**Introduction to JSON and YAML And Basics Data Types**

Elasticsearch supports several basic data types for indexing and storing data. These data types can be categorized as follows:

**1. Core Data Types**

* **text** – Used for full-text search, analyzed for tokenization.
* **keyword** – Used for exact matching, not analyzed.
* **long** – 64-bit integer (e.g., 123456789).
* **integer** – 32-bit integer (e.g., 1000).
* **short** – 16-bit integer (e.g., 255).
* **byte** – 8-bit integer (e.g., 127).
* **double** – 64-bit floating point number (e.g., 123.456).
* **float** – 32-bit floating point number.
* **half\_float** – 16-bit floating point number.
* **scaled\_float** – Floating point number with fixed scaling factor.

**2. Boolean Type**

* **boolean** – Stores true or false.

**3. Date Type**

* **date** – Stores date/time values in ISO 8601 format (e.g., 2024-02-02T15:00:00Z).

**4. Complex Data Types**

* **object** – Stores JSON objects.
* **nested** – Similar to object, but enables independent indexing of nested fields.

**5. Special Data Types**

* **geo\_point** – Stores latitude and longitude for geospatial queries.
* **geo\_shape** – Stores complex geographical shapes (e.g., polygons).
* **ip** – Stores IPv4 and IPv6 addresses.
* **completion** – Used for autocomplete suggestions.
* **token\_count** – Stores the number of tokens in a text field.
* **percolator** – Used for storing queries to be matched against incoming documents.

**1. JSON Basics**

JSON (JavaScript Object Notation) is a lightweight data-interchange format that is easy for humans to read and write and easy for machines to parse and generate.

**Example of a simple JSON object:**

{

"name": "Alice",

"age": 25,

"city": "New York"

}

* JSON is built on key-value pairs.
* Keys are always strings, enclosed in double quotes.
* Values can be strings, numbers, objects, arrays, or boolean values.

**2. Nested JSON**

Nested JSON is when a JSON object contains another JSON object as a value.

**Example of nested JSON:**

{

"student": {

"name": "Bob",

"age": 22,

"subjects": ["Math", "Science"],

"address": {

"street": "123 Main St",

"city": "Los Angeles",

"zipcode": "90001"

}

}

}

* The student object contains another object address.
* The subjects key contains an array.

**3. JSON Array**

A JSON array is a collection of values enclosed in square brackets.

**Example of a JSON array:**

{

"employees": [

{"name": "John", "age": 30},

{"name": "Jane", "age": 28},

{"name": "Mike", "age": 35}

]

}

**4. YAML Basics**

YAML (YAML Ain’t Markup Language) is a human-readable data format often used for configuration files.

**Example of a simple YAML file:**

name: Alice

age: 25

city: New York

* YAML does not use quotation marks around keys unless necessary.
* Indentation is used instead of braces {}.
* Lists are represented using dashes -.

**5. Nested YAML**

YAML allows nesting using indentation.

**Example of nested YAML:**

student:

name: Bob

age: 22

subjects:

- Math

- Science

address:

street: 123 Main St

city: Los Angeles

zipcode: 90001

**6. YAML vs. JSON Comparison**

|  |  |  |
| --- | --- | --- |
| **Feature** | **JSON** | **YAML** |
| Syntax Complexity | More structured with braces {} | Simpler, indentation-based |
| Readability | Harder for humans | Easier to read |
| Use Case | APIs, web applications | Configuration files, Kubernetes |

**Conclusion**

* JSON is widely used for data exchange in APIs.
* YAML is often preferred for configuration files due to its readability.
* Both formats support nesting and arrays.