**Lambda Cold Start**

**Introduction**

A "[cold start](https://docs.aws.amazon.com/lambda/latest/operatorguide/execution-environments.html#cold-start-latency)" in AWS Lambda is the initial delay that occurs when a function is invoked for the first time or after being idle. This includes downloading the function code and starting the execution environment. [Source](https://cloudtweaks.com/2023/11/lambda-cold-starts-how-to-fix-them/#:~:text=One%2520of%2520the%2520key%2520factors,help%2520you%2520manage%2520them%2520better.)

This results in a high initial response time for the first request to a "cold" lambda, while the response time for subsequent requests when the lambda is "warm" are lower. Though AWS says this typically occurs in under 1% of invocations, it is a real problem for the industry in developing cloud native serverless applications.

Reducing package size and using nodejs can reduce the time lost due to cold starts [source](https://medium.com/gitconnected/aws-lambda-cold-start-language-comparisons-2019-edition-%25EF%25B8%258F-1946d32a0244) , but are not a permanent solution.

To mitigate cold starts, [provisioned concurrency](https://docs.aws.amazon.com/lambda/latest/dg/provisioned-concurrency.html) can be used where a set amount of lambdas are always warm. However, this has the obvious downside of being more expensive and is like introducing a server in serverless. Usually, other warming techniques are used, such as periodically invoking the lambda to keep it warm, but this has the same downside of increased cost.

**Approach**

Depending on the cost, is it possible to keep a ***tiny*** provisioned instance always on, which knows whether there is a lambda that is currently warm to which the request can be directed? If there is no warm lambda, the instance handles the request itself and creates an event to start up the lambda. If there is a warm lambda, the request is redirected to the lambda. The lambdas can send events when they shutdown so that the provisioned instance can track their status.

**Factors**

The approach only makes sense if it is cheaper than keeping a provisioned concurrency instance of a lambda always warm. It is also feasible only for applications where the normal response time is lesser than the cold start time, since if the normal response time is in the order of seconds, few 100 ms may not matter. Also, an analysis of the average and max response times in provisioned concurrency vs cold start allowed vs this method should be done since there is an extra network call in routing the request to a warm lambda.

**Use case**

Develop a chat application that use different serverless functions:

1. To get the user query
2. To store the user query (this can be a dynamo db)
3. To handle the query and for providing the response.

The chat can simply take any user query and generate random outputs to demonstrate this particular functionality but it should offer a look and feel of real application.

**Requirements**

1. Create a user interface which displays various parameters of each serverless function in the application:
   1. Runtime of each function per request
   2. Cost estimates (if available)
   3. Response time of each function
   4. Number of request processed by each function
   5. The RAM / CPU of each function

2. The goal is to ensure that there is no cold start related response time by having a tiny provisioned ec2 on.