

Develop Elasticsearch based data model and automated catalog import pipeline

2018

|  |
| --- |
|  |
|  |
| November 13  A joint paper by JP Tokyo & Co and MRIIRS  Authored by: Somya Dwivedi Reviewed by: Rajesh Nath, Dr. Madhulika |

**Contents**

**1.) Introduction…………………………………………………………………………………………….3**

**2.) Description, Platforms and Technologies…….……………………………………………4**

**3.) Findings and verification….……………………………………………………………………….6**

**4.) About……………………………………………………………………………………………………….7**

**5.) Project artifacts….…………………………………………………………………………………….8**

**6.) References………………………………………………………………………………………………10**

# Introduction

|  |
| --- |
| This White Paper is for informational purposes only which is adhered to indexing the product catalogs in the Elasticsearch instance so as to obtain the search query result information in this document. Products comprising e-commerce and search engines with huge databases usually face issues such as product information retrieval taking too long. This leads to poor user experience and in turn, turns off potential customers. This issue can be solved by Elasticsearch as it provides simplicity in building filters, facets and aggregation. It also has incredibly fast response time, easy to maintain and provides easy analysis of product performance.  Elasticsearch is the most popular open source search engine generally used for the purpose of full-text search and to perform analytics on huge volume, high velocity and variety of data. It is developed alongside a data-collection and log-parsing engine called Logstash, and an analytics and visualization platform called Kibana. The three products are designed for use as an integrated solution, known as the "Elastic Stack" (formerly the "ELK stack").  Elasticsearch uses Lucene and tries to make all its features available through the JSON and Java API. It supports faceting and percolating, which can be useful for notifying if new documents match for registered queries. FSCrawler is also used to index the documents (like PDF). It uses local file system crawling to index new files, update existing files and remove old files. It also has a REST interface which helps us to upload the documents to Elasticsearch.  Here, we aim to develop an Elasticsearch based data model which can automatically import the catalogs (primarily PDFs), index them and make them searchable. The main steps involved in the development of this project are as follows:   1. Develop Elasticsearch data model to store various types of Product Catalogs 2. Product Catalogs can be in various file formats, primarily in PDF 3. Product Catalogs can be in different languages, primarily in Japanese and English. Product catalogs can have text, hyperlinks, images, icons etc. 4. Create data input pipeline to automatically input product catalogs and make them indexed and searchable. This should be operable in the form of batch programs. 5. Create basic data classification algorithms and programs. Also create basic page index and page location mapping algorithms and programs 6. Create data model for storing output of Data Classification algorithms 7. Create data model for storing output of Data Cleansing algorithms 8. Create data model for storing output of data validation algorithms 9. Create data model for storing output of data filtering algorithms 10. Imported catalog data must be indexed and must be quickly searchable (search time reduction and efficiency) 11. All output data of above algorithms must be quickly searchable (search time reduction and efficiency) 12. Do performance improvements of data models based on benchmarking results shared by other project teams |

# Description

|  |
| --- |
| Elasticsearch being a document-oriented database is designed to store, retrieve and manage document oriented or semi-structured data. It uses Lucene Standard-Analyzer to index for automatic category guessing and high precision.  Every feature of Elasticsearch is uncovered as a REST API:   * Index API – Used to document the Index * Get API – Used to retrieve the document * Search API – Used to submit your query and get the result * Put Mapping API – Used to override default choices and define our own mapping   Another important feature is the Ingest API which permits data manipulation and enhancement by configuring a pipeline through which every document is subject to pass. This pipeline is created with a set of processors, each of which do specific tasks that enrich our data. Ingest aims to deliver a lightweight solution for preprocessing and enriching documents within Elasticsearch itself before they are indexed. It interrupts the indexing requests, transforms the documents and passes it back for indexing. The source field must be a base64 encoded binary. To avoid the overhead of converting back and forth between base64, we used base64 python library which automatically does the work.  Steps to configure FSCrawler after downloading Elasticsearch 6.4.2-  1. Create a folder C:\tmp\jp  2. Download all the unprotected PDFs to this folder  3. Download FSCrawler-2.5 from GitHub to the above tmp folder  4. Open cmd and set directory as C:\tmp\fscrawler-2.3\bin  5. Run the command as> fscrawler --config\_dir ./jp catalogs  6. After running type y  7. Now go to "bin\jp\catalogs\\_settings" and change the URL as "C:\\tmp\\jp"  8. Now again run the command as> fscrawler --config\_dir ./jp catalogs --loop 1  The files are indexed now, use the python source code [github link] to run Elasticsearch and test the queries.    Platform and technologies-  The platforms and technologies used in this project are-   * Elasticsearch 6.4.2 - one of the powerful open source search engine * FSCrawler 2.5 – open source tool to index binary documents * cURL- a PC programming venture giving a library and command line tool for exchanging information utilizing different conventions. * Ingest API - permits data manipulation and enhancement by configuring a pipeline * Jupyter notebook (python 3.6) * Python libraries – elasticsearch, base64, sys, json, requests |
| Findings and verificationThe findings of this project are as follows:The protected catalog pdfs were successfully unprotected by saving them to google drive and again downloading themThe task of downloading and unprotecting the catalogs was automated using the UIPath toolThe catalogs were successfully indexed using FSCrawler and imported into the Elasticsearch instanceThe default indexing mechanism by FSCrawler wasn’t very efficient in terms of search query resultsThe catalogs were successfully indexed using Ingest pipeline and customizing the fields by curl commandsThe search query results were now more efficient in Elasticsearch At each step the project was under constant supervision and guidance of our mentors and other group members. Periodic updates were provided to the mentors from MRIIRS and JP Tokyo and co. which were also verified at each step. The implementation of the project was also verified by the peer members and mentors. |
| AboutContributor- Somya Dwivedi“A data science and machine learning enthusiast, passionate about learning new things and applying them practically” Areas of Interest: Data Science, Machine learning, Big Data Analytics, Business Intelligence  Languages: C, C++, Python, R, Java, HTML, CSS, SQL   * Research fellowship under Research Cluster of Computing (MRIIRS) * Achieved 1st position for aptitude in Zenith-2018 * Achieved 3rd position in paper presentation hackathon on IBM ICE day 2018 * Research paper accepted in EECCMC-2018 * Website development trainer for 3 months at Research Cluster of Computing (MRIIRS) * Conducted a Virtual Reality workshop for InnoSkill-2017 * Achieved 2nd position in intra-college website designing competition supported by IEEE computer society  1. Mentors  * Mr. Rajesh Nath - Associate Vice-President of JP Tokyo and co.   rajesh.nath@jptokyo.co.jp   * Dr. Madhulika Bhatia - Associate professor in MRIIRS   madhulika.fet@mriu.edu.in   * JP Tokyo - info@jptokyo.co.jp * MRIIRS - delhi@mrei.ac.in |
| Project artifactsThe source code of curl commands, to create a pipeline to index the documents and perform search query, is as follows://1. starting elastic searchsudo service elasticsearch start// to stop 🡪 sudo service elasticsearch stop//2. creating pipelinecurl -XPUT 'localhost:9200/\_ingest/pipeline/attachment?pretty' -H 'Content-Type: application/json' -d '{"description" : "Extract attachment information encoded in Base64 with UTF-8 charset","processors" : [{"attachment" : {"field" : "data"}}]}'//3. copy the pdfs to same folder containing the python file.//4. How to use the scriptpython pdf\_es.py 10th.pdf//the name of the pdf to be indexed should be provided as the argument and the same will also become the id of the pdf//5. to viewcurl -XGET 'localhost:9200/test\_index/test\_type/test\_id?pretty'the datacurl -XGET 'localhost:9200/test\_index/\_search?q=hello'curl -XGET 'http://localhost:9200/test\_index/\_search?pretty=true&q=\*:\*'The python source code to import and view documents in Elasticsearch is as follows: import base64  import sys  from elasticsearch import Elasticsearch  import requests  import json  es = Elasticsearch()  print(sys.argv[1])  file = open(sys.argv[1], 'rb')  file\_content = file.read()  encoded = base64.encodestring(file\_content)  encoded = encoded.replace ('\n', '')  final = {  'data': encoded  }  print(final)  url = 'http://localhost:9200/test\_index/test\_type/'+sys.argv[1]  headers = {"content-type": "application/json" }  response = requests.put(url, params={ 'pipeline' : 'attachment' }, json=final)  #Print Response  print(response.text) | |

# References

|  |
| --- |
| <https://www.elastic.co/products/elasticsearch><http://www2.lixil.co.jp/rp/dfw/exsas6/webcatalog/kentikumuke.aspx?cate_cd=K1021&_ga=2.14736374.1555774946.1539170516-874854234.1539170516><https://discuss.elastic.co/t/indexing-large-pdf-document/22917><https://github.com/dadoonet/fscrawler/blob/master/docs/source/admin/fs/elasticsearch.rst><https://fscrawler.readthedocs.io/en/fscrawler-2.5/><https://www.elastic.co/guide/en/elasticsearch/guide/current/index-doc.html><https://qbox.io/blog/how-to-index-attachments-and-files-to-elasticsearch-5-0-using-ingest-api><https://elasticsearch-py.readthedocs.io/en/master/><https://www.elastic.co/guide/en/elasticsearch/reference/current/ingest-apis.html> |
|  |