In this lab, our goal is to create a system that will automatically generate and store thumbnails of images that are uploaded with a notification on any failures. We will accomplish this by using Python (with the nifty Pillow library), Lambda, S3 storage buckets, and Simple Notification Service.

[Task 0: Setup](#_ayhu9tn2ul5u)

[Step 1: Download application](#_bxv2b3si5qbe)

[Step 2: Test the application locally](#_h1oz0slkemi3)

[Task 1: Create s3 buckets using the CLI](#_wvgkrw5imk0v)

[Step 1: Create source and destination buckets](#_akozbza59ypv)

[Step 2: Upload the test image](#_26pvaooala3)

[Task 2: Update code to read and write to our buckets](#_iypijkdh4cto)

[Task 3: Create our lambda dependencies](#_20z05rth0bzu)

[Task 4: Build lambda and deploy](#_d9xueotbl81v)

[Step 1: Create a zip file containing our code and it's dependencies](#_pxcwu63323cn)

[Step 2: Create a new lambda function](#_1xtayv7k9xhn)

[Step 3: Test your function](#_efnll8spgk1y)

[Task 5: Update lambda to read output bucket name from environment variable](#_fpp330urqyid)

[Task 6: Trigger conversion on upload](#_1663w4o17euh)

[Step 1: Update lambda to expect the s3 event format](#_ap155m2ugjbw)

[Step 2: Add S3 trigger to lambda function](#_ulb0grsf5dnj)

[Step 3: Test the trigger by uploading to your bucket](#_eupll1g95rb8)

[Task 7: Add SNS notifications on Thumbnail creation](#_6yq9bb5j1brn)

[Step 1: Create SNS topic](#_m8y5vu8wxcjw)

[Step 2: Setup an email subscription to your email](#_wqnyk6l6iry6)

[Step 3: Have the lambda notify you when run](#_ql5srd575mrx)

[Step 4: Test your lambda](#_dtbvzlheaklb)

[Task 8: Coding Bonus](#_phbtb6297rjp)

[Step 1: Properly handle non-image files!](#_9xq9wuba27o6)

### Task 0: Setup

#### Step 1: Download application

Lucky you! I have already written a thumbnail conversion script.

|  |
| --- |
| git clone <https://github.com/redrivet/thumbnail-converter-lambda.git>  cd thumbnail-converter-lambda |

#### Step 2: Test the application locally

|  |
| --- |
| pip3 install --target . -r requirements.txt  python3 thumbnail.py |

If everything worked, you should now see the new thumbnail. Compare the different sizes!

|  |
| --- |
| # View original  ls -alh flower.png  # View thumbnail  ls -alh /tmp/flower.jpg |

### Task 1: Create s3 buckets using the CLI

#### Step 1: Create source and destination buckets

|  |
| --- |
| aws s3 mb s3://**<YOUR-NAME>**-original-images  aws s3 mb s3://**<YOUR-NAME>**-thumbnail-images |

#### Step 2: Upload the test image

Upload ***flower.png*** in the repo to your original images bucket

### Task 2: Update code to read and write to our buckets

Using boto3, we need to have our app download from our original-images bucket and upload the result to our thumbnail-images bucket.

Currently, the app looks for the source image in the current directory and stores the result in /tmp/thumbnails. We can continue to use /tmp/ for this intermediate storage. Use the boto3 documentation: <https://boto3.amazonaws.com/v1/documentation/api/latest/guide/s3-examples.html>

Step 1. Update the input\_bucket and output\_bucket to point to your buckets

Step 2. Implement the download function

Step 3. Implement the upload function

Step 4. You can test your upload by checking if your bucket contains the thumbnail after running the script

### Task 3: Create our lambda dependencies

We must create a role to be used by our lambda so it can write objects to our thumbnail bucket.

Step 1. Examine the role-policy-document.json

Step 2. Create the role

|  |
| --- |
| aws iam create-role --role-name **<username>-s3-thumbnail-lambda** --assume-role-policy-document file://role-policy-document.json  # Give our role permission to write to cloudwatch logging  aws iam attach-role-policy --role-name **<username>-s3-thumbnail-lambda** --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole" |

Step 3. Find the ARN for AmazonS3FullAccess policy and, using the CLI, attach it to our role, or bonus points if you create a policy that only allows our function to write to your specific thumbnail bucket.

### Task 4: Build lambda and deploy

#### Step 1: Create a zip file containing our code and it's dependencies

|  |
| --- |
| pip3 install --target . -r requirements.txt  zip -r9 function.zip . |

#### Step 2: Create a new lambda function

Using your new zip file and the following settings, create your lambda function

name: <your name>-thumbnail

handler: "thumbnail.handler"

role: [The arn of the role you created previously]

runtime: "python3.7"

#### Step 3: Test your function

This may be easiest via the console as you will need to create a test *event* that will be passed to your lambda, similar to the following:

|  |
| --- |
| {  "bucket": "jrrickerson-original-images",  "object": "flower.png"  } |

### Task 5: Update lambda to read output bucket name from environment variable

Update code and iterate. Think of ways you can test your changes quickly without having to update the lambda and functionally test it. Make sure you add your env variable in your lambda configuration.

### Task 6: Trigger conversion on upload

#### Step 1: Update lambda to expect the s3 event format

<https://docs.aws.amazon.com/AmazonS3/latest/dev/notification-content-structure.html>

Update code and iterate. Think of ways you can test your changes quickly without having to update the lambda and manually testing it.

#### Step 2: Add S3 trigger to lambda function

Trigger a new thumbnail to be created any time an object is uploaded. Hint, this is easily done when looking at your lambda configuration in the console.

#### Step 3: Test the trigger by uploading to your bucket

### Task 7: Add SNS notifications on Thumbnail creation

#### Step 1: Create SNS topic

#### Step 2: Setup an email subscription to your email

#### Step 3: Have the lambda notify you when run

Hint, add the following policy to the lambdas role allowing it to publish to SNS

|  |
| --- |
| aws iam attach-role-policy --role-name jrrickerson-s3-thumbnail-lambda --policy-arn "arn:aws:iam::aws:policy/AmazonSNSFullAccess" |

#### Step 4: Test your lambda

You should receive an email within 5 minutes. When done, update your lambda to only put a message on the SNS topic on failure.

### Task 8: Coding Bonus

#### Step 1: Properly handle non-image files!