An Enterprise Model to Monitor Serverless Applications using AWS CloudWatch

Sai Teja Makani

ABSTRACT

This abstract explores the integration of AWS CloudWatch dashboards for effective monitoring and alerting within serverless applications. As organizations increasingly adopt serverless computing, the need for streamlined monitoring and proactive issue identification becomes paramount. AWS CloudWatch dashboards offer a comprehensive solution by providing a consolidated and customizable visual interface for monitoring key metrics, logs, and events across various serverless components. The abstract delves into the benefits of employing CloudWatch dashboards, highlighting their role in facilitating real-time tracking of serverless resources, such as AWS Lambda functions, API Gateway, and more. It sheds light on the seamless integration of CloudWatch Metrics, Logs, and Alarms, empowering users to create insightful visualizations that aid in diagnosing performance bottlenecks, identifying trends, and optimizing resource utilization. Furthermore, the abstract examines the dynamic alerting capabilities enabled by CloudWatch Alarms, ensuring timely notifications for critical thresholds or anomalies. This proactive alerting mechanism enhances application reliability and minimizes downtime. The abstract also touches upon best practices for designing effective dashboards, ensuring they cater to specific business needs and objectives.

In conclusion, the abstract underscores how AWS CloudWatch dashboards serve as a pivotal tool for monitoring and alerting within serverless architectures, enabling organizations to achieve optimal performance, swift issue resolution, and a superior end-user experience.

Keywords: AWS CloudWatch, Serverless Application, Monitoring, Alerting and Dashboards.

I. INTRODUCTION

In recent years, the paradigm shift towards cloud computing has revolutionized the landscape of application development and deployment (i.e., [1]). Among the innovative architectures that have emerged, serverless computing has garnered significant attention due to its promises of enhanced scalability, reduced operational complexity, and cost-efficiency (i.e., [2]). This novel approach eliminates the need for provisioning and managing underlying infrastructure, allowing developers to focus solely on code logic and functionality. While serverless architecture offers undeniable benefits, it introduces new challenges in terms of monitoring and alerting that differ from traditional infrastructure-based models (i.e., [3]).

This paper introduces a critical solution to address the monitoring and alerting intricacies within serverless applications: the integration of AWS CloudWatch dashboards. AWS

CloudWatch, a comprehensive monitoring service offered by Amazon Web Services (AWS), provides a suite of tools to collect, analyze, and visualize real-time data from various AWS resources (i.e., [4]). Within the context of serverless computing, CloudWatch dashboards emerge as indispensable instruments, enabling seamless monitoring and proactive issue identification.

Serverless applications are composed of dynamically scaling components, such as AWS Lambda functions and API Gateway endpoints, creating a complex and fluid ecosystem. Traditional monitoring approaches struggle to capture the nuances of these ephemeral components and their interactions, necessitating tailored strategies (i.e., [5]). CloudWatch dashboards empower developers and operations teams to design customized visualizations that encapsulate the health, performance, and behavior of serverless resources, facilitating comprehensive insights. A fundamental aspect of effective serverless application monitoring is the ability to proactively identify and mitigate issues before they impact user experience. CloudWatch Alarms enable the establishment of predefined thresholds for various metrics, triggering real-time notifications when breaches or anomalies occur (i.e., [6]). This dynamic alerting mechanism ensures timely responses, contributing to improved application reliability and reduced downtime. As organizations increasingly embrace serverless computing to enhance agility and innovation, the implementation of AWS CloudWatch dashboards becomes a pivotal factor in ensuring smooth operations and optimal performance. This paper further explores CloudWatch's capabilities, delves into best practices for dashboard design, and highlights its role in driving efficient serverless application monitoring and alerting strategies.

II. COMPARISON OF SERVERLESS AND SERVER BASED LOGGING

In the context of server-based logging, the conventional approach involves the installation of logging plugins or tools such as Log4j, strategically configured to capture logs across various levels, including DEBUG, Logging, or ERROR (i.e., [7]). These logs are subsequently transported to specialized monitoring and visualization services like Prometheus and Grafana, where they are transformed into insightful dashboards for analysis and diagnosis (i.e., [8]). While this method is effective, it introduces a complex ecosystem of components, where a failure in any one element can reverberate throughout the entire system, leading to potential disruptions.

Conversely, within the realm of AWS serverless applications, logging operates within the native AWS environment, primarily utilizing CloudWatch logs (i.e., [9]). Remarkably, CloudWatch logs persist even in the event of application crashes or failures, offering a built-in and resilient logging solution. This eliminates the need for external exports to platforms like Grafana or Prometheus, streamlining the process significantly. The transition from the traditional server-based model to the serverless paradigm reduces potential points of failure and simplifies the monitoring architecture.

AWS CloudWatch dashboards emerge as a transformative aspect of this shift. Instead of duplicating logs on servers and exporting them to external tools like Grafana or Prometheus, CloudWatch dashboards provide a consolidated and user-friendly interface for creating custom visualizations (i.e., [10]). The need for separate licenses and the effort of maintaining external

dashboards are circumvented, further streamlining the monitoring process. One of the noteworthy advantages of CloudWatch dashboards is their seamless integration within the AWS ecosystem, eliminating the requirement for additional licenses. Organizations can optimize costs by paying only for the resources and features they utilize, making CloudWatch dashboards a cost-effective solution (i.e., [11]). This integration aligns well with the broader principles of serverless architecture, where the emphasis lies on scalability, cost-efficiency, and operational simplicity.

In conclusion, the transition from traditional server-based logging to AWS serverless application logging represents a paradigm shift that simplifies and enhances monitoring processes. The utilization of CloudWatch logs and dashboards offers an integrated and resilient solution that eliminates the complexities associated with external logging and visualization tools. By leveraging the native capabilities of CloudWatch dashboards, organizations can achieve comprehensive monitoring, proactive issue identification, and cost-effective practices within their serverless environments.

III. CLOUDWATCH LOG INSIGHTS

At the core of our CloudWatch dashboard, CloudWatch Logs Insights serves as a foundational component for every widget. These widgets encompass a diverse range, including charts, maps, numerical displays, graphs, and logs. The versatility of CloudWatch Logs Insights allows us to tailor these widgets to our specific requirements, enabling the creation of a highly customized dashboard that precisely aligns with our intended objectives. This adaptability empowers us to craft an intuitive and comprehensive monitoring interface for seamless insights and efficient decision-making. On top of logging AWS provides metrics on all of our resources, those can't be transferred to external tools directly.

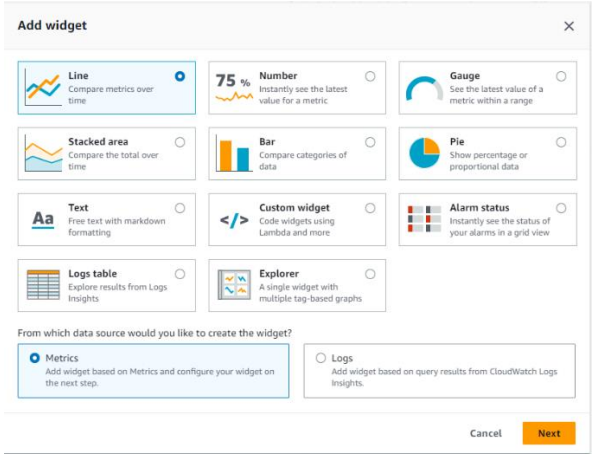

Fig: 1 AWS CloudWatch Widget Options

AWS CloudWatch Logs Insights offers powerful querying and analysis capabilities for log data, enabling users to derive valuable insights from their logs. Below are two examples of how CloudWatch Logs Insights can be used.

Anomaly Detection in Application Logs: CloudWatch Logs Insights can help identify anomalies or patterns within application logs. For instance, consider an e-commerce platform experiencing an unexpected surge in failed transactions. By querying CloudWatch Logs Insights, developers can analyze error logs across various microservices and pinpoint the root cause of the issue. They could query for specific error messages or status codes over a defined time period, allowing them to detect patterns or spikes in failures. This real-time analysis aids in quickly diagnosing and addressing the problem, thus enhancing the overall reliability of the application (i.e., [12]).

Security Investigation and Threat Detection: CloudWatch Logs Insights can also play a crucial role in security investigations. Imagine a scenario where a security breach is suspected within a serverless application. By querying CloudWatch Logs Insights, security teams can search for suspicious activity or unauthorized access attempts across the application's logs. They can identify IP addresses, user agents, or specific endpoints associated with potentially malicious behavior. This enables timely responses to security incidents and aids in enhancing the application's overall security posture (i.e., [13]).

Below is one of the real-world example to write a cloud watch insight snippet majorly

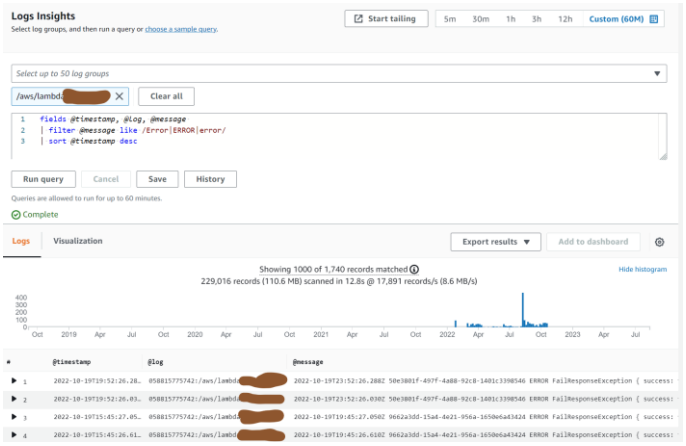

Fig: 2 AWS CloudWatch Insights to Capture Errors

IV. BLOCK DIAGRAM

The provided block diagram illustrates the practical sequence of CloudWatch logging, showcasing its efficient design in contrast to employing a distinct service for uploading logs to Prometheus and Grafana processing nodes. CloudWatch dashboards excel in terms of cost-effectiveness, operational continuity, availability, and user-friendliness.

In the outlined block diagram, AWS assumes the role of capturing diverse metrics and logs from various services. Subsequently, these metrics and logs can be seamlessly subscribed to widgets using CloudWatch Insights. Once the insights derived from these subscriptions meet desired criteria, they can be approved and integrated into the final dashboard design. The resulting dashboard, rich with multiple widgets, takes shape as depicted on the far right side of the diagram. This streamlined process exemplifies the optimization achieved through CloudWatch dashboards, eliminating the need for intermediary services and simplifying the workflow for logging, metrics analysis, and visualization. With CloudWatch dashboards' seamless integration into the AWS environment, organizations can capitalize on a comprehensive and intuitive monitoring solution that not only enhances operational efficiency but also aligns harmoniously with the broader goals of cost-conscious and agile application management.

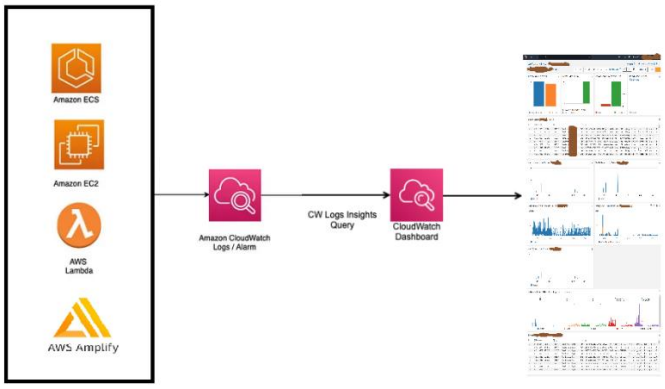

Fig: 3 Architecture of the Dashboard

V. RESULTS

The depicted CloudWatch dashboard offers a comprehensive insight into the monitoring and logging framework of an entire project, illustrating an efficient approach to overseeing diverse components. In scenarios where all components originate from AWS, the process involves the creation of widgets strategically aligned with specific metrics and logs of the tracked components. This consolidation facilitates convenient data comparison and analysis, enhancing the overall monitoring experience. Consider an example involving two Lambda functions within the project. By utilizing CloudWatch log insights, you can effortlessly point to various metrics of both functions, such as invocation count, runtime duration, and error occurrences. This arrangement allows for swift performance evaluation and issue identification. Furthermore, for a granular examination of error origins, a separate widget can provide access to the actual logs responsible for the errors, enhancing diagnostic capabilities (ref: Fig. 2). Within this dashboard an alarm widget could be added to raise a flag when there is an error occurred or a flow failed.

Comparatively, this CloudWatch-based approach proves notably more streamlined than the utilization of Prometheus and Grafana. While these platforms offer similar functionality, CloudWatch's native integration within the AWS environment simplifies configuration and

usage, mitigating potential complexities and reducing the operational burden. The illustrated CloudWatch dashboard underscores the synergy between intuitive visualization and comprehensive monitoring, reinforcing the advantages of a unified approach for metrics, logs, and error analysis. By harnessing CloudWatch's capabilities, organizations can achieve heightened observability, effective troubleshooting, and seamless performance optimization, making it a compelling choice for modern monitoring needs.


Backend-Lambda -Invocations
Backend-Lambda -Duration
Backend-Lambda -Error count and succes...
Backend-Lambda -Throttles
Backend-Lambda Error logs
Amplify 4xx errors (sum) Front-end
Amplify 5xx errors (sum) Front-end
Amplify Time to first byte (average) Front-end
Amplify Data transfer (sum) Front-end
Amplify Requests (sum) Front-end
API Gateway 4XXError, 5XXError, Count, IntegrationLatency, Latency
Lambda

Fig: 4 Final Results of the CloudWatch Dashboard

VI. CONCLUSION

In conclusion, the integration of AWS CloudWatch dashboards into serverless application monitoring and alerting represents a transformative shift that addresses the unique challenges posed by dynamic and event-driven architectures. As organizations increasingly embrace serverless computing to enhance agility and optimize resource utilization, CloudWatch dashboards emerge as a pivotal tool for ensuring optimal performance, efficient issue resolution, and a seamless end-user experience. The shift from traditional server-based logging to CloudWatch-based monitoring signifies a paradigmatic evolution in the way applications are monitored and managed. CloudWatch's native integration within the AWS ecosystem simplifies the monitoring process, eliminating the need for separate licenses and reducing operational complexity. This transition empowers organizations to embrace the core tenets of serverless architecture—scalability, cost-efficiency, and reduced operational overhead—while maintaining robust observability. The flexibility of CloudWatch dashboards provides the ability to design tailored visualizations that encapsulate the health, behavior, and performance of serverless components. This real-time monitoring and alerting mechanism ensures timely issue detection and proactive resolution, thereby minimizing downtime and bolstering application reliability. By relying on CloudWatch's dynamic alerting capabilities, organizations can swiftly respond to anomalies, breaches, or performance degradation, safeguarding their systems and bolstering user trust.

Furthermore, the cost-effective pay-as-you-go model of CloudWatch dashboards aligns seamlessly with the financial advantages offered by serverless computing. Organizations can optimize their monitoring costs by only paying for the resources and features they utilize, ensuring a scalable and budget-friendly solution. As the digital landscape continues to evolve, the synergy between serverless computing and CloudWatch dashboards demonstrates the forward trajectory of application development and monitoring. This paper has illuminated the transformative potential of CloudWatch dashboards, underscoring their indispensable role in achieving robust, efficient, and proactive monitoring and alerting within the context of serverless applications. By harnessing the capabilities of AWS CloudWatch dashboards, organizations can not only adapt to the changing technological landscape but also excel in delivering resilient and high-performance serverless applications.

VII. REFERENCES

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., ... & Zaharia, M. (2010). A view of cloud computing. Communications of the ACM, 53(4), 50-58.

2. Book: Roberts, J., & Farrell, R. (2018). Serverless Architectures. O'Reilly Media.

3. Hao, H., & Viswanathan, M. (2017). Challenges in serverless computing. In Proceedings of the 2017 ACM International Conference on Management of Data (pp. 2051-2054).

4. Amazon Web Services. (2021). Amazon CloudWatch. https://aws.amazon.com/cloudwatch/

5. Zhang, L., Cheng, J., Boutaba, R., & Qian, F. (2019). A comprehensive look at serverless computing. IEEE Transactions on Cloud Computing.

6. Amazon Web Services. (2021). Amazon CloudWatch Alarms. https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/AlarmThatSendsEmail.html

7. Apache Log4j Project. (2021). Introduction to Log4j. https://logging.apache.org/log4j/2.x/manual/introduction.html

8. Grafana Labs. (2021). What is Grafana? https://grafana.com/docs/grafana/latest/

9. Amazon Web Services. (2021). Amazon CloudWatch Logs. https://aws.amazon.com/cloudwatch/features/logs/

10. Amazon Web Services. (2021). Amazon CloudWatch Dashboards. https://aws.amazon.com/cloudwatch/features/dashboards/

11. Amazon Web Services. (2021). Amazon CloudWatch Pricing. https://aws.amazon.com/cloudwatch/pricing/

12. Amazon Web Services. (2021). CloudWatch Logs Insights. https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/AnalyzingLogData.html

13. Amazon Web Services. (2021). Using CloudWatch Logs Insights to Analyze Application Logs. https://aws.amazon.com/blogs/devops/using-cloudwatch-logs-insights-to-analyze-application-logs/