# 

# Notes on Kibana Query Language (KQL)

## 1. Introduction to KQL

1. Before we get to exploring the **Discover app** in detail, we need to talk about the **Kibana Query Language**, abbreviated **KQL**.

2. This query language is a central part of the Discover app, but it’s used in several other apps as well.

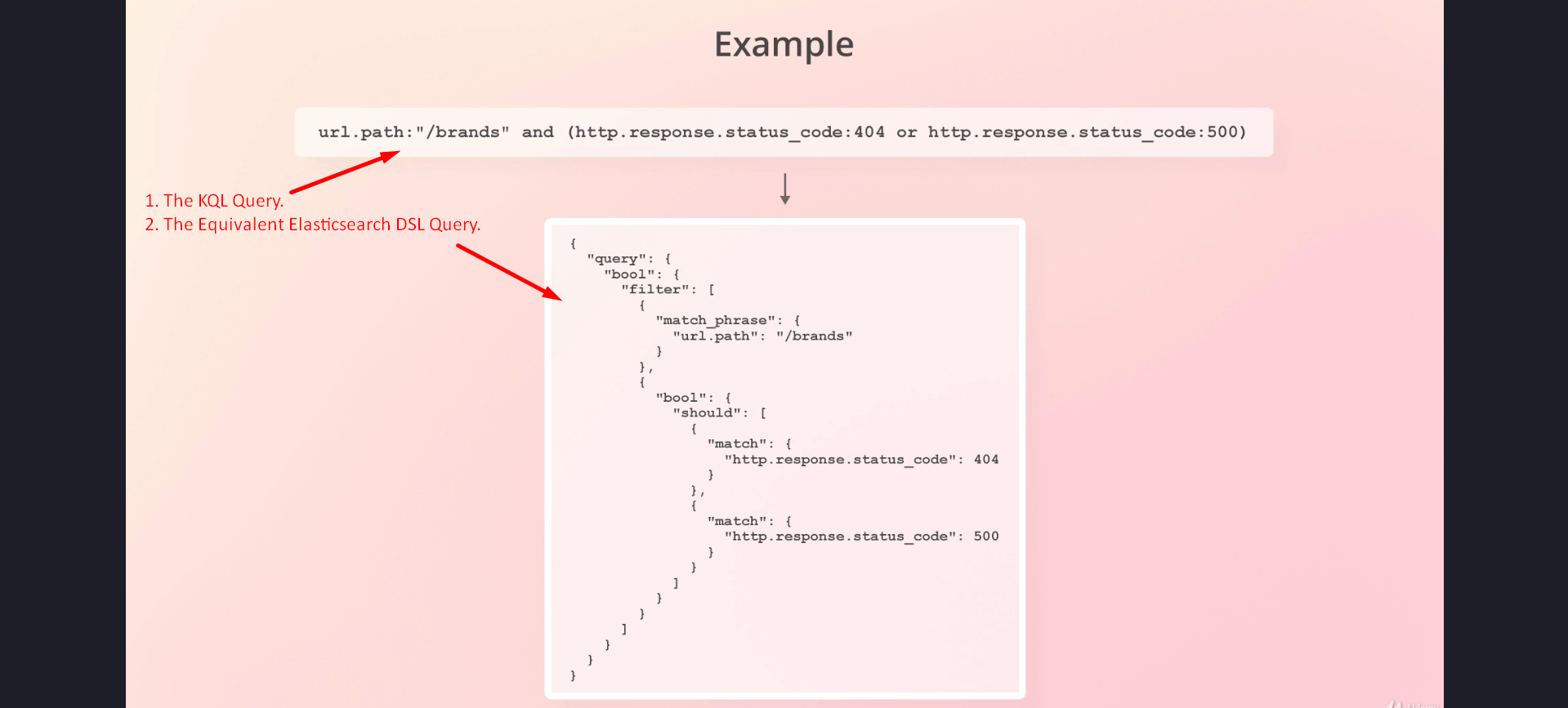
3. KQL is a query language in which you can write **short and concise search queries** to filter documents.

4. Those search queries are then automatically translated into Elasticsearch queries.

5. The purpose of KQL is to support a much more concise syntax than **Elasticsearch’s Query DSL**, which is quite verbose.

6. The Query DSL syntax is often not short, and the JSON request body would not look pretty within a text field.

7. KQL’s syntax is much shorter, but still provides enough flexibility for most queries.



## 2. Examples and Syntax of KQL

1. Let’s go through some KQL queries to get you familiar with the syntax.

2. For some of the queries, I will show you the equivalent queries for the Query DSL.

3. Note that for many of them, I have taken the liberty to rewrite the generated Query DSL.

4. The reason is that the KQL is often translated to some very verbose Query DSL, which is hard to read.

5. That’s just a small price for the flexibility that KQL provides, but I want to keep things as simple as possible for you.

## 3. Simple KQL Queries

1. Let’s begin with the simplest example of a KQL query; simply typing in the text that we want to search for.

2. In this example, we search for the terms “products” and “fischer.”

3. We could of course have searched for a single term, but I want to show you how Kibana handles a free text search including multiple terms.

4. Free text searches are translated into a multi\_match query that searches all available fields for the specified terms.

5. The type parameter is set to best\_fields, meaning that the best matching field’s relevance score is used for each document.

6. In the case of queries that contain multiple terms, the results will contain documents that match those terms, regardless of the order in which they appear.

7. This means that switching the terms around has no effect on which documents match the query.

8. If that sounds confusing, then don’t worry about it, since that’s just some Elasticsearch-specific stuff.

## 4. Phrase Searches in KQL

1. If you want to search for a phrase instead, you can surround the terms within quotation marks.

2. In that case the type of the multi\_match query is set to phrase as you can see on your screen.

3. Under the hood, a match\_phrase query is run on each field, requiring the terms to appear in the specified order.

## 5. Searching Specific Fields in KQL

1. While free text searches can be convenient sometimes, we often want to search specific fields.

2. We can do that by specifying the field name followed by an operator and a value.

3. Here is an example of searching for requests that have an HTTP status code of 404.

4. We use the colon operator — or the “equals operator” — when searching for specific values.

5. You will see which other operators are available in a moment.