Auto Configuration Module for Logstash in Elasticsearch Ecosystem

Hammad Ahmed, Yoosang Park, Jongsun Choi, Jaeyoung Choi
Dept. of Computer Science and Engineering, Soongsil University
e-mail: malikhammad124@gmail.com, yspark@soongsil.ac.kr, {jongsun.choi, choi}@ssu.ac.kr

Abstract

Log analysis and monitoring have a significant importance in most of the systems. Log management has core importance in applications like distributed applications, cloud based applications, and applications designed for big data. These applications produce a large number of log files which contain essential information. This information can be used for log analytics to understand the relevant patterns from varying log data. However, they need some tools for the purpose of parsing, storing, and visualizing log information. “Elasticsearch, Logstash, and Kibana” (ELK Stack) is one of the most popular analyzing tools for log management. For the ingestion of log files configuration files have a key importance, as they cover all the services needed to input, process, and output the log files. However, creating configuration files is sometimes very complicated and time consuming in many applications as it requires domain expertise and manual creation. In this paper, an auto configuration module for Logstash is proposed which aims to auto generate the configuration files for Logstash. The primary purpose of this paper is to provide a mechanism, which can be used to auto generate the configuration files for corresponding log files in less time. The proposed module aims to provide an overall efficiency in the log management system.

Keywords: Auto configuration, Logstash, Elasticsearch

1. Introduction

Nowadays log files are an important aspect in most of the computer systems. These files are widely used for analyzing and monitoring of the systems and networks. Log analysis and monitoring is a big concern for the development of the complex systems. Log management is exceptionally important in distributed applications, cloud based applications, and in applications for big data. In these applications, a large number of log files have been produced, which are essential to monitor the performance and security issues in the application.

In order to analyze and monitor different log files collected from heterogeneous devices or networks, we need some specific tools. Therefore, in most of the systems, Elasticsearch, Logstash, and Kibana (ELK Stack) [1] is used, which is the most popular tool to analyze and visualize the log files collected from different devices, networks, or systems. ELK Stack is the combination of Elasticsearch, Logstash, and Kibana. Elasticsearch is an open source, document based, RESTful search and analytics engine. Elasticsearch is the core of ELK Stack where everything is stored and searched. Kibana is the user interface, which is used to visualize log files and can be used to interact with Elasticsearch. Logstash is used to parse, filter, and transform data into the Elasticsearch. It is the most important component in the log management system. In log management systems, log files are stored in Elasticsearch using Logstash, and then visualize using Kibana. The process is shown in the Figure 1.

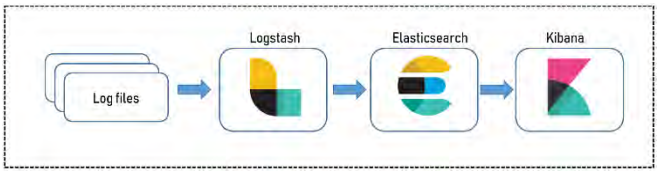

Figure 1 ELK log management

Logstash facilitates the ingestion of data into the Elasticsearch. There are systems generating logs, and those logs can be ingested into Elasticsearch in real time using the data pipeline that Logstash facilitates. The data pipeline of Logstash includes an input, a filter, and the output, these plugin functions perform their functionality according to the configuration information. Input plugin receives the data from the log file. Filter is the plugin which will transform or process the data. After the data has been processed, the output plugin will send the data to the Elasticsearch.

Configuration information is the necessary part of Logstash. The configuration information includes input information of the log file to be parsed. Information about how the file should be formatted or processed, and where the log file should be forwarded as an output. Therefore, configuration information has a very key role in log management system. By knowing the above information, a configuration file for Logstash is created. However, domain expertise is required for creating these configuration files. Moreover, if more than one configuration files are to be created, users must manually write each file from scratch which can be time consuming.

Therefore, in order to solve these problems, a plugin module for Logstash is proposed. The module is named as Auto Configuration Module. The proposed module aims to receive log files as an input and generate configuration files as output which will be compatible with Logstash. This module consists of three submodules, Configuration File Manager, Template Manager, and Template Pool, which will be explained in Section 3. Main objective is to provide reusability of configuration files as well as efficiency in the system.

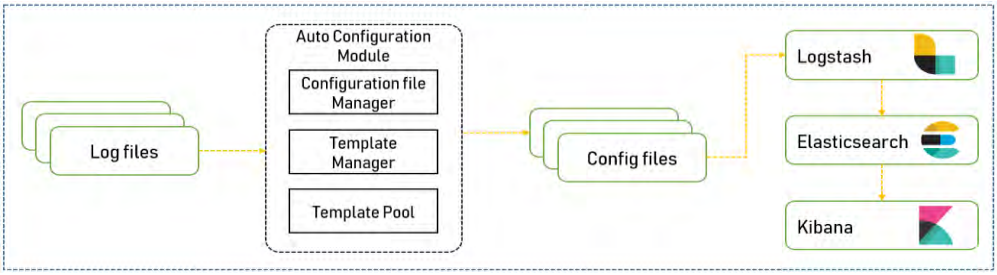

Figure 2 Log Management with Auto Configuration Module

2. Related Work

Log management has a key importance in these researches.

Lei et al. [2] suggested a system for combining Logstash, Elasticsearch, and Kibana for the monitoring and troubleshooting the logs. The proposed system in the above research uses log data as the source of information and analyze the potential relationship in different fields. For the log management system Logstash is the core component. Configuration files have the key importance in Logstash which are covering all the services.

Agrawal et al. [3] suggested a system in which log files are used for monitoring and determining the activities in the cloud systems. In this system log files data is send to Logstash which parses it and forwards to Elasticsearch. This system shows that even for the cloud based system log management has a key importance.

Doan and Iuhasz [4] mentioned that log management is equally important in big data applications. Hence, for the heterogeneous logs, Logstash is used for centralizing and parsing of the log files.

Bajer [5] suggested a method to Building an IoT Data Hub with Elasticsearch ecosystem. In this paper, Bajer organized how to config ecosystem modules such as Elasticsearch core, Logstash, and Kibana. However, there are difficulties for declaring specifications of input data because heterogeneous devices or sensors produce various data format and structure.

The researches mentioned above conclude that nowadays log management has a very key importance in most of the applications. Either they are distributed applications or cloud applications, the log management is equally important in these applications. Every research explained above uses Logstash for data parsing which requires manual creation of configuration files. So, if there is a module that can auto generate these configuration files based on the log files, a large amount of time can be saved. This module can improve their performance and provide efficiency in the process. Therefore, this research proposes a plugin module named Auto Configuration Module which aims to auto generate configuration files for Logstash.

3. Auto Configuration Module

Any complex system which requires log analysis will produce large numbers of heterogeneous log files. Hence, that system will require different configuration files in order to parse those log files. However, writing configuration files for every novice user becomes complicated. Sometimes configuration files are extremely large that they become unmanageable. Therefore, to solve these problems, a module is required which can auto generate the configuration files. This module will help the users to easily create configuration files promptly without any domain expertise.

The proposed Auto Configuration Module will automatically generate configuration files from the log files generated by different devices. Auto Configuration Module receives the log files as an input from the user and generates the corresponding configuration files for Logstash as shown in Figure 2. Auto Configuration Module also generates JSON schema, which is used for input type validation.

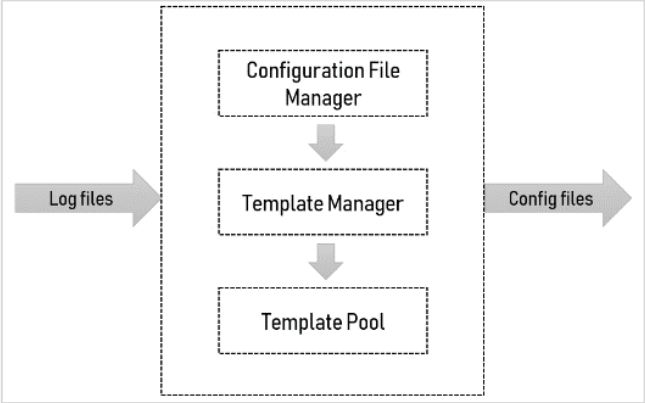

Figure 3 Auto Configuration Module

As shown in Figure 3, Auto Configuration Module consists of three submodules; Configuration File Manager, Template Manager, and Template Pool. Each submodule has different functionality in Auto Configuration Module. In order to produce configuration files, each submodule interact with each other.

3.1 Configuration File Manager

Configuration File Manager is the core module of Auto Configuration Module. It creates configuration files based on some inputs. It interacts with template manager to initialize a template based on user's preference and by writing information of log files in the selected template, it creates configuration files. Configuration File Manager also allows users to have their own file naming schema.

3.2 Template Manager:

Template Manager is responsible for generating configuration file templates. It interacts with Template Pool to load the template files according to the input request by the user. Template manager scans the template line by line and inserts appropriate information. Template manager also stores the new templates into the template Pool.

3.3 Template Pool:

In order to store and retrieve templates, Template Pool is created. This submodule of Auto Configuration Module provides reusability. It consists of sample templates which can be used according to the user's preference.

4. Implementation of Auto Configuration Module

Our module is divided into three submodules named Configuration File Manager, Template Manager, and Template Pool as explained in Section 3. These three modules perform their own tasks according to their functionality. For testing the Auto Configuration Module, we have used different log files which were collected from different devices. These log files have been used as an input for Auto Configuration Module. In response, it generates configuration files for Logstash.

4.1 Experiment

For the experiment, we collected sample input data from the Samsung galaxy Gear S2 as shown in the Figure 4. Based on this data Auto Configuration Module generates JSON schema and Configuration file as shown in the Figures 5 and 6 respectively.

Template manager in Auto Configuration Module will search for the appropriate template for Input Data Log File from the template pool and process the information in the configuration file, for example, the source path of the file, schema location, port number, and character set encoding the output. Generated configuration file is shown in the Figure 6.


```
{
"deviceId":"gear_2",
"userId":"u2",
"regTime":1518134637094,
"timestamp":"2018-02-09T09:03:57Z",
"cardiac":0,
"gyro":[3.78,-1.82,-13.79],
"acc":[7.95,-4.06,4.71]
}
```


Figure 4 Input Data Log File example


```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Bio",
  "description": "Bio Data",
  "type": "object",
  "properties": {
    "deviceId": {
      "type": "string"
    },
    "userId": {
      "type": "string"
    },
    "regTime": {
      "type": "number",
    },
     "timestamp": {
      "type": "string",
    },
      "cardiac": {
      "type": "integer",
     },
       "gyro": {
      "type": "array",
     },
       "acc": {
      "type": "array",
       }
  },
  "required": ["deviceId", "userId", "regTime","timestamp","cardiac", "gyro", "acc"]
}
```


Figure 5 Generated JSON Schema

```
input{
  file{
    path => ["/repository01/logs/bio.log"]
    type => "bio"
    codec => "json"
  }
}
filter{
  date{
    timezone => "Asia/Seoul"
    match => ["timestamp", "yyyy-MM-dd'T'HH:mm:ssZ", "yyyy-MM-dd'T'HH:mm:ss.SSSZ"]
    target => "@timestamp"
  }
  schema_validation{
   schema => "/usr/share/logstash/schemas/Bio_Schema.json"
   report_field => "_validation_errors"
   strict => true
  }
}

output{
  elasticsearch{
    hosts => ["localhost:9200"]
    document_type => "%{type}"
  }
  stdout{
    codec => "rubydebug"
  }
}
```

Figure 6 Generated Configuration File

Configuration file manager will receive the name of the configuration file and JSON schema as an input from the user and will generate the corresponding configuration file and JSON schema with the extension named "conf" and "json", respectively. Auto Configuration Module will interact with template manager and request for the appropriate template according to the user's input information. Configuration file manager will then store this file in the Logstash default directory. Similarly, configuration files are created for the other input log files using Auto Configuration Module.

5. Conclusion and Future Work

In this paper, a module is proposed that aims to auto generate configuration files for Logstash. Configuration files require domain specific knowledge and are created manually, therefore, they require lots of time. The proposed system aims to auto generate configuration files within less time and much more efficiently. Auto Configuration Module uses log files as an input and auto generates corresponding configuration files. This module can also generate JSON Schema which we use as type checking for validation of data. In the near future, configuration files can be generated for different parsing engines other than Logstash.

Acknowledgements

This work was supported by the Industrial Convergence Core Technology Development Program (No. 10062501) funded by the Ministry of Trade, Industry & Energy (MOTIE), Korea and was supported by Basic Science Research Program (No. 2016R1C1B2009744) through the National Research Foundation of Korea(NRF) funded by the Ministry of Science and ICT(MSIT)

References

[1] Elasticsearch, Logstash, Kibana, “ELK Stack”, www.elastic.co/elk-stack

[2] Xiafei Lei, Zhe Wang, Yuzhen He, “Log Real-time Management Scheme Based on LEK”, (IWMECS 2015)

[3] Vaibhav Agrawal, Devanjal Kotia, Kamelia Moshirian, Mihui Kim, “Log-Based Cloud Monitoring System for OpenStack”, 2018 IEEE Fourth International Conference on Big Data Computing Service and Applications

[4] Dong Nguyen Doan, Gabriel Iuhasz, “Tuning Logstash Garbage Collection for High Throughput in a Monitoring Platform”, 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2016)

[5] Marcin Bajer, “Building an IoT Data Hub with Elasticsearch, Logstash, and Kibana”, 2017 5th International Conference on Future Internet of Things and Cloud Workshops