**How To Monitor Spring Boot Microservices Using ELK Stack?**

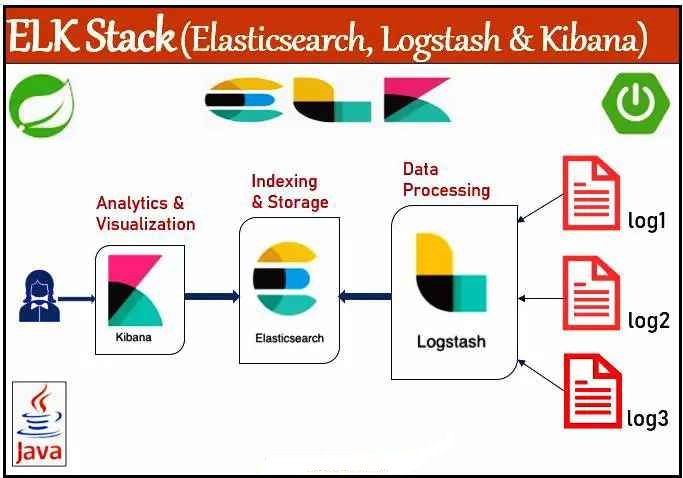
While developing an application, we always adding a feature in our code that can tells us what went wrong if the application execution fails. This feature is nothing but logging.

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, all user activities, names of classes, methods, timestamp, occurred exceptions with stack trace etc.

Sometimes, these log files become larger in size and finding the exact issue manually becomes difficulty. Here ELK Stack helps us in analysing our log files at runtime.

ELK is an acronym of a combination of three tools:

* Elasticsearch
* Logstash
* Kibana



## **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.

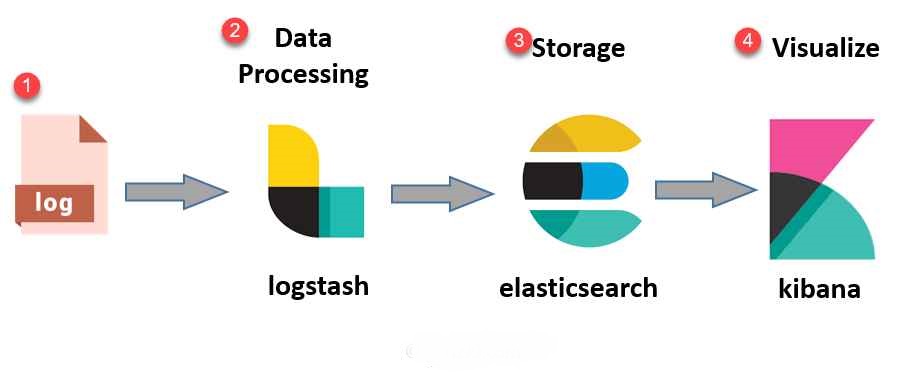
An organized logging plays an important role in fixing production time issues.

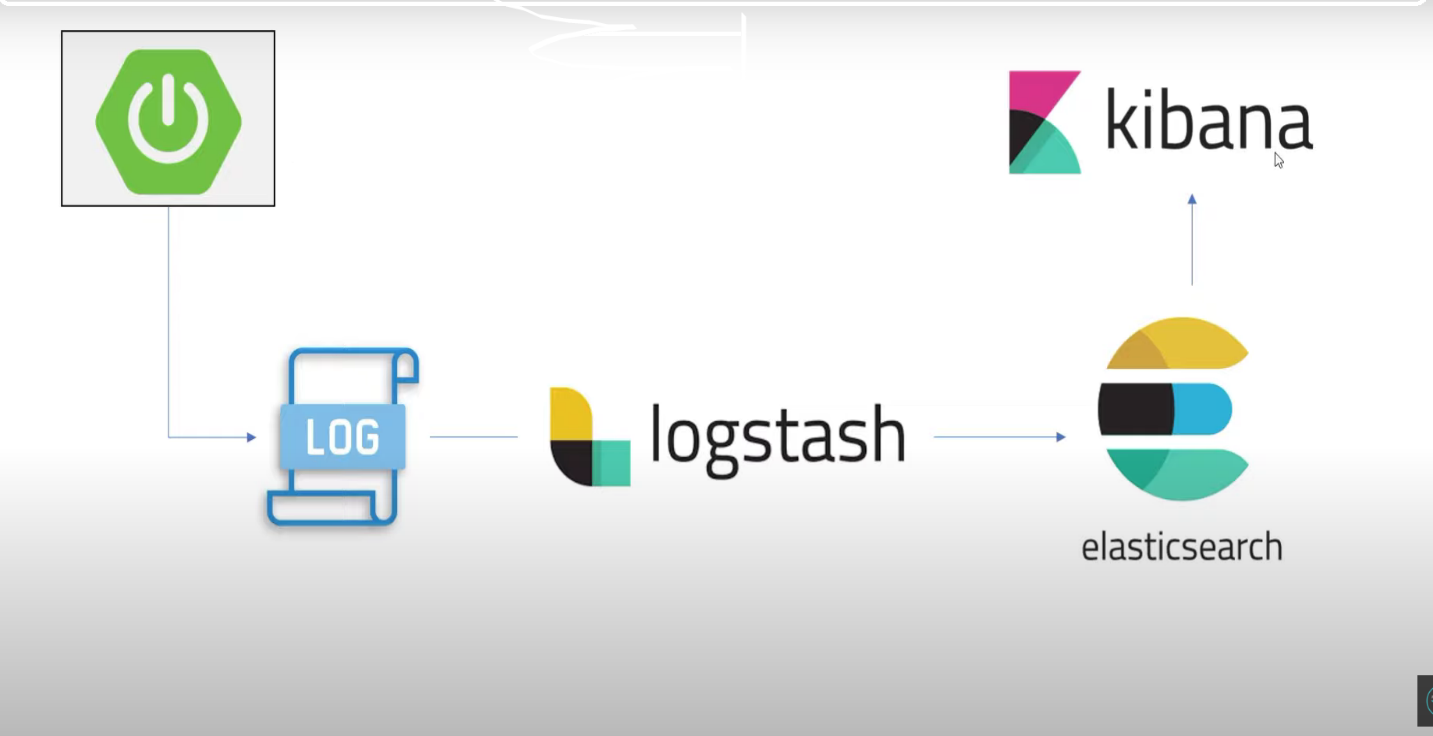
## What is ELK Stack?

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/).

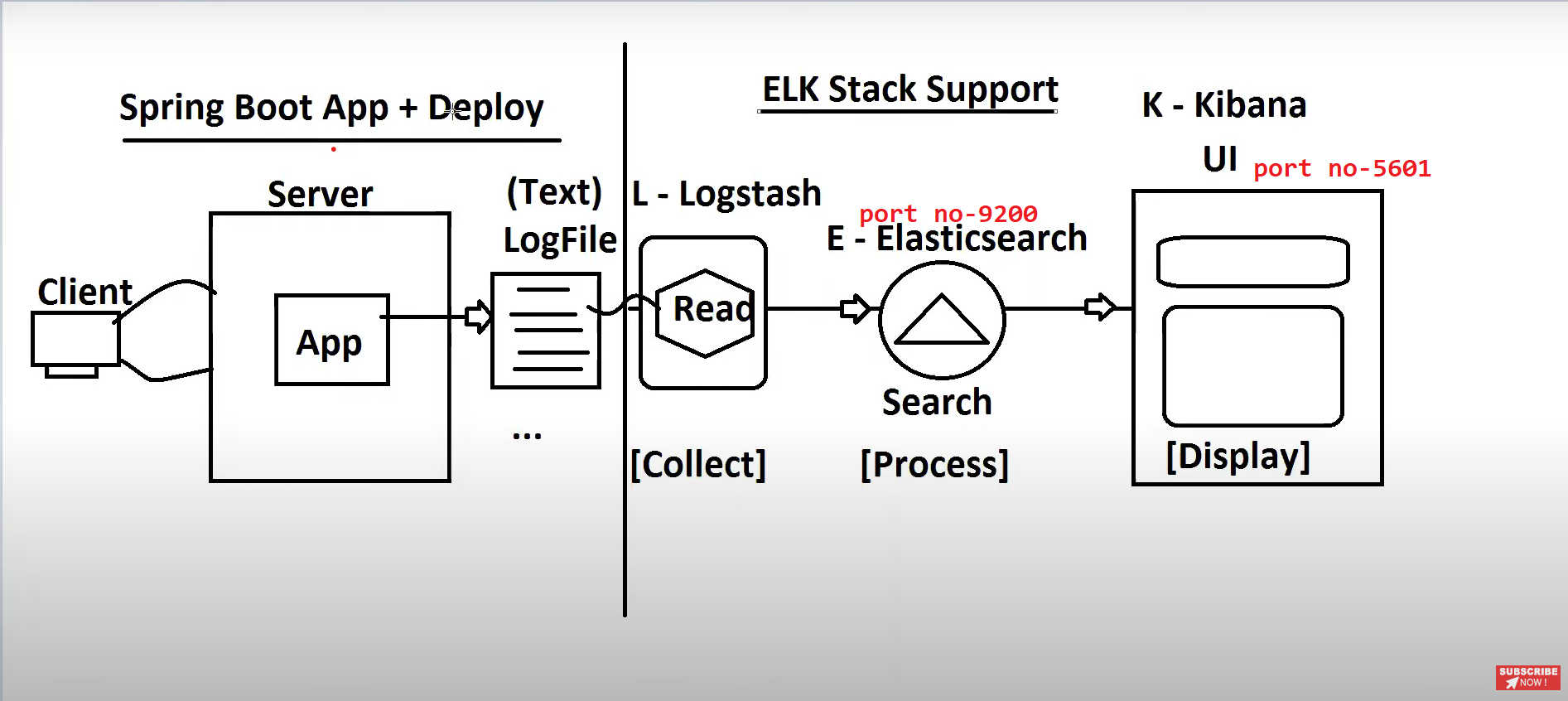
**Elastic Search(E)** :- Elasticsearch is a NoSQL database that is based on the Lucene search engine which will helps us to store inputs/logs **[Storing]**

**Logstash(L)** :- Logstash is a log pipeline tool that accepts input/logs from various sources & exports the data to various targets. **[Collecting]**

**Kibana(K)** :- Kibana is a visualization UI layer, which helps developer to monitor application logs. **[Display/ UI Layer]**

****

**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) Open **“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) Open “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.

### **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.

|  |
| --- |
| InvoiceController.java |
| 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 {**

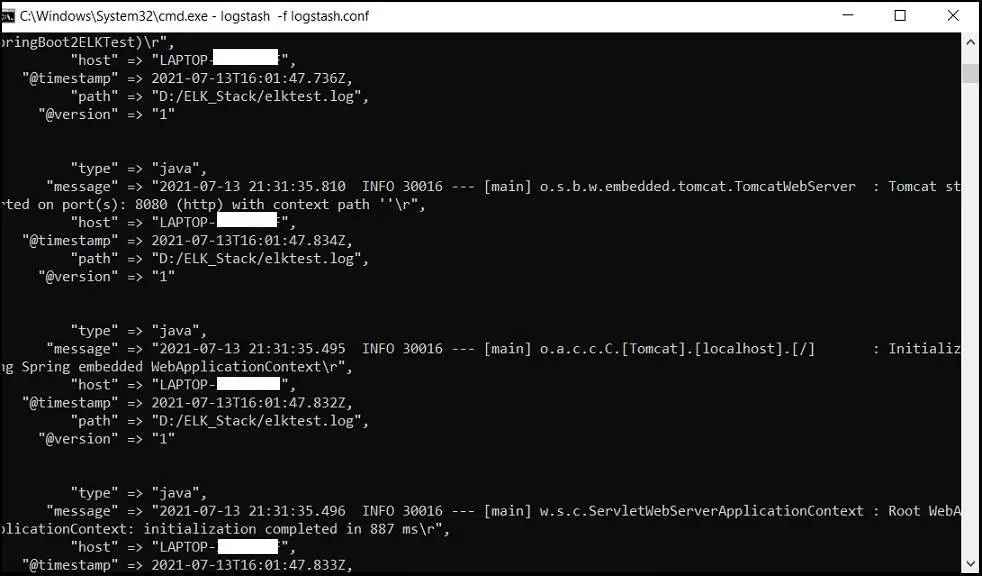
**hosts => ["localhost:9200"]**

**}**

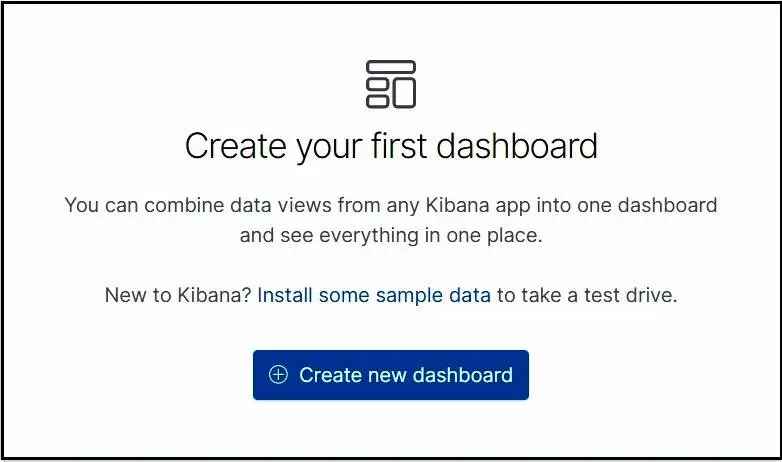
**}**

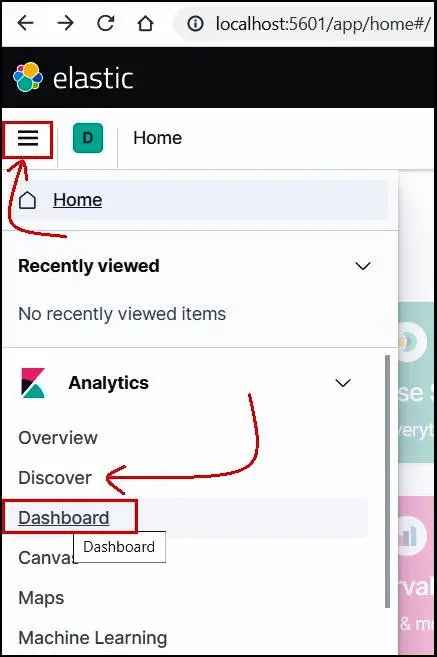
### **Step#5: Run your application & ELK Stack**

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.

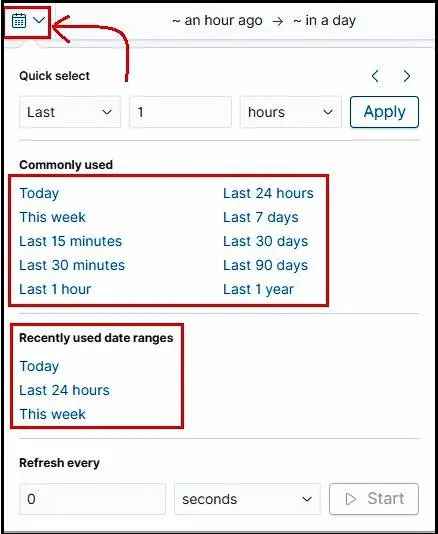
## **How to test in Kibana Dashboard?**

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