

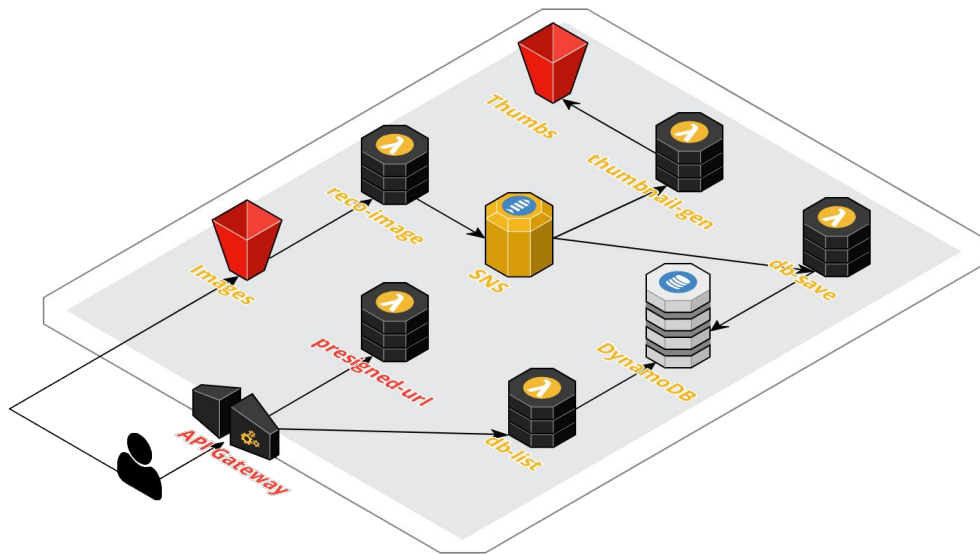
# AWS training - Serverless

Hands On #2 - Lambda | API Gateway - Upload Image

# Overview

This Hands-on is composed of 2 parts:

1. [Lambda Part](#) : new function to generate a S3 pre-signed url
2. [API Gateway Part](#) : new REST API Gateway and new GET method to route the ingress traffic to the Lambda function



# Let's go! | Lambda Part

Go to Virginia region

N. Virginia ▼

Create a lambda function having these properties:

- **Name:** py-aws-lambda-presigned-url
- **Runtime:** Python 3.6
- **Role:** serverless\_lambda\_role
- Add a new **Environment variable** containing the bucket name created previously
  - Name: bucketName
  - Value: serverless-training-img-<xxx>
- Upload the **Function code** from the S3 bucket:  
<https://s3.amazonaws.com/awstacktraining-serverless-resources/code-templates/py-aws-lambda-presigned-url-template.zip>

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

## Context:

In this part we create a lambda in charge of generating a S3 pre-signed url. This url will be used later by the front-end web-ui to upload a new image in the bucket

## Documentation:

[https://docs.aws.amazon.com/fr\\_fr/IAM/latest/UserGuide/introduction.html](https://docs.aws.amazon.com/fr_fr/IAM/latest/UserGuide/introduction.html)

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


[https://docs.aws.amazon.com/fr\\_fr/apigateway/latest/developerguide/welcome.html](https://docs.aws.amazon.com/fr_fr/apigateway/latest/developerguide/welcome.html)

# Hint 1

**Lambda Creation** - create a new  
Lambda function  
“py-aws-lambda-presigned-url”  
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.

**Role**

Defines 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 2

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

**Function code** [Info](#)

Code entry type

Upload a file from Amazon S3 ▼

Runtime

Python 3.6 ▼

Handler

[Info](#)

lambda\_function.lambda\_handler

Amazon S3 link URL

Paste an S3 link URL to your function code .zip.

<https://s3.amazonaws.com/mybucket/path/to/object.zip>

# Hint 3

**Lambda Configuration** - Add a new environment variable  
"bucketName"

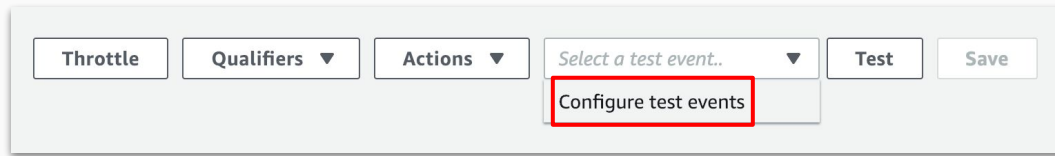
### Environment variables

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.](#)

bucketName	serverless-training-img-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, create the test event and test !

# Let's go! | API Gateway Part

---

Create a new **REST** API Gateway having these properties:

- **Name:** reco-image

Create a new resource and a GET method on this API :

- **Name:** upload
- **Resource:** /upload
- Create a new **GET** method
- **CORS** has to be enabled
- Use **Lambda proxy integration**
- Integrate with the **py-aws-lambda-presigned-url** lambda function

Before deploying -> Go to [Testing part](#) to test the API endpoint

- Deploy the API on a new stage **dev** (test the API first)

After deploying -> Go to [Done ! part](#) to test the full integration

## Context:

In this part we create the API Gateway resource in front of the lambda. The proxy mode will route the ingress traffic directly to the function without any modification.

## Documentation:

[https://docs.aws.amazon.com/fr\\_fr/apigateway/latest/developerguide/welcome.html](https://docs.aws.amazon.com/fr_fr/apigateway/latest/developerguide/welcome.html)



# Hint 4

**API Gateway** - create a new REST  
API Gateway "reco-image"

## Choose the protocol

Select whether you would like to create a REST API or a WebSocket API.

☒ **REST** ☐ **WebSocket**

## Create new API

In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

☒ **New API** ☐ **Clone from existing API** ☐ **Import from Swagger or Open API 3** ☐ **Example API**

## Settings

Choose a friendly name and description for your API.

**API name\***

reco-image

**Description**

**Endpoint Type**

Regional



# Hint 5

**API Gateway** - create a new resource “upload”, with CORS enabled

The diagram illustrates the process of creating a new resource in API Gateway. It starts with a screenshot of the 'Actions' menu, where the 'Create Resource' option is highlighted with a red box. A blue arrow points from this option to a 'Configure as proxy resource' dialog box. In this dialog, the 'Resource Name' is set to 'upload' and the 'Resource Path' is set to '/ upload'. The 'Enable API Gateway CORS' checkbox is checked. A help text box explains that path parameters can be added using brackets, such as {username} for a path parameter called 'username'.

**Actions** ▾ / Methods

**RESOURCE ACTIONS**

- Create Method
- Create Resource**
- Enable CORS
- Edit Resource Documentation

**API ACTIONS**

- Deploy API
- Import API
- Edit API Documentation
- Delete API

**Configure as [proxy resource](#)** ⓘ

**Resource Name\*** upload

**Resource Path\*** / upload

You can add path parameters using brackets. For example, the resource path {username} represents a path parameter called 'username'. Configuring /{proxy+} as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to /foo. To handle requests to /, add a new ANY method on the / resource.

**Enable API Gateway CORS** ☒ ⓘ

# Hint 6

**API Gateway** - create a new **GET** method pointing to the lambda function created previously. Use the proxy mode integration

## /upload - GET - Setup

Choose the integration point for your new method.

**Integration type** ☒ Lambda Function ⓘ

☐ HTTP ⓘ

☐ Mock ⓘ

☐ AWS Service ⓘ

☐ VPC Link ⓘ

**Use Lambda Proxy integration** ☒ ⓘ

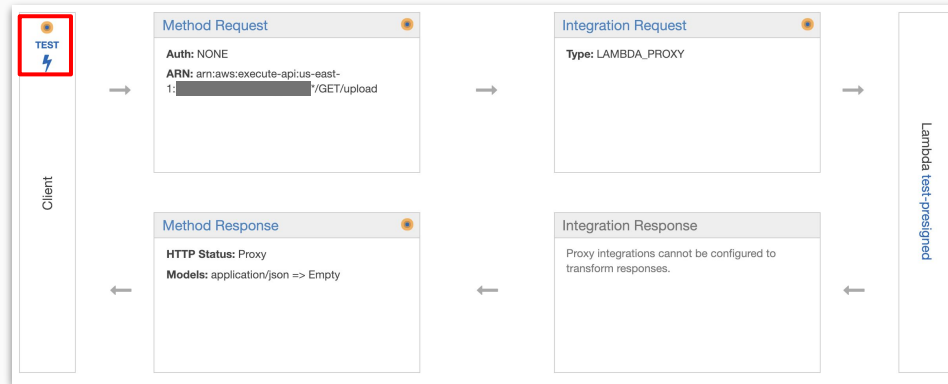
**Lambda Region**

**Lambda Function**

**Use Default Timeout** ☒ ⓘ

# Test the API Gateway resource

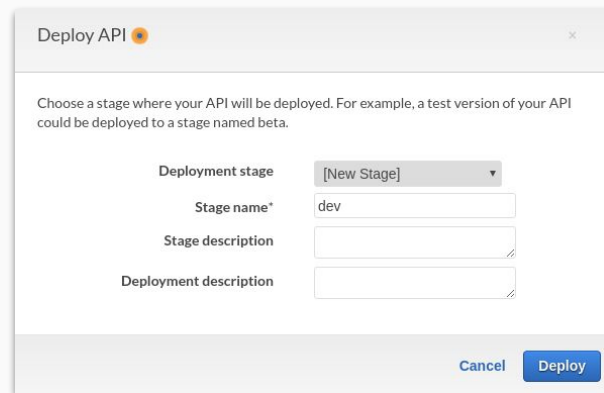
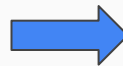
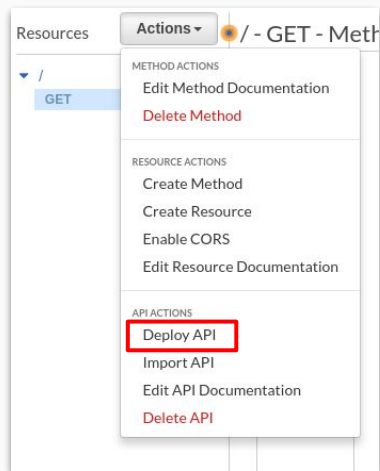
1. Click on test icon



2. Set the **Query String** with **"filename=image-test.jpg"** and test !

# Hint 7

**API Gateway** - Deploy the API to the stage "dev" (create it if needed)



# Done !

Test the API endpoint using the invoke URL !

<YOUR\_API\_INVOKE\_URL>/upload?filename=image-test.jpg

You can download the Lambda code at



<https://github.com/laurentnoireterre/py-aws-lambda-presigned-url>