Czech Technical University in Prague

Faculty of Electrical Engineering

Telemetry Data Collection, Analysis and Representation

BACHELOR THESIS

Tomáš Zámostný

Supervisor: **Ing. Bešťák Robert Ph.D.**

summer **2021**

CTU CZECH TECHNICAL UNIVERSITY IN PRAGUE

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Zámostný Tomáš**	Personal ID number:	**466202**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Telecommunications Engineering**		
Study program:	**Electronics and Communications**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Telemetry Data Collection, Analysis and Representation

Bachelor’s thesis title in Czech:

Sběr telemetrických dat, jejich analýza a representace

Guidelines:

The aim of work is to qualify and select a solution for telemetry data collection, to implement a data collector agent for Debian based PBX servers, and to create a backend service for data storage and analysis. Further, to create a web-based analytics dashboard.

Bibliography / sources:

[1] B. Marr. Big Data: Using SMART Big Data, Analytics and Metrics To Make Better Decisions and Improve Performance. Wiley. 2015.
[2] Martin Sauter. From GSM to LTE-Advanced Pro and 5G: An Introduction to Mobile Networks and Mobile Broadband. Wiley. 2017.
[3] B. Baesens. Analytics in a Big Data World: The Essential Guide to Data Science and its Applications, Wiley. 2014.

Name and workplace of bachelor’s thesis supervisor:

Ing. Robert Bešťák, Ph.D., Department of Telecommunications Engineering, FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **16.09.2020** Deadline for bachelor thesis submission: **05.01.2021**

Assignment valid until: **19.02.2022**

Ing. Robert Bešťák, Ph.D.	Head of department’s signature	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature		Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Declaration of Authorship

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all mate- rial and results that are not original to this work.

Prague, date: Tomáš Zámostný

Prohlášení o autorství

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Praha, dne: Tomáš Zámostný

Acknowledgements

This project was made in a very short period of time between challenges and difficulties, none of this would have been possible without the help and support of a very special group of people.

"Thank you" to my family, who has supported me since day one in all my decisions and achievements.

Finally, I would like to thank my thesis supervisor Ing. Robert Bešťák, PhD. of the Department of Telecommunication Engineering at Czech Technical University in Prague for proposing this thesis and giving me the opportunity to make it happen.

Tomáš Zámostný

Czech Technical University in Prague

Abstract

This thesis aims to search, analyse, and visualise real-time data in telemetry and log data from various servers using open source products such as Elasticsearch, Logstash, and Kibana. The log and data located on several different processing machines can be challenging to handle as the volume of data increases with the scale of software systems, networks, and services.

This thesis discusses the concepts of centralised logging, telemetry, and existing software solutions for implementing such a system. It also allows you to search for problems across multiple servers by connecting their telemetry data over a specific period. These operations are valuable and essential in an environment with hundreds of containers producing terabytes of data per day.

This thesis discusses the benefits of collecting and analysing all telemetry data from the newly created system. The finished framework and additional results from the implementation process are also described. The final setup worked well and did not necessitate any improvements to our system.

Keywords: *Telemetry, Data analysis, Collect data, Agent, Elasticsearch, Logstash, Kibana, Real-time, ELK Stack, Metricbeat*

Abstrakt

Cílem této práce je vyhledávat, analyzovat a vizualizovat telemetrická data v reálném čase a protokolovat data z různých serverů za pomoci open-source nástrojů jako je Elasticsearch, Logstash a Kibana. Je komplikované sbírat data umístěná na mnoha různých serverech, vzhledem k množství dat, které se zvyšuje s velikostí sítě.

Práce pojednává o konceptech centralizované telemetrie, a hledá vhodnou variantu ze stávajících softwarových řešení, aby bylo možné hledat a analyzovat problémy na serverech za delší časové období. Takové řešení je přínosné a klíčové v prostředí, které se skládá z velkého množství serverů a generuje stovky terabajtů dat denně.

V dalších kapitolách se věnuji výhodám sběru a analýzy všech telemetrických dat z našeho vytvořeného systému. Konečné nastavení fungovalo správně a nevyžadovalo žádná další vylepšení našeho systému.

Klíčová slova: *Telemetrie, Analýza dat, Sběr dat, Agent, Elasticsearch, Logstash, Kibana, Real-time, ELK Stack, Metricbeat*

CONTENT

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ELK	Elasticsearch Logstash Kibana
PHP	Personal Home Page
HTTP	Hypertext Transfer Protocol
API	Application Program Interface
JSON	JavaScript Object Notation
GUI	Graphical User Interface
VM	Virtual Machine
UI	User Interface
ISO	International Organization for Standardisation
GPS	Global Positioning System
RAM	Random Access Memory
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
SQL	Structured Query Language
DAM	Digital Asset Management
CTI	Computer Telephony Integration
VOIP	Voice Over Internet Protocol
PBX	Private Branch Exchange
CSV	Comma-separated values
UTC	Coordinated Universal Time

1. INTRODUCTION

Telemetry data is data recorded and transmitted from a remote source and is later used for monitoring [1]. The type of data measured can come in many forms, such as HTTP request logs, CPU / Memory usage, etc. Once this data has been received, it can be used for many important functions. For instance: monitoring of performance and security of a system, examining the health of servers and ultimately improving the customer experience of the product or service which is being monitored.

There are several professional softwares capable of analysing telemetry data, such as Hadoop and ELK which means Elasticsearch Logstash Kibana. Each software has its own unique advantages and disadvantages, with specific functions more compatible with particular software. For example, Hadoop is more suitable for analysing big data (large volumes of data that cannot be treated with conventional methods).

This thesis aims to describe and implement a solution for telemetry data collection and analysis using the ELK stack. Namely, to implement a collector agent for Debian based PBX servers (Private Branch Exchange), used in private telephone networks, to create a back-end service for data storage and to create a web-based analytics dashboard. ELK is a collection of open-source tools that work together to search and analyse data from multiple servers or a single location.

1.1 Aim of Bachelor Thesis

This thesis aims to implement the collection and subsequent analysis of telemetry data from multiple PBX servers. This data is used for determining the state of a PBX server network. A PBX network is often crucial for the daily operations of companies and other institutions performing health checks. Preventing and minimising downtime is necessary for smooth operations and cost avoidance. A diagram of a PBX network is shown in Figure 1.

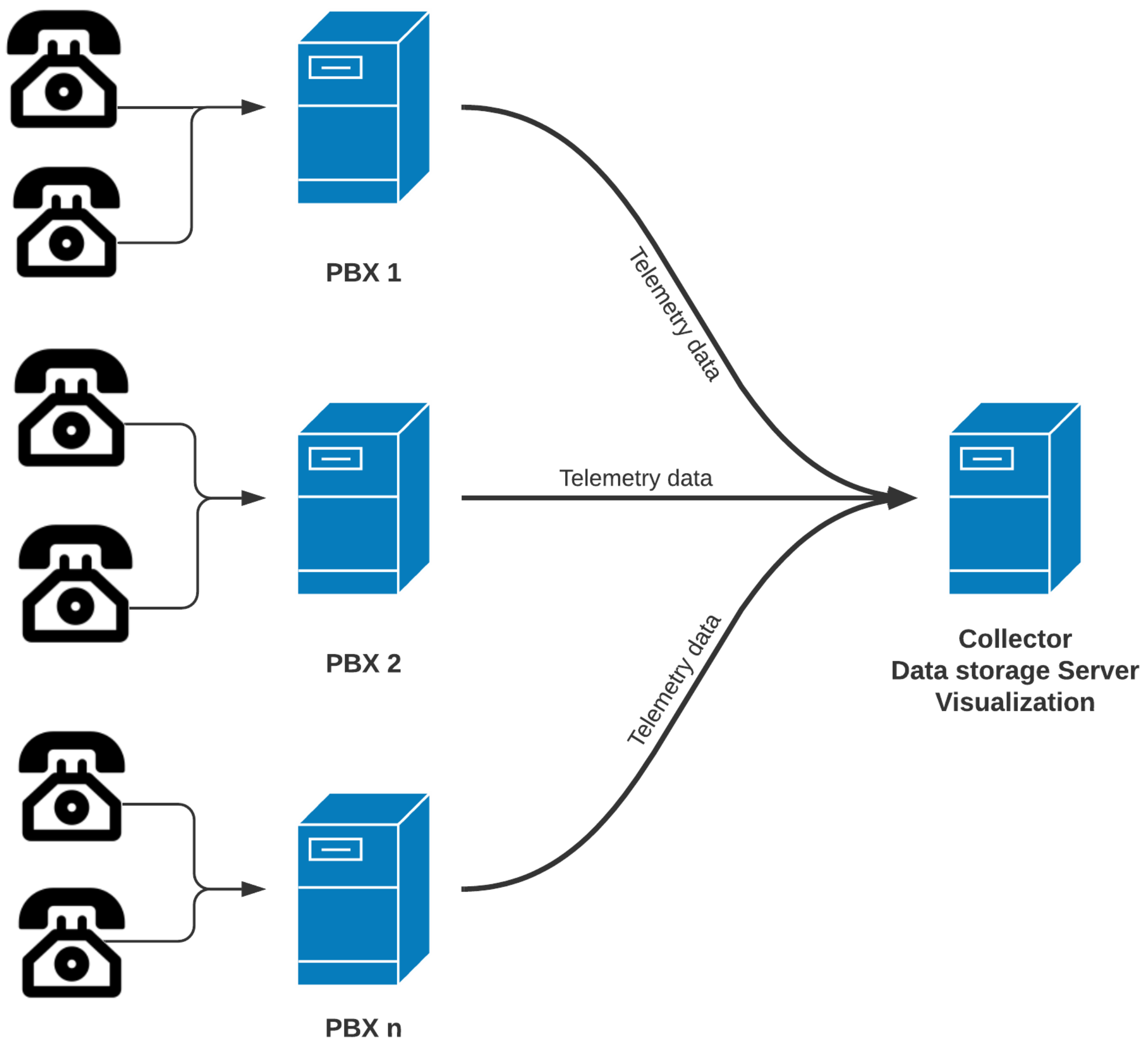


Figure 1 – Network structure

To achieve our goal, we use the ELK stack and Beats (Metricbeat and Filebeat). Figure 2 shows how ELK is used to centralise data management. Logstash is the component that receives logs transmitted by Beats from remote servers, pre-process them and pass them on directly to Elasticsearch. Elasticsearch serves as the database, search and analytics engine, and finally, we use Kibana, a user interface component that displays Elasticsearch data in graphs, dashboards, and charts. The whole stack is then run in a Docker container.

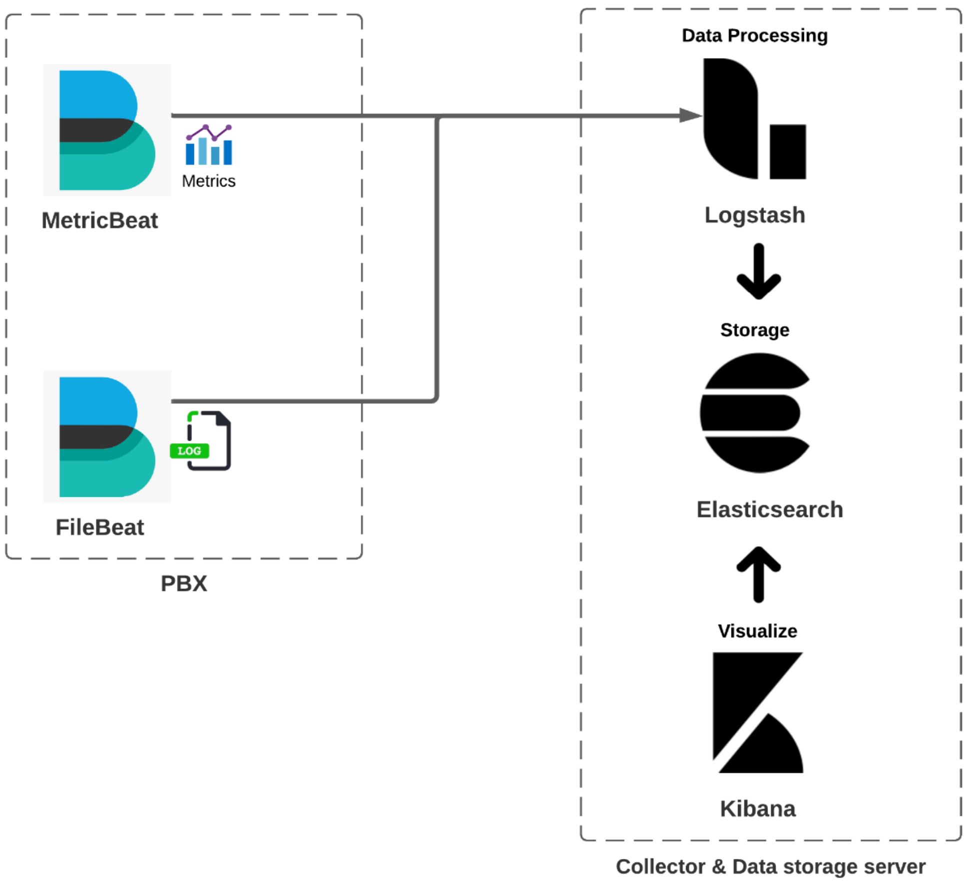


Figure 2 – System architecture

1.2 Structure

This document contains seven chapters and thirty-five subchapters.

- Chapter 1 - This chapter provides an introduction to the thesis and explains the Aim of the thesis along with illustrations.
- Chapter 2 - This chapter introduces ELK, which is the main software used in this thesis.
- Chapter 3 - This chapter explains the used environments, which helps the reader to understand more about ELK and its implementation in the deployed environment.
- Chapter 4 - This chapter has an explanation of the contents of the different datasets which are being used for testing centralised telemetry monitoring.
- Chapter 5 - This chapter covers an in-depth explanation of creating and implementing our system.
- Chapter 6 - This chapter consists of descriptions of visual output from Kibana.
- Chapter 7 - The thesis conclusion provides an overall review and suggests further development possibilities.

2. ELK STACK

The architecture of software services has evolved from monoliths to microservices in the last few years or decades. Microservices are easily scalable and can therefore be used very flexibly. This development opens up many options for adapting the services to the required load for operation in production.

Monitoring is an essential part of systems in production, especially finding errors in log files or analysation telemetry data. While searching the data files in a single-system service, in which there are only one or possibly several chronologically divided data files, is still relatively easy. Things are very different with distributed microservices. An incoming request is forwarded from one service to the next, so troubleshooting can be difficult and take up a lot of time.

We want to take a closer look at the ELK stack here, as it is becoming increasingly popular and the technologies can be used free of charge. ELK stands for the technologies on which the telemetry and logging framework is based: Elasticsearch, Logstash and Kibana. Logstash collects data files from various sources, filters them, changes them, and passes them on to several databases. One possibility is the Logstash, which receives data entries via TCP or UDP. In the ELK stack, the data entries from Logstash are passed on to Elasticsearch and stored. Elasticsearch is a document-based NoSQL database [2].

Elasticsearch is based on Apache Lucene and is highly efficient when searching for texts and is therefore perfect for telemetry or log file entries. Kibana is a graphical user interface that reads and displays the log entries from Elasticsearch. The aggregated telemetry and log data can be visualised graphically with diagrams and summarised in a dashboard [3].

The ELK stack can also be built up a little differently. Logstash can react to TCP or UPD calls and receive the data entries in this way and read them directly from a data file. In this setup, Logstash must be installed for each service and one or more files are defined as input (shows in Figure 3). Logstash monitors the specified data files, reacts to changes in the file and sends new entries to Elasticsearch [4].

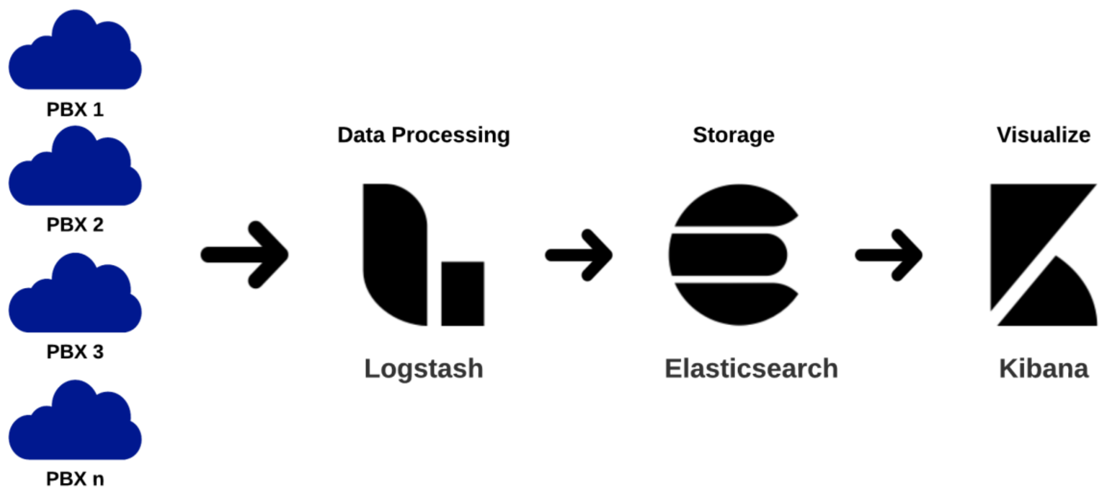


Figure 3 – ELK architecture

2.1 Elasticsearch

Elasticsearch is an open-source search engine based on Apache Lucene. It works with indices, which consist of JSON documents in NoSQL format. The search engine works very quickly and can be used to search in large amounts of data (big data) and support distributed architectures for high availability. Together with Kibana and Logstash, Elasticsearch forms the Elastic Stack [5].

Elasticsearch is a search engine released in 2010 and developed by Elastic (formerly Elasticsearch N.V). It is based on Apache Lucene and enables full-text searches in various types of structured and unstructured data such as text data, numerical data or geodata. Elasticsearch works with indices, consisting of JSON documents in NoSQL format, and provides a REST API for indexing and searching through the data. Due to the high search speed, Elasticsearch can be used to search through large amounts of data, for example, in big data and business intelligence applications.

The search engine supports distributed computer architectures and offers high availability. Together with Kibana and Logstash, Elasticsearch forms the so-called Elastic Stack, also called ELK-Stack. It is a collection of open-source software that can be used for analysing and visualising data. Elasticsearch is one of the most widely used search software and is used for searching websites, company data or log files [3].

The core software of Elasticsearch is subject to the Apache License 2.0. In addition, the company Elastic offers commercial products and fee-based services related to Elasticsearch. Elasticsearch is programmed in Java, and it is platform-independent.

2.1.1 How Elasticsearch works

The focus of the search with Elasticsearch is the data structure of the inverted index. The index is created before the search by indexing the raw data and storing it in a database. The data that is to be searched is stored there. The index is divided into types and documents. For the search engine, types behave similarly to tables in a database. An index can be of many types. The JSON documents with their properties are located below the types. They are the smallest unit of the index and contain the actual data for the search. JSON documents are made up of pairs of keys and values. Elasticsearch uses a REST API both for indexing the raw data and for executing the search queries.

2.1.2 Applications of Elasticsearch

Elasticsearch can be used as a search engine for many different applications. Typical applications are:

- Search on websites
- the search in application data
- the search in business data
- the search in log files
- the search in geospatial data as well
- the search in security and monitoring data

Elasticsearch uses a data structure called an inverted index designed to allow quick full-text searches. An inverted index lists every unique word in any document and identifies all of the documents each word occurs in. The architecture of the system is structured based on the following concepts [6],[7]:

- **Clusters**: It is a collection of one or more nodes(servers) which are running the ELK software components. All these nodes will hold the entire data and provides federated indexing. The logs are stored in different nodes and search capability is across all nodes and makes simple to output the requested document. A cluster is identified by a unique name which shouldn't be matched with other cluster names.
- **Node**: It is a single server that is with Elasticsearch package installed into it. This server is a part of a cluster with a unique identifier name inside the cluster. This node also accesses the user to store, search and analyse the documents.
- **Index**: It is a collection of documents (logs) that have somewhat similar characteristics. In a single cluster, many indexes can be declared and be defined.

- **Document**: It is a basic unit in Node which is information that can be indexed. Many documents can be stored within an index/type. Although a document physically resides in an index, a document actually must be indexed/assigned to a type inside an index.
- **Shards and Replicas:** Elasticsearch has the best facility to subdivide the index into multiple pieces into shards. When there is large data, this facility acts more efficiently. When an index is created, simply number of shards can be defined and each shard is in itself a fully-functional and independent.

2.1.3 Advantages of Elasticsearch

The following points explain the various advantages of Elasticsearch:

1. Elasticsearch is fast. Since Elasticsearch is based on Lucene, it can play its trump cards, especially with full-text searches. Elasticsearch is also a near-real-time search engine: the time between indexing a document and finding the document through the search is concise - usually only a second. This makes Elasticsearch particularly suitable for use cases in which time plays a role, such as security analytics and infrastructure monitoring.
2. Elasticsearch is naturally distributed. The documents stored in Elasticsearch are distributed to various containers, the so-called shards, which are duplicated so that redundant copies are available in the event of a hardware failure. The distributed nature of Elasticsearch makes it possible to scale Elasticsearch to hundreds (or even thousands) of servers and process data petabyte by petabyte.
3. Elasticsearch has a variety of features. In addition to its speed, scalability and resilience, Elasticsearch has several powerful built-in functions that make it even more efficient to store and search for data. This includes, for example, data rollups and index lifecycle management.
4. The Elastic Stack makes it easy to ingest, visualise, and report on data. Thanks to the Beats and Logstash integrations, data can be quickly processed before indexing in Elasticsearch. Furthermore, Kibana offers real-time visualisation of Elasticsearch data and functions for quick access to APM data, log data and infrastructure metrics.

2.1.4 Disadvantages of Elasticsearch

The following points explain the various disadvantages of Elasticsearch:

1. Language constraint: Elasticsearch does not have multi-language support in terms of handling request and response data.
2. Structure specific: To run the queries correctly, you need to take care of a hierarchy of indices, IDs, and types.

2.2 Beats

Data such as CPU utilisation or the network status of the server are also interesting for the analysis. Elastic Stack offers so-called beats for this. Beats are lightweight platforms that act as data shippers and transmit data files from various sources to Logstash or Elasticsearch. Many beats are supported by Elastic Stack, such as Metricbeat, Filebeat, Packetbeat, Heartbeat, Win-log beat, Libbeat (in Figure 4). Metricbeat collects the metrics about the system and services. With the help of Metricbeat, you can quickly get an overview of CPU usage at a system level, memory and file systems, the IO of hard drives, the network, etc. Metric Beat is part of Elastic Stack and is easy to use with Logstash, Elasticsearch and Kibana integrable. With the default setting, Metricbeat sends the data to Elasticsearch in localhost: 9200, but it is easy to configure via the metricbeat.yml file which metrics we need and where the metrics should be sent. Packetbeat collects the network data and database analysis. With Filebeat, you can send log files to a central server. Libbeat is an open-source framework that makes it easier to create your beat [8].

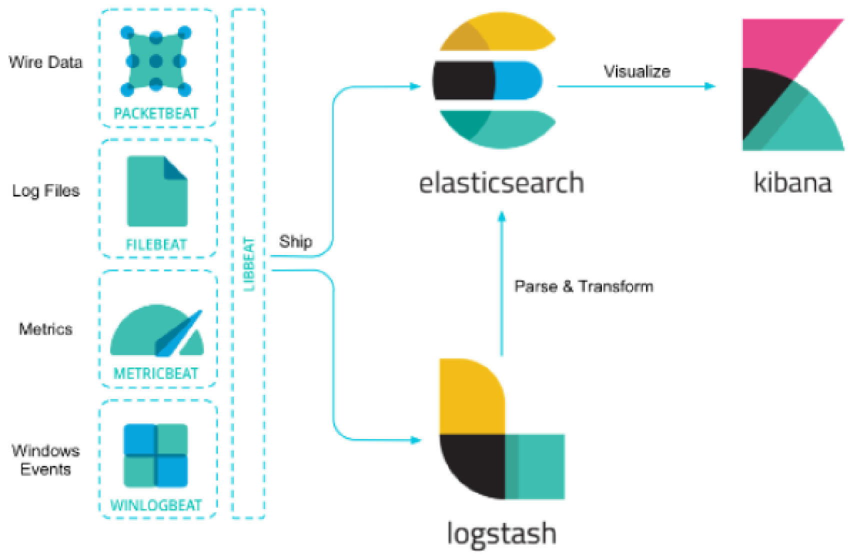


Figure 4 - Structure Beats and Elasticsearch [8]

2.2.1 Metricbeat agent

Metricbeat collects data about system resources and sends it to the monitoring cluster. The advantage of using Metricbeat instead of the internal collection is that the monitoring agent continues to function even if the Metricbeat instance is terminated. Because you will be monitoring Metricbeat, you will need to run two instances of Metricbeat: one that collects metrics from the system and services running on the Web server and another that collects metrics only from Metricbeat. You can send monitoring data to a dedicated monitoring cluster using a separate instance as a monitoring agent. The monitoring agent remains active even if the leading agent fails [8].

2.3 Logstash

Logstash [9] processes and normalises telemetry and log files. The application draws its information from various data sources that users define as input modules. Sources can be, for example, data streams from Syslog or log files. In a second step, filter plug-ins process the data according to user specifications - the phase can also be omitted, the material then moves on unprocessed. Finally, the output modules output the results, everything goes to the Elasticsearch service in the test environment. Figure 5 shows how the components interact.

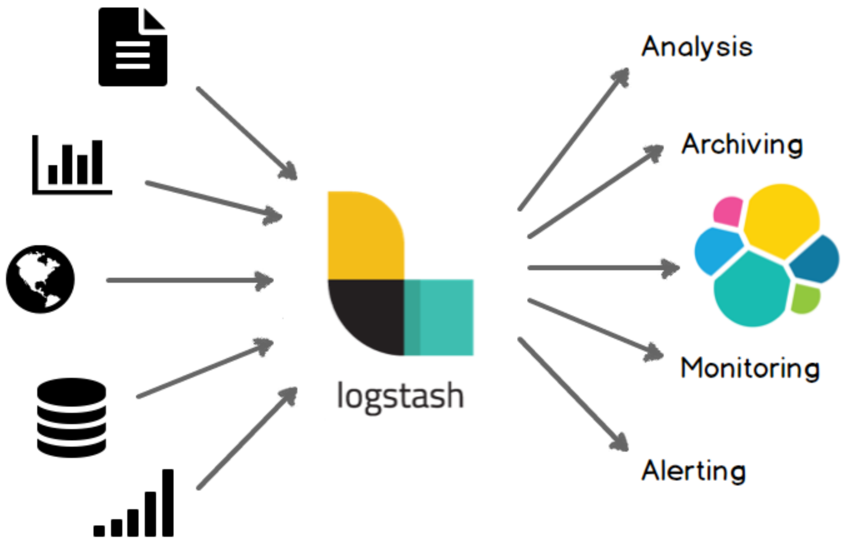


Figure 5 – Basic functioning of Logstash [9]

Logstash is free and, like Elasticsearch, comes under the Apache license. The sources, Debian and RPM packages, and information on the online repository can be found on the project page.

Logstash has a pluggable platform that supports over 200 plug-ins. This function facilitates the mixing, matching, and arranging of various inputs, as well as the streaming of outputs to operate in a pipeline. With its persistent queue, Logstash guarantees at least once delivery for in flight events even in the case of node failure.

Logstash can efficiently minimise obtained data by removing only the useful information. Logstash filters parse each occurrence, identifying named fields to create structure, and transforming them to converge on a standard format for more efficient analysis and business value. When it comes to data output, Logstash supports a wide variety of destination types. All events are typically sent to Elasticsearch, but Logstash can also be used separately to save data to a CSV file, a SQL database, or a data analytics algorithm [10].

2.3.1 Logstash with ELK

Indexing data in Elasticsearch can be done either by manually sending an HTTP request for indexing with the data or using some data senders like the Elastic Beats family. If the data comes from several different sources without a consistent data structure, a separate filter pipeline such as Logstash may be required to normalise the data before indexing. Logstash is an event processing pipeline that consists of three stages. The input stage is where the data enters the pipeline, and an event is created. Possible inputs are via a UDP port or from a Beats data sender. There is no rule in which format the data has to be available. The event is built from fields that contain the data. From the input stage, the event is passed on to the filter stage. A proper filter is the so-called Grok filter. It parses plain text from a field and recognises patterns. This way, additional fields can be created and filled with the parsed data, and an unstructured text from a field can be brought into a uniformly structured and queryable form [2].

Additionally, the fields can be manipulated by the mutate filter. Several filters can be applied to an event, one after the other. The final stage after the filter stage is the. Here one or more outputs can be defined, where and in which format the data should be sent. For example, the Elasticsearch output uses the convenient bulk index request to send a series of events at once [4].

2.4 Kibana

The information that is now stored in the Elasticsearch database is visualised with the data visualisation tool Kibana. Kibana is a web interface that was specially developed for

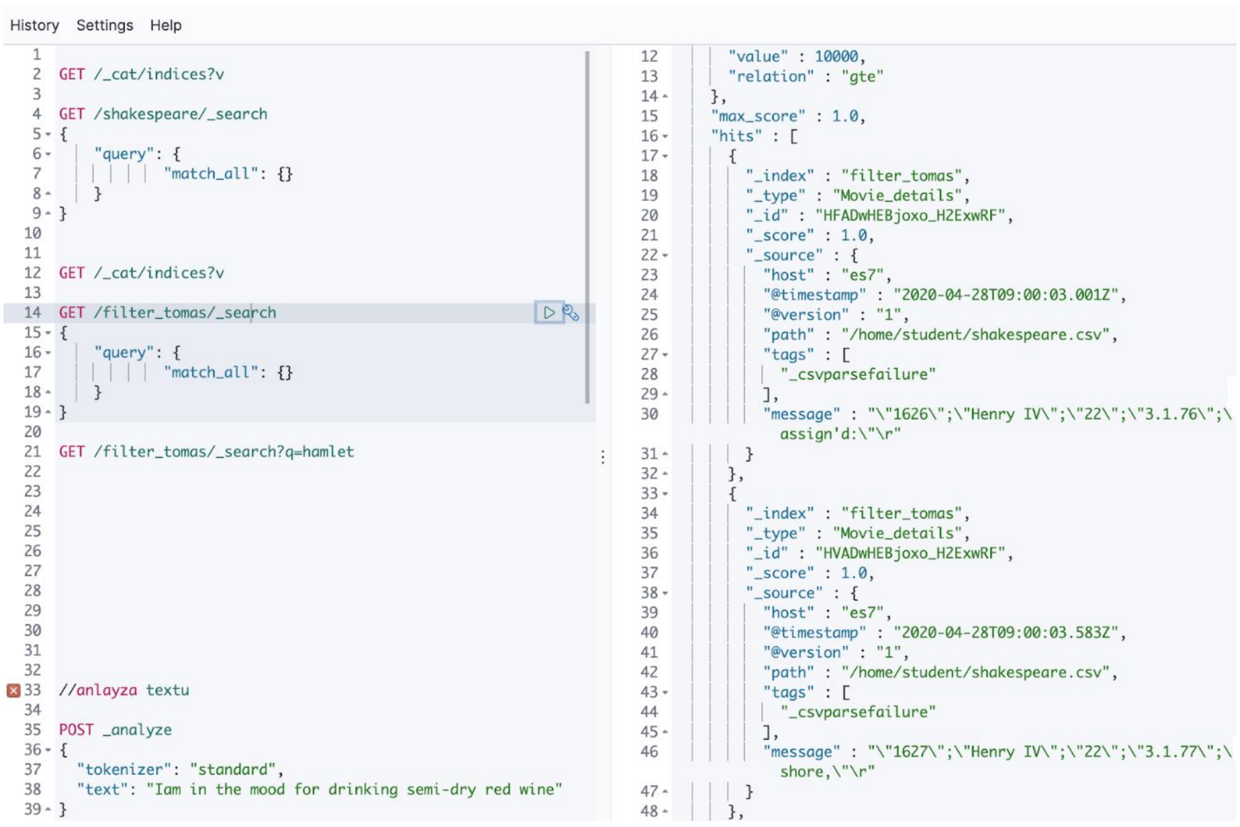
Elasticsearch. It also comes from Elastic and forms the third component of the ELK Stack. The graphical user interface interacts with Elasticsearch via REST and offers the user the option of visualising this data Figure 15 Kibana overview and filtering it as desired. With Kibana, the data can be analysed, evaluated and visualised. Kibana offers the possibility to create different types of visualisations. This includes histograms, graphs, pie charts, donut charts, and more. In order to summarise the various visualisations and adapt them to your own search, Kibana offers so-called dashboards (Figure 4 Kibana overview). A dashboard consists of any number of visualisations that can be freely arranged next to one another and one below the other. You can also interact with Kibana via a REST API. The user can call up the data directly from a visualisation via the API. Certain processes can be automated connection with Kibana can be established by external software products.

Kibana's interface can be accessed using: *http://localhost:5601/*. The default port on which Kibana is available is 5601. The main view of Kibana is divided into 3 main components – Discover, Dashboards and other managements. Discover allows to interactively browse and analyse pure data entries. Elasticsearch documents are categorised based on Index patterns defined in Management section. They can be easily searched and filtered by time or by document properties using Apache Lucene query syntax. One can also see the number of documents that match the search query or get field value statistics.

Kibana uses the data from Elasticsearch to represent the data to the user in the form of bar graphs, pie charts, heat maps etc. Visualise section provides possibility to visualise data. When we have the visualisations ready, all of them can be placed on one board – the Dashboard. Observing different sections together gives you a clear overall idea about what exactly is happening. Information can be shown in form of tables, charts, maps, histograms, and many others. Large volume can be easily shown in form of pie charts, bar charts, line, or scatter plot [11].

2.4.1 Kibana DEV Tool

Kibana is a front-end GUI for displaying and analysing data in Elasticsearch. It was created in JavaScript and runs in a normal web browser. Located on the Dev Tools page the development tools that are used to interact with Elasticsearch can. Kibana Dev Tool has a very simple user interface and is in a request and divided an answer window (see figure below). Features like syntax highlighting, auto-completion and automatic indentation make it easier to process search queries in the request window considerably. In the answer window, the respective result of the search query is shown as a JSON document presented. The folding and unfolding of document parts are also here possible [11].



```

GET /_cat/indices?v

GET /shakespeare/_search
{
    "query": {
            "match_all": {}
    }
}


GET /_cat/indices?v

GET /filter_tomas/_search
{
    "query": {
            "match_all": {}
    }
}

GET /filter_tomas/_search?q=hamlet











//anlayza textu

POST _analyze
{
  "tokenizer": "standard",
  "text": "Iam in the mood for drinking semi-dry red wine"
}
```

```
      "value" : 10000,
      "relation" : "gte"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "filter_tomas",
        "_type" : "Movie_details",
        "_id" : "HFADwHEBjoxo_H2ExwRF",
        "_score" : 1.0,
        "_source" : {
          "host" : "es7",
          "@timestamp" : "2020-04-28T09:00:03.001Z",
          "@version" : "1",
          "path" : "/home/student/shakespeare.csv",
          "tags" : [
            "_csvparsefailure"
          ],
          "message" : "\"1626\";\"Henry IV\";\"22\";\"3.1.76\";\
            assign'd:\"\r"
        }
      },
      {
        "_index" : "filter_tomas",
        "_type" : "Movie_details",
        "_id" : "HVADwHEBjoxo_H2ExwRF",
        "_score" : 1.0,
        "_source" : {
          "host" : "es7",
          "@timestamp" : "2020-04-28T09:00:03.583Z",
          "@version" : "1",
          "path" : "/home/student/shakespeare.csv",
          "tags" : [
            "_csvparsefailure"
          ],
          "message" : "\"1627\";\"Henry IV\";\"22\";\"3.1.77\";\
            shore,\"\r"
        }
      },
```

Figure 6 – Screenshot of Kibana Dev-tool

Figure 6 on the left side shows the beginning of an approximately 100-line long code with a test request. In the answer window, the right of it is all that match the request Elasticsearch documents clearly displayed. Because of this easy-to-use console uses Kibana Dev-Tools to code them to test programmed Java applications or to trace errors that occur and to be able to fix them.

2.5 Why ELK Stack

Each server in a network with several servers running different processes generates a different set of log messages. It is difficult to decipher a large number of telemetry and log messages and separate telemetry or log files scattered across each server. The Elastic Stack is a collection of open-source tools that work together to search and analyse log messages from multiple servers from a single location. Because the Elastic Stack's components are designed to be fully scalable, the monitored infrastructure can expand without the Elastic Stack becoming a "bottleneck."

This is why the Elastic Stack has become so well-known in the IT sector and industry. The Elastic Stack is used by companies such as Cisco, Docker, Microsoft, and eBay [7]. As a result, a sizable community is dedicated to Elastic Stack's continued development and maintenance. The popularity of open-source software like this one is growing every year.

The Elastic website has comprehensive documentation of the Elastic Stack tools. In addition to platforms like Stack Overflow, which are a valuable source of help and inspiration, there is an Elastic user forum that helps solve problems. Various blogs exist on the Internet that offer advice on how to prevent administrators from falling into traps.

Other systems exist in addition to the Elastic Stack. The commercial software Splunk, which is used (for example) by the ATLAS experiment, is probably the most common. Splunk only offers a free licence with a data volume limit of 500 MB. When using Splunk, the costs rise as the volume of data grows. The Elastic Stack basic licence comes with most of the necessary tools and functionalities, as well as an unlimited data volume. However, when purchasing extended licences, support and special plug-ins such as warning functions and security packages are available. On top of that, other third-party open-source tools can be used to set up the tool for the warning function, for example [3].

2.6 Finding the best solution

One of the most critical aspects of the choice is that both systems/tools are free and open source. Kibana is easy to use, and there is a free trial version available to try it out. Since Kibana is used to visualise data, it requires the use of Elasticsearch as the telemetry and log database, which requires the log collecting system's ability to store data in Elasticsearch. A pure Rsyslog based solution would not work in this thesis, though Rsyslog could be used to send data to Logstash. An agent to collect several types of telemetry data, mainly CPU and Memory usage, would be implemented Metricbeat. For collecting logs, Filebeat would be a better choice because it can parse log data and detect multiline log messages. Filebeat, on the other hand, would be a better choice for logs because it is capable of parsing log data and detecting multiline log messages. In this case, Filebeat, Metricbeat and Logstash are used because they are very similar. They can both send data to Elasticsearch, and Kibana can visualise any data stored in the Elasticsearch database. However, this organisation has some experience with Filebeat and Logstash. These software solutions have been used in the past. In the long run, choosing something already familiar will make maintenance and software updates easier. Although either Filebeat or Logstash could be removed from the centralised logging chain, both would be used.

Because multiple instances of the case can run on different servers, using Filebeat to collect log data instead of Logstash can save memory and CPU resources. Following that a single Logstash instance is used to parse and process the log data, process the log, which is the critical step for this system, given the challenging nature of the process. Filebeat alone does not provide adequate data processing capabilities, making sense to consolidate log processing in one place to simplify configuration changes [12].

3. USED ENVIRONMENTS

As part of my work, I worked with various environments, which I describe in detail in this chapter.

3.1 Debian

There are many Linux distributions such as Ubuntu, Red Hat, Deepin, Fedora, Black Arch, Mageia. In this thesis, it has been chosen the distribution of the Debian operating system [13].

3.1.1 Desktop and Server

Debian is a popular open-source software and operating system that can be used on both a desktop and a server. The open-source freeware from the Debian project includes desktops such as KDE, GNOME, Mate and XFCE. LXDE is recommended for older computers with limited RAM because it only takes a small amount of Memory. Older versions of Debian include Iceweasel as a browser, while Debian 8 and later include Firefox as an ESR edition. Software such as LibreOffice, Wireshark, machine tools, and multimedia tools are also available and installed using the terminal command line or the Synaptic graphic package manager.

3.1.2 Stable version

Above everything, Debian has a positive reputation for server use since it must run consistently for a long time without the need to load new versions continually. The stable version, which is released every two years and receives security updates for five years, ensures this. Old stable is the previous version of the new stable version, and it can be used until the end of support since Debian can continue to provide security updates until then. There are also experimental, unstable, and testing versions that are becoming stable version.

3.1.3 Manage packages

Debian uses the Advanced Packaging Tool (APT) as a package manager for installing software. If the software was not installed during the installation, it could now be installed from the command line. When you instal software, the package manager checks to see if all of the required packages are present and, if not, install them as well. Furthermore, the APT updates and removes packages and kernels.

To update Debian packages only a few command lines are needed. To begin, update the package details with "sudo apt-get update." Only managers, who must enter their password after the instruction, are permitted to perform an update. "sudo apt-get dist-upgrade" is used to download new package versions and update the packages if the package information is up to date. After that, use "sudo apt-get autoremove" to uninstall any packages that are no longer required. This also removes old kernel versions, freeing up a significant amount of Memory. If you do not do this, the hard disc partition will quickly fill up, and the file system will be unable to accept any more files, despite the available storage space [13].

3.2 CRON

This is a feature in the operating system. In the different variants of Windows, defined as Programmable Tasks, you can define the frequency and batch file or program to run through a graphical interface. In the Linux systems, it varies a bit depending on the distribution, but basically, you configure a particular text file, where the frequencies and program, command or shell script to run. In either case, there is abundant documentation about it, and it is not a complex task [PT-Win] and [Cron]. Besides, the same script or batch file must be run in both cases, with tiny differences, but constantly invoking the same extension module described in the next section. Care should be taken to establish how to refresh rate, the lowest possible frequency of all tasks programmable. This means that if tasks are scheduled that must be executed once a week and tasks need to be run once a day, the latter involves a particular schedule. If this schedule is segmented by every 30 minutes, the task scheduled should have 30 minutes. If, on the other hand, this schedule use only exact hours, the scheduled task should have 1 hour [13].

3.3 Docker

Docker is mainly a software development platform and a kind of virtualisation technology that makes it easy for us to develop and deploy apps inside neatly packaged virtual containerised environments. Meaning apps run the same, no matter where they are or what machine they are running on. Docker containers can be deployed to just about any machine without any compatibility issues, so your software stays system diagnostic, making the software more straightforward to use, less work to develop, and easy to maintain and deploy. These containers running on your computer or server act like little microcomputers with concrete jobs, each with their operating system and its own isolated CPU processes, Memory, and Network resources.

Containers are similar to virtual machines (VMs), but they are far more specific and granular. They separate a single application and its dependencies from all external software libraries required to run the app, as well as the underlying operating system. The containerised applications all run on the same operating system, but they are divided into compartments.

Containerized applications use significantly less Memory than virtual machines, boot up and finish faster, and can be carried much more densely on their host hardware. Enterprise software must be able to respond quickly to changing needs. That means easy scaling to meet demand as well as easy updating to add new features as the project demands. Docker containers make it simple to instal new software versions with custom features.

Docker allows for portability, and Docker containers contain everything an application requires to run. They make it simple to move applications between environments. That application can be run in a Docker container on any system that has the Docker runtime installed, such as a developer's laptop or a public cloud. Docker and containers provide developers with more independence while also allowing them to develop applications that respond quickly to changing business conditions [14].

3.3.1 Docker-Image

The Docker image is built with the information from the Docker le. It's kind of a snapshot of a system. Images are built up internally in layers, whereby each layer can be an independent image. For example, the operating system is an image (base layer) that is loaded when another image is built. If another image is created that is based on the same base layer, it does not have to be reloaded. It is sufficient that reference is made to the base layer. This saves resources, especially when there are several images based on the same layers. In summary, the image is that element of Docker that is portable and can be distributed.

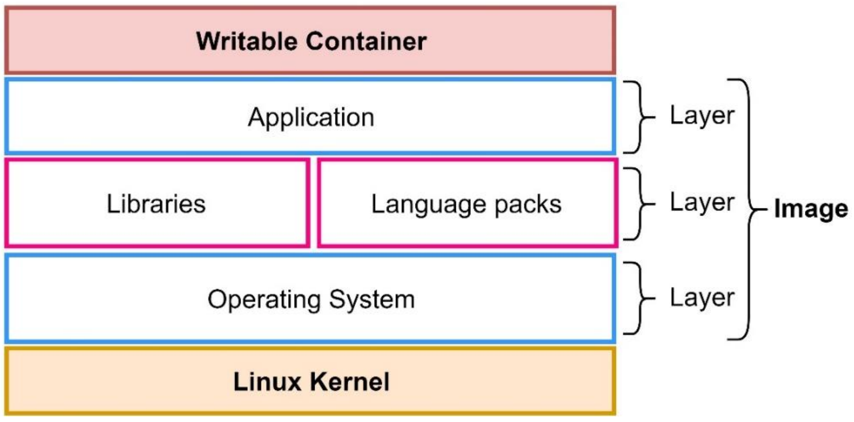


Figure 7 – Docker structure

3.3.2 Docker-Volume

One function of Docker is that a new container is set up when the software updates an image. This then replaces the old container after a successful deployment. To avoid losing the files of the old container, Docker volume is used. Volumes are directories in the host's file system that are separate from the container. This is where the files are saved that are to be retained when a container change takes place. By default, Docker automatically creates a volume as soon as a container is created. This volume directory is provided with a UID, a unique ID. However, the volumes can also be named with their own name. You can also choose your own storage location. The information about a volume can be viewed with docker inspect [14].

3.3.3 Docker Container

In a Docker container, the software of an application becomes an invisible Box containing everything the application needs to do. This includes Operating system, application code, runtime, system tools, system libraries etc. Docker containers are created from Docker images. Because images are read-only. Docker adds a read/write file system over the read-only. The file system of the image to create a container. When creating the container, Docker also creates a network interface so that the Container can communicate with the localhost. Available IP address appended. After a container has been successfully created, it can be run in any environment without making changes to have to [15].

3.3.4 Service and Stacks

If the aim is to distribute services over several computers, this increases the administrative effort immensely. Services were therefore introduced to better scale Docker. A service describes a task. As with a container, with a service, you specify which image you are referring to. The difference is that you don't take care of running services yourself. Docker takes on this task itself simplified. Such services are usually combined in groups. The stacks are used to manage these service groups. simple. The prerequisite for using services and stacks is the creation of a cluster.

3.3.5 Base Image

A base image is a basic image. Other images are mostly derived from them. The name and version are specified with the FROM command. Here you should pay attention to o labeled images. Examples of this would be Ubuntu, Alpine Linux or Debian.

3.3.6 Docker Volumes

The individual Docker containers are completely isolated from each other. So no Containers access the host's file system or from other containers. In many cases, different containers need data with each other or with the host change. In this case, volumes can be used. A volume is a path which the host and one or more containers can access. The path to a host directory can be specified when starting a Docker container with the Flag -v must be passed.

3.3.7 Cluster and Nodes

One of the advantages of ES is its ability to run as a distributed system across several nodes (servers), allowing for a higher throughput of operations than a single machine could handle. The Clusters acts as a single running system, scaled out horizontally when more space and processing power is needed for additional data. The underlying processes of ES manage the distribution of data among nodes in order to optimise the parallel execution of queries and provide data redundancy, all while remaining transparent to the user [15],[16].

A cluster consists of different types of nodes. A master node that controls the cluster, a data node that contains the data and can perform operations related to the data such as read, writes, update and delete, and a Client node that behaves as an intelligent router or coordinator, rerouting data requests to the data nodes and cluster requests to the master node.

The data imported into the system is split into several packets called shards, which are the basic scaling units in ES and can have zero or more replicas. The number of shards the data should be split into can be defined in the configuration settings. These shards are then distributed across multiple nodes, enabling several machines to work on parts of the same dataset in parallel.

There are two types of shards, primary and replica. The primary shard is the original data, and the replica is an exact copy functioning as a backup. The primary and replica shard of the same data are stored on different nodes to prevent any loss of information should a node fail. If a node fails, the replica shard automatically becomes primary, and a new replica is created on a separate node. A query can then be sent to any node, which in turn becomes the coordinator for the request. That node decides which shards the request is routed to, according to the availability of the shards and the specifications that the user has defined. Every node of every selected shard processes the query, and the results are then returned to the coordinator. The coordinator then combines the results and sends them to the user, along with the source of the documents used [15].

3.4 Events, Logs & Metrics

The data collected as part of the monitoring is grouped into three categories:

- Logs
- Metrics
- Events

In the event of a malfunction, logs are mainly used for diagnostic purposes. They can give you information about what is going on inside an operating system, an application, or a network device. In most cases, data is exchanged via textual representation and stored in log files. As a result, Syslog was created to manage logs in Linux distributions centrally. Syslog is a log format that specifies how a log message should be constructed. The official RFC3164 standard was released in 2001. Various services have been established, which receive and manage log messages according to the Syslog protocol, i.e. make them available to operating system users. Syslogd, rsyslog, and Syslog-ng are well-known examples. However, on all systemd-based Linux distributions, the Syslog protocol has been replaced by the journal by default. RHEL, CentOS, and Debian, for example, are all affected. Journal can also receive and send Syslog-formatted messages. It also gives you more options for adding metadata to your logs [18].

Metrics make it possible to quantify the properties of software and hardware. Metrics must be recorded regularly in order to be helpful. An observation or data point is a single acquisition. One Timestamp, a value, and, if necessary, several additional descriptive character attributes are included. A time series is a collection of data points that are primarily found in chronological order. The number of visitors to a website is an example of a time series. The time of measurement and the number of visitors are both included in a data point. The server name could be an additional attribute.

Granularity or Resolution refers to the recording of data points at specific time intervals. At design time, determining the granularity is a crucial decision. If the intervals are too long, essential details may be missed. When intervals are too short, the amount of data that needs to be stored and processed increases.

3.5 Rsyslog

For documentation and debugging purposes, user activities - for example, login attempts - and system events are saved in the Rsyslog relation.

Identifier	**Data type**	**Comment**
id	INT (PK)	Item ID
stamp	TIMESTAMP	current timestamp
user	INT (FK)	ID of the acting user
si	MEDIUMINT (FK)	ID of the incident involved
action	VARCHAR (40)	carried out action
status	ENUM	OK, ERROR, INFO
details	VARCHAR (255)	Possibly parameters of the activities carried out

Table 1 – Rsyslog command

After the SSH / SCP method considered in the previous section, some If weak points are noticed, the transfer of log data with rsyslog is considered here. For example, in many current Linux distributions, Debian, rsyslog is the standard Syslog daemon, or, as with SUSE, it can be installed via the respective package manager [rsy09]. With rsyslog, Syslog events over the network on another. Log host. rsyslog supports both UDP and the more reliable transmission via TCP. Likewise, to ensure the confidentiality of the log data, the transmission can be encrypted via GSSAPI or TLS. Since data is only transmitted then, the method is generally transmitted if a new log event is also transmitted. More economical with network resources than the method via SSH [12].

After the SSH / SCP method considered in the previous section, some If weak points are noticed, the transfer of log data with rsyslog is considered here. The rsyslog project homepage is *http://www.rsyslog.com*. For example, in many current Linux distributions, Debian, rsyslog is the standard Syslog daemon, or, as with SUSE, it can be installed via the respective package manager [rsy09]. With rsyslog, Syslog events over the network on another. Log host. rsyslog supports both UDP and the more reliable transmission via TCP. Likewise, to ensure the confidentiality of the log data, the transmission can be encrypted via GSSAPI or TLS. Since data is only transmitted then, the method is generally transmitted if a new log event is also transmitted. More economical with network resources than the method via SSH [19].

For rsyslog to work correctly, the file in which rsyslog the log entries writes is not in the input folder of LoginIDS. LoginIDS removes the file there regularly when it is processed. That disturbs the function of rsyslog. For example, it is better, For example, the rsyslog log file via a cron job, at intervals of several minutes ten into the input folder. In principle, too many log entries are copied, but since the files are on the same host, the I / O overhead is not as high and annoying as if the file were constantly being copied over the network. How often the log file has to be copied depends on how often LoginIDS is evaluated.

Rsyslog is a high-performance and feature-rich log processing system. It is open-source, but development and maintenance are mainly contributed by Adiscon. Adiscon also offers enterprise support for rsyslog. Rsyslog started as a regular UNIX Syslog daemon but has evolved into a system providing high-performance log processing, advanced security features, modular design with a high number of inputs, outputs and transforming options.

At the time of writing, the currently available stable version of rsyslog is 8.25.0. Rsyslog runs on Linux and can be used as a log originator, relay or Collector. It can read log messages from a variety of local inputs like flat files, UNIX sockets, systemd journal but also receive messages from remote systems over Transmission Control Protocol (TCP), User Datagram Protocol (UDP) or Reliable Event Logging Protocol (RELP). Rsyslog can process log messages from inputs using parser and message modification modules and output them using output modules. RELP can provide reliable log transfer, using application layer acknowledgement, between systems with rsyslog or other solutions supporting RELP installed. Rsyslog can also provide log integrity by specifying the use of Guardtime's Keyless Signature. Infrastructure (KSI) in file output module [19].

3.6 SSH/SCP

Using this method can be implemented on Unix operating systems with very little configuration effort, since SSH / SCP is usually pre-installed. With a classic Linux file system rights management, each service can deliver its log files via SCP If you would like to create your own user on the LoginIDS host. If the distinction between the different service users is not important, only one could be used Users are created and a different public key is available for each of the service providers Authentication. All these users are then assigned to their own group, which has write and execute rights on the LoginIDS input folder. This is the minimum of rights that are required to copy a file into the folder allowed to. With this method it is unfortunately not possible to completely prevent that too Files can be copied in the other direction, i.e. from the LoginIDS host. The group should not hold any read rights on the folder, so the contents of the folder not displayed or wildcards can be used for copying. Nevertheless, the file can be copied if the exact file name is known. Since the scheme of the filename is given relatively strictly, it must be assumed that there are only a few It is possible to try to guess the filenames of other log files. The format to which the log file names must correspond was described in Section 4.1. By implementing ¨ the real-time evaluation, in which the input folder is constantly monitored by LoginIDS, ¨ Such an attack is only possible to a very limited extent via a race condition. LoginIDS detects via Inotify when the process that copies a file into the input folder has its fi ¨ lehandle closes and is obviously finished with the copying process. Then will the file was immediately moved to a working folder that no one except LoginIDS needs to have access to. So to be able to copy log files from the input folder, the attacker would have to access the file in this short period of time. Without further testing I suspect that the chances of success are very slim, especially if the SSH access is limited to once per minute, for example by an iptables rule. In case of doubt, the log file can be given a random file extension [20].

Another possible case that LoginIDS is not currently on the LoginIDS host is running and therefore cannot remove the log files from the input folder, can be covered by the fact that LoginIDS of the service group has wx rights when starting the input folder and removes it when you exit. This enables services cannot access the folder at all while LoginIDS is not running. This requires a certain amount of error handling on the service host, which in the event of a failed transmission.

Furthermore, it must be considered how often log files are transferred with SCP. LoginIDS performs a difference detection so that the same log file, can be copied several times to the LoginIDS host without duplicating the first entries to be analysed. However, depending on the network load, it does not make sense to copy the complete log file every minute without knowing exactly whether new log entries have been written at all were. The SSH / SCP method is therefore not suitable for real-time evaluation. Depending on The size of the network and the log files was an hourly to daily transmission keep the network load within limits [21].

4. Data Characteristic

Telemetry data is textual data produced by software systems to record system runtime information. Telemetry is often the only way to get detailed runtime information about a system. Virtually any computer programme generates some kind of telemetry data.

Typically, this data consists of a timestamp, the message's verbosity, followed by a free-text message that could contain any information capturing the state of the application.

Given we are monitoring multiple distributed sources, the volumes quickly become too large and impractical to be analysed without a specialised telemetry monitoring system.

4.1 Telemetry data example

For the purpose of testing a test batch of data was required. This data was gathered over a span of two days (7th – 8th April) and contains telemetry information from a virtual machine running multiple applications (browser,..). The amount of data gathered for testing would not usually require the use of ELK however it is sufficient to demonstrate the functionality.

This data entry consists of a timestamp, a system's hostname (where the message originated from), a record ID, and a CPU message itself.

Below are two different examples of how RAW data looks like. Each message contains an average of about 40 attributes. In our case, we use only a part of the parameters. In general, incoming data characterises server performance, network usage, server load...

The first information in the message says which service created the record. In our case was created by metricbeat_1 agent. The next parameter 2021-04-07T14:27:42.101Z represents the time stamp when exactly the given record was created.

Example 1:

```
metricbeat_1|2021-04-07T14:27:42.101Z  INFO [monitoring] log/log.go:145 Non-zero
metrics in the last 30s           {"monitoring":{"metrics":
{"beat":{"cpu":{"system":{"ticks":79920,"time":{"ms":151}},"total":{"ticks":149740,"time
":{"ms":287},"value":149740},"user":{"ticks":69820,"time":{"ms":136}}},"hadles":{"limit":
{"hard":1048576,"soft":1048576},"open":7},"info":{"ephemeral_id":"2ef928e4-5e67-
4212-
819f583a025ff563","uptime":{"ms":13350140}},"memstats":{"gc_next":17176608,"memo
ry_alloc":13084216,"memory_total":22456700976},"runtime":{"goroutines":27}},"libbeat
":{"config":{"module":{"running":0}},"output":{"events":{"acked":60,"batches":22,"total"
:60},"read":{"bytes":132},"write":{"bytes":18088}},"pipeline":{"clients":2,"events":{"activ
e":1,"published":60,"total":60},"queue":{"acked":60}}},"metricbeat":{"system":{"cpu":{"e
vents":30,"success":30},"memory":{"events":30,"success":30}}},"system":{"load":{"1":0.0
5,"15":0.02,"5":0.06,"norm":{"1":0.0167,"15":0.0067,"5":0.02}}}}}}
```

Example 2:

```
metricbeat_1  | 2021-04-08T11:22:06.628Z INFO  [monitoring]   log/log.go:145 Non-zero
metrics in the last 30s           {"monitoring": {"metrics":
{"beat":{"cpu":{"system":{"ticks":179890,"time":{"ms":290}},"total":{"ticks":325810,"tim
e":{"ms":564},"value":325810},"user":{"ticks":145920,"time":{"ms":274}}},"handles":{"lim
it":{"hard":1048576,"soft":1048576},"open":7},"info":{"ephemeral_id":"3e15220b-7287-
4557-a584-
63ca0bca063b","uptime":{"ms":18510821}},"memstats":{"gc_next":16782144,"memory_
alloc":10700704,"memory_total":37263831280},"runtime":{"goroutines":39}},"libbeat":{"
config":{"module":{"running":0}},"output":{"events":{"acked":243,"batches":27,"total":2
43},"read":{"bytes":162},"write":{"bytes":35954}},"pipeline":{"clients":5,"events":{"active
":3,"published":240,"total":240},"queue":{"acked":243}}},"metricbeat":{"system":{"core":
{"events":90,"success":90},"cpu":{"events":30,"success":30},"load":{"events":30,"success
":30},"memory":{"events":30,"success":30},"network":{"events":60,"success":60}}},"syste
m":{"load":{"1":0.05,"15":0.09,"5":0.09,"norm":{"1":0.0167,"15":0.03,"5":0.03
```

5. Experimental Setup

This chapter describes the test setup used. It consists of two dockerised Debian servers run locally on virtual machines, first acting as the source of data (imitation of a PBX server) second implementing the centralised data collector and storage. The second also hosts the data visualisation (Kibana).

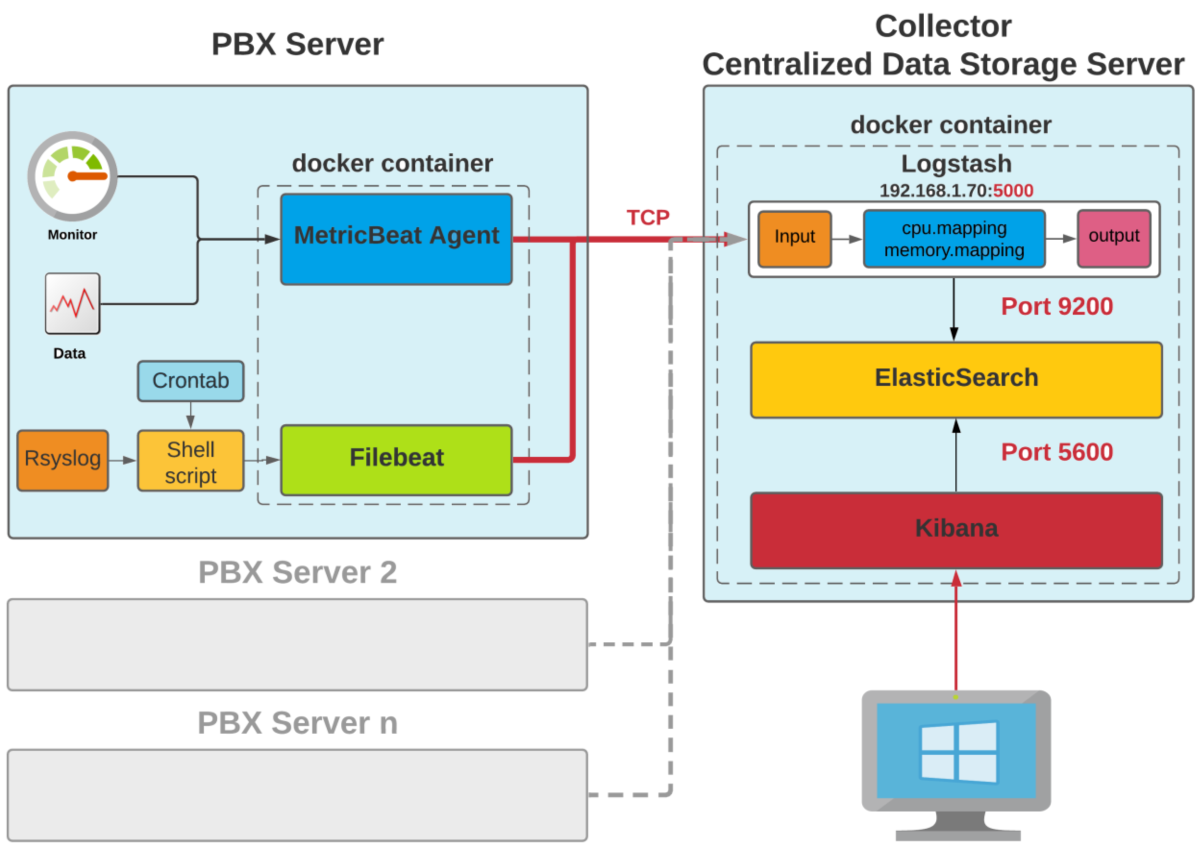


Figure 8 – Experimental Setup scheme

In this thesis, the implemented centralised telemetry monitoring framework achieves its goals and provides several benefits. Most importantly, all data is now accessible in a single location, eliminating the need to search through multiple data files stored on multiple server machines, which was the centralised telemetry section's sole purpose. Although the centralised telemetry system did not introduce any significant additional issues to the overall setup, it did require some manual configuration changes due to the possibility of data files being stored in multiple locations.

The data analysis framework also achieves its goals and provides numerous benefits for system telemetry monitoring. Kibana's time series visualisations show details about the overall system health and whether data messages are present

5.1 Configurations of virtual machines

As mentioned above, the test setup components run on virtual machines that communicate with each other and simulate a simplified production environment. The VMs parameters are describes following in Table 2.

Data Collector & Storage server	Configuration	8GB Memory
		4 processors
		80GB disk
		2 network adapters
		IP address 192.168.1.78
	Installed Software	Elasticsearch 7.7
		Kibana 7.7
		Docker
PBX server	Configuration	4GB Memory
		3 processors
		40GB disk
		2 network adapters
	Installed Software	Docker
		Filebeat 7.7
		Metricbeat
		Apache Web Server
		Free PBX 15

Table 2 – Configuration of Virtual machine

5.2 PBX Server

The following paragraph describes the configuration of the PBX server in *Figure 8*. This server is the source of telemetry data.

5.2.1 Configuring docker-compose

Docker-compose can build a multi-component application by composing a set of components, each of which is made up of an image and a set of options that determine how the component shall behave. The same image can be used for multiple components, but the reused images will create different components. To specify the composition of components, a configuration file such as docker-compose.yml is used. A Docker Compose file is shown below, which assembles a multi-component framework from the web components. A local image is used to represent the web component, which is generated by a Docker file.

The following script aims to create two services. The first service is filebeat, which is used to send telemetry data from a PBX server to the receiving server. The second service that runs within the Docker-compose file is Metricbeat. This is an agent that downloads telemetry information about the PBX server. An essential part of the script is setting volumes. If we did not set the four lines under the volumes tag, the telemetry data would not be collected from the PBX server but the Docker container.

```
# docker-compose.yml
version: '2.1'
services:
  filebeat:
    image: elastic/filebeat:7.7.0
    restart: always
    volumes:
      - ./filebeat/filebeat.yml:/usr/share/filebeat/filebeat.yml:ro
      - /home/tom/data:/usr/share/filebeat/data
    hostname: filebeat
   metricbeat:
    image: elastic/metricbeat:7.7.0
    restart: always
    volumes:
      - ./metricbeat/metricbeat.yml:/usr/share/metricbeat/metric-
beat.yml:ro
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - /sys/fs/cgroup:/hostfs/sys/fs/cgroup:ro
      - /proc:/hostfs/proc:ro
      - /:/hostfs:ro
    hostname: metricbeat
```

5.2.2 Configuring Filebeat

Filebeat's job is to read log files and send the log messages across the network to a Logstash. Filebeat's configuration lets you choose which fields appear in the JSON format messages it sends to Logstash or elsewhere. Log parsing and field modifications must be done in Logstash otherwise.

The example below demonstrates how to set up Filebeat to send logs to a Logstash. Configuring Encryption and authentication is optional if the Filebeat and Logstash servers are connected to a secure network.

```
#/home/tom/filebeat.yml
filebeat.inputs:
  -type:log
    paths:
      -/usr/share/filebeat/data/sys-data*
    document_type:log

#setup.template.name:"filebeat-"
#setup.template.pattern:"filebeat-*"
#setup.dashboards.enabled:true

output.logstash:
  hosts:["192.168.1.70:5000"]
#   index: "filebeat-%{+yyyy.MM.dd}"
```

Filebeat requires complete file paths, which can be problematic if the log files are stored in different locations. Because Docker-compose is usually run in the user's home folder who installed it, the full path to the log files is frequently different. A new log file is created every day, and each time the device is restarted. Because it's made to track all files with the .log extension, Filebeat detects a new log file and starts following it.

```
2021-04-08T00:15:01Z debian INF  (root) CMD (/root/scripts/copy.sh)
2021-04-08T00:17:01Z debian INF  (root) CMD (/root/scripts/copy.sh)
2021-04-08T00:17:01Z debian INF  (root) CMD (cd/&&run-report/etc/cron.hourly)
2021-04-08T00:18:01Z debian INF  (root) CMD (/root/scripts/copy.sh)
```

5.2.3 Configuring Metricbeat agent

The Metricbeat configuration is straightforward. We set the Metricbeat to monitor the CPU and memory usage. This telemetry data will then be essential to us when creating the graphical output. The sampling frequency is set to 1 second. The configuration is similar to the previous configuration of Filebeat.

```yaml
#metricbeat.yml
metricbeat.modules:
- module: system
  metricsets: ["cpu","memory","load","network","core","process","process_summary","socket_summary","filesystem","fsstat","uptime"]
  enabled: true
  period: 1s
  processes: ['.*']
  cpu.metrics: ["normalized_percentages"]

output.logstash:
  hosts: ["192.168.1.83:5000"]
```

5.2.4 Configuring rsyslog

The instructions for installing rsyslog can be found on the rsyslog home page. Because rsyslog is Debian's default Syslog daemon, an update was performed rather than a fresh installation. The repository for version eight was set up, and rsyslog 8.25 was installed alongside the additional package rsyslog-help-8.25.0, which met the requirements. Rsyslog can read both the log socket (/dev/log) and the system journal. Log socket input should be used instead of journal input, according to the rsyslog documentation. In the Rsyslog configuration, it is necessary to add line number 24. This line adds a timestamp in UTC format to the beginning of the log record.

```
# root@debian: /var/log# vim /etc/rsyslog.conf
# /etc/rsyslog.conf configuration file for rsyslog
# For more information install rsyslog-doc and see
# /usr/share/doc/rsyslog-doc/html/configuration/index.html
#################
#### MODULES ####
#################
module(load="imuxsock") # provides support for local system logging
module(load="imklog")   # provides kernel logging support
#module(load="immark")  # provides --MARK-- message capability
# provides UDP syslog reception
#module(load="imudp")
#input(type="imudp" port="514")
# provides TCP syslog reception
#module(load="imtcp")
#input(type="imtcp" port="514")
###########################
#### GLOBAL DIRECTIVES ####
###########################
#
# Use traditional timestamp format.
# To enable high precision timestamps, comment out the following line.
#
$template Mytemplate,"%$year%-%$month%-%$day%T%timegenerated:12:19:date-rfc3339%Z
%HOSTNAME% %syslogseverity-text:0:3:uppercase% %msg%\n"
$ActionFileDefaultTemplate Mytemplate
#
# Set the default permissions for all log files.
#
```

5.2.5 Configuring crontab

For our purposes, it was necessary to write a new script. The script runs every minute and invokes our script "copy.sh".

```
#root@debian:~# vim /etc/crontab
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
#  .---------------- minute (0 - 59)
#  |  .------------- hour (0 - 23)
#  |  |  .---------- day of month (1 - 31)
#  |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
#  |  |  |  |  .---- day of week (0-6)(Sunday=0 or 7)OR sun,mon,tue,wed,thu,fri,sat
#  |  |  |  |  |
#  *  *  *  *  * user-name command to be executed
17 * * * *   root cd / && run-parts --report /etc/cron.hourly
25 6 * * *   root test-x/usr/sbin/anacron||(cd/&&run-parts--report/etc/cron.daily)
47 6 * * 7   root test-x/usr/sbin/anacron||(cd/&&run-parts--report/etc/cron.weekly)
52 6 1 * *   root test-x/usr/sbin/anacron||(cd/&&run-parts--report/etc/cron.monthly)
#
* * * * * root /root/scripts/copy.sh
```

5.2.6 Configuring shell script

After each call by Crontab, this script downloads (line 5) the last entry from the rsyslog (/var/log/syslog) and copies it to the filebeat folder (*home/tom/data /sys-data folder*).

```
1 #root@debian:~# vim scripts/copy.sh
2 #! /bin/bash
3
4 echo DEBUG >>/root/bla.txt
5 cp /var/log/syslog /home/tom/data/sys-data
```

5.3 Collector, Data storage, visualisation server

The following paragraph describes the configuration of the Collector, Data storage, visualisation server. The server is depicted in Figure 9 on the right side. This server is the receiver of telemetry data and logs data.

5.3.1 Configuring Docker-compose

The first step for getting an ELK Stack up and running is to install Collector, a Data storage server on a virtual machine. The advantage of running Elasticsearch on the Debian server is that much of the Elasticsearch optimisation work is done on Linux. Furthermore, most documentation, including the official one on Elastic's website, assumes the user runs Elasticsearch on Linux.

Java 8 is installed in the virtual machine because the entire ELK Stack is written in Java and requires a Java runtime environment to function.

The applications are run in a Docker container. The benefit is that it starts all the applications at once, eliminating the need to launch them separately. The configuration of the docker-compose.yml file developed for this purpose is displayed in List (docker-compose.yml).

```
#docker-compose.yml
version:'2.1'
services:
  kibana:
    image:docker.elastic.co/kibana/kibana:7.7.0
    ports:
      -"5601:5601"
    environment:
      -"ELASTICSEARCH_URL=http://localhost:9200"
    depends_on:
      -elasticsearch
    hostname:kibana
  elasticsearch:
    image:docker.elastic.co/elasticsearch/elasticsearch:7.7.0
    environment:
      -"discovery.type=single-node"
    hostname:elasticsearch
    ports:
      -"9200:9200"
      -"9300:9300"
  logstash:
    image:logstash:7.7.0
    hostname:logstash
    volumes:
     -./logstash/comm:/usr/share/logstash/comm:ro
     -./logstash/logstash.yml:/usr/share/logstash/config/logstash.yml:ro
    ports:
     - "5000:5000"
```

Elasticsearch does not need any modification because it automatically binds to loopback addresses and listens on port 9200, and Logstash would be able to link to Elasticsearch if it is running on the same server machine. By default, Kibana will use port 5601.

5.3.2 Configuring Logstash

A single Logstash instance running on a single server machine might be sufficient in a centralised telemetry system if the telemetry processing throughput is not too high. Multiple Logstash instances could also be run to scale them horizontally, allowing Metricbeat to distribute data equally to each instance. Logstash aims to receive data events from one or more Metricbeat instances and forward all of these events into a single Elasticsearch database.

Logstash, like Metricbeat, uses the same configuration file format. We can configure parameters such as worker thread count and internal event queue settings. However, the other configuration file is in a different format and is used to set inputs, output, and data message processing. Again, inside a closed network, Encryption and authentication are not entirely necessary. Set the input port for Collector at address 5000. This configuration is shown in the example below:

```
//input.conf
input{
  beats{
        port=>5000
        ssl=>false
  }

```

The connection to Elasticsearch uses HTTP or HTTPS because Elasticsearch must be contacted through its REST API. Although HTTPS can be used for secure connection and server authentication, Elasticsearch and Logstash do not support it because HTTPS rarely enforces client authentication. In a closed network, authentication is also unnecessary.

The example below shows the configuration with localhost and indexing data. Appropriate filters are designed to analyse and process data. Based on the tag, the filters create a specific index for Elasticsearch. For example, on lines 10 to 16, if the incoming data is in the system memory tag, we will create *index_memory* plus the date.

```
output{
if "system log" in [tags] {
  elasticsearch{
    hosts=> ["elasticsearch:9200"]
    manage_template=> false
    document_type=> "log"
    index=> "index_log-%{+YYYY.MM.dd}"
   }
}
else if "system memory" in [tags] {
  elasticsearch{
    hosts=> ["elasticsearch:9200"]
    manage_template=> false
    document_type=> "log"
    index=> "index_memory-%{+YYYY.MM.dd}"
   }
}
else if "system cpu" in [tags] {
  elasticsearch{
    hosts=> ["elasticsearch:9200"]
    manage_template=> false
    document_type=> "log"
    index=> "index_cpu-%{+YYYY.MM.dd}"
   }
}
else if "system core" in [tags] {
  elasticsearch{
    hosts=> ["elasticsearch:9200"]
    manage_template=> false
    document_type=> "log"
    index=> "index_core-%{+YYYY.MM.dd}"
   }
}
else if "system load" in [tags] {
  elasticsearch{
    hosts=> ["elasticsearch:9200"]
    manage_template=> false
    document_type=> "log"
    index=> "index_load-%{+YYYY.MM.dd}"
   }
}
else if "system network" in [tags] {
  elasticsearch{
    hosts=> ["elasticsearch:9200"]
    manage_template=> false
    document_type=> "log"
    index=> "index_network-%{+YYYY.MM.dd}"
   }
}
else {
    elasticsearch{
    hosts=> ["elasticsearch:9200"]
```

A Logstash filter can be used to remove unnecessary data fields from a case. Furthermore, if the file path is not needed, it makes sense to delete it from the source field and retain the filename. However, telemetry and log files in separate directories have the same name, which should be considered when developing the system. The Logstash configuration below demonstrates how to use the grok filter to construct a state. The filter can be split up into many configuration files. Filters are automatically sorted by file name. Elasticsearch is not very resistant to errors, which the administrator must therefore handle.

Firstly, we check if the field "*message*" entry contains our UTC timestamp pattern at all. In case it does, the message is tagged with the "correct timestamp" label. This can later be used to filter out only the correctly formatted messages.

Secondly, the condition is used to convert the Timestamp to a different format for practicality.

Lastly, in case one of the two conditions fails, the message is stored in a new field, "*rest_message*", to prevent data loss.

```
filter{

 if [agent][type] in "filebeat" {

 if [message] {
  grok {
  match => {"message" => "%{TIMESTAMP_ISO8601:[@metadata][isotimestamp]} debian
%{GREEDYDATA:[@metadata][true_message]}"}
  break_on_match => false
  add_tag => ["system log"]
}
  if [@metadata][isotimestamp] {
  date {
 match=> ["[@metadata][isotimestamp]", "ISO8601" ]
}
}
if [@metadata][true_message]
{
mutate {
add_field => {"rest_message" => "%{[@metadata][true_message]}"}
}
}
}
}
}
```

The script described below works similarly to the previous script. The following lines are designed to map the "system memory" tag to when the filter condition is met. The telemetry data in this tag describes how much RAM is occupied. For data processing, there is a problem that all information is in bytes. Therefore, we will create a new field, "RAM_GB" on line 6, and to which we will copy our original data from the field "[system] [memory] [used] [bytes]". Then we use the ruby function and divide the data into a suitable format.

```
 1 # memory_mapping
 2 filter{
 3  if [agent][type] in "metricbeat" and [event][dataset] in "system.memory"
   {
 4  mutate{add_tag => "system memory" }
 5  if [system][memory][used][bytes] {
 6   mutate{add_field => { "RAM_B" => "%{[system][memory][used][bytes]}"}}
 7
 8   ruby {
 9    code => "ramgb = event.get('[RAM_B]').to_f/1000000000
10 event.set('[RAM_GB]', ramgb.round(2))
11 "
12 }
13 }
14 }
15 }
```

6. RESULTS

This chapter will showcase the results obtained in this bachelor's thesis - namely, the resulting Dashboard. This Dashboard is generated from data gathered from a test PBX server and collected and stored on a Collector, Data Storage Server. The setup is described in detail in chapters Experimental setup.

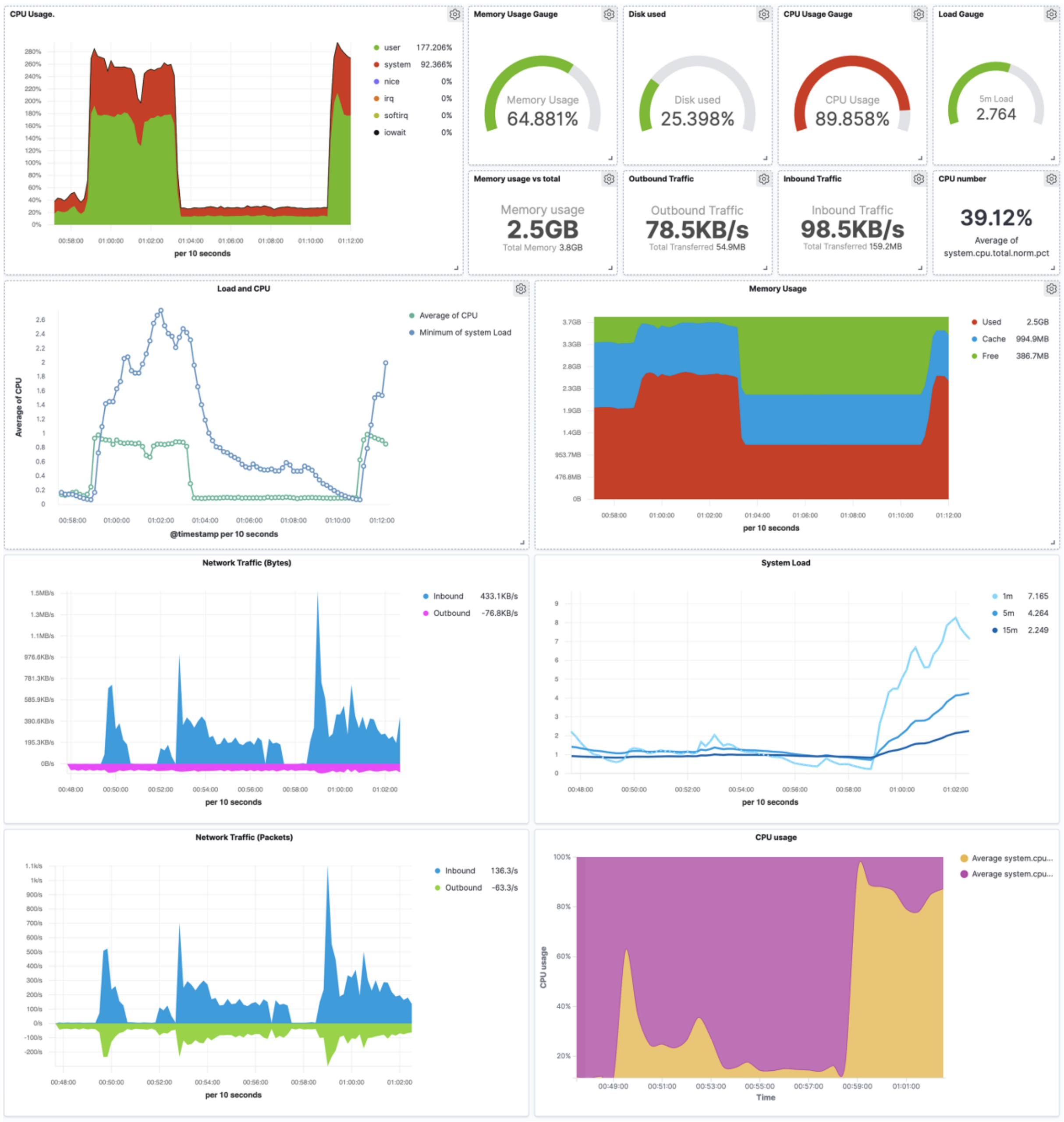


Figure 9 - Overview dashboard

In **Figure 9**, we can see the whole Dashboard. The Dashboard displays essential information about the PBX server. We chose the time window April 8 from 0:48 to 1:03 for this preview. According to the user's needs, these parameters can be changed to select only a specific time period (from-to).

Figure 9 is only illustrative to show what the entire Dashboard looks like. Only a subset of the graphs can be seen in one view on the browser. This is due to the limited screen size. For this purpose, I have combined several screenshots to create a better idea of the Dashboard.

The layout of the individual charts in the Dashboard was selected based on experience gained while creating a graphic interface for a web application.

The purpose of the Dashboard is to provide the user with a comprehensive overview of the selected PBX server. The information a user needs to know about the remote server is CPU, RAM usage, network usage, load usage. In our case, we monitor only one PBX server. The plotted graphs always concern only one server. Currently, there is no option to display multiple servers at once. Displaying data from all servers at once will be a future task. At the moment, we did not solve this due to time constraints. Another problem with displaying data from all servers is the readability of the graphs. We select the appropriate server using filters in Kibana.

The rest of the chapter describes the selected graphs that I considered necessary or interesting to bring to the reader.

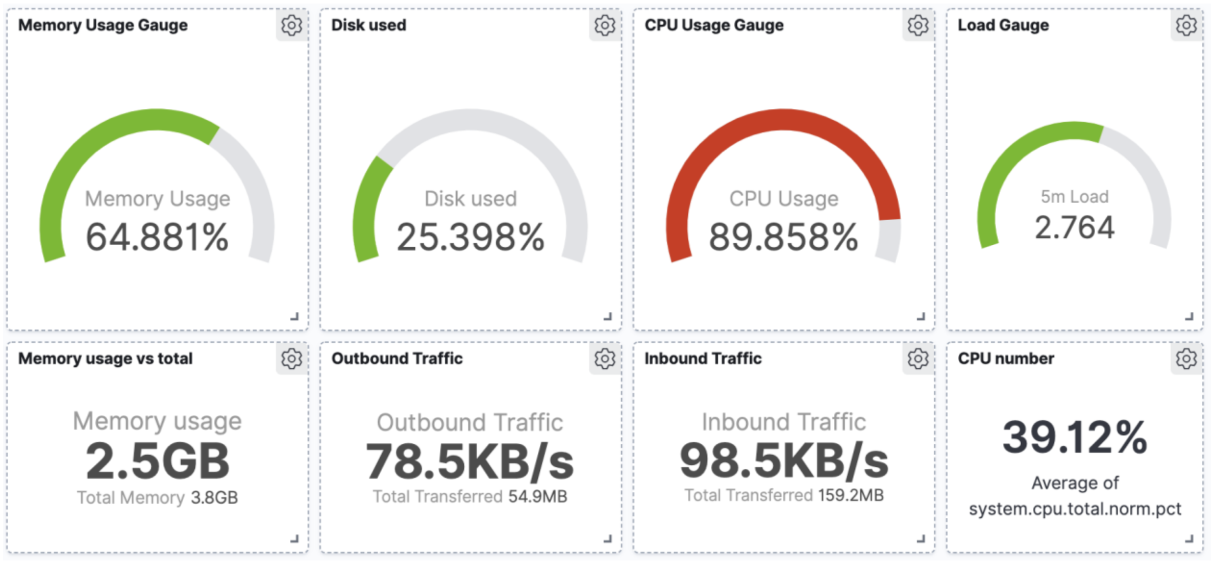


Figure 10 - Dashboard gauges

Figure 10 depicts four gauges in the first row and four text information on the second row of cells.

- The first indicator gives us data on how much RAM is currently used.
- The second gauge, "Disk used", shows the total capacity of the hard drive used.
- The third gauge shows the current CPU usage.

As you can see in our case, the total performance is 89%. The last gauge shows us the average server load for the last 5 minutes.

The Memory usage vs total indicator shows us the data on how much Memory is used from the total allocated RAM. This indicator shows the current load on the Internet. Here we can see that 78.5KB per second is currently being sent. The following indicator shows us how much data is currently being received from the network (98.5KB per second).

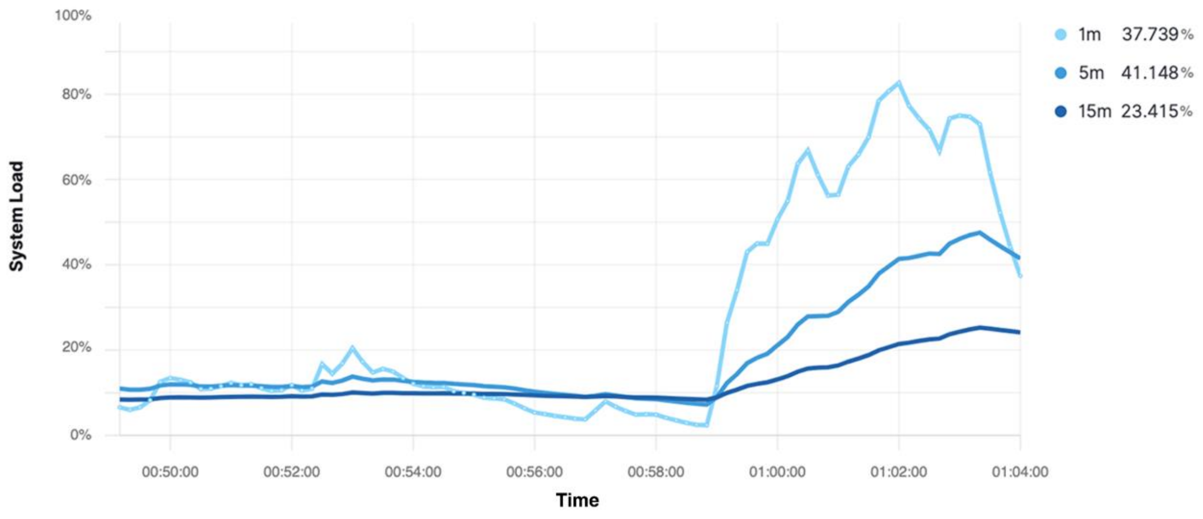


Figure 11 - Load

Figure 11 shows the traffic (system load). The x-axis shows the timeline for the last 15 minutes and the timestep set (10 seconds). Y-axis shows the volume of the traffic. With the maximum value being 100%, there are three different curves in the graph. It is possible to see that all waveforms show us a volume. However, they are averaged to a different value (1min, 5 min, 15min). At 00:59, there is sudden server traffic. While the load curve starts to rise sharply within 1 minute, the change in values in 5 and 15 minutes is smoother.

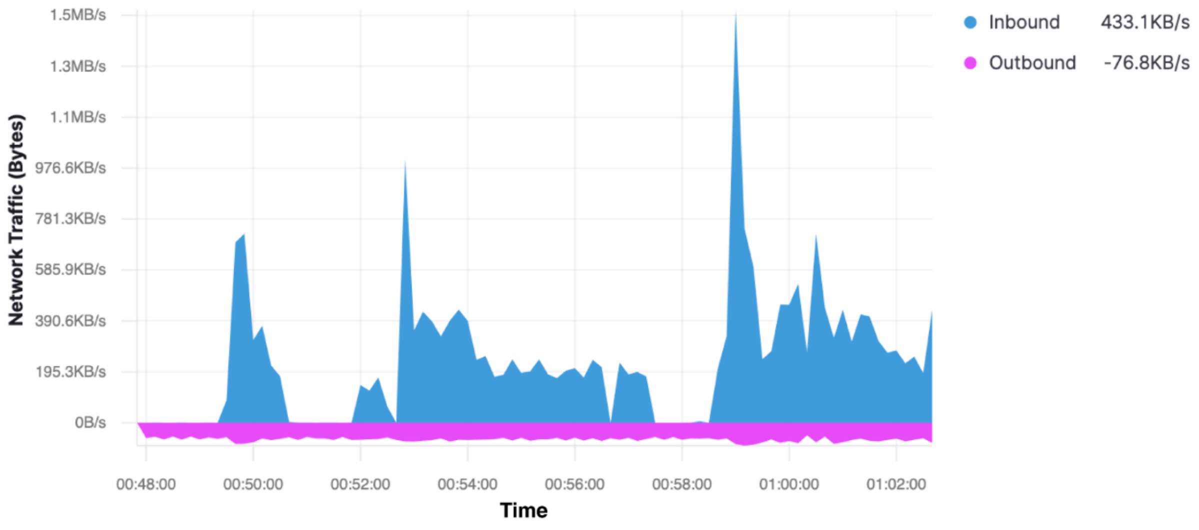


Figure 12 – Network traffic

Figure 12 depicts the use of the network interface on server PBX. The x-axis shows the timeline for the last 15 minutes and the timestep set for 10 seconds. Y-axis shows how much data flowed from our server or, vice-versa, how much data the server received. The data received from the network is shown in blue colour. The purple part of the graph represents outgoing data from the server. Based on the legend at the top right, we can see the current volume of data received and data sent.

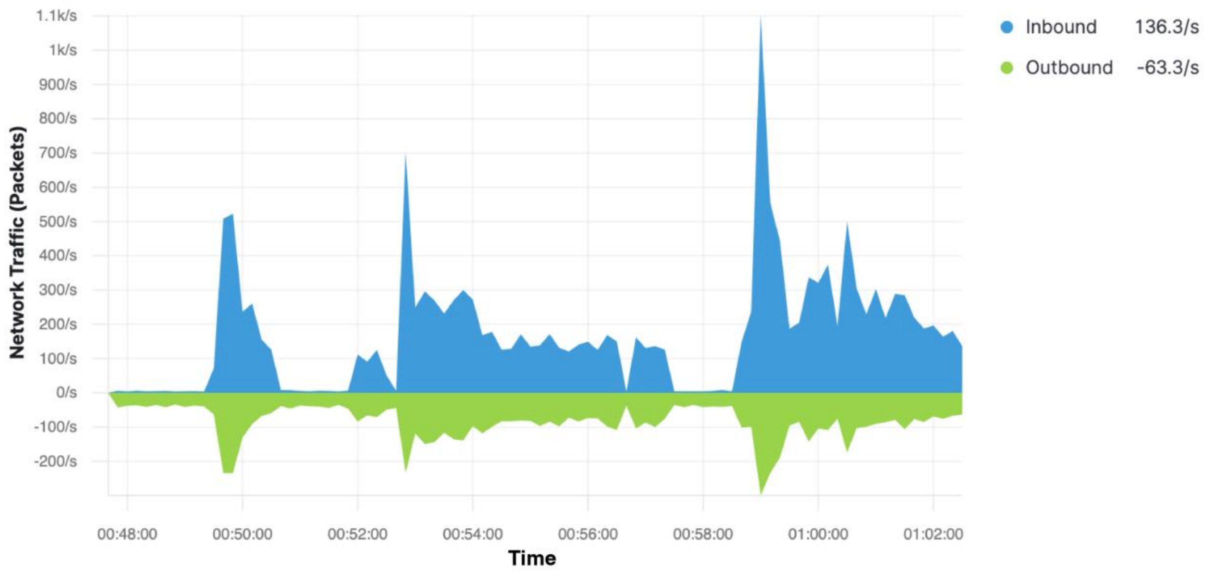


Figure 13 – Network traffic packets

Figure 13 directly correlates with **Figure 12**. When shows the number of packets received/sent. Comparing these two graphs shows that the increased traffic of the network has increased the volume of packets sent.

We can see a total of 3 peaks at the following times: 0:50, 0:53, 0:59. These sharp fluctuations of the site were probably caused by a significant increase in incoming queries to our PBX server. A pattern can be found in the signal, followed by a gradual attenuation after each peak.

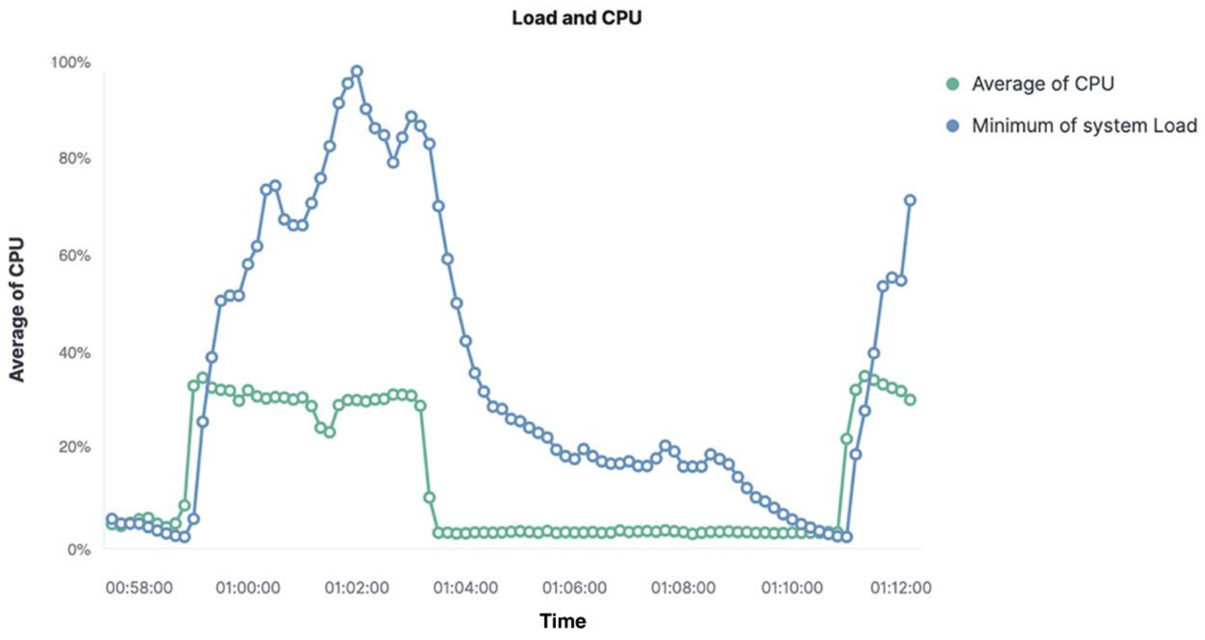


Figure 14 – Load vs CPU usage

Figure 14 shows the CPU dependence on traffic (server load). The graph nicely shows a corresponding increase in server performance in the case of an increase in workload (more queries per one server).

In Figure 14, we can see the load peak, which started to increase at 0:59. As the workload increases, so does the overall performance of the server. Performance reaches the limit of 40%, which is constantly held until 1:04. It is essential to say that the CPU power value is averaged. The actual current power curve would look a little different.

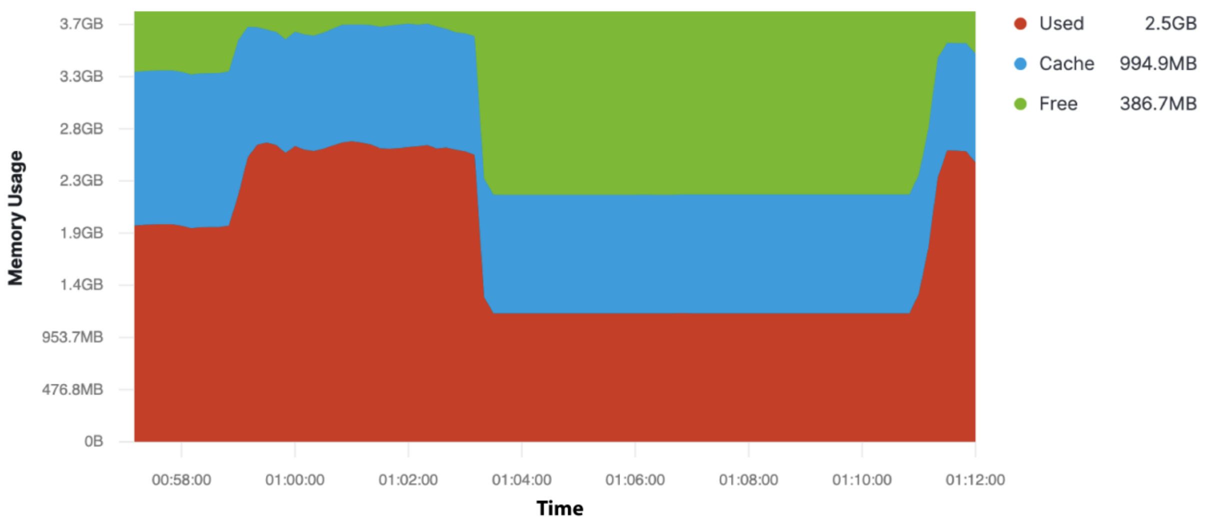


Figure 15 – Memory usage

Figure 15 describes the use of RAM. The y-axis shows the capacity in GB, where the maximum value is 3.8 GB. The x-axis shows the timeframe of the last 15 minutes. The correlation between used Memory and cache memory is well visible in the course. In our case, the cache memory holds about 1 GB which meets the desirable values.

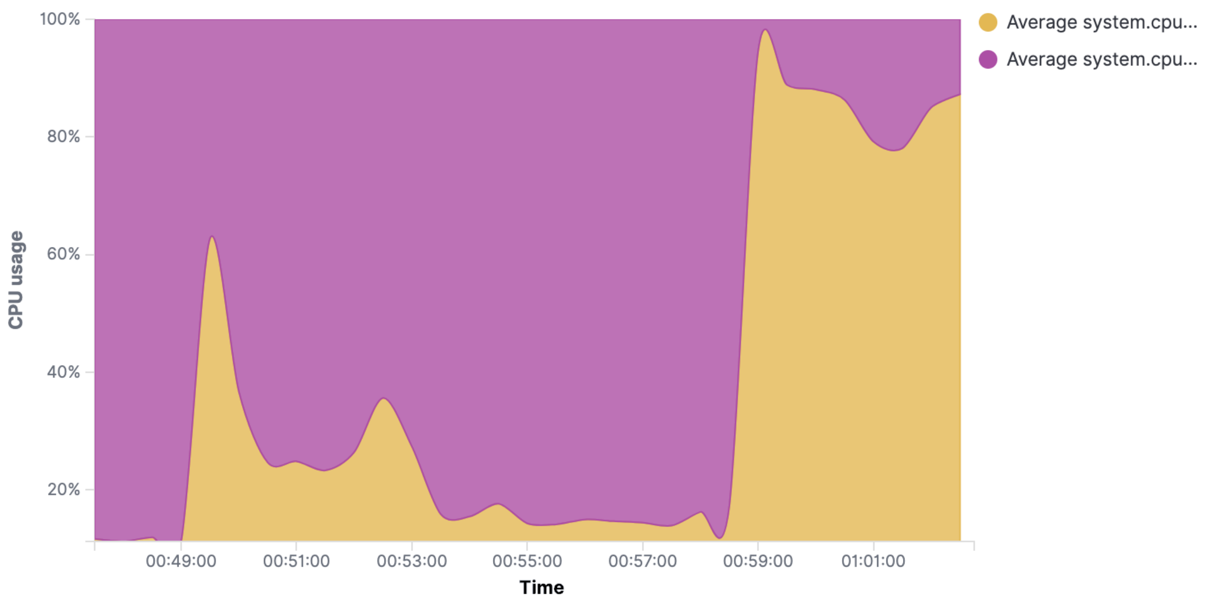


Figure 16 – Basic CPU usage

The following **Figure 16** shows the simple use of CPU over time. The y-axis shows the percentage of power utilisation. Here the scale is normalised up to 100% and cannot exceed this limit (mentioning this fact since, e.g. in Figure 17, we go beyond this limit). The x-axis shows the time window of the last 15 minutes. We can see a total of 3 peaks. The first occurred at 0:49 and the second at 0:53, and the third at 0:59. The most significant change was the third peak at

0:59 when the server was at maximum load. In our case, this is mainly since our server is virtualised and the computing capacity is undersized compared to the actual physical server. Therefore, it is not difficult to load our PBX server to the maximum. In a relational application, such a flow would mean a redesign of the server and a change in hardware parameters.

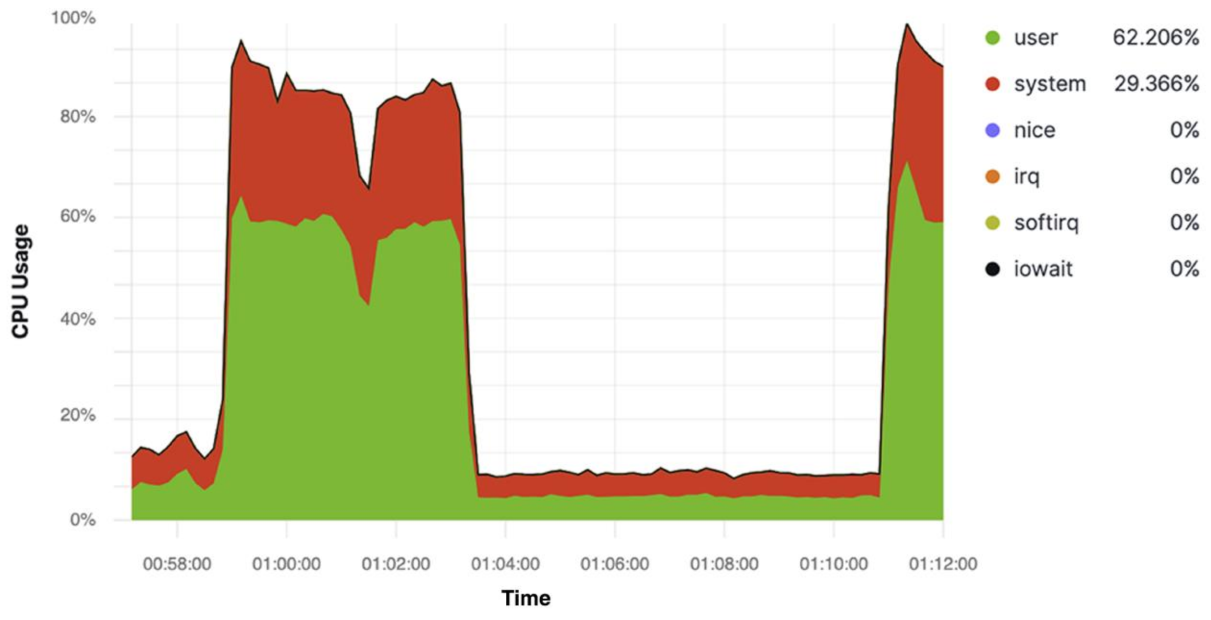


Figure 17 – CPU usage by services

Figure 17 describes the distribution of performance between individual "services". In total, the course consists of two services in our case. It clearly shows what percentage the system uses.

The green colour describes One level up, and the "user" CPU state offers CPU time used by user-space processes. These are higher-level processes, like your application or the database server running on your machine. In short, every CPU time used by anything other than the kernel is marked "user", even if it was not started from any user account. If a user-space process needs access to the hardware, it needs to ask the kernel, which would count towards the "system" state.

The red colour shows how much power the server operation itself consumes. It is clear from the figure that these are constant values ranging between 5% and 30%.

The x-axis shows the power in percentages. The monitored PBX server contains three cores. Therefore, the maximum value would be 100%.

In this case, we can see a total of 2 large tips. The first occurred at 0:59 and the second at 1:11. These peaks were caused by the launch of several applications that required these system resources. As a result of undersizing our PBX server's parameters, the RAMs were relatively fast at maximum filling (100%). In a practical application, we would solve this problem by increasing the number of RAM memory.

7. CONCLUSION

This work qualified and investigated various solutions for the collection and analysis of telemetry data. In addition to the input itself, a logs collection has been added to the metric data collection. The overall goal is to create a robust solution for collecting data from a PBX Server based on the Debian operating system. Implement a suitable data collection agent on this server. Lastly, it was necessary to create a Server that will be used for centralised data collection and storage. It is possible to filter, analyse and visualise data in real-time on this server.

In the first step of the solution, we qualified the OpenTelemetry system. This solution was found to be very sophisticated. By the end of my work, my knowledge did not reach the level to fully grasp the whole structure of the solution and implement it in another application. OpenTelemetry is a tool containing a wide range of services. Examples are Prometheus, Zipkin and Jaeger tools.

Next, I explored another way to see if communication between the two servers could be resolved using SSH / SCP technologies. After educating myself in this technology, I created a service between two Debian servers. I was able to send log data to another server. I sent the data in text form. Even though the solution met the "communication between two servers" task, it nevertheless had several disadvantages. For example, sending data was insecure - the solution could not resolve duplicate data. Even though the applied technology was very unsuitable for my application combined with Big Data, I found this experience very beneficial. In chapter 5, Experimental setup, I decided to omit this methodology because it did not lead directly to a successful solution to the given problem.

Based on this previous experience, I came to the consideration described in more detail in Chapter 5. This chapter deals with detailed back-end implementation. After configuring and creating all services, I managed to create a robust centralised data monitoring system. Whole solution development took place in a closed virtual network in VirtualBox. After creating a functional solution, it was possible to analyse incoming data in Kibana. The graphical output was in the form of dashboards plotting incoming telemetry data in real-time. The final testing showed that the communication occurred with a more negligible delay, which did not exceed more than 10 seconds.

A centralised telemetry monitoring offers some obvious benefits. One of the advantages is the ability to scale the created solution and deploy it to other instances of PBX servers based on the Debian operating system. Another benefit is creating a service based on Docker containers technology when it is possible to create an image of a given solution and deploy it on another server.

Due to the short time period of the bachelor's thesis, together with security reasons, only testing servers environments were used. Although our system did not work with big data that way, the potential service was to process big data. In addition to the scope of the considered assignment, I also collected logs from the PBX server. In my work, I describe the solution only marginally because it was not the aim of the work. In any case, I can undoubtedly say that the solution created for collecting logs is not optimal. The problem, for example, is the impossibility of using a Docker container and a simple deployment.

Based on this work presented, it would be interesting to visualise data from multiple servers PBX in one Dashboard. Another interesting thing would be to try out a deployment of the service on a large number of PBX servers.

BIBLIOGRAPHY

[1] What is Telemetry? How Telemetry Works, Benefits, and Tutorial. *Stackify* [online]. 26. April 2017 [cit. 2021-05-21]. Retrieved: https://stackify.com/telemetry-tutorial/

[2] The Complete Guide to the ELK Stack. *Logz.io* [online]. 1. April 2021 [cit. 2021-05-21]. Retrieved: https://logz.io/learn/complete-guide-elk-stack/

[3] *ELK Stack Tutorial: What is Kibana, Logstash & Elasticsearch?* [online]. [cit. 2021-05-21]. Retrieved: https://www.guru99.com/elk-stack-tutorial.html

[4] SHARMA, Vishal. *Beginning Elastic Stack* [online]. 1st ed. 2016. Berkeley, CA: Apress : Imprint: Apress, 2016. ISBN 978-1-4842-1694-1. Retrieved: doi:10.1007/978-1-4842-1694-1

[5] Elasticsearch: The Official Distributed Search & Analytics Engine. *Elastic* [online]. [cit. 2021-05-21]. Retrieved: https://www.elastic.co/elasticsearch

[6] *Elasticsearch* [online]. 2021 [cit. 2021-05-21]. Retrieved: https://en.wikipedia.org/w/index.php?title=Elasticsearch&oldid=1022518233

[7] ZAMFIR, Vlad-Andrei, M. CARABAS, Costin CARABAS a N. TAPUS. Systems Monitoring and Big Data Analysis Using the Elasticsearch System. *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*. 2019, 188–193.

[8] Beats: Data Shippers for Elasticsearch. *Elastic* [online]. [cit. 2021-05-21]. Retrieved: https://www.elastic.co/beats

[9] Logstash: Collect, Parse, Transform Logs. *Elastic* [online]. [cit. 2021-05-21]. Retrieved: https://www.elastic.co/logstash

[10] *Logstash - Introduction - Tutorialspoint* [online]. [cit. 2021-05-21]. Retrieved: https://www.tutorialspoint.com/logstash/logstash_introduction.htm

[11] *Kibana: Explore, Visualize, Discover Data | Elastic* [online]. [cit. 2021-05-21]. Retrieved: https://www.elastic.co/kibana

[12] Enterprise Support. *rsyslog* [online]. [cit. 2021-05-21]. Retrieved: https://www.rsyslog.com/professional-services/enterprise-support/

[13] *Debian Reference* [online]. [cit. 2021-05-21]. Retrieved: https://www.debian.org/doc/manuals/debian-reference/

[14] *Why you should use Docker and containers | InfoWorld* [online]. [cit. 2021-05-21]. Retrieved: https://www.infoworld.com/article/3310941/why-you-should-use-docker-and-containers.html&quot

[15] VAUGHAN-NICHOLS, Steven J. What is Docker and why is it so darn popular? *ZDNet* [online]. [cit. 2021-05-21]. Retrieved: https://www.zdnet.com/article/what-is-docker-and-why-is-it-so-darn-popular/

[16] AGARWAL, Nitin. Docker Container's Filesystem Demystified. *Medium* [online]. 30. January 2017 [cit. 2021-05-21]. Retrieved: https://medium.com/@BeNitinAgarwal/docker-containers-filesystem-demystified-b6ed8112a04a

[17] Elasticsearch from the Top Down. *Elastic Blog* [online]. 15. October 2014 [cit. 2021-05-21]. Retrieved: https://www.elastic.co/blog/found-elasticsearch-top-down

[18] VAN DONGEN, B F. A Meta Model for Process Mining Data. nedatováno, 12.

[19] RSyslog Documentation. *rsyslog* [online]. [cit. 2021-05-21]. Retrieved: https://www.rsyslog.com/doc/

[20] *OpenSSH: Security* [online]. [cit. 2021-05-21]. Retrieved: https://www.openssh.com/security.html

[21] *How an SSH tunnel can bypass firewalls, add Encryption to application protocols, and help access services remotely.* [online]. [cit. 2021-05-21]. Retrieved: https://www.ssh.com/academy/ssh/tunneling