ELASTIC SEARCH FOR CODEX

Elasticsearch is an open-source, enterprise-grade search engine which can power extremely fast searches that support all data discovery applications. With Elasticsearch we can store, search, and analyze big volumes of data quickly and in near real time. It is generally used as the underlying search engine that powers applications that have simple/complex search features and requirements.

Terminologies:

- **Cluster:** A cluster is a collection of nodes that shares data.
- **Node:** A node is a single server that is part of the cluster, stores the data, and participates in the cluster's indexing and search capabilities.
- **Index:** An index is a collection of documents with similar characteristics. An index is more equivalent to a schema in RDBMS.
- **Type:** There can be multiple types within an index. For example, an ecommerce application can have used products in one type and new products in another type of the same index. One index can have multiple types as multiple tables in one database.
- **Document:** A document is a basic unit of information that can be indexed. It is like a row in a table.
- **Shards and Replicas:** Elastic Search indexes are divided into multiple pieces called shards, which allows the index to scale horizontally. Elastic Search also allows us to make copies of index shards, which are called replicas.

Setup:

Install latest Elastic Search and run it. By default it runs at 9200 port and you can check it by your browser (localhost:9200).

NOTE**:* For better understanding and ease of access, use *Chrome Extension* ***Elastic Head.

STEPS USED IN BUILDING SEARCH ENGINE FOR CODEX 3.0

Before you start to learning this, you need to have knowledge of database, clusters, keywords, stop words, nlp etc. Learn these from my blog on chatbots.

Assuming that you know above concepts lets begin.

PUSHING DATA TO ES:

- First step is building your own cluster. To build it we need data.
- Once you know what data you want to search, store it or push it in JSON format to Elastic Search.
- Each Json Object has to be stored in unique combination of *index, type & id.*

  Example: If you have 2 Employees data i.e

```
"employee":
{
 "name":      "sonoo",
 "salary":     56000,
 "married":    true
  "empid":    115366
 }

"employee":
{
 "name":      "monoo",
 "salary":     56000,
 "married":    true
 "empid":     115364
 }
```

You can push these to ES by making index as *employee*, type as *name* & id as *empid.* Note that **type** should be string without space so you can use employee phone number if there.

- As shown in above example, Every data from CodeX is pushed to ES.
- For each feature in CodeX, a index is created i.e
  Example: For Escalation data:
  Index*-> escalation*
  Type*->Codex_ID*
  Id*-> Escalation Id*

- If you want to add a new feature, add it similarly.

RETRIEVING DATA FROM ES:

- To retrieve data, from elastic search we must process the user text to get index, type, id.
  Example: Search Text is *"Escalation raised by Sachin Bainur for project MOVE"*
  Here you must get index->escalation and from that cluster, search data related to Sachin and Move.
- Refer chatbot blog, how you can get keywords from sentence.
- Once you have your keywords, compare with all indices to get related match of index for ES.
- Use pattern match to get data from particular index.
- If no index matches, Search pattern from all the data and get result.