Elastic Search

Elasticsearch is the distributed search and analytics engine at the heart of the Elastic Stac.

* **Purpose**: Primarily used for full-text search, structured search, analytics, and log/event data management.
* **Key Features**:
  + **Real-time search**: Allows for quick search across large datasets.
  + **Scalability**: Designed to scale horizontally, meaning it can handle large amounts of data by distributing it across multiple servers.
  + **RESTful API**: Uses a JSON-based API for interacting with data.
  + **Advanced Query Capabilities**: Supports complex queries, aggregations, and full-text search capabilities.

# 1. Elastic Stack (ELK Stack)

* **Definition**: A collection of open-source tools (Elasticsearch, Logstash, Kibana, and Beats) developed by Elastic for data ingestion, search, analytics, and visualization.
* **Components**:
  + **Elasticsearch**: The core search and analytics engine.
  + **Logstash**: A data processing pipeline tool that collects, processes, and forwards data to Elasticsearch.
  + **Kibana**: A visualization tool used for exploring and visualizing data stored in Elasticsearch, creating dashboards, and monitoring system metrics.
  + **Beats**: Lightweight data shippers that collect data from various sources (like logs, metrics, and network data) and send it to Logstash or Elasticsearch.

1. **Use Cases**

* **Log and Event Data Management**: Used for storing and analyzing logs generated by servers, applications, or network devices.
* **Monitoring and Observability**: Provides real-time monitoring and observability for infrastructure, applications, and networks.
* **Security Analytics**: Helps in identifying and responding to security threats.
* **Search Applications**: Powers search functionality in websites and applications.

# How to Add the data in the Elastic Search

## Document:-

In Elasticsearch, a document can be more than just text, it can be any structured data encoded in JSON. That data can be things like numbers, strings, and dates. Each document has a unique ID and a given data type, which describes what kind of entity the document is.

## Index:-

An index can be thought of as an optimized collection of documents and each document is a collection of fields, which are the key-value pairs that contain your data.

By default, Elasticsearch indexes all data in every field and each indexed field has a dedicated, optimized data structure. For example, text fields are stored in inverted indices, and numeric and geo fields are stored in BKD trees. The ability to use the per-field data structures to assemble and return search results is what makes Elasticsearch so fast.

Elasticsearch also has the **ability to be schema-less**, which means that documents can be indexed without explicitly specifying how to handle each of the different fields that might occur in a document. When **dynamic mapping** is enabled, Elasticsearch automatically detects and adds new fields to the index. This default behavior makes it easy to index and explore your data—​just start indexing documents and Elasticsearch will detect and map booleans, floating point and integer values, dates, and strings to the appropriate Elasticsearch data types.

## Send requests to Elasticsearch

You send data and other requests to Elasticsearch using REST APIs. This lets you interact with Elasticsearch using any client that sends HTTP requests, such as [curl](https://curl.se/). You can also use Kibana’s Console to send requests to Elasticsearch.

# Searching Data

A *search query*, or *query*, is a request for information about data in Elasticsearch data streams or indices.

## Search in all the documents

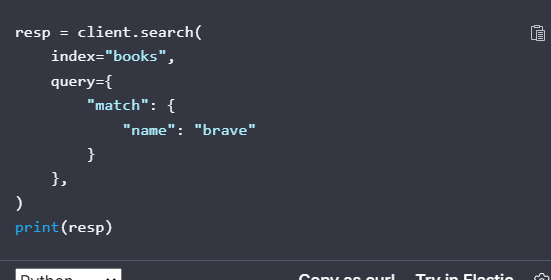
Run the following command to search the books index for all documents:

**resp = es.search(index=<index\_name>)**

The \_source of each hit contains the original JSON object submitted during indexing.

## Match Query

You can use the match query to search for documents that contain a specific value in a specific field. This is the standard query for performing full-text search, including fuzzy matching and phrase searches.



1. **What are Elasticsearch mappings?**

Elasticsearch mappings are data structure definitions that determine how storage and searching work within an Elasticsearch index.

When you create an index, you can define the mappings for each field in the index, including its data type and how it is indexed. In the following sections, we’ll take a look at four key Elasticsearch mapping concepts:

* **Data types:**Indicators of the type data stored in a field.
* **Field properties:**Sub-fields that describe data.
* **Mapping types:**Define whether mappings are static or dynamic.
* **Analyzers:**Text-processing mechanisms that define how characters are interpreted.

### Key Differences Between Dynamic and Static Mapping

| **Feature** | **Dynamic Mapping** | **Static (Explicit) Mapping** |
| --- | --- | --- |
| **Definition** | Automatically infers field types. | Manually defined by the user. |
| **Control** | Less control, depends on Elasticsearch's guess. | Full control over data types and indexing behavior. |
| **Use Case** | Quick and easy indexing, good for prototypes. | Production environments where consistency is crucial. |
| **Flexibility** | Adds new fields automatically. | Only allows pre-defined fields unless updated. |
| **Risk of Incorrect Types** | Possible, due to automatic inference. | Minimal, since fields are explicitly defined. |

1. Term-level queries

You can use **term-level queries** to find documents based on precise values in structured data. Examples of structured data include date ranges, IP addresses, prices, or product IDs.

Unlike [full-text queries](https://www.elastic.co/guide/en/elasticsearch/reference/current/full-text-queries.html), term-level queries do not analyze search terms. Instead, term-level queries match the exact terms stored in a field.