TAMPEREEN
AMMATTIKORKEAKOULU

IMPLEMENTING AUTOMATED LOG BASED ALERTS IN A PATIENT INFORMATION SYSTEM

Henri Virtanen

Opinnäytetyö
Toukokuu 2017
Tietojenkäsittely
Tietoverkkopalvelut

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittely
Tietoverkkopalvelut

VIRTANEN, HENRI:
Automatisoidun hälytysjärjestelmän toteuttaminen potilastietojärjestelmässä

Opinnäytetyö 27 sivua
Toukokuu 2017

Opinnäytetyön tavoitteena oli Acuvitec Oy:n Acute-potilastietojärjestelmän eri palveluiden ja liitoskohtien reaaliaikainen seuranta lokien perusteella ja niihin pohjautuvat automatisoidut hälytykset. Ympäristön laajentuessa loki-pohjainen monitorointi tulee yhä haastavammaksi, koska muodostuvien lokirivien määrä kasvaa. Tämän haasteen ratkaisemiseksi muokattiin nykyistä seurantajärjestelmää ja lisättiin siihen uusia ominaisuuksia. Tavoitteena oli kohentaa reaaliaikaista tilannetajua ympäristössä. Käytettävät työkalut ovat ELK Stack (Elasticsearch, Logstash ja Kibana) sekä Elasticalert.

Opinnäytetyö suoritettiin konstruktiivisena tutkimuksena. Tavoitteen toteuttaminen aloitettiin kartoittamalla tämänhetkinen tilanne. Kartoituksessa tutkittiin senhetkistä seurantajärjestelmää ja haastateltiin Acuvitecin työntekijöitä. Haastatteluiden tavoitteena oli selvittää, mitä tulevalta seurantajärjestelmältä halutaan ja millaisia tavoitteita järjestelmälle asetetaan.

Toteutettua automatisoitua hälytysjärjestelmää ei ole vielä otettu käyttöön tuotantoympäristössä, mutta tehdyissä testeissä se on todettu toimivaksi työkaluksi. Suurin tämänhetkinen hyöty toteutuneesta opinnäytetyöstä on nykyiseen seurantajärjestelmään tehdyt muutokset hälytyksiä varten, koska niiden avulla saadaan graafisesta seurantatyökalusta huomattavasti enemmän irti. Tulevaisuudessa olisi vielä tarkoitus ottaa kaikki ympäristön palvelut ja integraatiot mukaan automatisoidun seurannan piiriin.

Asiasanat: ELK-Stack, Elasticalert, potilastietojärjestelmä, loki, hälytys

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Option of Network Services

VIRTANEN, HENRI:
Implementing Automated Log Based Alerts in a Patient Information System

Bachelor's thesis 27 pages
May 2017

The goal of this thesis was to implement a real-time log based situational awareness and alert system to a patient information system. To achieve real-time awareness in the system, the pre-existing solution was modified and a number of new features were added to it. The tools used in the implementation were ELK-Stack (Elasticsearch, Logstash and Kibana) and Elasticalert.

The process started by mapping the current situation. The mapping was achieved by investigating the existing solution and interviewing the employees of Acuvitec. Interviews were also held to find out about the desires, concerns and requirements the staff had regarding the situational awareness system.

The new automated log based alert system developed and implemented in this project is yet to be deployed to the production environment, but in the tests done in a test environment proved it as a functioning tool. The greatest benefits were brought about by the modifications made to the former alert system. These changes increased the capability to use the graphical visualization tool Kibana more efficiently.

Key words: ELK-Stack, Elasticalert, patient information system, log, alarms

SISÄLLYS

ACRONYMS AND ABBREVIATIONS

IIS	Internet information services
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
SaaP	Software as a Product
ELK Stack	A group of services combined from Elasticsearch, Logastash, Kibana
ePrescription	An electronical medication prescription
eArchive	Centralazied database of patient data maintained by Kela
SFTP	Secure File Transfer Protocol
Sysadmin	System administrator
DMZ	Demilitarized zone

1 INTRODUCTION

In the scope of this thesis the patient information system of Acuvitec Oy the daughter company of Vitec Software Group will be analyzed. The patient information system will be later referred as Acute. Acute is a browser based Software a as Service product. Acute is used by multiple private, public and occupational health centers. Physiotherapists also use Acute. Users handle highly confidential data with Acute so secure and reliable data management is a necessity.

The system and environment changed on 2016 when the student Timo Hulkkonen of Jyväskylä University of applied sciences produced a thesis for his master's degree program in information technology. In the thesis carrying the name of "Implementing situational awareness" Hulkkonen created a system to the environment that helped follow the vast data flow in the Acute ecosystem. This was also required in the new auditing of service providers to achieve an A-grade. A-grade service providers were required to distinguish anomalies from the system in real-time. The mentioned anomalies could be found from DNS-services, firewall, windows event log. To solve this challenge ELK-Stack was implemented as a part of the Acute environment.

ELK-Stack now produced a near real-time log flow but requires active following thus requiring at least one employee's contribution. Therefore, the next step in development would be implementing a system that would automatically follow the log flow and in pre-configured thresholds set off alarms which would be forwarded to the system administrators. In this thesis, a solution is researched and implemented to the system to automate the following of logged data.

Employees of Acuvitec will be interviewed to find out what information should be followed and alarmed of. A plugin called Elastalert will be used to analyze data and set off alarms. Elastalert is a functionality developed to be used with ELK-Stack.

2 BACKGROUND

2.1 Necessity

The company requires an automated log alert solution to follow an increasing amount of log data. As Hector Angulo mentions in his article (2015), most log management platforms simply throw tools your way and don't do much to remove the onerous work that is log analysis. (Hector Angulo, 2015.)

There is also a reliability aspect in this because a human eye can miss trends that a computer might not. In addition, considering the fact the millions of lines of log data are produced every day, there is no way a person can analyze that much of data. (Aviv Raff, 2014)

2.2 Benefits

By implementing an automatic log analyzer, the company will be able to direct resources to other tasks and projects when employees do not need to actively follow logs. In an annual scale, the savings for the company will be tremendous.

Automatic log management ensures that log data is stored centralized and in a sufficient detail. Reviewing log history, comparing logs from different sources and following trends helps establishing a baseline for the system. This will help system administrators in analyzing system health. (Karen Kent, Murugiah Souppaya, 2006)

2.3 Situation now

The log data is collected from hosts with an agent called NxLog and forwarded to a centralized log server. The log server is composed of three services Logstash, Elasticsearch and Kibana. Logstash is a log-parsing engine and is used to forward the incoming data to Elasticseach. Elasticsearch is used to index the data and it works as a search engine. Kibana is a tool to make queries and form graphical interfaces of saved Elasticsearch indices. The combination of these three parts is called ELK Stack. The

current system requires an employee to follow the log flow actively from Kibana, which eats resources.

As shown in figure 1 Nxlog is outside the ELK-Stack ensemble since it is installed on all target servers for log collection. ELK-Stack services are installed only on one server, the ELK-Stack server.

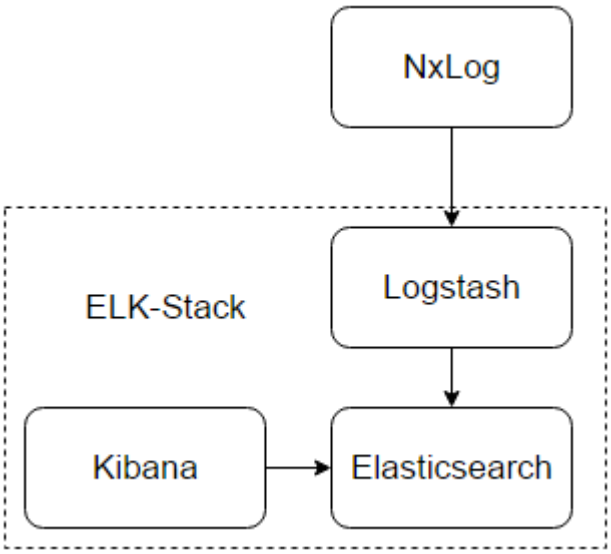

FIGURE 1. Representation of the log processing

3 CLOUD ARCHITECTURE

3.1 Cloud computing

The keywords in cloud architecture and computing are flexibility, scalability, reliability and security. Whether your company needs tens or hundreds of email addresses or computing power for a hundred to thousand user software, can the cloud handle them with flexibility and dynamic scalability. Pricing of the provided services is usually based on use, so you “pay for what you use”. This is a compelling pricing model because of its small initial costs. Reliability and security come hand in hand since cloud computing services have usually failover features and meet the latest security audits. On the other hand, cloud architecture might require new or different security measures compared to traditional server hosting because of centralized instances.

A good example of cloud architecture was demonstrated in a publication (2014) of George Hynes. He referred the cloud service as Android phone. IaaS being the hardware, PaaS being the Android operating system, and SaaS being the apps available for download at Google Play. This can also be illustrated in a pyramid model as shown in figure 2.

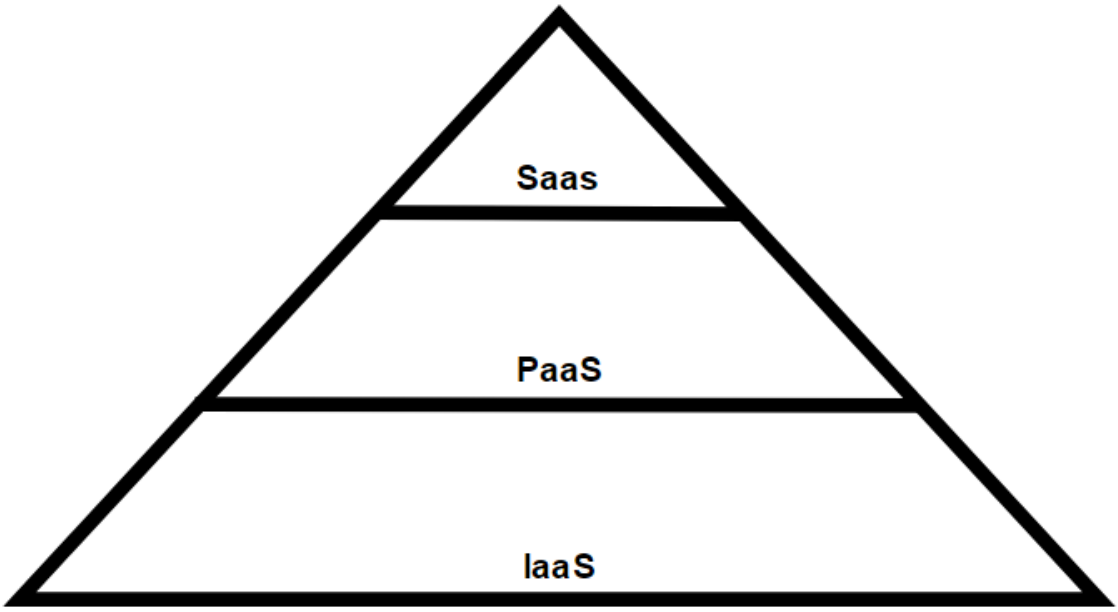

FIGURE 2. Representation of cloud architecture building blocks

As demonstrated in figure 3, IaaS is the least used model because of it having only the most basic components. IaaS model is used by network and infrastructure architects.

Following IaaS is PaaS which has more components and is often used by application developers. Last but the most used model SaaS is already a finished product and is used by the end users.

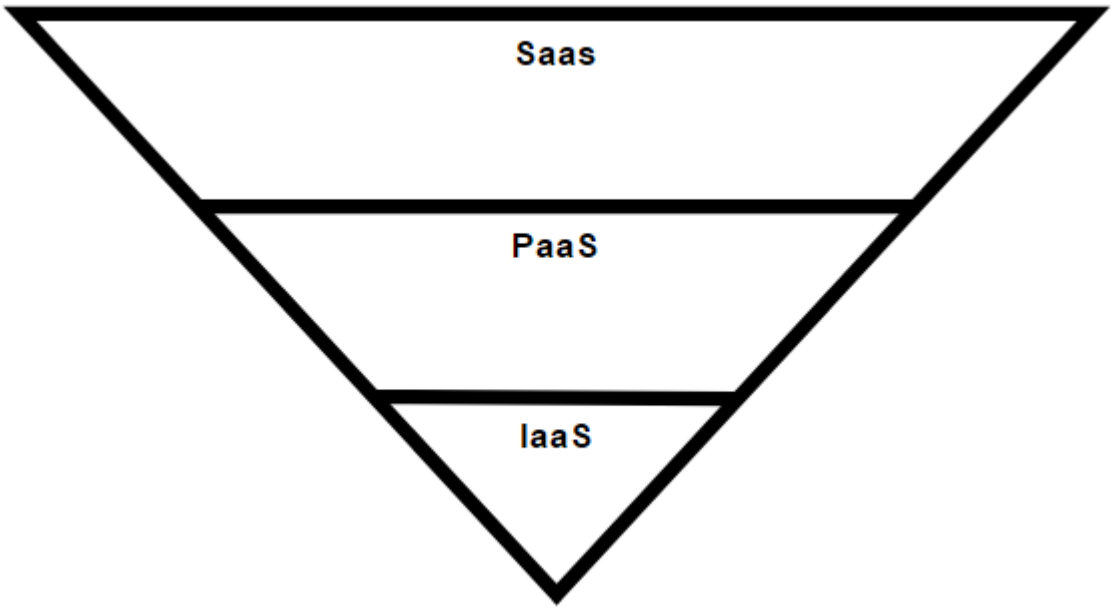

FIGURE 3. Inverted pyramid of Cloud computing based on number of potential users

3.2 Infrastructure as a Service

IaaS (Infrastructure as a Service) is the backbone of cloud architecture. In an IaaS environment, all the basics are provided such as computing, storage and networking. It removes the aspect of hardware and operational maintenance. The users of this model have programmatic access to configure and manage the mentioned blocks, but do not control the cloud infrastructure itself.

IaaS is usually deployed as a shared service thus lowering initial costs compared to purchasing the hardware yourself. The flexibility and dynamic scalability of resources are useful in situations where usage varies. The burden of malfunctioning hardware and networking problems are transferred to the provider. The disadvantages of this model might occur in greater long-term costs and dependence on the vendor’s capabilities.

3.3 Platform as a Service

PaaS (Platform as a Service) is built on top of IaaS. In PaaS, all the basic blocks mentioned in IaaS are provided. In addition tools, libraries and platforms for software development and hosting are provided from the provider. PaaS can be found in three types of systems; add-on development, stand-alone and application delivery-only environments. In addition to these, there is Open Platform as a Service.

Add-on development meaning customization of an existing SaaS product. This type often requires developers to purchase license to the SaaS product in hand. Stand-alone environments supply a generic development environment. Application delivery-only environments provide platforms for only hosting-level services. Open Platform as Service, which provides a sandbox for development and hosting, fully customizable by the developer.

The advantages of PaaS is stripping out all the imaginable non-development and hosting related aspects. This allows the developers to center their full attention to developing and hosting. Monetary savings will consist of mainly human resources since no network or architecture engineers are needed.

3.4 Software as a Service

SaaS (Software as a Service) is the tip of cloud computing. In this model the software is installed and maintained in the cloud. The software is used by clients from their cloud clients over the Internet or Intranet. SaaS can be implemented with or without multitenancy or a hybrid of these. In a multitenant environment, a large number of customers is managed in a single instance. In a non-multitenant environment, the instances can be divided to an instance specific per customer with mechanisms such as virtualization. An example of a hybrid could be virtualized multitenant environments grouped by customer groups.

The advantages of the SaaS model are version control, iterative updates, scalability, flexibility and reducing initial costs for both end user and developer. Version control and iterative updates are easy to manage because only one version of the software is used in a single instance multitenant environment. However, if more than one version is desired

to be deployed, is this possible in a virtualized non-multitenant or a hybrid environment. Multitenancy also being the reason for lowered costs because of shared use.

3.5 Cloud computing case Acute

Acute is mainly provided to the customers as a Software as a Service. Acuvitec hosts the software and its services on an Open Platform as a Service model provided by a 3rd party company. End users use the software over a secured connection over internet or a using a virtual private network. Acuvitec has a hybrid virtualized multitenant model, which homes one to many client instances per virtualized servers. The hybrid model enables having different versions deployed simultaneously, but also making it easier to release updates to the multitenant virtualized servers.

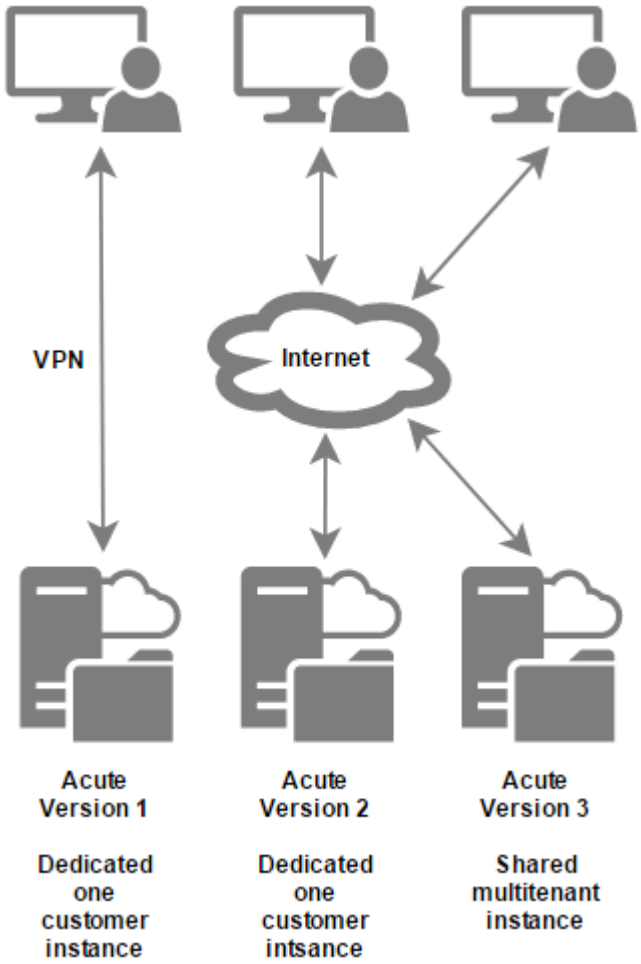

FIGURE 4. Examples of SaaS models for customers

4 ENVIRONMENT

4.1 SaaS

4.1.1 Platform

The environment the setup will be applied to is vast structure of database, application, message and integration servers. The most simple basic server structure would consist of an integrated application, database and messaging server. This kind of environments are usually only seen on license clientele. License clients maintain their own server environment and buy the product as a SaaP instead of SaaS. However, in the scope of this thesis we will apply the alert solution to a SaaS environment with dedicated servers as shown in figure 5.

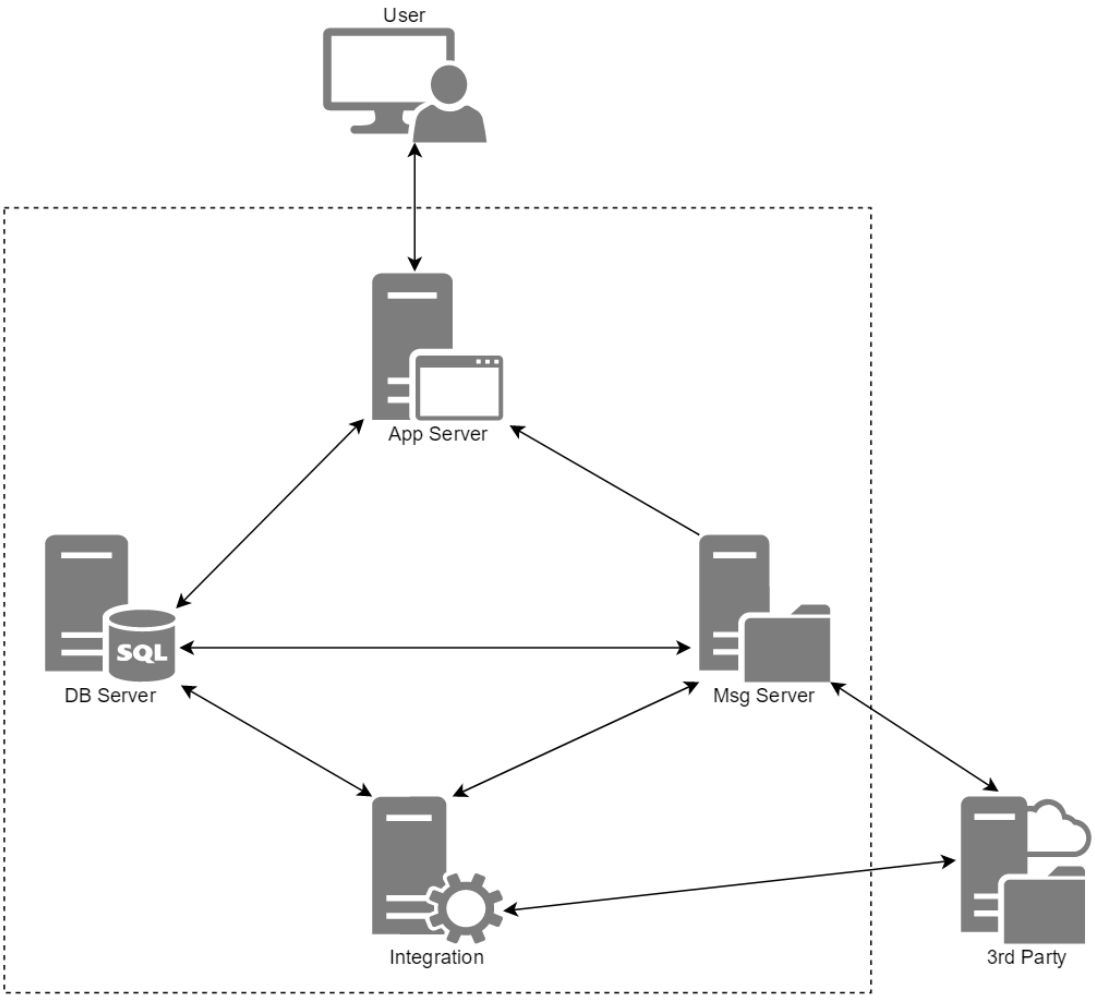

FIGURE 5. Basic environment.

4.1.2 Servers

The entire environment consists of mainly Windows servers except the Ubuntu servers used for ELK Stack. All servers are virtualized on a VMware and are managed by a vCenter client. Both products are from VMware Inc. All the configurations and tests run in this thesis were done on Windows server 2008 R2 and 2012. ELK-Stack was installed on Ubuntu 14.04.4 LTS.

4.1.3 Acute software

Acute is a browser-based software used by medical professionals to document patient data. The most simple form of usage is storing information regarding patient sessions to the patient journal. The patient journal can be imagined as a patient unique folder where all data for that patient is stored. Acute also has various complex functions and integrations. Integrations are features that communicate with other software remotely. Such remote features are ePrescription, eArchive and electronic examination referrals.

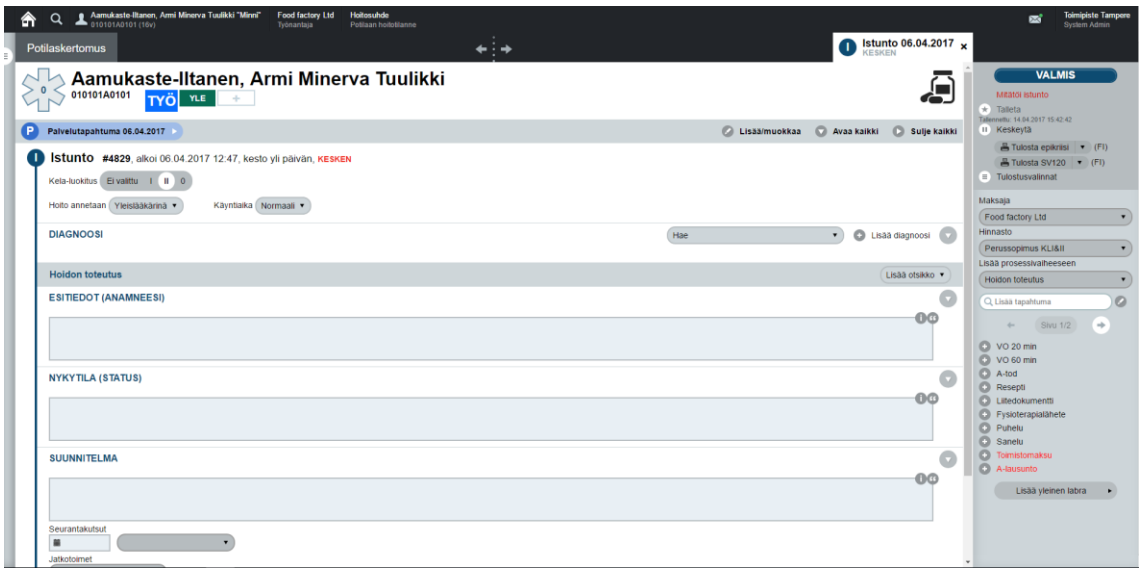

FIGURE 6. Acute

4.1.4 Logs

Monitored logs in the scope of the thesis are event, firewall, IIS and custom integration logs. Firewall and IIS logs are mainly collected from DMZ server. DMZ servers are

servers to which users can connect from outside the trusted network. For example, application servers where users connect to use their Acute instance are DMZ servers. On these servers data like blocked IPs, HTTP post and get method URLs and delay can be obtained.

Integration and message server logs are usually line-based custom logs. These logs contain information about status and health of various integration transactions. Laboratory, ePrescription and eArchive transactions are among these logged events.

Database servers have their own logging and alert system embedded in the SQL engine. Event logs could be followed for scheduled task failures. An example of a scheduled task is a data dump extracted from the database and exported to a SFTP server for clients to download.

5 RESEARCH

5.1 Interviews

5.1.1 The interview

There is a lot of information that could help the company serve their customers faster and more efficiently. Imagine a situation where specialists would already be fixing a problem before receiving a complaint of the problem from the users. In minor cases, fixing the problem without the users even noticing there was one.

Interviews were held to find out what trends and problematic situations should be followed and informed of. In addition, to narrow down the scope of wanted alerts and information that would be delivered to the specialists and customer support.

Various personnel were handpicked from different departments of the company. Personnel from customer service to system specialists were chosen. Experience of the interviewees varied from a year to seven years in the company. Chosen personnel were selected based on their potential in helping to solve what data is important from that vast data flow. Thus, people with a known good insight to the different aspects in customers and the system itself were chosen to potentially improve the business of the company.

The questions represented to the interviewees are as follows.

1. What information would you consider good or vital to receive in real time from the system?
2. What information about the system status would be helpful information?
3. What are subjects of which clients often give feedback?
4. Is there information you would like to have more easily obtainable?
5. Would it be a good idea to send automatically information to customers?

5.2 Results

5.2.1 First question

What information would you consider good or vital to receive in real time from the system?

There was a slight variance in the answers to the first question depending on the occupation. Customer service was more eager to receive real-time info about integrations such as ePrescription and electronic examination referrals. System administrators wanted more info about server side problems like failed scheduled tasks and blocked IPs or failed transfers on the SFTP server. Everyone agreed on the fact that everything that could lead to a nonconformity, should be informed of as soon as possible. Thus possibly avoiding it affecting the clients.

5.2.2 Second question

What information about the system status would be helpful information?

Customer service was straightforward with the second question saying that information about the different features, if they are working or not, would be good. The sysadmins gave more in detail answers, but also agreeing that knowing if it works or not is necessary.

The sysadmins were hoping for alerts that could help them anticipate upcoming defects before hearing it from the customers. An example was given from the electronic laboratory referrals. Even though the integration itself would be up and running, single transactions could fail resulting in some clients not getting lab results. These kind of situations could be fixed before the customers notice the problem. It was also noted that every log should have its own alert thresholds.

5.2.3 Third question

What are subjects of which clients often give feedback?

The interviewees were asked what are the top subjects of feedback coming from the customers. The most common were integration related troubles. Lab referrals not transferring to the laboratory or vice versa and ePresciptions not transferring to the prescription center. Second most common subject was general delay. General delay meaning sluggishness of the Acute software for the users. The third most common subject brought up was scheduled tasks. Under scheduled tasks was combined automated task failures like report creation.

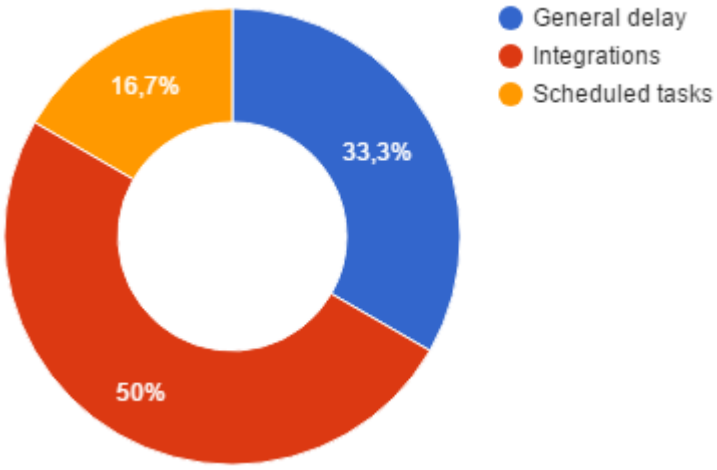

FIGURE 7. Top feedback subjects

5.2.4 Fourth question

Is there information you would like to have more easily obtainable?

The sysadmins were unanimous about what information they would want more obtainable concerning the alerts. They want more details and instructions to the alerts. The general situation now with the form of alerts is source and cause of the alert. The future desired situation would be if the alerts would tell the user what, where, why and instructions on how to solve it. This would save time and effort. The customer support thought information about the overall health and latency of the system would be beneficial.

5.2.5 Fifth question

Would it be a good idea to send automatically information to customers?

The general opinion was a definite no. Clients could be “over informed” and information sent to clients should always be checked before sending. Sysadmins pointed out that in some cases automated alerts could be sent outside the company. In situations, where for example a third party is involved in maintaining a feature. In these cases, the faster the other party is informed the better.

6 SOLUTION

6.1 Services

6.1.1 ELK Stack

The main services in the environment for log data collection is formed from 4 services. Nxlog, Logstach, Elasticsearch and Kibana. The three mentioned last form the so-called ELK Stack. Nxlog is not considered part of the ELK Stack and can be replaced with any other log forwarder. ELK abbreviation formed from each service first letter. All the services are open source.

6.1.2 NxLog

Nxlog is the service that starts the whole chain of actions. Nxlog follows configured archives and files for changes and forwards them to Logstash. Log data can be shipped from different sources. Eventlog, IIS log, Firewall log or as in figure 8 a custom laboratory transaction log can be forwarded.

The configuration can consist of extension, input, output and processor modules. In figure 8 all except processor modules are used. In the extension modules, you can import modules that you can use inside other modules. In figure 8 xm_json module is imported and used inside the input module to convert a line-based log to JSON format. Inside the input module, the monitored log is configured and modified. Output module is used to configure forwarding of the log data. In the route node input and output modules are infused. Finally NxLog forwards the log data to Logstash.

```
22  <Input lablog>
23      Module im_file
24      # Default lablog log location
25      File '[redacted]MSMQ*.log'
26      SavePos TRUE
27      InputType LineBased
28      Exec $Message = $raw_event;
29      # drop empty messages
30      Exec if $Message == '' drop();
31      # drop comment lines
32      Exec if $Message =~ /^#/ drop();
33      # add tag from which folder log is from
34      Exec $folderId = '[redacted]';
35      # format the message as json
36      Exec to_json();
37  </Input>
38
39  <Output out_iis>
40      Module om_tcp
41      Host [redacted]
42      Port 4006
43  </Output>
44
45  <Output out_debug>
46      Module om_file
47      File '[redacted]nxlog_debug.log'
48  </Output>
49
50  <Route 1>
51      Path lablog => out_iis
52  </Route>
53
54  <Route 2>
55      Path lablog, => out_debug
56  </Route>
```

FIGURE 8. NxLog configuration

6.1.3 Logstash

Logstash is the service in the middle. Logstash works as the receiver, normalizer, filter and sender. Normalizing means to bring or return something to a normal or standard condition or state. Filtering in this scenario means modification of fields, if they meet certain criteria.

At receiving log data Logstash tags the data per input port. In figure 9 for all entries coming to TCP port 4006, a field type with a value “lablog” is given. At this point, it is also a good idea to tell the system what codec is used, so decoding can be done before it enters the input module.

```
  # lab logs
  tcp {
    type => "lablog"
    port => 4006
    codec => "json"
  }

}
10-input.conf (END)
```

FIGURE 9. Logstash input configuration

After the initial tagging, the data is filtered if the type matches a filter. In this case all entries that entered using the port 4006 will match type with a value of “lablog” as shown in figure 10. Fields can be removed, modified or added before sending them to Elasticsearch.

```
filter {
        if [type] == "lablog" {
                mutate  {
                        remove_field => [ "EventReceivedTime" ]
                }
                if [host] == "[redacted]" {
                        mutate {
                                add_field => { "Hostname" => "rndintg01" }
                        }
                }
                if [Message] =~ /.Commit.ACK.hasn't.been./ {
                        mutate {
                                add_field => { "Error" => "1" }
                        }
                }
                if [Message] =~ /.Commit.ACK.has.been/ {
                        mutate {
                                add_field => { "Error" => "0"}
                        }
                }
        }


}
20-lablog.conf (END)
```

FIGURE 10. Logstash filter configuration

6.1.4 Elasticsearch

After Logstash has imported and filtered the entries, all the data is forwarded to Elasticsearch. It is the heart of the ELK Stack. Everything is either eventually forwarded to Elasticsearch or read from it. It is a search engine based on Lucene and is developed in

Java. Elasticsearch creates indices which contain parsed and normalized log data in various predetermined fields. Fields can be for example timestamp, error code, message, source and event id. Fields consists of a header and the data itself. The data types can vary from core datatypes like integers, strings or Booleans to more specialized datatypes like geo-points, JSON objects and arrays.

6.2 Situational awareness

6.2.1 Kibana

Kibana is used to graphically present the Elasticsearch indecies. Kibana can also be used to make queries to the indices with Lucene query language. From saved search queries, visual presentations can be formed and from which dashboards can be created. A dashboard is a group of visualizations.

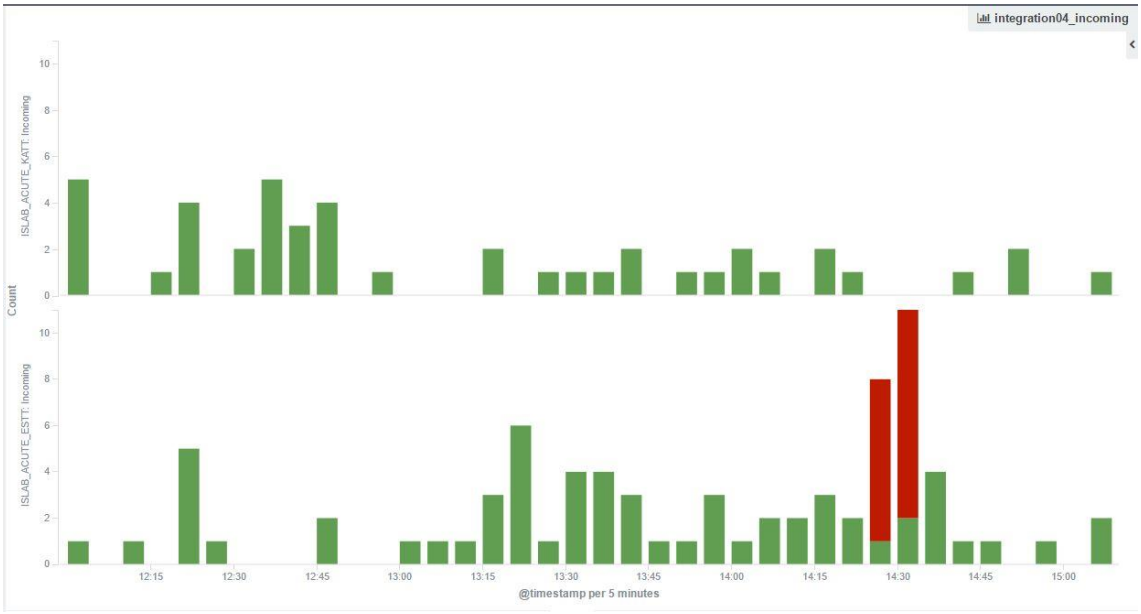

FIGURE 11. Visualization of laboratory transactions count and status

6.2.2 Elastalert

Elastalert is a tool created by Yelp to follow Elasticsearch indices and patterns in them. If a preconfigured pattern is detected a configured action will occur. Such actions can be email, Telegram and Jira alerts. As Yelp says it "ElastAlert is a simple framework for

alerting on anomalies, spikes, or other patterns of interest from data in Elasticsearch". (Running ElastAlert for the First Time, 2016)

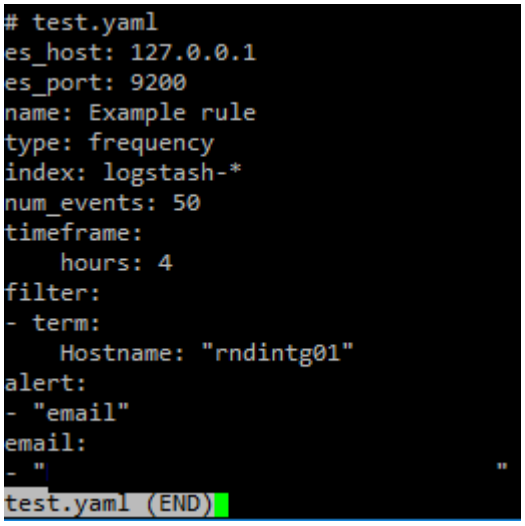

FIGURE 12. Elastalert example alert configuration

7 CONCLUSION

The question was to solve what information would be critical to be informed of, how and when to be informed, from all the logged data. In addition, a major problem in the former system was that it required an employee to actively monitor it. The original rules and configuration were setup by Hulkkonen and they required some modification. In the scope of the thesis, only operating system and application logs were included.

The necessity and benefits for the thesis was easy to back up by various articles and publications but mainly by personal and colleague experiences. Research was done by interviewing company personnel, which also supported the importance of this implementation. Perhaps the biggest agenda being financial savings in man-hours and avoiding sanctions from clients.

The challenge in this thesis was to include the important data and exclude the unimportant data from the millions of lines of log data. Especially the custom logs proved to be challenging with varying fields and message bodies because all the indexed data had to be in a normalized form to be suitable for comparison. Forming the alerts themselves was not as big of a challenge as I expected.

As the result of this thesis, the previous implemented system was modified to fulfill the new needs for the situational awareness system. The newly configured system was tested on incoming laboratory transactions statuses and REST service IIS logs. Observed subjects were specific error messages, long delays in https post and get methods and http errors. In these cases, the system seemed to catch wanted anomalies efficiently.

The next step in developing the situational awareness system would be involving all the services and features from all the servers. Normalizing logs and alerts so customization would not be needed on different parts in the system. Including guide lines how to fix the situation if needed. Now that the company has started to use Jira, perhaps a Elastalert – Jira integration. In the far future, an interesting step forward would be a self-learning system to analyze new data and patterns. (Weixi Li, 2013)

SOURCES

Angulo, H. 2015. Three Ways That Automated Log Summaries Take the Grunt Work out of Log Analysis. Read 20.04.2017. https://www.loggly.com/blog/three-ways-automated-log-summaries-take-grunt-work-log-analysis/

Haynes, G. 2014. IaaS, PaaS, SaaS, & the Cloud 101. Read 09.05.2017. https://www.linkedin.com/pulse/20140907071547-305726885-iaas-pass-saas-the-cloud-101

Hulkkonen, T. 2016, Implementing situational awareness. Technology, communication and transport, Jyväskylä Univeristy of Applied Sciences, Master's thesis. Read 15.12.2016. https://www.theseus.fi/bitstream/handle/10024/113392/Hulkkonen_Timo.pdf?sequence=1

Kent, K. and Souppaya M. 2006. National Institute of Standards and Technology. Guide to Computer Security Log Management. Read 15.04.2017. http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-92.pdf

Li, W. 2013. Automatic Log Analysis using Machine Learning, Awesome Automatic Log Analysis version 2.0. Read 20.04.2017. https://pdfs.semanticscholar.org/0711/4afea22ccf212ca7f31a73904e67ab40a785.pdf

Raff, A. 2014. Automated Traffic Log Analysis: A Must Have for Advanced Threat Protection. Read 20.04.2017. http://www.securityweek.com/automated-traffic-log-analysis-must-have-advanced-threat-protection

Running ElastAlert for the First Time. ElastAlert - Easy & Flexible Alerting With Elasticsearch. Read 04.05.2017. http://elastalert.readthedocs.io/en/latest/running_elastalert.html