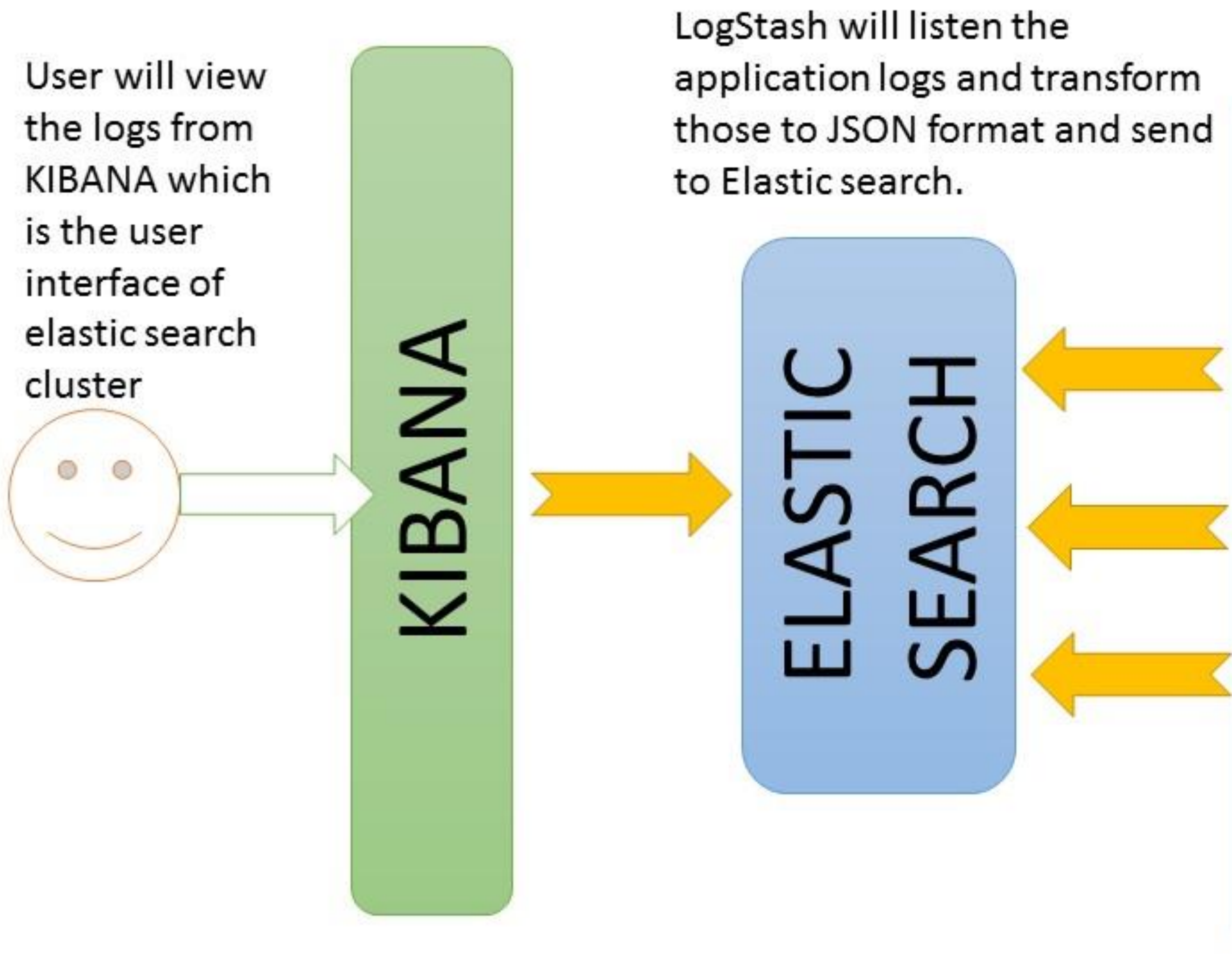
ELK Stack

1. What is ELK Stack

1. **Elasticsearch** is a distributed, JSON-based *search and analytics engine* designed for horizontal scalability, maximum reliability, and easy management.
2. **Logstash** is a dynamic *data collection pipeline* with an extensible plugin ecosystem and strong Elasticsearch synergy.
3. **Kibana** gives the *visualization* of data through a UI.

1.1. ELK Stack Architecture

Logstash processes the application log files based on the filter criteria we set and sends those logs to Elasticsearch. Through Kibana, we view and analyze those logs when required.


User will view the logs from KIBANA which is the user interface of elastic search cluster
KIBANA
LogStash will listen the application logs and transform those to JSON format and send to Elastic search.
ELASTIC SEARCH

ELK stack interaction with different applicati

ELK In Action

2. ELK stack configuration

All these three tools are based on JVM and before start installing them, please verify that JDK has been properly configured. Check that standard JDK 1.8 installation, JAVA_HOME and PATH set up is already done.

2.1. Elasticsearch

- Download latest version of Elasticsearch from this download page and unzip it any folder.

- Run bin\elasticsearch.bat from command prompt.
- By default, it would start at http://localhost:9200

2.2. Kibana

- Download the latest distribution from download page and unzip into any folder.
- Open config/kibana.yml in an editor and set elasticsearch.url to point at your Elasticsearch instance. In our case as we will use the local instance just uncomment elasticsearch.url: "http://localhost:9200"
- Run bin\kibana.bat from command prompt.
- Once started successfully, Kibana will start on default port 5601 and Kibana UI will be available at http://localhost:5601

2.3. Logstash

- Download the latest distribution from download page and unzip into any folder.
- Create one file logstash.conf as per configuration instructions. We will again come to this point during actual demo time for exact configuration.

Now run bin/logstash -f logstash.conf to start logstash

ELK stack is not up and running. Now we need to create few microservices and point logstash to the API log path.

3. ELK stack example – Create Microservice

3.1. Create Spring Boot Project

Let’s create an application using spring boot for faster development time. Follow those steps to start this service.

3.2. Add REST Endpoints

Add one RestController class which will expose few endpoints like /elk, /elkdemo, /exception. Actually we are going to test few log statements only, so feel free to add/modify logs as per your choice.

```
package com.example.howtodoinjava.elkexamplespringboot;


import java.io.PrintWriter;
import java.io.StringWriter;
import java.util.Date;


import org.apache.log4j.Level;
import org.apache.log4j.Logger;
```

```
}

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.core.ParameterizedTypeReference;
import org.springframework.http.HttpMethod;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;

@SpringBootApplication
public class ElkExampleSpringBootApplication {

   public static void main(String[] args) {
      SpringApplication.run(ElkExampleSpringBootApplication.class, args);
   }
}

@RestController
class ELKController {
   private static final Logger LOG = Logger.getLogger(ELKController.class.getName());

   @Autowired
   RestTemplate restTemplete;

   @Bean
   RestTemplate restTemplate() {
      return new RestTemplate();
   }

   @RequestMapping(value = "/elkdemo")
```

```
public String helloWorld() {
    String response = "Hello user ! " + new Date();
    LOG.log(Level.INFO, "/elkdemo - &gt; " + response);

    return response;
}

@RequestMapping(value = "/elk")
public String helloWorld1() {

    String response = restTemplete.exchange("http://localhost:8080/elkdemo", HttpMethod.GET, null, new Paramet
    }).getBody();
    LOG.log(Level.INFO, "/elk - &gt; " + response);

    try {
        String exceptionrsp = restTemplete.exchange("http://localhost:8080/exception", HttpMethod.GET, null, new P
        }).getBody();
        LOG.log(Level.INFO, "/elk trying to print exception - &gt; " + exceptionrsp);
        response = response + " === " + exceptionrsp;
    } catch (Exception e) {
        // exception should not reach here. Really bad practice :)
    }

    return response;
}

@RequestMapping(value = "/exception")
public String exception() {
    String rsp = "";
    try {
        int i = 1 / 0;
```

```
            // should get exception
        } catch (Exception e) {
            e.printStackTrace();
            LOG.error(e);

            StringWriter sw = new StringWriter();
            PrintWriter pw = new PrintWriter(sw);
            e.printStackTrace(pw);
            String sStackTrace = sw.toString(); // stack trace as a string
            LOG.error("Exception As String :: - &gt; "+sStackTrace);

            rsp = sStackTrace;
        }

        return rsp;
    }
}
```

3.3. Configure Spring boot Logging

Open application.properties under resources folder and add below configuration entries.

```
logging.file=elk-example.log
spring.application.name = elk-example
```

Read More: Spring Boot Logging Example

3.4. Verify Microservice Generated Logs

Do a final maven build using mvn clean install and start the application using command java -jar target\elk-example-spring-boot-0.0.1-SNAPSHOT.jar and test by browsing http://localhost:8080/elk.

Don’t be afraid by seeing the big stack trace in the screen as it has been done intentionally to see how ELK handles exception message.

Go to the application root directory and verify that the log file i.e. elk-example.log has been created and do a couple of visits to the endpoints and verify that logs are getting added in the log file.

4. Logstash Configuration

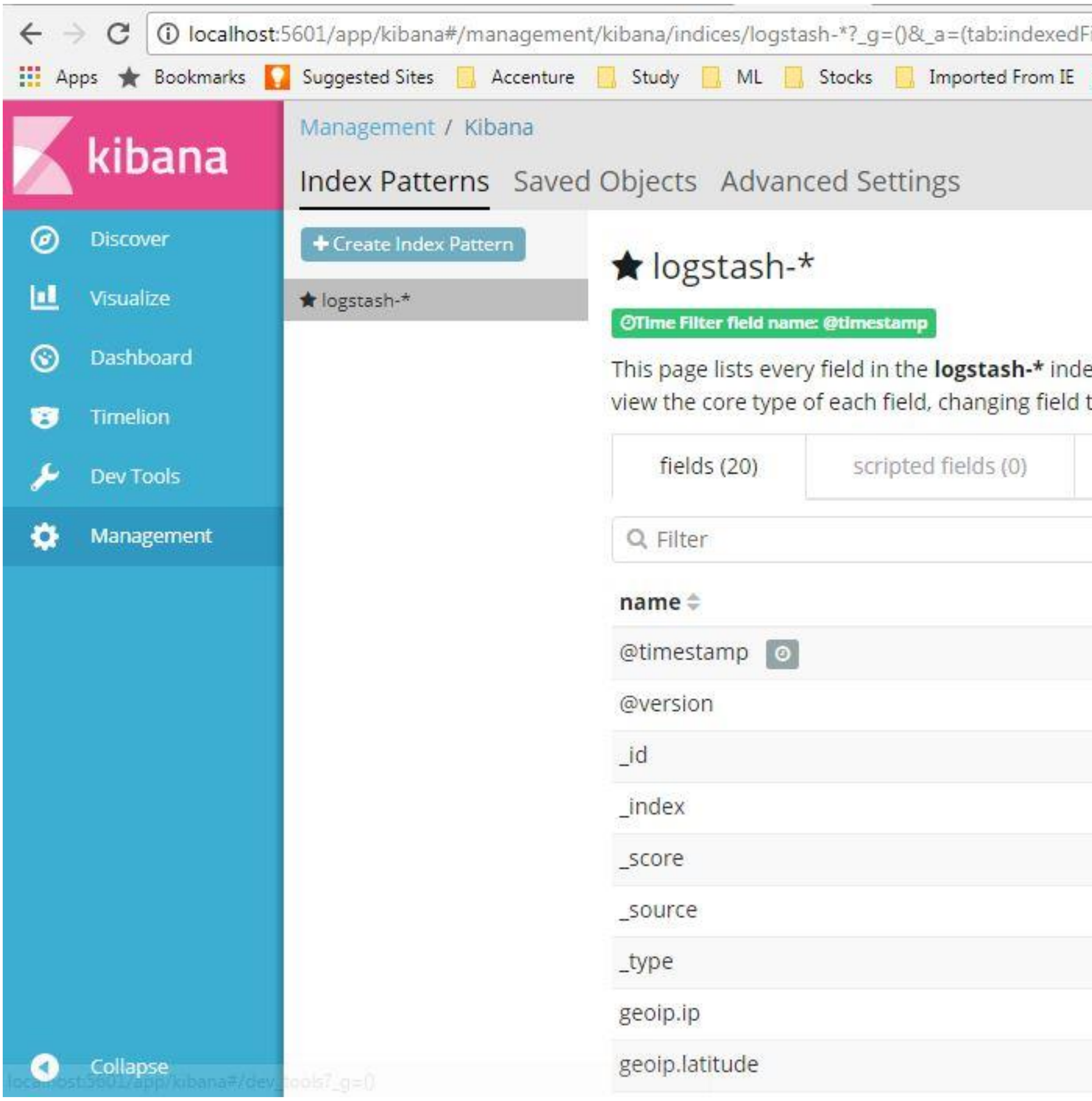
We need to create a logstash configuration file so that it listen to the log file and push log messages to elastic search. Here is the logstash configuration used in the example, please change the log path as per your setup.

```
input {
  file {
    type => "java"
    path => "F:/Study/eclipse_workspace_mars/elk-example-spring-boot/elk-example.log"
    codec => multiline {
      pattern => "^%{YEAR}-%{MONTHNUM}-%{MONTHDAY} %{TIME}.*"
      negate => "true"
      what => "previous"
    }
  }
}

filter {
  #If log line contains tab character followed by 'at' then we will tag that entry as stacktrace
  if [message] =~ "\tat" {
    grok {
      match => ["message", "^(\tat)"]
      add_tag => ["stacktrace"]
    }
  }

 grok {
    match => [ "message",
               "(?<timestamp>%{YEAR}-%{MONTHNUM}-%{MONTHDAY} %{TIME})  %{LOGLEVEL:level} %{NUMBER:pid} --- \
               "message",
               "(?<timestamp>%{YEAR}-%{MONTHNUM}-%{MONTHDAY} %{TIME})  %{LOGLEVEL:level} %{NUMBER:pid} --- .
             ]
  }


  date {
    match => [ "timestamp" , "yyyy-MM-dd HH:mm:ss.SSS" ]
  }
}

output {

  stdout {
    codec => rubydebug
  }

  # Sending properly parsed log events to elasticsearch
  elasticsearch {
    hosts => ["localhost:9200"]
  }
}
```

5. Kibana Configuration

Before viewing the logs in Kibana, we need to configure the Index Patterns. We can configure logstash-* as default configuration. We can always change this index pattern in logstash side and configure in Kibana. For simplicity, we will work with default configuration.

The index pattern management page will look like below. With this configuration we are pointing Kibana to Elasticsearch index(s) of your choice. Logstash creates indices with the name pattern of logstash-YYYY.MM.DD We can do all those configuration in Kibana console http://localhost:5601/app/kibana and going to Management link in left panel.



This page lists every field in the **logstash-*** inde
view the core type of each field, changing field t

```
@timestamp
@version
_id
_index
_score
_source
_type
geoip.ip
geoip.latitude
```

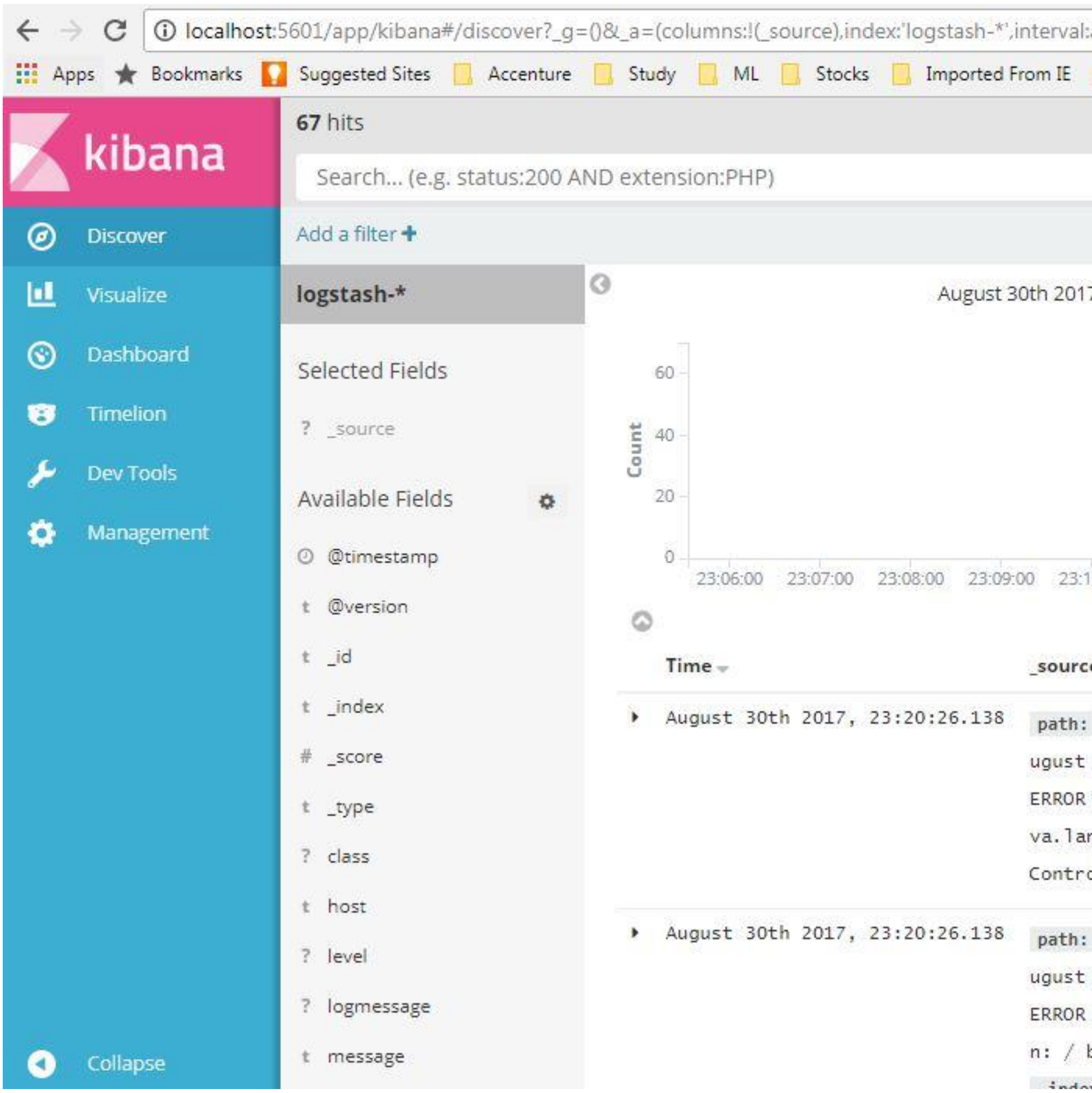
Logstash configuration in Kibana

6. Verify ELK Stack

Now when all components are up and running, let's verify the whole ecosystem.

Go to application and test the end points couple of times so that logs got generated and then go to Kibana console and see that logs are properly stacked in the Kibana with lots of extra feature like we can filter, see different graphs etc in built.

Here is the view of generated logs in Kibana.



localhost:5601/app/kibana#/discover?_g=()&_a=(columns:!(_source),index:'logstash-*',interval:

Apps Bookmarks Suggested Sites Accenture Study ML Stocks Imported From IE

67 hits

Search... (e.g. status:200 AND extension:PHP)

Add a filter +

logstash-*

Selected Fields

? _source

Available Fields

@timestamp

t @version

t _id

t _index

_score

t _type

? class

t host

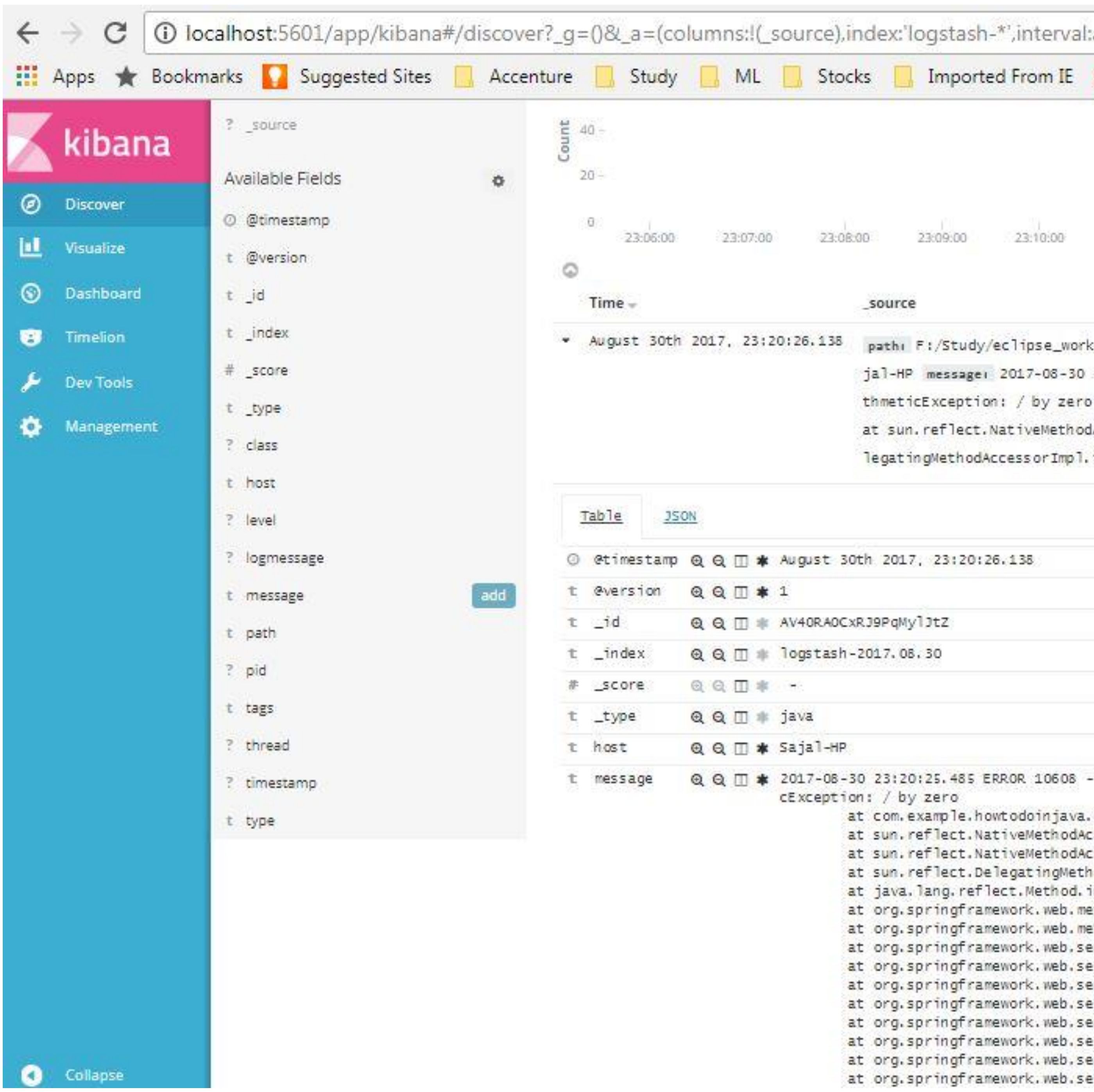
? level

? logmessage

t message

```
Time                                   _sourc
▸ August 30th 2017, 23:20:26.138       path:
                                       ugust
                                       ERROR
                                       va.lar
                                       Contro
▸ August 30th 2017, 23:20:26.138       path:
                                       ugust
                                       ERROR
                                       n: / b
```

Kibana Logs Overview



```
localhost:5601/app/kibana#/discover?_g=()&_a=(columns:!(_source),index:'logstash-*',interval:
```

```
August 30th 2017, 23:20:26.138   path: F:/Study/eclipse_work
                                 jal-HP message: 2017-08-30
                                 thmeticException: / by zero
                                 at sun.reflect.NativeMethod
                                 legatingMethodAccessorImpl.
```

```
@timestamp  August 30th 2017, 23:20:26.138
@version    1
_id         AV40RAOCxRJ9PqMylJtZ
_index      logstash-2017.08.30
_score      -
_type       java
host        Sajal-HP
message     2017-08-30 23:20:25.485 ERROR 10608 -
            cException: / by zero
                     at com.example.howtodoinjava.
                     at sun.reflect.NativeMethodAc
                     at sun.reflect.NativeMethodAc
                     at sun.reflect.DelegatingMeth
                     at java.lang.reflect.Method.i
                     at org.springframework.web.me
                     at org.springframework.web.me
                     at org.springframework.web.se
                     at org.springframework.web.se
                     at org.springframework.web.se
                     at org.springframework.web.se
                     at org.springframework.web.se
                     at org.springframework.web.se
                     at org.springframework.web.se
                     at org.springframework.web.se
```

Kibana Logs details screen

7. ELK Stack Tutorial – Summary

In this **ELK example**, we learned to **configure ELK stack** and the saw how we can point our application log files to ELK and view and analyze logs in Kibana. I will suggest you to play with configurations and share your learnings with us. e.g.

- Instead logstash to listen to our logs, we can use logback configuration to use TCP appender to send logs to a remote Logstash instance via TCP protocol.
- We can point multiple log files with Logstash.

- We can use more sophisticated filters in the logstash configuration file to do more as per our need.
- We can use remote ELK cluster to point to our log files, or push logs into, this is basically required when all the applications will be deployed in cloud.
- Create different index patterns in logstash.