JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

A Survey on Deploying Prometheus and Grafana for Application Monitoring

Abhijeet Dhane[1], Atharva Joshi[2], Chinmay Borgaonkar[3], Manas Deshpande[4], Prof. Pravin Patil[5]
Department of Computer Engineering,
Pune Institute of Computer Technology, Pune, India.

Abstract : Today's age of increased digital footprint demands software applications to be robust and efficient like never before, leading to the development of comprehensive application monitoring to be a prominent directive of the moment ensuring elegant performance with an ability to address and understand problems faster. Our literature review dives into the domain of application monitoring, focusing on the incorporation of well-known open-source tools Prometheus and Grafana, to be precise.

Prometheus being a software suite for monitoring and alerting enables collection and analysis of metrics from applications and their respective infrastructures, when used along with Grafana, a diverse open-source platform which offers visually appealing and comprehensible dashboards to help visualize and interpret data, seems to be a promising duo to solve the stated problem.

After thorough review of state of the art through existing literatures, this paper explores the core concepts, architectures, and methodologies needed to leverage offerings of Prometheus and Grafana. It studies their core strengths and explores the abilities when used them together for real-time application monitoring. In addition, the paper explores case studies and practical implementations, highlighting the varied range of applications that utilize these tools in various industries.

***IndexTerms* - Prometheus, Grafana, application monitoring, time series database, visualization tool, case studies.**

I. INTRODUCTION

Cloud computing is an emerging model of enterprise computing. A cloud computing platform shares extensive infrastructure, data storage, and software with each other to form a vast resource pool that enables users to obtain computing power, storage space, and information services as needed. You need to monitor your servers for resources, performance, and errors, as well as monitor the applications they serve. Application monitoring is basically a preventative measure to help you catch any problems before they cause any serious problems that affect your productivity and your customer. Application monitoring is the process of continuously scanning the servers on a designated network and scanning the network for any failures or anomalies detected by the server monitoring software. This survey delves into the field of application monitoring, a domain that is constantly evolving to meet the challenges posed by complex, dynamic, and highly distributed applications. Traditional monitoring approaches are often insufficient to capture the complexity of modern applications that span multiple infrastructures, use microservices architectures, and use cloud technologies. In response to these challenges, the open source monitoring ecosystem has seen the rise of innovative solutions, among which Prometheus and Grafana have gained significant attention.

There is an increased inclination of devops engineers towards Prometheus and Grafana as a primary duo to address the processing task of time series data. Storage backend and analytics along with visualization interface are the core offerings of Prometheus and Grafana respectively. Fact of being open-source and having a friendly learning curve are the prominent reasons of the combination for becoming increasingly popular. Both are cloud-based and thereby highly scalable, hence can be used to monitor all sized applications and can be deployed on Kubernetes or other cloud platforms. Stability and reliability being a major concern of cloud platforms can be assured by a comprehensive, smart, and effective monitoring combination of Prometheus and Grafana.

II. APPLICATION MONITORING

Application monitoring refers to the process of monitoring, analyzing, and controlling the performance, availability, and overall behavior of software applications. It plays a key role in ensuring that applications run efficiently, reliably, and securely. Monitoring helps identify performance bottlenecks, detect errors, and ensure timely resolution of issues, thereby increasing user experience and satisfaction. Monitoring provides a real-time overview of system behavior and enables early detection of anomalies and potential security threats. By analyzing performance metrics, monitoring enables proactive capacity planning and ensures systems can handle varying workloads. Additionally, it helps with compliance management by monitoring security measures and ensuring that applications adhere to regulatory standards. Modern applications are increasingly complex, distributed, and dynamic. This makes them difficult to track. Some of the challenges in monitoring modern applications include:

2.1 Microservices

Microservices-based applications consist of many small independent services. This can make it difficult to track dependencies between services and identify the root cause of problems.

2.2 Containers

Containerized applications can be dynamically scaled up and down. This can make it difficult to collect and analyze metrics from containerized applications.

2.3 Cloud-native Applications

Cloud-native applications are often deployed in multiple cloud environments. This can make it difficult to get a full view of the performance and health of cloud-native applications.

In short, application monitoring is essential to maintaining the health and functionality of modern software applications. It addresses the challenges presented by complex architectures such as microservices and containers, ensuring that applications run smoothly and securely. Through continuous monitoring, organizations can increase user satisfaction, optimize performance, and protect their systems from potential threats and failures.

III. OVERVIEW OF PROMETHEUS AND GRAFANA

3.1 Prometheus

Prometheus is an open source toolkit for system monitoring and awakening that is utilized as free software for event monitoring and waking. Either directly or through a central drive gateway for temporary work, Prometheus extracts criteria from instrumented jobs. It keeps all of the scraped samples locally and applies rules to this data in order to generate warnings or total and record new time series from the data. Prometheus' primary characteristics are:

[1] A multidimensional data model that uses key/value pairs and metrics names to link time series data.
[2] One versatile query language to take advantage of this dimension is PromQL.
[3] Individual nodes are independent; there is no dependency on a distributed storage.
[4] A pull model over HTTP is used to collect time series data.

Several tools make up a typical Prometheus monitoring plat-form:

3.1.1 Prometheus Server

The main architectural component is the Prometheus server. The Golang writing is excellent, and it is tested in combat. It carries out the subsequent duties:

[1] Use service discovery to find targets.
[2] Extract the targets' requirements. The objectives could be operations, motivate exporters or gateway.
[3] Save the criteria as a sequence of data.
[4] Use regulations to establish new standards and encourage prudence.
[5] Provides time-series data through an HTTP API for future usage, like data exporting and visualization.

3.1.2 Client Libraries

You must add instrumentation to their law using one of the Prometheus client libraries before you may cover your services. The Prometheus metric kinds are applied in these. There are four primary metric types available in the Prometheus user library. The only places these can currently be found are in the line protocol and client libraries (to allow APIs accustomed to the operation of the particular types). The Prometheus server flattens all data into untyped time series and does not yet utilize the type information. The operation should be written in a language compatible with a Prometheus customer library. With the help of an HTTP endpoint on the case of your action, you can establish and reveal internal criteria.

3.1.3 Push Gateway

For batch and deciduous jobs to expose their criteria to Prometheus, there is a Prometheus Push Gateway. These jobs could not last long enough to be eliminated, so they could instead push their requirements through a Push Gateway. Prometheus is also exposed to these criteria through the Push Gateway.

3.1.4 Exporters

Many libraries and servers are available to assist in exporting third-party system criteria as Prometheuscriteria. When it is not possible to directly instrument a system using Prometheus criteria (for example, HAProxy or Linux system stats), this is helpful.

3.1.5 Alertmanager

Similar to the Prometheus server, the Alertmanager manages alerts sent by customer operations. It handles grouping, deduplication, and routing to the appropriate receiver integration, much like OpsGenie, PagerDuty, or dispatch. It also handles the suppression and inhibition of warnings.

3.2 What are Metrics?

In common parlance, metrics are numerical measurements. A time series is used to track changes over time. Depending on the application, different metrics are being sought after by users. These could include the number of connections or active queries in the database, the web server's request times, etc. Because they enable you to comprehend the reasons behind your application's performance, metrics are crucial. Assume that the web application you are using is lagging. You will need some information to figure out what is happening with your app. The application might lag, for instance, if there are a lot of requests. If you know how many requests there are, you can figure out what is causing it and add more servers to handle the traffic. Prometheus offers a proprietary query language called PromQL (Prometheus Query Language) that lets users pick and combine data. Because PromQL is specifically designed to operate in the Time-Series database convention, it offers time-related query functions. A few instances are the range vector, the instantaneous vector, and the rate() function, whichcan return multiple samples for each time series that is queried. PromQL components in Prometheus revolve around four distinct categories of metrics. They are as follows:

3.2.1 Counter

The counter is among the simplest metrics. It is helpful in tracking and assessing values that are not yet increasing.

3.2.2 Gauge

Gauges are used to measure values that rise and fall.This covers the quantity of simultaneous requests and the memory usage right now. A metric usually takes the form of a single numerical value.

3.2.3 Summary

Following the sampling of the observations, the summary shows the total number of observations as well as the sum of the observed values. Furthermore, it calculates variable quantiles within a sliding time interval.

3.2.4 Histogram

The sum of all observed values and counts in the segments is typically displayed on a histogram. It is possible to modify the metric computation to meet your needs. Keep in mind that there is a big difference between summaries and histograms depending on where and how statistical quantiles are calculated.

3.3 Grafana

Users can view their data through charts and graphs that are combined into a single dashboard (or multiple dashboards!)for simpler interpretation and comprehension using Grafana, an open source interactive data visualization platform created by Grafana Labs. Additionally, regardless of where your data is kept—on traditional server environments, Kubernetes clusters, different cloud services, etc.—you can query and set up alerts on your metrics and information. After that, you will be able to examine the data more readily, spot patterns and irregularities, and eventually improve the effectiveness of your operations. Grafana was founded on the ideas of openness and the notion that information ought to be accessible to everyone in the company, not just a select few. As a result, teams are encouraged to be more transparent, creative, and cooperative by creating an environment where data is easily accessible and utilized by anyone who needs it.

3.3.1 Dashboards

Data gathered from multiple sources takes on new significance thanks to Grafana dashboards. Collaborating and further exploring the data and its implications can be facilitated by sharing these dashboards with other teams and team members. Using sophisticated querying and transformation tools, create dashboards that are unique for you and your team and modify them to produce the visualizations you desire. In order to identify the root cause of an incident or unexpected system behavior as soon as possible, it is essential to comprehend all pertinent data and data relationships. In order to swiftly identify and address the source of the issue, Grafana facilitates the smooth visualization and transfer of data between teams and team members.

3.3.1.1 Key Properties

[1] Dashboards: use graphs, heatmaps, geomaps, histograms, and other visualizations to see your data however you would like.
[2] Plug-ins: Connect to current data sources with dashboard plug-ins to render your data in real-time on an intuitive API; no data migration is necessary. Additionally, plugins for data sources can be made to retrieve metrics from any custom API.
[3] Alerts: All of your alerts can be created, consolidated, and managed from a single user interface.
[4] Transformations: Rename, condense, merge, compute across queries and data sources.
[5] Remarks: Utilize rich events from various data sources to add annotations to graphs.
[6]] Panel Editor: A standardized user interface for setting up and personalizing your panels.

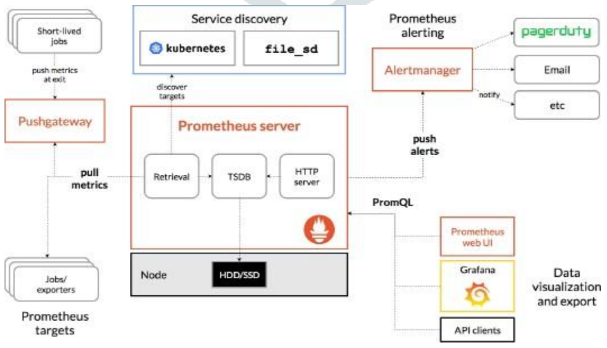

Figure 1 Prometheus and Grafana Architecture

IV. INTEGRATION WITH OTHER TOOLS

4.1 Integration with Container Orchestration Platforms

To monitor containerized applications, Prometheus and Grafana can be connected with container orchestration technologies like Docker Swarm and Kubernetes. Such incorporation may be accomplished by a variety of techniques, including:

4.1.1 Making use of Kubernetes API

Prometheus can scrape the Kubernetes API to obtain the Kubernetes API and collect Kubernetes metrics nodes, pods, and other sources. Grafana is useful for build alerts and dashboards with Kubernetes metrics gathered through Prometheus.

4.1.2 Using Helm charts

Helm charts can be utilized by Prometheus running on Kubernetes with Grafana. Helm charts also offer several pre-made dashboards and notifications for keeping an eye on Kubernetes applications.

4.1.3 Using Docker Swarm Plug-ins

Through the usage of Docker Swarm plugins, Prometheus and Grafana can be integrated with Docker Swarm in a variety of ways. These plugins allow you to install Prometheus and Grafana on Docker Swarm and scrape metrics from the service.

4.2 Integrating Prometheus and Grafana with Logging and Tracing Systems

Tracing and logging are integral components of effective application monitoring, in addition to metrics. To collect comprehensive logs for analysis, Prometheus may be linked with logging platforms like Elasticsearch and Fluentd. Grafana's flexible data source features enable it to obtain log data from various systems and contrast logs with metrics. Prometheus can also be used in conjunction with distributed tracking systems like Jaeger or Zipkin, which enables developers to monitor requests made by different microservices. By incorporating trace data into Grafana dashboards, one can get a comprehensive understanding of the behavior of an application and aid in debugging and performance optimization.

4.3 Third-Party Plugins and Extensions for Enhanced Functionality

The ability to incorporate third-party plugins and extensions into Prometheus and Grafana's flexible designs expands their capability. The range of services that can be watched is increased by using different plugins that allow interface with messaging platforms, databases, and cloud services. One way to seamlessly integrate with cloud resources is using plugins for AWS CloudWatch or Azure Monitor. Furthermore, Grafana's visualization capabilities can be expanded by community contributed plugins, which provide customized tables and graphs for certain monitoring needs. Users can personalize monitoring parameters and application performance analysis methods using these plug-ins and extensions.

4.4 Benefits of integrating Prometheus and Grafana with other tools

There are several advantages to integrating Prometheus and Grafana with other tools, such as:

[1] A more thorough understanding of the health and performance of the application.

[2] The capacity to keep an eye on programs running invarious settings (e.g. on-premises, cloud). Prometheusand Grafana can be utilized to monitor a diverse arrayof technologies, such as microservices, containers, and legacy systems.

[3] The capacity to add more features to Prometheus and Gloria by utilizing third-party plugins and extensions.

V. RELATED WORKS

Prometheus, the monitoring system used to track data from the OpenStack cloud platform, is introduced and its benefits are discussed in the paper [1], which also uses Grafana. To construct a real-time data visualization system by visualizing the tracking apparatus. Every system transmits the information. Information gathered by the exporters, including system data describes the RAM, disk, I/O, and other components of each node server Grafana and Prometheus, the data collector and storage brings the statistics to life. Conclusion of the study states that the system is effective, therefore making OpenStack a more comprehensive platform. Software supports the reliability and scalability of the system.

On-premise server monitoring is a problem that the paper [2] attempts to address. Since these servers need to be extremely secure, regular monitoring is challenging. According to the document, administrators will just need to examine the server for initial configuration and recurring maintenance because a Telegram bot would supply all data remotely. Prometheus serves as a surveillance and notification system. Telegram bots communicate with Python services to deliver notifications and obtain data from the database. Furthermore, the report delineates future research avenues by verifying the service's interoperability with alternative messaging platforms and monitoring technologies.

In the past, servers were observed using the analytics repository Cloudwatch [3]. The restrictions of Cloudwatch included the ability to only do five actions per alarm, a monthly limit of ten alarms, and a one-day monitoring period. Furthermore, exporting alerting alarm data for additional analysis was not possible. There were numerous screens to travel through, making the user interface (UI) untidy. It was unable to monitor servers because it offered no recommendations of any kind.

Machine learning is proposed as a means of monitoring servers in the study [4]. Based on some physical attributes like system memory, RAM, and swap tool, it employs a novel K Nearest Neighbor (KNN) machine learning method to categorize the server state as healthy or sick. Additionally, a training dataset is used to train the model. Nvd3.js is used to visualize the findings as a bar graph.

Paper [5] "Observability in Kubernetes Cluster: Automatic Anomalies Detection using Prometheus" has researched observability in Kubernetes Cluster, identified potential enhancements, and then created, put into practice, and assessed a novel approach in comparison to existing methods for the automatic detection of anomalies in server performance metrics. Addressing this issue becomes more important when one realizes that human intervention is an inevitable part of using a Kubernetes monitoring system. They believed that the key elements in a Kubernetes cluster to watch out for were logs, metrics, traces, etc. The paper's solution, which cleverly takes advantage of Kubernetes' scalability features, thus focuses on training the data model and giving Prometheus additional metrics for warnings.

In the era of big data, data centers and computing organizations face a challenge that must be met by a system that can support heterogeneity, complex next-generation systems, intensive power and cooling requirements, complex network and storage requirements, and rapid monitoring and control of their operations. This challenge is recognized in Paper [6], "Towards a Framework for Monitoring and Analyzing High Performance Computing Environments Using Kubernetes and Prometheus."

Because data is being generated at a faster rate, there is a need to monitor large amounts of data in real time, and the traditional monitoring and alerting system causes confusion by sending many alerts.

Paper [7] the study ”A Resource Utilization Analytics Platform Using Grafana and Telegraf for the Savio Supercluster” looked into a number of system resource usage monitoring options and found that while traditional systems are helpful for managers, users, and system administrators because they can provide meaningful performance statistics via the command line, they don’t provide a thorough visualization of system resource usage trends. The authors suggested a method for gathering system state and visualizing it that included using Slurm to obtain job states, Telegraf to gather CPU metrics, InfluxDB to store time series data, and Grafana to visualize and obtain insights into system resource usage and alerts via commonly used industry accepted communication channels. The method provided revolves around the efficient removal of system task data, in conjunction with other performance measures and their insightful display.

Paper [8] the impact of using Prometheus and Grafana for performance monitoring of HPC environments involving CPUs and especially GPUs has been thoroughly studied in ”Jobstats: A Slurm-Compatible Job Monitoring Platform for CPU and GPU Clusters.” In these environments, file system slowdowns, CPU overloads, and job failure rates are challenging as they increase with the complexity of HPC clusters. Therefore, the authors proposed the Jobstats job monitoring platform, which is further configured with four Prometheus exporters that monitor CPU and memory usage, GPU Job statistics, and Node Specific Statistics, all of which are then visualized by Grafana. Individual job statistics are stored in the Slurm database. The primary contributions of this study are the Jobstats monitoring platform, the tools developed on this platform, and the time series database’s processing speed.

VI. USE CASES

Applications running in containers: Prometheus and Grafana can be used to track the performance of applications running in containers on Docker Swarm or Kubernetes. Metrics can be obtained from the Docker Swarm or Kubernetes APIs, or they can be exported to Prometheus via exporters from container orchestration platforms.

6.1 Monitoring microservices-based applications’ health

Prometheus and Grafana can be used to track metrics like resource usage, error rates, and response times to keep an eye on the wellbeing of microservices-based applications. Before they result in disruptions or outages, these data can be used to detect and fix microservices problems.

6.2 Legacy application performance monitoring

Prometheus and Grafana can be used to scrape metrics from application servers, databases, and other infrastructure components in order to monitor the performance of legacy applications. Performance bottlenecks in legacy applications can be located and fixed with the help of this information.

6.3 Application Security Monitoring

Security events, failed authentication attempts, and login attempts are just a few of the metrics that can be tracked using Prometheus and Grafana. It is possible to swiftly identify and address security threats with the help of this information.

VII. CASE STUDY

7.1 DHL Express Switzerland

Use Prometheus, Grafana, and Grafana k6 to lower MTTR: Within DHL’s observability stack with DHL’s premium Time Definite International service, more than 296 million packages are shipped globally annually. Additionally, for every package that passes through Switzerland, the IT team at DHL Express Switzerland, the local branch of the global logistics and transport company, offers solutions for tracking and analyzing customs clearance, mobile and optical character recognition (OCR) scanning, and warehouse management. Every minute and every shipment counts in this intricate operation, which calls for a deep business and IT foundation. Translation: Errors, false positives, downtime, and unsuccessful requests are all highly unlikely. IT Lead Djamel Djedid and Chief Architect Michael Lerch discussed how their methodical approach to implementing Grafana-centric observability solution helped DHL Express Switzerland solve problems more quickly, save manpower, and extend its observability beyond traditional IT monitoring in their recent talk at GrafanaCON 2023, ”Transforming IT and Business Flows at DHL Express with Grafana, k6 and Prometheus”, which is now available on demand.

7.1.1 Phase one: POC with Prometheus + Grafana

The group determined that a more contemporary and scalable Site Reliability Engineering (SRE) solution was required early in 2020. Their staff were frequently reactive when issues emerged since their outdated monitoring system was on pause. Prometheus and Grafana were used as a proof of concept, and in the end, it was decided to move the skeptical elder observers to Prometheus. The team successfully integrated their new suite at scale just in time to handle a busy period for the company, with the help of Grafana training and self-learning. They have created Grafana dashboards that track worldwide customs clearance data, similar to the one below. The team may immediately detect bottlenecks and communicate information about their location and resolution thanks to these dashboards.

7.1.2 Phase 2: Full speed ahead with Grafana Alerting

By 2021, the group was prepared to advance to a more sophisticated Grafana Alerting and graphical dashboard deployment. They expanded on their existing dashboards and linked their internal Wiki and Microsoft Teams with warnings. ”Our notifications contain the name of the app, an explanation of the issue, a link to the Grafana dashboard, and a link to the Microsoft Teams wiki containing instructions on how to fix it,” Lerch stated. Whoever is in charge can deal with problems as soon as they come up and find speedier solutions than ever before.

7.1.3 Phase 3: Load testing with Grafana k6 for smooth cloud migration

In order to relocate some of its servers from local data centers to the public cloud, the team began an infrastructure modernization project in 2022 after deploying Grafana. According to Djedid, ”We needed to monitor performance and make sure the migration didn’t negatively impact end-user performance. What we needed was a tool to measure the latency between the user and both the cloud and local servers.” Grafana k6 performance testing eliminated uncertainty when DHL Express Switzerland moved from on-premise servers to the public cloud. The results of the stress testing showed that the cloud servers were far more reliable for a higher user base. According to Lerch, ”Grafana K6 really helped us gain confidence that the solution we implemented was reliable and scalable.”

VIII. CONCLUSION

With the growing digital footprint and the difficulties presented by contemporary application architectures, system monitoring tools have become essential resources for developers, operations teams, and companies looking to achieve peak efficiency, dependability, and customer happiness. The combination of Prometheus and Grafana acts as a beacon, pointing organizations in the direction of a future in which apps are not only tracked but also intelligently optimized for optimal user experience. The survey’s findings provide a basis for additional investigation and real-world applications, guaranteeing that the field of application monitoring keeps evolving hand-in-hand with technological advancements.

REFERENCES

[1] L. Chen, M. Xian and J. Liu, ”Monitoring System of OpenStack Cloud Platform Based on Prometheus,” 2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL), Chongqing, China, 2020, pp. 206-209, doi:10.1109/CVIDL51233.2020.0-100.

[2] Rawoof F. M., Tajammul M., “A Survey on Remote On-Premise Server Monitoring”, 2022 Journal of Emerging Technologies and Innovative Research (JETIR) Volume 9, Issue 2.

[3] Kumar A. K., Vinutha B. S., Vinayaditya B.V., “ REAL TIME MONITORING OF SERVERS WITH PROMETHEUS AND GRAFANA FOR HIGH AVAILABILITY”, Apr 2019, International Research Journal of Engineering and Technology (IRJET).

[4] Bose S., Rakesh K.R., “Server status Monitoring using Advanced Machine Learning Algorithms”, 6 June 2020, International Journal of Creative Research Thoughts (IJCRT) Volume 8.

[5] O. Mart, C. Negru, F. Pop and A. Castiglione, ”Observability in Kubernetes Cluster: Automatic Anomalies Detection using Prometheus,” 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), Yanuca Island, Cuvu, Fiji, 2020, pp. 565-570, doi:10.1109/HPCC-SmartCity-DSS 50907.2020.00071.

[6] N. Sukhija and E. Bautista, ”Towards a Framework for Monitoring and Analyzing High Performance Computing Environments Using Kubernetes and Prometheus,” 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), Leicester, UK, 2019, pp. 257-262, doi: 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00087.

[7] Nicolas Chan. 2019. A Resource Utilization Analytics Platform Using Grafana and Telegraf for the Savio Supercluster. In Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning) (PEARC ’19). Association for Computing Machinery, New York, NY, USA, Article 31, 1–6.

[8] 1. Josko Plazonic, Jonathan Halverson, and Troy Comi. 2023. Jobstats: A Slurm-Compatible Job Monitoring Platform for CPU and GPU Clusters. In Practice and Experience in Advanced Research Computing (PEARC ’23). Association for Computing Machinery, New York, NY, USA,102–108. https://doi.org/10.1145/3569951.3604396.

[9] Zhang Yu-Long, Cao Heng, Hu Jing-Feng, Wang Jin-Cheng, “Design and implementation of remote monitoring system for welding machine based on web”, 2018.

[10] A. Kaushik, “Use of Open Source Technologies for Enterprise Server Monitoring Using SNMP”, IJCSE, Vol. 2, No. 7, pp. 2246-2252, 2010.

[11] J. Swarna, C. S. Raja, D. Ravichandran, “Cloud Monitoring Based on SNMP”, Journal of Theoretical and Applied Information Technology, Vol. 40, No. 2, pp. 188-193, 2012.

[12] Andreas witting and Michael witting, Amazon web services in actions, ISBN- 1617292885, 17/10/2015.

[13] G. Suciu, V. Suciu, R. Gheorghe, C. Dobre, F. Pop, and A. Castiglione. “Analysis of Network Management and Monitoring Using Cloud Computing”, Computational Intelligence and Intelligent Systems, Springer, pp. 343-352, 2016.

[14] Ikram Hawramani, Cloud computing for complete beginners: Building and scaling high performance web servers on the amazon cloud.

[15] Chakraborty M, Kundan AP. Grafana. In: Monitoring Cloud-Native Applications [Internet]. Berkeley, CA, Apress; 2021. [cited 15th August 2022] Available from: https://doi.org/10.1007/978-1-4842-6888-96.