AWS Cloud Developer (Javascript) Coding Assessment

The goal of this assessment is for us to take a look at how you write and organize your code. This is the most technical part of the Valence interview process, and we want to make sure that you are not under any stress and have plenty of time to complete this assessment.

It should take you approximately 2-3 hours to complete, and we promise we'll never ask you any white-boarding pop-quizzes during the culture fit interview that follows 🙂

Guidelines

- Take as much time as you need
- Feel free to use an IDE of your choice
- Use Google as you normally would
- Use of third party frameworks is encouraged
- Clearly define dependencies within `package.json`
- No need to write any unit tests

To remove any potential bias we will scrub any personally identifiable information from your submission before handing it over to an engineer for a code review.

Evaluation Criteria

We are looking at the following things:

- Does the app compile and run? Are all the base features implemented?
- How well is your code organized? (Hint: Break your code into meaningful functions and compose them in the endpoint hook)
- Are your variable and class names descriptive? Do they make sense? (Tip: don't name things with a single character)
- Are you using proper encapsulation?

- Is your memory management sound? Are there any memory leaks?
- Are you adhering to the Javascript code style guidelines and naming conventions?
- Are all public methods and classes appropriately documented?
- Did you implement any bonus features? Points for appropriate asynchronous operations (network/disk access), and parallelization as necessary
- Less code = more 👍👍. Brevity is the soul of wit 🙂

Note that we will test your function by running it from the command line and comparing to some expected output, so make sure your formatting is correct (order of entries doesn't matter).

Submission

When you are finished, please perform the following steps:

1. Remove any personally identifiable information from the project and file headers (such as the name of the author, provisioning information, etc).
2. Zip up the code, excluding any build folders or compiled artifacts.
3. **Email your submission to your Valence recruiter.**

AWS Lambda Function

We are using Serverless as the framework to enable running of this Lambda function locally without the need to deploy to AWS.

What to Do

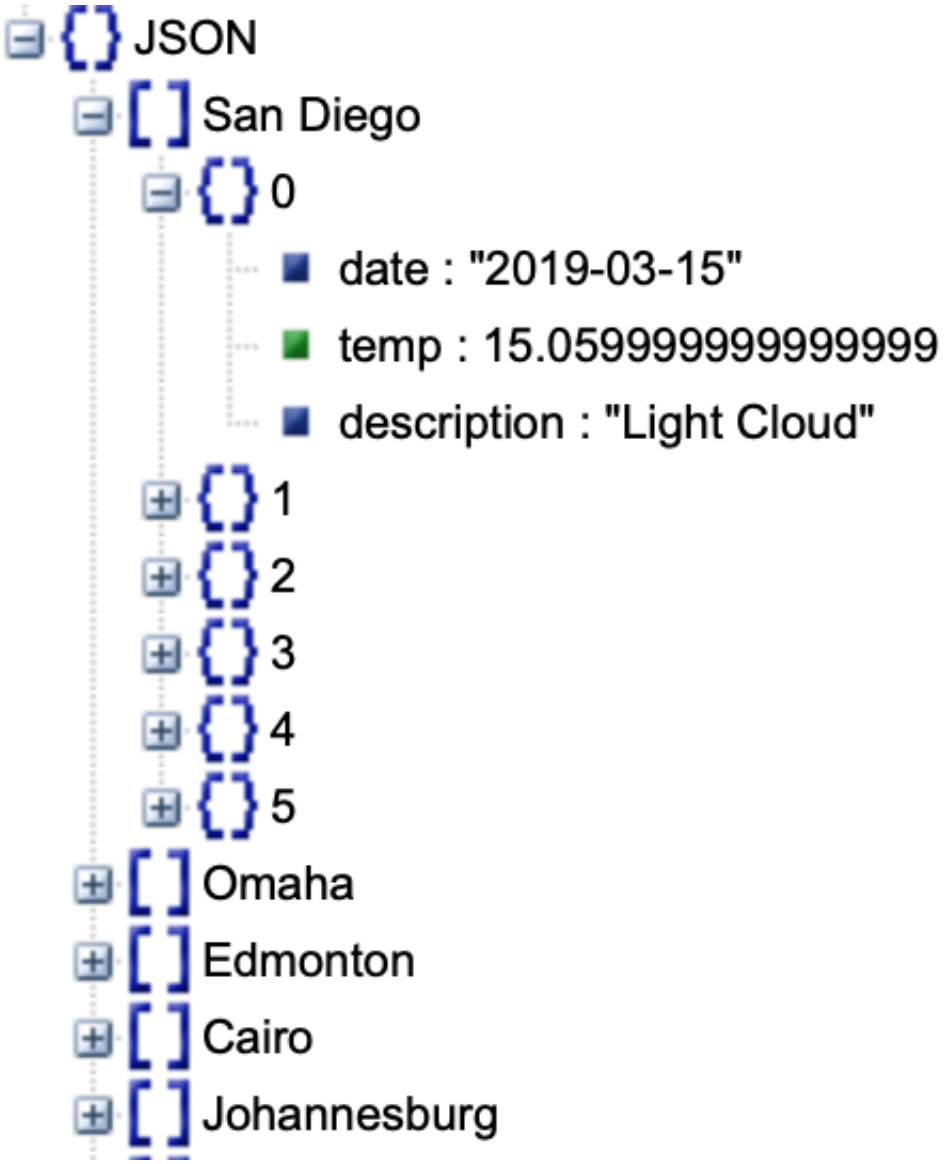
Your mission, should you choose to accept it, is to develop an AWS Lambda function (using the serverless framework setup described below), to act as an API endpoint that reads a CSV file from S3, queries weather data from an external API for each of the locations, and returns a single JSON document describing the weather across all locations.

The API you will be querying is the *MetaWeather* service - it provides an aggregated forecast across many different weather services into a single 5-day forecast for each location. You will parse the CSV passed in to your API endpoint, and for each location name, you need to devise a scalable and reliable strategy to acquire the 5-day forecast data for each location name, and return it in a single, useful JSON document format intended to be consumed by a client application, or data processing service.

Your output should be a simplified version of the weather data, across multiple locations as defined in the CSV file:

```
{
    "location_1_name" : [
        {
            "date" : "some_date",
            "temp" : "some_temp",
            "weather" : "some_weather"
        },
        ...
    ],
    "location_2_name" : [
        ...
    ],
    ...
}
```

Example:



Resources

- Locations CSV
- MetaWeather API
- Using Promises

Notes

- You must fetch the CSV file from S3, it can't be locally embedded into your function.
- Not all data within the CSV needs to be used. You should decide which data is relevant, and which isn't.
- Make sure to read the API documentation extensively - querying the required forecast data for this assignment is not completely straightforward and **may require multiple queries** to get the data you need.
- Proper **parallelization and usage of asynchronous operations** is extremely important in contexts such as this - as you know, network operations are blocking, and an efficient program doesn't spend

unnecessary time waiting.

Setup

Installation

1. Install Serverless
2. Download Serverless Template

Using an IDE (VSCode)

You are welcome to use an IDE to develop the Lambda function. One IDE that works really well with the Serverless framework (allowing for local debugging) is VSCode. You will want to install the serverless-vscode extension if you want to use it. This should work equally well on a Mac and on Windows (most of us use Macs).

To open the template and create a debug configuration, do the following:

1. In **VSCode** go to **File** > **Open** and point to the template folder.
2. Open the `handler.js` file by double-clicking it. This step allows VSCode to infer that you'll be working with JS/Node.
3. Click the **Debug** tab (leftmost menu).
4. You will see a drop-down near the Run symbol that says "No Configuration". Click on that dropdown and select Add Configuration. Select Node.js as your environment.
5. A `launch.json` file will be automatically generated in the hidden `.vscode` folder within your project's root.
6. Replace the contents of `launch.json` with the following.

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "type": "node",
            "request": "launch",
            "name": "Launch Lambda",
            "program": "${workspaceFolder}/node_modules/serverless/bin/serverless",
            "args": [
                "invoke",
                "local",
                "-f",
                "myFunction"
            ],

            // Required, otherwise Serverless thinks stdin is a stream and tries to r
ead options from it.
            "console": "integratedTerminal"
        }
    ]
}
```

Usage Local

```
serverless invoke local -f myFunction -l
```

Deploy

```
serverless deploy
```

Usage Remote

```
serverless invoke -f myFunction -l
```