

ELASTICSEARCH LOGSTASH KIBANA DASHBOARD TUTORIAL FOR WIRESHARK CSV FILES

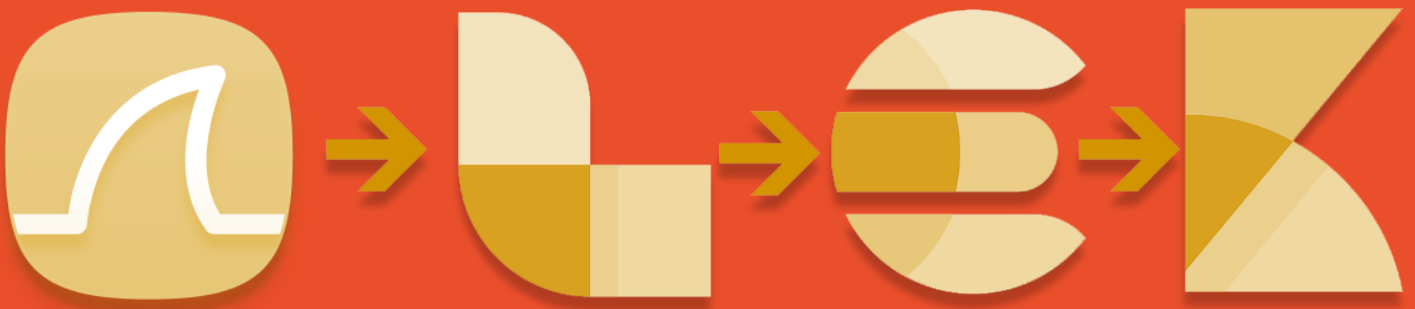


Table of Contents

	Page
Introduction	1
Installation	2
Running Elasticsearch and Kibana	4
Elasticsearch mapping	6
Converting csv file through Logstash	9
Kibana settings	15
Creating visualisations through Kibana	16
Creating dashboard through Kibana	29
References	30
Appendices	31

Introduction

Elasticsearch-Logstash-Kibana Stack

The Elasticsearch-Logstash-Kibana (ELK) stack is a pipeline for storing and retrieving data with Elasticsearch as the central executor. The ELK stack was selected to best achieve our objective of creating an external behaviour monitoring platform for IoT devices that near instantaneously retrieves data even when voluminous IoT devices are connected. Elasticsearch is a NoSQL distributed database and therefore retrieves data faster when compared to a relational database (Mu, Zhao, Yang, Zhang & Yan, 2018). A brief description of each agent in the stack is provided below:

① Elasticsearch

Open source information retrieval system developed by Elastic and built on Lucene. It is scalable, supports real-time indexing and searching, and utilizes JavaScript Object Notation. Due to its optimization for performance, the system is popularly used as a monitoring platform by service providing companies such as Facebook, GitHub, Stack Exchange and eCommerce websites like Ebay and Netflix (Mu, Zhao, Yang, Zhang & Yan, 2018).

② Logstash

Open source tool that collects and parses events or data to store it as logs. A pipeline needs to be specified as a configuration file (Arnold, 2014).

③ Kibana

Open source data visualisation plugin for Elasticsearch data on a web interface. Users can define various visualizations of data in Elasticsearch and combine them to produce dashboards. The visualizations and dashboards are stored to an Elasticsearch index (Takase, Nakamura, Watase & Sasaki, 2017).

In addition to ELK, the following software was also used to create the dashboard:

① Ubuntu

Open source operating system and Debian-based Linux distribution. The Ubuntu terminal is used over the Windows command prompt terminal as it offers better compatibility with the ELK stack.

Installation

Installing Ubuntu for Windows

- ① Download and install *Ubuntu* for Windows from the Microsoft store, located at:
<https://www.microsoft.com/en-us/p/ubuntu/9nblggh4msv6?activetab=pivot:overviewtab>
- ② After installing Ubuntu, run it and follow the prompts to create a username and password.

```
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: kcho7166
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

- ③ Install all updates.

1	<code>sudo apt update && apt upgrade -y</code>
---	--

- ④ Once the update is complete, close the current Ubuntu terminal and open a new one.

- ⑤ Install Java SDK 8 as required by Elasticsearch.

1	<code>sudo add-apt-repository ppa:webupd8team/java</code>
2	<code>sudo apt install -y oracle-java8-set-default</code>

- ⑥ Verify Java 8 installation.

1	<code>java -version</code>
---	----------------------------

The output should be similar to below:

```
java version "1.8.0_181"
Java(TM) SE Runtime Environment (build 1.8.0_181-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.181-b13, mixed mode)
```

Installing Elasticsearch (Version used for the tutorial: 6.4.0)

- ① Download *Elasticsearch* for LINUX, located at:
<https://www.elastic.co/downloads/elasticsearch>
- ② Extract Elasticsearch to a desired directory. For ease of use, it is recommended that all Elasticsearch, Logstash and Kibana are extracted to the same directory (In this tutorial, all were extracted to the Downloads folder located in the D drive).

Installing Logstash (Version used for the tutorial: 6.4.0)

- ① Download *Logstash* TAR file format, located at:
<https://www.elastic.co/downloads/logstash>
- ② Extract Logstash to the same directory as Elasticsearch.

Installing Kibana (Version used for the tutorial: 6.4.0)

- ① Download *Kibana* for LINUX 64-BIT, located at:
<https://www.elastic.co/downloads/logstash>
- ② Extract Kibana to the same directory as Elasticsearch.

Installing Notepad++ (for creating Logstash pipeline)

- ① Download and install the latest version of *notepad++* Installer 32-bit x86, located at:
<https://notepad-plus-plus.org/download/>

Running Elasticsearch and Kibana

Running Elasticsearch

- ① Run Ubuntu.
- ② Change the directory to where Elasticsearch files are extracted by entering the command:

```
1 cd /mnt/drive of directory/Elasticsearch directory/Elasticsearch folder/
```

In this tutorial, the files were extracted to the Downloads folder in the D drive. Thus, the command used was:

```
1 cd /mnt/d/Downloads/elasticsearch-6.4.0/
```

(Note: the name of Elasticsearch folder will be different depending on the version downloaded)

Tip: Use **TAB-autocomplete** for faster command input by entering some letters and pressing the TAB button to perform autocompletion e.g. *elastic* + TAB ➔ *elasticsearch-6.4.0*

- ③ Run Elasticsearch (This process may take a while).

```
1 bin/elasticsearch
```

The output should be similar to below:

```
[2018-11-27T01:28:20,651][INFO ][o.e.p.PluginsService] ] [daZnZA6] loaded module [aggs-matrix-stats]
[2018-11-27T01:28:20,652][INFO ][o.e.p.PluginsService] ] [daZnZA6] loaded module [analysis-common]
[2018-11-27T01:28:20,653][INFO ][o.e.p.PluginsService] ] [daZnZA6] loaded module [ingest-common]
[2018-11-27T01:28:20,653][INFO ][o.e.p.PluginsService] ] [daZnZA6] loaded module [lang-expression]
[2018-11-27T01:28:20,654][INFO ][o.e.p.PluginsService] ] [daZnZA6] loaded module [lang-mustache]
[2018-11-27T01:28:20,654][INFO ][o.e.p.PluginsService] ] [daZnZA6] loaded module [lang-painless]
[2018-11-27T01:28:20,655][INFO ][o.e.p.PluginsService] ] [daZnZA6] loaded module [mapper-extras]
[2018-11-27T01:28:20,655][INFO ][o.e.p.PluginsService] ] [daZnZA6] loaded module [parent-join]
[2018-11-27T01:28:20,656][INFO ][o.e.p.PluginsService] ] [daZnZA6] loaded module [percolator]
[2018-11-27T01:28:20,656][INFO ][o.e.p.PluginsService] ] [daZnZA6] loaded module [rank-eval]
[2018-11-27T01:28:20,656][INFO ][o.e.p.PluginsService] ] [daZnZA6] loaded module [reindex]
[2018-11-27T01:28:20,657][INFO ][o.e.p.PluginsService] ] [daZnZA6] loaded module [repository-url]
[2018-11-27T01:28:20,657][INFO ][o.e.p.PluginsService] ] [daZnZA6] loaded module [transport-netty4]
[2018-11-27T01:28:20,657][INFO ][o.e.p.PluginsService] ] [daZnZA6] loaded module [tribe]
[2018-11-27T01:28:20,658][INFO ][o.e.p.PluginsService] ] [daZnZA6] loaded module [x-pack-core]
[2018-11-27T01:28:20,658][INFO ][o.e.p.PluginsService] ] [daZnZA6] loaded module [x-pack-deprecation]
[2018-11-27T01:28:20,658][INFO ][o.e.p.PluginsService] ] [daZnZA6] loaded module [x-pack-graph]
[2018-11-27T01:28:20,658][INFO ][o.e.p.PluginsService] ] [daZnZA6] loaded module [x-pack-logstash]
[2018-11-27T01:28:20,659][INFO ][o.e.p.PluginsService] ] [daZnZA6] loaded module [x-pack-ml]
[2018-11-27T01:28:20,659][INFO ][o.e.p.PluginsService] ] [daZnZA6] loaded module [x-pack-monitoring]
[2018-11-27T01:28:20,659][INFO ][o.e.p.PluginsService] ] [daZnZA6] loaded module [x-pack-rollop]
```

Running Kibana

- ① Open a new Ubuntu terminal and change the directory to where Kibana files are extracted by entering the command:

```
1 cd /mnt/drive of directory/Kibana directory/Kibana folder/
```

In this tutorial, the files were extracted to the Downloads folder in the D drive, thus the command used was:

```
1 cd /mnt/d/DownLoads/kibana-6.4.0-linux-x86_64/
```

(Note: the name of Kibana folder will be different depending on the version downloaded)

- ② Run Kibana (This process may take a while).

```
1 bin/kibana
```

The output should be similar to below:

```
log [01:42:38.040] [info][status][plugin:kibana@6.4.0] Status changed from uninitialized to green - Ready
log [01:42:38.115] [info][status][plugin:elasticsearch@6.4.0] Status changed from uninitialized to yellow - Waiting for Elasticsearch
log [01:42:38.120] [info][status][plugin:xpack_main@6.4.0] Status changed from uninitialized to yellow - Waiting for Elasticsearch
log [01:42:38.124] [info][status][plugin:searchprofiler@6.4.0] Status changed from uninitialized to yellow - Waiting for Elasticsearch
log [01:42:38.129] [info][status][plugin:ml@6.4.0] Status changed from uninitialized to yellow - Waiting for Elasticsearch
log [01:42:38.173] [info][status][plugin:tilemap@6.4.0] Status changed from uninitialized to yellow - Waiting for Elasticsearch
log [01:42:38.176] [info][status][plugin:watcher@6.4.0] Status changed from uninitialized to yellow - Waiting for Elasticsearch
log [01:42:38.187] [info][status][plugin:license_management@6.4.0] Status changed from uninitialized to green - Ready
log [01:42:38.189] [info][status][plugin:index_management@6.4.0] Status changed from uninitialized to yellow - Waiting for Elasticsearch
log [01:42:39.366] [info][status][plugin:timelion@6.4.0] Status changed from uninitialized to green - Ready
log [01:42:39.368] [info][status][plugin:graph@6.4.0] Status changed from uninitialized to yellow - Waiting for Elasticsearch
log [01:42:39.379] [info][status][plugin:monitoring@6.4.0] Status changed from uninitialized to green - Ready
log [01:42:39.381] [warning][security] Generating a random key for xpack.security.encryptionKey. To prevent sessions from being invalidated on restart
log [01:42:39.383] [warning][security] Session cookies will be transmitted over insecure connections. This is not recommended.
log [01:42:39.390] [info][status][plugin:security@6.4.0] Status changed from uninitialized to yellow - Waiting for Elasticsearch
log [01:42:39.408] [info][status][plugin:grokdebugger@6.4.0] Status changed from uninitialized to yellow - Waiting for Elasticsearch
log [01:42:39.414] [info][status][plugin:dashboard_mode@6.4.0] Status changed from uninitialized to green - Ready
log [01:42:39.417] [info][status][plugin:logstash@6.4.0] Status changed from uninitialized to yellow - Waiting for Elasticsearch
log [01:42:39.434] [info][status][plugin:apm@6.4.0] Status changed from uninitialized to green - Ready
log [01:42:39.444] [info][status][plugin:console@6.4.0] Status changed from uninitialized to green - Ready
log [01:42:39.446] [info][status][plugin:console_extensions@6.4.0] Status changed from uninitialized to green - Ready
log [01:42:39.450] [info][status][plugin:notifications@6.4.0] Status changed from uninitialized to green - Ready
log [01:42:39.452] [info][status][plugin:metrics@6.4.0] Status changed from uninitialized to green - Ready
log [01:42:41.278] [warning][reporting] Generating a random key for xpack.reporting.encryptionKey. To prevent pending reports from failing on restart
log [01:42:41.283] [info][status][plugin:reporting@6.4.0] Status changed from uninitialized to yellow - Waiting for Elasticsearch
log [01:42:41.417] [info][listening][server][http] Server running at http://localhost:5601
log [01:42:41.518] [info][status][plugin:elasticsearch@6.4.0] Status changed from yellow to green - Ready
log [01:42:41.596] [info][license][xpack] Imported license information from Elasticsearch for the [data] cluster: mode: basic | status: active
log [01:42:41.600] [info][status][plugin:xpack_main@6.4.0] Status changed from yellow to green - Ready
log [01:42:41.601] [info][status][plugin:searchprofiler@6.4.0] Status changed from yellow to green - Ready
log [01:42:41.601] [info][status][plugin:ml@6.4.0] Status changed from yellow to green - Ready
log [01:42:41.602] [info][status][plugin:tilemap@6.4.0] Status changed from yellow to green - Ready
log [01:42:41.603] [info][status][plugin:watcher@6.4.0] Status changed from yellow to green - Ready
log [01:42:41.603] [info][status][plugin:index_management@6.4.0] Status changed from yellow to green - Ready
log [01:42:41.604] [info][status][plugin:graph@6.4.0] Status changed from yellow to green - Ready
log [01:42:41.605] [info][status][plugin:grokdebugger@6.4.0] Status changed from yellow to green - Ready
log [01:42:41.606] [info][status][plugin:logstash@6.4.0] Status changed from yellow to green - Ready
log [01:42:41.608] [info][status][plugin:reporting@6.4.0] Status changed from yellow to green - Ready
log [01:42:41.609] [info][kibana-monitoring][monitoring-ui] Starting monitoring stats collection
log [01:42:41.612] [info][status][plugin:security@6.4.0] Status changed from yellow to green - Ready
log [01:42:41.701] [info][license][xpack] Imported license information from Elasticsearch for the [monitoring] cluster: mode: basic | status: active
```

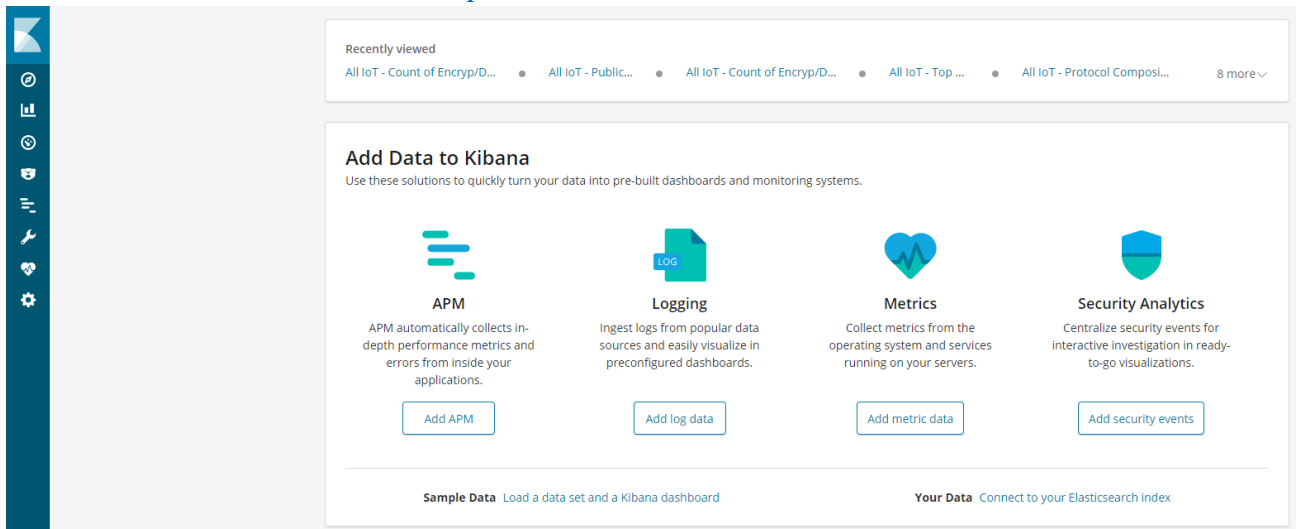
- ③ Kibana/Elasticsearch is now accessible on a web-browser by entering the following as the URL:
<http://localhost:5601>


Elasticsearch mapping

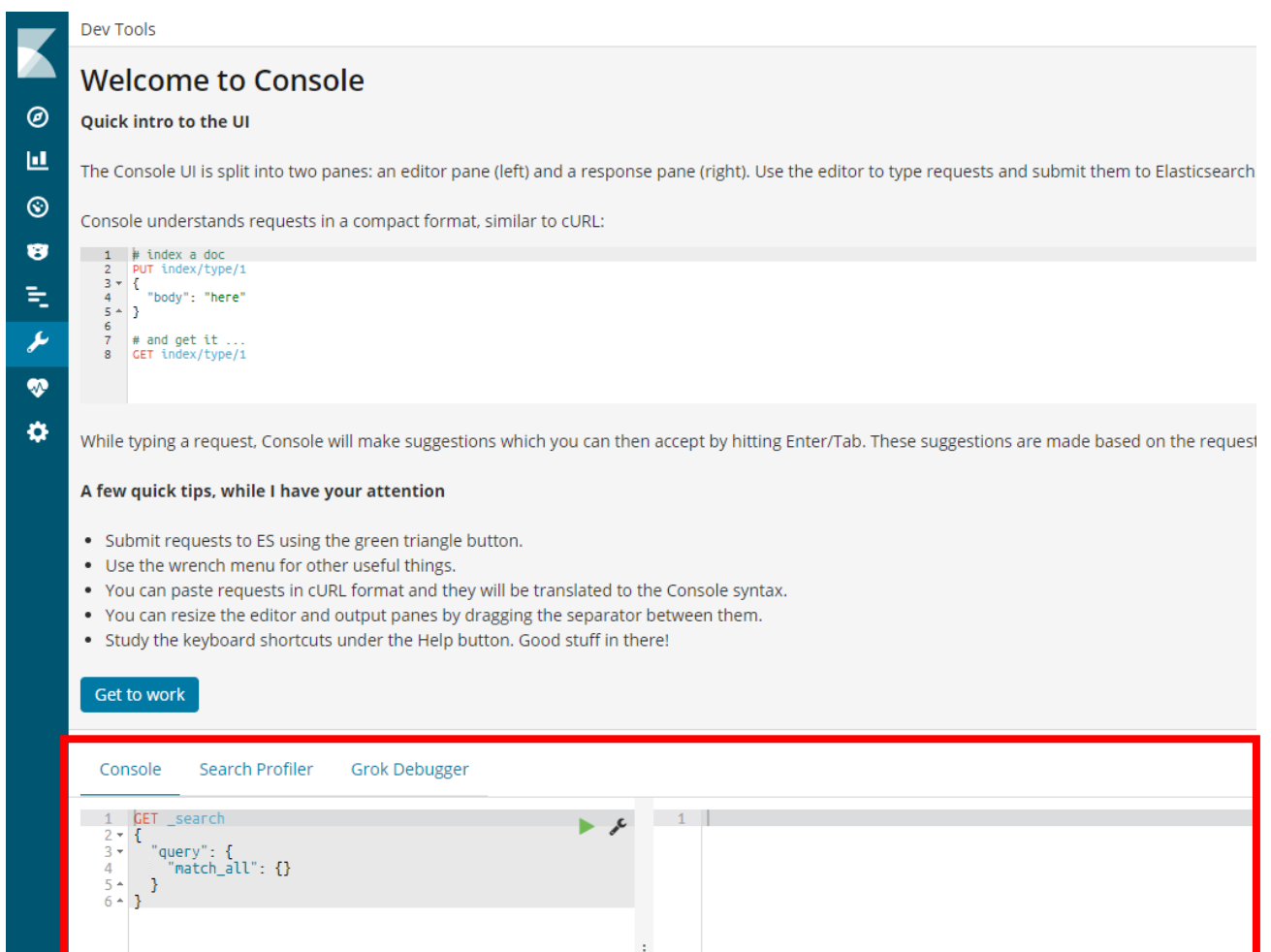
Why Elastic mapping is required

Elasticsearch mapping allows a more sophisticated categorization of the imported data. It offers various types such as timestamp, IP, geo-point and keyword. Furthermore, it enables prevents explosion of indexed fields by explicitly specifying which index field should be created.

- ① On a new internet browser, enter <http://localhost:5601> as the URL address to access Kibana.



- ② Click on the  icon from the tabs on the left-hand side to be directed to 'Dev Tools'.



- ③ In the console (highlighted above with a red box), enter the following code.

```
1 PUT _template/loro
2 {
3   "index_patterns": "packets-*",
4   "mappings": {
5     "pcap_file": {
6       "dynamic": "false",
7       "properties": {
8         "col": {
9           "properties": {
10             "frequency": {
11               "type": "integer"
12             },
13             "datarate": {
14               "type": "text"
15             },
16             "RSSI": {
17               "type": "integer"
18             },
19             "SNR": {
20               "type": "float"
21             },
22             "size": {
23               "type": "integer"
24             },
25             "DevEUI or DevAddr": {
26               "type": "text"
27             },
28             "AppEUI": {
29               "type": "text"
30             },
31             "fcnt": {
32               "type": "integer"
33             },
34             "mtype_desc": {
35               "type": "text"
36             },
37             "activation": {
38               "type": "text"
39             },
40             "network": {
41               "type": "text"
42             },
43             "deveui_manufacturer": {
44               "type": "text"
45             },
46             "appeui_manufacturer": {
47               "type": "text"
48             },
49             "location1": {
50               "type": "text"
51             },
52             "tx_interval": {
53               "type": "float"
54             },
55             "dev_number": {
56               "type": "integer"
57             }
58           }
59         }
60       }
61     }
62   }
63 }
```

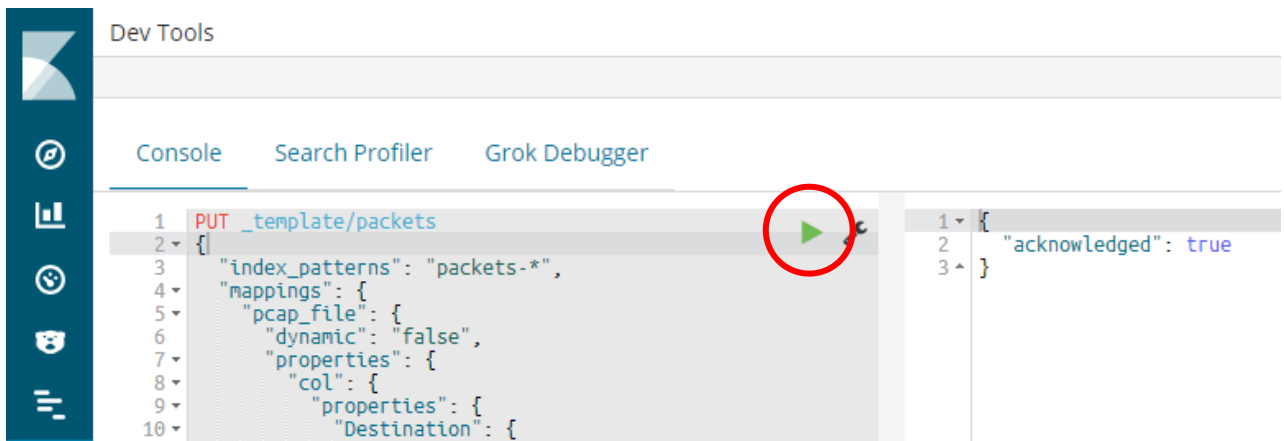
"index_patterns": "packets-*" specifies that this template should be applied to all new indices created that match this pattern

"dynamic": "false" specifies that fields not explicitly specified in the mapping should not be indexed. However, all the non-indexed fields will still be stored in Elasticsearch and you will see them in the results of your searches. However, you will not be able to search or aggregate them.

"properties": { "col": specifies the data type for each of the columns. As can be seen from **"type":**, Elasticsearch mapping supports a wide range of data types such as *ip* and *keyword*.

"geoip_dst": and **"geoip_src":** are the names of columns that will be generated by Logstash for storing the latitude and longitude information of destination and source IP address. Through Elasticsearch mapping, we are able to convert the latitude and longitude coordinates as geo points.

Once all codes have been entered, click on the green arrow located at the top right of the console (highlighted with a **red circle** below). Upon successfully creating a mapping, the following message should appear to the right: “acknowledged”: true



Converting csv file through Logstash

Creating a data processing pipeline for Logstash

Logstash is preferred over ingest pipeline for importing data onto Elasticsearch as it offers more flexibility in data selection and transformation. In order to convert csv files as logs, a data processing pipeline in configuration file format needs to be created.

- ① Open *notepad++* and follow the instructions below to create a pipeline on a new notepad.

```
1 input {
2   file {
3     path =>
4     "/path_to_csv_data/name_of_data_file.csv"
5     start_position => "beginning"
6     sincedb_path => "/dev/null"
7   }
8 }
9 filter {
10  csv {
11    separator => ","
12    source => "message"
13    columns => [ "col.date", "col.time",
14 "col.us count", "col.frequency", "col.RF chain", "col.RX
15 chain", "col.status", "col.bandwidth", "col.datarate",
16 "col.coderate", "col.RSSI", "col.SNR", "col.size",
17 "col.DevEUI or DevAddr", "col.AppEUI", "col.fctrl",
18 "col.fctrl_bin", "col.class_b", "col.fcnt", "col.mhdr",
19 "col.mtype", "col.mtype_desc", "col.MIC",
20 "col.activation", "col.network",
21 "col.deveui_manufacturer", "col.appelui_manufacturer",
22 "col.location", "col.location1", "col.BitRate[bitsps]",
23 "col.Freq_Plan", "col.payload", "col.sec_diff",
24 "col.tx_interval", "col.dev_number" ]
25 }
26 mutate {
27   convert => {
28     "col.frequency" => "integer"
29     "col.bandwidth" => "integer"
30     "col.size" => "integer"
31     "col.RSSI" => "integer"
32     "col.SNR" => "float"
33     "col.fcnt" => "integer"
34     "col.tx_interval" => "float"
35     "col.dev_number" => "integer"
36   }
37 }
38 }
39 output {
40   elasticsearch {
41     hosts => "localhost"
42     index => "lora-filtered-new-ant"
43     document_type => "pcap_file"
44   }
45   stdout {}
46 }
```

'*path*' refers to the directory of the csv file. In this tutorial, the csv file is "alliot_labels.csv" and is located within the Logstash folder.

'*start position*' refers to where the data should be read from.

Here we specify the type of file as csv by indicating the separator as ",". Also, the desired columns of csv are selected with the "col." prefix.

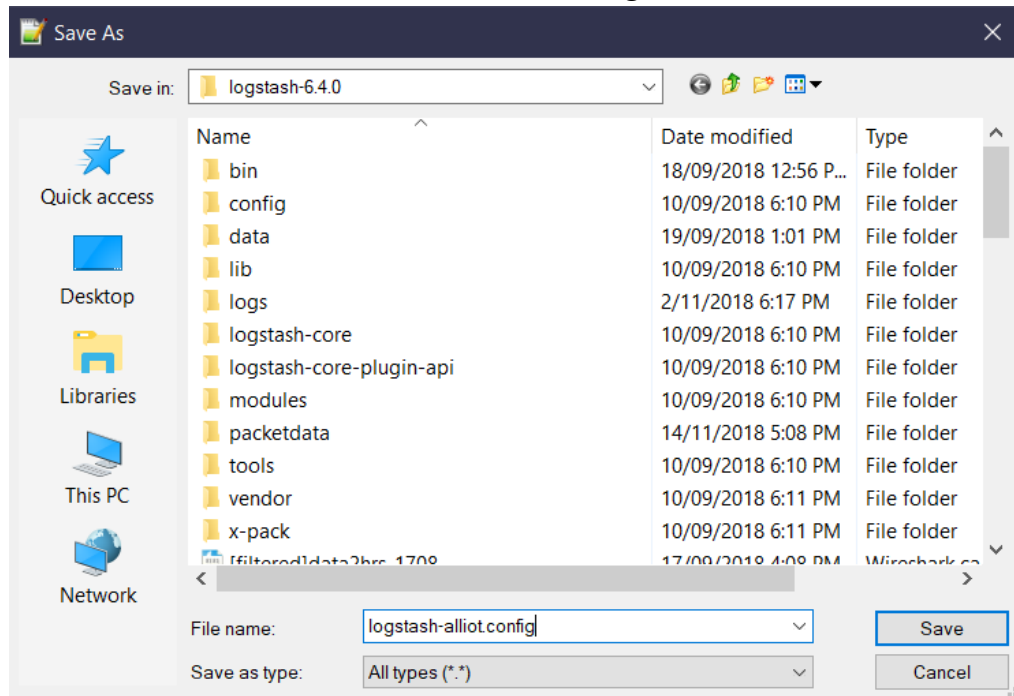
We obtain the latitude and longitude coordinates for source IP and destination IP separately by looking up the IP address in the GeoLite2-City database. The '*geoip*' filter allows the obtained latitude and longitude to be imported to Elasticsearch.

'*source*' refers to the column containing the IP address, while '*target*' is the new column name for storing the coordinates.

By default, Logstash sets all data types as strings. As we need numerical values to allow aggregation for visualisation in Kibana, we convert columns with numerical values to their corresponding types. Logstash only offers basic numeric types such as integer and float.

Once all the desired data are correctly converted, we export the logs to Elasticsearch. We assign the '*packets-*' prefix to the index name deliberately, to comply with Elasticsearch mapping that we have created. Logs with same index names will be stored together.

- ② Save it as a **.config** file in the same directory as Logstash, with a user-defined name e.g. *packetsdata*. This can be achieved by setting data type as **All types (*.*)**, and adding the **.config** suffix after the user-defined name. In this tutorial, the file was saved as '*logstash-alliot*'.



Converting and uploading the csv data to Elasticsearch using Logstash

- ③ Open a new Ubuntu terminal and change the terminal directory to the Logstash folder.

```
1 cd /mnt/drive of directory/Logstash directory/Logstash folder/
```

In this tutorial, the Logstash was extracted to the Downloads folder in the D drive, thus the command used was:

```
1 cd /mnt/d/Downloads/Logstash-6.4.0/
```

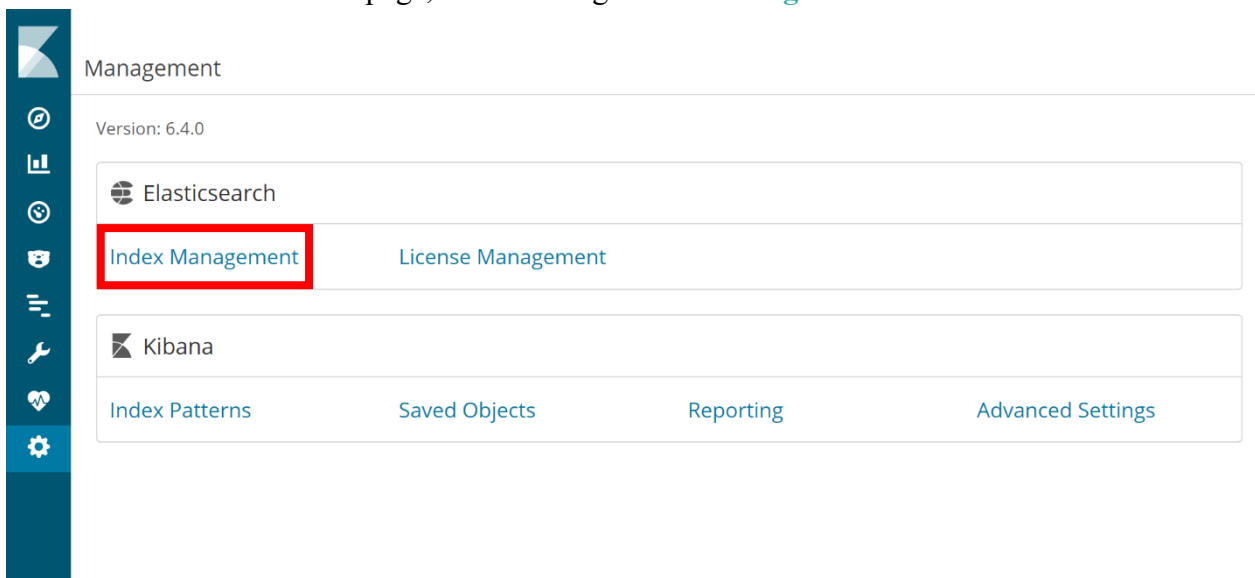
(Note: the name of Logstash folder will be different depending on the version downloaded)

```
kcho7166@KWON-Dell: /mnt/d/Downloads/logstash-6.4.0
kcho7166@KWON-Dell:/$ cd /mnt/d/Downloads/logstash-6.4.0/
kcho7166@KWON-Dell: /mnt/d/Downloads/logstash-6.4.0$
```

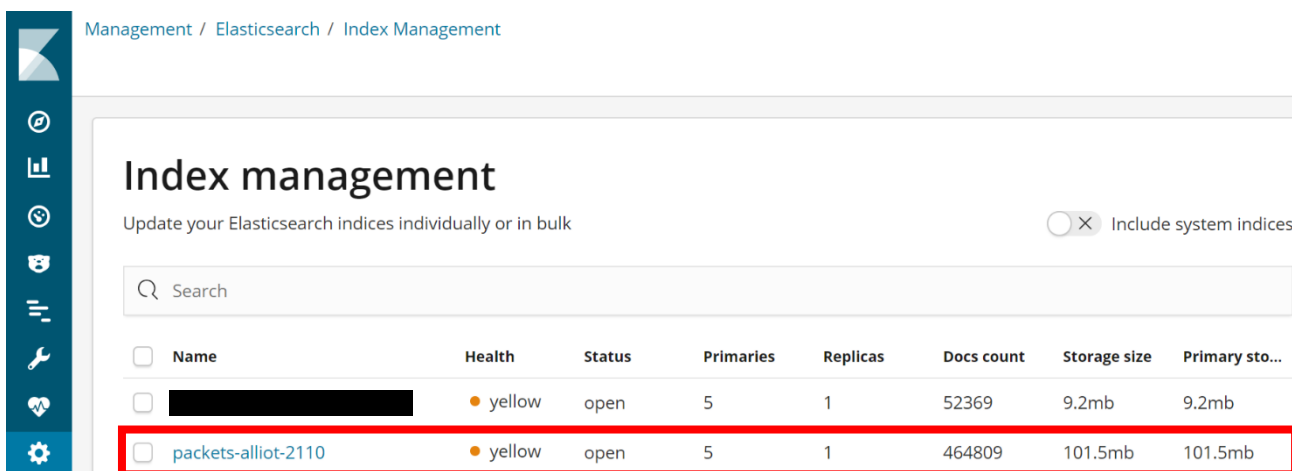
- ④ Run Logstash while specifying the data processing pipeline configuration file after '**-f**'. This will begin importing data from Logstash to Elasticsearch and may take a while depending on the size of the imported data. The example below illustrates how the configuration file created in the tutorial was specified after '**-f**'.

```
1 bin/logstash -f logstash-alliot.config
```

- ④ You can see that a new index has been created in Elasticsearch by clicking the  icon on the left-hand side tab of the Kibana page, then clicking **Index Management** as shown below.

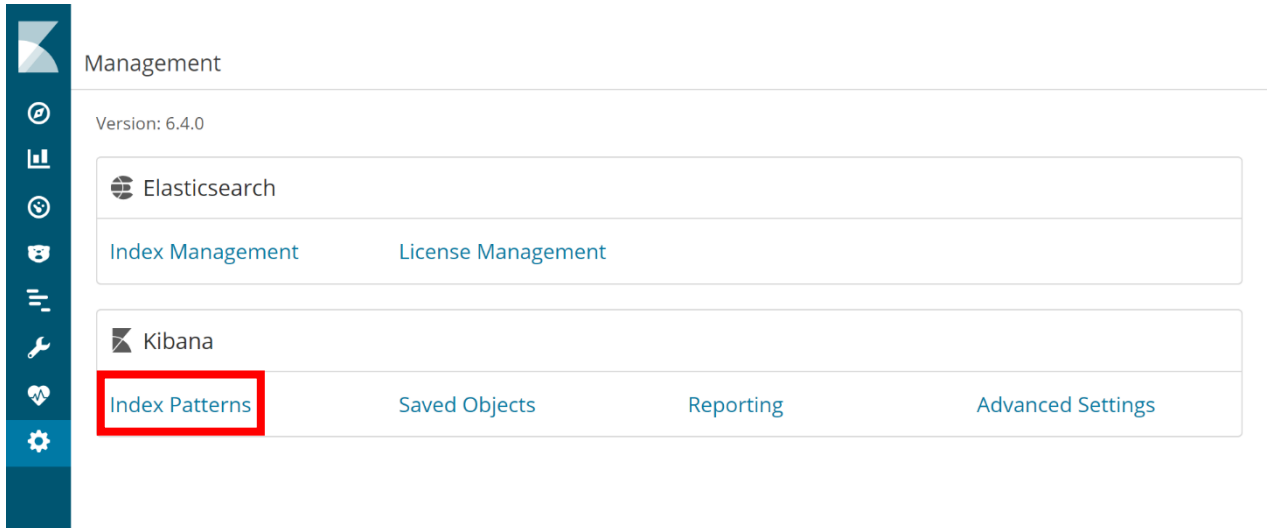


In this tutorial, the designated index name was *'packets-alliot-2110'*, which has been successfully created on Elasticsearch as shown below:

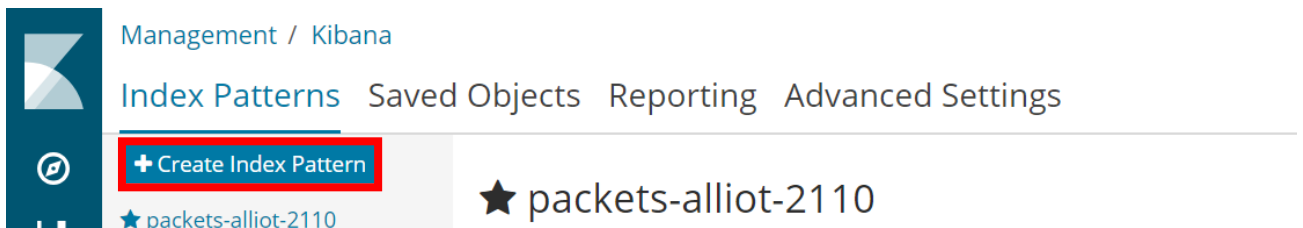




- ⑤ We now need to create an index for Kibana to allow the creation of visualisations by clicking the icon on the left-hand side tab of the Kibana page, then clicking **Index Patterns** as shown below.



- ⑥ Click on the **+ Create Index Pattern** button.



- ⑦ In the input box below **Index pattern**, enter the index name that corresponds to the recently created Elasticsearch index which we saw in step 4 then click **Next step**. Afterwards, click the **Create index pattern** button to complete the process. In this tutorial, *'packets-alliot-2110'* was the index pattern.

Management / Kibana

Index Patterns Saved Objects Reporting Advanced Settings

★ packets-alliot-2110

airbeam

echoamamusic

echospotify

googlehomemusic

huawei-baseline

packets-airbeam

packets-applewatch-podcast

packets-echo-amazonmusic

packets-echo-spotify

packets-huawei-baseline

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

packets-alliot-2110

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.

packets-alliot-2110

Rows per page: 10

- ⑧ With the Kibana index pattern successfully created, you will now be directed to a page showing the fields in the index. Change *Rows per page* at the bottom of the page from 10 to 50 to display all the fields.

Rows per page: 50

- ⑨ In order to create correct units for our fields of interest, we need to specify the format for some fields. For this, place the cursor on the *c.Size* field and click on the icon.

- ⑩ Under **Format**, select *Bytes* as the desired format and click **Save field**.

Edit col.Length

Type

number

Format (Default: Number)

- Default -

- Default -

Url

Bytes

Duration

Number

Percentage

String

Color

Boolean

Save field Cancel


- ⑪ Change the format for *c.Info*, *c.IoT*, *c.No* and *c.Time* to correspond to the format shown below. For *c.Time*, also set **Output format** as *Seconds*.

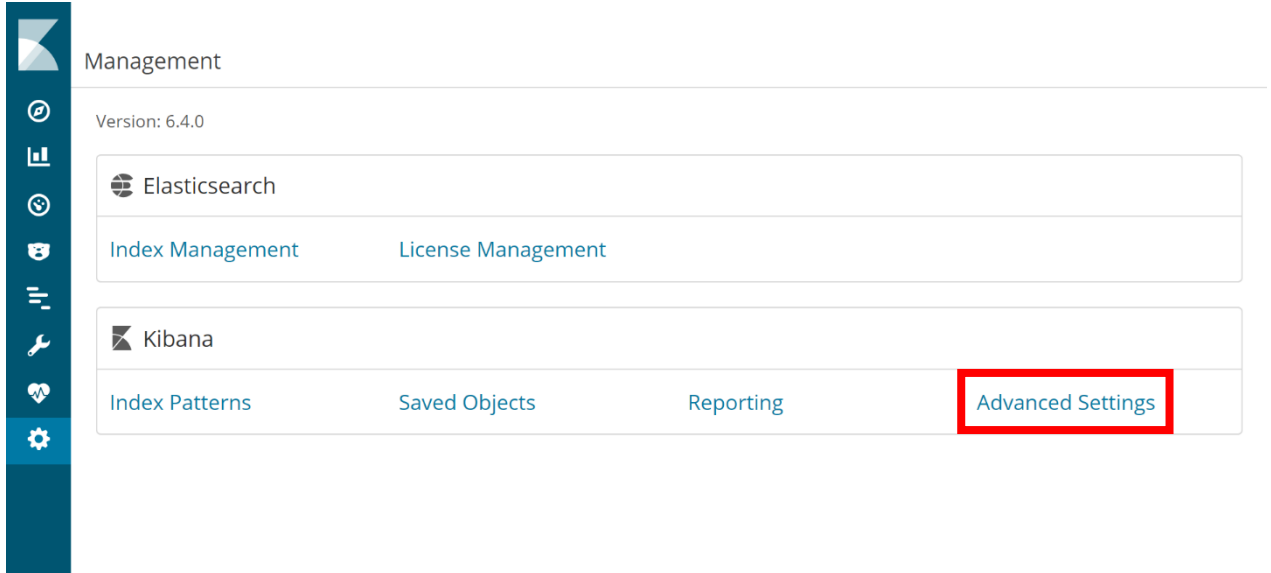
c.Info	string	String	•	
c.IoT	string	String	•	•
c.Length	number	Bytes	•	•
c.No	number	Number	•	•
c.Protocol	string		•	•
c.Source	ip		•	•
c.Time	number	Duration	•	•
g.location	geo_point		•	•
g.location	geo_point		•	•

Rows per page: 50 ▾

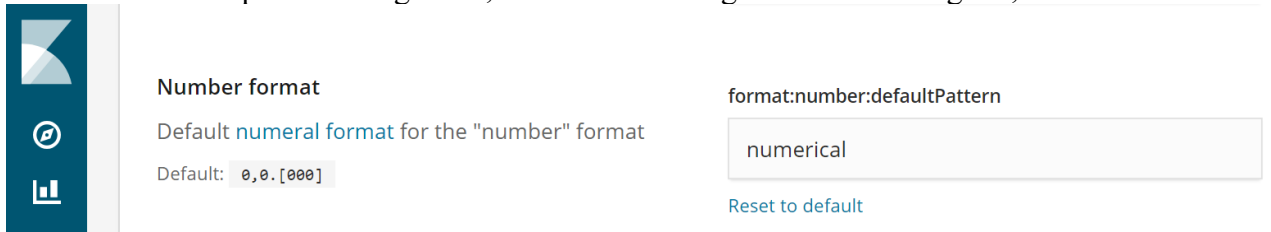
Kibana settings

Prior to creating visualisations on Kibana, we need to make some changes in Kibana's advanced settings to change the numerical format for simplifying long numerical values, as well as to maximise the tile map precision of the geographical map of IP addresses.

- ① Click the  icon on the left-hand side tab of the Kibana page, then click **Advanced Settings** as shown below.

















- ② Search for 'Number format' via Ctrl + F, and change the format from the default to *numerical*. This allows letters to replace trailing zeros, therefore allowing shorter labels e.g. 10,000 → 10k.



Creating visualisations through Kibana

Kibana offers various types of visualisations. For this tutorial, we will explore how pie chart, vertical bar chart, metric, coordinate map, and controls can be created to illustrate information regarding traffic patterns. Visualisation offered in Kibana 6.4.0 can be categorized into five different types, as described below ("A Kibana Tutorial - Part 2: Creating Visualizations | Logz.io", 2018).

1. Basic Charts		
	Area	For visualising time series data and for splitting lines on fields.
	Heat Map	For showing statistical outliers and dare often sued for latency values.
	Horizontal Bar	Good for showing relationships between two fields.
	Line	Simple way to show time series. Good for splitting lines to show anomalies.
	Pie	Useful for displaying parts of a whole.
	Vertical Bar	Great for time series data and for splitting lines across fields.
2. Data		
	Data Table	Best way to split across multiple fields in a custom way.
	Gauge	A way to show the status of a specific metric using thresholds you define.
	Goal	Similar to a Gauge, useful for monitoring a specific metric defined as a goal.
	Metric	Useful visualisation for displaying a calculation as a single number <i>e.g. number of encrypted traffic</i> .
3. Maps		
	Coordinate Map	Help add a geographical dimension to IP-based logs.
	Region Map	
4. Time Series		
	Timelion	Allows you to create more advanced queries based on time series data <i>e.g. percentage of 500 errors over time</i> .
	Visual Builder	

5. Other



Controls

Allows you to create selectors or sliders to filter the data being visualised.



Markdown

For adding customized text or image-based visualisation to the dashboard using markdown syntax *e.g. company logo or a description of a dashboard.*



Tag Cloud

Helps display groups of words sized by their importance.





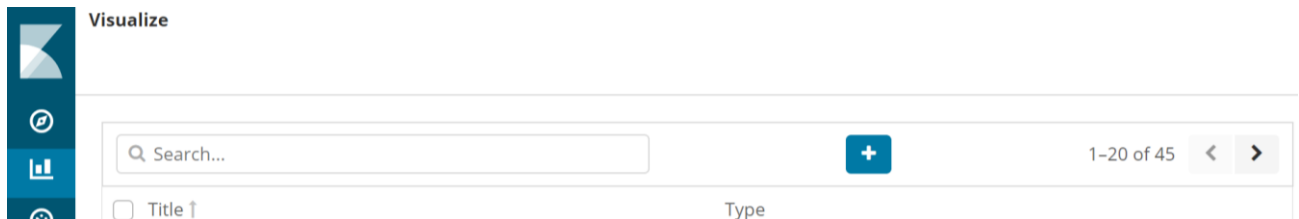
Vega

Allows you to add custom visualisations based on Vega and VegaLite.


Tutorials on creating different visualizations are provided below, using a different dataset. The dataset used is for a pcap file exported from Wireshark. Although the LoRadar dataset has different field names, the examples below can be used to understand how figures can be created in Kibana.

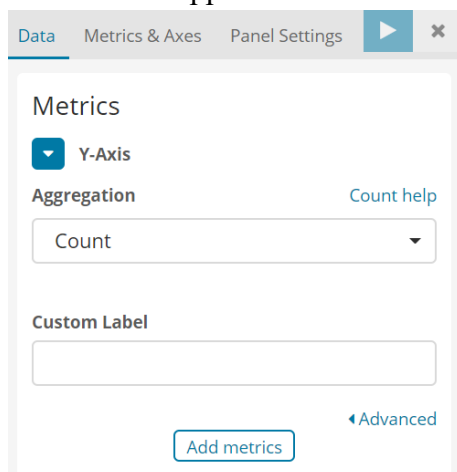
Vertical bar chart (For visualising packet count or packet length across time)

- ① Open Visualize by clicking on the  button on the left-hand side tab of the Kibana page. Then click  to create a new visualisation.

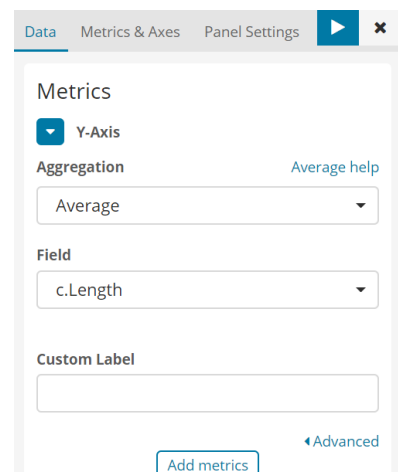


- ② Click *Vertical Bar* and select the index that should be used to create the visualisation. In this tutorial, the index of interest was *'packets-alliot-2110'*.

- ③ For visualisation of packet count, select *Count* by clicking  under **Metrics** and selecting *Count* as the **Aggregation** option. For visualisation of average packet length, select *Average* as the **Aggregation** option and *c.Length* as the **Field** option. Y-Axis label can be specified in **Custom Label** e.g. Average Packet Length (Bytes). A more detailed description of each aggregation options for Metrics can be seen in Appendix A.



(a) Packet count visualisation

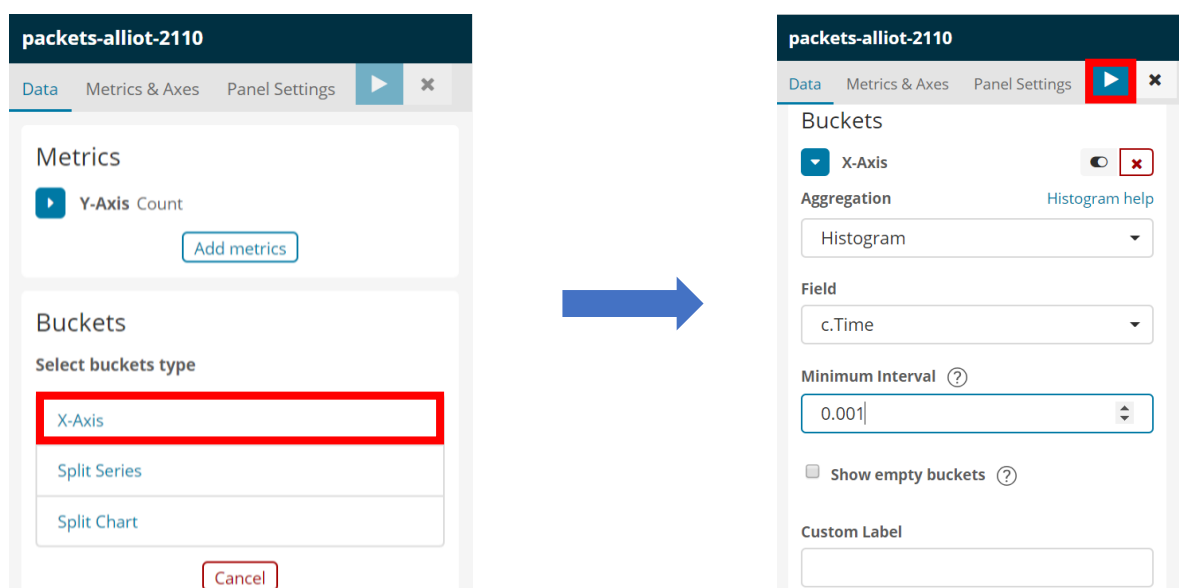


(b) Average packet length visualisation

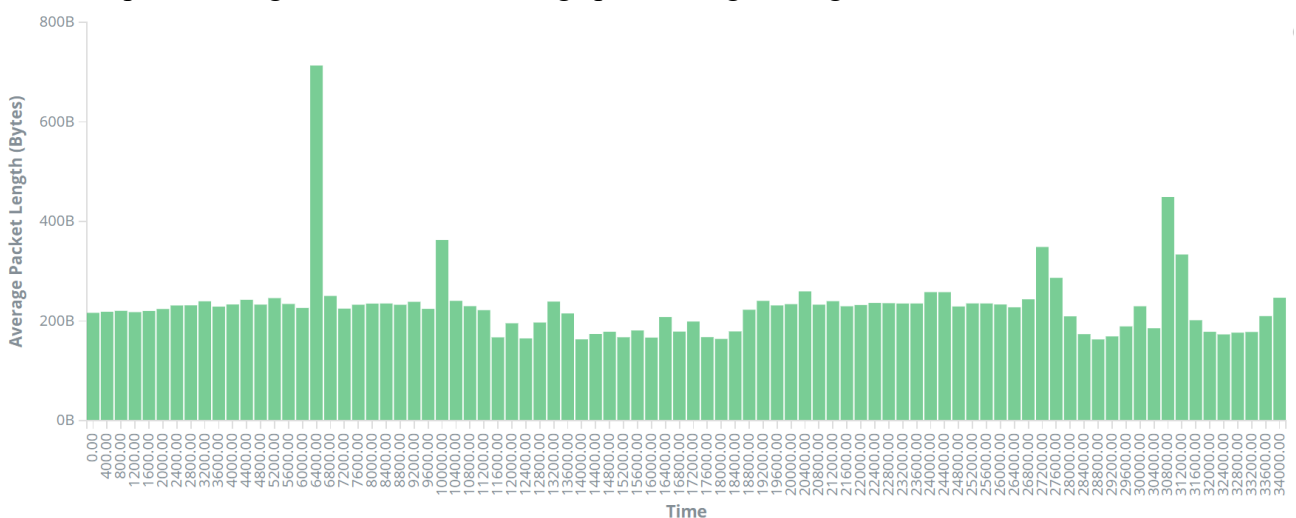
- ④ Select **X-Axis** under **Buckets**, and select *Histogram* option for **Aggregation** and *c.Time* option for **Field**. Enter 0.001 as the **Minimum Interval** to specify the bin size for the histogram that displays information for up to milliseconds. Label for the X-Axis can be specified in **Custom Label**.

Afterwards, click  to create the histogram.

(Note: The 0.001 translates to milliseconds because the default unit for time from Wireshark is in seconds. Kibana automatically scales the interval when the number of buckets exceed the maximum number of buckets specified in Kibana's advanced settings, *histogram:maxBars*. However, when the user interacts with the vertical bar chart to zoom in on a particular time range, our vertical bar chart will support up to millisecond units.)



An example of histogram created for average packet length using the tutorial's index:

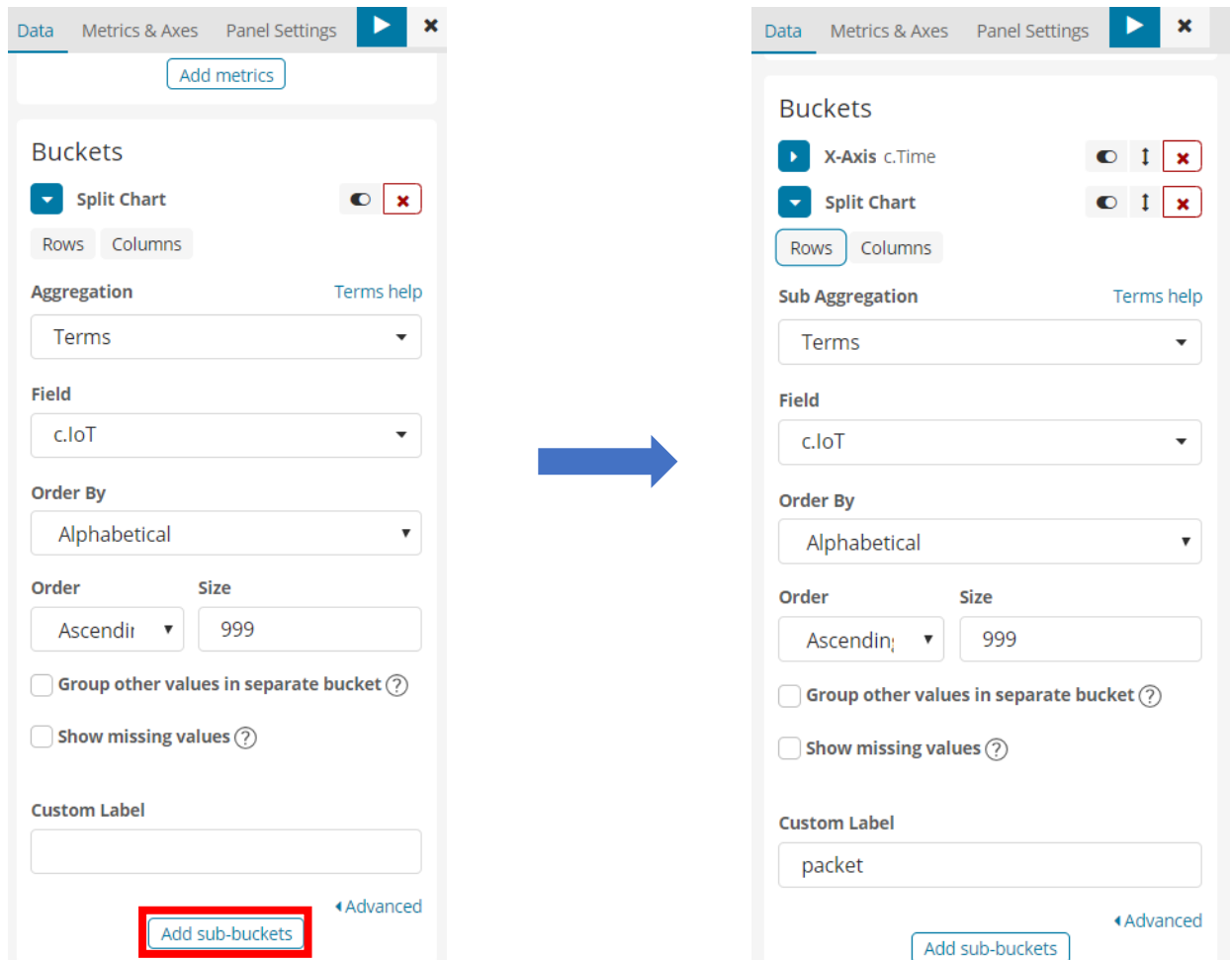


- ⑤ Save the visualisation by clicking *Save* on the top of the page and providing a name for it.

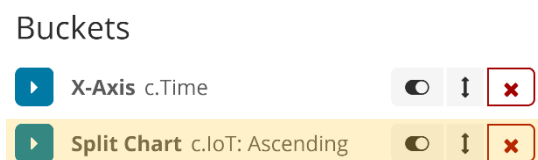


Extension: Splitting the overall traffic pattern into IoT and non-IoT device

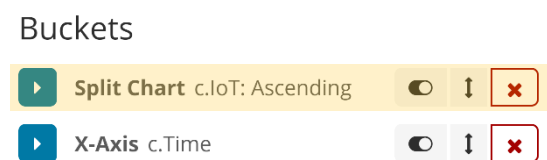
- ⑥ Under **Buckets**, click [Add sub-buckets](#) followed by **Split Chart**. Select **Rows** to create a vertically aligned bar chart that allows the comparison of traffic patterns across time. Select **Terms** for **Sub Aggregation** and **c.IoT** for **Field** to allow partitioning by the terms in the IoT column. Order the histograms in ascending alphabetical order by selecting **Alphabetical** in **Order By** and **Ascending** in **Order**. Increase the value of **Size** to 999 (or some large number) to allow all IoT terms to be used for the visualisation.



- ⑦ Even with the Alphabetical ordering specified, the visualisation at present will produce incorrect ordering due to the **Split Chart** being executed after **X-Axis** (You can confirm this by clicking on the button next to **Panel Settings** and observing the y-axis label for each histogram). To correct the ordering, we use the button next to **Split Chart** to drag **Split Chart** above **X-Axis** to ensure that alphabetical ordering is applied first. Click to produce the histogram with these changes applied.



(a) Before dragging



(b) After dragging

- ⑧ Save the visualisation as a *new visualisation* by clicking *Save* on the top of the page and checking *Save as a new visualisation* option after providing a name for the new visualisation.

Visualize / New Visualization Save Share Inspect Refresh Auto-refresh

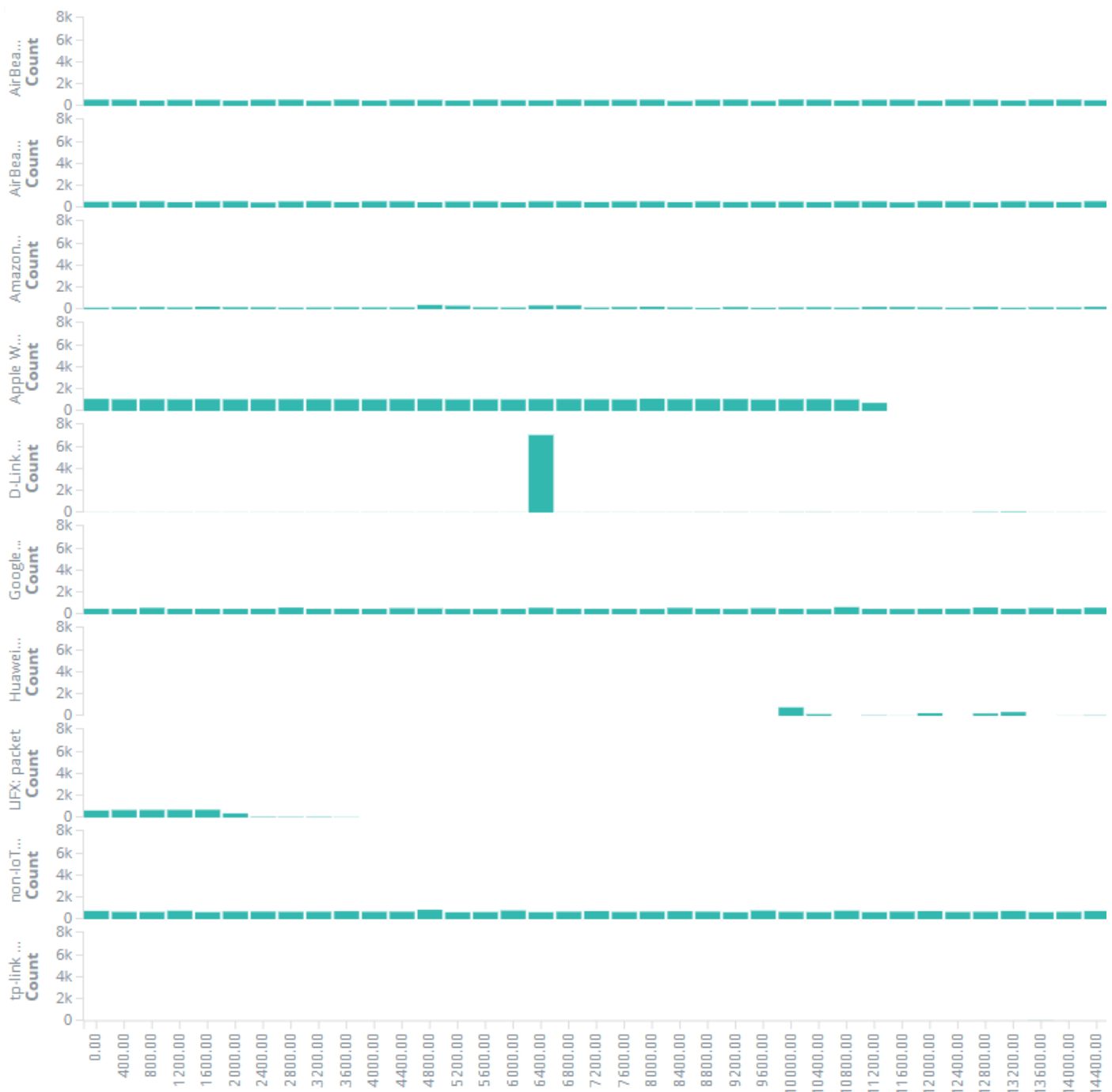
Save Visualization

New Visualization


☒ Save as a new visualization

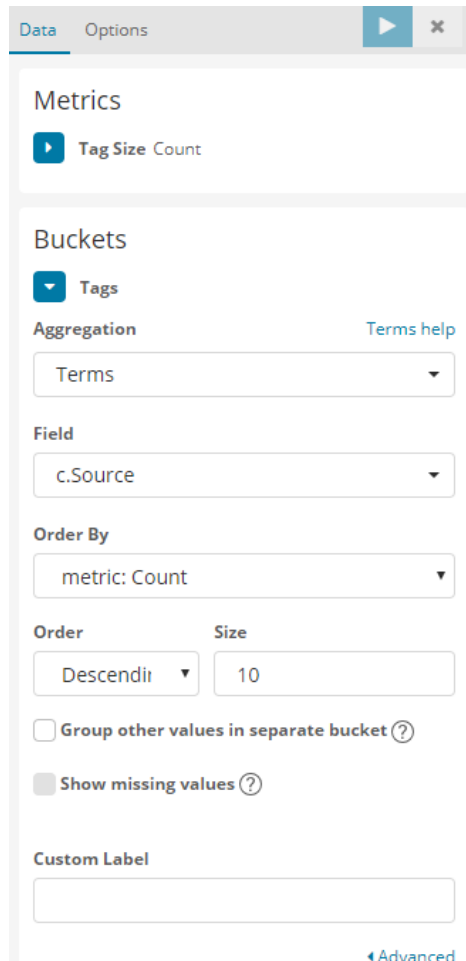
Save

An example of a histogram of packet count with split chart can be seen below, where a histogram is produced for each of the IoT devices in the dataset, as well as non-IoT device.



Tag cloud (For visualising top 10 Source and Destination IP)

- ① Repeat steps 1 and 2 of Vertical Bar Chart, but selecting *Tag Cloud* instead of *Vertical Bar*.
- ② We set *Count* as the **Aggregation** option under **Metrics** as we wish to rank IP address by packet count. Under **Buckets**, click **Tags** and select *Terms* as the **Aggregation** option. For top 10 Source IP, select *c.Source* as the **Field** (*c.Destination* for top 10 Destination IP) and select *Descending* for **Order** and 10 for **Size**. Apply these changes by clicking the  button.



The screenshot shows the 'Options' tab of a visualization configuration window. The 'Metrics' section has 'Tag Size Count' selected. The 'Buckets' section has 'Tags' selected. Under 'Aggregation', 'Terms' is selected. The 'Field' is set to 'c.Source'. 'Order By' is set to 'metric: Count'. 'Order' is set to 'Descending' and 'Size' is set to '10'. There are checkboxes for 'Group other values in separate bucket' and 'Show missing values', both of which are unchecked. A 'Custom Label' field is empty. An 'Advanced' link is at the bottom right.

- ③ In the **Options** tab, uncheck **Show Label** to remove the unwanted label at the bottom of the Tag cloud. Text scale, orientations and font size can be changed if desired (High packet count = large text size). Save the visualisation.

An example of Tag cloud showing top 10 Source and Destination IP addresses:

192.168.137.224
fe80::917d:6673:3342:a95f
192.168.137.204
192.168.137.1
66.228.47.244
17.132.149.5
192.168.137.114
192.168.137.174

(a) Source IP

192.168.137.1
192.168.137.114
ff02::fb 192.168.137.204
66.228.47.244
17.132.149.5
192.168.137.174
192.168.137.224

(b) Destination IP

Pie chart (For visualising top 3 protocols used by each top 10 Source or Destination IP)

- ① Repeat steps 1 and 2 of Vertical Bar Chart, but selecting *Pie* instead of *Vertical Bar*.
- ② We set *Count* as the **Aggregation** option under **Metrics** as we wish to rank protocols by packet count. Under **Buckets**, click **Split Slices** and select *Terms* as the **Aggregation** option. For top 10 Source IP, select *c.Source* as the **Field** (*c.Destination* for top 10 Destination IP) and select *Descending* for **Order** and 10 for **Size**.

Metrics

Slice Size Count

Buckets

Split Slices ☒ **Terms help**

Aggregation Terms

Field c.Source

Order By metric: Count

Order Descending **Size** 10

☐ Group other values in separate bucket ?

☐ Show missing values ?

Custom Label

[Add sub-buckets](#) **Advanced**

- ③ Click [Add sub-buckets](#) at the bottom of **Buckets** and click **Split Slices**, again selecting *Terms* as the **Aggregation** option. Choose *c.Protocol* as the **Field** and select *Descending* for **Order** and 3 for **Size**.


Split Slices ☒ **Terms help**

Sub Aggregation Terms

Field c.Protocol

Order By metric: Count

Order Descend **Size** 3

- ④ In the **Options** tab, uncheck **Show Label** to remove the unwanted label at the bottom of the Pie chart. Also uncheck **Donut** to make the inner Pie chart a filled circle. Apply these changes by clicking the  button. Save the visualisation.

Data
Options
▶
✕

Pie Settings

Donut ☐

Legend Position right ▼

Show Tooltip ☒

Labels Settings

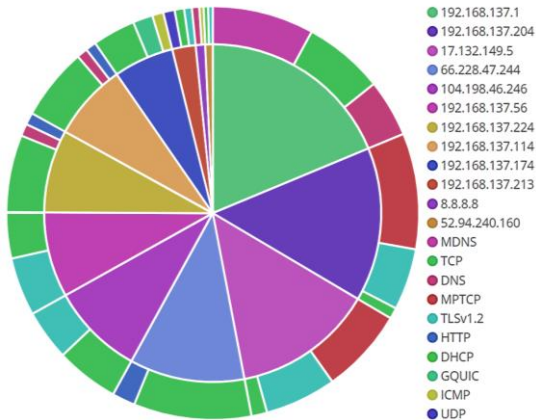
Show Labels ☐

Show Top Level Only ☒

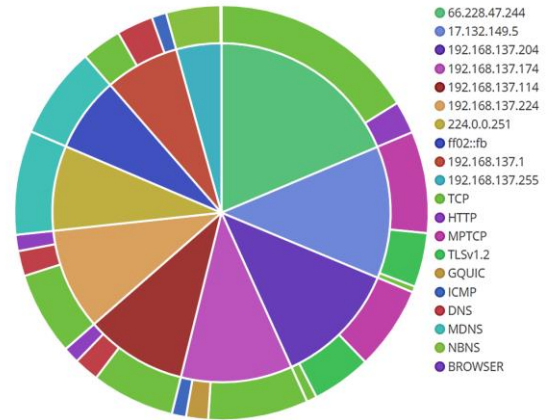
Show Values ☒

Truncate 100

An example of Pie chart with top 10 Source or Destination IP addresses as the inner layer and top 3 protocols of each IP address as the outer layer:



(a) Source IP




(b) Destination IP


Metric (For visualising the number of encrypted and decrypted traffic for Source and Destination IP)

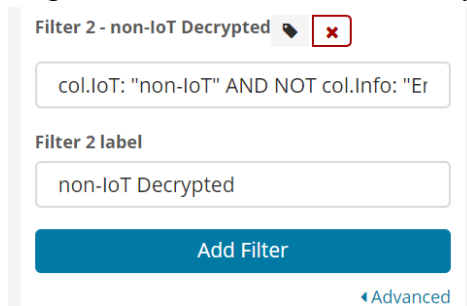
- ① Repeat steps 1 and 2 of Vertical Bar Chart, but selecting *Metric* instead of *Vertical Bar*.
- ② We define traffic as unique IP addresses, with encrypted traffic being the IP address with the term 'encrypted' present in the Info column of Wireshark's captured data. Hence, we set *Unique Count* as the **Metric** option and *c.Source* as the **Field** option under **Metrics** to count the number of unique Source IP addresses that are encrypted or decrypted (select *c.Destination* for unique Destination IP).


The screenshot shows the 'Metrics' configuration panel. At the top, there are tabs for 'Data' and 'Options', with 'Options' being the active tab. Below the tabs, there is a 'Metrics' section with a dropdown menu set to 'Metric'. Underneath, the 'Aggregation' dropdown is set to 'Unique Count', with a 'Unique Count help' link to its right. The 'Field' dropdown is set to 'c.Source'. There is an empty 'Custom Label' input field. At the bottom right, there is an 'Advanced' link with a left-pointing arrow. At the bottom center, there is an 'Add metrics' button.

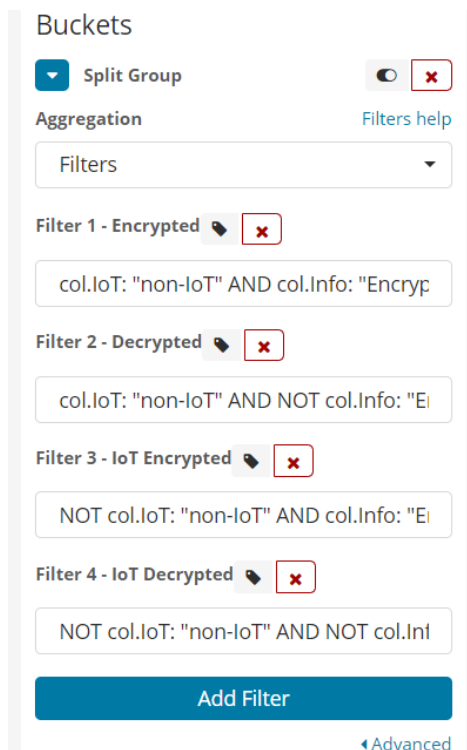
- ③ Under **Buckets**, click **Split Group** and select *Filters* as the **Aggregation** option. In the input box below **Filter 1**, enter *col.IoT: "non-IoT" AND col.Info: "Encrypted"* to create a metric for encrypted traffic associated with non-IoT devices. Assign the proper label (e.g. non-IoT Encrypted) for ease of identification by clicking on  next to **Filter 1** and typing in the desired label.

The screenshot shows the 'Buckets' configuration panel. At the top, there is a 'Split Group' dropdown menu. To its right is a toggle switch and a red 'X' icon. Below this, the 'Aggregation' dropdown is set to 'Filters', with a 'Filters help' link to its right. Underneath, there is a section for 'Filter 1 - non-IoT Encrypted'. To the left of this text is a tag icon and a red 'X' icon. Below this text is an input field containing the filter expression: 'col.IoT: "non-IoT" AND col.Info: "Encryp"'. Below the input field is a 'Filter 1 label' section with an input field containing the label 'non-IoT Encrypted'. At the bottom center, there is an 'Add Filter' button. At the bottom right, there is an 'Advanced' link with a left-pointing arrow.

- ④ To create a metric for decrypted traffic, click **Add Filter** at the bottom of **Buckets** and simply copy the input of Filter 1 while adding *NOT* after *AND*. Assign the proper label (e.g. non-IoT Decrypted) for ease of identification by clicking on  next to **Filter 2** and typing in the desired label.



- ⑤ Repeats steps 3 and 4 above to create metric for IoT devices by adding *NOT* in front of *col.IoT*. Apply these changes by clicking the  button. Save the visualisation.



An example of Metric showing the number of encrypted and decrypted traffic for Source and Destination IP:

0	7
Encrypted - Traffic	Decrypted - Traffic
61	102
IoT Encrypted - Traffic	IoT Decrypted - Traffic


(a) Source IP

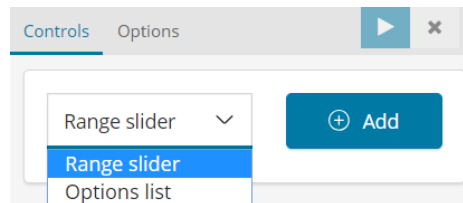
0	14
Encrypted - Traffic	Decrypted - Traffic
61	117
IoT Encrypted - Traffic	IoT Decrypted - Traffic

(b) Destination IP

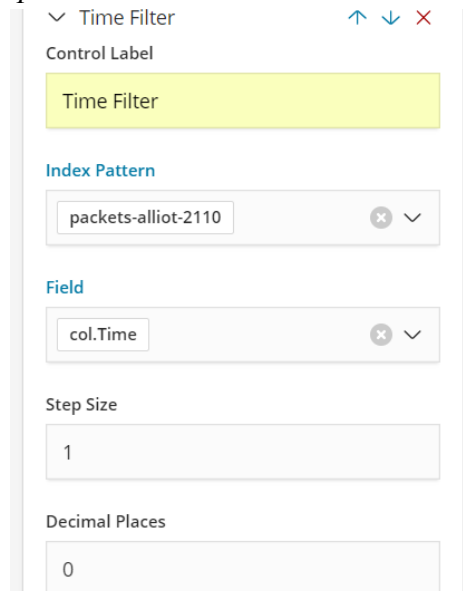
Controls (For creating filters for time, protocol and IoT device)

- ① Repeat steps 1 and 2 of Vertical Bar Chart, but selecting *Controls* instead of *Vertical Bar*.

- ② To create a filter for time, select *Range slider* under **Controls** and click  .

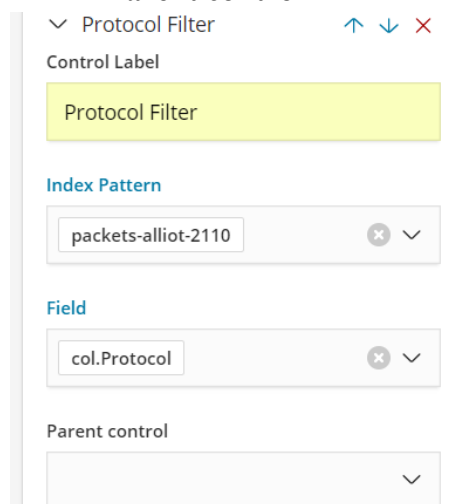



- ③ Type in *Time Filter* for **Control Label** and select the desired index as the **Index Pattern**. The index pattern used for this tutorial was *packets-alliot-2110*. Choose *col.Time* option for **Field**.

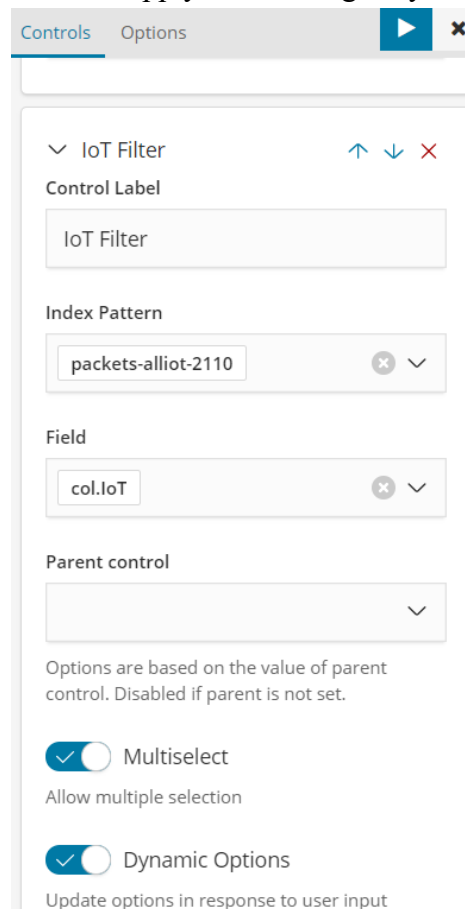


- ④ Create a new filter for protocol by selecting *Options list* under **Controls** and click  .

- ⑤ Type in *Protocol Filter* for **Control Label** and select the same index pattern as step 3. Choose *col.Protocol* option for **Field** and leave **Parent control** blank.




- ⑥ Create a new filter for IoT device by repeating steps 4 and 5 above, but setting *IoT Filter* as the **Control Label** and *col.IoT* as the **Field**. Apply these changes by clicking the  button.



The screenshot shows the 'Controls' tab of a Kibana filter configuration panel. At the top, there are tabs for 'Controls' and 'Options', with a play button and a close button to the right. The main section is titled 'IoT Filter' with a dropdown arrow, up/down arrows, and a close button. Below this, there are three input fields: 'Control Label' with the value 'IoT Filter', 'Index Pattern' with the value 'packets-alliot-2110', and 'Field' with the value 'col.IoT'. Each of these fields has a dropdown arrow and a close button. Below these fields is a 'Parent control' dropdown menu. At the bottom, there are two toggle switches: 'Multiselect' (checked) and 'Dynamic Options' (checked). Below the 'Multiselect' toggle is the text 'Allow multiple selection'. Below the 'Dynamic Options' toggle is the text 'Update options in response to user input'.


- ⑦ In **Options**, turn *Update Kibana filters on each change* on to automatically apply the filter without having to click *Apply changes* (this also removes such buttons, making the visualisation more compact). Save the visualisation.


An example of Controls for time, protocol and IoT device:



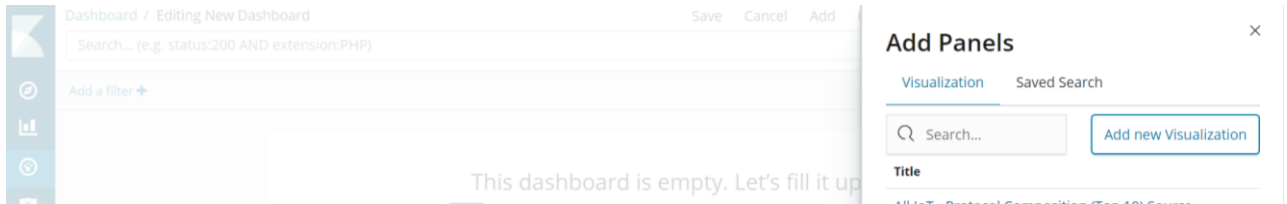
The screenshot shows three filter controls side-by-side. The first is 'Time Filter' with a range slider from 0 to 34170. The second is 'Protocol Filter' with a dropdown menu showing 'Select...'. The third is 'IoT Device Filter' with a dropdown menu showing 'Select...'.

Creating dashboard through Kibana

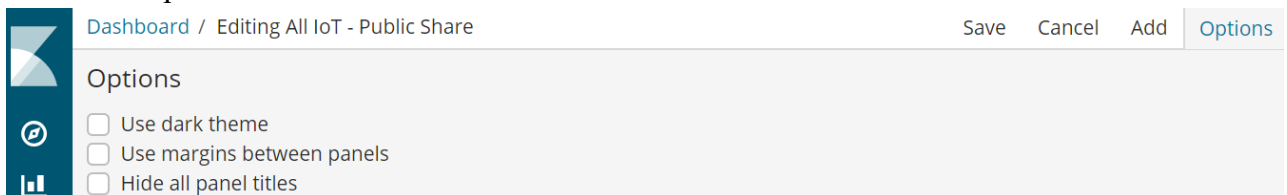
- ① Open Dashboard by clicking on the  button on the left-hand side tab of the Kibana page.


- ② Create a new dashboard by clicking  on the top right.


- ③ Click the *Add* button in the top menu bar and click on the desired visualisation under **Add Panels** to add a visualization to the dashboard (NOT the *Add new Visualisation* button).

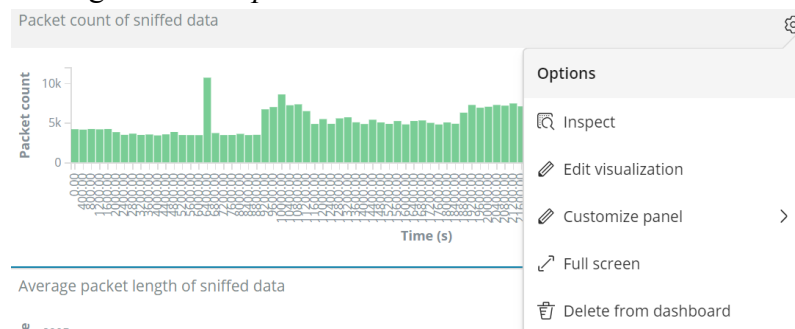


- ④ The *Options* button in the top menu allows users to use dark theme, remove margins between panels, and hide all panel titles.



- ⑤ Each visualisation on the dashboard can be resized by clicking on the  button located at the bottom-right of the visualisation. Positions of visualisations can also be changed by drawing the visualisation around the dashboard.

- ⑥ Title can be assigned to visualisations by clicking the  button located at the top-right of the visualisation and selecting *Customize panel*.



- ⑦ Save the dashboard by clicking *Save* on the top menu bar. Saved dashboards can be edited anytime by selecting *Edit* on the top menu bar. Full screen view of the dashboard is also possible through the top menu bar.

References

- A Kibana Tutorial - Part 2: Creating Visualizations | Logz.io. (2018). Retrieved from <https://logz.io/blog/kibana-tutorial-2/>
- Arnold, S. E. (2014). Rumble in the search jungle: The ELK invasion. *Online Searcher*, 38(4), 16-20. Retrieved from <http://ezproxy.library.usyd.edu.au/login?url=https://search-proquest-com.ezproxy1.library.usyd.edu.au/docview/1555641812?accountid=14757>
- Bullock, J., & Parker, J. T. (2017). Chapter 1: Introducing Wireshark. In *Wireshark for security professionals: Using Wireshark and the Metasploit framework*(pp. 1-18). Indianapolis, IN: Wiley.
- How much of the IP Address space do you cover? | MaxMind Support Center. (2018). Retrieved from <https://support.maxmind.com/geoip-faq/geoip2-and-geoip-legacy-databases/how-much-of-the-ip-address-space-do-you-cover/>
- Mu, C., Zhao, J., Yang, G., Zhang, J., & Yan, Z. (2018). Towards Practical Visual Search Engine Within Elasticsearch. *arXiv preprint arXiv:1806.08896*.
- Takase, W., Nakamura, T., Watase, Y., & Sasaki, T. (2017). A solution for secure use of Kibana and Elasticsearch in multi-user environment. *arXiv preprint arXiv:1706.10040*.

Appendices

Appendix A. Metric aggregation options available for line, area and bar charts

Metric Aggregations	
Count	Returns a raw count of the elements in the selected index pattern.
Average	Returns the average of a numeric field.
Sum	Returns the total sum of a numeric field.
Min	Returns the minimum value of a numeric field.
Max	Returns the maximum value of a numeric field.
Unique Count	Returns the number of unique values in a field.
Standard Deviation	Returns the standard deviation of data in a numeric field.
Top Hit	Returns one or more of the top values from a specific field in user documents. Select a field from the drop-down, how you want to sort the documents and choose the top fields, and how many values should be returned.
Percentiles	Divides the values in a numeric field into user-specified percentile bands.
Percentile Rank	Returns the percentile rankings for the values in the user-specified numeric field.
Parent Pipeline Aggregations	
For each of the parent pipeline aggregations, the metric for which the aggregation is calculated needs to be defined. This may be a new or existing metric. These aggregations can also be nested e.g. to produce 3rd derivative.	
Derivative	Calculates the derivative of specific metrics.
Cumulative Sum	Calculates the cumulative sum of a specified metric in a parent histogram.
Moving Average	Slides a window across the data and emit the average value of that window.
Serial Diff	A technique where values in a time series are subtracted from itself at different time lags or period.
Sibling Pipeline Aggregations	
A metric for which to calculate the sibling aggregation needs to be provided. Additionally, a bucket aggregation—that defines the buckets on which the sibling aggregation will run—needs to be provided.	
Average Bucket	Calculates the (mean) average value of a specified metric in a sibling aggregation.
Sum Bucket	Calculates the sum of values of a specified metric in a sibling aggregation.
Min Bucket	Calculates the minimum value of a specified metric in a sibling aggregation.
Max Bucket	Calculates the maximum value of a specified metric in a sibling aggregation.