Elasticsearch on Cloud with Kibana

Elasticsearch

Elasticsearch is a tool for querying written words. It can perform some other nifty tasks, but at its core it's made for wading through text, returning text similar to a given query and/or statistical analyses of a corpus of text.

More specifically, elasticsearch is a standalone database server, written in Java, that takes data in and stores it in a sophisticated format optimized for language based searches. Working with it is convenient as its main protocol is implemented with HTTP/JSON. Elasticsearch is also easily scalable, supporting clustering and leader election out of the box.

Elasticsearch implements a clustered architecture that uses sharding to distribute data across multiple nodes, and replication to provide high availability. Documents are stored in indexes. The user can specify which fields in a document are used to uniquely identify it within an index, or the system can generate a key field and values automatically. The index is used to physically organize documents and is the principal means for locating documents.

The core of elasticsearch intelligent search engine is largely another software project: *Lucene*. It is perhaps easiest to understand elasticsearch as a piece of infrastructure built *around* Lucene's Java libraries. Everything in elasticsearch that pertains to the actual algorithms for matching text and storing optimized indexes of searchable terms is implemented by Lucene. Elasticsearch itself provides a more useable and concise API, scalability, and operational tools on top of Lucene's search implementation.

Elasticsearch uses an implementation which automatically indexes the documents which provides high retrieval speeds which you cannot imagine. Also as it uses the concept of distributed/clustered computing which helps the user to put petabytes of data into their cluster and retrieve them in lightning speed.

Also, Elasticsearch provides clients in a variety of programming languages such as Python, Java, .Net, JavaScript etc. This helps They're easy to work with, feel natural to use, and, just like Elasticsearch, don't limit what you might want to do with them.

Kibana

Kibana is an open source analytics and visualization platform designed to work with Elasticsearch. You use Kibana to search, view, and interact with data stored in Elasticsearch indices. You can easily perform advanced data analysis and visualize your data in a variety of

charts, tables, and maps. Kibana makes it easy to understand large volumes of data. It's simple, browser-based interface enables you to quickly create and share dynamic dashboards that display changes to Elasticsearch queries in real time.

The Kibana interface is divided into four main sections:

- **Discover**: By default, this page will display all of your ELK stacks most recently received logs. Here, you can filter through and find specific log messages based on Search Queries, then narrow the search results to a specific time range with the Time Filter.
- **Visualize**: The Kibana Visualize page is where you can create, modify, and view your own custom visualizations. There are several different types of visualizations, ranging from Vertical bar and Pie charts to Tile maps (for displaying data on a map) and Data tables. Visualizations can also be shared with other users who have access to your Kibana instance.
- **Dashboard**: The Kibana Dashboard page is where you can create, modify, and view your own custom dashboards. With a dashboard, you can combine multiple visualizations onto a single page, then filter them by providing a search query or by selecting filters by clicking elements in the visualization. Dashboards are useful for when you want to get an overview of your logs and make correlations among various visualizations and logs.
- **Settings:** The Kibana Settings page lets you change a variety of things like default values or index patterns.

Problems that Elasticsearch Solve well

There are myriad cases in which elasticsearch is useful. Some use cases more clearly call for it than others. Listed below are some tasks which for which elasticsearch is particularly well suited.

- Auto-completing a search box based on partially typed words based on previously issued searches while accounting for mis-spellings
- Storing a large quantity of semi-structured (JSON) data in a distributed fashion, with a specified level of redundancy across a cluster of machines
- Searching a large number of product descriptions for the best match for a specific phrase and returning the best results

Elasticsearch is generally fantastic at providing approximate answers from data, such as scoring the results by *quality*. While elasticsearch can perform exact matching and statistical calculations, its primary task of search is an inherently approximate task. Finding approximate answers is a property that separates elasticsearch from more traditional databases. That being

said, traditional relational databases excel at precision and data integrity, for which elasticsearch and Lucene have few provisions.

Installation of Elasticsearch

Elasticsearch requires at least Java 8. The binaries are available from www.elastic.co/downloads along with all the releases that have been made in the past. For each release, you have a choice among a zip or tar archive, a DEB or RPM package, or a Windows MS installation package.

For simplicity let's go with installation of Elasticsearch using tar archive:

1. Let's download the Elasticsearch 5.6.3 tar as follows:

```
curl -L -O
https://artifacts.elastic.co/downloads/elasticsearch/elasticsear
ch-5.6.3.tar.gz
```

2. Then extract it as follows:

```
tar -xvf elasticsearch-5.6.3.tar.gz
```

3. It will then create a bunch of files and folders in your current directory. We then go into the bin directory as follows:

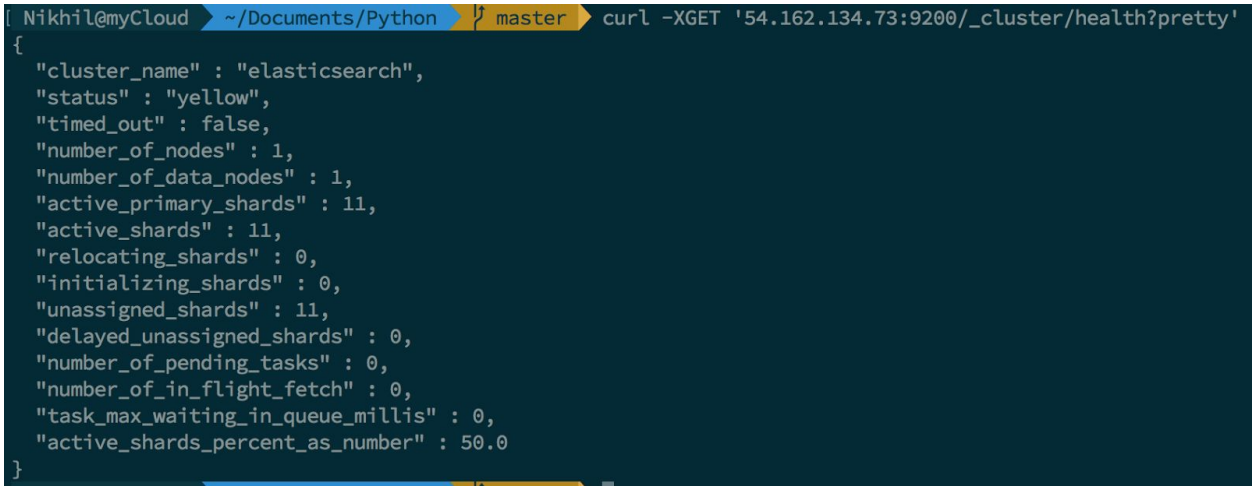
```
cd elasticsearch-5.6.3/bin
```

4. And now we are ready to start our node and single cluster:

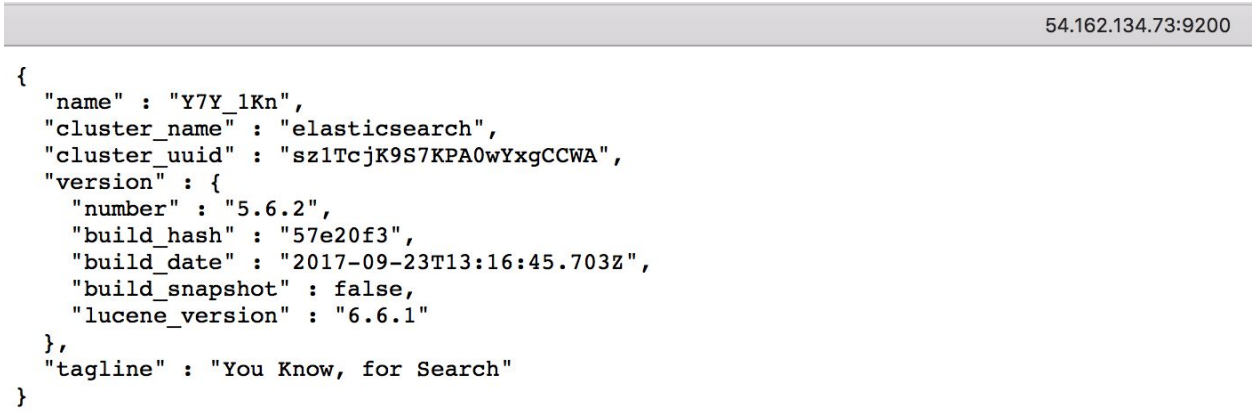
```
./elasticsearch
```

This will start a single node cluster which can be accessed only inside the localhost. So in order to make sure that it is available outside the cluster, we need to add specify the host to which the elasticsearch service will bind. This can be specified in the `config/kibana.yml` file which is inside the elasticsearch folder. You have to change the `network.host = 0.0.0.0`

Once the above change has been done, we can restart the elasticsearch service and this service will bind to the public IP address of the instance. We can check the health of the cluster by using the following command which is shown in the screenshot.



Also we can check if the Elasticsearch service is up and running by just accessing the public IP address in your browser.
In our case : http://54.162.134.73:9200



Installation of Kibana

1. You can download the latest kibana tar package by running the following command.

```
Curl -O -L
https://artifacts.elastic.co/downloads/kibana/kibana-5.6.3-linux
-x86_64.tar.gz
```

2. Extract the Kibana package using the below command

```
tar -xzf kibana-5.6.3-linux-x86_64.tar.gz
```

3. It will then create a bunch of files and folders in your current directory. We then go into the bin directory as follows:

```
cd kibana-5.6.3-linux-x86_64/bin
```

4. We can start the kibana service using the following command.

```
./kibana
```

Similar to Elasticsearch, Kibana will need some configuration adjustments to be able to access from an external network. We need to change server.host and the server.port in the `kibana.yml` file and find the lines referring to server.port and ensure they say server.port: 5601 and `server.host: "0.0.0.0"`. It should only be necessary to uncomment these lines. Once this is done, you can start the Kibana service.

Snapshot of Simple Wikipedia Data

As we know from the previous section, Elasticsearch only supports JSON documents. We have chosen simplewiki document to index into our Elasticsearch instance. Below is the format in which one the document looks like.


```
{
       "template": ["Template:Tl"],
       "redirect": [],
       "version_type": "external",
       "heading": ["July 2016", "Quick deletion of User:Ruhul KH"],
       "source_text": "== July 2016 ==\n==Quick deletion of User:Ruhul KH==\n[[Image:Ambox deletion.png|left|25px]]",
       "wiki": "simplewiki",
       "opening_text": null,
       "coordinates": [],
       "auxiliary_text": [],
       "language": "en",
       "title": "Ruhul KH",
       "version": 5451479,
       "external_link": [],
       "namespace_text": "User talk",
       "namespace": 3,
       "text_bytes": 1533,
       "incoming_links": 1,
       "text": "The page you wrote, User:Ruhul KH, has been selected for quick deletion. If you think this page should be kept, please add {{wait}} below the line {{QD}} and say why on the talk page",
       "category": [],
       "outgoing_link": ["User:Macdonald-ross", "User:MBlaze_Lightning", "User:Ruhul_KH", "User_talk:MBlaze_Lightning", "User_talk:Macdonald-ross", "Wikipedia:Deletion_policy", "Wikipedia:Introduction", "Wikipedia:NPOV", "Wikipedia:Requests_for_undeletion", "Wikipedia:What_Wikipedia_is_not", "Template:QD", "Template:Wait"],
       "timestamp": "2016-07-30T15:48:59Z"
}
```

Uploading the Data into Elasticsearch

We can use the Bulk API to upload a lot of documents together with very less time. The way in which a document can be uploaded is again using the RESTful service of the Elasticsearch service.

```
curl -s -XPOST $es/$index/_bulk?pretty --data-binary @file
```

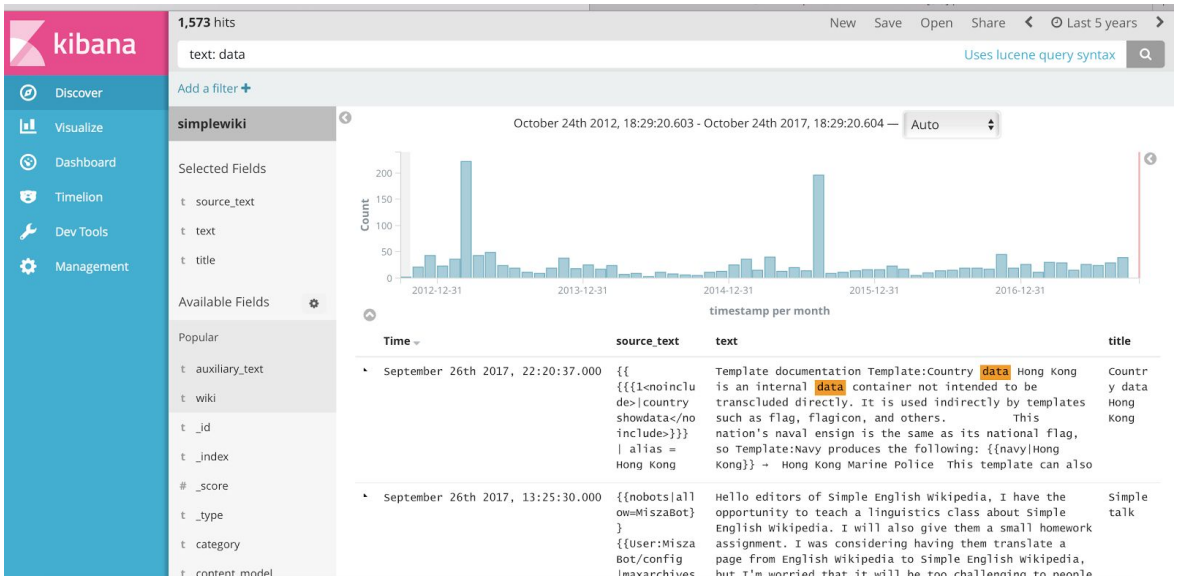
Here `file` represents the file to be indexed.

Once the data is uploaded we can check the number of documents which are present in the elasticsearch by again making a request to the instance. Screenshot below shows the result.

```
[ec2-user@elasticme elasticme]$ curl -XGET 'localhost:9200/simplewiki/page/_count?pretty' -H 'Content-Type: application/json'
{
  "count" : 244349,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  }
}
```

Visualizing through Kibana

Instead of making a request from the command line or using a client from a programming language we have connected kibana to our elasticsearch instance which has helped us in visualizing the data based on a time series. Here is a snapshot of the query through Kibana.



Future Works

We have currently set up elasticsearch and Kibana in Amazon Web Service. For the next phase we would implement both elasticsearch and kibana on Microsoft Azure and Google Cloud and searching for the data through querying and visualizing it through Kibana.

Citations

https://logz.io/blog/install-elk-stack-azure/
https://www.digitalocean.com/community/tutorials/how-to-use-kibana-dashboards-and-visualizations
https://www.3pillarglobal.com/insights/advantages-of-elastic-search
http://exploringelasticsearch.com/overview.html#sec-over-what-is-elasticsearch
https://www.elastic.co/guide/en/kibana/current/targz.html
https://www.elastic.co/products/elasticsearch
https://www.elastic.co/guide/en/elasticsearch/reference/current/_installation.html