Elasticsearch and Visualization Tools

Mustafa Can AYDIN

September 8, 2021

Contents

.

1 Introduction

1.1 What is Elasticsearch

Elasticsearch is a s nosql database with a search engine build using Apache Lucene search engine. Database of the elasticsearch designed so as to perform very quick searches with the downsides of more storage space requirements and slow write operations. So it is commonly agreed that elasticsearch's purpose is not to store data but help increase speeds of data reading. It is really useful in systems where there are few writes and many read at any given time.

1.2 What is Grafana

Grafana is data visulazation tool used with many different data sources.
It is widely used to analyze metric data from the systems such as Cpu/memory-utilization and temprature of the servers.

1.3 What is Kibana

Kibana is data visulazation tool used with elasticsearch as data source.
It is a part of the ELK(elastic-stack) where colllected, filtered, and indexed data is visualized by the help of it. Kibana is widely used for visualization of the logs, since it has a text based searching capabilities it is very easy to analyze log text and visuazlize them for the human reader with Kibana.

1.4 What is Logstash & Beats

Logstash and Beats are mainly data collectors for the elastic stack.

2 Kibana vs Grafana

2.1 Introduction

Both KIbana and Grafana are open-source data visualization tools. Kibana is part of the Elastic stack consisting of Elasticsearch, Logstash, Beats, and Kibana. It is widely used with elasticsearch nodes to visualize data. On the other hand Grafana is used with other tools as data storage such

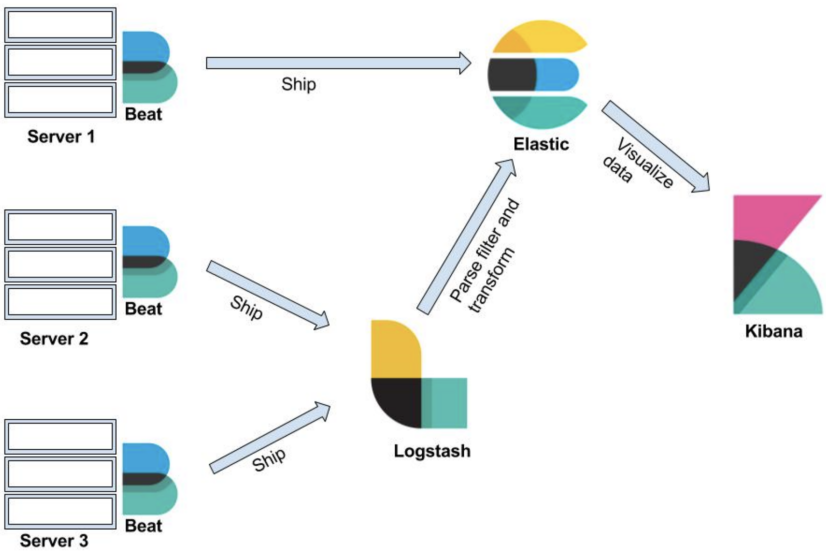

Figure 1: Query inspector in grafana user interface.

as Prometheus, Graphite, InfluxDB, MySQL, and PostgreSQL in addition to elasticsearch. It is widely considered that Kibana is best suited for Log analysis and Grafana is best suited for metric analysis. Both of them have extensive access control & authentication systems. They have some differences when it comes to querying the data. Grafana offers query editor to visualize the data whereas Kibana can be used with elasticsearch query language of KQL. Kibana is very effective for querying the text on the other hand grafana cannot query texts. For instance how many times a word appears in a log file cannot be queried with Grafana. Both Tools have extensive visualization option which can be used in many different types of data to achive desired visualizations for people. Alerting features are richer for the grafana but the most important alerting features can be found in both of them.

2.2 Pros & Cons of Grafana & Kibana

2.2.1 Logging vs. Monitoring

The key difference between the two visualization tools stems from their purpose. Grafana's design for caters to analyzing and visualizing metrics such as system CPU, memory, disk and I/O utilization. The platform does not allow full-text data querying. Kibana, on the other hand, runs

on top of Elasticsearch and is used primarily for analyzing log messages.
If you are building a monitoring system, both can do the job pretty well, though there are still some differences that will be outlined below. If it's logs you're after, for any of the use cases that logs support — troubleshooting, forensics, development, security, Kibana is your only option.

2.2.2 Setup, installation and configuration

Both Kibana and Grafana are pretty easy to install and configure. Both support installation on Linux, Mac, Windows, Docker or building from source. Kibana supports a wider array of installation options per operating system, but all in all — there is no big difference here. Since Kibana is used on top of Elasticsearch, a connection with your Elasticsearch instance is required. Grafana is configured using an .ini file which is relatively easier to handle compared to Kibana's syntax-sensitive YAML configuration files. Grafana also allows you to override configuration options using environment variables.

2.2.3 Data sources and integrations

Grafana was designed to work as a UI for analyzing metrics. As such, it can work with multiple time-series data stores, including built-in integrations with Graphite, Prometheus, InfluxDB, MySQL, PostgreSQL, and Elasticsearch, and additional data sources using plugins. For each data source, Grafana has a specific query editor that is customized for the features and capabilities that are included in that data source. Kibana on the other hand, is designed to work only with Elasticsearch and thus does not support any other type of data source. Kibana on the other hand, is designed to work only with Elasticsearch and thus does not support any other type of data source. In order to extrapolate data from other sources, it needs to be shipped into the ELK Stack (via Filebeat or Metricbeat, then Logstash, then Elasticsearch) in order to apply Kibana to it.

2.2.4 Access control and authentication

By default, and unless you are using either the X-Pack (a commercial bundle of ELK add- ons, including for access control and authentication)

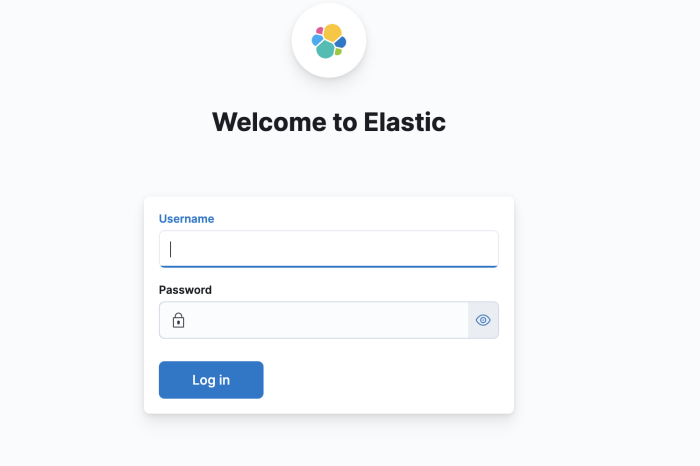

Figure 2: Kibana login page after x-pack enabled.

or open source solutions such as SearchGuard, your Kibana dashboards are open and accessible to the public. In comparison, Grafana ships with built-in user control and authentication mechanisms that allow you to restrict and control access to your dashboards, including using an external SQL or LDAP server. In addition, Grafana's API can be used for tasks such as saving a specific dashboard, creating users, and updating data sources. You can also create specific API keys and assign them to specific roles. Also, for the authentication one might use proxy server on top of kibana and use desired authentication method before reaching to Kibana. For instance, nginx server can be authenticated with the use of htpasswd.

2.2.5 Querying

Querying and searching logs is one of Kibana's more powerful features. Using either Lucene syntax, the Elasticsearch Query DSL or the experimental Kuery, the data stored in Elasticsearch indices can be searched with results displayed in the main log display area in chronological order. Lucene is quite a powerful querying language but is not intuitive and involves a certain learning curve. Grafana, users use what is called a Query Editor for querying. Each data source has a different Query Editor tailored for the specific data source, meaning that the syntax used varies according to the data source. Graphite querying will be different than Prometheus querying, for example.

2.2.5.1 Query examples on Kibana

Kibana's console allow easier cURL commands like below to quickly interact with ES. Some examples :

```
GET _cat/indices
```

```
GET nameOfTheIndex/_search
```

Creating an index with the name of employees

```
PUT employees
```

Defining mapping for the above created index

```
PUT employees/_mapping { "properties": { "date_of_birth": {
    "type": "date", "format": "dd/MM/yyyy" } } }
```

inserting data into the index using bulk api of ES

```
POST _bulk { "index" : { "_index" : "employees", "_id" : "1"
    } } {"id":1,"name":"Huntlee Dargavel","email":"hdargavel
    0@japanpost.jp","gender":"male","ip_address":"58.11.89.19
    3","date_of_birth":"11/09/1990","company":"Talane","
    position":"Research Associate","experience":7,"country":"
    China","phrase":"Multi-channelled coherent leverage","
    salary":180025} { "index" : { "_index" : "employees", "
    _id" : "2" } } {"id":2,"name":"Othilia Cathel","email":"
    ocathel1@senate.gov","gender":"female","ip_address":"3.16
    4.153.228","date_of_birth":"22/07/1987","company":"
    Edgepulse","position":"Structural Engineer","experience":
    11,"country":"China","phrase":"Grass-roots heuristic help
    -desk","salary":193530} { "index" : { "_index" : "
    employees", "_id" : "3" } } {"id":3,"name":"Winston Waren
    ","email":"wwaren2@4shared.com","gender":"male","
    ip_address":"202.37.210.94","date_of_birth":"10/11/1985",
    "company":"Yozio","position":"Human Resources Manager","
    experience":12,"country":"China","phrase":"Versatile
    object-oriented emulation","salary":50616} { "index" : {
    "_index" : "employees", "_id" : "4" } } {"id" : 4,"name"
    : "Alan Thomas","email" : "athomas2@example.com","gender"
```

```
    : "male","ip_address" : "200.47.210.95","date_of_birth"
   : "11/12/1985","company" : "Yamaha","position" : "
   Resources Manager","experience" : 12,"country" : "China",
   "phrase" : "Emulation of roots heuristic coherent systems
   ","salary" : 300000}
```

Search for match of the word heuristic in the phrase field

```
POST employees/_search
{"query": {
"match": {
"phrase": {
"query" : "heuristic"}}}}
```

Search for match of the either word heuristic or word roots or help(words are connected to each other by OR operator by default)

```
POST employees/_search
{"query": {
"match": {
"phrase": {
"query" : "heuristic roots help"}}}}
```

Search for match of the either word heuristic and word roots and help

```
POST employees/_search
{"query": {
"match": {
"phrase": {
"query" : "heuristic roots help",
"operator" : "AND"}}}}
```

2.2.5.2 Query examples on Grafana

Grafana doesnt support Elastic search query domain specific language and only option for querying is to use lucene query language which is harder than ES query language. It has support for ES query language thorugh its UI. For instance, selecting group by filter will result in ES query of

```
{
```

```
  "size": 0,
      "query": {
          "bool": {
              "filter": [
                  {
                      "range": {
                          "@timestamp": {
                              "gte": 1590118795217,
                              "lte": 1590140395217,
                              "format": "epoch_millis"
                          }
                      }
                  },
                  {
                      "query_string": {
                          "analyze_wildcard": true,
                          "query": "*"
                      }
                  }
              ]
          }
      },
      "aggs": {
          "3": {
              "terms": {
                  "field": "application.keyword",
                  "size": 10,
                  "order": {
                      "_term": "desc"
                  },
                  "min_doc_count": 0
              },
              "aggs": {
                  "2": {
                      "date_histogram": {
                          "interval": "15s",
                          "field": "@timestamp",
                          "min_doc_count": 0,
                          "extended_bounds": {
```

```
                                "min": 1590118795217,
                                "max": 1590140395217
                            },
                            "format": "epoch_millis"
                        },
                        "aggs": {}
                    }
                }
            }
        }
    }
```

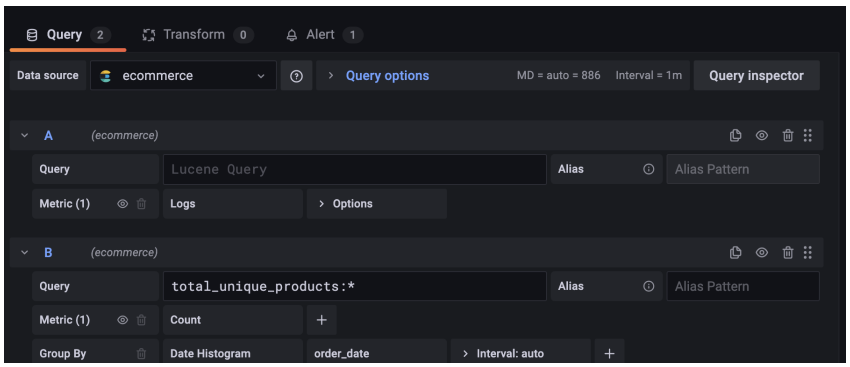

Figure 3: Query inspector in grafana user interface.

One can check the option called query inspector then see the generated query by grafana.

Lucene Query examples on Grafana

As it can be seen from below lucene queries can be shortly entered into grafana and grafana embeds it so as to unify it with the rest of the filters into the final query json.

```
name_of_the_field_to_match:search_term
```

```
total_unique_products:*
```

```
total_quantity:80
```

2.2.6 Dashboards and visualizations

Both have very rich visualization capabilities. Kibana offers a rich variety of visualization types, allowing you to create pie charts, line charts, data tables, single metric visualizations, geo maps, time series and markdown visualizations, and combine all these into dashboards. Dashboards in Kibana are extremely dynamic and versatile — data can be filtered on the fly, and dashboards can easily be edited and opened in full-page format. Kibana ships with default dashboards for various data sets for easier setup time. Grafana dashboards are what made Grafana such a popular visualization tool. They are infamous for being completely versatile. Visualizations in the software are called panels, and users can create a dashboard containing panels for different data sources. Grafana supports graph, singlestat, table, heatmap and free-text panel types. The software's users can make use of a large ecosystem of ready-made dashboards for different data types and sources. Functionality wise — both Grafana and Kibana offer many customization options that allow users to slice and dice data in any way they want. Users can play around with panel colors, labels, X and Y axis, the size of panels, and plenty more. All in all though, Grafana has a wider array of customization options and also makes changing the different setting easier with panel editors and collapsible rows.

2.2.7 Alerts

A key difference between Kibana and Grafana is alerts. Since version 4.x, Grafana has shipped with a built-in alerting engine that allows users to attach conditional rules to dashboard panels that result in triggered alerts to a notification endpoint of your choice (e.g. email, Slack, PagerDuty, custom webhooks). Kibana does not come with an out-of-the-box alerting capability. To add alerting to Kibana users can either opt for a hosted ELK Stack such as Logz.io, implement ElastAlert or use X-Pack.

2.2.8 Community

Both open source tools have a powerful community of users and active contributors. But when looking at the two projects on GitHub, Kibana seems to have the edge. Grafana has about 14,000 code commits while

Kibana has more than 17,000. Both projects are highly active, but taking a closer look at the frequency of commits reflects a certain edge to Kibana. In terms of popularity, we can take a look at Google trends to get an indication. Again, Kibana seemed to have the advantage until 2020. But for a number of reasons, Grafana has eclipsed Kibana in popularity. As changes to Kibana's license result in policy changes from different companies and projects, that gap might widen.

Figure 4: Just like yarn and npm, grafana and kibana have similar popularities with one being slightly ahead.

2.3 Assessment for the use cases

Both Kibana and Grafana are powerful visualization tools. However, at their core, they are both used for different data types and use cases. Grafana, together with a time-series database such as Graphite or InfluxDB is a combination used for metrics analysis; on the other hand. Kibana is part of the popular ELK Stack, used for exploring log data. Both platforms are good options and can even sometimes complement each other. As mentioned above, a significant amount of organizations will use both tools as part of their overall monitoring stack. At Logz.io we use both tools to monitor our production environment, with Grafana hooked up to Graphite, Prometheus and Elasticsearch.

3 Direct interaction with Elasticsearch

3.1 Elasticsearch with Python

3.1.1 Structure

In order to interact with ES using python, one needs to import the elastic search library. After that we can create indices and upload documents to the indices we created. For instance, we can read data from a csv file and then upload it into the elasticsearch. In order to upload the csv file we need to convert it into ELK format. After conversion python elasticsearch library can achieve uploading of the file to the desired index.

3.1.2 Execution

```
python3 esInteraction.py
```

3.2 Elasticsearch with Java

3.2.1 Structure

This project aims to achieve basic crud operation task into ES using spring boot. Application is not only responsible for inserting documents into desired ES indices but only it is able to create, delete, and update indices.

3.2.2 Compilation & Execution

```
mvn install java -jar target/elasticsearchInteractionWithSpring.jar
```

4 R&D Discussion

- Instead of
- Regular

5 Examples

6 Conclusion & Assessment