Report on using elastic stack for log processing and monitoring by Dan Coleman

Abstract:

As time goes on and we shift from the old client-server architectures to that of microservices logging becomes more and more of a critical component. Many microservices can be scaled in size on demand, to do this, resources are automatically being provided dynamically as workloads spike. Much of this information such as CPU, memory, network or disk utilization of the service allocated over time is stored or “logged” as to help monitor the system. As to not lose the logging information when resources are removed there is a need to abstract the storing of logging information from the physical servers the logs are created on [3]. In this report we will look at how each of the tools in the log stash, why you would use logging and how you would use elastic stack for log processing.

Introduction:

Elasticsearch, Logstash, and Kibana (formerly known as the ELK Stack, now renamed to the **Elastic Stack** with the addition of Beats) is a set of open source tools made by Elastic designed to help users take data from any type of source and in any format and search, analyse and visualize that data in real time at any scale [2][7]. Elastic Stack can be deployed on premise or made available as Software as a Service (SaaS) [7]. Logstash provides an input stream to Elasticsearch for storage and search, and Kibana accesses the data for visualizations such as dashboards. In this first section we shall look at each tool within the Stack. Elastic also provides "Beats" packages which can be configured to provide pre-made Kibana visualizations and dashboards about various database and application technologies.[10]

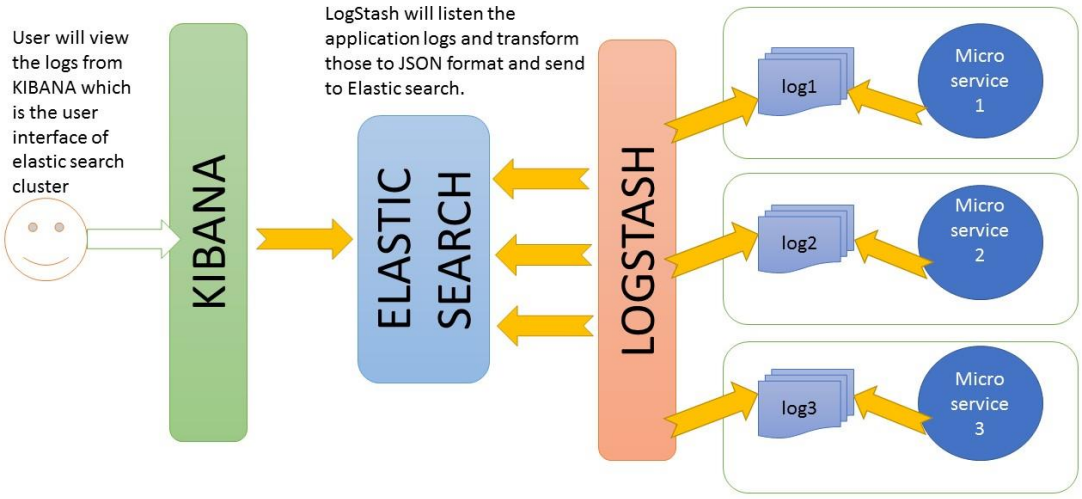

Figure 1 How Elasticstack works together [12]

Tools:

Elasticsearch

Elasticsearch is the central component of the Elastic Stack. Elasticsearch is a distributed RESTful, search and analytic engine for many kinds of data. It was developed in Java and based on the Apache Lucene search engine library [1]. According to db-engines.com's ranking it is the most popular database management search engine [4]. Elasticsearch takes the logging information and stores it in JSON document format. Each document is stored as a key-value pairing. It uses a data structure known as *inverted index* to allow fast full-text searches. An inverted index lists every unique word that appears in a document and identifies all the documents each word occurs in. During the indexing process, Elasticsearch builds an inverted index to make data searchable in near real-time [1]. The data is stored clusters which is a collection of one or more servers that together hold the entire data and give federated indexing. With this system you can nest queries to quickly receive the fields you wish in a single query [5]. As of May 2019, the core security features of the Elastic Stack have been made available to the public free of charge [6].

Logstash

Logstash is server-side data processing pipeline, it allows you to simultaneously collect log events from multiple different sources, aggregate and process this data into a standardized log format and distribute it to Elasticsearch [1][7]. As data travels from source to store Logstash filters parse each event, identify if it has named fields to build structure, and transform them to converge on a common format regardless of previous format or complexity. Along with Elasticsearch it is possible to route the data to a variety of outputs. Logstash is highly customisable with over 200 plugins to allow you to customise it to best suit your needs.[8]

Kibana

Kibana is a data visualization and management tool for Elasticsearch that provides real-time updating histograms, line graphs, pie charts, and maps [1]. It is specialized for large volumes of streaming, real-time data.[7] It also provides Canvas, a presentation tool, that allows users to create slide decks that pull live data directly from Elasticsearch. [9]

Beats

Finally, beats are "data shippers", separately installable products that are installed on servers as agents used to send different types of operational data to Elasticsearch either directly or through Logstash, where data may be enhanced or archived. [7]

Uses of Logging Files:

On a microservice having a sound logging strategy is a critical component of building a secure, manageable system. Log files hold information on the behaviour of database activity, users, error and debugging information. As systems get bigger and a system have more and more servers finding data in logs becomes more difficult, this the need of centralised logging. In this section we shall cover several of the main uses for log files in systems:

- **Auditing:** Having a trackable trail of data for auditors is often needed to pass audits. Those documents need have real data from logs.
- **Security:** Intrusion detection and fraud detection analysis rely on collecting the logs of all user access both successful and unsuccessful.
- **Troubleshooting:** Having logs of debugging information and error messages collected makes analysing what problems may occur in the environment significantly easier.

- **Monitoring:** Identifying reoccurring trends in the service along with thresholds allows a programmer to predict what issues may occur in the future and thus solve them proactively before they effect the end user. [3]

It is worth noting that log data alone is not an optimal solution as typically it records too much to search through easily therefore, tools such as Elasticstack are incredibly useful as they include ways to search through the logs more effectively. [11]

Integrating elastic search:

This section will briefly cover how one would integrate the Elasticstack into a cloud-based microservice application.

Logging Requirements

There are two key requirements for architecting a centralized logging strategy. The first requirement is to **Direct Logs to a redundant and isolated storage area**. A possible solution is depicted in the figure below, but we shall show how this design could be changed to suit Elasticstack. In the case of Elasticstack this would be done by Logstash. All logs would be pipelined from the local machines and aggregate them to a standardised format which would then be sent to Elasticsearch (Logging Service) which would allow the following benefits:

- Auditing would be easier as all logs are together in one location.
- Since the logging data is sent to Elasticsearch trend analysis is much easier as you can query the information.
- Loss of log data is minimised as the data is not stored on the local disk of servers that may become deprovisioned on the spot. [3]

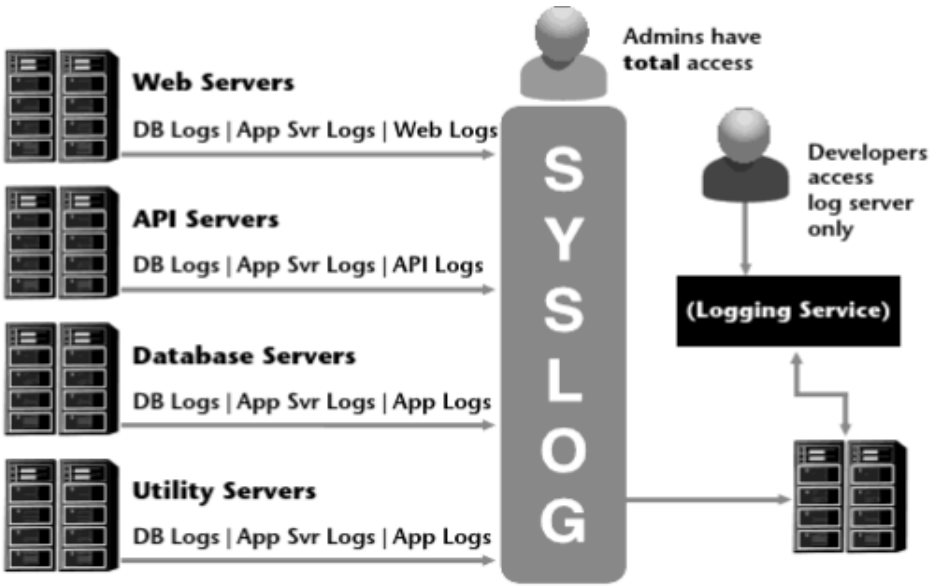

Figure 2 Centralized Logging Strategy [3]

The second requirement is to have a **standardized log format.** If log messages aren't designed in a standardized format, the value of the data is rather limited, searches will be sub-optimal, and it will be harder to produce consistent results.[3] Using Elasticstack Logstash automatically standardises the logs to a single format, allowing a greater degree of automation. This in turn makes it much easier to detect patterns and proactively solve issues that may occur in the service in the future.

Stack configuration

The three main tools (Elasticsearch, Logstash, Kibana) are based on the Java Virtual Machine so a JDK must be installed. Download each of the 3 tools from Elastic's official website and run the .bat for Elasticsearch. The default port is 9200. Next open config/kibana on an editor and set the port for Elasticsearch; in our case this would be elasticsearch.url: "http://localhost:9200". Next Run the Kibana .bat file in cmd. The default port is 5601, to access the UI use "http://localhost:5601/". Finally create a logstash.conf file. A default file should look like:

```
input { stdin { } }
output {
  elasticsearch { hosts => ["localhost:9200"] }
  stdout { codec => rubydebug }
}
```

Figure 3 A default logstash.conf file

Then run "bin/logstash -f logstash.conf" to start Logstash in cmd. The stack is now up and running. In the next step we will cover and pointing Logstash to the API log path of a microservice. [12]

Adding REST endpoints

In your microservice add a RestController class to expose a few endpoints. Below is an example provided by howtodoinjava.com [12]

```
@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 ParameterizedTypeReference() {
        }).getBody();
        LOG.log(Level.INFO, "/elk - &gt; " + response);
        try {
            String exceptionrsp = restTemplete.exchange("http://localhost:8080/exception", HttpMethod.GET, null, new ParameterizedTypeReference() {
            }).getBody();
            LOG.log(Level.INFO, "/elk trying to print exception - &gt; " + exceptionrsp);
            response = response + " === " + exceptionrsp;
        } catch (Exception e) {}
        return response;
    }
    @RequestMapping(value = "/exception")
    public String exception() {
        String rsp = "";
        try {
            int i = 1 / 0;
        } 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;
    }
}
```

Be sure to configure the logging file correctly. In the sample provided this would involve going to application.properties in spring boot and adding the following entries:

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

If you follow along with this example to test you would start the application and browse http://localhost:8080/elk then verify that the log file “elk-example.log” has been created in the applications root directory. [12]

Configuring Logstash

Next, we need to set the logstash.conf file to let it know what it needs to listen for and what it should output to. Below is the conf file for sample provided. [12]

logstash.conf

```
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} ---
               \[(?<thread>[A-Za-z0-9-]+)\] [A-Za-z0-9.]*\.(?<class>[A-Za-z0-9#_]+)\s*:\s+(?<logmessage>.*)",
               "message",
               "(?<timestamp>%{YEAR}-%{MONTHNUM}-%{MONTHDAY} %{TIME})  %{LOGLEVEL:level} %{NUMBER:pid} --- .+? :\s+(?<logmessage>.*)"
             ]
  }


  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"]
  }
}
```

Configuring Kibana

Before the logs can be viewed in Kibana you must configure what is called an “Index Pattern”. You can configure logstash-* as default configuration. Index patterns tell Kibana which Elasticsearch indices you wish to explore. An index pattern can match the name of a single index or you can use wildcard (*) to match multiple indices. [13] For the sample we are using we shall use the default configuration. With this configuration Kibana is pointed to Elasticsearch index(es) of your choice. Logstash will create indices with the name pattern “logstash-YYYY.MM.DD”. All configuration can be done in Kibana’s console (http://localhost:5601/app/kibana). [12]

Verify Stack

Finally, to verify that all components are working correctly test a few of the endpoints and then check the Kibana console to see that the logs are properly stacked in Kibana, allowing filtering and viewing graphs. The Elasticstack is now set up correctly.

Beats

After installing and configuring the Elastic stack you can install beats if needed. Each one has a different setup and these setups also vary by what operating system you use and if you are using docker. However for each there is a detailed installation page on elastics official website (https://www.elastic.co/guide/en/beats/libbeat/6.2/installing-beats.html)

References

1. https://www.elastic.co/what-is/elasticsearch
2. https://www.elastic.co/webinars/introduction-elk-stack
3. Kavis, M. J. (2014). Architecting the Cloud : Design Decisions for Cloud Computing Service Models (SaaS, PaaS, IaaS).
4. https://db-engines.com/en/ranking/search+engine
5. https://dzone.com/articles/what-is-elasticsearch-and-how-it-can-be-useful
6. https://www.elastic.co/blog/security-for-elasticsearch-is-now-free
7. https://searchitoperations.techtarget.com/definition/Elastic-Stack
8. https://www.elastic.co/products/logstash
9. https://www.elastic.co/blog/getting-started-with-canvas-in-kibana
10. https://en.wikipedia.org/wiki/Kibana
11. https://thenewstack.io/logging-and-monitoring-why-you-need-both/
12. https://howtodoinjava.com/microservices/elk-stack-tutorial-example/#microservice
13. https://www.elastic.co/guide/en/kibana/current/tutorial-define-index.html