Lab: Getting Started with the Model Asset Exchange and the Data Asset Exchange

In this lab you will explore the Model Asset Exchange (MAX) and the Data Asset Exchange (DAX), which are two open source Data Science resources on IBM Developer.

Upon completion of part 1 of this lab ("Explore deep learning models") you will be able to:

- Find ready-to-use deep learning models on the Model Asset Exchange
- Locate resources that guide you through deployment in a local or cloud environment
- Explore the deep learning model-serving microservice API using your web browser
- Articulate how developers can consume those microservices

Upon completion of part 2 ("Explore deep learning data sets") you will know:

- Where to find open data sets on IBM Developer
- How to explore those data sets

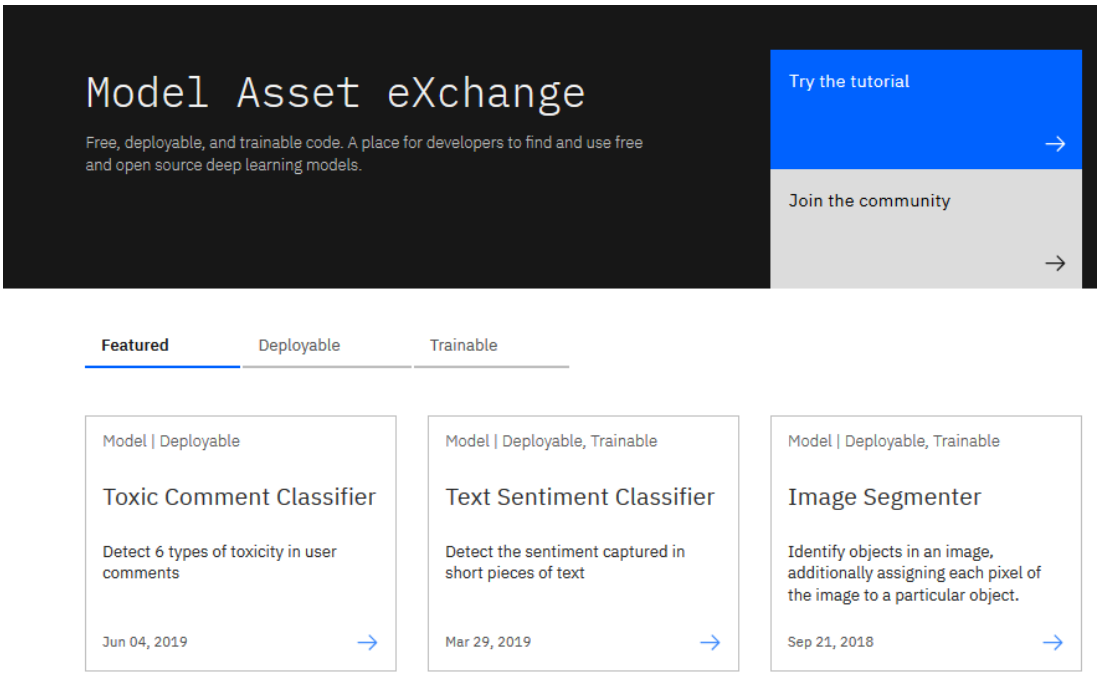
It will take you approximately 30 minutes to complete the lab. Only a web browser is required to complete the tasks.

Part 1: Explore deep learning models

The Model Asset Exchange is a curated repository of open source deep learning models for a variety of domains, such as text, image, audio, and video processing.

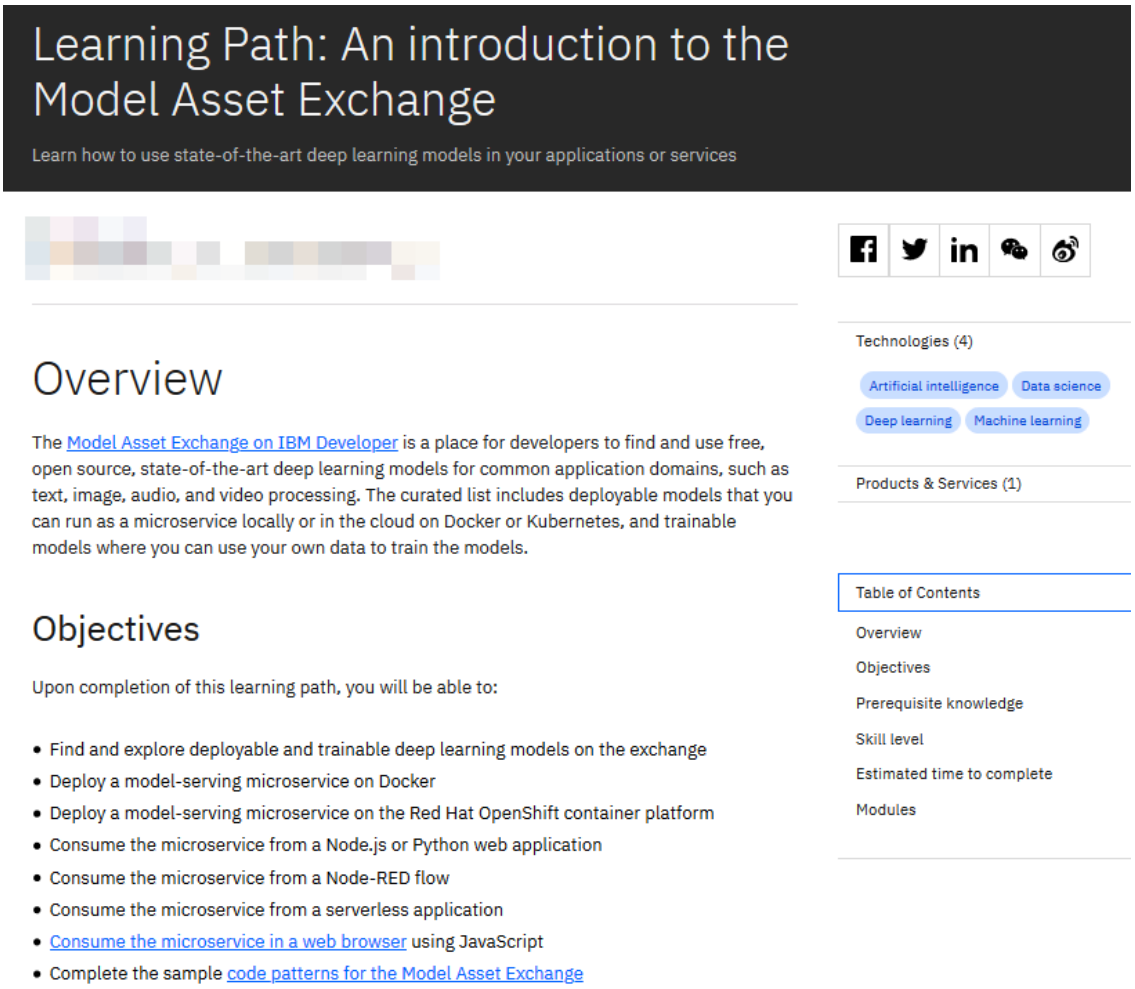
1. Open https://developer.ibm.com/ in your web browser.

2. From the main menu select **Open Source at IBM** > **Model Asset eXchange**. The MAX home page is displayed.



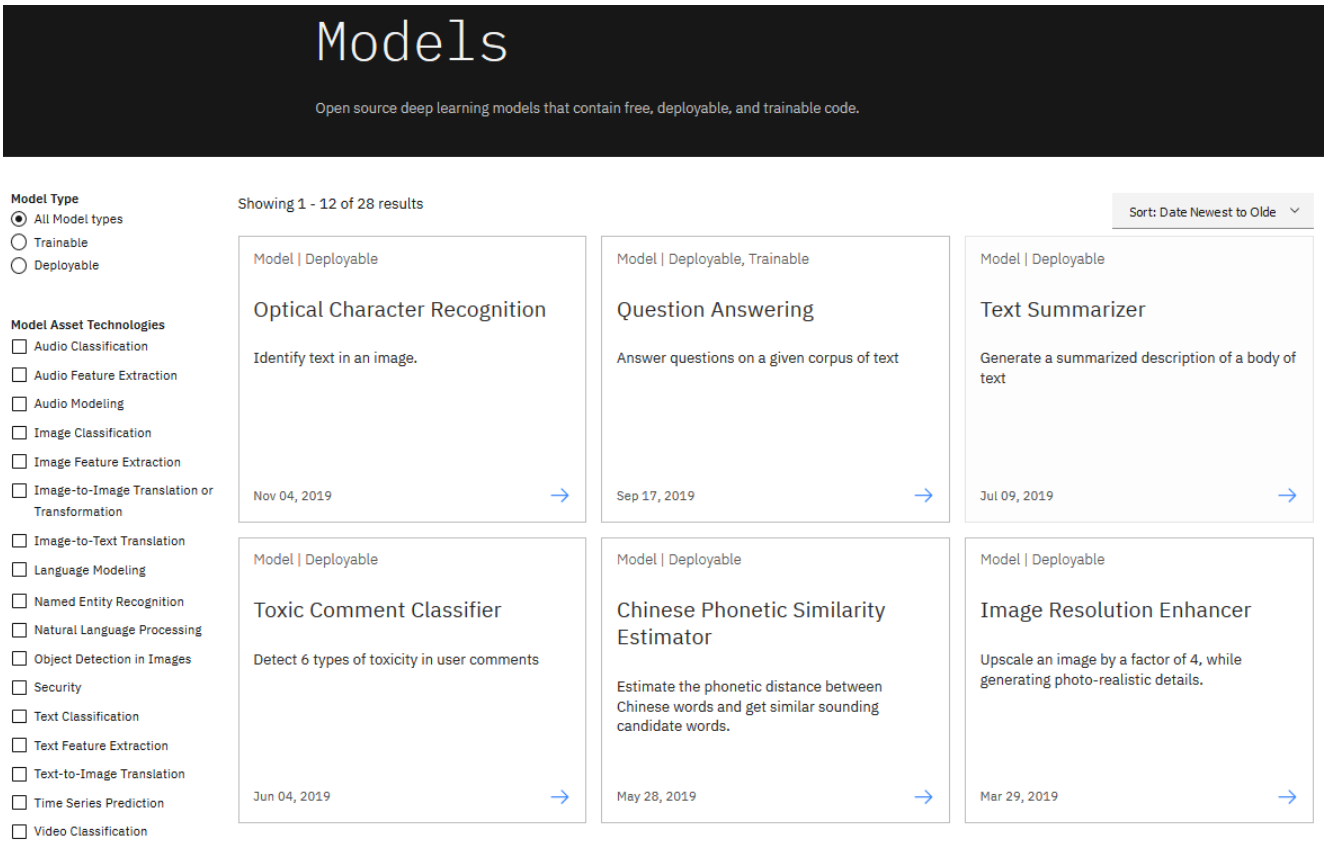
In this introductory lab exercise, we are going to focus on a few MAX key features. More detailed information is available in the Learning Path, which covers common deployment and consumption scenarios.

3. Open the Learning Path (**Try the tutorial**) in a new browser window (or tab) and briefly review the objectives and modules.



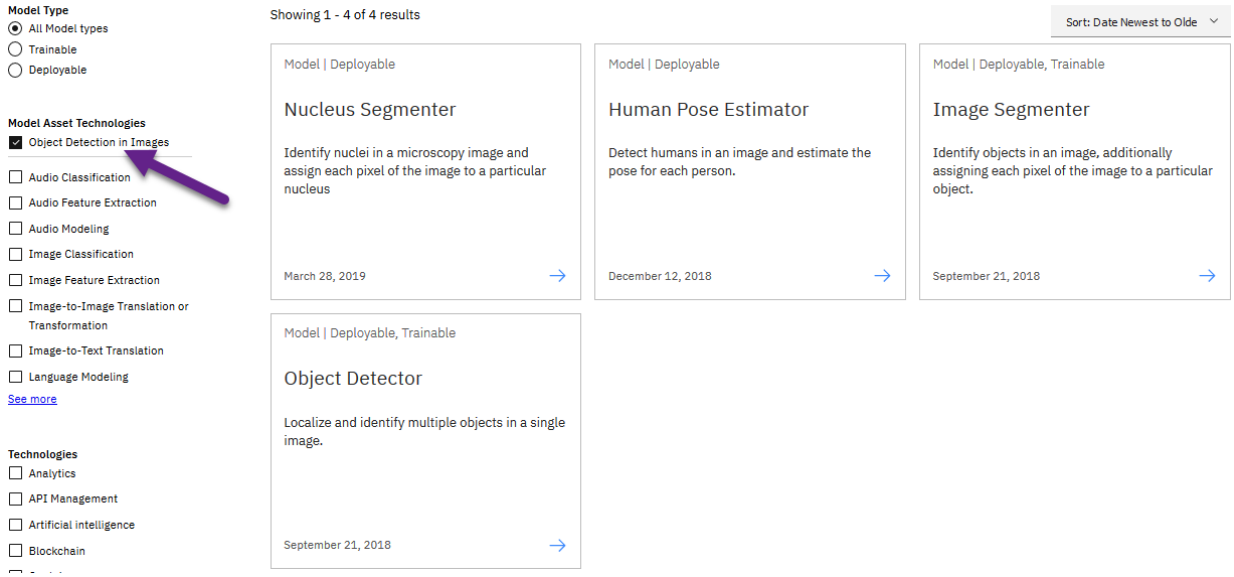
4. Close the Learning Path browser window (or tab).

5. The main page displays a small model selection. Scroll down and click **View all models** to review the complete list.



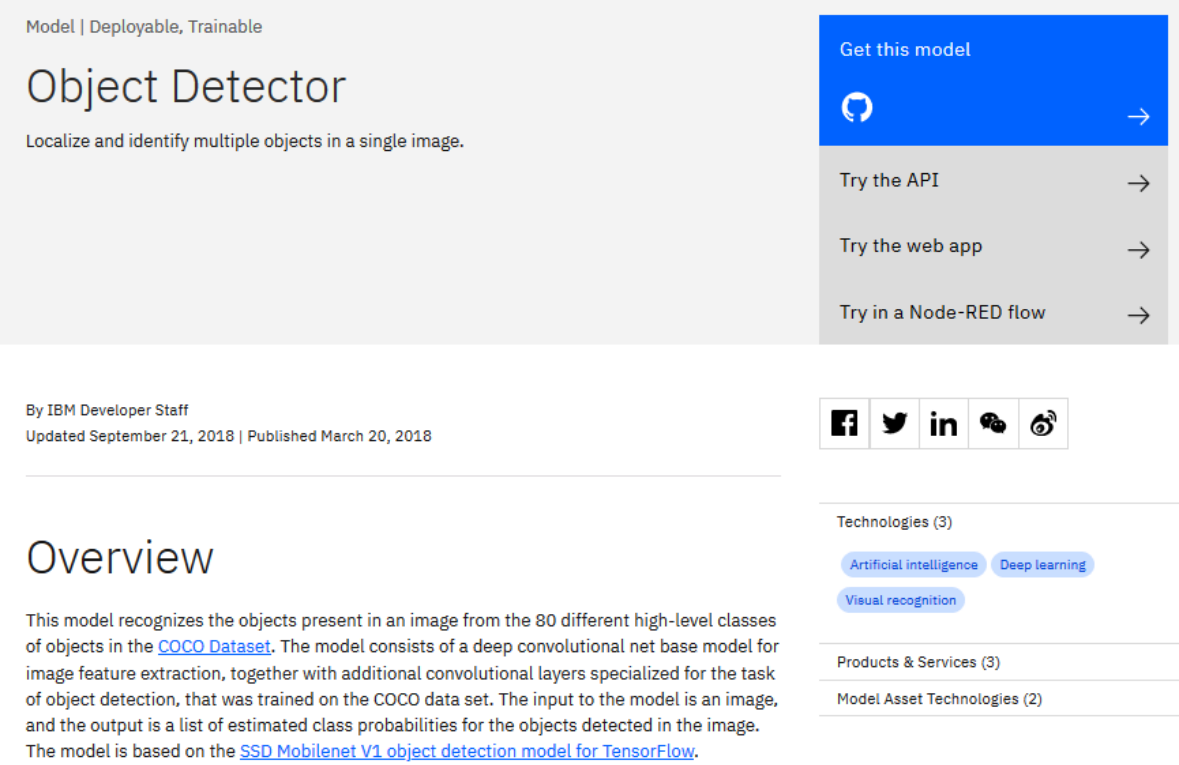
You can filter models by type and domain using the filter on the left-hand side.

6. Select **Object Detection in Images** from the **Model Asset Technologies** list on the left side.



Four models should be displayed: Nucleus Segmenter, Human Pose Estimator, Image Segmenter, and Object Detector.

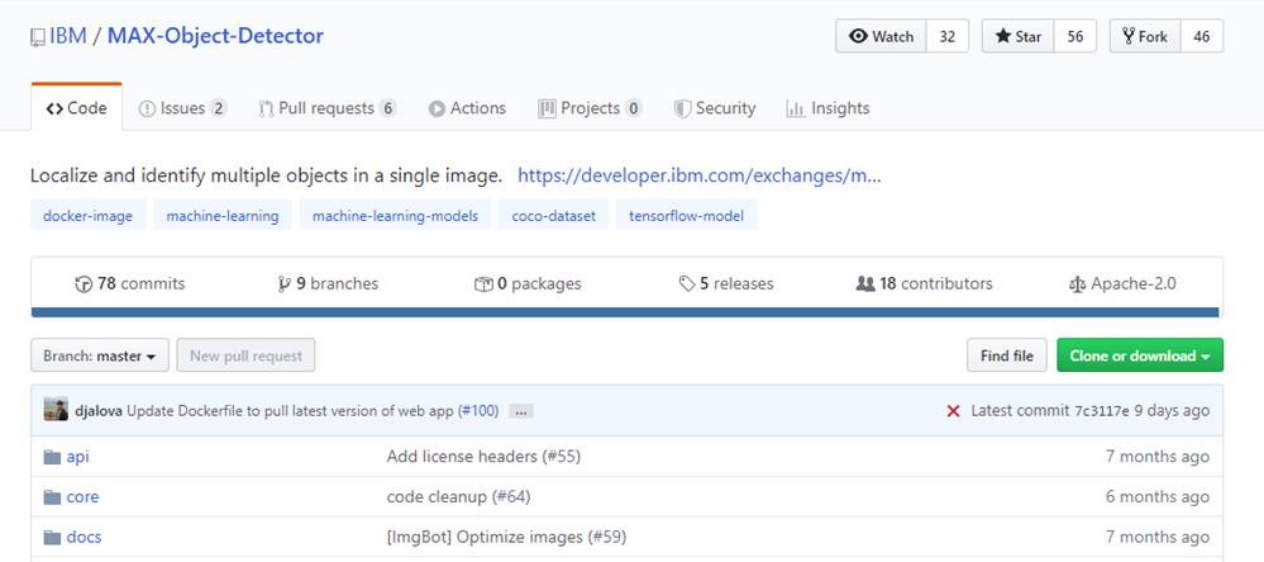
7. Select the **Object Detector** model.



On the model page you can learn about the underlying technology, read up on related research, and explore deployment and consumption options.

8. The model's source is published on GitHub and can be downloaded and modified if desired. Click **Get this model** to open the repository in a new browser tab (or window).

The repository contains everything a developer needs to manually build a customized version of the model-serving microservice.



9. Close the GitHub browser tab (or window).

10. On the Object Detector page (https://developer.ibm.com/exchanges/models/all/max-object-detector/) navigate to the **Deployment** section. All MAX models are distributed as containerized applications, which can be deployed as a microservice in a local environment, a hybrid cloud environment or a cloud environment using Docker or Kubernetes.

- Deploy from Dockerhub:

```
docker run -it -p 5000:5000 codait/max-object-detector
```

Show more

- Deploy on Red Hat OpenShift:

  Follow the instructions for the OpenShift web console or the OpenShift Container Platform CLI in this tutorial and specify `codait/max-object-detector` as the image name.

- Deploy on Kubernetes:

```
kubectl apply -f https://raw.githubusercontent.com/IBM/MAX-Object-Detector/master/m
```

Show more

  A more elaborate tutorial on how to deploy this MAX model to production on IBM Cloud can be found here.

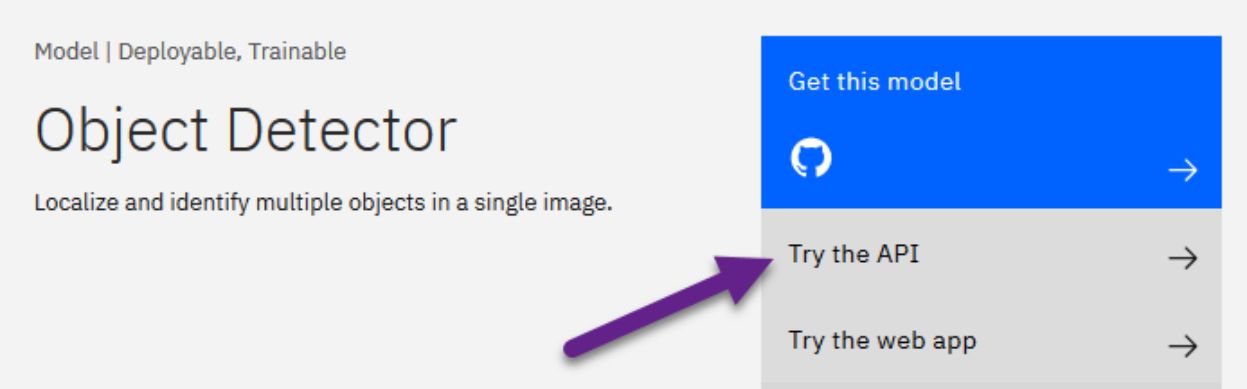
- Locally: follow the instructions in the model README on GitHub

It is beyond the scope of this lab to discuss the deployment steps. However, you can explore the model-serving microservice without having to deploy anything, which we'll discuss in the next section.

Explore a MAX model-serving microservice

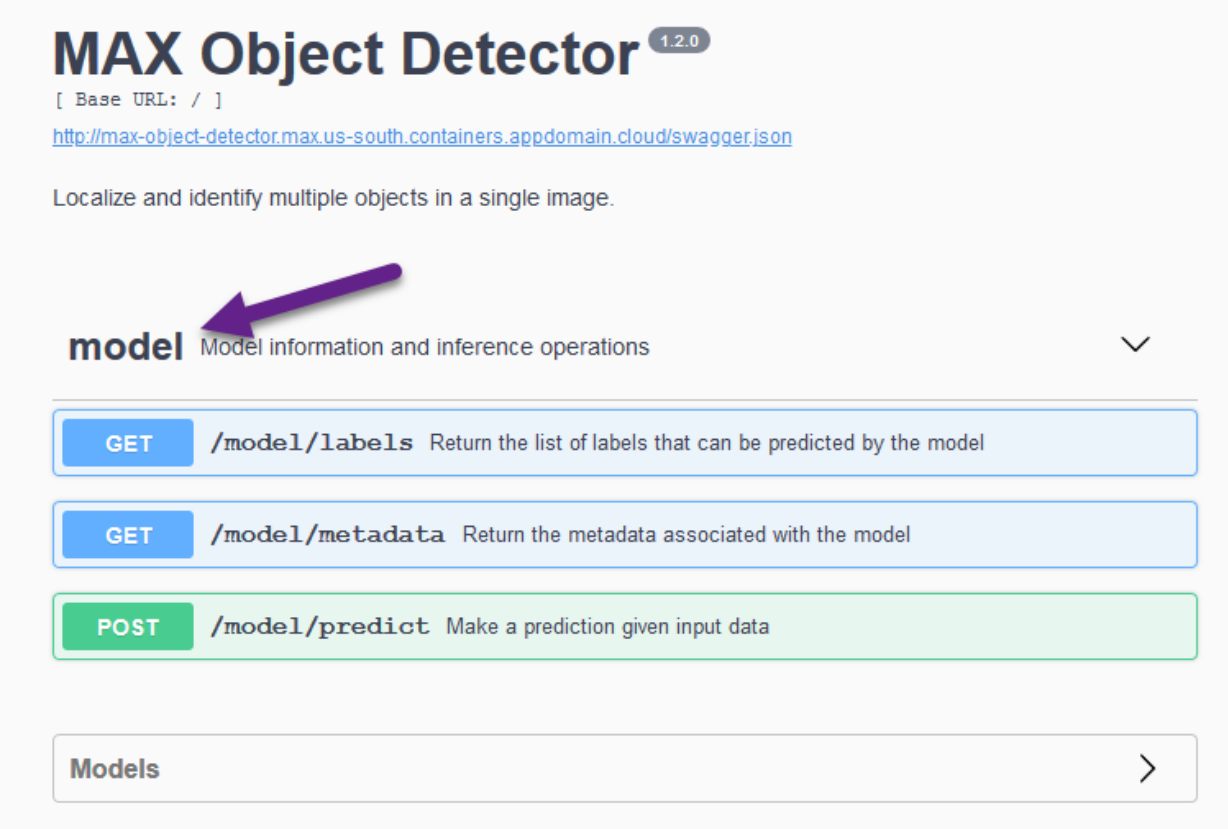
To explore a microservice you can access an evaluation microservice instance that is hosted on IBM Cloud.

11. Scroll to the top of the page and click **Try the API**.



12. The microservice's API specification page opens in a new browser tab (or window).

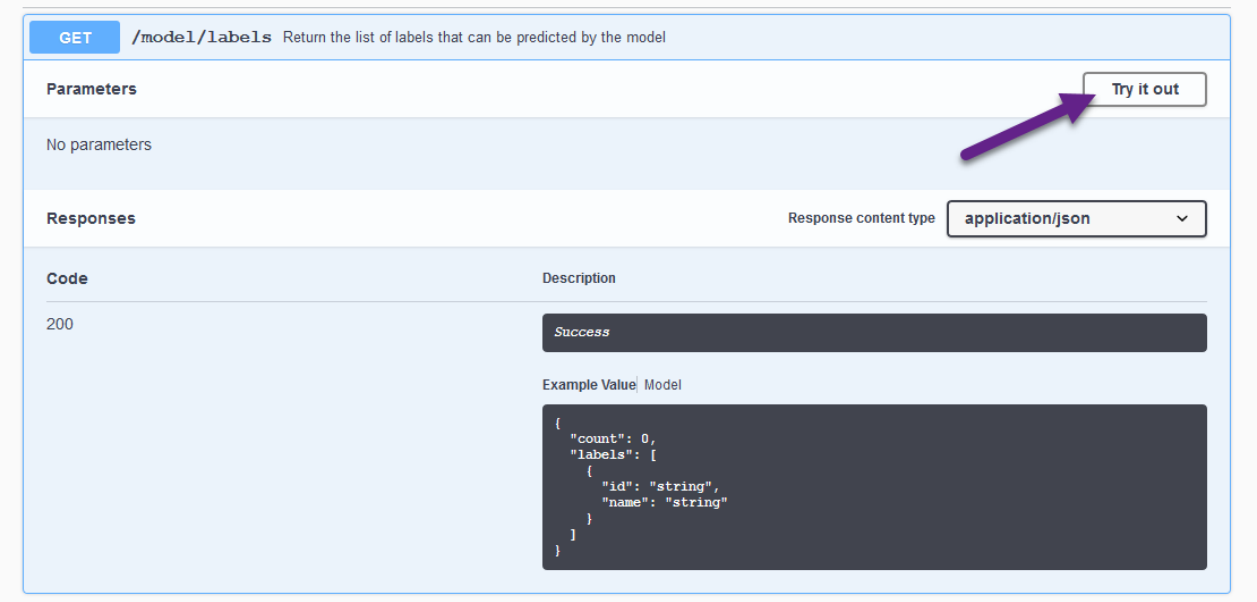
13. Expand the **model** section. The microservice's REST endpoints are displayed.



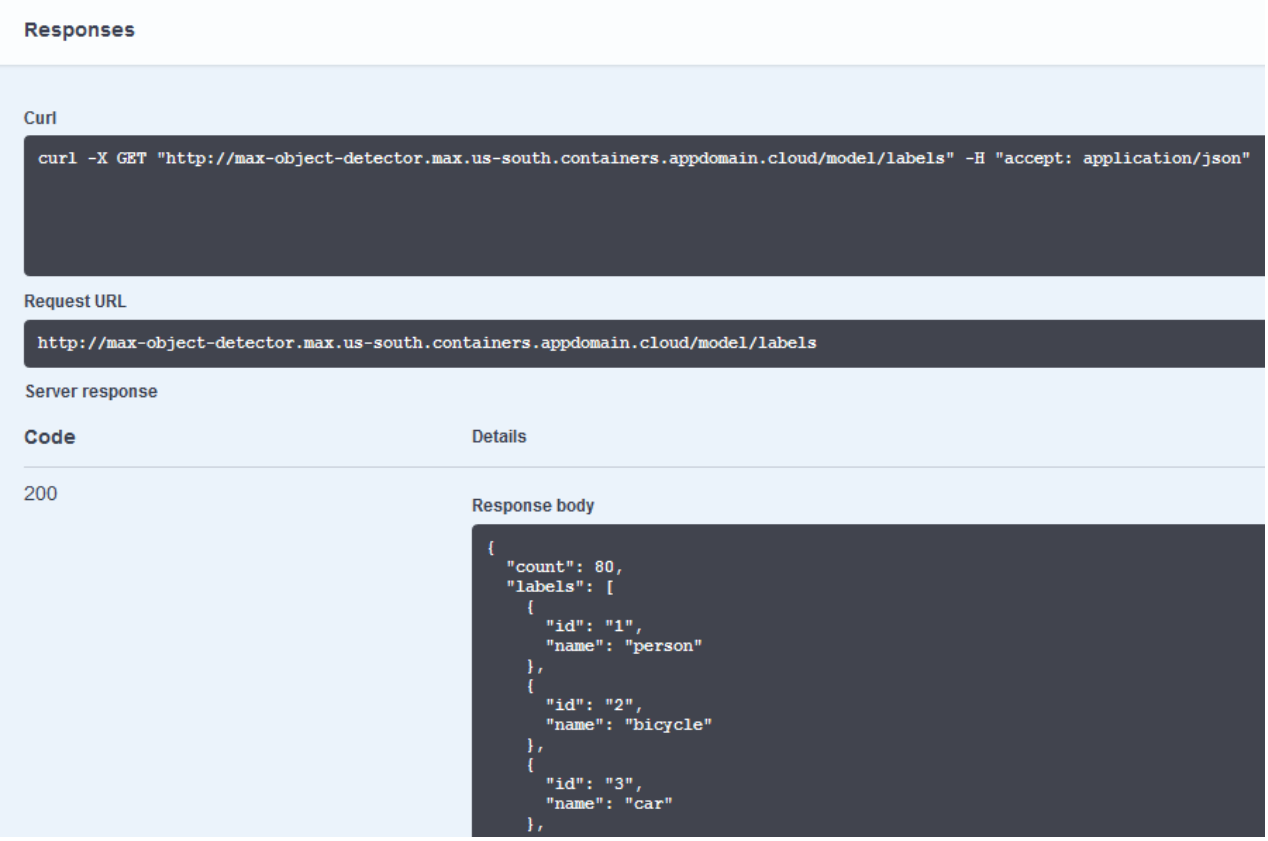
The Object Detector microservice exposes three endpoints that applications can access: a *labels* endpoint, a *metadata* endpoint and a *predict* endpoint. Let's explore the *labels* endpoint.

14. Click **GET /model/labels**. This endpoint returns a list of objects that this model can detect.

15. Click **Try it out** and **Execute**.

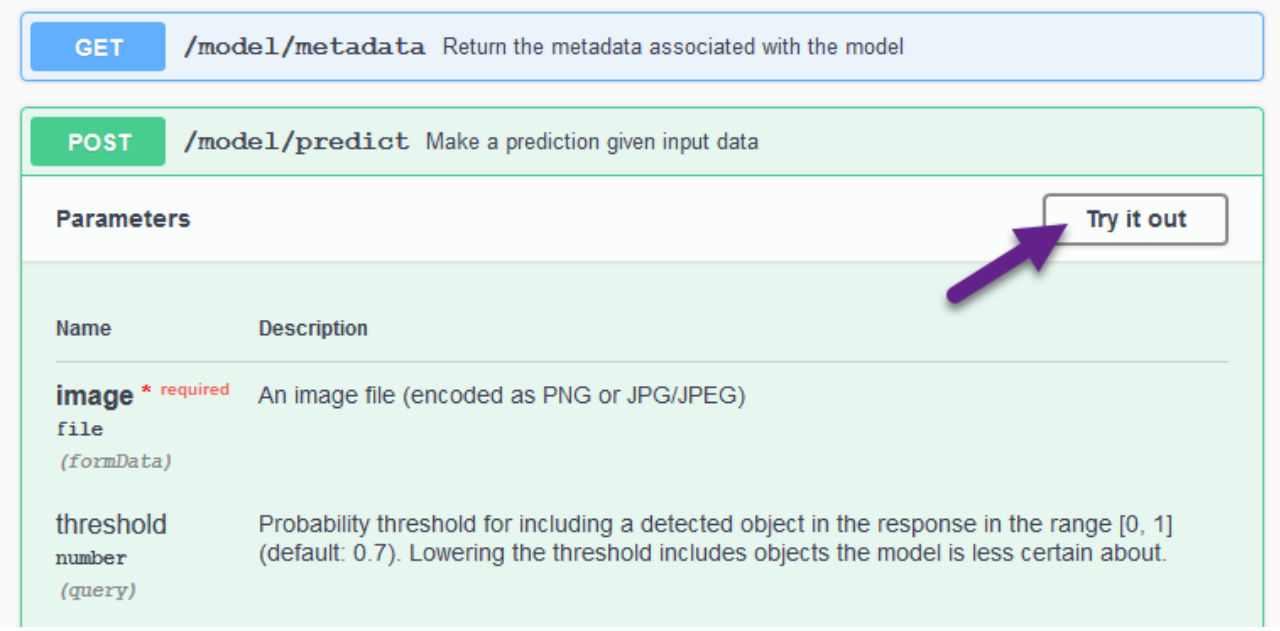


The response indicates that this model was trained to identify 80 different types of objects, such as persons, bicycles, and cars.

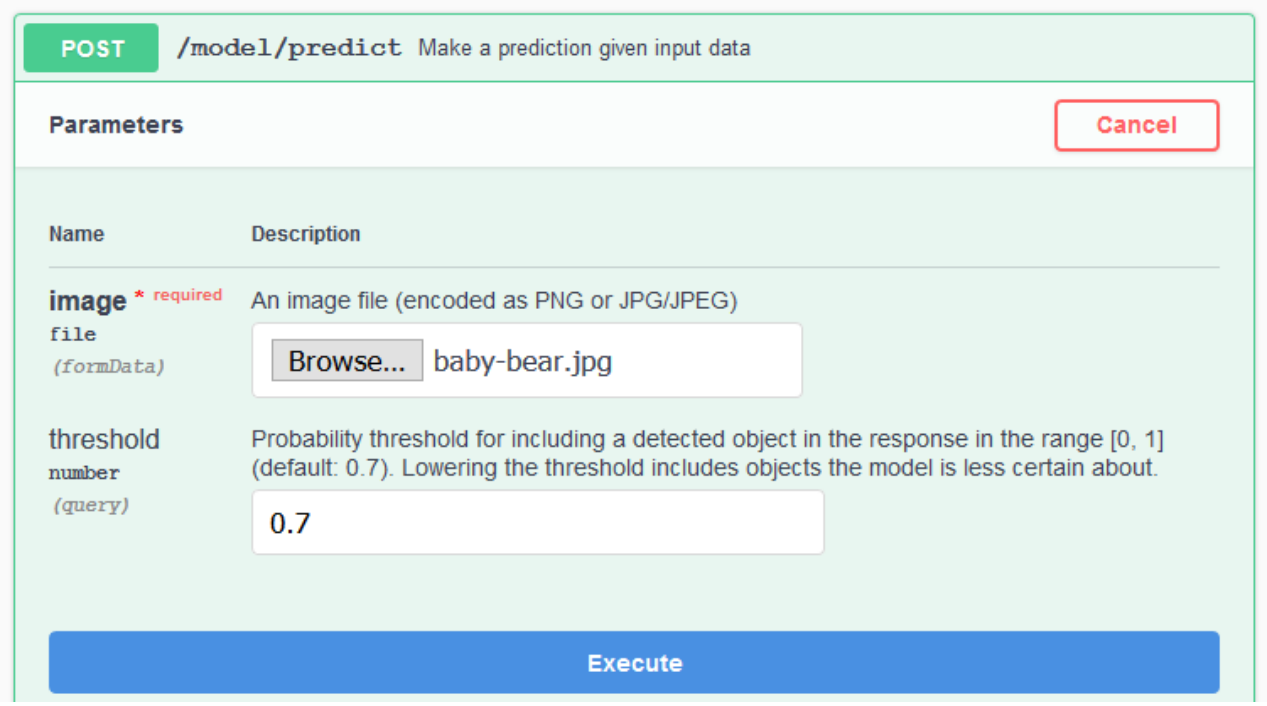


You can train some models, like the Object Detector, using your own data with the help of the Watson Machine Learning service on IBM Cloud. (You won't be doing that in this lab though.)

16. Click **POST /models/predict** and **Try it out** to test the model-serving endpoint. This endpoint is what your applications would use to analyze the content of an image.



17. Browse to a JPG or PNG file on your local machine, accept the default threshold parameter and click **Execute** to invoke the endpoint. The microservice analyzes the image and returns a list of identified objects.



18. Review the response. For each detected object an entry is returned in the predictions list, identifying the object class (e.g. person), the probability that the identified object class is correct (e.g.

0.9879012107849121), and the normalized coordinates where in the image the object was detected. Probability is a numeric value between 0 and 1. The higher the value the more confident the model is about its prediction accuracy.


```
{
 "status": "ok",
 "predictions": [
  {
   "label_id": "88",
   "label": "teddy bear",
   "probability": 0.9896332025527954,
   "detection_box": [
    0.27832502126693726,
    0.5611844062805176,
    0.643224835395813,
    0.8432191610336304
   ]
  },
  {
   "label_id": "1",
   "label": "person",
   "probability": 0.9879012107849121,
   "detection_box": [
    0.24251864850521088,
    0.26926857233047485,
    0.6558930277824402,
    0.5768759846687317
   ]
  }
 ]
}
```


The threshold input value was used as a filter, eliminating objects from the result that the model is not confident enough about.

19. Lower (or increase) the threshold value from its default of 0.7 and invoke the endpoint again. The returned results might change depending on the value you've selected.

20. Close the browser tab (or window).

In the next section we'll briefly review how applications can consume the prediction endpoint to analyze the input.

Consume a MAX model-serving microservice

Since the microservice exposes a REST API, developers can implement applications and services that consume the prediction endpoint in any programming language.

1. On the Object Detector page (https://developer.ibm.com/exchanges/models/all/max-object-detector/) navigate to the **Example Usage** section.

Example Usage

You can test or use this model

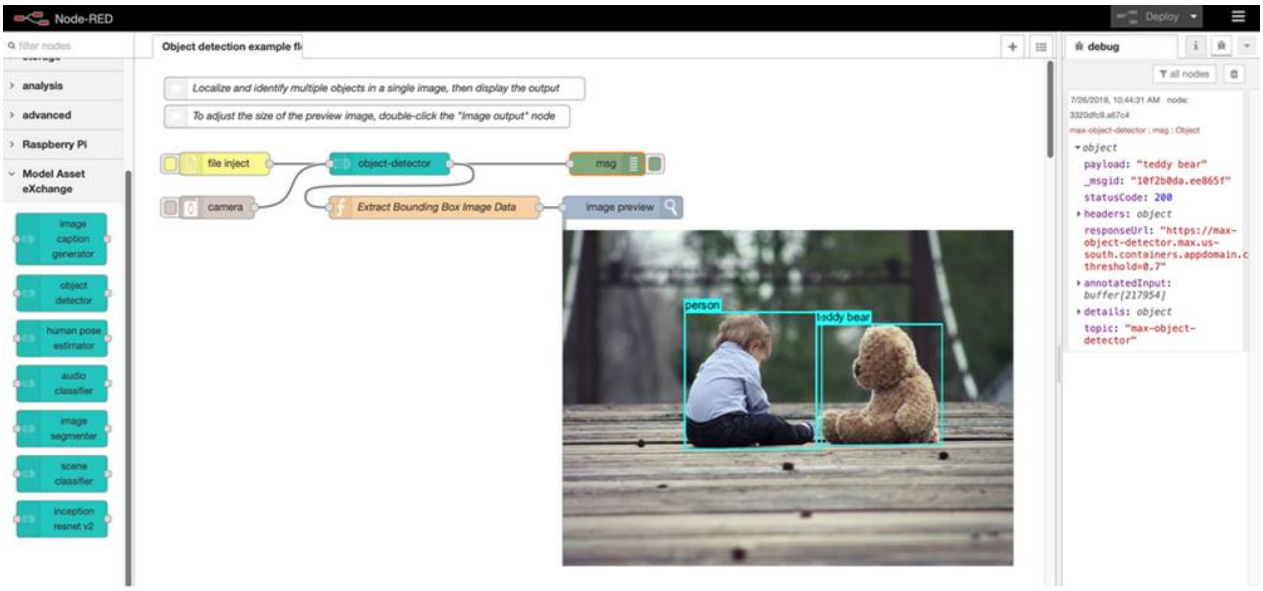
- using cURL
- in a Node-RED flow
- in CodePen
- in a serverless app

2. To make it easy for developers to get started, each model comes with a set of examples. For the Object Detector model these samples include:

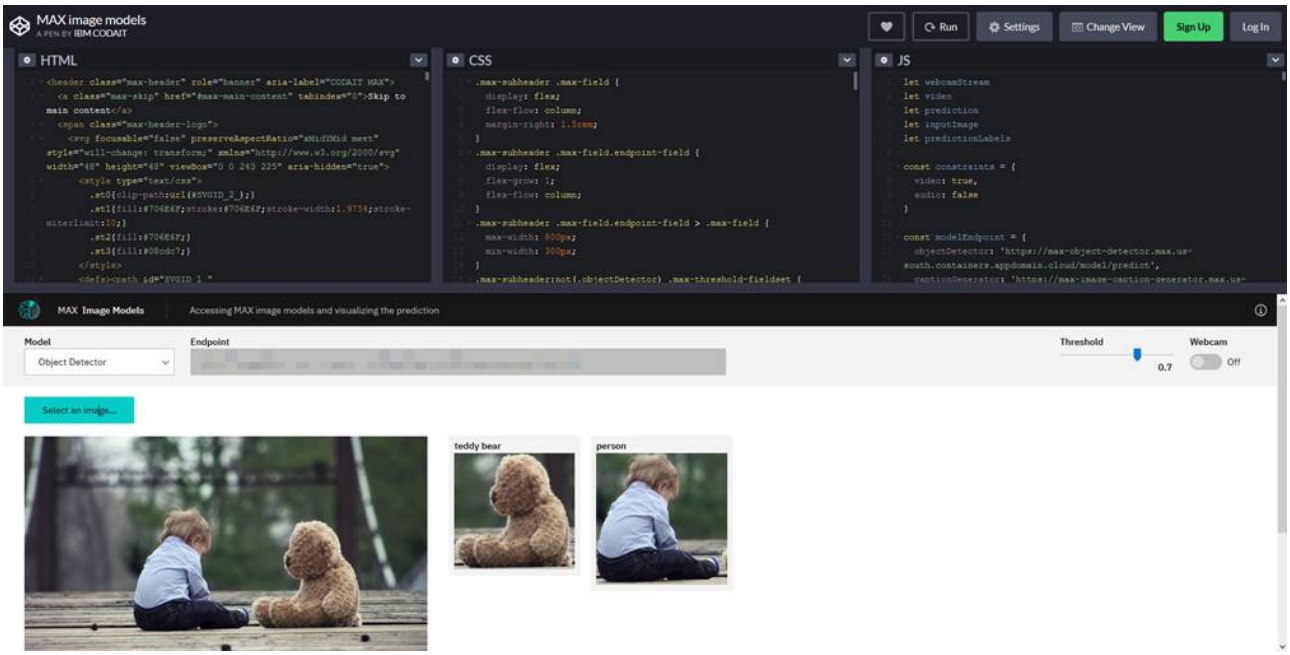
- A command line example that uses curl (https://curl.haxx.se/), a popular open source tool to download and upload files from the web.

```
bigmac:MAX-Object-Detector patti$ curl -F "image=@samples/dog-human.jpg" -XPOST
http://[redacted]/model/predict
 | python -m json.tool
{
    "predictions": [
        {
            "detection_box": [
                0.1242099404335022,
                0.12507186830043793,
                0.8423267006874084,
                0.5974075794219971
            ],
            "label": "person",
            "label_id": "1",
            "probability": 0.944034993648529
        },
        {
            "detection_box": [
                0.10447660088539124,
                0.17799153923988342,
                0.8422801494598389,
                0.732001781463623
            ],
            "label": "dog",
            "label_id": "18",
            "probability": 0.8645513653755188
        }
    ],
    "status": "ok"
}
```

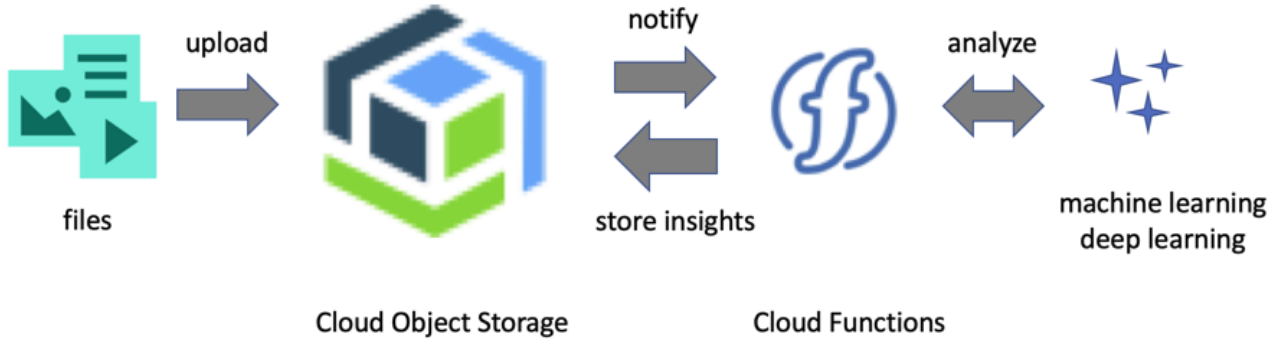
- Node-RED flows that illustrate how to incorporate object detection into Internet of Things (IOT) applications.



- JavaScript code pens that illustrate how to utilize object detection in a web browser.



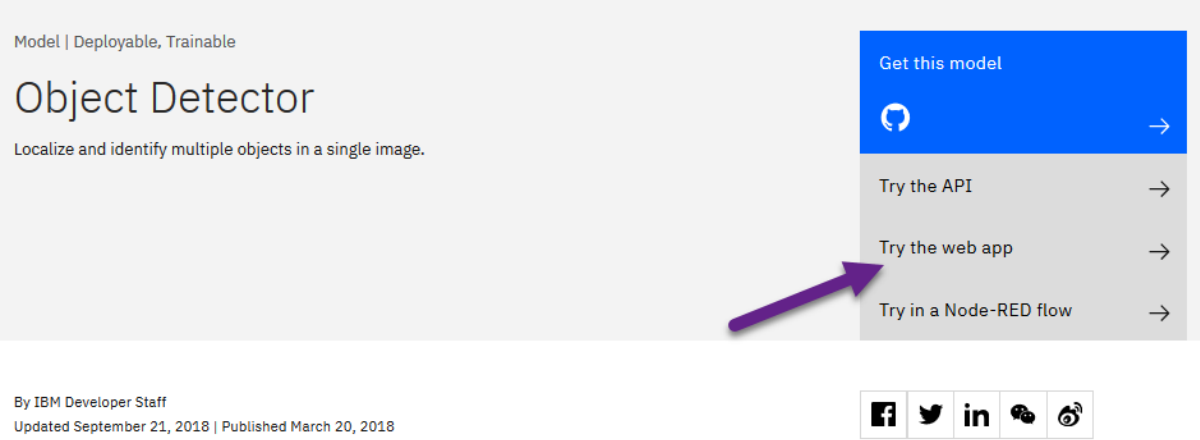
- A tutorial that outlines how to build a serverless application using IBM Cloud Functions, which is a Functions as a Service platform based on Apache OpenWhisk.



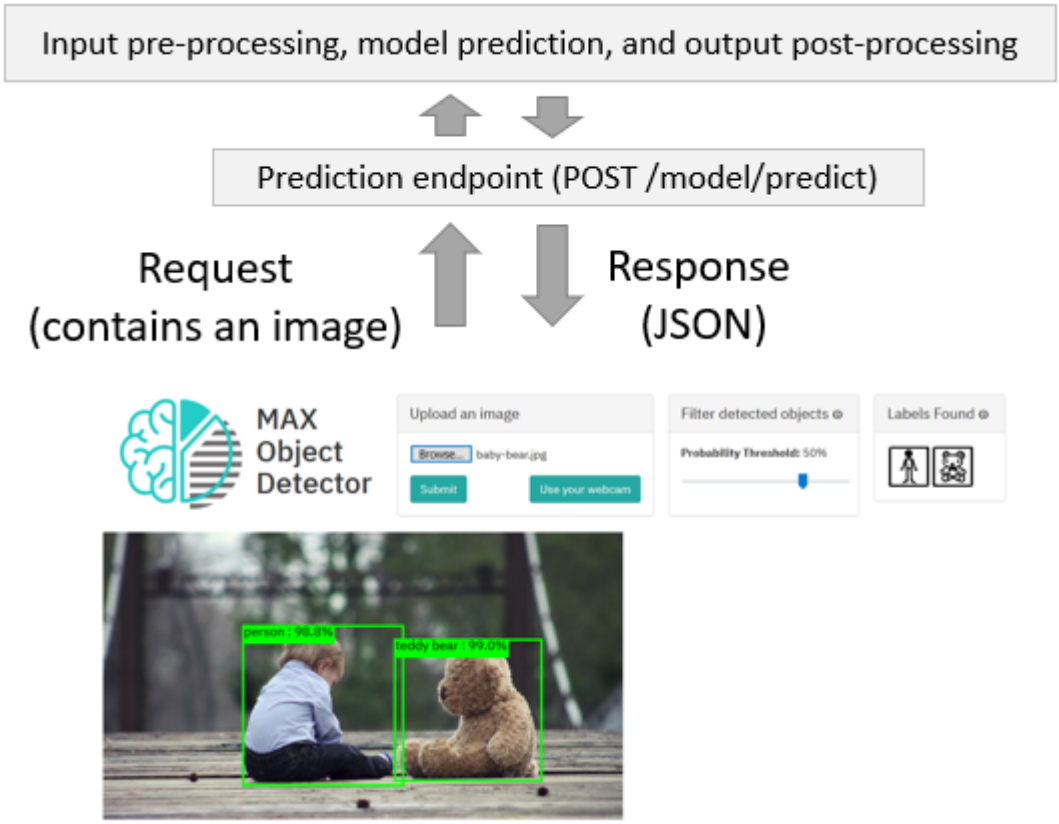
Some models, like the Object Detector, also come with a sample application that you can try out without having to install anything.

3. On the Object Detector page (https://developer.ibm.com/exchanges/models/all/max-object-detector/) scroll to the top.

4. Click **Try the web app** to launch the demo application in your web browser.



The demo web application uploads an image (or takes a picture using the web cam), sends a request to an Object Detector microservice and visualizes the response by drawing bounding boxes around detected objects and attaching a label.



5. Upload an image (or take a picture using the web camera) and inspect the visualized results.

6. Change the filter conditions (e.g. lower the probability threshold) and observe how they impact the visualized results. Note the web application caches the response and applies the filter on the cached data. This was done for two reasons:

- It eliminates the need to re-process the image using the deep learning microservice (which significantly lowers the application's response time)
- It illustrates how client-side filtering can be applied to the results if the prediction endpoint doesn't support filtering by the desired criteria.

If you are interested in learning more about how the application was implemented, take a look at the code pattern: https://developer.ibm.com/patterns/create-a-web-app-to-interact-with-objects-detected-using-machine-learning/

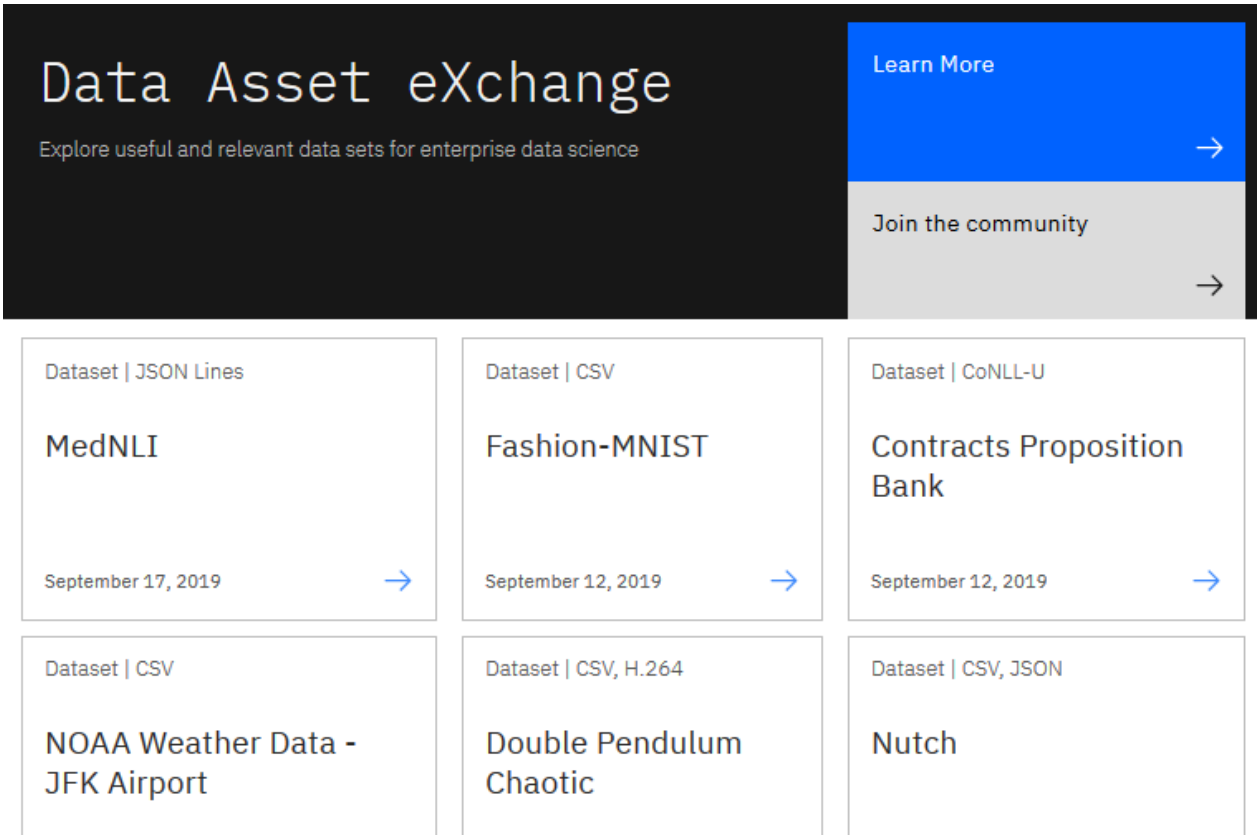
This concludes part 1 of this lab, which introduced the Model Asset Exchange.

Part 2: Explore deep learning data sets

The Data Asset Exchange is a curated collection of open data sets from IBM Research and 3rd parties that you can use to train models.

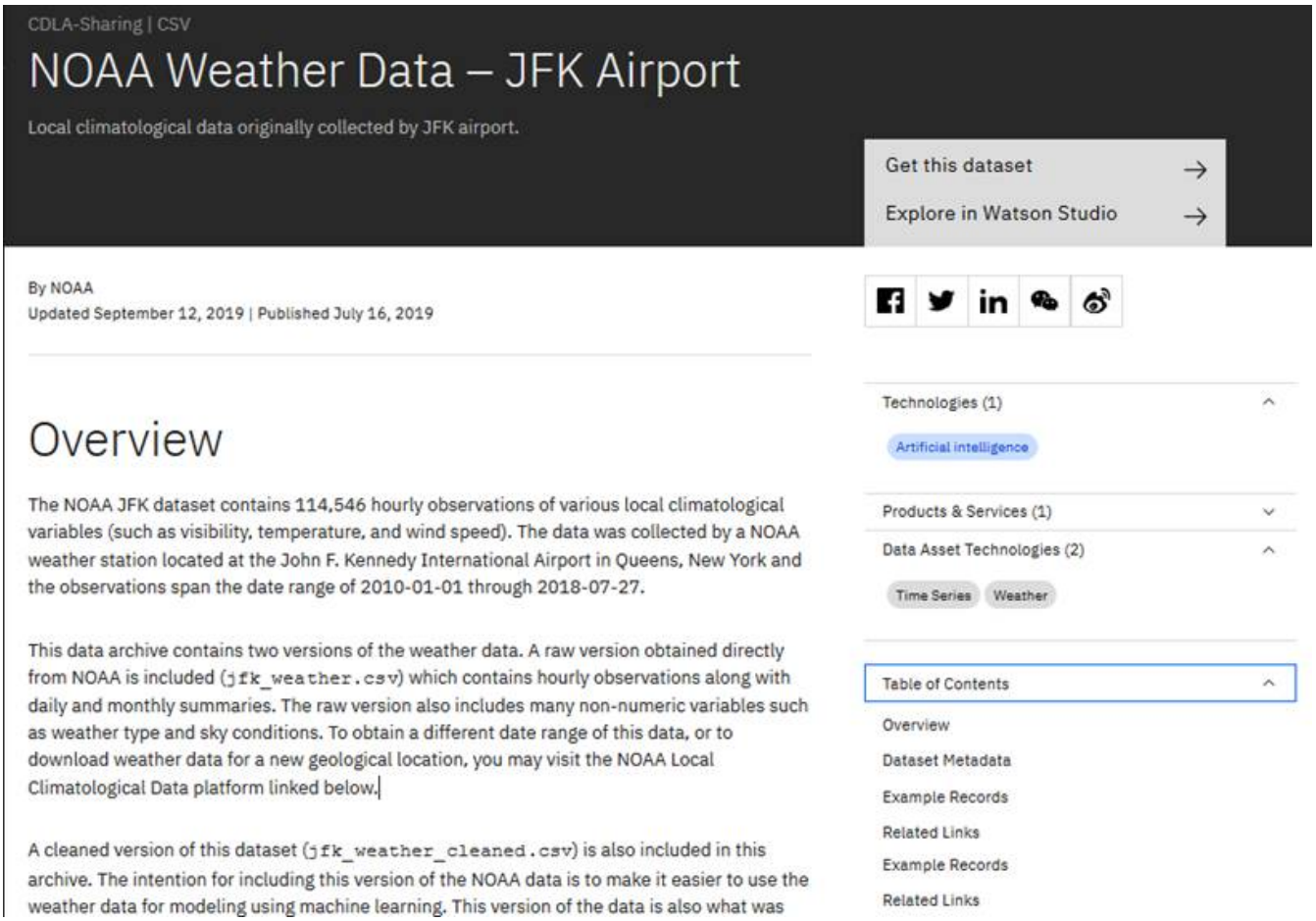
1. Open https://developer.ibm.com/ in your web browser.

2. From the main menu select **Open Source at IBM** > **Data Asset eXchange**. The DAX homepage is displayed.



The collection includes data sets from the Debater project (https://www.research.ibm.com/artificial-intelligence/project-debater/), data sets that can be used to train models to perform document layout analysis, natural language processing, time series analysis and more.

3. Open the NOAA Weather Data dataset (https://developer.ibm.com/exchanges/data/all/jfk-weather-data/), which contains data from a weather station at the John F. Kennedy Airport in New York spanning 8 years. This data set was used to train the weather forecaster model on MAX (https://developer.ibm.com/exchanges/models/all/max-weather-forecaster/).

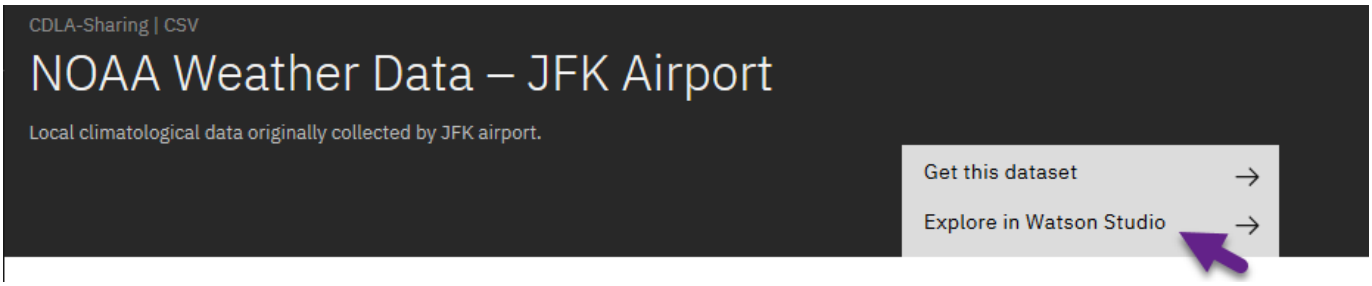


You can download the data set using the **Get this dataset** link. Data sets are stored as compressed archives, which you can extract using any utility that supports the tar.gz format. If you are not familiar with this file format take a look at this short open source tutorial https://opensource.com/article/17/7/how-unzip-targz-file.

4. Inspect the data set's metadata.

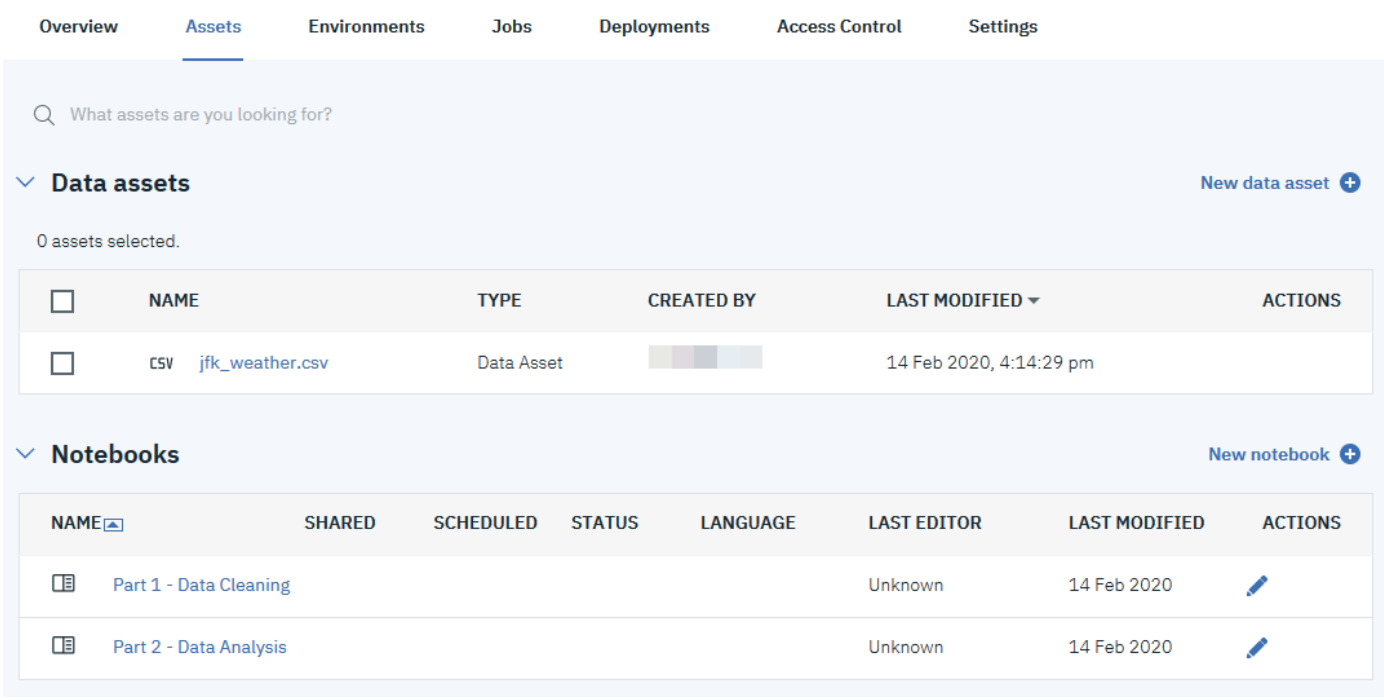
- This data set is stored as tabular data and formatted as a comma separated value (CSV) file, which is a very popular basic data exchange format.
- The data set was published under the data science friendly CDLA-Sharing license (https://cdla.io/).
- The data set contains time-series data and can be used to predict weather trends.

5. Most data sets are complemented by Python notebooks that you can use to explore, pre-process, and analyze the data. You can access the notebook (or notebooks) by clicking the **Explore in Watson Studio** or **Try the notebook** link:



The notebooks are hosted on Watson Studio, IBM's Data Science platform. Later in this course you'll learn more about Watson Studio, notebooks and how to run them.

6. [Optional] If you are already familiar with notebooks and Watson Studio feel free to open the link and import the project or notebook. The example below depicts the weather data set project assets, which include the raw data file and two notebooks.



This concludes part 2 of this lab, which introduced the Data Asset Exchange.