Technical Paper- Spring boot logs streaming with Kafka and ELK stack

Version No: 1.0

Date: 07-FEB-2020

|  |  |
| --- | --- |
| Prepared by | Prashanth Kartan |
| Reviewed by |  |
| Approved by: |  |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version No | Date | Prepared by / Modified by | Significant Changes |
| 1.0 | 22-Jan-2020 | Prashanth Kartan | Initial version |

Table of Contents

[1. Abstract 4](#_Toc31912430)

[2. Introduction 5](#_Toc31912431)

[3. System Design 5](#_Toc31912432)

[4. Stream Log4J logs to Apache Kafka 5](#_Toc31912433)

[5. Consuming Kafka Message in Logstash 6](#_Toc31912434)

[6. Visualize Kafka message with Kibana 8](#_Toc31912435)

[7. Result and discussion 8](#_Toc31912437)

[8. Challenges 9](#_Toc31912438)

[9. Conclusion 9](#_Toc31912439)

[10. References 9](#_Toc31912440)

[11. Annexure 9](#_Toc31912441)

# 1. Abstract

|  |  |
| --- | --- |
| Objective | There are log4j logging system implemented which slowly eat disk space over time. This leads to decrease application performance when disk is full. To overcome this problem, we will be setting up apache Kafka, Logstash and Elasticsearch to stream log4j logs directly to Kafka from an application and visualize the logs in Kibana dashboard. |
| Abstract | This paper explores apache Kafka, Logstash and Elasticsearch to stream log4j logs directly to Kafka from a web application and visualize the logs in Kibana dashboard. The application logs that is streamed to Kafka will be consumed by Logstash and pushed to Elasticsearch and Kibana which are combined to perform operations on logs.  *Kafka* is a distributed streaming platform which building real-time streaming data pipeline that reliably get data between systems/applications and transform/react to the streams of data.  *Logstash* is used managing logs. It supports virtually any type of log, including system logs, error logs and custom application logs. It can retrieve logs from several sources, including syslog, messaging queue, and JMX, it can output data in a variety of ways, including email, webserver and to Elasticsearch.  *Elasticsearch* could be a full-text, real-time search and analytics engine that stores the data. It is built on the Apache Lucene search engine programmed library and exposes data through REST and java APIs. Elasticsearch is scalable and is built to be used by distributed systems.  *Kibana* is a web-based graphical interface for searching, analyzing and visualizing log data stored within the Elasticsearch indices. It utilizes the REST interface of Elasticsearch to retrieve the data and not only enables users to create customized dashboard views of their data, but also allows them to build DSL query and filter the data in ad hoc manner. |

# 

# 2. Introduction

In today’s world, utilization of applications is huge and processing millions of transactions per day and applications are generating huge amount of data in the form of logs that are significant source of valuable information. It is difficult to handle all logs in one place and no easy way to analyze it. Analysis of these high amounts of data needs appropriate and sophisticated methods to process them promptly, efficiently and precisely. Data logging is an important asset in web application monitoring and reporting as it contains massive amounts of data about the application behavior. Analysis of logged data can be a great help with reporting of malicious use, intruder’s detection, compliance assurance and the anomalies that might lead to actual damage.

The processing mode of the solutions in use is to collect and manage logs on a single device. With the increasing of the number of equipment and the service items, the total amounts of logs in actual production has reached the limit of the performance of traditional solutions. The upgrade is usually to update the hardware, which is useless on the problem of Big Data. For traditional solutions, it is difficult for relational database to carry out full-text retrieval for unstructured data, visualization is also difficult to meet the increasingly complex needs.

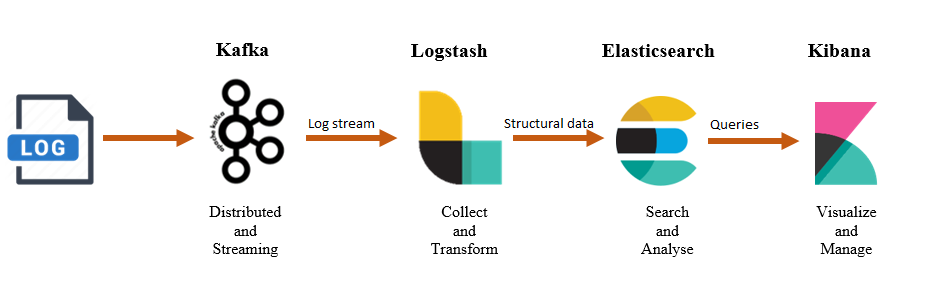
The appearance of the ELK technology stack effectively solves the problems of the traditional log system. All the data are stored in each node by the distributed cluster and have great horizontal expansion ability. Adding new node in the cluster can solve the problem of performance bottleneck and improve the overall robustness of the system.

# 3. System Design

The below block diagram of proposed system illustrates how the ELK stack components are used to collect the log data.



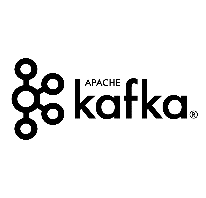
Loggregator



# 4. Stream Log4J logs to Apache Kafka

Because of limitations in existing systems, we developed a new messaging-based log aggregator Kafka. Let’s first introduce the basic concepts of Kafka. A stream of messages of a type is defined by a *topic*. A producer can publish messages to a topic. The published messages are then stored at a ser of servers called *brokers*. A consumer can subscribe to one or more topics from the brokers and consume the subscribed messages by pulling data from the brokers.

Here is the block diagram of streaming logs from spring boot application to Kafka.



Kafka producer

To stream log4j logs to Apache Kafka using maven artifact **kafka-log4j-appender**. The spring boot application will have log4j configuration with Kafka Appender that will stream the logs generated in the application to Kafka.

Here is the maven dependency of kafka-log4j-appender.

<dependency>

<groupId>org.apache.kafka</groupId>

<artifactId>kafka-log4j-appender</artifactId>

<version>1.0.0</version>

</dependency>

**Kafka configuration in Log4J2.xml**

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

<Configuration status="info" name="spring-boot-kafka-log" packages="<package-name>">

<Appenders>

<Kafka name="Kafka" topic="<topic-name>">

<PatternLayout pattern="%date %message"/>

<Property name="bootstrap.servers">localhost:9092,localhost:9091</Property>

</Kafka>

<Async name="Async">

<AppenderRef ref="Kafka"/>

</Async>

<Console name="stdout" target="SYSTEM\_OUT">

<PatternLayout pattern="%d{HH:mm:ss.SSS} %-5p [%-7t] %F:%L - %m%n"/>

</Console>

</Appenders>

<Loggers>

<Root level="INFO">

<AppenderRef ref="Kafka"/>

<AppenderRef ref="stdout"/>

</Root>

<Logger name="org.apache.kafka" level="WARN" />

</Loggers>

</Configuration>

# 5. Consuming Kafka Message in Logstash

Let’s introduce about the Logstash and its concepts. Logstash is an open source data collection engine. It also does index on large amount of data or logs that are collected from different servers. This data is passed to the Elasticsearch for further query processing. It can dynamically unify data from variety of sources and normalize the data into framework of user’s choice. It is a tool to collect, process, and forward events and log messages. Collection is accomplished via configurable input plugins including raw socket/packet communication, file tailing, and several message bus clients. It does the four main tasks such as parsing the data and logs, extracting the data and logs, managing the logs and structuring it. We get server logs event viewer logs and application custom logs.

Logstash Architecture

Outputs

Filters

Inputs

**LOGSTASH PIPELINE**

Data sources

To consume Kafka message into Logstash, we need to provide the configuration file to Logstash. In this file, we will have configuration related our Kafka. Typically, Logstash config file consists of three main sections: input, filter and output. Each section contains plugins that do relevant part of the processing.

Following is the configuration that tells Logstash about Kafka server address and the topic name from which it can consume the messages.

**logstash-kafka.conf**

input {

kafka {

bootstrap\_servers => "localhost:9092,localhost:9091"

topics => ["<topic-name>"]

}

}

filter {

grok {

match => [ “message”, “%{GREEDYDATA}”]

}

}

output {

elasticsearch {

hosts => ["localhost:9200"]

index => "<index-name>"

workers => 1

}

}

**Input section**

Input section defines from where Logstash will read input data – in our case it will be a application server details from where the logs are published in topic.

**Filter section**

Filter section contains plugins that perform intermediary processing on a log event. In our case, event will either be a single log line or multiline log event grouped according to the rules described above. In filter section we will do several things:

* Tag a log event if it contains a stack trace. This will be useful when searching for expectations later.
* Parse out (or grok, in Logstash terminology) timestamp, log level, pid, thread, class name and log message.
* Specified timestamp field and format – Kibana will use that later for time-based searches.

**Output section**

Output section contains output plugins that send event data to a destination. Outputs are the final stage in the event pipeline. We will be sending our logs events to stdout (console output) and to Elasticsearch.

Once the config file is in place and Elasticsearch is running, we can run Logstash by following command:

<logstash-path>/logstash -f <logstash-config-file-name>.conf

Now, we have Kafka, Elasticsearch and Logstash is up and running and our application log is directly getting pushed to Kafka and Logstash is reading from it pushing to Elasticsearch.

# 6 Visualize Kafka message with Kibana

Kibana is an open source data visualization platform that allows you to interact with your data through stunning, powerful graphics that can be combined into custom dashboards that help you share in sites from your data far and wide. This tool does the tasks such as exploring, visualization and discovering data. Depending upon the query and the JSON response the result is generated. Kibana creates tables, graphs, pie charts etc. Thus, Kibana does easy representation of large volumes of data and provides analytics. Our system using these tools helps different organizations generating huge amount of data in managing, storing and provides understanding, representation of data in easy graphical forms. It also converts complex and unstructured data to structured data and indexing for its easy representation.

# Kibana query

Kibana Overview

# 7 Result and discussion

In this system basically we are streaming the application logs to the ELK stack. The ELK stack is used which is implemented on cloud platform. Logs from different servers are collected and given to the Logstash which does indexing sorting of these logs which was otherwise difficult. This indexed data when passed to elastic search can then be used for query processing. This query is given by the users to the Elasticsearch and results are displayed on Kibana in an easy graphical representation. These results are combined to form dashboards.

# 8 Challenges

* There are couple of things we need to take care while configuring Kafka Appender in log4j2 configuration. To avoid recursive logging, we have configured apache Kafka logging level as WARN. One thing to note here is, the logging level can’t be debugged to avoid recursive logging. This appender is synchronous by default and will block until the record has been acknowledged by the Kafka server and hence, we have wrapped it with Async Appender to log asynchronously.

# 9 Conclusion

This system provides log streaming and log analytics which is based on Apache Kafka and ELK stack using the cloud platform. As companies produce large chunks of data and there is no easy way to analyze this data. Elasticsearch, Logstash and Kibana are the main tools of this system which provides an easy way to analyze the data. This system using the above three tools to help different organizations generating huge amount of data in different tasks such as managing, storing, and analyzing and provides understanding, representation of data in easy graphical forms. It also converts complex and unstructured data to structured data and indexing for its easy representation which helps the end user in understanding the data with less effort. These results are finally displayed on dashboards.

# 10 References

International Journal of Advanced Research in Computer and Communication Engineering - Log analytics using ELK stack on Cloud platform

# 11 Annexure

<https://kafka.apache.org/>

<https://www.elastic.co/guide>

<https://www.devglan.com/apache-kafka/stream-log4j-logs-to-kafka>

<https://www.devglan.com/apache-kafka/kafka-elasticsearch-logstash-example>

<https://moz.com/devblog/log-aggregation-with-elk-kafka>