1. Overview dev tools
2. Overview study map
3. Program analyzation
4. Reference

Contents

[**Development Tools** 3](#_Toc535247812)

[**Study Guide** 4](#_Toc535247813)

[**Back-End** 4](#_Toc535247814)

[1. Elasticearch 4](#_Toc535247815)

[2. Logstash 5](#_Toc535247816)

[**WebServer** 7](#_Toc535247817)

[1. Nodejs 7](#_Toc535247818)

[2. Ajax 8](#_Toc535247819)

[3. JQuery 9](#_Toc535247820)

[**Front-End** 9](#_Toc535247821)

[1. Kibana 9](#_Toc535247822)

[2. Plotly 10](#_Toc535247823)

[**Program Language** 11](#_Toc535247824)

[**Architecture** 11](#_Toc535247825)

[**Back-End** 11](#_Toc535247826)

[1. Elasticearch 11](#_Toc535247827)

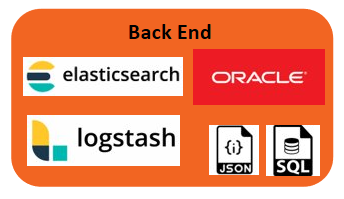
[2. Logstash 13](#_Toc535247828)

[**WebServer** 15](#_Toc535247829)

[1. Nodejs 15](#_Toc535247830)

# Development Tools

Three parts of the whole system:

****

# Study Guide

For the sake of time, the future developer taking this project is suggested to flow this learning map drawn on the previous experience and errors.

## Back-End

The back-end of this system is mainly the search engine and data pre-processing system.

### Elasticearch

As the core of this system, the fundamental or even advanced knowledge is required at the first beginning in order to understand the basic requirement of this project.

*Learning Map*:

*Reference*:

Official Website:

<https://www.elastic.co/guide/cn/index.html>

<https://www.elastic.co/guide/index.html> *(English)*

Overall Introduce:

<https://blog.csdn.net/yezonggang/article/details/80064394>

Download and Installation:

<https://blog.csdn.net/weidong22/article/details/79062851>

Query (use combine with Kibana):

<https://www.elastic.co/guide/en/elasticsearch/client/javascript-api/current/quick-start.html> *(English)*

<https://blog.csdn.net/tototuzuoquan/article/details/78303095>

<https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-multi-match-query.html>

<https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-match-query-phrase-prefix.html> *(English)*

<http://n3xtchen.github.io/n3xtchen/elasticsearch/2017/07/05/elasticsearch-23-useful-query-example> *(Recommend)*

Blog Series (Recommend):

<http://www.cnblogs.com/ginb/p/6637236.html>

<http://www.cnblogs.com/ginb/p/elasticsearch.html>

<http://www.cnblogs.com/ginb/p/6993299.html>

<http://www.cnblogs.com/ginb/p/7000427.html>

JavaScript API:

<https://www.elastic.co/guide/en/elasticsearch/client/javascript-api/current/quick-start.html> *(English)*

Mapping:

<https://my.oschina.net/davidzhang/blog/811511>

https://stackoverflow.com/questions/37861279/how-to-index-a-pdf-file-in-elasticsearch-5-0-0-with-ingest-attachment-plugin?rq=1

### Logstash

Logstash is the tool to import and update data in Elasticseach’s database. It has the ability to parse unstructured files such as .log, .txt using regular expression. As well as transfer the data in existing Oracle/MySQL database into the Elasticsearch Database directly.

*Learning Map*:

*Reference*:

Official Website:

<https://www.elastic.co/products/logstash>

<https://www.elastic.co/guide/en/logstash/current/getting-started-with-logstash.html> (Series of Install, Guide) *(English)*

Basic Using:

<https://www.cnblogs.com/yincheng/p/logstash.html>

Parse Test Files:

<https://www.elastic.co/guide/en/logstash/6.5/advanced-pipeline.html> *(English)* <http://trumandu.github.io/2016/10/24/logstash%E4%BD%BF%E7%94%A8%E6%95%99%E7%A8%8B/>

Connect to ES: <https://blog.csdn.net/wangnan9279/article/details/79287820>

Connect to Oracle Database (JDBC):

<https://blog.csdn.net/wjacketcn/article/details/50960843> <https://blog.csdn.net/laoyang360/article/details/75452953>

<https://discuss.elastic.co/t/logstash-jdbc-input-oracle-settings/26996> *(English)*

## WebServer

The Webserver Part is responsible for transferring the keywords user inputs and return the result that the elasticsearch engine generates back to the front-end.

### Nodejs

Nodejs is a powerful tool built on JavaScript, it’s easy to operate and expand with other API such as Elasticsearch.

*Learning Map*:

*Reference*:

Official Website + Download:

<https://nodejs.org/en/> *(English)*

Introduce:

<https://codeburst.io/the-only-nodejs-introduction-youll-ever-need-d969a47ef219> *(English)*

Tutorial Series (Recommend):

<https://www.w3schools.com/nodejs/> *(English)*

Connect to ES:

<https://www.oschina.net/translate/search-engine-node-elasticsearch>

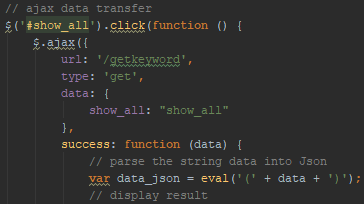
Asynchronization (Advanced):

<https://www.jb51.net/article/63070.htm>

<https://m.jb51.net/article/84148.htm>

### Ajax

Ajax is a popular tool used when building webpage, here we also use it as the tool to transfer data between the web server and the front end.



*Transfer data by Ajax combined with JQuery*

*Learning Map*:

*Reference*:

Introduce:

<https://www.w3schools.com/xml/ajax_intro.asp>

Tutorial Series (Recommend):

<https://www.w3schools.com/xml/ajax_intro.asp>  *(English)*

<http://www.runoob.com/ajax/ajax-tutorial.html>

Data Transmission:

<https://www.jb51.net/article/57874.htm> *(Recommend)*

### JQuery

JQuery is a simple alternative of JavaScript, but it has lots of special functions and unique language format. It is used in this system to combine with Ajax.

It’s not as important as other modules in this system, thus no detailed study of this section in this doc.

## Front-End

The work of front-end is to reformat and present the search result that transferred by the server and generated by the back-end engine, as well as collect the keywords inputted by user.

### Kibana

Kibana is part of the ELK (Elasticsearch – Logstash – Kibana) application stack. It is a tool to visualize the result, but it is replaced by our own webpage and only serve as the query – testing tool to inspect the elasticsearch database.

*Learning Map*:

*Reference*:

Official Website: <https://www.elastic.co/guide/cn/kibana/current/index.html> <https://www.elastic.co/guide/en/kibana/current/index.html> *(English)*

Download:

<https://www.elastic.co/downloads/kibana>

Connect with ES:

<https://www.elastic.co/guide/cn/kibana/current/connect-to-elasticsearch.html>

Tutorial:

<https://www.elastic.co/guide/en/kibana/current/index.html> *(English)*

### Plotly

Plotly is a powerful and flexible graphic plugin built for JavaScript. It is used for visualization in this system.

*Learning Map*:

*Reference*:

Official Website, Download, Tutorial:

<https://plot.ly/javascript/>

## Program Language

I believe a qualified software engineer is capable of handling any unfamiliar program language in very short term in grammar–level. Language is too easy to worry that much for a brilliant programmer. Therefore, this documentation dose not contain detailed guide of any program language, the reader could find various tutorials of basic language grammar knowledge in Google.

# Architecture

This section introduces technical details and explains the code.

## Back-End

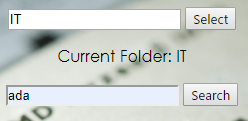
### Elasticearch

The most operation in elasticsearch is query according to the given keywords.

*File: /server\_searchengine\_v3.js*

There are three different ways to search:

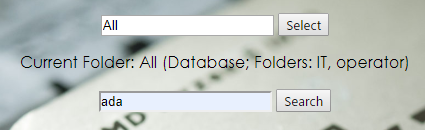
1. Target Folder : text files only



index: index, // the ES database name   
type: 'doc',  
size: results\_number, // number of return result  
body:{   
 query:{ // query body  
 bool:{  
 must:[  
 {match:{"type":"txt"}}, // search in all test files  
 {match\_phrase:{"message": keyword}},// search keyword as whole phrase  
 {match:{"log\_folder": folder}} // match the keyword  
 ]  
 }  
 },  
 aggs: { // group the query result  
 type: {  
 terms: {  
 "field": "type.keyword"  
 },  
 aggs: {  
 folder: { // group by folder firstly  
 terms: {  
 "field": "log\_folder.keyword"  
 },  
 aggs: {   
 log: { // group by log name then   
 terms: {  
 "field": "log\_name.keyword"  
 }  
 }  
 }  
 }  
 }  
 }  
 },  
 \_source:["log\_time", "log\_date","log\_name","message","log\_folder","type"]

// only show partial attributes  
}

1. Search in both database and all files: all data



index: index,  
type: 'doc',  
size: results\_number,  
body:{  
 query:{  
 bool:{  
 should:[   
 {match:{db\_message:{"query":keyword, "operator":"and"}}}, // database  
 {match\_phrase:{"message":keyword}} // files  
 ]  
 }  
 },  
 aggs: {  
 type: {  
 terms: {  
 "field": "type.keyword"  
 },  
 aggs: {  
 folder: {  
 terms: {  
 "field": "log\_folder.keyword"  
 },  
 aggs: {  
 log: {  
 terms: {  
 "field": "log\_name.keyword"  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
}

1. Show the overall data statics



index: index,  
type: 'doc',  
size: results\_number,  
body: {  
 aggs: {  
 type: {  
 terms: {  
 "field": "type.keyword"  
 },  
 aggs: {  
 folder: {  
 terms: {  
 "field": "log\_folder.keyword"  
 },  
 aggs: {  
 log: {  
 terms: {  
 "field": "log\_name.keyword"  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
}

### Logstash

Logstash is needed when import and update data, as well as connect to the conventional database and parse text file.

*File: /conf/logstash/*

1. Import and parse data from text files under folders

input {

file{

path => "C:/Mulong/logs/operator/480b5c800056afd8-(BRCM\_PL\_TNR-BARTMP36).txt"

Set file path

start\_position => "beginning"

codec => multiline{

Define log segment

negate => true

pattern => "(^|\[INFO\]\[)(\d+\-\w+\-\d+)\s(\d+\:\d+\:\d+)"

what => "previous"

Differentiate from database

}

add\_field => {"type" => "txt"}

}

}

filter {

grok {

Parse unique log format

break\_on\_match => false

match => {"message"=>

"(^|\[INFO\]\[)(?<log\_date>\d+\-\w+\-\d+)\s(?<log\_time>\d+\:\d+\:\d+\.?\d\*)[\]\s\t]\*(?<log\_content>.\*)"}

match => {"path"=>"(?<log\_folder>[^\/]\*)\/(?<log\_name>[^\/]\*\.(log|txt))"}

}

}

1. Connect to database

input{

jdbc{

jdbc\_driver\_library => "c:\Mulong\jdbc\ojdbc8.jar"

jdbc\_driver\_class => "Java::oracle.jdbc.driver.OracleDriver"

jdbc\_connection\_string => "jdbc:oracle:thin:edr\_admin/edr\_admin32229@//vpngorasvdlstg:1521/svdlqa"

Connect to the DB server

jdbc\_user => "edr\_admin"

jdbc\_password => "edr\_admin32229"

statement\_filepath => "C:\Mulong\git\github\Elasticsearch-nodejs-UI\nodejs\_project\_AMD\sql\1.sql"

Table name

Content you wanna select

type => "todo"

}

jdbc{

jdbc\_driver\_library => "c:\Mulong\jdbc\ojdbc8.jar"

jdbc\_driver\_class => "Java::oracle.jdbc.driver.OracleDriver"

jdbc\_connection\_string => "jdbc:oracle:thin:edr\_admin/edr\_admin32229@//vpngorasvdlstg:1521/svdlqa"

jdbc\_user => "edr\_admin"

jdbc\_password => "edr\_admin32229"

statement\_filepath => "C:\Mulong\git\github\Elasticsearch-nodejs-UI\nodejs\_project\_AMD\sql\2.sql"

Adding when import new table

type => "userpreferences"

}

## WebServer

### Nodejs

Used to build the web server and excuse the search query according to input keyword, as well as send back the result to front-end.

*File: /server\_router\_v3.js*

Initial page

The first page sent when link to the system.

// \*\*\* router start \*\*\*  
app.get('/', **function** (req, res) {  
 res.sendfile(\_\_dirname + '/public/index\_v3.html');  
});

1. Main server: trigger the search engine.

app.get('/getkeyword', **function** (req, res) {  
 // get keyword from request  
 **var** search = {  
 'folder': req.query.folder,  
 'keyword': req.query.keyword,  
 'show\_all': req.query.show\_all  
 };  
 keycontent = search['keyword'];  
 folder = search['folder'];  
 console.log("\n\ninput folder: " + folder);  
 console.log("Input keyword: " + keycontent);  
  
 // trigger the search engine  
 // search by given keywords  
 es.elasticSearch(search, **function** (result) {  
 **var** response = {}; // the final return response  
 **if**(result){  
 // return variables  
 **var** disp = {}; // the table on website  
 **var** draw\_data = {}; // the data for drawing diagram  
 // show all or show details  
 **if**(search['show\_all']){  
 disp = ui.disp\_overview(result, draw\_data);  
 }  
 **else**{  
 **var** overview = ui.disp\_overview(result, draw\_data);  
 disp = ui.disp\_detail(result, keycontent);  
  
 disp['txt'] = overview['txt'] + disp['txt'];  
 disp['db'] = overview['db'] + disp['db'];  
 }  
  
 // gather the results  
 response['disp'] = disp;  
 response['draw\_data'] = draw\_data;  
 response['status'] = 1;  
 res.setHeader('Content-Type', 'text/html');  
 res.end(JSON.stringify(response));  
 }  
 **else**{  
 response['disp'] = "<h3>No related result found by given keyword in target folder: " + search['folder'] + '/' + search['keyword'] + "</h3>";  
 response['status'] = -1;  
 res.end(JSON.stringify(response));  
 }  
 });  
});

1. Download

app.get('/download', **function** (req, res) {   
 **var** file = \_\_dirname + '\\logs\\' + req.query.file;  
 console.log("Download " + file);  
 res.set({  
 'Content-Type': 'application/octet-stream',  
 'Content-Disposition': 'attachment; filename=' + req.query.file  
 });  
 fs.createReadStream(file).pipe(res);  
});

### Ajax

Use to transfer data and collect the return data from the server, as well as defining the on\_click function.

*File: /public/index\_v3.html*

<script type="application/javascript">  
 // ajax data transfer  
 $('#show\_all').click(**function** () {  
 $.ajax({  
 url: '/getkeyword',  
 type: 'get',  
 data: {  
 show\_all: "show\_all"  
 },  
 success: **function** (data) {  
 // parse the string data into Json  
 **var** data\_json = eval('(' + data + ')');  
 // display result  
 document.getElementById('folder\_name').innerText = 'Current Folder: All logs and database on server';  
 document.getElementById('table\_txt').innerHTML = data\_json['disp']['txt'];  
 document.getElementById('table\_db').innerHTML = data\_json['disp']['db'];  
  
 // change display area  
 document.getElementById('rt').style.display = 'block';  
 document.getElementById('rt').style.backgroundColor = '#00B7FF';  
 document.getElementById('nav\_bar').style.display = 'block';  
 document.getElementById('nav\_data\_source').style.display = 'block';  
 document.getElementById('search\_failed').style.display = 'none';  
 document.getElementById('disp\_table').style.display = 'block';  
 document.getElementById('disp\_plot').style.display = 'none';  
  
 // change button status  
 document.getElementById('butt\_table').className = 'active';  
 document.getElementById('butt\_plot').className = '';  
  
 plot(data\_json['draw\_data']);  
 }  
 })  
 });