AWS Lambda Guide

Created	@February 4, 2022 5:33 PM
Last Edited Time	@February 14, 2022 11:08 AM
Type	Technical Spec
Status	In Progress 🙌
Created By	
Last Edited By	
Stakeholders	
Tags	

Summary

This document explains the steps to setup a lambda function, and how we can deploy our code and run these serverless functions automatically based on triggers.

Steps to create a Lambda Function

Please run and test your Python code locally, and make sure that the code is error-free. Then encapsulate all your code into methods that can be called by the lambda handler discussed in the next step. Then create a new function in your code called ‘lambda_handler’, and it would look something like this:

```
def lambda_handler(event, context):
    # TODO implement
    webscraping()
    results = candidate_match()
    print(results)
```

Please note, the structure of your python file would be like this:

1. package import statements
2. lambda_handler function
3. other functions and methods that are to be called inside the handler

Now, once you have made the above changes then open you terminal and run the following commands to prepare the zip file for AWS lambda:

In your terminal go to a directory of your choice for creating a virtual environment. Then run the following commands:

1. *virtualenv venv*

If the above commands gives an error then run this:

python

>> pip3 install virtualenv

Press Ctrl+Z and then go back to command line and try step 1 again.

2. cd venv
3. cd bin
4. source activate

Now, the venv virtual instance would be active.

5. Now, look at your Python script, and all the packages you are importing, first install them here in this virtual environment, by using the following command:
   pip3 install (package name)

   For example: pip3 install requests
6. Once all the packages are installed then open this venv directory in your finder, and go to /dir/venv/lib/pythonxx/site-packages
7. Copy your python file containing your code here, and please make sure to name your Python file “lambda_function.py”
8. Once copied zip all the contents of the ‘site-packages’ folder, and name it ‘lambda_function’.

AWS Console

In the Aws console, search and open AWS Lambda, then on the left pane click on “Functions” and then click on “New Function”.

1. Give your function a name.
2. Select the runtime, which should be your Python runtime like Python 3.9.
3. Then create the function.
4. In the function now, click on the “Code” tab.
5. There is a button on the right hand corner, called “Upload Code”, click on that to upload your “lambda_handler.zip” file.
6. Upload and click on Save.
7. Please note if the zip is over 50 MB, you have to put it in an S3 bucket and then load it from there to Lambda. (Same process)
8. Once loaded, then click on “Test” tab.
9. If we don’t require any input parameters to our lambda function then just create the default test and save it with any name of your choice.
10. Then hit the “Test” button.
11. If everything was done correctly AND packages were loaded properly then the execution should be successful.

Creating an AWS Layer

When packages are not properly loaded into lambda and lambda function gives an error because of incompatibility then we have to create a lambda layer for that package. Please note, in your lambda function in the Code tab, you have a ‘Layers’ option on the bottom, click on ‘Add a new layer’ and then select from AWS’s default available layers. Based on your runtime, you would have ready to use layers, which you can add to your function and test your code again in lambda, if these AWS layers don’t fix the problem then you can go ahead and create your own layer as described below:

Following are the steps to create a layer:

- Let’s create a layer for a package called ‘numpy’.

  Go to this webpage https://pypi.org/ and search for your package, once found then click on ‘Downloads’ and download the package with your python runtime and which has ‘manylinux’ and ‘x86-64’ in the name. By default lambda runs on Linux, therefore we need the package that is compatible for that OS.
- Download the .whl file and then decompress it.
- Now throw all the contents of the decompressed file, into a new folder called ‘python’.
- Create a zip of the python folder and also name it python.zip.
- Now in the Lambda console, create a new layer, and then upload your zip using the upload option available.
- Choose your desired runtime and select the architecture which should be x86-64.
- Create the layer.
- Once, created add this layer to your lambda function and now your lambda function should work as intended and would not throw any errors related to pandas not available for use.

Triggers

There are many kinds of triggers that one can set on a lambda function. Triggers can be set using S3, Redshift, SNS, etc.

Steps to setup a trigger on AWS Lambda Function:

1. In your Lambda function, click on the “Configuration” tab in the center of the screen. Once opened please click on the “Triggers” option on the left hand pane.
2. Inside triggers, click on “create a new trigger”.
3. From the drop-down select the trigger you want to setup to trigger your S3 function. In this example, let’s setup a trigger which activates an instance of lambda when a “.csv” file is dropped in a specific S3 bucket.
4. Search and select S3 in the drop-down.
5. Then choose the S3 bucket, or specify the path to the bucket. (You can get the path from the S3 AWS console)
6. In the suffix field put ‘.csv’.
7. For the action click on the drop-down and select ‘PUT’ object, this woud trigger the lambda only if a csv object is “PUT” into the S3 bucket.
8. Save the Trigger, and your trigger should be up and running.

Best Practices to setup a Lambda Function

Introduction

There are a couple of ways you can deploy your code in Lambda, and let’s discuss these ways here with some advantages and disadvantages.

Important Points to Note:

A few important things to note is that your main python code file should be named ‘lambda_handler.py’ for it to work inside Lambda, if you change this filename then it also has to be changed inside the AWS Lambda function’s configuration. So to keep it simple let’s name is lambda_handler.py.

There would always be a ‘lamda_handler(event, context)’ function inside your code, and this would act as the main function of the script. The ‘event’ parameter contains any information related to variables that were passed to the function or actual events that would trigger the lambda, for example, the details about the file dropped in S3 that triggered the lambda.

How to Deploy your Code

There are a couple of ways that you can use to deploy code into lambda. Once of which includes us writing a lambda function locally inside a virtual environment (the steps for which were detailed above), testing it, and then deploying a zip of the python file and all packages to the S3 Lambda Function, and then testing it on the AWS Lambda console.

The other recommended way which reduces a lot of time deploy time, and makes the process easier for the developer to leverage. This method allows us to upload all our dependencies first in the form of a layer, and then use the lambda code editor inside AWS Lambda console to write out our code, and test the function on the AWS Lambda console seamlessly. This saves a lot of time as we don't have to make code changes locally everytime, zip up all the contents again and then upload them to S3 just to test the function, and also it allows us to actually see our code in Lambda, whereas the complete zip does not make our code visible on the Lambda Code editor for editing and review. This process of adding dependencies as layers is described below:

1. Create a virtual environment locally in any folder locally.
2. For that follow the steps:
3. In a Terminal, write the following commands, and this guide assumes you are creating your environment inside "/Users/sharjeel/Documents/Work-Dir/".

   ```
   cd /Users/sharjeel/Documents/Work-Dir
   virtualenv venv
   cd venv
   ls
   cd bin
   source activate
   pip install packages (Run this command multiple times to install all your packages
   needed for deployment)
   cd "/Users/sharjeel/Documents/virtualenvironments/venv/lib/python3.9/site-
   packages" (Please note your might differ based on your python runtime, but there
   would be a site-packages folder in your mac)
   ```

4. Now, cd to your directory:

   ```
   cd /Users/sharjeel/Documents/virtualenvironments/venv/lib/python3.9/
   ```

5. Then run the command to zip all contents INSIDE of 'site-packages' into 'python.zip', using the command:

   ```
   zip site-packages/  python.zip
   ```

6. Open your lambda function in the lambda console, and then click on "Code".
7. On the bottom there is an option of "Add a Layer", click that and follow the steps on adding a layer. Give your new layer a name of your choice, and upload your "python.zip" file to this layer, and then choose the Python runtime and then Create the layer.
8. Now, in your lambda function, you should be able to add this new layer to your code, using the same Layer window on the bottom of the "Code" tab.
9. Add this layer and then in Lambda code editor, start writing your lambda function.
10. As mentioned above, 'lambda_handler' function must be there as it would act as your main function, whereas other functions and and modules can be added to the console, by using the code editor, and can be imported referenced in the 'lambda_handler.py' module.
11. Once, done run and test the lambda function, before deploying to production.

Important Resources

https://towardsdatascience.com/python-packages-in-aws-lambda-made-easy-8fbc78520e30

The above link is another way of creating a lambda layer, with the use of Cloud9 which is a cloud-based IDE of AWS.

https://medium.com/@samme/setting-up-python-3-6-aws-lambda-deployment-package-with-numpy-scipy-pillow-and-scikit-image-de488b2afca6

The above link is important to create an Amazon Linux docker container that contains all the dependencies installed and then load that container in lambda as a layer.