Jose Miguel Arrieta Follow
Jun 26 · 5 min read

Creating an API using scikit-learn, AWS Lambda, S3 and Amazon API Gateway

This tutorial will help you build a classifier as a service. The classifier will be trained using ***iris flower data set*** witch consists on 3 different types of irises' (*Setosa*, *Versicolour*, and *Virginica*). The rows being the samples and the columns being features: *sepal length*, *sepal width*, *petal length* and *petal width*. *Scikit-learn* library will be used for machine-learning algorithms.The classifier will be stored in a S3 bucket and a lambda function will used to make classifications, finally an Amazon API Gateway will be used to trigger the lambda function. The goal in this tutorial will be that given *sepal length, sepal width, petal length and petal* in a POST request, the API will return the corresponding classification. Some familiarity with machine learning concepts is assumed.

Prerequisites

- *An AWS account* https://aws.amazon.com/

Image 1. Create an AWS Account

- You will need to get *scikit-learn* library and its dependencies built and packaged for Lambda. To continue, please follow instructions here https://serverlesscode.com/post/scikitlearn-with-amazon-linux-container/. You will need to install *docker*, `git clone` a repository and run some scripts. At the end you'll

have a `venv.zip` file containing *scikit-learn* and all it's dependencies. We will come back to this later.

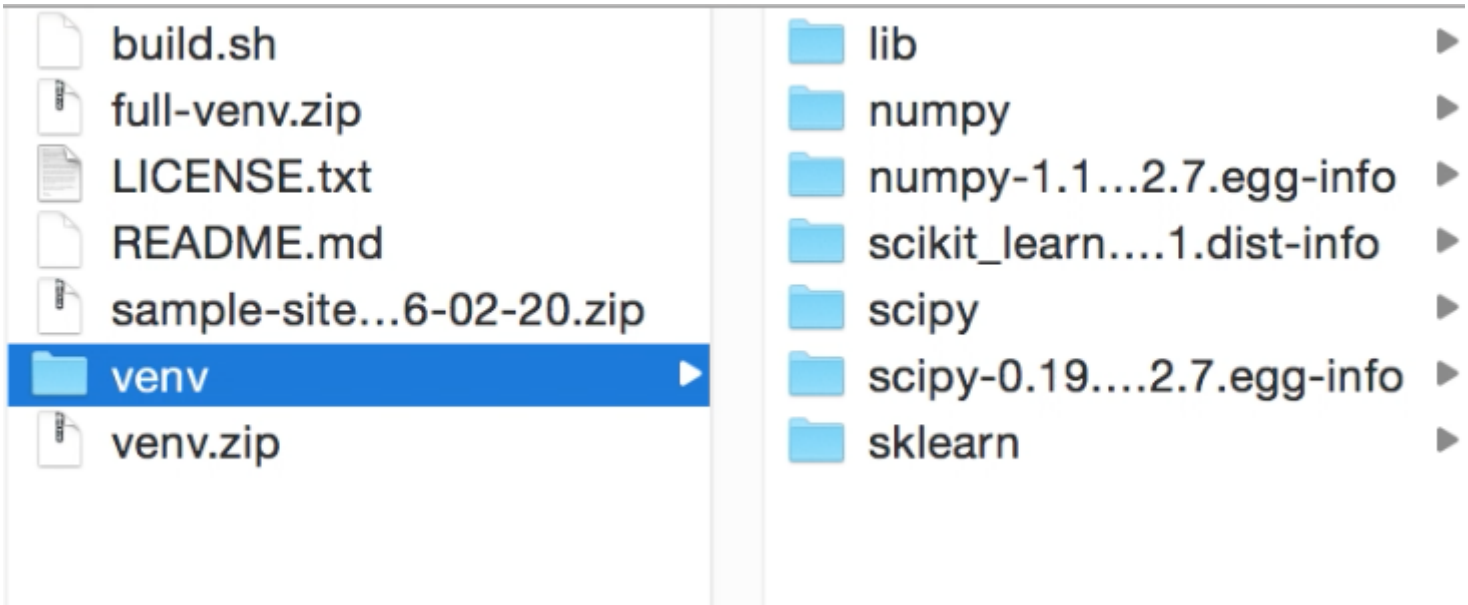

Image 2. Scikit-learn and all it's dependencies.

Build a Classifier

For this tutorial *Scikit-learn*, a machine library for the python programming language will be used. A *Support Vector machine* algorithm is selected to build a classifier using the *iris dataset*. The classifier is stored locally using *pickle* module and later uploaded to an Amazon S3 bucket.

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-

from sklearn import svm, datasets
import pickle
import numpy as np

#Load data
iris = datasets.load_iris()
X = iris.data
y = iris.target

#Train model with all data
```

Amazon S3

> *Amazon Simple Storage Service (Amazon S3) is object storage with a simple web service interface to store and retrieve any amount of data from anywhere on the web, designed to deliver durability and scale. https://aws.amazon.com/s3/*

Next, we will upload the classifier saved locally to an S3 bucket.

1. **Create a bucket.**

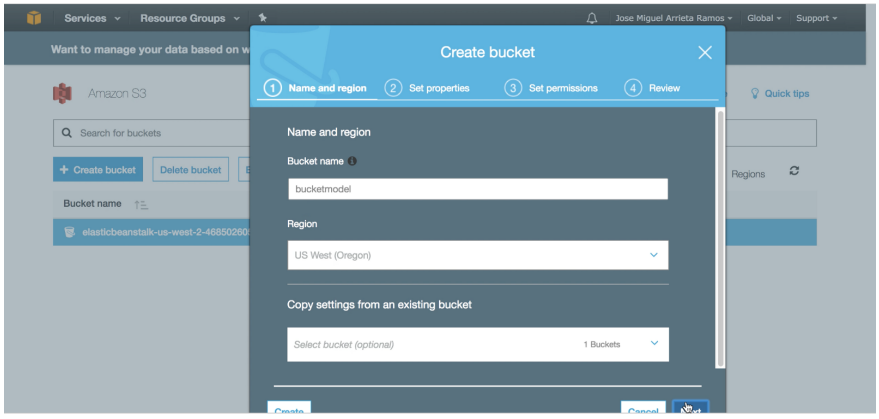

Image 3. Create S3 bucket

2. **Upload Model**

Here you will need to upload the classifier trained in the previous steps. You can upload the classifier using Amazon GUI or CLI.

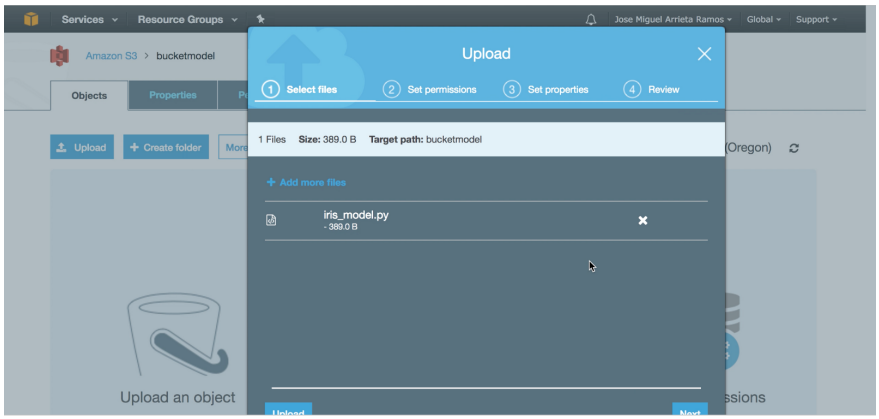

Image 4. Upload classifier to S3 bucket.

3. **Manage privileges.**

For this example the file uploaded will be public.

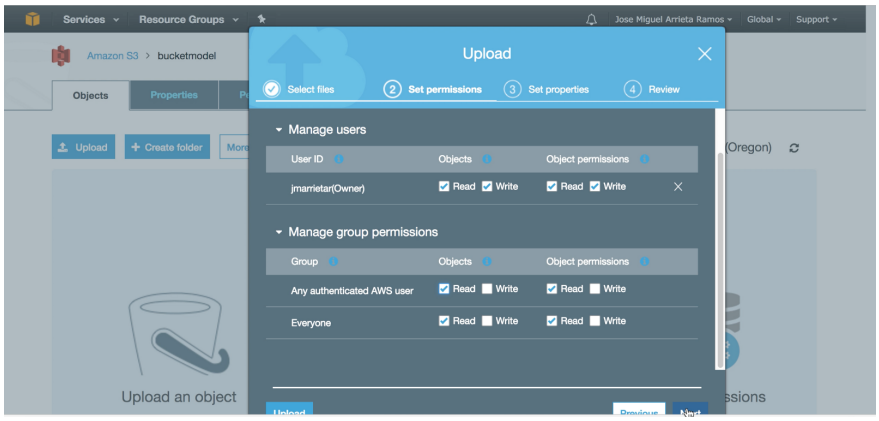

Image 5. Set permissions

Iris classifier successfully uploaded.

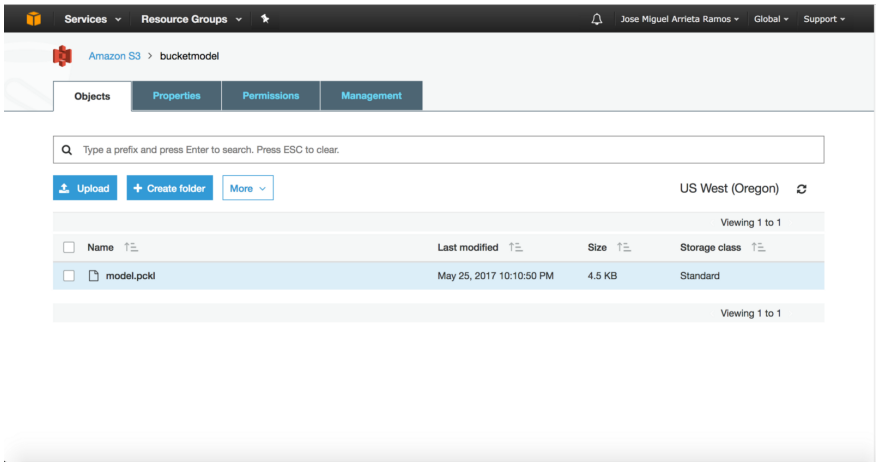

Image 6. Model.pckl

AWS Lambda

> *AWS Lambda is an event-driven, serverless computing platform provided by Amazon. It is a compute service that runs code in response to events and automatically manages the compute resources required by that code. https://aws.amazon.com/lambda/*

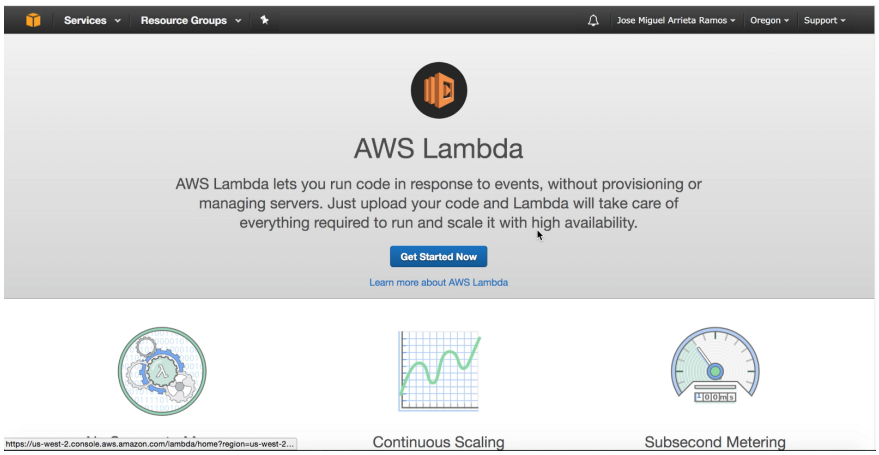

Image 7. AWS Lambda

1. **Select blueprint**

For this tutorial select Black function.

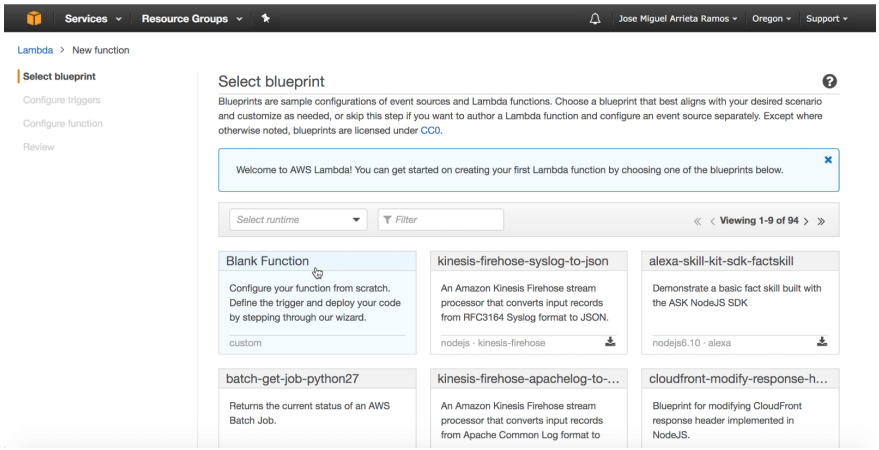

Image 8. Select blueprint.

The triggers configurations will be done later.

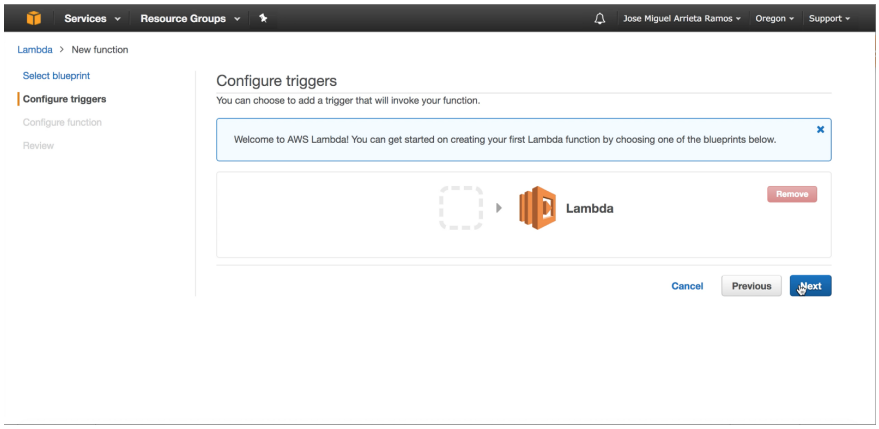

Image 9. Configure triggers.

2. **Configure function.**

The language used for this tutorial will be Python 2.7.

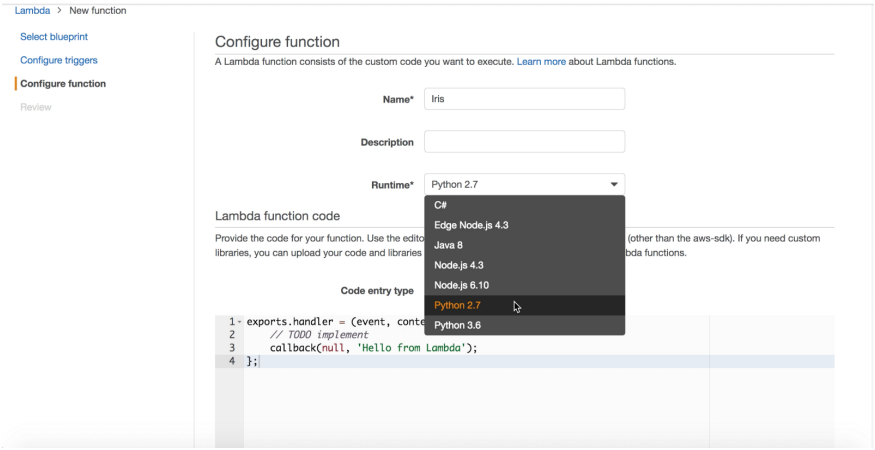

Image 10. Configure function.

> **Note**: Lambda supports recently Python 3.6 recently https://aws.amazon.com/about-aws/whats-new/2017/04/aws-lambda-supports-python-3-6

> *You will be asked to create a role or choose an exiting one http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html, keep all other default configurations.*

The lambda function was successfully created and we will now need to upload a compressed file with the lambda function (`function.py`) and the libraries needed.

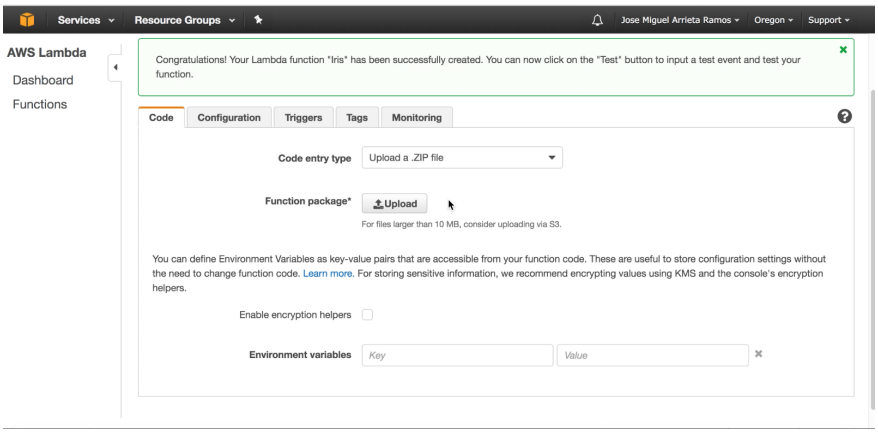

Image 11. Lambda function 'Iris' created.

In order to use *Numpy* and *Scipy* witch requires external libraries *ctypes* module is used, additionally *boto3* module is used to write software that makes use of Amazon services like S3 and *pickle* module

to load stored classifier. The lambda function code would be as follows:

```
import boto3
import os
import ctypes
import uuid
import sklearn
import pickle
import numpy as np

for d, _, files in os.walk('lib'):
    for f in files:
        if f.endswith('.a'):
            continue
        ctypes.cdll.LoadLibrary(os.path.join(d, f))

s3_client = boto3.client('s3')

def handler(event, context):

    #Info
    sepal_length = float(event.get('Iris')['sepal_leng
    sepal_width = float(event.get('Iris')['sepal_width
    petal_length = float(event.get('Iris')['petal_leng
    petal_width = float(event.get('Iris')['petal_width

    #Row
    row_arti = np.array([sepal_length,sepal_width,peta

```

*3. **Compress files and upload.***

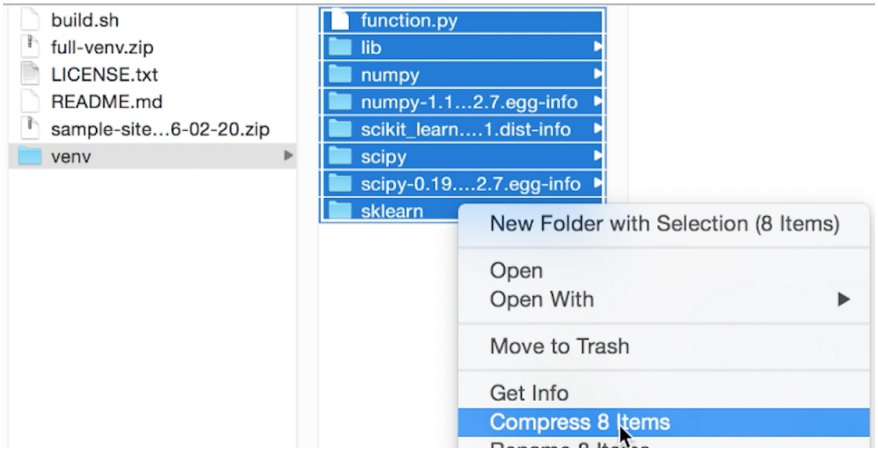

Image 12. Compress files and upload zip.

API Amazon Gateway

> *Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale.https://aws.amazon.com/api-gateway/*

1. In the options of the lambda function previously created, on the '*Triggers*' part. Click on '*Add trigger*'.

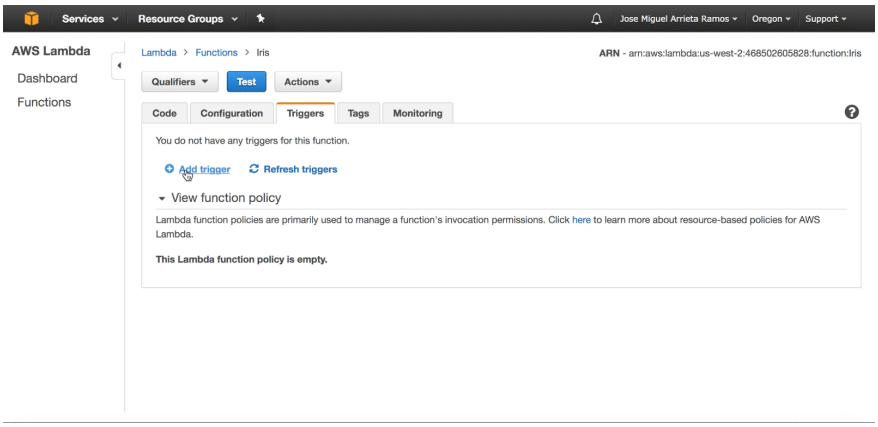

Image 13. Add trigger to lambda function.

2. *Choose **Api Gateway** Trigger.*

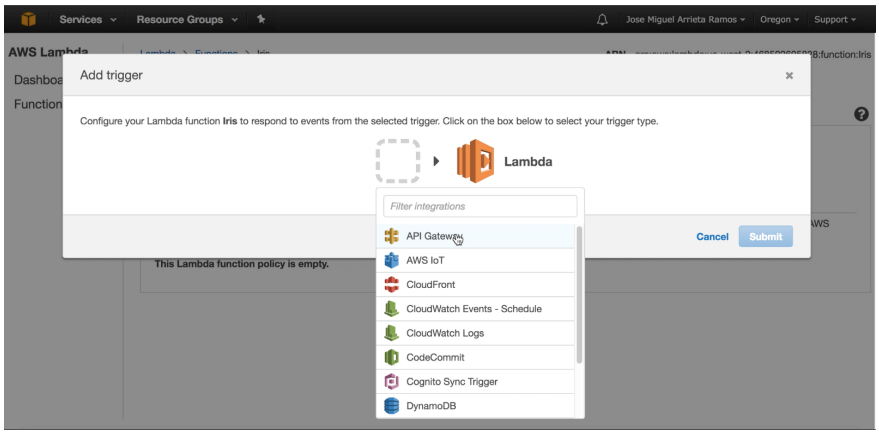

Image 14. Select API Gateway

3. Configure API Gateway.

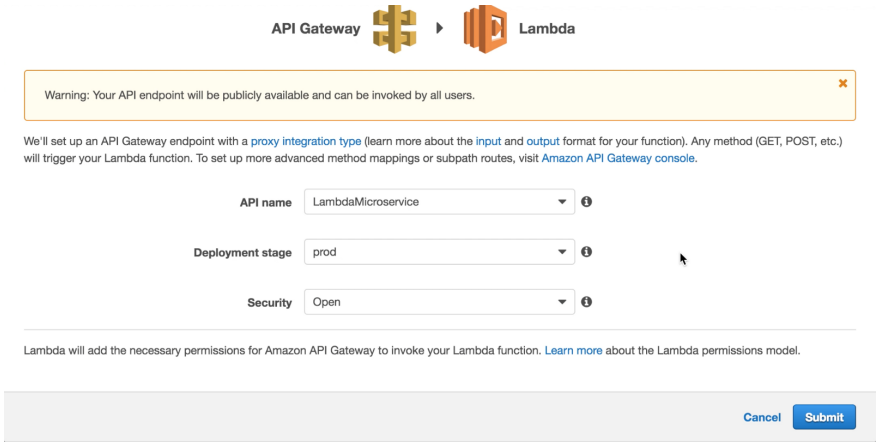

Image 15. Set up API Gateway

> *In this tutorial endpoint will be open and publicly available*

Successfully added trigger to function, and now function is receiving events from trigger.

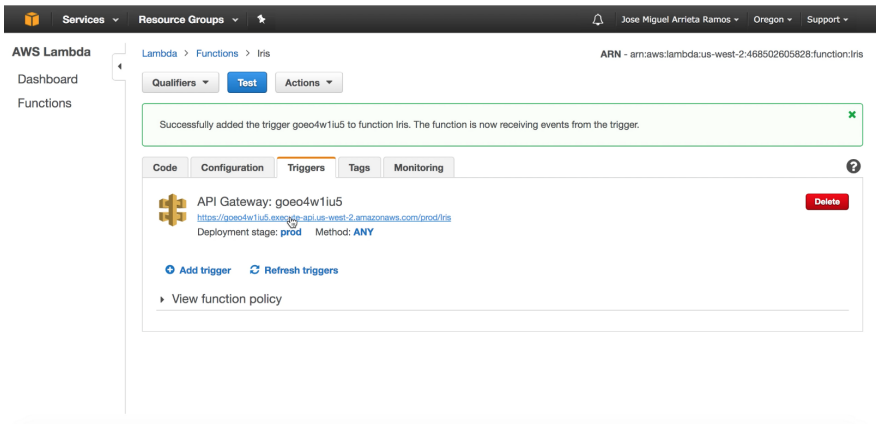

Image 16. Trigger added to Lambda function 'Iris'

4. Go to Amazon API Gateway Panel

Set up a POST method.

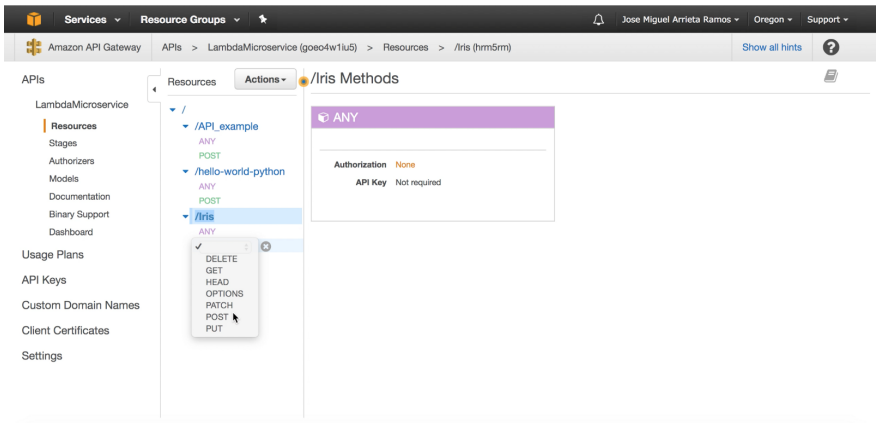

Image 17. Add POST Method.

Configure Lambda Integration

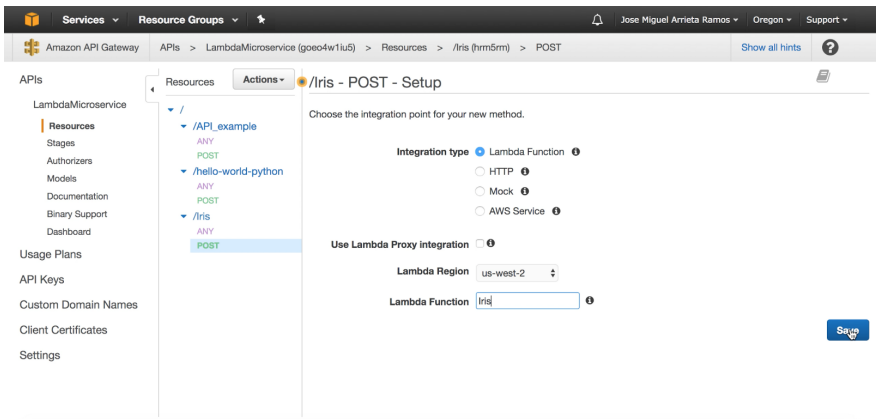

Image 18. Setup Method

Deploy API

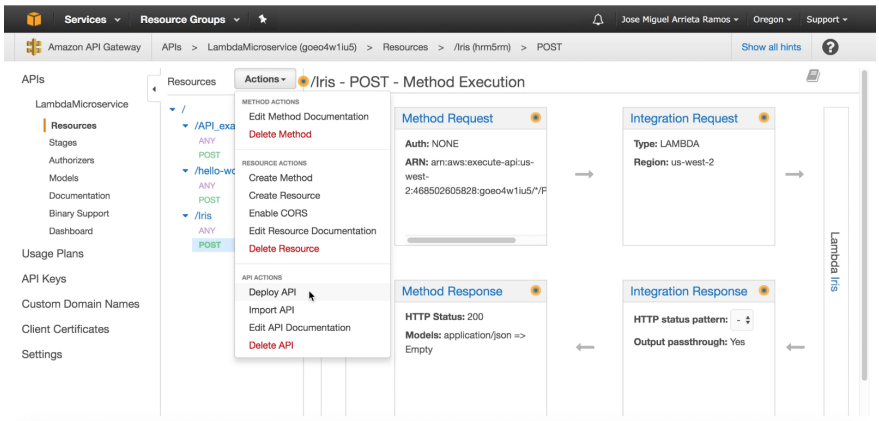

Image 19. Deploy API

API Endpoint

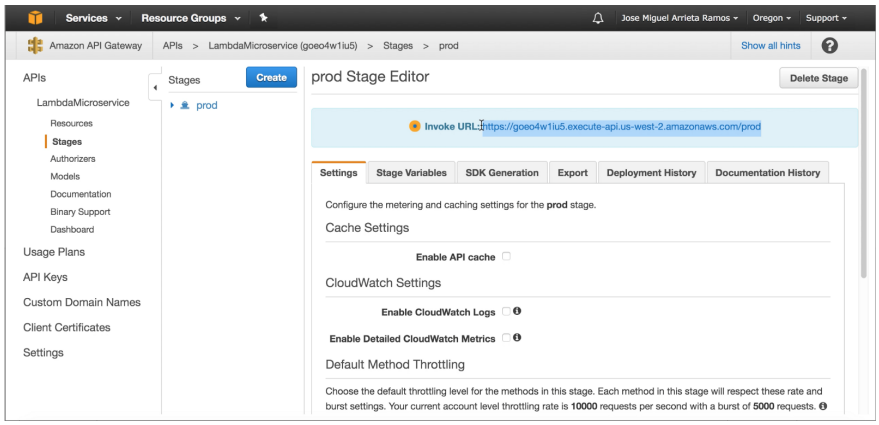

Image 20. End point

API is deployed and is ready to be consumed for applications.

Note: *Add lambda function name to the end of endpoint.*

Test API

Create two instances features in JSON format.

1. Instance #1

```
{
   "Iris":{
      "sepal_length":"5.1",
      "sepal_width":"3.5",
      "petal_length":"1.4",
      "petal_width":"0.2"
```

2. Instances #2

```
{
   "Iris":{
      "sepal_length":"5.1",
      "sepal_width":"1.5",
      "petal_length":"3.4",
      "petal_width":"0.5"
```

Send POST request via curl.

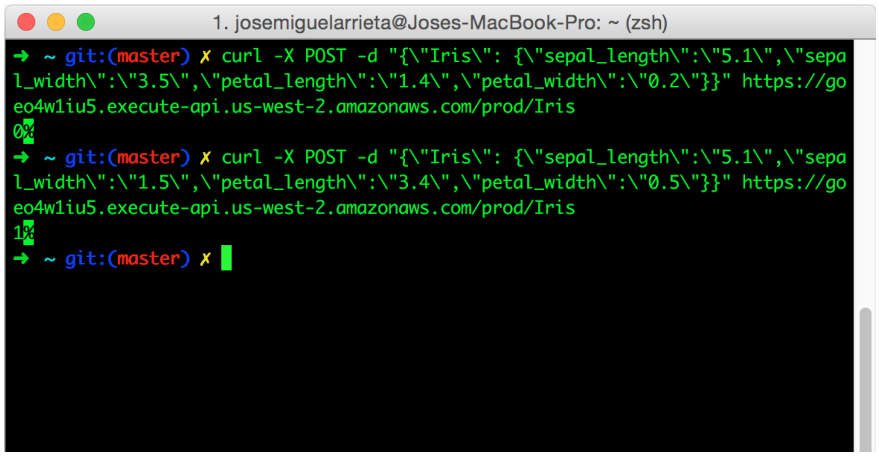

Image 21. API responses.

> Successfully, the two instances were classified. For the first request the features were classified as 0 (*Setosa*) and for the second one as 1 (*Versicolour*).

Thanks.

Jose Miguel
June 27, 2017