Standardizing deep learning model deployment with MAX

Nick Pentreath
Principal Engineer

@MLnick

About

– *@MLnick* on Twitter & Github

– Principal Engineer, IBM CODAIT (Center for Open-Source Data & AI Technologies)

– Machine Learning & AI

– Apache Spark committer & PMC

– Author of *Machine Learning with Spark*

– Various conferences & meetups

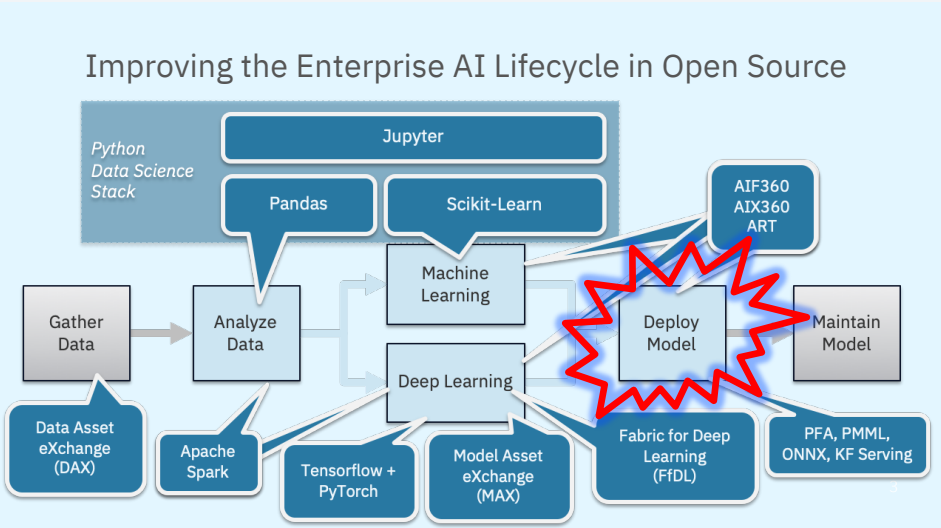
Center for Open Source Data & AI Technologies

CODAIT aims to make AI solutions dramatically easier to create, deploy, and manage in the enterprise.

We contribute to and advocate for the open-source technologies that are foundational to IBM's AI offerings.

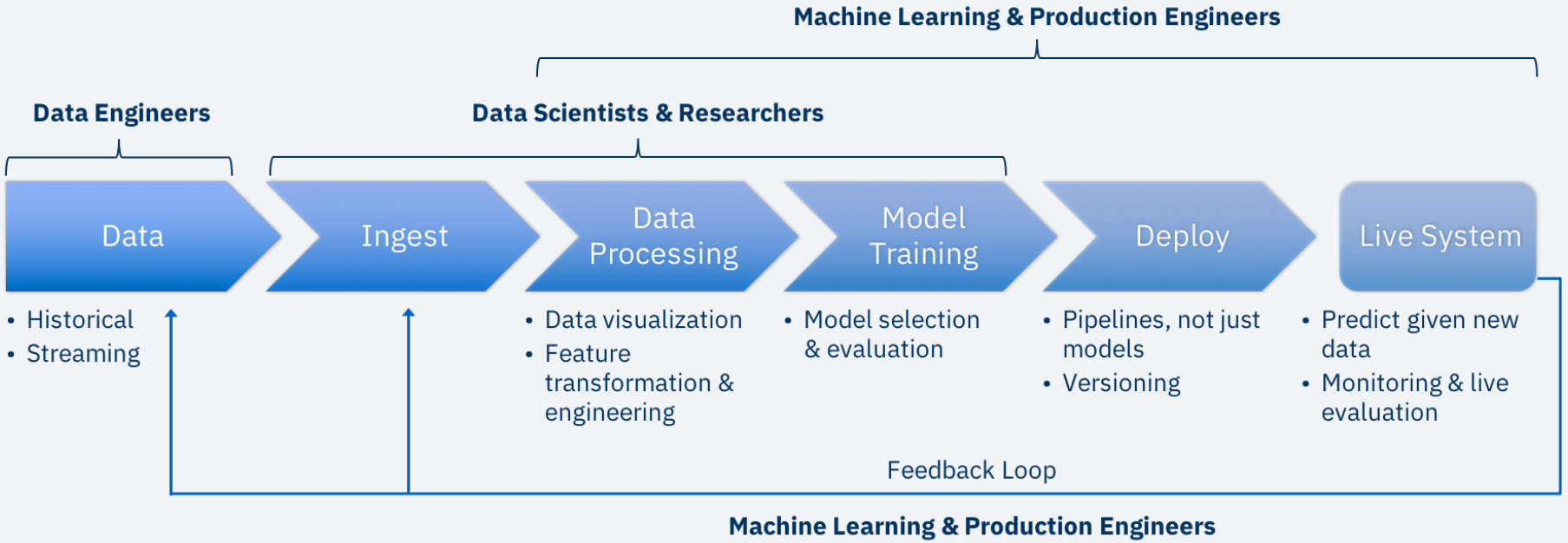
30+ open-source developers!

CODAIT

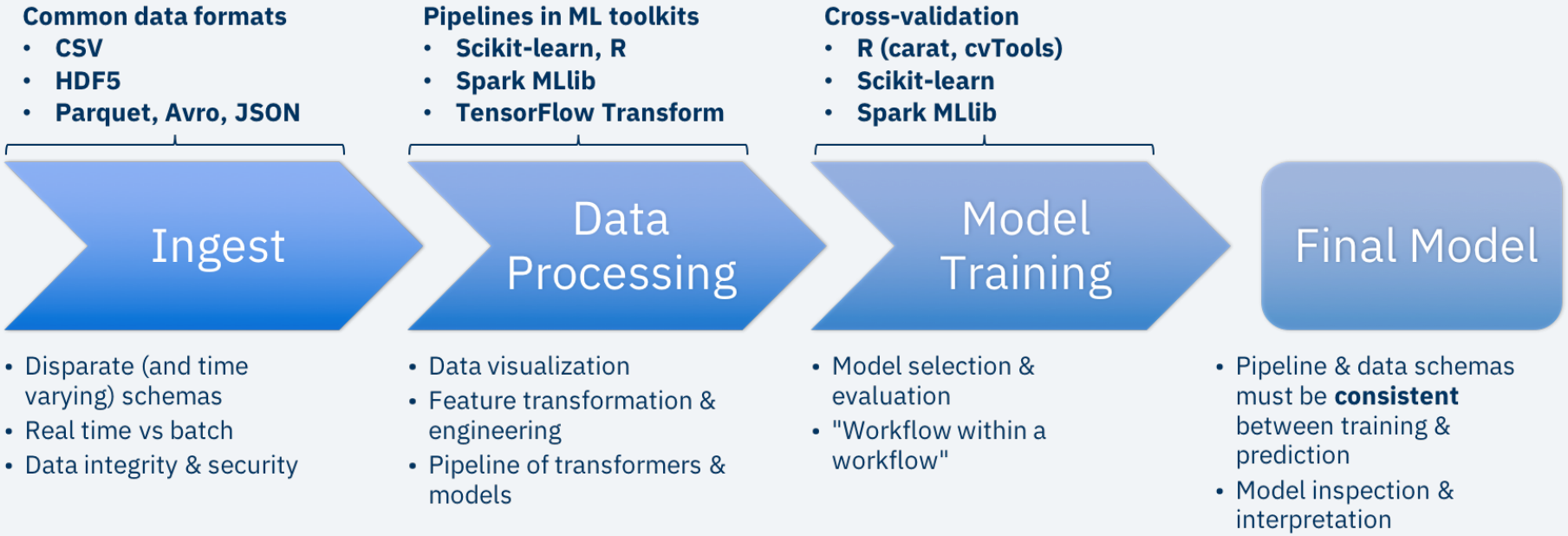


The Machine Learning Workflow

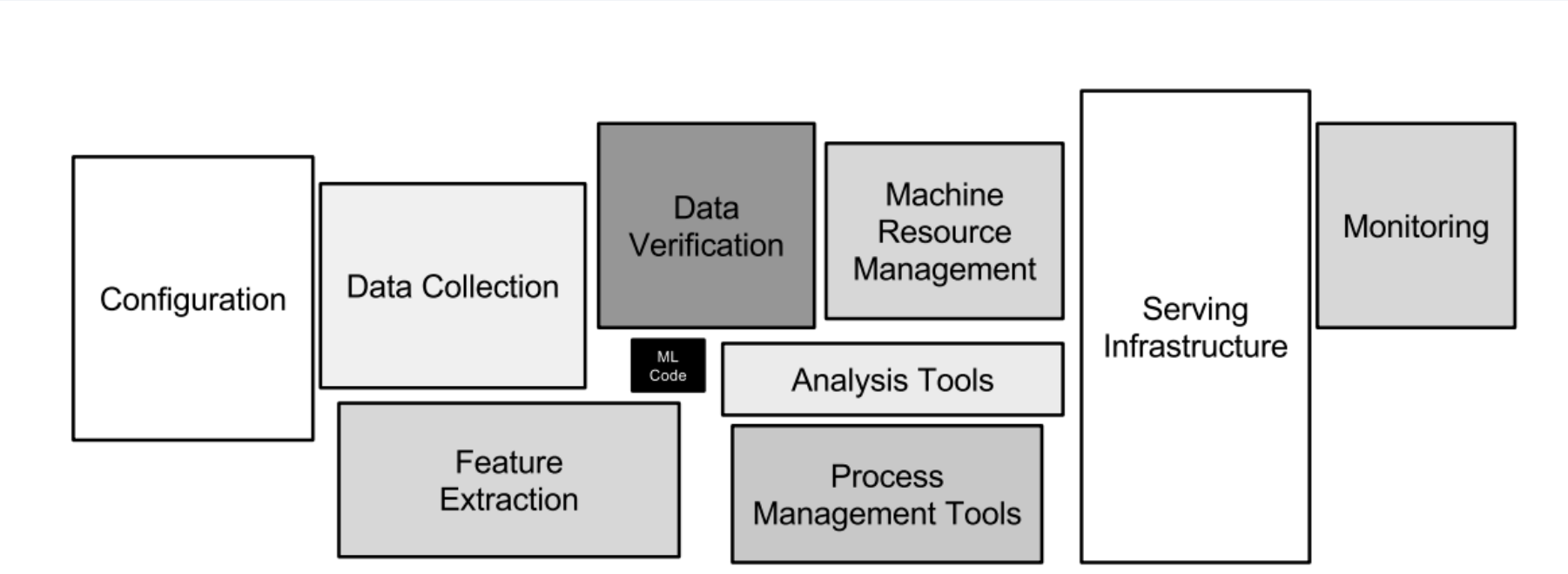
The workflow spans teams …



… and tools …


Common data formats
• CSV
• HDF5
• Parquet, Avro, JSON
Pipelines in ML toolkits
• Scikit-learn, R
• Spark MLlib
• TensorFlow Transform
Cross-validation
• R (carat, cvTools)
• Scikit-learn
• Spark MLlib
Ingest
Data Processing
Model Training
Final Model
• Disparate (and time varying) schemas
• Real time vs batch
• Data integrity & security
• Data visualization
• Feature transformation & engineering
• Pipeline of transformers & models
• Model selection & evaluation
• "Workflow within a workflow"
• Pipeline & data schemas must be consistent between training & prediction
• Model inspection & interpretation

… and is a small (but critical!) piece of the puzzle



*Source: Hidden Technical Debt in Machine Learning Systems

Machine Learning Deployment

What, Where, How?

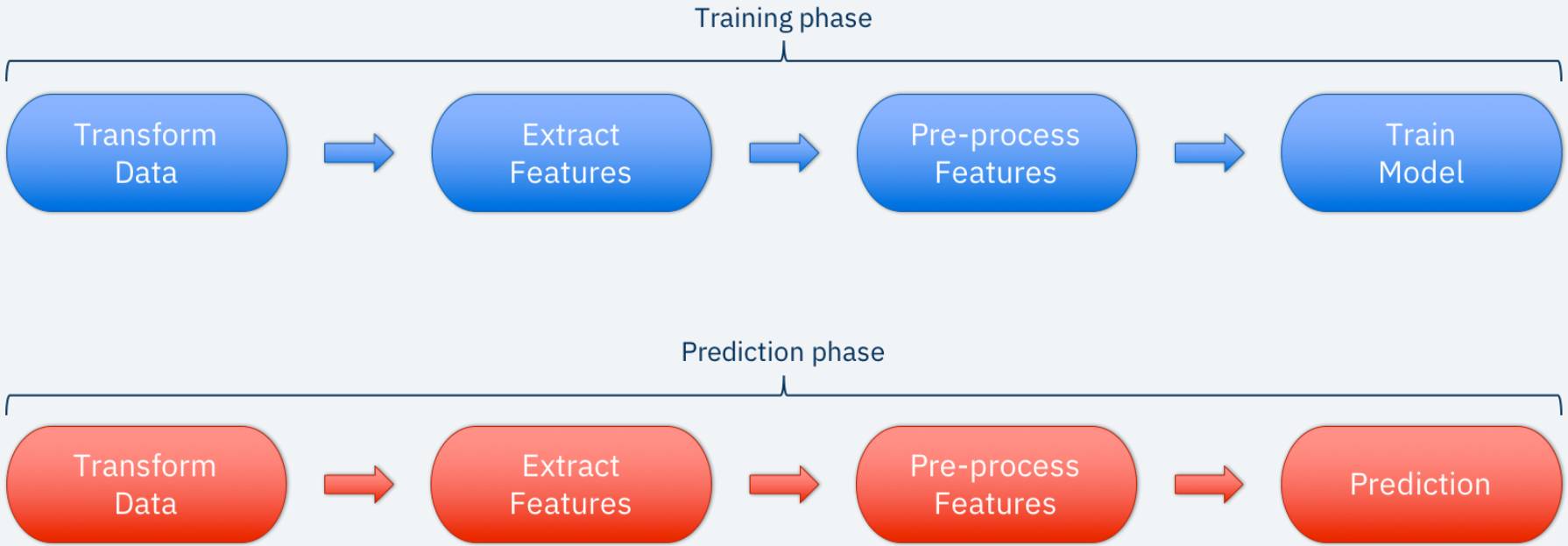
– What are you deploying?
  - What is a “model”

– Where are you deploying?
  - Target environment (cloud, browser, edge)
  - Batch, streaming, real-time?

– How are you deploying?
  - “devops” deployment mechanism
  - Serving framework

We will talk about the what & how

What is a “model”?



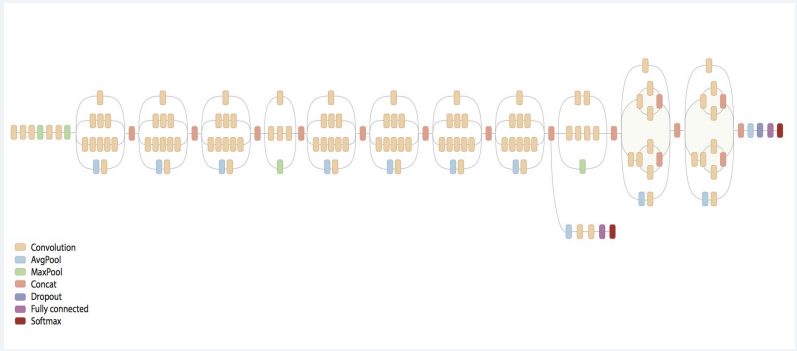
Deep Learning doesn't need feature engineering or data processing …

right?

Deep learning pipeline?

Input image

Inference

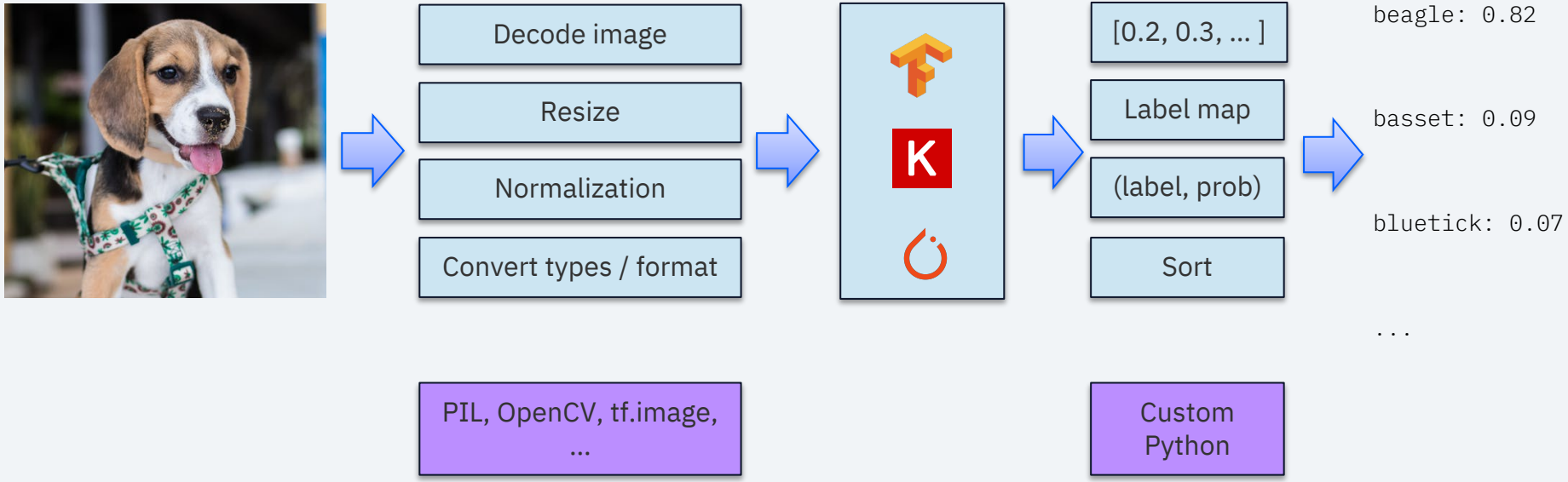


Prediction

```
beagle: 0.82
```


Source: https://ai.googleblog.com/2016/03/train-your-own-image-classifier-with.html

Deep learning pipeline!



** Logos trademarks of their respective projects*

Pipelines, not Models

- Deploying just the model part of the workflow is not enough
- Entire pipeline must be deployed
  - Data transforms
  - Feature extraction & pre-processing
  - DL / ML model
  - Prediction transformation
- Even ETL is part of the pipeline!

- Pipelines in frameworks
  - scikit-learn
  - Spark ML pipelines
  - TensorFlow Transform
  - pipeliner (R)

Many Challenges

– Need to manage and bridge many different:

- Languages - Python, R, Notebooks, Scala / Java / C
- Frameworks – too many to count!
- Dependencies
- Versions

– Friction between teams

- Data scientists & researchers – latest & greatest
- Production – stability, control, minimize changes, performance
- Business – metrics, business impact, product must always work!

– Formats

- Each framework does things differently
- Proprietary formats: lock-in, not portable

– Lack of standardization leads to custom solutions and extensions

** Logos trademarks of their respective projects*

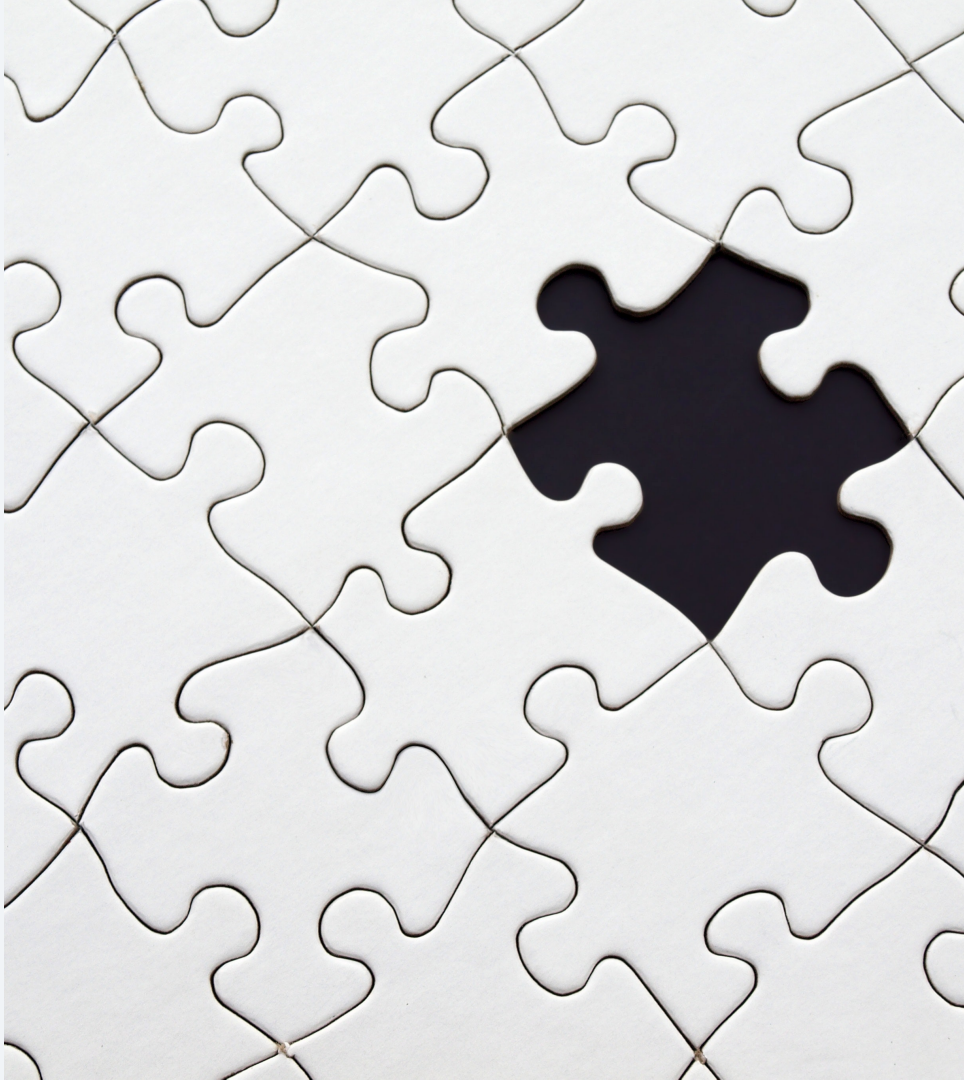
Containers for ML Deployment

Containers for ML Deployment

– Container-based deployment has significant benefits
  - Repeatability
  - Ease of configuration
  - Separation of concerns – focus on what, not how
  - Allow data scientists & researchers to use their language / framework of choice
  - Container frameworks take care of (certain) monitoring, fault tolerance, HA, etc.

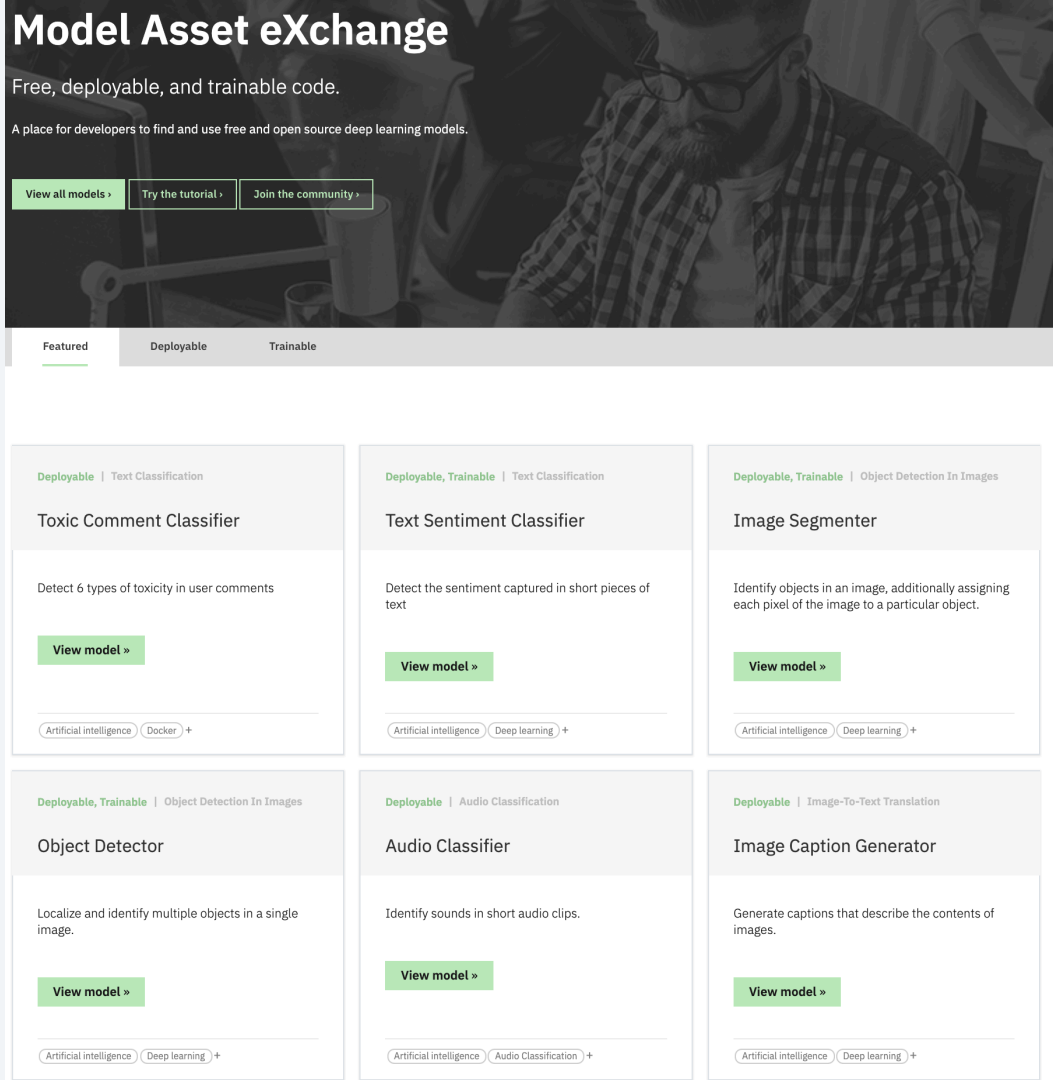
– But …
  - What goes in the container is still the most important factor
  - Performance can be highly variable across language, framework, version
  - Requires devops knowledge, CI / deployment pipelines, good practices
  - Does not solve the issue of standardization
    - Formats
    - APIs exposed
  - A serving framework is still required on top

The Model Asset Exchange



Model Asset eXchange (MAX)

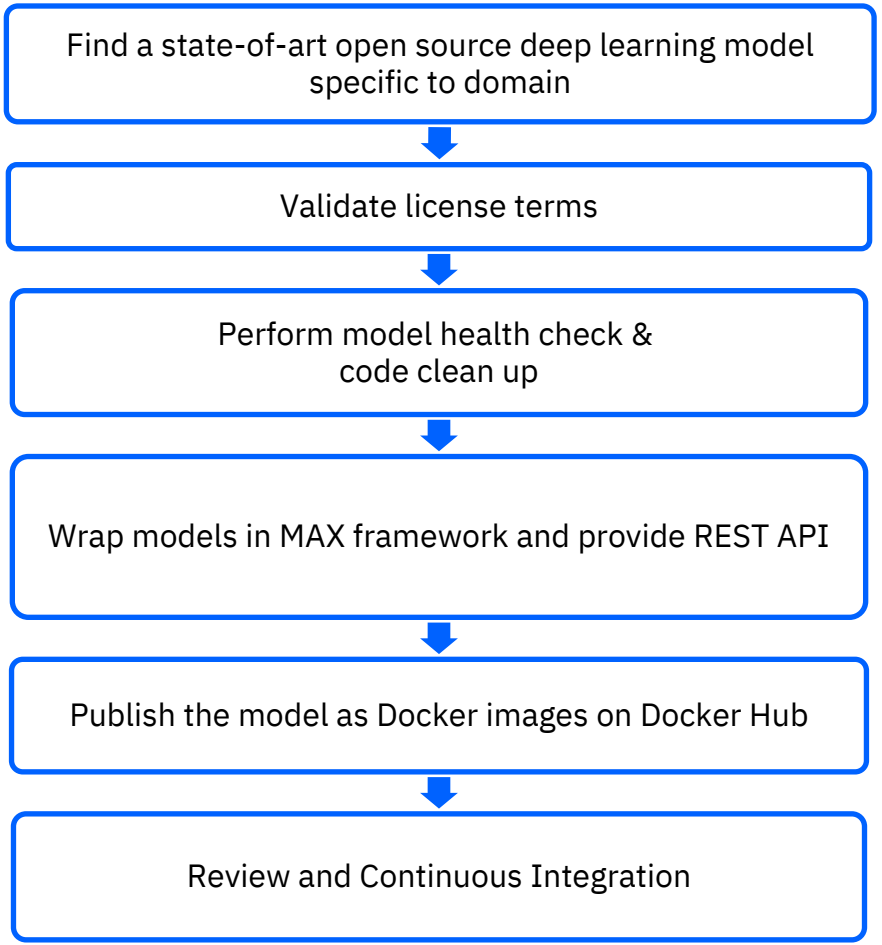
ibm.biz/model-exchange



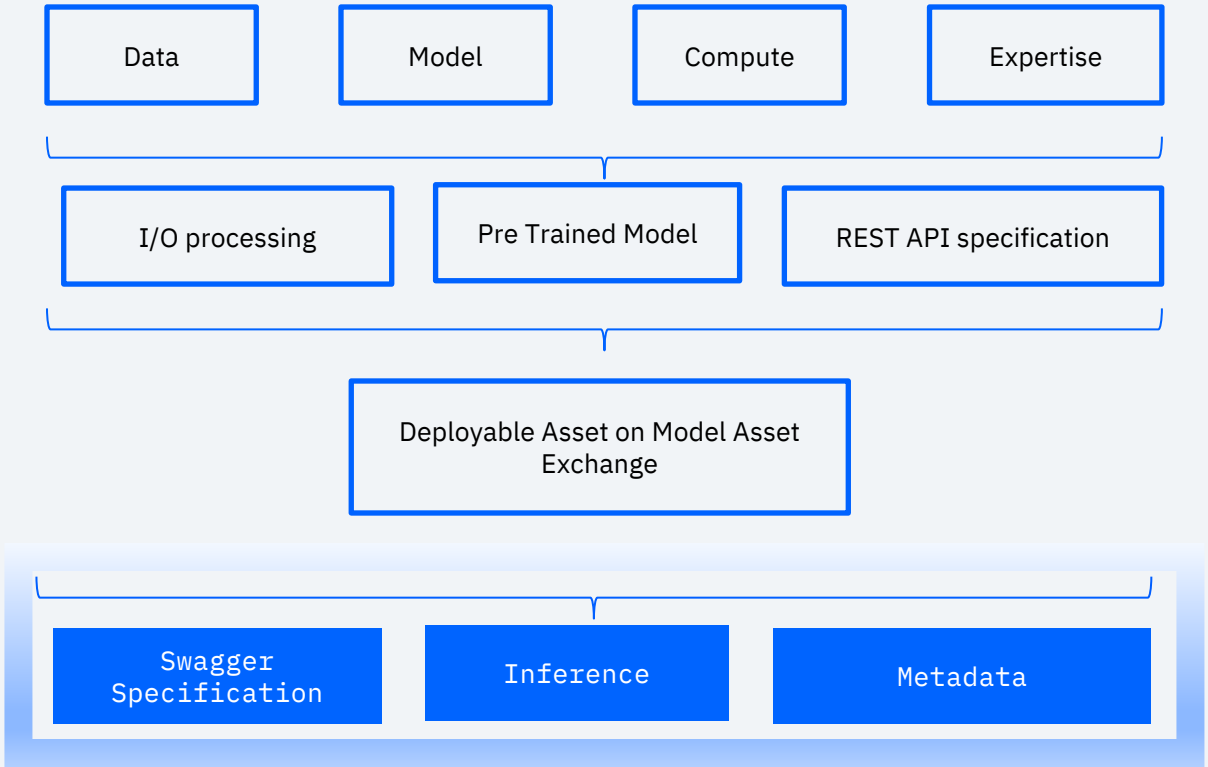
What is MAX?

- One place for **state-of-art** open source deep learning models
- Wide variety of domains
- Tested code and IP
- **Free and open** source
- Both trainable and trained versions

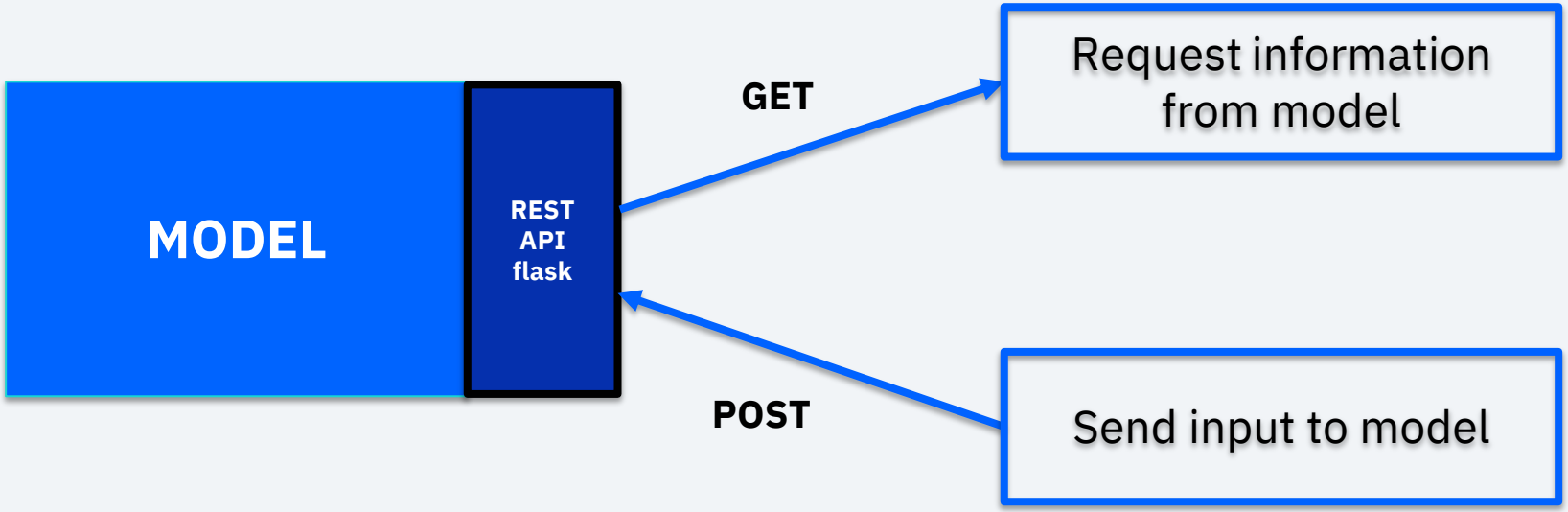
Behind the Scenes



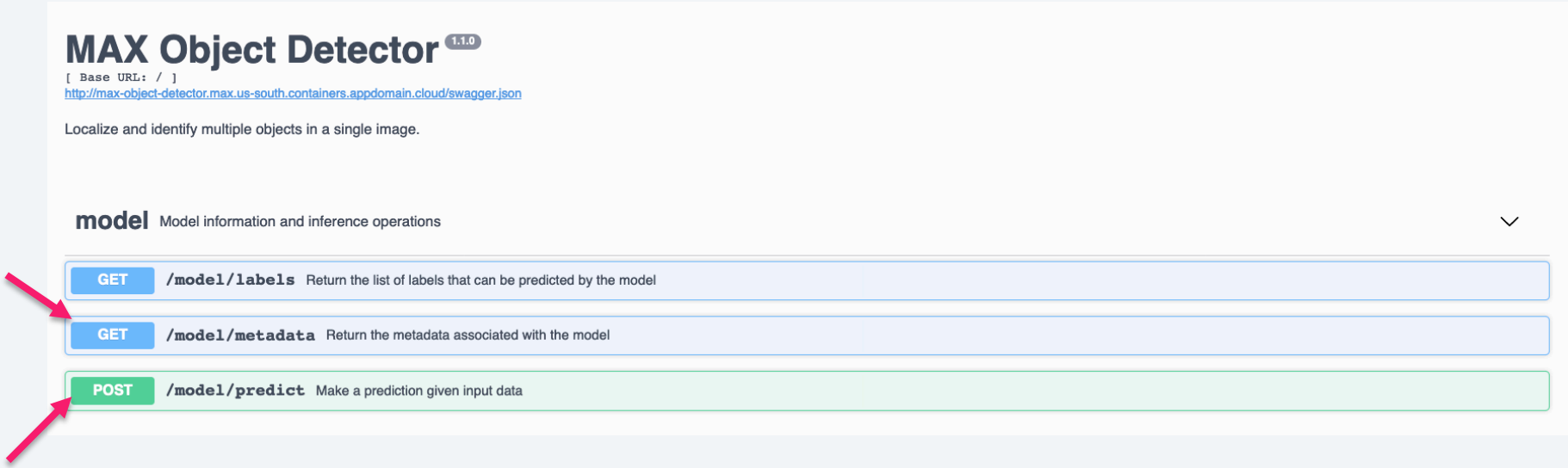
Deployable Asset on MAX



MAX Model Consumption – REST API



MAX Model Consumption – REST API



2. Python

```
# Model
url = 'http://max-object-detector.max.us-south.containers.appdomain.cloud/'
model_endpoint =  'model/predict'
complete_url = url + model_endpoint

# Upload an image to the MAX model's rest API
path_to_input_image = 'baby-bear.jpg'

with open(path_to_input_image, 'rb') as file:
    file_form = {'image': (path_to_input_image, file, 'image/jpeg')}
    # Post the image to the rest API using the requests library
    r = requests.post(url=complete_url, files=file_form)
    # Return the JSON
    response = r.json()

IPython.display.Image(path_to_input_image, width = 450)
```

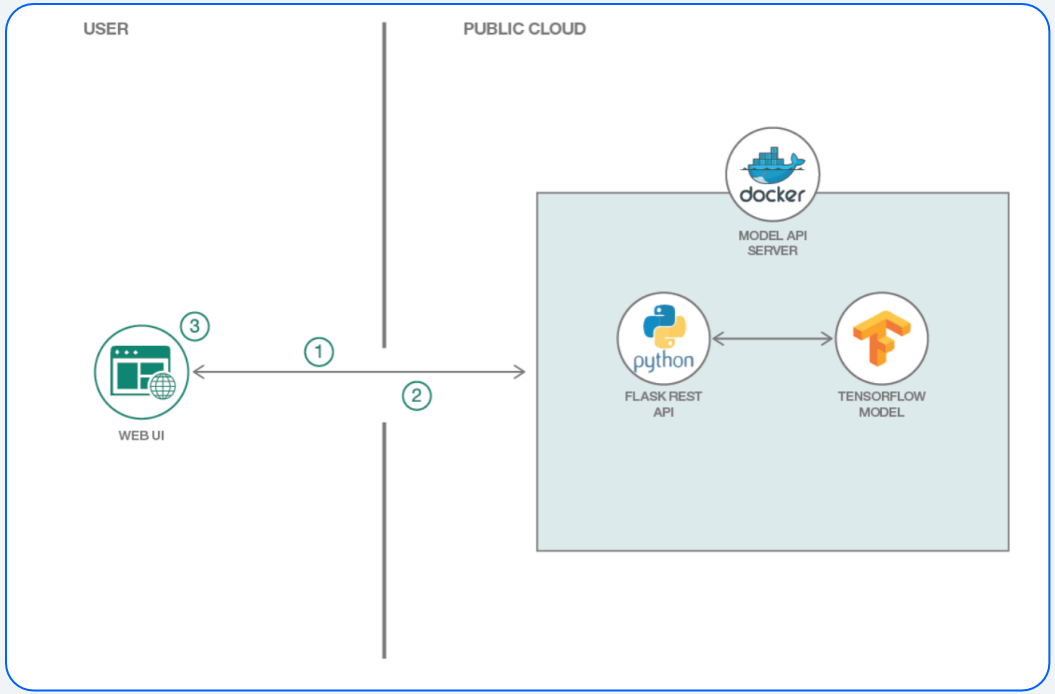
ibm.biz/max-notebook

2. Python


```
{'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': [0.24251867830753326,
    0.26926860213279724,
    0.655893087387085,
    0.5768759250640869]}]}
```

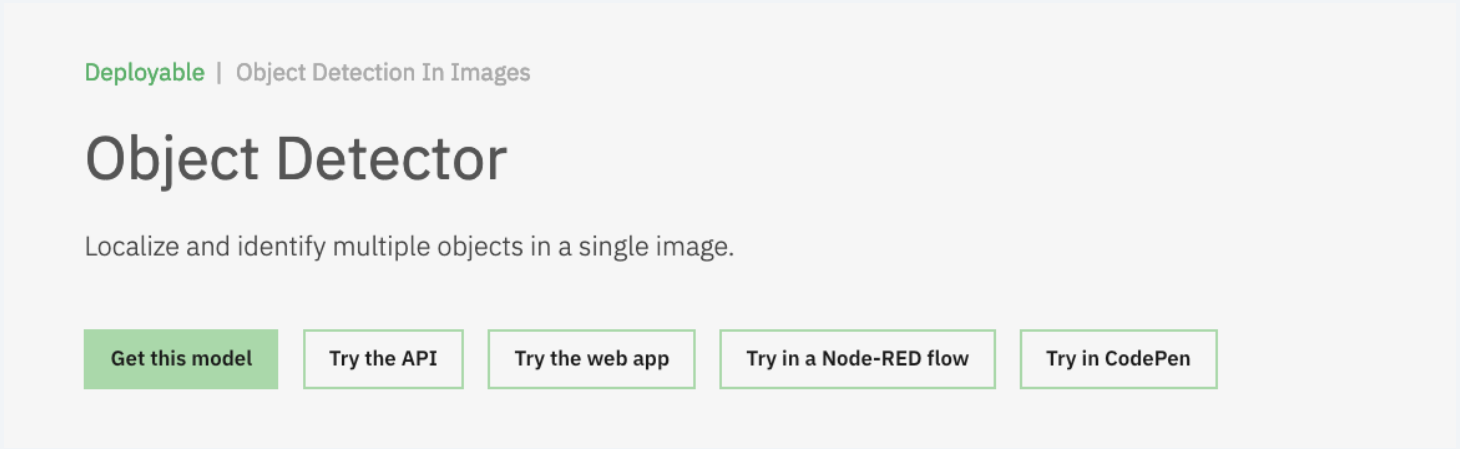

ibm.biz/max-notebook

3. Web App



- User uses Web UI to send an image to Model API.
- Model API returns object data and Web UI displays detected objects.

... and a few others



MAX-Framework

- A pip installable python library.
- Wrapper around Flask
- Abstracts out all basic functionality of the MAX model into `MAXApp` and `MAXApi` abstract classes.

github.com/IBM/MAX-Framework

MAX-Skeleton

- Template repository to create a deployable MAX model.
- Contains all the code scaffolding and imports MAX Framework.

github.com/IBM/MAX-Skeleton

MAX Framework

```
class ModelWrapper(MAXModelWrapper):

    MODEL_META_DATA = {
        'id': 'ID',
        'name': 'MODEL NAME',
        'description': 'DESCRIPTION',
        'type': 'MODEL TYPE',
        'source': 'MODEL SOURCE'
        'license': 'LICENSE'
    }

    def __init__(self, path=DEFAULT_MODEL_PATH):
        pass

    def _pre_process(self, inp):
        return inp

    def _post_process(self, result):
        return result

    def _predict(self, x):
        return x
```

MAX Framework

```python
class ModelPredictAPI(PredictAPI):

    model_wrapper = ModelWrapper()

    @MAX_API.doc('predict')
    @MAX_API.expect(input_parser)
    @MAX_API.marshal_with(predict_response)
    def post(self):
        """Make a prediction given input data"""
        result = {'status': 'error'}

        args = input_parser.parse_args()
        input_data = args['file'].read()
        preds = self.model_wrapper.predict(input_data)

        # Modify this code if the schema is changed
        label_preds = [{'label_id': p[0], 'label': p[1], 'probability': p[2]} for p in [x for x in preds]]
        result['predictions'] = label_preds
        result['status'] = 'ok'

        return result
```

MAX Framework

```
class ImageProcessor(object):
    """Composes several transforms together.

    Args:
        transforms (list of ``Transform`` objects): sequence of transforms to compose.

    Example:
        >>> pipeline = ImageProcessor([
        >>>     Rotate(150),
        >>>     Resize([100,100])
        >>> ])
        >>> pipeline.apply_transforms(img)
    """

    def __init__(self, transforms=[]):
        assert isinstance(transforms, Sequence)
        self.transforms = transforms

    def apply_transforms(self, img):
```

Requirements for Wrapping a Model

- Docker
- Python IDE or code editors
- Pre-trained model weights stored in a downloadable location
- List of required python packages
- Input pre-processing code
- Prediction/Inference code
- Output post-processing code

Steps to wrap your own model

Use MAX-Skeleton	Update Dockerfile	Requirements update	Update metadata	Update scripts	Test the model
• Login to GitHub • Click on `Use this template` • Clone the new repository	• Update model file location • Calculate hash value for model files and update md5sum.txt	• Add required python packages along with version number	• Update REST API metadata • Update model related metadata	• Add pre-processing, prediction and post-processing code.	• Build the docker image • Run the model server

Thank you!

codait.org

twitter.com/Mlnick & twitter.com/ibmcodait

github.com/MLnick

developer.ibm.com

- MAX on IBM Developer
https://ibm.biz/model-exchange

- DAX on IBM Developer
https://ibm.biz/data-exchange

- Learning path
https://ibm.biz/max-learning-path

- Ecosystem status
https://ibm.biz/max-status

- MAX Framework
https://ibm.biz/max-framework

- MAX Node Red
https://ibm.biz/max-node-red