



This study guide is based on the video lesson available on TrainerTests.com

Study Guide: Synchronous and Asynchronous Invocation

Lambda functions in AWS offer powerful capabilities, and understanding how they handle invocations is crucial for designing efficient and cost-effective applications. This article delves into the concepts of synchronous and asynchronous lambda invocation using a simple analogy and real-world examples, shedding light on the benefits and trade-offs associated with each.

The Restaurant Analogy: Synchronous Invocation

To comprehend synchronous invocation, let's imagine a group of waiters in a restaurant. They take orders from customers, write them down, and personally wait for the chef to cook the meals. This is akin to synchronous invocation, where the invoker (waiter) pauses, ensuring the invoked action (food preparation) is completed in real-time. Though effective, it leads to downtime as the waiter waits for the food to be ready.

The Lambda Scenario: Synchronous Invocation

Translating this to Lambda, consider a scenario where an S3 bucket triggers a Lambda function upon object creation. This main Lambda function, akin to the waiter, then synchronously invokes two other Lambda functions. The challenge here lies in the fact that the main Lambda function cannot conclude until the invoked Lambda functions complete their tasks. Even if one function takes a significantly longer time, the main function continues running, incurring unnecessary costs.

Drawbacks of Synchronous Invocation

The synchronous approach, while ensuring real-time response and error handling, may lead to inefficient resource usage and higher costs. The main function is bound by the completion time of the invoked functions, potentially resulting in idle time and increased expenses.

Introducing Asynchronous Invocation

Now, let's explore asynchronous invocation, likened to a scenario where waiters drop orders on a counter and leave without waiting for food preparation. Similarly, in Lambda, the main function, after triggering other functions, does not wait for their completion. Instead, it places events in a queue, and the invoked functions process these events independently.

Efficiency with Asynchronous Invocation

In the asynchronous model, the main Lambda function concludes once it triggers the other functions, leading to more efficient resource utilization. While invoked functions may take varying durations, the main function doesn't incur costs during their execution, resulting in potential cost savings.

Downsides of Asynchronous Invocation

While asynchronous invocation offers efficiency, it lacks the real-time error monitoring present in synchronous invocation. If an error occurs in an invoked function, it will be retried, but the main function won't actively monitor its success. This places the responsibility on developers to handle potential errors in the distributed architecture.

Choosing the Right Approach

Selecting between synchronous and asynchronous invocation depends on specific use cases. Synchronous invocation suits scenarios where real-time response and error handling are critical. On the other hand, asynchronous invocation proves advantageous when efficiency and cost savings are prioritized, and developers can implement robust error handling mechanisms.

Conclusion: Balancing Act

Lambda invocation methods represent a balancing act between real-time responsiveness, error handling, and resource efficiency. Whether synchronous or asynchronous, understanding the nuances empowers developers to design applications that align with performance, cost, and error-handling requirements. As the digital landscape evolves, Lambda invocation strategies continue to play a pivotal role in optimizing cloud-based solutions.