



Enhancing Centralised Log Management for Distributed WordPress Sites with ELK Stack

An applied research project presented in partial fulfilment of the requirements

for the degree of

Master of Information Technology

at Whitireia/WelTec, Wellington, New Zealand

Jigang Guo

2025

ABSTRACT

Maintaining distributed WordPress websites across different servers can be a demanding task, especially for developers or system admins managing them alone. Relying on manual methods to check for security issues, track how resources are being used, or review log files often requires a lot of time and leaves room for mistakes. This project investigates using the ELK stack - Elasticsearch, Logstash, and Kibana - as a centralised solution to simplify these tasks. The goal is to see if ELK can make it easier to monitor and manage multiple WordPress setups by offering real-time data and automatic alerts.

The study centres on three main areas: spotting security threats, keeping an eye on server performance and usage, and interpreting meaningful application logs. I will compare ELK's performance to that of traditional, manual logging practices, focusing on efficiency, accuracy, and system safety. By doing so, I aim to understand whether ELK truly offers practical improvements. The results should provide helpful suggestions for WordPress users or administrators who are considering choosing modern monitoring tools to supervise websites. Ideally, the study will show how ELK used as a centralised log system can ease maintenance tasks, reduce the workload on administrators, and make WordPress sites more secure and stable.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Clement Swarnappa, for his invaluable guidance, support, and encouragement throughout the course of this research. His insightful feedback and patient instruction have greatly contributed to the development and completion of this project.

I would also like to thank my lecturers and classmates at Whitireia and WelTec for their continuous support and inspiration during my studies.

Finally, I am deeply grateful to my family for their unwavering love and encouragement, which have been a source of strength throughout my academic journey.

TABLE OF CONTENTS

ABSTRACT	ii
ACKNOWLEDGEMENTS.....	iii
LIST OF TABLES.....	vi
LIST OF FIGURES	vii
LIST OF ABBREVIATIONS.....	xi
CHAPTER 1 INTRODUCTION	1
1.1 General Background	1
1.2 Significance of the Study	2
1.3 Scope of Study	2
1.4 Research Problem	2
1.5 Research Questions.....	3
1.6 Research Objectives.....	3
1.7 Thesis Organisation.....	4
CHAPTER 2 LITERATURE REVIEW	6
2.1 Importance of Log Data Analysis	6
2.2 Centralised Log Management Overview	7
2.3 The ELK Stack in Log Management	7
2.2.1 Beats Module.....	8
2.2.2 Structured vs. Unstructured Data.....	10
2.2.3 Elasticsearch Module.....	11
2.2.4 Logstash Module	13
2.2.5 Kibana Module	13
2.2.6 Application of ELK Stack	14
2.4 Security Monitoring through Log Analysis	15
2.5 Resource Utilisation Monitoring.....	15
2.6 Overview of WordPress.....	16
2.7 WordPress Management Challenges	27
2.8 Existing Solutions for WordPress Monitoring.....	27
2.9 Gaps in the Current Research.....	28
2.10 Summary	28
CHAPTER 3 METHODOLOGY	28
3.1 Introduction.....	28
3.2 Design Science Research Overview	29
3.3 Problem Identification Stage.....	29
3.4 Solution Objectives Stage	30
3.5 Design and Development Stage	31
3.5.1 Architecture Overview.....	32
3.5.2 Artefact Configuration Overview.....	33
3.5.2.1 WordPress and MySQL Services	34
3.5.2.2 Reverse Proxy with Nginx	35
3.5.2.3 Log Collection: Filebeat.....	37
3.5.2.4 Resource Monitoring: Metricbeat.....	39
3.5.2.5 Centralised Logging Stack: Elasticsearch, Logstash, Kibana	39
3.5.3 Log Collection Pipeline	40

3.5.3.1	Filebeat Configuration.....	40
3.5.3.2	Logstash Configuration	42
3.5.4	Custom Logging Extensions.....	44
3.5.4.1	Custom Plugin: Slow Query Simulator	44
3.5.4.2	Custom Plugin: User Activity Logger.....	45
3.5.5	Kibana Configuration	46
3.5.5.1	Index Creation in Kibana.....	46
3.5.5.2	Alerts Rule Configuration	47
3.5.5.3	Dashboard Creation in Kibana	49
3.5.6	Brute-Force Detection Script for WordPress Login Attempts.....	54
3.5.7	User Behaviour Log Analysis Script	55
3.6	Evaluation Stage	56
3.6.1	Brute-Force Attack Detection and Alerting.....	57
3.6.2	System Performance and Slow Query Monitoring.....	57
3.6.3	User Behaviour Analysis	57
3.6.4	Summary.....	58
CHAPTER 4	EXPERIMENTAL RESULTS AND DISCUSSION	58
4.1	Introduction to Experimental Setup	58
4.1.1	Experimentation on Security Monitoring	59
4.1.1.1	Discussion	60
4.1.2	Experimentation on Performance Monitoring.....	61
4.1.2.1	MySQL Slow Query monitoring.....	64
4.1.2.2	Discussion	65
4.1.3	Experimentation on Decision-Making Support Using Custom User Activity Logs	66
4.1.3.1	Discussion	68
4.2	Summary.....	69
CHAPTER 5	CONCLUSION AND FUTURE WORK	69
5.1	Conclusion	69
5.2	Recommendations and Future Work.....	70
5.3	Limitations	70
REFERENCES	71
APPENDICES	74

LIST OF TABLES

<u>Table No.</u>	<u>Page No.</u>
Table 1:Beats library from the Elastic.....	9

LIST OF FIGURES

<u>Figure No.</u>	<u>Page No.</u>
Figure 1: ELK workflow pipeline.....	8
Figure 2: Comparison between Structured and Unstructured Data	11
Figure 3: How does Elasticsearch works?	12
Figure 4: WordPress Block Editor.....	17
Figure 5: Popular WordPress plugins	18
Figure 6: Popular WordPress themes	19
Figure 7: Official website of Taylor Swift.....	21
Figure 8: Official website of Disney Books	22
Figure 9: Official website of Sony Music	23
Figure 10: The official website of Time magazine	25
Figure 11 System Architecture	33
Figure 12: WordPress1 Service in Docker Compose.....	34
Figure 13: WordPress2 Service in Docker Compose.....	35
Figure 14: MySQL 1 Service in Docker Compose	35
Figure 15:MySQL2 Service in Docker Compose	35

Figure 16: Nginx Service in Docker Compose.....	36
Figure 17: Nginx Configuration	37
Figure 18: Filebeat1 Service in Docker Compose	38
Figure 19: Filebeat2 Service in Docker Compose	38
Figure 20: Filebeat-kibana Service in Docker Compose	38
Figure 21: Metricbeat Services in Docker Compose	39
Figure 22: Elasticsearch Service in Docker Compose	40
Figure 23: Filebeat 1 Configuration.....	41
Figure 24: Filebeat 2 Configuration.....	42
Figure 25: Logstash Configuration	44
Figure 26:Slow Query Simulator Plugin.....	45
Figure 27: User Activity Log Plugin.....	46
Figure 28: Index Patterns Created in Kibana.....	47
Figure 29: Alert Rule Created in Kibana	48
Figure 30: Alert Rule Configuration.....	49
Figure 31: Count of Visitors Created in Dashboard.....	50
Figure 32: Count of Visitors Configuration in Dashboard	50

Figure 33: User Duration Time Created in Dashboard	51
Figure 34: User Duration Time Vertical Axis Configuration in Dashboard.....	51
Figure 35: User Duration Time Horizontal Axis Configuration in Dashboard ...	52
Figure 36: Count of Users Scroll Down 50% Created in Dashboard	52
Figure 37: Count of Users Scroll Down 50% Configuration in Dashboard.....	53
Figure 38: Users Triggered Click Event Times Created in Dashboard.....	53
Figure 39: Users Triggered Click Event Times Configuration in Dashboard	54
Figure 40: Shell Script to Calculate Brute-Force Times	55
Figure 41: Shell Script to Analysis User Behaviours.....	56
Figure 42: User Login Records in Apache Log.....	59
Figure 43: Result of User Login Times by Shell Script.....	59
Figure 44: User Login Logs Appeared in Kibana.....	60
Figure 45: Alert Message Stored in Kibana Log.....	60
Figure 46: Alert Email Received From Kibana Log.....	60
Figure 47: Metricbeat Containers Running Status.....	61
Figure 48: Metrics Collected By Metricbeats in Dashboard	61
Figure 49: Metrics Collected by Metricbeat 1.....	62

Figure 50: Metrics Collected by Metricbeat 2.....	62
Figure 51: Logs Collected by Metricbeat 1 Shown in Kibana	63
Figure 52: Logs Collected by Metricbeat 2 Shown in Kibana	63
Figure 53: Metrics Shown on WordPress 1 Server by HTOP	63
Figure 54: Metrics Shown on WordPress 2 Server by HTOP	64
Figure 55: Slow SQL Query Result.....	64
Figure 56: Slow SQL Query Raw Logs.....	65
Figure 57: Slow SQL Query Logs Shown in Kibana	65
Figure 58: Alert Message of Slow SQL Query Via Email Notification.....	65
Figure 59: An E-Commerce Store Built On Kadence Theme.....	67
Figure 60: User Behaviour Raw Logs	67
Figure 61: User Behaviour Stats Shown in Dashboard.....	68
Figure 62: User Behaviour Stats Calculated By Shell Script.....	68
Figure 63: User Behaviour Logs Shown in Kibana	68

LIST OF ABBREVIATIONS

Centralised Log Management	CLM
Elasticsearch, Logstash, Kibana	ELK
Content Management System	CMS
Online Analytical Processing	OLAP
Advanced Persistent Threats	APT
Security Information and Event Management	SIEM

CHAPTER 1 INTRODUCTION

1.1 General Background

In recent years, the popularity of WordPress has surged, making it one of the most widely used content management systems (CMS) globally. Many developers and website administrators manage multiple WordPress sites simultaneously, especially within small to medium-sized enterprises (Toddplex, 2016). Due to its user-friendly interface and rich ecosystem of plugins, many non-technical users feel confident managing their sites (Murphy et al., 2021). However, this ease of use can lead to overconfidence, resulting in risky actions such as installing insecure plugins or modifying unfamiliar settings, potentially compromising the website's security and performance. Even minor issues can cause user dissatisfaction or financial losses (Sunil et al., 2023). From a developer's perspective, it becomes crucial to establish mechanisms for quickly identifying and resolving system anomalies or bugs introduced by non-technical users. Log tracking and analysis are key techniques in this regard, enabling early detection of errors or warnings and providing actionable insights for system optimisation and user experience improvements. Most industries maintain real-time alert tools to ensure system safety and stability, and make this a top priority (He et al., 2022). However, log management in practice presents numerous challenges, particularly in industrial contexts. These include the high volume and complexity of log data, heterogeneous log formats, and difficulties in correlating and understanding log content. System maintenance, along with troubleshooting, becomes excessively complex when key data is hidden deep within extensive log records. Finding the specific reason behind an issue within such datasets proves both time-consuming and frustrating. The practice of manual monitoring through traditional methods tends to be inefficient and laborious, and the complexity of modern IT systems makes manual log interpretation increasingly impossible. These approaches create space for human mistakes and

oversight which raises the probability of security breaches, resource exhaustion and outdated software component usage. Many standard logging tools fail to effectively display complex system data through clear visual presentations (Sunil et al., 2023). A centralised log management system serves as the main solution to address the problems of log monitoring. The ELK stack represents the most widely used solution because it consists of Elasticsearch, Logstash and Kibana as an open-source suite (Ahmed et al., 2020). The ELK system provides real-time data analysis through interactive dashboards that display visualised metrics which makes it valuable for understanding system health and user behaviours. This research evaluates ELK stack implementation in distributed WordPress networks to assess its effectiveness for website monitoring as well as fault detection and user behaviour analysis.

1.2 Significance of the Study

This research aims to construct a project that decreases administrative and developer workloads through ELK stack implementation to improve security issue detection, server resource and performance monitoring, and application log utilisation.

1.3 Scope of Study

The research examines log data management between WordPress websites using the ELK stack while concentrating on security vulnerability detection, system resource monitoring, and application logs tracking. The study does not explore content management systems outside of WordPress or WordPress installations with extensive custom modifications. The study will use a comparative method to evaluate ELK performance against standard manual log monitoring practices to demonstrate specific system management and efficiency benefits.

1.4 Research Problem

Maintaining multiple WordPress websites distributed across different servers or environments presents a range of operational challenges. Developers and website administrators often face difficulties in identifying security risks, analysing performance metrics, managing system

resources, and analysing user behaviours through application logs. Manual monitoring techniques are not only labour-intensive but also prone to oversight, particularly as the number grows. This study investigates whether the ELK stack - comprising Elasticsearch, Logstash, and Kibana - can provide a more efficient and reliable approach for addressing these recurring challenges. The focus is on assessing how ELK can support the centralised management and real-time monitoring of multiple WordPress instances.

1.5 Research Questions

- In what ways does the ELK stack improve the detection of security threats across multiple WordPress sites when compared with the traditional manual monitoring method?
- To what extent does the ELK stack enhance visibility of system resource usage in distributed WordPress environments?
- How effectively can the ELK stack assist in analysing user behaviour to support data-driven decision-making?

1.6 Research Objectives

The primary objectives of this research are as follows:

To investigate how the Elastic Stack (Elasticsearch, Logstash, and Kibana) can be applied to monitor and manage distributed WordPress environments.

To design and implement a centralised logging system that collects and analyses logs from multiple WordPress websites.

To evaluate the effectiveness of ELK Stack in detecting potential security vulnerabilities, resource usage, and application logs analysis in a distributed environment.

To conduct experiments that simulate different usage and attack scenarios, and measure the system's performance, scalability, and responsiveness.

To propose recommendations for WordPress developers and administrators on how to integrate ELK Stack into their systems to enhance operational efficiency and security monitoring.

1.7 Thesis Organisation

This thesis is organised into five chapters; each designed to systematically address the research problem and contribute to the achievement of the research objectives.

- **Chapter 1 – Introduction**

This chapter introduces the research topic, outlining the background, problem statement, research significance, objectives, and the scope of the study. This establishes the foundation for the subsequent investigation into centralized logging and monitoring systems for distributed WordPress environments.

- **Chapter 2 – Literature Review**

This chapter critically reviews existing studies related to the Elastic Stack components - Elasticsearch, Logstash, and Kibana - as well as their deployment in distributed system monitoring. Additionally, it examines previous research on WordPress performance, security monitoring techniques, and distributed logging challenges. Gaps in the current knowledge are identified to position this research within the broader academic context.

- **Chapter 3 – Research Methodology**

This chapter details the methodological approach adopted in this study. It describes the design of the experimental environment, the configuration of the ELK Stack for log collection and analysis, and the techniques used to simulate distributed WordPress operations. Methods for evaluating performance, anomaly detection, and scalability are also articulated.

- **Chapter 4 – Results and Discussion**

This chapter presents the empirical findings derived from the experiments. It analyses system performance metrics, evaluates the effectiveness of the ELK Stack in detecting anomalies and

security events, and discusses the scalability of the proposed solution. The results are critically compared with existing benchmarks and research objectives.

- **Chapter 5 – Conclusion and Future Work**

This chapter presents the main research findings and contributions. The study encountered specific challenges, which are discussed in this chapter, together with suggested directions for future research that focus on improving centralised monitoring systems for extensive distributed WordPress networks.

CHAPTER 2 LITERATURE REVIEW

2.1 Importance of Log Data Analysis

System logs serve as event records which document multiple system occurrences. System The system generates events whenever users initiate or terminate operations and applications start or terminate, and software gets installed or uninstalled and users modify the system clock and finish authentication protocols. System administrators receive security breach alerts through logs which track all unauthorized access attempts (Ahmed et al., 2020). System component behaviour becomes most accessible for analysis through monitoring its operational activities. System runtime logs enable developers and administrators to detect bugs, errors, and abnormal outputs while simultaneously helping to detect security threats and possible intrusions (Svacina et al., 2020). The output of logging statements creates unstructured printed text which results in time-ordered text-based data collection.

A log message contains several items, which include system variables and parameters such as hostname and username, together with message level (INFO/DEBUG/ERROR), IP address and other items that present semantic information through plain text words (Wang, 2023). Logs have been widely adopted in software system development and maintenance. In the IT industry, it is a common practice to record detailed software runtime information into logs (Zhu et al., 2023).

According to Wang et al. (2020), applying web log mining technology to the development of e-commerce systems not only enhances user experience and supports personalised recommendations, but also assists companies in understanding customer intent, identifying potential users, improving website infrastructure, and advancing the overall e-commerce industry. For effective comparison across different log files, log analysis systems must be capable of parsing content within the specific context in which it was generated. Overall, log

analysis contributes not only to better business decision-making but also to improved service quality.

2.2 Centralised Log Management Overview

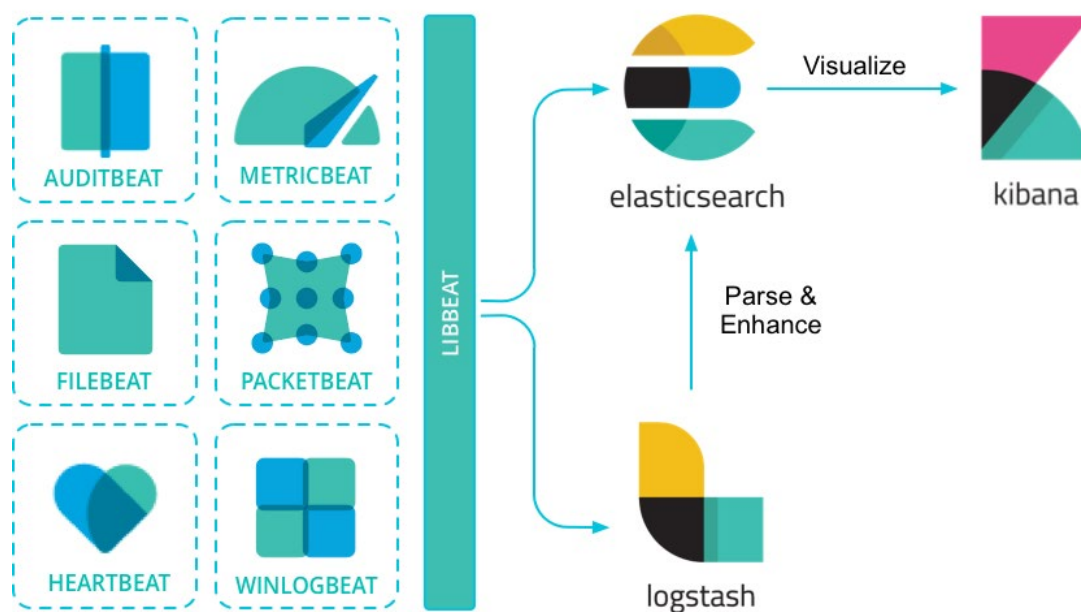
In current IT systems especially those with distributed systems of many servers and services, centralised log management (CLM) has become a key strategy. Kent (2018) points out that when log data is brought together in one location, administrators can better understand system-wide behaviour. The system provides administrators with enhanced visibility which enhances their capabilities for problem debugging system protection and performance evaluation. Through this system administrators gain access to a unified interface that enables them to search, filter and view logs. The current method of log management through one interface stands in sharp contrast to traditional manual approaches which required server-by-server log searches. The traditional approach to log management proves inefficient while simultaneously raising the chances of errors and oversight in complex system deployments.

2.3 The ELK Stack in Log Management

The ELK Stack, consisting of Beats, Elasticsearch, Logstash and Kibana, serves as a popular open-source framework for implementing CLM. Elasticsearch operates as the central storage system, which also functions as a search engine for log data. Logstash serves as the component that gathers logs before transforming them, while Kibana enables users to visualise metrics and patterns through real-time system activity monitoring. Beats serve as a lightweight data shipping system that collects logs and metrics from different sources before sending them to Logstash or Elasticsearch for additional processing. The ELK workflow pipeline functions as depicted in Figure 1 when it operates. The Beats library enables installation on target servers to send log data directly to Elasticsearch or Logstash for data processing. Elasticsearch handles data indexing and storage operations while Kibana enables users to execute queries and create visualisations. Instead of accessing individual servers to diagnose issues, administrators can

use Kibana's dashboards to search and analyse logs across multiple systems at once. This approach not only saves time but also enhances accuracy by enabling the correlation of events from different machines within the same timeframe, helping to uncover underlying issues that may span across services or servers.

Figure 1: ELK workflow pipeline



Note. The figure illustrates the ELK workflow pipeline. From *What are Beats?*, by Elasticsearch B.V., 2025 (<https://www.elastic.co/guide/en/beats/libbeat/current/beats-reference.html>). Copyright 2025 by Elasticsearch B.V.

2.2.1 Beats Module

The Elastic Stack includes lightweight data collection tools called Beats which efficiently gathers data from log files network traffic, and system metrics before sending it to Logstash or Elasticsearch for additional processing and analysis. Beats consist of multiple modules, each designed for a specific type of data source, and are primarily used for forwarding and centralising log data. Notably, Beats only collects data without performing any processing,

making them highly efficient. One of their key advantages is their low CPU and memory consumption, which ensures minimal impact on business servers (Xu et al., 2024).

For example, Filebeat is a lightweight log shipper that reads data from local or remote log files and forwards it to Logstash or directly to Elasticsearch. Other tools include Metricbeat, which collects system and service metrics, and Packetbeat, which captures network traffic data, Etc.

Table 1: Beats library from the Elastic

Log Type	Log Module	Beat framework
Audit data	Auditbeat	Collects audit framework data and monitors file integrity
Log files	Filebeat	Ships log files from various sources
Service availability	Heartbeat	Checks uptime and monitors service availability
System metrics	Metricbeat	Collects CPU, memory, and disk usage data
Network traffic	Packetbeat	Analyses network packets for performance and troubleshooting

Windows event logs	Winlogbeat	Winlogbeat - Lightweight shipper for Windows event logs
--------------------	------------	---

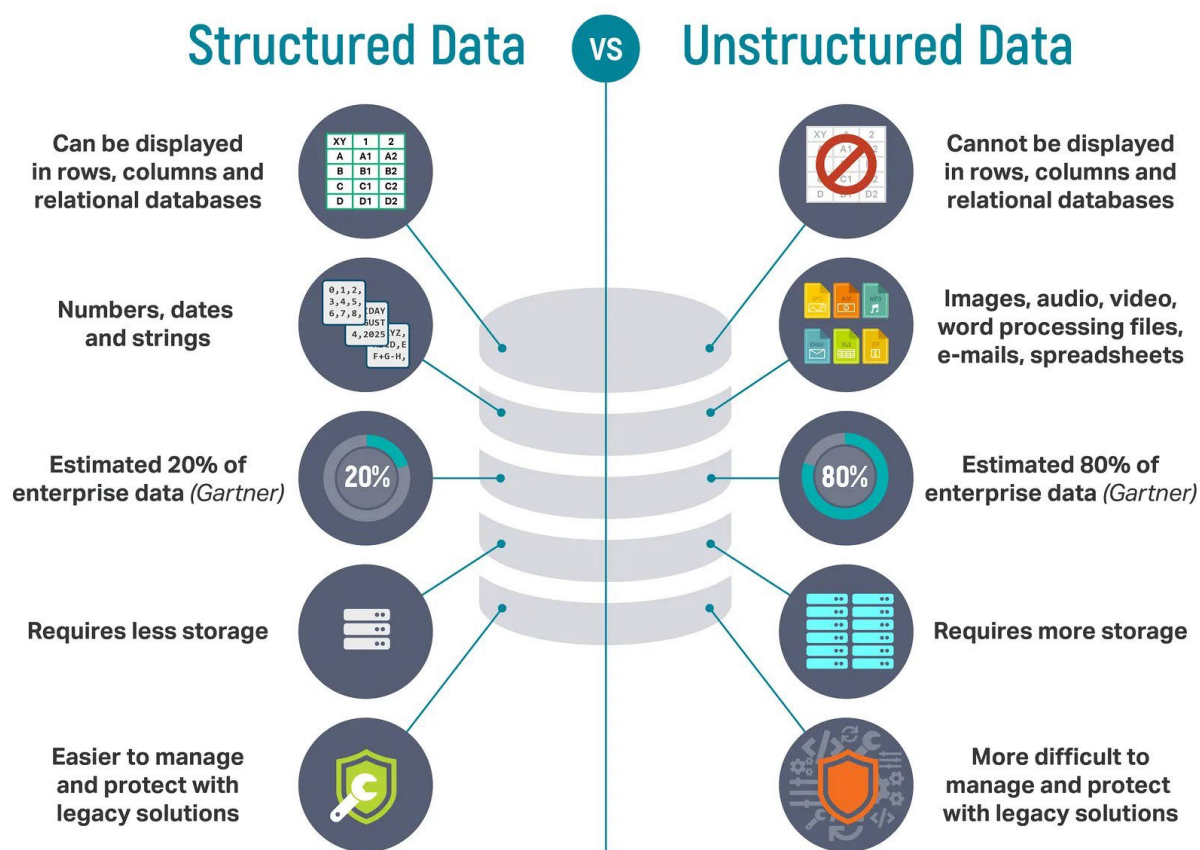
Note. Adapted from <https://www.elastic.co/guide/en/beats/libbeat/current/beats-reference.html>. Copyright 2025. Elasticsearch B.V.

2.2.2 Structured vs. Unstructured Data

In modern IT systems, data is generally categorised as either structured or unstructured. Structured data refers to information that adheres to a predefined data model and is often stored in relational databases such as MySQL or Oracle. It includes data types like names, dates, or numerical values that are easily searchable using SQL. In real life, the objects we search for are not always relational data. The goal of a search is to quickly locate the information that is most relevant to a user's needs. Given that both structured and unstructured data coexist in modern systems, the ability to accurately and efficiently retrieve information from both types has become increasingly important.

In contrast, unstructured data lacks a consistent format and cannot be easily stored in traditional relational databases. This category includes text documents, images, videos, and, most relevantly, log files. Because of its inconsistent nature, unstructured data poses challenges for storage, indexing, and analysis using traditional tools. Figure 2 shows the difference between structured data and unstructured data in storing and retrieving data.

Figure 2: Comparison between Structured and Unstructured Data



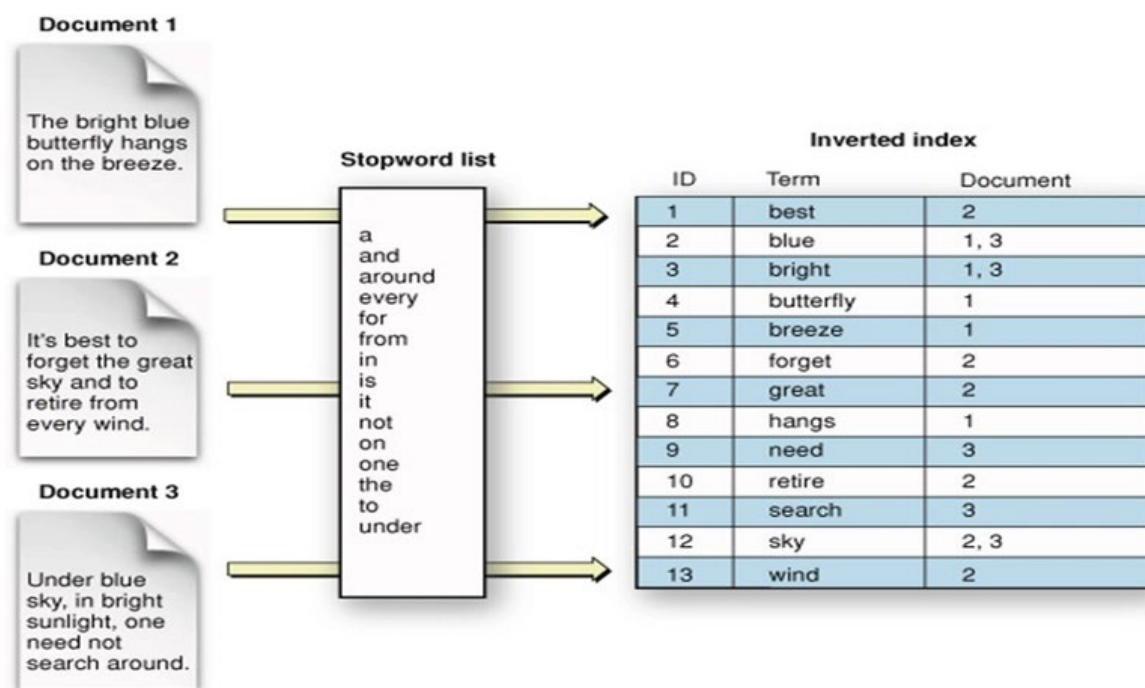
Note. The figure illustrates the comparison between structured and unstructured data. From *Structured vs Unstructured Data: An Overview*, by MongoDB, Inc., 2025 (<https://www.mongodb.com/resources/basics/unstructured-data/structured-vs-unstructured>). Copyright 2025 by MongoDB, Inc.

2.2.3 Elasticsearch Module

Elasticsearch is a powerful, distributed, open-source engine built on Apache Lucene that serves as the core component of the ELK Stack. It is optimised for search and analytics across large-scale datasets, functioning as both a scalable data store and a vector database. Elasticsearch allows for fast indexing, storage, querying, and real-time analysis of structured and unstructured data (Elastic Stack, 2025).

To address the challenges posed by unstructured data, Elasticsearch treats each piece of data as a document, which is then indexed for rapid querying. It applies an inverted index structure that maps each word or term to its location in the document set. During the indexing process, Elasticsearch tokenises the data, removes stop words (such as “a”, “the” or “and”), and stores relevant tokens in a highly searchable format. This makes it particularly well-suited for log analysis, full-text search, and real-time data exploration at scale.

Figure 3: How does Elasticsearch work?



Note. The figure illustrates the process of how Elasticsearch works. From *Elasticsearch, the advanced Search and Analytics Engine*, by [Kartikay Sawhney](#), 2019 (<https://medium.com/@kartikaysawhney1506/elasticsearch-the-advanced-search-and-analytics-engine-8ebbe7dd39f3>). Copyright 2019 by [Kartikay Sawhney](#).

As illustrated in Figure 3, when a term is queried in Documents 1, 2 and 3, Elasticsearch can quickly look up the term and locate it in the inverted index and retrieve the list of documents that contain that term. During the indexing process, stop words - such as "the", "is", and "and"

- are typically filtered out to optimise the efficiency and accuracy of the search, minimise the unnecessary matches, making it a powerful tool for handling large volumes of textual data and providing relevant search results in near real-time.

2.2.4 Logstash Module

Logstash is a free and open-source framework designed for collecting and parsing a large variety of both structured and unstructured data types. By unifying data collection and transformation, Logstash allows for real-time analytics and enables structured insights from diverse input formats (*Logstash Introduction | Logstash Reference [8.17] | Elastic, 2025*). As a plugin-based event forwarder, it ingests data from multiple sources, processes it, and then ships it to various destinations. These input plugins capture data from CSV files, TCP/UDP sockets, and HTTP APIs. Once the data is ingested, Logstash applies filter plugins to handle event processing, these filters transform the incoming data by removing unwanted elements, enriching the events with additional information, and preparing them for output to the designated targets (Bajer, 2017). In the data filtering process, the Grok filter stands as a popular choice because it uses predefined patterns to extract structured information from unorganised log entries. This approach enables users to efficiently parse and transform raw logs into a more readable and analysable format. Users can also achieve advanced data processing through the Ruby filter, which allows the embedding of custom Ruby scripts to implement complex transformations (Doan & Iuhasz, 2016). By default, Logstash sends the processed data to Elasticsearch as its primary output destination. However, it also supports transmitting data to other targets, such as CSV files, relational databases, and external platforms including Azure Machine Learning (Bajer, 2017).

2.2.5 Kibana Module

Kibana functions as the visual interface of the ELK Stack to enable users to interactively explore and display data through a wide array of visual formats - such as bar charts, tables,

heatmaps, and geographic maps. Kibana operates efficiently with large volumes of data through its web-based interface which enables users to create dynamic dashboards. Users can interact with real-time data through the platform without programming because queries operate using JSON-like syntax. Bhatnagar et al. (2020) state that this functionality provides users with better access to deep system insights which leads to improved decision-making.

2.2.6 Application of ELK Stack

ELK Stack showcases practical applications in various industrial fields. In their research on a Uday and Mamatha (2019) demonstrated how the system facilitates healthcare projects by aggregating logs from multiple sources to achieve real-time visualisation. This assists engineers in quick identifying which system issues require immediate attention based on their needs, locations and severity. Therefore, this process accelerates maintenance response, reduces system downtime. In a simulation of communication network system, ELK has been used to manage different real-time data streams. Through Kibana's visualization feature users can detect anomalies and monitor network traffic more effectively (Yang et al., 2022).

ELK serves as an effective cybersecurity solution to detect advanced persistent threats (APT) because it enables real-time log processing and anomaly detection and machine learning integration which allows security teams to detect threats rapidly (Stoleriu et al., 2021). Laingo Nantenaina and Zo (2024) conducted research which proved that OLAP technologies work well with ELK through Elasticsearch indexing to enhance the speed of large dataset classification and analysis tasks under multidimensional scenarios analysis.

The log management system for Docker environments proposed by Chen et al. (2020) combines ELK with Kafka to track logs in real-time and filter them structurally while enhancing DevOps workflows through visualization. Smith and Jones (2020) demonstrated Elasticsearch's real-time query functionality through Logstash data pipelines and Kibana dashboards. Lee et al. (2021) confirmed the system's ability to scale for deployments of both

small and large enterprises. The ELK solution provides organizations with a budget-friendly alternative to Splunk which enables robust security log analysis and visualization capabilities (Son & Kwon, 2017). The ELK applications provide outstanding value to systems that require quick decision-making insights.

2.4 Security Monitoring through Log Analysis

Security monitoring operations in the present era heavily depend on system log analysis as their fundamental operational component. System log analysis helps identify irregular system activities that could indicate criminal behaviours including unauthorized access attempts and malware infections. Log analysis according to Ahmad and Patel (2019) enables the discovery of security risks that were previously undetected. The ELK Stack generates automated alerts for recognized attack patterns and behavioural anomalies when its configuration is correct. The ELK system enhances its threat detection and response features through its integration with a Security Information and Event Management (SIEM) system (Yang et al., 2022).

The log analysis system developed by Ahmed et al. (2020) protected cloud applications. The system collected logs through Log4j before it implemented dual detection mechanisms to detect SQL injection attacks. The system performed log analysis through two independent methods which included both Bayesian classifier-based log categorization and visual pattern matching for security analysts. The research showed that application security requires instant analysis together with multiple detection approaches.

2.5 Resource Utilisation Monitoring

Resource consumption monitoring such as CPU, memory and disk storage usage helps prevent system bottlenecks and outages. The ELK Stack enables the collection and visualisation of such metrics, which allows administrators to detect unusual patterns or resource strains more easily. Brown (2020) highlighted that system health information exists within log data and ELK stack delivers exceptional value for distributed architectures because it unifies logs from

multiple servers that handle workload. The system enables better resource planning and system tuning through its ability to combine performance indicators from different nodes.

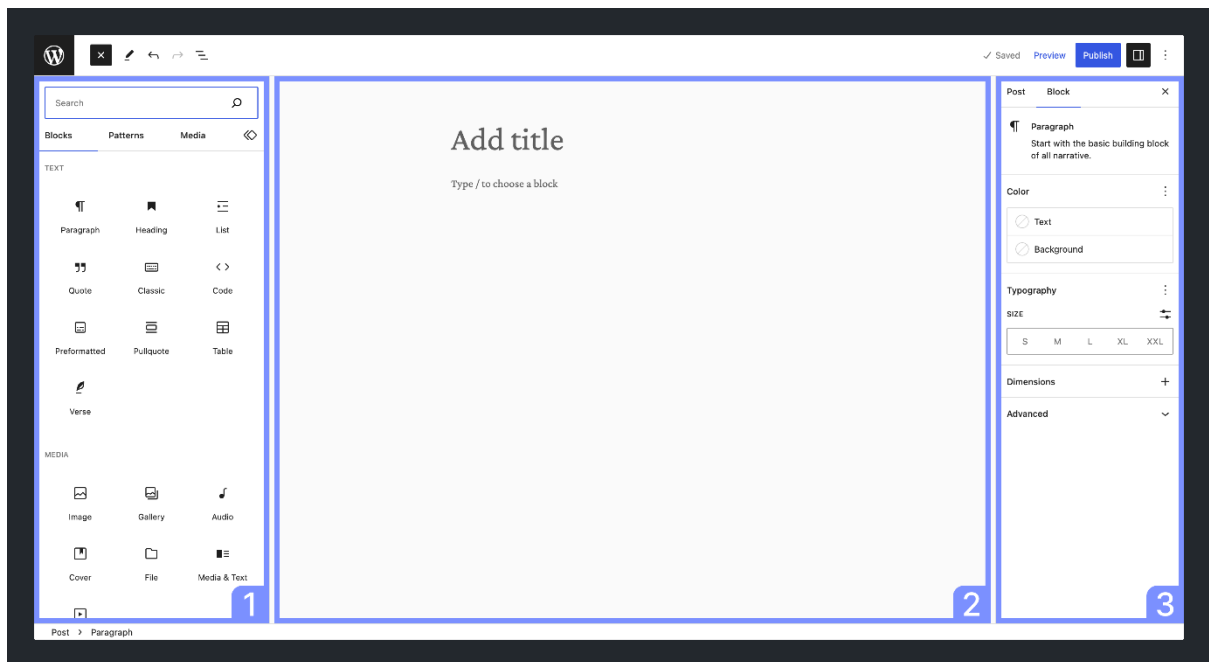
2.6 Overview of WordPress

WordPress launched in 2003 has evolved into a leading content management system (CMS). WordPress stands out for its user-friendly interface and broad plugin selection which enables users to create everything from basic blogs to advanced e-commerce websites. Murphy et al. (2021) stated that WordPress powers approximately 35% of internet content worldwide and is possibly the most used content management system, serving as the back end for hundreds of millions of websites and accounting for 60.3% of all content management systems in use. WordPress provides an intuitive content editing interface that allows users to create and publish content without needing to master complex programming languages (Achar, 2021). With its "Block Editor," users can quickly build page content through drag-and-drop operations, such as adding text, images, and videos, without writing HTML or CSS code. This simplified editing approach dramatically lowers the barriers to website creation, making it accessible for non-technical users to manage website content.

As shown in Figure 4, the WordPress Block Editor consists of several key components:

1. **inserter:** A panel for users to insert predefined blocks into the content canvas.
2. **Content canvas:** The content editor, which serves as the workspace where content is created and organised using blocks.
3. **Settings Panel:** A panel that allows users to configure the settings of a selected block or adjust the settings of the entire post.

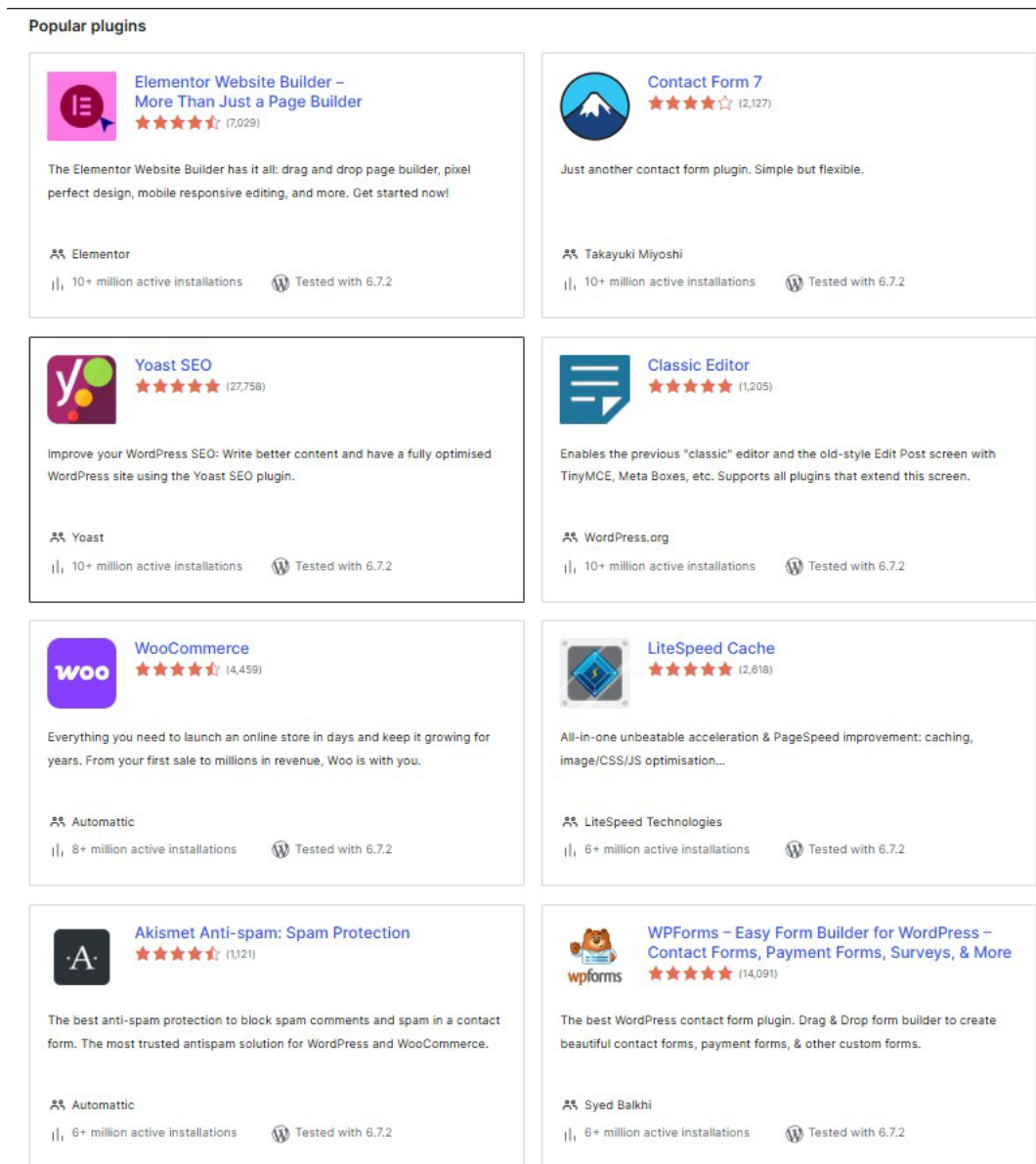
Figure 4: WordPress Block Editor



Note. The figure illustrates the primary elements of the Block Editor. From *Block Editor Handbook*, by WordPress.ORG, 2019 (<https://developer.wordpress.org/block-editor/>). in the public domain.

One of the significant advantages of WordPress is its plugin system, which allows users to add various website features, such as SEO optimisation, e-commerce, and social media integration. WordPress has an extensive plugin library, enabling users to install plugins as needed to extend the website's functionality. Each plugin provides its service based on user needs, such as enhancing search functionality, managing user permissions, and enabling electronic payments. Figure 5 illustrates some of the most popular WordPress plugins commonly used to expand a website's capabilities.

Figure 5: Popular WordPress plugins

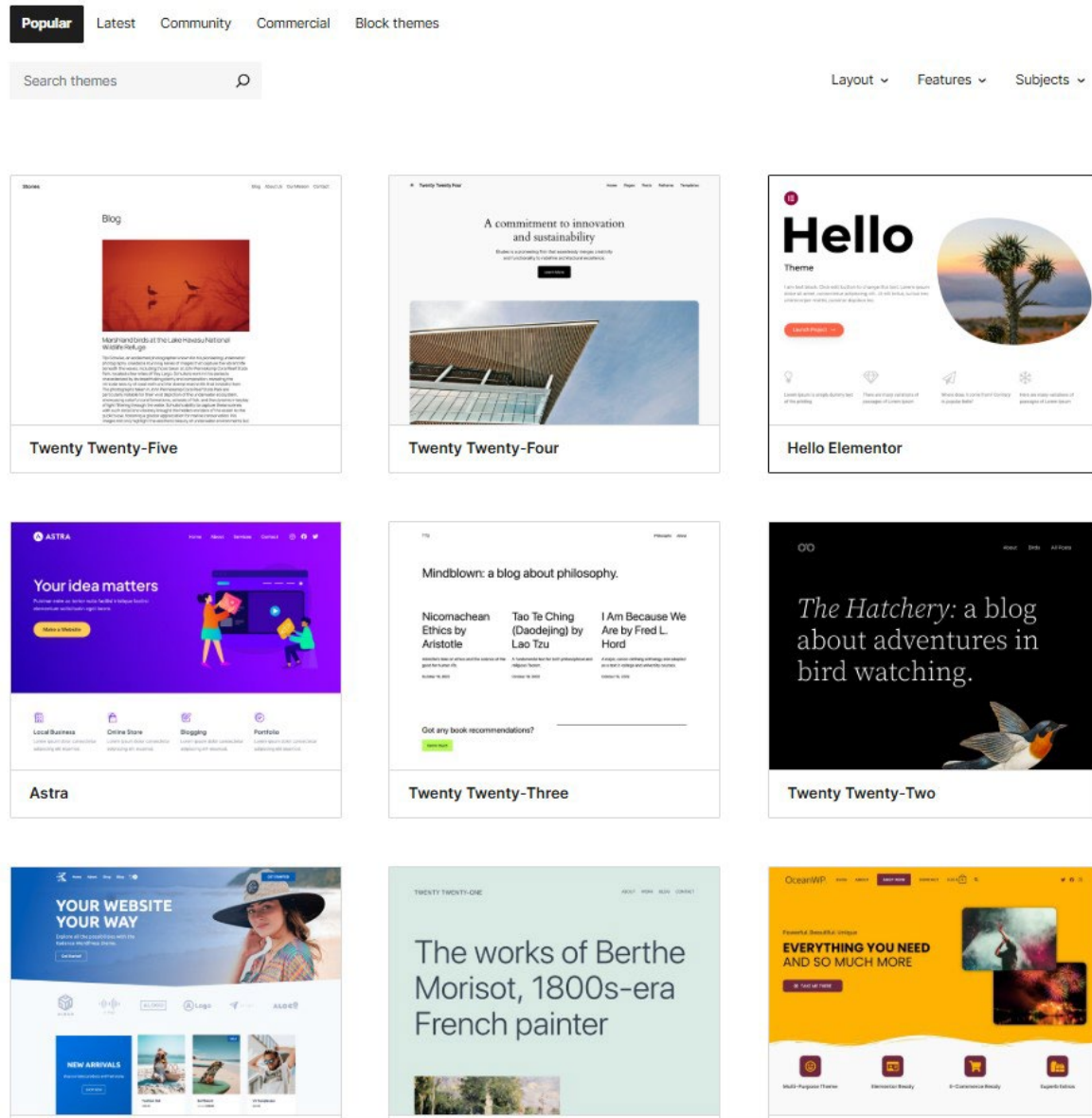


Note. The figure illustrates the popular plugins of WordPress. From *Plugins*, by WordPress.ORG, n.d. (<https://en-nz.wordpress.org/plugins/>). In the public domain.

WordPress also offers powerful theme customisation options, allowing users to tailor a website's appearance to match branding guidelines or meet specific functional requirements. Themes control the overall design, layout, and visual style of a site, providing both prebuilt templates and extensive customisation possibilities.

Figure 6 showcases some of the most popular WordPress themes widely adopted for creating visually appealing and professional websites.

Figure 6: Popular WordPress themes



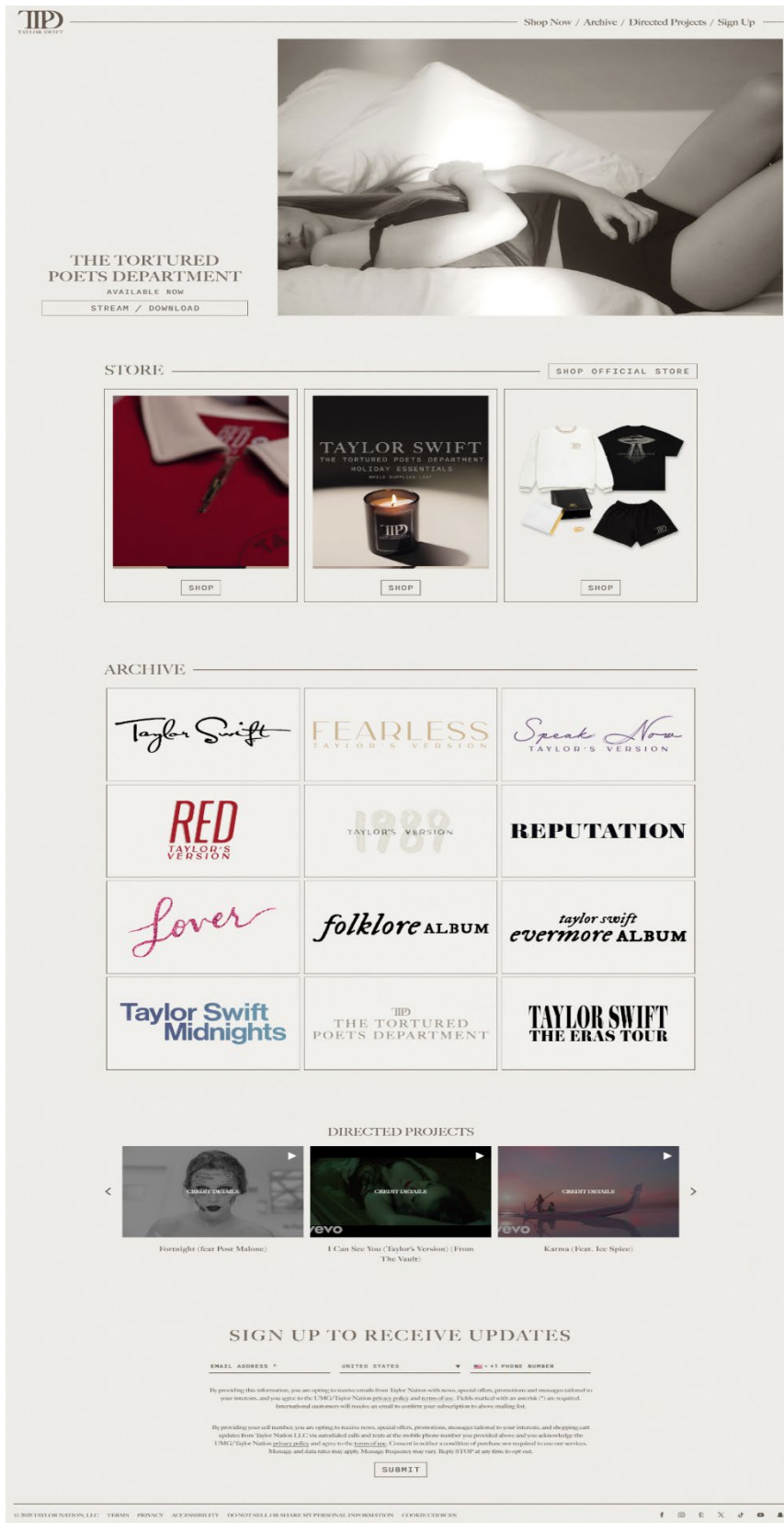
Note. The figure illustrates the popular themes of WordPress. From *Themes*, by

WordPress.ORG, n.d. (<https://en-nz.wordpress.org/themes/>). In the public domain.

The high flexibility of plugins and themes allows WordPress to be applied to nearly any type of website, whether it is a simple blog, a complex business website, or a feature-rich e-commerce platform. For example, Taylor Swift's official website (shown in Figure 7), Disney

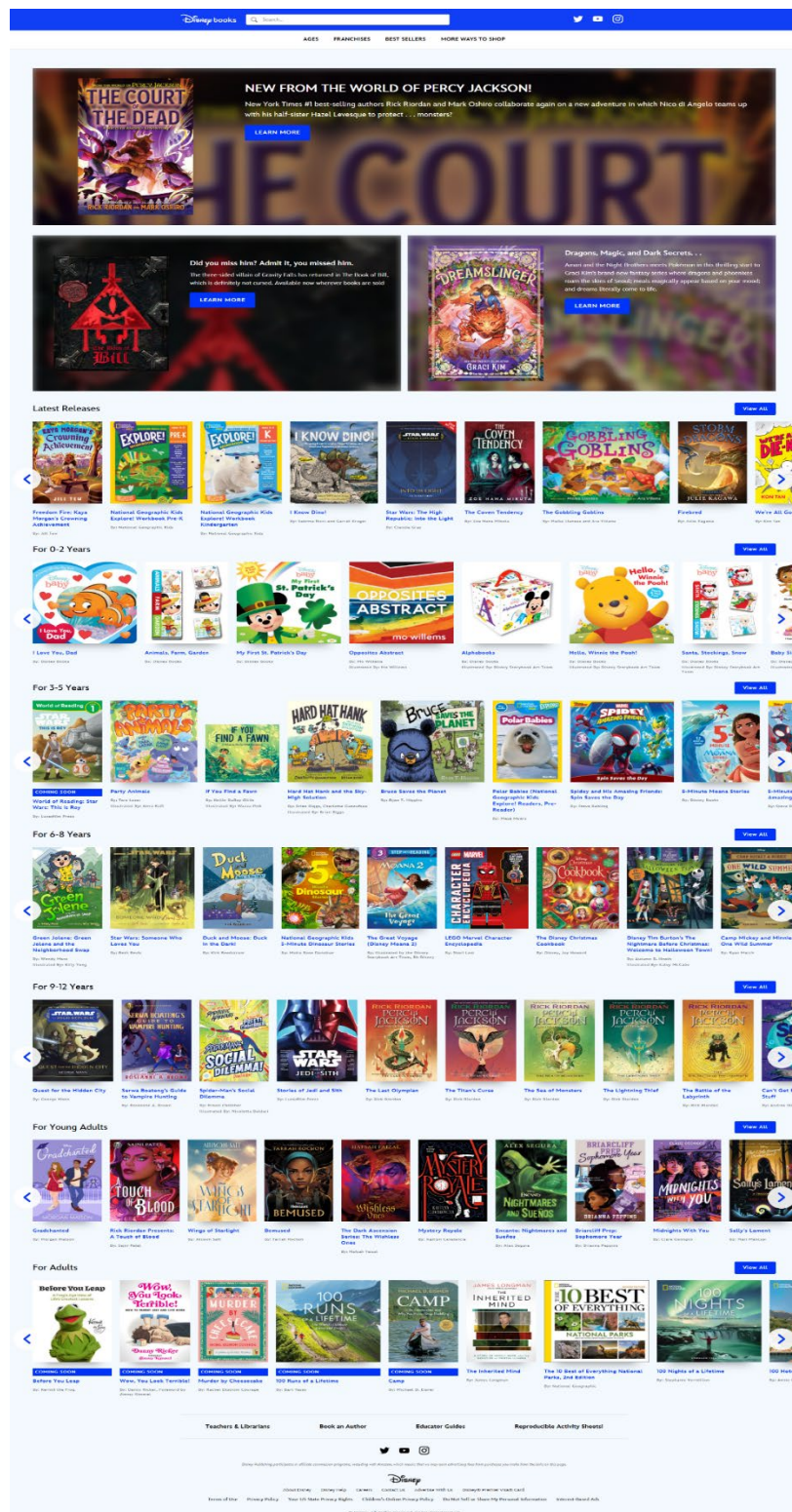
Books (Figure 8), Sony Music (Figure 9), and Time magazine (Figure 10) are all built on WordPress. These examples highlight WordPress's versatility. By leveraging a combination of plugins and themes, developers can customise websites to meet a wide range of needs, significantly enhancing flexibility and adaptability.

Figure 7: Official website of Taylor Swift



Note. The figure illustrates the official website of Taylor Swift. From *Taylor Swift*, by taylorswift.com, n.d. (<https://www.taylorswift.com/>). Copyright 2019 by Taylor Nation, LLC.

Figure 8: Official website of Disney Books



Note. The figure illustrates the official website of Disney Books. From *Disney*, by books.disney.com, n.d. (<https://books.disney.com/>). Copyright by Disney.

Figure 9: Official website of Sony Music



[About](#) [News](#) [Careers](#) [FAQ](#) [For Artists](#)

Red Clay Strays



Labels

ALAMO
RECORDS

ARISTA

AWAL



Epic

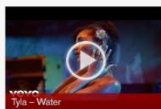
Featured Videos



LISA - NEW WOMAN feat. ...



Lil Nas X - J CHRIS ...



Tyla - Water

News

Sony Music Entertainment India and Tiger Baby Announce New Tiger Baby Records Joint Venture

Mar 27, 2025

Acclaimed Filmmakers and Creative Powerhouse Behind Coke Studio to Lead New Label 'City Sessions' Initiative to Support Artist Collaboration Mumbai, 27 March 2025 – Sony Music Entertainment India and Tiger Baby announced today the

[Read More »](#)

Sony Music Latin Iberia Announces New Structure in Brazil

Mar 27, 2025

Fernando Cabral de Mello Named CEO of Sony Music Entertainment Brazil Miami, FL (March 27, 2025) – Sony Music Latin Iberia today announced a new organizational structure for its operations in Brazil, which will now be led by Fernando Cabral de Mello

[Read More »](#)

First Details Revealed of Film in Conjunction With the Oasis Live '25 Tour

Mar 13, 2025

As this summer's wildly anticipated Oasis Live '25 tour approaches, Oasis can now confirm that a new film is being made in conjunction with the tour. The film will be created and produced by Steven Knight, the BAFTA®

[Read More »](#)

RCA Records Greater China Partners with Rising Chinese Indie Hip Hop Label IRIS Chengdu

Mar 10, 2025

Launch of five albums from five artists combines RCA China's expertise with indie label IRIS Chengdu's roster, bringing Chinese rap and hip hop music to new fans worldwide Shanghai, 10 March 2025 – RCA Records Greater China, a label di

[Read More »](#)

Top Latin Record Label Rancho Humilde, Sony Music Latin, and Sony's Columbia Pictures Join Forces to Release 'CLIK!'

Mar 04, 2025

A Groundbreaking Feature Film Defining the Global Impact and New Era of Mexican-American Culture Set August 15th, 2025 Theatrical Release Jimmy Humilde, CEO of Rancho Humilde, the powerhouse Mexican American recording label, is expanding his legacy

[Read More »](#)

[More News »](#)

Have a Question for Sony Music?

Visit our FAQ page for information regarding

- Our demo policy
- Royalties
- Employment & internship opportunities
- Press contacts & more

[View Our FAQs »](#)

[Back To Top](#)



[ROYALTY INFO](#)

[COPYRIGHT INFO](#)

Copyright © 2025 SONY MUSIC ENTERTAINMENT. All Rights Reserved.
Reggie Information | About Us | Pro Center | Copyright Information | Privacy Policy / Your Privacy Rights | How We Use Your Data | Do Not Sell/Share My Personal Information | Your California Privacy Rights | Terms & Conditions | AI Usage Terms | Send Us Feedback | Why Music Matters

Note. The figure illustrates the official website of Sony Music. From *Sony Music*, by sonymusic.com, n.d. (<https://www.sonymusic.com/>). Copyright by 2025 SONY MUSIC ENTERTAINMENT.

Figure 10: The official website of Time magazine

Note. The figure illustrates the official website of Time magazine. From *Time*, by time.com, n.d. (<https://time.com/>). Copyright by 2025 TIME USA, LLC.

2.7 WordPress Management Challenges

Although WordPress is a powerful platform, its widespread usage makes it a common target for hackers. WordPress's most common security risks are outdated software, insecure plugins and themes, insecure passwords, SQL injection, cross-site scripting (XSS), brute force, and file inclusion exploits. (Yi et al., 2024). All WordPress websites are susceptible to brute force attacks to enumerate users and their passwords with the WPSCAN tool (Shah & Ayoade, 2023). The WordPress community regularly releases security updates and patches to fix known vulnerabilities, and administrators must keep the core, plugins, and themes up to date to minimise security risks (Releases – WordPress News, 2025). Additionally, installing appropriate security plugins, implementing strong password policies, and regularly backing up website data are standard measures to protect WordPress website security.

2.8 Existing Solutions for WordPress Monitoring

Popular WordPress monitoring plugins like Wordfence and Jetpack focus mainly on security, uptime monitoring, and backup management. While these tools offer essential protective functions, they lack built-in support for comprehensive logging and log analysis (*Jetpack Vs Wordfence*, 2021). As Murphy (2021) identified, finding a free-tier WordPress plugin that effectively detects plugin vulnerabilities is impossible. Moreover, these solutions typically offer limited customisation and do not allow developers or administrators to define what log data should be collected, how it should be processed, or how alerts should be generated. This lack of flexibility and visibility makes them unsuitable for scenarios that require in-depth monitoring, especially across distributed WordPress environments.

2.9 Gaps in the Current Research

While the effectiveness of the ELK stack in general system monitoring has been well-documented, limited research specifically applies ELK to the context of distributed WordPress site management. Few studies address how ELK can streamline security vulnerability detection, resource usage monitoring, and user behaviour analysis based on application logs in the WordPress environment. This gap highlights the need for targeted research exploring the integration of ELK into WordPress maintenance workflows.

2.10 Summary

The existing literature establishes the importance of centralised log management and the capabilities of the ELK stack in enhancing system monitoring and security. However, specific applications of ELK in the management of distributed WordPress websites remain underexplored. This research aims to fill this gap by investigating how ELK can effectively support developers and administrators in maintaining multiple WordPress installations more efficiently and securely.

CHAPTER 3 METHODOLOGY

3.1 Introduction

The research follows Design Science Research (DSR) principles to develop and assess a centralised logging system for distributed WordPress environments which uses ELK Stack (Elasticsearch, Logstash, Kibana). The research follows DSR because it focuses on improving practical artefacts through design iterations to resolve real-world problems, such as effective log management across multiple WordPress sites.

WordPress's content management system is one of the most popular tools globally, with millions of websites under its management. The complexity of environment monitoring and maintenance grows significantly when deployments increase in scale, whether they involve

multiple servers or websites. System administrators face difficulties when working with scattered log data because it makes performance metric tracking, security issue detection, and user behaviour analysis inefficient and delayed. This research develops a central logging solution powered by the ELK Stack to solve these problems. The platform offers advanced capabilities for log aggregation and analysis, as well as visualisation that provides a scalable solution over basic shell-based log management approaches. The system works to increase transparency and operational efficiency while enabling data-driven choices by linking ELK to distributed WordPress systems.

The research demonstrates the artefact design, followed by implementation details and performance evaluation through experimental approaches. It evaluates ELK-based solution performance against traditional shell scripting methods to explain the benefits of usability and flexibility, along with actionable insights. The research aims to establish a framework that developers and administrators can use to manage distributed WordPress systems more effectively.

3.2 Design Science Research Overview

The research implements Hevner et al.'s (2004) Design Science Research (DSR) framework through its six essential activities, starting with problem identification, followed by solution objectives definition, and then design and development stages, demonstration, evaluation, and communication. The systematic development and evaluation of IT artifacts for real-world problem solutions follows the stages outlined in this framework. The following sections of this report correspond to each activity, starting with problem identification in Section 3.3.

3.3 Problem Identification Stage

A systematic literature review was performed to determine research gaps, which led to defining a specific problem statement. The focus was on the ELK (Elasticsearch, Logstash, Kibana) stack, distributed logging systems, and performance monitoring in WordPress environments.

The review covered the period from 2018 to 2024, using academic databases such as Google Scholar, IEEE Xplore, and ACM Digital Library. Search terms included combinations like “ELK stack,” “Elasticsearch logging,” “WordPress performance monitoring,” “centralised logging,” and “distributed logging system,” enhanced with Boolean operators to improve relevance.

Inclusion criteria prioritised peer-reviewed journal articles and conference papers in English that explored ELK implementations in web applications or CMS platforms like WordPress. Studies unrelated to distributed systems or outdated models were excluded.

The findings revealed that while the ELK stack is widely applied in enterprise-level logging systems, few studies address its application in distributed WordPress deployments, especially regarding illegal request detection, resource usage imbalance, and user behaviour analysis across multiple WordPress instances.

This literature gap highlights the need for a focused investigation into how the ELK stack can support real-time, centralised monitoring of multiple WordPress sites, forming the basis of the current study.

3.4 Solution Objectives Stage

The solution objectives stage defined the conditions required to address the research questions regarding the capabilities of the ELK (Elasticsearch, Logstash, Kibana) stack in managing distributed WordPress environments. These objectives were directly aligned with the core research aim: to explore how ELK can improve detection, visibility, and insight generation compared to traditional manual methods.

The primary objective was to design and develop a centralised logging solution based on the ELK stack to enhance the detection of security threats across multiple WordPress sites. This required system administrators to establish automated log collection procedures for distributed WordPress websites while setting up rule-based mechanism.

The project's second goal involved improving system resource monitoring capabilities through ELK pipeline integration of CPU information and memory metrics and disk I/O and database performance logs. The system enables real-time and historical analysis of site data which helps administrators perform performance tuning and capacity planning.

Another goal of this project involved using log information to evaluate WordPress website user activity through observations of login events and page visits as well as feature utilisation. The system aimed to convert raw log data into useful information through Kibana visual dashboards and queries to support site administrators in their data-based choices.

The objectives can be accomplished through developing an ELK system that links multiple WordPress sites for demonstrating log data collection and processing with visualisation capabilities for security, system performance and behavioural analysis. The evaluation will include simulated conditions to measure ELK performance in comparison to traditional manual monitoring approaches.

This artefact exists as a practical tool for WordPress system administrators while also acting as academic research regarding centralized logging and CMS performance monitoring systems.

3.5 Design and Development Stage

The ELK-based logging system with multiple WordPress sites integration occurs during the Design and Development phase. The artefact showcases ELK's ability to automate log collection while simultaneously enhancing threat detection and providing better resource visibility and supporting user behaviour analysis. This stage defines the architectural framework together with technology selection and deployment procedures to build a operational distributed WordPress prototype. System administrators require scalable designs with real-time capabilities and user-friendly interfaces.

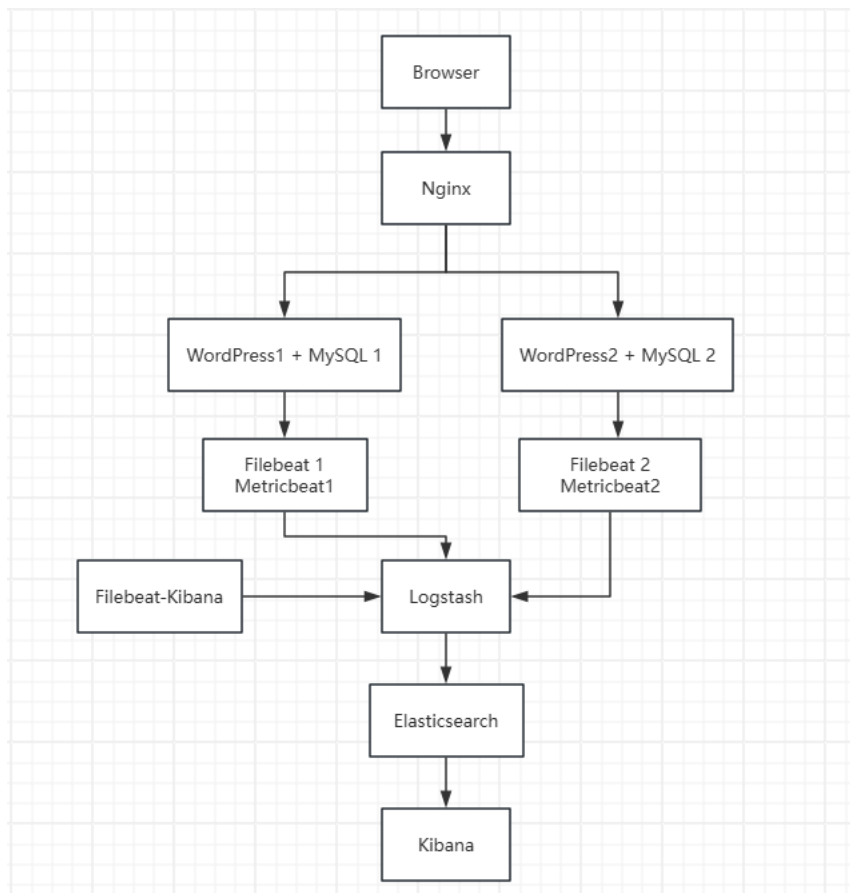
3.5.1 Architecture Overview

The system applies Docker's containerisation technology to build WordPress websites and ELK (Elasticsearch, Logstash, Kibana) logging framework.

The system consists of the following key components (see Figure 11):

- A nginx reverse proxy simulates a load balancer or gateway in real deployments.
- Two independent WordPress containers (WordPress1 and WordPress2), each paired with a dedicated MySQL instance (MySQL1 and MySQL2).
- Filebeat agents (Filebeat1 and Filebeat2) are configured per site to forward application-level logs (e.g., user activity and MySQL slow query logs) to Logstash.
- Metricbeat agents (Metricbeat1 and Metricbeat2) are deployed alongside each WordPress container to monitor container-level metrics such as CPU usage, memory consumption, and disk I/O.
- A central Logstash container, responsible for parsing and enriching log data before indexing it into Elasticsearch.
- A single-node Elasticsearch service that stores logs and metrics data, enabling full-text search and aggregations.
- A Kibana dashboard interface visualises system behaviour, analyses performance trends, and monitors potential security threats.

Figure 11: System Architecture



All components are orchestrated using Docker Compose and connected via a custom bridge network (`wp_network`) to facilitate inter-service communication. Log and configuration files are managed via mounted volumes, ensuring observability and traceability of the system's inner workings. Its modular, scalable design enables easy replication of additional WordPress sites and Beats agents, making it suitable for testing centralised logging scenarios in multi-site WordPress deployments.

3.5.2 Artefact Configuration Overview

This configuration file orchestrates multiple services, including two independent WordPress sites, their respective MySQL databases, Nginx reverse proxy, and various Elastic Stack components such as Filebeat, Metricbeat, Logstash, Elasticsearch, and Kibana. The system operates within a Docker bridge network to enable seamless inter-service communication.

3.5.2.1 WordPress and MySQL Services

The artefact simulates a distributed WordPress environment by deploying two separate WordPress instances, WordPress1 and WordPress2, each connected to a dedicated MySQL database container (MySQL1 and MySQL2). This design reflects a realistic multi-site scenario, allowing independent operation, logging, and monitoring for each WordPress instance (see Figure 12). In the configuration of WordPress 1, a custom volume is mounted at `wp-content/plugins/custom-logs/user-activity.log`, which allows a plugin to record user activity logs persistently. This log file later serves as input for Filebeat to forward to the ELK stack (see Figure 13). Meanwhile, MySQL1 is configured with the “slow-query-log” flag and “long-query-time=1” setting, enabling the capture of any SQL queries that exceed one second. These slow query logs are written to “`/var/log/mysql/slow.log`”, which is also mounted as a Docker volume for persistent access (see Figure 14). The second WordPress instance, WordPress2, follows a similar structure without custom logging. The database container, MySQL2, maintains the standard configuration and stores data through a dedicated volume (see Figure 15).

Figure 12: WordPress1 Service in Docker Compose

```
# WordPress 1
wordpress1:
  image: wordpress:latest
  restart: always
  depends_on:
    - mysql1
  environment:
    WORDPRESS_DB_HOST: mysql1:3306
    WORDPRESS_DB_USER: root
    WORDPRESS_DB_PASSWORD: root
    WORDPRESS_DB_NAME: wordpress1
  volumes:
    - ./wp1/wp-content:/var/www/html/wp-content
    - ./wp1/user-activity.log:/var/www/html/wp-content/plugins/custom-logs/user-activity.log
  networks:
    - wp_network
```

Figure 13: WordPress2 Service in Docker Compose

```
# WordPress 2
wordpress2:
  image: wordpress:latest
  restart: always
  depends_on:
    - mysql2
  environment:
    WORDPRESS_DB_HOST: mysql2:3306
    WORDPRESS_DB_USER: root
    WORDPRESS_DB_PASSWORD: root
    WORDPRESS_DB_NAME: wordpress2
  volumes:
    - ./wp2/wp-content:/var/www/html/wp-content
  networks:
    - wp_network
```

Figure 14: MySQL 1 Service in Docker Compose

```
# MySQL 1
mysql1:
  image: mysql:5.7
  restart: always
  environment:
    MYSQL_ROOT_PASSWORD: root
    MYSQL_DATABASE: wordpress1
  volumes:
    - ./mysql1_logs:/var/log/mysql
  command: --slow-query-log=1 --slow-query-log-file=/var/log/mysql/slow.log --long-query-time=1
  networks:
    - wp_network
```

Figure 15: MySQL2 Service in Docker Compose

```
# MySQL 2
mysql2:
  image: mysql:5.7
  restart: always
  environment:
    MYSQL_ROOT_PASSWORD: root
    MYSQL_DATABASE: wordpress2
  volumes:
    - mysql2_data:/var/lib/mysql
  networks:
    - wp_network
```

Together, these components form the foundational layer of the artefact's monitoring system, enabling later log collection and performance analysis.

3.5.2.2 Reverse Proxy with Nginx

A reverse proxy is configured using Nginx to enable unified access to all services (see Figure 16). The Nginx proxy container listens on port 80 and uses a mounted “nginx.conf” file to route requests based on domain names.

As shown in Figure 17, the configuration defines upstream servers for WordPress1, WordPress2, and Kibana. Requests to wordpress1.arp.com, wordpress2.arp.com, and elk.arp.com are forwarded to their respective containers. This reverse proxy mechanism plays a vital role in the artefact by mimicking a real-world infrastructure and allowing seamless monitoring of distributed WordPress instances and the centralised ELK interface.

Figure 16: Nginx Service in Docker Compose

```
nginx:
  image: nginx:latest
  container_name: nginx_proxy
  restart: always
  ports:
    - "80:80"
  volumes:
    - ./nginx.conf:/etc/nginx/nginx.conf:ro
  networks:
    - wp_network
```

Figure 17: Nginx Configuration

```
server {
    listen 80;
    server_name wordpress1.arp.com;
    location / {
        proxy_pass http://wordpress1;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}

server {
    listen 80;
    server_name wordpress2.arp.com;
    location / {
        proxy_pass http://wordpress2;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}

server {
    listen 80;
    server_name elk.arp.com;
    location / {
        proxy_pass http://kibana;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}
```

3.5.2.3 Log Collection: Filebeat

Three Filebeat containers were deployed to collect and ship logs from the WordPress environment to Logstash. Filebeat1 is configured to collect both user activity logs from WordPress 1 (user-activity.log) and slow query logs from MySQL 1 (/var/log/mysql/slow.log) (see Figure 18). The Filebeat2 instance is set up for WordPress 2 and reads configuration from a separate Filebeat2.yml file, ensuring independence in log parsing (see Figure 19). Finally, filebeat-kibana, shown in Figure 20, collects Docker-related logs for the Kibana interface, enabling monitoring of access and error events.

Figure 18: Filebeat1 Service in Docker Compose

```
# Filebeat for WordPress 1
filebeat1:
  image: elastic/filebeat:7.17.0
  user: root
  volumes:
    - /var/lib/docker/containers:/var/lib/docker/containers:ro
    - ./wp1/user-activity.log:/var/log/user-activity.log
    - ./mysql1_logs:/var/log/mysql
    - ./filebeat1.yml:/usr/share/filebeat/filebeat.yml
  depends_on:
    - wordpress1
    - elasticsearch
    - logstash
  restart: always
  command: ["--strict.perms=false"]
  networks:
    - wp_network
```

Figure 19: Filebeat2 Service in Docker Compose

```
# Filebeat for WordPress 2
filebeat2:
  image: elastic/filebeat:7.17.0
  user: root
  volumes:
    - /var/lib/docker/containers:/var/lib/docker/containers:ro
    - ./filebeat2.yml:/usr/share/filebeat/filebeat.yml
  depends_on:
    - wordpress2
    - elasticsearch
    - logstash #
  restart: always
  command: ["--strict.perms=false"]
  networks:
    - wp_network
```

Figure 20: Filebeat-kibana Service in Docker Compose

```
# Filebeat for Kibana
filebeat-kibana:
  image: elastic/filebeat:7.17.0
  user: root
  volumes:
    - /var/lib/docker/containers:/var/lib/docker/containers:ro
    - /var/run/docker.sock:/var/run/docker.sock
    - ./filebeat-kibana.yml:/usr/share/filebeat/filebeat.yml
  depends_on:
    - kibana
    - logstash
  command: ["--strict.perms=false"]
  networks:
    - wp_network
```

Each Filebeat instance uses a dedicated configuration file (e.g., filebeat1.yml) and is connected to the shared wp_network, allowing communication with WordPress, Elasticsearch, and Logstash services.

3.5.2.4 Resource Monitoring: Metricbeat

Two Metricbeat instances (Metricbeat1 and Metricbeat2) are deployed for each WordPress environment to monitor system-level metrics such as CPU, memory, and I/O usage (see Figure 21). These services also rely on mounted configuration files and forward metrics to the ELK stack. This enables real-time performance tracking of individual WordPress containers, allowing for performance benchmarking and resource usage analysis.

Figure 21: Metricbeat Services in Docker Compose

```
# Metricbeat for WordPress 1
metricbeat1:
  image: docker.elastic.co/beats/metricbeat:7.17.0
  user: root
  volumes:
    - ./metricbeat/metricbeat1.yml:/usr/share/metricbeat/metricbeat.yml
    - /var/lib/docker/containers:/var/lib/docker/containers:ro
    - /var/log:/var/log:ro
  depends_on:
    - wordpress1
  command: ["--strict.perms=false"]
  networks:
    - wp_network

# Metricbeat for WordPress 2
metricbeat2:
  image: docker.elastic.co/beats/metricbeat:7.17.0
  user: root
  volumes:
    - ./metricbeat/metricbeat2.yml:/usr/share/metricbeat/metricbeat.yml
    - /var/lib/docker/containers:/var/lib/docker/containers:ro
    - /var/log:/var/log:ro
  depends_on:
    - wordpress2
  command: ["--strict.perms=false"]
  networks:
    - wp_network
```

3.5.2.5 Centralised Logging Stack: Elasticsearch, Logstash, Kibana

To enable centralised log aggregation and analysis, this project integrates the ELK stack, composed of Elasticsearch, Logstash, and Kibana (see Figure 22). Elasticsearch is the core data store, indexing incoming logs from multiple WordPress and MySQL sources. Logstash acts as the data processing pipeline, receiving logs from Filebeat, parsing and filtering them using custom grok patterns and conditionals, and then forwarding structured data to Elasticsearch. Kibana provides a web-based interface for visualising and analysing the aggregated log data. By accessing elk.arp.com, users can search logs, build dashboards, and detect anomalies or

performance issues across WordPress instances in real-time. The reverse proxy configuration (see previous section) ensures convenient access via friendly domain names.

Figure 22: Elasticsearch Service in Docker Compose

```
# ELK Stack
elasticsearch:
  image: docker.elastic.co/elasticsearch/elasticsearch:7.17.0
  environment:
    - discovery.type=single-node
  volumes:
    - esdata:/usr/share/elasticsearch/data
  ports:
    - "9200:9200"
  networks:
    - wp_network

logstash:
  image: docker.elastic.co/logstash/logstash:7.17.0
  depends_on:
    - elasticsearch
  volumes:
    - ./logstash.conf:/usr/share/logstash/pipeline/logstash.conf
  ports:
    - "5044:5044"
  networks:
    - wp_network

kibana:
  image: docker.elastic.co/kibana/kibana:7.17.0
  depends_on:
    - elasticsearch
  ports:
    - "5601:5601"
  volumes:
    - ./kibana-logs:/usr/share/kibana/logs
  networks:
    - wp_network
```

3.5.3 Log Collection Pipeline

3.5.3.1 Filebeat Configuration

Filebeat is deployed alongside each key component (WordPress 1, WordPress 2, and Kibana) as a lightweight log shipper. Each Filebeat instance is configured with specific inputs to collect relevant log files and forward them to Logstash for processing.

For WordPress 1 (see Figure 23), the filebeat.yml file includes three types of inputs:

- Container logs: It monitors Docker container logs under “/var/lib/docker/containers” and filters lines containing the domain wordpress1.arp.com, tagging them with the custom field “log_source: apache1”.

- User activity logs: A local file “/var/log/user-activity.log” is monitored and labelled with “log_source: user-activity”, enabling later distinction in Logstash and Elasticsearch.
- MySQL slow query logs: Filebeat captures logs from “/var/log/mysql/slow.log”, applying a multiline pattern to combine multi-line entries beginning with a numeric date. This input is tagged as “log_source: mysql-slow-query”.

For WordPress 2 (see Figure 24), a similar configuration captures container logs filtered by the keyword wordpress2.arp.com and tags them with “log_source: apache2”. For Kibana, Filebeat uses Docker metadata enrichment and a conditional “drop_event” processor only to retain logs from the Kibana container (arp-kibana-1). The logs are tagged as “log_source: kibana-stdout”. All Filebeat instances are configured to send output to the centralised Logstash service via port 5044, enabling further filtering and transformation.

Figure 23: Filebeat 1 Configuration

```

1  ! filebeat1.yml
2  filebeat.inputs:
3  - type: container
4    paths:
5      - "/var/lib/docker/containers/*/*.log"
6    fields:
7      log_source: apache1
8      include_lines: ['wordpress1.arp.com']
9
10 - type: log
11   enabled: true
12   paths:
13     - /var/log/user-activity.log
14   fields:
15     log_source: user-activity
16
17 - type: log
18   enabled: true
19   paths:
20     - /var/log/mysql/slow.log
21   fields:
22     log_source: mysql-slow-query
23     multiline.pattern: '^d{6}' # MySQL slow query logs start with a date in numeric format
24     multiline.negate: true
25     multiline.match: after
26
27 output.logstash:
28   hosts: ["logstash:5044"] # Logstash server address and port
29 http:
30   enabled: true
31   host: "0.0.0.0"
32   port: 5066

```

Figure 24: Filebeat 2 Configuration

```
! filebeat2.yml
1  filebeat.inputs:
2    - type: container
3      paths:
4        - "/var/lib/docker/containers/*//*.log"
5      fields:
6        log_source: apache2
7        include_lines: ['wordpress2.arp.com']
8
9  output.logstash:
10    hosts: ["logstash:5044"]
11
```

3.5.3.2 Logstash Configuration

Logstash is configured to serve as the central processing unit in the logging pipeline. It listens to incoming logs from Filebeat via the Beats input plugin on port 5044. Once logs are received, Logstash uses conditional filters to parse and enrich the data based on its source, ensuring each log type is properly structured before indexing into Elasticsearch.

In the filter stage, Logstash distinguishes logs based on the “log_source” field set earlier by Filebeat:

- For Apache1 and Apache2 logs (from WordPress containers), the standard `%{COMBINEDAPACHELOG}` pattern is applied using the Grok plugin to extract common fields such as IP address, timestamp, HTTP method, URL, and status code.
- For user-activity logs, a custom Grok pattern is defined to parse structured fields like “msg”, “timestamp”, “level_name”, “user_ip”, “request_id”, and “duration”.
- For mysql-slow-query, no parsing is required; instead, a new field alert with value “slow_query_alert” is added to mark the event as potentially problematic.
- Additionally, if the message field matches a pattern containing “AlertName”, such as those from Kibana alerts, it is parsed to extract “alert_name” and “log_message”. An alert field with value “kibana_alert” is then appended.

In the output stage, logs are indexed into Elasticsearch with a dynamic index name pattern:

wordpress-[log_source]-logs-[date].

This approach organises logs by their source and date, making them easy to retrieve and analyse in Kibana. Furthermore, Logstash includes an email alert mechanism:

- If a log entry is marked as “slow_query_alert”, an email is sent to the administrator with details about the slow query.
- Similarly, “kibana_alert” entries trigger an alert email containing the alert message details.

This setup allows the system to centralise and structure log data and actively notify administrators when critical issues arise.

Figure 25: Logstash Configuration

```
logstash.conf

1 input {
2   beats {
3     port => 5044
4   }
5 }
6
7 filter {
8   if [fields][log_source] == "apache1" or [fields][log_source] == "apache2" {
9     grok {
10      match => { "message" => "%{COMBINEDAPACHELOG}" }
11    }
12   } else if [fields][log_source] == "user-activity" {
13     grok {
14      match => {
15        "message" => "msg=%{GREEDYDATA:msg},timestamp=%{TIMESTAMP_ISO8601:timestamp},level_name"
16      }
17    }
18    mutate {
19      convert => { "duration" => "integer" }
20    }
21   } else if [fields][log_source] == "mysql-slow-query" {
22     mutate {
23       add_field => { "alert" => "slow_query_alert" }
24     }
25   } else if [message] =~ /AlertName/ {
26     grok {
27       match => {
28         "message" => "AlertName: %{WORD:alert_name};- LogMessage: %{GREEDYDATA:log_message}"
29       }
30     }
31     mutate {
32       add_field => { "alert" => "kibana_alert" }
33     }
34   }
35 }
36
37 output {
38   elasticsearch {
39     hosts => ["http://elasticsearch:9200"]
40     index => "wordpress-%{[fields][log_source]}-logs-%{+YYYY.MM.dd}"
41   }
42
43   if [alert] == "slow_query_alert" {
44     email {
45       to => "jigang.guo@gmail.com"
46       from => "logstash@gmail.com"
47       subject => "Slow Query Alert"
48       body => "A slow query was detected with the following details: %{message}"
49       via => "smtp"
50       address => "smtp.gmail.com"
```

3.5.4 Custom Logging Extensions

3.5.4.1 Custom Plugin: Slow Query Simulator

I developed a lightweight WordPress plugin called Slow Query Simulator to simulate performance bottlenecks and evaluate logging and monitoring capabilities. It executes an SQL query using `SELECT SLEEP(5)`, forcing a 5-second delay at the database level. The plugin records the start and end time using `microtime(true)` and outputs the total execution duration. This allows for the controlled generation of slow queries, which can be detected and logged by

monitoring tools such as Filebeat and Logstash. This is beneficial for verifying whether the ELK stack can accurately capture and visualise database performance anomalies.

Figure 26:Slow Query Simulator Plugin

```
<?php
/*
Plugin Name: Slow Query Simulator
Description: This plugin simulates slow queries for testing purposes by using the `SLEEP()` SQL function.
Version: 1.0
Author: Jigang Guo
License: GPL2
*/

add_action('init', function() {
    global $wpdb;

    if ( isset($_GET['do_slow_query']) ) {
        $start = microtime(true);

        $wpdb->query("SELECT SLEEP(5)");

        $end = microtime(true);

        echo "Query executed in " . ($end - $start) . " seconds.";

        exit;
    }
});
```

3.5.4.2 Custom Plugin: User Activity Logger

To monitor front-end user behaviours on the WordPress site, I developed a User Activity Logger plugin. This plugin leverages the “Monolog” library to record structured logs in a custom format compatible with the Logstash GROK filter.

These user behaviours include scrolling past 50% of the height, page stay duration time, button click times, and accordion toggles. Each action is logged with key metadata such as timestamp, user IP, request ID, and interaction duration time. Logs are written to user-activity.log. These logs are later shipped via Filebeat and parsed by Logstash for further analysis in Kibana.

Figure 27: User Activity Log Plugin

```
p1 > wp-content > plugins > custom-logs > user-activity-logger.php
23 $log = new Logger('user_activity');
24
25 // Define the log format
26 $logFormat = "msg=%message%,timestamp=%datetime%,level_name=%level_name%,user_ip=%extra.user_ip%,request_id=%extra.request_id%,";
27 $logFormat .= "duration=%context.duration%\n";
28
29 // Use a StreamHandler with custom formatting
30 $handler = new StreamHandler(__DIR__ . '/user-activity.log', Logger::INFO);
31 $handler->setFormatter(new LineFormatter($logFormat, null, true, true));
32
33 $log->pushHandler($handler);
34 $log->pushProcessor(new UidProcessor());
35
36 > function log_user_activity_on_click() { ...
63 }
64
65 // Record user scroll down
66 > function log_scroll_event( $message ) { ...
74 }
75
76 // Record page stay time
77 > function log_page_stay_duration( ) { ...
85 }
86
87 // Record accordion toggle behaviour
88 > function log_accordion_toggle( ) { ...
99 }
100
101 // Record Click behaviour
102 > function log_click_event( ) { ...
110 }
111
112 add_action('init', 'log_user_activity_on_click');
113
114 // Trigger user behaviour logs
115 > function add_user_activity_scripts() { ...
126 }
127
128 add_action('wp_footer', 'add_user_activity_scripts');
```

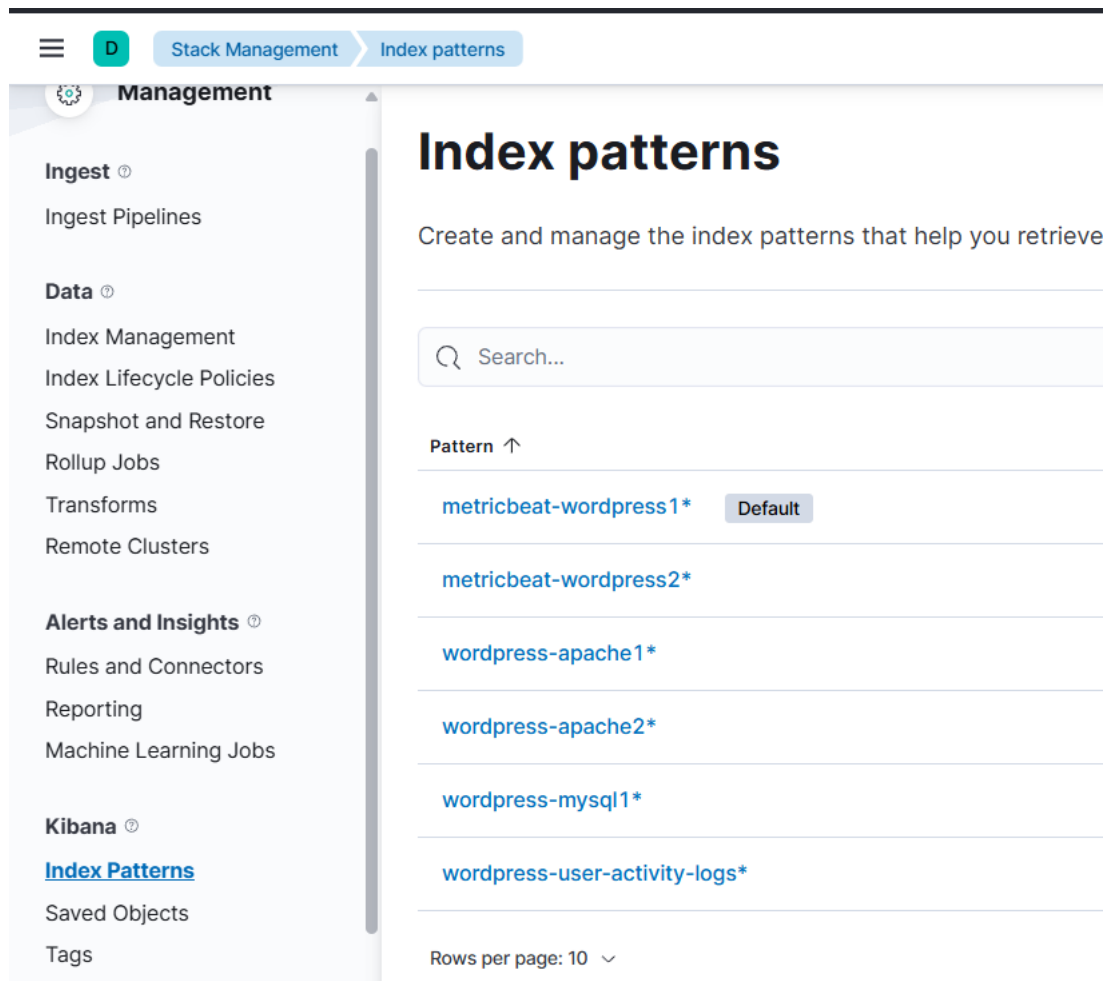
3.5.5 Kibana Configuration

3.5.5.1 Index Creation in Kibana

To facilitate structured querying and visualisation in Kibana, I created a series of index patterns under Stack Management. These include: “metricbeat-wordpress1*”, “metricbeat-wordpress2*”, “metricbeat-apache1*”, “metricbeat-apache2*”, “wordpress-mysql1*”, and “wordpress-user-activity-logs*”. Each pattern corresponds to logs collected from specific components across different virtual machines. For example, the “metricbeat-*” indices are used for system and service metrics, while “wordpress-user-activity-logs*” stores custom logs from the WordPress plugin. Defining these patterns enables efficient filtering, dashboard creation,

and time-based analysis, which are essential for performance comparison and anomaly detection.

Figure 28: Index Patterns Created in Kibana



3.5.5.2 Alerts Rule Configuration

In Kibana, I configured an alert rule to record when a certain threshold of POST requests was exceeded. As illustrated in Figure 29, the configuration of a Kibana alert involves four key steps. First, the execution interval is defined to determine how frequently the rule is evaluated (e.g., every minute). Second, the specific resource being monitored is selected, in this case, the “wp-login.php” file, which is often targeted in brute-force attacks. Third, the trigger

condition is configured, specifying that the alert should be activated if the number of requests to this endpoint exceeds five times per minute. Finally, a message template is composed to present the message content (see Figure 30).

Figure 29: Alert Rule Created in Kibana

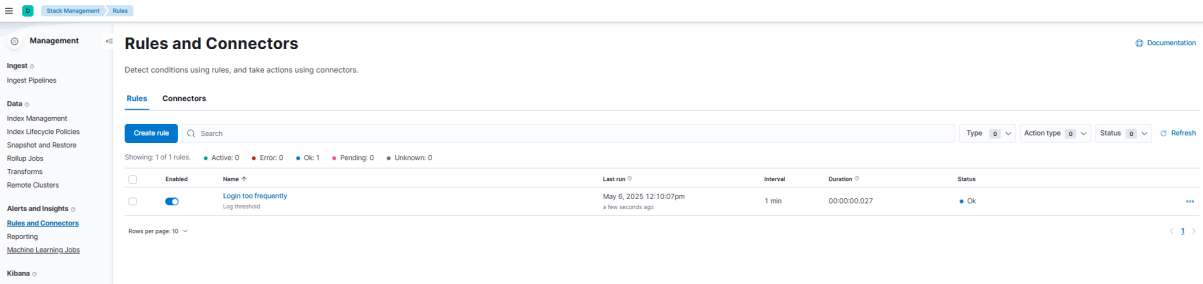


Figure 30: Alert Rule Configuration

Edit rule

Name: Login too frequently

Tags (optional):

Check every: 1 minute

Notify: Only on status change

Log threshold

Alert when the log aggregation exceeds the threshold. [Documentation](#)

WHEN THE count OF LOG ENTRIES

WITH request.keyword IS /wp-login.php

AND verb.keyword IS POST

+ Add condition

IS more than 5

FOR THE LAST 1 minute

GROUP BY clientip.keyword

Actions

Login request too frequency

Run when: Fired

Server log connector: Login request too frequency

Level: Error

Message: - AlertName: {{rule.name}}

Cancel Save

3.5.5.3 Dashboard Creation in Kibana

To analyse user interactions on the WordPress site, I created a custom dashboard in Kibana using “wordpress-user-activity-logs*” as the source index (see Figure 31). One key metric displayed on the dashboard is the number of unique visitors, which is approximated by counting the number of distinct “request_id” values. Each “request_id” serves as a session identifier generated per user visit, enabling estimation of session-based traffic (see Figure 32).

Figure 31: Count of Visitors Created in Dashboard

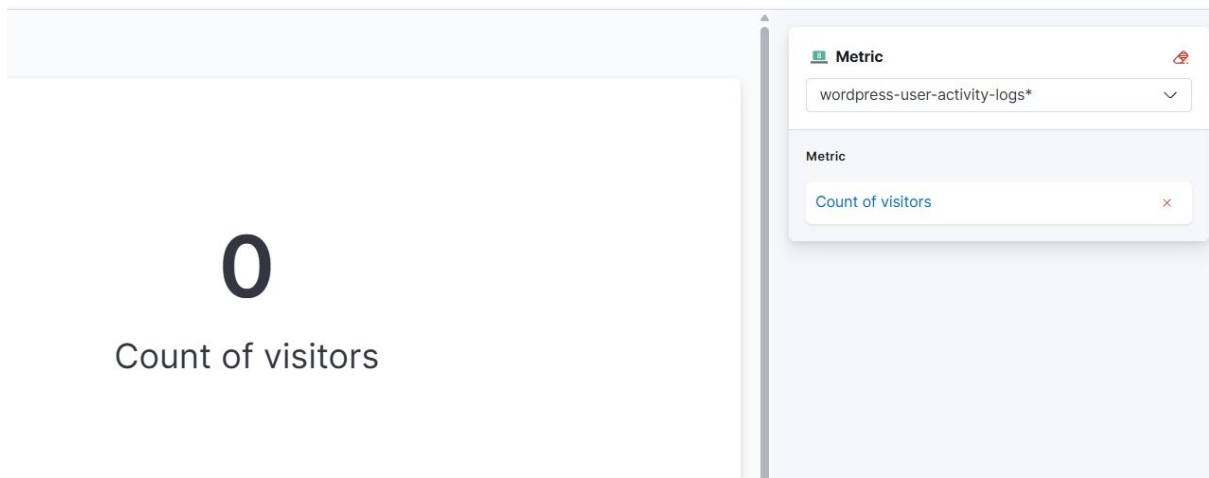
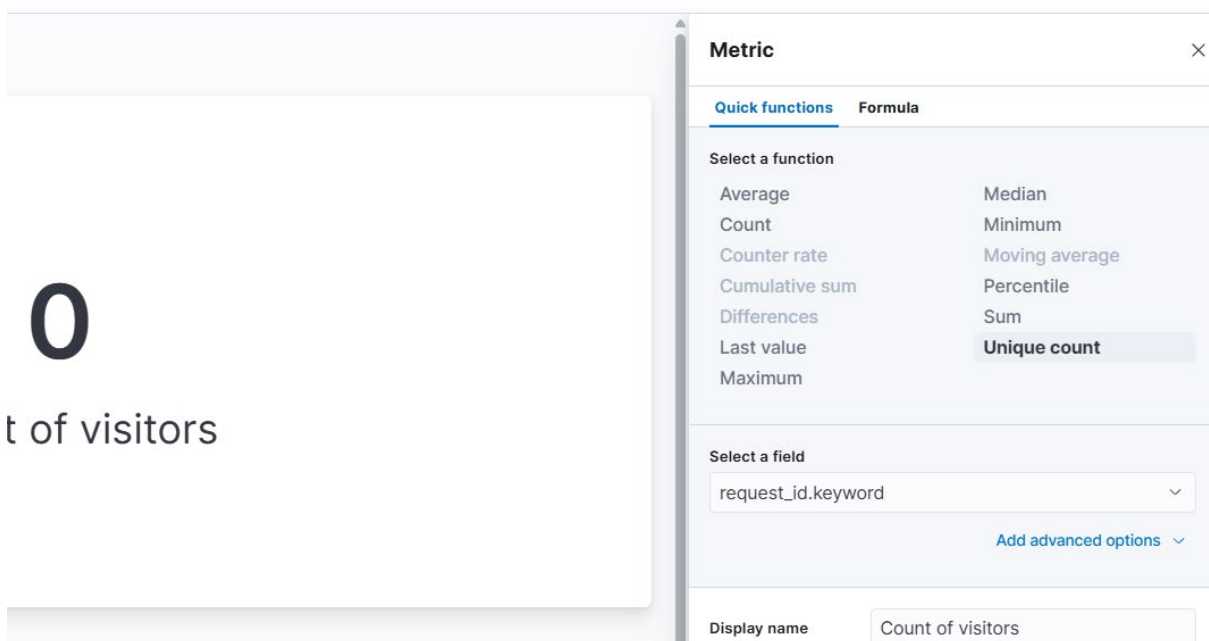


Figure 32: Count of Visitors Configuration in Dashboard



To gain deeper insights into user engagement, I utilised a vertical bar chart to visualise visitor stay time (see Figure 33). In this chart, the X-axis represents the unique count of “request_id”, which corresponds to individual user sessions (see Figure 35), while the Y-axis shows the sum of the duration field, which records the total time each visitor spent on the page (see Figure 34).

Figure 33: User Duration Time Created in Dashboard

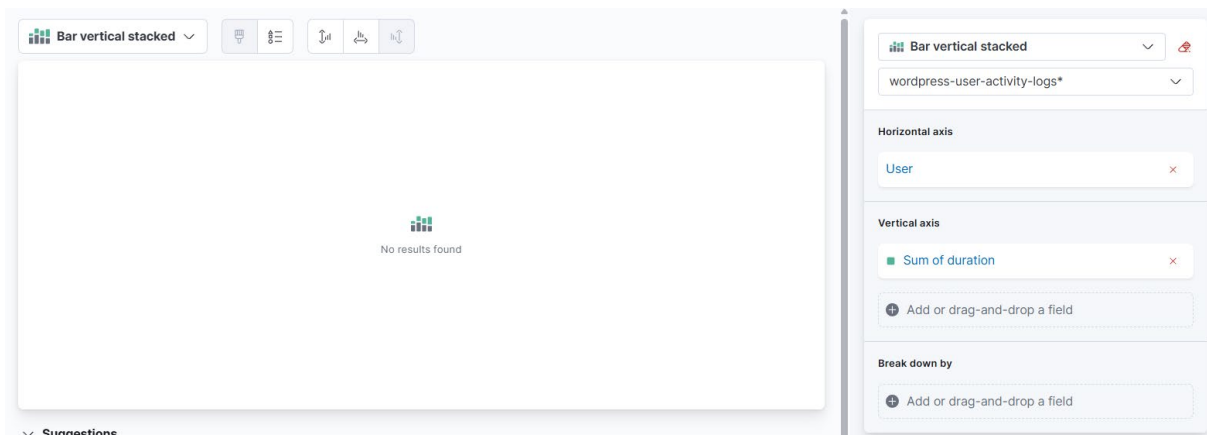
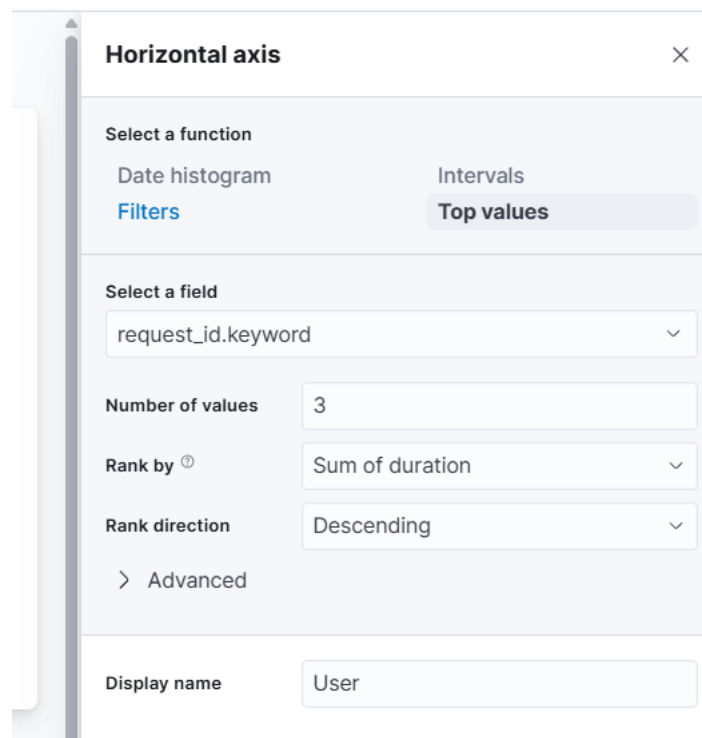


Figure 34: User Duration Time Vertical Axis Configuration in Dashboard

The 'Vertical axis' configuration dialog is shown with the following details:

- Vertical axis** (Title bar with a close button)
- Quick functions** (Selected tab) | **Formula** (Unselected tab)
- Select a function:**
 - Average
 - Count
 - Counter rate
 - Cumulative sum
 - Differences
 - Last value
 - Maximum
 - Median
 - Minimum
 - Moving average
 - Percentile
 - Sum** (Selected)
 - Unique count
- Select a field:**
 - duration (Selected in dropdown)
 - [Add advanced options](#) (Link)
- Display name:** Sum of duration

Figure 35: User Duration Time Horizontal Axis Configuration in Dashboard



A configuration panel titled "Horizontal axis" with a close button (X) in the top right corner. It contains several sections: "Select a function" with three tabs: "Date histogram", "Intervals", and "Filters" (which is selected and highlighted in blue); "Select a field" with a dropdown menu showing "request_id.keyword"; "Number of values" with a text input field containing "3"; "Rank by" with a dropdown menu showing "Sum of duration"; "Rank direction" with a dropdown menu showing "Descending"; an "Advanced" section with a right-pointing chevron; and "Display name" with a text input field containing "User".

To analyse how many users were actively engaging with the content, I applied a filter on the field “msg.keyword” with the value: “User scroll down to the 50% position” (see Figure 36). This ensures we only analyse logs that record scroll behaviour. Then, I calculated the unique count of “request_id”, which represents individual sessions or visitors (see Figure 37).

Figure 36: Count of Users Scroll Down 50% Created in Dashboard

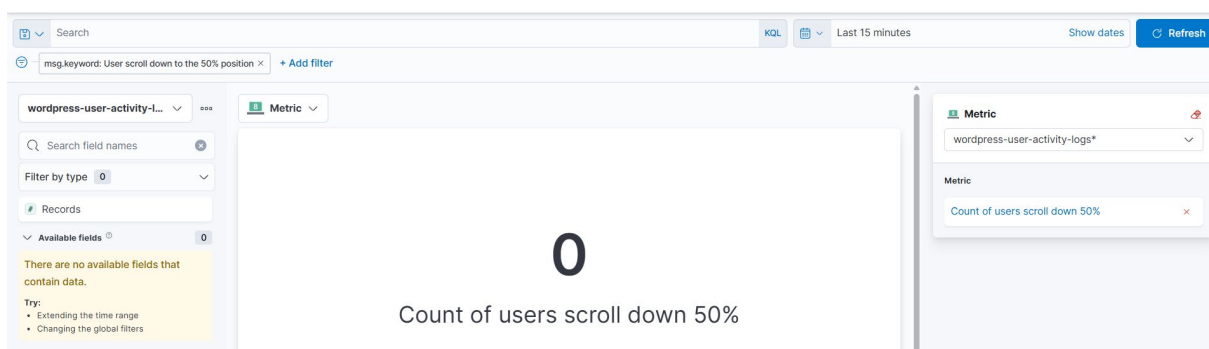


Figure 37: Count of Users Scroll Down 50% Configuration in Dashboard

The screenshot shows a 'Metric' configuration panel with a close button (X) in the top right corner. It has two tabs: 'Quick functions' (active) and 'Formula'. Under 'Quick functions', there is a 'Select a function' section with a list of functions: Average, Count, Counter rate, Cumulative sum, Differences, Last value, Maximum, Median, Minimum, Moving average, Percentile, Sum, and Unique count. The 'Unique count' function is highlighted. Below this is a 'Select a field' section with a dropdown menu showing 'request_id.keyword'. There is a link 'Add advanced options' with a downward arrow. At the bottom, there are two fields: 'Display name' with the value 'Count of users scroll down 50%' and 'Value format' with the value 'Default'.

To measure user interaction through clicks, I applied a filter on the field “msg.keyword” with the value “User triggered click event”. Then I set the horizontal axis to display “request_id” (representing individual visitors, see Figure 38), and the vertical axis to count the number of logs matching each session (see Figure 39). This allowed me to visualise how many times each visitor clicked during their session.

Figure 38: Users Triggered Click Event Times Created in Dashboard

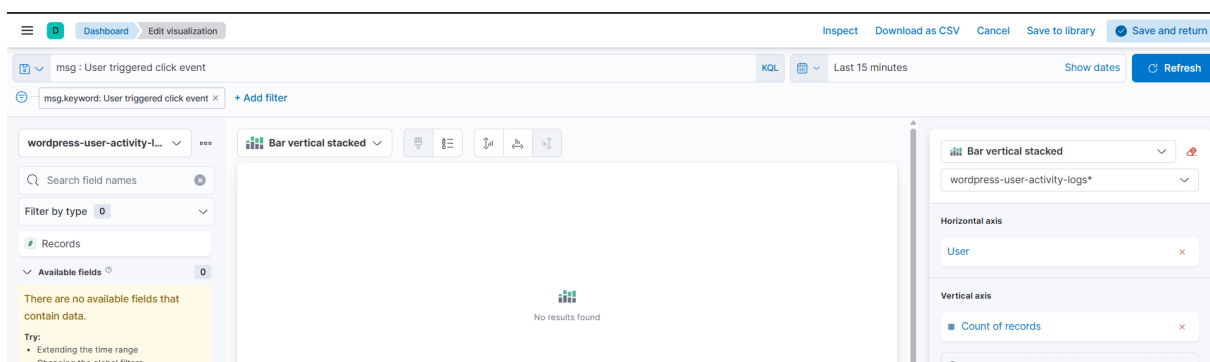
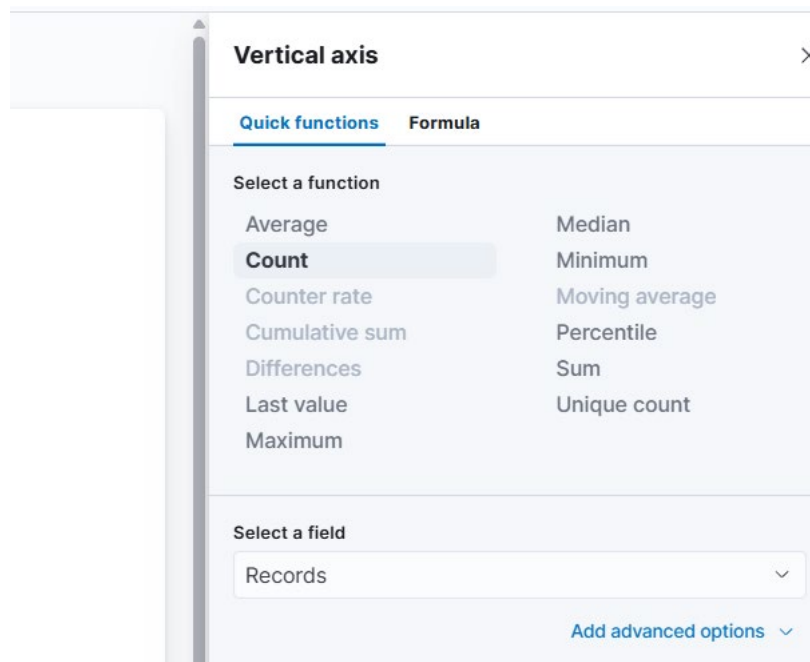


Figure 39: Users Triggered Click Event Times Configuration in Dashboard



3.5.6 Brute-Force Detection Script for WordPress Login Attempts

A simple shell script was implemented to monitor potential brute-force attacks targeting the WordPress login endpoint. This script analyses the Docker container logs to identify IP addresses that have made excessive login attempts to “/wp-login.php”. Specifically, it extracts all POST requests to this endpoint, counts the number of occurrences per IP address, and outputs any IP with more than five attempts (see Figure 40). This lightweight detection can be extended to automatically block malicious IPs using firewall rules or intrusion prevention systems.

Figure 40: Shell Script to Calculate Brute-Force Times

```
$ count_wp_logins.sh
1  #!/bin/bash
2
3  CONTAINER_ID="ff10f57c81ea"
4
5  CURRENT_TIME=$(date +%s)
6  THRESHOLD=$((CURRENT_TIME - 900))
7
8  docker logs "$CONTAINER_ID" 2>&1 \
9  | grep 'POST /wp-login.php' \
10 | awk -v threshold="$THRESHOLD" '
11   function parse_time(str) {
12     gsub("/", " ", str)
13     gsub(":", " ", str)
14     split(str, parts, " ")
15     return parts[1] " " parts[2] " " parts[3] " " parts[4] ":" parts[5] ":" parts[6] " +1200"
16   }
17
18   {
19     match($0, /\[([0-9]{2})\/([A-Za-z]{3})\/([0-9]{4}):([0-9]{2}):([0-9]{2}):([0-9]{2})/, m)
20     if (m[1] != "") {
21       cmd = "date -d \"" parse_time(m[1]) "\" +%s"
22       cmd | getline log_time
23       close(cmd)
24
25       if (log_time >= threshold) {
26         ip = $1
27         ip_count[ip]++
28       }
29     }
30   }
31   END {
32     for (ip in ip_count)
33       if (ip_count[ip] > 5)
34         print ip, ip_count[ip]
35   }
36   '
37
```

3.5.7 User Behaviour Log Analysis Script

To verify the accuracy of the dashboard data, I wrote a Bash script that analyses raw logs from user-activity.log (see Figure 41). The script filters logs from the past hour, extracts key fields such as “request_id”, “msg”, and “duration”, and summarises user actions per session. It counts click events, scroll events (specifically "scroll down to 50%"), and accumulates total stay duration for each visitor.

Figure 41: Shell Script to Analysis User Behaviours

```
BEGIN {
    start_epoch = to_epoch(start)
}

{
    timestamp = ""
    request_id = ""
    msg = ""
    duration = 0

    for (i = 1; i <= NF; i++) {
        if ($i ~ /timestamp=/) {
            timestamp = $i
        }
        if ($i ~ /request_id=/) {
            split($i, a, "="); request_id = a[2]
        }
        if ($i ~ /msg=/) {
            msg = substr($i, 5)
        }
        if ($i ~ /duration=/) {
            split($i, c, "="); duration = c[2]
        }
    }

    log_epoch = to_epoch(timestamp)

    if (log_epoch >= start_epoch) {
        click[request_id] += (msg ~ /click/) ? 1 : 0
        scroll[request_id] += (msg ~ /User scroll down to the 50% position/) ? 1 : 0
        stay[request_id] += (msg ~ /stayed on the page/) ? 1 : 0
        total_duration[request_id] += duration
    }
}

END {
    printf "%-36s %-10s %-10s %-10s\n", "Request ID", "Clicks", "Scrolls", "Duration"
    for (id in click) {
        printf "%-36s %-10d %-10d %-10d\n", id, click[id], scroll[id], total_duration[id]
    }
}

' "$LOGFILE"
```

3.6 Evaluation Stage

This evaluation aims to assess the effectiveness and reliability of the proposed logging and monitoring system across three functional components: brute-force attack detection and alerting, system performance and slow query monitoring, and user behaviour analysis. Each component was tested through practical scenarios to verify data collection, visualisation, and system responsiveness.

3.6.1 Brute-Force Attack Detection and Alerting

To evaluate the system's ability to detect brute-force login attempts, simulated attacks were conducted by repeatedly sending POST requests to “/wp-login.php” using automated tools. The ELK stack successfully captured these attempts through logs collected by Filebeat. A custom Bash script was used to aggregate IP addresses with excessive failed attempts.

Elast-alert was configured to trigger an email alert if an IP exceeded five login attempts within a short timeframe. The delay from attack detection to email delivery was roughly estimated at 10 - 20 seconds, demonstrating that the alerting mechanism is timely and practical for intrusion detection in real-world use cases.

3.6.2 System Performance and Slow Query Monitoring

The system monitoring process utilised Metricbeat to track CPU and memory usage metrics for Apache, MySQL and WordPress services. The system forwarded data to Elasticsearch before Kibana dashboards displayed the information.

A simulated SQL query with intentional delay (SELECT SLEEP(5)) was executed to generate a slow query log for slow query analysis. Filebeat processed the logs before Kibana displayed them in its visual interface. The dashboard displayed query results within less than thirty seconds from query execution to prove its capability to identify performance bottlenecks through real-time detection.

3.6.3 User Behaviour Analysis

The system collected user behaviour data through structured logs that contained user ID information along with timestamps, event types (clicks, scrolls, page stays) and duration measurements. A Bash script was written specifically for this project to calculate user ID-based metrics, including click counts, scroll events, and total user duration.

The results were compared with visualisations created in Kibana using filters and aggregations. Metrics like the number of users who scrolled past 50% of the page and the frequency of clicks

events were consistent between the script output and the Kibana dashboards. This validated the system's ability to interpret and visualise user interaction patterns accurately.

3.6.4 Summary

Across all three modules, the evaluation confirmed that the ELK-based logging system can:

- Detect and alert on brute-force login attempts in near real-time.
- Monitor system resource usage and effectively identify slow database queries.
- Analyse and visualise user behaviour patterns for performance and UX insights.

CHAPTER 4 EXPERIMENTAL RESULTS AND DISCUSSION

4.1 Introduction to Experimental Setup

This experiment was designed to explore how the ELK stack improves log analysis, monitoring, and user insight in distributed WordPress environments. All services, including two separate WordPress sites, their MySQL databases, and the ELK stack, were deployed using Docker containers to simulate a realistic, scalable setup. For security threat detection, I simulated brute-force login attempts, I also integrated email alerts for suspicious requests. To monitor system resources, Metricbeat was used to track CPU and memory usage across WordPress containers, MySQL slow queries were enabled and forwarded to ELK using Filebeat. For user behaviour analysis, a custom plugin was created to log user actions, these logs were sent to ELK to help analyse interaction patterns.

4.1.1 Experimentation on Security Monitoring

The experiment successfully demonstrated the ELK stack's capability to detect and respond to suspicious login behaviour on a WordPress site. A simulated brute-force attack generated six consecutive “/wp-login.php” post requests from the same IP address. These events were captured in the raw Docker logs (see Figure 42) and processed using a custom shell script, which accurately identified the excessive login attempts (see Figure 43). The same events were visualised in Kibana, confirming that the logs were correctly ingested and parsed (see Figure 44). The Watcher rule was triggered upon detecting the threshold breach (see Figure 45), and an alert email was sent automatically (see Figure 46). This verified that the security monitoring pipeline, from log collection to alert delivery, was functioning correctly and in real time.

Figure 42: User Login Records in Apache Log

```
172.19.0.1 - - [10/May/2025:10:06:06 +1200] "POST /wp-login.php HTTP/1.0" 200 2363 "http://wordpress1.arp.d
Webkit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36"
172.19.0.1 - - [10/May/2025:10:06:08 +1200] "POST /wp-login.php HTTP/1.0" 200 2363 "http://wordpress1.arp.d
Webkit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36"
172.19.0.1 - - [10/May/2025:10:06:11 +1200] "POST /wp-login.php HTTP/1.0" 200 2363 "http://wordpress1.arp.d
Webkit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36"
172.19.0.1 - - [10/May/2025:10:06:13 +1200] "POST /wp-login.php HTTP/1.0" 200 2363 "http://wordpress1.arp.d
Webkit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36"
172.19.0.1 - - [10/May/2025:10:06:15 +1200] "POST /wp-login.php HTTP/1.0" 200 2363 "http://wordpress1.arp.d
Webkit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36"
172.19.0.1 - - [10/May/2025:10:06:18 +1200] "POST /wp-login.php HTTP/1.0" 200 2363 "http://wordpress1.arp.d
Webkit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36"
```

Figure 43: Result of User Login Times by Shell Script

```
Administrator@Jigang MINGW64 /d/weltec/arp (master)
$ ./count_wp_logins.sh
172.19.0.1 6
```

Figure 44: User Login Logs Appeared in Kibana

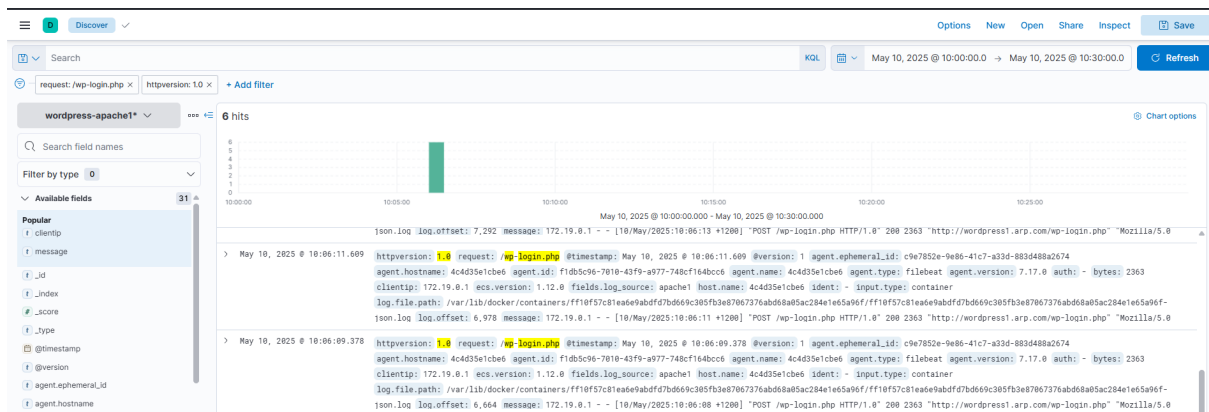


Figure 45: Alert Message Stored in Kibana Log

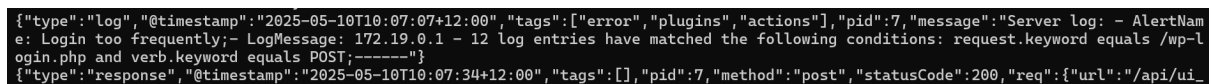
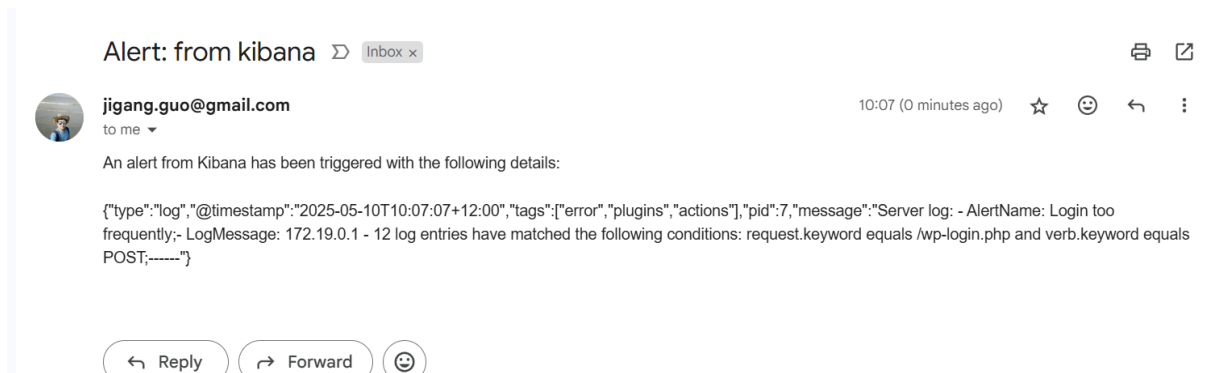


Figure 46: Alert Email Received from Kibana Log



4.1.1.1 Discussion

This experiment demonstrated the effectiveness of ELK Stack in real-time security monitoring for WordPress login activity. The system successfully identified repeated suspicious login attempts from a single IP address by simulating a brute-force attack. The seamless flow, from Docker log collection and shell-based preprocessing to Kibana visualisation and Watcher-triggered alerts, validated the reliability and responsiveness of the ELK pipeline. The timely

email alert confirmed that potential threats can be detected and reported automatically, highlighting ELK’s practical value in enhancing WordPress site security.

4.1.2 Experimentation on Performance Monitoring

This experiment verified that Metricbeat, integrated with the ELK Stack, can effectively monitor system performance for distributed WordPress containers. As shown in Figure 47, the “docker ps” command identified two running Metric containers. Their corresponding dashboards, shown in Figure 48, Figure 49 and Figure 50, display real-time metrics such as CPU, memory, load, and network traffic. To validate accuracy, “htop” was run inside each WordPress container. The results (Figure 53 and Figure 54) showed CPU and memory values consistent with those in the dashboards. Metricbeat logs in Elasticsearch (Figure 52 and Figure 53) further confirmed successful data collection and transmission. These results demonstrate that Metricbeat reliably captures and visualises system metrics across multiple WordPress instances.

Figure 47: Metricbeat Containers Running Status

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
169ea5a09a8	docker.elastic.co/beats/metricbeat:7.17.0	"/usr/bin/tini -- /u..."	6 minutes ago	Up About a minute		arp-metricbeat1-1
292a36255ce	docker.elastic.co/beats/metricbeat:7.17.0	"/usr/bin/tini -- /u..."	6 minutes ago	Up About a minute		arp-metricbeat2-1
abcdd6a3b5ce	elastic/filebeat:7.17.0	"/usr/bin/tini -- /u..."	6 minutes ago	Up 6 minutes		arp-filebeat1-1
634b918a87ff	elastic/filebeat:7.17.0	"/usr/bin/tini -- /u..."	6 minutes ago	Up 6 minutes		arp-filebeat2-1
0972f4ab1796	wordpress:latest	"docker-entrypoint.s..."	6 minutes ago	Up 6 minutes	80/tcp	arp-wordpress1-1
0b08bac1fcac	wordpress:latest	"docker-entrypoint.s..."	6 minutes ago	Up 6 minutes	80/tcp	arp-wordpress2-1
d0ba8101a2f1	docker.elastic.co/logstash/logstash:7.17.0	"/usr/local/bin/dock..."	6 minutes ago	Up 6 minutes	0.0.0.0:5044->5044/tcp, 9600/tcp	arp-logstash-1
c2b9ba0f118d	docker.elastic.co/kibana/kibana:7.17.0	"/bin/tini -- /usr/L..."	6 minutes ago	Up 6 minutes	0.0.0.0:5601->5601/tcp	arp-kibana-1
7387558e559f	nginx:latest	"docker-entrypoint.s..."	6 minutes ago	Up 6 minutes	0.0.0.0:80->80/tcp	nginx_proxy
2a3c85ee4317	mysql:5.7	"docker-entrypoint.s..."	6 minutes ago	Up 6 minutes	3306/tcp, 33060/tcp	arp-mysql1-1
25f28afd256e	mysql:5.7	"docker-entrypoint.s..."	6 minutes ago	Up 6 minutes	3306/tcp, 33060/tcp	arp-mysql2-1
383a276617cd	docker.elastic.co/elasticsearch/elasticsearch:7.17.0	"/bin/tini -- /usr/L..."	6 minutes ago	Up 6 minutes	0.0.0.0:9200->9200/tcp, 9300/tcp	arp-elasticsearch-1

Figure 48: Metrics Collected By Metricbeats in Dashboard

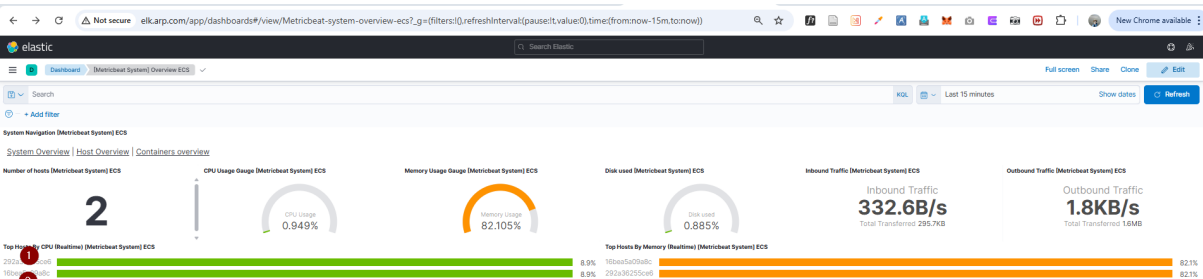


Figure 49: Metrics Collected by Metricbeat 1

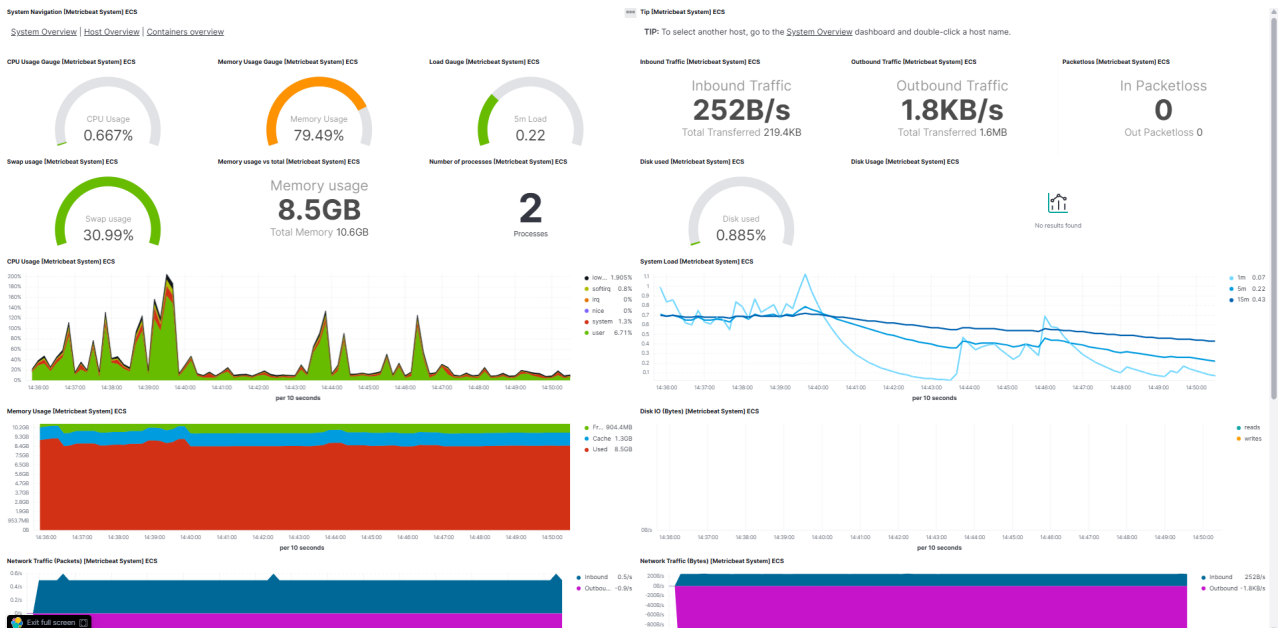


Figure 50: Metrics Collected by Metricbeat 2

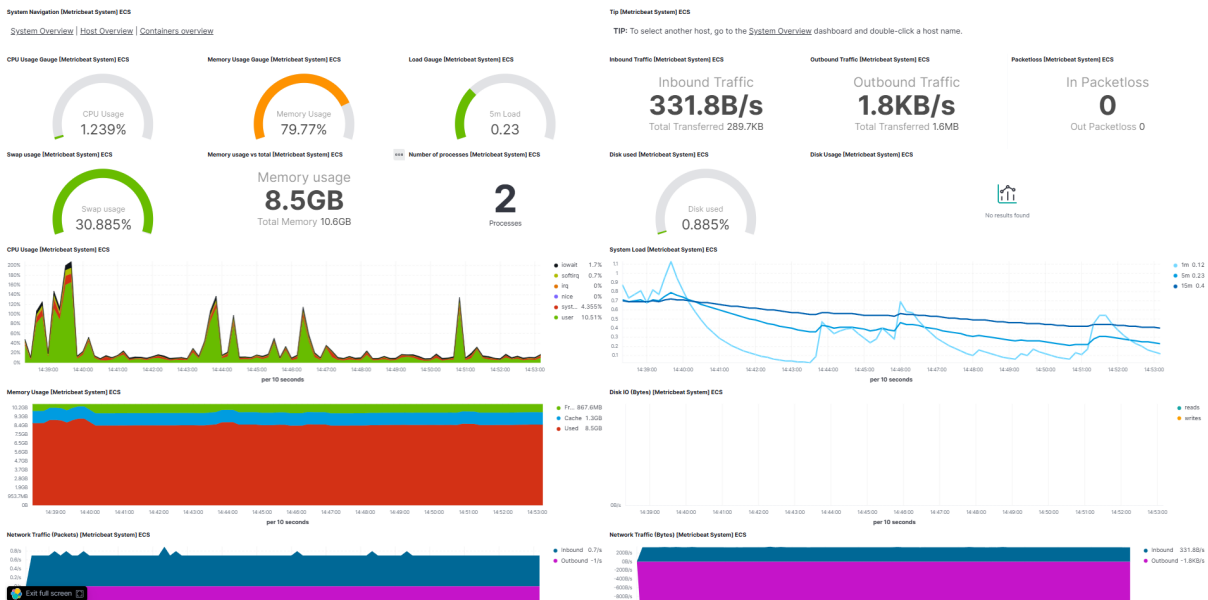


Figure 51: Logs Collected by Metricbeat 1 Shown in Kibana

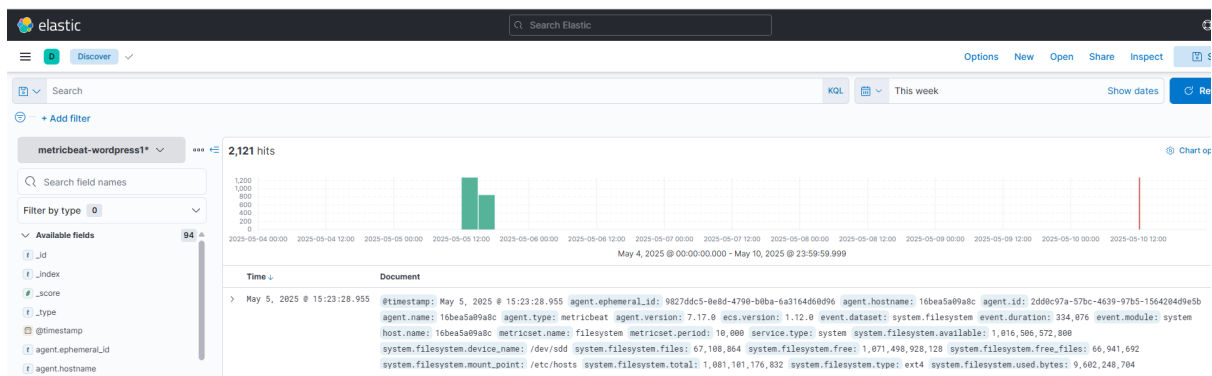


Figure 52: Logs Collected by Metricbeat 2 Shown in Kibana



Figure 53: Metrics Shown on WordPress 1 Server by HTOP

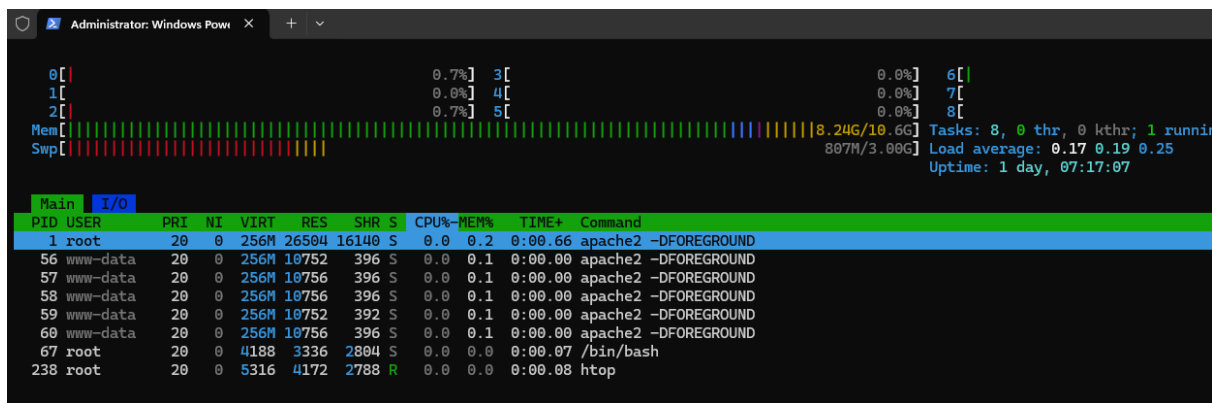
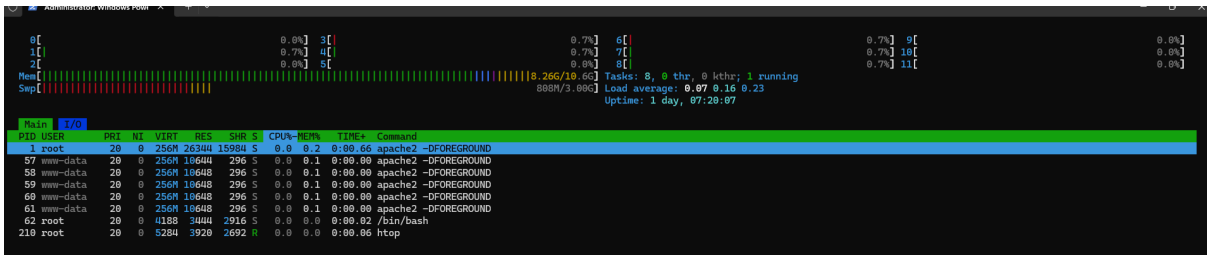


Figure 54: Metrics Shown on WordPress 2 Server by HTOP



4.1.2.1 MySQL Slow Query monitoring

Figure 55 displays a captured slow SQL query that took 5 seconds to execute, exceeding the predefined threshold. The slow query log was successfully recorded (see Figure 56) and parsed by Logstash. The structured output was visualised in Kibana, as shown in Figure 57, confirming proper data ingestion and processing. Consequently, an alert email was automatically triggered, as illustrated in Figure 58, demonstrating the effectiveness of the monitoring and alerting setup.

Figure 55: Slow SQL Query Result

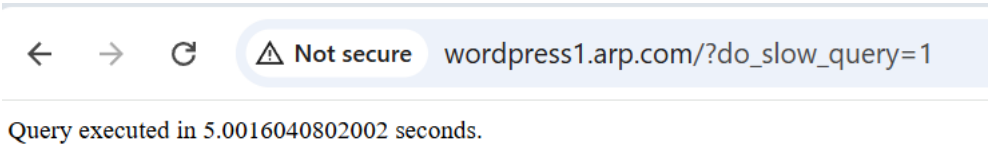


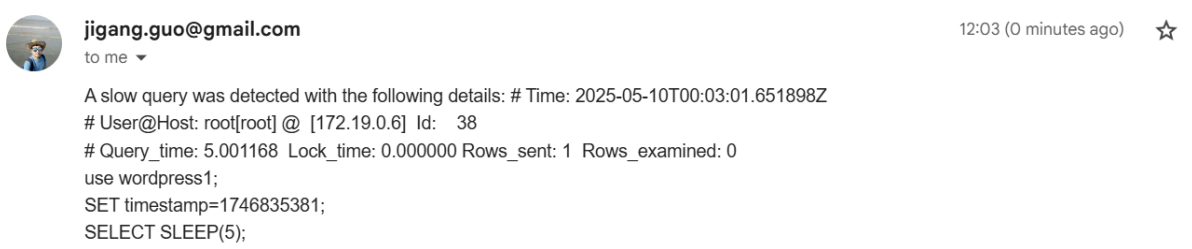
Figure 56: Slow SQL Query Raw Logs

```
822 ,(@time_zone_id, 2138198427, 0);
823 mysqld, Version: 5.7.44-log (MySQL Community Server (GPL)). started with:
824 Tcp port: 0 Unix socket: /var/run/mysqld/mysqld.sock
825 Time Id Command Argument
826 # Time: 2025-05-10T00:03:01.651898Z
827 # User@Host: root[root] @ [172.19.0.6] Id: 38
828 # Query_time: 5.001168 Lock_time: 0.000000 Rows_sent: 1 Rows_examined: 0
829 use wordpress1;
830 SET timestamp=1746835381;
831 SELECT SLEEP(5);
832
```

Figure 57: Slow SQL Query Logs Shown in Kibana



Figure 58: Alert Message of Slow SQL Query Via Email Notification



4.1.2.2 Discussion

The experiments confirmed the effectiveness of using Metricbeats and the ELK Stack for real-time monitoring and alerting in a distributed WordPress environment. Metricbeats successfully captured system-level metrics from multiple containers, such as CPU, memory usage, load, and network traffic, with data accuracy validated against the native “htop” tool. This

demonstrates the reliability of Metricbeat in providing consistent and actionable system performance insights.

Additionally, the slow query monitoring setup proved effective in detecting performance issues. The ELK Stack accurately ingested and visualised slow SQL queries, and the alerting mechanism functioned as expected by notifying administrators when predefined thresholds were exceeded. Together, these results showcase ELK's robust capabilities for both performance tracking and proactive issue detection in a containerised web environment.

4.1.3 Experimentation on Decision-Making Support Using Custom User Activity Logs

To evaluate user behaviour in this experiment, a typical e-commerce website was built using the Kadence theme in WordPress (see Figure 59). Four simulated users performed various browsing actions. Figure 60 displays the original log entries captured during these sessions. Figure 61 shows the Kibana dashboard visualising key metrics: the number of users who visited the homepage, clicked promotional buttons, the average time spent on the homepage, and how many users scrolled to the bottom of the page. These metrics were further verified through a custom shell script, and the results matched the visualised data (see Figure 62). Additionally, Figure 63 shows ELK logs confirming the successful ingestion and recording of all user activities.

Figure 59: An E-Commerce Store Built on Kadence Theme

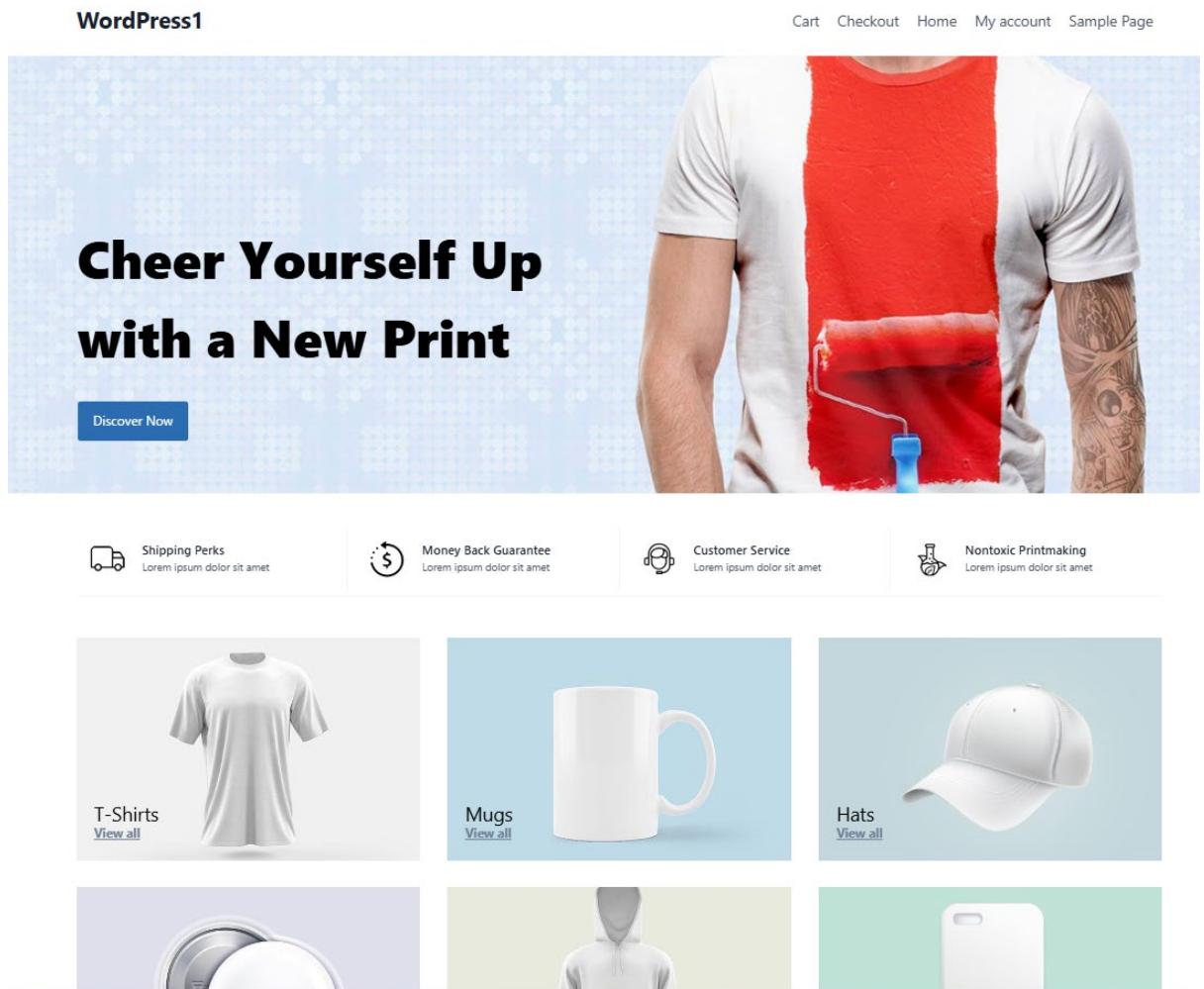


Figure 60: User Behaviour Raw Logs

```
msg=User scroll down to the 50% position,timestamp=2025-05-06T21:13:19.916682+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=ea91ec19465e30fb6829eeead0411224,duration=0
msg=User scroll down to the 50% position,timestamp=2025-05-06T21:13:28.063158+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=ea91ec19465e30fb6829eeead0411224,duration=0
msg=User scroll down to the 50% position,timestamp=2025-05-06T21:13:33.571203+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=ea91ec19465e30fb6829eeead0411224,duration=0
msg=User scroll down to the 50% position,timestamp=2025-05-06T21:13:33.572333+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=8fe13165bf34d3eb6fac13dd40c84201,duration=0
msg=User scroll down to the 50% position,timestamp=2025-05-06T21:13:36.879048+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=8fe13165bf34d3eb6fac13dd40c84201,duration=0
msg=User scroll down to the 50% position,timestamp=2025-05-06T21:13:37.606559+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=ea91ec19465e30fb6829eeead0411224,duration=0
msg=User scroll down to the 50% position,timestamp=2025-05-06T21:13:41.384669+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=ea91ec19465e30fb6829eeead0411224,duration=0
msg=User scroll down to the 50% position,timestamp=2025-05-06T21:13:41.385889+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=8fe13165bf34d3eb6fac13dd40c84201,duration=0
msg=User scroll down to the 50% position,timestamp=2025-05-06T21:13:45.572228+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=8fe13165bf34d3eb6fac13dd40c84201,duration=0
msg=User scroll down to the 50% position,timestamp=2025-05-06T21:13:45.573417+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=ea91ec19465e30fb6829eeead0411224,duration=0
msg=User scroll down to the 50% position,timestamp=2025-05-06T21:13:48.702125+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=ea91ec19465e30fb6829eeead0411224,duration=0
msg=User scroll down to the 50% position,timestamp=2025-05-06T21:13:48.707675+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=8fe13165bf34d3eb6fac13dd40c84201,duration=0
msg=User scroll down to the 50% position,timestamp=2025-05-06T21:13:53.047968+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=8fe13165bf34d3eb6fac13dd40c84201,duration=0
msg=User scroll down to the 50% position,timestamp=2025-05-06T21:13:53.068360+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=ea91ec19465e30fb6829eeead0411224,duration=0
msg=User stayed on the page,timestamp=2025-05-06T21:13:56.443087+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=8fe13165bf34d3eb6fac13dd40c84201,duration=4
msg=User scroll down to the 50% position,timestamp=2025-05-06T21:13:59.653339+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=ea91ec19465e30fb6829eeead0411224,duration=0
msg=User stayed on the page,timestamp=2025-05-06T21:14:08.044929+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=ea91ec19465e30fb6829eeead0411224,duration=172
msg=User stayed on the page,timestamp=2025-05-06T21:16:08.082612+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=ea91ec19465e30fb6829eeead0411224,duration=8
msg=User stayed on the page,timestamp=2025-05-06T21:16:20.659459+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=ea91ec19465e30fb6829eeead0411224,duration=13
msg=User scroll down to the 50% position,timestamp=2025-05-06T21:19:54.173598+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=485571875c9162de624eb677858853b7,duration=0
msg=User scroll down to the 50% position,timestamp=2025-05-06T21:20:05.650657+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=485571875c9162de624eb677858853b7,duration=0
msg=User triggered click event,timestamp=2025-05-06T21:20:11.997314+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=485571875c9162de624eb677858853b7,duration=0
msg=User stayed on the page,timestamp=2025-05-06T21:20:17.882929+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=485571875c9162de624eb677858853b7,duration=46
msg=User triggered click event,timestamp=2025-05-06T21:22:01.375912+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=681e6083bf7ae638c4a88300ed0807cb7,duration=0
msg=User stayed on the page,timestamp=2025-05-06T21:46:21.226534+00:00,level_name=INFO,user_ip=172.19.0.1,request_id=681e6083bf7ae638c4a88300ed0807cb7,duration=1468
```

Figure 61: User Behaviour Stats Shown in Dashboard

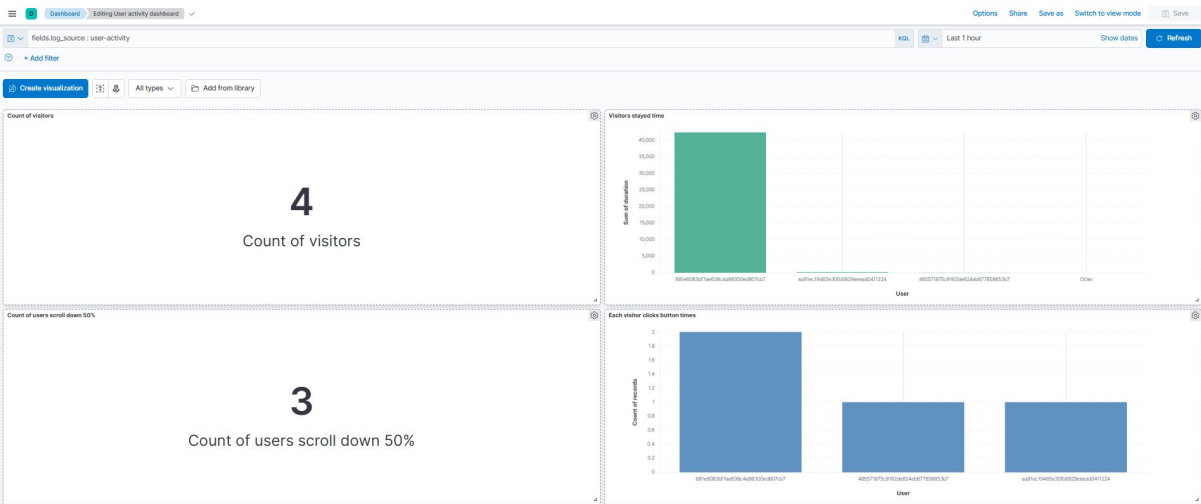


Figure 62: User Behaviour Stats Calculated By Shell Script

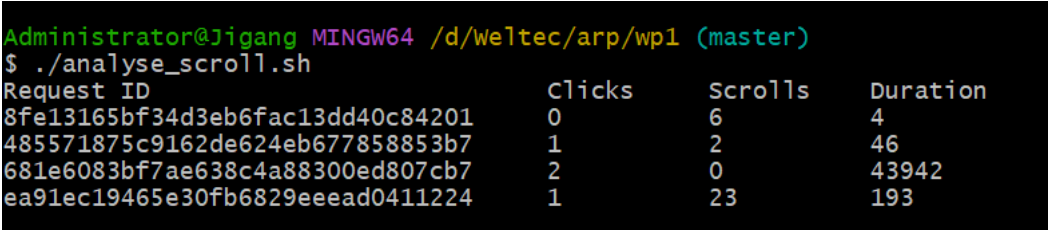
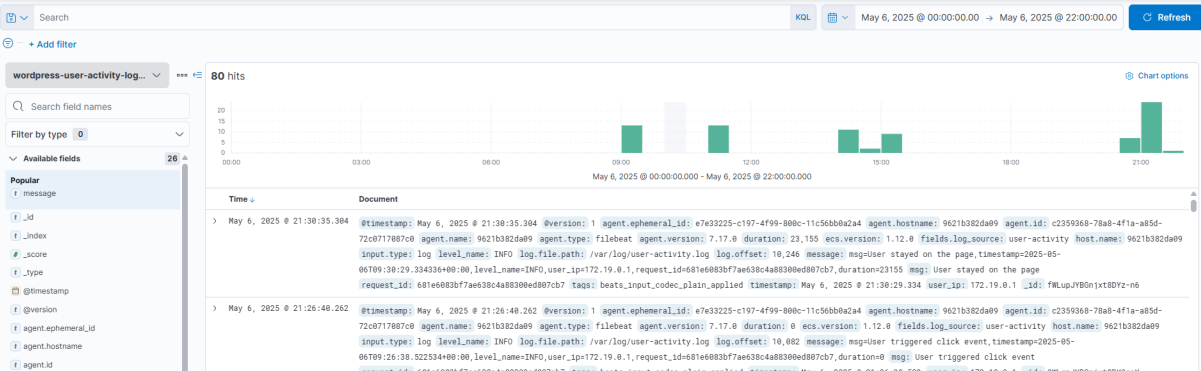


Figure 63: User Behaviour Logs Shown in Kibana



4.1.3.1 Discussion

The user behaviour monitoring experiment demonstrated the effectiveness of ELK Stack in tracking and analysing frontend interactions on a WordPress-based e-commerce site. Key

metrics such as homepage visits, promotional button clicks, time spent on the homepage, and scroll depth were successfully captured by simulating different user sessions. The data visualised in Kibana aligned with the raw logs and manual verification using shell scripts, confirming the reliability of the logging setup. These findings prove how users interact with the system and demonstrate how ELK can help optimise UX and drive data-based marketing decisions.

4.2 Summary

This chapter analysed the ELK Stack's functionality through security alerting, system performance monitoring, and user behaviour analytics in distributed WordPress environments. Security monitoring was validated through a simulated brute-force login attempt, where suspicious behaviour was detected, visualised, and alerted in real time. Metricbeat proved effective in capturing and visualising real-time performance metrics such as CPU, memory, and network usage across multiple WordPress containers. Finally, custom dashboards and log analysis enabled detailed tracking of user interactions, including visit counts, button clicks, scroll depth, and session duration. These results demonstrate that the ELK Stack offers a comprehensive and reliable solution for threat detection, operational visibility, and user behaviour analysis in dynamic web applications.

CHAPTER 5 CONCLUSION AND FUTURE WORK

5.1 Conclusion

This research demonstrated that the ELK Stack can effectively monitor and manage distributed WordPress websites. By implementing a centralised logging and visualisation system, the study addressed key challenges in security alerting, system performance monitoring, and user behaviour tracking. The experiments showed that Metricbeat reliably captured system-level

metrics across multiple WordPress containers, slow SQL queries were effectively ingested and alerted on, and user behaviour data could be analysed in real time through visual dashboards. These findings highlight the ELK Stack's strong capability to support developers and administrators in improving the security, performance and maintainability of modern WordPress-based applications.

5.2 Recommendations and Future Work

Future work could expand on this study by exploring the use of other Beats modules (e.g., Auditbeat) for more comprehensive system monitoring. In addition, integrating machine learning capabilities within the ELK Stack (via Elastic ML) could help automatically detect anomalies in traffic patterns or unusual user behaviour. More complex user interaction scenarios and higher concurrency loads could also be tested to evaluate ELK's scalability. Another valuable direction would be implementing role-based access control within Kibana for better multi-user data governance.

5.3 Limitations

While the study demonstrated the core functionality of ELK in a controlled environment, it was limited to a small number of WordPress instances and simulated user scenarios. The testing environment lacked high traffic or real-world data variety, which could affect the generalizability of results. Additionally, alert thresholds were manually defined, which may not scale well in dynamic production environments. Lastly, the security analysis focused on login attempts, but did not cover deeper intrusion detection or cross-site scripting (XSS) threats.

REFERENCES

- Achar, S. (2021). An Overview of Environmental Scalability and Security in Hybrid Cloud Infrastructure Designs. *Asia Pacific Journal of Energy and Environment*, 8(2), Article 2. <https://doi.org/10.18034/apjee.v8i2.650>
- Ahmed, F., Jahangir, U., Rahim, H., Ali, K., & Agha, D.-S. (2020). Centralized Log Management Using Elasticsearch, Logstash and Kibana. *2020 International Conference on Information Science and Communication Technology (ICISCT)*, 1–7. <https://doi.org/10.1109/ICISCT49550.2020.9080053>
- Elastic Stack. (2025). *What is Elasticsearch? | Elasticsearch Guide [8.17] | Elastic [Learn/Docs/Elasticsearch/Reference/8.17]. What Is Elasticsearch? | Elasticsearch Guide [8.17] | Elastic.* <https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro-what-is-es.html>
- Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, 28(1), 75–105.
- He, S., Zhang, X., He, P., Xu, Y., Li, L., Kang, Y., Ma, M., Wei, Y., Dang, Y., Rajmohan, S., & Lin, Q. (2022). An empirical study of log analysis at Microsoft. *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 1465–1476. <https://doi.org/10.1145/3540250.3558963>
- Jetpack Vs Wordfence: Which Is Better Security Plugin?* (2021, December 28). <https://www.malcare.com/blog/jetpack-vs-wordfence/>

Logstash Introduction | *Logstash Reference [8.17]* | *Elastic*. (2025).
[Learn/Docs/Logstash/Reference/8.17].

<https://www.elastic.co/guide/en/logstash/8.17/introduction.html>

Murphy, D. T., Zibran, M. F., & Eishita, F. Z. (2021). Plugins to Detect Vulnerable Plugins: An Empirical Assessment of the Security Scanner Plugins for WordPress. *2021 IEEE/ACIS 19th International Conference on Software Engineering Research, Management and Applications (SERA)*, 39–44.
<https://doi.org/10.1109/SERA51205.2021.9509274>

Releases – WordPress News. (2025, March 25). <https://wordpress.org/news/category/releases/>

Shah, P. G., & Ayoade, J. (2023). An Empricial Study of Brute Force Attack on Wordpress Website. *2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, 659–662. <https://doi.org/10.1109/ICSSIT55814.2023.10060966>

Sunil, S., Suresh, A., & Hemamalini, V. (2023). Log Based Anomaly Detection: Relation Between The Logs. *2023 International Conference on Networking and Communications (ICNWC)*, 1–5.
<https://doi.org/10.1109/ICNWC57852.2023.10127571>

Svacina, J., Raffety, J., Woodahl, C., Stone, B., Cerny, T., Bures, M., Shin, D., Frajtek, K., & Tisnovsky, P. (2020). On Vulnerability and Security Log analysis: A Systematic Literature Review on Recent Trends. *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, 175–180.
<https://doi.org/10.1145/3400286.3418261>

toddplex. (2016, August 2). Your business doesn't need WordPress. *Unconventional Website Advice*. <https://medium.com/unconventional-website-advice/your-business-doesn-t-need-wordpress-464c69da34a2>

- Wang, Y. (2023). Design of Visual Log Analysis System. *2023 IEEE International Conference on Sensors, Electronics and Computer Engineering (ICSECE)*, 1649–1652. <https://doi.org/10.1109/ICSECE58870.2023.10263397>
- Xu, S., Meng, Z., & Wang, H. (n.d.). *Research and Implementation of Log-Based Anomaly Detection Platform Based on ELK+Kafka*.
- Yang, L., Chen, Z., Bai, Y., Zhang, M., & Yu, J. (2022). Research on Data Processing and Visualization of Simulation System. *Proceedings of the 5th International Conference on Big Data Technologies*, 131–134. <https://doi.org/10.1145/3565291.3565312>
- Yi, K. M., Kyaw, L. Y., & Thandar, P. (2024). Enhancing Security of WordPress Websites built on Virtual Machine Using Cloud Computing Technology. *2024 5th International Conference on Advanced Information Technologies (ICAIT)*, 1–6. <https://doi.org/10.1109/ICAIT65209.2024.10754914>
- Zhu, J., He, S., He, P., Liu, J., & Lyu, M. R. (2023). Loghub: A Large Collection of System Log Datasets for AI-driven Log Analytics. *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*, 355–366. <https://doi.org/10.1109/ISSRE59848.2023.00071>

APPENDICES

This paper's source code and resources are publicly available at the following GitHub repository: <https://github.com/jigangmissyou/arp.git>.