CS-554 Lab 3

Yue Lin

10479231

Scenario1: Logging

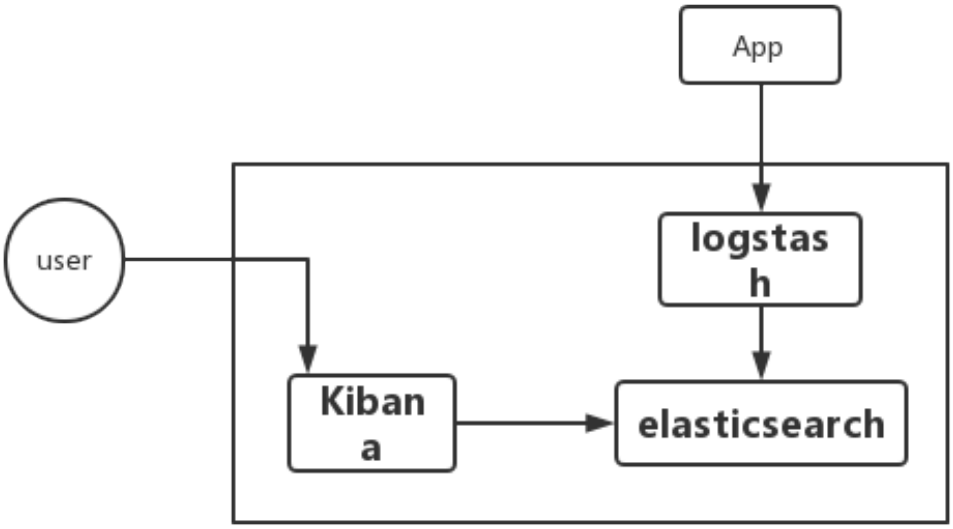
I will use ELK tech stacks to create a log server which my former company has used. ELK is the combination of Elasticsearch, logstash and Kibana.

Elasticsearch is a NoSQL database which is based on document.It stores data and provides many flexible and practical Restful APIs, so upper-layer applications can query data, use data, and analyze data as needed. In the log platform, all log data is stored in ElasticSearch. With its powerful search capabilities, logs can be flexibly queried quickly.

Logstash is mainly used to collect data and save the data into ElasticSearch. There are a lot of flexible plugins for Logstash , after collecting data with Logstash, we can parse the data and do other processes before sending to ElasticSearch.

Kibana is used to read the data in ElasticSearch and make it visualize like a good UI. And it brings good tool for us to query Rest API so that we don’t need to query with command lines.

Flow chart:



How would you store your log entries?

I will store my log entries as JSON type, because ElasticSearch the smallest unit to be searched(record element) is the document. The record will have the following common fields(which is similar with column in mysql): UID, username, log_level, url_path, url_method,datastamp.

How would you allow users to submit log entries?

My application will not allow users to submit log entries, because my app can collect log automatically. The collectors of the log system are deployed on each application server. They monitor the local log files. If any log is generated, it will obtain the newly generated log, obtain the log and process it. Usually, the line-based text log will be converted into JSON format data. , and then store the log in an ordered storage system and build an index to provide search services for subsequent clients. This completes the collection and persistence of log data.

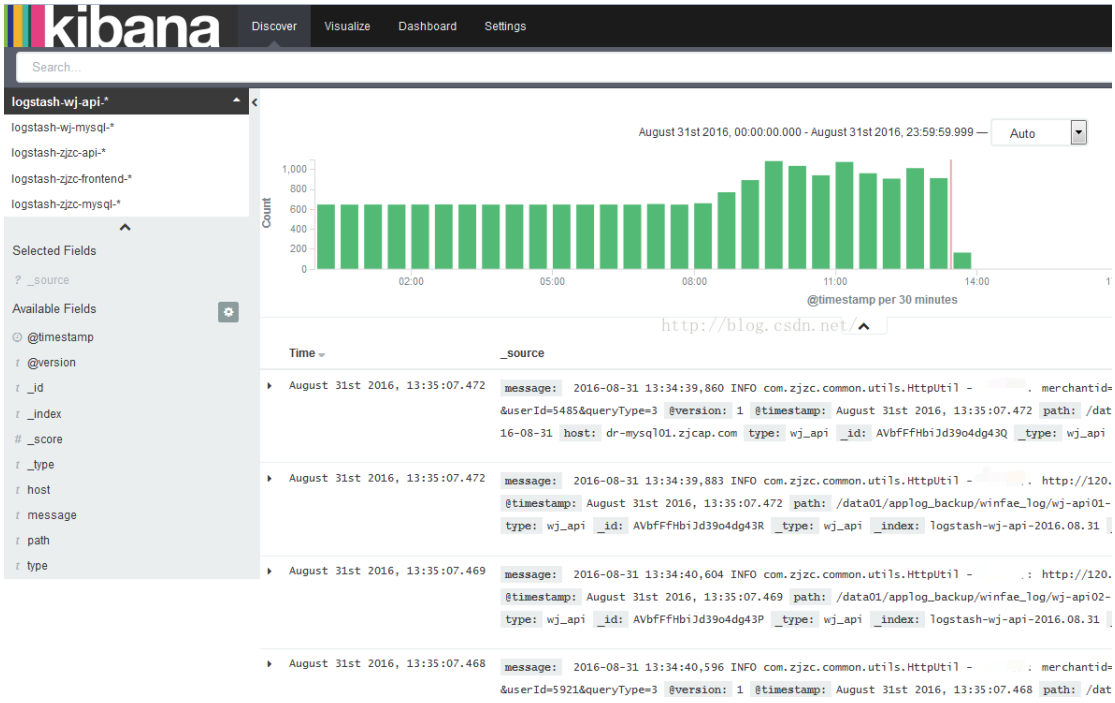
How would you allow them to query log entries?

As I mentioned before, I will use Kibana to show the data and query. By entering search terms in the search bar, you can search in indexes that match the current index pattern. You can do simple text queries, or use Lucene syntax, or use the JSON-based Elasticsearch query DSL.

For example, you want to search the username with “Tom”, you can write

```
{ “query”:{
        “match”:{
            “username”:”Tom”
            }
        }
}
```

The picture below is the one of the pages with Kibana. It is very clearly for us to read the log.s



How would you allow them to see their log entries?

Kibana is a web application so if you put it into a web-server, just open the web address with port of 5601(the default port) or you can put in your localhost. So just type the http://YOURDOMAIN.com:5601 in your browse.

What would be your web server?

I will choose Apache. It can run in Unix and Windows. It is free and open source. Instead of other complex configuration, it has simple setup and configuration, highly scalable and adaptable through modules. What's more it supports multi-languages.

Scenario2: Expense Reports

Because these reports data are important and may be used in the future, so we should persist the data, and the data will be stored in the SQL database such as Mysql or Oracle.

For the server, I will choose node as my server, because NodeJs does not create a new thread for each client connection, but only uses one thread. When a user is connected, an internal event is triggered. Through the non-blocking I/O and event-driven mechanism, the Node.js program is macroscopically parallel. Thus the node server can support more users connected at the same time.

In order to send email, I will use the third-party plugin----nodemailer. There are about three steps for sending email. Firstly, use SMTP to create the Nodemailer transport. Secondly, set the configuration about the subject. Last, use the function in Nodemailer to send the email.

To print the HTML to PDF, I will use wkhtmltopdf library. It can transfer any page to PDF in the browse. When generating PDF with wkhtmltopdf, it will automatically generate a tree directory structure based on your H tags in the HTML page. It can run in Linux, and windows.

To send email to users with pdf file, I will generate PDF with wkhtmltopdf tech in the server-side and then send the corresponding users emails with the attached file.

For the web templates, I may use Handlebars which we used mostly last semester. Handlebars is a dynamic templating language that is very simple to use. It doesn't have a lot of extra unique features so it is easy for us to learn and used into small projects. The Handlebars template looks like regular text, but it has embedded Handlebars expressions.

Scenario3: A Twitter Streaming Safety Service

To search or scan the keywords, I will use API like this

https://api.twitter.com/2/tweets/search/stream?tweet.fields=created_at&expansions=author_id&user.fields=created_at, because we can add rules into the stream, and any tweets which may match my rules will connect with the stream. And to search the content you want, you can add the tags which are keywords at the same when passing rules.

Parse the IP address of the sent tweet, and distinguish the address according to the IP address. For example, NJ and NY use different servers to accept and store the tweets. At the same time build a total server to store the alarm log around.

To protect the website, firstly when setting up the web application, we must take any script attack under consideration, write good code to avoid these attacks. Then make sure we have enough servers that when the main server shuts down, the copy will respond quickly. Last but not least, we must update and maintain periodically and handle the bugs which are reported.

My web server technology will be Nginx, because Nginx has the function of hot deployment and it can support high concurrent connections, the upper limit of concurrent connections supported by Nginx depends on your memory. During peak periods, Nginx can respond to requests faster than other web servers.

I will use MySQL as my database, the reasons are that, MySQL is free and more common in the

market and it has six different trigger to use. And for the historical log of tweets, I will store in MySQL, although there are countless tweets, but we can use different port and Redis cache to decrease the press.

For the real time, streaming incident report, I can use MessageQueue - Kafka. Its throughput is at the level of 100,000, but on the topic, once there is too much, the throughput will be greatly reduced. High availability is based on the master-slave architecture, and 0 loss can also be achieved after configuration parameter optimization. Currently mainly used by large companies for real-time computing and diary collection.

To store the media, I will choose cloud client storage to store it. If I store in the local disk, there will be no enough space for it. And when I store the media successfully, it will return a string of its address.

As I said before, I will choose Nginx as my web server. It is very easy to start, and it can run almost 24/7 without restarting even if it runs for several months

Scenario4: A Mildly Interesting Mobile Application

To handle the geospatial data store images, I will use Amazon Cloud to store these resources. When receiving the request of uploading images by users, the server side will find the place where the first folder is geographical location and the second folder is timestamp, and these files will named by the location, date and their original name, in this way we can find the find them very fast.

In my upload API, I will generate the file name by the location, date and its name. Then when upload successfully, it will return a cloud address to store in the database. And in my show images API, I will first pares the user's IP address to get his location and send a request to cloud to get the location's images ordered by time.

I will choose mongoDB as my database, because it has the its own fuzzy query for me to get the answer quickly instead of MySQL which I have to connect first and then traverse all the data.