**Event-Driven Data Ingestion in AWS (S3 to S3)**

Let’s say you have data coming into S3 in your AWS environment every 15 minutes and want to ingest it as it comes. The best approach for this near real-time ingestion is to use [AWS lambda function](https://docs.aws.amazon.com/lambda/latest/dg/welcome.html). To demonstrate how to develop and deploy lambda function in AWS, we will have a look at a simple use case of moving file from source S3 to target S3 as the file is created in the source.

AWS lambda supports a few different programming languages. I will use Python for this example. To deploy lambda functions, you need to package the modules used in the function. For some reason, modules downloaded and packaged from Windows sometimes do not work (e.g. pyscopg2 never works even if I download the binary version). From my experience, Python lambda development is the best to be done in Linux or Mac.

In this example, we will use Amazon Linux EC2 instance to develop the package and deploy it from there. You can of course use your local Mac or Linux machine for this.

To deploy lambda functions, I usually use the deployment framework called Serverless, which makes deployment much easier. Here, I would like to use the manual deployment from the AWS documentation [here](https://docs.aws.amazon.com/lambda/latest/dg/with-s3-example-deployment-pkg.html#with-s3-example-deployment-pkg-python). Doing the deployment manually will teach you how the lambda function works. I think it is good to try it at least once before using more automated solutions.

**Summary Steps**

1. Launch Linux EC2 instance
2. Install Python 3 and create virtual environment
3. Install required packages
4. Code Lambda function
5. Package function & modules into a zip file
6. Create Lambda Function Role
7. Deploy Lambda
8. Test function manually
9. Configure S3 source bucket as event source
10. Execute the function by pushing file to the source bucket

**Steps**

1. **Launch a Linux EC2 Instance**

Micro instance is usually good enough for developing lambda function. After deployment, you can terminate the instance. Make sure to choose the right security group and subnet so that the instance can access the appropriate resources.

1. **Install Python 3 and create virtual environment**

This step and the next step can be bootstrapped ([How to Launch EC2 With Bootstrap in AWS](https://www.mydatahack.com/how-to-launch-ec2-with-bootstrap-in-aws/)). For Centos, Redhat or Amazon Linux, follow the steps here: [How to Install Python 3 and Create Virtual Environment in Centos, Redhat and Amazon Linux](https://www.mydatahack.com/how-to-install-python-3-and-create-virtual-environment-in-centos-redhat-and-amazon-linux/). For Ubuntu, follow the steps here: [How to Install Python 3 and Create Virtual Environment In Ubuntu](https://www.mydatahack.com/how-to-install-python-3-and-create-virtual-environment-in-ubuntu/).

1. **Install required packages**

Make sure to activate the virtual environment where you want to develop the lambda function. The installed packages will be in the site-packages folder. You need to pack all the site-packages content into a zip file. AWS SDK for Python (like boto3 or botocore) is pre-installed for Lambda function and you do not need to pack them. Although we do not use any other package for this code example, let’s pack psycopg2 and import it in the function to make sure the packaged modules work.

Note that you need to install the binary version for psycopg2.

[cc lang="python" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

source your-virtual-environment-path/activate

pip install psycopg2-binary

[/cc]

1. **Code Lambda Function**

The PUT API call to the source S3 is the event to trigger the function. In Python, the event data is processed as dictionary. See how the input file looks in the AWS documentation [here](https://docs.aws.amazon.com/lambda/latest/dg/with-s3-example-upload-deployment-pkg.html). Make sure to change the parameter for your environment. Save this as inputFile.txt.

[cc lang="text" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

[/cc]

From the event input, you can grab bucket name and key to specify the newly created file in the source bucket.

The function grab the data from S3 and copy to a local tmp folder. Then, push it back to the target bucket. You can add transformation logic here if you need to.

[cc lang="python" tab\_size="4" lines="-1" theme="GeSHi" line\_numbers="false"]

[/cc]

Once the function is created, use the input file (inputFile.txt) to test the function. You can add this for local testing.

[cc lang="python" tab\_size="4" lines="-1" theme="GeSHi" line\_numbers="false"]

f = open('inputFile.txt', 'r')

event = json.load(f)

target\_bucket = 'lambda.test.target'

download\_file(event, 'hello')

[/cc]

1. Package function & modules into a zip file

Now that your function is working, you can package the modules and code. You need to go to the site-packages folder for your virtual environment first then pack all the modules and then pack the python file.

[cc lang="bash" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

cd /home/ec2-user/python3/lambda/lib/python3.6/site-packages

zip -r9 /tmp/MoveS3Data.zip \*

zip -g MoveS3Data.zip MoveS3Data.py

[/cc]

1. **Create Lambda Function Role**

By using IAM, you need to create a role for Lambda function. Follow the steps in the documentation [here](https://docs.aws.amazon.com/lambda/latest/dg/with-s3-example-create-iam-role.html).

1. **Deploy Lambda**

Once you have the zip file and lambda role, you can deploy the function. Make sure that you have the correct role and region selected. Handler has to be your-python-file-name.main-function-name.

[cc lang="bash" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

[/cc]

1. **Test the function manually**

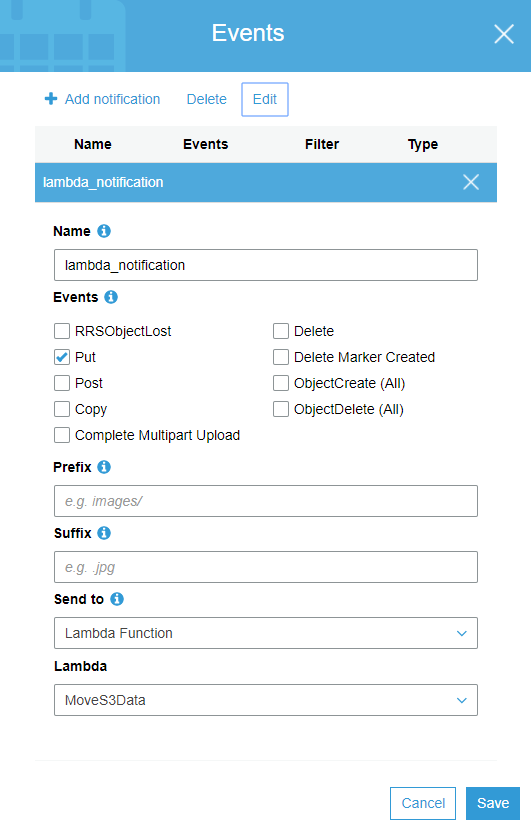
When your deployment is successful, invoke the function with the input file (inputFile.txt) set as payload.

[cc lang="bash" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

[/cc]

1. **Configure S3 source bucket as event source**

Go to the source bucket and configure event as below. In this example, my source bucket is lambda.test.source.



1. **Execute the function by pushing file to the source bucket**

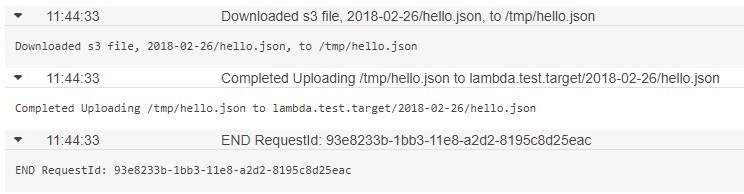
Let’s push a file to S3 with AWS console and check if the function moved the data into the target bucket. [Here](https://docs.aws.amazon.com/cli/latest/reference/s3/cp.html) is the s3 copy command reference.

[cc lang="bash" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

aws s3 cp your\_file\_name.json s3://lambda.test.source/2018-03-01/your\_file\_name.json

[/cc]

You can check the execution outcome in CloudWatch, too.



Yay!