

AWS Lambda

Tutorial

© 2010 VMware Inc. All rights reserved

Face Detection

Input



Output



Sources

cv2.so	OpenCV statically bind library
lambda_function.py	Libraries used by numpy
lib	numpy: Python library for library bindings
numpy	
share	OpenCV configuration files

Face Detection

```
import ...

print('Loading function')

dstBucket = 'ece1779winter2017demo'
dstPrefix = 'tmp/'
outputDomain = 's3.amazonaws.com/ece1779winter2017demo'

cascPath = 'share/OpenCV/haarcascades/haarcascade_frontalface_olt.xml'

s3 = boto3.resource('s3')
faceCascade = cv2.CascadeClassifier(cascPath)

def lambda_handler(event, context):
    print('Received event: ' + json.dumps(event, indent=2))
    imageUrl = event['imageUrl']
    imageFile = urllib2.urlopen(imageUrl)
    imageBytes = numpy.asarray(bytearray(imageFile.read()), dtype=numpy.uint8)
    image = cv2.imdecode(imageBytes, cv2.CV_LOAD_IMAGE_UNCHANGED)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(30, 30),
        flags = cv2.CV_HAAR_SCALE_IMAGE
    )

    for (x, y, w, h) in faces:
        cv2.rectangle(image, (x, y), (x+w, y+h), (255, 255, 255), 2)

    r, outputImage = cv2.imencode('.jpg', image)
    if False==r:
        raise Exception('Error encoding image')

    dstKey = dstPrefix + datetime.datetime.now().strftime('%Y%m%d%H%M%S') + '-' + str(uuid.uuid4()) + '.jpg'

    s3.Bucket(dstBucket).put_object(Key=dstKey,
        Body=outputImage.tostring(),
        ContentType='image/jpeg',
        ACL='public-read'
    )

    outputUrl = 'https://' + outputDomain + '/' + dstKey
    return "<html><body><img src='{}' /></body></html>".format(outputUrl)
```

Replace with YOUR S3 bucket name

Replace with URL to your S3 bucket

Create Deployment Package

- Inside source directory type:

```
> zip -r face_detection_code.zip *
```

Create Lambda Function

The screenshot shows the AWS Lambda console interface for creating a new function. The browser address bar shows the URL: `https://console.aws.amazon.com/lambda/home?region=us-east-1#/create`. The page title is "Lambda Management Console". The navigation bar includes "Services", "Resource Groups", and a user profile "Eyal de Lara" in "N. Virginia". The breadcrumb trail is "Lambda > Functions > Create function > Author from scratch".

The "Basic information" section is active, showing the following fields and options:

- Name***: `face_recognition`
- Role***: `Create new role from template(s)` (dropdown menu). Below this, a note states: "Lambda will automatically create a role with permissions from the selected policy templates. Note that basic Lambda permissions (logging to CloudWatch) will automatically be added. If your function accesses a VPC, the required permissions will also be added."
- Role name***: `face_recognition_role`
- Policy templates**: A dropdown menu showing `SS object read-only permissions` with a close button (X).

At the bottom of the form, there is a note: "* These fields are required." and three buttons: "Cancel", "Previous", and "Create function".

The footer of the console shows "Feedback", "English (US)", and copyright information: "© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved." along with links for "Privacy Policy" and "Terms of Use".

Create Lambda Function

The screenshot shows the AWS Lambda console for a function named 'face_recognition'. The 'Configuration' tab is active. A message states: 'The deployment package of your Lambda function "face_recognition" is too large to enable inline code editing. However, you can still invoke your function right now.' The 'Code entry type' is set to 'Upload a .ZIP file'. The 'Runtime' is 'Python 2.7' and the 'Handler' is 'lambda_function.lambda_handler'. The 'Function package*' section has an 'Upload' button and a note: 'For files larger than 10 MB, consider uploading via S3.' Below this are sections for 'Environment variables', 'Tags', 'Execution role', and 'Basic settings'.

Create Lambda Function

This screenshot shows the same AWS Lambda console configuration for the 'face_recognition' function, but with more details visible. The 'Function package*' section shows an uploaded file 'face_detection_code.zip (17.3 MB)' with a note: 'For files larger than 10 MB, consider uploading via S3.' The 'Execution role' section shows 'Choose an existing role' and 'Existing role' set to 'service-role/face_recognition_role'. The 'Basic settings' section shows 'Memory (MB)' set to 1536 MB and 'Timeout' set to 3 minutes. A red text overlay on the right side of the console reads: 'Increase memory allocation and timeout'.

Configure Security Role

The screenshot shows the AWS IAM Management Console interface. The left sidebar contains navigation links: Dashboard, Groups, Users, Roles (selected), Policies, Identity providers, Account settings, Credential report, and Encryption keys. The main content area is titled 'Role Actions' and displays a table of roles. The table has columns for 'Role Name' and 'Creation Time'. Two roles are listed: 'face_recognition' and 'lambda_basic_execution'. A 'Filter' input field is at the top of the table, and it indicates 'Showing 2 results'.

Role Name	Creation Time
face_recognition	2017-03-24 08:54 EDT
lambda_basic_execution	2017-03-23 20:26 EDT

Configure Security Role

The screenshot shows the configuration details for the 'face_recognition' role in the AWS IAM Management Console. The left sidebar is the same as the previous screenshot. The main content area shows the role's details: Role ARN (arn:aws:iam::470901565805:role/service-role/face_recognition), Instance Profile ARN(s), Path (/service-role/), and Creation Time (2017-03-24 08:54 EDT). Below this, there are tabs for 'Permissions', 'Trust Relationships', 'Access Advisor', and 'Revoke Sessions'. The 'Permissions' tab is active, showing 'Managed Policies' and 'Inline Policies'. Under 'Managed Policies', it states 'The following managed policies are attached to this role. You can attach up to 10 managed policies.' and provides an 'Attach Policy' button. A table lists the attached policies:

Policy Name	Actions
AWSLambdaS3ExecutionRole-aef084d0-e70f-48a7-9267-3dd30f46926b	Show Policy Detach Policy Simulate Policy
AWSLambdaBasicExecutionRole-8af675aee-5e47-4539-a77b-0066aafa1631	Show Policy Detach Policy Simulate Policy

Below the table is a section for 'Inline Policies' with a dropdown arrow.

Configure Security Role

Search IAM

Policy ARN: `arn:aws:iam:470901565805:policy/service-role/AWSLambdaS3ExecutionRole-ae084d0-e70f-48a7-9267-3dd30f46926b`

Description

Permissions Attached entities (1) Policy versions Access Advisor

Policy summary JSON

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "s3:GetObject"
8       ],
9       "Resource": "arn:aws:s3:*:*"
10    }
11  ]
12 }
```

Replace with: `s3:*`

☒ Use autoforamtting for policy editing ☒ Save as default version

Feedback English © 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Configure Test Values

Lambda Management Console

Configure test event

A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

☐ Create new test event ☒ Edit saved test events

Saved Test Event

event1509555289

```
1 {
2   "imageUrl": "https://pbs.twimg.com/media/CB6KmZ9UEAAyOwU.jpg:large"
3 }
```

Test Function

The screenshot shows the AWS Lambda console interface. The left sidebar contains a navigation menu with 'Dashboard' and 'Functions'. The main content area displays the 'face_recognition' function details. At the top, there are tabs for 'Qualifiers' and 'Actions', and a 'Test' button. Below this, a green box indicates 'Execution result: succeeded (logs)'. A 'Details' section shows a log snippet: `{ "html">body</body></html>" }`. A 'Summary' section lists various metrics: Code SHA-256 (WnZc5TWCM5V1xk20heHk1jnf0S41G02QIQMB55F18=), Request ID (3d2b0cb-bf28-11e7-ad68-7f56dc873331), Duration (1421.62 ms), Billed duration (1500 ms), Resources configured (1536 MB), and Max memory used (65 MB). The bottom of the console shows a footer with 'Feedback', 'English (US)', and copyright information.

Create Web API

The screenshot shows the AWS API Gateway console interface. The left sidebar contains a navigation menu with 'APIs', 'Usage Plans', 'API Keys', 'Custom Domain Names', 'Client Certificates', and 'Settings'. The main content area displays the 'Create new API' form. At the top, there are radio buttons for 'New API' (selected), 'Clone from existing API', 'Import from Swagger', and 'Example API'. Below this, a 'Name and description' section has a text input for 'API name*' with the value 'face_recognition_api' and an empty text input for 'Description'. A 'Create API' button is located at the bottom right. The bottom of the console shows a footer with 'Feedback', 'English', and copyright information.

Create Method

The screenshot shows the AWS API Gateway console. The breadcrumb trail is: Amazon API Gateway > APIs > face_recognition_api (s73grnmt04) > Resources > / (rxvtediut6) > GET. The left sidebar shows the API structure: face_recognition_api > Resources > / > GET. The main content area is titled '/ - GET - Setup' and contains the following fields:

- Integration type:** ☒ Lambda Function ⓘ
☐ HTTP ⓘ
☐ Mock ⓘ
☐ AWS Service ⓘ
- Use Lambda Proxy integration:** ☐ ⓘ
- Lambda Region:** us-east-1
- Lambda Function:** face_recognition ⓘ

A **Save** button is located at the bottom right of the form.

Map HTTP Parameter for Function Argument

The screenshot shows the AWS API Gateway console. The breadcrumb trail is: Amazon API Gateway > APIs > face_recognition_api (s73grnmt04) > Resources > / (rxvtediut6) > GET. The left sidebar shows the API structure: face_recognition_api > Resources > / > GET. The main content area is titled '/ - GET - Integration Request' and contains the following fields:

- Integration type:** ☒ Lambda Function ⓘ
☐ HTTP ⓘ
☐ Mock ⓘ
☐ AWS Service ⓘ
- Use Lambda Proxy integration:** ☐ ⓘ
- Lambda Region:** us-east-1
- Lambda Function:** face_recognition
- Invoke with caller credentials:** ☐ ⓘ
- Credentials cache:** Do not add caller credentials to cache key

The **Body Mapping Templates** section is expanded, showing the following options:

- Request body passthrough:** ☐ When no template matches the request Content-Type header ⓘ
☐ When there are no templates defined (recommended) ⓘ

Map HTTP Parameter for Function Argument

Request body passthrough ☐ When no template matches the request Content-Type header ⓘ
☐ When there are no templates defined (recommended) ⓘ
☒ **Never** ⓘ

Content-Type	
application/json	⊖

[+ Add mapping template](#)

application/json

Generate template: [↕](#)

```
1 {"imageUrl": "$input.params('image')}"
```

Add Parameter to Method Request (for testing purposes)

The screenshot shows the AWS API Gateway console interface. The breadcrumb navigation at the top indicates the path: **Services** > **Resource Groups** > **APIs** > **face_recognition_api (s73grnmt04)** > **Resources** > **/ (rxvtediut6)** > **GET**. The left-hand navigation pane lists various API Gateway components, with **Resources** selected under the **face_recognition_api** resource. The main content area is titled **Method Execution / - GET - Method Request**. It includes a description: "Provide information about this method's authorization settings and the parameters it can receive." Under the **Authorization Settings** section, **Authorization** is set to **NONE** and **API Key Required** is set to **false**. The **URL Query String Parameters** section is expanded, displaying a table with the following data:

Name	Caching
image	<input type="checkbox"/>

Below the table, there is a link to **Add query string**. Other sections like **HTTP Request Headers** and **Request Body** are visible but collapsed. The footer of the console shows **Feedback**, **English**, and copyright information for Amazon Web Services, Inc.

Change Method Response Type to HTML

API Gateway

CloudWatch Management ...

https://console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/s73grnmt04/resources

Services Resource Groups

Amazon API Gateway APIs > face_recognition_api (s73grnmt04) > Resources > / (xxvtediuf6) > GET

APIs

- face_recognition_api
 - Resources
 - Stages
 - Authorizers
 - Models
 - Documentation
 - Binary Support
- facereco
- Usage Plans
- API Keys
- Custom Domain Names
- Client Certificates
- Settings

Method Execution / - GET - Method Response

Provide information about this method's response types, their headers and content types.

HTTP Status

200

Response Headers for 200

Name
No headers

Add Header

Response Body for 200

Content type	Models
text/html	Empty

Add Response Model

Add Response

Feedback English

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Deploy API

API Gateway

CloudWatch Management ...

https://console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/s73grnmt04/resources

Services Resource Groups

Amazon API Gateway APIs > face_recognition_api (s73grnmt04) > Resources > / (xxvtediuf6) > GET

APIs

- face_recognition_api
 - Resources
 - Stages
 - Authorizers
 - Models
 - Documentation
 - Binary Support
- facereco
- Usage Plans
- API Keys
- Custom Domain Names
- Client Certificates
- Settings

Deploy API

Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

Deployment stage: [New Stage]

Stage name*: prod

Stage description: Production

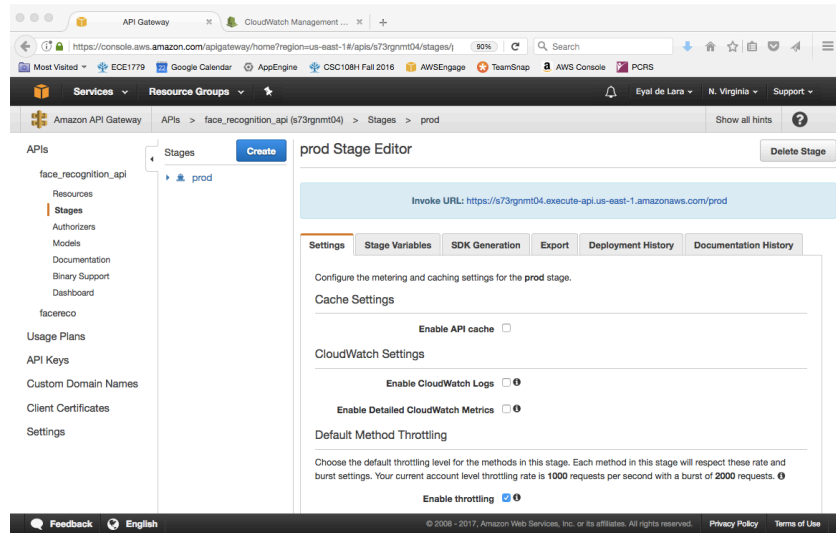
Deployment description:

Cancel Deploy

Feedback English

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Deploy API



Test with Browser

