ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA034

SCADA STATISTICS MONITORING USING THE Elastic Stack (Elasticsearch, Logstash, Kibana)

James Hamilton, Brad Schofield, Manuel Gonzalez Berges, Jean-Charles Tournier
CERN, Geneva, Switzerland

Abstract

The Industrial Controls and Safety systems group at CERN, in collaboration with other groups, has developed and currently maintains around 200 controls applications that include domains such as LHC magnet protection, cryogenics and electrical network supervision systems. Millions of value changes and alarms from many devices are archived to a centralised Oracle database but it is not easy to obtain high-level statistics from such an archive. A system based on Elasticsearch, Logstash and Kibana (the Elastic Stack [1]) has been implemented in order to provide easy access to these statistics. This system provides aggregated statistics based on the number of value changes and alarms, classified according to several criteria such as time, application domain, system and device. The system can be used, for example, to detect abnormal situations and alarm misconfiguration. In addition to these statistics each application generates text-based log files which are parsed, collected and displayed using the Elastic Stack to provide centralised access to all the application logs. Further work will explore the possibilities of combining the statistics and logs to better understand the behaviour of CERN's controls applications.

INTRODUCTION

There are around 200 controls applications maintained by the Industrial Controls and Safety systems group at CERN. Managing statistics, logs and detecting misconfigurations from these applications is difficult. This paper describes the service that we have implemented in order to more easily obtain statistics and error logs from all applications through a centralised web application.

Value Change and Alarms Statistics

Most of the controls applications archive value changes and alarms to a centralised Oracle database. The history of values changes and alarms for hundreds of thousands of devices are archived in a centralised Oracle High Performance Real Application Cluster (RAC), from around 200 controls applications. Mainly due to the structure of the database and the huge amount of data it is not easy to view, analyse or use this data to obtain high-level statistics.

It is not possible to easily obtain, for example, a list of devices that are archiving an excessive number of value changes. This is important to detect as it could indicate a faulty or misconfigured device.

The controls applications also generate alarms and it should also be possible to easily obtain information on the number of alarms from each application and device.

WinCC OA Logs

All the controls applications log errors, warnings and information messages to local log files on the servers on which they are running. It is difficult to examine these logs as they are distributed across many servers. For example, developers of the applications cannot easily or search these logs. It should be possible to collect and examine these logs centrally without putting an additional load on the production systems. In addition, since the log files on the servers are rolling, old log entries are lost when the log files are overwritten.

TECHNOLOGY

The implementation of the SCADA statistics monitoring service is built on the Elastic Stack [1] – Elasticsearch, Logstash, Kibana and Filebeat.

Elasticsearch is a distributed database that stores JSON documents designed specifically for search and analytics of semi-structured data. Unlike a relational database management system it is schema-free although it is not schema-less (like MongoDB for example) meaning that it is not required to define the types (string, number, etc) of the data before inserting it but it is possible to define types. The underlying technology is the Apache Lucene text search engine [2].

Logstash is a "data processing pipeline" that can ingest data from various sources, transform it and send it to various consumers; Elasticsearch is one of the many consumers that can be used with Logstash.

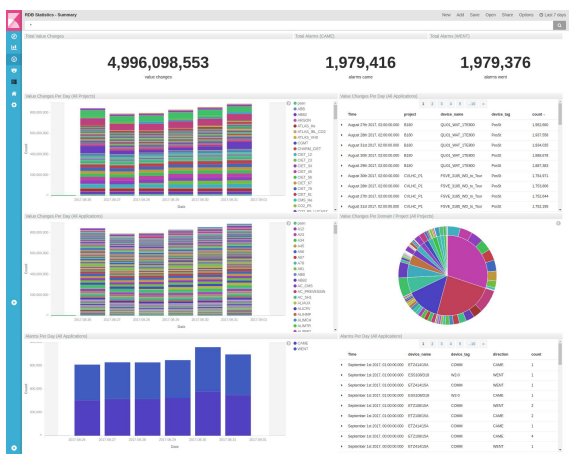

Figure 1: Value Change and Alarm Statistics Dashboard.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA034

Kibana is a web based visualisation tool that integrates with Elasticsearch to provide easy ways to navigate and visualise data, using a variety of graphs, charts and tables. See Figure 1 for an example of a Kibana dashboard.

Filebeat is one of the many lightweight 'data shippers' available as part of the Elastic Stack known as Beats. Beats data shippers are single purpose and designed to be installed on the machine that generates the data without having any impact on the performance of the machine. Filebeat reads text-based log files and forwards them either directly to Elasticsearch or to Logstash.

The Elastic Stack is open source but certain features are available only as part of the commercial X-Pack extension; for many use-cases (including ours) the open source stack is sufficient.

IMPLEMENTATION

Using the Elastic Stack we have implemented a web service which allows users to examine, visualise and query statistics; and to view and search application logs on the web.

Value Changes & Alarms

We solve the problem of gathering statistics on value changes & alarms by executing queries daily to obtain aggregate statistics, as shown in Figure 2. The basic statistics gathered for all applications are:

- The number of value changes per day, for each device
- The number of alarms per day, for each device
- The number of equipment disconnections
- The equipment disconnection durations

So, when users perform queries in Kibana they will not see live data but aggregated data from the previous days. Apart from those listed above, some applications require domain specific reporting which means running other specialised queries.

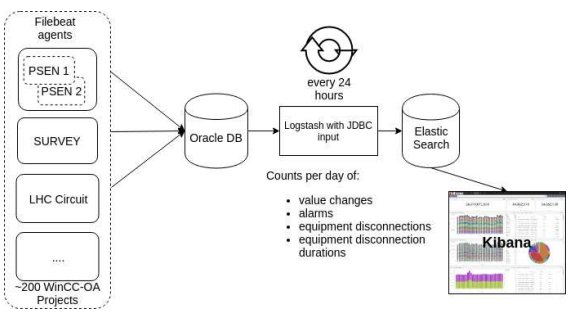

Figure 2: Logstash setup for collecting statistics.

Each JSON document in Elasticsearch contains the count of value changes or alarms, and metadata such as the domain, application and device names. This allows building visualisations in Kibana at the domain, application or device level.

We execute the queries using Logstash and the *logstash-input-jdbc* plugin [3] which allows queries to be scheduled for daily execution for each application[1]. This instance of Logstash is always running, waiting for the scheduled queries to be executed.

WinCC OA Logs

In order to collect the text-based logs from every application, which are spread across many servers, we use Filebeat [4]. This is a lightweight application that is designed to read log files without putting too much load on the system which is a production machine.

Filebeat itself does very little processing – it's main task is to watch a folder of files for changes and forward the changes, line-by-line, to Logstash.

Logstash instances are awaiting input from Filebeat instances. Logstash is concerned with receiving lines from a log file, collating multi-line messages and parsing the text into a structured JSON message; the structured JSON message is then sent to Elasticsearch for storage.

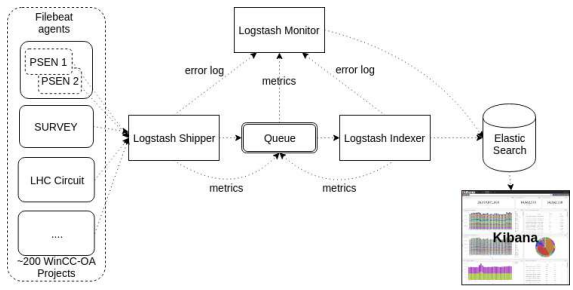

Figure 3: Logstash setup for collecting logs.

The main challenge with parsing log files generated by the controls applications is their unstructured nature and the variety of different formats. We make heavy use of the Logstash filter called *grok* to convert these unstructured log entries into structured documents for storing in Elasticsearch.

An example of a basic WinCC OA log entry is shown in listing 1; this is a log entry composed of several fields that we are interested in parsing. We can use a regex pattern to extract the WinCC OA manager, timestamp, severity, error code and message.

```
WCCOAui (2), 2016.04.05 13:47:15.070, PARAM,SEVERE, 19, The attribute does not
        exist in this config
```

Listing 1: Example WinCC OA log

The biggest problem is the huge variety of non-standard log files generated by the controls applications. We attempt to parse as many, as correctly as we can but cannot always guarantee that every message will be parsed correctly. There are around 80 different regular expressions that are used

[1] each application stores data in its own separate schema but within the same database

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA034

depending on the type of log file and most log file types require trying multiple regular expressions until a match is found. In any single type of log file there could be, for example, differently formatted dates and various forms of error message.

A useful tool to help create such regular expressions is the online Grok Constructor tool [5] which enables them to be created incrementally from example log entries.

Figure 3 shows the Logstash pipeline for collecting and parsing the WinCC OA logs – we divide the pipeline into two: the *shipper* and the *indexer*. The shipper receives log messages from Filebeat and concatenates multi-line messages. The concatenated messages are sent to a queue[2] [6] and one or more indexers read logs from the queue, parse them and send the structured log entry to Elasticsearch. The main advantage of using a queue is that we can easily scale up the parsing simply by adding more indexers.

Logstash Configuration Generator

Soon after we started development of the service using the Elastic Stack, we realised that the Logstash configurations became unmanageable – especially due to the high number of SQL queries that we require (currently ~500) where each query requires a separate entry within the configuration file.

To solve this problem we developed a Python application that generates our configs (see Figure 4). The Logstash Configuration Generator (*LCG*) uses the Jinja2 template engine [7] which allows us to create templates from which the configuration files are generated.

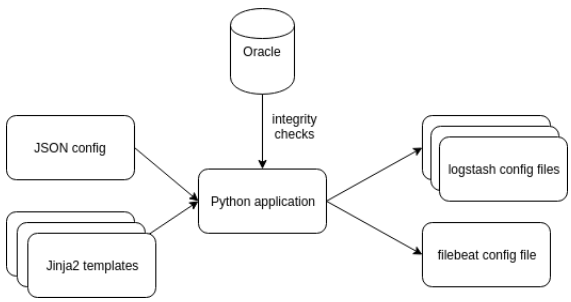

Figure 4: Logstash Configuration Generator Architecture.

We have created a custom configuration format (encoded as JSON) that allows us to define reports (and which templates those reports use) and list the controls applications for which reports should be generated. The generator will then generate, for each application and report, a section of the configuration file.

During execution, the *LCG* also does some integrity checks against the Oracle database – to check, for example, that the applications listed in the config exist and have the correct parameters, such as the correct application ID.

We generate files for the Logstash statistics, shipper, indexer and monitor instances. The generated configuration file for the Logstash statistics is the largest and contains approximately 30,000 lines, comprised mostly of SQL queries.

[2] Redis, the open source in-memory data structure store [6].

We generate one Filebeat configuration file with all the controls applications log file paths listed – Filebeat will ignore a path if it doesn't exist – which simplifies the deployment of the Filebeat configuration.

Logstash Monitoring

We have an additional Logstash instance, known as the monitor (as shown in Figure 3), that collects error logs and throughput statistics from the other Logstash instances and queue. This allows us to use a dashboard in Kibana (see Figure 5) to monitor the status of the setup, for example, to see errors, the throughput of the shippers & indexers and the size of the queues.

Figure 5: Logstash Monitoring Dashboard.

DEPLOYMENT

We have deployed the service using the OpenStack service provided by the CERN IT department which enables a quick setup and the easy addition/removal of Logstash indexers. The CERN IT department also provides Elasticsearch as a service.

For redundancy there are two Logstash shippers, two queues, and three Logstash indexers.

All Filebeat instances will send logs to either of the shippers; while the output from the shippers is sent to the primary queue or the secondary queue if the first queue is not available. If the queue becomes full it will reject new inputs meaning that Logstash will not be able to send any more logs and therefore the Filebeat instances will stop sending logs until the queue is cleared.

There are three Logstash indexers that read from the queues which parse the log entries. We can add as many

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA034

indexers as required without any change to any of the configurations.

We use Jenkins [8] to execute the Logstash Configuration Generator application whenever we make changes to its configuration which will generate the new Logstash and Filebeat configuration files. We also have Jenkins jobs to allow us to deploy, restart and reload configurations with a single click.

SERVICE STATISTICS

The SCADA statistics service has been running for more than a year now and covers 26 domains comprising 145 applications. There are around 800, 000, 000 value changes per day and around 600, 000 alarms per day recorded across all applications. This means an average of around 2, 000, 000 documents stored in Elasticsearch per day for value change & alarm statistics (as each JSON document stores an aggregated count value).

WinCC OA text logs are collected for 142 applications with around 1 million log entries per day across all applications. Some applications produce more logs than others and there are often spikes of log activity (as can be seen at the bottom of Figure 6).

CASE STUDIES

This section describes some specific use cases where the service has been useful.

Faulty Archiving Configuration

The high level statistics for value changes has proven very useful in identifying misconfigurations in archive settings for process data. Since the majority of the 200 control systems archive their data to a common database cluster, misconfigurations on any one individual system may be easily missed.

As an example, the application LHC Circuit which monitors the roughly 1600 power converters for the LHC was found to have a faulty archiving setting for the reference voltages of many devices. As an analog value, the voltage reference would typically be expected to have the customary deadband filtering.

However, a number of devices had been configured with 'on change' archiving, meaning that voltage reference values were being sent to the archive at their sampling rate of approximately 2Hz. This translated to over one hundred thousand value changes per day, which stood out very clearly in the value change statistics. For example, in Figure 1 the bar charts are split into slices for each application and it can be seen which applications have higher numbers of value changes & alarms. Thanks to the service the archive settings could be corrected for all affected devices.

WinCC OA Log Availability

The statistics service provides a central entry point for accessing log information for all WinCC OA applications at CERN (see Figure 6). This is particularly valuable for application domains in which the control systems are highly

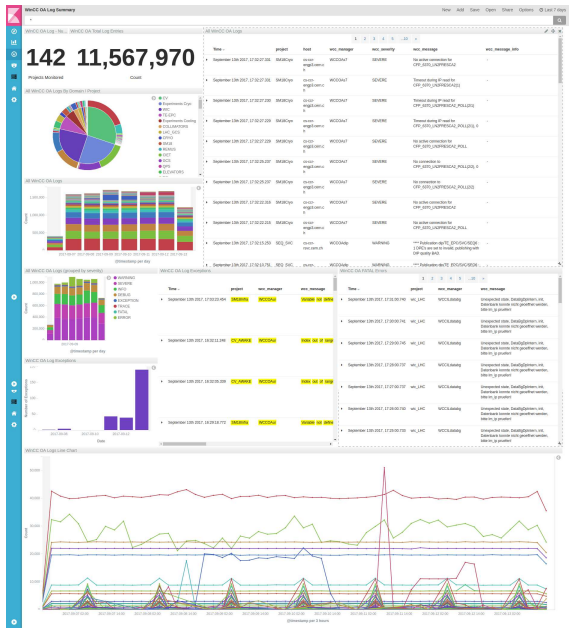

Figure 6: WinCC OA Log Dashboard.

distributed, which implies that logs are located on many different servers.

Another important feature of the service is that it provides greater persistence of log entries. Since the WinCC OA logs are rotated, eventually log entries will be overwritten and will be lost on the server. This rotation may depend on the rate at which entries are being made in the logs, which at certain times can be very high. Typically, times at which the log entry rate is high are also times at which the system experts are most interested in accessing the log entries, such as during control system upgrades. The service ensures that all log entries are available for longer periods of time.

WinCC OA Log Entry Statistics

In addition to simply making the logs available centrally, the service's ability to parse the log entries enables statistics to be created over various entry types. This in turn allows diagnostics to be made at a high level. As an example, a large number of log messages related to middleware communication was used to identify the loss of communication with certain pieces of equipment, which could then be rectified.

Electrical Network Supervision

The supervision system monitoring the CERN electrical network has benefited from the service in two different ways and for two different audiences (Figure 7 shows the Electrical network supervision dashboard).

Firstly, it has benefited the operation team in charge of the electrical network by providing the means to compute key performance indicators (KPI) such as SAIDI (System Average Interruption Duration Index) or SAIFI (System Average Interruption Frequency Index) in order to provide a quick overview of the level of service provided to the different CERN consumers. These KPIs can now be easily computed

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA034

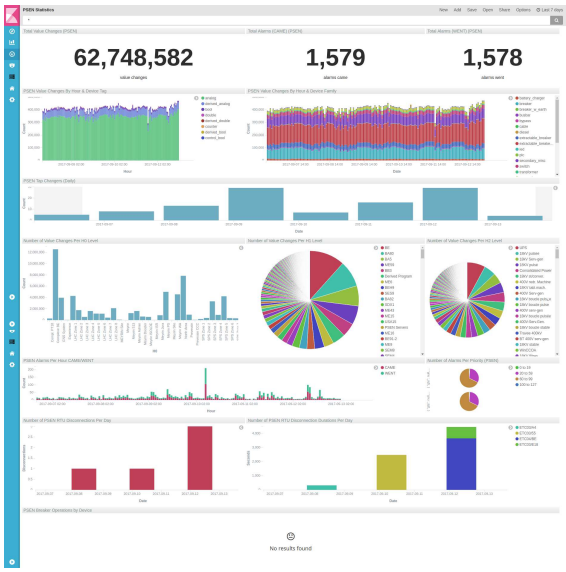

Figure 7: Electrical Network Supervision Dashboard.

for the primary equipment (e.g. how often a specific client has been without power), but also for the secondary equipment (e.g. how often and for how long a part of the network was not supervised because of the loss of connection with the acquisition devices).

Secondly, it has benefited the team in charge of the development of the supervision system itself by providing key indicators related to the log messages created by the system. It is therefore an additional tool to detect and diagnose abnormal situations such as a sudden change in the number of log messages registered for a particular process. It is also a complementary tool to the current supervision mechanisms as it allows storing & searching log messages for a longer period than the local logging storage (typically years vs. days).

This use case goes beyond the original intentions of the service by doing process related calculations which shows the usefulness of the service and how further services can be built on-top of it.

CONCLUSION

Industrial controls applications at CERN produce lots of data in different areas including an archive of process and system data and log files. There are many activities going on to analyse this data and get value from it [9, 10] This paper presents a high-level approach where the data is mainly viewed as aggregated statistics and correlations. Only in some cases is the actual data processed in detail. This approach gives good results for the use cases mentioned in the paper, among others, and has the potential to be extended. The service presented has been built on a modern state of the art set of tools – the Elastic Stack – that greatly facilitated the task.

FUTURE WORK

There is a huge amount of 'noise' in the WinCC OA controls application logs which provides a good candidate for anomaly detection using machine learning. It is currently difficult to find the 'true' errors when looking through the log and an automated method of detecting anomalies would be very helpful for the application developers. One option to consider is the newly released machine learning features of X-Pack.

Another X-Pack feature that could greatly improve the service would be the Alerting component. Currently, users have to look at the dashboards to diagnose problems but with the Alerting component we could notify users whenever thresholds are exceeded.

Furthermore, we believe that the combination of value & alarm statistics and application logs could lead to some interesting results to better understand the behaviour of CERN's controls applications. For example, if there are errors in the log there could be corresponding alarms.

REFERENCES

[1] Elasitc, "Elastic stack", https://www.elastic.co/, 2017.

[2] Apache, "Lucene", https://lucene.apache.org/core/, 2017.

[3] Elastic, "logstash-input-jdbc plugin", https://www.elastic.co/guide/en/logstash/current/plugins-inputs-jdbc.html, 2017.

[4] Elastic, "Filebeat", https://www.elastic.co/products/beats/filebeat, 2017.

[5] H.-P. Stőrr, "Grok constructor", http://grokconstructor.appspot.com/, 2017.

[6] R. Labs, "Redis", https://redis.io/, 2017.

[7] A. Ronacher and community, "Jinja2", http://jinja.pocoo.org/, 2017.

[8] Jenkins, "Jenkins", https://jenkins.io/, 2017.

[9] P. J. Seweryn, F. M. Tilaro, J. B. Schofield, and M. G. Berges, "Data analytics reporting tool for cern scada systems", presented at ICALEPCS'17, Barcelona, Spain, Oct 2017, paper TUPHA035, this conference.

[10] F. Tilaro, B. Bradu, M. G. Berges, F. Varela, and M. Roshchin, "Model learning algorithms for faulty sensors detection in cern control systems", presented at ICALEPCS'17, Barcelona, Spain, Oct 2017, paper TUCPA04, this conference.