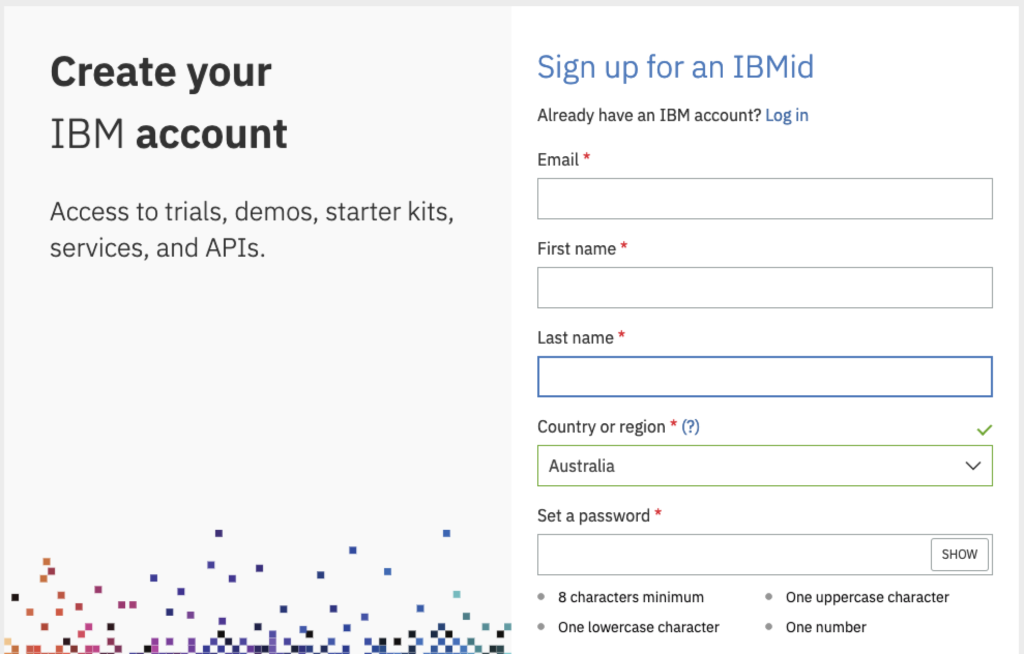
**How to Build Your First Image Recognition Classifier with IBM Watson Visual Recognition**

Before cracking on with actually building a classifier the first thing you’ll need is an IBMid. Why?

Well, you’re going to be using the Watson Visual Recognition API to train and test your classifier. These credentials allow you to access the free tier of IBM AI products through the IBM Cloud platform (side note: this was previously called IBM Bluemix).

If you don’t have an IBMid already head on over to the link below and sign up for one. Once you’ve got an account setup, take note of your credentials as you’ll need them in the next step!

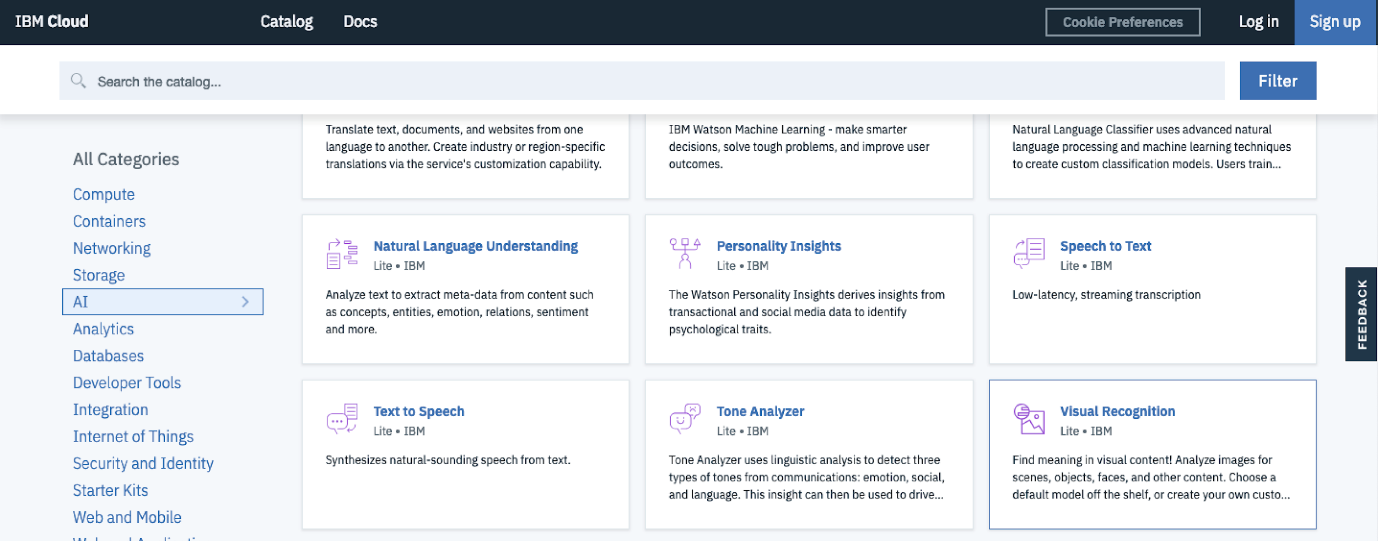


## STEP 2 – CREATE A SERVICE

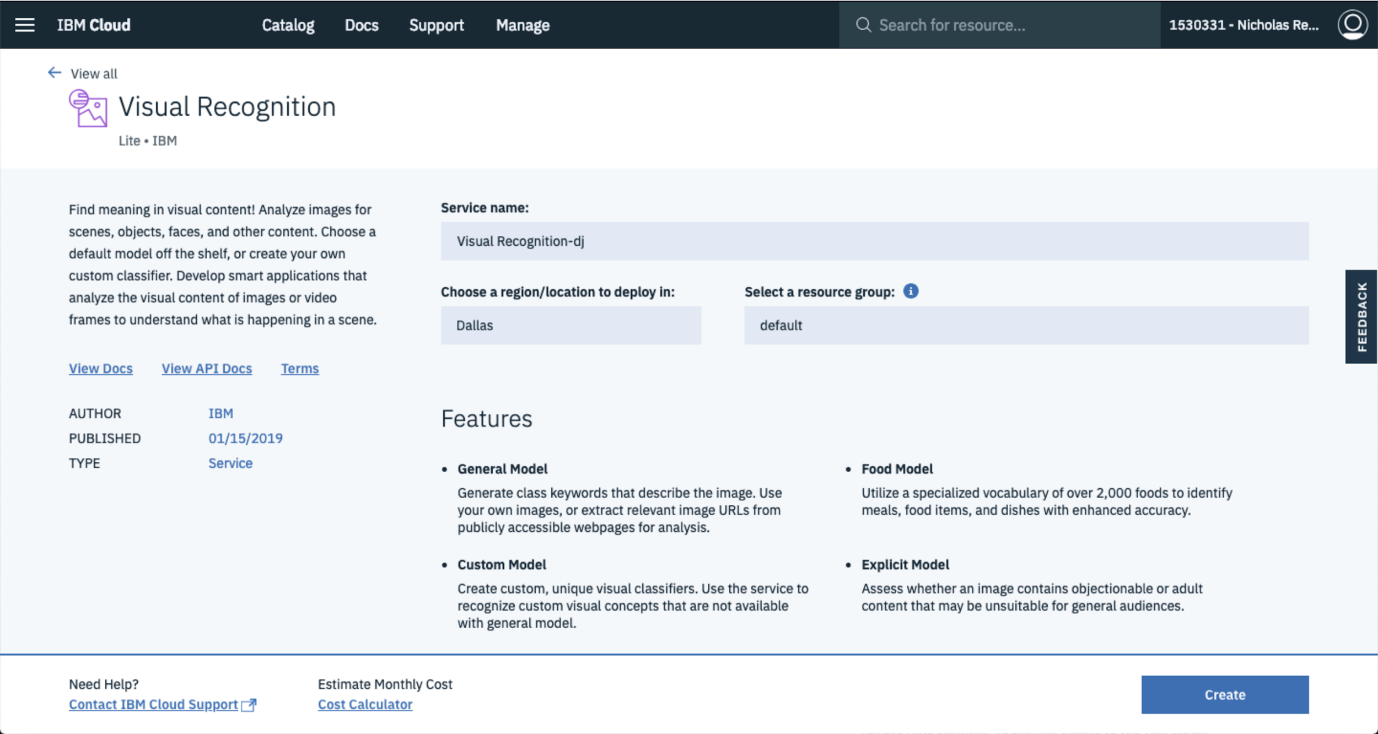
The next step is to spin up a service on IBM Cloud which will allow you to access the Visual Recognition API. This can all be done through the IBM Cloud console. Start by heading over to the

[IBM Cloud Catalog](https://console.bluemix.net/catalog/)

From there, go to AI



Here you’ll be asked to name your service, select a region and select a resource group. These can be left with their defaults.

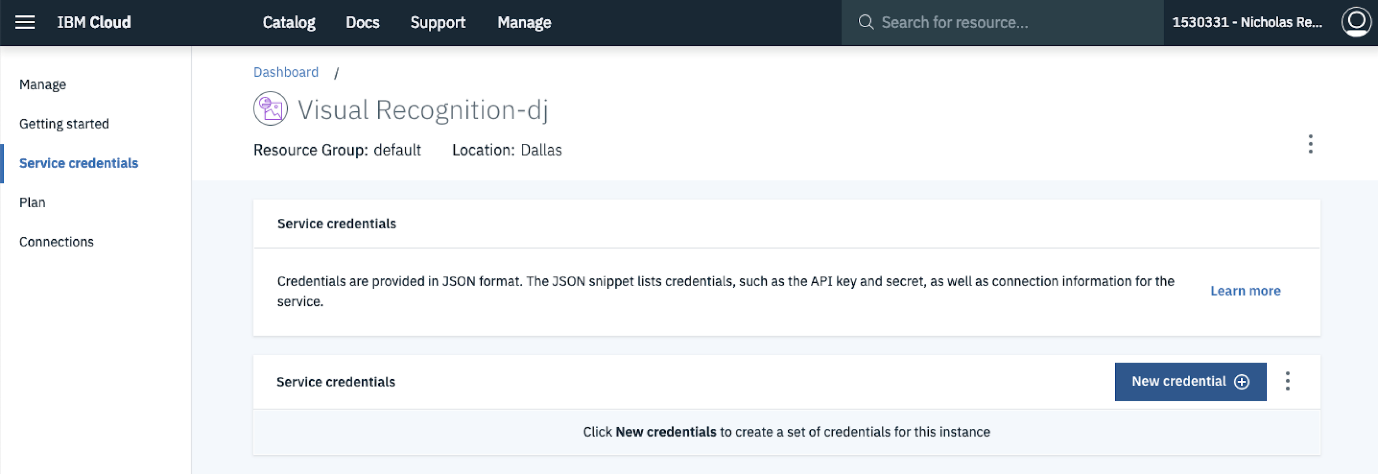


If you scroll down you’ll be asked to select a plan. The Lite plan is more than sufficient when just starting out. Select that one and click **Create**

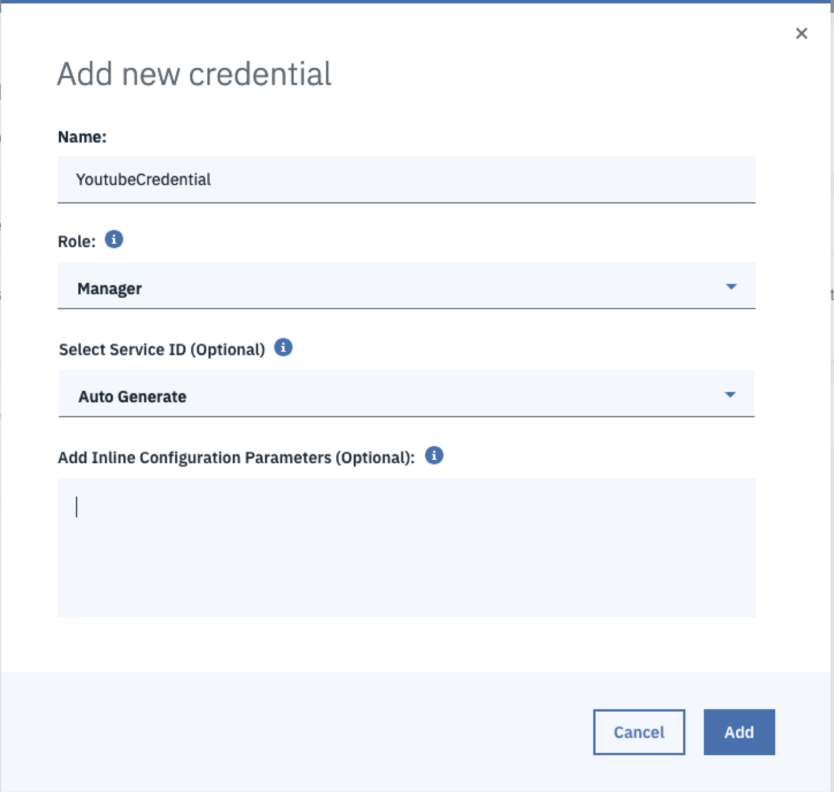


## **STEP 3 – GENERATE API CREDENTIALS**

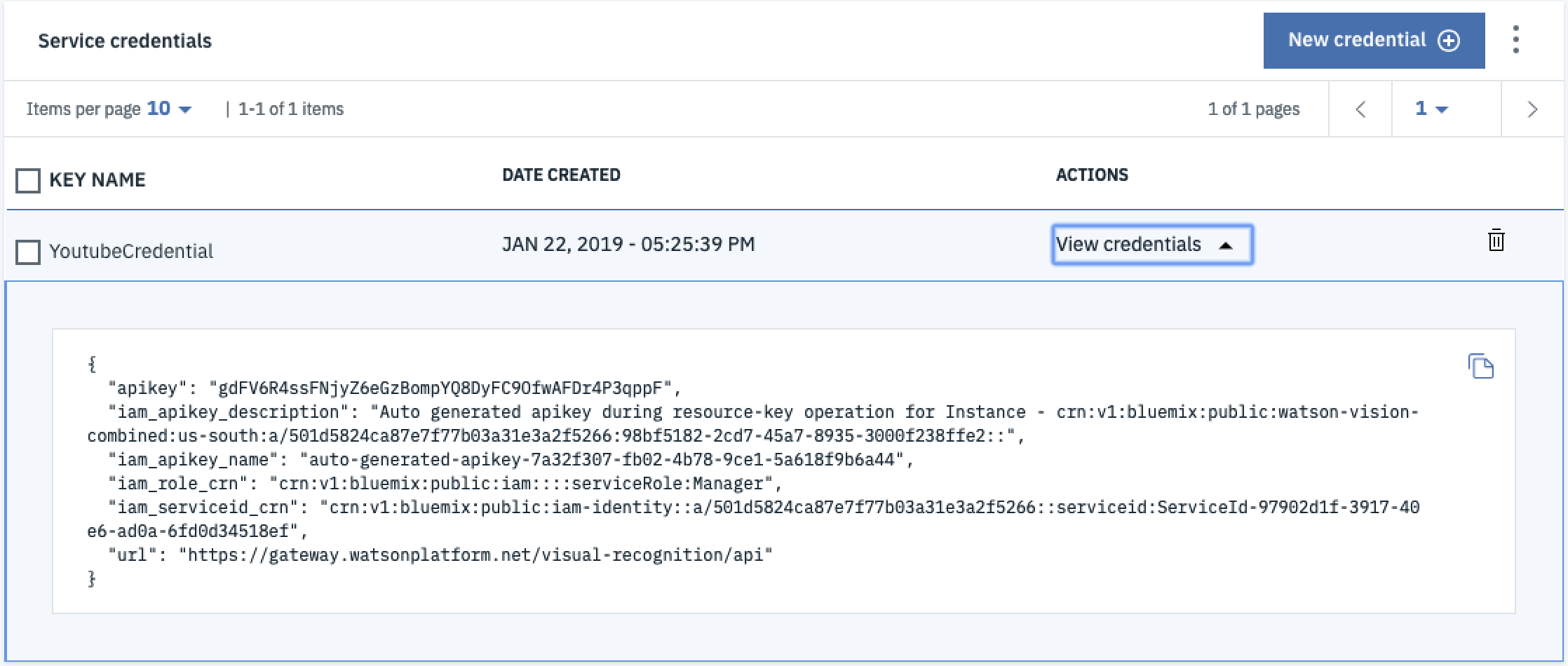
The Visual Recognition API uses token-based [IAM to authenticate requests](https://console.bluemix.net/apidocs/visual-recognition#authentication). The IAM credentials can be generated once you’ve created your service by going to Service Credential then Selecting New Credential.



Once there, create a new credential by entering a name, changing the role to Manager and selecting Auto Generate (optional) for the Select Service ID then hitting Add.



This will generate new credentials which will allow you to use the API when accessing it using Python later on. Click View Credentials and copy the apikey, you’ll need it in Step 5.

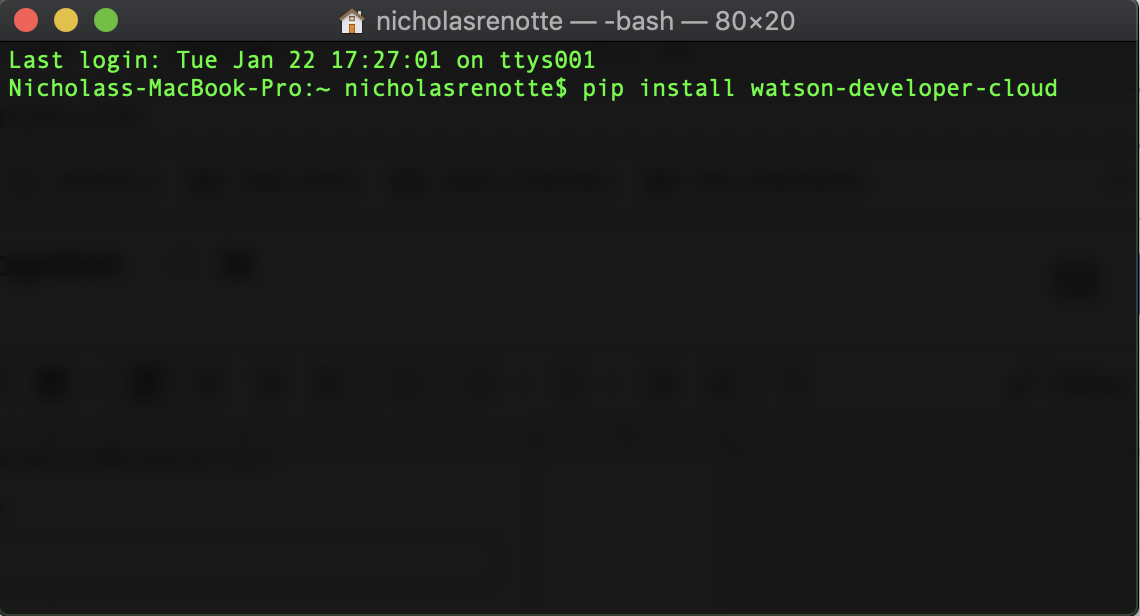


## STEP 4 – INSTALL WATSON DEVELOPER CLOUD

We’re out of setup world now and we’re finally delving into some code. Because this example is going to use Python to access the visual recognition service, we’re going to need some modules to help us out.

The key module that we’ll be using is watson-developer-cloud. To install it run:

pip install watson-developer-cloud



## STEP 5 – CLASSIFYING GENERAL IMAGES

It’s now time to start classifying some stuff. This example uses jupyter notebooks to interact with the API. If you’re not familiar with how to use notebooks, check out this [quick tutorial](https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/).

The first step is to import the json module and the visual recognition method from the watsno\_developer\_cloud module.

import json

from watson\_developer\_cloud import VisualRecognitionV3

Then create a new instance of the visual recognition service and update *iam\_apikey* to the apikey you generated in Step 3.

# Create instance of VR Service

visual\_recognition = VisualRecognitionV3(

'2018-03-19',

iam\_apikey='gdFV6R4ssFNjyZ6eGzBompYQ8DyFC9OfwAFDr4P3qppF') # Replace this with your APIKEY

Grab a url of an image you’d like to classify and update the url variable. Then run the classify method against the visual\_recognition service to retrieve the classification.

# URL that you want to classify

url = 'https://cdn.shopclues.com/images/thumbnails/18729/320/320/78279966PC14275391591432900471.jpg'

# Call classify method from service

classes\_result = visual\_recognition.classify(url=url).get\_result()

# Pretty print JSON result

print(json.dumps(classes\_result, indent=2))

The response can be pretty-printed using json.dumps and should look similar to the result shown below. The image used was a basic desktop computer that looked something like this:



Looking at the classes returned you can see that the classifier accurately classified the image as a desktop computer.

# Check classes returned

classes\_result['images'][0]['classifiers'][0]['classes']

# Expected results

[{'class': 'desktop computer',

'score': 0.959,

'type\_hierarchy': '/machine/computer/digital computer/personal computer/desktop computer'},

{'class': 'personal computer', 'score': 0.977},

{'class': 'digital computer', 'score': 0.977},

{'class': 'computer', 'score': 0.984},

{'class': 'machine', 'score': 0.984},

{'class': 'system', 'score': 0.77},

{'class': 'coal black color', 'score': 0.901}]

## STEP 6 – CLASSIFYING FOOD

One of the cool things about the visual recognition API is that you’re able to use more than just their basic classifier. For example the devs over at IBM have also developed a classifier that allows you to classify food.

To use that classifier just pass through an extra argument to the classify method. The argument required is classifier\_ids=[“food”].

# Food URL that you want to classify

url = 'http://soappotions.com/wp-content/uploads/2017/10/orange.jpg'

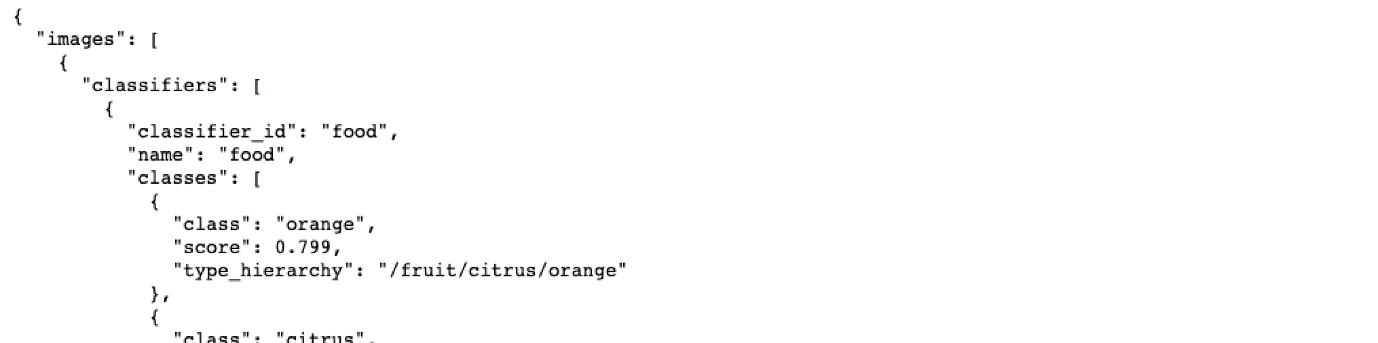
# Call classify method from service with clasifier\_ids parameter set

classes\_result = visual\_recognition.classify(url=url, classifier\_ids=["food"]).get\_result()

# Pretty print JSON result

print(json.dumps(classes\_result, indent=2))

When you run this classifier you’ll actually notice that the response shows that the classifier\_id being used is the food classifier. This might not seem all that important now but it becomes increasingly important when you start training your own models.



## STEP 7 – DETECTING FACES

Watson VR also opens up the ability to detect faces. Rather than using the classify method we’ll use detect\_faces to leverage the facial recognition api. One of the really neat things about this function is that it actually picks up multiple faces within a particular image and also estimates the gender and the age of that face!

To switch our code over so that it can detect faces, simply change the classify method to detect\_faces and run the code. (NB: Update the url to one that has images of faces as well)

# Face URL that you want to classify

url = 'https://upload.wikimedia.org/wikipedia/commons/thumb/2/2a/Donald\_Glover\_TIFF\_2015.jpg/220px-Donald\_Glover\_TIFF\_2015.jpg'

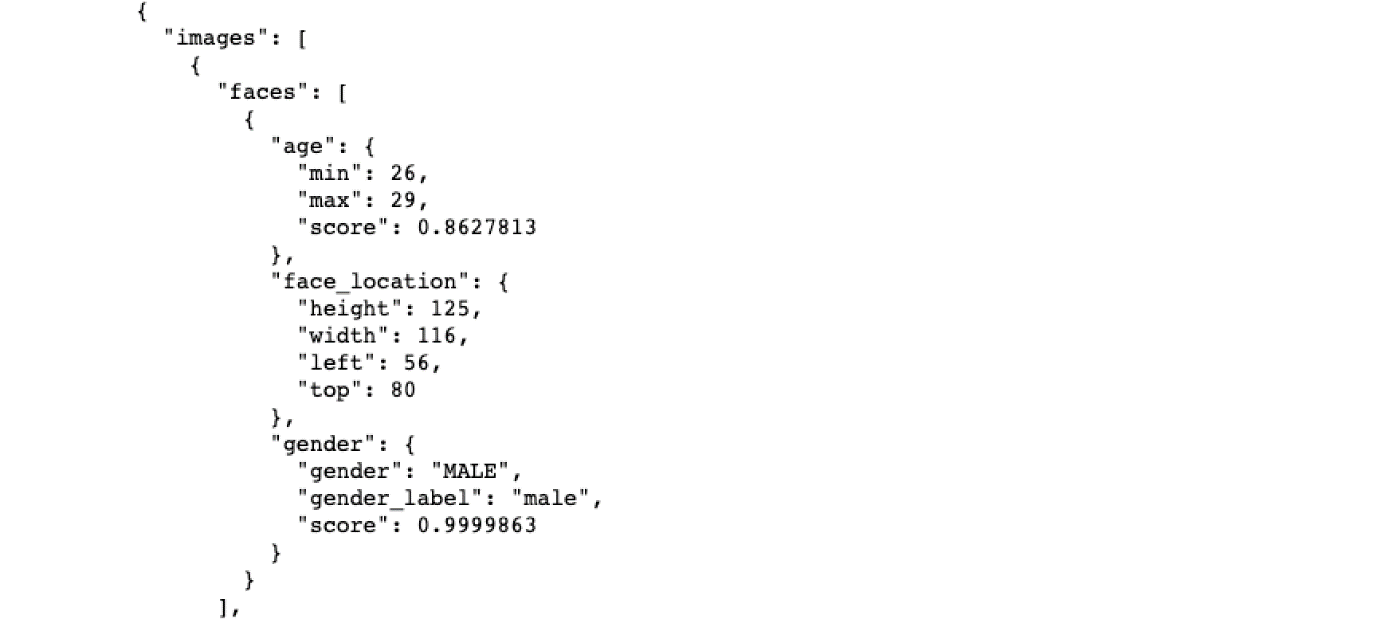
# Call detect faces method from service

classes\_result = visual\_recognition.detect\_faces(url=url).get\_result()

# Pretty print JSON result

print(json.dumps(classes\_result, indent=2))

The response should return an array of faces as well as the estimated age of that person, the location of the person’s face within the photo and the guestimated age



## STEP 8 – CUSTOM CLASSIFICATIONS

“This is great Nick, but i don’t really care about classifying fruit or detecting faces”

Well, you’re in luck because the visual recognition API allows you to train custom classes. This would be like [training your own CNN using Tensorflow](https://www.tensorflow.org/tutorials/images/image_recognition) but way easier.

To do this, you need to load zip files containing images of what you’re trying to classify as well as images of things that don’t form part of that class. Then run the create\_classifier method against your visual recognition service.

# Open each image zip file

with open('./beagle.zip', 'rb') as beagle, \

open('./golden-retriever.zip', 'rb') as goldenretriever, \

open('./husky.zip', 'rb') as husky, \

open('./cats.zip', 'rb') as cats:

# Create new classifier category

model = visual\_recognition.create\_classifier('dogs',

beagle\_positive\_examples=beagle,

goldenretriever\_positive\_examples=goldenretriever,

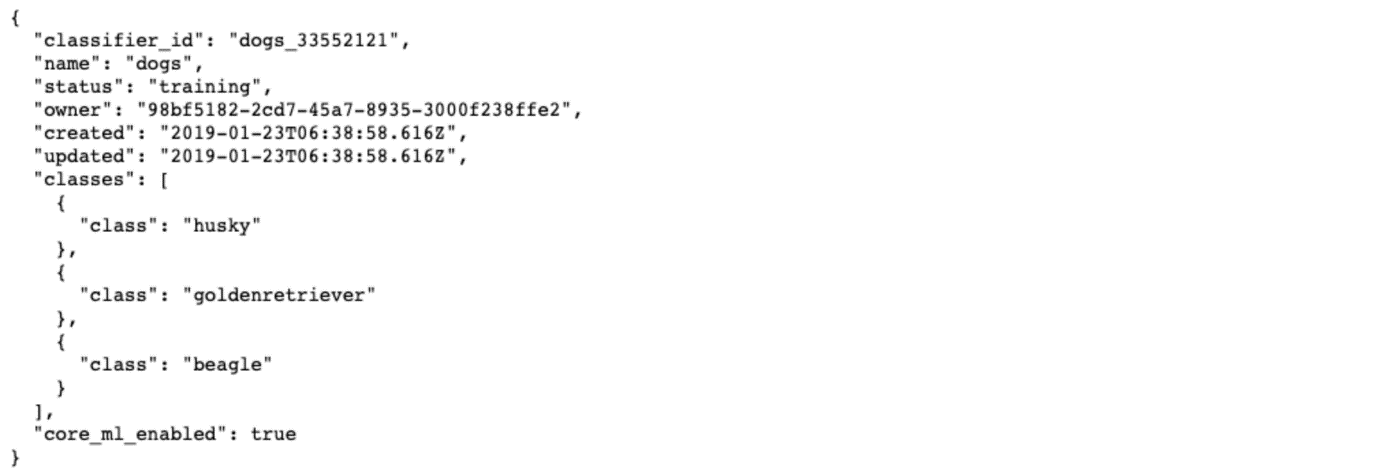
husky\_positive\_examples=husky,

negative\_examples=cats).get\_result()

# Pretty print JSON result

print(json.dumps(model, indent=2))

Assuming everything went well this will return a response that shows that the model has started training.



We can check on the training status using the get\_classifier method.

# Check status of classifier

classifier = visual\_recognition.get\_classifier(classifier\_id='dogs\_33552121').get\_result()

# Pretty print JSON result

print(json.dumps(classifier, indent=2))



Once the status has changed from training to ready we can use the model to classify custom images in this case, pictures of dogs.

# URL from custom class that you want to classify

url = 'https://i.ytimg.com/vi/bx7BjjqHf2U/maxresdefault.jpg'

# Run classifier using classifier ID from custom classifier result

result = visual\_recognition.classify(url=url, classifier\_ids=["dogs\_33552121"]).get\_result()

# Pretty print JSON result

print(json.dumps(result, indent=2))

