Scraper Service

Shake Shack provides real-time images of the store-front on the Shake Shack website. In order to count the number of people in line, we use these images as our starting point.

This scraper service downloads the images and saves them to S3. From S3, the images are used by downstream services. The service can be run on a schedule or *ad hoc*.

# Prerequisites

This service has three dependencies: 1. AWS Lambda 2. Python Libraries 3. Amazon S3

## 1. AWS Lambda

This is an Amazon product that allows you to create serverless applications. One can create an account and get started with Lambda here: <https://aws.amazon.com/lambda/>

## 2. Python Libraries

This service depends on two python libraries:

### a. requests

This library allows you to make HTTP requests with Python. This is library is readily available with AWS Lambda. Simply including the following code in your Lambda function: from botocore.vendored import requests

### b. fake\_useragent

We use this library in conjunction with the requests library because, when making repeated HTTP requests, we want to simulate using different browsers for each programmatic request.

Because this library is not readily available to AWS Lambda, we have to upload it to AWS. The process is as follows ([see here for detailed instructions](https://medium.com/@qtangs/creating-new-aws-lambda-layer-for-python-pandas-library-348b126e9f3e)):

Install package locally by creating a requirements.txt file containing the following:

fake-useragent==0.1.11

Create the following packaging script. When executed, this will install the packages in your requirements file:

export PKG\_DIR="python"  
  
rm -rf ${PKG\_DIR} && mkdir -p ${PKG\_DIR}  
  
docker run --rm -v $(pwd):/foo -w /foo lambci/lambda:build-python3.6 \  
 pip install -r requirements.txt --no-deps -t ${PKG\_DIR}

Execute the packaging script to create zip file:

chmod +x get\_layer\_packages.sh  
./get\_layer\_packages.sh  
zip -r my-Python36-Pandas23.zip .

Once you have created a zipfile with the fake\_useragent library, upload it to your Lambda function with the UI.

## 3. Amazon S3

We chose Amazon S3 to store scraped images. After setting up an S3 bucket, we needed to grant our scraper service "full access" to the S3 bucket. This enables the service to write to S3. Access to S3 can be granted through Amazon's IAM UI.

# Running the Service

The scraper service does not require any inputs, so it can be triggered at any time. We trigger the service in one of two ways:

## a. On a Schedule

AWS Cloudwatch provides a mechanism to schedule the execution of AWS Lambda functions. We scheduled the service to run every 30 minutes.

## b. *Ad hoc*

In a subsequent iteration, we will use another service (check\_line) to make an API call to trigger the scraper service.

# Deployment

The scraper service is manually deployed from Github (used for version control and managing codebase). We will set up continuous integration in a subsequent version.

# Built With

* [AWS Lambda](https://aws.amazon.com/lambda/) - Serverless framework used
* [S3](https://aws.amazon.com/s3/getting-started/) - File management