# Creating Field Mappings in Elasticsearch

## 1. Introduction

Field mappings in Elasticsearch define the structure and data types of fields within an index. You can add mappings either when creating an index or afterwards. This lecture focuses on defining mappings during index creation, as it is the most common initial interaction with mapping.

## 2. Defining Mappings While Creating an Index

Mappings are defined within a 'mappings' key when creating a new index. All field mappings are specified within a 'properties' key, which applies to fields at every level of the hierarchy, including nested objects.

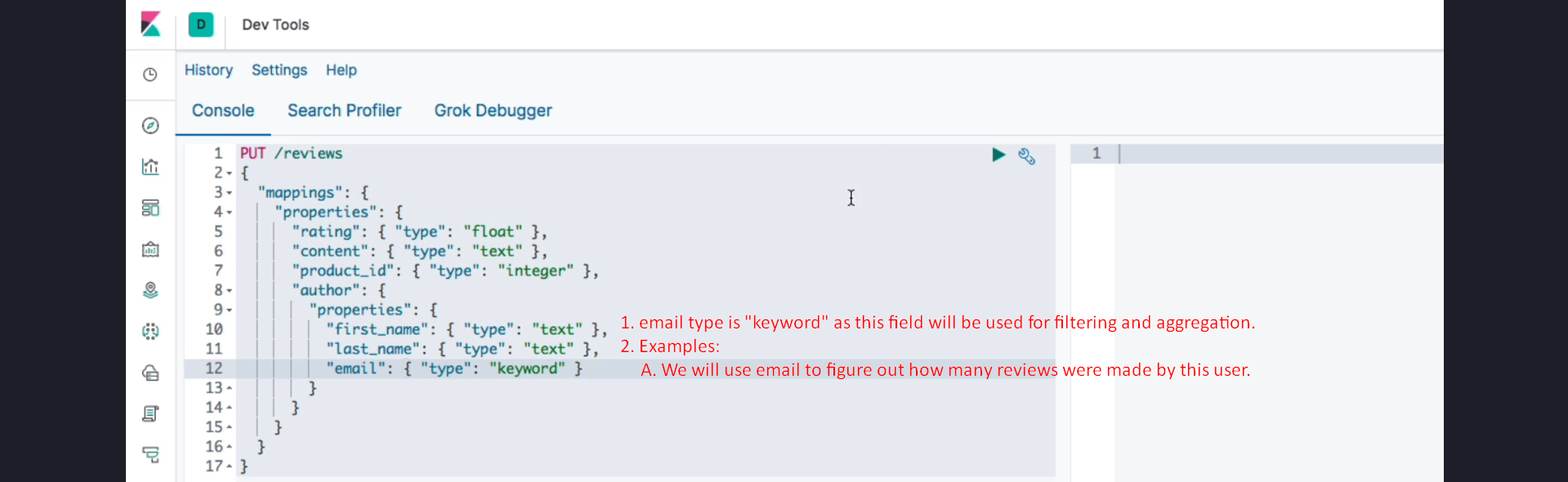
### 2.1 Example Mapping Structure

Here is an example of creating mappings for a new index with four fields: 'rating', 'content', 'product\_id', and 'author'. The 'author' field is an object containing additional nested fields.

PUT /reviews  
{  
 "mappings": {  
 "properties": {  
 "rating": { "type": "float" },  
 "content": { "type": "text" },  
 "product\_id": { "type": "integer" },  
 "author": {  
 "properties": {  
 "first\_name": { "type": "text" },  
 "last\_name": { "type": "text" },  
 "email": { "type": "keyword" }  
 }  
 }  
 }  
 }  
}

## 3. Choosing Data Types

When defining field mappings, it is important to select the correct data type for each field. For example, the 'rating' field is a floating-point number, while the 'content' field is a 'text' field for full-text search.

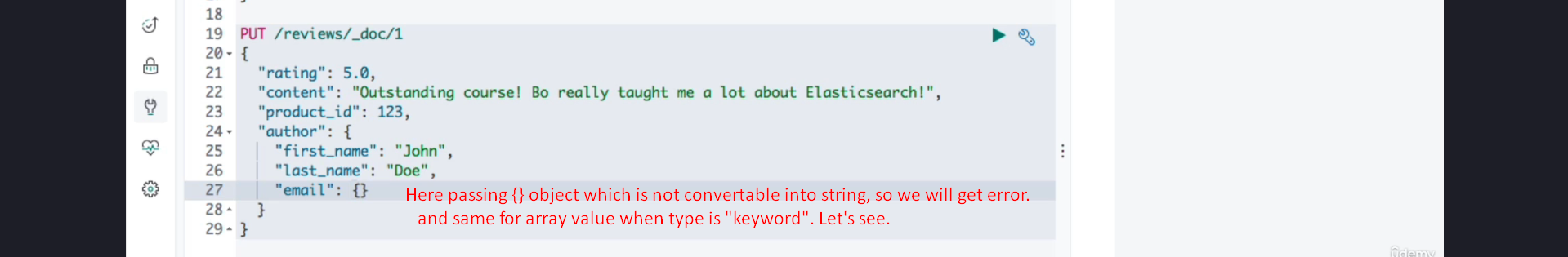
The 'email' field in the nested 'author' object is mapped as a 'keyword' data type because it is likely to be used for filtering, aggregations, or exact matches, rather than full-text search.  
Below is the example for creating mapping at the time of creation of index.  
  
  

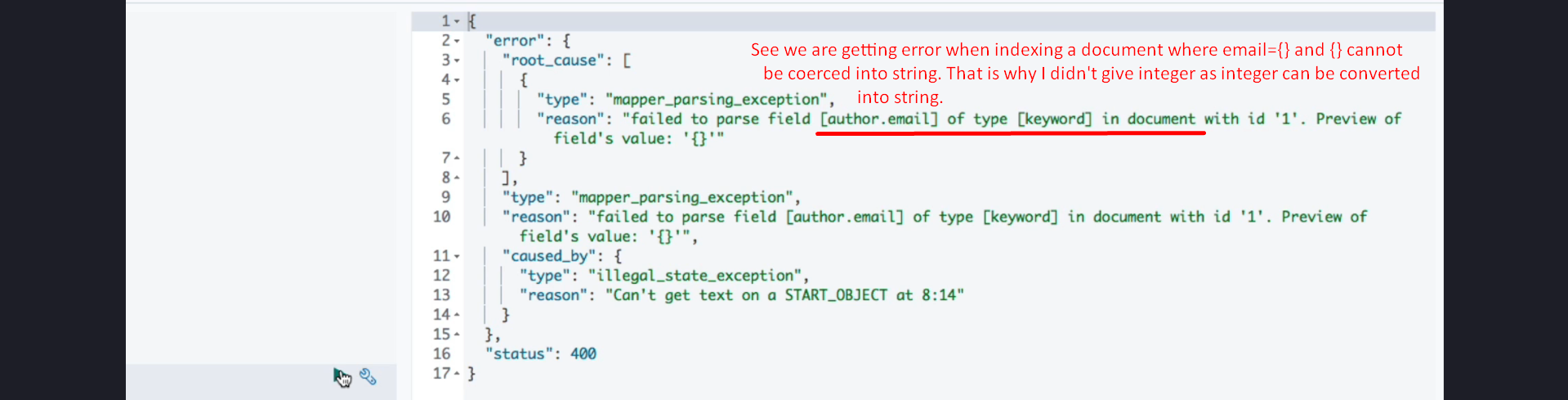

Choosing the correct data type ensures efficient querying and accurate results.

## 4. Indexing Documents with Mapping Constraints

After creating an index with the defined mapping, documents can be indexed into it. However, if a document's field values do not match the mapping, Elasticsearch will reject the document.

For instance, if the 'email' field is supplied as an object instead of a string, Elasticsearch will throw an error. On the other hand, if coercion is enabled and the mismatch is minor (e.g., providing an integer for a string field), Elasticsearch will attempt to convert the value.

Here is an example of indexing a document:  
POST /reviews/\_doc  
{  
 "rating": 4.5,  
 "content": "Great product!",  
 "product\_id": 12345,  
 "author": {  
 "first\_name": "John",  
 "last\_name": "Doe",  
 "email": "john.doe@example.com"  
 }  
}  
  




## 5. Naming Conventions for Fields

Elasticsearch does not enforce a specific naming convention for field names. However, it is recommended to follow consistent naming practices such as camelCase or snake\_case, as these are widely used in JSON.

## 6. Summary

1. Field mappings define the structure and data types of fields in an index.  
2. Mappings are defined within a 'mappings' key and use a 'properties' key to specify fields and their types.  
3. Nested objects require additional 'properties' keys to define their internal structure.  
4. Selecting appropriate data types ensures efficient querying and accurate results.  
5. Elasticsearch rejects documents with field values that do not match the defined mapping, unless coercion is enabled.  
6. Consistent naming conventions, such as camelCase or snake\_case, are recommended for field names.