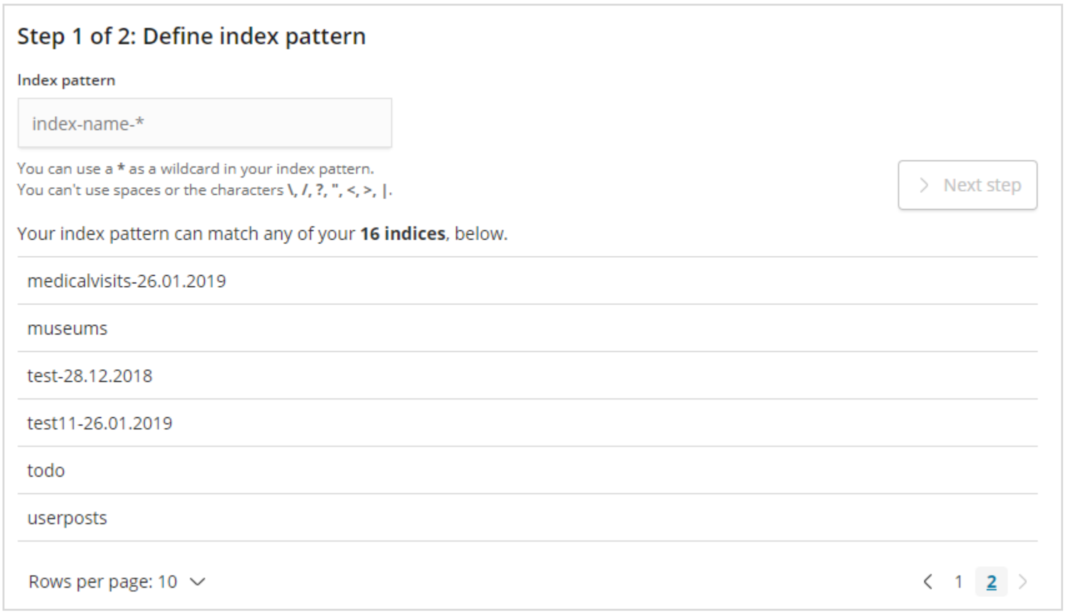
This page lists every field in the **countriesdata-28.12.2018** index and the field's associated core type as recorded by Elasticsearch. To change a field type, use the Elasticsearch Mapping API

Name	Type	Format	Searchable	Aggregatable	Excluded
@timestamp	date		●	●	
@version	string		●		
@version.keyword	string		●	●	
Area	number		●	●	
Country	string		●		
Country.keyword	string		●	●	
Population	number		●	●	
Region	string		●		

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.



Step 1 of 2: Define index pattern

Index pattern

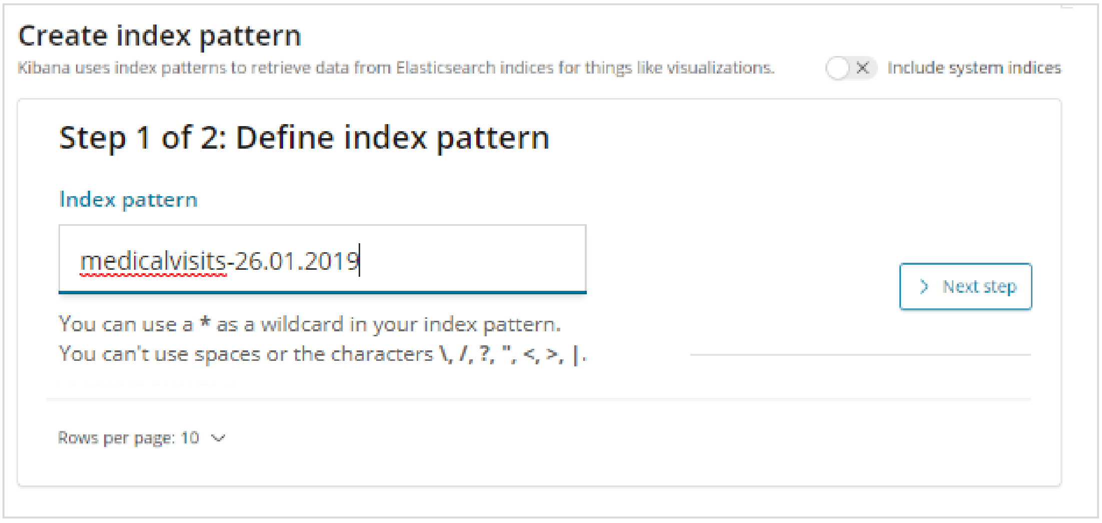
index-name-*

You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, ", <, >, |.

Your index pattern can match any of your **16 indices**, below.

medicalvisits-26.01.2019
museums
test-28.12.2018
test11-26.01.2019
todo
userposts

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



Create index pattern

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

Step 1 of 2: Define index pattern

Index pattern

medicalvisits-26.01.2019

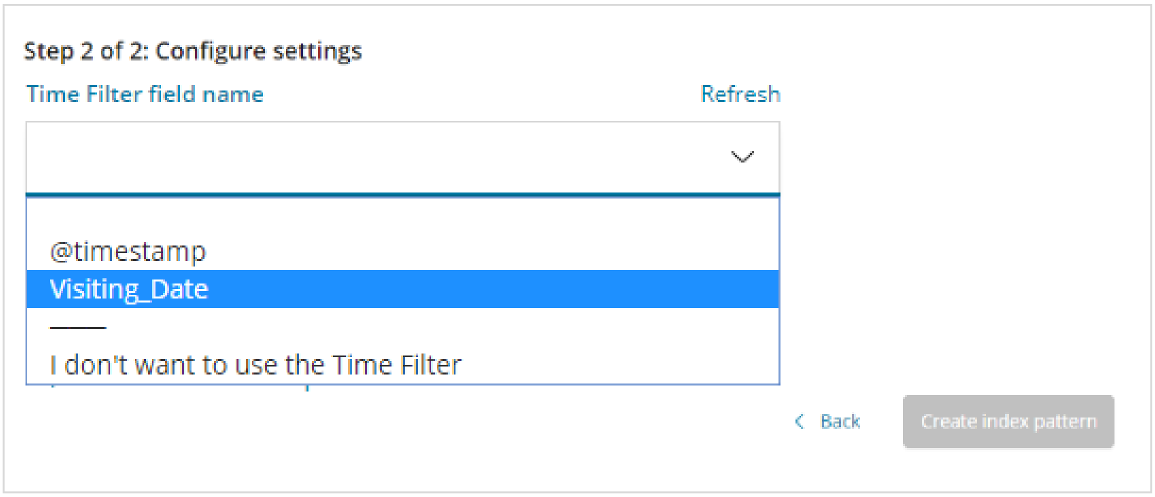
You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, ", <, >, |.

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.

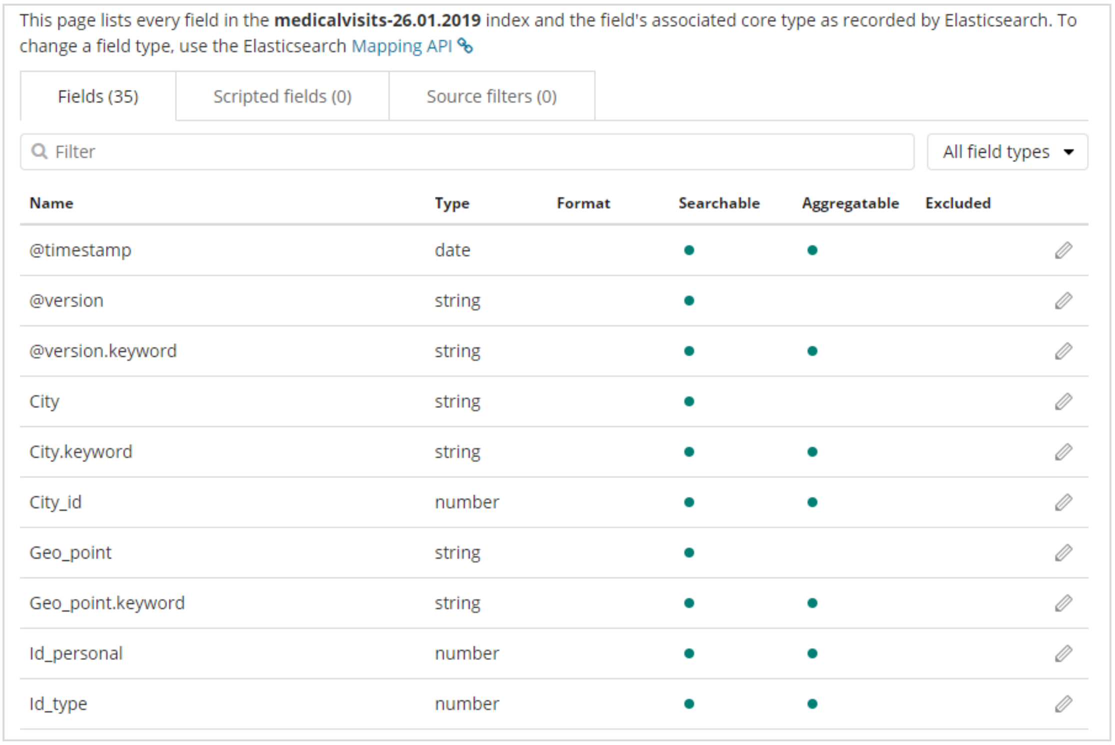


Click ***Create index pattern*** button to create the index. Once done it will display all the fields present in your index medicalvisits-26.01.2019 as shown below:

We have following fields in the index medicalvisits-26.01.2019 :

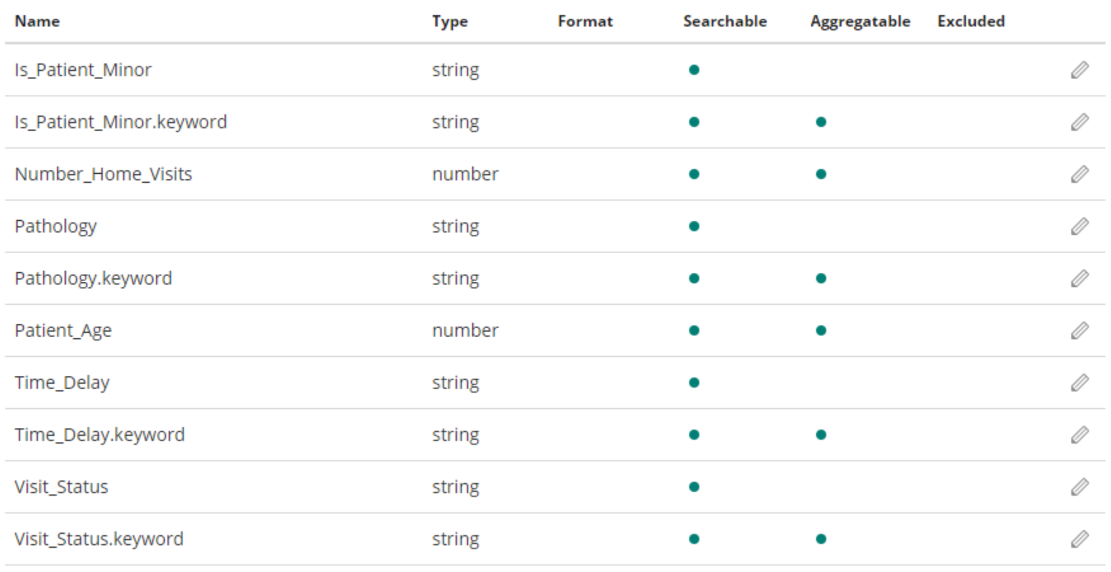
```
["Visit_Status","Time_Delay","City","City_id","Patient_Age","Zipcode","Latitude
","Longitude","Pathology","Visiting_Date","Id_type","Id_personal","Number_Home_
Visits","Is_Patient_Minor","Geo_point"].
```

The index has all the data for home medical visits. There are some additional fields added by elasticsearch when inserted from logstash.

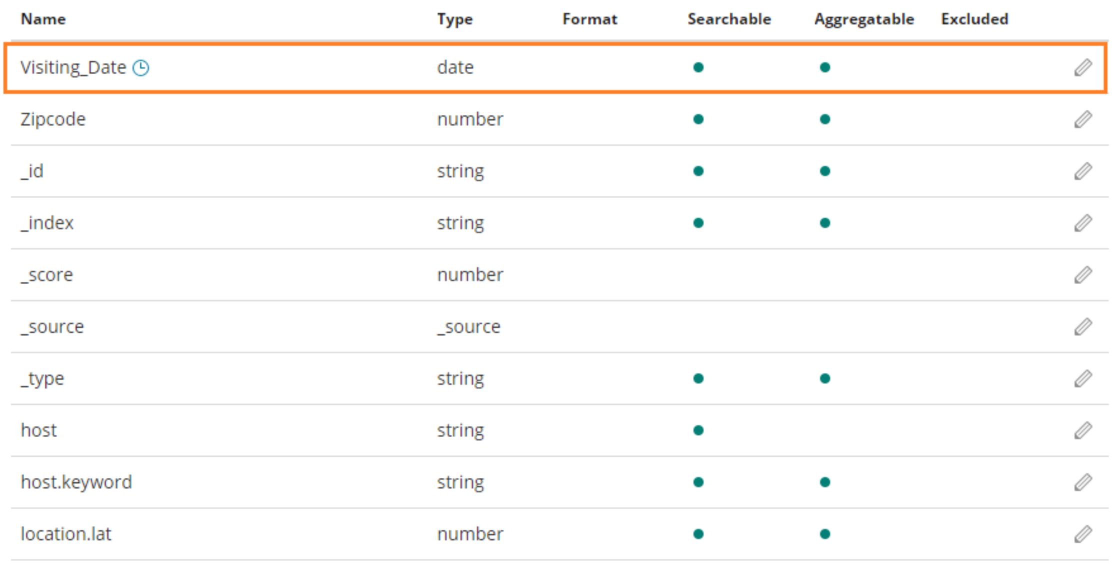


This page lists every field in the **medicalvisits-26.01.2019** index and the field's associated core type as recorded by Elasticsearch. To change a field type, use the Elasticsearch Mapping API

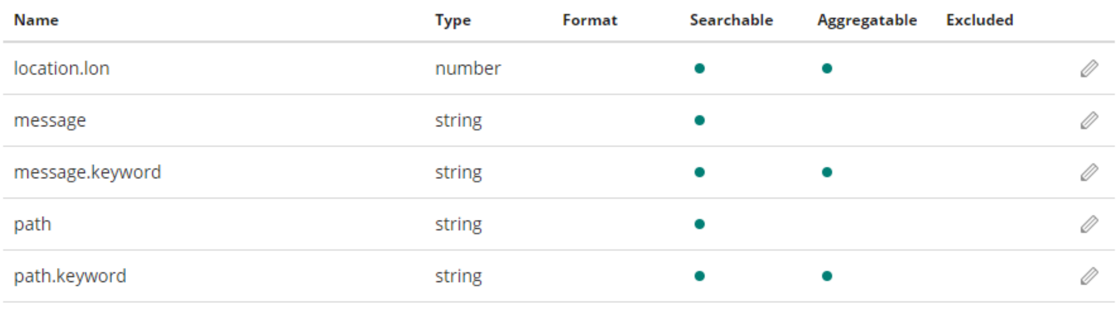
Name	Type	Format	Searchable	Aggregatable	Excluded
@timestamp	date		●	●	
@version	string		●		
@version.keyword	string		●	●	
City	string		●		
City.keyword	string		●	●	
City_id	number		●	●	
Geo_point	string		●		
Geo_point.keyword	string		●	●	
Id_personal	number		●	●	
Id_type	number		●	●	



Name	Type	Format	Searchable	Aggregatable	Excluded
Is_Patient_Minor	string		●		
Is_Patient_Minor.keyword	string		●	●	
Number_Home_Visits	number		●	●	
Pathology	string		●		
Pathology.keyword	string		●	●	
Patient_Age	number		●	●	
Time_Delay	string		●		
Time_Delay.keyword	string		●	●	
Visit_Status	string		●		
Visit_Status.keyword	string		●	●	



Name	Type	Format	Searchable	Aggregatable	Excluded
Visiting_Date	date		●	●	
Zipcode	number		●	●	
_id	string		●	●	
_index	string		●	●	
_score	number				
_source	_source				
_type	string		●	●	
host	string		●		
host.keyword	string		●	●	
location.lat	number		●	●	



Name	Type	Format	Searchable	Aggregatable	Excluded
location.lon	number		●	●	
message	string		●		
message.keyword	string		●	●	
path	string		●		
path.keyword	string		●	●	

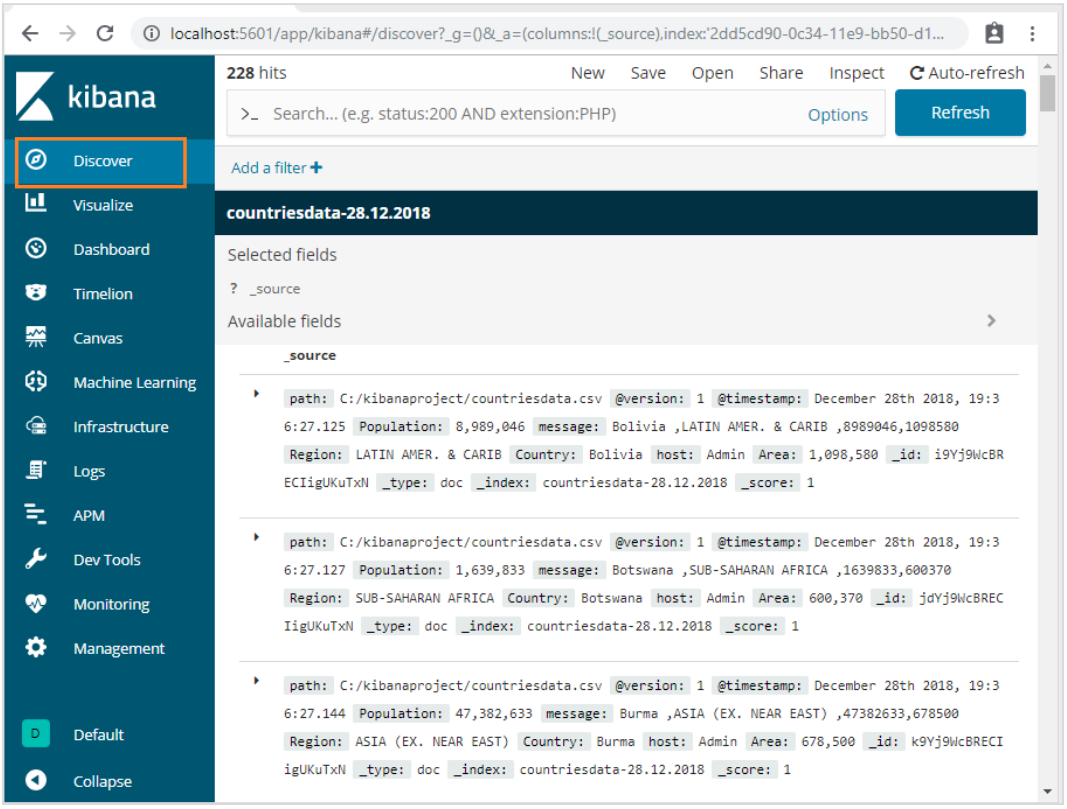
6. Kibana — Discover

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

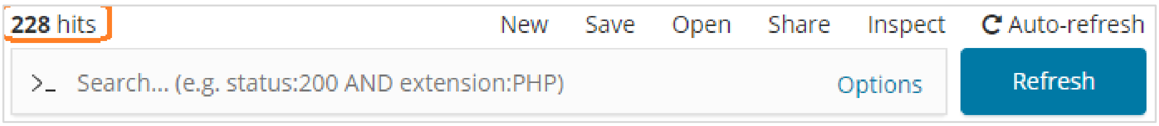
Select Discover on the left side menu as shown below:



```
_source

path: C:/kibanaproject/countriesdata.csv @version: 1 @timestamp: December 28th 2018, 19:3
6:27.125 Population: 8,989,046 message: Bolivia ,LATIN AMER. & CARIB ,8989046,1098580
Region: LATIN AMER. & CARIB Country: Bolivia host: Admin Area: 1,098,580 _id: i9Yj9WcBR
ECIigUKuTxN _type: doc _index: countriesdata-28.12.2018 _score: 1

path: C:/kibanaproject/countriesdata.csv @version: 1 @timestamp: December 28th 2018, 19:3
6:27.127 Population: 1,639,833 message: Botswana ,SUB-SAHARAN AFRICA ,1639833,600370
Region: SUB-SAHARAN AFRICA Country: Botswana host: Admin Area: 600,370 _id: jdYj9WcBREC
IigUKuTxN _type: doc _index: countriesdata-28.12.2018 _score: 1

path: C:/kibanaproject/countriesdata.csv @version: 1 @timestamp: December 28th 2018, 19:3
6:27.144 Population: 47,382,633 message: Burma ,ASIA (EX. NEAR EAST) ,47382633,678500
Region: ASIA (EX. NEAR EAST) Country: Burma host: Admin Area: 678,500 _id: k9Yj9WcBRECI
igUKuTxN _type: doc _index: countriesdata-28.12.2018 _score: 1
```

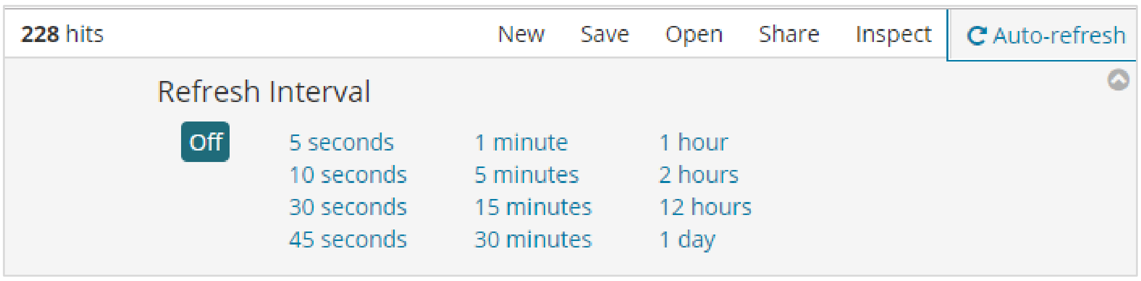
On the right side, it displays the details of the data available in ***countriesdata-28.12.2018*** index we created in previous chapter.

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.

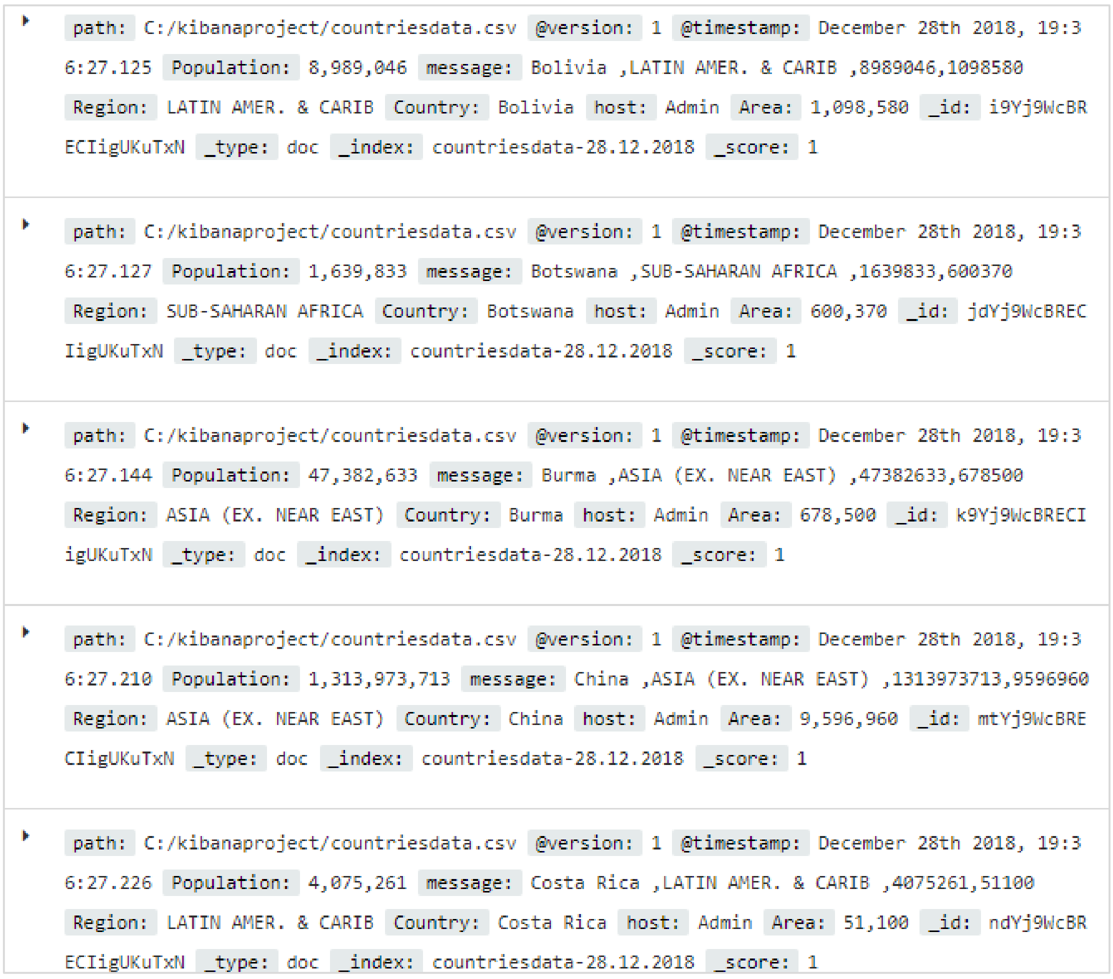
If you click Auto-refresh, it will display the screen as shown below:



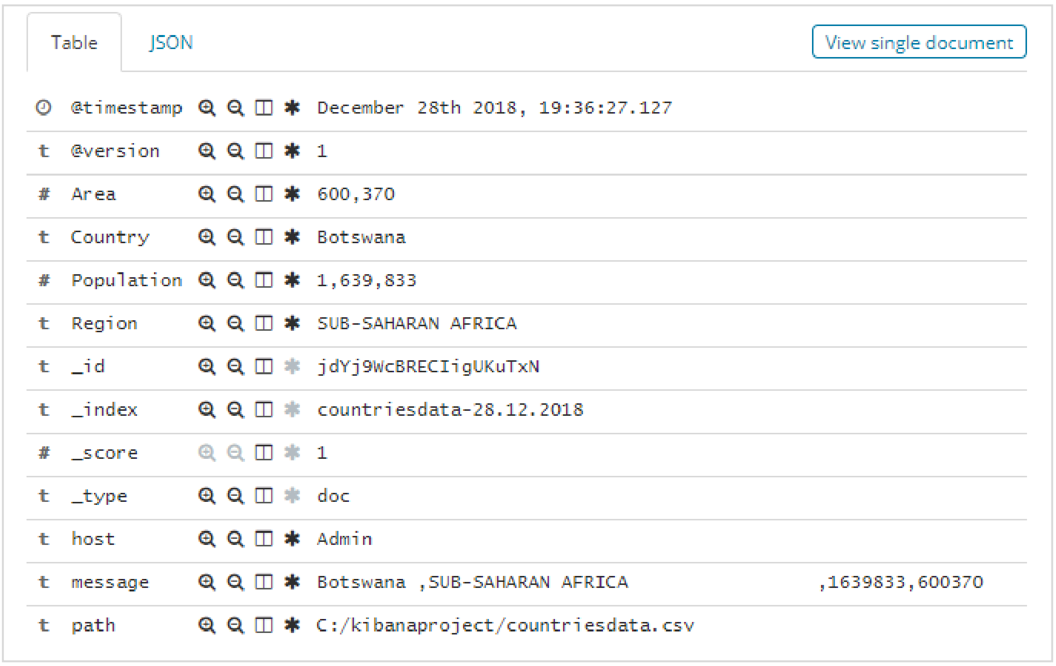
You can set the auto-refresh interval by clicking on the seconds, minutes or hour from above. Kibana will auto-refresh the screen and get fresh data after every interval timer you set.

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

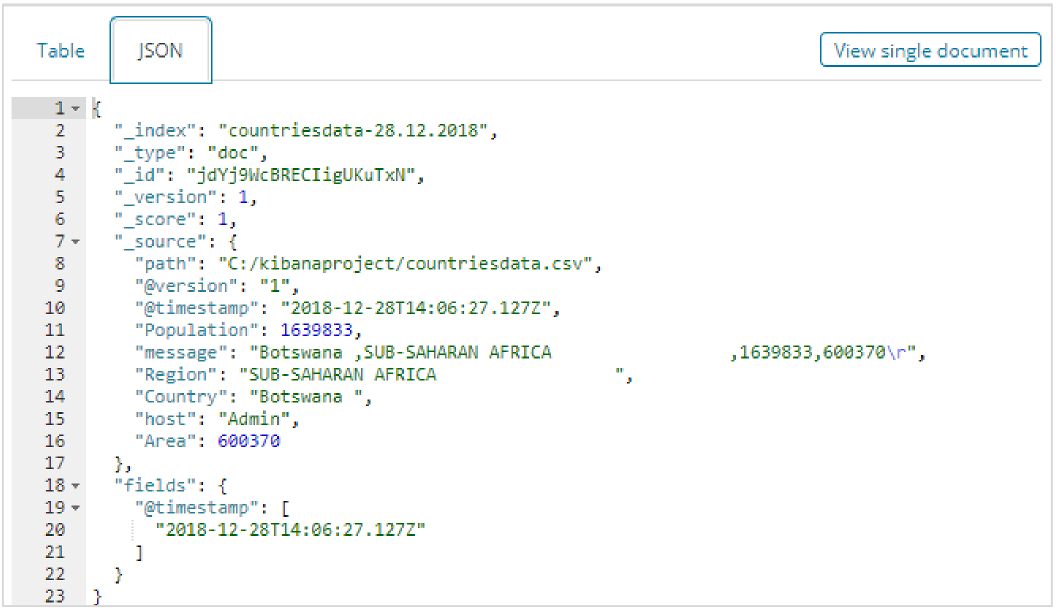


```
path: C:/kibanaproject/countriesdata.csv @version: 1 @timestamp: December 28th 2018, 19:36:27.125 Population: 8,989,046 message: Bolivia ,LATIN AMER. & CARIB ,8989046,1098580 Region: LATIN AMER. & CARIB Country: Bolivia host: Admin Area: 1,098,580 _id: i9Yj9WcBRECIigUKuTxN _type: doc _index: countriesdata-28.12.2018 _score: 1

path: C:/kibanaproject/countriesdata.csv @version: 1 @timestamp: December 28th 2018, 19:36:27.127 Population: 1,639,833 message: Botswana ,SUB-SAHARAN AFRICA ,1639833,600370 Region: SUB-SAHARAN AFRICA Country: Botswana host: Admin Area: 600,370 _id: jdYj9WcBRECIigUKuTxN _type: doc _index: countriesdata-28.12.2018 _score: 1

path: C:/kibanaproject/countriesdata.csv @version: 1 @timestamp: December 28th 2018, 19:36:27.144 Population: 47,382,633 message: Burma ,ASIA (EX. NEAR EAST) ,47382633,678500 Region: ASIA (EX. NEAR EAST) Country: Burma host: Admin Area: 678,500 _id: k9Yj9WcBRECIigUKuTxN _type: doc _index: countriesdata-28.12.2018 _score: 1

path: C:/kibanaproject/countriesdata.csv @version: 1 @timestamp: December 28th 2018, 19:36:27.210 Population: 1,313,973,713 message: China ,ASIA (EX. NEAR EAST) ,1313973713,9596960 Region: ASIA (EX. NEAR EAST) Country: China host: Admin Area: 9,596,960 _id: mtYj9WcBRECIigUKuTxN _type: doc _index: countriesdata-28.12.2018 _score: 1

path: C:/kibanaproject/countriesdata.csv @version: 1 @timestamp: December 28th 2018, 19:36:27.226 Population: 4,075,261 message: Costa Rica ,LATIN AMER. & CARIB ,4075261,51100 Region: LATIN AMER. & CARIB Country: Costa Rica host: Admin Area: 51,100 _id: ndYj9WcBRECIigUKuTxN _type: doc _index: countriesdata-28.12.2018 _score: 1
```



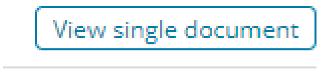
⏲	@timestamp	December 28th 2018, 19:36:27.127
t	@version	1
#	Area	600,370
t	Country	Botswana
#	Population	1,639,833
t	Region	SUB-SAHARAN AFRICA
t	_id	jdYj9WcBRECIigUKuTxN
t	_index	countriesdata-28.12.2018
#	_score	1
t	_type	doc
t	host	Admin
t	message	Botswana ,SUB-SAHARAN AFRICA ,1639833,600370
t	path	C:/kibanaproject/countriesdata.csv

JSON Format



```
{
  "_index": "countriesdata-28.12.2018",
  "_type": "doc",
  "_id": "jdYj9WcBRECIigUKuTxN",
  "_version": 1,
  "_score": 1,
  "_source": {
    "path": "C:/kibanaproject/countriesdata.csv",
    "@version": "1",
    "@timestamp": "2018-12-28T14:06:27.127Z",
    "Population": 1639833,
    "message": "Botswana ,SUB-SAHARAN AFRICA                 ,1639833,600370\r",
    "Region": "SUB-SAHARAN AFRICA             ",
    "Country": "Botswana ",
    "host": "Admin",
    "Area": 600370
  },
  "fields": {
    "@timestamp": [
      "2018-12-28T14:06:27.127Z"
    ]
  }
}
```

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:

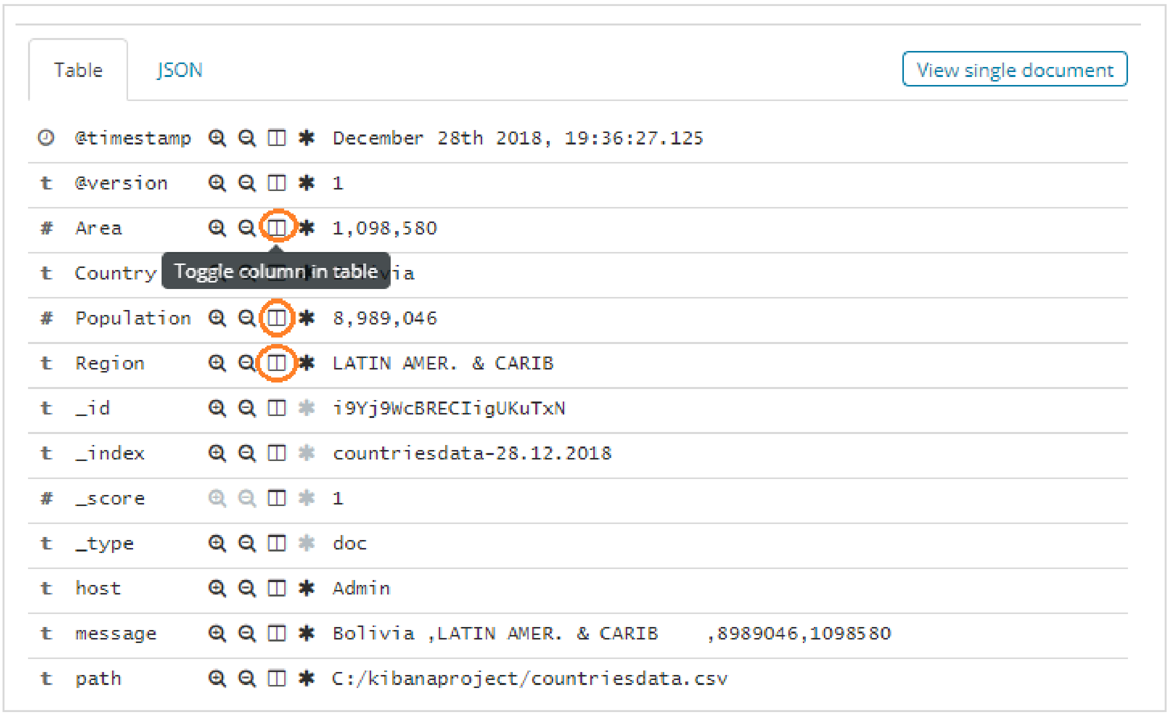
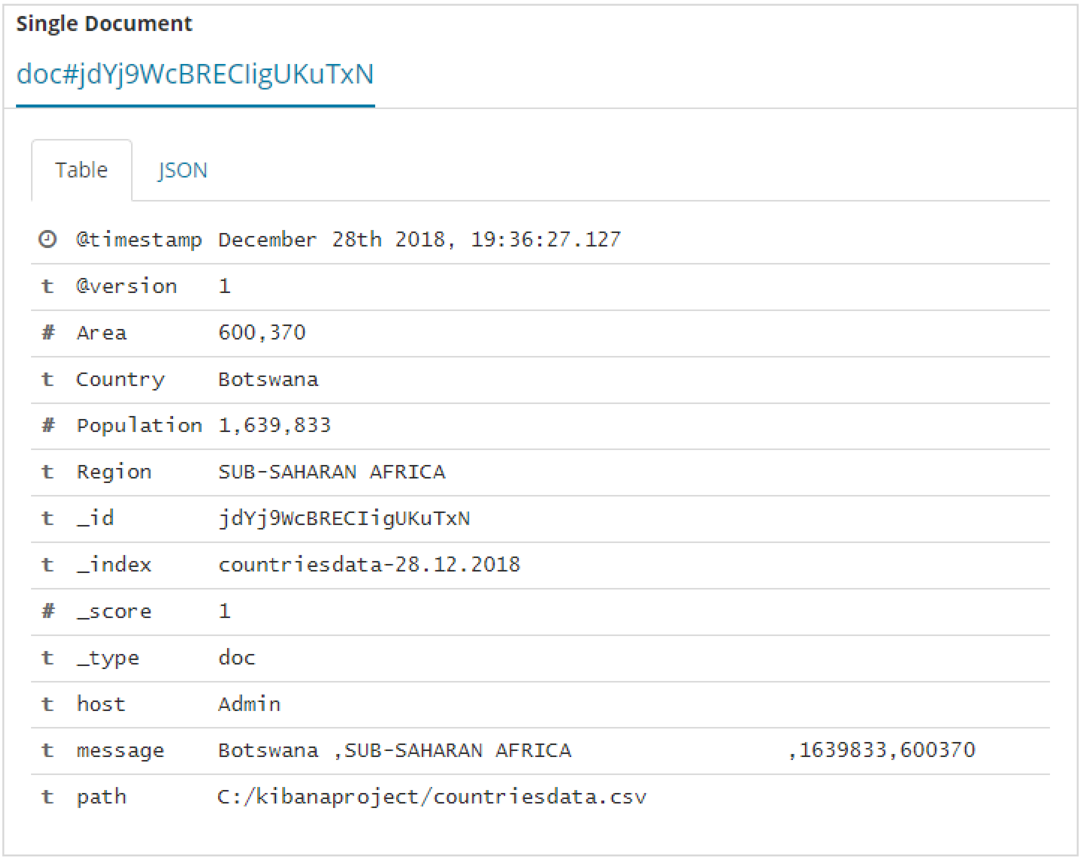


Table | JSON | View single document

⊙	@timestamp		December 28th 2018, 19:36:27.125
t	@version		1
#	Area		1,098,580
t	Country	Toggle column in table	ia
#	Population		8,989,046
t	Region		LATIN AMER. & CARIB
t	_id		i9Yj9WcBRECIigUKuTxN
t	_index		countriesdata-28.12.2018
#	_score		1
t	_type		doc
t	host		Admin
t	message		Bolivia ,LATIN AMER. & CARIB ,8989046,1098580
t	path		C:/kibanaproject/countriesdata.csv



Single Document

doc#jdYj9WcBRECIigUKuTxN

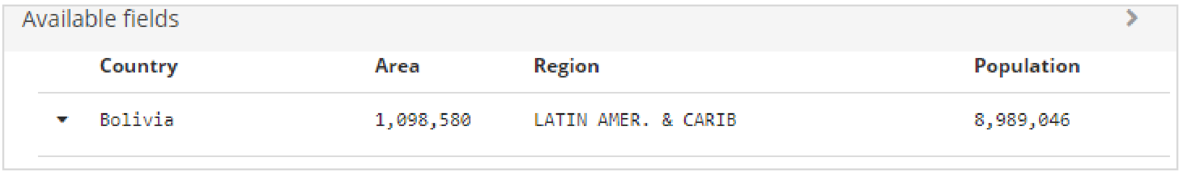
Table | JSON

⊙	@timestamp	December 28th 2018, 19:36:27.127
t	@version	1
#	Area	600,370
t	Country	Botswana
#	Population	1,639,833
t	Region	SUB-SAHARAN AFRICA
t	_id	jdYj9WcBRECIigUKuTxN
t	_index	countriesdata-28.12.2018
#	_score	1
t	_type	doc
t	host	Admin
t	message	Botswana ,SUB-SAHARAN AFRICA ,1639833,600370
t	path	C:/kibanaproject/countriesdata.csv

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:

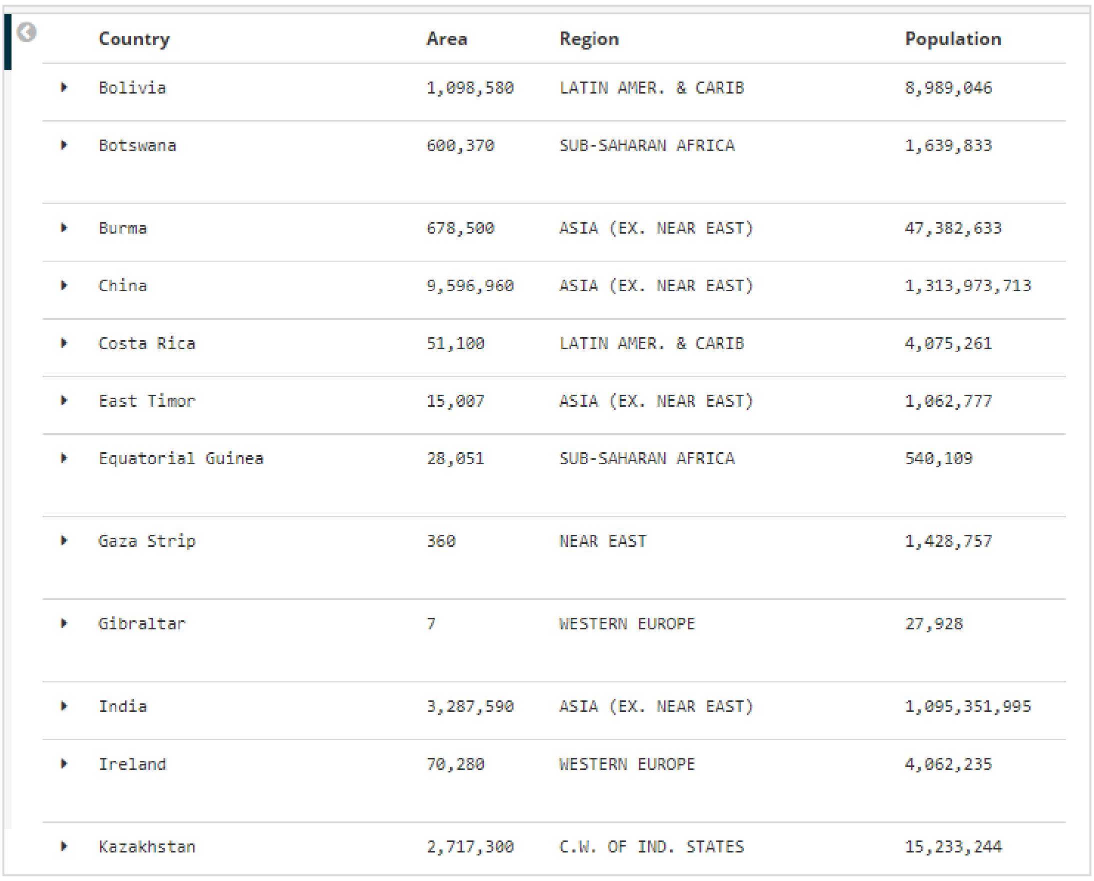
Click on Toggle column in table option available for each and you will notice the data being shown in table format:



Available fields

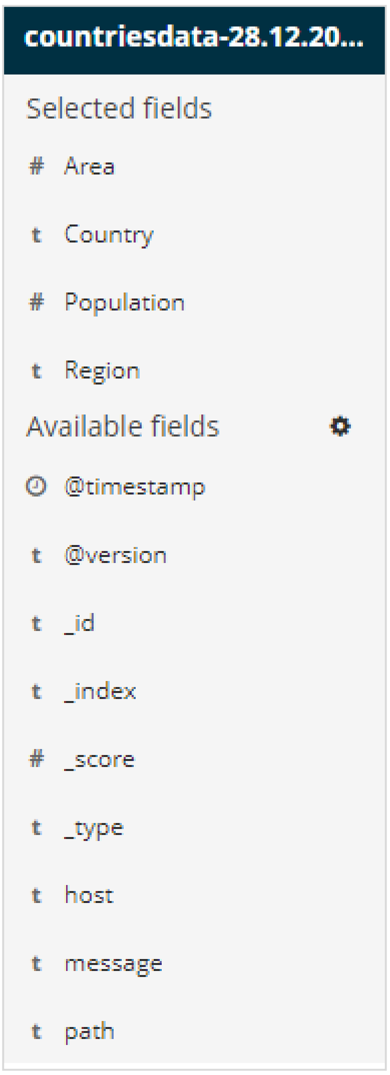
	Country	Area	Region	Population
▾	Bolivia	1,098,580	LATIN AMER. & CARIB	8,989,046

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.

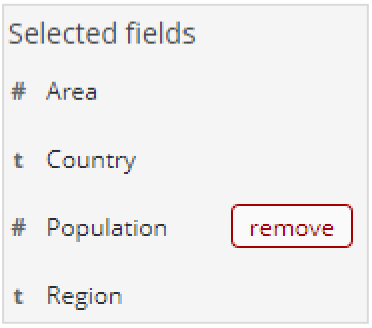


	Country	Area	Region	Population
▸	Bolivia	1,098,580	LATIN AMER. & CARIB	8,989,046
▸	Botswana	600,370	SUB-SAHARAN AFRICA	1,639,833
▸	Burma	678,500	ASIA (EX. NEAR EAST)	47,382,633
▸	China	9,596,960	ASIA (EX. NEAR EAST)	1,313,973,713
▸	Costa Rica	51,100	LATIN AMER. & CARIB	4,075,261
▸	East Timor	15,007	ASIA (EX. NEAR EAST)	1,062,777
▸	Equatorial Guinea	28,051	SUB-SAHARAN AFRICA	540,109
▸	Gaza Strip	360	NEAR EAST	1,428,757
▸	Gibraltar	7	WESTERN EUROPE	27,928
▸	India	3,287,590	ASIA (EX. NEAR EAST)	1,095,351,995
▸	Ireland	70,280	WESTERN EUROPE	4,062,235
▸	Kazakhstan	2,717,300	C.W. OF IND. STATES	15,233,244

The fields we selected are displayed on the left side of the screen as shown below:



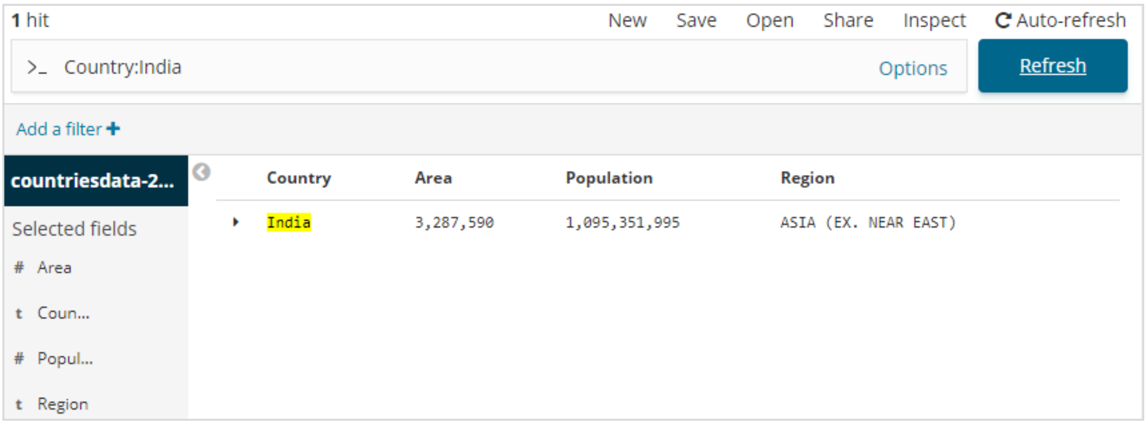
Observe that there are 2 options: *Selected fields* and *Available fields*. The fields we have selected to show in tabular format are a part of selected fields. In case you want to remove any field you can do so by clicking the remove button which will be seen across the field name in selected field 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*.

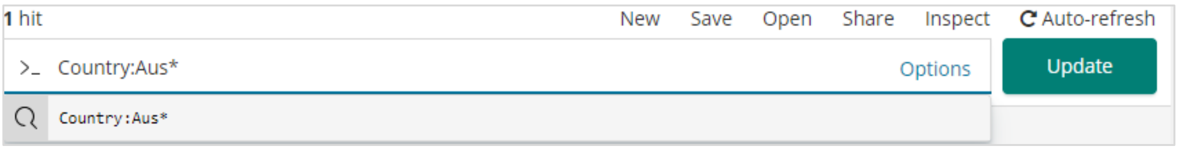
We have a search option available in Discover, which we can use to search for data inside the index. Let us try examples related to search option here:

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

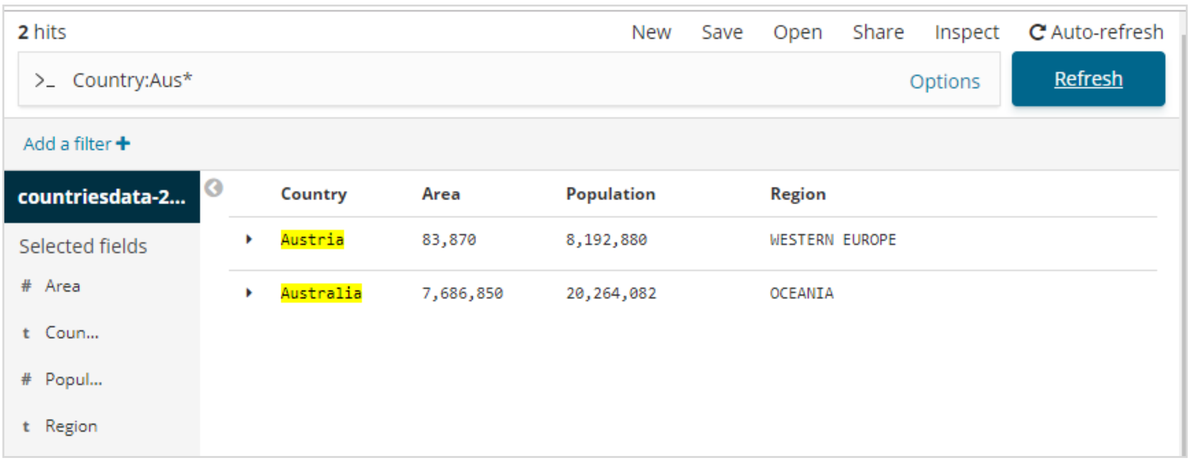


Country	Area	Population	Region
India	3,287,590	1,095,351,995	ASIA (EX. NEAR EAST)

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:

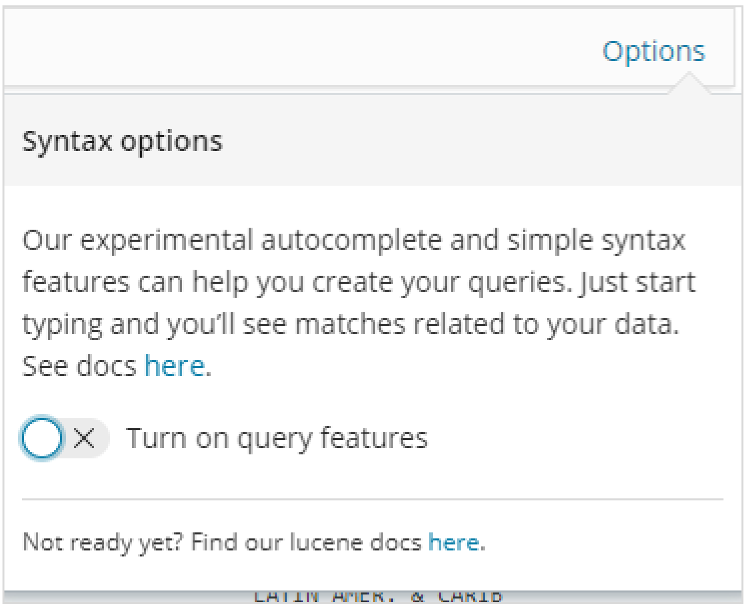


Click Update to see the results



Country	Area	Population	Region
Austria	83,870	8,192,880	WESTERN EUROPE
Australia	7,686,850	20,264,082	OCEANIA

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.


Options

Syntax options

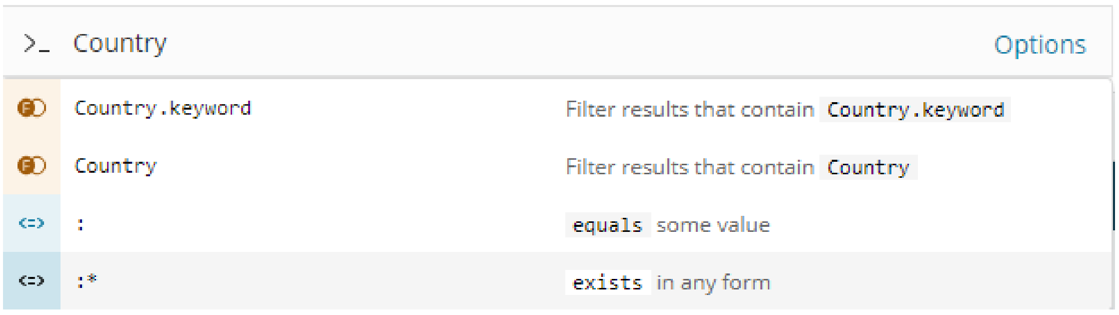
Our experimental autocomplete and simple syntax features can help you create your queries. Just start typing and you'll see matches related to your data. See docs here.

Turn on query features

Not ready yet? Find our lucene docs here.

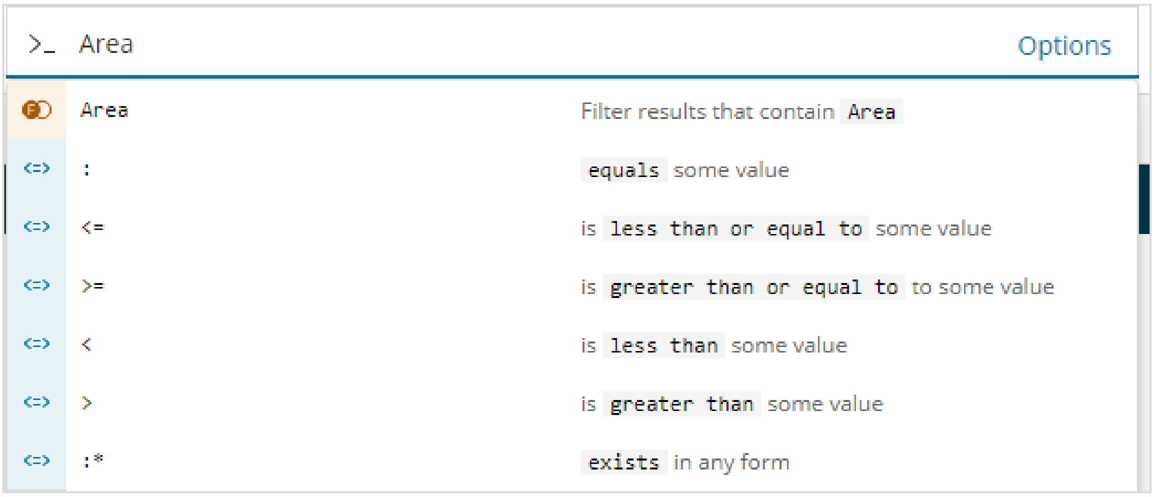
Turn on query features and type the field name in search, it will display the options available for that field.

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


>_ Country Options

F	Country.keyword	Filter results that contain `Country.keyword`
F	Country	Filter results that contain `Country`
<=>	:	`equals` some value
<=>	:*	`exists` in any form

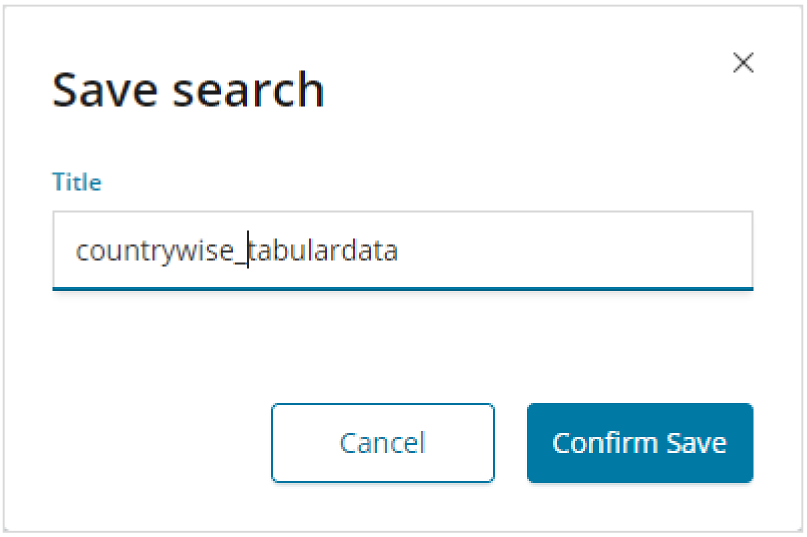
Similarly, Area is a Number field and it displays following options for Number field:



Area	Filter results that contain Area
:	equals some value
<=	is less than or equal to some value
>=	is greater than or equal to to some value
<	is less than some value
>	is greater than some value
:*	exists in any form

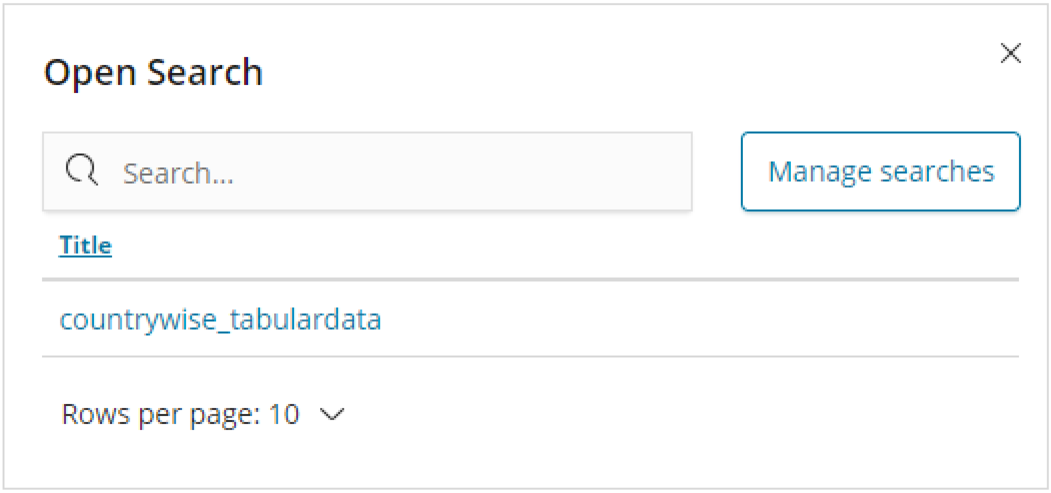
You can try out different combination and filter the data as per your choice in Discover field. The data inside the Discover tab can be saved using the Save button, so that you can use it for future purpose.

To save the data inside discover click on the save button on top right corner as shown below:

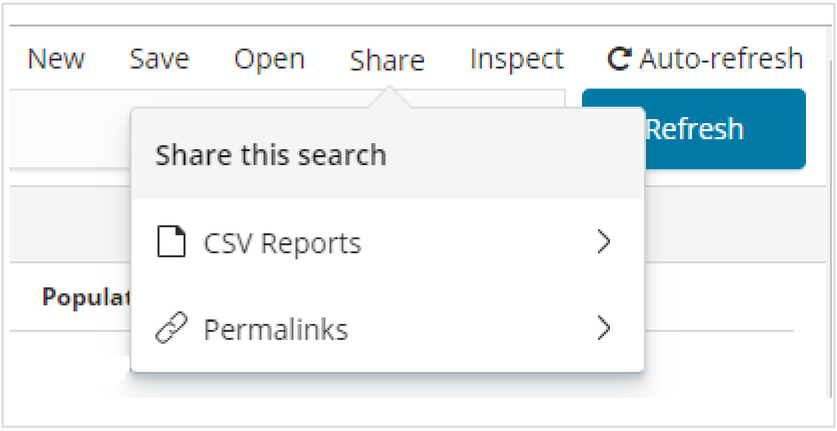


Title: countrywise_tabulardata

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:

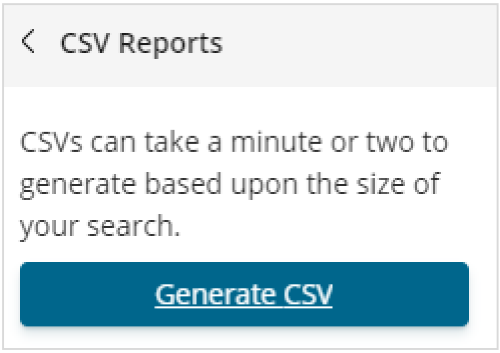


You can also share the data with others using the Share button available on top right corner. If you click it, you can find sharing options as shown below:



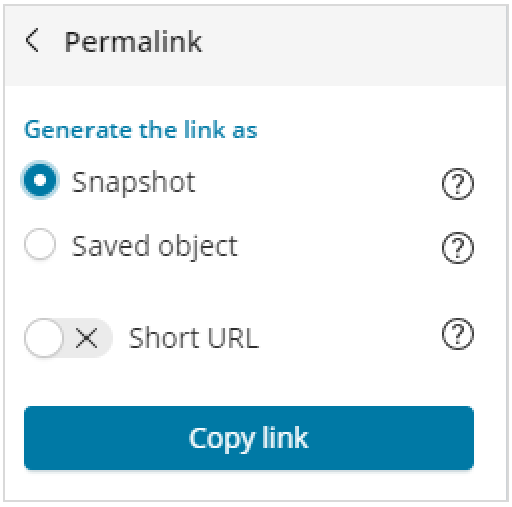
You can share it using CSV Reports or in the form of Permalinks.

The option available onclick on CSV Reports are:


CSV Reports
CSVs can take a minute or two to generate based upon the size of your search.
Generate CSV

Click Generate CSV to get the report to be shared with others.

The option available onclick of Permalinks are as follows:



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.

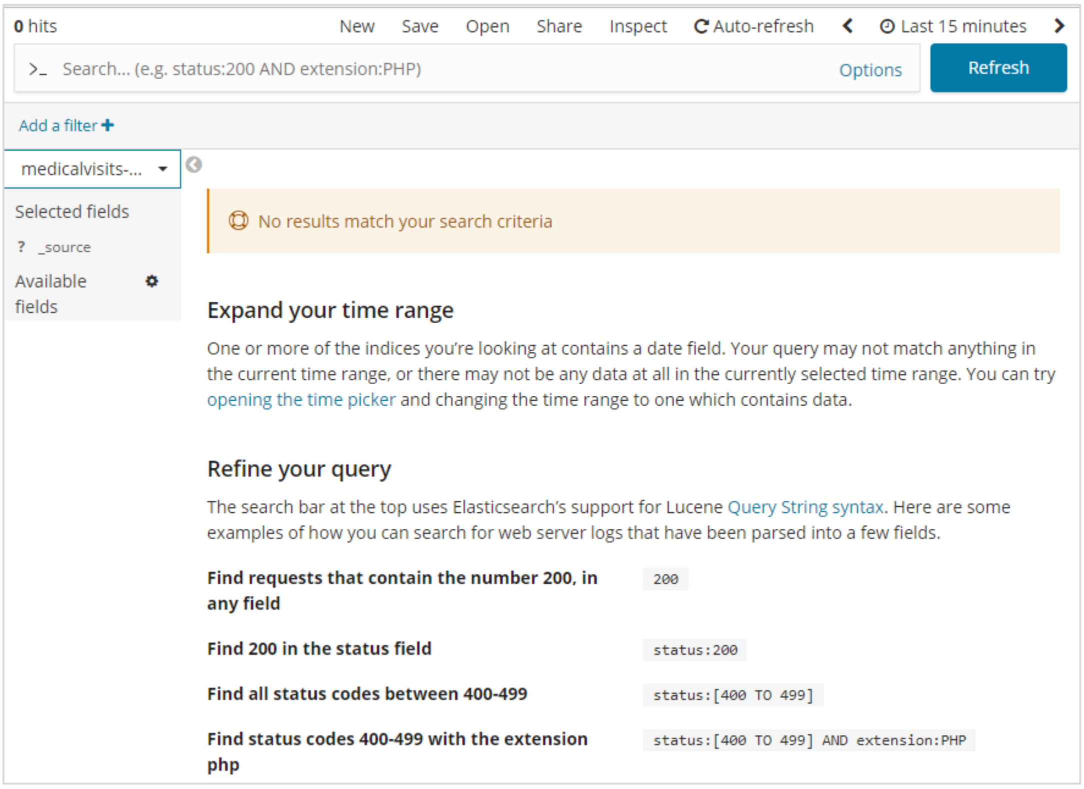
In general, Kibana gives longer links; there is option at the bottom to get Short URL. A sample short url for Snapshot and Saved object are shown here:

Snapshot: http://localhost:5601/goto/309a983483fccd423950cfb708fabfa5
Saved Object :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***



No results match your search criteria

Expand your time range

One or more of the indices you're looking at contains a date field. Your query may not match anything in the current time range, or there may not be any data at all in the currently selected time range. You can try opening the time picker and changing the time range to one which contains data.

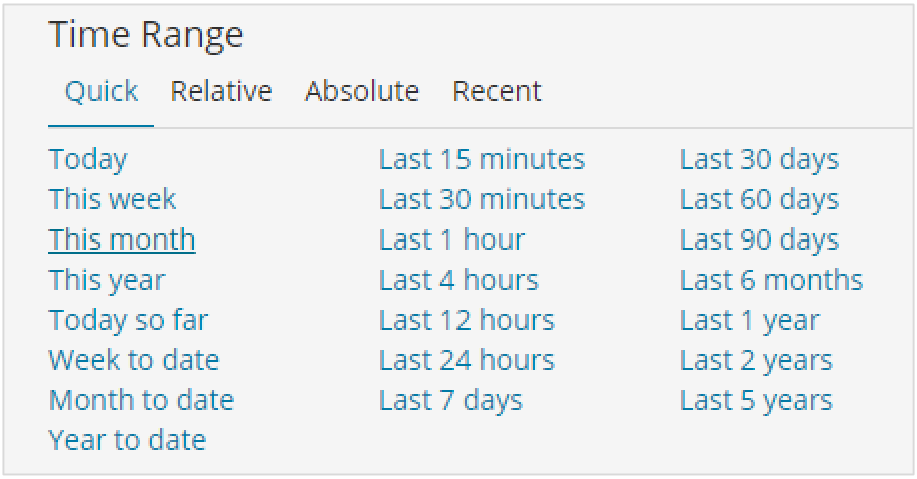
Refine your query

The search bar at the top uses Elasticsearch's support for Lucene Query String syntax. Here are some examples of how you can search for web server logs that have been parsed into a few fields.

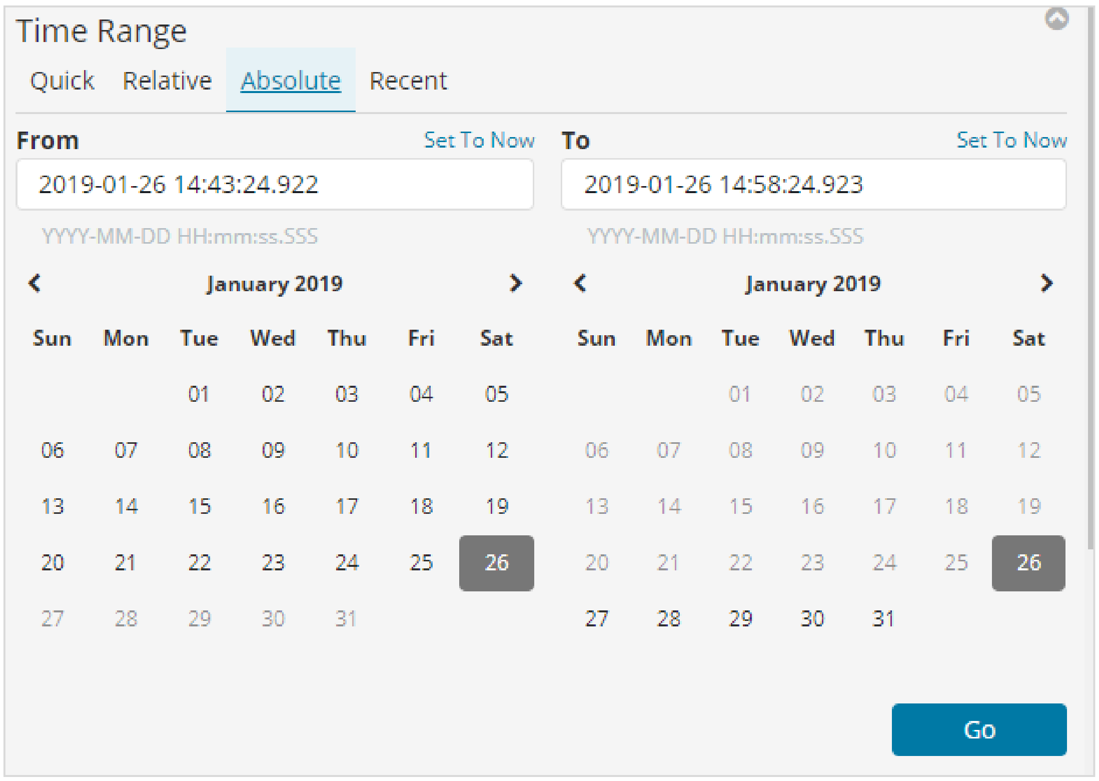
Find requests that contain the number 200, in any field	`200`
Find 200 in the status field	`status:200`
Find all status codes between 400-499	`status:[400 TO 499]`
Find status codes 400-499 with the extension php	`status:[400 TO 499] AND extension:PHP`

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.

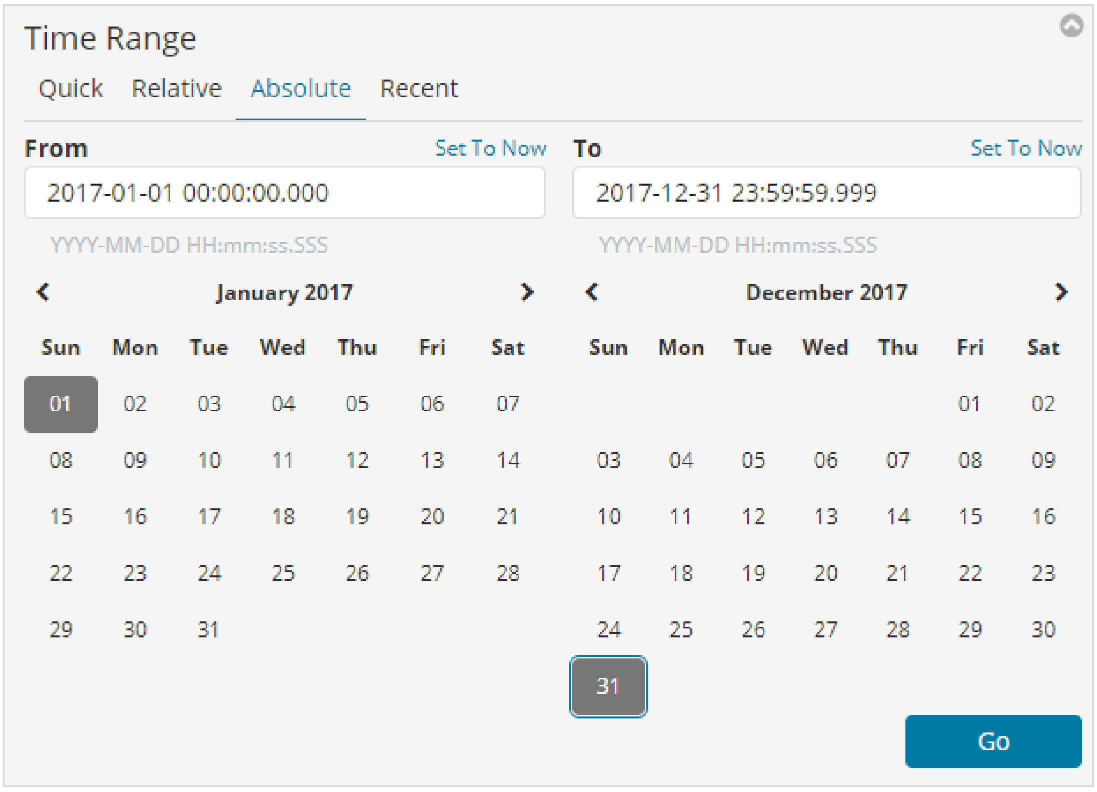
Change the time range as shown below:



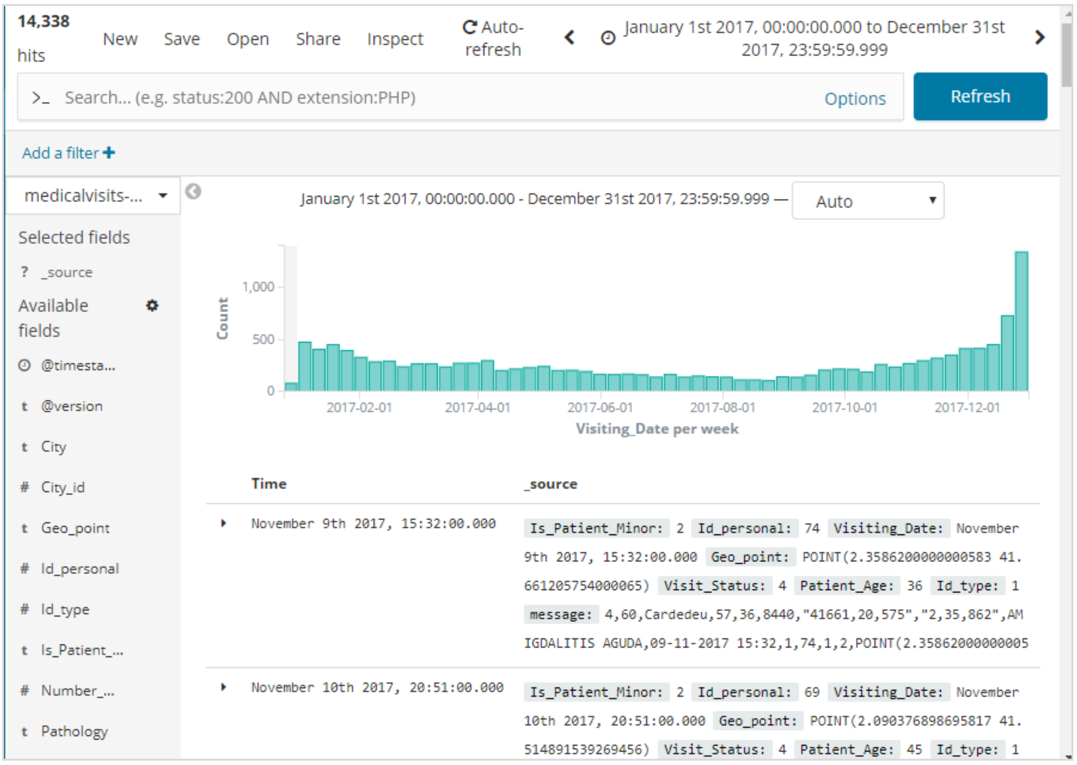
Click Absolute tab.



Select the date From: 1st Jan 2017 and To : 31st Dec2017 as we will analyze data for year 2017.

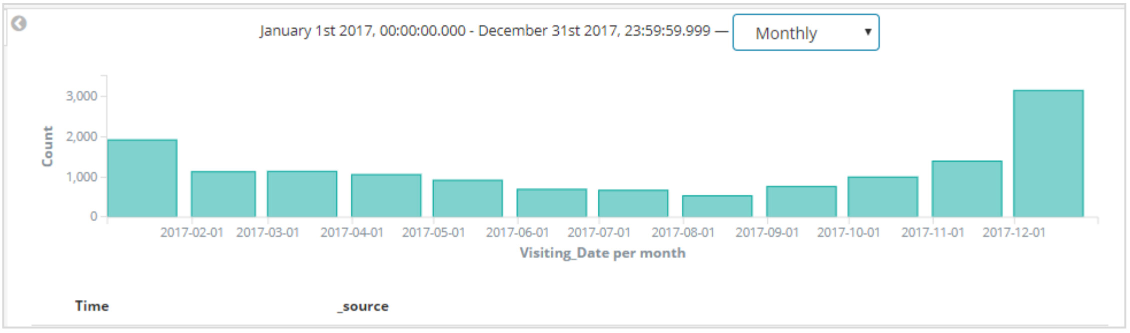


Click the Go button to add the timerange. It will display you the data and bar chart as follows:

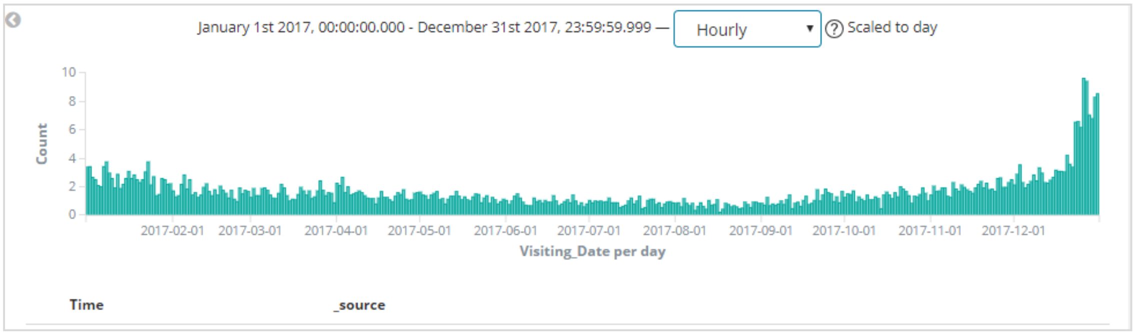


```
Time                              _source

November 9th 2017, 15:32:00.000   Is_Patient_Minor: 2 Id_personal: 74 Visiting_Date: November
                                  9th 2017, 15:32:00.000 Geo_point: POINT(2.358620000000583 41.
                                  661205754000065) Visit_Status: 4 Patient_Age: 36 Id_type: 1
                                  message: 4,60,Cardedeu,57,36,8440,"41661,20,575","2,35,862",AM
                                  IGDALITIS AGUDA,09-11-2017 15:32,1,74,1,2,POINT(2.35862000000005

November 10th 2017, 20:51:00.000  Is_Patient_Minor: 2 Id_personal: 69 Visiting_Date: November
                                  10th 2017, 20:51:00.000 Geo_point: POINT(2.090376898695817 41.
                                  514891539269456) Visit_Status: 4 Patient_Age: 45 Id_type: 1
```

This is the monthly data for the year 2017:

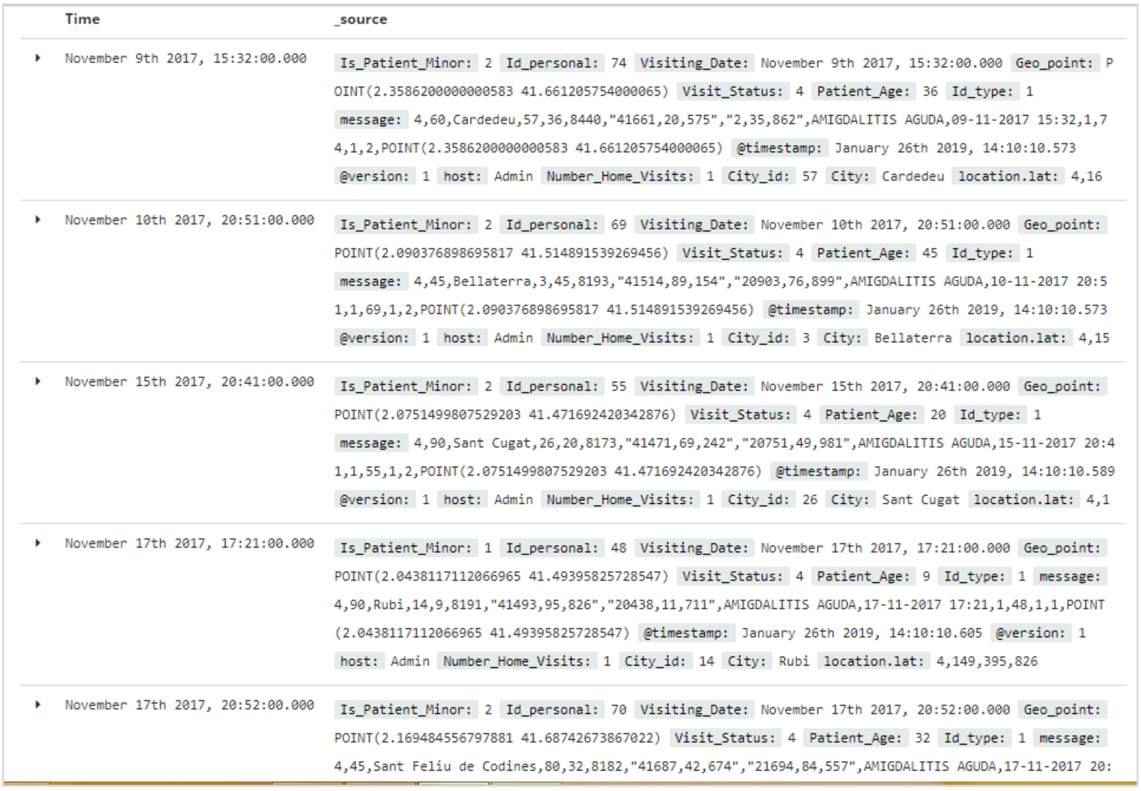


Since we also have the time stored along with date, we can filter the data on hours and minutes too.



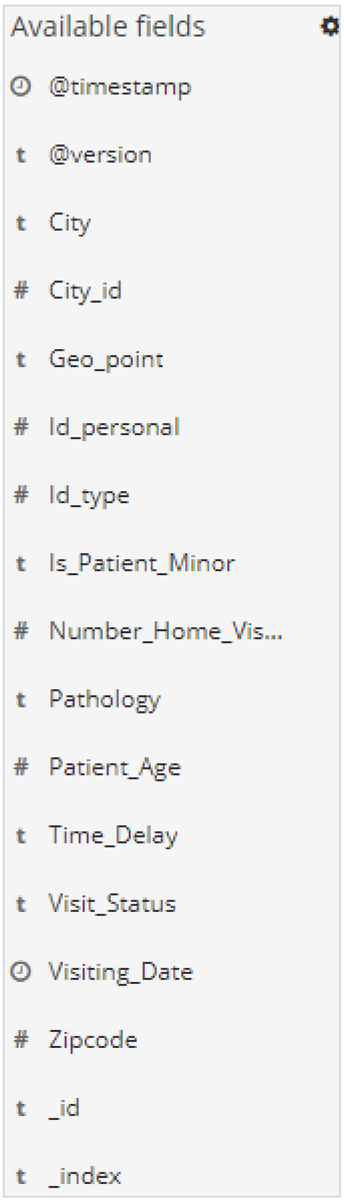
The figure shown above displays the hourly data for the year 2017.

Here the fields displayed from the index: medicalvisits-26.01.2019

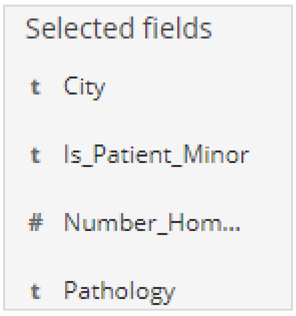


Time	_source
November 9th 2017, 15:32:00.000	Is_Patient_Minor: 2 Id_personal: 74 Visiting_Date: November 9th 2017, 15:32:00.000 Geo_point: POINT(2.3586200000000583 41.661205754000065) Visit_Status: 4 Patient_Age: 36 Id_type: 1 message: 4,60,Cardedeu,57,36,8440,"41661,20,575","2,35,862",AMIGDALITIS AGUDA,09-11-2017 15:32,1,74,1,2,POINT(2.3586200000000583 41.661205754000065) @timestamp: January 26th 2019, 14:10:10.573 @version: 1 host: Admin Number_Home_Visits: 1 City_id: 57 City: Cardedeu location.lat: 4,16
November 10th 2017, 20:51:00.000	Is_Patient_Minor: 2 Id_personal: 69 Visiting_Date: November 10th 2017, 20:51:00.000 Geo_point: POINT(2.090376898695817 41.514891539269456) Visit_Status: 4 Patient_Age: 45 Id_type: 1 message: 4,45,Bellaterra,3,45,8193,"41514,89,154","20903,76,899",AMIGDALITIS AGUDA,10-11-2017 20:51,1,69,1,2,POINT(2.090376898695817 41.514891539269456) @timestamp: January 26th 2019, 14:10:10.573 @version: 1 host: Admin Number_Home_Visits: 1 City_id: 3 City: Bellaterra location.lat: 4,15
November 15th 2017, 20:41:00.000	Is_Patient_Minor: 2 Id_personal: 55 Visiting_Date: November 15th 2017, 20:41:00.000 Geo_point: POINT(2.0751499807529203 41.471692420342876) Visit_Status: 4 Patient_Age: 20 Id_type: 1 message: 4,90,Sant Cugat,26,20,8173,"41471,69,242","20751,49,981",AMIGDALITIS AGUDA,15-11-2017 20:41,1,55,1,2,POINT(2.0751499807529203 41.471692420342876) @timestamp: January 26th 2019, 14:10:10.589 @version: 1 host: Admin Number_Home_Visits: 1 City_id: 26 City: Sant Cugat location.lat: 4,1
November 17th 2017, 17:21:00.000	Is_Patient_Minor: 1 Id_personal: 48 Visiting_Date: November 17th 2017, 17:21:00.000 Geo_point: POINT(2.0438117112066965 41.49395825728547) Visit_Status: 4 Patient_Age: 9 Id_type: 1 message: 4,90,Rubi,14,9,8191,"41493,95,826","20438,11,711",AMIGDALITIS AGUDA,17-11-2017 17:21,1,48,1,1,POINT (2.0438117112066965 41.49395825728547) @timestamp: January 26th 2019, 14:10:10.605 @version: 1 host: Admin Number_Home_Visits: 1 City_id: 14 City: Rubi location.lat: 4,149,395,826
November 17th 2017, 20:52:00.000	Is_Patient_Minor: 2 Id_personal: 70 Visiting_Date: November 17th 2017, 20:52:00.000 Geo_point: POINT(2.169484556797881 41.68742673867022) Visit_Status: 4 Patient_Age: 32 Id_type: 1 message: 4,45,Sant Feliu de Codines,80,32,8182,"41687,42,674","21694,84,557",AMIGDALITIS AGUDA,17-11-2017 20:

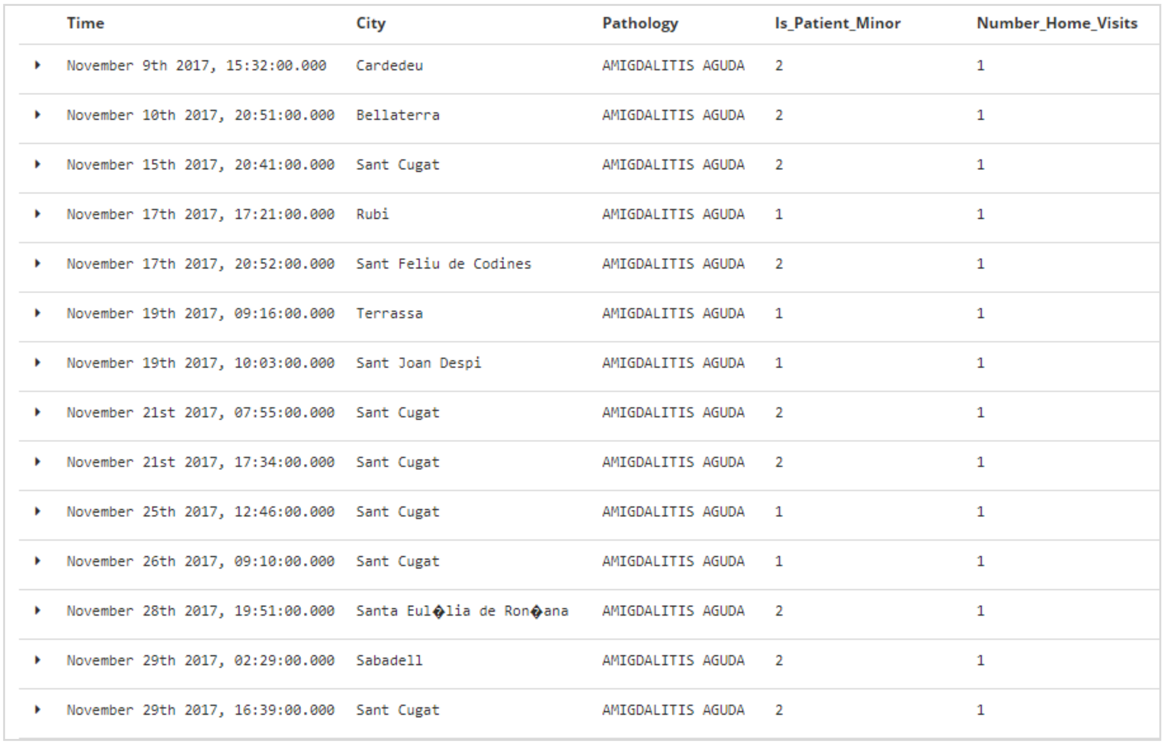
We have the available fields on left side as shown below:



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:



The tabular data for above fields is shown here:



	Time	City	Pathology	Is_Patient_Minor	Number_Home_Visits
▸	November 9th 2017, 15:32:00.000	Cardedeu	AMIGDALITIS AGUDA	2	1
▸	November 10th 2017, 20:51:00.000	Bellaterra	AMIGDALITIS AGUDA	2	1
▸	November 15th 2017, 20:41:00.000	Sant Cugat	AMIGDALITIS AGUDA	2	1
▸	November 17th 2017, 17:21:00.000	Rubi	AMIGDALITIS AGUDA	1	1
▸	November 17th 2017, 20:52:00.000	Sant Feliu de Codines	AMIGDALITIS AGUDA	2	1
▸	November 19th 2017, 09:16:00.000	Terrassa	AMIGDALITIS AGUDA	1	1
▸	November 19th 2017, 10:03:00.000	Sant Joan Despi	AMIGDALITIS AGUDA	1	1
▸	November 21st 2017, 07:55:00.000	Sant Cugat	AMIGDALITIS AGUDA	2	1
▸	November 21st 2017, 17:34:00.000	Sant Cugat	AMIGDALITIS AGUDA	2	1
▸	November 25th 2017, 12:46:00.000	Sant Cugat	AMIGDALITIS AGUDA	1	1
▸	November 26th 2017, 09:10:00.000	Sant Cugat	AMIGDALITIS AGUDA	1	1
▸	November 28th 2017, 19:51:00.000	Santa Eul�lia de Ron�ana	AMIGDALITIS AGUDA	2	1
▸	November 29th 2017, 02:29:00.000	Sabadell	AMIGDALITIS AGUDA	2	1
▸	November 29th 2017, 16:39:00.000	Sant Cugat	AMIGDALITIS AGUDA	2	1

7. Kibana — Aggregations and Metrics

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.

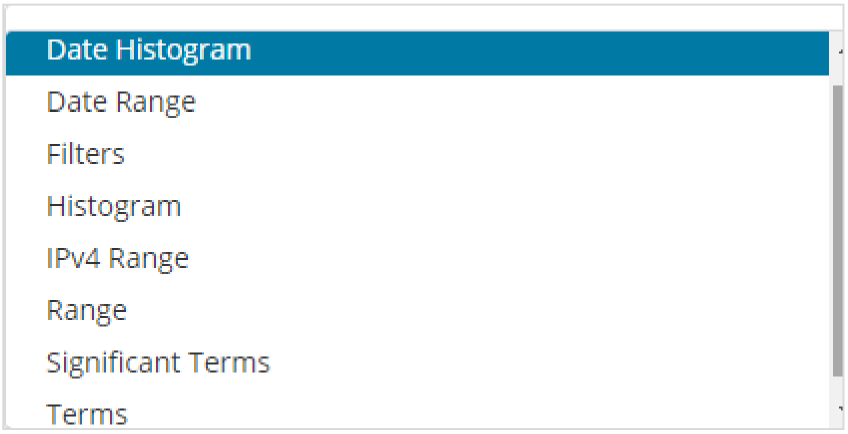
Whenever you perform any visualization, you need to decide the criteria, which means in which way you want to group the data to perform the metric on it.

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:



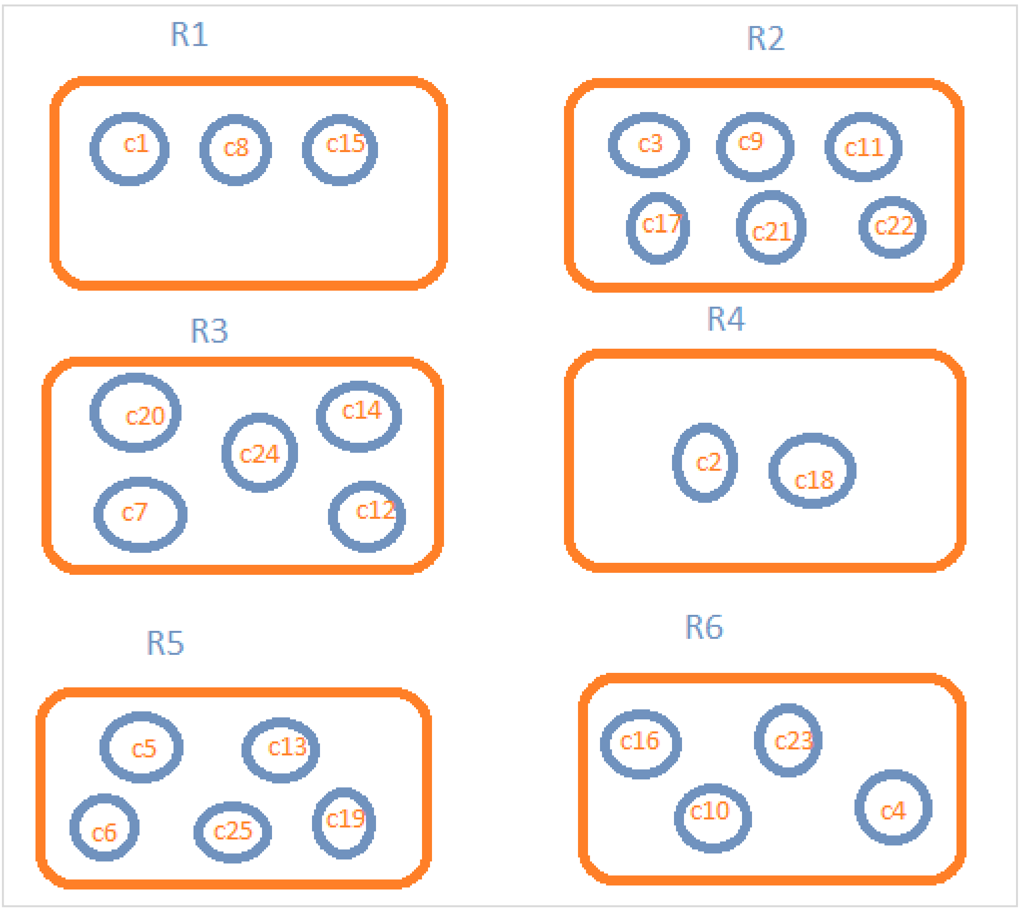
Bucket Aggregation has the following list:

- 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.

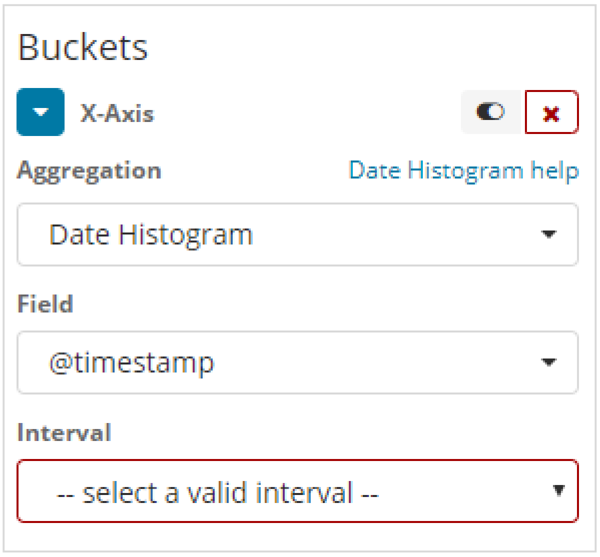
The list of Bucket Aggregation we have so far is:

- 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:


Buckets

X-Axis

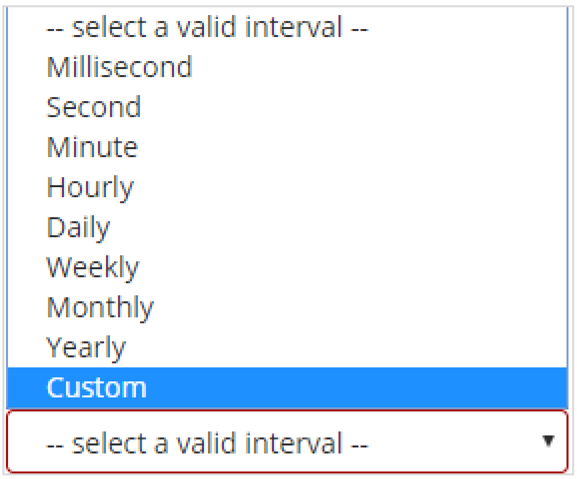
Aggregation: Date Histogram

Date Histogram help

Field: @timestamp

Interval: -- select a valid interval --

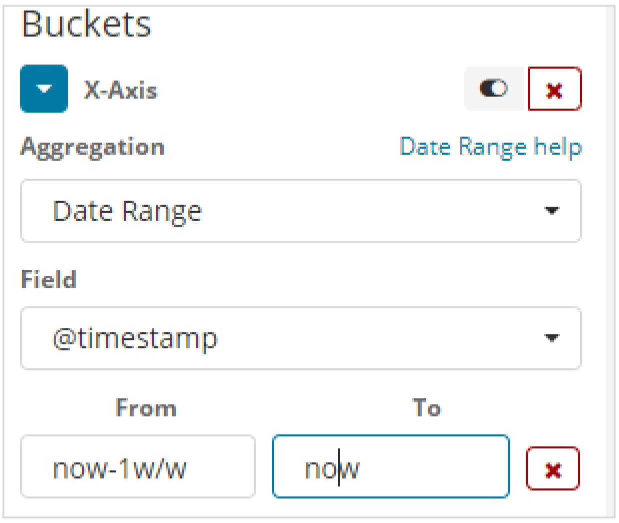
When you Select Buckets Aggregation as Date Histogram, it will display the Field option which will give only the date related fields. Once you select your field, you need to select the Interval which has the following details:



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.

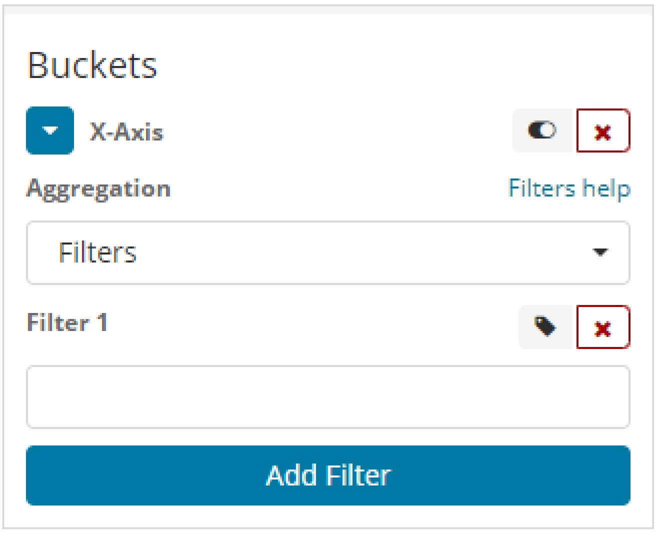


Aggregation: Date Range
Field: @timestamp
From: now-1w/w
To: now

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.

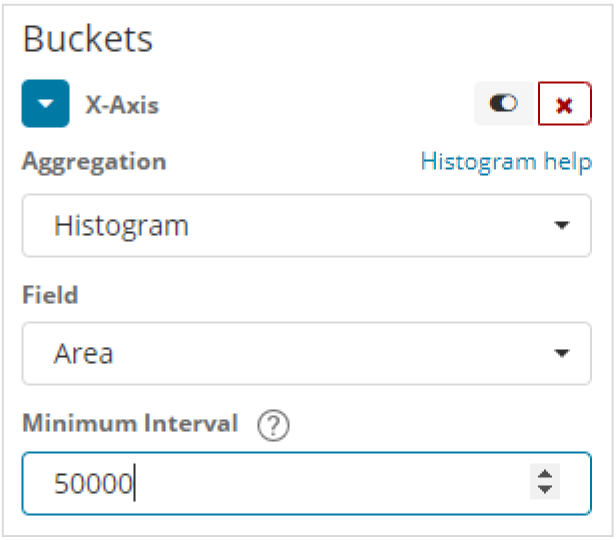
Using filters, users can write their queries in the filter option as shown below:



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

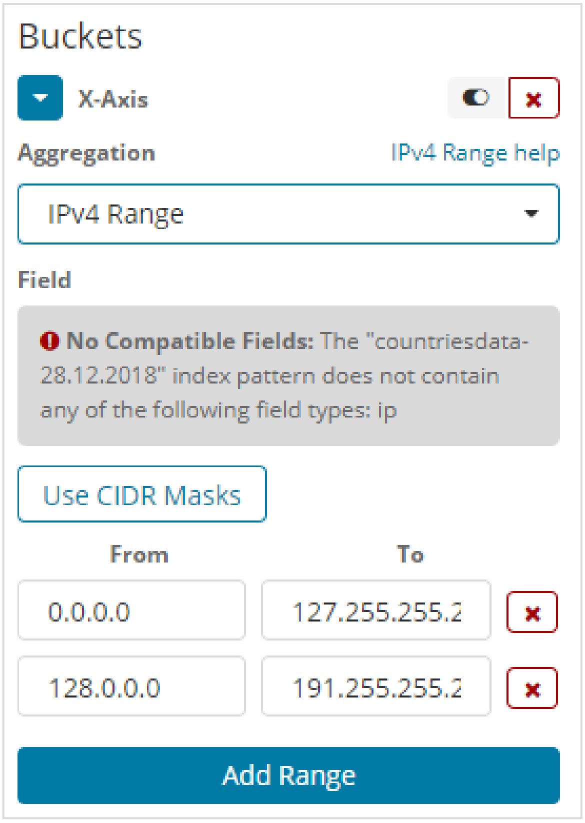
Histogram

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.



Buckets

X-Axis

Aggregation — IPv4 Range help

IPv4 Range

Field

No Compatible Fields: The "countriesdata-28.12.2018" index pattern does not contain any of the following field types: ip

Use CIDR Masks

From	To
0.0.0.0	127.255.255.2
128.0.0.0	191.255.255.2

Add Range

The index that we have that is the contriesdata-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.

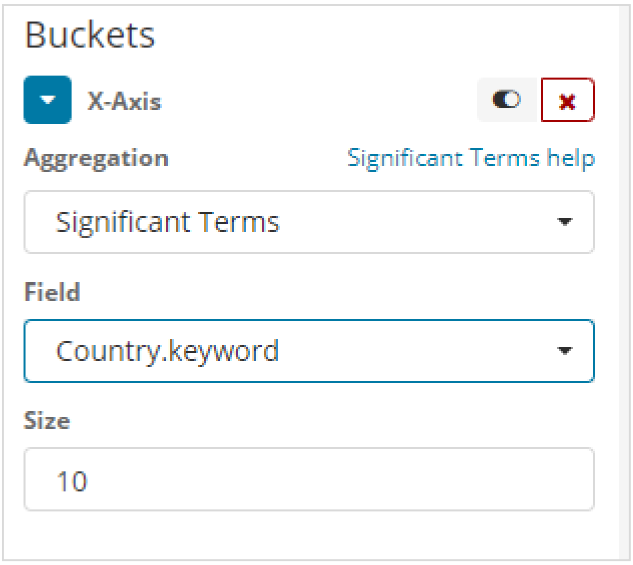
Range

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.



Buckets

X-Axis

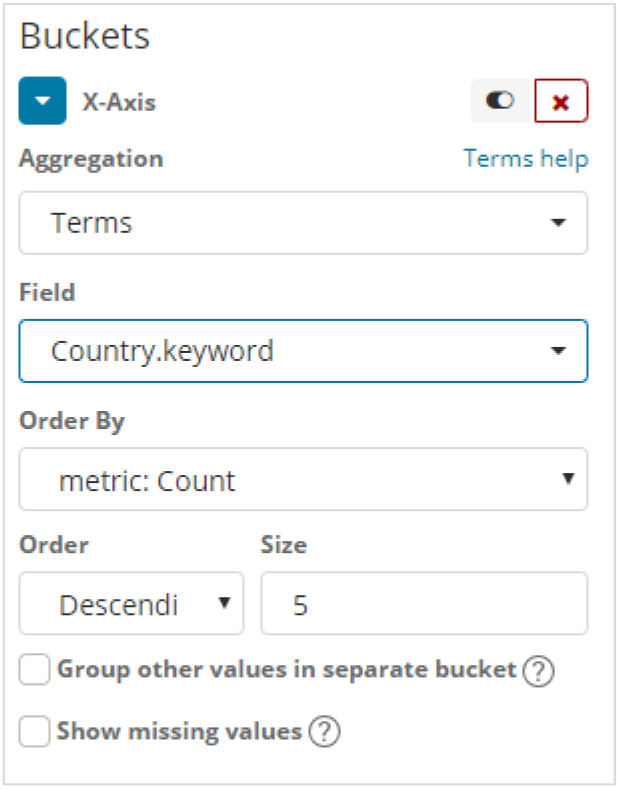
Aggregation: Significant Terms

Field: Country.keyword

Size: 10

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.



Buckets

X-Axis

Aggregation: Terms

Field: Country.keyword

Order By: metric: Count

Order: Descendi

Size: 5

Group other values in separate bucket

Show missing values

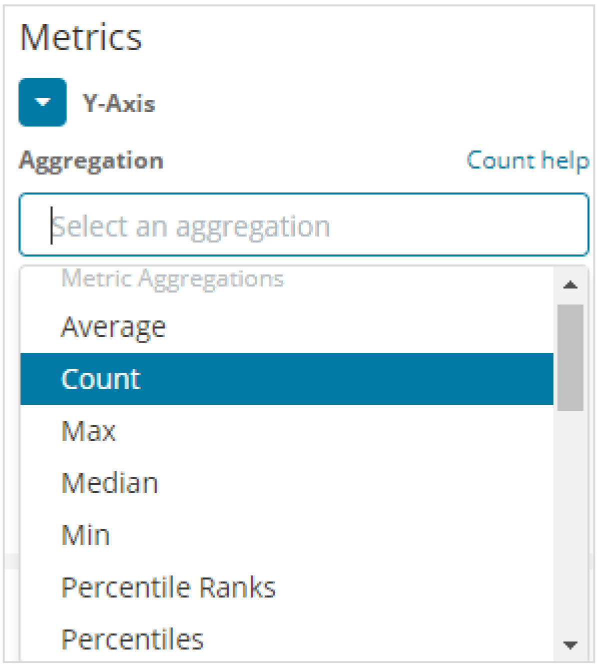
We have an option order by which we will group the data based on the metric we select. The size refers to the number of buckets you want to display in the visualization.

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:



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

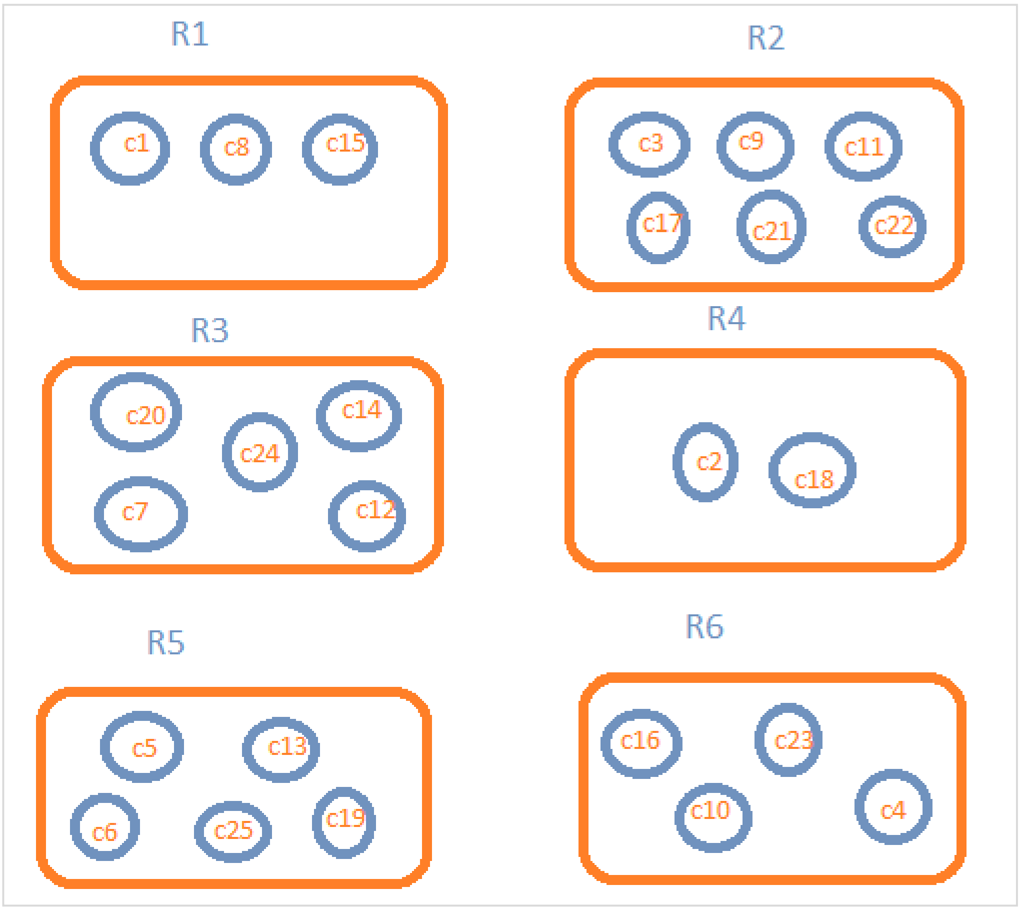
- Average
- Count
- Max
- Min
- Sum

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 is500 and c15 is 700. Now to get the average value of R1 bucket

R1 = value of c1 + value of c8 + value of c15 / 3
= 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.

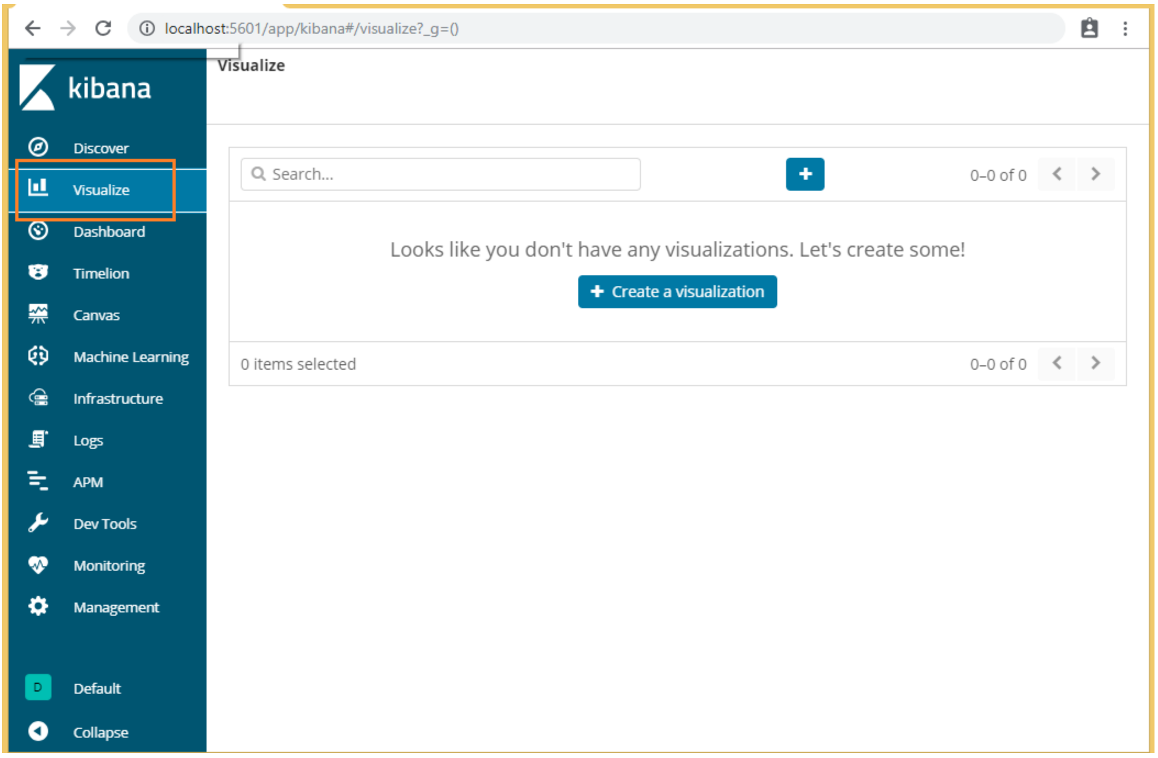
In case we have documents with area in the region than R1 to R6 will have the country wise area summed up for the region.

8. Kibana — Create Visualization

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

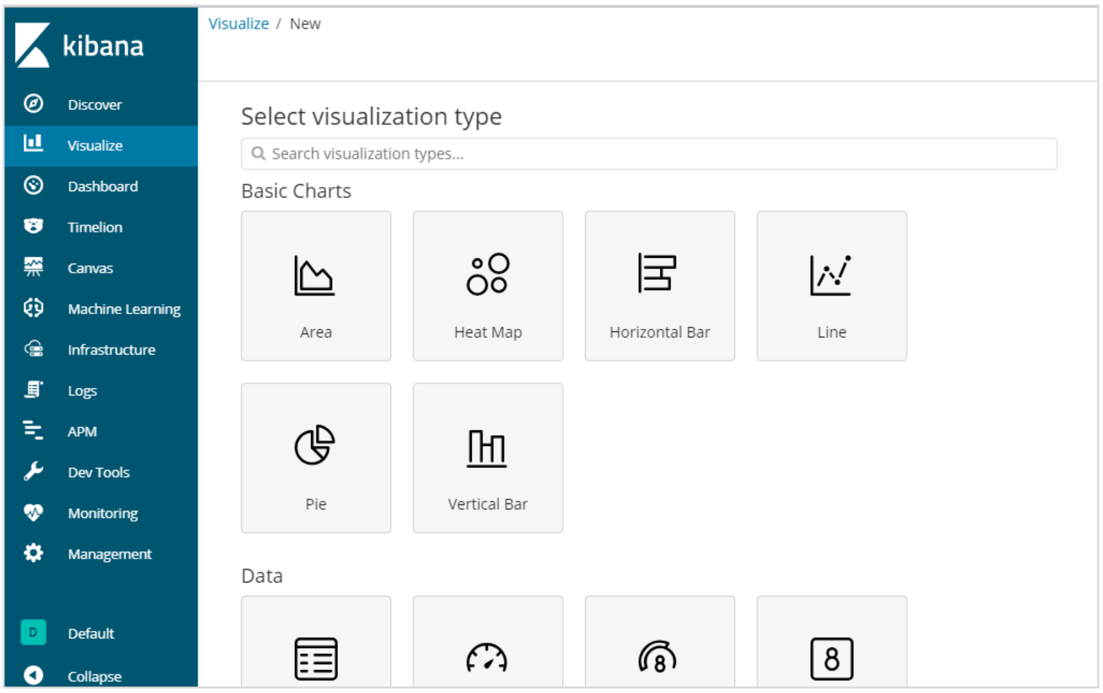
Go to Kibana Visualization as shown below:



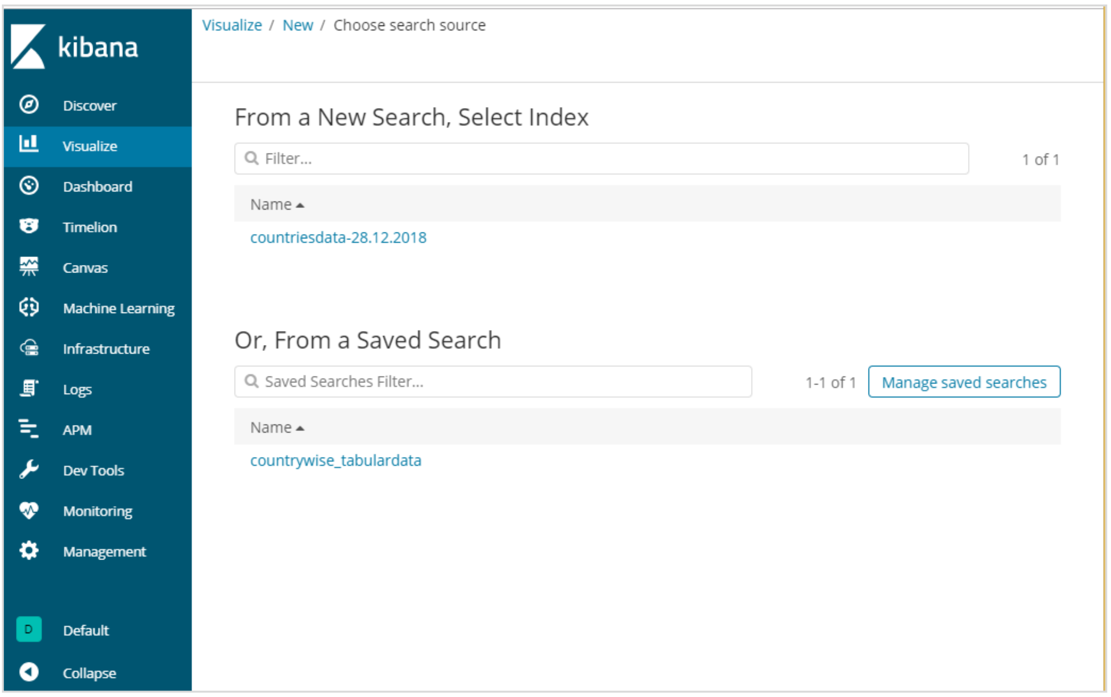
Looks like you don't have any visualizations. Let's create some!

We do not have any visualization created, so it shows blank and there is a button to create one.

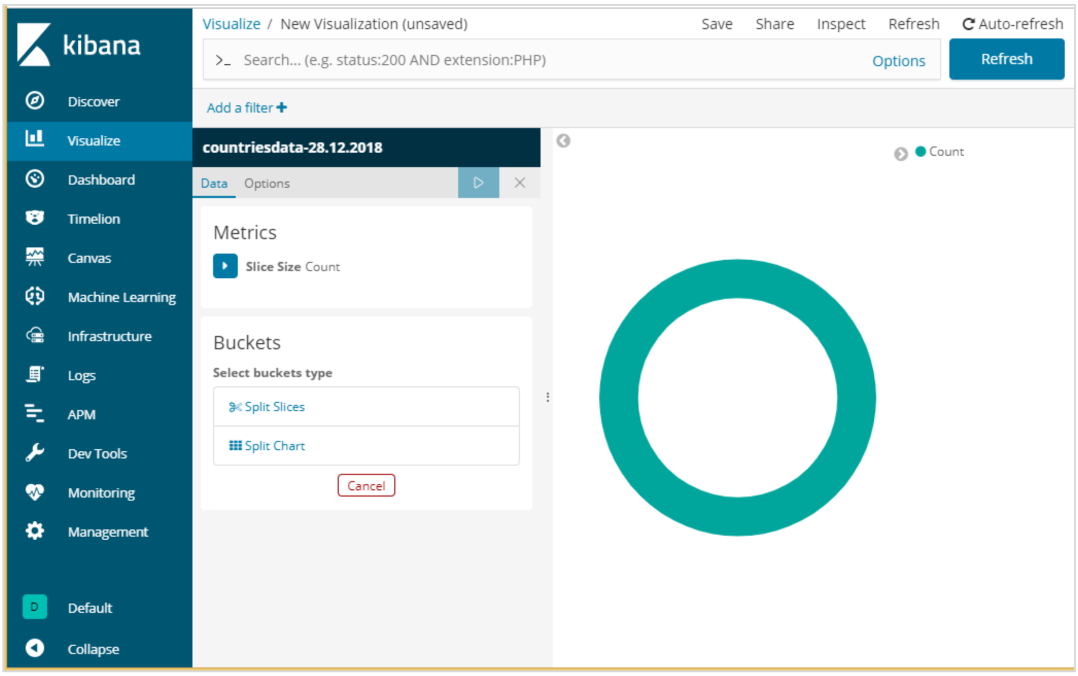
Click the button ***Create a visualization*** as shown in the screen above and it will take you to the screen as shown below:



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.



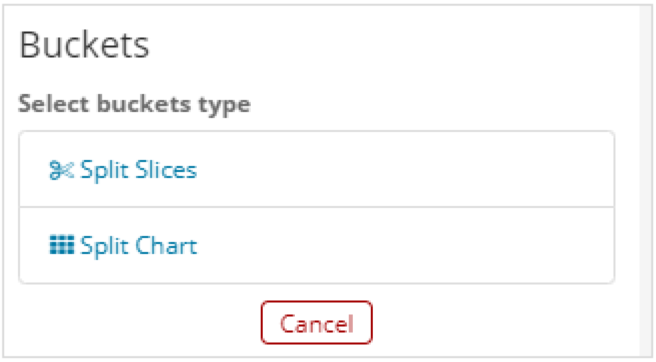
Once you select the visualization type, now you need to select the index on which you want to work on, and it will take you the screen as shown below:



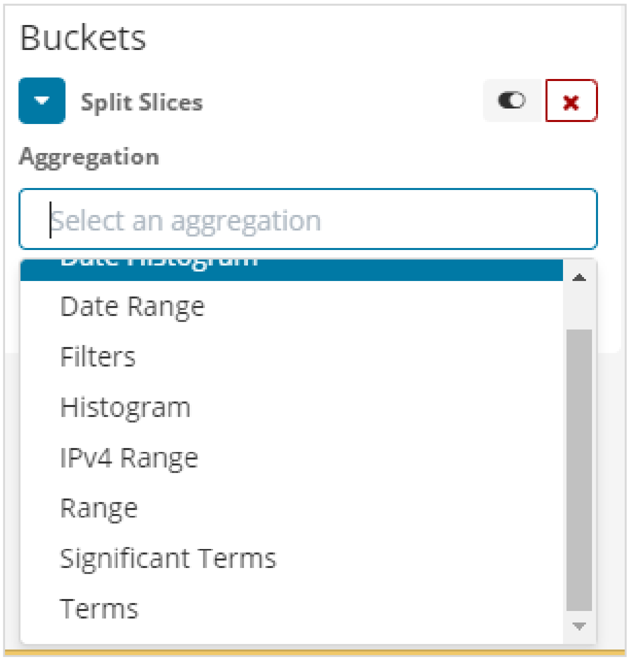
Now we have a default pie chart. We will use the countriesdata-28.12.2018 to get the count of regions available in the countries data in pie chart format.

Bucket and Metric Aggregation

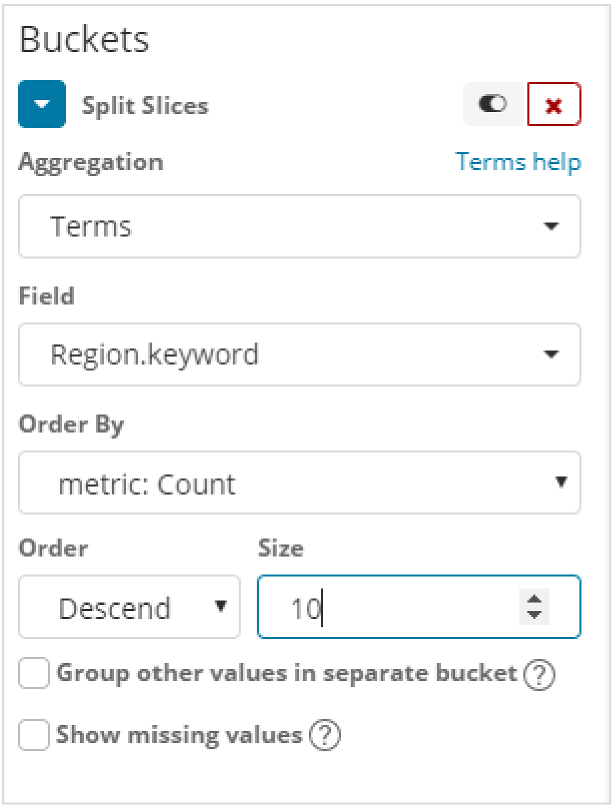
The left side has metrics, which we will select as count. In Buckets, there are 2 options Split slices and split chart. We will use the option Split slices.



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:



Aggregation: Terms

Field: Region.keyword

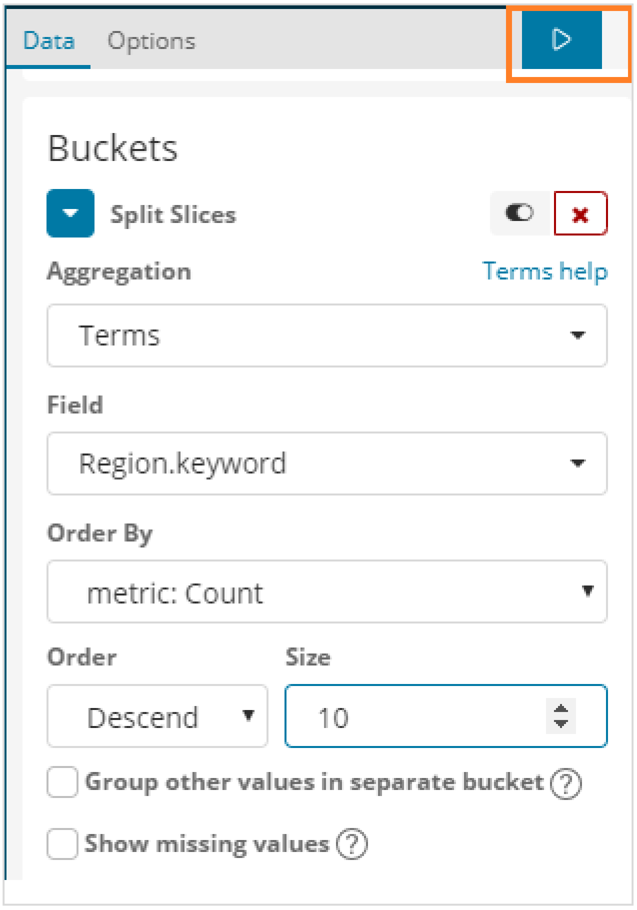
Order By: metric: Count

Order: Descend

Size: 10

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.



Aggregation: Terms

Field: Region.keyword

Order By: metric: Count

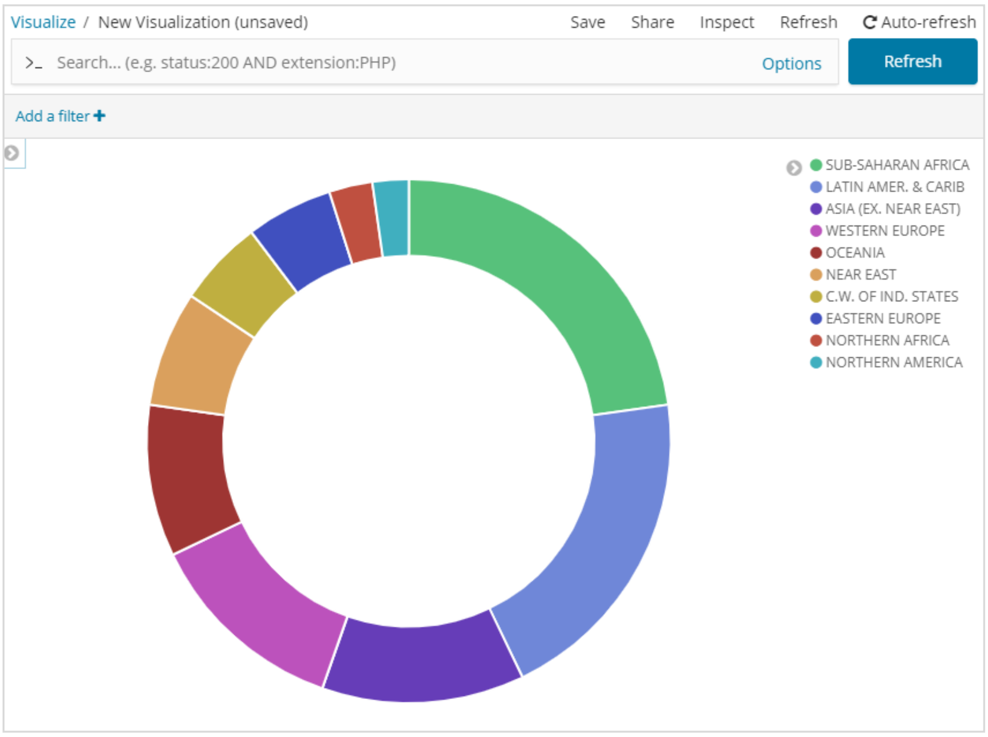
Order: Descend

Size: 10

☐ Group other values in separate bucket

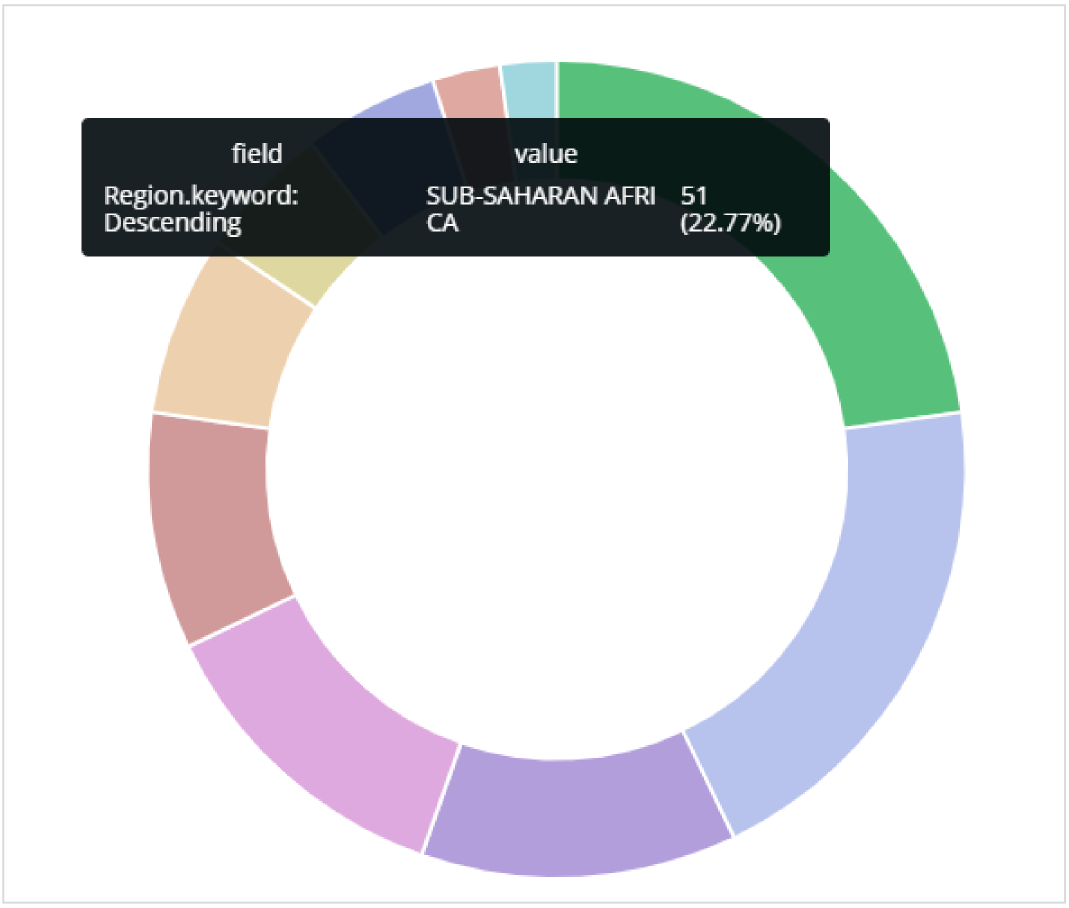
☐ Show missing values

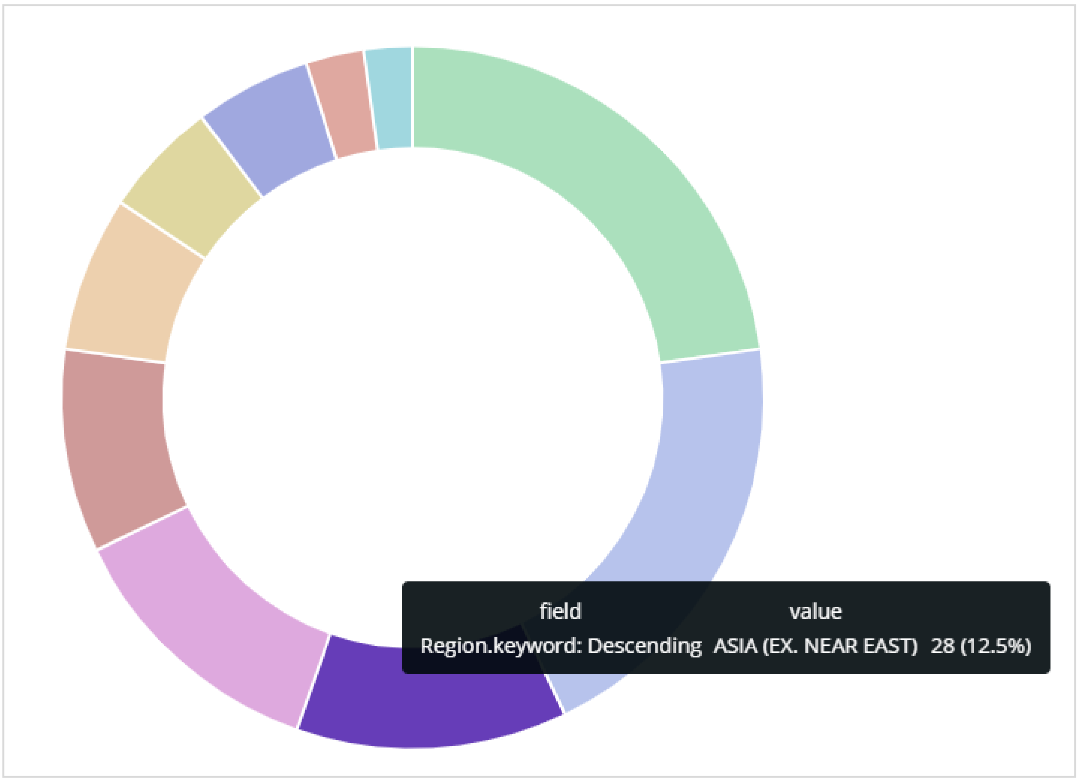
Pie chart display



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:

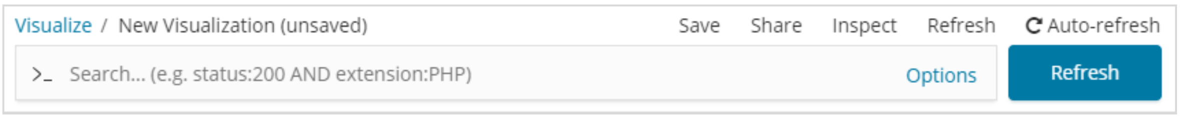
So it tells us that 22.77% of region is occupied by Sub-Saharan Afri from the countries data we have uploaded.

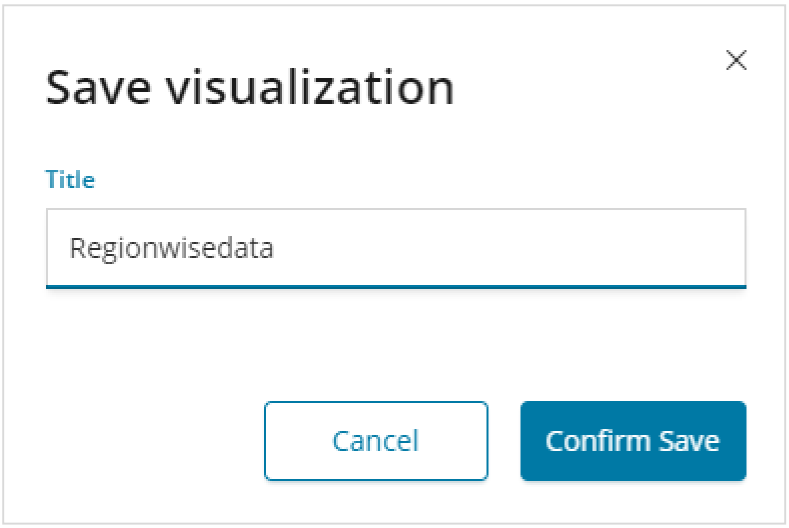




Asia region covers 12.5% and the count is 28.

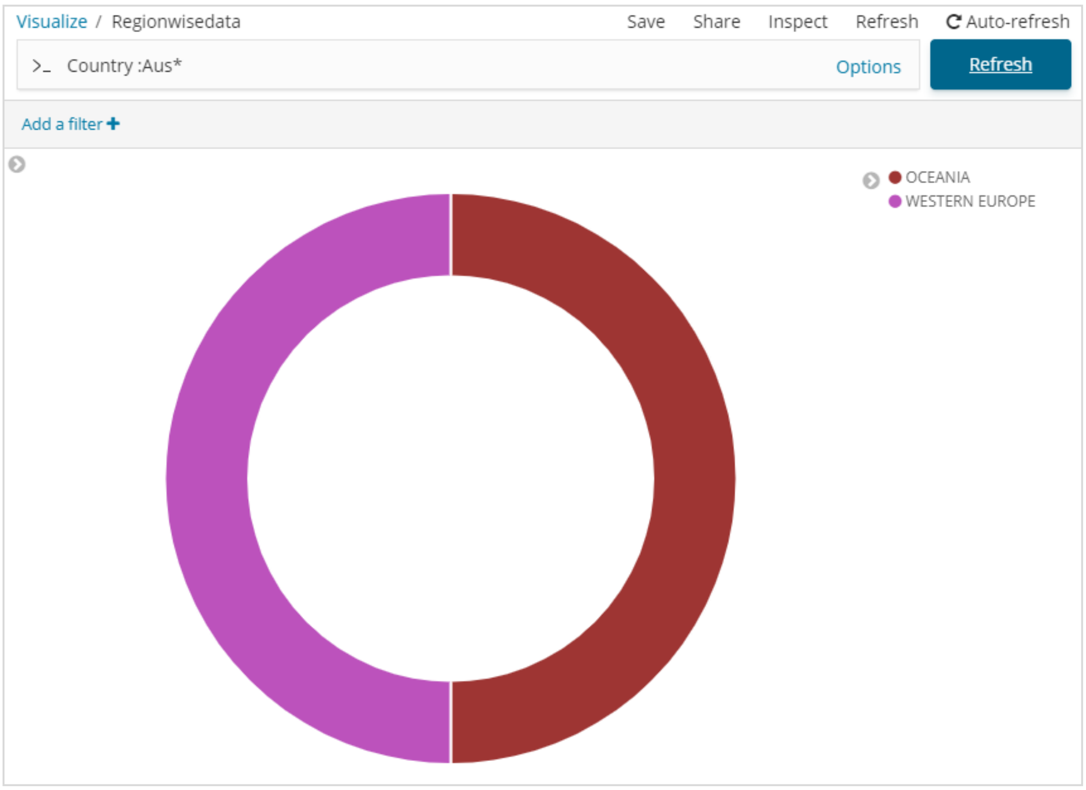
Now we can save the visualization by clicking on the save button on top right corner as shown below:


Visualize / New Visualization (unsaved)
Save
Share
Inspect
Refresh
Auto-refresh
>_ Search... (e.g. status:200 AND extension:PHP)
Options
Refresh



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.

9. Kibana — Working with Charts

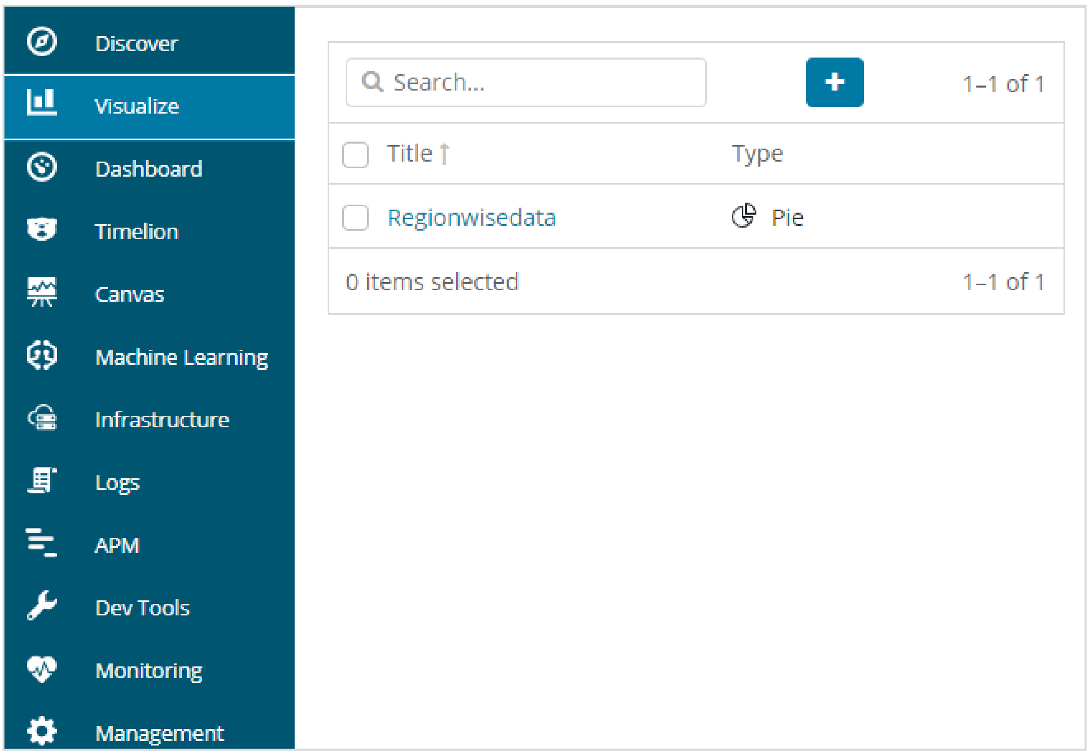
Let us explore and understand the most commonly used charts in visualization.

- 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:

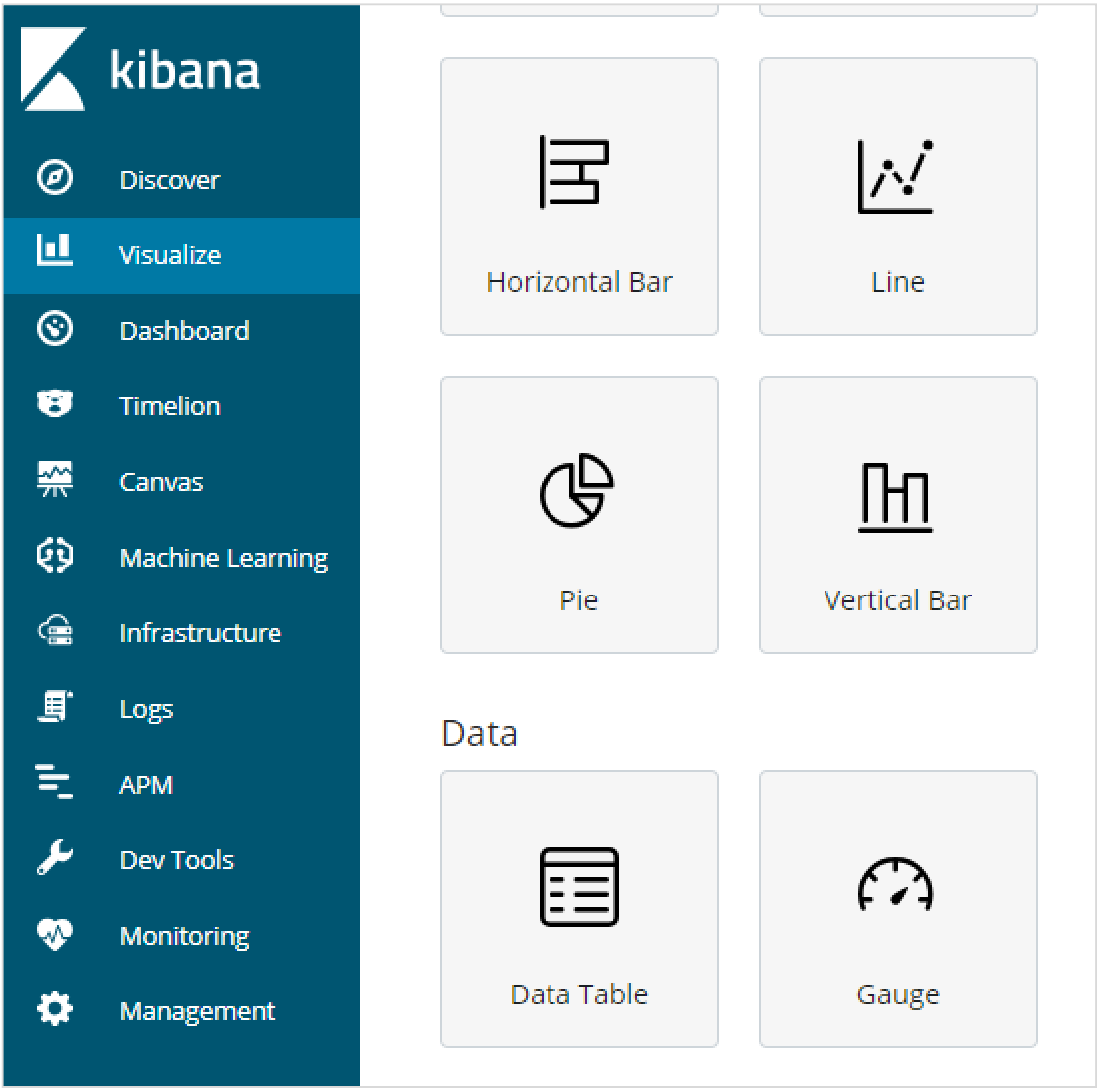


Title ↑	Type
Regionwisedata	Pie

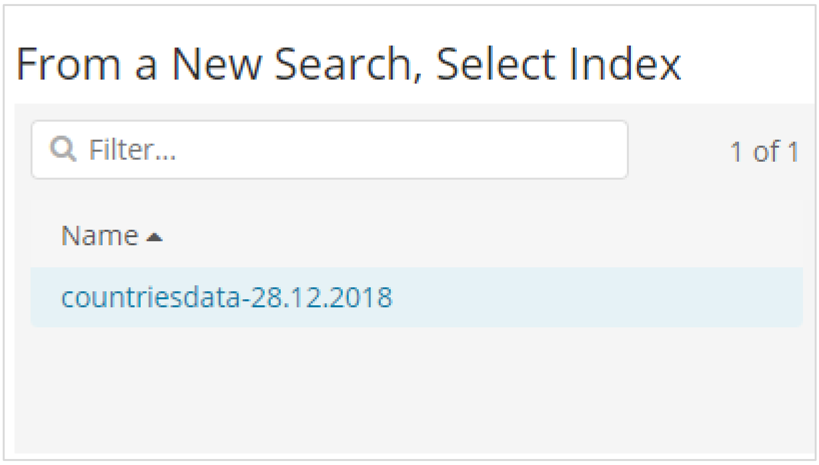
1–1 of 1

0 items selected 1–1 of 1

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

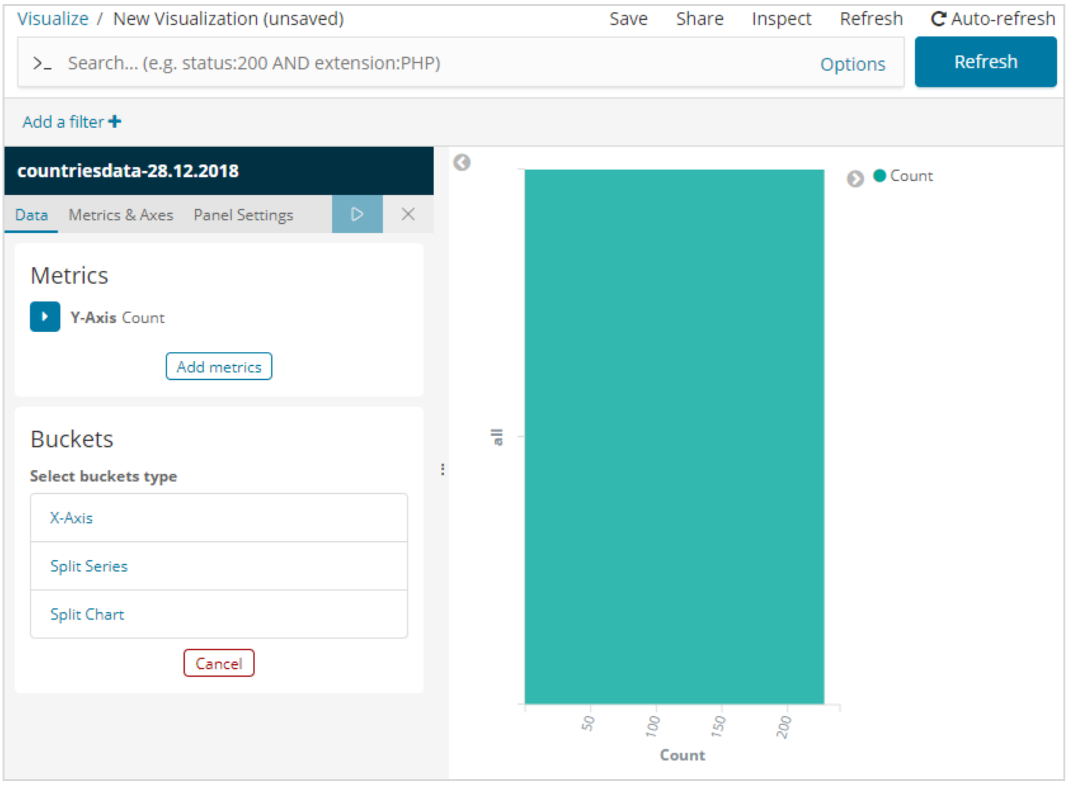
Filter...

1 of 1

Name ▲

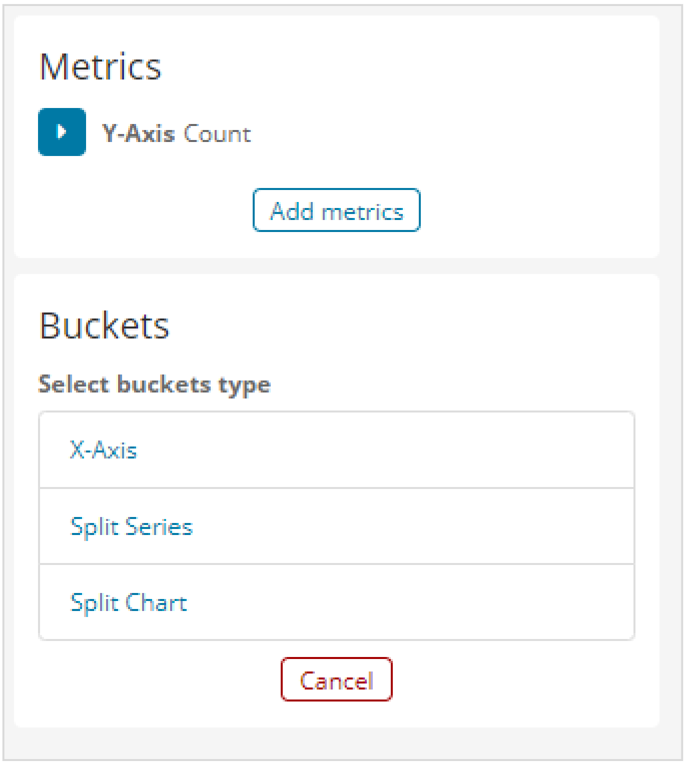
countriesdata-28.12.2018

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

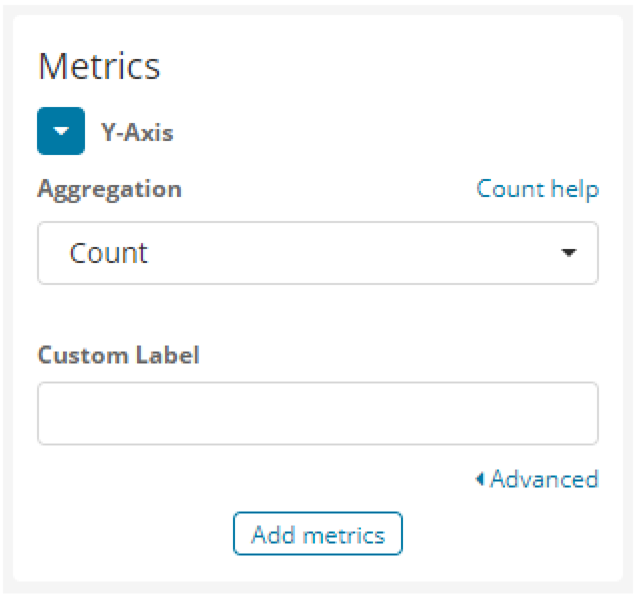


It shows a default count. Now, let us plot a horizontal graph where we can see the data of top 10 country wise populations.

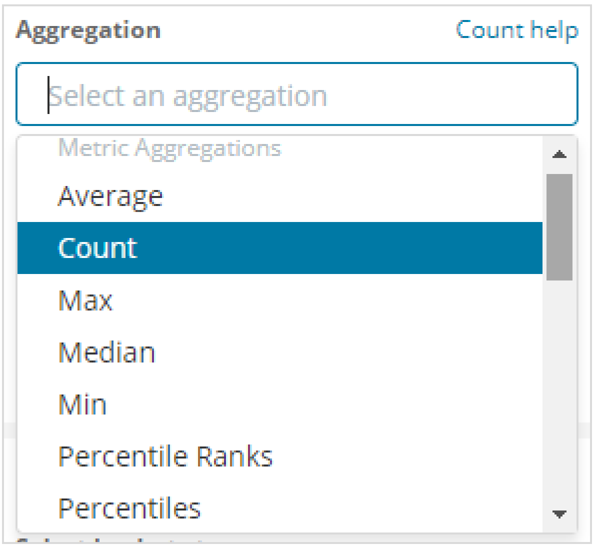
For this purpose, we need to select what we want on the Y and X axis. Hence, select the Bucket and Metric Aggregation:



Now, if you click on Y-Axis, it will display the screen as shown below:



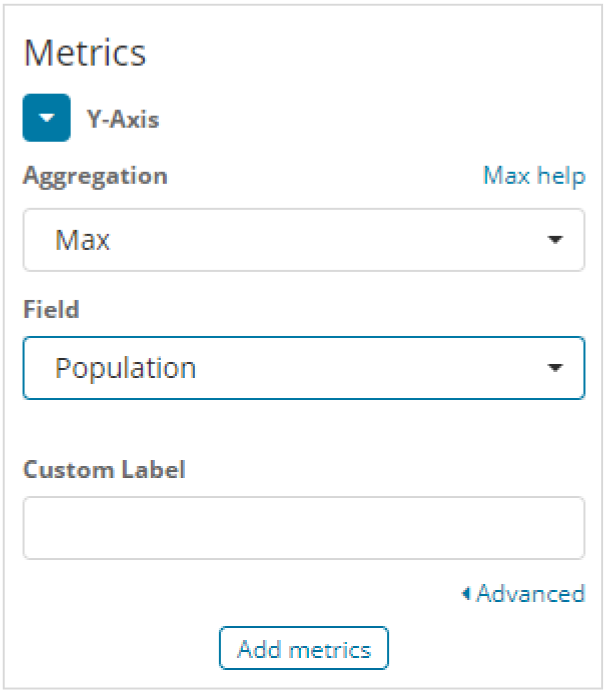
Now, select the Aggregation that you want from the options shown here:



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:



Metrics

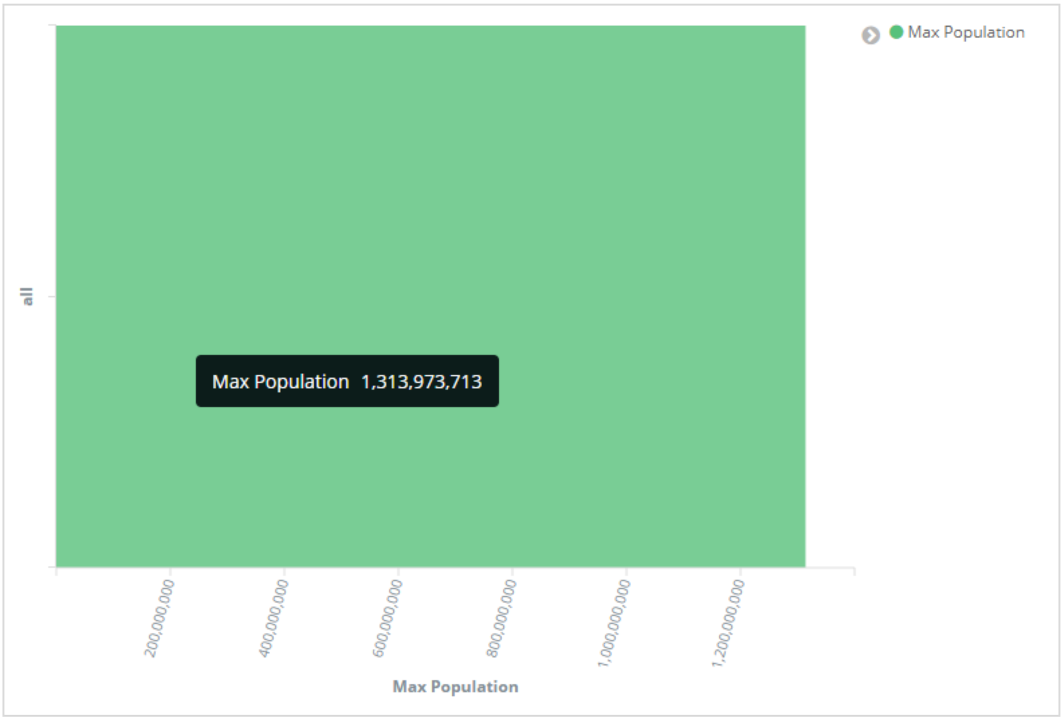
Y-Axis

Aggregation: Max

Field: Population

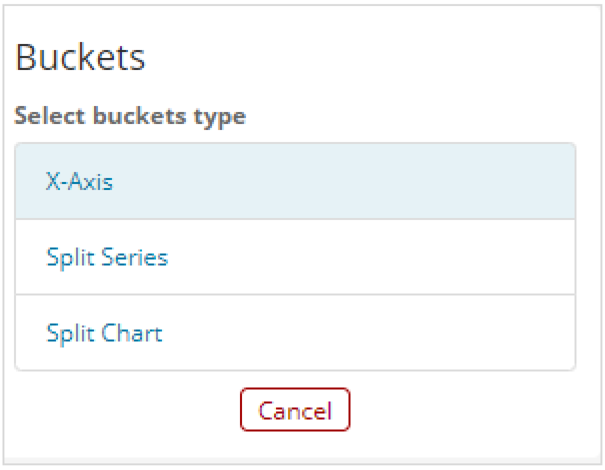
Custom Label:

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

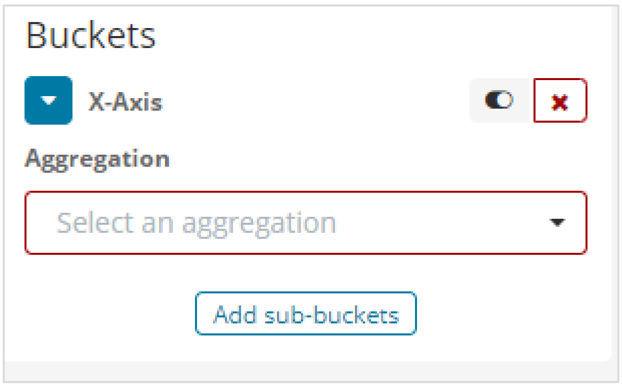


Max Population 1,313,973,713

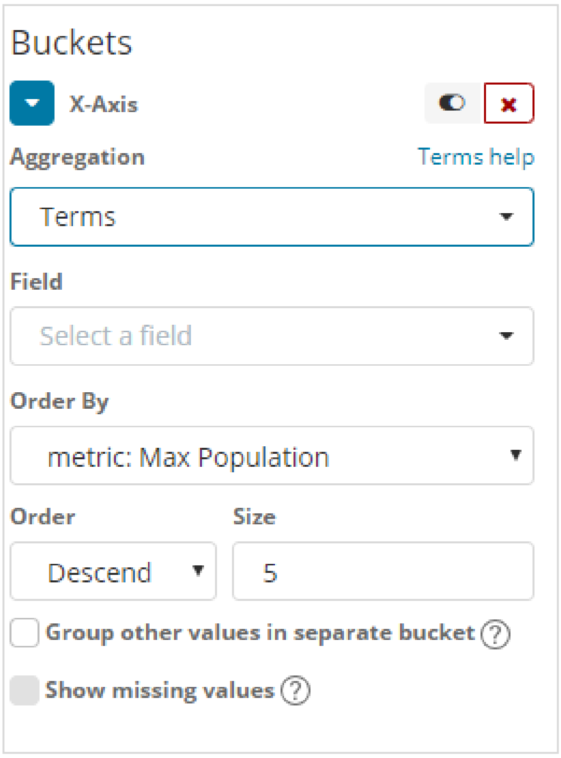
Now let us select the X-axis as shown below:



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



Choose Aggregation as Terms.



Aggregation: Terms

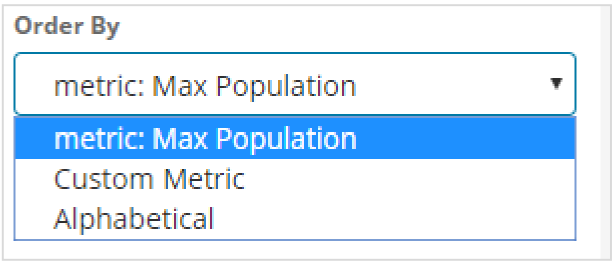
Field: Select a field

Order By: metric: Max Population

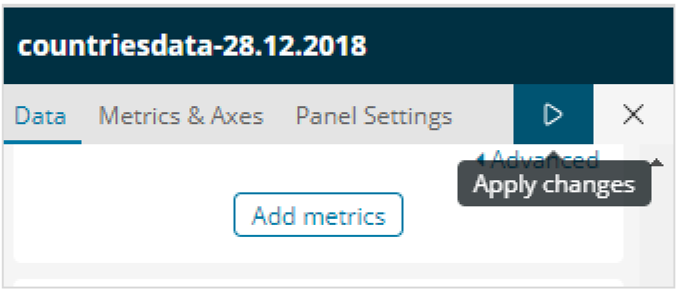
Order: Descend

Size: 5

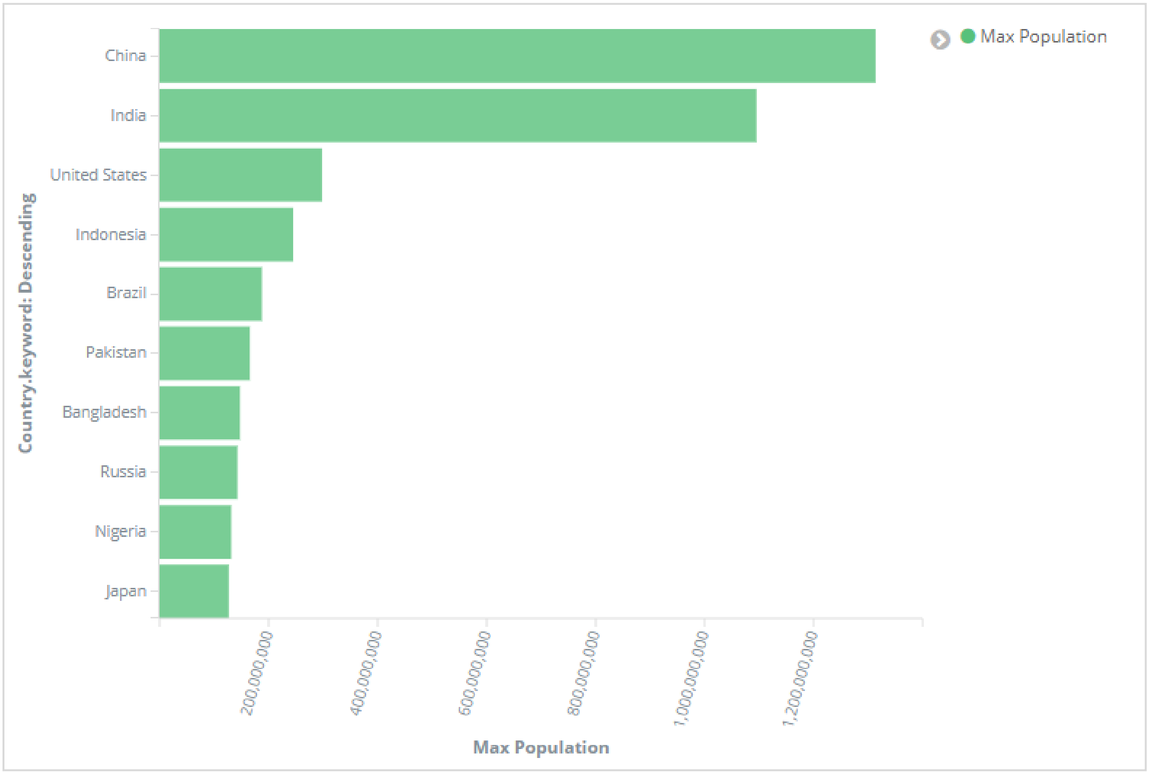
Choose the field from the dropdown. We want country wise population so select country field. Order by we have following options:



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.

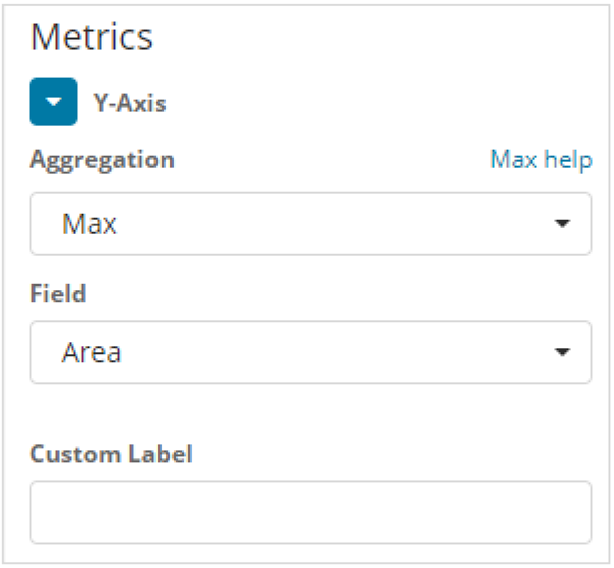
Vertical Bar Chart

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:

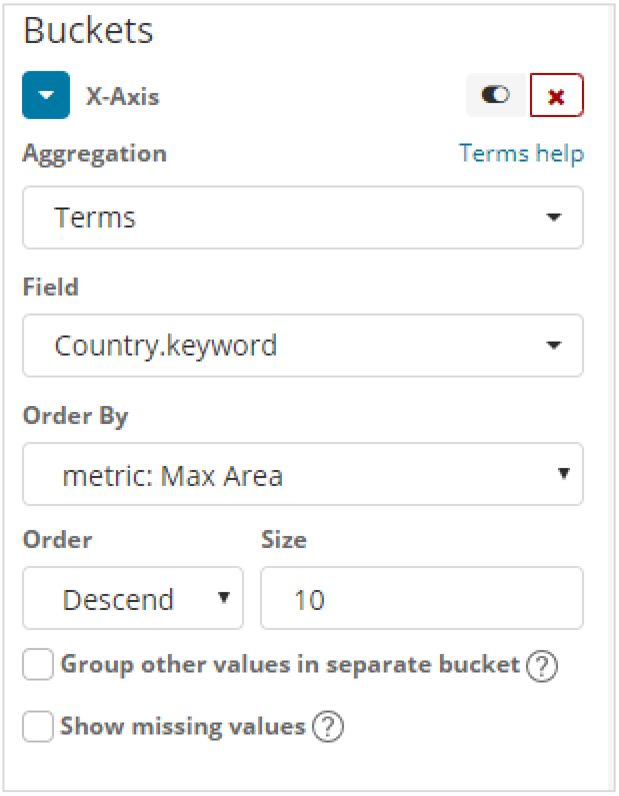
Y-axis



Aggregation: Max

Field: Area

X-axis



Buckets

X-Axis

Aggregation: Terms

Field: Country.keyword

Order By: metric: Max Area

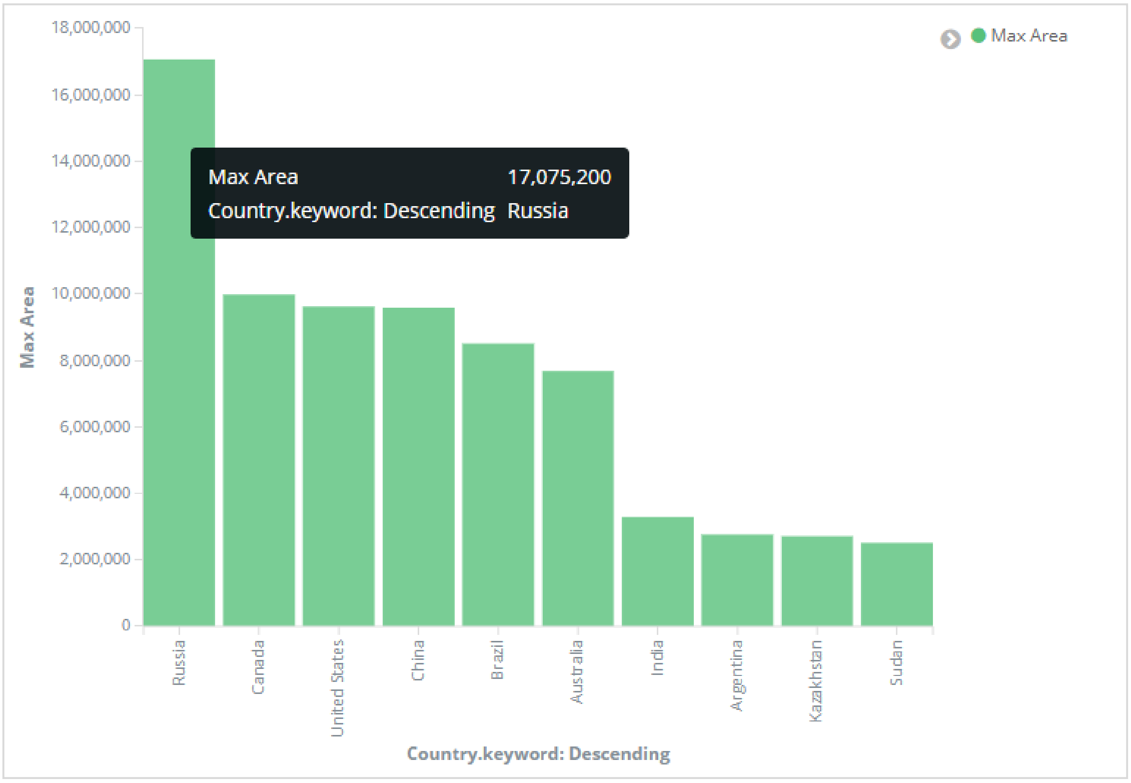
Order: Descend

Size: 10

Group other values in separate bucket

Show missing values

When we apply the changes here, we can see the output as shown below:



Max Area 17,075,200
Country.keyword: Descending Russia

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.

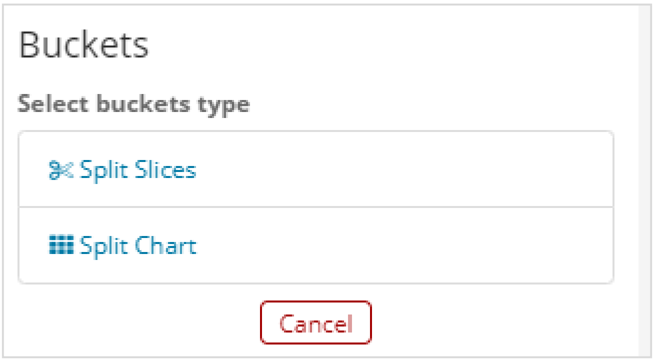
Let us save this visualization as *countrywise_maxarea* to be used with dashboard later.

Next, let us work on Pie chart.

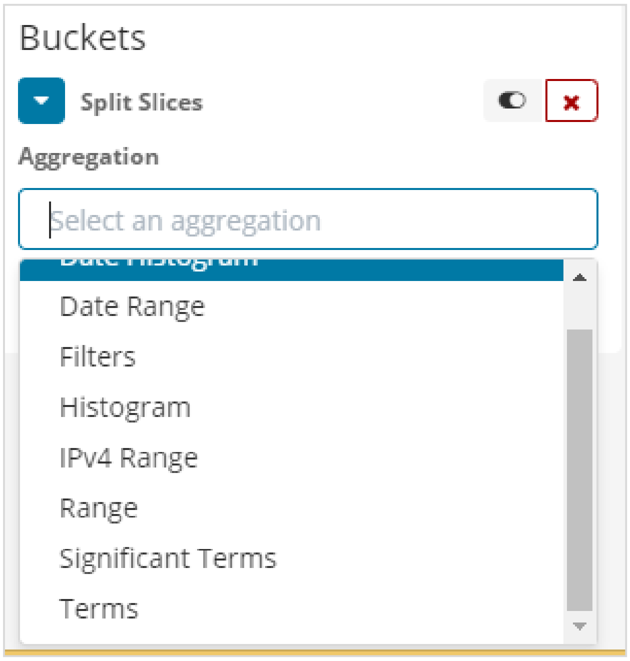
Pie Chart

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.

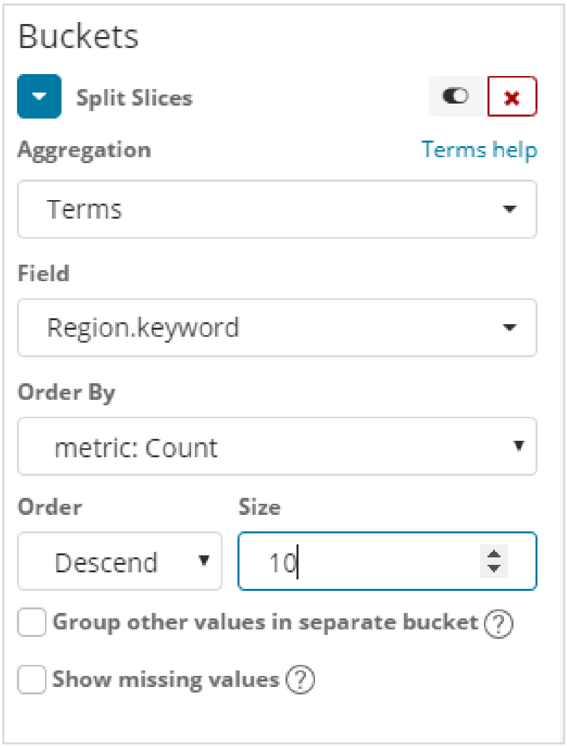
The left side has metrics which will give count. In Buckets, there are 2 options: Split slices and split chart. Now, we will use the option Split slices.



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:



Buckets

Split Slices

Aggregation: Terms

Field: Region.keyword

Order By: metric: Count

Order: Descend

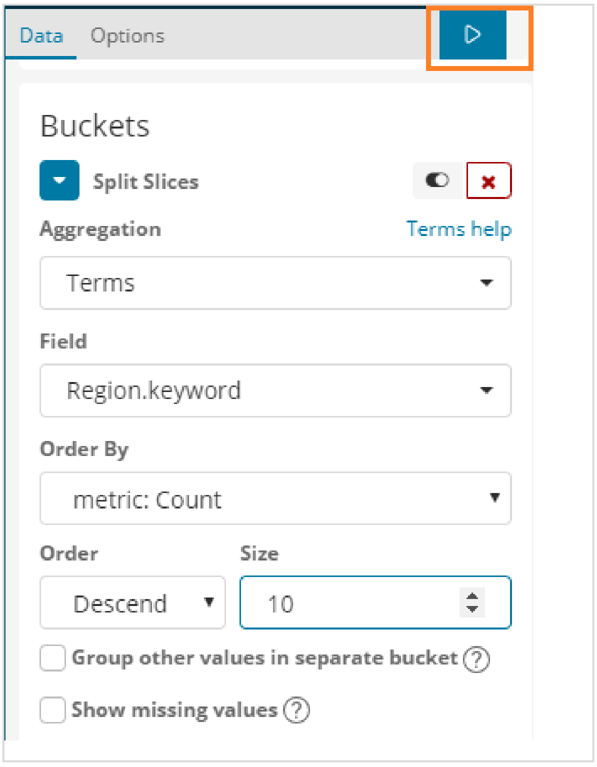
Size: 10

Group other values in separate bucket

Show missing values

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 be 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.



Buckets

Split Slices

Aggregation: Terms

Field: Region.keyword

Order By: metric: Count

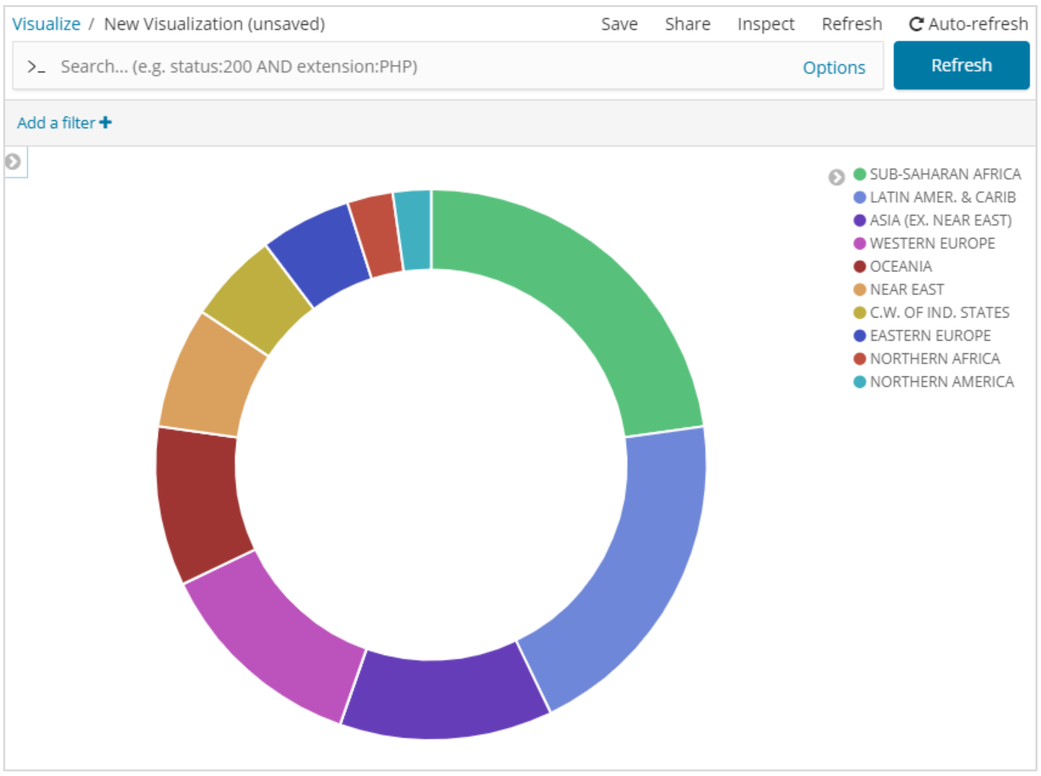
Order: Descend

Size: 10

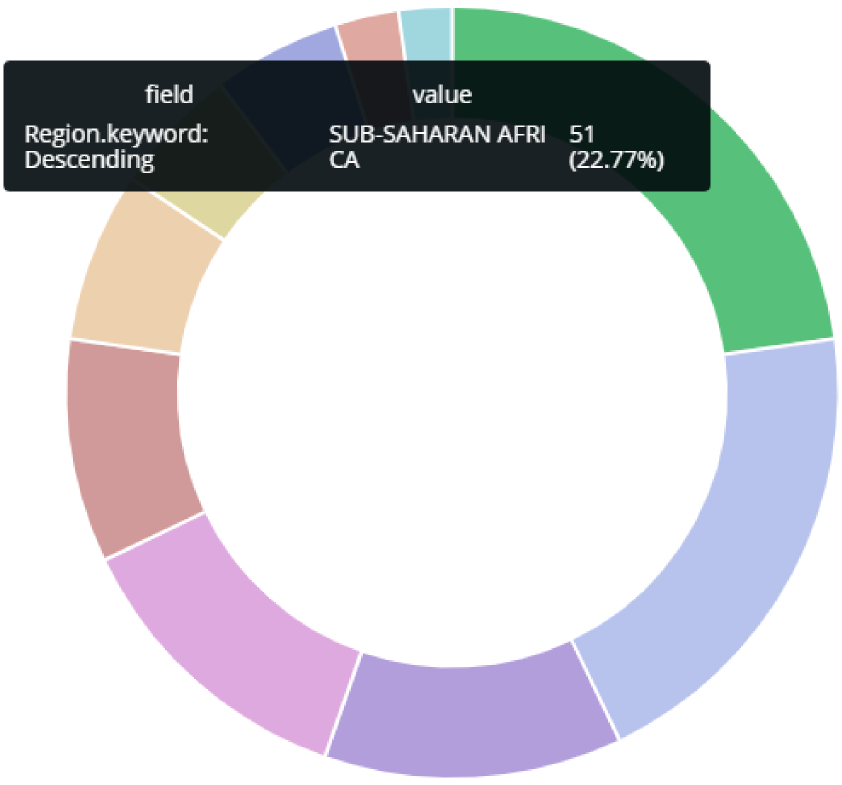
Group other values in separate bucket

Show missing values

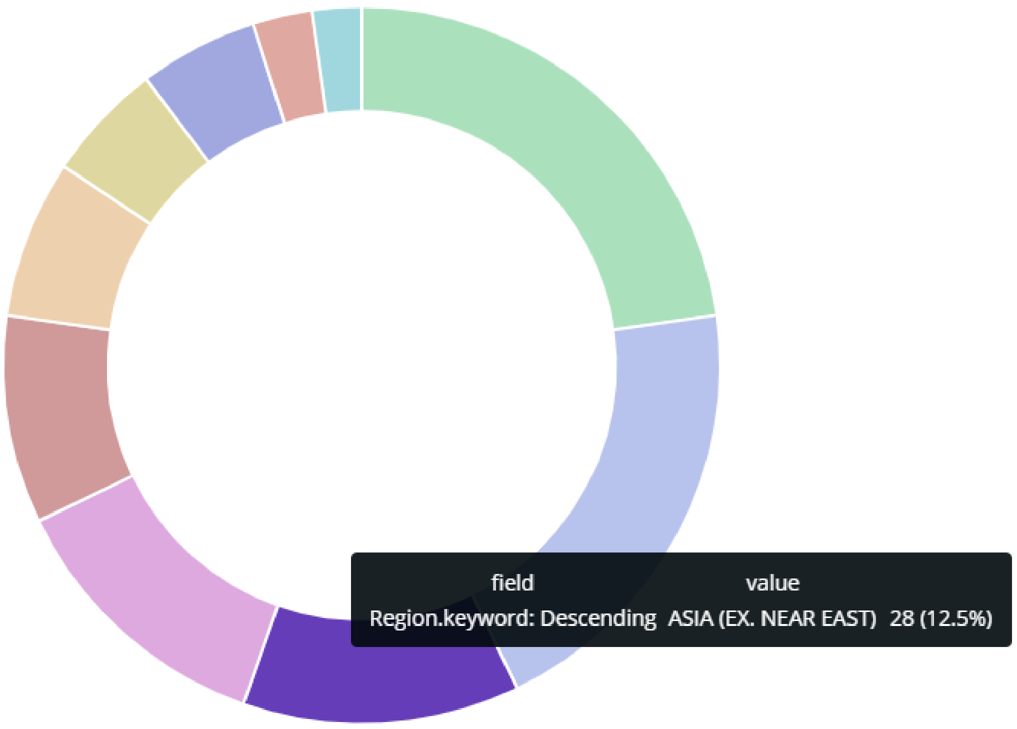
Pie chart display



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:



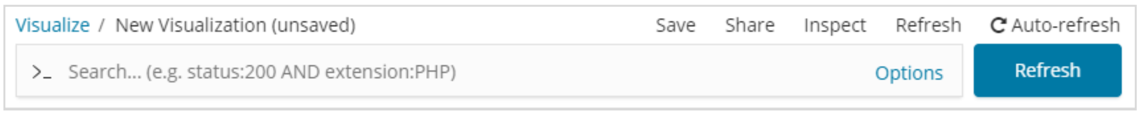
field	value	
Region.keyword: Descending	SUB-SAHARAN AFRICA	51 (22.77%)



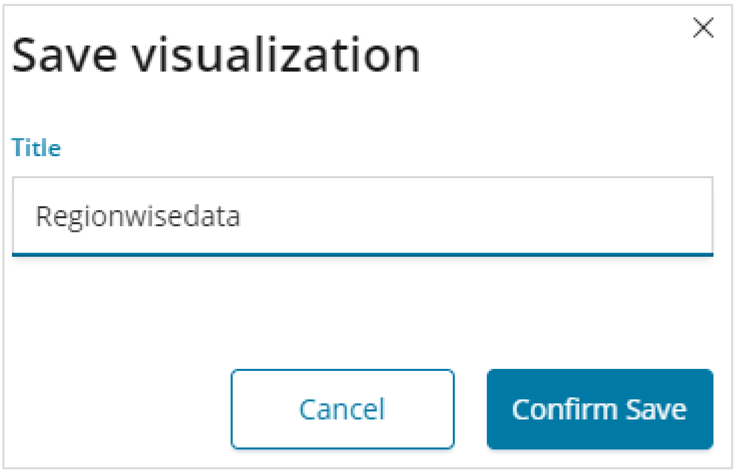
Thus, it tells us that 22.77% of region is occupied by Sub-Saharan Afri in the countries data we have uploaded.

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:


Visualize / New Visualization (unsaved)
Save
Share
Inspect
Refresh
Auto-refresh
>_ Search... (e.g. status:200 AND extension:PHP)
Options
Refresh

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



10. Kibana — Working with Graphs

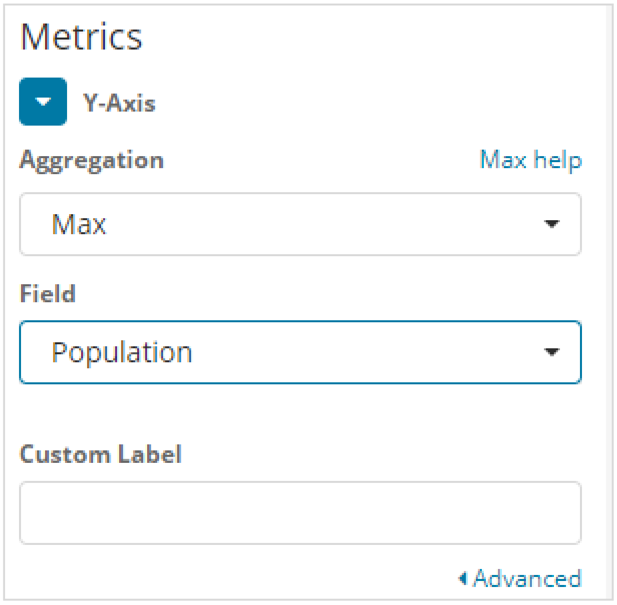
In this chapter, we will discuss the two types of graphs used in visualization:

- Line Graph
- Area

Line Graph

To start with, let us create a visualization, choosing a line graph to display the data and use *contriesdata* 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


Metrics

Y-Axis

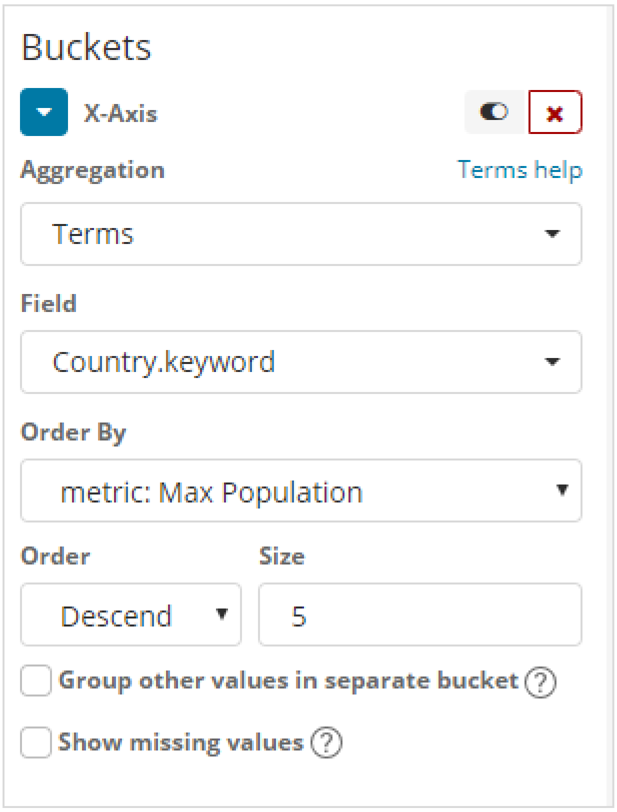
Aggregation: Max

Field: Population

Custom Label:

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

X-Axis

Aggregation: Terms

Field: Country.keyword

Order By: metric: Max Population

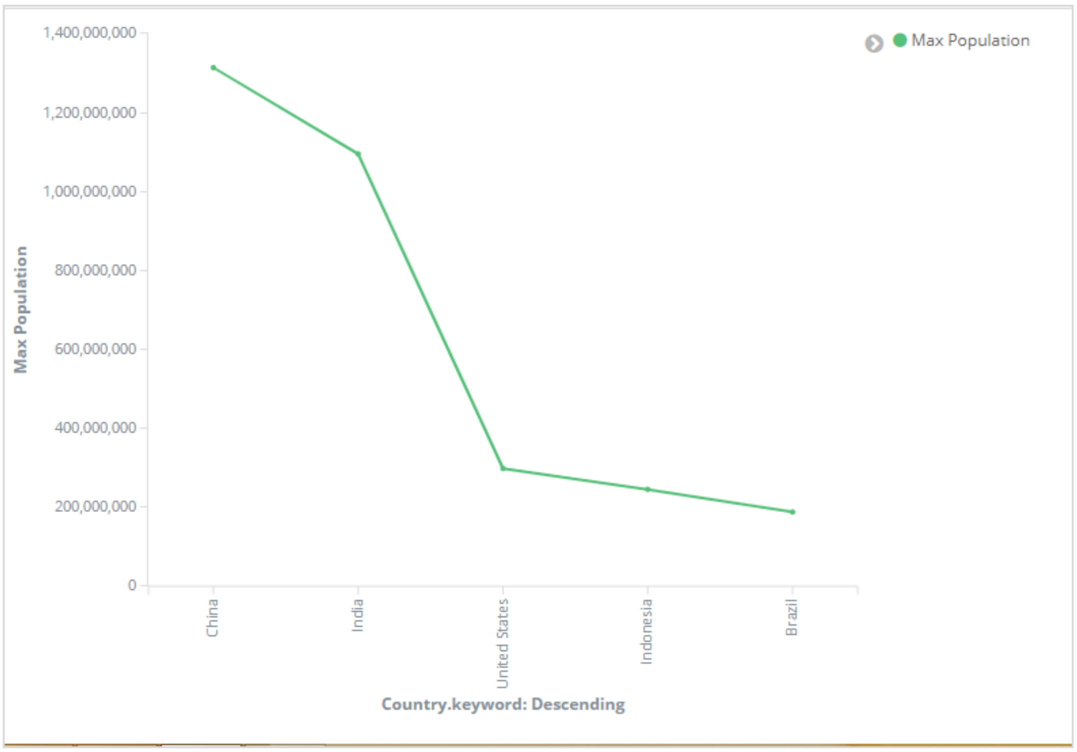
Order: Descend

Size: 5

☐ Group other values in separate bucket

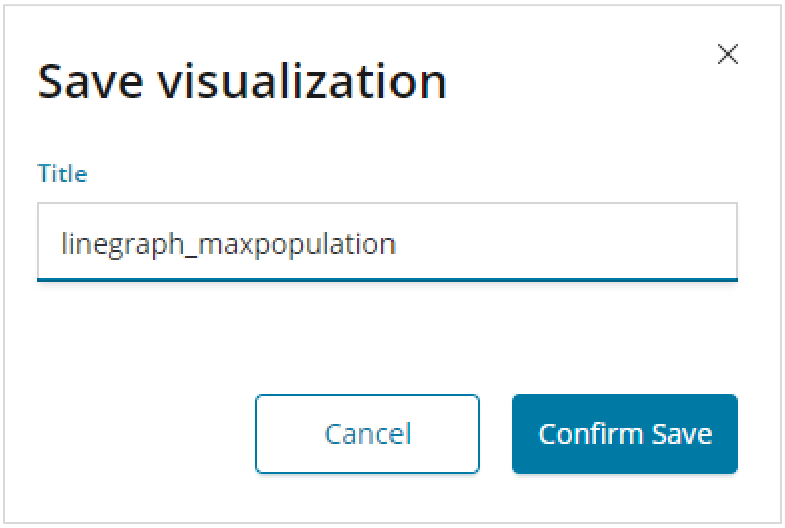
☐ Show missing values

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.



Save visualization

Title

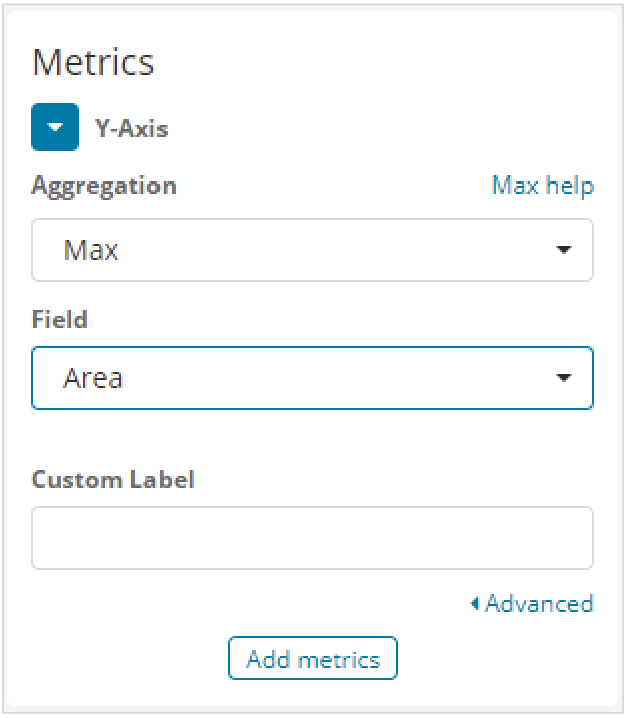
linegraph_maxpopulation

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:



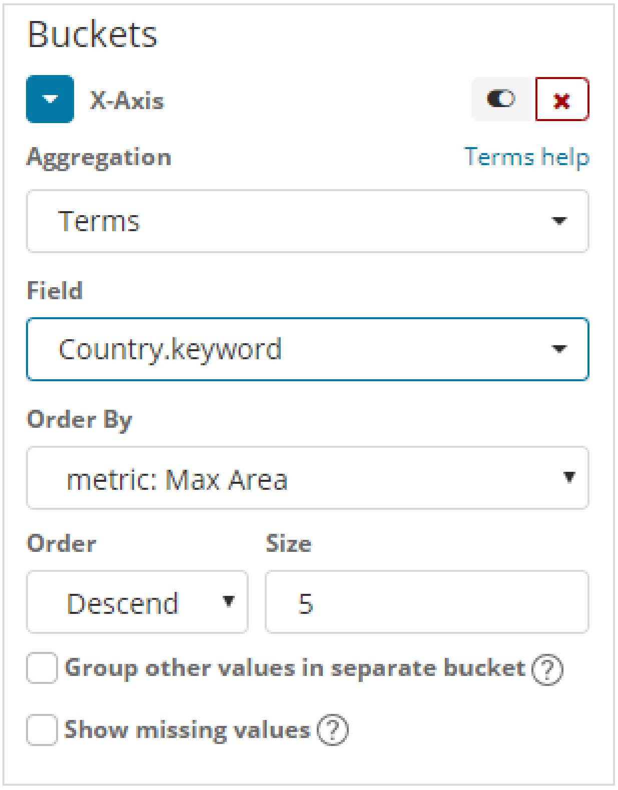
Metrics

Y-Axis

Aggregation: Max

Field: Area

Custom Label:



Buckets

X-Axis

Aggregation: Terms

Field: Country.keyword

Order By: metric: Max Area

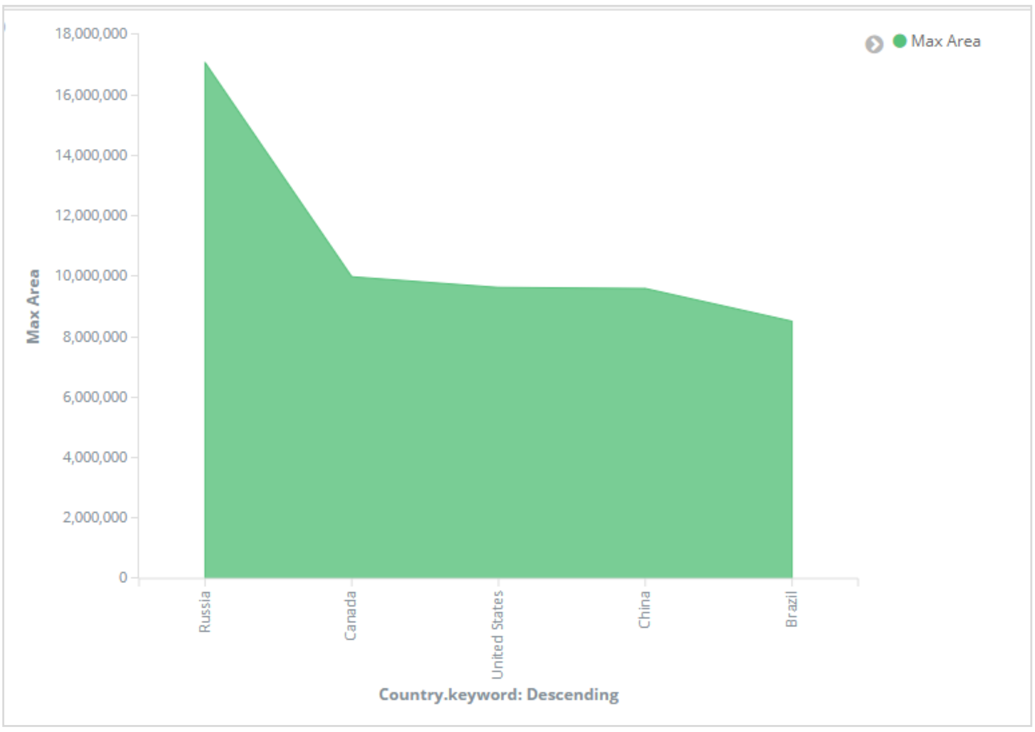
Order: Descend

Size: 5

Group other values in separate bucket

Show missing values

After you click the apply changes button, the output that we can see is as shown below:



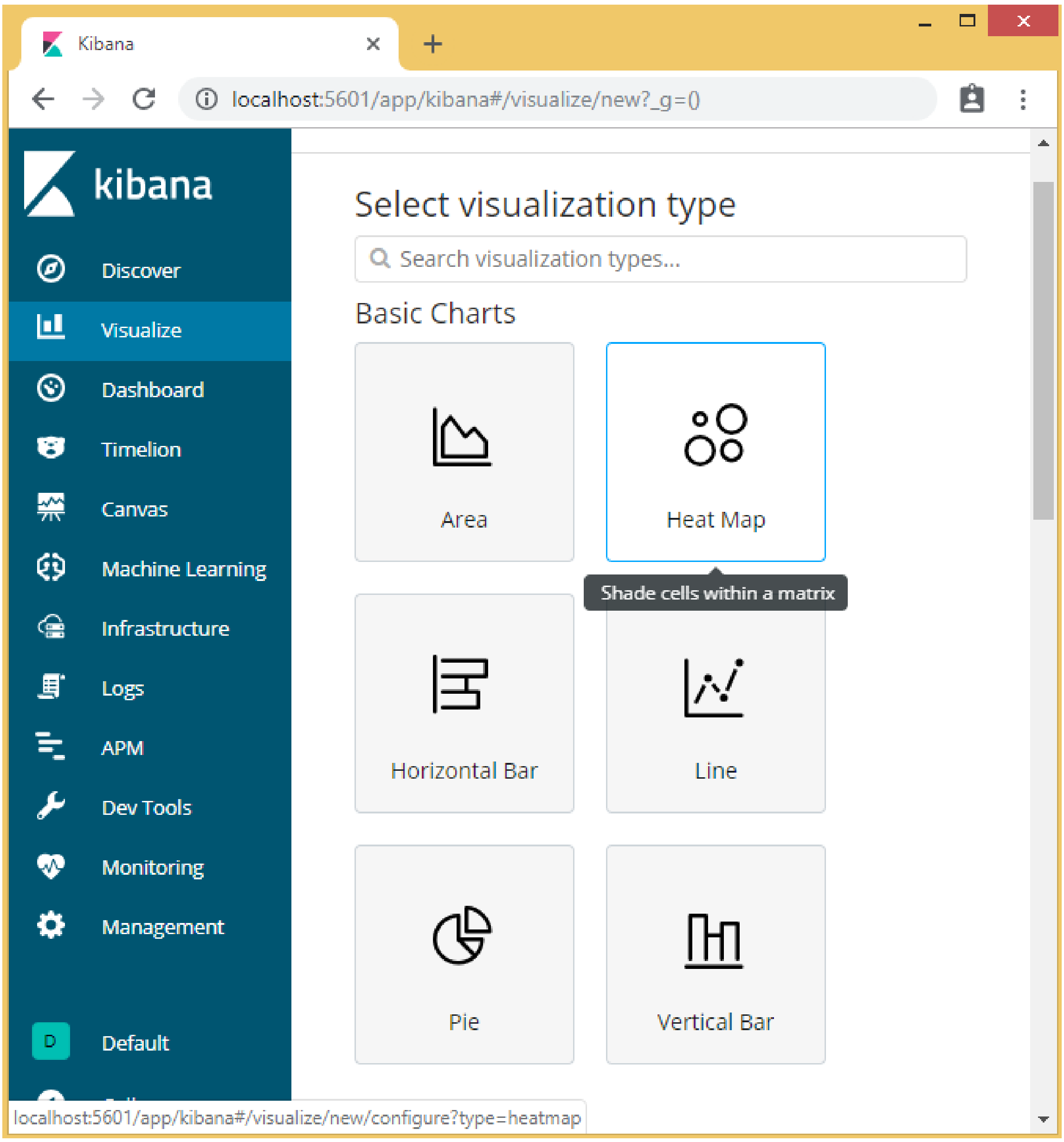
From the graph, we can observe that Russia has the highest area, followed by Canada, United States , China and Brazil. Save the visualization to use it later.

11. Kibana — Working with Heat Map

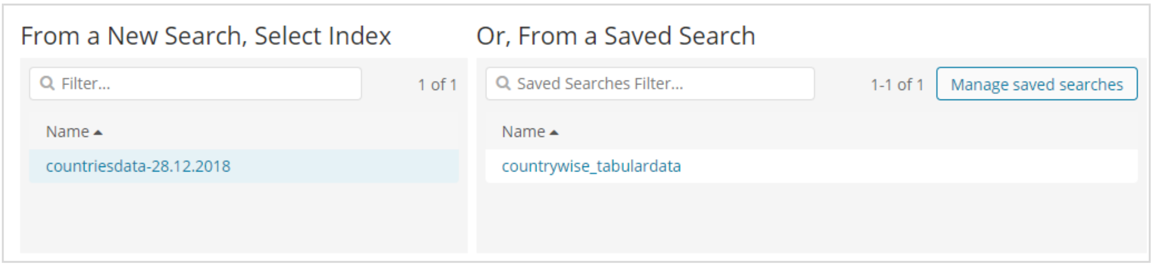
In this chapter we will understand how to work with heat map. Heat map will show the data presentation in different colours for the range selected in the data metrics.

Getting Started with Heat Map

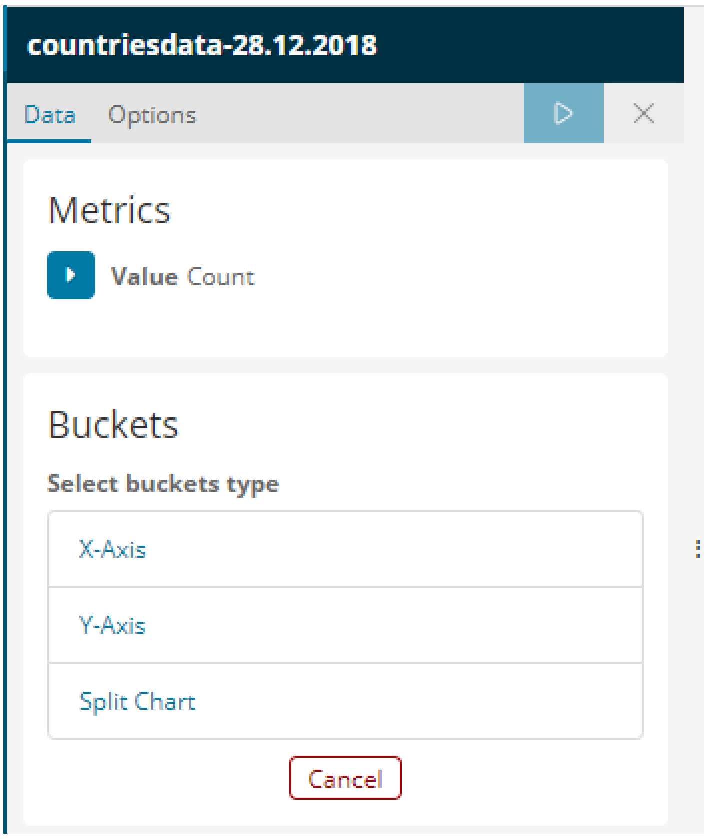
To start with, we need to create visualization by clicking on the visualization tab on the left side as shown below:


Kibana
localhost:5601/app/kibana#/visualize/new?_g=()
kibana
Discover
Visualize
Dashboard
Timelion
Canvas
Machine Learning
Infrastructure
Logs
APM
Dev Tools
Monitoring
Management
Default
Select visualization type
Search visualization types...
Basic Charts
Area
Heat Map
Shade cells within a matrix
Horizontal Bar
Line
Pie
Vertical Bar
localhost:5601/app/kibana#/visualize/new/configure?type=heatmap

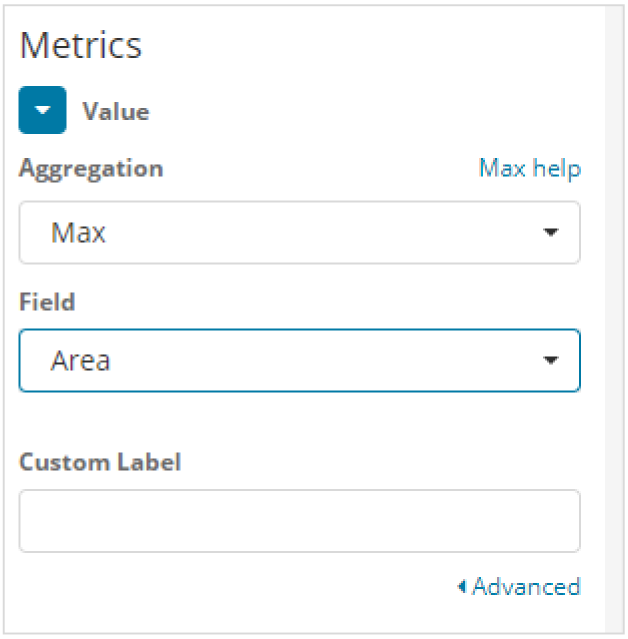
Select visualization type as heat map as shown above. It will ask you to choose the index as shown below:



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

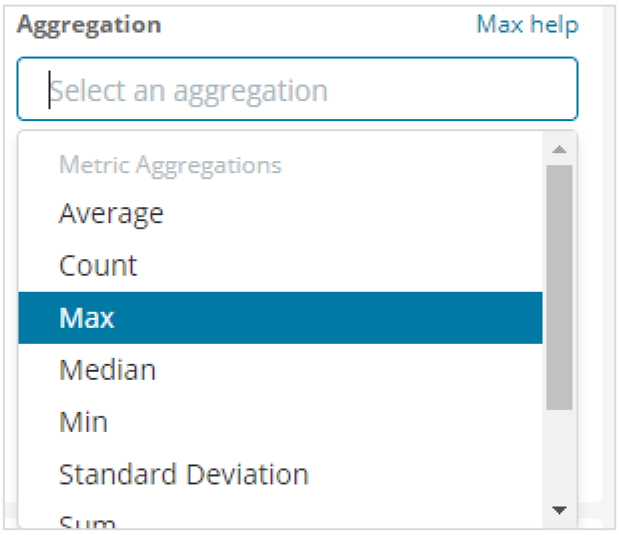
Value

Aggregation: Max

Field: Area

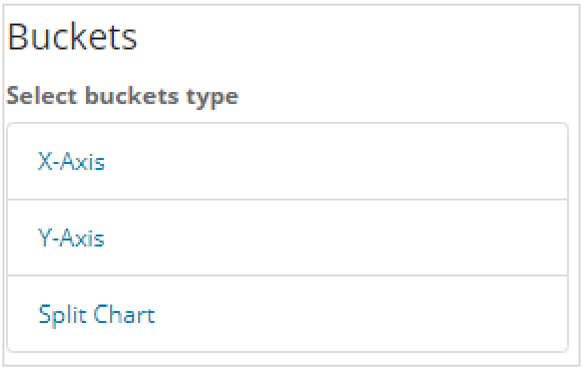
Custom Label:

Select Max Aggregation from dropdown as shown below:



We have select Max since we want to plot Max Area country wise.

Now will select the values for Buckets as shown below:



Buckets

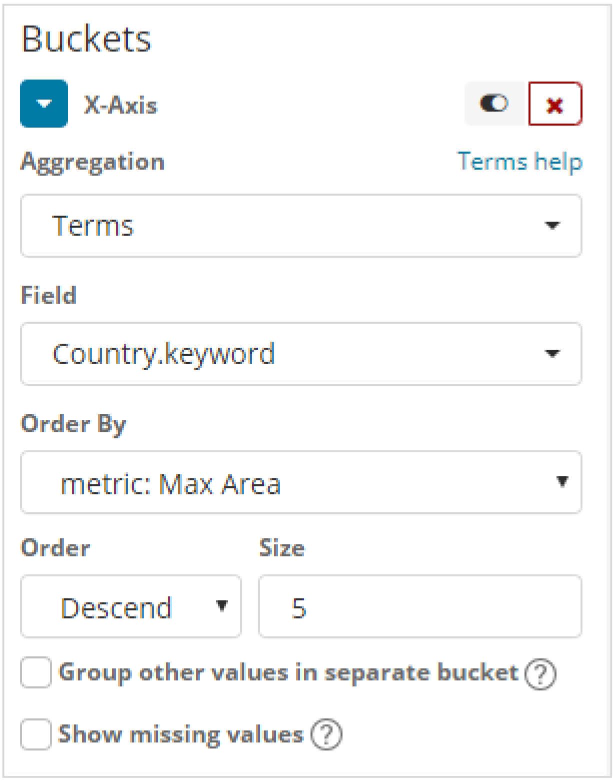
Select buckets type

X-Axis

Y-Axis

Split Chart

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



Buckets

X-Axis

Aggregation: Terms

Field: Country.keyword

Order By: metric: Max Area

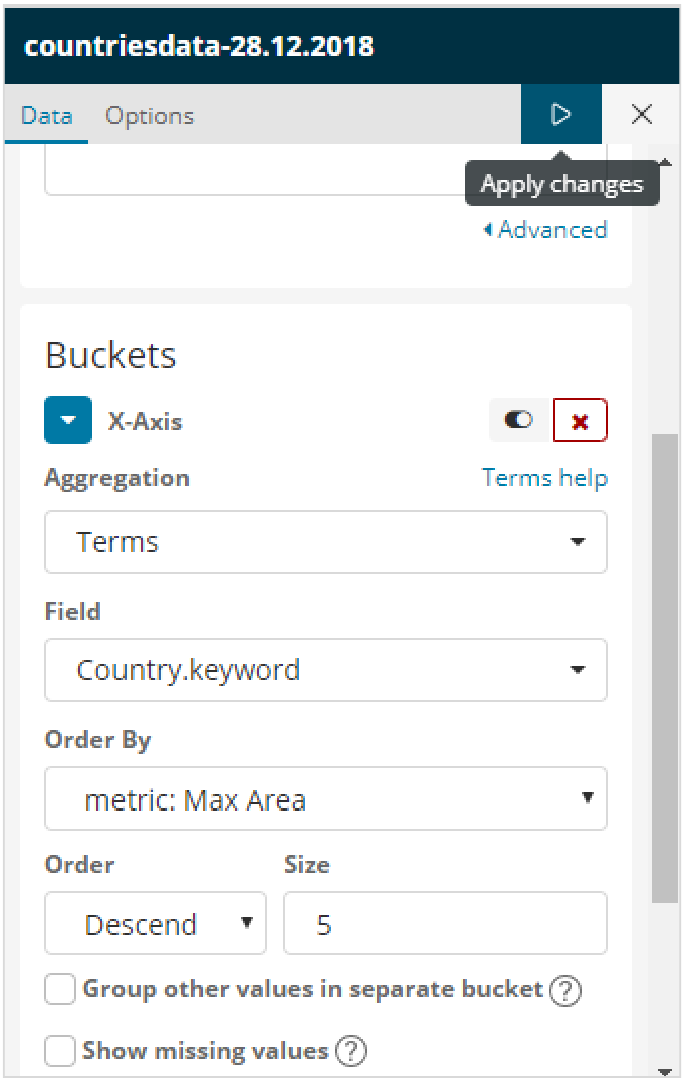
Order: Descend

Size: 5

Group other values in separate bucket

Show missing values

We have used Aggregation as Terms, Field as Country and Order By Max Area. Click on Apply Changes as shown below:



countriesdata-28.12.2018

Buckets

X-Axis

Aggregation: Terms

Field: Country.keyword

Order By: metric: Max Area

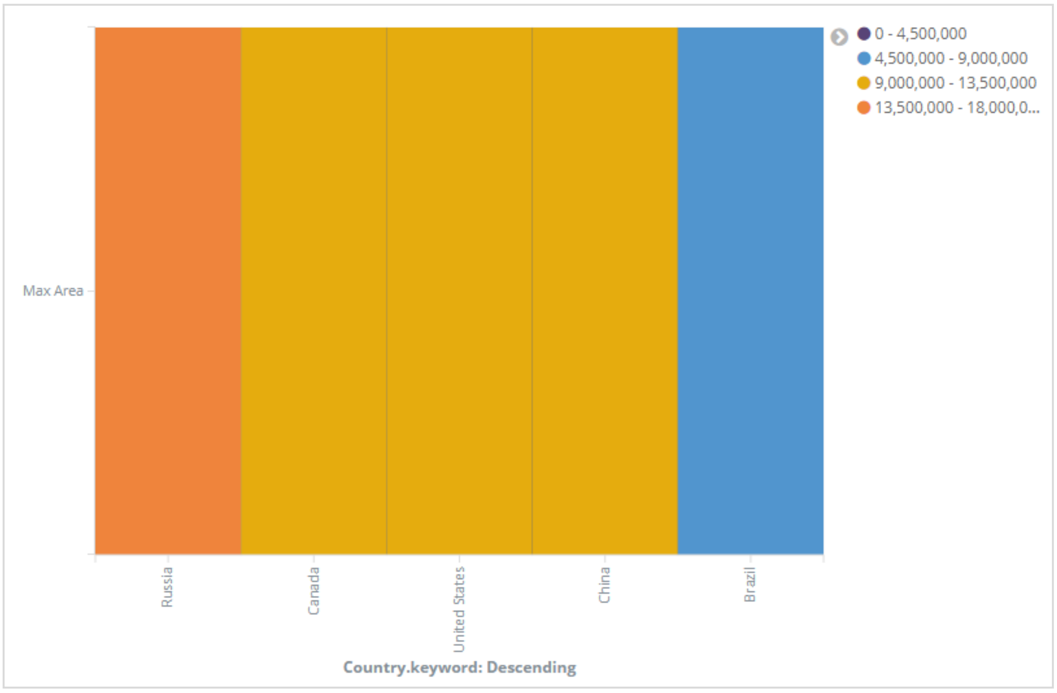
Order: Descend

Size: 5

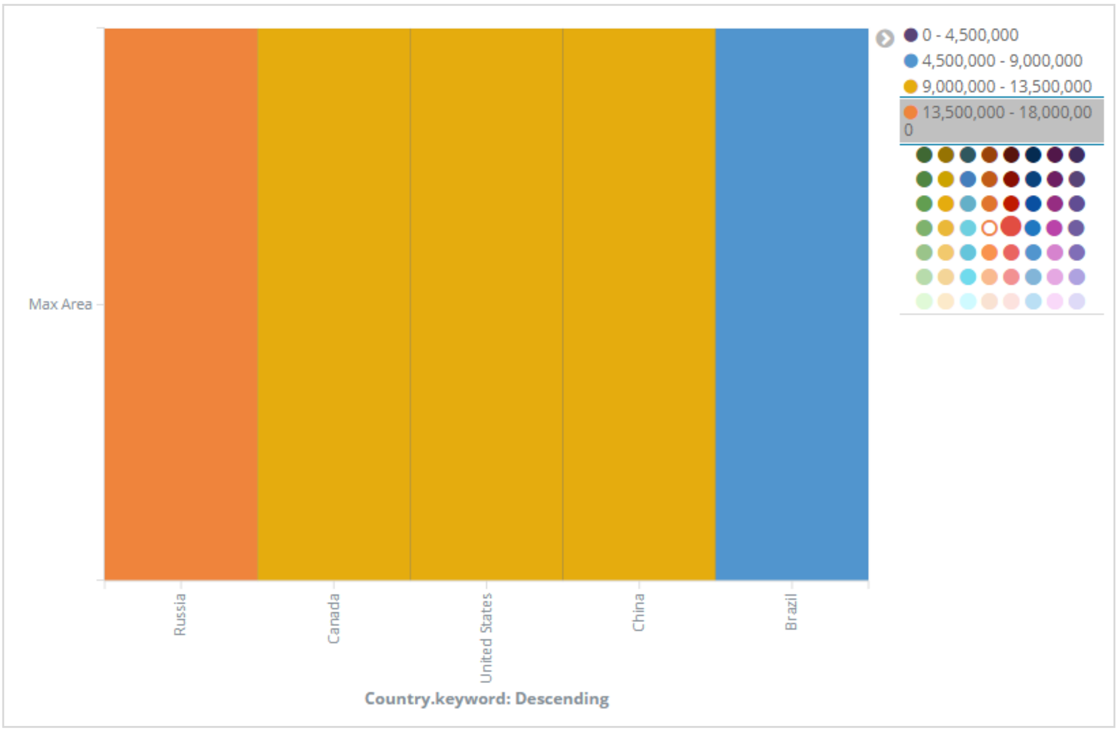
Group other values in separate bucket

Show missing values

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:



12. Kibana — Working with Coordinate Map

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:

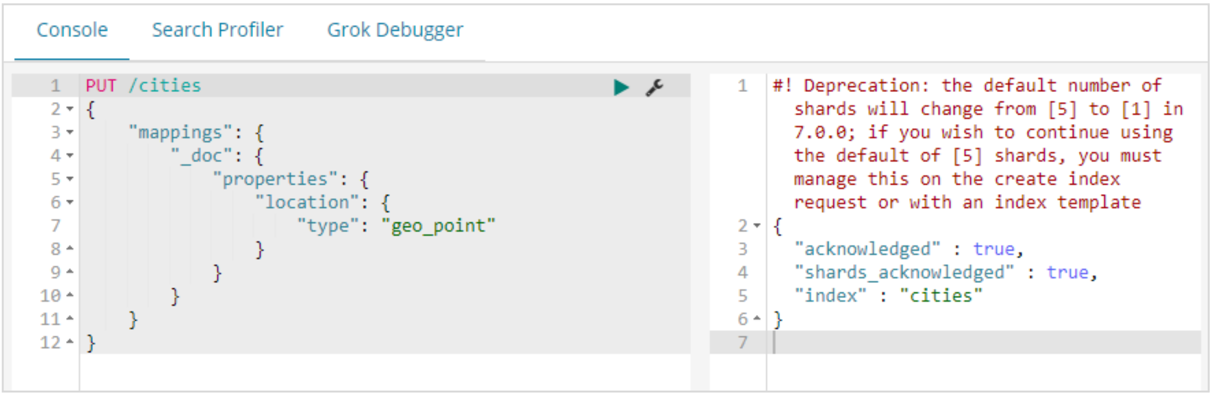
```
{"index":{"_id":1}}
{"location": "2.089330000000046,41.47367000000008", "city": "SantCugat"}
{"index":{"_id":2}}
{"location": "2.2947825000000677,41.601800991000076", "city": "Granollers"}
{"index":{"_id":3}}
{"location": "2.1105957495300474,41.5496295760424", "city": "Sabadell"}
{"index":{"_id":4}}
{"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": "2.0819450306711165,41.385491966414705", "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": "2.0819450306711165,41.385491966414705", "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 above commands in Kibana dev tools:

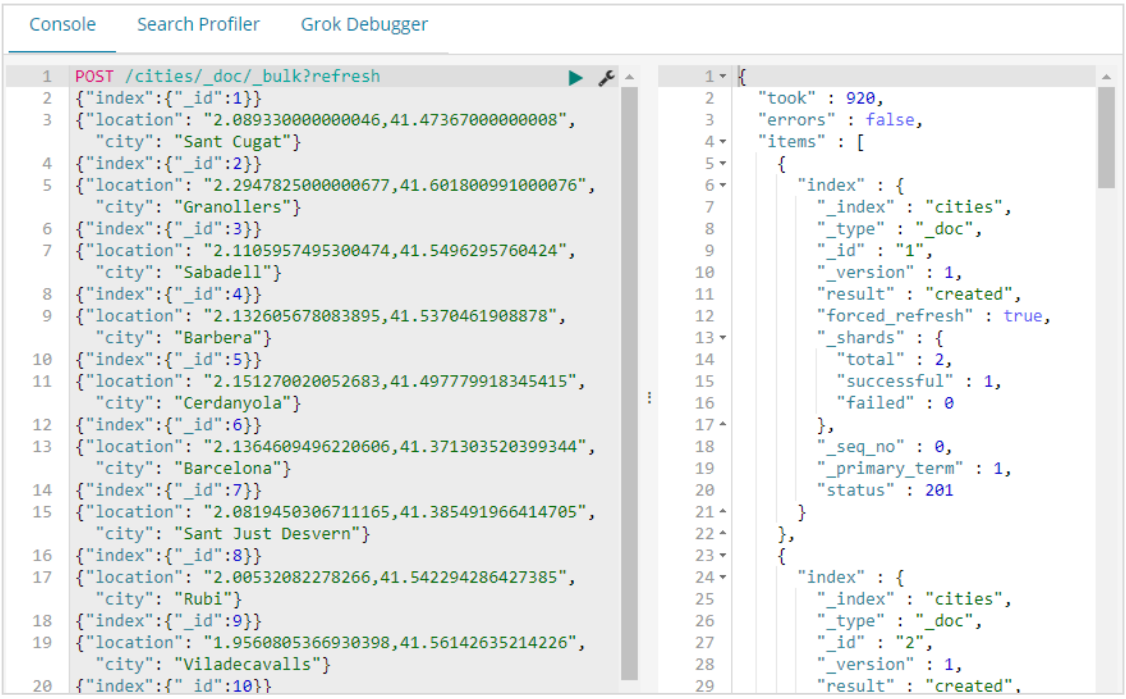


```
PUT /cities
{
    "mappings": {
        "_doc": {
            "properties": {
                "location": {
                    "type": "geo_point"
                }
            }
        }
    }
}
```

```
#! Deprecation: the default number of
   shards will change from [5] to [1] in
   7.0.0; if you wish to continue using
   the default of [5] shards, you must
   manage this on the create index
   request or with an index template
{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "cities"
}
```

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

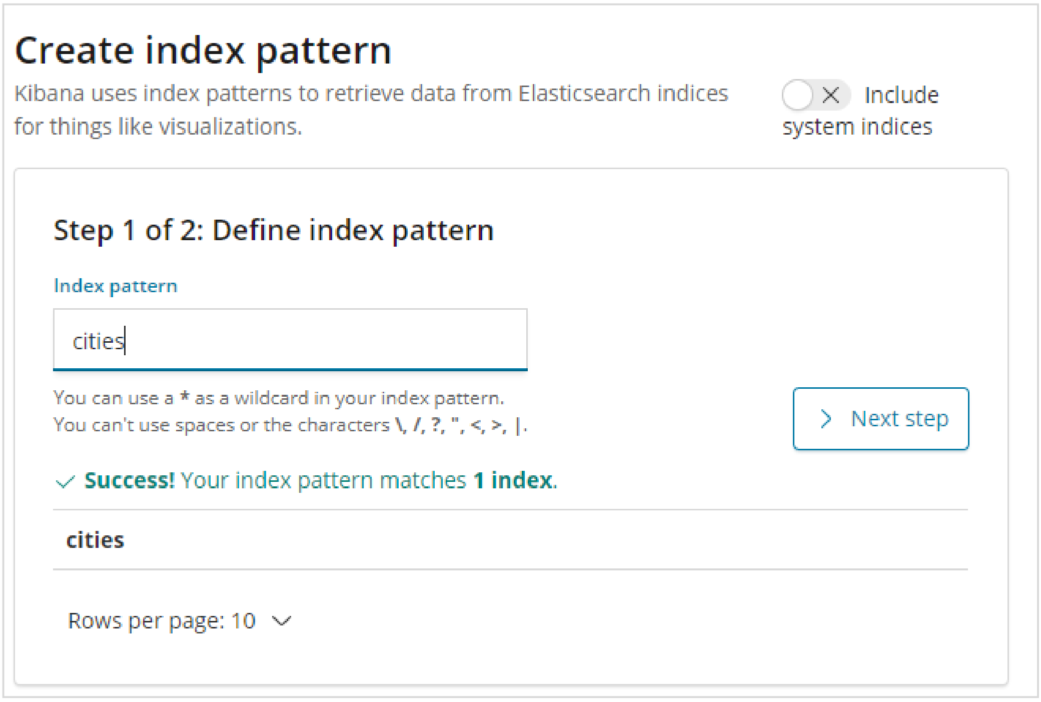
Now let's add data to the index: cities:



```
POST /cities/_doc/_bulk?refresh
{"index":{"_id":1}}
{"location": "2.089330000000046,41.47367000000008",
  "city": "Sant Cugat"}
{"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": "2.0819450306711165,41.385491966414705",
  "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}}
```

```
{
  "took" : 920,
  "errors" : false,
  "items" : [
    {
      "index" : {
        "_index" : "cities",
        "_type" : "_doc",
        "_id" : "1",
        "_version" : 1,
        "result" : "created",
        "forced_refresh" : true,
        "_shards" : {
          "total" : 2,
          "successful" : 1,
          "failed" : 0
        },
        "_seq_no" : 0,
        "_primary_term" : 1,
        "status" : 201
      }
    },
    {
      "index" : {
        "_index" : "cities",
        "_type" : "_doc",
        "_id" : "2",
        "_version" : 1,
        "result" : "created",
```

We are done creating index name cites with data. Now let us Create index pattern for cities using Management tab.



Create index pattern

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

Include system indices

Step 1 of 2: Define index pattern

Index pattern

cities

You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, ", <, >, |.

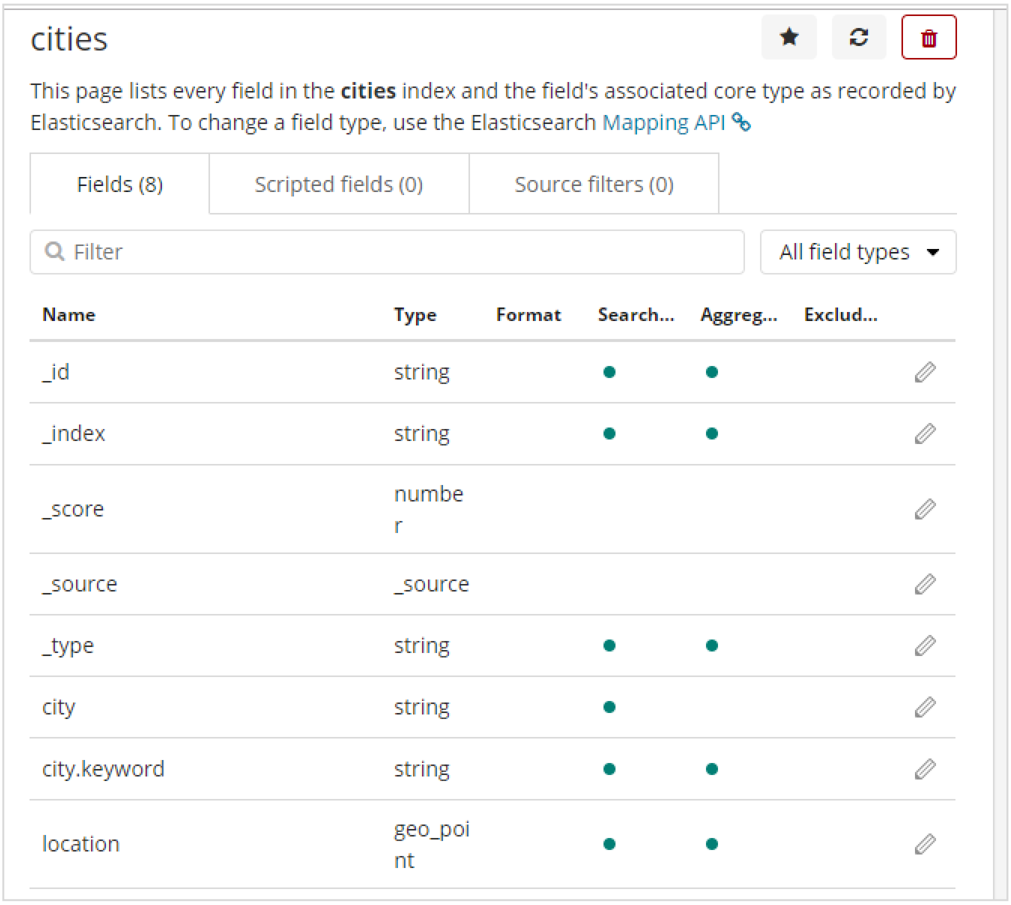
Next step

✓ **Success!** Your index pattern matches **1 index**.

cities

Rows per page: 10

The details of fields inside cities index are shown here:



cities

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

Fields (8) | Scripted fields (0) | Source filters (0)

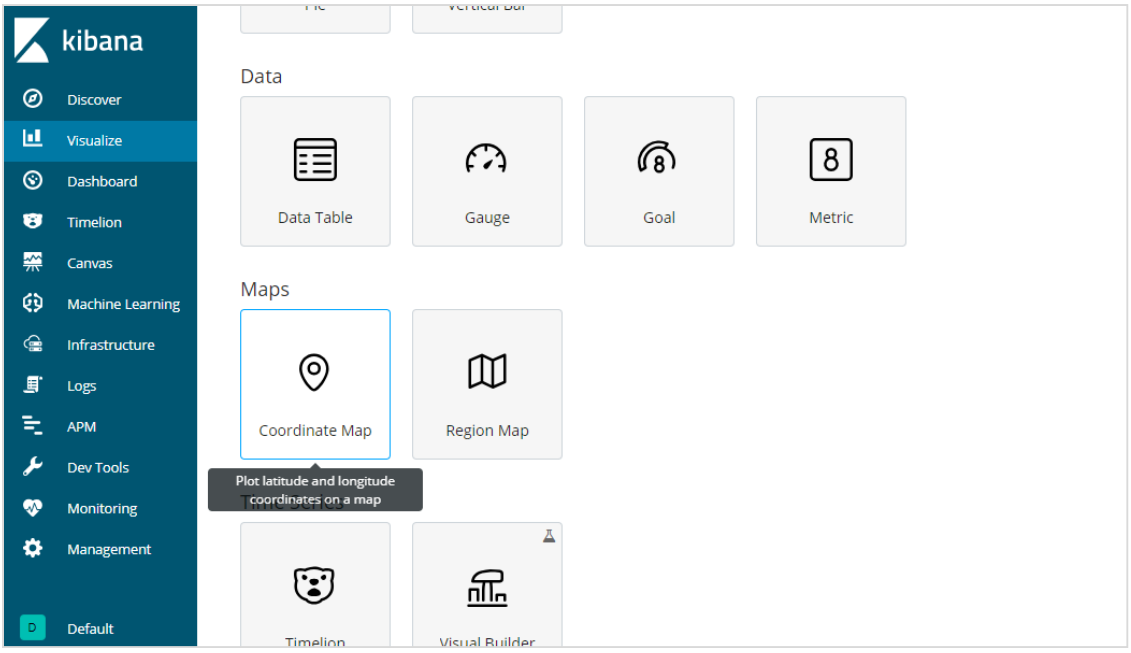
Filter | All field types

Name	Type	Format	Search...	Aggreg...	Exclud...
_id	string		●	●	
_index	string		●	●	
_score	number				
_source	_source				
_type	string		●	●	
city	string		●		
city.keyword	string		●	●	
location	geo_point		●	●	

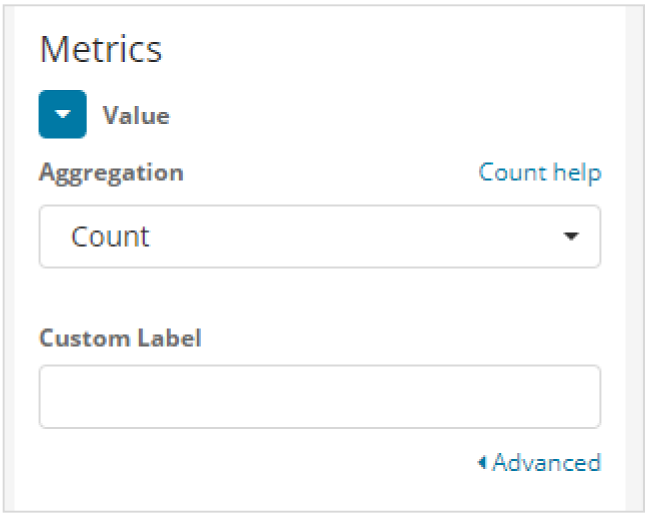
We can see that location is of type geo_point. We can now use it to create visualization.

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:

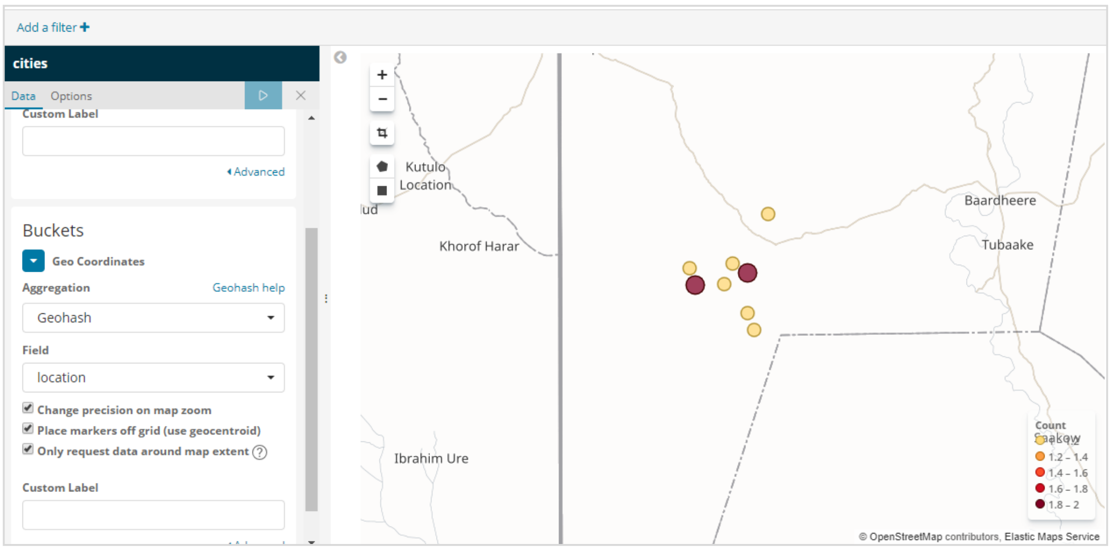

Metrics

Value

Aggregation: Count

Custom Label:

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.

13. Kibana — Working with Region Map

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:

```
{"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.

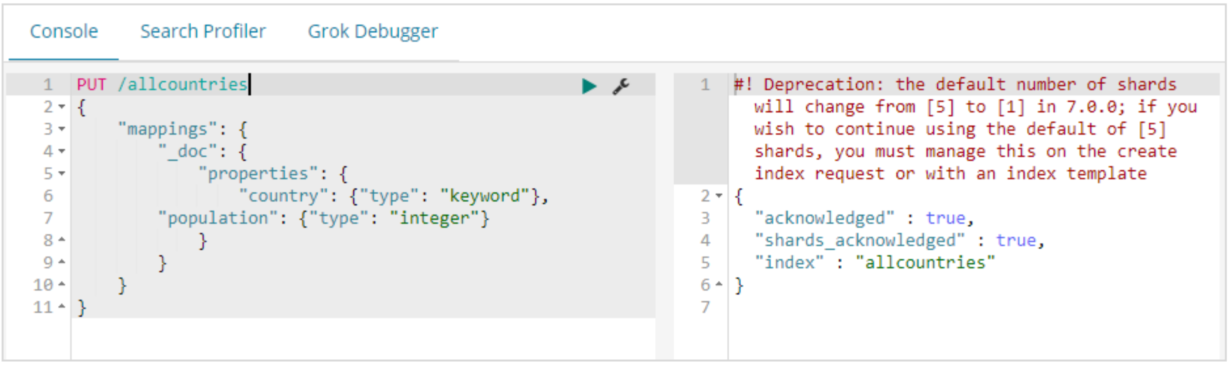
Now, go to Kibana Dev Tools and execute following queries:

```
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:**

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

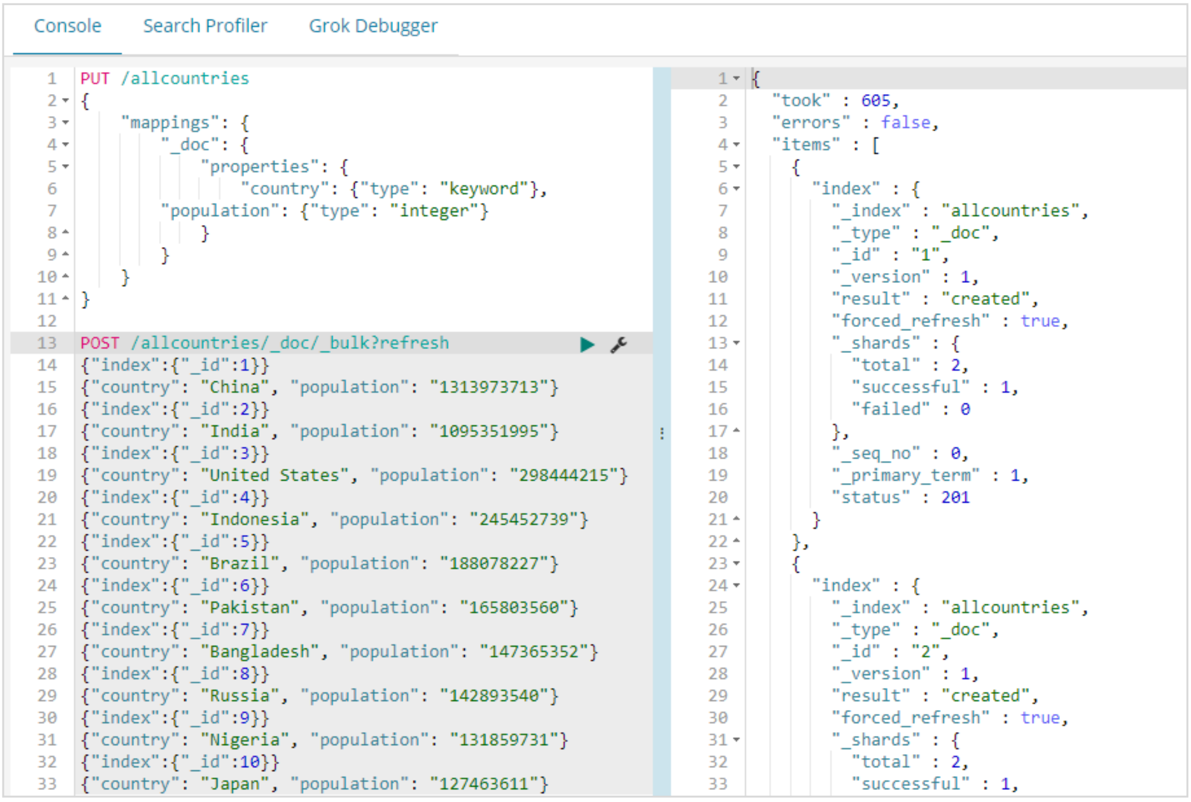
Note: To work with region maps we need to specify the field type to be used with aggregation as type as keyword.



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

```
#! Deprecation: the default number of shards
  will change from [5] to [1] in 7.0.0; if you
  wish to continue using the default of [5]
  shards, you must manage this on the create
  index request or with an index template
{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "allcountries"
}
```

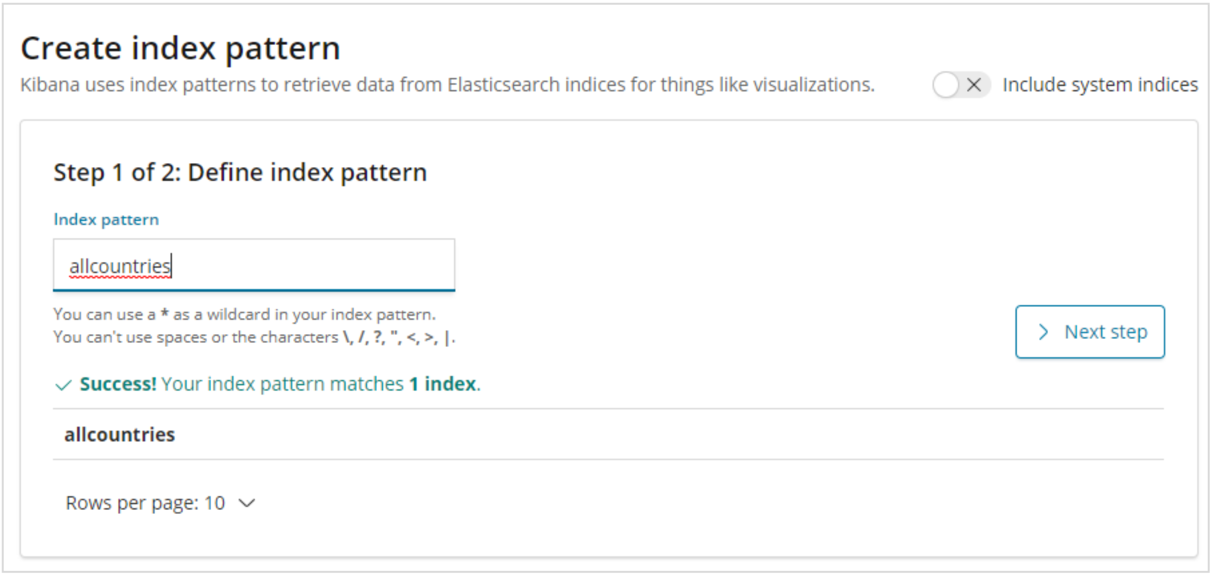
Once done, upload the data using _bulk command.



```
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"}
```

```
{
  "took" : 605,
  "errors" : false,
  "items" : [
    {
      "index" : {
        "_index" : "allcountries",
        "_type" : "_doc",
        "_id" : "1",
        "_version" : 1,
        "result" : "created",
        "forced_refresh" : true,
        "_shards" : {
          "total" : 2,
          "successful" : 1,
          "failed" : 0
        },
        "_seq_no" : 0,
        "_primary_term" : 1,
        "status" : 201
      }
    },
    {
      "index" : {
        "_index" : "allcountries",
        "_type" : "_doc",
        "_id" : "2",
        "_version" : 1,
        "result" : "created",
        "forced_refresh" : true,
        "_shards" : {
          "total" : 2,
          "successful" : 1,
```

We will now create index pattern. Go to Kibana Management tab and select create index pattern.



Create index pattern

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

Include system indices

Step 1 of 2: Define index pattern

Index pattern

allcountries

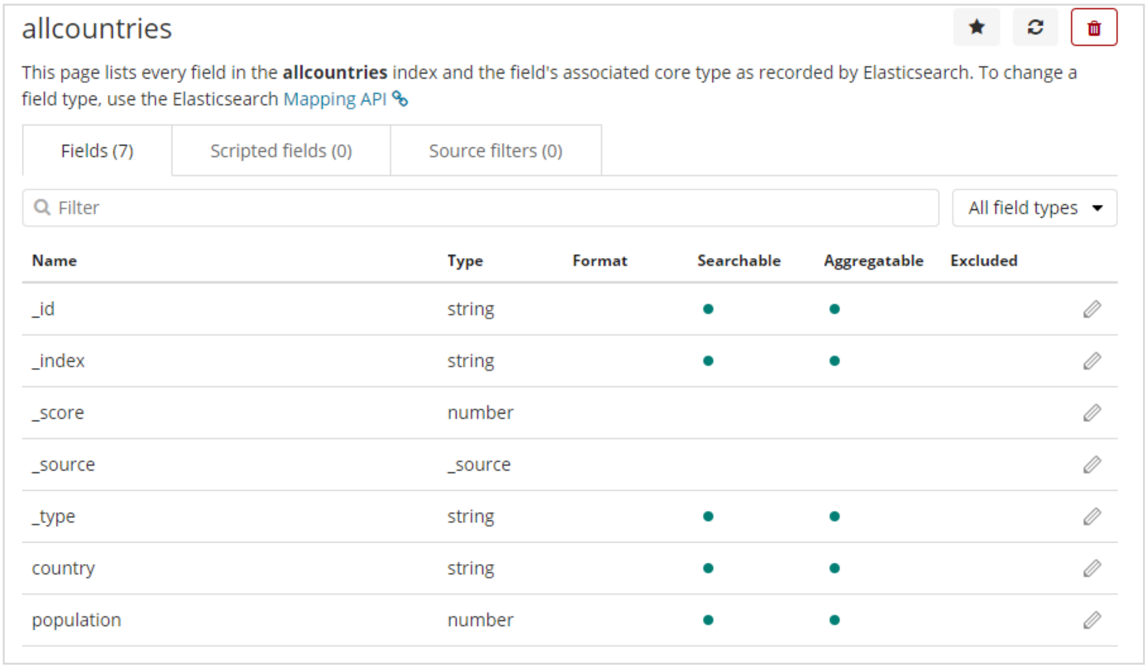
You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, ", <, >, |.

Success! Your index pattern matches 1 index.

allcountries

Rows per page: 10

Here are the fields displayed from allcountries index.



allcountries

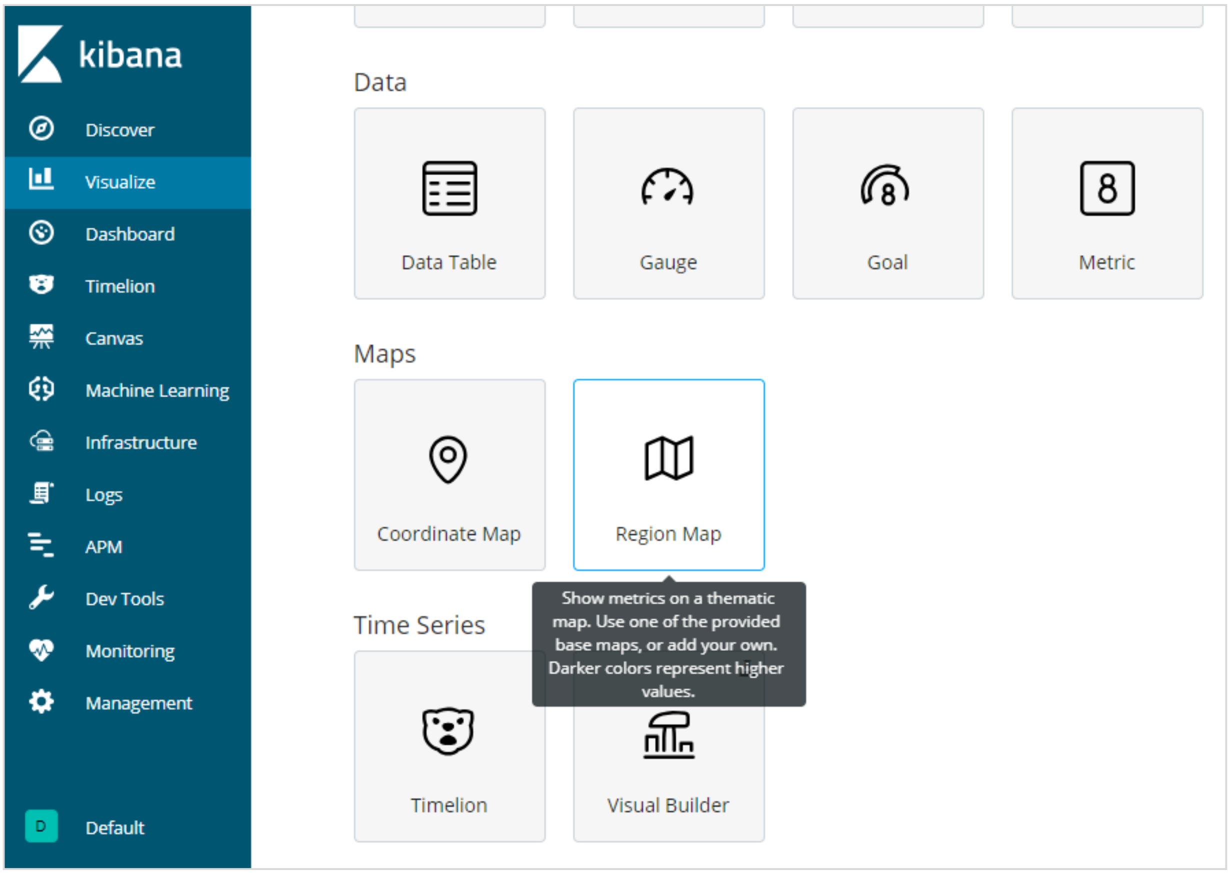
This page lists every field in the **allcountries** index and the field's associated core type as recorded by Elasticsearch. To change a field type, use the Elasticsearch Mapping API

Fields (7) | Scripted fields (0) | Source filters (0)

Name	Type	Format	Searchable	Aggregatable	Excluded
_id	string		●	●	
_index	string		●	●	
_score	number				
_source	_source				
_type	string		●	●	
country	string		●	●	
population	number		●	●	

Getting Started with Region Maps

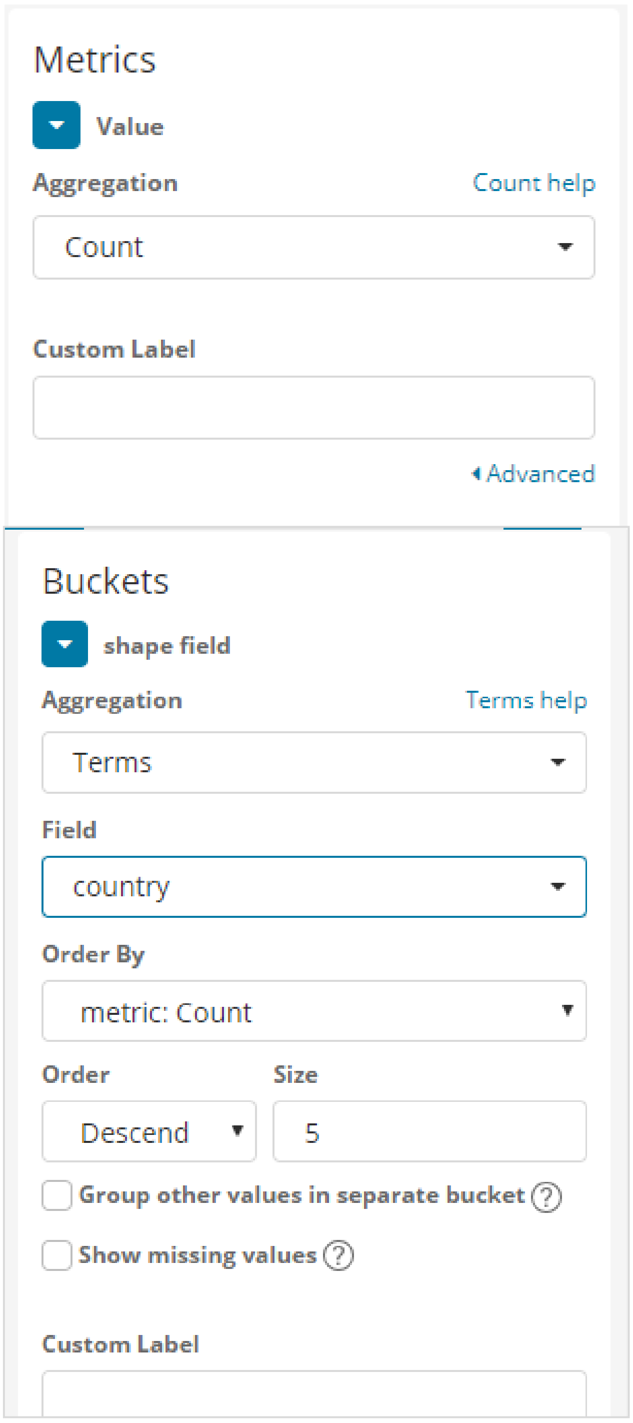
We will now create the visualization using Region Maps. Go to Visualization and select Region Maps.



Show metrics on a thematic map. Use one of the provided base maps, or add your own. Darker colors represent higher values.

Once done select index as *allcountries* and proceed.

Select Aggregation Metrics and Bucket Metrics as shown below:



Metrics

Value

Aggregation: Count

Custom Label:

Buckets

shape field

Aggregation: Terms

Field: country

Order By: metric: Count

Order: Descend

Size: 5

☐ Group other values in separate bucket

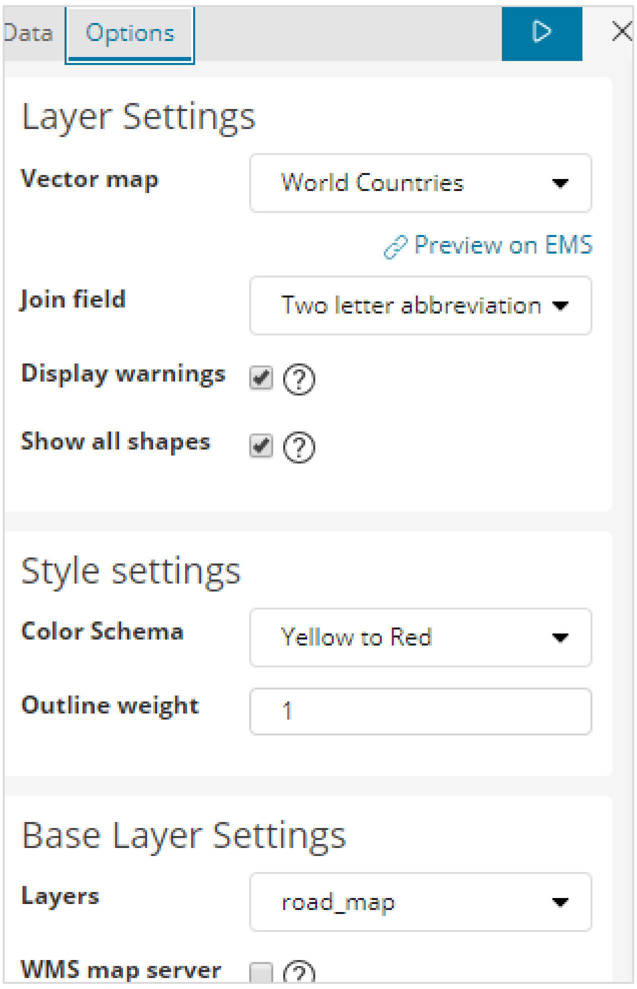
☐ Show missing values

Custom Label:

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:



Layer Settings

Vector map: World Countries

Preview on EMS

Join field: Two letter abbreviation

Display warnings ✓

Show all shapes ✓

Style settings

Color Schema: Yellow to Red

Outline weight: 1

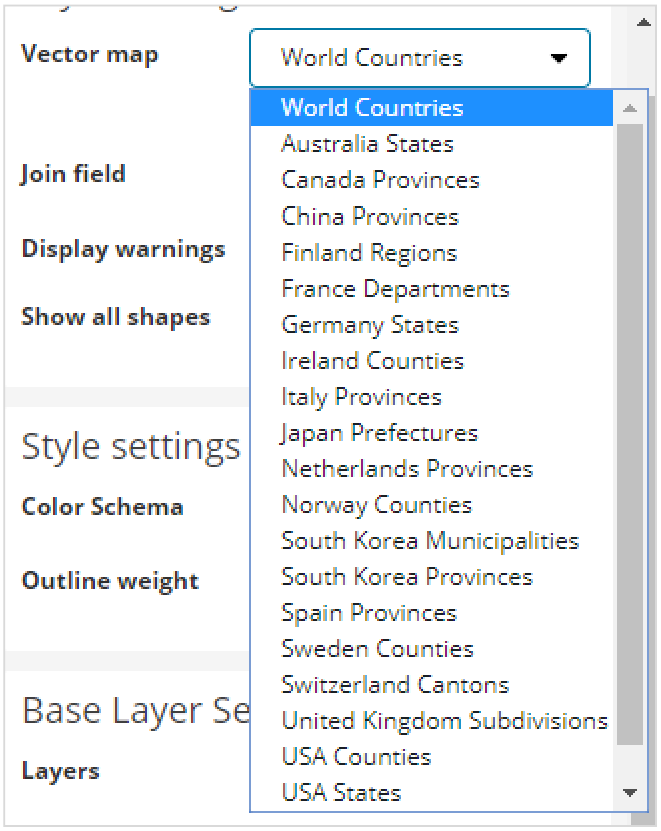
Base Layer Settings

Layers: road_map

WMS map server

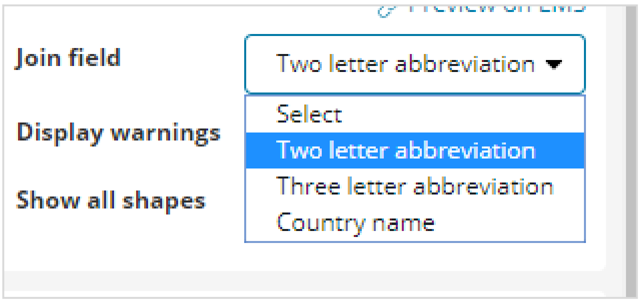
The options tab has Layer Settings configuration which are required to plot the data on the world map.

A Vector Map has the following options:



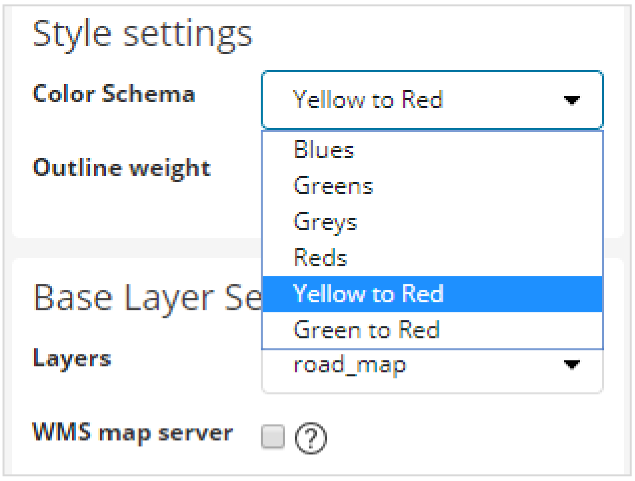
Here we will select world countries as i have countries data.

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:


Style settings

Color Schema: Yellow to Red

Outline weight

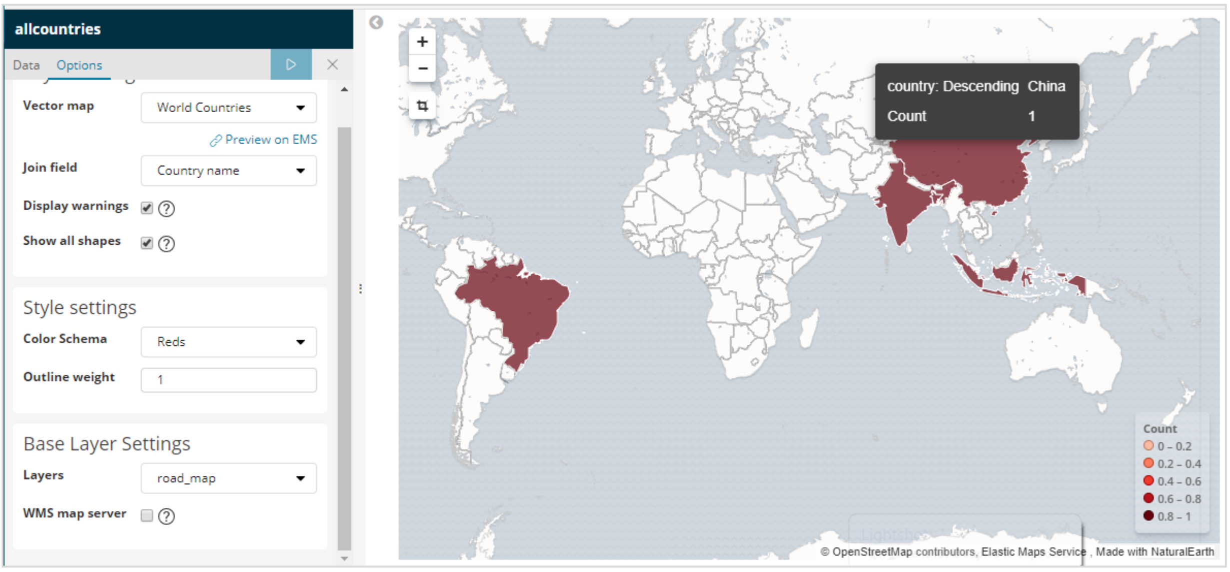
Base Layer Se

Layers: road_map

WMS map server ☐

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:


allcountries

Vector map: World Countries

Join field: Country name

Display warnings ✓

Show all shapes ✓

Style settings

Color Schema: Reds

Outline weight: 1

Base Layer Settings

Layers: road_map

WMS map server ☐

country: Descending China
Count 1

Count
● 0 – 0.2
● 0.2 – 0.4
● 0.4 – 0.6
● 0.6 – 0.8
● 0.8 – 1

© OpenStreetMap contributors, Elastic Maps Service , Made with NaturalEarth

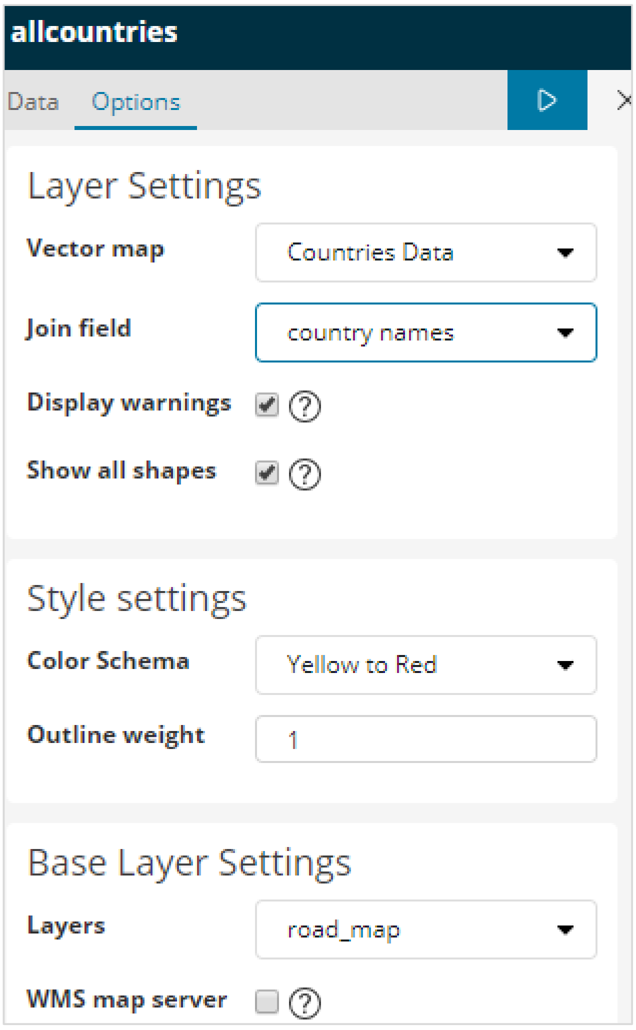
Self-hosted Vector Map and Join Field in Kibana

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:

```
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:



allcountries

Data Options

Layer Settings

Vector map: Countries Data

Join field: country names

Display warnings ✓

Show all shapes ✓

Style settings

Color Schema: Yellow to Red

Outline weight: 1

Base Layer Settings

Layers: road_map

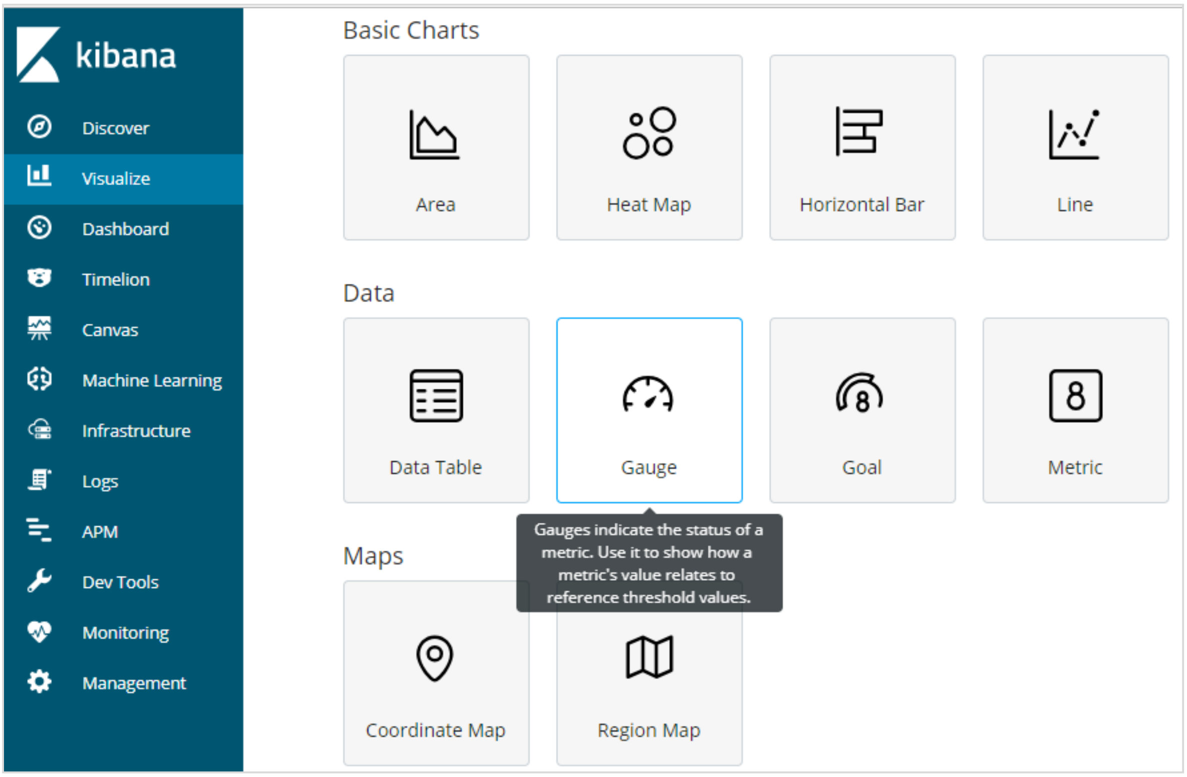
WMS map server ☐

14. Kibana — Working with Guage and Goal

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.

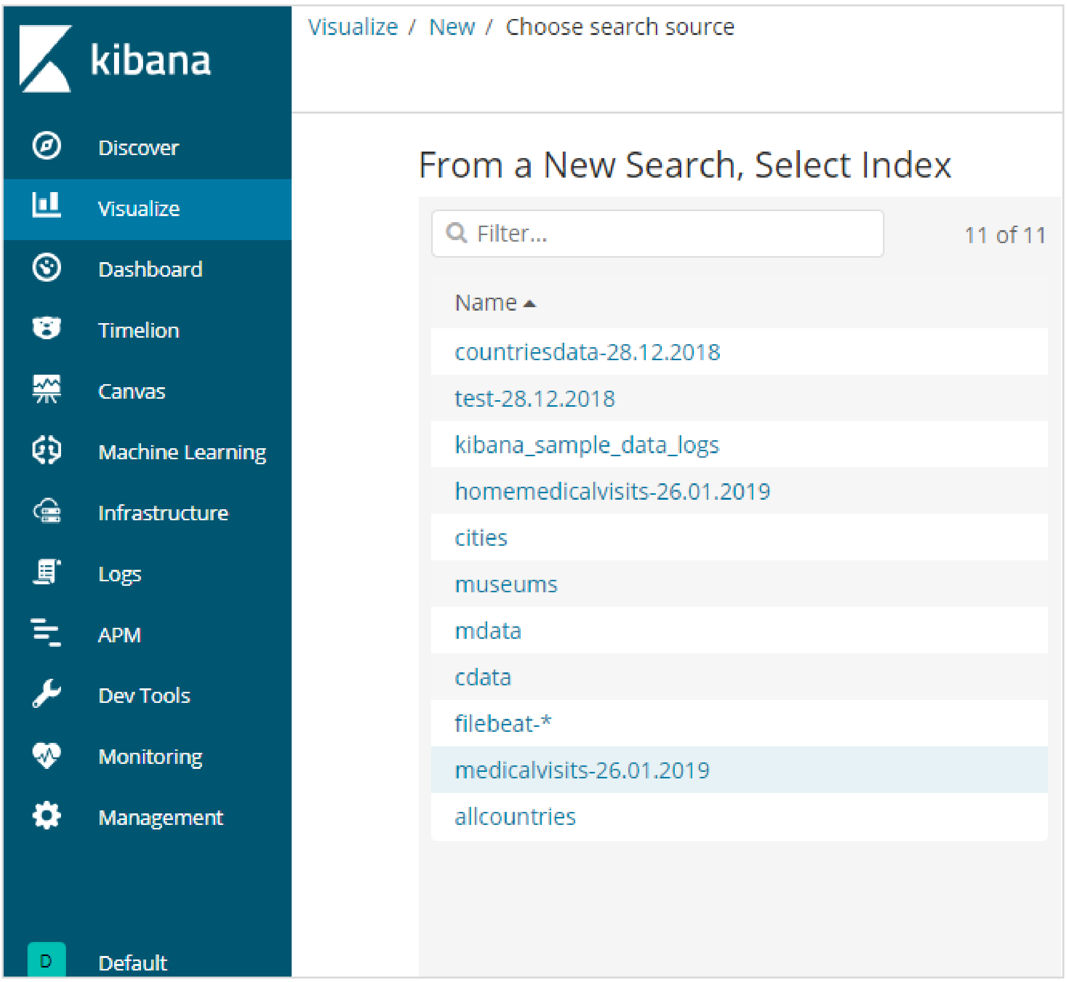
Working with Gauge

To start using Gauge, go to visualization and select Visualize tab from Kibana UI.



Gauges indicate the status of a metric. Use it to show how a metric's value relates to reference threshold values.

Click on Gauge and select the index you want to use.

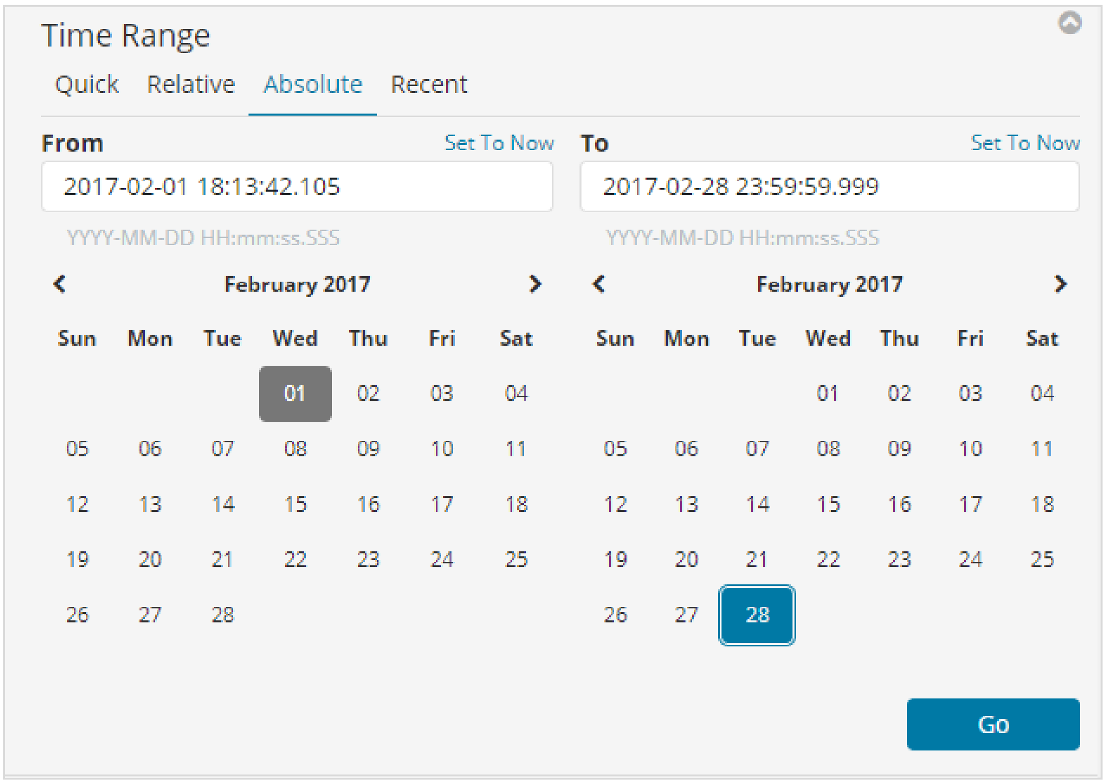


From a New Search, Select Index

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

We are going to work on *medicalvisits-26.01.2019* index.

Select the time range of February 2017

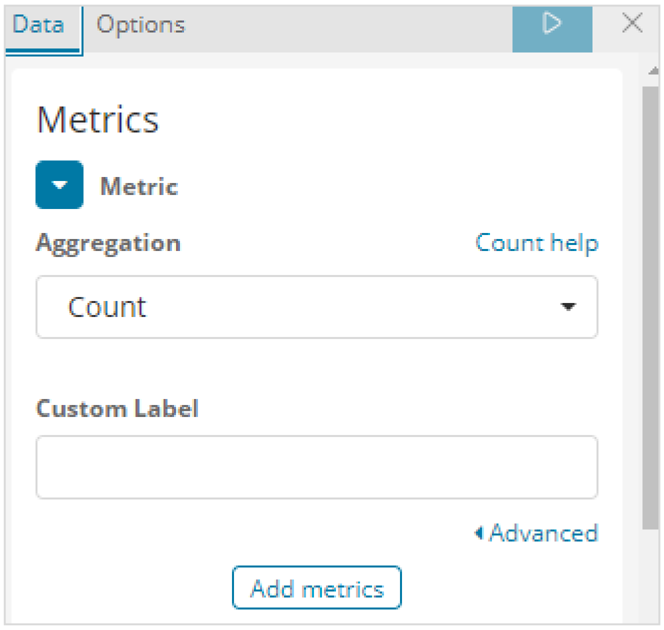


From: 2017-02-01 18:13:42.105

To: 2017-02-28 23:59:59.999

YYYY-MM-DD HH:mm:ss.SSS

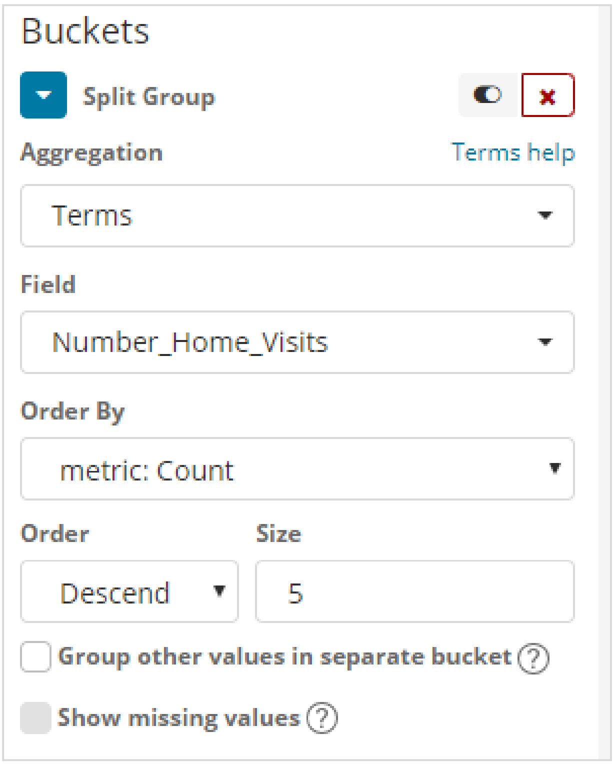
Now you can select the metric and bucket aggregation.



Aggregation: Count

Custom Label:

We have selected the metric aggregation as Count.


Buckets

Split Group

Aggregation: Terms

Field: Number_Home_Visits

Order By: metric: Count

Order: Descend

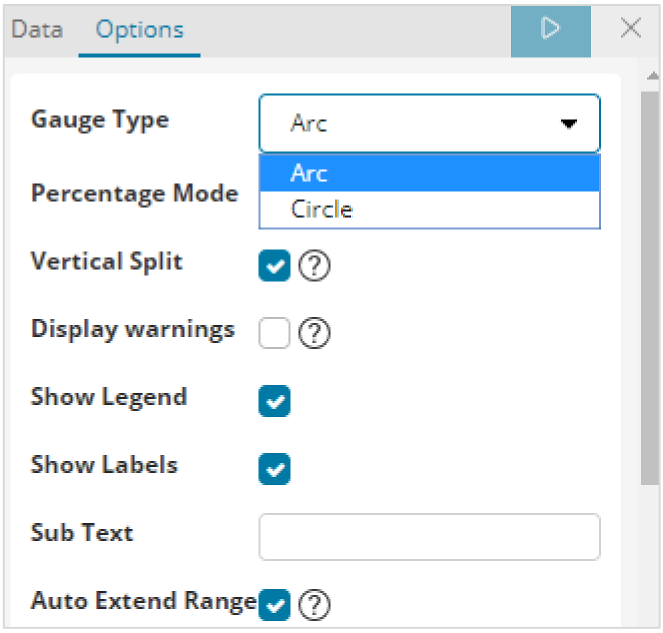
Size: 5

Group other values in separate bucket

Show missing values

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: Arc

Percentage Mode

Vertical Split ✓

Display warnings

Show Legend ✓

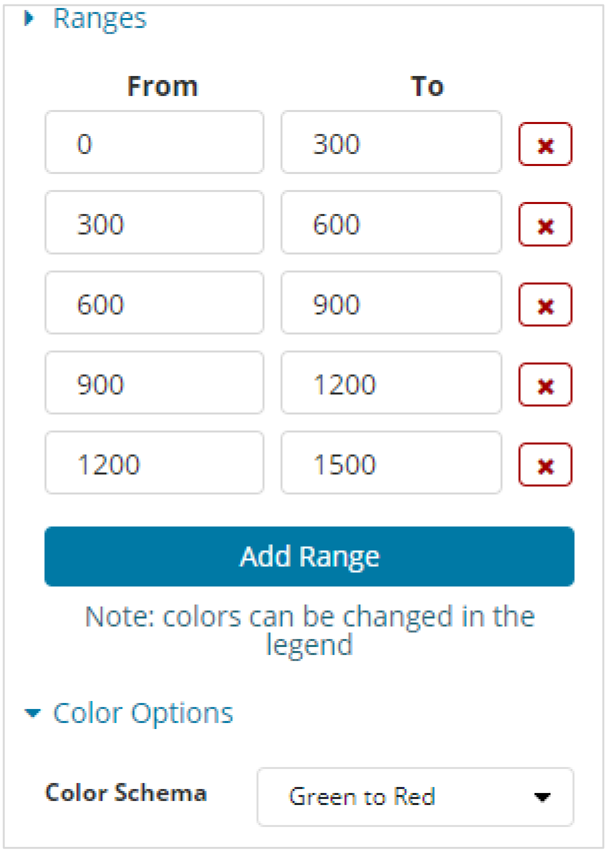
Show Labels ✓

Sub Text

Auto Extend Range ✓

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:

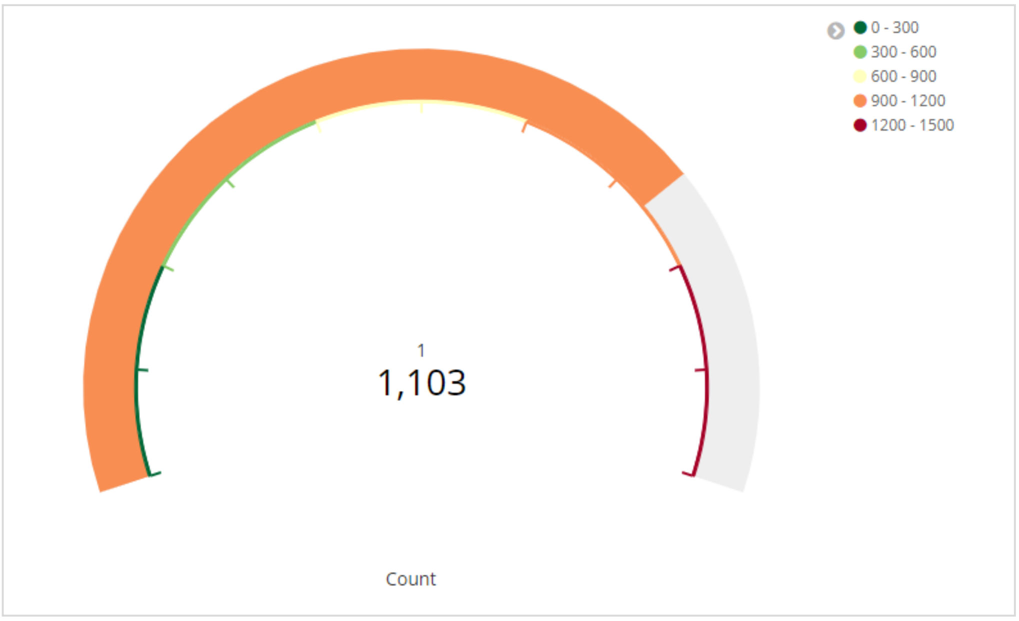


From	To
0	300
300	600
600	900
900	1200
1200	1500

Color Schema: Green to Red

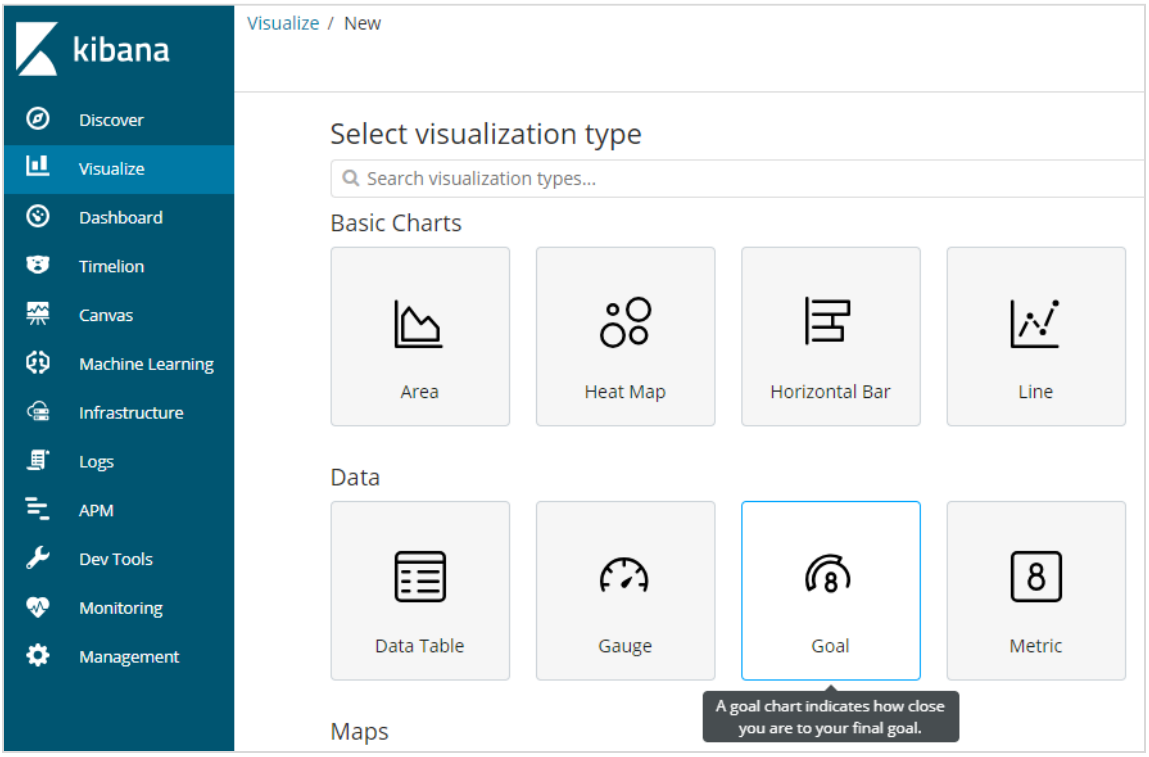
The colour selected is Green To Red.

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



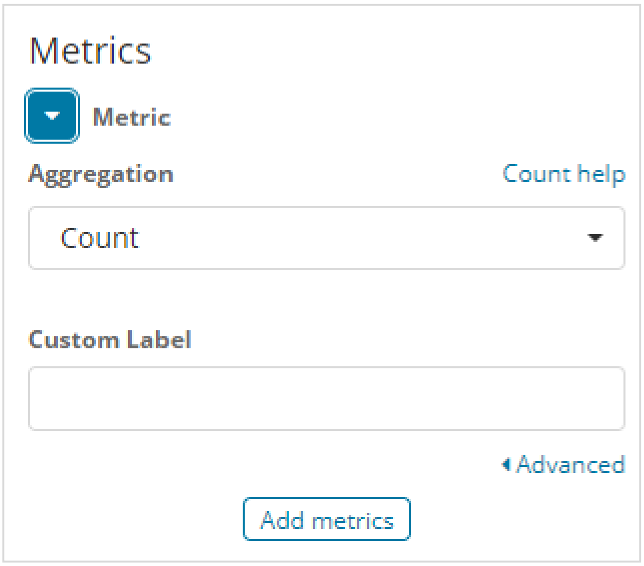
Working with Goal

Go to Visualize Tab and select Goal as shown below:



Select Goal and select the index.

Use *medicalvisits-26.01.2019* as the index.



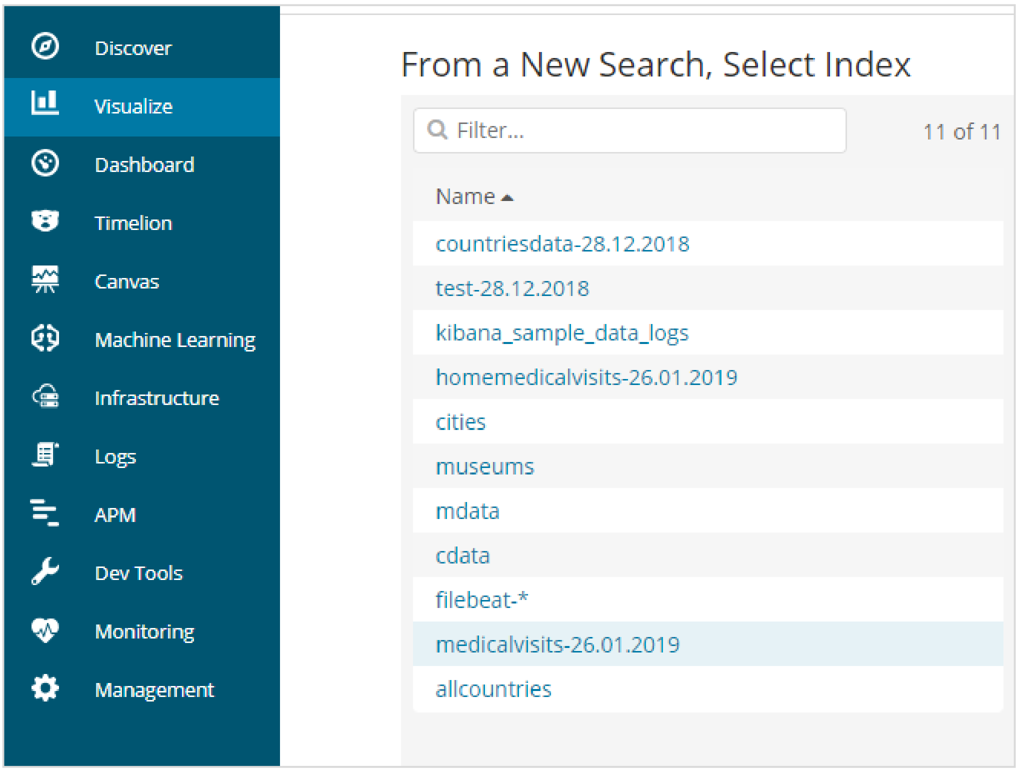
Metrics

Metric

Aggregation: Count

Custom Label:

Select the metric aggregation and bucket aggregation.
Metric Aggregation:



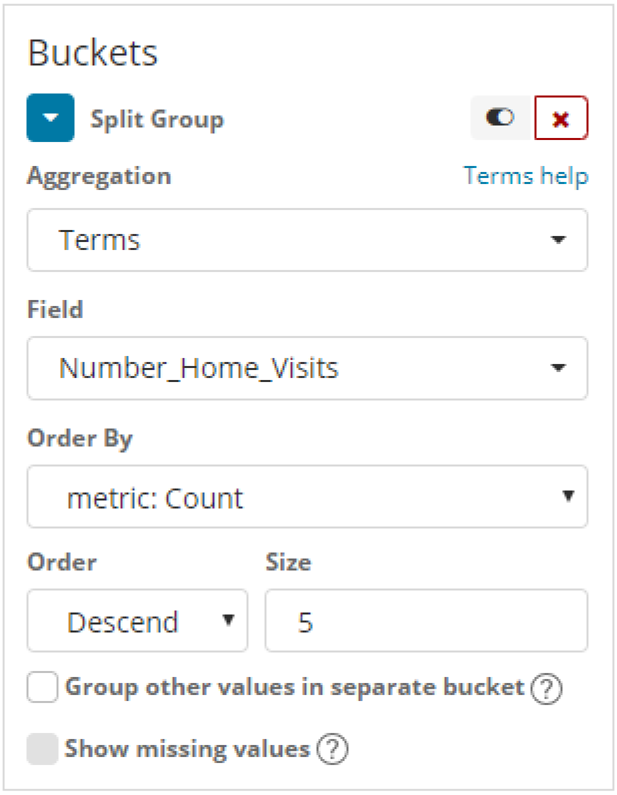
From a New Search, Select Index

11 of 11

Name
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

We have selected Count as the metric aggregation.

Bucket Aggregation



Buckets

Split Group

Aggregation: Terms

Field: Number_Home_Visits

Order By: metric: Count

Order: Descend

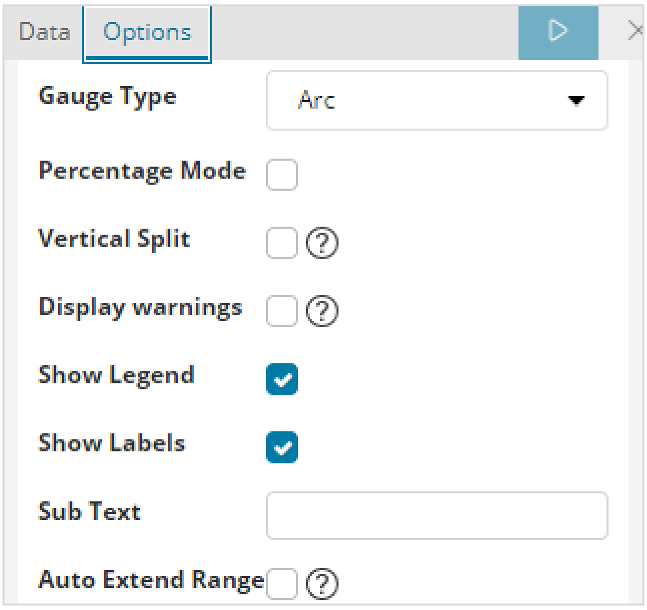
Size: 5

Group other values in separate bucket

Show missing values

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

The options selected are as follows:



Gauge Type: Arc

Percentage Mode

Vertical Split

Display warnings

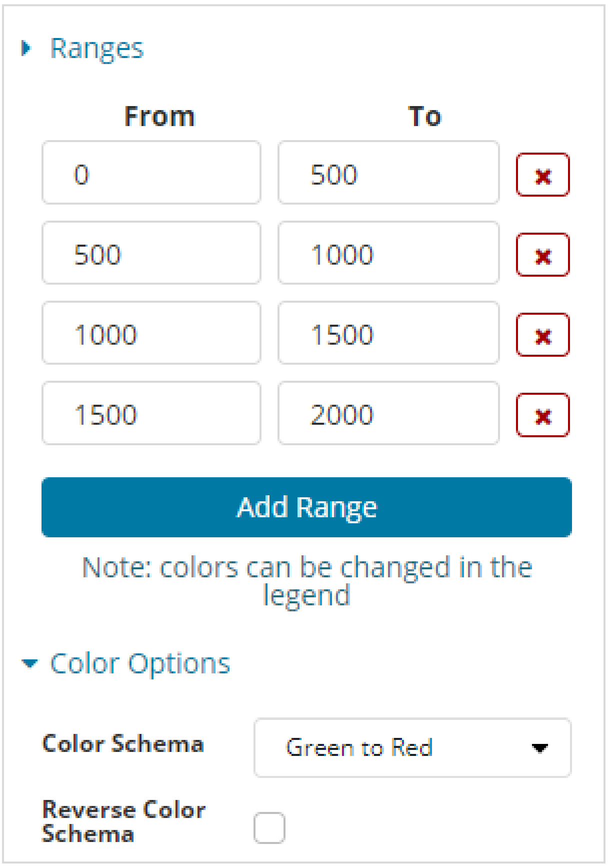
Show Legend ✓

Show Labels ✓

Sub Text

Auto Extend Range

The Range selected is as follows:



Ranges

From	To
0	500
500	1000
1000	1500
1500	2000

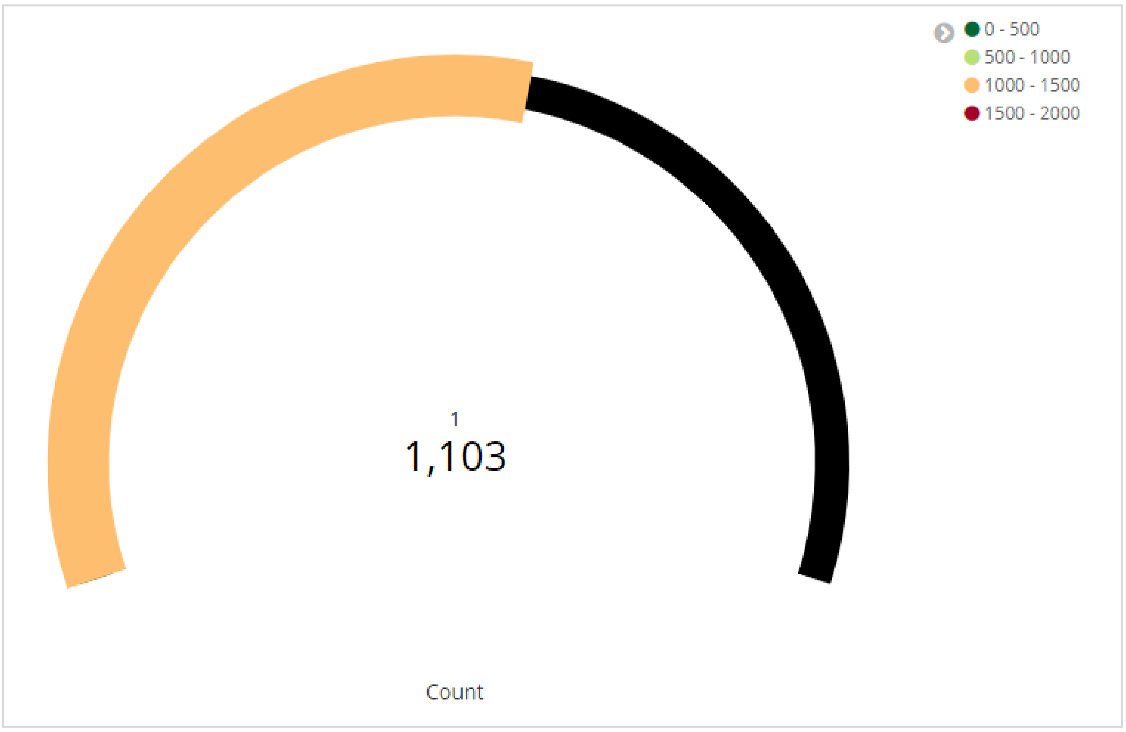
Note: colors can be changed in the legend

Color Options

Color Schema: Green to Red

Reverse Color Schema: ☐

Click on Analyze and you see the goal displayed as follows:



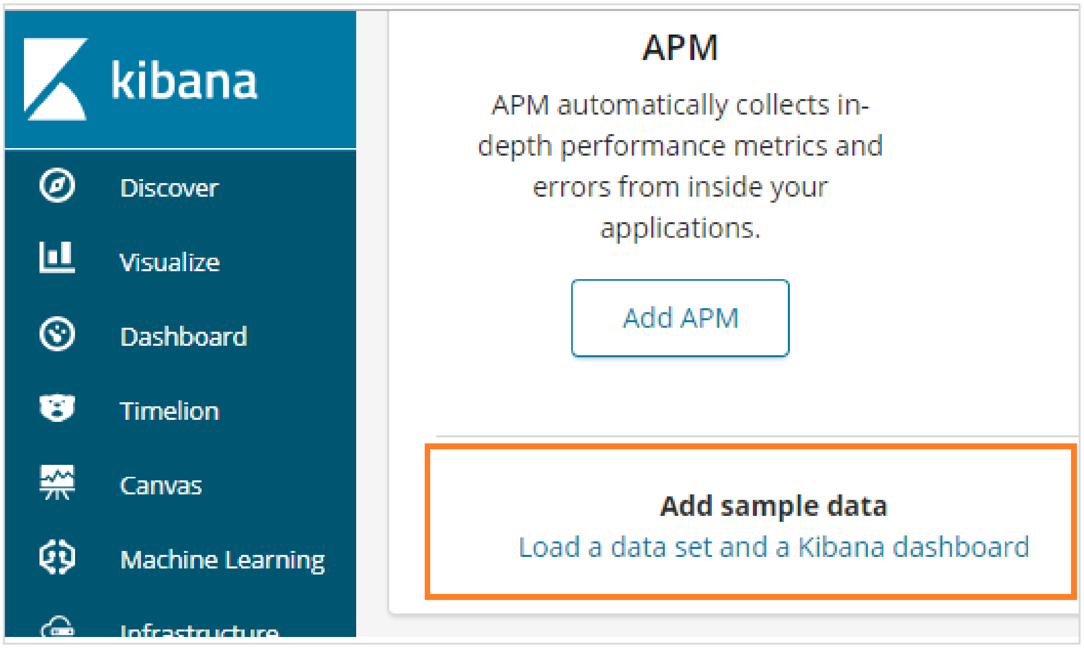
15. Kibana — Working with Canvas

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.

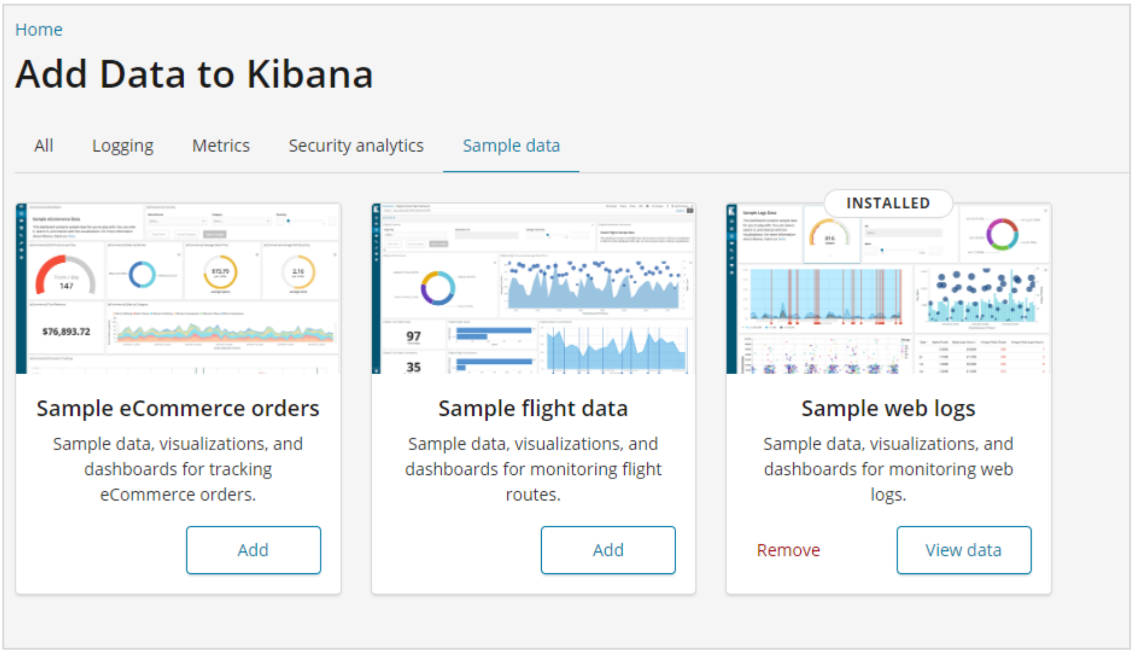
Loading Sample Data for Canvas Creation

To get the sample data go to Kibana home page and click on Add sample data as shown below:



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

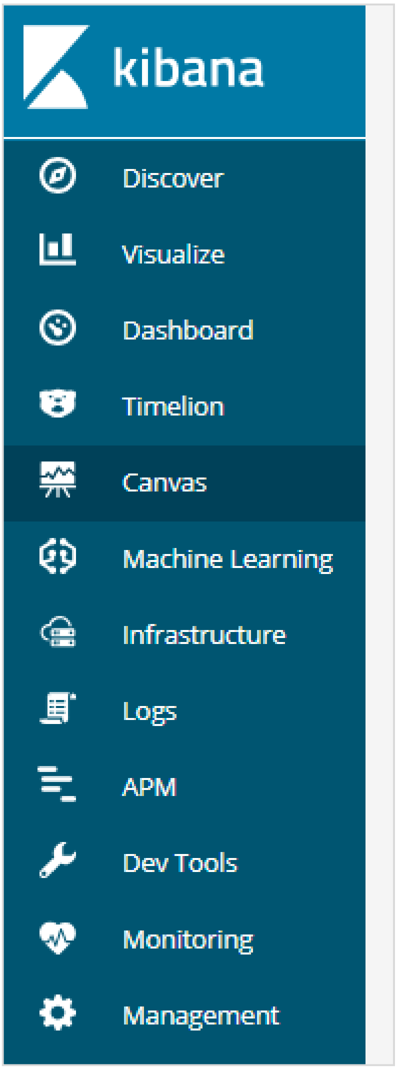
Click on Load a data set and a Kibana dashboard. It will take you to the screen as shown below:


Home
Add Data to Kibana
All
Logging
Metrics
Security analytics
Sample data
INSTALLED
Sample eCommerce orders
Sample data, visualizations, and dashboards for tracking eCommerce orders.
Add
Sample flight data
Sample data, visualizations, and dashboards for monitoring flight routes.
Add
Sample web logs
Sample data, visualizations, and dashboards for monitoring web logs.
Remove
View data

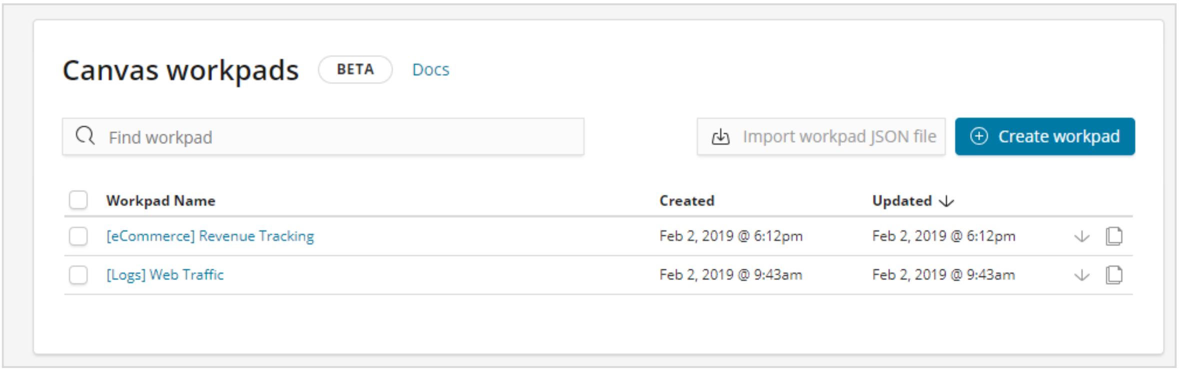
Click on Add button for Sample eCommerce orders. It will take some time to load the sample data. Once done you will get an alert message showing " Sample eCommerce data loaded."

Getting Started with Canvas Visualization

Now go to Canvas Visualization as shown below:



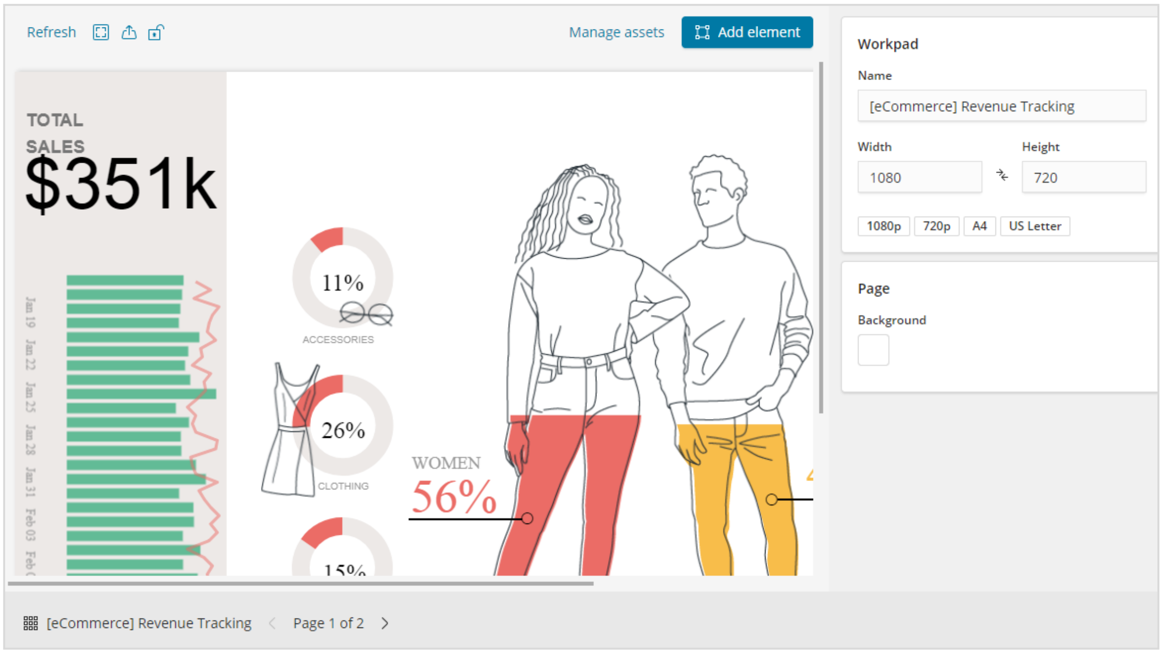
Click on Canvas and it will display screen as shown below:



Workpad Name	Created	Updated
[eCommerce] Revenue Tracking	Feb 2, 2019 @ 6:12pm	Feb 2, 2019 @ 6:12pm
[Logs] Web Traffic	Feb 2, 2019 @ 9:43am	Feb 2, 2019 @ 9:43am

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:



Workpad

Name: [eCommerce] Revenue Tracking

Width: 1080

Height: 720

1080p 720p A4 US Letter

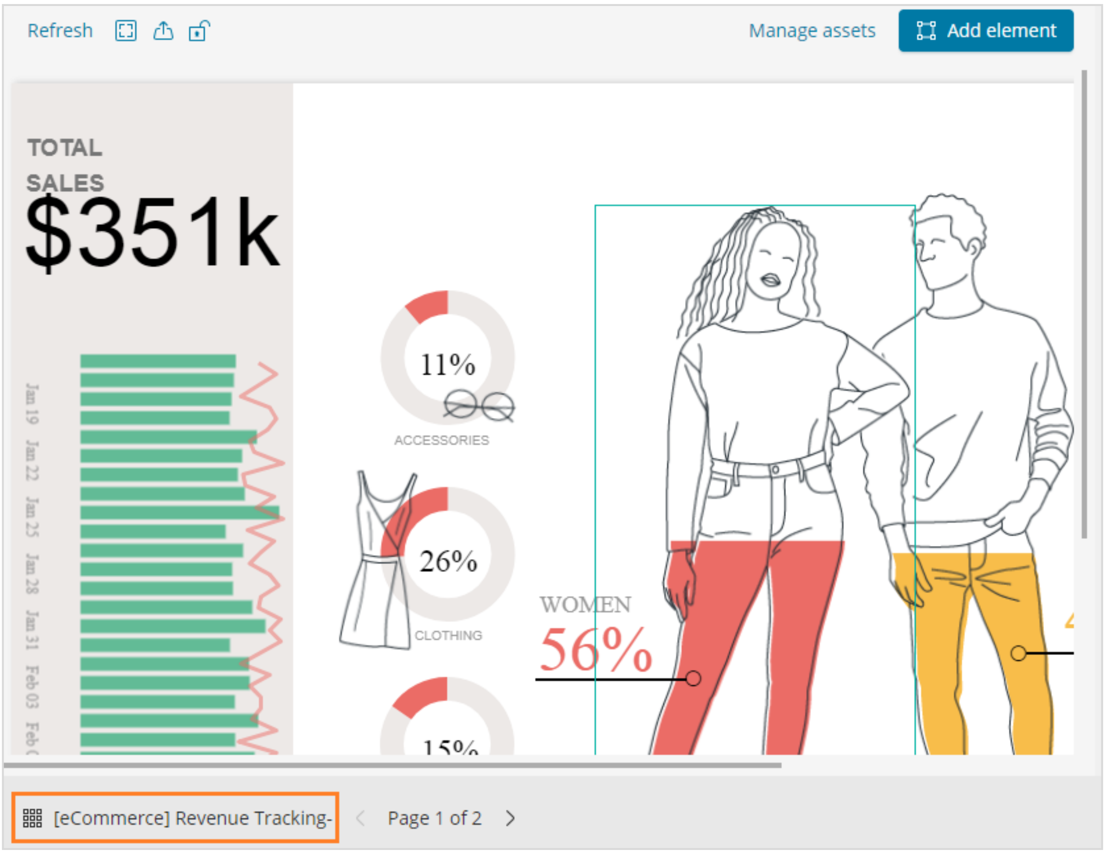
Page

Background

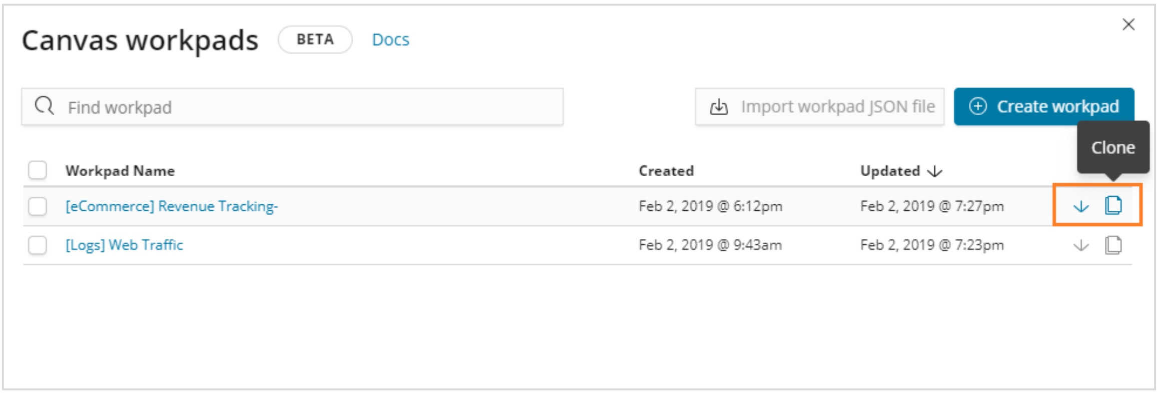
[eCommerce] Revenue Tracking

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:

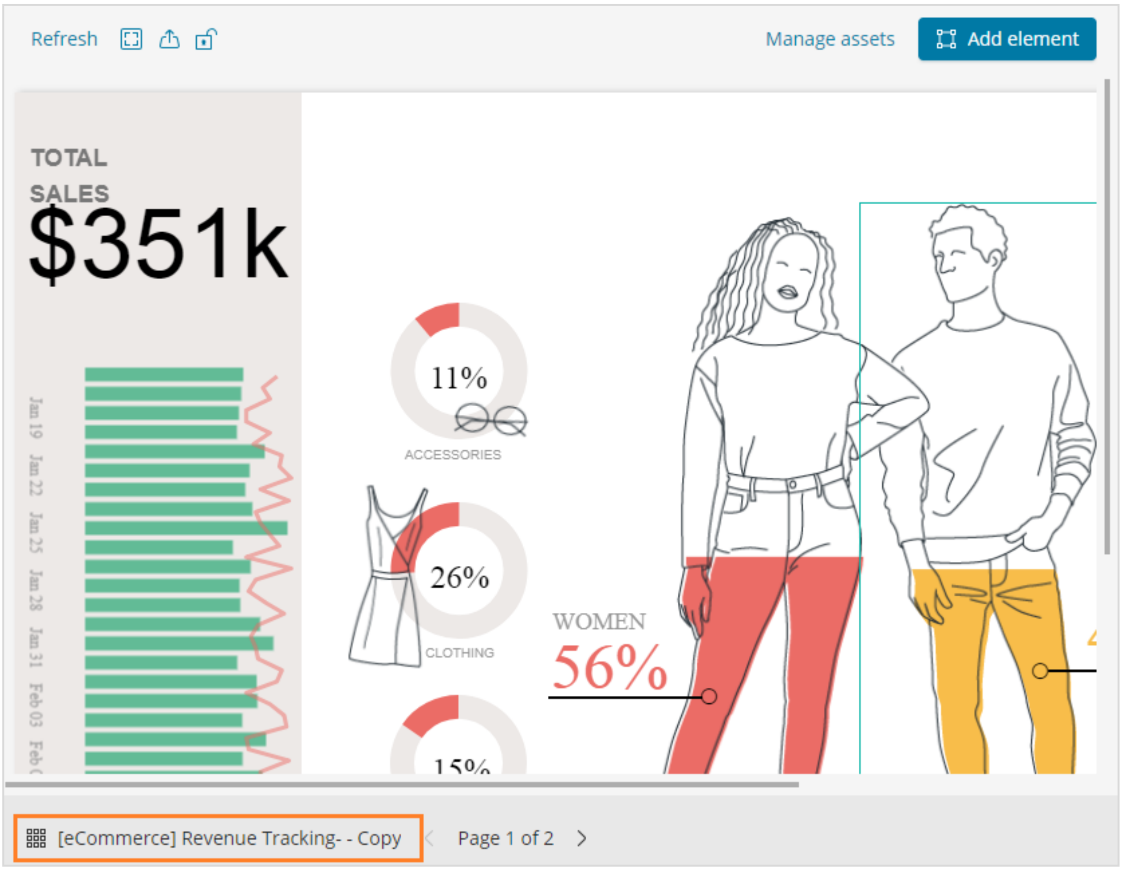


Click on the name and select clone option as shown below:



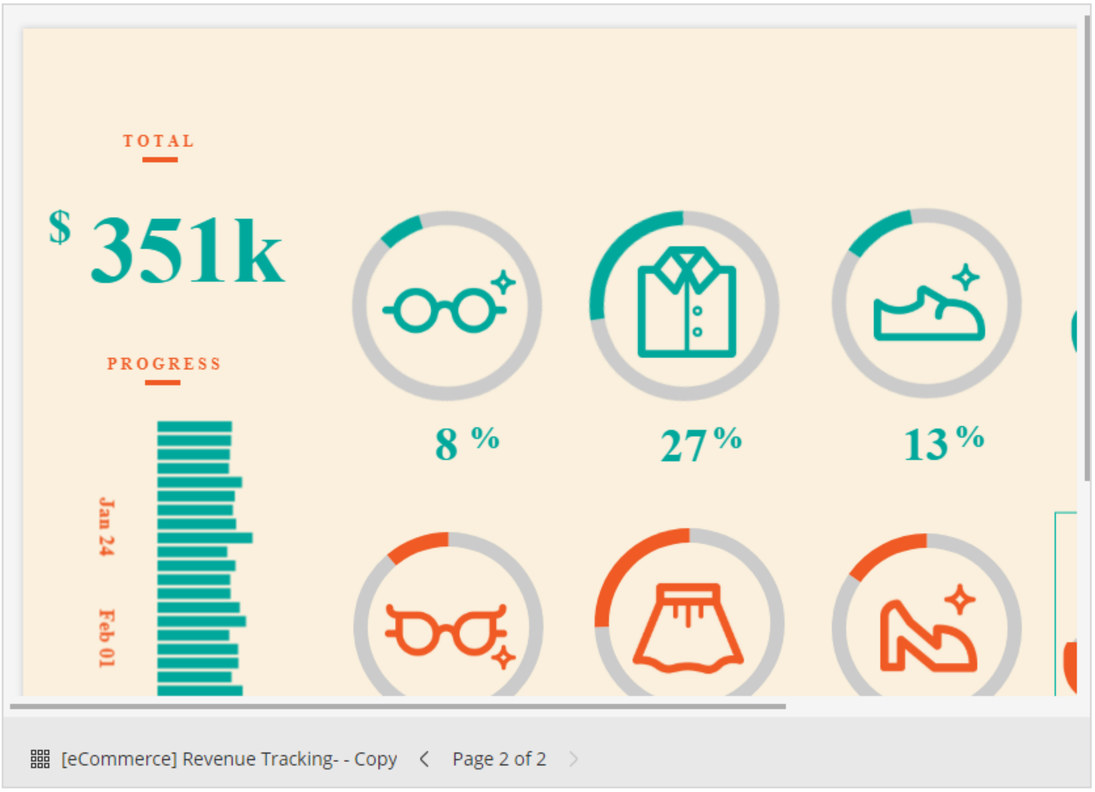
Workpad Name	Created	Updated
[eCommerce] Revenue Tracking-	Feb 2, 2019 @ 6:12pm	Feb 2, 2019 @ 7:27pm
[Logs] Web Traffic	Feb 2, 2019 @ 9:43am	Feb 2, 2019 @ 7:23pm

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

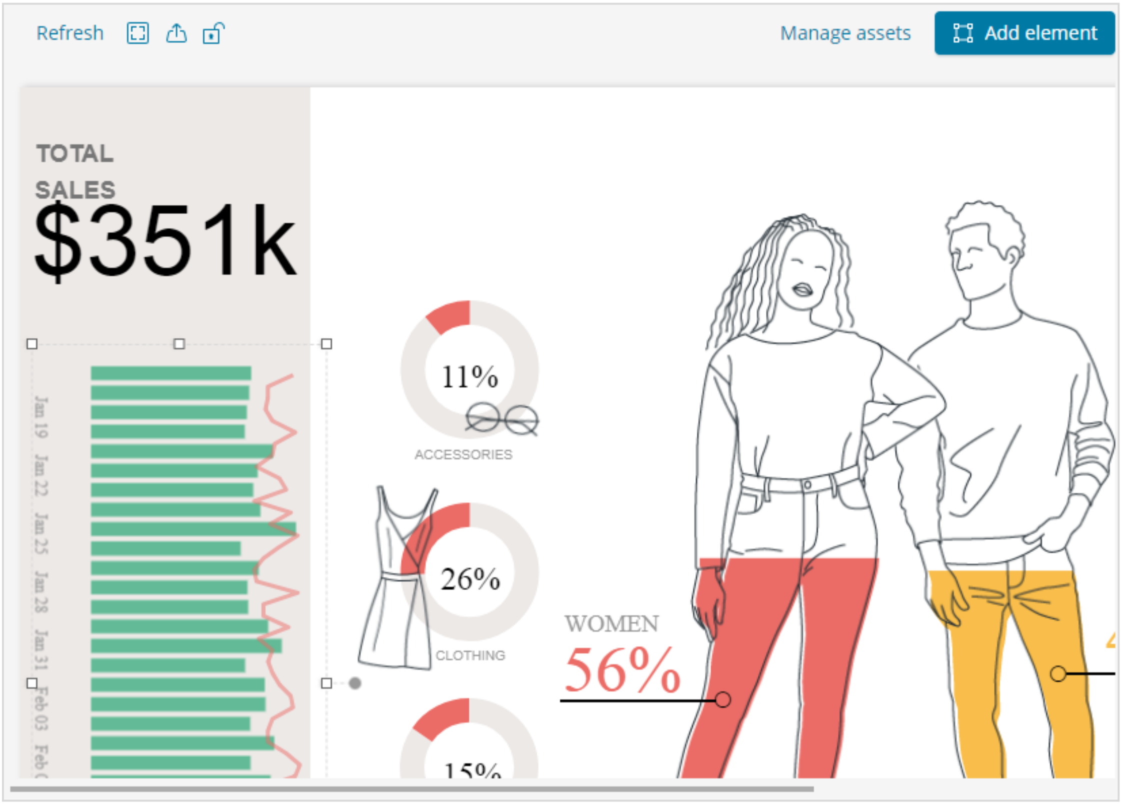


In this section, let us understand how to use the workpad. If you see above workpad, there are 2 pages for it. So in canvas we can represent the data in multiple pages.

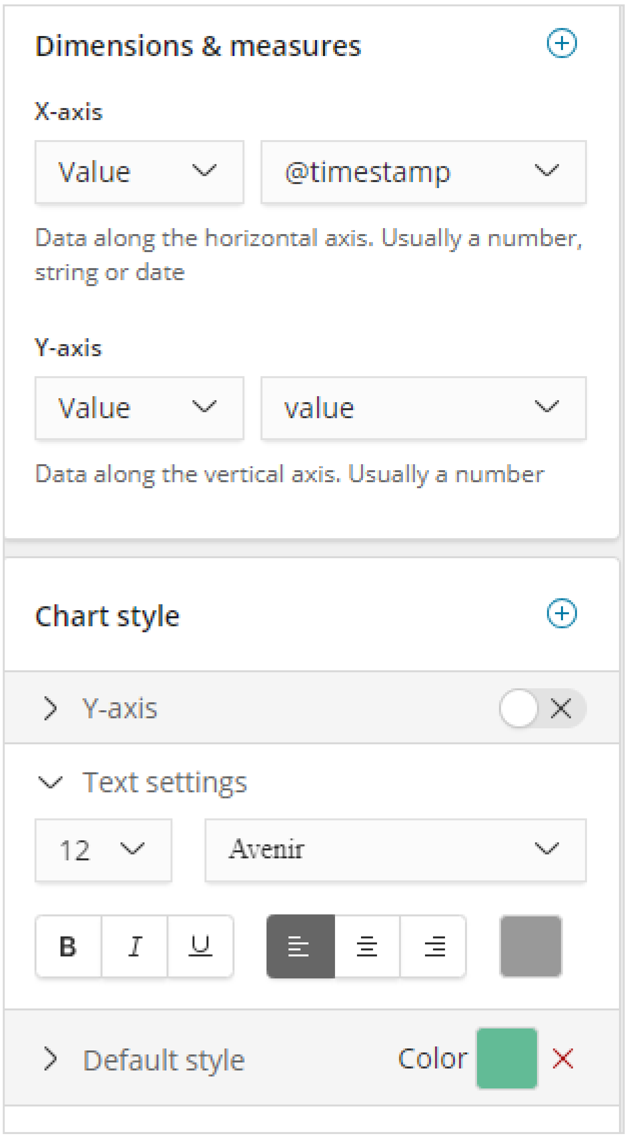
The page 2 display is as shown below:



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:


Dimensions & measures

X-axis

Value | @timestamp

Data along the horizontal axis. Usually a number, string or date

Y-axis

Value | value

Data along the vertical axis. Usually a number

Chart style

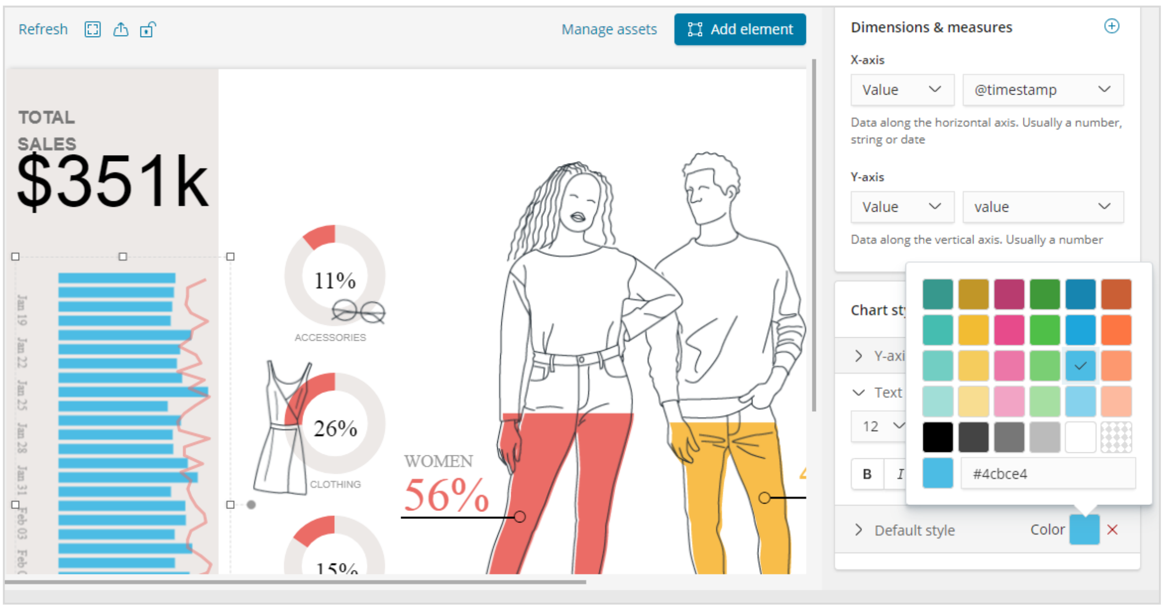
Y-axis

Text settings

12 | Avenir

Default style Color

Right now the default style used is green colour. We can change the colour here and check the display of same.



Dimensions & measures

X-axis

Value | @timestamp

Data along the horizontal axis. Usually a number, string or date

Y-axis

Value | value

Data along the vertical axis. Usually a number

Chart st

Y-axi

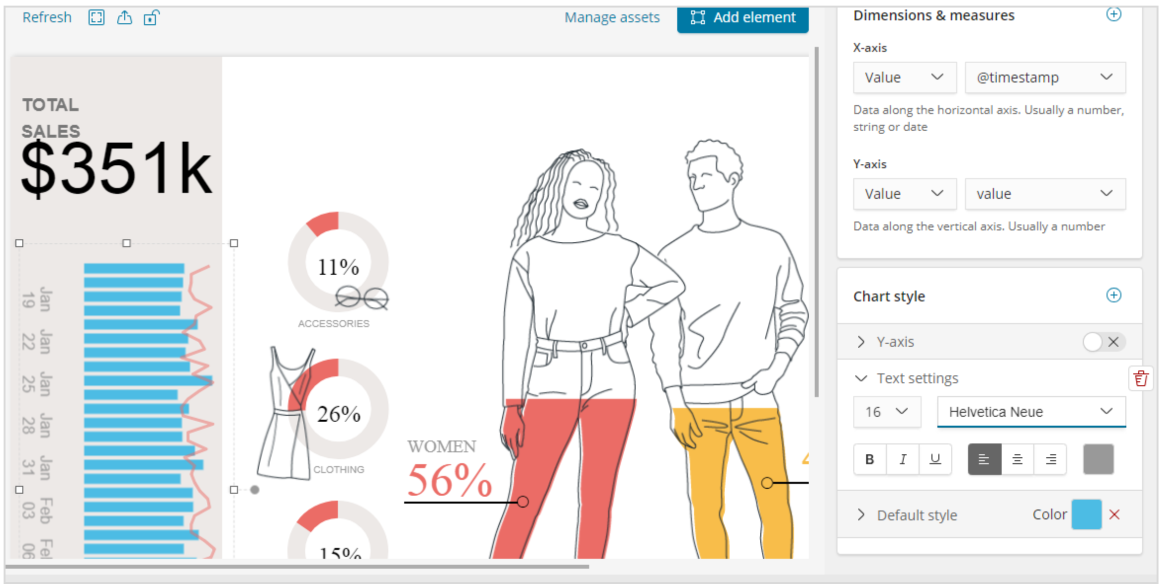
Text

12

#4cbce4

Default style Color

We have also changed the font and size for text settings as shown below:



Dimensions & measures

X-axis

Value | @timestamp

Data along the horizontal axis. Usually a number, string or date

Y-axis

Value | value

Data along the vertical axis. Usually a number

Chart style

Y-axis

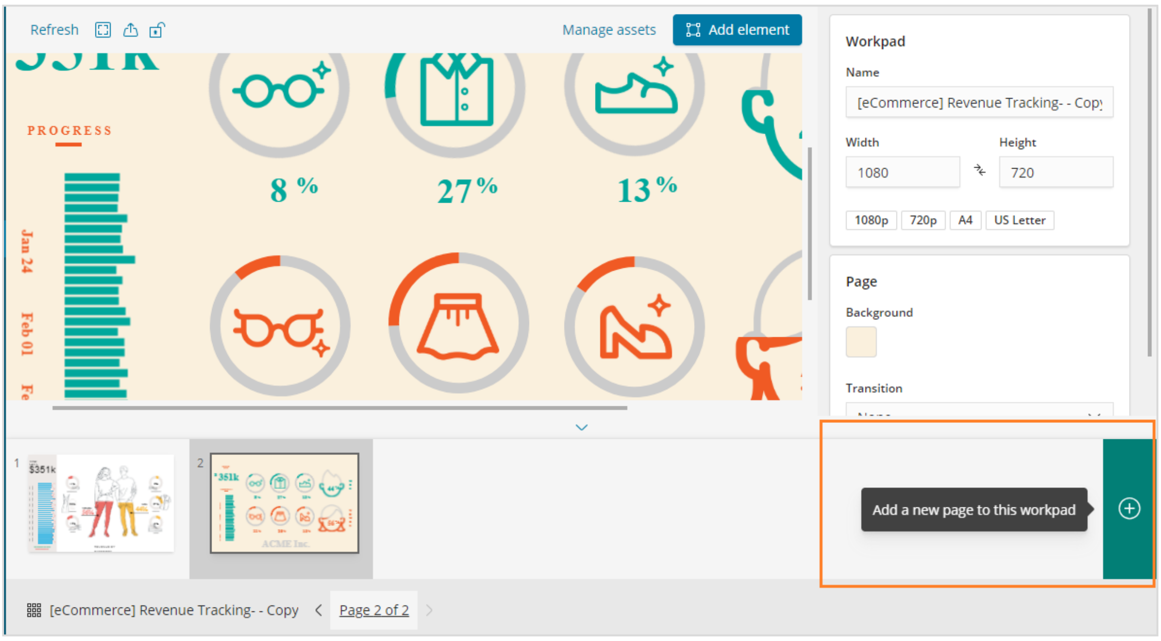
Text settings

16 | Helvetica Neue

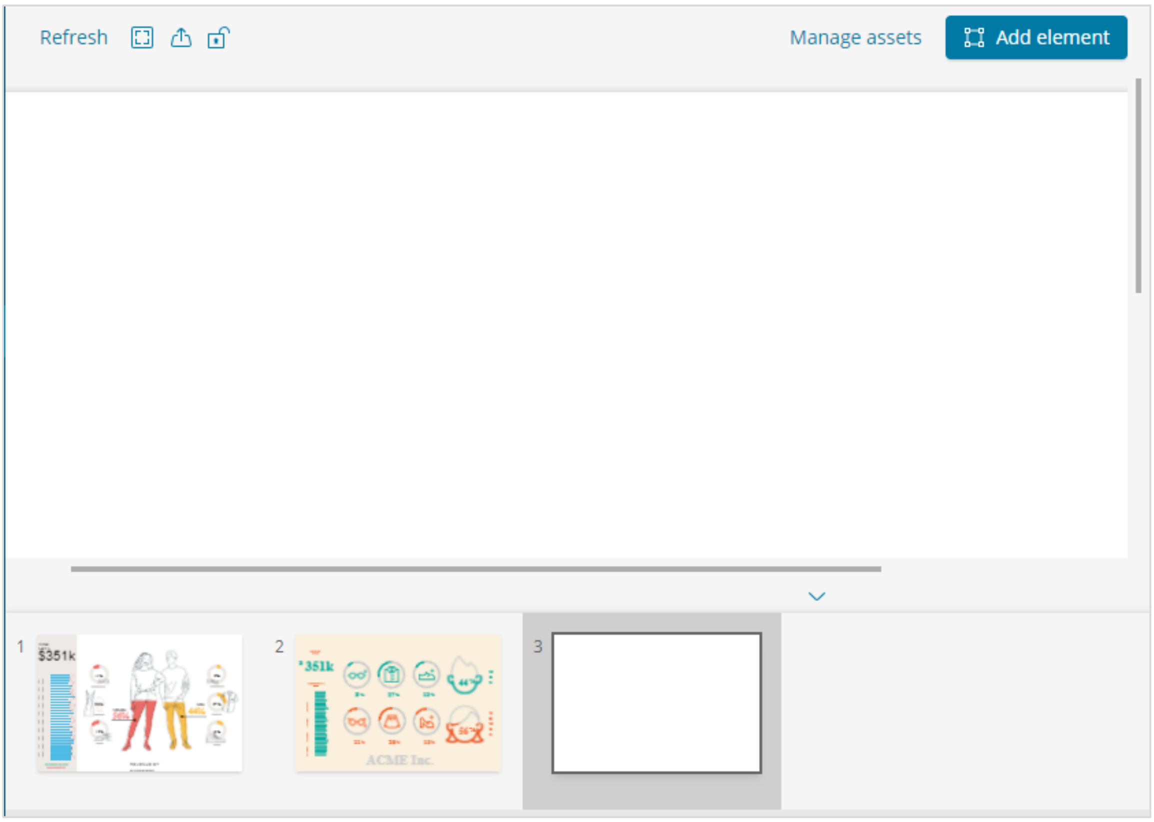
Default style Color

Adding New Page to Workpad Inside Canvas

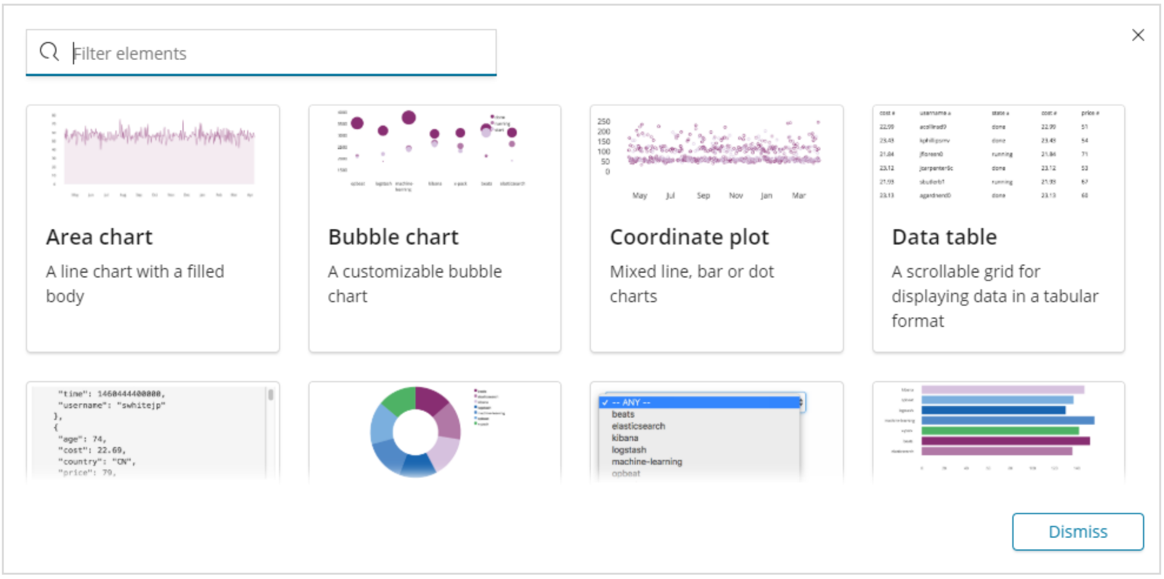
To add new page to the workpad, do as shown below:



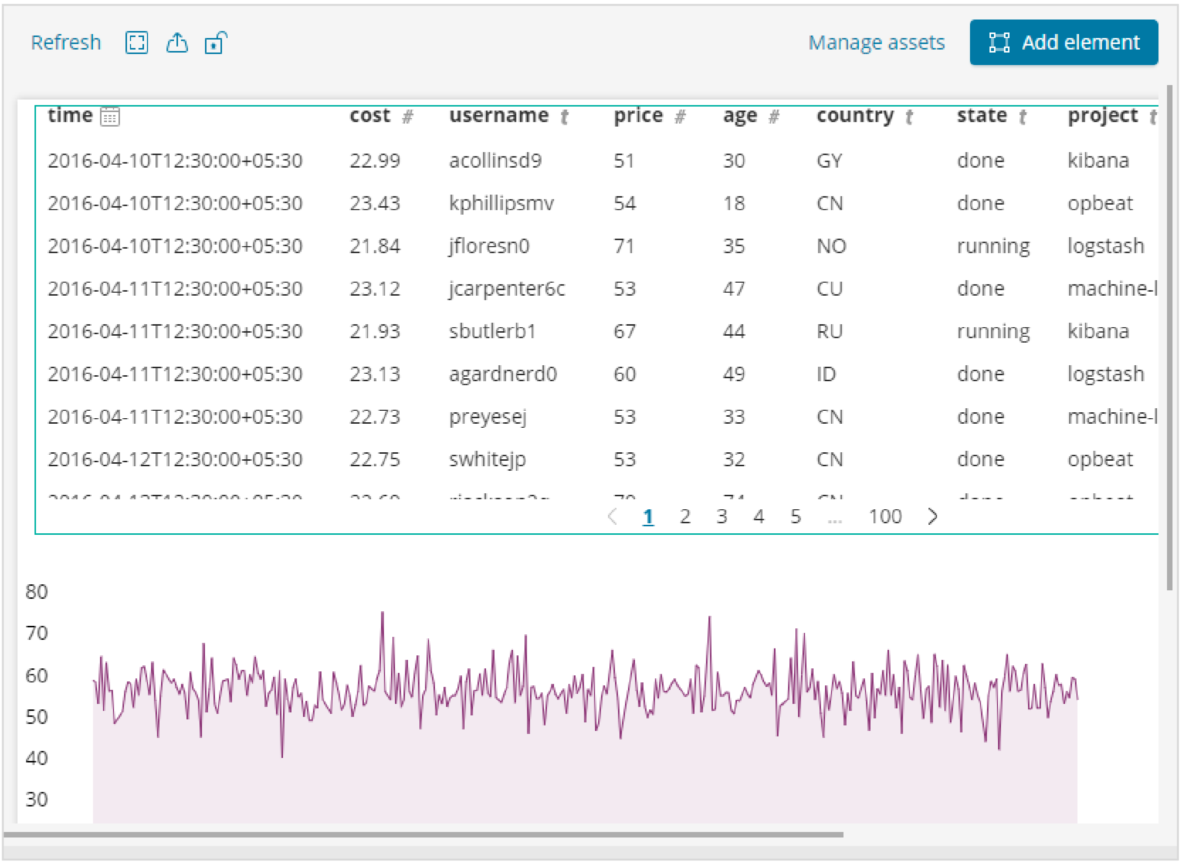
Once the page is created as shown below:



Click on Add element and it will display all possible visualization as shown below:


Filter elements
Area chart
A line chart with a filled body
Bubble chart
A customizable bubble chart
Coordinate plot
Mixed line, bar or dot charts
Data table
A scrollable grid for displaying data in a tabular format
Dismiss

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



time	cost	username	price	age	country	state	project
2016-04-10T12:30:00+05:30	22.99	acollinsd9	51	30	GY	done	kibana
2016-04-10T12:30:00+05:30	23.43	kphillipsmv	54	18	CN	done	opbeat
2016-04-10T12:30:00+05:30	21.84	jfloresn0	71	35	NO	running	logstash
2016-04-11T12:30:00+05:30	23.12	jcarpenter6c	53	47	CU	done	machine-l
2016-04-11T12:30:00+05:30	21.93	sbutlerb1	67	44	RU	running	kibana
2016-04-11T12:30:00+05:30	23.13	agardnerd0	60	49	ID	done	logstash
2016-04-11T12:30:00+05:30	22.73	preyesej	53	33	CN	done	machine-l
2016-04-12T12:30:00+05:30	22.75	swhitejp	53	32	CN	done	opbeat

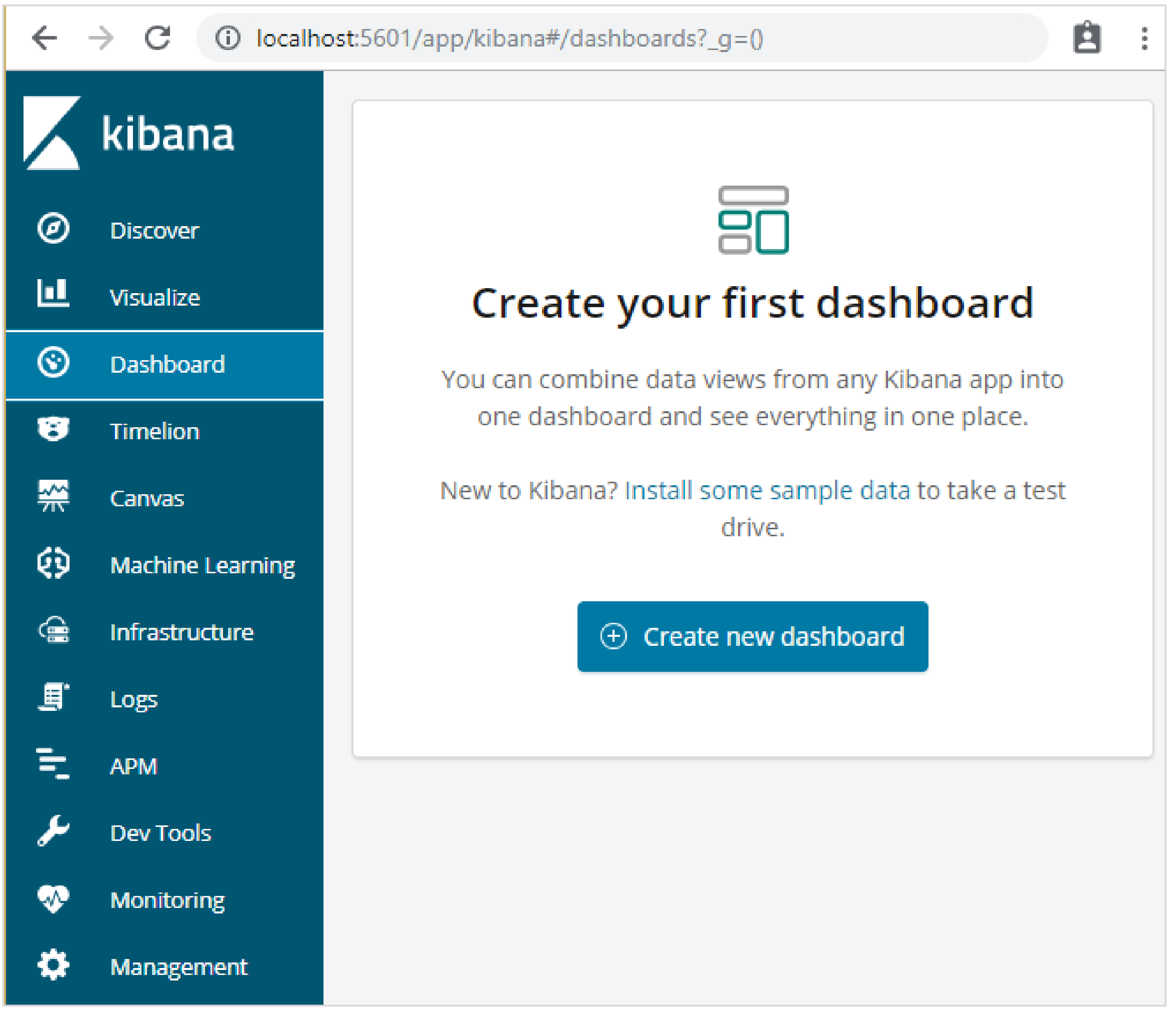
You can add more data elements to the same page or add more pages too.

16. Kibana — Create Dashboard

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:

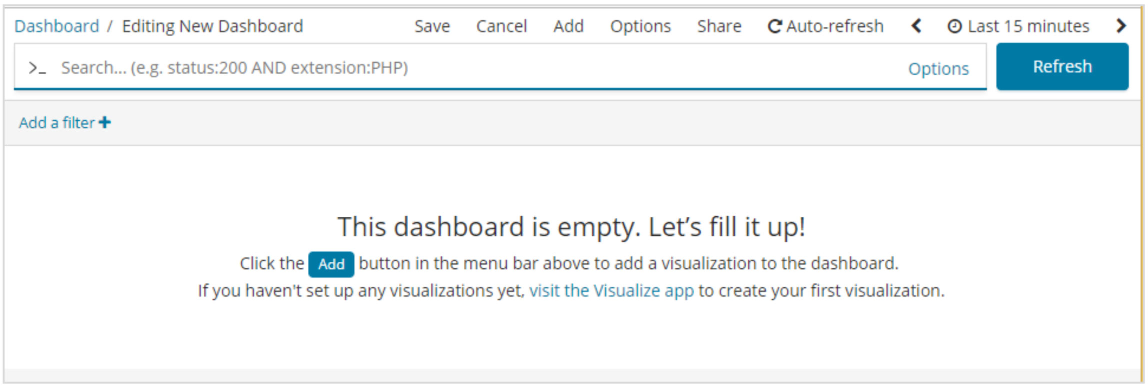


Create your first dashboard

You can combine data views from any Kibana app into one dashboard and see everything in one place.

New to Kibana? Install some sample data to take a test drive.

Now, click on Create new dashboard button as shown above. It will take us to the screen as shown below:


Dashboard / Editing New Dashboard
Save
Cancel
Add
Options
Share
Auto-refresh
Last 15 minutes
Search... (e.g. status:200 AND extension:PHP)
Options
Refresh
Add a filter

This dashboard is empty. Let's fill it up!

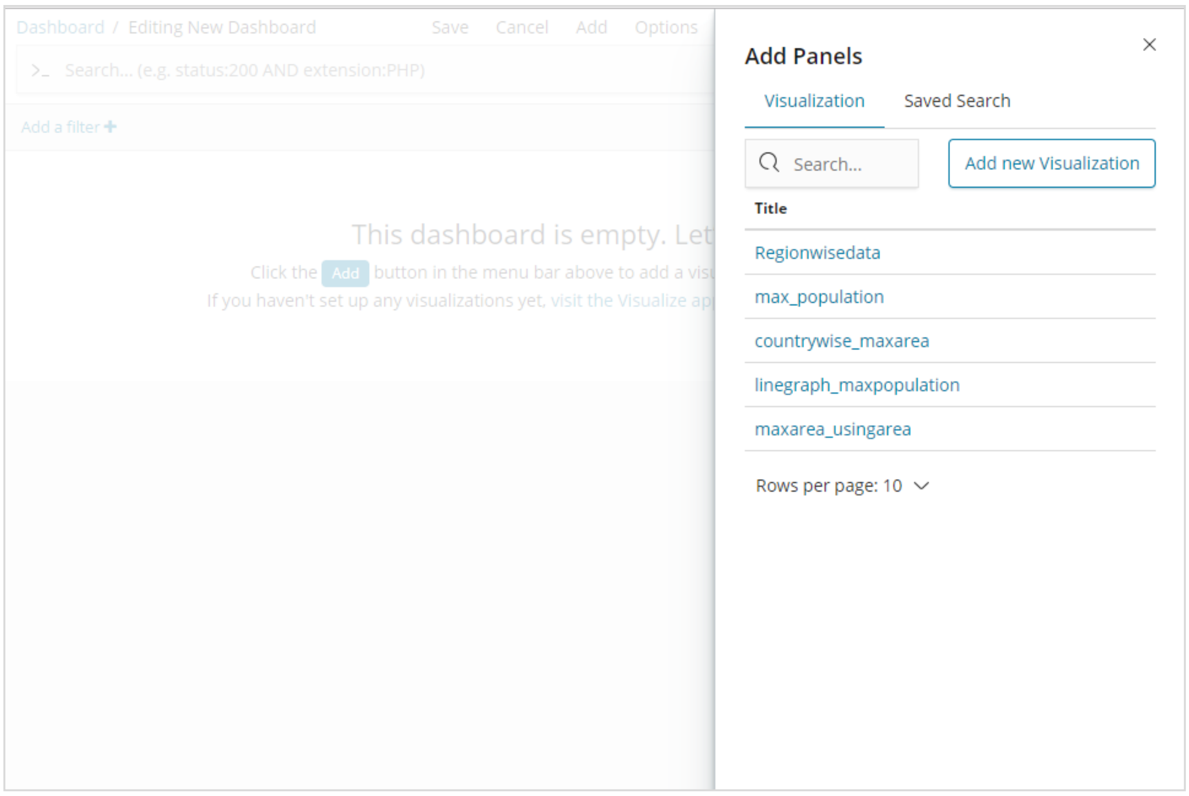
Click the Add button in the menu bar above to add a visualization to the dashboard.

If you haven't set up any visualizations yet, visit the Visualize app to create your first visualization.

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.

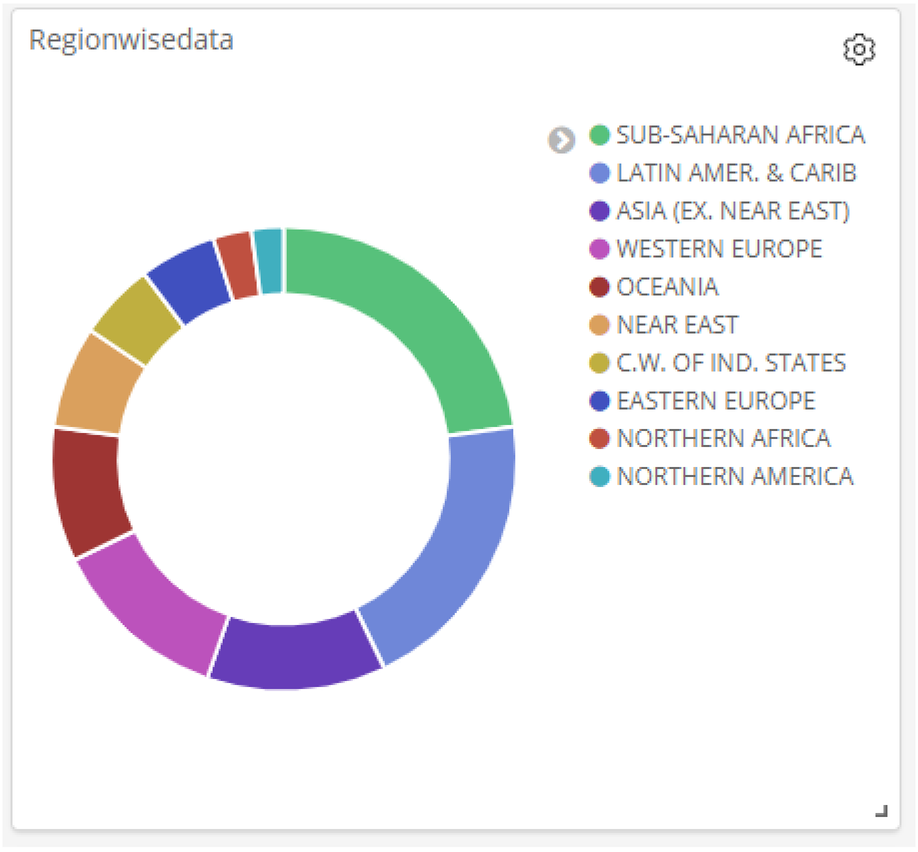
Add Visualization to Dashboard

When we click the Add button (top left corner), it displays us the visualization we created as shown below:

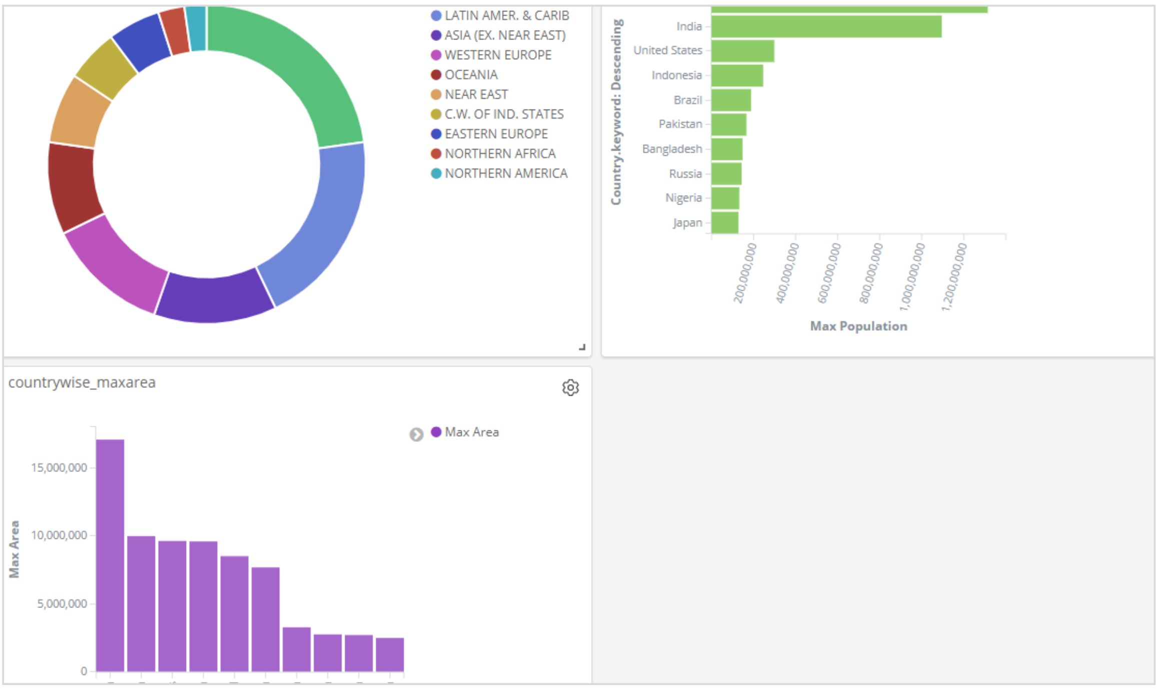


Title
Regionwisedata
max_population
countrywise_maxarea
linegraph_maxpopulation
maxarea_usingarea

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.

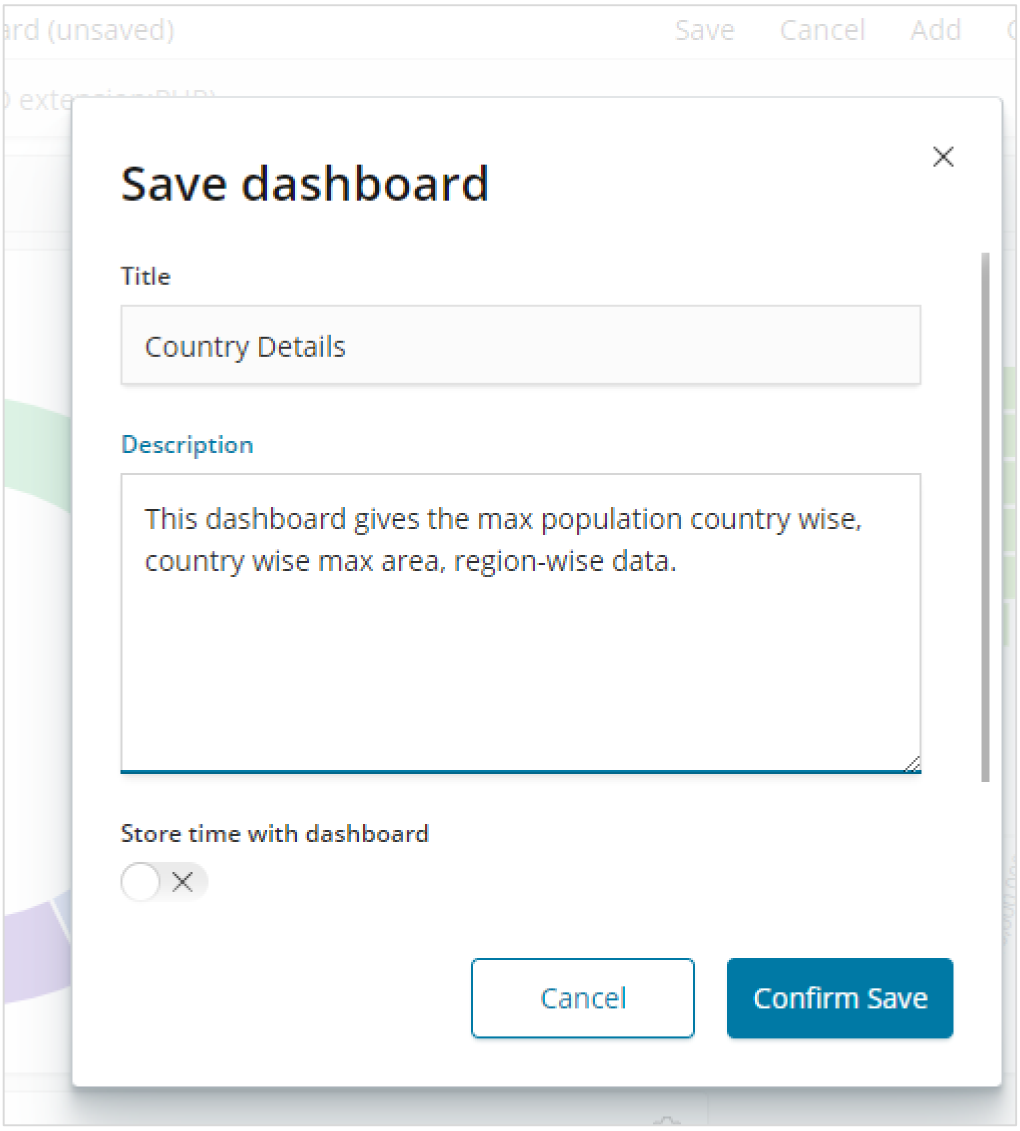
So now we know all the regions available, the max population country wise in descending order, the max area etc.

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.


Save dashboard

Title: Country Details

Description: This dashboard gives the max population country wise, country wise max area, region-wise data.

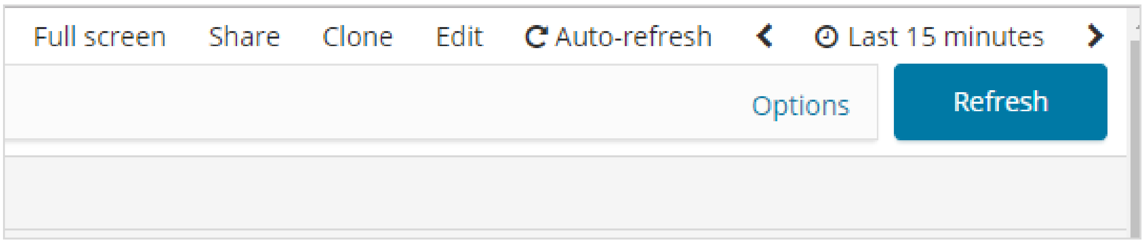
Store time with dashboard: off

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

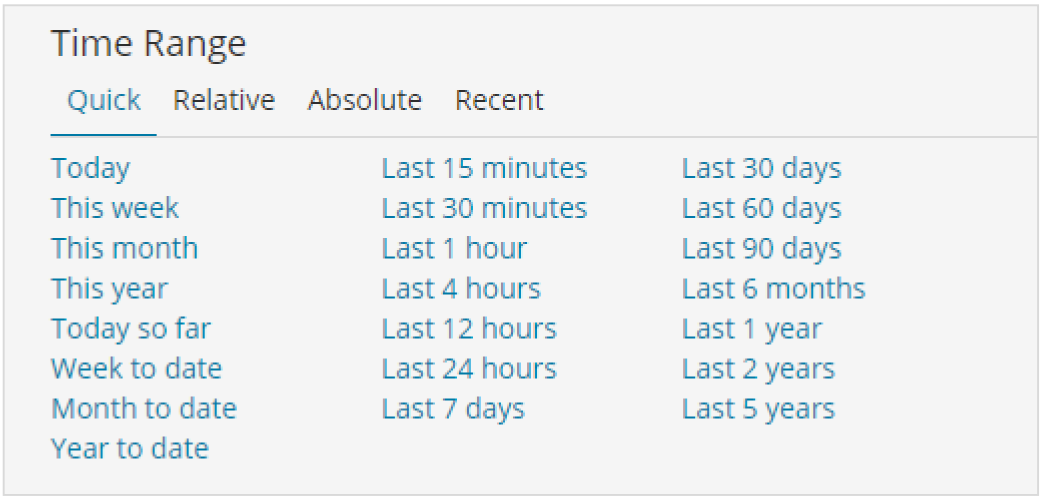
At present you can see the data shown is of **Last 15 minutes**. Please note this is a static data without any time field so the data displayed will not change. When you have the data connected to real time system changing the time, will also show the data reflecting.

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.

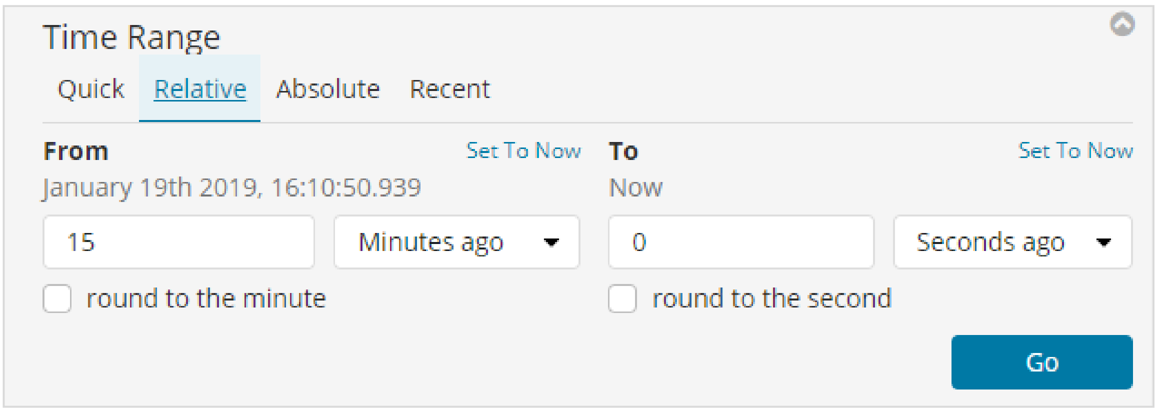
Observe that there are Quick, Relative, Absolute and Recent options. The following screenshot shows the details for Quick option:


Time Range

Quick Relative Absolute Recent

Today	Last 15 minutes	Last 30 days
This week	Last 30 minutes	Last 60 days
This month	Last 1 hour	Last 90 days
This year	Last 4 hours	Last 6 months
Today so far	Last 12 hours	Last 1 year
Week to date	Last 24 hours	Last 2 years
Month to date	Last 7 days	Last 5 years
Year to date		

Now, click on Relative to see the option available:


Time Range

Quick Relative Absolute Recent

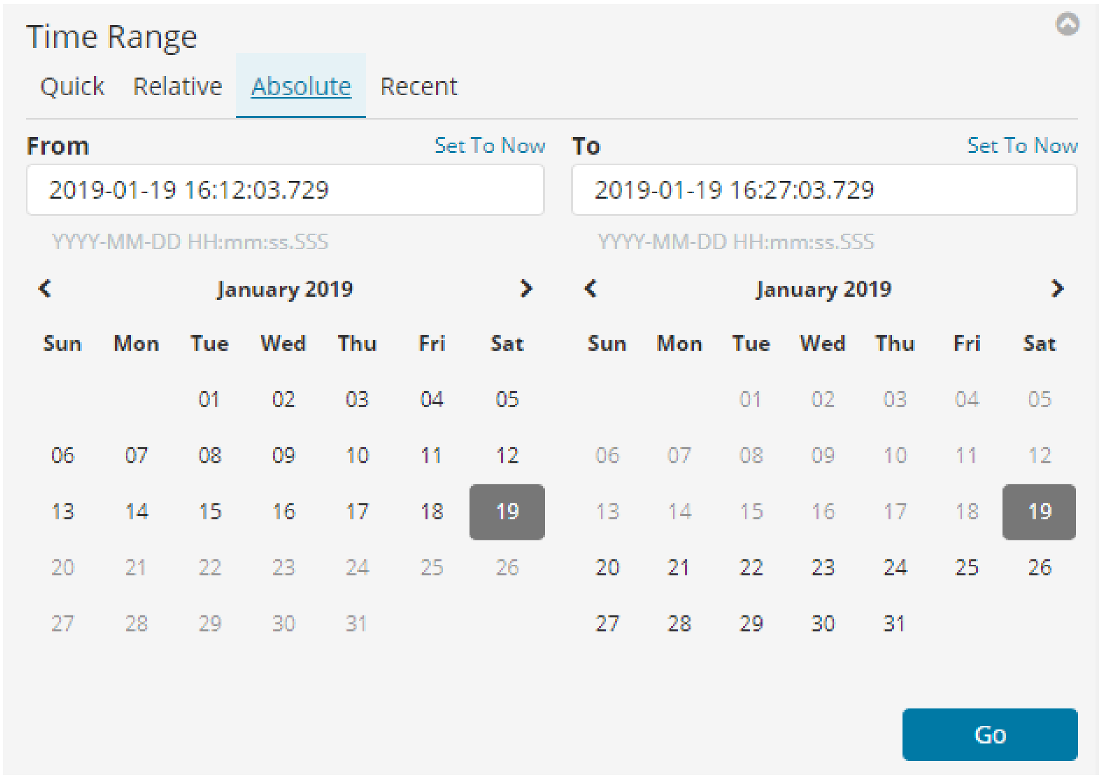
From Set To Now
January 19th 2019, 16:10:50.939
15 Minutes ago
round to the minute

To Set To Now
Now
0 Seconds ago
round to the second

Go

Here you can specify the From and To date in minutes , hours, seconds, months, years ago.

The Absolute option has the following details:

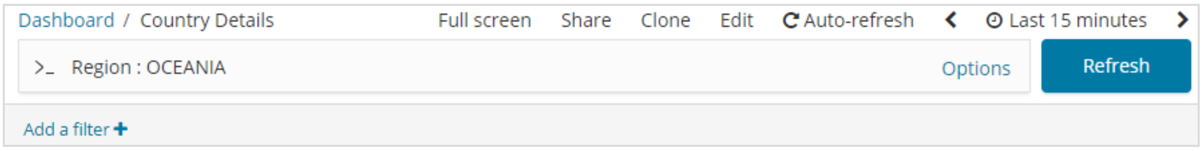


You can see the calendar option and can select a date range.

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

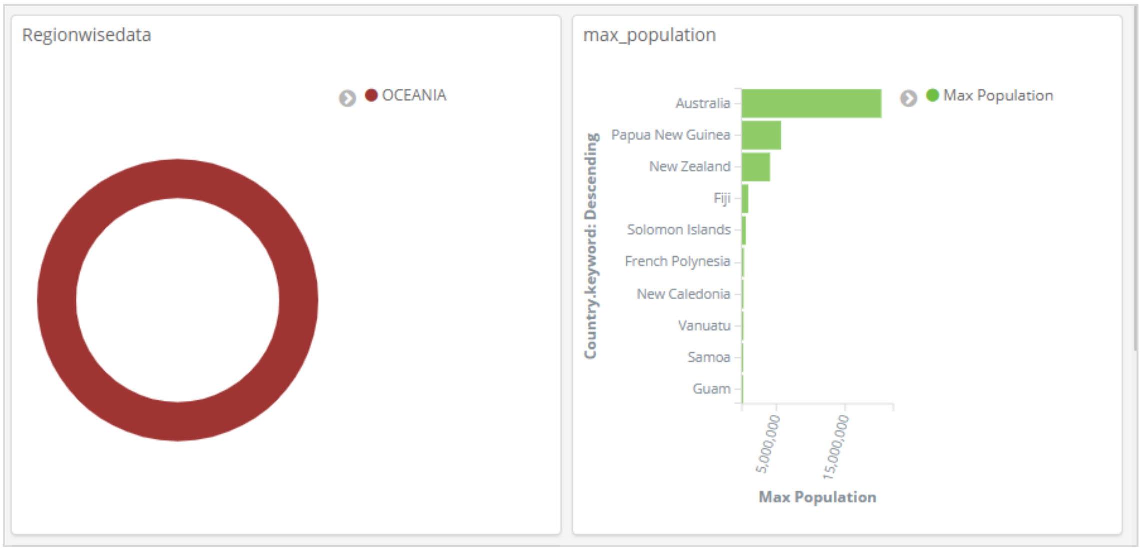
We can also use search and filter on the dashboard. In search suppose if we want to get the details of a particular region, we can add a search as shown below:



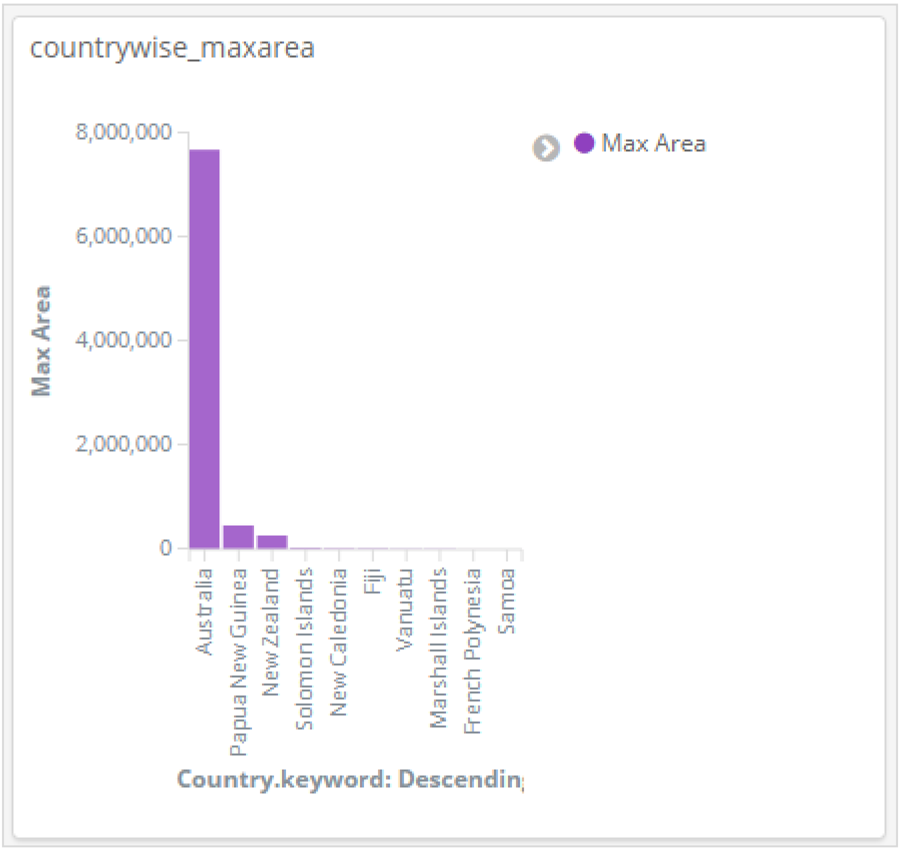
Region : OCEANIA

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

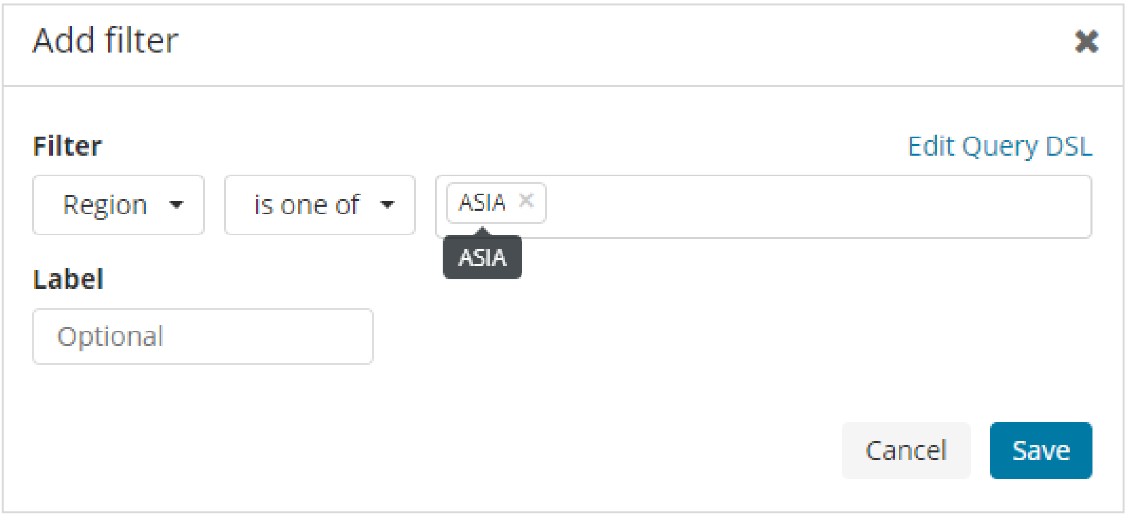
We get following results:

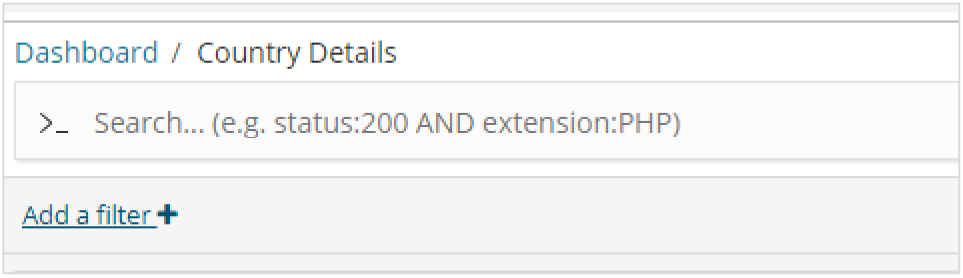


Looking at the above data we can say that in OCEANIA region, Australia has the max population and Area.

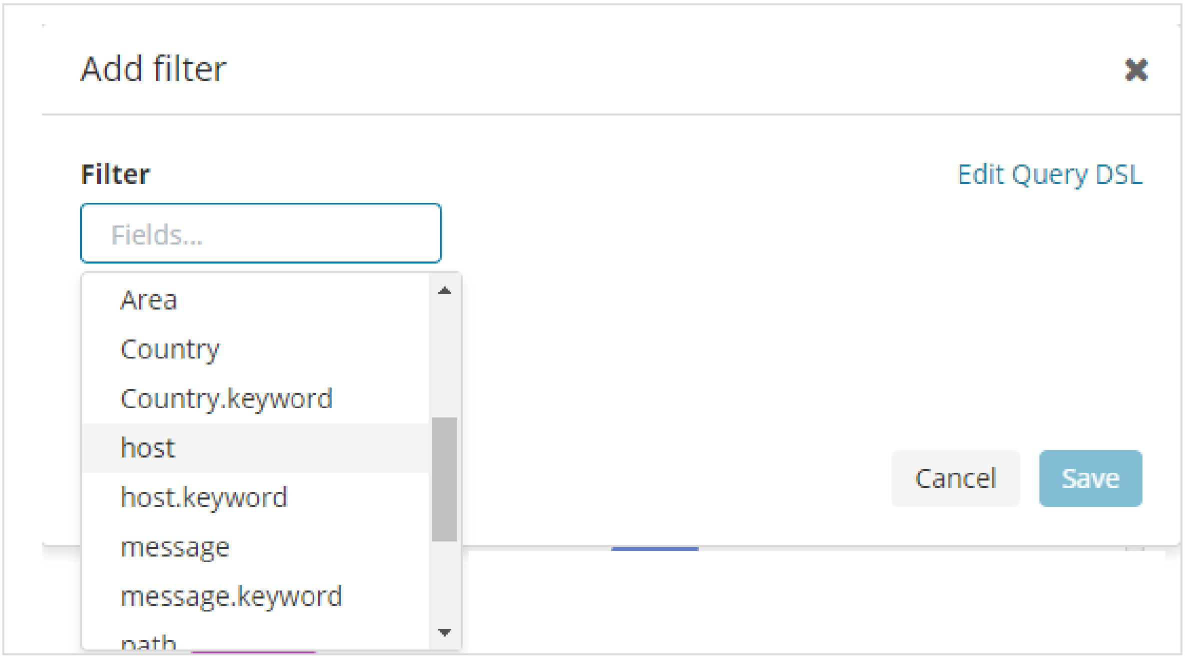


Similarly, we can add a filter as shown below:



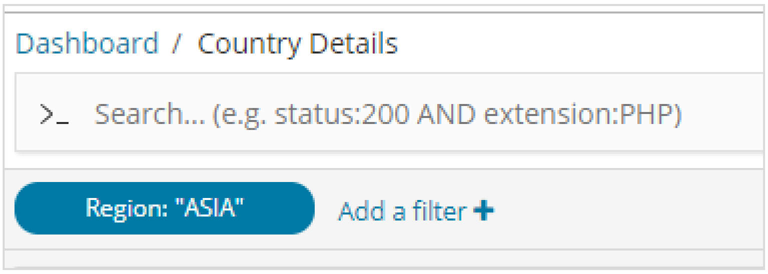

Dashboard / Country Details
>_ Search... (e.g. status:200 AND extension:PHP)
Add a filter +

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

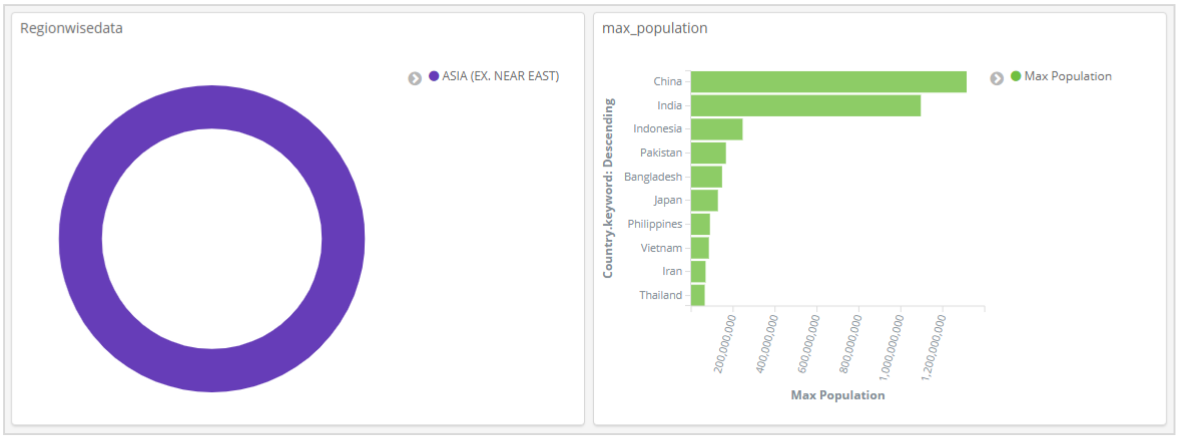


Choose the field you want to filter on. I will use Region field to get the details of ASIA region as shown below:

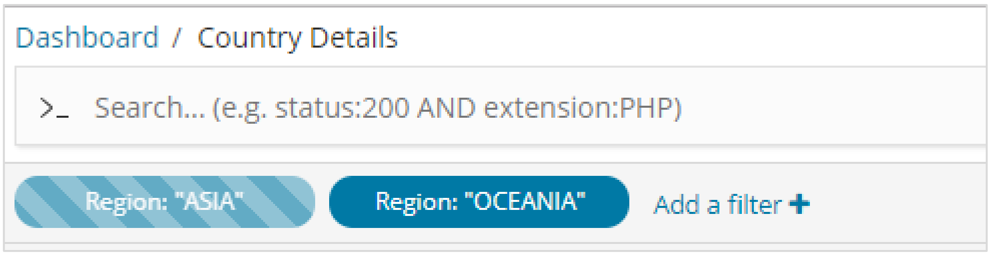
Save the filter and you should see the filter as follows:


Dashboard / Country Details
>_ Search... (e.g. status:200 AND extension:PHP)
Region: "ASIA"
Add a filter +

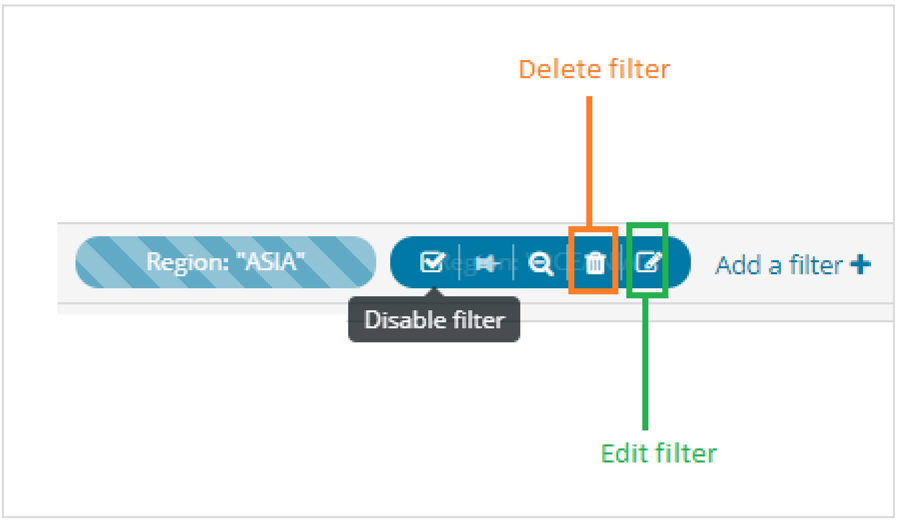
The data will now be shown as per the filter added:



You can also add more filters as shown below:

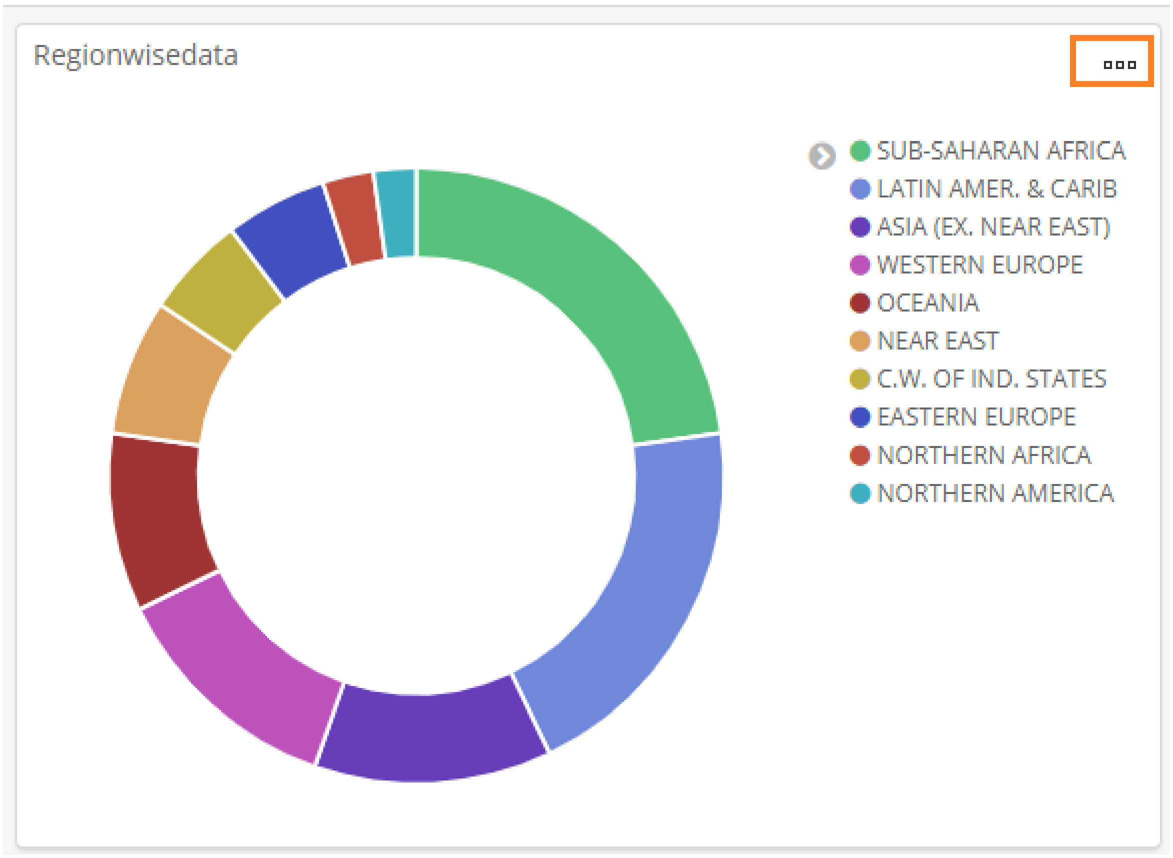

Dashboard / Country Details
>_ Search... (e.g. status:200 AND extension:PHP)
Region: "ASIA"
Region: "OCEANIA"
Add a filter +

You can disable the filter by clicking on the disable checkbox as shown below.

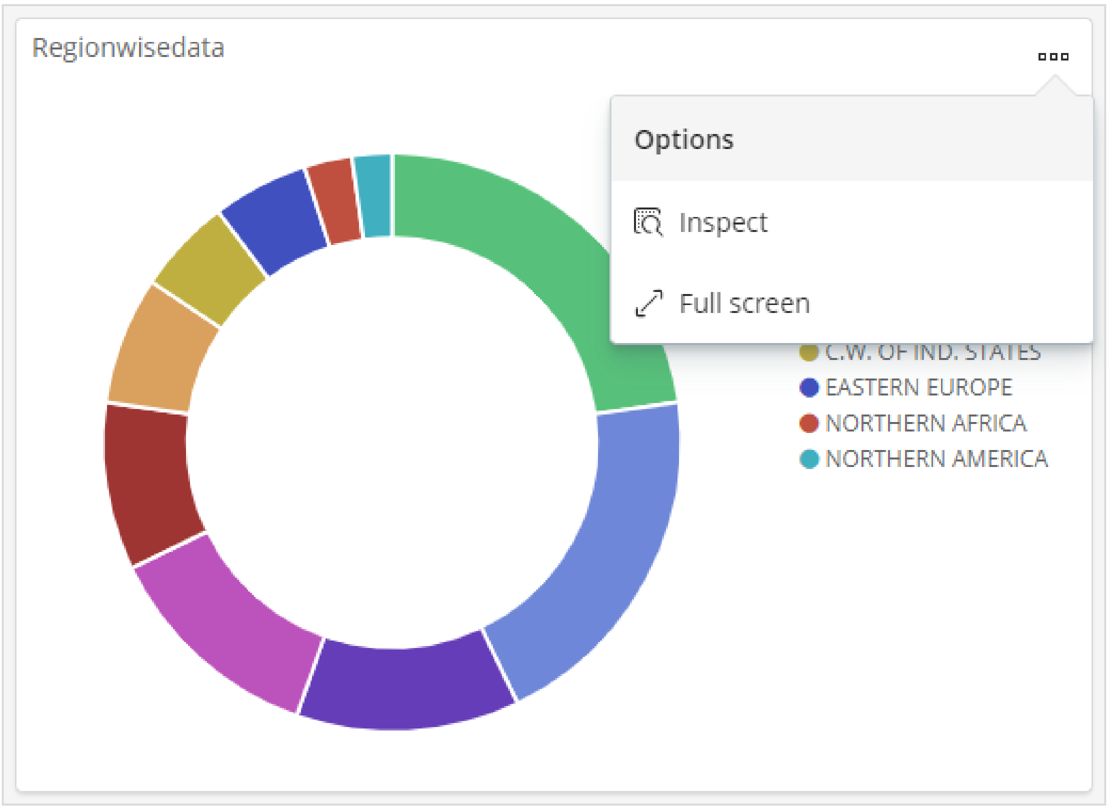


You can activate the filter by clicking on the same checkbox to activate it. Observe that there is delete button to delete the filter. Edit button to edit the filter or change the filter options.

For the visualization displayed, you will notice three dots as shown below:

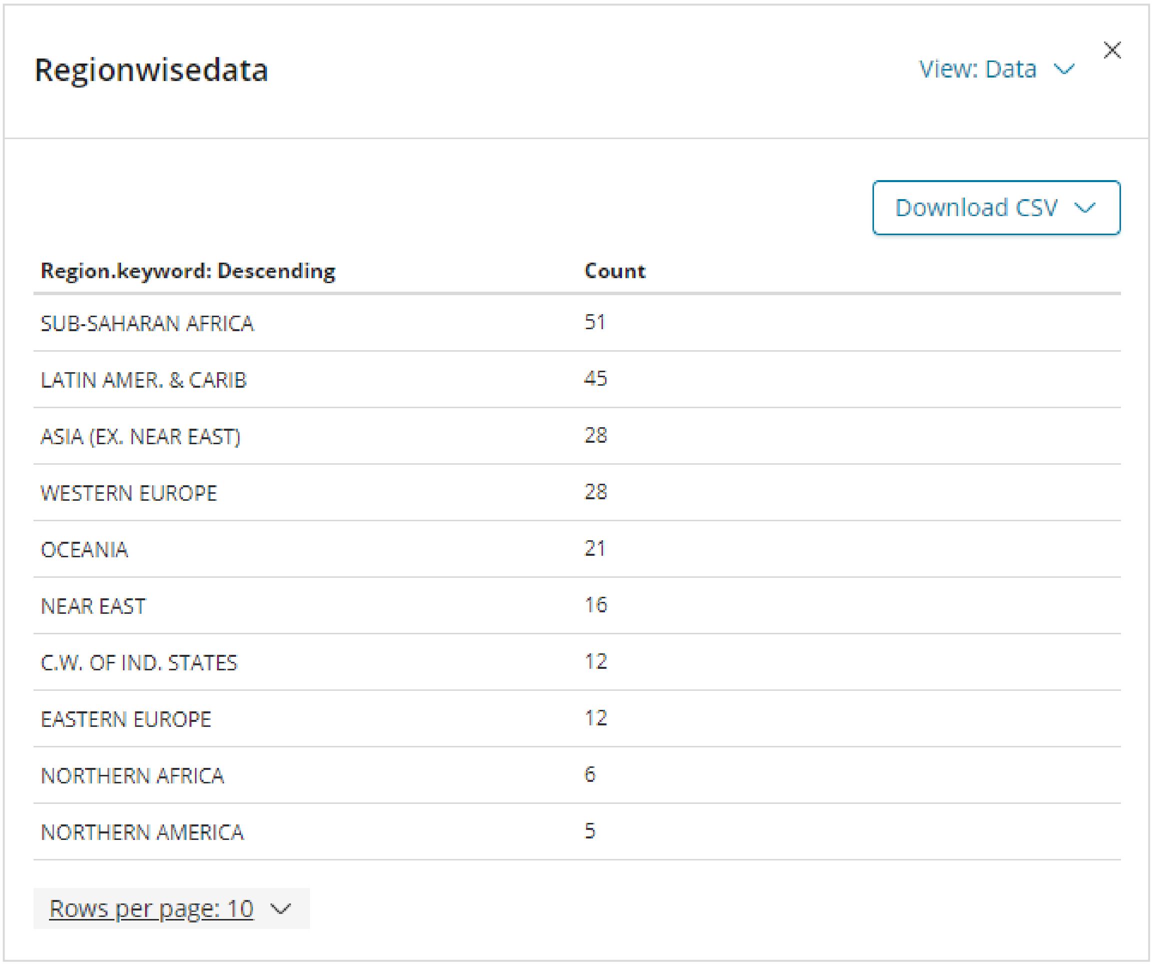


Click on it and it will display options as shown below:



Inspect and Fullscreen

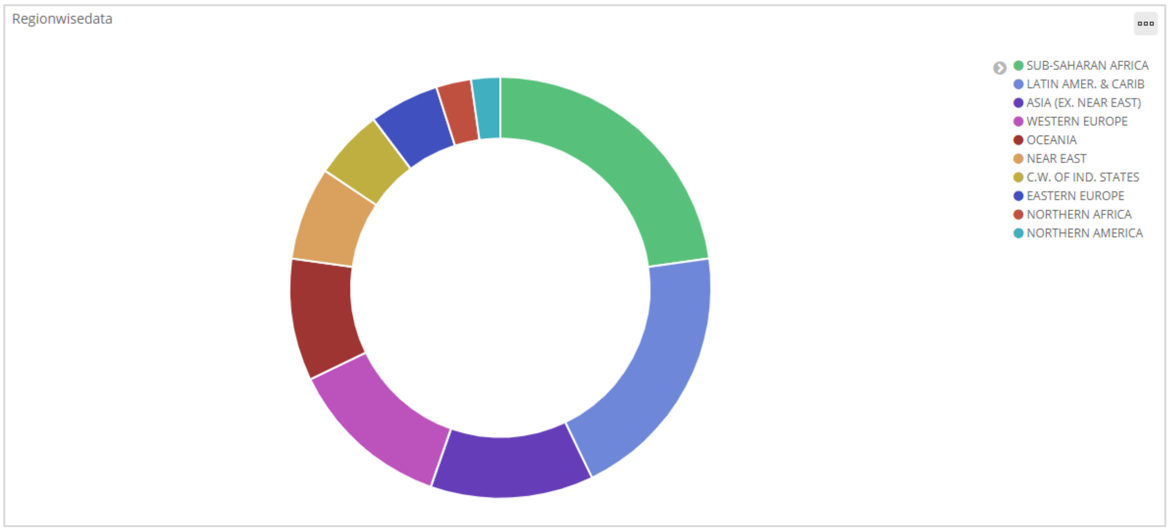
Click on Inspect and it gives the details of the region in tabular format as shown below:



Region.keyword: Descending	Count
SUB-SAHARAN AFRICA	51
LATIN AMER. & CARIB	45
ASIA (EX. NEAR EAST)	28
WESTERN EUROPE	28
OCEANIA	21
NEAR EAST	16
C.W. OF IND. STATES	12
EASTERN EUROPE	12
NORTHERN AFRICA	6
NORTHERN AMERICA	5

There is an option to download the visualization in CSV format in-case you want to see it in excel sheet.

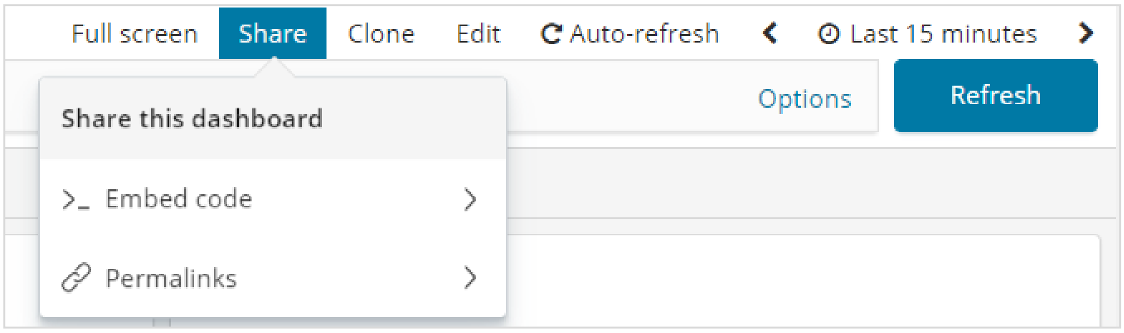
The next option fullscreen will get the visualization in a fullscreenmode as shown below:



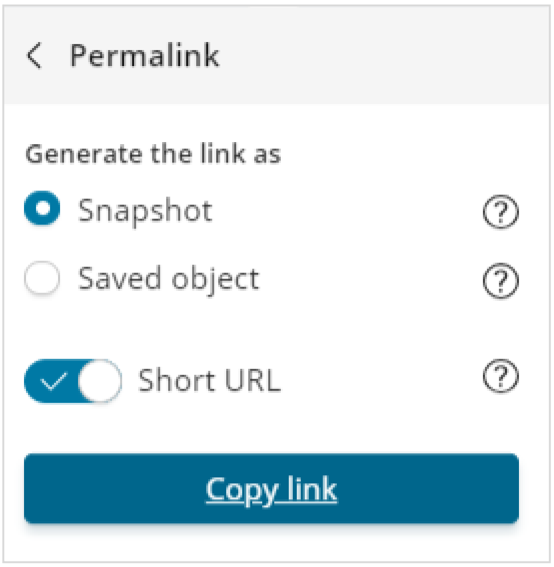
You can use the same button to exit the fullscreen mode.

Sharing Dashboard

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



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.



Generate the link as

- Snapshot (selected)
- Saved object

Short URL: on

The url will be as follows:

```
http://localhost:5601/goto/519c1a088d5d0f8703937d754923b84b
```

17. KIBANA — Timelion

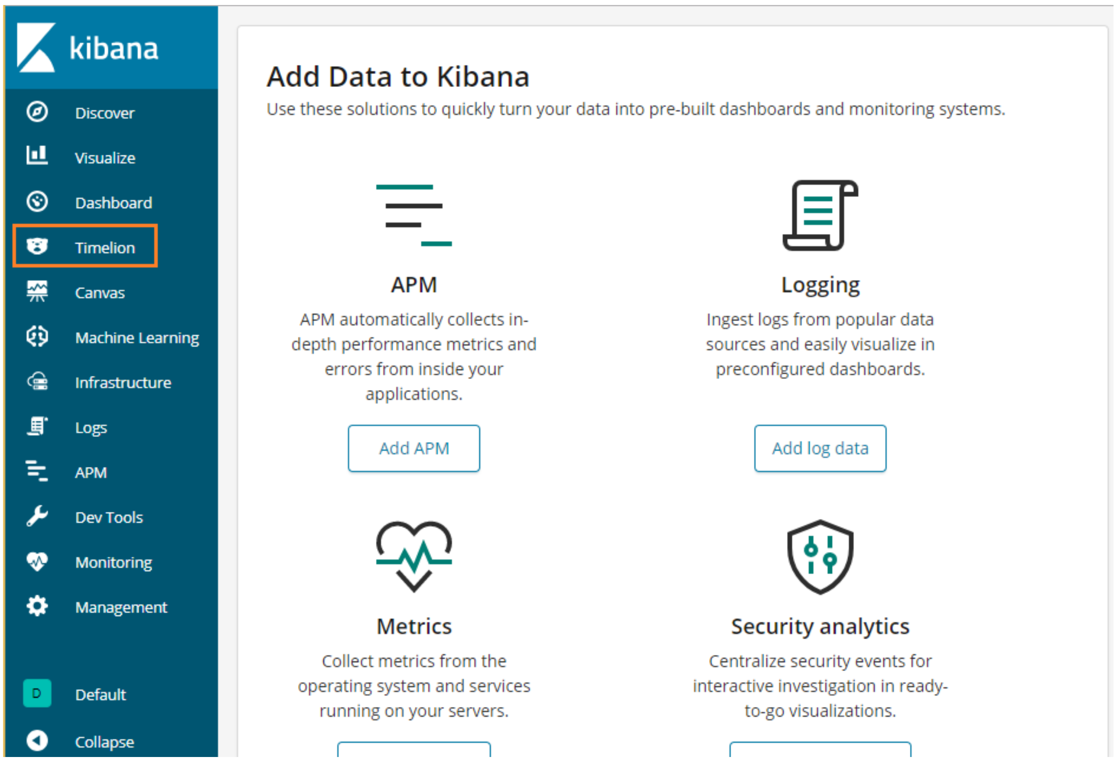
Timelion, also called as *timeline* is yet another visualization tool which is mainly used for time based data analysis. To work with timeline, we need to use simple expression language which will help us connect to the index and also perform calculations on the data to get the results we need.

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:



Add Data to Kibana

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

APM

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

Logging

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

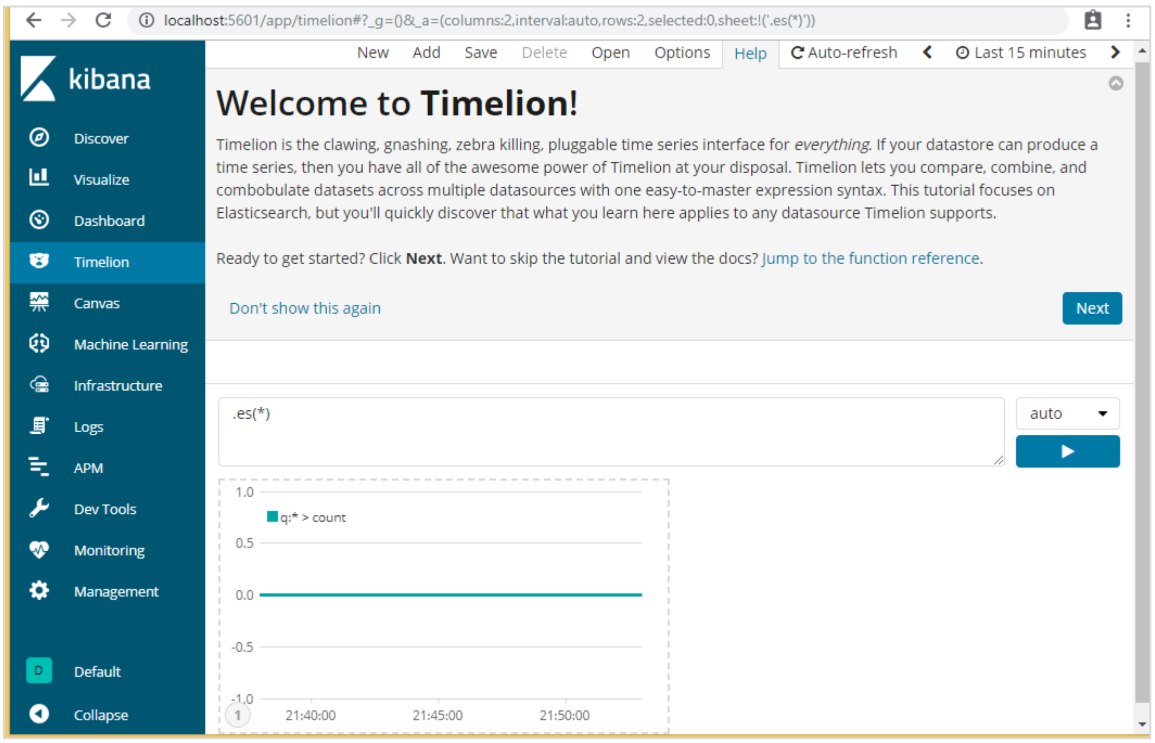
Metrics

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

Security analytics

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

Timelion by default shows the timeline of all indexes as shown below:



Welcome to Timelion!

Timelion is the clawing, gnashing, zebra killing, pluggable time series interface for *everything*. If your datastore can produce a time series, then you have all of the awesome power of Timelion at your disposal. Timelion lets you compare, combine, and combobulate datasets across multiple datasources with one easy-to-master expression syntax. This tutorial focuses on Elasticsearch, but you'll quickly discover that what you learn here applies to any datasource Timelion supports.

Ready to get started? Click **Next**. Want to skip the tutorial and view the docs? Jump to the function reference.

Don't show this again

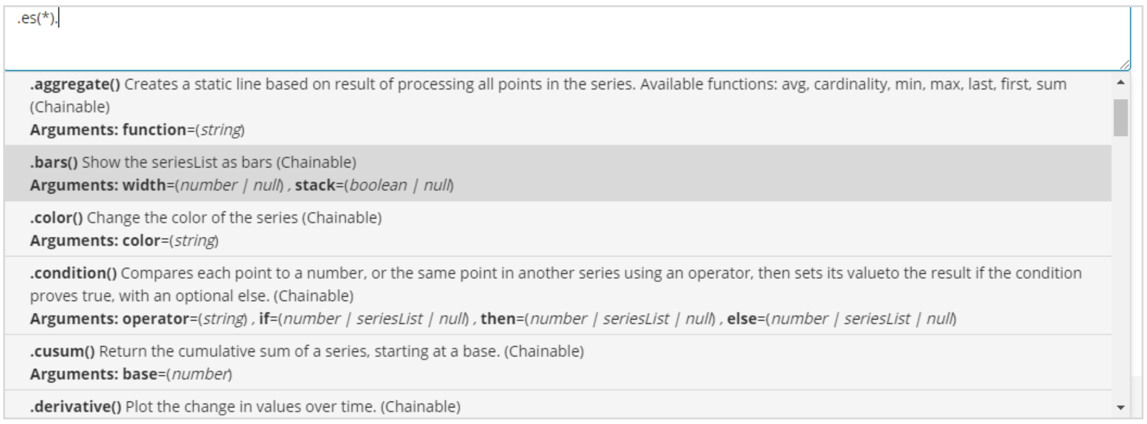
Next

```
.es(*)
```

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:



```
.es(*).
```

.aggregate() Creates a static line based on result of processing all points in the series. Available functions: avg, cardinality, min, max, last, first, sum (Chainable)
Arguments: function=(*string*)

.bars() Show the seriesList as bars (Chainable)
Arguments: width=(*number | null*) , **stack**=(*boolean | null*)

.color() Change the color of the series (Chainable)
Arguments: color=(*string*)

.condition() Compares each point to a number, or the same point in another series using an operator, then sets its valueto the result if the condition proves true, with an optional else. (Chainable)
Arguments: operator=(*string*) , **if**=(*number | seriesList | null*) , **then**=(*number | seriesList | null*) , **else**=(*number | seriesList | null*)

.cusum() Return the cumulative sum of a series, starting at a base. (Chainable)
Arguments: base=(*number*)

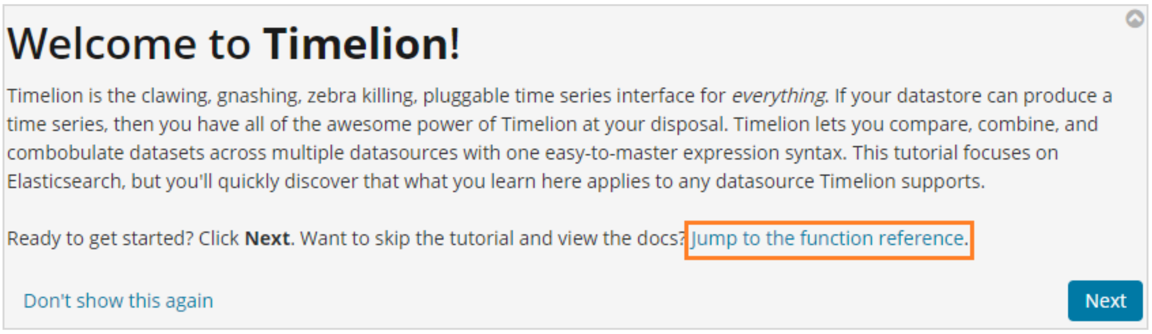
.derivative() Plot the change in values over time. (Chainable)

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:

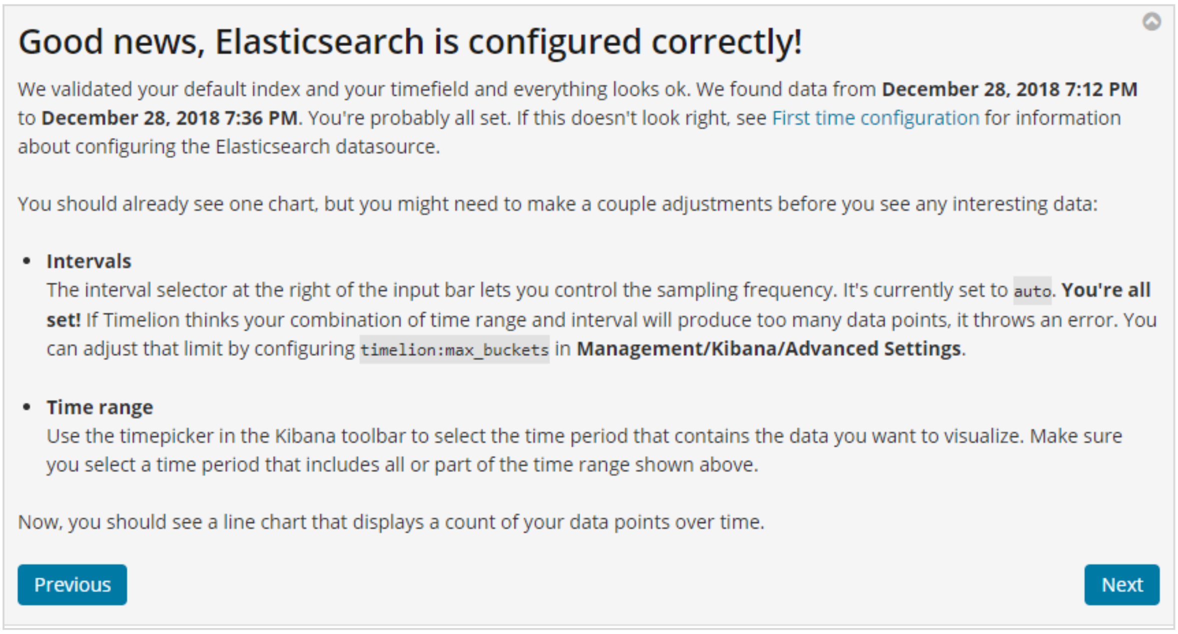


Welcome to Timelion!

Timelion is the clawing, gnashing, zebra killing, pluggable time series interface for *everything*. If your datastore can produce a time series, then you have all of the awesome power of Timelion at your disposal. Timelion lets you compare, combine, and combobulate datasets across multiple datasources with one easy-to-master expression syntax. This tutorial focuses on Elasticsearch, but you'll quickly discover that what you learn here applies to any datasource Timelion supports.

Ready to get started? Click **Next**. Want to skip the tutorial and view the docs? Jump to the function reference.

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!

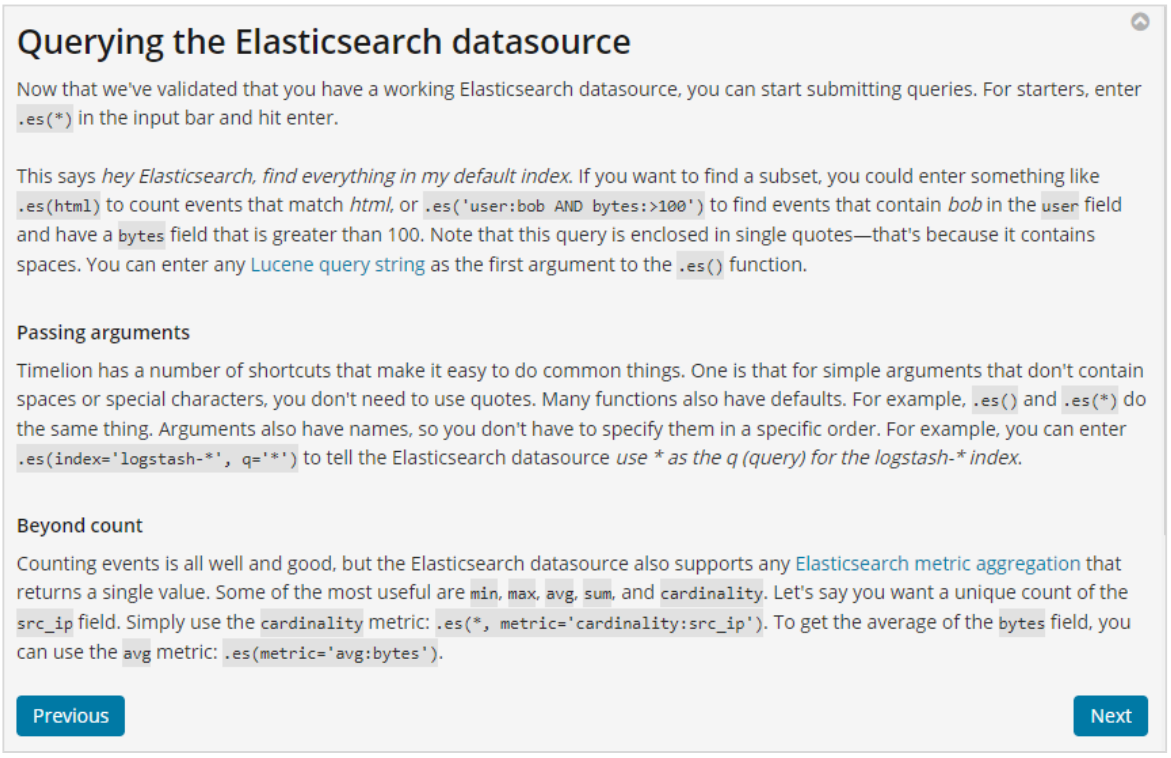
We validated your default index and your timefield and everything looks ok. We found data from **December 28, 2018 7:12 PM** to **December 28, 2018 7:36 PM**. You're probably all set. If this doesn't look right, see First time configuration for information about configuring the Elasticsearch datasource.

You should already see one chart, but you might need to make a couple adjustments before you see any interesting data:

- **Intervals**
  The interval selector at the right of the input bar lets you control the sampling frequency. It's currently set to `auto`. **You're all set!** If Timelion thinks your combination of time range and interval will produce too many data points, it throws an error. You can adjust that limit by configuring `timelion:max_buckets` in **Management/Kibana/Advanced Settings**.

- **Time range**
  Use the timepicker in the Kibana toolbar to select the time period that contains the data you want to visualize. Make sure you select a time period that includes all or part of the time range shown above.

Now, you should see a line chart that displays a count of your data points over time.



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.

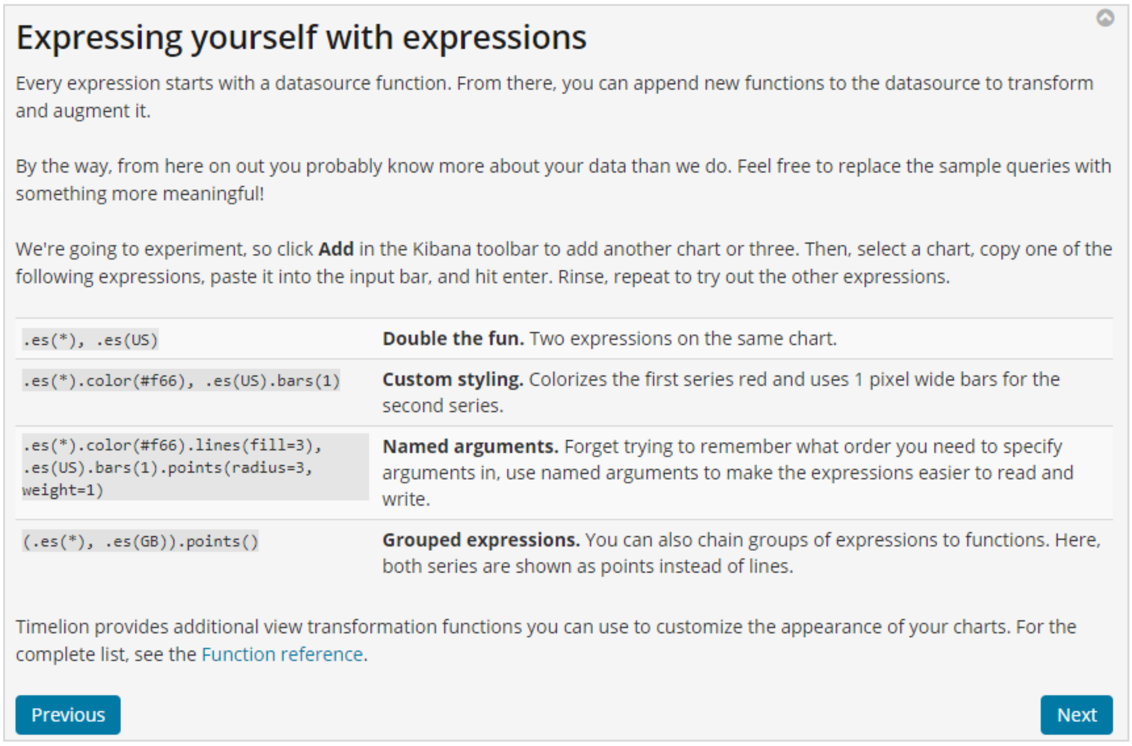
This says *hey Elasticsearch, find everything in my default index.* If you want to find a subset, you could enter something like `.es(html)` to count events that match *html*, or `.es('user:bob AND bytes:>100')` to find events that contain *bob* in the `user` field and have a `bytes` field that is greater 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 `.es(*)` 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

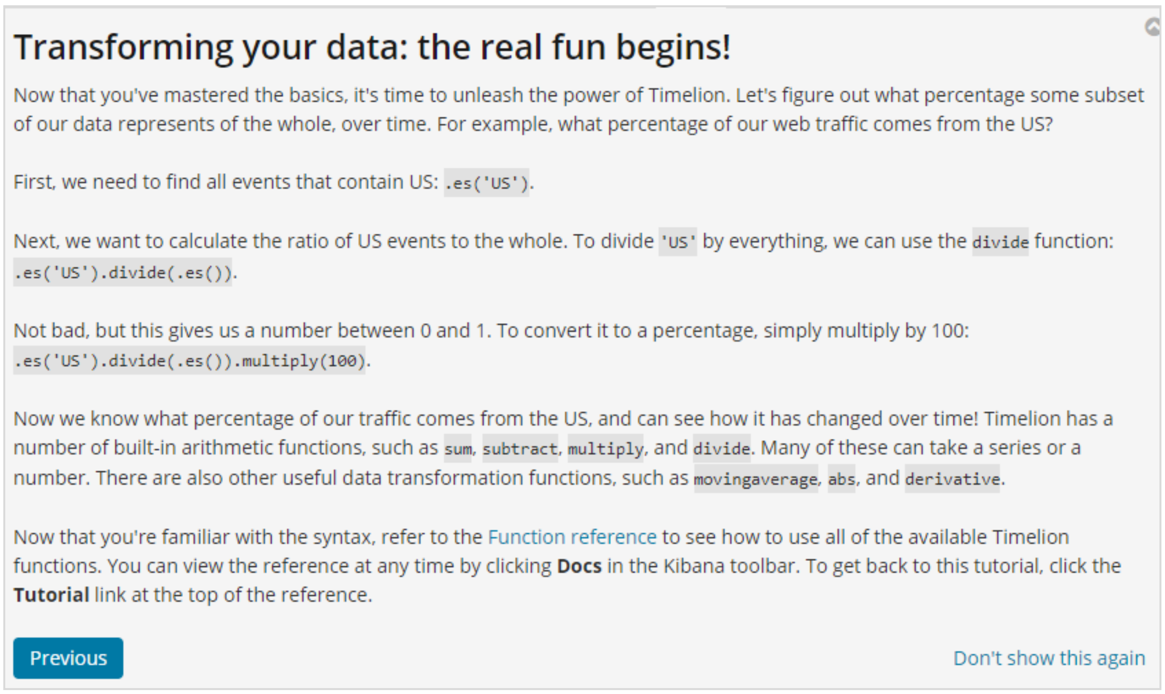
Every expression starts with a datasource function. From there, you can append new functions to the datasource to transform and augment it.

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.

`.es(*), .es(US)`	**Double the fun.** Two expressions on the same chart.
`.es(*).color(#f66), .es(US).bars(1)`	**Custom styling.** Colorizes the first series red and uses 1 pixel wide bars for the second series.
`.es(*).color(#f66).lines(fill=3), .es(US).bars(1).points(radius=3, weight=1)`	**Named arguments.** Forget trying to remember what order you need to specify arguments in, use named arguments to make the expressions easier to read and write.
`(.es(*), .es(GB)).points()`	**Grouped expressions.** You can also chain groups of expressions to functions. Here, both series are shown as points instead of lines.

Timelion provides additional view transformation functions you can use to customize the appearance of your charts. For the complete list, see the Function reference.



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())`.

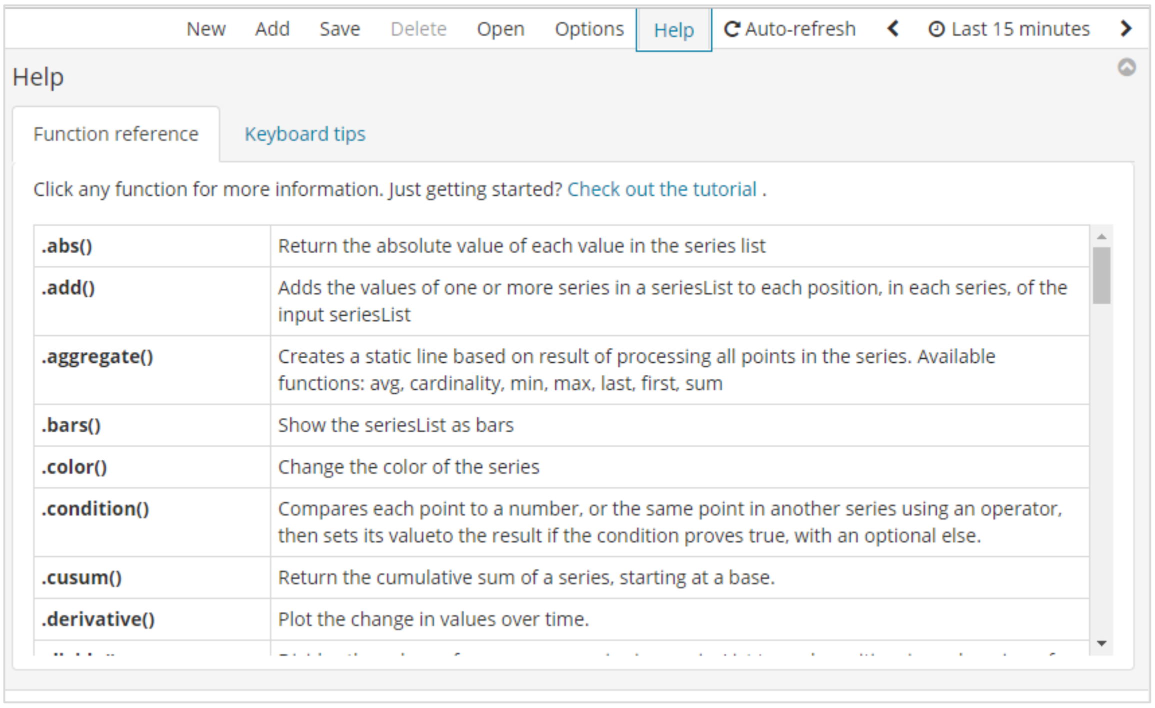
Not bad, but this gives us a number between 0 and 1. To convert it to a percentage, simply multiply by 100: `.es('US').divide(.es()).multiply(100)`.

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 `sum`, `subtract`, `multiply`, and `divide`. Many of these can take a series or a number. There are also other useful data transformation functions, such as `movingaverage`, `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:

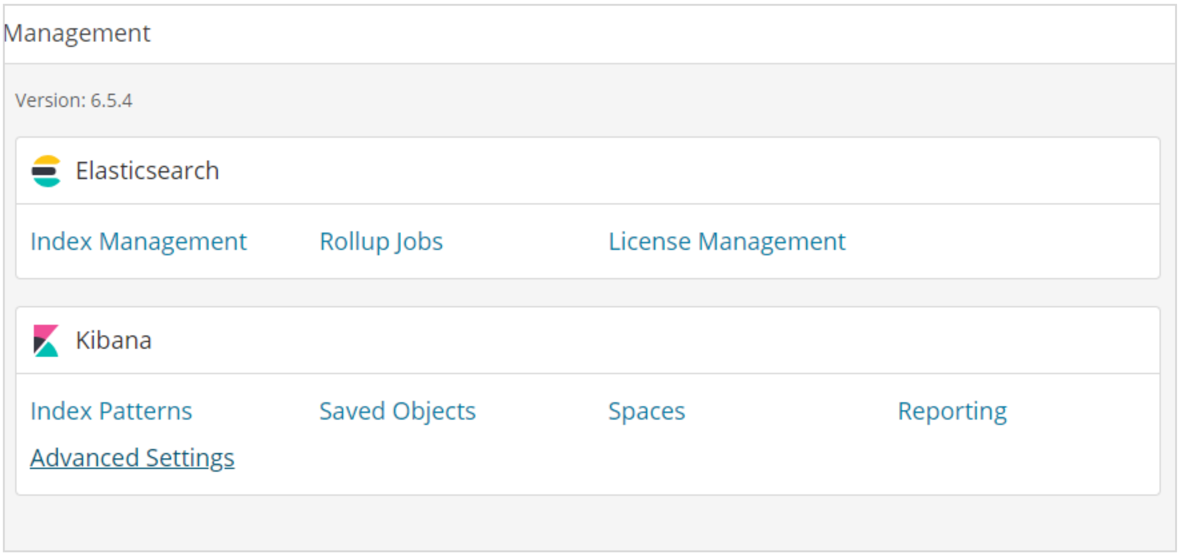


Click any function for more information. Just getting started? Check out the tutorial .

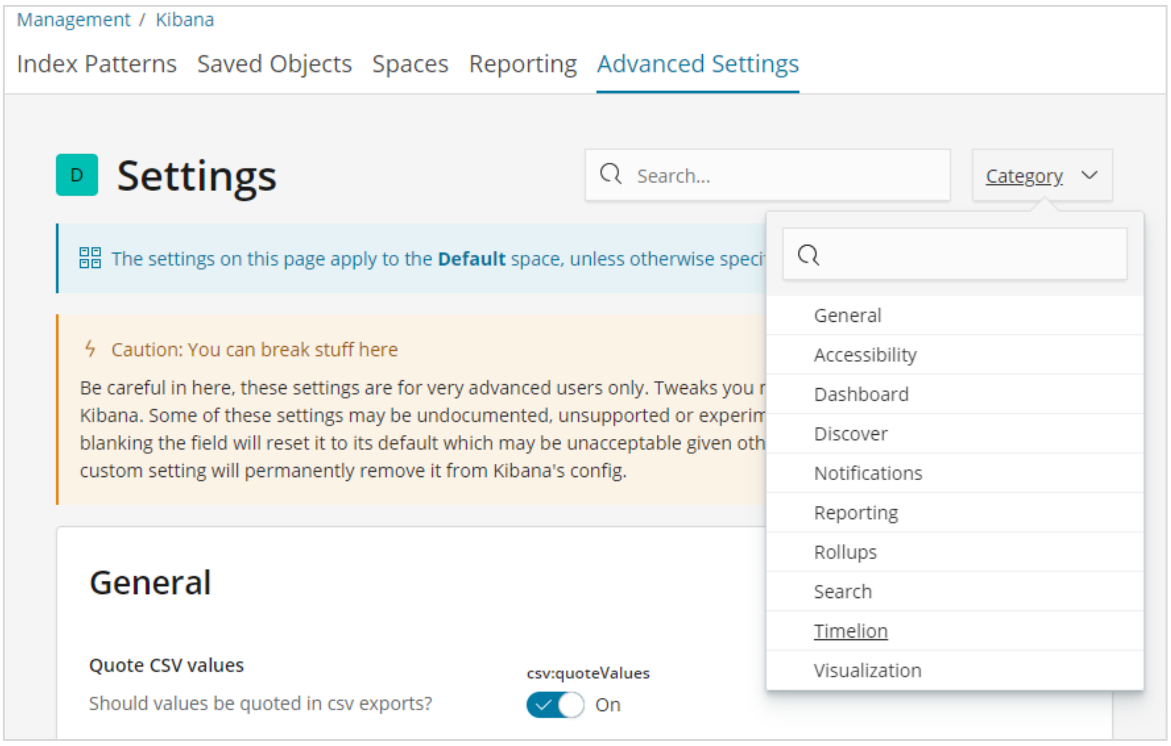
.abs()	Return the absolute value of each value in the series list
.add()	Adds the values of one or more series in a seriesList to each position, in each series, of the input seriesList
.aggregate()	Creates a static line based on result of processing all points in the series. Available functions: avg, cardinality, min, max, last, first, sum
.bars()	Show the seriesList as bars
.color()	Change the color of the series
.condition()	Compares each point to a number, or the same point in another series using an operator, then sets its valueto the result if the condition proves true, with an optional else.
.cusum()	Return the cumulative sum of a series, starting at a base.
.derivative()	Plot the change in values over time.

Timelion Configuration

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



Click on Advanced Settings and select Timelion from Category



The settings on this page apply to the **Default** space, unless otherwise speci

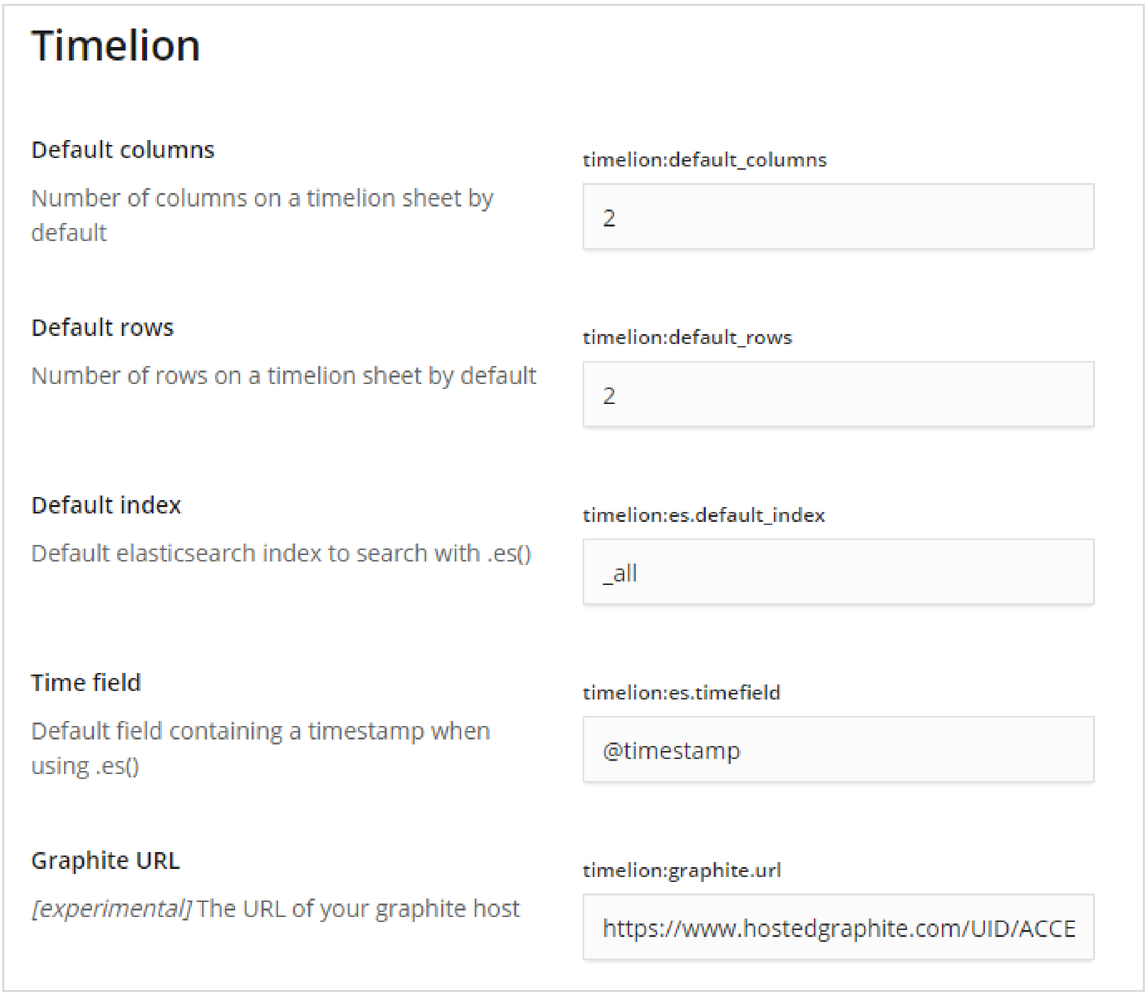
Caution: You can break stuff here

Be careful in here, these settings are for very advanced users only. Tweaks you r
Kibana. Some of these settings may be undocumented, unsupported or experim
blanking the field will reset it to its default which may be unacceptable given oth
custom setting will permanently remove it from Kibana's config.

General

Quote CSV values
Should values be quoted in csv exports?
csv:quoteValues
On

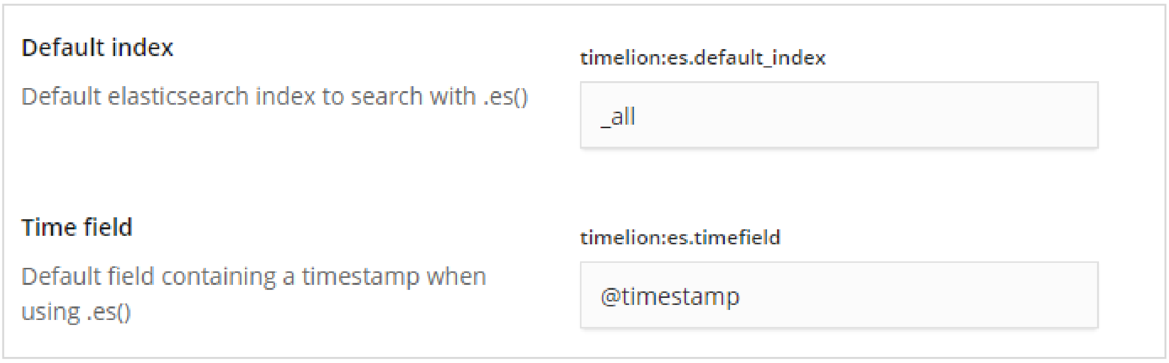
Once Timelion is selected it will display all the necessary fields required for timelion configuration.



Timelion

Default columns Number of columns on a timelion sheet by default	timelion:default_columns: 2
Default rows Number of rows on a timelion sheet by default	timelion:default_rows: 2
Default index Default elasticsearch index to search with .es()	timelion:es.default_index: _all
Time field Default field containing a timestamp when using .es()	timelion:es.timefield: @timestamp
Graphite URL *[experimental]* The URL of your graphite host	timelion:graphite.url: 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:

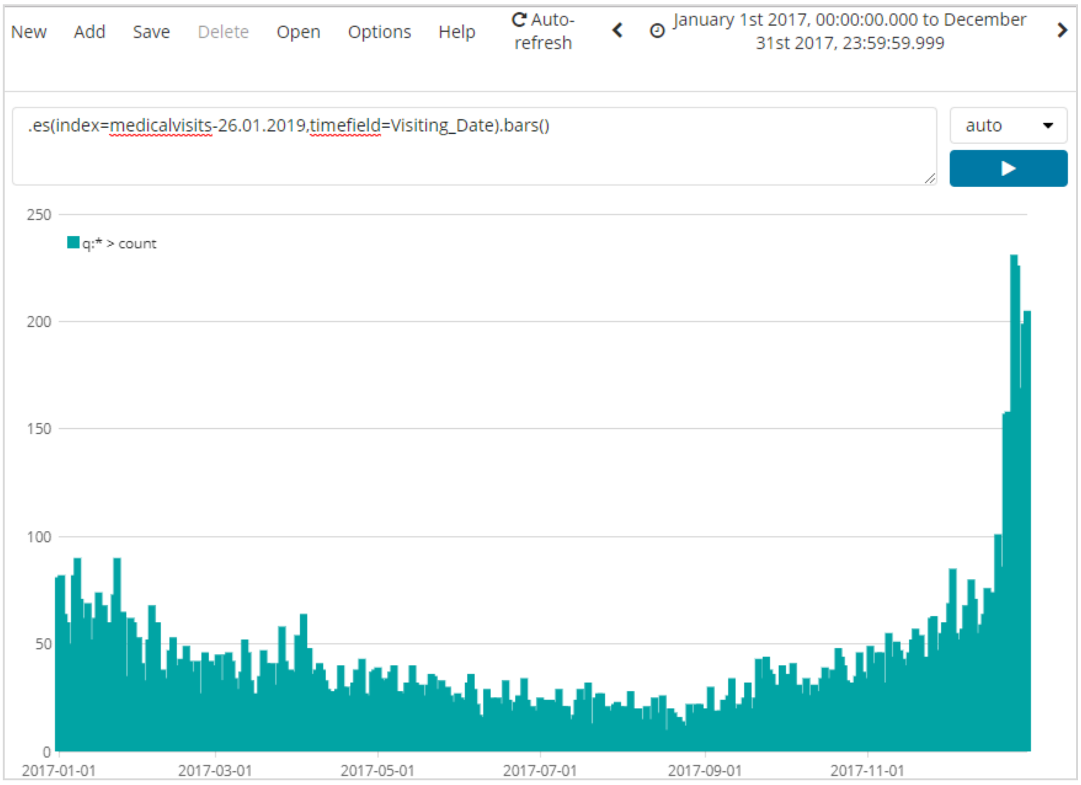


Default index Default elasticsearch index to search with .es()	timelion:es.default_index: _all
Time field Default field containing a timestamp when using .es()	timelion:es.timefield: @timestamp

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 Dec2017:



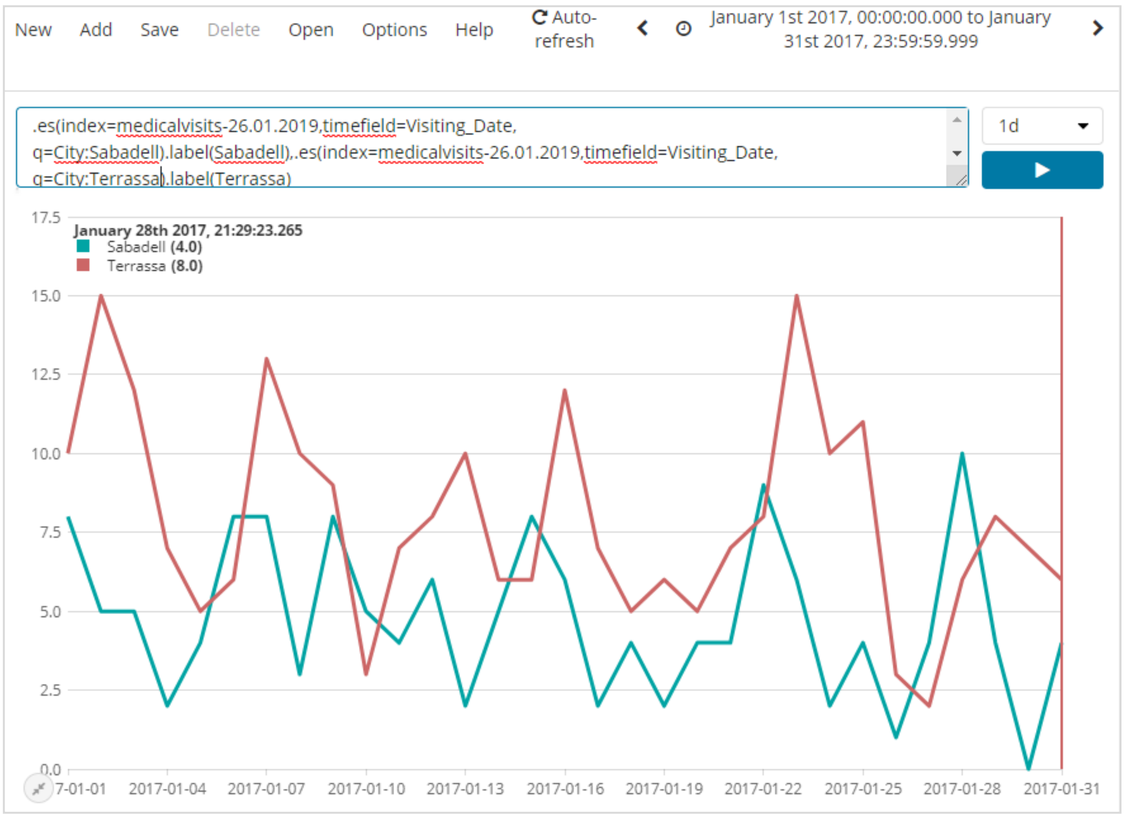
```
.es(index=medicalvisits-26.01.2019,timefield=Visiting_Date).bars()
```

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 jan2017, day wise.



```
.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 expression used is:

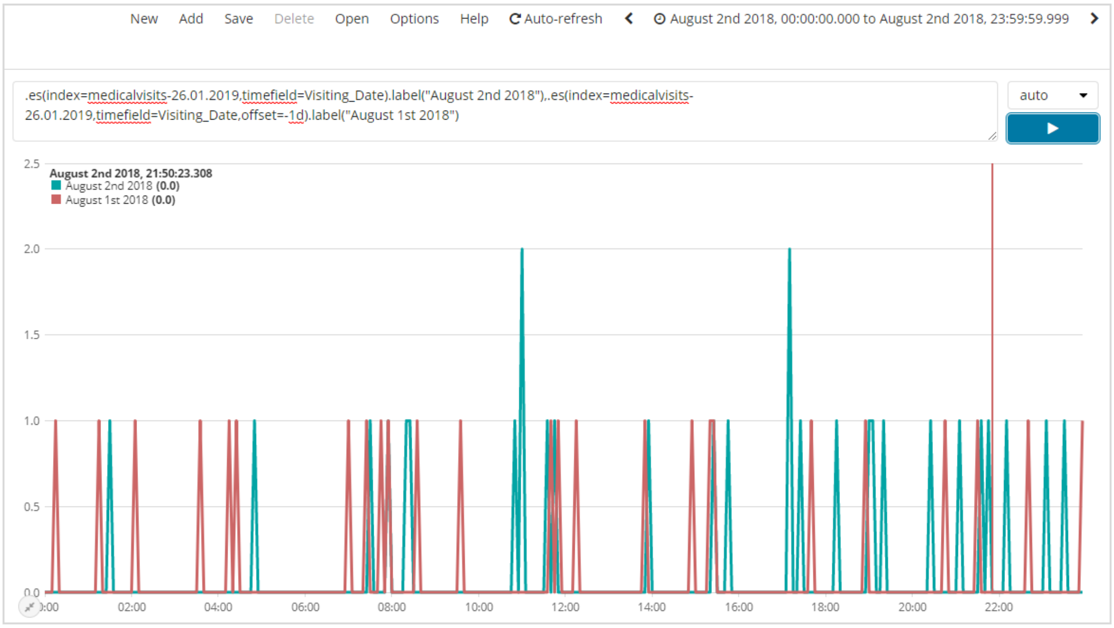
```
.es(index=medicalvisits-26.01.2019,timefield=Visiting_Date,
q=City:Sabadell).label(Sabadell),.es(index=medicalvisits-
26.01.2019,timefield=Visiting_Date, q=City:Terrassa).label(Terrassa)
```

The timeline comparison for 2 days is shown here:

Expression:

```
.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")
```

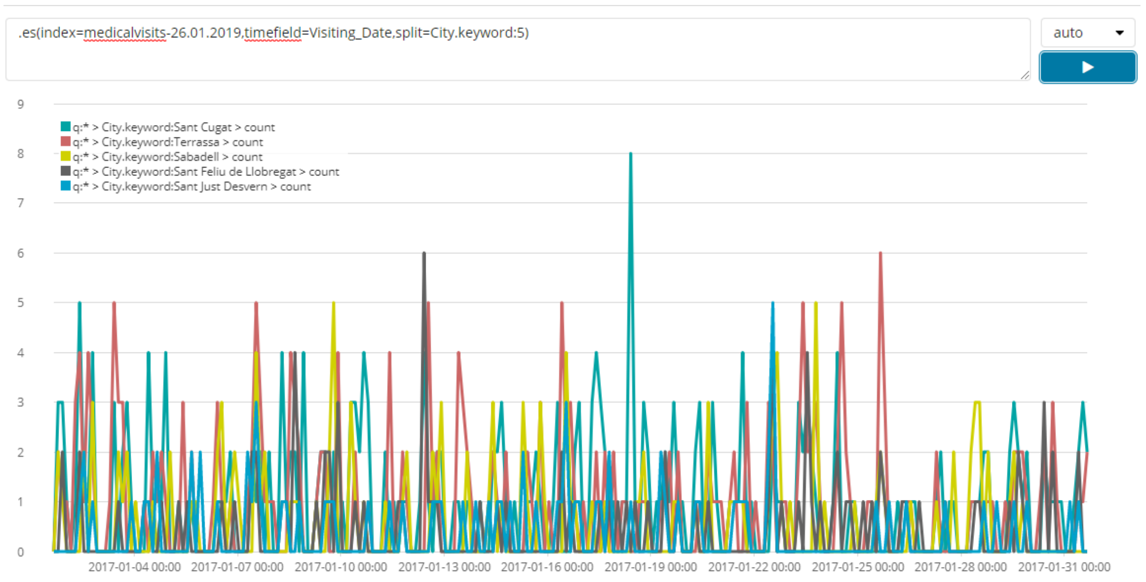
Here we have used offset and given a difference of 1day. We have selected the current date as 2nd August 2018. So it gives data difference for 2nd Aug 2018 and 1st Aug 2018.



```
.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")
```

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)
```


auto
🟩 q:* > City.keyword:Sant Cugat > count
🟥 q:* > City.keyword:Terrassa > count
🟨 q:* > City.keyword:Sabadell > count
⬛ q:* > City.keyword:Sant Feliu de Llobregat > count
🟦 q:* > City.keyword:Sant Just Desvern > count

```
.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.

18. Kibana — Dev Tools

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.

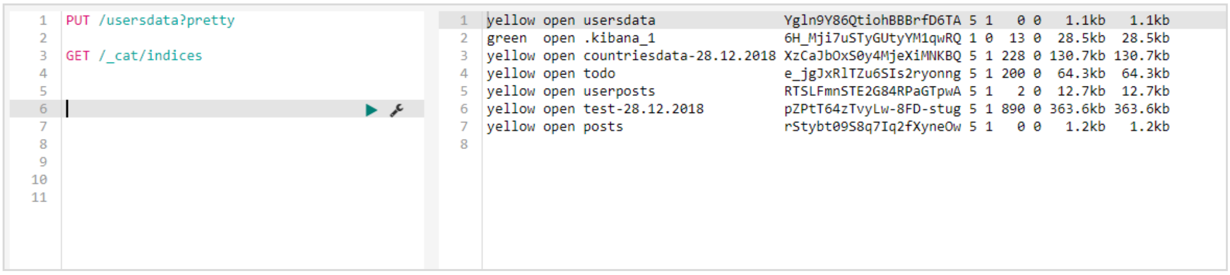
To create new index in Kibana we can use following command in dev tools:

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.



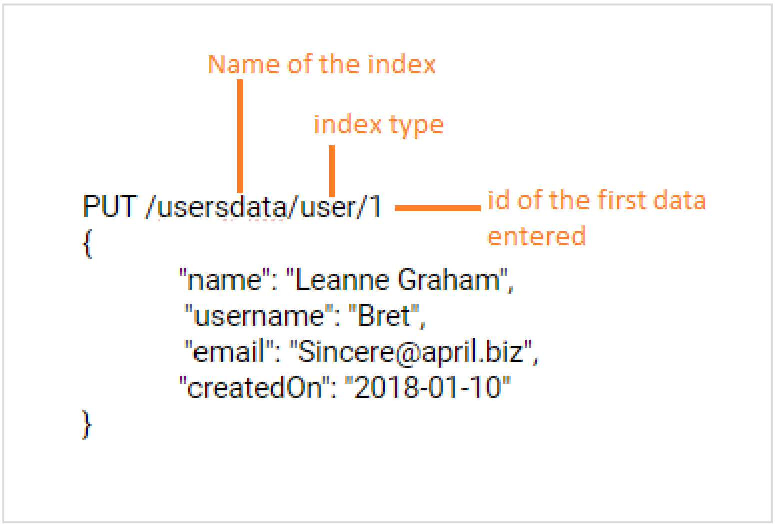
```
PUT /usersdata?pretty

GET /_cat/indices
```

```
yellow open usersdata                  Ygln9Y86QtiohBBBrfD6TA 5 1   0 0   1.1kb   1.1kb
green  open .kibana_1                  6H_Mji7uSTyGUtyYM1qwRQ 1 0  13 0  28.5kb  28.5kb
yellow open countriesdata-28.12.2018 XzCaJbOxS0y4MjeXiMNKBQ 5 1 228 0 130.7kb 130.7kb
yellow open todo                       e_jgJxRlTZu6SIs2ryonng 5 1 200 0  64.3kb  64.3kb
yellow open userposts                  RTSLFmnSTE2G84RPaGTpwA 5 1   2 0  12.7kb  12.7kb
yellow open test-28.12.2018           pZPtT64zTvyLw-8FD-stug 5 1 890 0 363.6kb 363.6kb
yellow open posts                      rStybt09S8q7Iq2fXyneOw 5 1   0 0   1.2kb   1.2kb
```

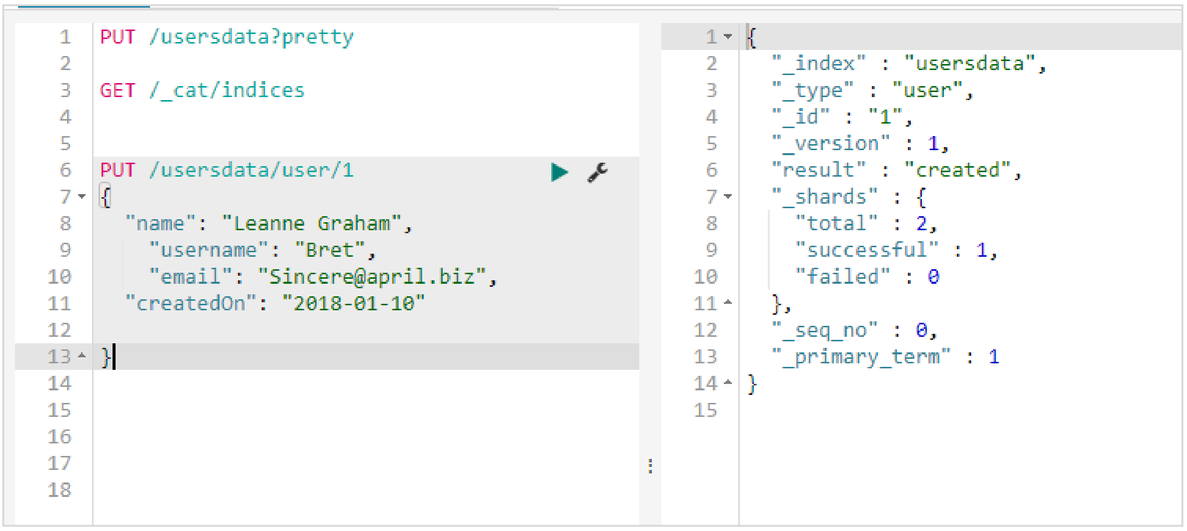
We are done with the index creation. Now will add the data in the index:

Add Data to Index Using PUT

You can add data to an index as follows:



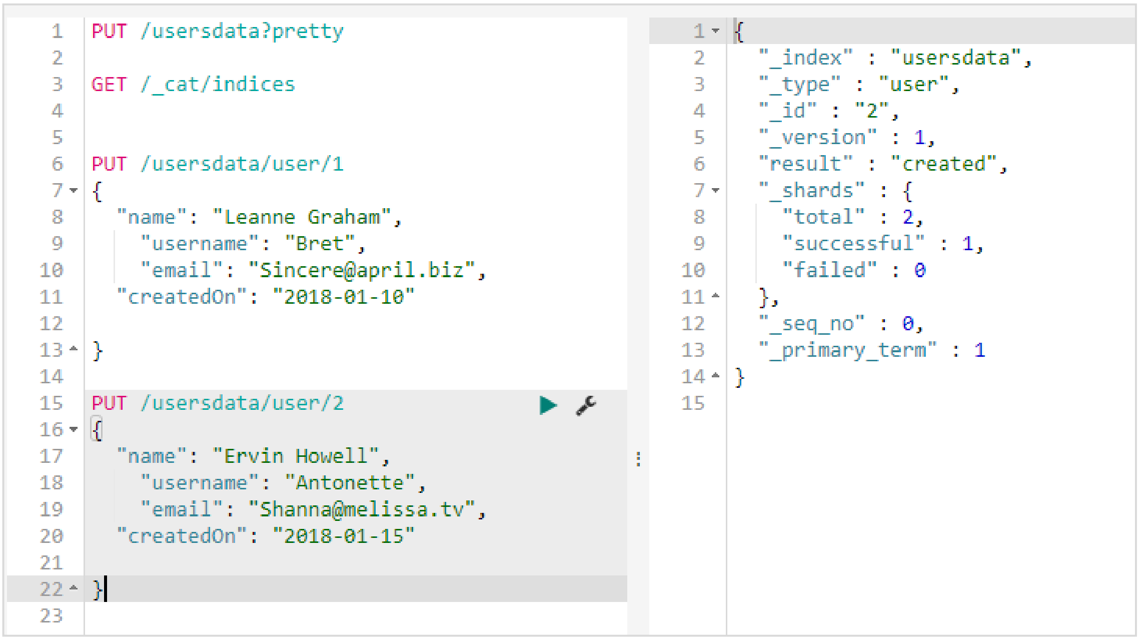
```
PUT /usersdata/user/1
{
        "name": "Leanne Graham",
        "username": "Bret",
        "email": "Sincere@april.biz",
        "createdOn": "2018-01-10"
}
```



```
PUT /usersdata?pretty

GET /_cat/indices


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

}
```

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

We will add one more record in usersdata index:



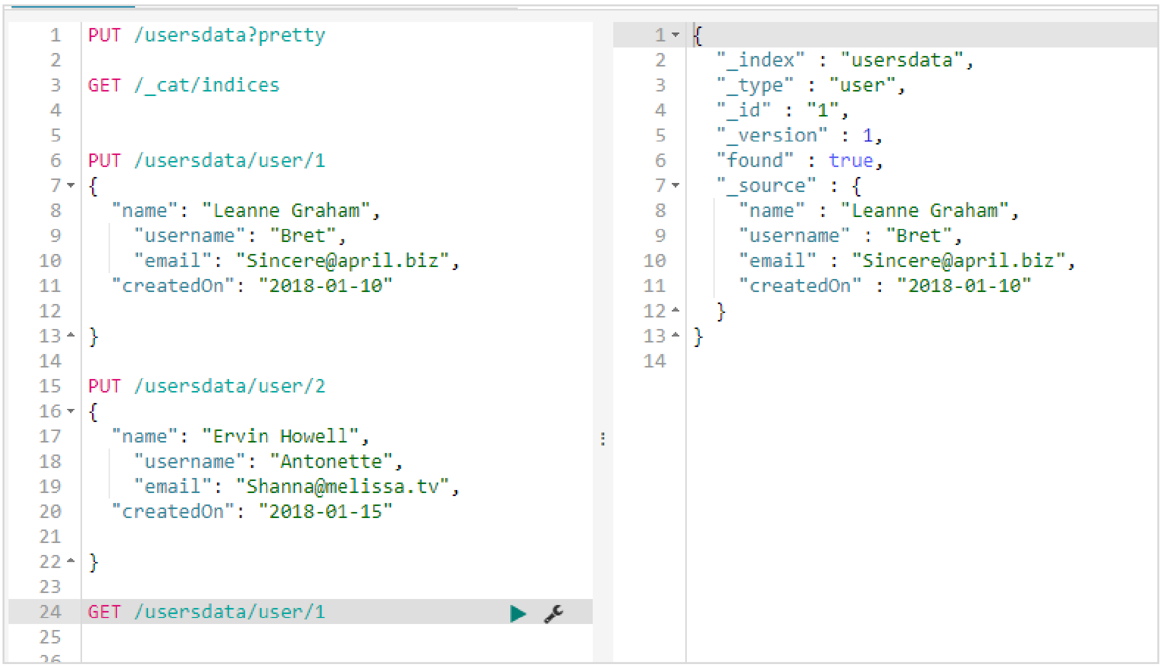
```
PUT /usersdata?pretty

GET /_cat/indices


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

}

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

}
```

```
{
  "_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

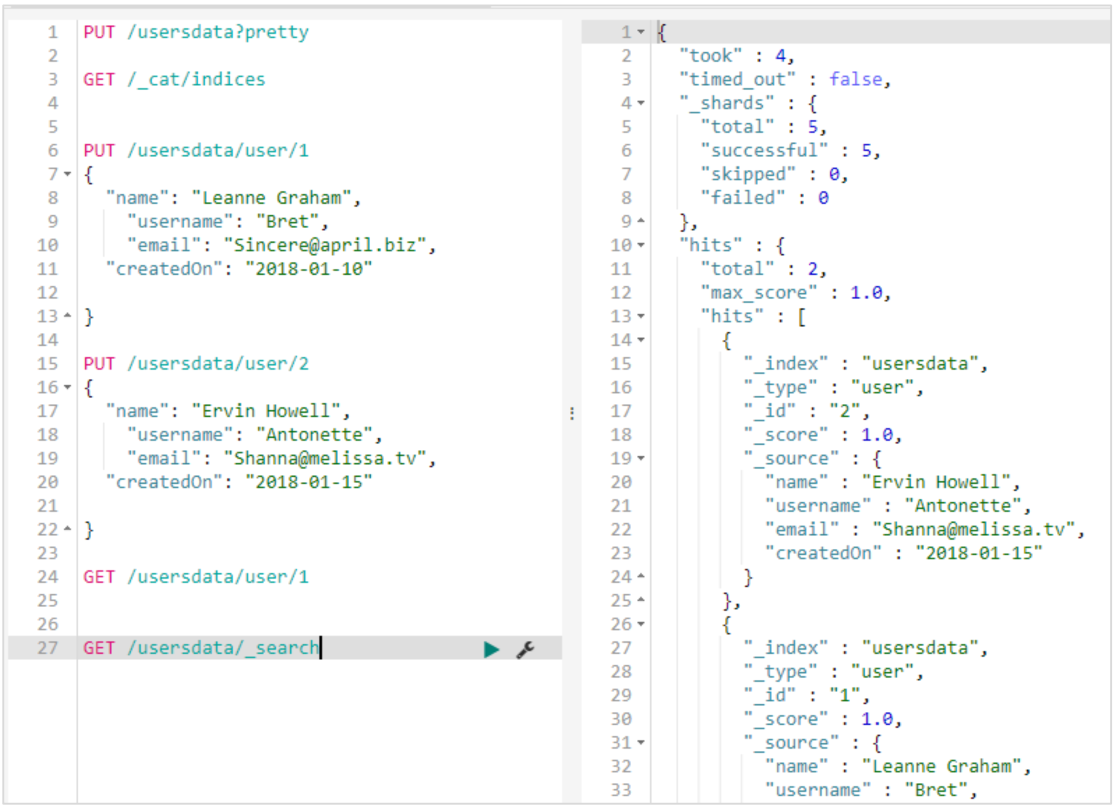
We can get the details of record 1 as follows:



```
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
```

```
{
  "_index" : "usersdata",
  "_type" : "user",
  "_id" : "1",
  "_version" : 1,
  "found" : true,
  "_source" : {
    "name" : "Leanne Graham",
    "username" : "Bret",
    "email" : "Sincere@april.biz",
    "createdOn" : "2018-01-10"
  }
}
```

You can get all records as follows:



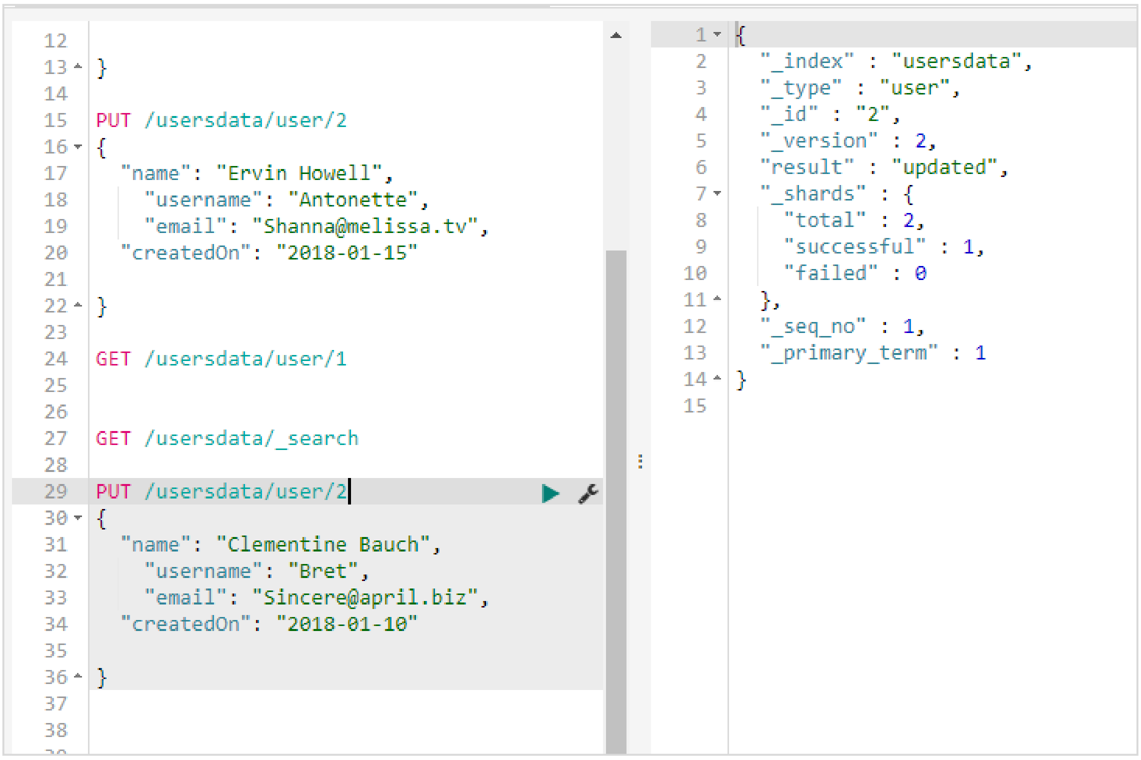
```
PUT /usersdata?pretty

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/user/1


GET /usersdata/_search
```

```
{
  "took" : 4,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 2,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "usersdata",
        "_type" : "user",
        "_id" : "2",
        "_score" : 1.0,
        "_source" : {
          "name" : "Ervin Howell",
          "username" : "Antonette",
          "email" : "Shanna@melissa.tv",
          "createdOn" : "2018-01-15"
        }
      },
      {
        "_index" : "usersdata",
        "_type" : "user",
        "_id" : "1",
        "_score" : 1.0,
        "_source" : {
          "name" : "Leanne Graham",
          "username" : "Bret",
```

Thus, we can get all the records from usersdata as shown above.

Update data in Index using PUT

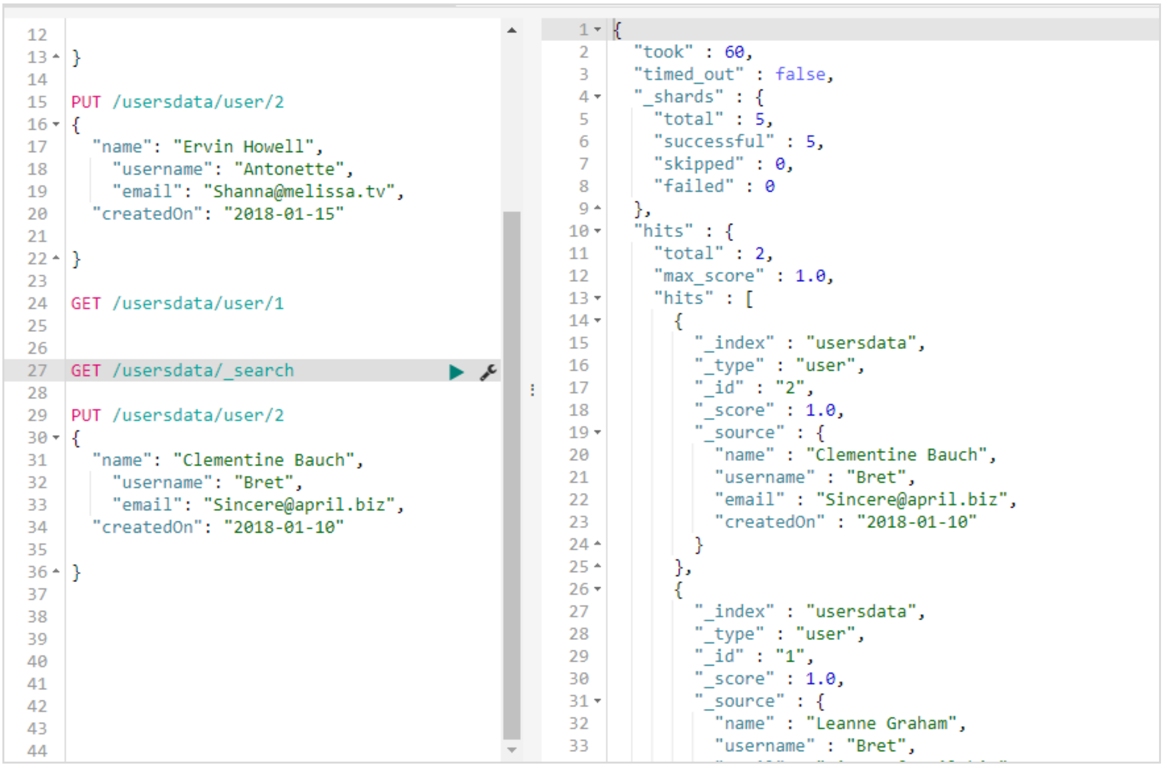
To update the record, you can do as follows:



```
}

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"

}
```

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

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:

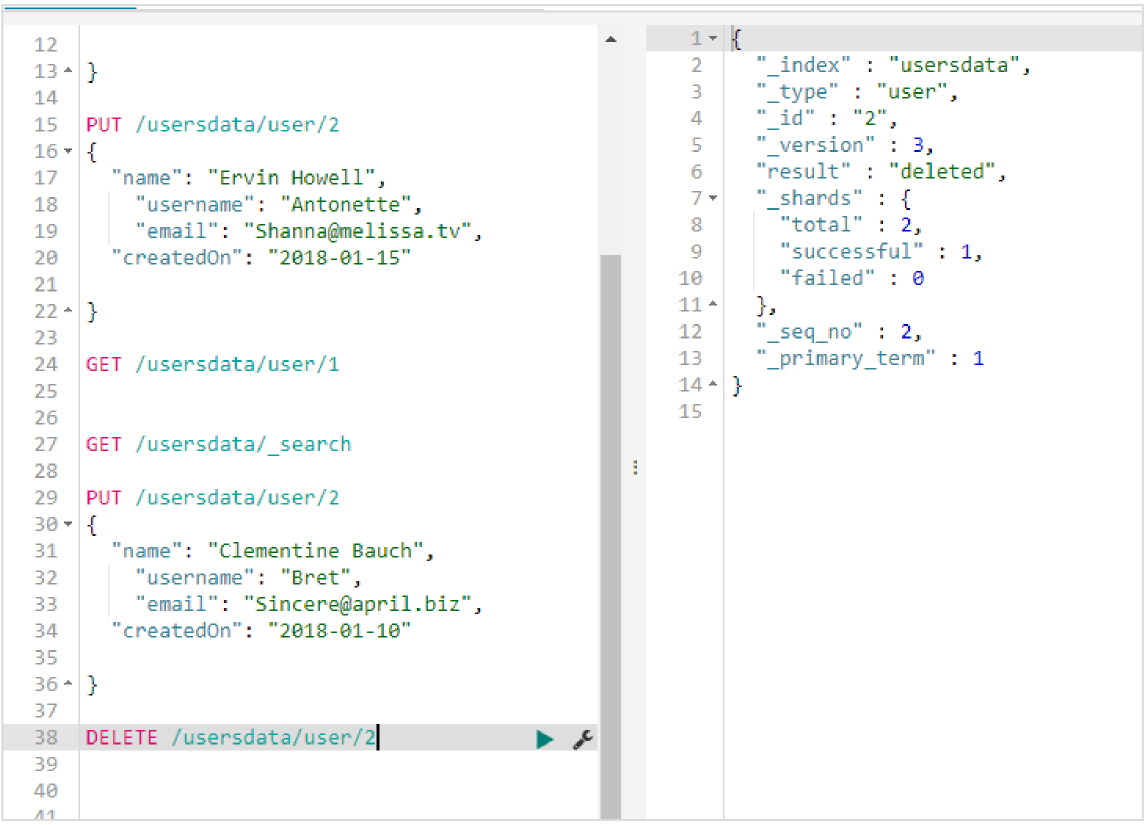


```
}

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"

}
```

```
{
  "took" : 60,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 2,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "usersdata",
        "_type" : "user",
        "_id" : "2",
        "_score" : 1.0,
        "_source" : {
          "name" : "Clementine Bauch",
          "username" : "Bret",
          "email" : "Sincere@april.biz",
          "createdOn" : "2018-01-10"
        }
      },
      {
        "_index" : "usersdata",
        "_type" : "user",
        "_id" : "1",
        "_score" : 1.0,
        "_source" : {
          "name" : "Leanne Graham",
          "username" : "Bret",
```

Delete data from index using DELETE

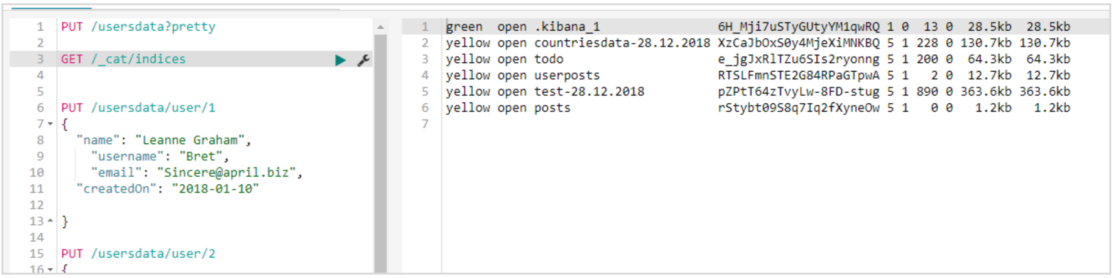
You can delete the record as shown here:



```
}

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
```

```
{
  "_index" : "usersdata",
  "_type" : "user",
  "_id" : "2",
  "_version" : 3,
  "result" : "deleted",
  "_shards" : {
    "total" : 2,
    "successful" : 1,
    "failed" : 0
  },
  "_seq_no" : 2,
  "_primary_term" : 1
}
```

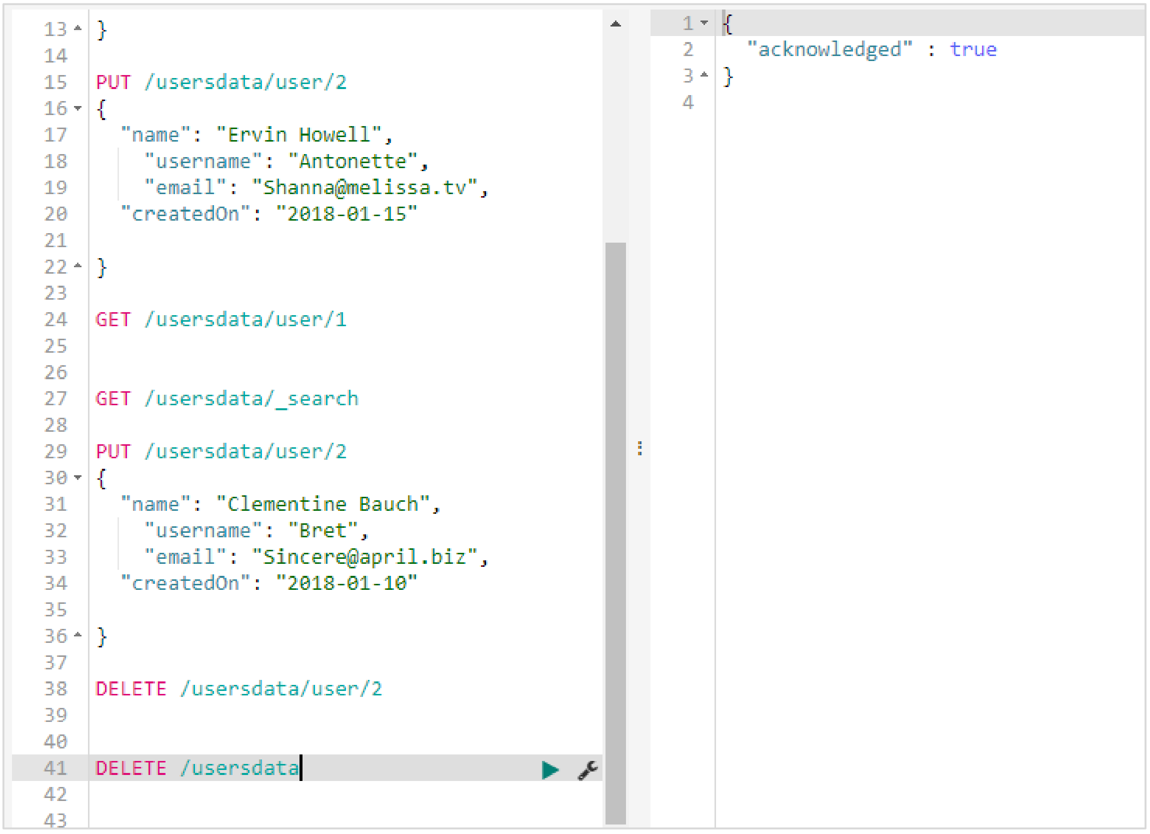
Now if you see the total records we will have only one record:
We can delete the index created as follows:



```
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
{
```

```
green  open .kibana_1                 6H_Mji7uSTyGUtyYM1qwRQ 1 0  13 0  28.5kb  28.5kb
yellow open countriesdata-28.12.2018 XzCaJbOxS0y4MjeXiMNKBQ 5 1 228 0 130.7kb 130.7kb
yellow open todo                     e_jgJxRlTZu6SIs2ryonng 5 1 200 0  64.3kb  64.3kb
yellow open userposts                RTSLFmnSTE2G84RPaGTpwA 5 1   2 0  12.7kb  12.7kb
yellow open test-28.12.2018          pZPtT64zTvyLw-8FD-stug 5 1 890 0 363.6kb 363.6kb
yellow open posts                    rStybt09S8q7Iq2fXyneOw 5 1   0 0   1.2kb   1.2kb
```

19.



```
}

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


DELETE /usersdata
```

```
{
  "acknowledged" : true
}
```

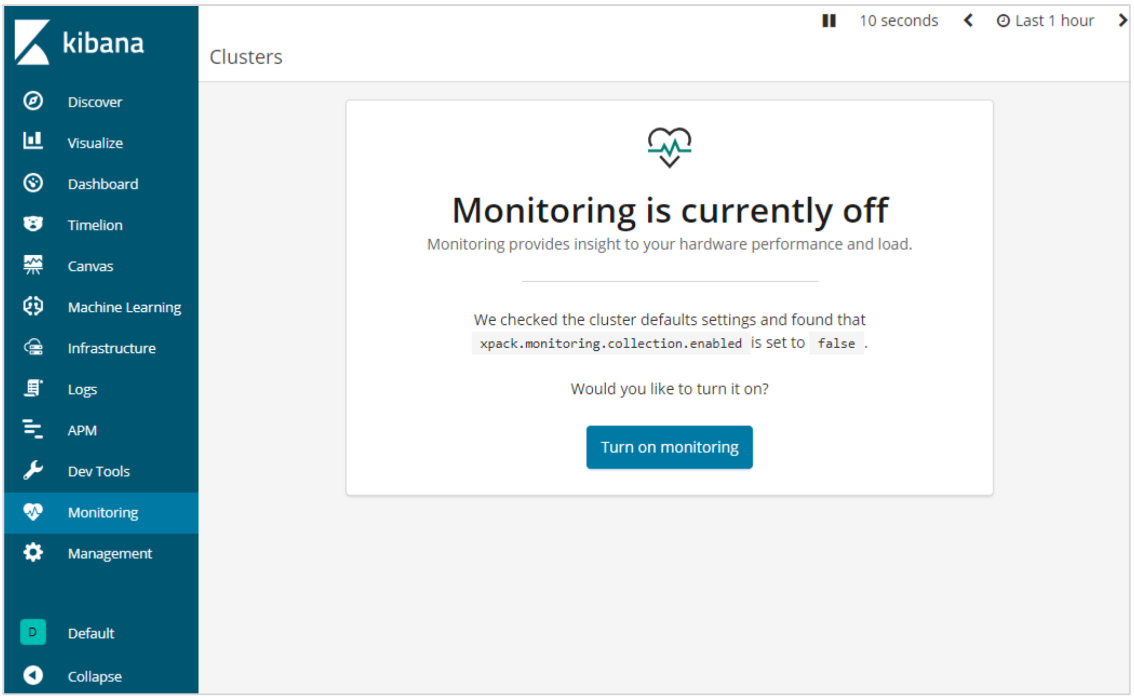
Now if you check the indices available we will not have usersdata index in it as deleted the index.

19. Kibana — Monitoring

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:



Monitoring is currently off

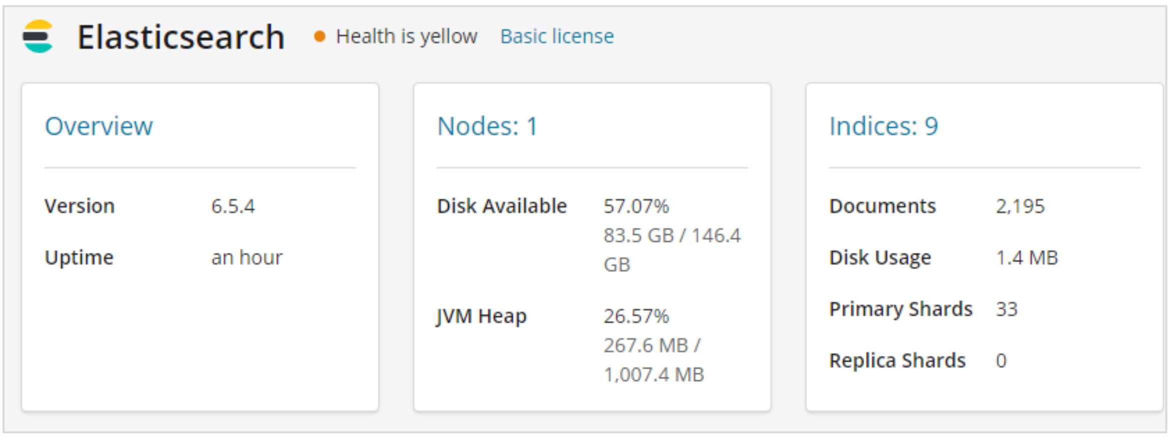
Monitoring provides insight to your hardware performance and load.

We checked the cluster defaults settings and found that `xpack.monitoring.collection.enabled` is set to `false`.

Would you like to turn it on?

`

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:



Elasticsearch • Health is yellow Basic license

Overview

Version	6.5.4
Uptime	an hour

Nodes: 1

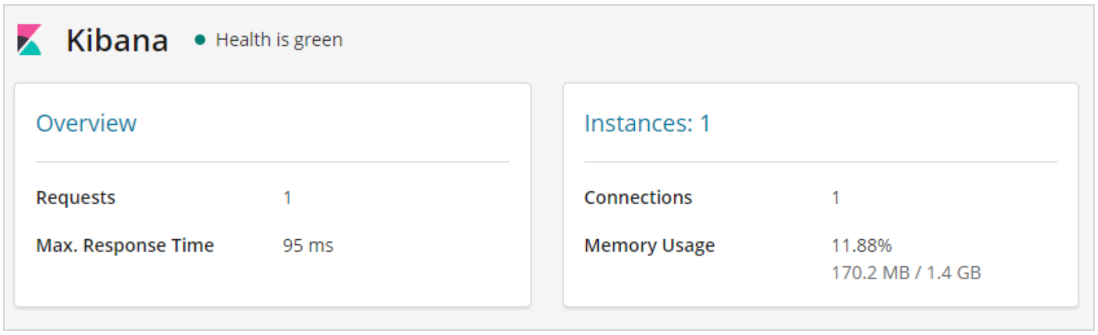
Disk Available	57.07% 83.5 GB / 146.4 GB
JVM Heap	26.57% 267.6 MB / 1,007.4 MB

Indices: 9

Documents	2,195
Disk Usage	1.4 MB
Primary Shards	33
Replica Shards	0

It gives the version of elasticsearch, disk available, indices added to elasticsearch, disk usage etc.

The monitoring details for Kibana are shown here:



Kibana ● Health is green

Overview	
Requests	1
Max. Response Time	95 ms

Instances: 1	
Connections	1
Memory Usage	11.88% 170.2 MB / 1.4 GB

It gives the Requests and max response time for the request and also the instances running and memory usage.

20. Kibana — Create Reports Using Kibana

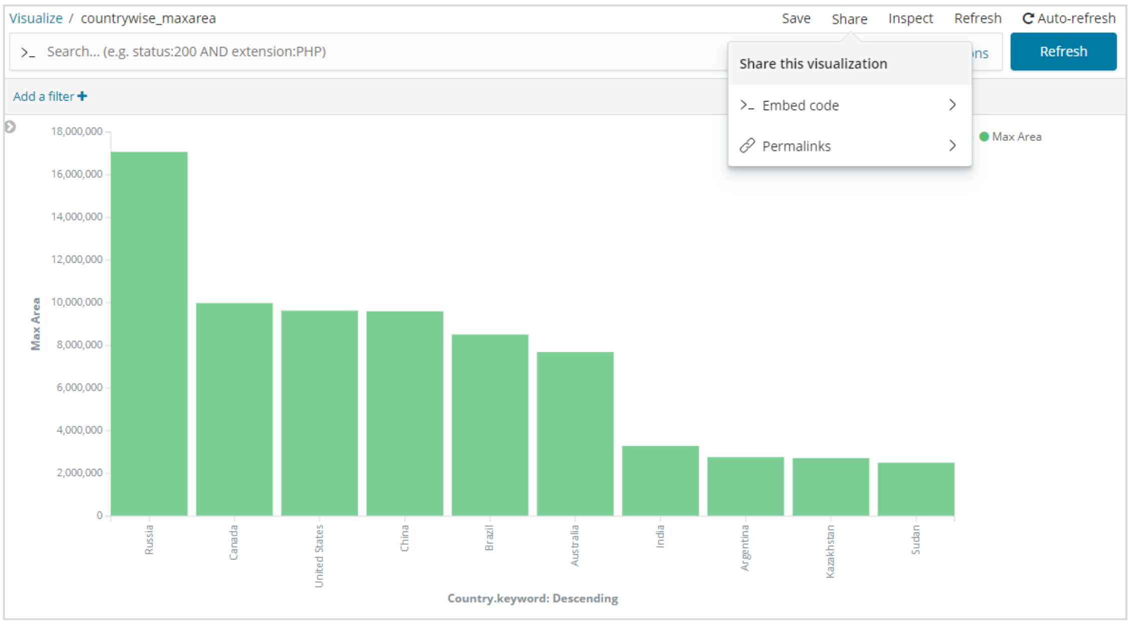
Reports can be easily created by using the Share button available in Kibana UI.

Reports in Kibana are available in the following two forms:

- Permalinks
- CSV Report

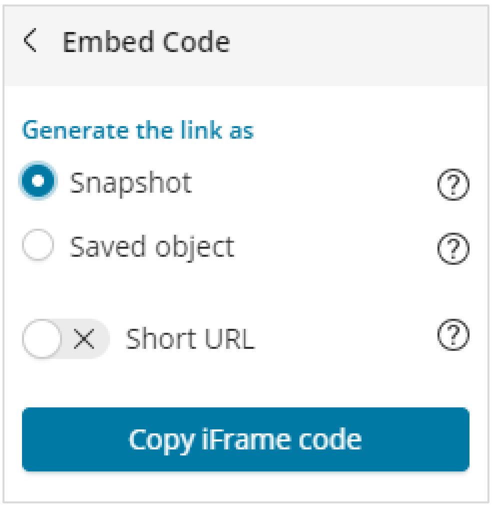
Report as Permalinks

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:

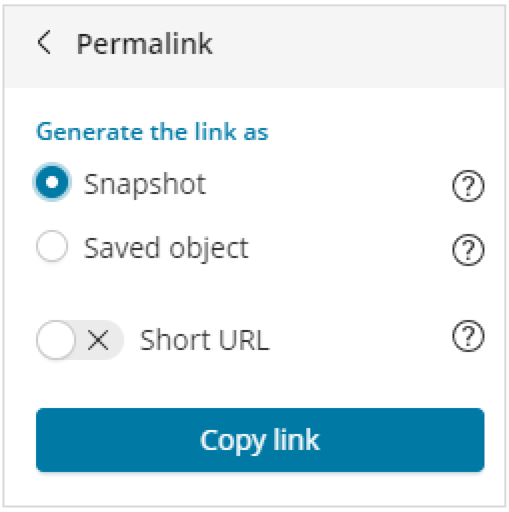
```
<iframe src="http://localhost:5601/app/kibana#/visualize/edit/87afcb60-165f-
11e9-aaf1-
3524d1f04792?embed=true&_g=()&_a=(filters:!(),linked:!f,query:(language:lucene,
query:''),uiState:(),vis:(aggs:!((enabled:!t,id:'1',params:(field:Area),schema:
metric,type:max),(enabled:!t,id:'2',params:(field:Country.keyword,missingBucket
:!f,missingBucketLabel:Missing,order:desc,orderBy:'1',otherBucket:!f,otherBucke
tLabel:Other,size:10),schema:segment,type:terms)),params:(addLegend:!t,addTimeM
arker:!f,addTooltip:!t,categoryAxes:!((id:CategoryAxis-
1,labels:(show:!t,truncate:100),position:bottom,scale:(type:linear),show:!t,sty
le:(),title:(),type:category)),grid:(categoryLines:!f,style:(color:%23eee)),leg
endPosition:right,seriesParams:!((data:(id:'1',label:'Max+Area'),drawLinesBetwe
enPoints:!t,mode:stacked,show:true,showCircles:!t,type:histogram,valueAxis:Valu
eAxis-1)),times:!(),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:'M
ax+Area'),type:value))),title:'countrywise_maxarea+',type:histogram))"
height="600" width="800"></iframe>
```

Snapshot Iframe code for short url:

```
<iframe
src="http://localhost:5601/goto/f0a6c852daedcb6b4fa74cce8c2ff6c4?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:



Permalink

Generate the link as

[✓ Snapshot] Saved object

Short URL: off

As snapshot and shot url.

With Short url:

```
http://localhost:5601/goto/f0a6c852daedcb6b4fa74cce8c2ff6c4
```

With Short url off, the link looks as below:

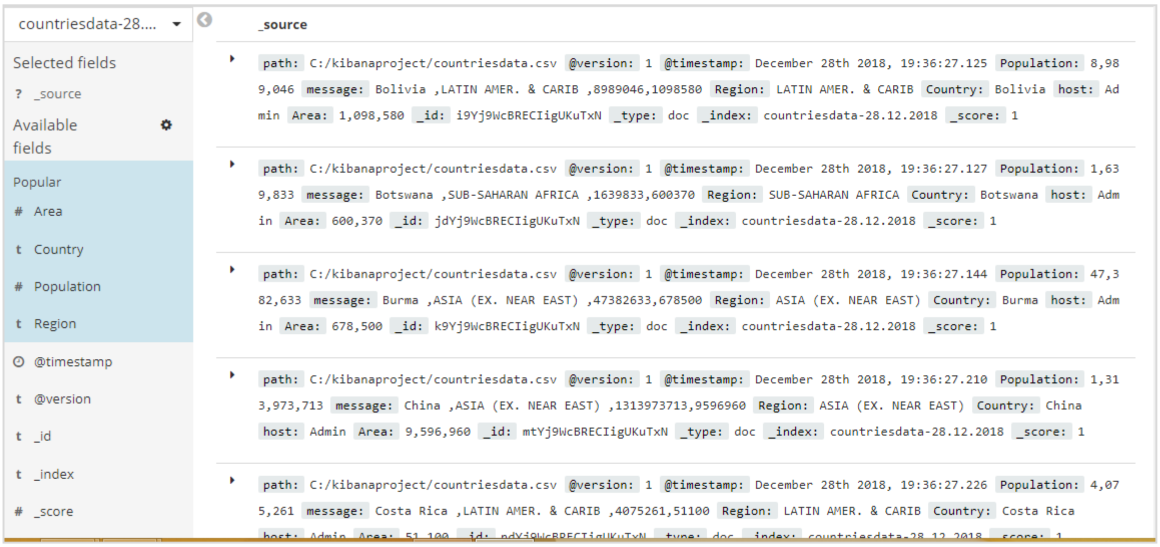
```
http://localhost:5601/app/kibana#/visualize/edit/87afcb60-165f-11e9-aaf1-
3524d1f04792?_g=()&_a=(filters:!(),linked:!f,query:(language:lucene,query:''),u
iState:(),vis:(aggs:!((enabled:!t,id:'1',params:(field:Area),schema:metric,type
:max),(enabled:!t,id:'2',params:(field:Country.keyword,missingBucket:!f,missing
BucketLabel:Missing,order:desc,orderBy:'1',otherBucket:!f,otherBucketLabel:Othe
r,size:10),schema:segment,type:terms)),params:(addLegend:!t,addTimeMarker:!f,ad
dTooltip:!t,categoryAxes:!((id:CategoryAxis-
1,labels:(show:!t,truncate:100),position:bottom,scale:(type:linear),show:!t,sty
le:(),title:(),type:category)),grid:(categoryLines:!f,style:(color:%23eee)),leg
endPosition:right,seriesParams:!((data:(id:'1',label:'Max%20Area'),drawLinesBet
weenPoints:!t,mode:stacked,show:true,showCircles:!t,type:histogram,valueAxis:Va
lueAxis-1)),times:!(),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:'M
ax%20Area'),type:value))),title:'countrywise_maxarea%20',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.

CSV Report

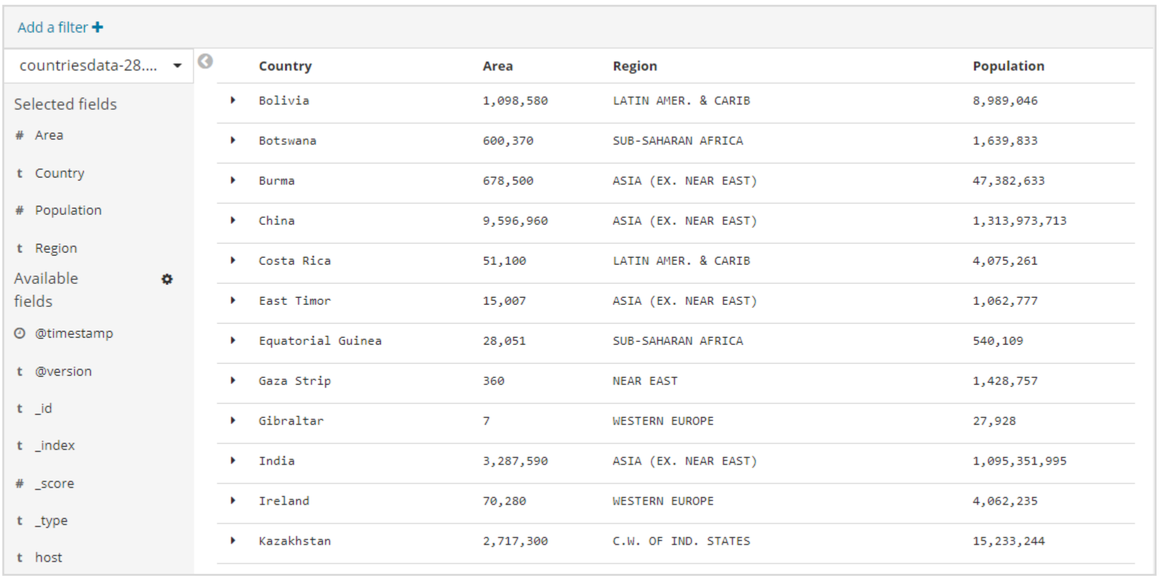
You can get CSV Report in Kibana where there is data, which is mostly in the Discover tab.

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:



```
_source

path: C:/kibanaproject/countriesdata.csv @version: 1 @timestamp: December 28th 2018, 19:36:27.125 Population: 8,989,046 message: Bolivia ,LATIN AMER. & CARIB ,8989046,1098580 Region: LATIN AMER. & CARIB Country: Bolivia host: Admin Area: 1,098,580 _id: i9Yj9WcBRECIigUKuTxN _type: doc _index: countriesdata-28.12.2018 _score: 1

path: C:/kibanaproject/countriesdata.csv @version: 1 @timestamp: December 28th 2018, 19:36:27.127 Population: 1,639,833 message: Botswana ,SUB-SAHARAN AFRICA ,1639833,600370 Region: SUB-SAHARAN AFRICA Country: Botswana host: Admin Area: 600,370 _id: jdYj9WcBRECIigUKuTxN _type: doc _index: countriesdata-28.12.2018 _score: 1

path: C:/kibanaproject/countriesdata.csv @version: 1 @timestamp: December 28th 2018, 19:36:27.144 Population: 47,382,633 message: Burma ,ASIA (EX. NEAR EAST) ,47382633,678500 Region: ASIA (EX. NEAR EAST) Country: Burma host: Admin Area: 678,500 _id: k9Yj9WcBRECIigUKuTxN _type: doc _index: countriesdata-28.12.2018 _score: 1

path: C:/kibanaproject/countriesdata.csv @version: 1 @timestamp: December 28th 2018, 19:36:27.210 Population: 1,313,973,713 message: China ,ASIA (EX. NEAR EAST) ,1313973713,9596960 Region: ASIA (EX. NEAR EAST) Country: China host: Admin Area: 9,596,960 _id: mtYj9WcBRECIigUKuTxN _type: doc _index: countriesdata-28.12.2018 _score: 1

path: C:/kibanaproject/countriesdata.csv @version: 1 @timestamp: December 28th 2018, 19:36:27.226 Population: 4,075,261 message: Costa Rica ,LATIN AMER. & CARIB ,4075261,51100 Region: LATIN AMER. & CARIB Country: Costa Rica host: Admin Area: 51,100 _id: pdYj9WcBRECIigUKuTxN _type: doc _index: countriesdata-28.12.2018 _score: 1
```

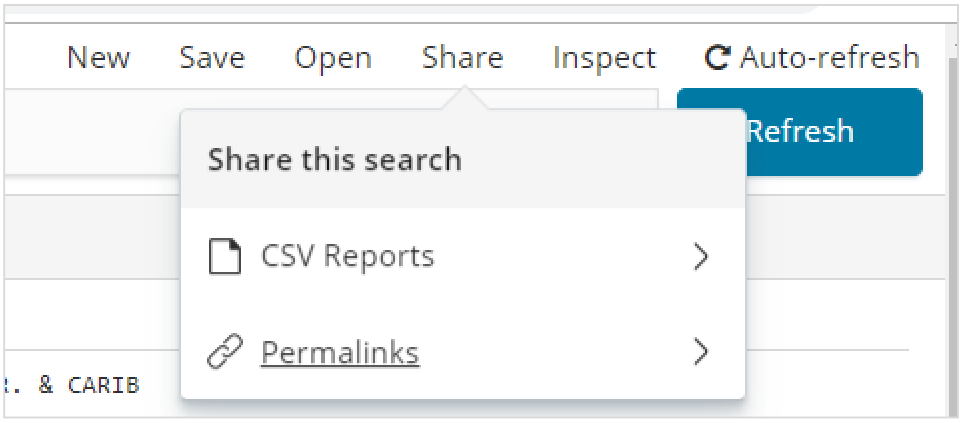
You can create tabular data from above data as shown below:



Country	Area	Region	Population
Bolivia	1,098,580	LATIN AMER. & CARIB	8,989,046
Botswana	600,370	SUB-SAHARAN AFRICA	1,639,833
Burma	678,500	ASIA (EX. NEAR EAST)	47,382,633
China	9,596,960	ASIA (EX. NEAR EAST)	1,313,973,713
Costa Rica	51,100	LATIN AMER. & CARIB	4,075,261
East Timor	15,007	ASIA (EX. NEAR EAST)	1,062,777
Equatorial Guinea	28,051	SUB-SAHARAN AFRICA	540,109
Gaza Strip	360	NEAR EAST	1,428,757
Gibraltar	7	WESTERN EUROPE	27,928
India	3,287,590	ASIA (EX. NEAR EAST)	1,095,351,995
Ireland	70,280	WESTERN EUROPE	4,062,235
Kazakhstan	2,717,300	C.W. OF IND. STATES	15,233,244

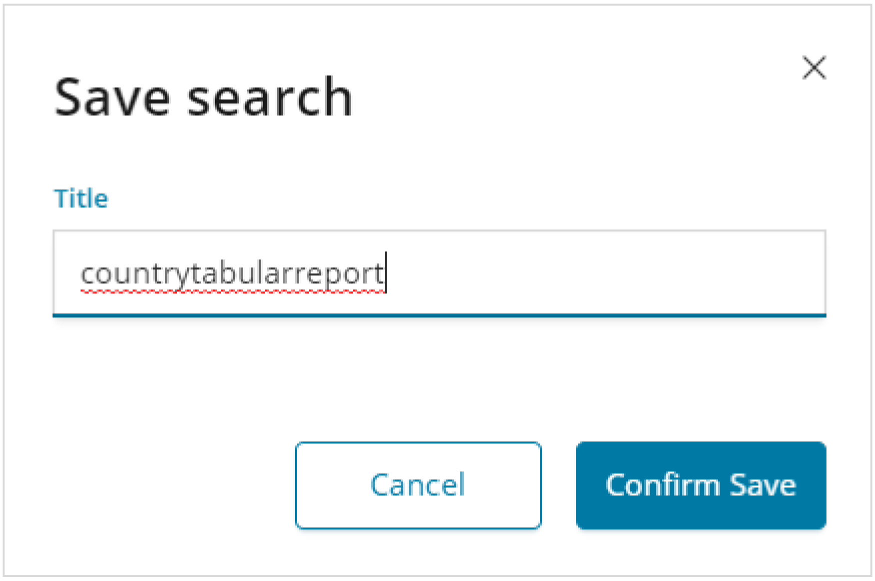
We have selected the fields from Available fields and the data seen earlier is converted into tabular format.

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

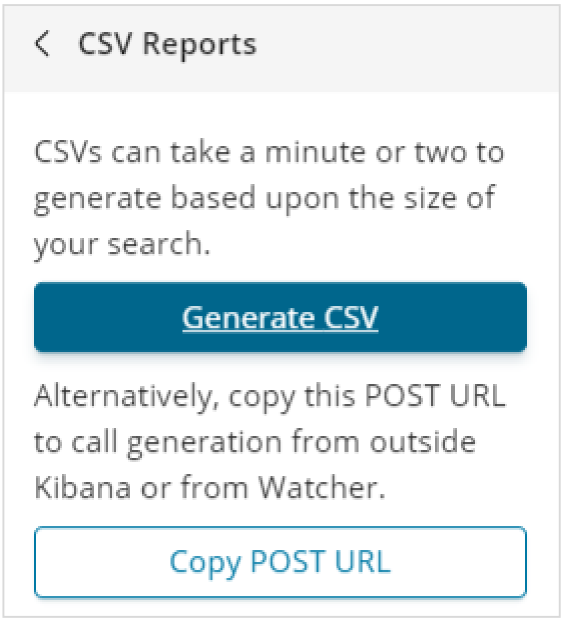


The share button has option for CSV report and permalinks. You can click on CSV Report and download the same.

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:

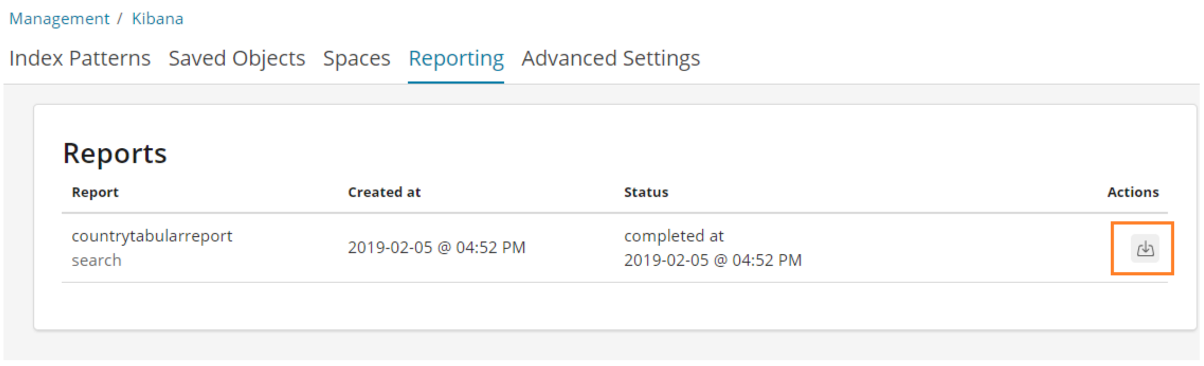


CSVs can take a minute or two to generate based upon the size of your search.

Alternatively, copy this POST URL to call generation from outside Kibana or from Watcher.

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

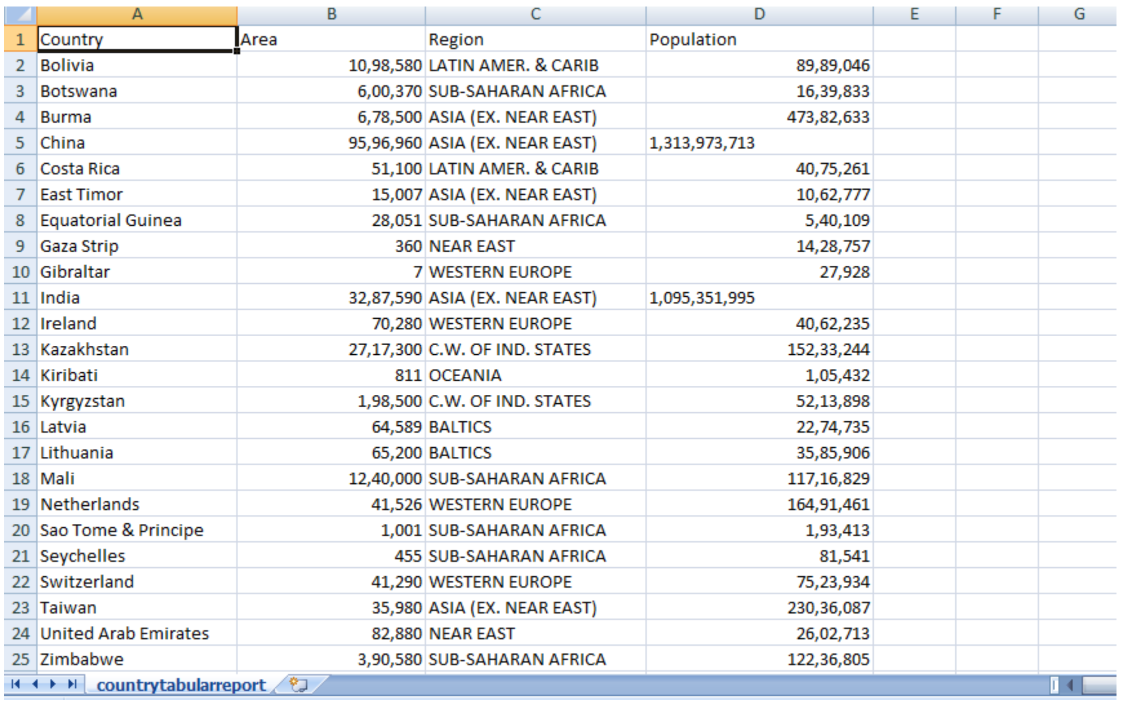
Go to Management Tab -> Reporting



Report	Created at	Status	Actions
countrytabularreport search	2019-02-05 @ 04:52 PM	completed at 2019-02-05 @ 04:52 PM	

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:



	A	B	C	D	E	F	G
1	Country	Area	Region	Population			
2	Bolivia	10,98,580	LATIN AMER. & CARIB	89,89,046			
3	Botswana	6,00,370	SUB-SAHARAN AFRICA	16,39,833			
4	Burma	6,78,500	ASIA (EX. NEAR EAST)	473,82,633			
5	China	95,96,960	ASIA (EX. NEAR EAST)	1,313,973,713			
6	Costa Rica	51,100	LATIN AMER. & CARIB	40,75,261			
7	East Timor	15,007	ASIA (EX. NEAR EAST)	10,62,777			
8	Equatorial Guinea	28,051	SUB-SAHARAN AFRICA	5,40,109			
9	Gaza Strip	360	NEAR EAST	14,28,757			
10	Gibraltar	7	WESTERN EUROPE	27,928			
11	India	32,87,590	ASIA (EX. NEAR EAST)	1,095,351,995			
12	Ireland	70,280	WESTERN EUROPE	40,62,235			
13	Kazakhstan	27,17,300	C.W. OF IND. STATES	152,33,244			
14	Kiribati	811	OCEANIA	1,05,432			
15	Kyrgyzstan	1,98,500	C.W. OF IND. STATES	52,13,898			
16	Latvia	64,589	BALTICS	22,74,735			
17	Lithuania	65,200	BALTICS	35,85,906			
18	Mali	12,40,000	SUB-SAHARAN AFRICA	117,16,829			
19	Netherlands	41,526	WESTERN EUROPE	164,91,461			
20	Sao Tome & Principe	1,001	SUB-SAHARAN AFRICA	1,93,413			
21	Seychelles	455	SUB-SAHARAN AFRICA	81,541			
22	Switzerland	41,290	WESTERN EUROPE	75,23,934			
23	Taiwan	35,980	ASIA (EX. NEAR EAST)	230,36,087			
24	United Arab Emirates	82,880	NEAR EAST	26,02,713			
25	Zimbabwe	3,90,580	SUB-SAHARAN AFRICA	122,36,805			