# 

# Detailed Notes on Dynamic Mapping in Elasticsearch

## Introduction to Dynamic Mapping

Till now we studied explicit mapping.

Dynamic mapping in Elasticsearch allows for automatic creation of field mappings when documents are indexed without requiring explicit mappings beforehand. This feature simplifies the process of indexing data, especially for fields that Elasticsearch encounters for the first time.

## Examples of Field Mappings

When Elasticsearch encounters a new field, it automatically assigns a mapping based on the field's data type. For instance:

- The 'created\_at' field is mapped to the 'date' data type due to Elasticsearch's date detection process, even if it was specified as a string.

- The 'in\_stock' field is mapped to the 'long' data type when a numeric value without decimals is provided.

- The 'tags' field is mapped as a multi-field with both 'text' and 'keyword' data types to support full-text searches and exact matches respectively.

## Optimizations and Considerations

Dynamic mapping provides sensible defaults but might not always be optimal. For example:

- The 'text' mapping for 'tags' may be unnecessary if only exact matches and aggregations are required.

- The 'keyword' mapping for 'description' may be redundant if only full-text searches are needed.

To optimize storage and indexing performance, explicitly define mappings when handling large datasets or when specific query behaviors are known.

## Rules for Dynamic Mapping

Elasticsearch uses a set of rules to determine field mappings dynamically:

- Strings are typically mapped as 'text' fields with a nested 'keyword' mapping unless they match date detection formats.

- Numeric detection maps strings containing numbers to 'float' or 'long' data types, though this is disabled by default.

- Integers are mapped to 'long', floating-point numbers to 'float', and booleans to 'boolean'.

- Objects are mapped using their key-value pairs, with the keys mapped like any other fields.

- Arrays are mapped based on the first non-null value in the array.

NULL values are ignored, and no mapping is created for such fields.

## Inspecting Dynamic Mappings

Dynamic mapping is enabled by default, as demonstrated with the 'products' index in an earlier example. Field mappings are created automatically based on the field values supplied during indexing.

For example, the 'description' and 'tags' fields were mapped as 'text' and 'keyword' fields. However, such mappings may not always be efficient. Optimizing mappings can reduce storage space and increase indexing throughput.

## Conclusion

Dynamic mapping simplifies Elasticsearch usage by creating field mappings automatically based on encountered data types. While it provides a convenient default behavior, explicitly defining mappings for large-scale or specific use cases ensures optimal performance and storage efficiency.