**Hands-On**

**Springboot Application**

I’ll start with the application; I created a REST API with 4 endpoints: This is how the application’s Dockerfile was:

FROM openjdk:17.0.1

WORKDIR /app

COPY .mvn/ .mvn

COPY mvnw pom.xml ./

RUN ./mvnw dependency:go-offline

COPY src ./src

CMD ["./mvnw", "spring-boot:run"]

EXPOSE 8080

Copy

Alternatively, you can use a Dockerfile like this:

FROM openjdk:17.0.1

WORKDIR /app

COPY ./target/\*.jar ./app.jar

ENTRYPOINT ["java", "-jar", "/app/app.jar"]

EXPOSE 8080

As we are going to use several containers, we are going to use docker compose.

At the root of your project, you will create the docker-compose.yml file. For now, docker-compose.yml will look like this.

version: '3.2'

services:

app:

container\_name: library\_app

build:

context: .

ports: - "8080:8080"

Copy

For now, docker-compose.yml will look like this.

To test, run from the root of the project: docker-compose up

Test by calling the endpoints via Postman, browser, or others of your choice.

**Elasticsearch**

I won’t go into details about Elasticsearch

elasticsearch:

container\_name: library\_elasticsearch

image: docker.elastic.co/elasticsearch/elasticsearch:7.15.2

volumes:

- type: volume

source: elasticsearch

target: /usr/share/elasticsearch/data

ports: - "9200:9200" - "9300:9300"

environment:

ES\_JAVA\_OPTS: "-Xmx256m -Xms256m"

ELASTIC\_PASSWORD: changeme

discovery.type: single-node

networks: - elk

networks:

elk:

driver: bridge

volumes:

elasticsearch

Here we are setting Elasticsearch settings like port, memory variables, base directory in docker, etc. We also created a network called elk and added our service to it. To test, run: docker-compose up, open this URL [http://localhost:9200/](http://localhost:9200/?ref=faun), and you will have a similar result to this:

This shows us that Elasticsearch is working correctly.

**Logstash**

For Logstash, the process is a little different: first let’s create a folder called .logstash, to store some settings. Inside it, we will create the logstash.conf file, which will have the following information:

input {

tcp {

mode => "server"

port => 4560

codec => json\_lines

}

file {

type => "java"

path => "/var/log/logs/library/application.log"

codec => multiline {

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

negate => "true"

what => "previous"

}

}

}

output {

stdout {

codec => rubydebug

}

elasticsearch {

index => "library-logstash-%{+YYYY.MM.dd}"

hosts => "elasticsearch:9200"

user => "elastic"

password => "changeme"

ecs\_compatibility => disabled

}

}

Here we will have the operating mode, which can be TCP or per file. In TCP mode, logstash will get real-time data from the port specified in the logback-spring.xml file, inside the project in the resources package.

logback-spring.xml file:

<?xml version="1.0" encoding="UTF-8"?>

<configuration>

<include resource="org/springframework/boot/logging/logback/defaults.xml"/>

<include resource="org/springframework/boot/logging/logback/base.xml"/>

<springProperty scope="context" name="appName" source="spring.application.name"/>

<property name="LOG\_FILE" value="${BUILD\_FOLDER:-build}/${appName}"/>

<property name="CONSOLE\_LOG\_PATTERN"

value="%clr(%d{yyyy-MM-dd HH:mm:ss.SSS}){faint} %clr(${LOG\_LEVEL\_PATTERN:-%5p}) %clr(${PID:- }){magenta} %clr(---){faint} %clr([%15.15t]){faint} %m%n${LOG\_EXCEPTION\_CONVERSION\_WORD:-%wEx}}"/><appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">

<filter class="ch.qos.logback.classic.filter.ThresholdFilter">

<level>INFO</level>

</filter>

<encoder>

<pattern>${CONSOLE\_LOG\_PATTERN}</pattern>

<charset>utf8</charset>

</encoder>

</appender><appender name="LOGSTASH" class="net.logstash.logback.appender.LogstashTcpSocketAppender">

<destination>logstash:4560</destination>

<encoder charset="UTF-8" class="net.logstash.logback.encoder.LoggingEventCompositeJsonEncoder">

<providers>

<timestamp>

<timeZone>UTC</timeZone>

</timestamp>

<pattern>

<pattern>

{

"logLevel": "%level",

"serviceName": "${springAppName:-}",

"pid": "${PID:-}",

"thread": "%thread",

"class": "%logger{40}",

"rest": "%message"

}

</pattern>

</pattern>

</providers>

</encoder><encoder class="net.logstash.logback.encoder.LogstashEncoder">

<includeCallerData>true</includeCallerData>

</encoder>

</appender><appender name="STASH" class="ch.qos.logback.core.rolling.RollingFileAppender">

<file>logback/redditApp.log</file>

<rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

<fileNamePattern>logback/redditApp.%d{yyyy-MM-dd}.log</fileNamePattern>

<maxHistory>7</maxHistory>

</rollingPolicy>

<encoder class="net.logstash.logback.encoder.LogstashEncoder"/>

</appender><root level="INFO">

<appender-ref ref="LOGSTASH"/>

<appender-ref ref="CONSOLE"/>

<appender-ref ref="FILE"/>

<appender-ref ref="STASH"/>

</root>

</configuration><include resource="org/springframework/boot/logging/logback/base.xml"/><springProperty scope="context" name="appName" source="spring.application.name"/>

It also contains the output which, in our case, will be for Elasticsearch, as described below. In the output information, we define the destination, elastic user and password (in this case it is with the default values), and index. The index will serve to filter the information only from this application in kibana. In docker-compose.yml, the container will look like this:

logstash:

container\_name: library\_logstash

image: docker.elastic.co/logstash/logstash:7.15.2

volumes:

- type: bind

source: .logstash

target: /usr/share/logstash/pipeline

read\_only: true

ports:

- "5044:5044"

- "5000:5000/tcp"

- "5000:5000/udp"

- "9600:9600"

- "4560:4560"

environment:

LS\_JAVA\_OPTS: "-Xmx256m -Xms256m"

networks:

- elk

depends\_on:

- elasticsearch

**Kibana**

Kibana has its simple configuration, just add to docker-compose.yml:

kibana:

container\_name: library\_kibana

image: docker.elastic.co/kibana/kibana:7.15.2

ports:

- "5601:5601"

networks:

- elk

depends\_on:

- elasticsearch

Copy

After that, it is now possible to access kibana through the browser at URL: [http://localhost:5601/](http://localhost:5601/?ref=faun). When using kibana, you will need to add the index we created earlier in logstash to get the information. To do this, access the kibana and enter: [http://localhost:5601/app/management/kibana/indexPatterns](http://localhost:5601/app/management/kibana/indexPatterns?ref=faun) Right menu/ Stack Management/ Index Patterns/ Create index pattern. As an example, the application we cited as an example was: library-logstash-\*.