

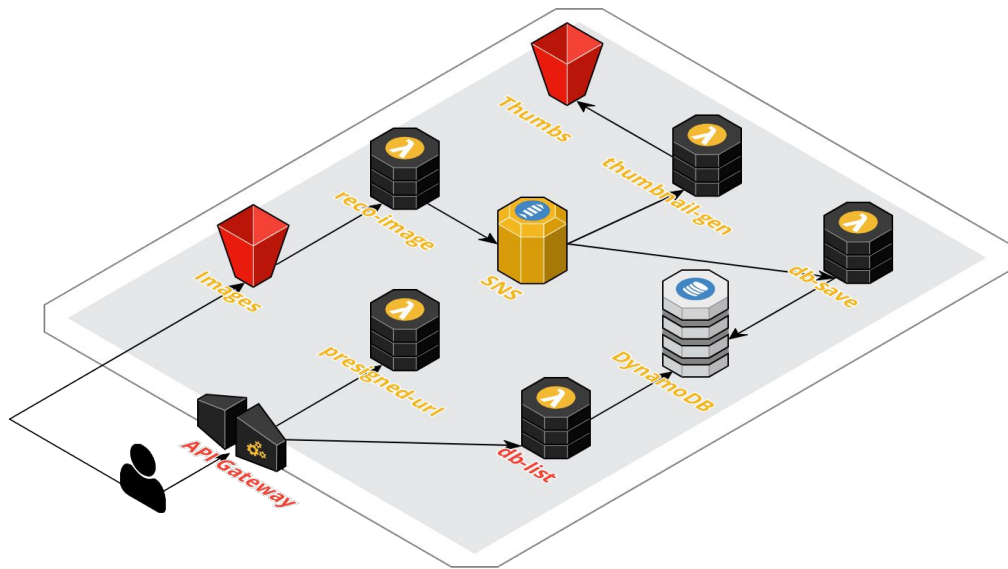
# AWS training - Serverless

Hands On #6 - API Gateway | Lambda - List items

# Overview

This Hands-on is composed of 2 parts:

1. [Lambda Part](#) : new function to list the recognition result items stored in DynamoDB
2. [API Gateway Part](#) : new GET method on existing API Gateway 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-db-list-items
- **Runtime:** Python 3.6
- **Role:** serverless\_lambda\_role
- Add a new **Environment variable** containing the table name created previously
  - Name: dynamodbTableName
  - Value: ImageRecoResults
- Upload the **Function code** from the S3 bucket:  
<https://s3.amazonaws.com/awstacktraining-serverless-resources/code-templates/py-aws-lambda-db-list-items-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 listing the recognition result items stored in DynamoDB.

## Documentation:


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

# Hint 1

**Lambda Creation** - create a new  
Lambda function  
“py-aws-lambda-db-list-items” 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  
“dynamodbTableName”

## 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.](#)

dynamodbTableName

ImageRecoResults

Remove

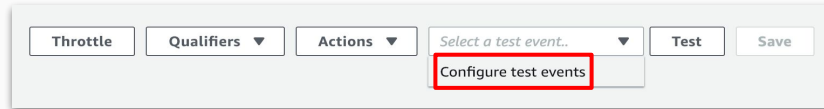
Key

Value

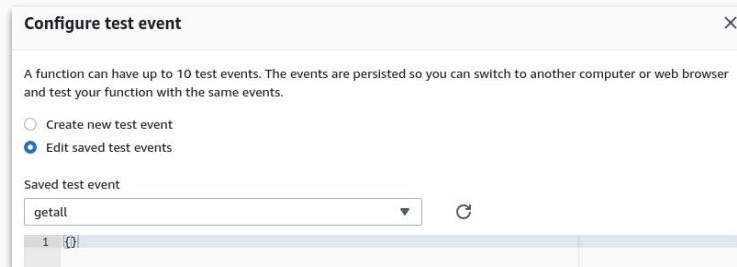
Remove

# Test the lambda function

1. Configure a new **Test event** from the top menu



2. Set an empty JSON content, create the test event and test !



# Let's go! | API Gateway Part

Create a new API resource on “image-reco” API Gateway, having these properties:

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

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

- Deploy the API on the **dev** stage

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:

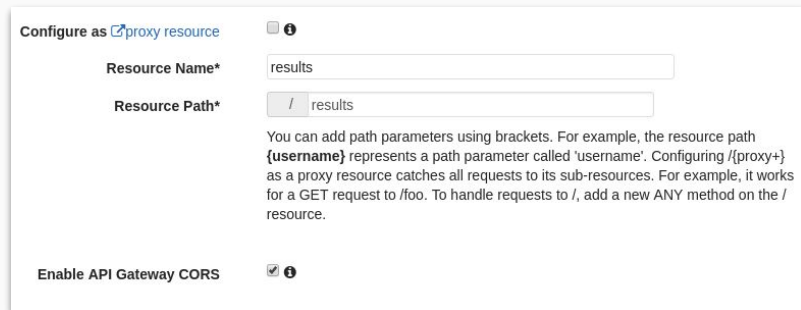
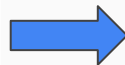
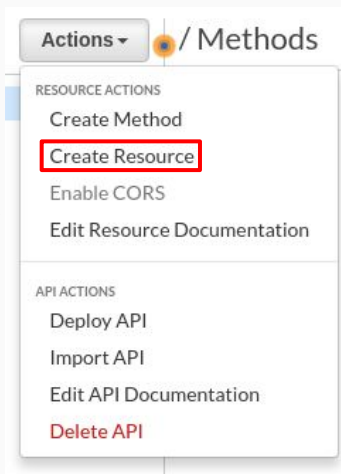
[https://docs.aws.amazon.com/fr\\_fr/lambda/latest/dg/with-s3-example-deployment-pkg.html#with-s3-example-deployment-pkg-python](https://docs.aws.amazon.com/fr_fr/lambda/latest/dg/with-s3-example-deployment-pkg.html#with-s3-example-deployment-pkg-python)

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



# Hint 4

**API Gateway** - Open the API Gateway “reco-image” and create a new resource “results”, with CORS enabled



# Hint 5

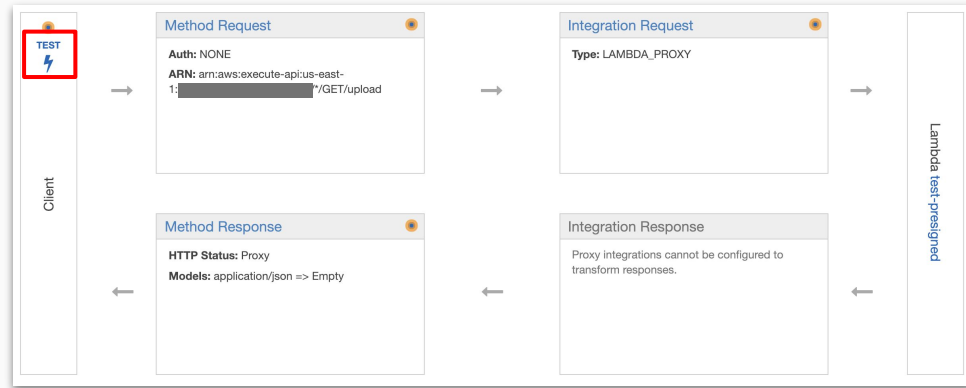
**API Gateway** - create a new GET method pointing to the lambda function created previously. Use the proxy mode integration

The screenshot shows the AWS API Gateway console interface. On the left, the 'Resources' pane shows a tree structure with a root '/' and a child resource '/results'. The 'GET' method is selected for the '/results' resource. The main pane is titled '/results - GET - Setup' and contains the following configuration options:

- Integration type:** A radio button selection where 'Lambda Function' is selected. Other options are HTTP, Mock, AWS Service, and VPC Link.
- Use Lambda Proxy integration:** A checkbox that is checked.
- Lambda Region:** A dropdown menu showing 'us-east-1'.
- Lambda Function:** A text input field containing 'py-aws-lambda-db-list-items'.
- Use Default Timeout:** A checkbox that is checked.

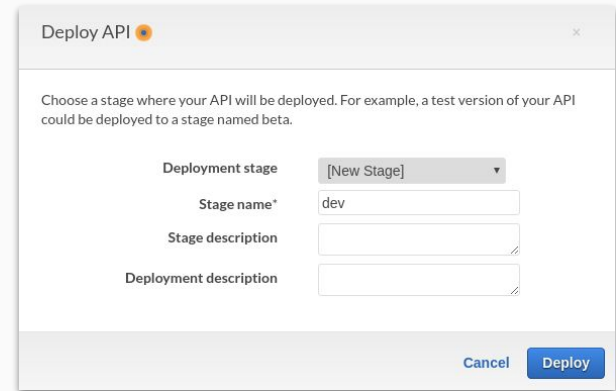
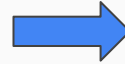
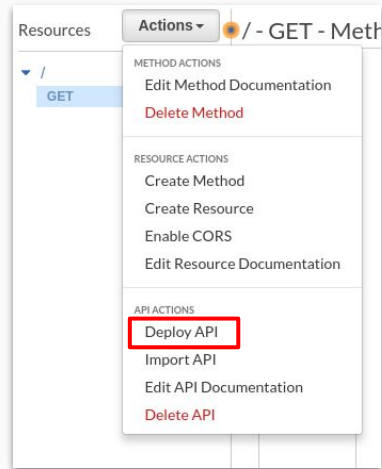
# Test the API Gateway resource

**API Gateway** - Test the GET method execution



# Hint 6

**API Gateway** - Deploy the API to the stage "dev"



# Done !

Test the API endpoint using the invoke URL !

<YOUR\_API\_INVOKE\_URL>/results

You can download the Lambda code at



<https://github.com/laurentnoireterre/py-aws-lambda-db-list-items>