LAB- ELK Integration

Step 1 – Starting ELK Docker Container

1) There is a docker image already with name sebp/elk . Visit https://hub.docker.com/r/sebp/elk/

2) For this docker container to run , we need to set vm.max_map_count to ateleast 262144 .

so, do the following :

docker-machine ssh

you will be connected to the docker vm machine.

Execute sudo sysctl vm.max_map_count=262144

3) pull the docker image using **docker pull sebp/elk**

4) Run the elk container using

```
docker run -p 5601:5601 -p 9200:9200 -p 5044:5044 -p 5000:5000 -it --name elk sebp/elk
```

This command publishes the following ports, which are needed for proper operation of the ELK stack:

5601 (Kibana web interface).
9200 (Elasticsearch JSON interface).
5044 (Logstash Beats interface, receives logs from Beats such as Filebeat)

We will be using 5000 port for sending logs to logstash from our application later

5) You can now access kibana at http://192.168.99.100:5601

Following are optional . If you have more time, you can follow. Otherwise go to STEP2.

6) Creating dummy log entries using logstash

First get the elk container id by executing the following command :

docker ps

```
docker exec -it <container-id> bash
```

At the command prompt , execute the following

```
cd /opt/logstash/bin

./ logstash   -e 'input { stdin { } } output { elasticsearch { hosts => ["localhost"] } }'
```

Wait for Logstash to start (as indicated by the message Logstash startup completed), then type some dummy text followed by Enter to create a log entry

Create 10 dummy entries

7) You Can search index elasticsearch using below URL:

http://198.168.99.100:9200/_search?pretty

8) You   can now visit kibana web interface   at http://192.168.99.100:5601

9) Make sure that the drop-down "Time-field name" field is pre-populated with the value @timestamp, then click on "Create".

Now click on discover. Did u observe the logs?

Step 2 – Emitting logs from our application to logstash and visualising using elasticsearch and kibana

In this step, you will be working on projects 11-elk working set

1) In 11-02-quotes-service-solution , open pom.xml and observe that we have added following logback dependencies :

```
<dependency>
      <groupId>net.logstash.logback</groupId>
      <artifactId>logstash-logback-encoder</artifactId>
      <version>4.9</version>
</dependency>
<dependency>
      <groupId>ch.qos.logback</groupId>
      <artifactId>logback-classic</artifactId>
      <version>1.2.3</version>
</dependency>
<dependency>
      <groupId>ch.qos.logback</groupId>
      <artifactId>logback-core</artifactId>
      <version>1.2.3</version>
</dependency>
```

2) Open logback.xml and observe how we have configured LogstashTcpSocketAppender.

Observe how we have configured destination of logstash input.

3) We asume that your elk container is still running. If not please start the elk container as described in step 1

We want to start logstash to start a pipeline which listens on tcp socket 5000 and sends to elasticsearch.

So, follow the below steps :

Get the id of your elk container by running **docker ps** command

Use the following command to connect to the elk container

docker exec –it <containerid> bash

At the command prompt , execute the following

```
cd /opt/logstash/bin
```

```
./logstash   -e 'input { tcp { port => 5000 codec => "json" } } output { elasticsearch { hosts
=> ["localhost"] index => "micro-%{serviceName}"} }'
```

6) Now start eureka server in 03-01-eureka-server-common. Then start 11-02-quotes-service-solution

Open eureka console at http://localhost:5001 and make sure that all the services are registered

Now give a request to /quotes?q=IBM

Now Some logs might have been generated. We want to view them in kibana. Goto http://192.168.99.100:5601 and Discover the logs.

Step 3 – Using Sleuth

In this step, you will be working on projects 12-sleuth-zipkin working set

1) In 12-01-quotes-service-solution , open pom.xml and observe that we have added following dependencies :

```
<dependency>
			<groupId>org.springframework.cloud</groupId>
			<artifactId>spring-cloud-starter-sleuth</artifactId>
		</dependency>
```

Open logback.xml unser src/main/resources and observe the encoder pattern. We have configured the pattern to print TraceId and SpanId

Start Eureka and then start this Quotes Application

Make a request to   for /quotes?q=IBM on this service and observe the TRACEID and SPANDID in the logs

Now start PortfolioApplication in 12-02-portfolio-service-solution.

Observe the logback.xml and pom.xml. they have same settings like in

Make a request to /portfolio/{youruserid}. Now observe the   TRACEID and SPANID in the logs of portfolio application.

Also observe the TRACEID and SPANID in the logs of quoteservice application.

Is the TRACEID Same in both the logs?

Congratulations !! you know how to correlate logs using TRACEID