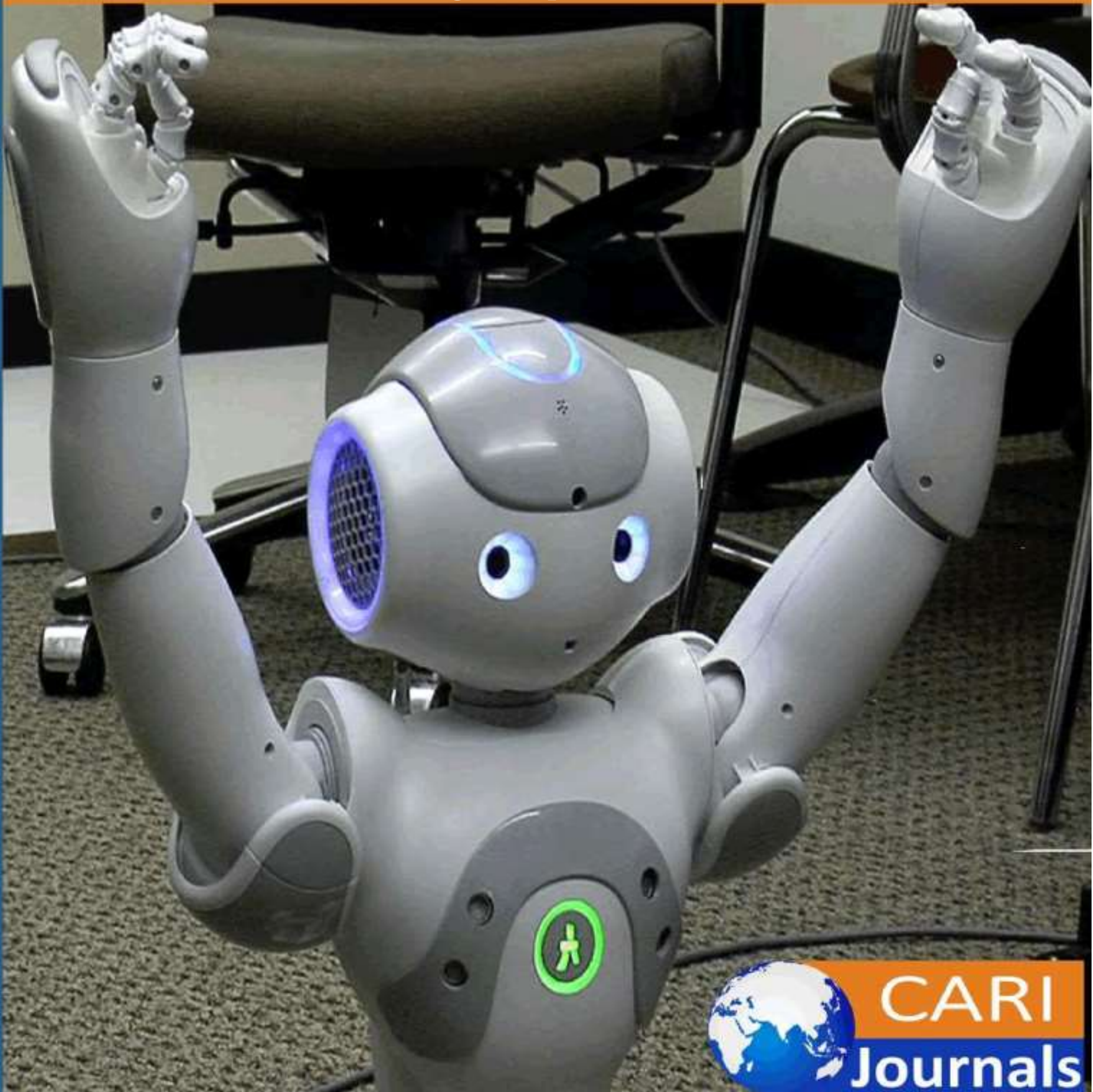
International Journal of
Computing and
Engineering
(IJCE)

Decoding Lambda API Architectures: Analyzing Monolithic Lambda Functions Versus Fine-Grained Single-Purpose Functions



CARI Journals

International Journal of Computing and Engineering

ISSN 2958-7425 (online)

Vol. 4, Issue No. 2, pp 20 - 28, 2023

www.carijournals.org

Decoding Lambda API Architectures: Analyzing Monolithic Lambda Functions Versus Fine-Grained Single-Purpose Functions

Balasubrahmanya Balakrishna

Richmond, VA, USA

https://orcid.org/0009-0000-1195-123X

Accepted: 2nd Dec 2023 Received in Revised Form: 16th Dec 2023 Published: 30th Dec 2023

Abstract

Purpose: With an emphasis on AWS Lambda specifically, this technical paper explores the trade-offs and architectural considerations between monolithic and single-purpose functions in server less computing systems.

Methodology: By examining the effects of using either a monolithic approach or a precisely calibrated, single-purpose function design, we hope to empower developers and architects. We investigate in-depth aspects, including resource usage, application monitoring, scalability, and performance. The approach strongly emphasizes a thorough examination of AWS Lambda's architectural issues, including the methods and resources utilized to produce insightful results.

Findings: The results emphasize carefully investigating server less computing's scalability, performance, and resource use, especially regarding single- and monolithic-purpose function architectures. These observations provide concise factors to take into account when developing server less applications.

Unique contributor to theory, policy and practice: The last section provides actionable insights to help developers of server less applications make well-informed decisions by condensing knowledge into useful suggestions for maximizing system responsiveness and resource management.

The author, coming from an AWS background, is committed to using these technologies to express the concept throughout.

Keywords: *AWS Lambda, Monolith Lambda Function, Single-Purpose Lambda Function, Lambda Function Security*

INTRODUCTION

Serverless computing has become a paradigm-shifting technology in recent years, providing pay-as-you-go pricing structures, reduced operational overhead, and unmatched scalability. AWS Lambda is a prominent player in the serverless market, offering a stable and adaptable environment for code execution without requiring server provisioning or management. A crucial issue in the development process is whether to use monolithic or single-purpose functions, as more entities use serverless architectures.

This technical article examines the trade-offs and architectural issues related to monolithic and single-purpose functions by navigating the complex terrain of serverless architecture, emphasizing AWS Lambda. It is critical to comprehend the effects of these architectural decisions as developers and architects work to maximize the serverless applications' performance, scalability, and resource consumption.

This analysis delves into the nuanced aspects that determine the success of serverless apps, surpassing the apparent similarities. We examine the potential outcomes of employing precisely calibrated, single-purpose functions compared to a monolithic approach. We break down performance, scalability, resource usage, and application monitoring to comprehensively understand how these decisions impact the system.

BACKGROUND

A. *AWS Lambda Functions: Configuration Essentials*

Serverless apps require the proper setup of AWS Lambda functions. Customizing the function to meet specific requirements entails setting the runtime, *memory allocation*, *timeouts*, *permission*, and *environment variables*.

a. Memory allocation affects performance, but timeouts prevent function overruns. Customize environment-related variables for different settings.

b. Permissions and triggers specify how a function interacts with other AWS services.

c. Configurations for error handling and logging facilitate troubleshooting.

d. Developers may optimize performance and resource consumption by fine-tuning functions for responsiveness, efficiency, and seamless interaction within the larger serverless ecosystem with AWS Lambda's various configuration choices.

DECODING MONOLITHIC AND SINGLE-PURPOSE LAMBDA APIS

A. *Decoding Monolithic Lambda APIs*

This section explores AWS Lambda's nuances, concentrating on monolithic functions. This brief investigation deconstructs the API architecture of using monoliths in serverless settings.

Fig. 1 illustrates a Monolith Serverless function consolidating multiple operations into a single handler. Recognizing that monolithic functions heighten application complexity by amalgamating various functionalities, one must meticulously consider distinct resources and HTTP methods for optimal performance. However, the execution role must extensively grant privileges to other AWS Services, contradicting the principle of least privileges. Additionally, passing method-specific variables for diverse HTTP methods renders environmental variables unmanageable. Lastly, configuring timeouts becomes imperative to accommodate the duration of the most time-consuming procedure.

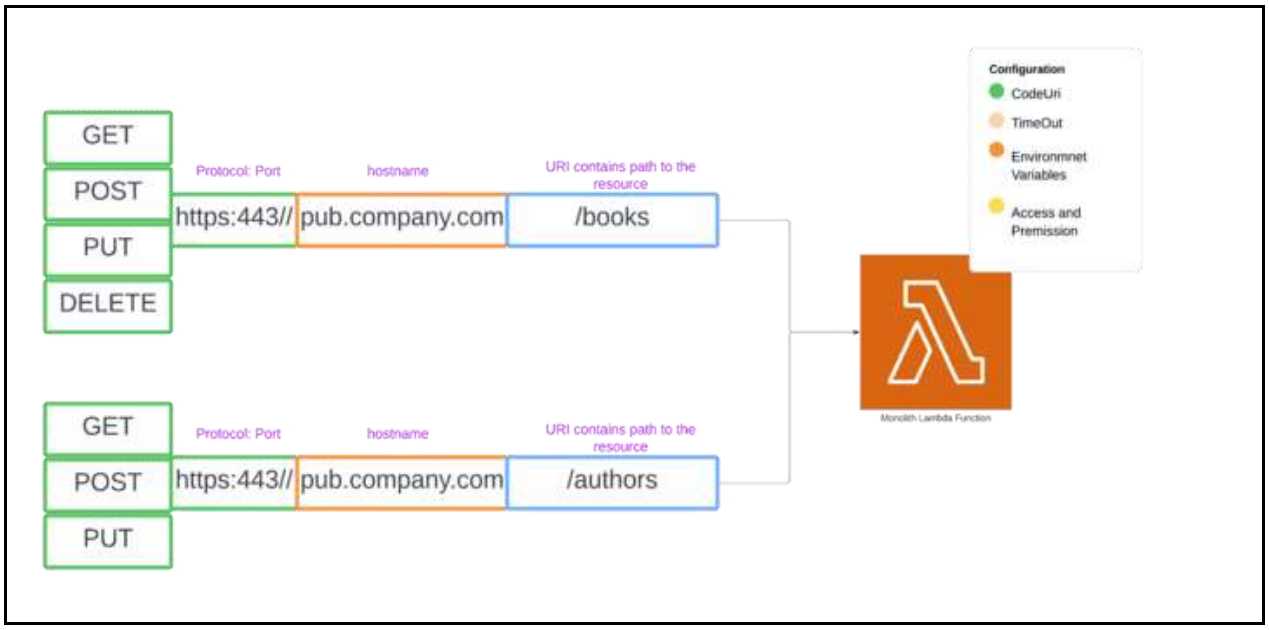


Fig.1 Monolith Lambda API

B. *Decoding Single Purpose Lambda APIs*

On the other hand, this section thoroughly examines AWS Lambda, explicitly focusing on single-purpose functions. Through the utilization of single-purpose functions, this concise analysis dissects the design of APIs within serverless settings.

In Fig. 1, we showcase a modified version of the Monolith Serverless function, and in Fig. 2, we closely examine the Single Purpose Serverless function. Significantly, we witness improved control over various Lambda configurations. Tailor the access and permissions of the single-purpose function to meet its specific requirements.

Each handler method now performs a unique task, enhancing code maintainability.

Furthermore, timeouts and environment variables are explicitly defined for every function, contributing to a more precise and effective setup.

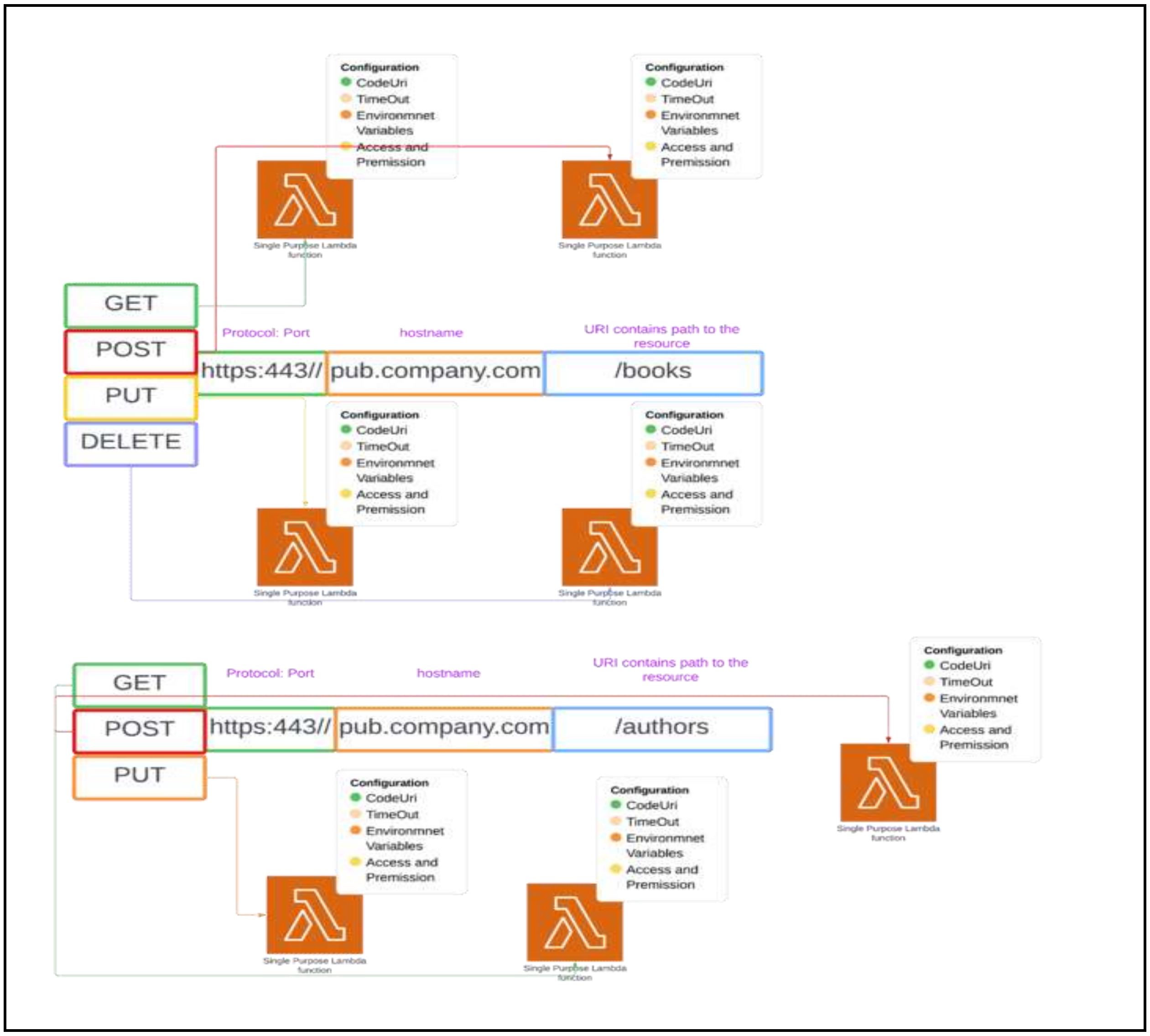


Fig.2 Single-Purpose Lambda API

LAMBDA ANALYSIS SERIES: PERFORMANCE, SECURITY, OBSERVABILITY

A. Performance Insights: Analyzing Single-Purpose and Monolithic Lambda APIs

The entire lambda function's performance may suffer significantly if the Monolithic Lambda API communicates with a downstream API with erratic response times. This response delay from downstream API raises AWS regional concurrency consumption, which impacts the monolithic lambda API's overall performance. It also cascades to other on-demand Lambda instances in the same AWS region, which may result in request throttling because there is insufficient capacity to start new instances and process new requests due to increased concurrency requirements because of the API's erratic response times.

The impact described above, resulting from increased concurrency consumption, becomes evident using the following calculation for synchronous Lambda function calls:

Concurrency = Duration (sec) x Request per Second (RPS).

Applying the formula[1] above, when configuring the Lambda API to accommodate *150 RPS* and observing a response time increase from *0.3 seconds* to *2 seconds*, the concurrency requirement escalates from *45* (150 x 0.3) to *300* (150 x 2).

Establishing a sensible Lambda API timeout remains effective in controlling over-consumed concurrency.

Nonetheless, as mentioned earlier, the timeout setting for a Monolithic Lambda API is generally configured to the most extended value for a specific resource and method.

Conversely, single-purpose Lambda APIs enable precise concurrency management and align timeouts with specific use cases. Shown in Fig.4 below

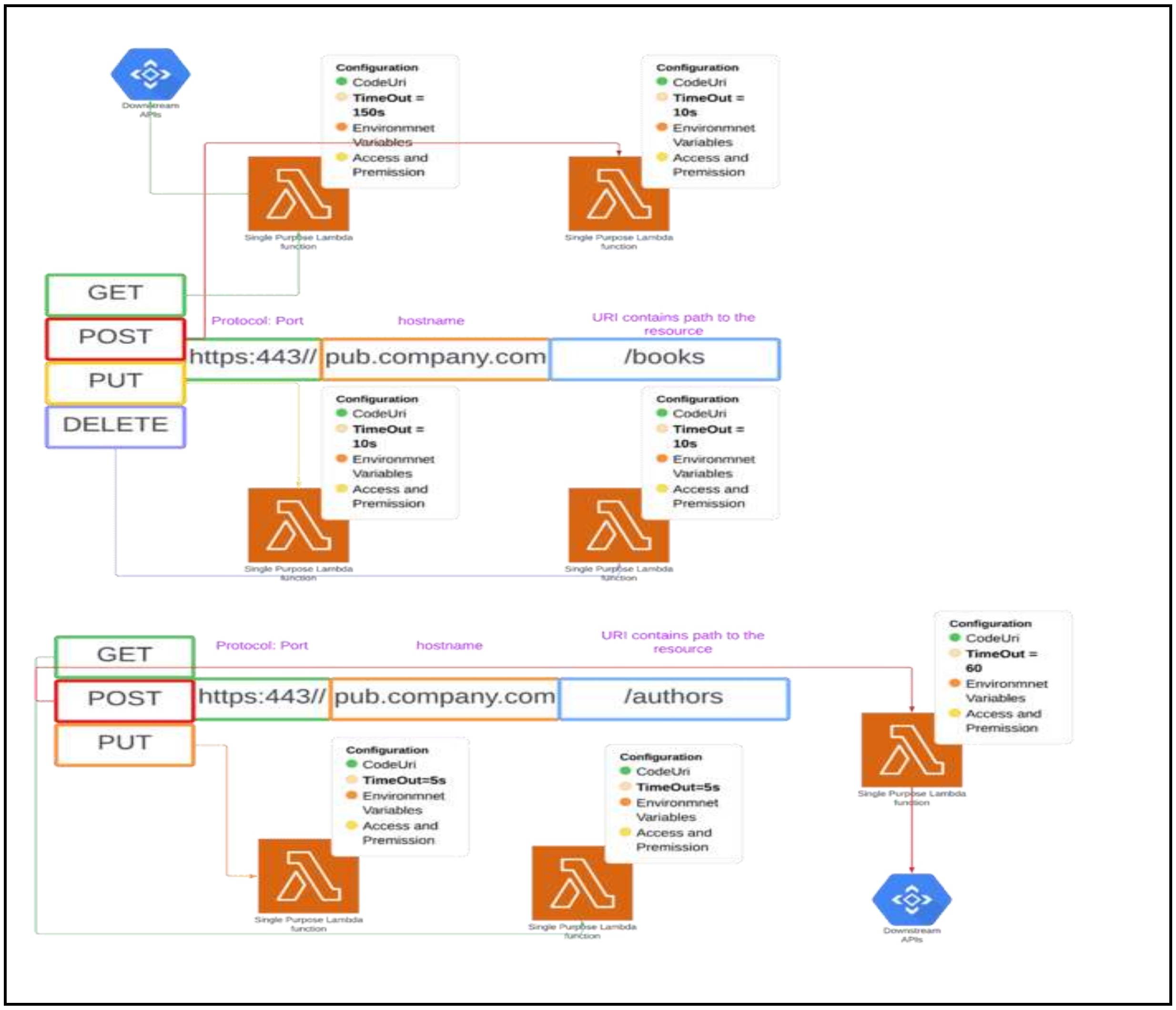


Fig. 4 Concurrency Management with Timeout set for each Lambda API

B. Security Insights: Analyzing Single-Purpose API and Monolithic Lambda API

The Execution Role[2] plays a crucial role in determining the AWS resources accessible through a Lambda API. Since the AWS Lambda function is limited to having a single assigned role, it forces an extensive role that contradicts the principle of least privilege. Fig. 5 illustrates an instance where the Lambda function necessitates broader permission settings to access and modify objects in an S3 bucket, emphasizing the challenge of maintaining granular permissions.

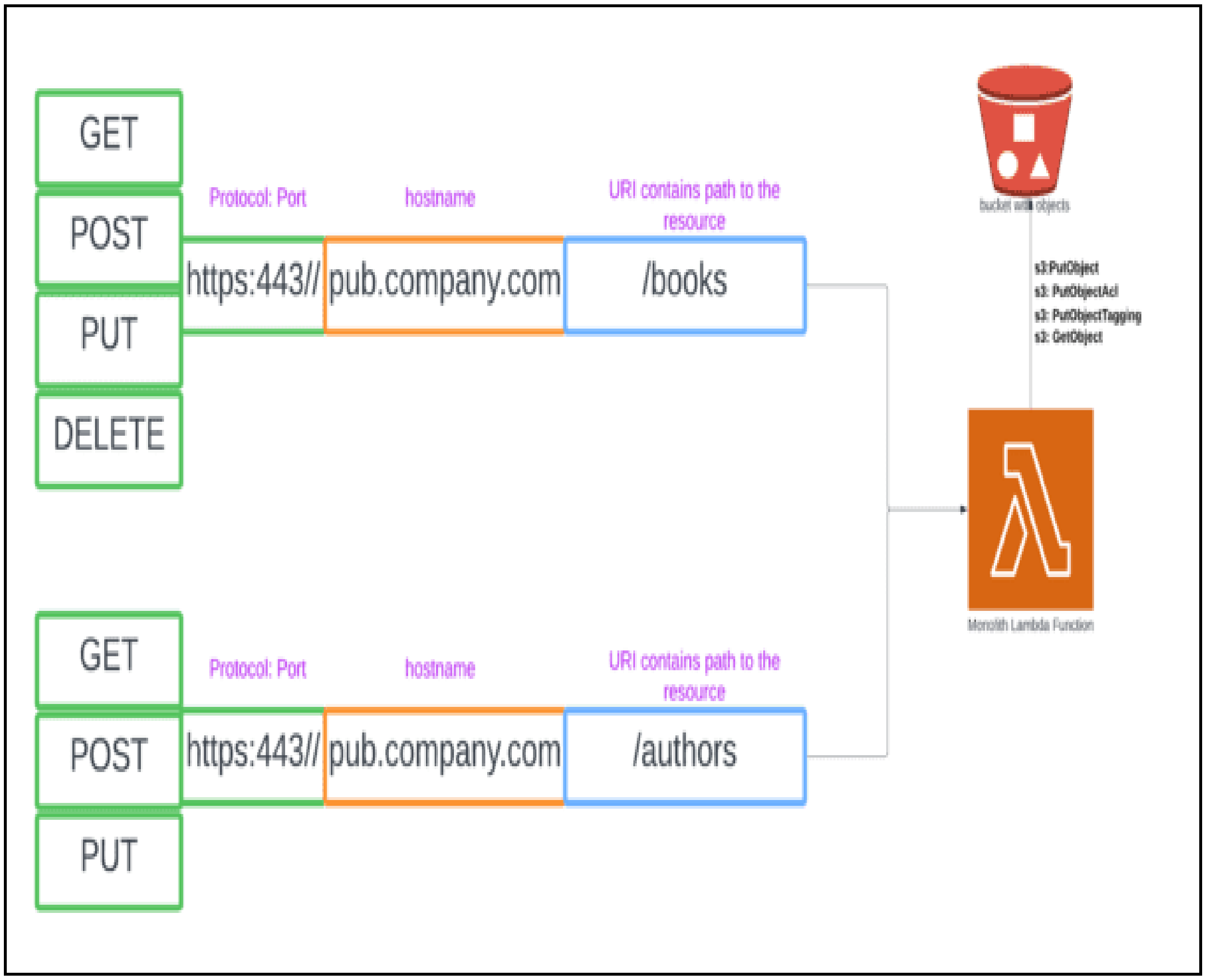


Fig. 5 Execution Role management with Monolithic Lambda APIs

Conversely, single-purpose lambda API can have a narrowed, function-specific execution role, as shown in Fig.6. This enables maintaining a minor action set necessary to fulfill a specific task.

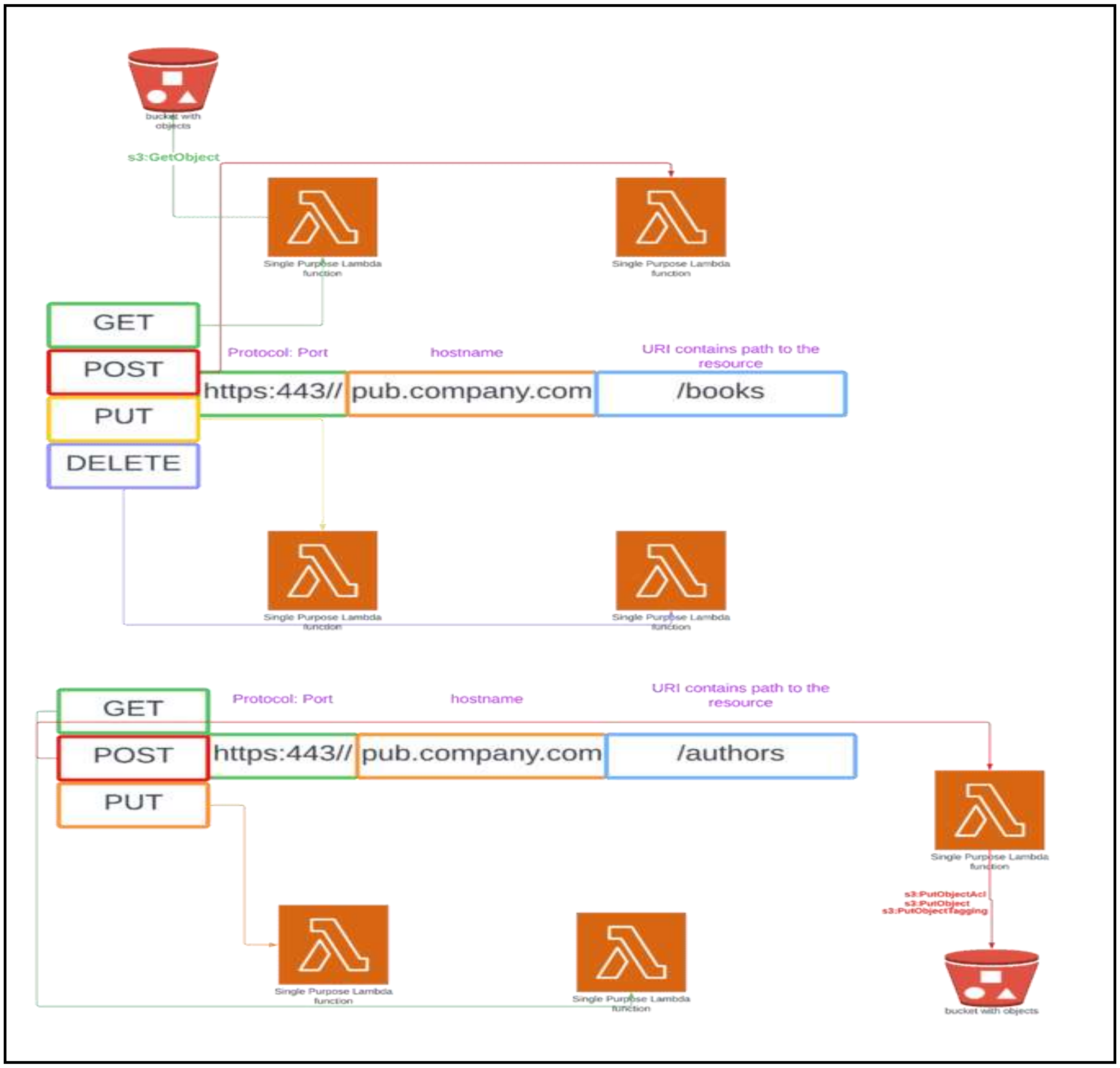

bucket with objects
s3:GetObject
Single Purpose Lambda function
Single Purpose Lambda function
GET
POST
PUT
DELETE
Protocol: Port
hostname
URI contains path to the resource
https:443//
pub.company.com
/books
Single Purpose Lambda function
Single Purpose Lambda function
GET
POST
PUT
Protocol: Port
hostname
URI contains path to the resource
https:443//
pub.company.com
/authors
Single Purpose Lambda function
s3:PutObjectAcl
s3:PutObject
s3:PutObjectTagging
Single Purpose Lambda function
Single Purpose Lambda function
bucket with objects

Fig. 6 Execution Role Management with Single-Purposed Lamabda APIs

C. Observability Insights: Analyzing Single-Purpose API and Monolithic Lambda API

I. Logging

Lambda API authors must implement middleware for intercepting requests and filtering logs in the context of a Monolithic Lambda API, shown in Fig.7. This middleware implementation is necessary because the AWS Lambda aggregates function logs into a single log group[3].

Fig. 7 Additional logging for Monolithic Lambda APIs

This middleware, on the other hand, is unnecessary for single-purposed functions. Having distinct log groups simplifies issue triage and avoids the need for additional middleware.

II. CloudWatch Metrics

CloudWatch metrics[4] are generated similarly at each function level. Custom metrics must be implemented by Lambda API authors to collect the route and other associated metrics, shown in Fig. 8.

Fig. 8 Additional Route Metrics for Monolithic Lambda APIs

In contrast, the Single-Purpose Lambda API can carefully scope errors, performance, and concurrency.

III. X-Ray[5] Traces

In the context of Monolithic Lambda API, Lambda API authors must include annotations and instrument functions. This information is critical for troubleshooting and aids in searching for traces connected with given routes. Fig. 9 depicts an instrumentation example.

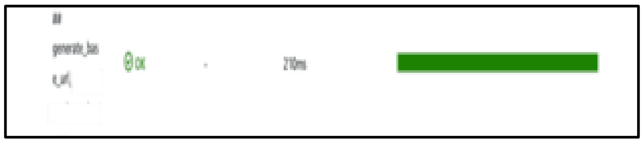


Fig. 9 Additional Tracing Segment for Monolithic Lambda APIs

On the contrary, adding annotations for Single Purpose Lambda API is a wonderful idea. However, it is not crucial because debugging difficulties are still manageable.

CONCLUSION

In conclusion, examining Monolithic Lambda Functions versus Fine-Grained Single-Purpose Functions exposes significant considerations for architects and developers engaged in serverless computing. Although monolithic functions offer simplicity, they entail trade-offs such as reduced control, added implementation and maintenance complexity, and diminished granularity. Conversely, single-purpose functions yield superior performance and security, leveraging precision and versatility. The trade-offs inherent in these approaches underscore the importance of aligning architectural choices with specific project objectives. As serverless computing expands, well-informed decisions regarding function design become paramount for optimizing application speed, scalability, and resource efficiency.

References

[1]AWS (n.d.). *How to calculate concurrency*. AWS Lambda. https://docs.aws.amazon.com/lambda/latest/dg/lambda-concurrency.html#calculating-concurrency

[2]AWS (n.d.). *Lambda execution role*. AWS Lambda. https://docs.aws.amazon.com/lambda/latest/dg/lambda-intro-execution-role.html

[3]AWS (n.d.). *What is Amazon CloudWatch Logs?* Amazon CloudWatch Logs. https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/WhatIsCloudWatchLogs.html

[4]AWS (n.d.). *Use Amazon CloudWatch metrics*. Amazon CloudWatch. https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/working_with_metrics.html

[5]AWS (n.d.). *What is AWS X-Ray?* AWS X-Ray. https://docs.aws.amazon.com/xray/latest/devguide/aws-xray.html