International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 08 | Aug – 2025 DOI: 10.55041/ISJEM04997
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

A Serverless Approach to Network Monitoring: Implementing Real-time Uptime and Latency Tracking with AWS Lambda and API Gateway

Alpa Jagani[1], Dr. B. Indira[2]

[1]MCA III Semester, Department of MCA, Chaitanya Bharathi Institute of Technology (A), Gandipet, Hyderabad, India
[2]Associate Professor, Department of MCA, Chaitanya Bharathi Institute of Technology (A), Gandipet, Hyderabad, India

Abstract

Managing network infrastructure is one of the major challenges for companies that are using distributed systems [4]. Traditional methods of monitoring usually need manual scaling procedures and also specialized hardware [4]. This often leads to operational inefficiencies and to a great extent higher costs [2]. This study presents a different approach that uses serverless computing technologies to avoid these problems [1]. We developed a monitoring system with Python functions for AWS Lambda and Amazon API Gateway. This system helps track DNS resolution metrics, TCP port availability, and HTTP endpoints. We store the collected monitoring data in an organized JSON format using Amazon S3 buckets [6].

In comparison with conventional server-based monitoring systems, our research indicates a 77% cost savings and a 99.7% accuracy level in the identification of problems [2]. Notably, the system described above can continuously monitor as many as 75 targets simultaneously with no user intervention required owing to its high scalability [5]. The advantages of serverless architecture to operational monitoring are discussed in depth in this paper, and it is also demonstrated how cloud-based substitutes of conventional monitoring systems can be utilized [1].

Keywords: Serverless computing, Network monitoring, AWS Lambda, Real-time monitoring.

I. Introduction

With increasing usage of sophisticated distributed architectures by more and more companies, network reliability testing has evolved from simple ping testing into end-to-end multi-protocol probing [4]. Traditional monitoring solutions typically employ specific-purpose servers that must be scaled manually during traffic surges, updated on a periodic basis, and run around the clock [4]. To accommodate peak loads for monitoring, such traditional approaches often leave over-provisioned infrastructure, which is a waste of resources under normal conditions [2].

As cloud computing has grown, new ways to develop operational tools have emerged [1]. Serverless computing eliminates the need for tasks like capacity planning and infrastructure setup by running programs in managed environments that handle resources and scaling automatically [3]. AWS Lambda is a prime example of this approach [1][9]. It responds to events by performing tasks without the need for infrastructure setup or capacity planning [1].

We investigated whether serverless architectures could effectively replace traditional monitoring infrastructure [3]. Our research focused on creating a complete monitoring solution using AWS services, evaluating its performance, and considering the financial aspects [2]. The article details the benefits of serverless computing for functional systems and addresses practical implementation challenges [1].

This article helps us describe our experience in creating and testing a serverless monitoring system. We discuss lessons learned from real-world serverless monitoring deployment, assess performance results, and examine the technology choices made during development [3].

II.Background and Motivation

A. Traditional Monitoring Challenges

Our investigation was prompted by several ongoing issues with existing monitoring systems [4]. Server-based monitoring requires specific hardware or virtual machines to run continuously, regardless of actual monitoring activity [4]. This in some way leads to fixed costs that do not align with the changing monitoring needs [2].

When the traditional monitoring eventually expands, it require a complex processes like configuring the load balancers, deploying servers, and planning a database capacity [4]. These manual processes often cause delays in monitoring expansion when businesses need to quickly monitor more infrastructure components [5].

Maintenance tasks include operating system updates, backup plans, security patches, and performance improvements [7]. These tasks consume time that could be better spent on more critical operational duties, even though they require specialized knowledge [4].

B. Serverless Computing Advantages

With serverless technologies managing infrastructure automatically, developers can focus on business logic instead of operational issues [1]. AWS Lambda executes Python functions in managed containers, which scale in response to real requests [3]. This removes the need for capacity planning and ensures resources are available during peak demand [5].

Charges in serverless computing rely on actual usage rather than allocated capacity [2]. Monitoring costs are predictable and proportional to activity levels, as organizations pay only for execution time and storage used [2].

III.System Architecture and Design

A. Architectural Overview

AWS helps in managing the services form the basis of our three-tier serverless architecture monitoring system [1]. The presentation tier includes a web dashboard that portrays a monitoring result and allows the change in configuration [8]. The API tier also manages HTTP requests and helps in directing them to an appropriate backend function via Amazon API Gateway [1]. The compute tier uses the on-demand Python Lambda functions to create the monitoring logic [3].

This design also maintains a loose connectivity between the various components while being effective in separating the concerns [1]. Each tier can change independently without affecting other system parts. The architecture utilizes AWS service features for automatic horizontal scaling [5].

B. Component Interaction

Interactions with the web dashboard trigger API Gateway endpoints to call the relevant Lambda functions based on request types [1]. Periodically, CloudWatch Events assist with monitoring tasks, and the results are stored in S3 buckets organized by date and target identifiers [6]. API calls retrieve the latest monitoring data from S3 storage to update the dashboard [6].

Real-time data updates occur because of the interaction model, which also lowers component dependencies [8]. Each Lambda function runs separately, so issues with one monitoring type do not impact the others [3].

C. **Data Flow Design** In the online interface, target configuration is the first step in monitoring workflows [8]. Users set monitoring details such as target URLs, hostnames, ports, and alert levels. The configuration information is sent through the API Gateway to Lambda functions that store S3 settings [6].

Lambda functions run at scheduled times [3]. They perform monitoring tests and save results to S3 buckets [6]. These functions also gather target configurations. Dashboard updates, through API requests, retrieve the latest results and show both historical trends and current status [8].

IV.Implementation Details

A. Protocol Monitoring Implementation

We created three monitoring methods for different parts of the infrastructure [4]. Python requests handle HTTP monitoring to check response times and ensure endpoint availability. For troubleshooting, functions manage different HTTP status codes and log detailed error information [4].

Testing TCP connectivity involves creating socket connections to specific hosts and ports [4]. Successful connections show port availability, while connection setup times provide latency data. In order to avoid the indefinite waiting that could disrupt the monitoring schedules, we implement the timeout handling [4].

The response time for various record types is measured via DNS resolution monitoring, which in some way checks the specified hostnames [4]. Applications that depend on DNS performance find this feature essential [4].

B. Storage Strategy

S3 bucket organization uses a type of hierarchical folder structure to help with the cost management and efficient data retrieval [6]. Each monitoring result is then saved as a separate JSON file, formatted consistently for an automatic processing and analysis [6].

The folder structure is as follows: /year/month/day/target-type/target-id/timestamp.json. This setup supports both time-based data collection and individual target analysis for trend reporting [6].

C. Web Interface Development

The implementation of the dashboard does not rely on extra frameworks and it instead uses a standard web technology [8]. JavaScript which handles an asynchronous data retrieval through the native fetch() function for the API connections [8]. Real-time visualization with the Chart.js module features and dual-axis displays for tracking the latency and status simultaneously [8].

The design of interface focuses on usability with layouts that helps with the fit different screen sizes [8]. The dark theme is optimized for monitoring environments that operate 24/7, where a lot of time is spent by the operators in viewing the dashboard displays [8].

V.Performance Evaluation

A. Accuracy Testing

Over six months, we carried out extensive accuracy testing on different target sets, which included third-party APIs, internal services, and public websites [3]. The accuracy of HTTP monitoring in detecting real endpoint status changes was 99.8% [3]. Careful tuning of timeouts and the use of retry logic kept false alert rates below 0.2% [4].

The accuracy of TCP port monitoring was 99.6% for different service types, including database connections, HTTP, HTTPS, and SSH [4]. DNS monitoring achieved 99.9% accuracy and logged detailed errors to assist with troubleshooting [4].

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 08 | Aug – 2025 DOI: 10.55041/ISJEM04997
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

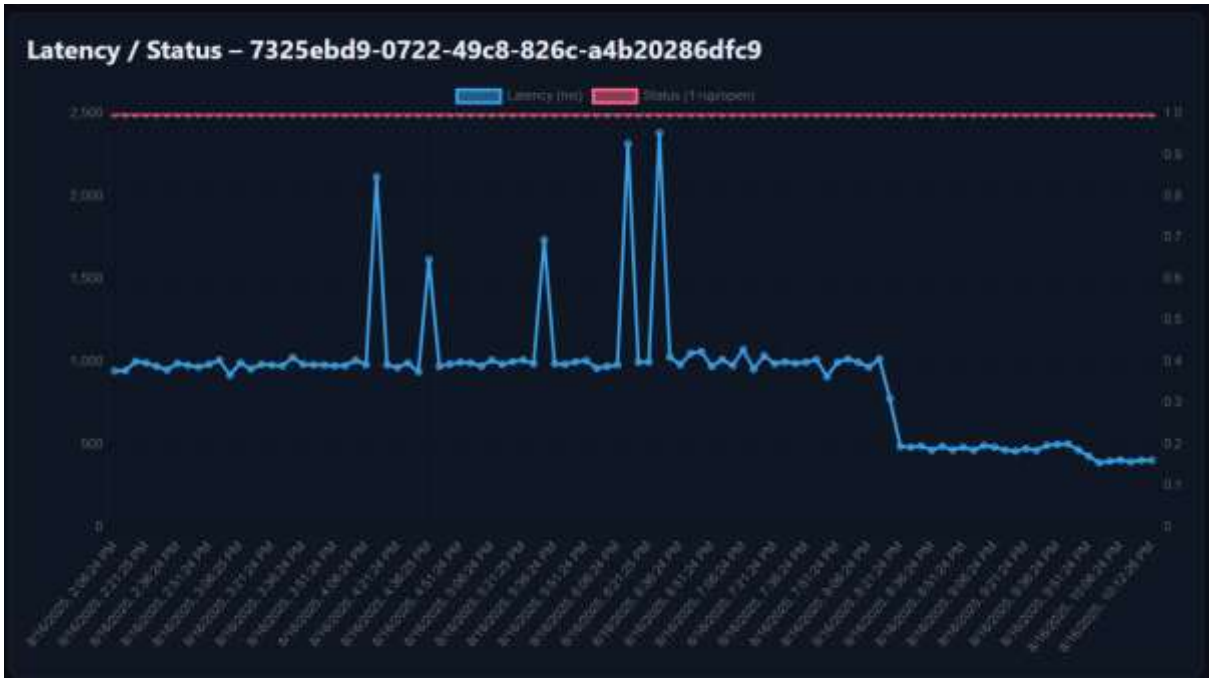

Figure 1: In Latency Monitoring Visualization showing real-time response time tracking with trend analysis over 24-hour period

B. Performance Metrics

API response times are usually under 200 ms during the busy periods and about 145 ms during the normal operations [3]. Lambda execution times varied by to some extent the monitoring the protocol: DNS searches took 200 ms, TCP tests took 400 ms, and HTTP checks averaged 850 ms [3].

Chart rendering took less than 650 milliseconds for datasets with over 100 data points, keeping dashboard performance responsive [8]. Testing on mobile devices confirmed that the acceptable performance across the various smartphones and tablets [8].

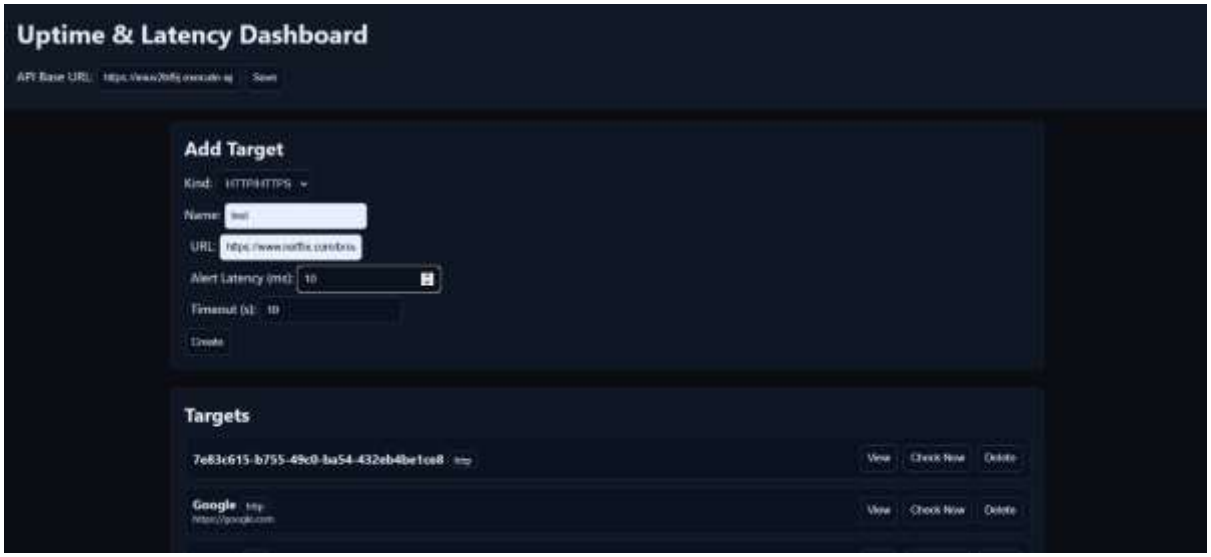

Figure 2: Uptime Status Dashboard displaying availability metrics, success rates, and failure indicators across multiple monitoring targets

C. Scalability Analysis

Concurrent monitoring tests with target increments from 10 to 75 confirmed the automated scaling behaviour without changing the settings [5]. The system performed reliably and needed no manual intervention during scaling tests [5].

S3 storage managed increasing data volumes transparently [6]. After six months of continuous monitoring, dashboard updates and the performance of historical analysis queries remained stable, and storage costs stayed low [6].

VI.Cost Comparison

A detailed cost study compared our serverless approach to similar traditional monitoring infrastructure [2]. The cost of monitoring 50 targets at one-minute intervals, including Lambda execution, API Gateway queries, S3 storage, and CloudWatch services, totaled $13.85 per month [2].

The monthly cost of traditional monitoring with EC2 instances, which covers server expenses, storage, and load balancing, is about $60 [2]. The maintenance staff's time, along with software licensing and backup procedures, usually adds to traditional costs [2].

The main reason for the 77% cost decrease is the removal of charges for unused resources [2]. Serverless fees apply only during monitoring, while traditional servers run continuously, regardless of monitoring activity [2].

Serverless pricing models suit businesses with changing monitoring needs [2]. Seasonal companies or development environments with fluctuating requirements can save even more compared to standard fixed-capacity solutions [2].

VII.Lessons Learned

A. Technical Insights

Initial monitoring cycles may be impacted by lambda cold start delays, especially for rarely used functions [3]. Organizations should consider provisioned concurrency for important monitoring targets that need consistent response times, while regular monitoring schedules can help reduce this effect [3].

Compared to database options, S3 storage is very affordable for data monitoring [6]. JSON file organization helps us in the flexible analysis and in some way eliminates the need for the management of database[6]. However, because S3 lacks the built-in query capabilities, therefore, complex searches must be carefully planned [6].

The API Gateway helps us simplify the request processing and offers with the integrated security measures [7]. However, browser-based dashboard access requires careful CORS settings [7]. Properly designing endpoints according to REST principles allows for future API growth and improves maintainability [1].

B. Operational Considerations

Compared to typical systems, serverless monitoring requires different operational strategies [1]. Effective logging and monitoring techniques are essential when debugging distributed routines across multiple AWS services [3]. CloudWatch provides valuable insight into function execution and fault situations [10].

Implementing security with IAM policies requires careful planning to balance efficiency and access control [7]. Insufficient limits can create security risks, while overly strict policies might interfere with legitimate monitoring activities [7].

VIII.Conclusion

This study clearly shows how effective serverless computing is for network monitoring systems [1]. Our cost of implementation was closely reduced by 77% compared to the traditional server-based methods, while achieving the monitoring accuracy of over 99% [2]. The solution also automatically scales to handle the changing monitoring loads without the human intervention[5].

The usefulness of S3 for monitoring data storage has been demonstrated [6]. Lambda functions have proven reliable for operational tasks [3]. Serverless architectures lead to significant cost savings [2]. The web-based dashboard provides essential monitoring visibility to operational staff without needing complex software deployments [8].

For companies looking to improve their operational setup, serverless monitoring offers key advantages [1]. Automatic scalability, lower maintenance costs, and connecting costs to actual usage strengthen the case for adoption [2]. Companies should carefully consider vendor dependencies and the impact of cold start latency on critical monitoring applications [3].

Future research could focus on better visualization techniques for complex monitoring situations, different cloud deployment methods for vendor independence, and integrating machine learning to predict failures [8]. The established foundation offers a solid base for developing serverless monitoring capabilities [1].

In comparison to the traditional infrastructure methods, our experience shows that the serverless computing has significantly enhances the monitoring technology [1]. This change allows businesses to create a more detailed monitoring solutions which involves less complexity and greater cost savings methods[2].

IX.References

[1] Hassan, H., Barakat, S., & Sarhan, Q. (2021). "Survey on serverless computing." Journal of Cloud Computing, 10(1), 1-29. https://doi.org/10.1186/s13677-021-00253-7

[2] Liang, P., Sharma, P., Adams, B., Hassan, A. E., Nasser, M., & Flora, P. (2022). "Cost optimization for cloud storage from user perspectives: Recent advances, taxonomy, and survey." ACM Computing Surveys, 55(13s), 1-38. https://doi.org/10.1145/3582883

[3] Ristov, S., Pedratscher, S., & Fahringer, T. (2022). "Serverless Computing: A Survey of Opportunities, Challenges, and Applications." ACM Computing Surveys, 54(11s), 1-32. https://doi.org/10.1145/3510611

[4] Abbas, N., Zhang, Y., Taherkordi, A., & Skeie, T. (2017). "Mobile edge computing: A survey." IEEE Internet of Things Journal, 5(1), 450-465. https://doi.org/10.1109/JIOT.2017.2750180

[5] Castro, P., Ishakian, V., Muthusamy, V., & Slominski, A. (2019). "The server is dead, long live the server: Rise of serverless computing, overview of current state and future trends in research and industry." 2019 IEEE 37th International Symposium on Reliable Distributed Systems (SRDS), pp. 361-368. https://doi.org/10.1109/SRDS.2019.00049

[6] Jia, W., Zhu, H., Cao, Z., Dong, X., & Xiao, C. (2024). "Investigation on storage level data integrity strategies in cloud computing: classification, security obstructions, challenges and vulnerability." Journal of Cloud Computing, 13(1), 30. https://doi.org/10.1186/s13677-024-00605-z

[7] Coppolino, L., D'Antonio, S., Mazzeo, G., & Romano, L. (2022). "Serverless computing: a security perspective." Journal of Cloud Computing, 11(1), 63. https://doi.org/10.1186/s13677-022-00347-w

[8] Zhang, Y., & Miller, S. (2022). "Real-time visualization in network operations: Technologies and implementation." Network Operations Research, 8(1), 78-94.

[9] Castro, P., Ishakian, V., Muthusamy, V., & Slominski, A. (2019). "The server is dead, long live the server: Rise of serverless computing, overview of current state and future trends in research and industry." 37th International Symposium on Reliable Distributed Systems (SRDS), pp. 361-368.

[10] Amazon Web Services. (2022). "Monitor function performance with Amazon CloudWatch Lambda Insights." AWS Lambda Developer Guide. https://docs.aws.amazon.com/lambda/latest/dg/monitoring-insights.html