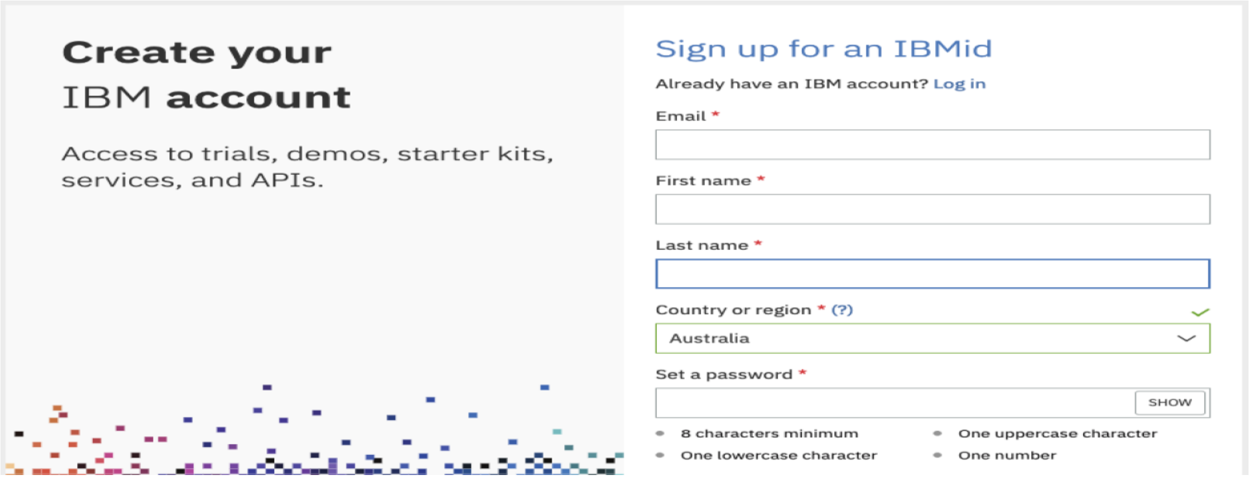
IMAGE RECOGNITION WITH IBM CLOUD CLOUD VISUAL RECOGNITION

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

How to use the IBM Watson Visual Recognition service to classify general images as well as how to train your own model to classify custom categories.

STEP 1 – GET AN IBM ACCOUNT:



Create your IBM account

Access to trials, demos, starter kits, services, and APIs.

Sign up for an IBMid

Already have an IBM account? Log in

Email *

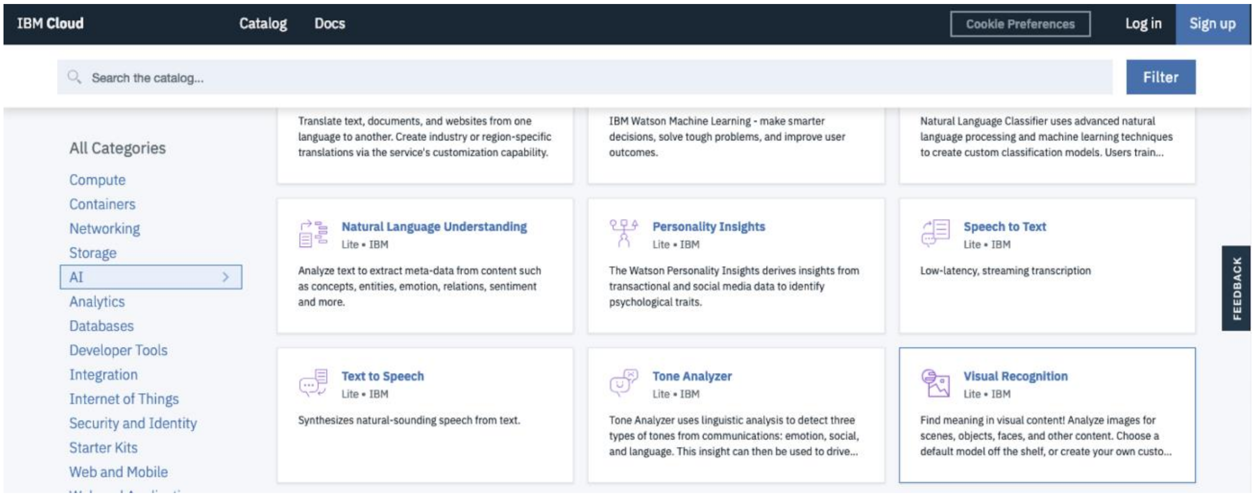
First name *

Last name *

Country or region * (?): Australia

Set a password *

STEP 2 – CREATE A SERVICE



Translate text, documents, and websites from one language to another. Create industry or region-specific translations via the service's customization capability.

IBM Watson Machine Learning - make smarter decisions, solve tough problems, and improve user outcomes.

Natural Language Classifier uses advanced natural language processing and machine learning techniques to create custom classification models. Users train...

Natural Language Understanding
Lite • IBM
Analyze text to extract meta-data from content such as concepts, entities, emotion, relations, sentiment and more.

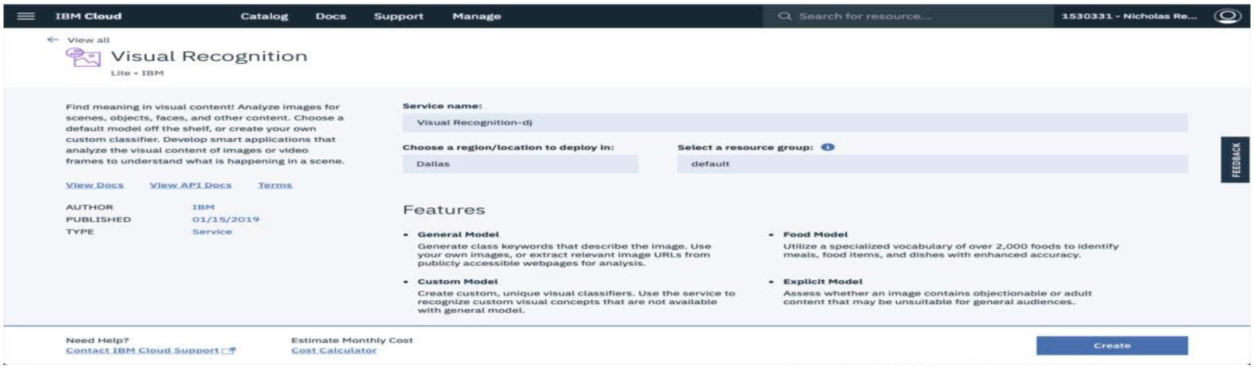
Personality Insights
Lite • IBM
The Watson Personality Insights derives insights from transactional and social media data to identify psychological traits.

Speech to Text
Lite • IBM
Low-latency, streaming transcription

Text to Speech
Lite • IBM
Synthesizes natural-sounding speech from text.

Tone Analyzer
Lite • IBM
Tone Analyzer uses linguistic analysis to detect three types of tones from communications: emotion, social, and language. This insight can then be used to drive...

Visual Recognition
Lite • IBM
Find meaning in visual content! Analyze images for scenes, objects, faces, and other content. Choose a default model off the shelf, or create your own custo...



Visual Recognition

Lite • IBM

Find meaning in visual content! Analyze images for scenes, objects, faces, and other content. Choose a default model off the shelf, or create your own custom classifier. Develop smart applications that analyze the visual content of images or video frames to understand what is happening in a scene.

AUTHOR IBM
PUBLISHED 01/15/2019
TYPE Service

Service name: Visual Recognition-dj

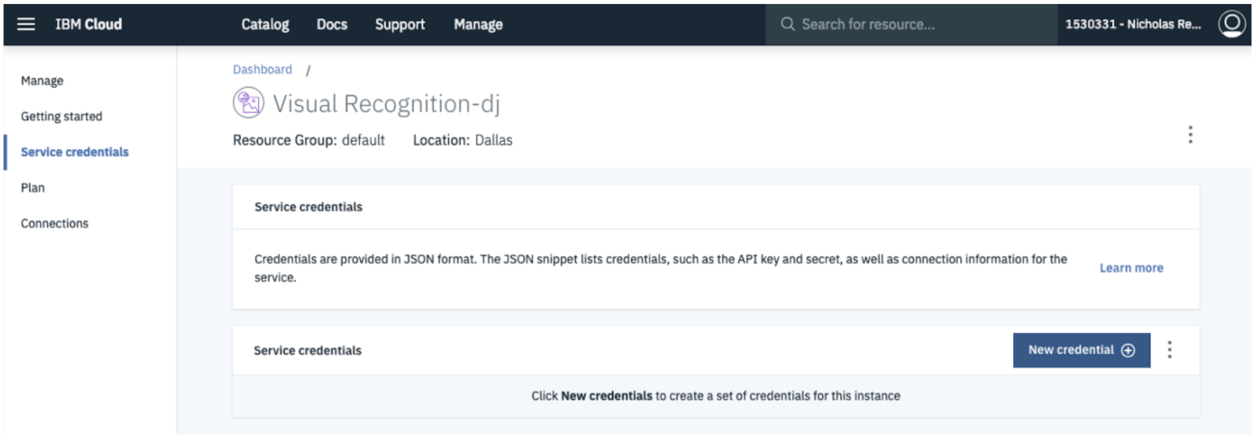
Choose a region/location to deploy in: Dallas

Select a resource group: default

Features

- **General Model** Generate class keywords that describe the image. Use your own images, or extract relevant image URLs from publicly accessible webpages for analysis.
- **Custom Model** Create custom, unique visual classifiers. Use the service to recognize custom visual concepts that are not available with general model.
- **Food Model** Utilize a specialized vocabulary of over 2,000 foods to identify meals, food items, and dishes with enhanced accuracy.
- **Explicit Model** Assess whether an image contains objectionable or adult content that may be unsuitable for general audiences.

STEP 3 – GENERATE API CREDENTIALS



Visual Recognition-dj

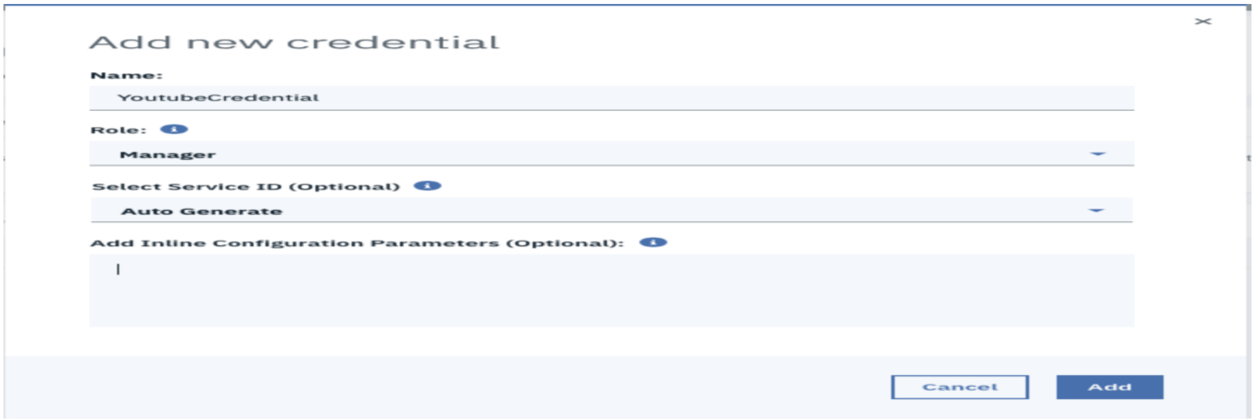
Resource Group: default Location: Dallas

Service credentials

Credentials are provided in JSON format. The JSON snippet lists credentials, such as the API key and secret, as well as connection information for the service.

Service credentials

Click **New credentials** to create a set of credentials for this instance



Add new credential

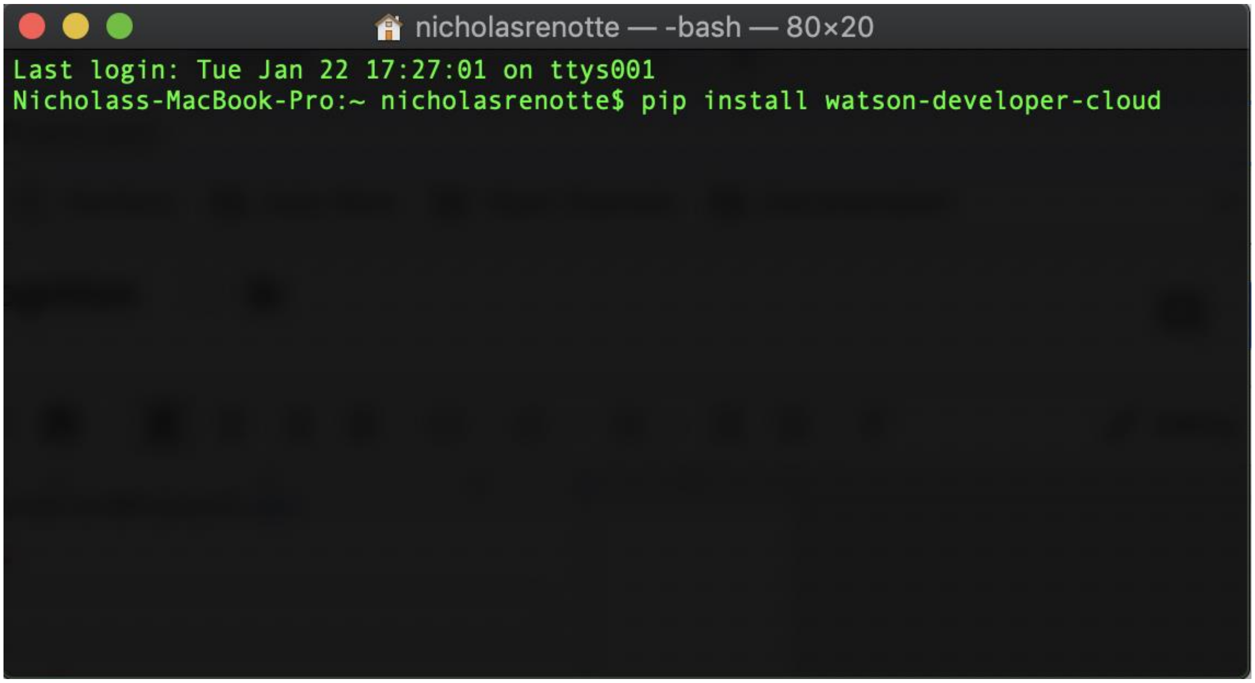
Name: YoutubeCredential

Role: Manager

Select Service ID (Optional) Auto Generate

Add Inline Configuration Parameters (Optional):

STEP 4 – INSTALL WATSON DEVELOPER CLOUD



```
Last login: Tue Jan 22 17:27:01 on ttys001
Nicholass-MacBook-Pro:~ nicholasrenotte$ pip install watson-developer-cloud
```

STEP 5 – CLASSIFYING GENERAL IMAGES

- Alright, all the setup is done. 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.
- 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.

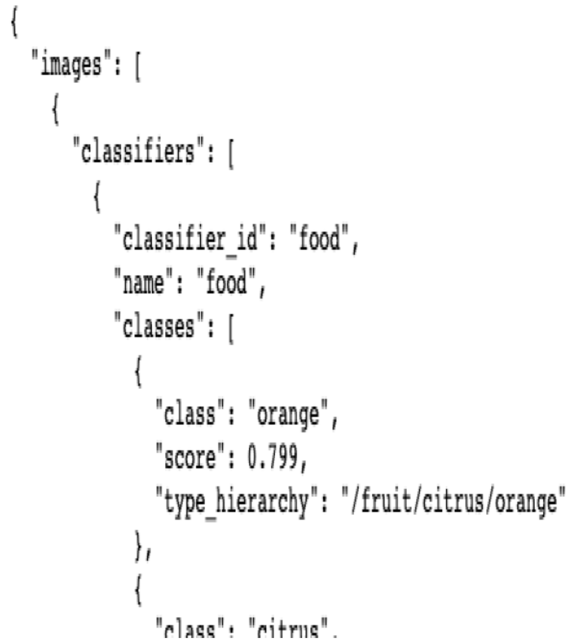
```
# Check classes returned
classes_result['images'][0]['classifiers'][0]['classes']
# Expected results
[{'class': 'desktop computer',
  'score': 0.959,
  'type_hierarchy': '/machine/computer/digital computer/personal computer/desktop computer'},
 {'class': 'personal computer', 'score': 0.977},
 {'class': '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

- 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))
```

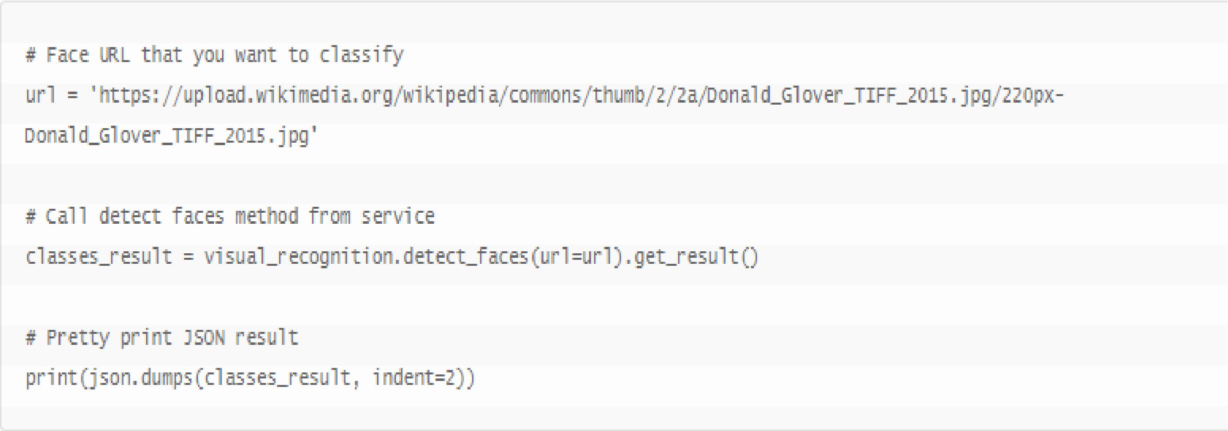
- When you run this classifier you’ll actually notice that the response shows that the classifier_id being used is the food classifier. This might not seem all that important now but it becomes increasingly important when you start training your own models.



```
{
  "images": [
    {
      "classifiers": [
        {
          "classifier_id": "food",
          "name": "food",
          "classes": [
            {
              "class": "orange",
              "score": 0.799,
              "type_hierarchy": "/fruit/citrus/orange"
            },
            {
              "class": "citrus",
```

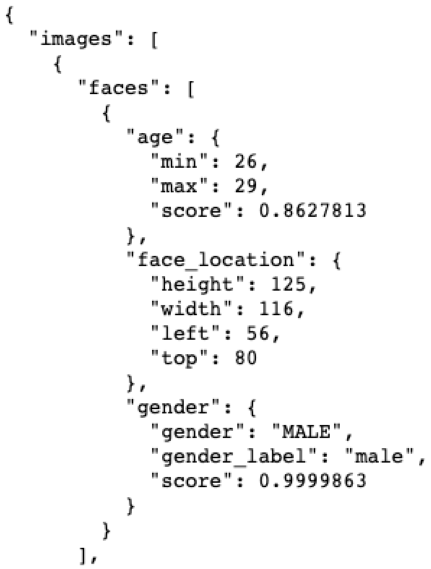
STEP 7 – DETECTING FACES

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))
```

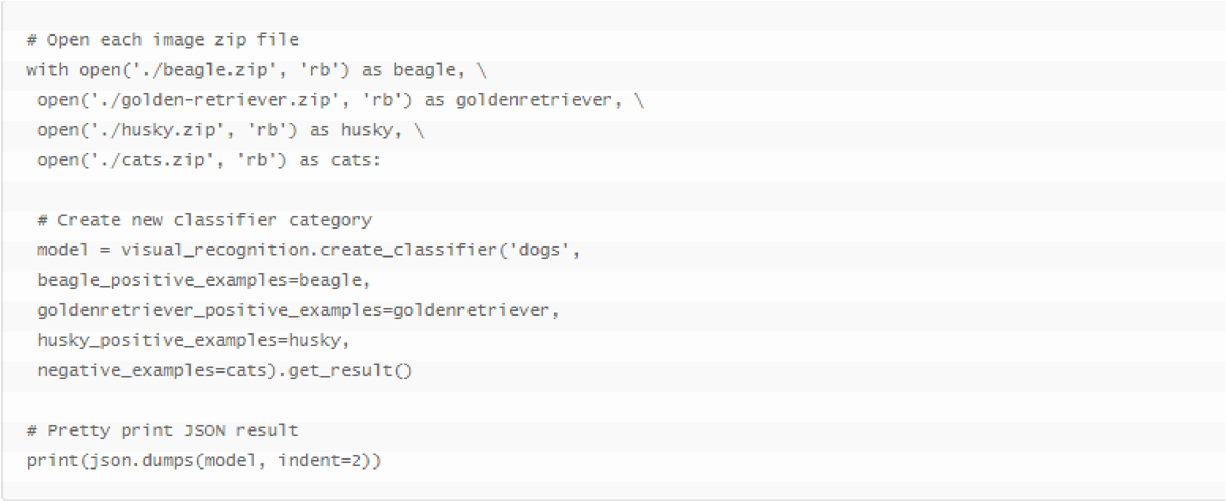
The response should return an array of faces as well as the estimated age of that person, the location of the person’s face within the photo and the guestimated age



```
{
  "images": [
    {
      "faces": [
        {
          "age": {
            "min": 26,
            "max": 29,
            "score": 0.8627813
          },
          "face_location": {
            "height": 125,
            "width": 116,
            "left": 56,
            "top": 80
          },
          "gender": {
            "gender": "MALE",
            "gender_label": "male",
            "score": 0.9999863
          }
        }
      ],
```

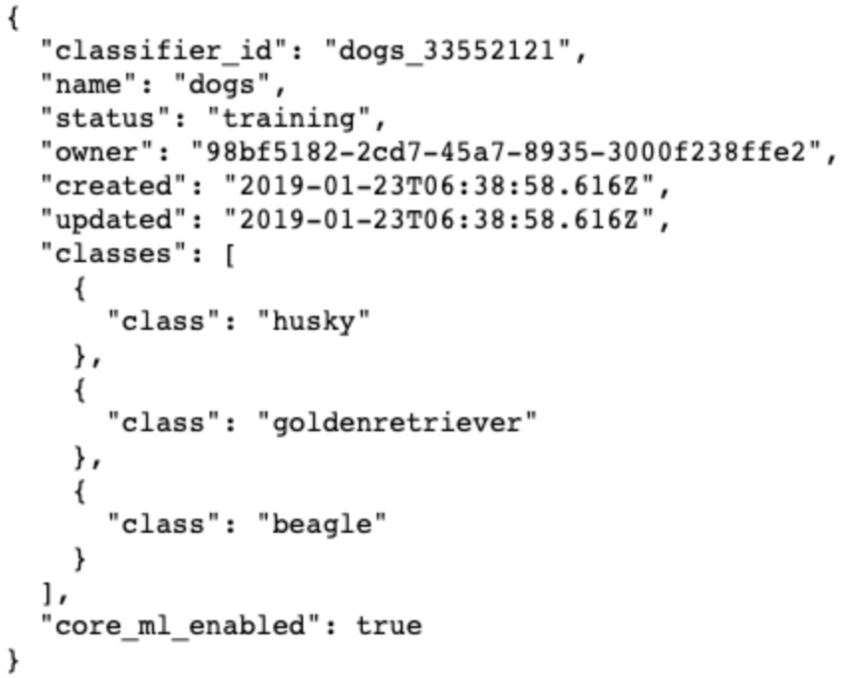
STEP 8 – CUSTOM CLASSIFICATIONS

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.



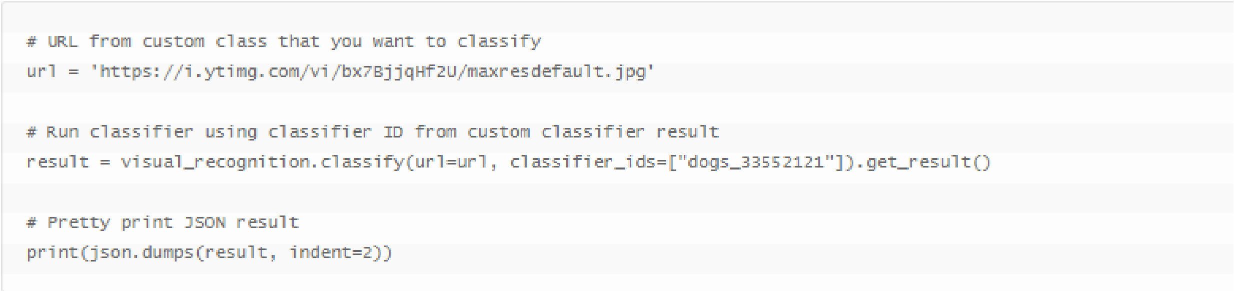
```
# Open each image zip file
with open('./beagle.zip', 'rb') as beagle, \
 open('./golden-retriever.zip', 'rb') as goldenretriever, \
 open('./husky.zip', 'rb') as husky, \
 open('./cats.zip', 'rb') as cats:

 # 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.



```
{
  "classifier_id": "dogs_33552121",
  "name": "dogs",
  "status": "training",
  "owner": "98bf5182-2cd7-45a7-8935-3000f238ffe2",
  "created": "2019-01-23T06:38:58.616Z",
  "updated": "2019-01-23T06:38:58.616Z",
  "classes": [
    {
      "class": "husky"
    },
    {
      "class": "goldenretriever"
    },
    {
      "class": "beagle"
    }
  ],
  "core_ml_enabled": true
}
```

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))
```

Thank you