

AWS training - Serverless

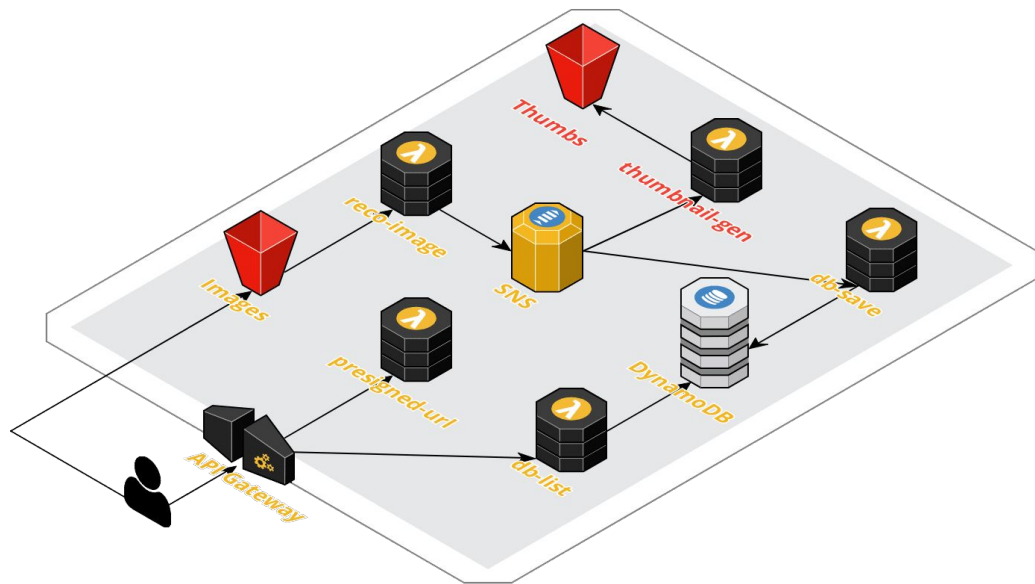
Hands On #7 - S3 | Lambda - Generate thumbnail



Overview

This Hands-on is composed of 2 parts:

1. [S3 Part](#) : new bucket to store thumbnails
2. [Lambda Part](#) : new function to create thumbnail from images analyzed by recognition function



Let's go! | S3 Part

Go to Virginia region

N. Virginia ▼

Create a S3 bucket in order to store the thumbnails:

- **Bucket Name:** serverless-training-thumb-`<xxx>`
- **Policy:** see Hint 3 to set the JSON policy

Once done -> Go to **Lambda part** to create your function

Context:

In this part we create a S3 bucket storing thumbnails of the images which have been analyzed by the recognition function.

Documentation:

https://docs.aws.amazon.com/fr_fr/lambda/latest/dg/with-s3-example-deployment-pkg.html#with-s3-example-deployment-pkg-python

<https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html>

Hint 1

S3 - Create a new bucket serverless-training-thumb-**<xxx>** and configure public access settings (uncheck all)

1

Name and region

2

Configure options

3

Set permissions

4

Review

Name and region

Bucket name ⓘ

serverless-training-thumb-1no

Region

US East (N. Virginia) ▾

Copy settings from an existing bucket

Select bucket (optional)4 Buckets ▾

✓

Name and region

✓

Configure options

3

Set permissions

4

Review

Note: You can grant access to specific users after you create the bucket.

Public access settings for this bucket

Use the Amazon S3 block public access settings to enforce that buckets don't allow public access to data. You can also configure the Amazon S3 block public access settings at the account level. [Learn more](#) ↗

Manage public access control lists (ACLs) for this bucket ⓘ

☐ Block new public ACLs and uploading public objects (Recommended) ⓘ

☐ Remove public access granted through public ACLs (Recommended) ⓘ

Manage public bucket policies for this bucket ⓘ

☐ Block new public bucket policies (Recommended) ⓘ

☐ Block public and cross-account access if bucket has public policies (Recommended) ⓘ

Manage system permissions

Do not grant Amazon S3 Log Delivery group write access to this bucket ▾

Hint 2

S3 - Under "Permission" > "Bucket Policy", set the following policy (replace YOUR_BUCKET_NAME)

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Sid": "AllowPublicRead",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::YOUR_BUCKET_NAME/*"
    }
  ]
}
```

Let's go! | Lambda Part

Create a lambda function having these properties:

- **Name:** py-aws-lambda-thumbnail-gen
- **Runtime:** Python 3.6
- **Trigger:** SNS
- **Role:** serverless_lambda_role
- Add two new **Environment variables** containing the source and thumbnail S3 bucket names
 - Name: imgSrcBucket
 - Value: serverless-training-img-<xxx>
 - Name: thumbDestBucket
 - Value: serverless-training-thumb-<xxx>
- Upload the **Function code** from the S3 bucket:
<https://s3.amazonaws.com/awstacktraining-serverless-resources/code-templates/py-aws-lambda-thumbnail-gen-template.zip>

Once done -> Go to [Testing part](#) to test your Lambda

Context:

In this part we create a new Lambda function in charge of resizing as a thumbnail an image analyzed by the recognition function and store it in the thumbnail S3 bucket just created.

This function is triggered by the SNS results topic.

Documentation:


https://docs.aws.amazon.com/fr_fr/lambda/latest/dg/with-s3-example-deployment-pkg.html#with-s3-example-deployment-pkg-python

Hint 3

Lambda Creation - create a new
Lambda function
“py-aws-lambda-thumbnail-gen”
using the existing role
“serverless_lambda_role”


Author from scratch ☒

Start with a simple "hello world" example.



Blueprints ☐

Choose a preconfigured template as a starting point for your Lambda function.



AWS Serverless Application Repository ☐

Find and deploy serverless applications published by AWS, AWS partners, and other developers.

**Author from scratch** [Info](#)**Name****Runtime**

You can select a supported AWS Lambda runtime or provide your own runtime as part of the function deployment package or Lambda layer after creating the function.

RoleDefines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more](#) about Lambda execution roles.**Existing role**

You can use an existing role with this function. Lambda must be able to assume this role, and the role must have Amazon CloudWatch Logs permissions.

Hint 4

Lambda Configuration - Upload the function code from the S3 link URL given in “Let’s Go” section

Function code [Info](#)

Code entry type	Runtime	Handler Info
<div>Upload a file from Amazon S3 ▼</div>	<div>Python 3.6 ▼</div>	<div>lambda_function.lambda_handler</div>
Amazon S3 link URL Paste an S3 link URL to your function code .zip.		
<div>https://s3.amazonaws.com/mybucket/path/to/object.zip</div>		

Hint 5

Lambda Configuration - Add a new
SNS Trigger from list on the left

Configure triggers

SNS topic
Select the SNS topic to subscribe to.

X

Lambda will add the necessary permissions for Amazon SNS to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

☒ **Enable trigger**
Enable the trigger now, or create it in a disabled state for testing (recommended).

Hint 6

Lambda Configuration - Add two new environment variables “imgSrcBucket” and “thumbDestBucket”

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more.](#)

imgSrcBucket

serverless-training-img-lno

Remove

thumbDestBucket

serverless-training-thumb-lno

Remove

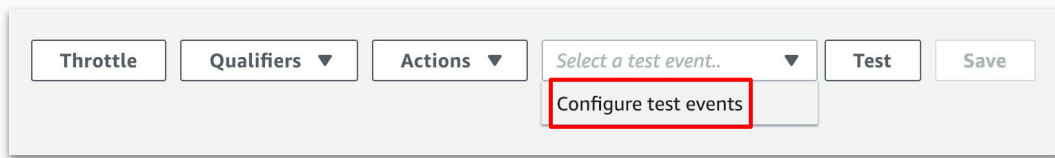
Key

Value

Remove

Test the lambda function

1. Copy the test json sample **test-sample.json** from **Function code**
2. Configure a new **Test event** from the top menu



3. Paste the json sample and replace `<IMAGE_NAME>` by an image name present in the S3 bucket
4. Create the test event and test !

Done !

Test the full integration by uploading an image in the S3 image bucket and checking the S3 thumbnail bucket

You can download the answer code at



<https://github.com/laurentnoireterre/py-aws-lambda-thumbnail-gen>