# ELK Stack Configuration and Example Code

The Statistical Dashboard has some dependencies on the Apache Proxy access logs, with the possibilities of using Tomcat access logs as well as the Apache Proxy error logs.  These logs require parsing and shipping to an Elasticsearch intense for parsing into indexed documents which can be served up, in turn, to an application framework such as Spring Boot, which can support a UI for dashboard display.  We are providing configuration files and sample code as prototyping support should we use this approach for a dashboard back end.

## ELK Stack

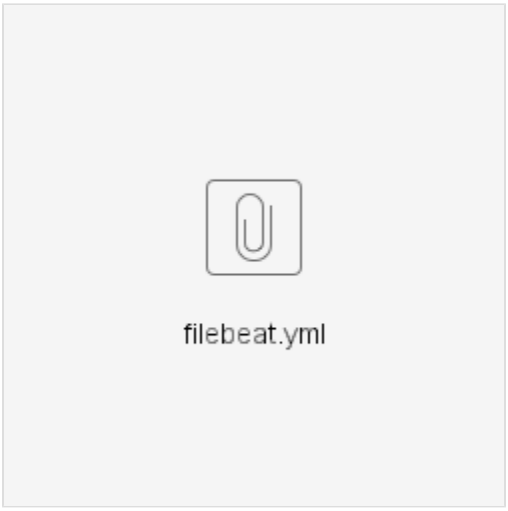
We built our prototypes with a minimal approach using the following applications as the tech stack.  We may add Logstash if we see a need for its filtering capabilities

- Filebeats
- Elasticsearch
- Kabana

### Elasticsearch Configuration - default

### Kabana Configuration - default

### Filebeat Configuration




### Apache Module Configuration

```
# Module: apache
# Docs: https://www.elastic.co/guide/en/beats/filebeat/7.9/filebeat-module-apache.html

- module: apache
  # Access logs
  access:
    enabled: true
    var.paths: ["/var/log/apache/access.log"]
    ##The type:access_log will help us point these logs to the right direction
    #input:
     # processors:
       # - add_fields:
           # target: fields
            #fields:
             # codec: plain
              #type: access_log
~
```

## Java Code Snippet for Elasticsearch Query Over Indexes of Parsed Logs

We'll need a java implementation to Elasticsearch's REST API to provide to the service side of a Spring Boot based web application UI.

**RestTest.java**

```
package search.client.test;

import java.io.IOException;
import java.util.concurrent.TimeUnit;

import org.apache.http.Header;
import org.apache.http.HttpHost;
import org.apache.http.client.config.RequestConfig;
import org.apache.http.message.BasicHeader;
import org.elasticsearch.action.search.SearchRequest;
import org.elasticsearch.action.search.SearchResponse;
import org.elasticsearch.client.Request;
import org.elasticsearch.client.RequestOptions;
import org.elasticsearch.client.Response;
import org.elasticsearch.client.RestClient;
import org.elasticsearch.client.RestClientBuilder;
import org.elasticsearch.client.RestHighLevelClient;
import org.elasticsearch.common.unit.TimeValue;
import org.elasticsearch.index.query.QueryBuilders;
import org.elasticsearch.search.SearchHits;
import org.elasticsearch.search.builder.SearchSourceBuilder;

public class RestTest {


        public void run() {

        RestClientBuilder builder =  RestClient.builder(
                new HttpHost("localhost", 9200, "http"));
        Header[] defaultHeaders = new Header[]{new BasicHeader("header", "value")};
        builder.setDefaultHeaders(defaultHeaders);
        builder.setRequestConfigCallback(
                    new RestClientBuilder.RequestConfigCallback() {
                        @Override
                        public RequestConfig.Builder customizeRequestConfig(
                                RequestConfig.Builder requestConfigBuilder) {
                            return requestConfigBuilder.setSocketTimeout(10000);
                        }
                    });

        RestHighLevelClient restClient = new RestHighLevelClient(builder);
        SearchSourceBuilder sourceBuilder = new SearchSourceBuilder();
        sourceBuilder.query(QueryBuilders.termQuery("event.outcome", "failure"));
        sourceBuilder.from(0);
        sourceBuilder.size(5);
        sourceBuilder.timeout(new TimeValue(60, TimeUnit.SECONDS));

        SearchRequest searchRequest = new SearchRequest();
        //searchRequest.indices("posts");
        searchRequest.source(sourceBuilder);

        SearchResponse searchResponse = null;


        try {
                searchResponse = restClient.search(searchRequest, RequestOptions.DEFAULT);
                restClient.close();
        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }

        SearchHits hits = searchResponse.getHits();
        hits.forEach(x -> System.out.println(x.toString()));
        }
```

```
        public static void main(String ...strings ) {
                new RestTest().run();
        }
}
```