# Reindexing Documents in Elasticsearch

## Introduction

As discussed in the previous lecture, changing how documents are mapped usually requires reindexing documents. This process allows us to adapt to new business requirements or accommodate changes in how data is stored as we scale.

## Creating a New Index

The first step in reindexing is creating a new index with the updated mapping. For example, if the 'product\_id' field is currently mapped as a 'long' type but now requires letters, we can update it to the 'keyword' data type.

To create the updated 'reviews' index, we retrieve the current mapping, modify the 'product\_id' field to 'keyword', and create the new index.

## Reindexing Documents

Instead of writing a custom script to migrate data from the old index to the new one, Elasticsearch provides the Reindex API. This API simplifies the reindexing process.

The basic query for reindexing requires defining the source and destination indices. Here's an example:

POST \_reindex  
{  
 "source": { "index": "old\_index" },  
 "dest": { "index": "new\_index" }  
}

## Modifying Documents During Reindexing

The Reindex API allows modification of documents during reindexing by supplying a script. For instance, we can convert the 'product\_id' field to a string to match its new 'keyword' type.

Example script for modifying documents:

{  
 "source": {  
 "script": {  
 "source": "if (ctx.\_source.product\_id != null) { ctx.\_source.product\_id = ctx.\_source.product\_id.toString() }"  
 }  
 }  
}

## Advanced Use Cases

1. \*\*Filtering Documents\*\*: Use a query within the 'source' parameter to reindex only documents matching specific criteria, such as reviews with a rating of at least 4.0.

2. \*\*Removing Fields\*\*: Use source filtering to exclude specific fields during reindexing.

3. \*\*Renaming Fields\*\*: Use a script to rename fields, such as changing 'content' to 'comment'.

4. \*\*Conditional Reindexing\*\*: Apply advanced logic to include or exclude documents based on custom conditions.

## Considerations for Reindexing

Reindexing may involve handling version conflicts, throttling, and other advanced parameters. For large-scale reindexing tasks, refer to the Elasticsearch documentation for best practices.

## Conclusion

Reindexing is a powerful feature in Elasticsearch, enabling changes to mappings and adapting to evolving business needs. By leveraging the Reindex API, we can simplify this process and ensure data consistency and integrity.