Monitoring AWS Lambda with CloudWatch

Satish Kumar Malaraju

Technology Architect
(DevSecOps), California-US

Abstract:
AWS Lambda is a serverless compute service that enables developers to execute code without the complexities of server management. It provides automatic scaling and a cost-effective pay-per-use model, making it a preferred choice for modern cloud applications. However, to maintain optimal performance, reliability, and cost-efficiency, robust monitoring mechanisms are essential. Amazon CloudWatch serves as a powerful observability tool that provides real-time metrics, logs, and event tracking for AWS resources, including Lambda functions. This paper explores the capabilities of CloudWatch in monitoring AWS Lambda, highlighting key metrics, best practices, and strategies for setting up alerts and dashboards. By leveraging CloudWatch effectively, organizations can enhance the performance and reliability of their Lambda-based applications while optimizing operational costs.

Keywords: AWS Lambda, serverless computing, Amazon CloudWatch, monitoring, observability, metrics, logs, performance optimization, cost-efficiency, scalability.

1. INTRODUCTION

Serverless computing has transformed the way modern applications are built and deployed by eliminating the need for developers to manage underlying infrastructure. Among the leading serverless solutions, AWS Lambda provides a highly scalable and cost-efficient compute service that allows users to execute code in response to events without provisioning or maintaining servers. By automatically scaling based on incoming requests and charging only for the compute time consumed, AWS Lambda has become a preferred choice for organizations looking to build dynamic, event-driven applications with minimal operational overhead.[12][13]

Despite its advantages, running applications on AWS Lambda introduces new challenges related to performance, reliability, and cost management. Since serverless functions operate in ephemeral environments with limited execution time, monitoring their behavior is critical to ensuring smooth operation. Without proper visibility into function execution, developers may struggle to detect performance bottlenecks, diagnose failures, or optimize resource utilization. This makes an effective monitoring strategy essential for maintaining the health and efficiency of serverless applications.

Amazon CloudWatch, a comprehensive monitoring and observability service provided by AWS, offers powerful capabilities for tracking and analyzing the performance of AWS Lambda functions. CloudWatch enables developers to collect real-time metrics, analyze logs, and set up automated alerts to proactively manage application health. By leveraging CloudWatch's monitoring tools, organizations can gain deep insights into function execution, identify anomalies, and optimize their applications for better efficiency and cost-effectiveness.[15]

This paper presents a detailed guide to monitoring AWS Lambda functions using Amazon CloudWatch. It explores the core features of CloudWatch, the key performance metrics that should be tracked, and best practices for configuring alarms and dashboards. By implementing an effective monitoring strategy, developers can enhance the reliability, performance, and cost-efficiency of their Lambda-based applications, ensuring that they operate seamlessly in production environments.

Amazon CloudWatch provides a comprehensive set of monitoring and observability tools specifically designed to enhance the visibility and performance tracking of AWS Lambda functions. By leveraging these features, developers can gain deeper insights into function behavior, troubleshoot issues effectively, and optimize application performance. [2][3]
The key features of CloudWatch for Lambda monitoring include:

- Metrics: CloudWatch automatically collects a range of key performance metrics for Lambda functions, including invocation count, duration, error rate, and throttling occurrences. These metrics provide valuable insights into how frequently functions are executed, how long they take to complete, and whether any errors or throttles impact their performance. Monitoring these metrics helps developers identify inefficiencies and ensure the smooth operation of their serverless applications.

- Logs: AWS Lambda integrates seamlessly with CloudWatch Logs, capturing all log outputs generated by function executions. This includes custom logs added by developers, as well as system-generated logs that provide execution details. These logs serve as a valuable resource for debugging and troubleshooting, allowing teams to trace function behavior, detect anomalies, and resolve errors efficiently.

- Alarms: CloudWatch allows users to configure alarms that trigger notifications when specific metrics exceed predefined thresholds. This proactive monitoring approach helps teams respond to potential issues before they escalate into significant problems. Setting up alarms for critical metrics, such as high error rates or increased function duration, ensures that developers can take corrective actions promptly, reducing the risk of application downtime.

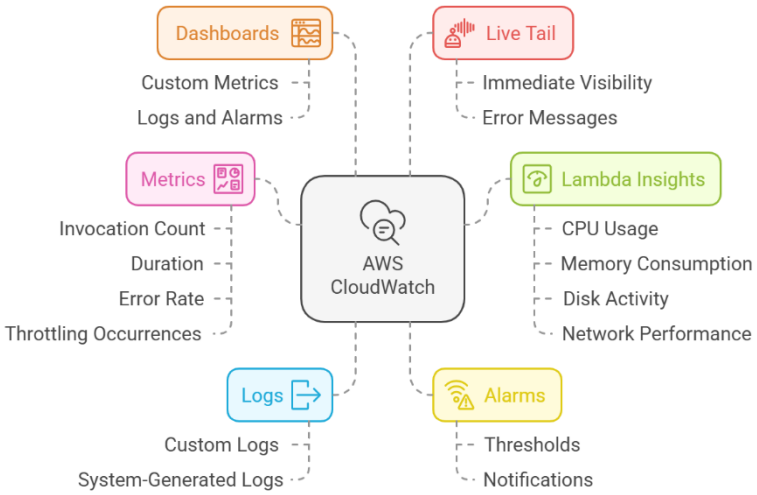
- Dashboards: CloudWatch dashboards provide a customizable, visual representation of Lambda function performance. Developers can create custom dashboards to display key metrics, logs, and alarms in one centralized view. These dashboards can be tailored to monitor individual functions or groups of functions and can also incorporate widgets for other AWS services, such as API Gateway and SQS, to provide a holistic view of an application's overall health and performance.

- CloudWatch Logs Live Tail: This feature enables real-time log streaming within the Lambda console, allowing developers to monitor function logs as they are generated. By providing immediate visibility into function execution and error messages, Live Tail helps teams quickly diagnose and resolve issues, minimizing downtime and improving overall application reliability.

- Lambda Insights: For more advanced monitoring capabilities, CloudWatch offers Lambda Insights, which provides deeper visibility into system-level performance metrics. This feature aggregates data such as CPU usage, memory consumption, disk activity, and network performance. By analyzing these metrics, developers can identify performance bottlenecks, optimize resource allocation, and enhance the efficiency of their Lambda functions.

By leveraging these CloudWatch features, organizations can ensure the reliability, performance, and cost-effectiveness of their AWS Lambda functions. A well-implemented monitoring strategy enables proactive issue resolution, enhances observability, and helps maintain seamless application operation in serverless environments.[6][7]

Figure 1: AWS CloudWatch Features for Lambda Monitoring



2. COMPREHENSIVE METRICS FOR MONITORING AWS LAMBDA FUNCTIONS WITH AMAZON CLOUDWATCH

Amazon CloudWatch provides an extensive set of metrics to monitor the behavior and performance of AWS Lambda functions. These metrics offer deep insights into function execution, enabling developers to track key performance indicators, diagnose issues, and optimize application efficiency. CloudWatch categorizes Lambda function metrics into different types, each serving a unique purpose in observability and monitoring.[3][4]

Figure 2: AWS Lambda Metrics Overview

- Invocation Metrics

Invocation metrics serve as fundamental indicators of a function's execution and its outcome. They provide essential data regarding the number of times a function is invoked and whether those invocations succeed or fail. Key invocation metrics include:

- Invocations: Tracks the number of times a function is triggered, helping developers analyze usage patterns and trends over time.
- Errors: Measures the number of invocations that result in an error, allowing teams to identify and troubleshoot failures effectively.

- Success Rate: Derived from the ratio of successful invocations to total invocations, providing insights into overall function reliability.
- Retries: Captures the number of times AWS automatically retries a failed invocation, which is particularly useful for monitoring functions triggered by asynchronous event sources.

- Performance Metrics

Performance metrics focus on measuring execution efficiency and response times. These metrics are crucial for identifying bottlenecks, optimizing resource allocation, and improving function responsiveness. The key performance metrics include:

- Duration: Measures the total execution time of a function from start to finish, helping developers gauge function efficiency and responsiveness.
- Billed Duration: Represents the execution time rounded up to the nearest 1 ms, which is used for cost calculation. Monitoring this metric allows teams to optimize function runtime and reduce unnecessary costs.
- Memory Usage: Tracks the amount of memory utilized during function execution, helping developers fine-tune allocated memory for optimal performance.
- Cold Start Duration: Captures the time taken for function initialization when a new execution environment is created, which can impact response times in latency-sensitive applications.

- Concurrency Metrics

Concurrency metrics track how many instances of a Lambda function are running simultaneously, providing insights into scalability and resource utilization. These metrics are particularly important for applications with variable workloads. Key concurrency metrics include:

- Concurrent Executions: Measures the number of function instances running simultaneously, which is crucial for ensuring efficient resource allocation.
- Throttles: Tracks the number of function invocations that are throttled due to exceeding concurrency limits. High throttle rates indicate the need for increasing concurrency limits or optimizing invocation patterns.
- Provisioned Concurrency Utilization: Indicates the percentage of reserved provisioned concurrency being utilized, helping teams ensure that pre-allocated instances are efficiently used.

- Asynchronous Invocation Metrics

Asynchronous invocation metrics provide insights into functions that process events asynchronously, allowing developers to monitor message failures and retries. These metrics are essential for ensuring the reliability of event-driven applications.

- DeadLetterErrors: Tracks the number of messages that fail to be processed and are sent to a Dead Letter Queue (DLQ). Monitoring this metric helps developers diagnose failed event processing.
- AsyncEventAge: Measures the age of an event when it is finally processed, providing insights into delays and backlog accumulation.
- AsyncDeliveryFailures: Captures the number of asynchronous event deliveries that fail permanently, indicating potential configuration or processing issues.

5. Event Source Mapping Metrics

For Lambda functions triggered by event sources such as Amazon Kinesis, DynamoDB Streams, or SQS, CloudWatch provides specific metrics to track event processing efficiency. These metrics help developers optimize data ingestion and processing rates.

- IteratorAge: Measures the age of the oldest unprocessed record in an event stream, helping identify lag in data processing.
- BatchSize: Tracks the number of records processed in a batch execution, allowing teams to fine-tune batch sizes for optimal performance.
- MessagesProcessed: Counts the number of messages successfully processed from an event source, indicating the function's throughput.

- Custom Metrics for Advanced Monitoring

Beyond the standard CloudWatch metrics, developers can create custom metrics to monitor application-specific performance indicators. These custom metrics provide granular insights tailored to unique business requirements. Some common use cases include:

- Business-Critical Errors: Tracking errors specific to business logic, such as failed transactions or API request failures.
- API Success Rate: Measuring the percentage of successful API requests handled by a Lambda function.
- User Activity Metrics: Monitoring the number of unique users triggering a function, providing insights into application usage.
- Custom Latency Metrics: Capturing latency for specific function operations, such as database queries or external API calls.

Amazon CloudWatch provides a rich set of built-in metrics for monitoring AWS Lambda functions, covering aspects such as invocation behavior, performance efficiency, concurrency, and event-driven processing. By leveraging these metrics, along with custom monitoring solutions, organizations can gain comprehensive visibility into their serverless applications. Effective use of these metrics enables proactive issue detection, performance optimization, and cost efficiency, ensuring that Lambda functions operate seamlessly and reliably in production environments. Here are some of the most important metrics to keep an eye on:

Table 1: Important metrics

Metric Name	Description	Importance
Invocations	Tracks the number of times your function is invoked.	A sudden spike or drop in invocations could indicate a change in traffic patterns or a potential issue.
Errors	Tracks the number of failed function executions.	A high error rate indicates a problem with your function's code or configuration.
Duration	Measures the time it takes for your function to execute.	Long durations can increase costs and impact user experience.
Throttles	Counts the number of invocation requests that were throttled due to exceeding concurrency limits.	Throttling can indicate that your function needs more resources or that you need to optimize your code.
DeadLetterErrors	Tracks the number of times Lambda failed to send an event to a dead-letter queue (DLQ).	This is important for asynchronous invocations, as it helps you identify and debug failed events.
IteratorAge	For functions processing events from streaming sources like Kinesis or DynamoDB Streams, this metric measures the age of the last processed event.	A high iterator age indicates that your function is falling behind in processing events.
ConcurrentExecutions	Tracks the number of function instances running concurrently.	Monitoring this metric helps you understand capacity utilization and manage resource allocation.

3. COMPREHENSIVE GUIDE TO MONITORING AWS LAMBDA WITH AMAZON CLOUDWATCH

CloudWatch alarms provide a proactive monitoring mechanism that allows you to define thresholds for various Lambda function metrics and receive notifications when those thresholds are exceeded. By configuring alarms for critical performance indicators, you can detect potential issues early and take corrective actions before they impact your application and users. [1][14]

The steps to setup CloudWatch Alarm are:

1. Access the CloudWatch Console
   - Sign in to the AWS Management Console and navigate to the Amazon CloudWatch service.
   - In the left panel, select Alarms, then click on Create Alarm.
2. Choose the Metric to Monitor
   - Click Select metric and choose the AWS Lambda namespace.
   - Browse through available metrics and select the one you want to monitor, such as Errors, Duration, or Throttles.
3. Define Alarm Conditions
   - Specify the threshold for triggering the alarm. For example, you can configure the alarm to trigger if the error rate exceeds 1% for five consecutive minutes or if the function execution duration consistently exceeds a defined limit.
   - Define the evaluation period, which determines how frequently the alarm checks for threshold breaches.
4. Set Notification Actions
   - Choose how you want to receive alerts. CloudWatch alarms can be configured to trigger actions such as sending an email notification, triggering an Amazon SNS (Simple Notification Service) topic, or invoking an AWS Lambda function for automated remediation.
5. Review and Create the Alarm
   - Verify the alarm settings and confirm that the configuration aligns with your monitoring objectives.
   - Click Create Alarm to finalize the setup.

By implementing CloudWatch alarms for key performance metrics, you ensure that you are notified about potential issues before they escalate into critical failures, enabling faster troubleshooting and improved system reliability.[12][3]

CloudWatch dashboards provide a customizable visual interface for monitoring Lambda function performance. By aggregating key metrics, logs, and alarms into a centralized view, dashboards allow developers and operations teams to quickly assess the health and efficiency of their serverless applications.

The Steps to Create a CloudWatch Dashboard are:

1. Access the CloudWatch Console
   - Open the Amazon CloudWatch service in the AWS Management Console.
   - In the left panel, navigate to Dashboards and click on Create Dashboard.
2. Name and Describe Your Dashboard
   - Enter a name and optional description for your dashboard to specify its purpose, such as monitoring a specific application, function group, or workload.
3. Add Widgets to Visualize Metrics and Logs
   - Click Add Widget and choose from various visualization options such as line graphs, bar charts, numerical data, and log streams.
   - Select the Lambda function metrics you want to display, such as invocations, errors, duration, and concurrency usage.
4. Customize Widget Configuration
   - Define the time range for data visualization (e.g., last 1 hour, 24 hours, or custom time range).
   - Customize the appearance of each widget for better readability.
5. Include Data from Multiple AWS Services
   - To gain a more comprehensive view of application performance, include widgets for related AWS services such as API Gateway, SQS, DynamoDB, and Kinesis.
6. Save and Monitor Your Dashboard
   - Click Save Dashboard and access it anytime to monitor real-time function performance and trends.

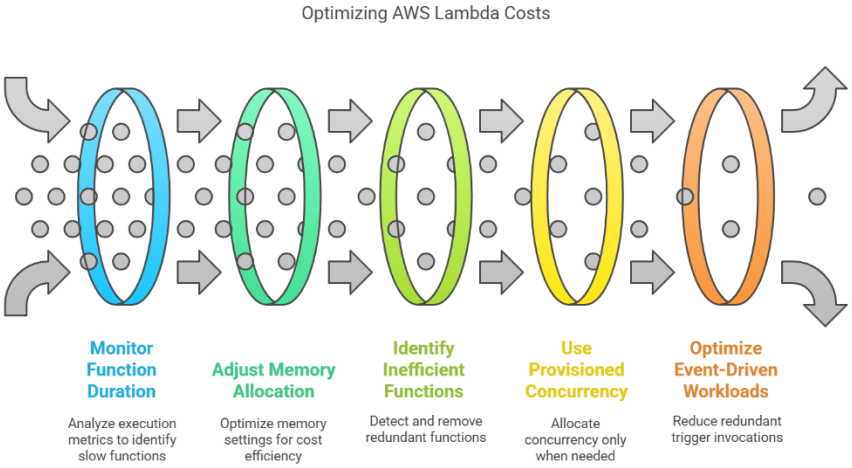
By leveraging CloudWatch dashboards, teams can track Lambda function health in an intuitive, visually structured format, ensuring efficient monitoring and faster response to anomalies.

CloudWatch Logs Insights is a powerful log analysis tool that enables developers to search, filter, and analyze Lambda function logs using SQL-like queries. It is particularly useful for debugging issues, identifying trends, and gaining deeper insights into function behavior.

- Detecting Frequent Errors: Use query-based analysis to identify the most common error messages and their frequency.
- Filtering Log Events: Search for log entries that contain specific keywords, such as "timeout" or "access denied," to quickly pinpoint issues.
- Analyzing Performance Trends: Extract execution duration patterns over time to detect performance degradation.
- Correlating Logs with Metrics: Combine log analysis with function metrics to trace the root cause of failures or performance bottlenecks.

By leveraging CloudWatch Logs Insights, developers can gain granular visibility into Lambda function behavior, making it easier to troubleshoot problems and optimize function performance.

Figure 3: Strategies to Reduce AWS Lambda Costs



Beyond performance monitoring, CloudWatch helps optimize Lambda costs by providing insights into function execution, resource usage, and billing patterns.
Amazon CloudWatch provides a comprehensive monitoring solution for AWS Lambda functions, offering capabilities such as real-time metric tracking, log analysis, alarms, and dashboards. By leveraging these features, organizations can:

- Proactively detect and resolve issues with alarms and notifications.
- Gain deep performance insights through dashboards and Logs Insights.
- Ensure cost efficiency by optimizing function execution and resource utilization.

By implementing an effective CloudWatch monitoring strategy, developers can enhance the reliability, performance, and cost-effectiveness of their AWS Lambda applications, ensuring a seamless experience for users.[16][17]

4. BEST PRACTICES FOR MONITORING LAMBDA WITH CLOUDWATCH

Monitoring AWS Lambda functions efficiently is critical to ensuring their performance, reliability, and cost-effectiveness. A well-structured monitoring approach using Amazon CloudWatch, combined with other AWS services, enables developers to detect issues early, optimize function execution, and manage resources efficiently. Implementing best practices ensures that Lambda functions run smoothly while minimizing unnecessary costs and overhead.[18][7]

A multi-layered monitoring approach is essential for gaining a complete view of Lambda function performance. Relying on a single monitoring tool is insufficient, as different monitoring solutions provide unique insights into function behavior. CloudWatch Metrics allows tracking of invocation counts, execution duration, error rates, and throttling events, which are fundamental for performance evaluation. CloudWatch Logs captures logs generated by the function, helping developers debug issues and analyze function execution

in real time. AWS Lambda Insights provides deeper system-level metrics, such as memory usage and CPU consumption, which are valuable for performance optimization. AWS X-Ray tracing enables distributed tracing, helping identify bottlenecks and dependencies affecting execution. Integrating third-party monitoring tools can further enhance observability by providing additional analytics and visualization capabilities. Combining these tools ensures a more comprehensive understanding of function performance and allows for proactive issue resolution.

Setting meaningful and actionable alerts is a crucial aspect of effective monitoring. Poorly configured alarms can lead to alert fatigue, where excessive notifications reduce responsiveness to critical issues. It is important to focus on key performance indicators, such as error rate, throttling occurrences, execution duration, and memory utilization, and configure CloudWatch alarms based on realistic thresholds that align with business requirements. Alerts should be routed through Amazon SNS or other notification services to ensure timely responses. Additionally, defining automated remediation actions, such as triggering a secondary Lambda function when an issue is detected, can help resolve problems before they impact users. By setting alerts that provide clear, actionable insights, teams can maintain high application availability while avoiding unnecessary distractions.

Regularly reviewing and updating monitoring configurations ensures that the system remains relevant as the application evolves. Applications change over time due to updates, increased traffic, or modifications in architecture, making it necessary to adjust monitoring settings accordingly. CloudWatch alarms should be periodically reviewed to ensure they still reflect critical conditions that require attention. Dashboards and visualization tools should be updated to display the most important real-time data, and log retention policies should be revisited to manage storage costs effectively. It is also essential to test alerting mechanisms periodically to verify that notifications are being received by the correct teams and response protocols are in place. Continuous refinement of the monitoring strategy allows organizations to stay ahead of potential issues and optimize function performance over time.[1][7][8]

Structured logging plays a vital role in making logs easier to search, analyze, and correlate with performance metrics. Traditional unstructured logs can be difficult to interpret, making troubleshooting time-consuming. Implementing structured logging formats, such as JSON, enhances the ability to filter and analyze logs using CloudWatch Logs Insights. Logging frameworks that support structured logging should be adopted to maintain consistency across log entries. This practice improves observability, enables faster debugging, and simplifies the correlation of log data with function execution details. With structured logs, teams can quickly identify errors, track execution patterns, and extract valuable insights without manual log parsing.

Monitoring memory usage is a key factor in optimizing Lambda function costs and performance. Since AWS Lambda pricing is based on memory allocation and execution time, excessive memory allocation can lead to unnecessary costs, while insufficient memory can result in performance bottlenecks. By analyzing memory utilization through CloudWatch metrics, developers can fine-tune memory settings to achieve an optimal balance between performance and cost. AWS Lambda Power Tuning tools can be used to test different configurations and determine the most efficient memory settings. By ensuring that functions have the right amount of allocated memory, organizations can reduce expenses while maintaining high execution efficiency.

Enabling AWS X-Ray tracing provides deeper insights into function execution by mapping out the request flow and highlighting performance bottlenecks. X-Ray allows developers to track requests across multiple AWS services, helping diagnose latency issues and service dependencies. By visualizing the execution path of a function, teams can identify inefficiencies in data processing and optimize workflows. X-Ray also supports annotations and metadata tagging, which can be used to filter traces and analyze specific execution patterns. Integrating X-Ray with CloudWatch further enhances monitoring capabilities by correlating trace data with performance metrics. This level of observability is especially valuable for troubleshooting microservices-based applications and serverless workflows.

Implementing log retention policies is important for managing storage costs while maintaining access to critical data. Without proper retention settings, CloudWatch logs can accumulate and result in high storage expenses. Organizations should define retention policies based on their operational and compliance needs. Development environments may only require logs to be retained for a few days, while production environments may need logs stored for several months. By setting appropriate retention periods, teams can ensure that historical logs are available when needed while avoiding unnecessary costs associated with long-term storage. Additionally, archiving important logs in Amazon S3 can provide an alternative storage solution for compliance and audit purposes.

By following these best practices, organizations can optimize their AWS Lambda monitoring strategy using CloudWatch, ensuring high availability, performance, and cost-efficiency. Effective monitoring not only helps in identifying and resolving issues proactively but also contributes to better resource management and system reliability.

Table 2: The Best practices for monitoring AWS Lambda functions using CloudWatch

Best Practice	Description	Tools/Techniques
Use a Multi-Layered Monitoring Approach	Combine multiple monitoring tools to gain a complete understanding of Lambda performance.	CloudWatch Metrics, Logs, Lambda Insights, AWS X-Ray, Third-Party Tools
Set Meaningful Alerts	Set up alerts for critical metrics based on realistic thresholds to avoid alert fatigue and to respond to significant issues effectively.	CloudWatch Alarms, SNS Notifications
Regularly Review and Update Monitoring Strategy	Continuously audit and adjust monitoring configurations as the application evolves, ensuring relevance and effectiveness.	CloudWatch Alarms, Dashboards, Log Retention Policies
Use Structured Logging	Implement structured logging formats (e.g., JSON) for easier search, analysis, and correlation with performance metrics.	CloudWatch Logs Insights, Logging Frameworks (winston, loguru)
Monitor Memory Usage for Cost Efficiency	Analyze memory usage metrics to optimize Lambda function cost and performance.	CloudWatch Metrics, AWS Lambda Power Tuning
Enable AWS X-Ray Tracing	Use X-Ray tracing to visualize execution paths, identify bottlenecks, and track dependencies for in-depth analysis and troubleshooting.	AWS X-Ray, CloudWatch
Implement Log Retention Policies	Define log retention periods to manage storage costs and ensure access to relevant logs for troubleshooting or compliance.	CloudWatch Log Retention Settings, S3 (for archiving)

5. CONCLUSION

Monitoring AWS Lambda functions with CloudWatch is a crucial aspect of ensuring their optimal performance, reliability, and cost-efficiency. As serverless applications scale, monitoring becomes even more essential to identify and resolve issues quickly. CloudWatch provides a comprehensive suite of tools to track key metrics, analyze logs, and set up automated alerts, all of which contribute to the smooth operation of Lambda functions. By utilizing CloudWatch metrics, such as invocation counts, execution duration, and error rates, you can gain real-time insights into your Lambda functions' performance. Logs capture detailed information about function execution, enabling efficient troubleshooting and debugging when issues arise.

Setting up alarms in CloudWatch helps you proactively manage potential issues by notifying you when specific thresholds are exceeded, allowing for a swift response before problems impact users. Creating dashboards further enhances visibility by providing an at-a-glance view of critical function metrics and logs, streamlining the process of monitoring multiple Lambda functions in one place.

By following best practices such as setting meaningful alerts, regularly reviewing monitoring configurations, enabling structured logging, and using AWS X-Ray for tracing, you can ensure that your Lambda functions are operating at peak performance. These practices not only help identify and resolve issues but also optimize resource usage, reducing unnecessary costs.

To deepen your understanding and put these best practices into action, it's important to explore the official AWS CloudWatch documentation. By fully leveraging CloudWatch's capabilities, you can implement a robust monitoring strategy that ensures the reliability, scalability, and cost-effectiveness of your serverless applications.

REFERENCES:

1. Diagboya, Ewere. *Infrastructure Monitoring with Amazon CloudWatch: Effectively monitor your AWS infrastructure to optimize resource allocation, detect anomalies, and set automated actions*. Packt Publishing Ltd, 2021.
2. Wadia, Yohan, and Udita Gupta. *Mastering AWS Lambda*. Packt Publishing Ltd, 2017.
3. Ghit, Raoul. *Monitoring serverless applications: an SLO-based approach*. MS thesis. 2021.
4. Routavaara, Ilkka. "Security monitoring in AWS public cloud." (2020).
5. Poccia, Danilo. *AWS Lambda in Action: Event-driven serverless applications*. Simon and Schuster, 2016.
6. Lin, Wei-Tsung, et al. "Tracking causal order in aws lambda applications." *2018 IEEE international conference on cloud engineering (IC2E)*. IEEE, 2018.
7. Kumar, Nunna Sravan, Manthri Sai Kiran, and P. Pagidala Raghunath. "Monitoring and Handling the Errors in Serverless Applications Using AWS Lambda."
8. Alpernas, Kalev, et al. "Cloud-scale runtime verification of serverless applications." *Proceedings of the ACM Symposium on Cloud Computing*. 2021.
9. Acharya, Bibek. "Building Serverless Application with AWS Lambda." (2020).
10. Sbarski, Peter, and Sam Kroonenburg. *Serverless architectures on AWS: with examples using Aws Lambda*. Simon and Schuster, 2017.
11. Dani, A., et al. "Case Study: Use of AWS Lambda for Building a Serverless Chat Application."
12. Patterson, Scott. *Learn AWS Serverless Computing: A Beginner's Guide to Using AWS Lambda, Amazon API Gateway, and Services from Amazon Web Services*. Packt Publishing Ltd, 2019.
13. Benedict, Shajulin. "Performance issues and monitoring mechanisms for serverless IoT applications—an exploratory study." *Smart Computing Techniques and Applications: Proceedings of the Fourth International Conference on Smart Computing and Informatics, Volume 1*. Springer Singapore, 2021.
14. Labouardy, Mohamed. *Hands-On Serverless Applications with Go: Build real-world, production-ready applications with AWS Lambda*. Packt Publishing Ltd, 2018.
15. Integrating Identity and Access Management for Critical Infrastructure: Ensuring Compliance and Security in Utility Systems. Suchismita Chatterjee. 2022. IJIRCT, Volume 8, Issue 2. Pages 1-8. https://www.ijirct.org/viewPaper.php?paperId=2412105
16. Suchismita Chatterjee , 2021. "*Advanced Malware Detection in Operational Technology: Signature-Based Vs. Behaviour-Based Approaches*", *ESP Journal of Engineering & Technology Advancements* 1(2): 272-279
17. McCarthy, Dave. "AWS at the Edge: A Cloud Without Boundaries." *International Data Corporation Accessed via https://d1. awsstatic. com/IoT/IDC-AWS-at-the-Edge-White-Paper. pdf* 1.1 (2020): 1-13.