SPRING BOOT + ELK

ELK stands for Elasticsearch, Logstash and Kibana.

- **Elasticsearch** is a NoSQL database that is based on the Lucene search engine.
- **Logstash** is a log pipeline tool that accepts inputs from various sources, executes different transformations, and exports the data to various targets. It is a dynamic data collection pipeline with an extensible plugin ecosystem and strong Elasticsearch synergy.
- **Kibana** is a visualization UI layer that works on top of Elasticsearch.

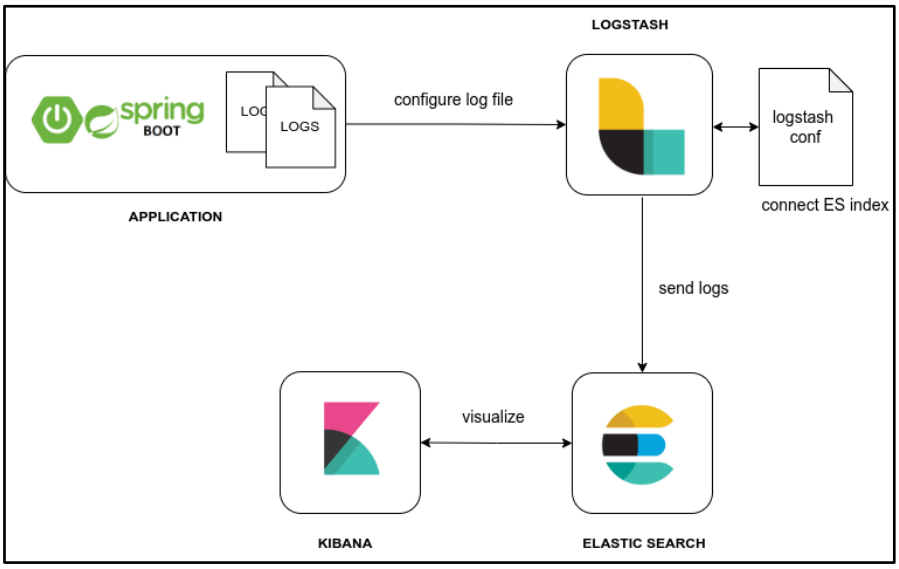
We can integrate this stack to any back-end application to achieve **centralized logging**. We can use ELK to analyze the logs more efficiently and also using more complex search scenarios. It provides log aggregation and efficient searching.

- Download ELK stack from these links.

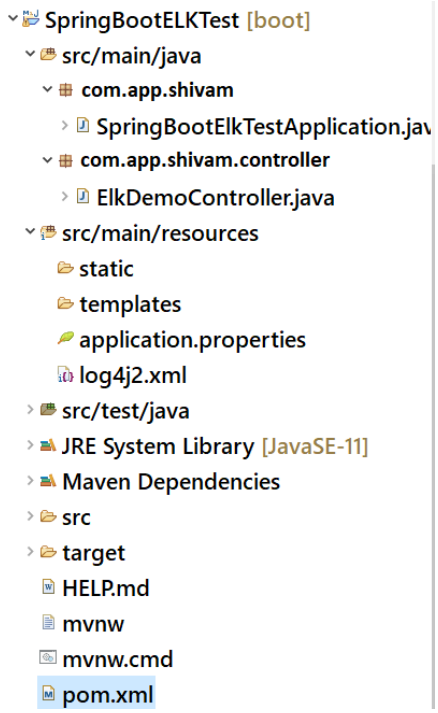
  — **ES**: https://www.elastic.co/downloads/elasticsearch

  — **Logstash**: https://www.elastic.co/downloads/logstash

  — **Kibana**: https://www.elastic.co/downloads/kibana
- Extract each into separate folders.
- **Recommended version is ELK stack version as 7.14.1**
- **If you need past releases of ELK stack, you can find them here:** https://www.elastic.co/downloads/past-releases



Spring Boot application:



Controller code:

```
package com.app.shivam.controller;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class ElkDemoController {

	private static final Logger logger = LogManager.getLogger(ElkDemoController.class);

	@GetMapping(path = "/welcome")
	public ResponseEntity<String> welcome() {
		logger.debug("Welcome to ELK demo service");
		return ResponseEntity.ok("Hello ELK Integration!!!");
	}

	@GetMapping(path = "/users/{name}")
	public ResponseEntity<String> getUserByName(@PathVariable String name) {
		if (name.equals("ADMIN")) {
			logger.debug("Access by ADMIN triggered");
			return ResponseEntity.ok("Access Granted to " + name);
		} else {
			logger.error("Access denied for: " + name);
			return new ResponseEntity<>("Access Denied for " + name,
HttpStatus.BAD_REQUEST);
		}
	}

}
```

Logging Configuration Code: -

log4j2.xml: -

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="DEBUG">
	<Properties>
		<Property name="LOG_PATTERN">
	    [%-5level] %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %c{1} - %msg%n
	</Property>
		<Property name="BASE_PATH">D:/elk/logs</Property>
	</Properties>
	<Appenders>
		<Console name="ConsoleAppender" target="SYSTEM_OUT" follow="true">
			<PatternLayout pattern="${LOG_PATTERN}" />
		</Console>
		<RollingFile name="FileAppender" fileName="${BASE_PATH}/elkdemo.log"
filePattern="${BASE_PATH}/elkdemo-%d{yyyy-MM-dd}.log">
			<PatternLayout>
				<Pattern>${LOG_PATTERN}</Pattern>
			</PatternLayout>
			<Policies>
				<SizeBasedTriggeringPolicy size="10MB" />
			</Policies>
			<DefaultRolloverStrategy max="10" />
		</RollingFile>
	</Appenders>
	<Loggers>
		<Logger name="com.app.shivam" level="DEBUG" additivity="false">
			<AppenderRef ref="FileAppender" />
			<AppenderRef ref="ConsoleAppender" />
		</Logger>
		<Root level="DEBUG">
			<AppenderRef ref="FileAppender" />
		</Root>
	</Loggers>
</Configuration>
```

pom.xml: -

```
<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
			<exclusions>
				<exclusion>
					<groupId>org.springframework.boot</groupId>
					<artifactId>spring-boot-starter-logging</artifactId>
				</exclusion>
			</exclusions>
		</dependency>

		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-log4j2</artifactId>
		</dependency>
```

Logging Configuration:
Create a **log4j2.xml** file inside resources folder. This includes log patterns, how log files should be created, how console logs should be shown and many more information.

RollingFile is the type of log file. It means we have a constraint on the log file to rotate log file creation. I have added size based constraint: when a log file is greater than 10MB, new file is created. File name format will be *elkdemo-<date>.log*.

Execution order:

1. Start Spring Boot application and Enter URL to Test application

http://localhost:8080/welcome

http://localhost:8080/users/sam

2. Start Elasticsearch

D:\ELK\elasticsearch-7.14.1\bin> elasticsearch.bat

3. Start kibana

Open File : **D:\ELK\kibana-7.14.1-windows-x86_64\config\kibana.yml** and un-comment line

elasticsearch.hosts: ["http://localhost:9200"]

then run below batch file

D:\ELK\kibana-7.14.1-windows-x86_64\bin> kibana.bat

http://localhost:5601/

4. Configure and Start Logstash

Create a file : **./config/logstash.conf**

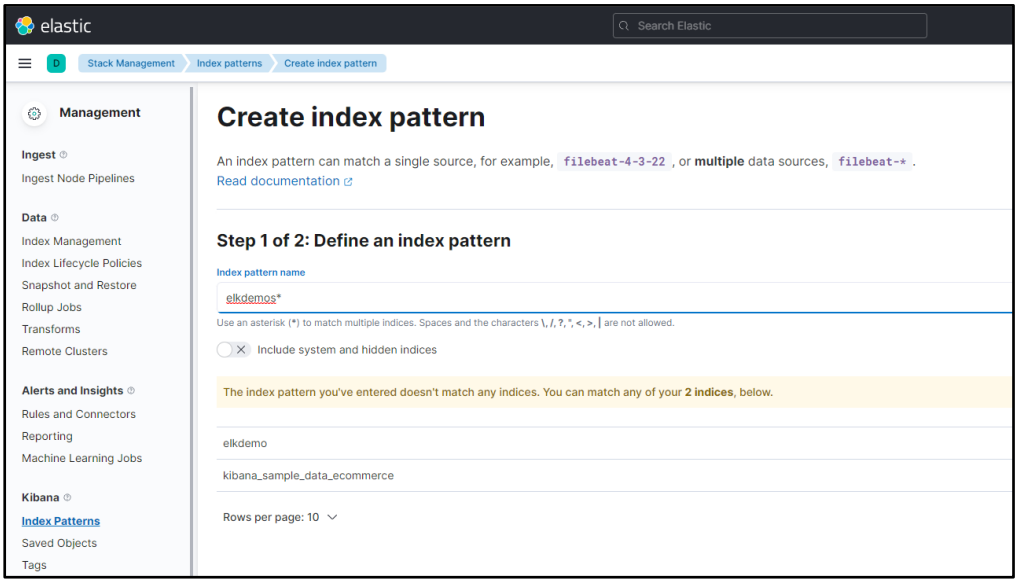
```
input {
  file {
    path => "D:/elk/logs/elkdemo.log"
    start_position => "beginning"
  }
}
output {
  stdout {
    codec => rubydebug
  }
  elasticsearch {
    hosts => ["localhost:9200"]
    index => "elkdemo"
  }
}
```

D:\ELK\logstash-7.14.1\bin> logstash.bat –f ../config/logstash.conf

5. Configure Index Pattern in kibana

Create Index Pattern:

Go to **Management > Stack Management > Kibana > Index Patterns** page from left side bar in Kibana UI



Go to “Discover” page from menu. In the left side, you will get a drop down to select the pattern. Select **elkdemo pattern** from there

Create a Filter to search: