**How to monitor Spring Boot Microservices using ELK Stack?**

* [java](https://javatechonline.com/java/)

* [Elasticsearch](https://javatechonline.com/elasticsearch/)

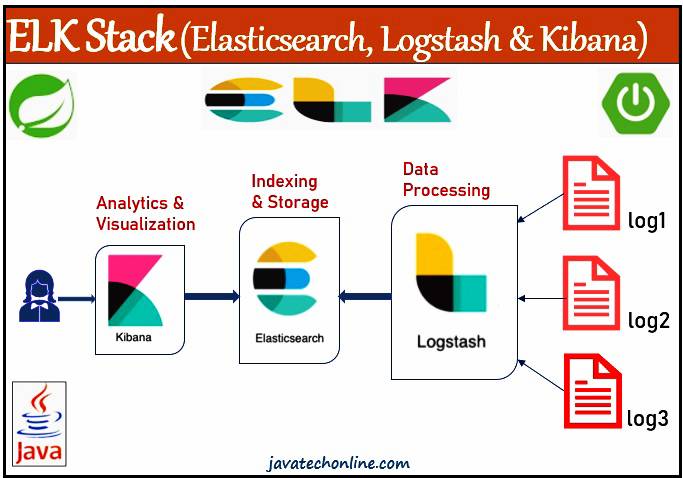
* [ELK Stack](https://javatechonline.com/elk-stack/)

* [Microservices](https://javatechonline.com/microservices/)

* [Spring Boot](https://javatechonline.com/spring-boot/)

*by*[*devs5003*](https://javatechonline.com/author/erdsingh24/)*- July 16, 2021*3

Last Updated on April 29th, 2024

While developing an application, we always incorporate a feature in our code that can direct us what went wrong if the application fails to run normally. This feature is nothing but logging. The more work you do with logging, the less work you have to do with fixing the application issue. Typically, we keep all logging information in a text format file, called a log file. This file captures all the details such as startup of a server, activities of all users, names of classes, methods, timestamp, occurred exceptions with stack trace etc. Moreover, it depends on us what all information we require in the log file.

Sometimes, these log files become larger in size and finding the exact issue manually becomes tedious. Here ELK Stack helps us in analyzing our log files at runtime. Hence, we will talk about ‘How to monitor Spring Boot Microservices using ELK Stack?’.

The term ‘ELK Stack’ is becoming more popular day by day. ELK is an acronym of a combination of three tools: Elasticsearch, Logstash and Kibana. Generally, we use all of them to monitor our application. However, each of them has a different purpose that we will discuss in below sections. ELK Stack and Splunk are the world’s most popular log management platform. Here, we will discuss about ELK Stack. Let’s start discussing our topic ‘How to monitor Spring Boot Microservices using ELK Stack?’ and its related concepts.

Table of Contents

* [Why is Monitoring of an Application Becoming More Important?](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#Why_is_Monitoring_of_an_Application_Becoming_More_Important)
* [What is ELK Stack?](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#What_is_ELK_Stack)
* [Why ELK Stack?](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#Why_ELK_Stack)
* [What is ELK Stack used for?](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#What_is_ELK_Stack_used_for)
* [How to download and install ELK Stack (Elasticsearch, Logstash and Kibana)?](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#How_to_download_and_install_ELK_Stack_Elasticsearch_Logstash_and_Kibana)
  + [1) Elasticsearch](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#1_Elasticsearch)
  + [2) Kibana](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#2_Kibana)
  + [3) Logstash](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#3_Logstash)
* [How to monitor Spring Boot Microservices using ELK Stack?](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#How_to_monitor_Spring_Boot_Microservices_using_ELK_Stack)
  + [Step#1: Create a new Spring Boot Starter Project using STS](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#Step1_Create_a_new_Spring_Boot_Starter_Project_using_STS)
  + [Step#2: Create a RestController](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#Step2_Create_a_RestController)
  + [Step#3: Update application.properties](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#Step3_Update_applicationproperties)
  + [Step#4: Create logstash.conf file](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#Step4_Create_logstashconf_file)
  + [Step#5: Run your application & ELK Stack](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#Step5_Run_your_application_ELK_Stack)
* [How to test in Kibana Dashboard?](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#How_to_test_in_Kibana_Dashboard)
* [How to search data in Kibana Dashboard?](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#How_to_search_data_in_Kibana_Dashboard)
  + [How to Switch between KQL and Lucene Syntax in Kibana?](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#How_to_Switch_between_KQL_and_Lucene_Syntax_in_Kibana)
  + [Search By Field (Lucence)](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#Search_By_Field_Lucence)
  + [Free Text (Lucence)](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#Free_Text_Lucence)
  + [Boolean Operators (Lucence) :  AND, OR,  NOT](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#Boolean_Operators_Lucence_AND_OR_NOT)
  + [Ranges (Lucence)  [ ], { }, :>, :>=, :<, :<=](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#Ranges_Lucence_%3E_%3E)
  + [Wildcards (Lucence)   \*, ?](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#Wildcards_Lucence)
  + [Regex (Lucence)   / [ ] /, / < > /](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#Regex_Lucence_%3C_%3E)
  + [Boolean Operators (KQL)   AND, OR, AND NOT](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#Boolean_Operators_KQL_AND_OR_AND_NOT)
  + [Search By Field (KQL)](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#Search_By_Field_KQL)
  + [Free Text (KQL)](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#Free_Text_KQL)
  + [Ranges (KQL)  >, >=, <, <=](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#Ranges_KQL_%3E_%3E)
  + [Wildcard(KQL)  \*](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#WildcardKQL)
* [Conclusion](https://javatechonline.com/how-to-monitor-spring-boot-microservices-using-elk-stack/#Conclusion)

**Why is Monitoring of an Application Becoming More Important?**

Any organization doesn’t want to afford a single second of downtime or slow performance of the applications. Moreover, performance issues can harm a brand name and even in some cases convert into a revenue loss. Hence, in order to ensure apps are accessible 24/7, efficient and secure at all times, developers utilize the different types of data produced by their applications and also the infrastructure supporting them. This data, generally in the form of logs, becomes important in the monitoring of these applications and the identification and resolution of any occurring issues. An organized logging plays an important role in fixing production time issues.

Before we begin discussing ‘How to monitor Spring Boot Microservices using ELK Stack?’, let’s get a basic understanding of ELK Stack.

**What is ELK Stack?**

ELK Stack is a log management platform. The word “ELK” is the acronym for three open source projects: Elasticsearch, Logstash, and Kibana all developed, managed and maintained by [Elastic](https://www.elastic.co/). Furthermore, **Elasticsearch** is a search and analytics engine, based on the Apache Lucene search engine. **Logstash** is a server‑side data processing pipeline that ingests data from multiple sources simultaneously, transforms it, and then sends it to a “stash” like Elasticsearch. **Kibana** is a visualization layer that works on top of Elasticsearch, providing users with the ability to analyze and visualize the data.

**Why ELK Stack?**

In today’s competitive world, the application’s architecture has turned into microservices, containers and orchestration infrastructure deployed in the cloud, across clouds or in hybrid environments. Moreover, the absolute volume of data produced by these environments is constantly increasing. Therefore, manual analysis of data such as log analysis is becoming a challenge in itself. This is where centralized log handling and analytics solutions such as the ELK Stack come into the picture. Hence, it offers developers to increase the visibility they need and ensure apps are available and responsive at all times.

**What is ELK Stack used for?**

The most common usage of ELK (together, three different components) are for monitoring, troubleshooting and securing IT environments. Moreover, there are many more use cases for the ELK Stack such as business intelligence and web analytics. Moreover, Logstash takes care of data collection and processing, Elasticsearch indexes and stores the data, and Kibana provides a user interface for querying the data and visualizing it.

**How to download and install ELK Stack (Elasticsearch, Logstash and Kibana)?**

In order to use ELK Stack, we have to download all three software i.e. Elasticserach, Logstash and Kibana. Below is the steps to download and install them in your system.

1) Elasticsearch

1) Go to <https://www.elastic.co/downloads/elasticsearch>  
2) Select an OS based link  
3) Extract ZIP file to a location in your system  
4) To start it, go to the bin folder and run below command, It will start on port : 9200  
> elasticsearch.bat

2) Kibana

1) Go to <https://www.elastic.co/downloads/elasticsearch>  
2) Select an OS based link  
3) Extract ZIP file to a location in your system  
4) Link Kibana with Elasticsearch : Open kibana.yml file from config/kibana.yml : uncomment below line  
elasticsearch.hosts : http://localhost:9200  
5) To start it, go to the bin folder and run below command, It will start on port : 5601  
> kibana.bat

3) Logstash

1) Go to <https://www.elastic.co/downloads/logstash>  
2) Select an OS based link  
3) Extract ZIP file to a location in your system  
4) Go to bin folder and create one file ‘logstash.conf’ with some configuration. Some examples of this file are given in below link.  
<https://www.elastic.co/guide/en/logstash/current/config-examples.html>  
5) To start it, go to the bin folder and run below command  
> logstash -f logstash.conf

**How to monitor Spring Boot Microservices using ELK Stack?**

Now, It’s time to create a Spring Boot application and integrate it with ELK Stack. However, it doesn’t matter whether you are working on a Microsevices based application or a Simple Spring Boot application. Here, our focus should be to create log files and the content of the log files will be captured by logstash. We can even create a Simple Java application that creates a log file. Any ways, the process of integration will generally be the same. Let’s create a Spring Boot application and integrate it with ELK Stack step by step.

Step#1: Create a new Spring Boot Starter Project using STS

Let’s create a Spring Boot Starter project using STS. While creating Starter Project select ‘Spring Web’, and ‘Sprong Boot DevTools’ as starter project dependencies. Even If you don’t know how to create a Spring Boot Starter Project, Kindly visit our [Internal Link](https://javatechonline.com/saving-data-into-database-using-spring-boot-data-jpa-step-by-step-tutorial/#Step_1_Creating_Starter_Project_using_STS).

Step#2: Create a RestController

Create a RestControlller as InvoiceController and write some methods that generate an ample amount of log messages to the log file as below.

**import java.io.PrintWriter;**

**import java.io.StringWriter;**

**import org.slf4j.Logger;**

**import org.slf4j.LoggerFactory;**

**import org.springframework.web.bind.annotation.GetMapping;**

**import org.springframework.web.bind.annotation.RequestMapping;**

**import org.springframework.web.bind.annotation.RestController;**

**@RestController**

**@RequestMapping("/invoice")**

**public class InvoiceController {**

**private static final Logger *logger*= LoggerFactory.getLogger(InvoiceController.class);**

**@GetMapping("/get")**

**public String getInvoice() {**

***logger*.info("Entering into method getInvoice()");**

**try {**

***logger*.info("finding Invices");**

**throw new RuntimeException("Invoice not available");**

**} catch (Exception e) {**

**logger.error(" Unable to find invoice" +e.getMessage());**

**e.printStackTrace();**

**StringWriter sw= new StringWriter();**

**PrintWriter pw= new PrintWriter(sw);**

**e.printStackTrace(pw);**

***logger*.error("Exception is -: " +sw.toString());**

**}**

**return "INVOICE";**

**}**

**}**

Step#3: Update application.properties

Update application.properties and provide the location of log file as below

**logging.file.name=D:/ELK\_Stack/elktest.log**

Step#4: Create logstash.conf file

In this step, we will create a new logstash.conf file at the bin folder of your logstash installtion. For example, in our case, the location is ‘D:\ELK\_Stack\logstash-7.13.3\bin’. We have created a sample file for java logs as below.

It generally contains three parts : input, filter, and output

**1)** **input** : indicates where to read from  
**2)** **filter** : indicates what to filter  
**3)** **output** : indicates how to provide output

**input {**

**file {**

**type => "java"**

**path => "D:/ELK\_Stack/elktest.log"**

**codec => multiline {**

**pattern => "^%{YEAR}-%{MONTHNUM}-%{MONTHDAY} %{TIME}.\*"**

**negate => "true"**

**what => "previous"**

**}**

**}**

**}**

**filter {**

**if [message] =~ "\tat" {**

**grok {**

**match => ["message", "^(\tat)"]**

**add\_tag => ["stacktrace"]**

**}**

**}**

**}**

**output {**

**stdout {**

**codec => rubydebug**

**}**

**elasticsearch {**

**hosts => ["localhost:9200"]**

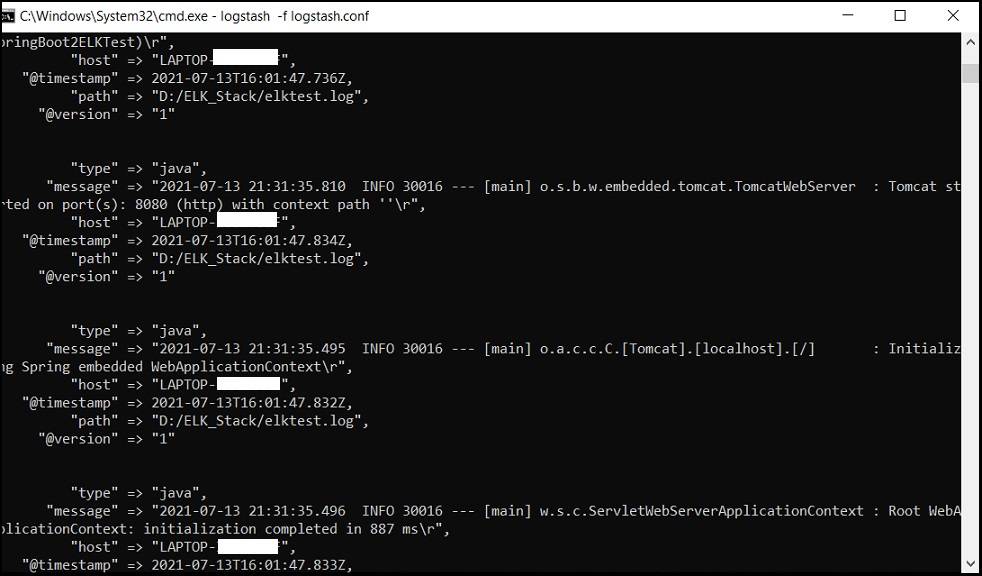
**}**

**}**

Step#5: Run your application & ELK Stack

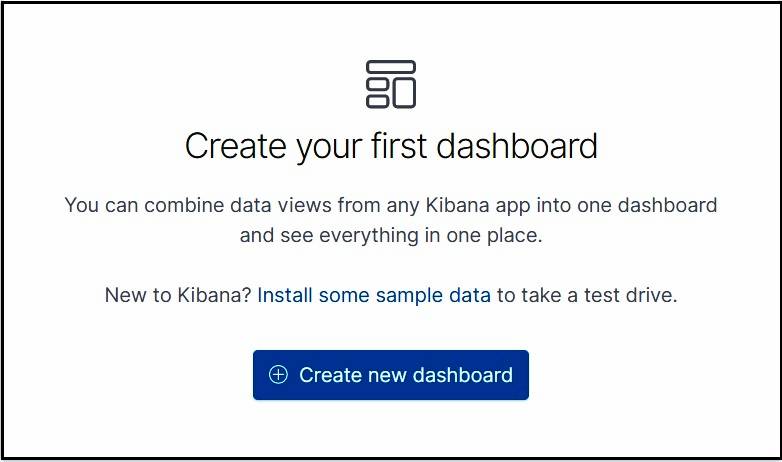
1) Run your Spring Boot Application  
2) Run Elasticsearch : Go to bin folder and Use below command  
> elasticsearch.bat  
3) Run Kibana : Go to bin folder and Use below command  
> kibana.bat  
4) Run Logstash : Go to bin folder and Use below command  
> logstash -f logstash.conf

Once you start the logstash, it will start parsing the log file and show like below traces.

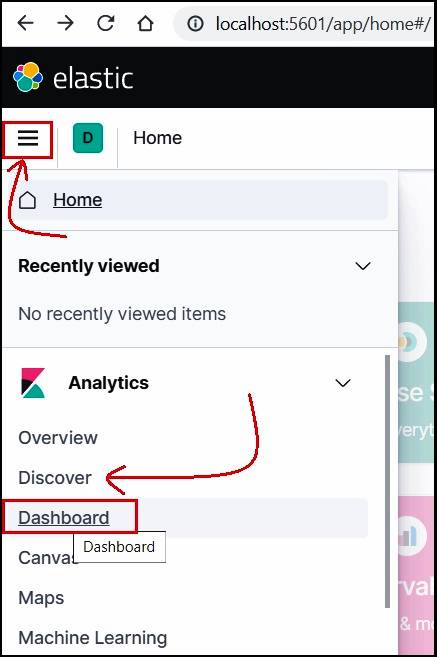


**How to test in Kibana Dashboard?**

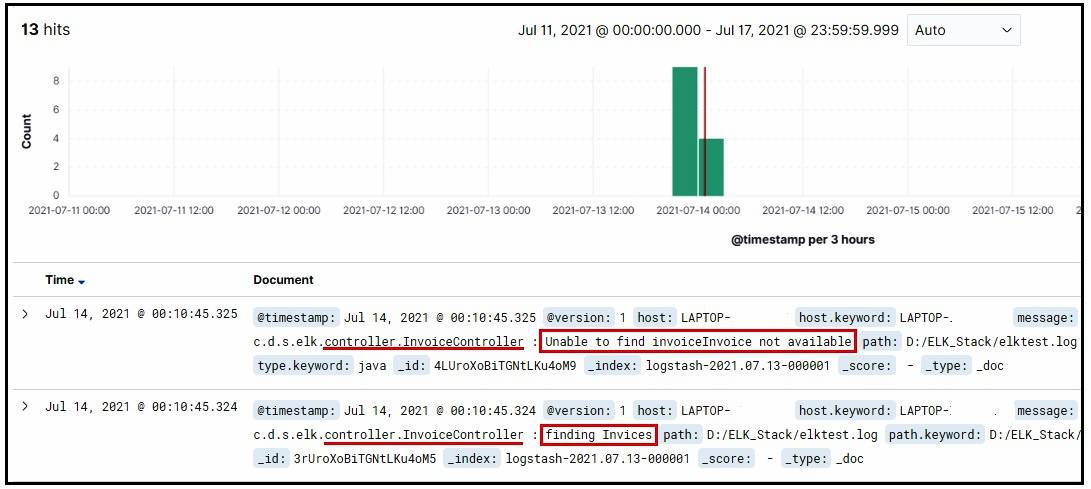
1) Go to Kibana UI : Open browser, and hit http://localhost:5601  
2) Click on the Dashboard, and then click on ‘Create new Dashboard’. Please refer screenshots attached below.  
3) Click on the ‘Create Index Pattern’ to provide one search index pattern  
4) Enter some pattern in Index pattern name field such as ‘logstash-\*’ and click on ‘Next step’ button  
5) In the time field let’s select ‘@timestamp’ and then click on ‘Create index pattern’  
6) Now click on the left bar and select ‘Discover’, you will get the data populated in the Dashboard.



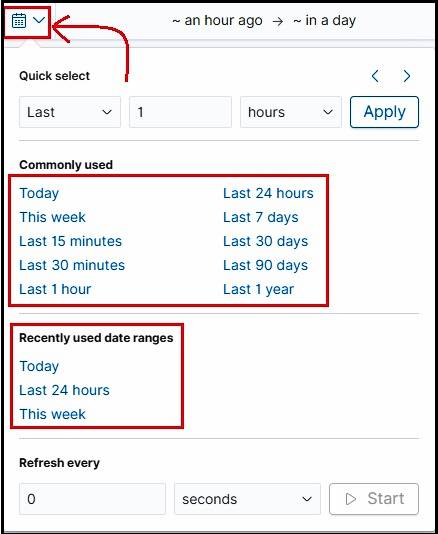
Below screenshot shows the place where you can go to ‘Dashboard’ and ‘Discover’.



Once you click on ‘Discover’, below results will apear in the Kibana UI.



We can also set the time duration as per our requirement as below.



**How to search data in Kibana Dashboard?**

Let’s discuss about some of the queries that we require while searching the results from the Kibana UI. As aforementioned, Elasticsearch is a search and analytics engine, based on the Apache Lucene search engine. It is completely open source and built with Java. In fact, Elasticsearch is classified as a NoSQL database. It means Elasticsearch stores data in an unstructured way. Hence, you could not query the data using SQL. The new Elasticsearch SQL project will allow using SQL statements to interact with the data. Being aware with the syntax and its variety of operators, will be helping you to query in Kibana UI.

We have two different ways of querying data in Kibana: either use the traditional **Lucene Query Syntax** or the most recent **KQL (Kibana Query Language)**. If you are using Kibana 7.0 or later, Kibana Query Language is included as a default. However, we will discuss the basics for both approaches including examples. One language may be better for your requirement than another. It totally depends on the nature of a search or your individual experience. However, KQL has some limitations such as not supporting fuzzy or regex searches. Moreover, we may expect Elastic team to concentrate on expanding KQL in the future releases.

How to Switch between KQL and Lucene Syntax in Kibana?

Clicking on the square on the right end of the search bar in Kibana. It will either read KQL or Lucene depending on which is activated. Once clicked, you can toggle the Kibana Query Language button either on or off. If it is in the off state, KQL is activated. Similarly, If it is in the off state, Lucence Syntax is activated.

Search By Field (Lucence)

Querying with field names is the most popular way of filtering data from Elasticsearch. You might be searching for a specific field that contains specific terms. Then you can do it like below:

**name:”Specific term”    Example⇒ message: ERROR**

The query above indicates that you are searching the term ‘ERROR’ in the message field. It will return the results that have ERROR in the message field.

Free Text (Lucence)

The simplest form of querying data, just like a Google search.

**Invoice** ⇒ returns results that include “Invoice” in any field

**“Invoice not Found”** ⇒ returns results that include “Invoice not Found” in any field

Boolean Operators (Lucence) :  AND, OR,  NOT

Like other programming languages, Elasticsearch also supports the OR, AND and NOT operators. Also, the meaning of these operators are same as any other programming language. Operators such as AND, OR, and NOT must be capitalized.

**♦ Invoice AND Found** ⇒ Will return results that contain both the terms Invoice and Found

**♦ Error NOT Warning** ⇒ Will return results that contain Error but not Warning

**♦ Exception OR Error** ⇒ Will return results that contain Exception or Error, or both

Ranges (Lucence)  [ ], { }, :>, :>=, :<, :<=

Lucence supports multiple types of Range searches : [ ], { }, :>, :>=, :<, :<=

* **price:[2 TO 24] ⇒** Will return results with price from 2 through 24, including 2 and 24
* **price:{2 TO 12} ⇒** Will return results with any price from 2 through 12
* **price:>2 ⇒**Will return results with any price greater than 2

Wildcards (Lucence)   \*, ?

* **pr\*** ⇒ will return results that include values that start with “pr”, such as price and protocol
* **pr\*e** ⇒ will return results that include values that start with “pr” and end in “e”, such as price and prime
* **stat?s** ⇒ will return results that include values that start with “stat”, end in “s”, and have one character in between, such as status

Regex (Lucence)   / [ ] /, / < > /

* **/pri[mc]e/** ⇒ will return results that include either prime or price

Kibana Query Language (KQL) was first introduced in version 6.3 and became available as a default starting with version 7.0. However, this new language was built to provide scripted field support and to simplify the syntax compared to the Lucene language discussed above.

Boolean Operators (KQL)   AND, OR, AND NOT

Unlike Lucence Syntax, KQL Boolean operators are not case-sensitive. We can use ‘and’, ‘or’, ‘and not’ in place of ‘AND’, ‘OR’, ‘AND NOT’ respectively. Also, ‘NOT’ is replaced by ‘AND NOT’.

**Exception AND NOT Error** ⇒ returns results that only include Exception, but not those results that include both Exception and Error

**Exception and not Error** ⇒ returns results that only include Exception, but not those results that include both Exception and Error

**By default, ‘AND*‘* will have higher precedence than ‘OR’. Parentheses can be used to override this default.**

**Exception and (Error or Warning)** ⇒ returns results that include Exception and either Error or Warning

**If we use *‘*NOT*‘* before a search term, it will revert its meaning.**

**not status:”on Hold”** ⇒ returns results that do not have on hold listed as their status

**We can also revert entire groups by using parentheses.**

**not(name:Michael or location:”Washington DC”)** ⇒ returns results that do not have Michael as the name or Washington DC as the location

Search By Field (KQL)

**message: Error** ⇒ returns results that have Error in the message field

**message: “Invoice Unavailable”** ⇒ returns results that have 'Invoice Unavailable' in the message field. Here, the value “Invoice Unavailable” is in quotes so that the search includes the words Invoice and Unavailable in the given order. Without the quotes, the results would also include Unavailable Invoice.

Searching a single field for multiple values is also possible in KQL as below.

**message: (“Found” or “Not Found”)** ⇒ returns results that have either Found or Not Found listed as the message

**location: (Chicago and New York and London)** ⇒ returns results that have all three Chicago, New York, and London listed as locations

Free Text (KQL)

The simplest form of querying data, just like a Google search and Lucence Syntax.

**Invoice** ⇒ returns results that include “Invoice” in any field

**“Invoice not Found”** ⇒ returns results that include “Invoice not Found” in any field

Ranges (KQL)  >, >=, <, <=

**Unlike Lucence Syntax, the colons are eliminated before the greater than, less than, etc. signs for searches.**

**price>2** ⇒ Will return results with any price greater than 2

Wildcard(KQL)  \*

The KQL supports \* wildcard.

However, we can use wildcards to search text and keyword versions of a field concurrently.

**host\*:localhost** ⇒ returns results that have localhost for both the host and host.keyword fields

**Conclusion**

After going through all the theoretical & example part of ‘How to monitor Spring Boot Microservices using ELK Stack?’, finally, we should be able to integrate ELK Stack with a Spring Boot Project or Microservices. Similarly, we expect from you to further extend these examples and implement them in your project accordingly. I hope you might be convinced with the article ‘How to monitor Spring Boot Microservices using ELK Stack?’. In addition, If there is any update in the future, we will also update the article accordingly. Moreover, Feel free to provide your comments in the comments section below.