

|  |
| --- |
| REPORT TITLE  2018 |
|  |
| November 13  A joint paper by JP Tokyo & Co and MRIIRS  Authored by: Sayem Uddin Khan Reviewed by: Rajesh Nath, Dr.Madhulika |

**Contents**

**1.) Introduction, purpose and heading………………………………………………………….3**

**2.) Description, Platforms and Technologies…….……………………………………………7**

**3.) Findings and verification….……………………………………………………………………….8**

**4.) About……………………………………………………………………………………………………….9**

**5.) Project artifacts….……………………………………………………………………………………10**

**6.) References………………………………………………………………………………………………11**

# Introduction.

|  |
| --- |
| This White Paper is for informational purposes only which is adhered to indexing the product catalogs in the elastic search instance so as to obtain the search query result information in this document. About Elastic search  Elastic search is a search engine based on Lucene. It provides a distributed, multitenant  Capable full-text search engine with an HTTP web interface and schema-free JSON documents. Elastic search is developed in Java and is released as open source under the terms of the   Apache license. Official clients are available in Java, .NET (C#), PHP, and Python, Apache Groovy, Ruby and many other languages. According to the DB-Engines ranking, Elasticsearch is the most popular enterprise search engine followed by Apache Solr, also based on Lucene.  Elasticsearch 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, referred to as the "Elastic Stack" (formerly the "ELK stack").  Elasticsearch can be used to search all kinds of documents. It provides scalable search, has near real-time search, and supports multitenancy. "Elasticsearch is distributed, which means that indices can be divided into shards and each shard can have zero or more replicas. Each node hosts one or more shards, and acts as a coordinator to delegate operations to the correct shard(s). Rebalancing and routing are done automatically". Related data is often stored in the same index, which consists of one or more primary shards, and zero or more replica shards. Once an index has been created, the number of primary shards cannot be changed.  Elasticsearch uses Lucene and tries to make all its features available through the JSON and Java API. It supports facetting and percolating, which can be useful for notifying if new documents match for registered queries.  Another feature is called "gateway" and handles the long-term persistence of the index;[]](https://en.wikipedia.org/wiki/Elasticsearch#cite_note-gateway-6) for example, an index can be recovered from the gateway in the event of a server crash. Elasticsearch supports real-time GET requests, which makes it suitable as a NoSQLdatastore, but it lacks distributed transactions.  2.) Description  This Data model of Elastic search is being made to search queries as fast as possible so as to remain in the continuous flow of getting the results. Elasticsearch is a database that stores, retrieves, and manages document-oriented and semi-structured data.  Products that involve e-commerce and search engines with huge databases are facing issues such as product information retrieval taking too long. This leads to poor user experience and in turn turns off potential customers.  Lag in search is attributed to the relational database used for the product design, where the data is scattered among multiple tables and retrieval of meaningful user information require fetching the data from them. The Relational Database works comparatively slow when it comes to huge data and fetching search results through queries from the database. Businesses nowadays looking for alternate ways where the data stored in such a way that the retrieval is quick. This can be achieved by adopting NOSQL rather than RDBMS for storing data. Elasticsearch is one such NOSQL distributed database. Elasticsearch relies on flexible data models to build and update visitors profiles to meet the demanding workload and low latency required for real-time engagement.  Relational database works comparatively slow when it comes to huge data and fetching search results through queries from the database. (There are ways to optimize this like indexing but then there are related limitations like we can’t index every field. Row updates to heavily indexed tables would take time. People also scale their RDBMS vertically to improve performance.) This is a problem is overcome by Elasticsearch. Below figure shows how RDBMS ideally work for searching things from the Database.  SQL works with RDBMS  Let’s understand what is so significant about Elasticsearch. It is a document-oriented database, designed to store, retrieve and manage document oriented or semi-structured data. When you use Elasticsearch you store data in JSON document form. Then you query them for retrieval. It is schema-less, using some defaults to index the data unless you provide mapping as per your need. Elasticsearch uses Lucene StandardAnalyzer for indexing for automatic type guessing and more precision.  Every feature of Elasticsearch is exposed as a REST API   1. Index API – Used to document the Index 2. Get API – Used to retrieve the document 3. Search API – Used to submit your query and get the result 4. Put Mapping API – Used to override default choices and define our own mapping   Elastic search has its own Query Domain Specific Language, where you specify the query in JSON format. You can also nest other queries based on your need. Real projects require search on different fields by applying some conditions, different weights, recent documents, values of some predefined fields and so on. All such complexity can be expressed through a single query. The query DSL is powerful and designed to handle the real world query complexity through a single query. Elasticsearch APIs are directly related to Lucene and it is using the same as Lucene operations name. Query DSL also using the Lucene TermQuery to execute it.  Below figure shows how the Elasticsearch query works.  indexing and searching in Elasticsearch  **Platforms and Technologies.**   1. Discussing about the platform and technologies, It is known that the most powerful search engine i.e Elastic search 6.4.2 is used which is based on Lucene on an operating system of Windows-10. 2. An open source tools i.e FS crawler 2.5 is used to index of the documents available in a pdf format. 3. Jupyter notebook(python 3.6) 4. Libraries-  * **PyPDF2**-PyPDF2 is a pure-python PDF library capable of splitting, merging together, cropping, and transforming the pages of PDF files.It can retrieve text and metadata from PDFs as well as merge entire files together. * **OS**-The OS module in Python provides a way of using operating system dependent functionality. The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on – be that Windows, Mac or Linux. * **Glob**-The glob module finds all the pathnames matching a specified pattern according to the rules used by the Unix shell, although results are returned in arbitrary order**.**  1. **Findings and verifications**   Firstly the catalogs are imported successfully into the elastic search instance using the  FS crawler module and configured thereby. This will put all the catalogs that need to be indexed in the ES instance.  **Steps:-**   * Go to tmp/fscrawler/bin and set this as the current directory * Configure the product catalogs using command “fscrawler –config\_dir ./jp catalogs”. * Now run the files into a loop so that they all would get indexed using command   “fscrawler --config\_dir ./jp catalogs --loop 1”.   * Files are now indexed and we can check whether our search query is working fine or not. Source code and all design part is being given in the next section along with their outputs.     The research and the implementation part was under a constant guidance of the respective mentors and other group members also. Periodic updates were also given by the mentors regularly and it was also constantly discussed with the peer members of the group without whom this project would never be successful.  **7.) About**     1. **Contributor-Sayem Uddin khan**   **C:\Users\HP\Desktop\Khansahab.jpg**  “A keen and a dedicated learner who wishes to learn new things, concepts and try to apply the learned knowledge and techniques so as to bring an edge out of it”     * Received an honor award from Manav Rachna International institute of research and studies in Excellence in Academic Research. * Won a Silver medal in the IBM day paper presentation competition. * Currently persuing training in core and advanced java. * Published a research paper i.e Big data management using cloud computing and different tools published in IJRFE journal issue 2 vol-6, * Good technical content writer. * Interested subjects include C++, DBMS ,Java, Data structures  1. **Mentors**  * **Mr.Rajesh Nath-**Associate Vice-President of Jp Tokyo and Company.   rajesh.nath@jptokyo.co.jp   * **Dr.Madhulika-**Associate professor in MRIIRS.   madhulika.fet@mriu.edu.in   1. **JP Tokyo-**info@jptokyo.co.jp 2. **MRIIRS-**delhi@mrei.ac.in 3. **Project Artifacts**   **Prototype**  The catalogs were imported successfully in elastic search by first setting the required directory as “C:\tmp\fscrawler-2.3\bin” and then configuring the catalogs into the elastic search. By default Elastic search uses its own default analyzer to index the documents until and unless we defined a new way to index them. So it will ask to create index or not. After creating the index for the documents, now we are ready to fire our query and see the results.  C:\Users\HP\Desktop\FSCRAWLER.1.PNG  Now after importing and configuring the product catalogs successfully into the elastic search, index the documents using the command **“fscrawler --config\_dir ./jp catalogs --loop 1”** .  This will indexed all the files and now we can search our queries into the Elastic search.  **Source code and Design**  Now some results.  In [1]:  **from** **elasticsearch** **import** Elasticsearch  **import** **os**  **import** **glob**  **import** **PyPDF2**  **import** **pandas** **as** **pd**  In [2]:  os.chdir("C:**\\**tmp**\\**jp")  files=glob.glob("\*.\*")  In [3]:  len(files)  Out[3]:  5  In [4]:  **for** catalog **in** files:  print(catalog)  “ACFrOgAHo4sXcSv5qAOuMBr0uEWAOKGCh5NigeYzfP7TjOf7nNJFi7eP4Bjlfp9yGFyB7PiQCcsnNGA7e9AH1\_LtGrKgCqCWqmsElMEw20u5aR0gDdBIX9\_y36-cpY0=.pdf  ACFrOgAI3Y336GlEyJBySMpJQbsxFJY4OLPbBLCV53g7eljlGnyrt2l3B3BApP8Dclnpdpk5Mu8Kn2usHrOzzoHQGXO4HmvgjqWtQLCHsfBEuZPNA1Nrf7ufW-NlDHQ=.pdf  ACFrOgB-HyyouVKN\_pDn9PD5SPsjIIurmD5B9iB-j0FSKTowpvV1bCQnR9mTT8Y8y59w6rR-0qGNWZVPCRXJi2s-ysapNxHp8mpRBvDSX\_A5waXIsH8ndVx6OphWYAo=.pdf  ACFrOgBJW4oArIaLPE2RJ3VdYwNQnVw\_\_wzSEzQlMnYhY1jbqlYAT9XpGaORTL7Kfnbqiu2Tpez-LLdGzbQ5HRP8izvROPCyECyXdbILE9boAzDiLiprCU9h3qvEDDo=.pdf  ACFrOgBWZenYp9goTfOZ-StQJqg2d72CeQh-GPM8uCHqWE2rHpE1PAXWXsnO9irrvvM7ZiifwP8XfZQEqnpQ-dDh8x74MjnHGsEWgaCRBR41L-TN54i2rrjR1Gxyn7E=.pdf”  In [5]:  **def** extractPdfFiles(files):  this\_loc=1  df=pd.DataFrame(columns=("name","content"))    **for** file **in** files:  pdfFileObj=open(file,'rb')  pdfReader=PyPDF2.PdfFileReader(pdfFileObj)  n\_pages=pdfReader.numPages  this\_doc=''  **for** i **in** range(n\_pages):  pageObj=pdfReader.getPage(i)  this\_text=pageObj.extractText()  this\_doc+=this\_text  df.loc[this\_loc]=file,this\_doc  this\_loc=this\_loc+1  **return** df  In [6]:  df=extractPdfFiles(files)  In [7]:  df.head()  Out[7]:    In [8]:  es=Elasticsearch()  In [9]:  col\_names=df.columns  **for** row\_number **in** range(df.shape[0]):  body=dict([(name,str(df.iloc[row\_number][name])) **for** name **in** col\_names])  es.index(index='catalogs',doc\_type='doc',body=body)  In [32]:  search\_results=es.search(index='catalogs',doc\_type='doc',  body={"\_source":"name",  "query":{  "match\_phrase":{"content":"掲載内容"},  }  })  In [33]:  search\_results  Out[33]:  {'took': 11,  'timed\_out': False,  '\_shards': {'total': 5, 'successful': 5, 'skipped': 0, 'failed': 0},  'hits': {'total': 1,  'max\_score': 1.8330884,  'hits': [{'\_index': 'catalogs',  '\_type': 'doc',  '\_id': '458fc5922a961f24ec68d86b9cdfae7',  '\_score': 1.8330884,  '\_source': {}}]}}  In [34]:  search\_results['hits']['total']  Out[34]:  1   1. **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> |