A Review of System Monitoring Architectures Using Prometheus, ELK Stack, and Custom Dashboards

EJIELO OGBUEFI[1], OYEJIDE TIMOTHY ODOFIN[2], ABRAHAM AYODEJI ABAYOMI[3], BOLAJI IYANU ADEKUNLE[4], OLUWADEMILADE ADEREMI AGBOOLA[5], SAMUEL OWOADE[6]
[1]*Mac-Umec Associate limited And NYSC @Enugu, Nigeria*
[2]*Comsoft Ltd, Abuja, Nigeria*
[3]*Hagital Consulting, Lagos State, Nigeria*
[4]*Data Analyst, Federal Ministry of Mines and Steel Development, Nigeria*
[5]*Data Culture, New York, USA*
[6]*Sammich Technologies*

Abstract- ***System monitoring architectures play a critical role in ensuring the operational efficiency, performance, and security of modern IT infrastructures. This paper provides a comprehensive review of three prominent monitoring solutions—Prometheus, the ELK Stack (Elasticsearch, Logstash, and Kibana), and Custom Dashboards—evaluating their features, architecture, use cases, and benefits. Prometheus, an open-source time-series monitoring system, is ideal for large-scale, dynamic environments, particularly those utilizing microservices and cloud-native architectures. The ELK Stack offers a robust solution for log aggregation, search, and visualization, making it effective for security monitoring, log analysis, and performance management. Custom Dashboards, integrated with Prometheus or ELK, facilitate real-time visualization of key performance indicators, improving decision-making and responsiveness. Despite their strengths, these solutions present challenges, including scalability concerns, resource requirements, and integration complexities. The paper concludes with suggestions for future advancements in AI-driven monitoring, cross-platform integration, and automated configurations, aiming to enhance the adaptability and efficiency of monitoring systems in increasingly complex IT environments.***

Indexed Terms- ***System Monitoring, Prometheus, ELK Stack, Custom Dashboards, Real-Time Data Visualization, Log Aggregation***

I. INTRODUCTION

1.1 Background and Importance

System monitoring architectures are critical to the smooth operation of modern IT infrastructure. These architectures enable real-time monitoring of complex systems, helping organizations maintain the performance and reliability of their services [1, 2]. As IT environments grow more intricate with the increasing use of cloud computing, microservices, and distributed systems, monitoring has become even more crucial. It ensures that performance bottlenecks, security vulnerabilities, and operational issues are detected and addressed before they lead to significant problems [3-5]. Without a robust monitoring system, organizations risk system failures, security breaches, and decreased user satisfaction, which can lead to substantial financial and reputational losses. Moreover, efficient monitoring enhances troubleshooting processes, optimizing resource allocation, and facilitating proactive maintenance [6-8].

In today's competitive environment, businesses demand continuous uptime, seamless performance, and rapid issue resolution. Therefore, system monitoring tools are indispensable, ensuring these objectives are met. Monitoring architectures contribute to the agility and reliability of IT environments, enabling teams to make informed, data-driven decisions that enhance operational effectiveness [9, 10]. Effective system monitoring encompasses various tools and techniques. The integration of monitoring solutions such as

Prometheus, the ELK Stack, and custom dashboards is vital for organizations to build an all-encompassing monitoring infrastructure. These tools provide visibility, real-time metrics, and actionable insights that can improve business decision-making processes [11-13].

1.2 Purpose and Scope

This paper aims to provide an in-depth review of system monitoring architectures, focusing specifically on Prometheus, the ELK Stack, and custom dashboards. The purpose is to evaluate these tools based on their functionality, use cases, and effectiveness in modern IT environments. Prometheus and the ELK Stack represent two of the most widely used open-source monitoring and logging solutions, while custom dashboards play a pivotal role in visualizing system data for actionable insights. Together, they form a robust monitoring ecosystem that ensures the availability, performance, and security of infrastructure.

The review will explore how each tool addresses specific monitoring challenges, such as scalability, data aggregation, alerting, and real-time visualization. Prometheus, for example, offers excellent support for time-series data and alerting, while the ELK Stack excels in log management and data analytics. Custom dashboards, on the other hand, are essential for tailoring the monitoring experience to organizational needs, improving the interpretation and accessibility of data.

The scope of this review is to analyze the integration of these technologies, their individual strengths, and their combined potential in achieving a holistic monitoring solution. The paper will also highlight the challenges and limitations associated with these tools, providing a balanced perspective on their deployment in production environments.

1.3 Overview of Methodology

The methodology for this review involves a systematic evaluation of each monitoring solution, based on a set of predefined criteria. These criteria include scalability, ease of integration, data aggregation capabilities, alerting functionality, and the ability to provide real-time insights. The review will analyze scholarly articles, case studies, and real-world applications of Prometheus, the ELK Stack, and custom dashboards, focusing on how these tools have been implemented in production environments. By examining their adoption in various industries, the paper aims to identify best practices, common challenges, and the overall effectiveness of these monitoring architectures.

The evaluation will consider the architecture of each tool, including its components, integration points, and how it contributes to a comprehensive monitoring strategy. Prometheus's time-series database and powerful query language, the ELK Stack's log aggregation and visualization features, and the flexibility of custom dashboards will be compared to determine their strengths and weaknesses in different contexts. Furthermore, the paper will explore the synergies between these tools when integrated, assessing how they can work together to provide a unified monitoring solution that meets the diverse needs of modern IT infrastructure. This methodical approach will ensure that the review is both thorough and objective, providing valuable insights into the state of system monitoring in today's digital landscape.

II. PROMETHEUS AS A MONITORING SOLUTION

2.1 Overview of Prometheus

Prometheus is a widely adopted open-source monitoring and alerting toolkit designed primarily for reliability and scalability. Initially developed by SoundCloud, it has since gained significant popularity in the open-source community due to its unique capabilities tailored for modern, dynamic environments [14]. Prometheus is particularly suited for cloud-native applications and microservices, where traditional monitoring tools may struggle with scalability and flexibility. The core strength of Prometheus lies in its ability to efficiently collect and store time-series data, which is essential for monitoring the performance of distributed systems [15-17].

The system is designed to handle large volumes of metric data, providing visibility into a system's

performance over time. Prometheus collects metrics through a pull model, meaning it periodically queries targets to retrieve data, such as system health, resource usage, and error rates [18-20]. It stores this data in a time-series database, allowing users to query and analyze the historical performance of various system components. Additionally, Prometheus integrates well with other monitoring tools, making it a versatile choice for organizations looking for a comprehensive monitoring solution [21-23].

Prometheus offers an open-source, robust alternative to proprietary solutions, ensuring that organizations can monitor their systems without vendor lock-in. Its scalability, ease of setup, and integration with popular visualization tools like Grafana make it a go-to choice for businesses aiming to implement efficient and cost-effective monitoring systems [24].

2.2 Architecture and Components

Prometheus operates based on a well-defined architecture consisting of several key components that work together to collect, store, and analyze metrics. The core of the architecture is the Prometheus server, which is responsible for collecting data, storing it in a time-series database, and executing queries on that data. Prometheus uses a powerful query language called PromQL (Prometheus Query Language), which allows users to retrieve and manipulate metrics in real-time, making it highly flexible for complex analysis [1, 25].

Another crucial component is exporters, which expose application-specific metrics to Prometheus. These exporters are typically deployed alongside the application or system being monitored, gathering metrics related to CPU usage, memory, disk space, or network activity. Prometheus pulls data from these exporters on a regular basis, ensuring that it is always up to date [15, 26, 27].

Prometheus also integrates with alerting rules, which are user-defined conditions that trigger notifications when certain thresholds are breached. This alerting system is designed to help operations teams respond proactively to issues before they impact system performance. The system includes Alertmanager, which handles the notifications triggered by alerting rules, allowing for custom routing and silencing of alerts based on predefined conditions [28]. The overall architecture ensures that Prometheus can scale horizontally, meaning it can efficiently handle an increasing number of monitored services or system components without performance degradation.

2.3 Use Cases and Benefits

Prometheus excels in several areas of system monitoring, particularly for environments that require high scalability and flexibility. One of its primary use cases is in cloud-native applications, where microservices and containers introduce dynamic and constantly changing workloads [29]. Prometheus is well-suited to monitor such environments due to its ability to collect data from a large number of endpoints and scale horizontally without requiring significant reconfiguration [30, 31].

In addition to cloud-native monitoring, Prometheus is widely used in the monitoring of systems and applications in data centers. It provides deep insights into infrastructure performance, enabling businesses to track and visualize resource usage, system health, and service uptime. It is particularly beneficial in environments where real-time data is crucial for detecting and responding to issues quickly [32]. One of the standout benefits of Prometheus is its cost-effectiveness and ease of use. Being an open-source tool, it eliminates licensing costs associated with proprietary monitoring solutions [33-35]. Furthermore, its robust integration capabilities with other tools, such as Grafana for visualization and Alertmanager for notification management, enhance its utility and make it an ideal choice for organizations looking to deploy a comprehensive monitoring solution. Overall, Prometheus enables organizations to maintain system reliability, ensure high availability, and improve troubleshooting efficiency through data-driven insights [36].

III. ELK STACK FOR MONITORING AND LOGGING

3.1 Overview of ELK Stack

The ELK Stack, comprising Elasticsearch, Logstash, and Kibana, is a powerful suite of open-source tools primarily used for log aggregation, search, and data analysis. Elasticsearch is a distributed search and analytics engine that enables fast, scalable data indexing and querying [37]. Logstash is a data processing pipeline that ingests, transforms, and forwards log data, making it suitable for a wide variety of sources. Kibana provides a user-friendly interface for data visualization and exploration, enabling users to gain valuable insights from the collected logs [38-40].

Together, these components form a unified solution for handling large volumes of log and event data in real-time. Elasticsearch's indexing and querying capabilities allow organizations to search through vast datasets quickly, while Logstash processes and structures data before feeding it into Elasticsearch [41, 42]. Kibana's intuitive dashboards make it easy to visualize and analyze log data, helping teams identify trends, diagnose issues, and monitor system performance. This makes the ELK Stack a popular choice for organizations seeking to improve their monitoring capabilities, particularly in environments with complex, distributed architectures [43-45].

The ELK Stack is particularly effective for use cases that involve large-scale log data, such as IT operations, security monitoring, and performance analysis. It enables real-time collection, analysis, and visualization of logs from diverse systems, giving organizations the ability to detect anomalies, troubleshoot issues, and gain a comprehensive view of their infrastructure [46].

3.2 Architecture and Integration

The ELK Stack operates as a highly integrated system, where each component plays a specialized role in the data pipeline. Logstash, the data processing layer, is responsible for ingesting data from various sources such as system logs, application logs, and metrics from services [47, 48]. It processes the data, applying filters and transformations (e.g., parsing, normalization), and then sends it to Elasticsearch for indexing and storage [49]. Elasticsearch, built on the Apache Lucene library, is a distributed, RESTful search engine optimized for handling large volumes of data at high speed. It indexes data in near real-time, allowing users to query it efficiently and gain insights through aggregation and filtering [37, 50].

On top of Elasticsearch, Kibana provides a web-based interface that allows users to visualize and explore the data stored in Elasticsearch [51, 52]. Kibana enables the creation of interactive dashboards, visualizations (e.g., bar charts, pie charts, time-series graphs), and search queries that make log data accessible and actionable for operational teams [53, 54]. The integration of all three components—Logstash, Elasticsearch, and Kibana—forms a cohesive solution that is capable of handling the end-to-end process of log collection, storage, analysis, and visualization [55, 56].

The modularity of the ELK Stack allows it to be deployed in various configurations, depending on the organization's requirements [57, 58]. For instance, Elasticsearch can be deployed on multiple nodes for horizontal scaling, while Logstash and Kibana can be adjusted based on the data volume and user needs. The stack also integrates well with other tools and systems, further enhancing its flexibility and usefulness in diverse environments [59, 60].

3.3 Applications and Effectiveness

The ELK Stack is particularly effective for several key monitoring use cases, especially those involving log management, security monitoring, and system performance analysis. In the realm of log aggregation, ELK enables organizations to consolidate logs from multiple sources, such as servers, databases, applications, and networking devices, into a single searchable repository. This centralization of log data simplifies troubleshooting, as system administrators can search and correlate logs across various systems to identify the root cause of issues quickly [39, 40].

In security monitoring, the ELK Stack's ability to collect and analyze log data in real-time makes it an invaluable tool for detecting and responding to security threats [59]. By aggregating logs from firewalls, intrusion detection systems, and application logs, it allows security teams to monitor for suspicious

activity, such as unauthorized access attempts or system vulnerabilities, in real-time. Kibana's powerful visualization tools enable security analysts to build custom dashboards that highlight key security metrics and potential threats [50, 61].

Another significant application of the ELK Stack is in performance monitoring. By integrating application and infrastructure logs into the stack, organizations can monitor the health and performance of their systems in real-time. ELK helps track critical metrics such as resource usage, response times, error rates, and user activity. These insights allow IT teams to identify performance bottlenecks, optimize resource allocation, and ensure the smooth operation of business-critical services [39, 62, 63].

The effectiveness of the ELK Stack is evident in its ability to scale with growing log data, handle complex queries, and provide real-time insights. It has become a go-to solution for organizations of all sizes, enabling them to achieve greater visibility and control over their IT infrastructure while enhancing their ability to detect issues proactively and ensure operational continuity [64].

IV. CUSTOM DASHBOARDS FOR REAL-TIME VISUALIZATION

4.1 Definition and Importance

Custom dashboards are highly customizable visual interfaces that display critical metrics, KPIs (Key Performance Indicators), and logs in real-time. These dashboards allow users to monitor and analyze system performance, providing a comprehensive view of the health of applications, services, and infrastructure components [64, 65]. The importance of custom dashboards lies in their ability to tailor the presentation of data according to the specific needs of an organization or team. Rather than relying on default templates or one-size-fits-all solutions, custom dashboards empower users to highlight the most relevant information for their specific monitoring objectives [66, 67].

Custom dashboards are essential for improving decision-making by allowing teams to focus on the most important data points and trends. Whether for IT operations, security monitoring, or business analytics, custom dashboards provide a means to visualize complex data sets in a simple, actionable format [68, 69]. They can be designed to show time-series data, alert statuses, resource utilization, error rates, or user activity, depending on the goals of the monitoring system [70-72]. Their flexibility ensures that all relevant stakeholders—from technical teams to executives—can access the insights they need in a manner that supports swift, data-driven decisions [73]. Additionally, custom dashboards facilitate quicker response times to potential system issues by consolidating diverse data streams in one location. By eliminating the need for teams to dig through different systems or tools, they make it easier to identify trends, spot problems, and take corrective action before issues escalate [74].

4.2 Designing Effective Dashboards

Designing effective dashboards requires a balance between aesthetics and functionality to ensure that they provide clear, actionable insights. Best practices for creating these dashboards include defining a clear objective for what the dashboard should achieve [75, 76]. The first step in dashboard design is identifying the key metrics and data points that will drive decision-making. These should be aligned with the organization's goals and the specific needs of the users, whether it is monitoring application performance, system health, or security events [65, 77].

Another important consideration is data visualization techniques. Visual elements like graphs, charts, and gauges should be used to represent data in a way that is easy to understand at a glance. For example, time-series graphs can display trends in system performance, while pie charts might show resource allocation percentages [78, 79]. The dashboard should avoid information overload by focusing on the most critical metrics and organizing them logically, such as grouping related metrics together. Dashboards should also allow users to drill down into the data for more detailed analysis when necessary [80, 81].

Additionally, effective dashboards should prioritize real-time data display and interactivity. They should automatically update in real-time, providing users with the most current information about system

performance and alert statuses. Interactivity allows users to filter, zoom, or click on specific data points to obtain a deeper understanding of trends or anomalies [66, 82]. Another key element is alert integration—critical thresholds for specific metrics should trigger visual cues such as color changes or pop-up alerts, helping users identify problems quickly. In summary, a well-designed custom dashboard simplifies decision-making by making complex data easily digestible and actionable [83].

4.3 Integration with Prometheus and ELK

Custom dashboards can significantly enhance the monitoring experience when integrated with tools like Prometheus and the ELK Stack. Both Prometheus and the ELK Stack are capable of collecting vast amounts of data, but their true power is unlocked when that data is visualized and acted upon in real-time through custom dashboards [1, 84]. For example, Prometheus can be integrated with visualization tools like Grafana to create custom dashboards that display time-series data collected from various system components [85, 86]. Grafana's rich visualization features allow users to create highly customized graphs, tables, and alerts that can be tailored to suit their monitoring needs. These dashboards can display metrics such as CPU usage, memory consumption, request rates, and error logs, all pulled from Prometheus [87-89]. Additionally, Prometheus's query language (PromQL) enables users to filter and aggregate the data in ways that align with business needs, providing deeper insights into system behavior and performance [72, 90].

Similarly, the ELK Stack integrates seamlessly with custom dashboards, especially through Kibana. Kibana allows users to build interactive visualizations from the log data stored in Elasticsearch, providing a clear and accessible view of log trends, system events, and security incidents [91, 92]. Custom dashboards in Kibana can be designed to display logs in real-time, highlight error patterns, or visualize system activity over a defined period. Integrating custom dashboards with ELK allows users to correlate log data with performance metrics, gaining a holistic view of system behavior [93-95].

The integration of Prometheus and the ELK Stack with custom dashboards enhances the monitoring process by offering real-time data visualizations, historical trends, and alerting mechanisms [96, 97]. This synergy allows teams to monitor both system performance and log data side by side, empowering them to make informed, proactive decisions. With the ability to customize data views, these dashboards ensure that users only see the most relevant information in a format that supports effective monitoring and troubleshooting [98-100].

V. CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

This review has examined three prominent monitoring architectures—Prometheus, the ELK Stack, and Custom Dashboards—highlighting their distinct advantages and areas of application. Prometheus stands out for its efficient time-series data collection and powerful query language, making it ideal for monitoring dynamic and cloud-native environments. It excels in scalability, especially for applications running in microservices architectures. The ELK Stack, with its robust log aggregation, search, and visualization capabilities, provides a comprehensive solution for log management, security monitoring, and performance analysis. By enabling real-time data collection and offering powerful search and aggregation features, it simplifies the detection of system anomalies and security threats. Finally, custom dashboards serve as a crucial component for real-time visualization and decision-making, allowing users to tailor their monitoring environment to display the most relevant metrics and trends. When integrated with Prometheus or the ELK Stack, custom dashboards further enhance the ability to monitor system health, identify potential issues, and respond proactively.

While Prometheus, the ELK Stack, and custom dashboards provide robust monitoring solutions, there are challenges and limitations associated with their implementation. One of the primary challenges for Prometheus is its reliance on a pull-based model, which can lead to issues in monitoring services with very high-frequency data or services that are behind firewalls or inaccessible. Additionally, Prometheus's

storage system, though highly scalable, can become expensive for large datasets, particularly when data retention periods are long. The ELK Stack, on the other hand, can experience performance degradation when managing large volumes of log data, especially in real-time environments. The need for substantial hardware resources to handle massive log data volumes and the complexity of configuring and maintaining Elasticsearch clusters can be significant obstacles for smaller organizations or teams with limited resources.

Custom dashboards, while highly flexible, can become complex to manage and maintain if not designed with scalability in mind. Additionally, creating dashboards that efficiently integrate data from diverse sources—such as Prometheus and the ELK Stack—requires careful design and an understanding of how data flows through these systems. Poorly designed dashboards can lead to information overload, with too many metrics being displayed at once or visual elements that do not convey meaningful insights. These challenges must be addressed to ensure the effective use of monitoring architectures.

5.2 Future Directions

As the need for real-time data insights and system reliability grows, future research and improvements in monitoring technologies will likely focus on several key areas. One potential direction is the further integration of artificial intelligence (AI) and machine learning (ML) algorithms within monitoring solutions. AI could be used to predict potential system failures before they occur by analyzing trends and anomalies in data, enabling more proactive responses. Moreover, advancements in data storage and compression methods could reduce the cost and resource burden associated with maintaining large-scale time-series and log data, making monitoring systems more accessible for organizations of all sizes.

Another area for future research is cross-platform integration, as organizations increasingly use hybrid and multi-cloud environments. Developing seamless integration strategies between Prometheus, the ELK Stack, and other monitoring tools will be crucial for ensuring consistency in monitoring across different infrastructures. Finally, as DevOps and Continuous Integration/Continuous Deployment (CI/CD) practices evolve, there will likely be a growing focus on automated monitoring configurations that can adjust based on real-time changes in system architecture. This will allow monitoring systems to be more adaptive and responsive to the dynamic nature of modern IT environments.

REFERENCES

[1] M. Chakraborty and A. P. Kundan, "Architecture of a modern monitoring system," in *Monitoring Cloud-Native Applications: Lead Agile Operations Confidently Using Open Source Software*: Springer, 2021, pp. 55-96.

[2] O. Cabrera, M. Oriol, X. Franch, and J. Marco, "A context-aware monitoring architecture for supporting system adaptation and reconfiguration," *Computing,* vol. 103, no. 8, pp. 1621-1655, 2021.

[3] J. Povedano-Molina, J. M. Lopez-Vega, J. M. Lopez-Soler, A. Corradi, and L. Foschini, "DARGOS: A highly adaptable and scalable monitoring architecture for multi-tenant Clouds," *Future Generation Computer Systems,* vol. 29, no. 8, pp. 2041-2056, 2013.

[4] M. A. Serhani, M. El Menshawy, and A. Benharref, "SME2EM: Smart mobile end-to-end monitoring architecture for life-long diseases," *Computers in Biology and Medicine,* vol. 68, pp. 137-154, 2016.

[5] O. S. Albahri *et al.*, "Systematic review of real-time remote health monitoring system in triage and priority-based sensor technology: Taxonomy, open challenges, motivation and recommendations," *Journal of medical systems,* vol. 42, pp. 1-27, 2018.

[6] N. Rozanski and E. Woods, *Software systems architecture: working with stakeholders using viewpoints and perspectives*. Addison-Wesley, 2012.

[7] S. Karnouskos, A. W. Colombo, F. Jammes, J. Delsing, and T. Bangemann, "Towards an architecture for service-oriented process monitoring and control," in *IECON 2010-36th Annual Conference on IEEE Industrial*

Electronics Society, 2010: IEEE, pp. 1385-1391.

[8] M. Andreolini, M. Colajanni, and M. Pietri, "A scalable architecture for real-time monitoring of large information systems," in *2012 Second Symposium on Network Cloud Computing and Applications*, 2012: IEEE, pp. 143-150.

[9] V. M. Tovarnitchi, "Cloud-based architectures for environment monitoring," in *2017 21st International Conference on Control Systems and Computer Science (CSCS)*, 2017: IEEE, pp. 708-714.

[10] A. Alamri, W. S. Ansari, M. M. Hassan, M. S. Hossain, A. Alelaiwi, and M. A. Hossain, "A survey on sensor-cloud: architecture, applications, and approaches," *International Journal of Distributed Sensor Networks,* vol. 9, no. 2, p. 917923, 2013.

[11] P. Oborski, "Integrated monitoring system of production processes," *Management and Production Engineering Review,* no. 4, 2016.

[12] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, "Cloud monitoring: A survey," *Computer Networks,* vol. 57, no. 9, pp. 2093-2115, 2013.

[13] S. Fang *et al.*, "An integrated system for regional environmental monitoring and management based on internet of things," *IEEE Transactions on Industrial Informatics,* vol. 10, no. 2, pp. 1596-1605, 2014.

[14] N. Sabharwal and P. Pandey, *Monitoring Microservices and Containerized Applications: Deployment, Configuration, and Best Practices for Prometheus and Alert Manager*. Springer, 2020.

[15] J. Bastos and P. Araújo, *Hands-On Infrastructure Monitoring with Prometheus: Implement and scale queries, dashboards, and alerting across machines and containers*. Packt Publishing Ltd, 2019.

[16] B. Manate, F. Fortiş, and P. Moore, "Applying the Prometheus methodology for an Internet of Things architecture," in *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, 2014: IEEE, pp. 435-442.

[17] M. Holopainen, "Monitoring container environment with prometheus and grafana," 2021.

[18] T. Leppänen, "Data visualization and monitoring with Grafana and Prometheus," 2021.

[19] N. Sukhija and E. Bautista, "Towards a framework for monitoring and analyzing high performance computing environments using kubernetes and prometheus," in *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 2019: IEEE, pp. 257-262.

[20] J. Turnbull, *Monitoring with Prometheus*. Turnbull Press, 2018.

[21] J. Sirviö, "Monitoring of a Cloud-Based IT Infrastructure," 2021.

[22] O. Mart, C. Negru, F. Pop, and A. Castiglione, "Observability in kubernetes cluster: Automatic anomalies detection using prometheus," in *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2020: IEEE, pp. 565-570.

[23] N. Sukhija *et al.*, "Event management and monitoring framework for HPC environments using ServiceNow and Prometheus," in *Proceedings of the 12th international conference on management of digital ecosystems*, 2020, pp. 149-156.

[24] M. Miglierina, "Monitoring modern distributed software applications: challenges and solutions," 2017.

[25] S. Sivakumar, "Performance Engineering for Hybrid Multi-Cloud Architectures," ed: ResearchGate, 2021.

[26] R. Boncea, A. Zamfiroiu, and I. Bacivarov, "A scalable architecture for automated monitoring of microservices," *Academy of Economic*

Studies. Economy Informatics, vol. 18, no. 1, pp. 13-22, 2018.

[27] N. Kigo, "An agent based system for monitoring and measuring performance of public cloud applications for users," University of Nairobi, 2016.

[28] C. Pais, J. Carrasco, D. L. Martell, A. Weintraub, and D. L. Woodruff, "Cell2Fire: A cell-based forest fire growth model to support strategic landscape management planning," *Frontiers in Forests and Global Change,* vol. 4, p. 692706, 2021.

[29] L. A. O. Confidently, M. Chakraborty, and A. P. Kundan, "Monitoring Cloud-Native Applications," 2021.

[30] T. V. Kumar, "Event-Driven App Design for High-Concurrency Microservices," 2018.

[31] P. Amogh, G. Veeramachaneni, A. K. Rangisetti, B. R. Tamma, and A. A. Franklin, "A cloud native solution for dynamic auto scaling of MME in LTE," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2017: IEEE, pp. 1-7.

[32] N. Marie-Magdelaine, "Observability and resources managements in cloud-native environnements," Université de Bordeaux, 2021.

[33] R. Alt, R. Beck, and M. T. Smits, "FinTech and the transformation of the financial industry," vol. 28, ed: Springer, 2018, pp. 235-243.

[34] E. Hofmann, U. M. Strewe, and N. Bosia, *Supply chain finance and blockchain technology: the case of reverse securitisation*. Springer, 2017.

[35] Y. Tao, X. Wang, X. Xu, and G. Liu, "Container-as-a-service architecture for business workflow," *International Journal of Simulation and Process Modelling,* vol. 13, no. 2, pp. 102-115, 2018.

[36] J.-C. Guédon *et al.*, "Future of scholarly publishing and scholarly communication: report of the Expert Group to the European Commission," 2019.

[37] R. Nishant, "Visual logging framework using ELK stack," 2017.

[38] T. Prakash, M. Kakkar, and K. Patel, "Geo-identification of web users through logs using ELK stack," in *2016 6th international conference-cloud system and big data engineering (confluence)*, 2016: IEEE, pp. 606-610.

[39] S. Dharur and K. Swaminathan, "Efficient surveillance and monitoring using the ELK stack for IoT powered Smart Buildings," in *2018 2nd international conference on inventive systems and control (ICISC)*, 2018: IEEE, pp. 700-705.

[40] A. F. Rochim, M. A. Aziz, and A. Fauzi, "Design log management system of computer network devices infrastructures based on elk stack," in *2019 International Conference on Electrical Engineering and Computer Science (ICECOS)*, 2019: IEEE, pp. 338-342.

[41] E. Almeida, I. Koga, M. Santana, P. Guimaraes, and T. Eklin, "Exploratory study of the ELK stack for meteorological observation system data analysis," *J Comput Interdiscip Sci,* vol. 8, pp. 131-142, 2017.

[42] M. A. STAN, "Automation of Log Analysis Using the Hunting ELK Stack," *Romanian Cyber Security Journal,* vol. 3, no. 1, 2021.

[43] C.-T. Yang, E. Kristiani, Y.-T. Wang, G. Min, C.-H. Lai, and W.-J. Jiang, "On construction of a network log management system using ELK Stack with Ceph," *The Journal of Supercomputing,* vol. 76, pp. 6344-6360, 2020.

[44] A. Tiwari and D. Mane, "Application Performance Monitoring Using Log File on ELK Stack," ed: IRJET, 2020.

[45] A. Mikula, D. Adamová, M. Adam, J. Chudoba, and J. Švec, "Grid site monitoring and log processing using elk," in *CEUR Workshop Proceedings*, 2016.

[46] R. Appleyard and J. Adams, "Using the ELK stack for CASTOR application logging at RAL," in *International Symposium on Grids and Clouds (ISGC)*, 2015, vol. 15, no. 20.

[47] B. R. Cherukuri, "Microservices and containerization: Accelerating web development cycles," 2020.

[48] S. P. R. Asaithambi, R. Venkatraman, and S. Venkatraman, "MOBDA: Microservice-oriented big data architecture for smart city transport systems," *Big Data and Cognitive Computing,* vol. 4, no. 3, p. 17, 2020.

[49] H. U. Rahman, R. U. Khan, and A. Ali, "Programming and pre-processing systems for Big data storage and visualization," in *Handbook of Research on Big Data Storage and Visualization Techniques*: IGI Global, 2018, pp. 228-253.

[50] G. S. Sachdeva, "Practical ELK Stack," *Practical ELK Stack. Apress,* 2017.

[51] Z. Zhong and R. Buyya, "A cost-efficient container orchestration strategy in kubernetes-based cloud computing infrastructures with heterogeneous resources," *ACM Transactions on Internet Technology (TOIT),* vol. 20, no. 2, pp. 1-24, 2020.

[52] S. Chinamanagonda, "Enhancing CI/CD Pipelines with Advanced Automation-Continuous integration and delivery becoming mainstream," *Journal of Innovative Technologies,* vol. 3, no. 1, 2020.

[53] P. S. Patchamatla and I. O. Owolabi, "Integrating serverless computing and kubernetes in OpenStack for dynamic AI workflow optimization," *International Journal of Multidisciplinary Research in Science, Engineering and Technology,* vol. 1, p. 12, 2020.

[54] J. Okwuibe, J. Haavisto, E. Harjula, I. Ahmad, and M. Ylianttila, "SDN enhanced resource orchestration of containerized edge applications for industrial IoT," *IEEE Access,* vol. 8, pp. 229117-229131, 2020.

[55] B. Azarmi, *Learning Kibana 5.0*. Packt Publishing Ltd, 2017.

[56] M. Bajer, "Building an IoT data hub with Elasticsearch, Logstash and Kibana," in *2017 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, 2017: IEEE, pp. 63-68.

[57] P. Raj, A. Raman, P. Raj, and A. Raman, "Automated multi-cloud operations and container orchestration," *Software-Defined Cloud Centers: Operational and Management Technologies and Tools,* pp. 185-218, 2018.

[58] S. V. Gogouvitis, H. Mueller, S. Premnadh, A. Seitz, and B. Bruegge, "Seamless computing in industrial systems using container orchestration," *Future Generation Computer Systems,* vol. 109, pp. 678-688, 2020.

[59] M. Ruokola, "Centralized log management," 2017.

[60] S. Bagnasco, D. Berzano, A. Guarise, S. Lusso, M. Masera, and S. Vallero, "Towards Monitoring-as-a-service for Scientific Computing Cloud applications using the ElasticSearch ecosystem," in *Journal of Physics: Conference Series*, 2015, vol. 664, no. 2: IOP Publishing, p. 022040.

[61] P. Delgado, "Developing an adaptive threat hunting solution: The elasticsearch stack," 2018.

[62] M. Hanikat, "Centralized log management for complex computer networks," ed, 2018.

[63] A. H. Almuraikhi, "Network Security Central Monitoring System," Hamad Bin Khalifa University (Qatar), 2020.

[64] V.-A. Zamfir, M. Carabas, C. Carabas, and N. Tapus, "Systems monitoring and big data analysis using the elasticsearch system," in *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*, 2019: IEEE, pp. 188-193.

[65] W. W. Eckerson, *Performance dashboards: measuring, monitoring, and managing your business*. John Wiley & Sons, 2010.

[66] Z. Khan, "Custom collaboration of data visualization and dashboard optimization; data simplicity and accessibility," 2021.

[67] S. Kumar and M. Belwal, "Performance dashboard: Cutting-edge business intelligence and data visualization," in *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)*, 2017: IEEE, pp. 1201-1207.

[68] S. van Bekhoven and M. Haisma, "THE DESIGN AND IMPLEMENTATION OF A KEY PERFORMANCE INDICATOR DASHBOARD."

[69] H. S. AHMAD KAMIL, "Business Intelligence Dashboard Design: KPI Reporting Portal through SM-ART Dashboard (SM-ART KPI)," 2012.

[70] Q. Zhu, *Game-theoretic methods for security and resilience in cyber-physical systems*. University of Illinois at Urbana-Champaign, 2013.

[71] K. Päivärinta, "Design and Implementation of Centralized APIs Platform and Application Portal," 2019.

[72] A. Laisi, "A reference architecture for event-driven microservice systems in the public cloud," 2019.

[73] M. M. Travessa, "Development of a Visual Analytical Solution: Dashboards Creation for Customer Care Area," Universidade NOVA de Lisboa (Portugal), 2018.

[74] P. Russom, D. Stodder, and F. Halper, "Real-time data, BI, and analytics," *Accelerating Business to Leverage Customer Relations, Competitiveness, and Insights. TDWI best practices report, fourth quarter,* pp. 5-25, 2014.

[75] C. K. Rudrabhatla, "Impacts of decomposition techniques on performance and latency of microservices," *International Journal of Advanced Computer Science and Applications,* vol. 11, no. 8, 2020.

[76] D. Taibi and K. Systä, "From monolithic systems to microservices: A decomposition framework based on process mining," in *International Conference on Cloud Computing and Services Science*, 2019: SciTePress, pp. 153-164.

[77] M. K. Allio, "Strategic dashboards: designing and deploying them to improve implementation," *Strategy & Leadership,* vol. 40, no. 5, pp. 24-31, 2012.

[78] A. Cahyadi, "Beyond functionality and a user interface: a design thinking perspective on the design of dashboards," Swinburne, 2016.

[79] M. Keulen, "Defining Dashboards."

[80] G. W. Young and R. Kitchin, "Creating design guidelines for building city dashboards from a user's perspectives," *International Journal of Human-Computer Studies,* vol. 140, p. 102429, 2020.

[81] A. Dhareshwar, "Make it easy to make analyses look good Support for information dashboard design," 2018.

[82] V. Munford, "Designing a centralized faculty performance dashboard: Optimizing feedback and scholarly data reporting," 2021.

[83] A. Cîmpan, "Applying design system in cybersecurity dashboard development," ETSI_Informatica, 2019.

[84] E. Persson, "Development of a prototype framework for monitoring application events," ed, 2020.

[85] J. Fritzsch, J. Bogner, S. Wagner, and A. Zimmermann, "Microservices migration in industry: intentions, strategies, and challenges," in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2019: IEEE, pp. 481-490.

[86] E. Hatami-Alamdari and Z. Etzioni, "Monolithic architecture vs. multi-layered cloud-based architecture in the CRM application domain," 2019.

[87] J. Watada, A. Roy, R. Kadikar, H. Pham, and B. Xu, "Emerging trends, techniques and open issues of containerization: A review," *IEEE Access,* vol. 7, pp. 152443-152472, 2019.

[88] M. Turis, "Domain-driven design with architectural patterns," Masarykova univerzita, Fakulta informatiky, 2019.

[89] V. Desai, Y. Koladia, and S. Pansambal, "Microservices: architecture and technologies," *Int. J. Res. Appl. Sci. Eng. Technol,* vol. 8, no. 10, pp. 679-686, 2020.

[90] G. Liu, B. Huang, Z. Liang, M. Qin, H. Zhou, and Z. Li, "Microservices: architecture, container, and challenges," in *2020 IEEE 20th international conference on software quality, reliability and security companion (QRS-C)*, 2020: IEEE, pp. 629-635.

[91] P. R. Chelliah, S. Naithani, and S. Singh, *Practical Site Reliability Engineering: Automate the process of designing, developing, and delivering highly reliable apps and services with SRE*. Packt Publishing Ltd, 2018.

[92] P. S. Kocher, *Microservices and containers*. Addison-Wesley Professional, 2018.

[93] K. Huang and P. Jumde, *Learn Kubernetes Security: Securely orchestrate, scale, and manage your microservices in Kubernetes deployments*. Packt Publishing Ltd, 2020.

[94] D. Containers, "Studying Dependency Updates and a Framework for Multi-Versioning in," University of Alberta, 2020.

[95] P. W. Khakame, "Development of a scalable microservice architecture for web services using os-level virtualization," University of Nairobi, 2016.

[96] E. Using Jakarta, E. MicroProfile, and V. Nair, "Practical Domain-Driven Design in Enterprise Java."

[97] C. Buenosvinos, C. Soronellas, and K. Akbary, *Domain-Driven Design in PHP*. Packt Publishing Ltd, 2017.

[98] V. Vernon, *Domain-driven design distilled*. Addison-Wesley Professional, 2016.

[99] B. Hippchen, M. Schneider, I. Landerer, P. Giessler, S. Abeck, and L. Lavazza, "Methodology for splitting business capabilities into a microservice architecture: Design and maintenance using a domain-driven approach," in *The Fifth International Conference on Advances and Trends in Software, Valencia, Spain*, 2019.

[100] B. Hippchen, M. Schneider, P. Giessler, and S. Abeck, "Systematic application of domain-driven design for a business-driven microservice architecture," *International Journal on Advances in Software Volume 12, Number 3 & 4, 2019,* 2019.