CSCI4145/5409: Serverless

Reminder: This is an individual assignment. You are not allowed to collaborate with anyone else when completing this assignment. You can borrow code and configuration snippets from internet sources that are not from students in this class, however that code must be cited and include comments for how you have modified the original code.

Introduction

This assignment will measure your understanding of some of the serverless mechanisms of our cloud provider AWS. This assignment assures us that you have attended the tutorials and learned about AWS Lambda and Step, or that you have found some other way to learn these services. In addition, you will have to do some self-learning to study how to use AWS Simple Queue Service (SQS).

Learning Outcomes

- Learn the benefits of serverless computing and apply that learning to implement a finite state machine in AWS Step Functions using serverless compute mechanisms (Lambda) and message buffering mechanisms (SQS).

Requirements

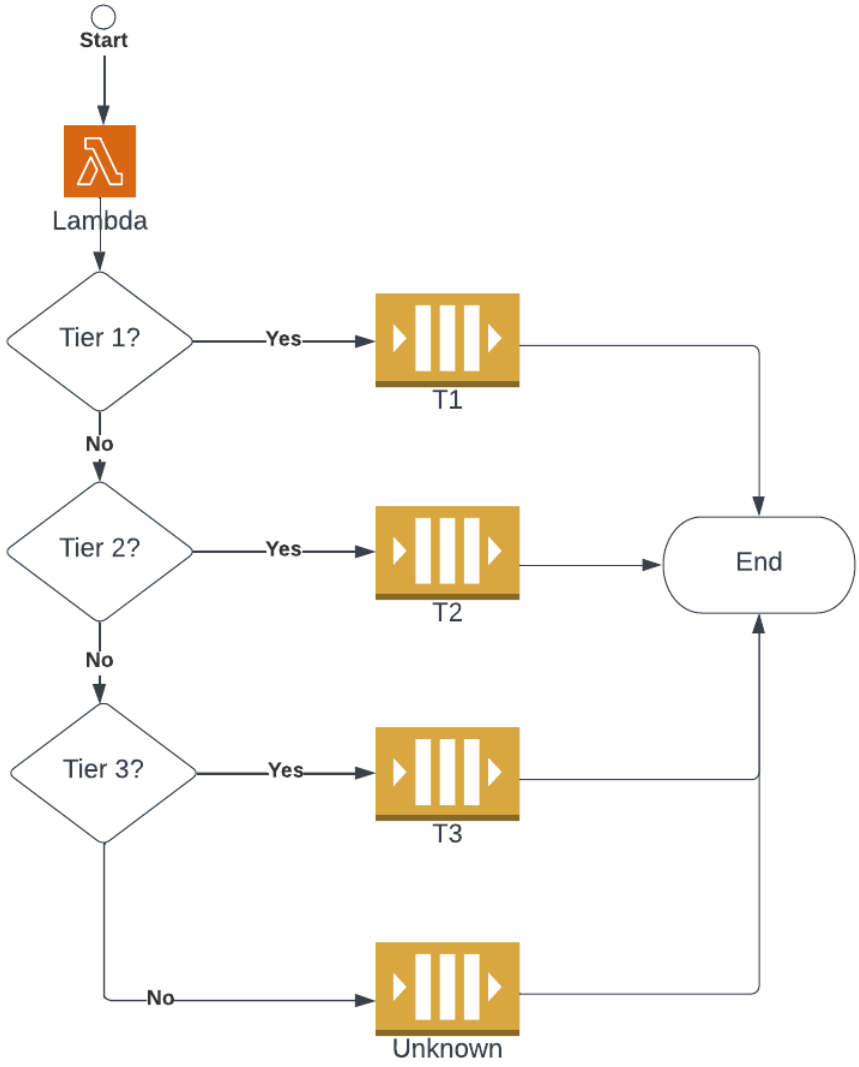
You will build the mock entry point to a producer/consumer model IT support application using serverless compute mechanisms. The purpose of your step function is to do preliminary sorting of new tickets into the support system for another process (that we won't create) to "consume" at a later time. The entry point to your system will be a step function, we can imagine in a real-world scenario a webpage that would launch the step function with input from a form that a user fills out when submitting a request for help from their IT department. The form would ask them various questions and then submit the data they enter to the step function via JSON.

Your state machine will use a lambda function to determine the "tier" of support required by the user:

- Tier 1: These are issues with accounts and passwords. For example, the user forgot their password and needs to have it reset. Or the user needs to set up a new account.
- Tier 2: These are issues with hardware that need an IT person to physically attend to the user's issue with their computer or printer.
- Tier 3: These are issues from high priority users that need to be addressed urgently.

Your step function's job is to sort incoming requests into one of 4 SQS queues: T1, T2, T3 and Unknown. The Unknown queue is the "catch all" queue to deposit requests that don't fit the criteria for tier's 1 to 3.

Here is a rough state diagram for your system:



Incoming JSON:

The JSON sent to your step function will have the following format:

```
{
      "email": "rhawkey@dal.ca",
      "message": "I do not remember my password! Please help!"
}
```

Lambda Task State:

The purpose of your lambda is to do the "thinking" of the state machine. It will do two things:

1. Parse the incoming JSON sent to it from the step function to determine what tier of support the user needs.

2. Return the tier to your step function so that your step function can use choice states to put the message in the correct queue.

Determine tiers as follows:

- Tier 1: Any message with the words "account" or "password" in the string.
- Tier 2: Any message with the words "computer", "laptop" or "printer" in the string.
- Tier 3: Any message with email address "rhawkey@dal.ca"
- Unknown: Anything that doesn't fit the other categories.
- **Note: The tiers are a hierarchy, e.g. if a message contains both "laptop" and sent from "rhawkey@dal.ca" then it is a tier 3 message.**

Once your decision has been made you should return some kind of JSON (that you define) to your step function for it to use in choice state decision making.

You can write your lambda in whatever language you like best.

Step Function Choice States

Your choice states will act on the input path to their state (which is the output path of your Lambda task state). They will decide which queue to place the message on. How you place the messages onto the SQS queues is up to you. You will need to do some self-learning to figure out how to create SQS queues, and how to insert messages onto them from step functions. You may be able to do it directly from inside the step function, or you may have to write a lambda function to do this task for you. This is left for you to figure out as part of the assignment.

The result of every execution of the step function is the same message that is passed into the step function is then placed on the appropriate SQS queue.

How To Submit

First, execute your step function with the following inputs:

```
{
     "email": "rhawkey@dal.ca",
     "message": "I do not remember my password! Please help!"
}

{
     "email": "person@dal.ca",
     "message": "I do not remember my password! Please help!"
}

{
     "email": "person@dal.ca",
     "message": "My laptop computer broke and I cannot use my
printer! Please help!"
}
```

```
{
     "email": "person@dal.ca",
     "message": "The registrar system is down! Please help!"
}
```

Then submit the following to Brightspace:

- The JSON definition of your step function in a text file.
- The code for your lambda function(s) (not zipped, just the .py or .js files)
- A screenshot for each of your 4 SQS queues showing each one having the correct message on it.

Marking Rubric

Your submission will be marked by the TAs reviewing your code and screenshots:

- **Your lambda correctly classifies incoming requests – 40%**
- **Your step function correctly sorts requests into correct SQS queues – 40%**
- **Your SQS queues display the correct incoming message for each execution – 20%**