DATA+AI SUMMIT 2021

FORMERLY SPARK+AI SUMMIT

mlflow Model Serving

27 May 2021

Andre Mesarovic

Sr. Resident Solutions Architect at Databricks

Agenda

- MLflow model serving
- How to score models with MLflow
    - Offline scoring with Spark
    - Online scoring with MLflow model server
- Custom model deployment and scoring
- Databricks model server

Prediction overview

Offline Prediction

- High throughput
- Bulk predictions
- Predict with Spark/Databricks
- Batch Prediction
- Structured Streaming Prediction

Online Prediction

- Low latency
- Individual predictions
- Real-time model serving
- MLflow scoring server
- MLflow deployment plugin
- Serving outside MLflow
- Databricks model serving

Vocabulary - Synonyms

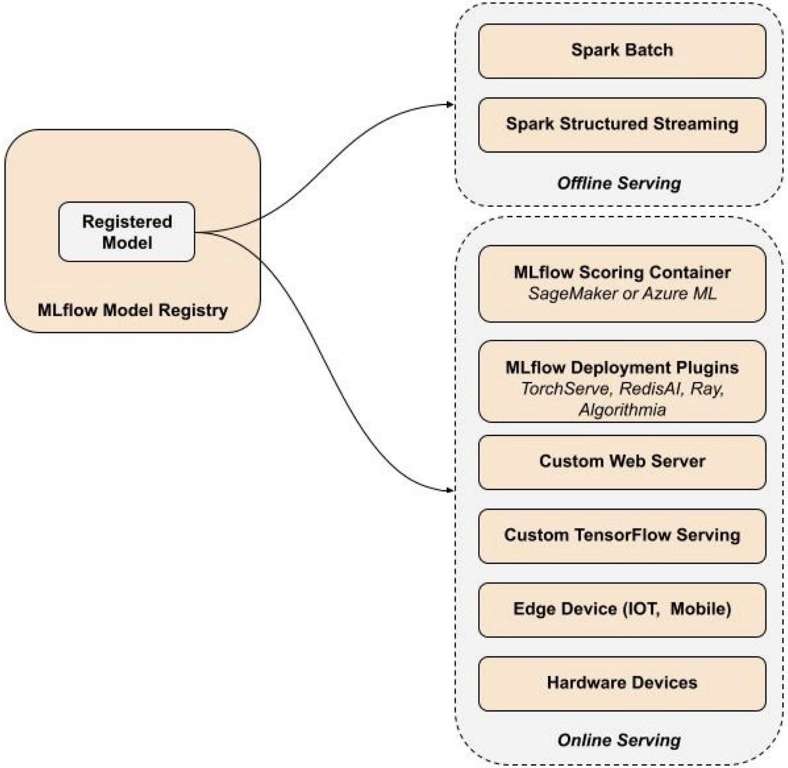
Model Serving

- Model serving
- Model scoring
- Model prediction
- Model inference
- Model deployment

Prediction Types

- Offline == batch prediction
  - Spark batch prediction
  - Spark streaming prediction
- Online == real-time prediction

MLflow Model Serving Ecosystem



Offline Scoring

- Spark is ideally suited for offline scoring
- Distributed scoring with Spark
- Can use either Spark batch or structured streaming
- Use MLflow UDF (Python or SQL) to parallelize scoring for single-node frameworks such as scikit-learn
- Create UDF model using Pyfunc flavor
- Load model from MLflow Model Registry

MLflow Offline Scoring Example

Model URI

```
model_uri = "models:/sklearn-wine/production"
```

Score with native flavor

```
model = mlflow.sklearn.load_model(model_uri)
predictions = model.predict(data)
```

Score with Pyfunc flavor

```
model = mlflow.pyfunc.load_model(model_uri)
predictions = model.predict(data)
```

Score with Python UDF

```
udf = mlflow.pyfunc.spark_udf(spark, model_uri)
predictions = data.withColumn("prediction", udf(*data.columns))
```

Score with SQL UDF

```
udf = mlflow.pyfunc.spark_udf(spark, model_uri)
spark.udf.register("predict", udf)
%sql select *, predict(*) as prediction from my_data
```

Online Scoring

- Score models outside of Spark
- Low-latency scoring one record at a time with immediate feedback
- Variety of real-time deployment options:
  - REST API - web servers - self-managed or as containers in cloud providers
  - Embedded in browsers, .e.g TensorFlow Lite
  - Edge devices: mobile phones, sensors, routers, etc.
  - Hardware accelerators - GPU, NVIDIA, Intel

Options to serve real-time models

- Embed model in application code
- Model deployed as a service
- Model published as data
- Martin Fowler's Continuous Delivery for Machine Learning

MLflow Model Server

- Cornerstone of different MLflow deployment targets
- Web server that exposes a standard REST API:
  - Input: CSV, JSON (pandas-split or pandas-records formats) or Tensor (JSON)
  - Output: JSON list of predictions
- Implementation: Python Flask or Java Jetty server (only MLeap)
- MLflow CLI builds a server with embedded model

MLflow Model Server deployment options

- Local web server
- SageMaker docker container
- Azure ML docker container
- Generic docker container
- Custom deployment plugin targets
  - TorchServe, RedisAi, Ray or Algorithmia

MLflow Model Server container types

Python Server

Model and its ML framework libraries embedded in Flask server. Only for non-Spark ML models.

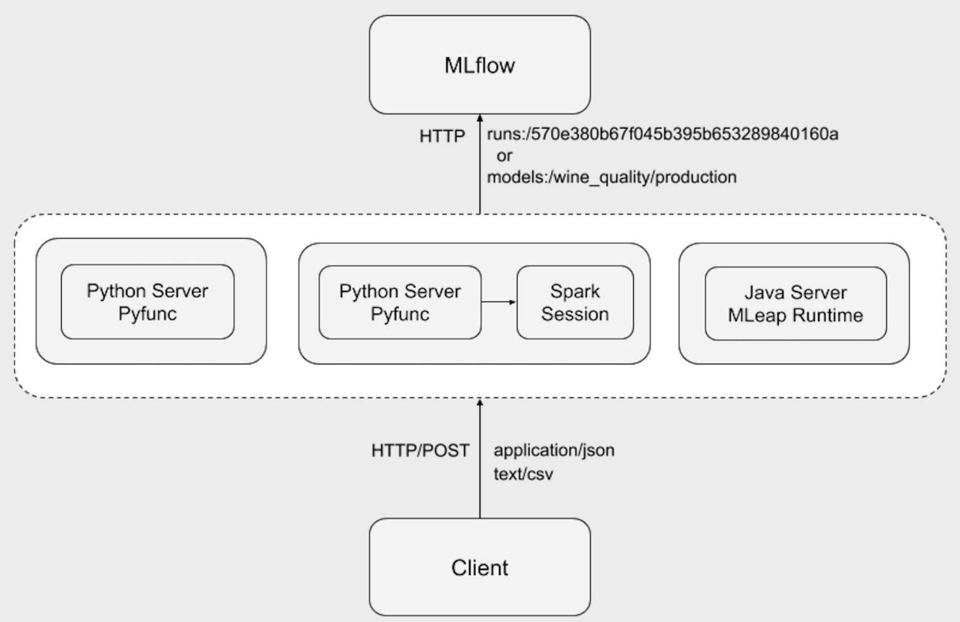
Python Server + Spark

Flask server delegates scoring to Spark server. Only for Spark ML models.

Java MLeap Server

Model and MLeap runtime are embedded in Jetty server. Only for Spark ML models. No Spark runtime.

MLflow Model Server container types



Launch Local MLflow Model Server

Launch local server

```
mlflow models serve --model-uri models:/sklearn-iris/production --port 5001
```

Score with CSV format

Request

```
curl  http://localhost:5001/invocations  \
  -H "Content-Type:text/csv" \
  -d '
    sepal_length,sepal_width,petal_length,petal_width
    5.1,3.5,1.4,0.2
    4.9,3.0,1.4,0.2'
```

Response

```
[0, 0]
```

Score with JSON split-oriented format

Request

```
curl  http://localhost:5001/invocations  \
  -H "Content-Type:application/json" \
  -d ' {
    "columns": ["sepal_length", "sepal_width", "petal_length", "petal_width"],
    "data": [
      [ 5.1, 3.5, 1.4, 0.2 ],
      [ 4.9, 3.0, 1.4, 0.2 ] ] }'
```

Response

```
[0, 0]
```

Score with JSON record-oriented format

Request

```
curl  http://localhost:5001/invocations  \
  -H "Content-Type:application/json; format=pandas-records" \
  -d '[
    { "sepal_length": 5.1, "sepal_width": 3.5, "petal_length": 1.4, "petal_width": 0.2 },
    { "sepal_length": 4.9, "sepal_width": 3.0, "petal_length": 1.4, "petal_width": 0.2 } ]'
```

Response

```
[0, 0]
```

Score with JSON Tensor

- Tensor Input Now Supported in MLflow - Databricks blog - 2021-03-18
- Request will be cast to Numpy arrays. This format is specified using a Content-Type request header value of application/json and the instances or inputs key in the request body dictionary
- Content-type is application/json - 2 request formats will be inferred
- See Deploy MLflow models - MLflow doc
- TFX Serving request format - TensorFlow doc

Score with JSON Tensor - numpy/tensor

TensorFlow serving "instances" format

```
curl http://127.0.0.1:5000/invocations -H 'Content-Type: application/json' -d '{
    "instances": [
        {"a": "s1", "b": 1, "c": [1, 2, 3]},
        {"a": "s2", "b": 2, "c": [4, 5, 6]},
        {"a": "s3", "b": 3, "c": [7, 8, 9]}
    ]
}'
```

TensorFlow serving "inputs" format

```
curl http://127.0.0.1:5000/invocations -H 'Content-Type: application/json' -d
'{
    "inputs": {
      "a": ["s1", "s2", "s3"],
      "b": [1, 2, 3]
}'
```

MLflow SageMaker and Azure ML containers

- Two types of containers are deployed to cloud providers
    - Python container - Flask web server process with embedded model
    - SparkML container - Flask web server process and Spark process
- SageMaker container
    - Most versatile container type
    - Can run in local mode on laptop as regular docker container

Python container

- Flask web server
- Flask server runs behind gunicorn to allow concurrent requests
- Serialized model (e.g. sklearn or Keras/TF) is embedded inside web server and wrapped by Pyfunc
- Server leverages Pyfunc flavor to make generic predictions

SparkML container

- Two processes:
  - Flask web server
  - Spark server - OSS Spark - no Databricks
- Web server delegates scoring to Spark server

MLflow SageMaker container deployment

- CLI:
  - mlflow sagemaker build-and-push-container
  - mlflow sagemaker deploy
  - mlflow sagemaker run-local
- API: mlflow.sagemaker API
- Deploy a model on Amazon SageMaker

Deploy MLflow SageMaker container

Launch container on SageMaker

```
mlflow sagemaker build-and-push-container --build --container my-container
mlflow sagemaker deploy --app-name wine-quality \
  --model-uri models:/wine-quality/production \
  --image-url my-image-url \
  --region us-west-2 --mode replace
```

Launch local container

```
mlflow sagemaker build-and-push-container --build --no-push \
  --container my-container
mlflow sagemaker run-local \
  --model-uri models:/wine-quality/production \
  --port 5051 --image my-container
```

MLflow AzureML container deployment

- Deploy to Azure Kubernetes Service (AKS) or Azure Container Instances (ACI)
- CLI - mlflow azureml build-image
- API - mlflow.azureml.build-image
- Deploy a python_function model on Microsoft Azure ML

Deploy MLflow Azure ML container

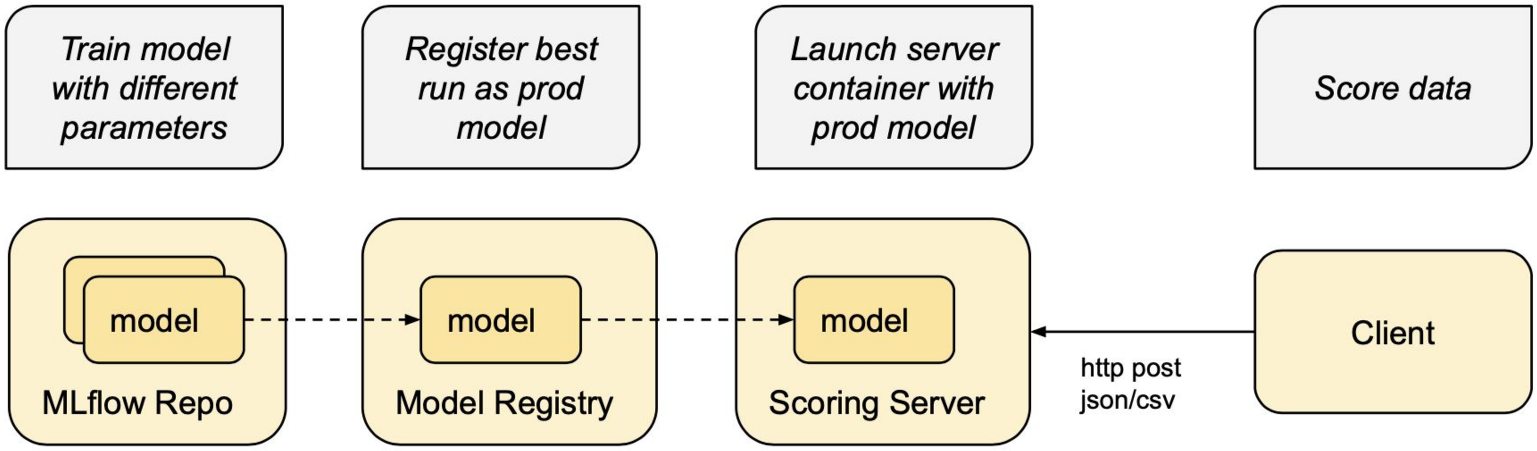
```
model_image, azure_model = mlflow.azureml.build_image(
    model_uri="models:/sklearn_wine",
    workspace="my_workspace",
    model_name="sklearn_wine",
    image_name="sklearn_wine_image",
    description="Sklearn Wine Quality",
    synchronous=True)
```

MLeap

- Non-Spark serialization format for Spark models
- Advantage is ability to score Spark model without overhead of Spark
- Faster than scoring Spark ML models with Spark
- Problem is stability, maturity and lack of dedicated commercial support

End-to-end ML Pipeline Example with MLflow

See [mlflow-examples - e2e-ml-pipeline]



MLflow Deployment Plugins

- MLflow Deployment Plugins - Deploy model to custom serving platform
- Current deployment plugins:
  - mlflow-torchserve
  - mlflow-redisai
  - mlflow-algorithmia
  - mlflow-ray-serve

MLflow Deployment Plugin Examples

TorchServe

```
mlflow deployments create -t torchserve ---name DEPLOYMENT_NAME -m <model uri> \
 -C 'MODEL_FILE=<model file path>' -C 'HANDLER=<handler file path>'
```

RedisAI

```
mlflow deployments create -t redisai --name <rediskey> -m <model uri> \
  -C <config option>
```

Ray

```
mlflow deployments create -t ray-serve --name <deployment name> -m <model uri> \
 -C num_replicas=<number of replicas>
```

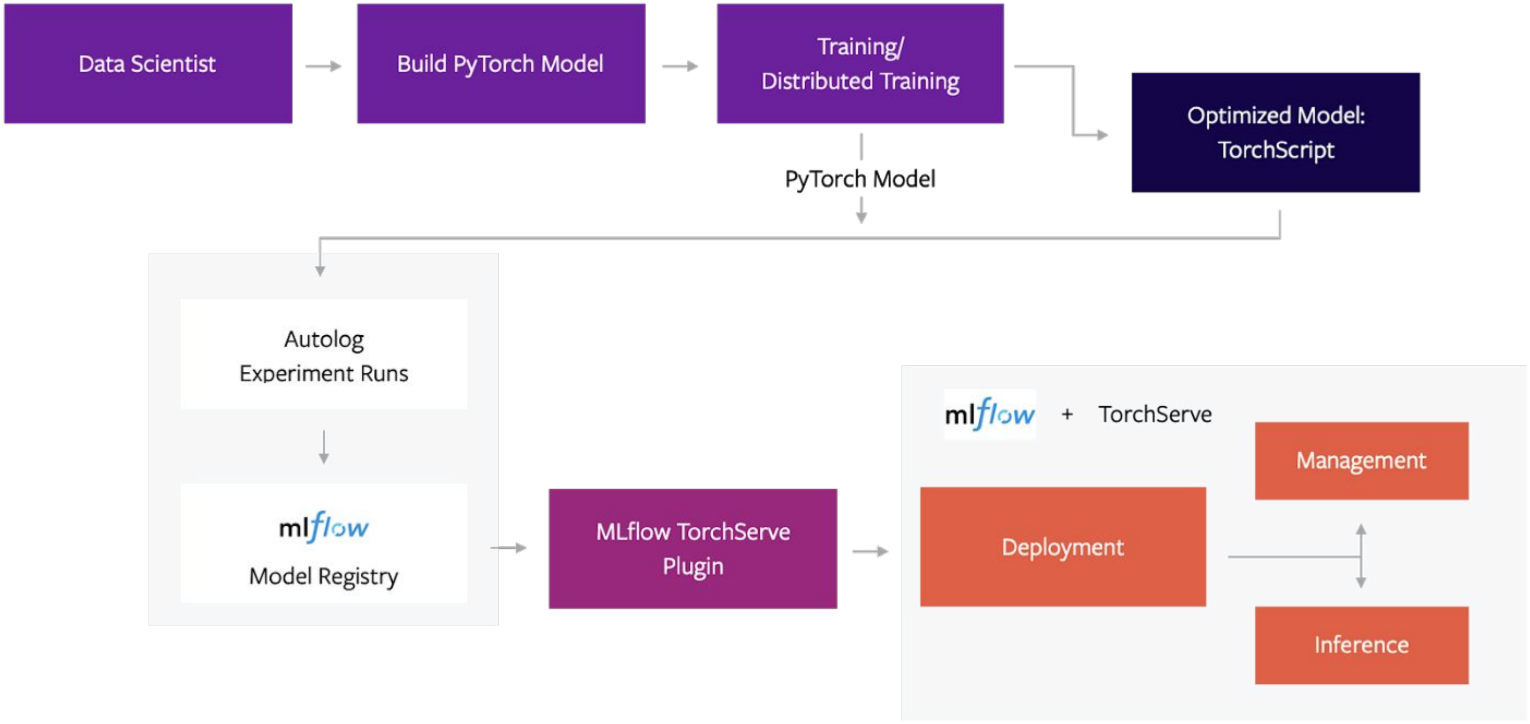
Algorithmia

```
mlflow deployments create -t algorithmia --name mlflow_sklearn_demo \
  -m mlruns/0/<run-id>/artifacts/model
```

MLflow + PyTorch = 🧡

- MLflow PyTorch integration released in MLflow 1.12.0 (Nov. 2020)
- Integrates with PyTorch Lightning, TorchScript and TorchServe
- Features:
  - Autologging with PyTorch Lightning
  - Translate to TorchScript model format for Python-free optimized scoring
  - Deploy MLflow models to TorchServe

MLflow + PyTorch = 🧡



MLflow TorchServe Resources

- PyTorch and MLflow Integration Announcement - 2020-11-12
- MLflow 1.12 Features Extended PyTorch Integration - 2012-11-13
- MLflow and PyTorch — Where Cutting Edge AI meets MLOps - medium - pytorch - 2020-11-12
- https://github.com/mlflow/mlflow-torchserve

MLflow RedisAI Resources

- https://github.com/RedisAI/mlflow-redisai
- All you need is PyTorch, MLflow, RedisAI and a cup of mocha latte - medium - 2020-08-28
- Taking Deep Learning to Production with MLflow & RedisAI – video - 30:16 - Sherin Thomas (RedisAI) - 2020-08-30
- PyPI mlflow-redisai

MLflow Ray Resources

- github.com/ray-project/mlflow-ray-serve
  - mlflow_example.py - test_mlflow_plugin.py
- Ray & MLflow: Taking Distributed Machine Learning Applications to Production blog
  - Databricks blog - 2021-02-03
  - Ray blog - 2021-01-13
- Using MLflow with Tune - Ray docs
- PyPI mlflow-ray-serve

MLflow Algorithmia Resources

- MLFlow Integrations - Development Center - Algorithmia
- https://github.com/algorithmiaio/mlflow-algorithmia
- Algorithmia and MLflow: Integrating open-source tooling with enterprise MLOps - Algorithmia blog - 2021-04-15
- pip install mlflow-algorithmia
- MLflow Integration from Azure ML and Algorithmia - video - 1:03:12 - DAIS-2021 -2021-04-22
- Algorithmia and MLflow: Integrating open-source tooling with enterprise MLOps - video - 11:19 - 2021-04-01

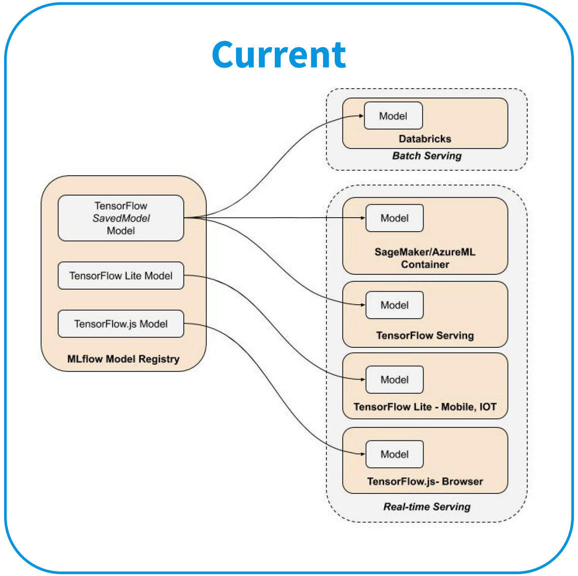
MLflow and TensorFlow Serving Custom Deployment

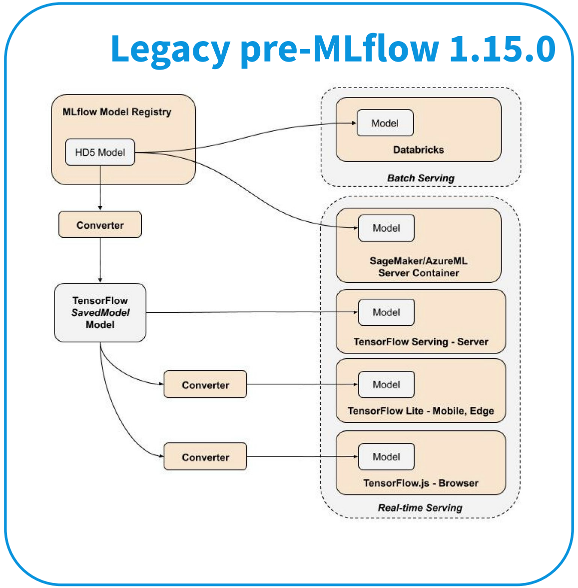
- Example how to deploy models to a custom deployment target
- MLflow deployment plugin has not yet been developed
- Deployment builds a standard TensorFlow Serving docker container with a MLflow model from Model Registry
- https://github.com/amesar/mlflow-tools/tree/master/mlflow_tools/tensorflow_serving

Keras/TensorFlow Model Formats

- SavedModel format
    - TensorFlow standard format is default in Keras TF 2.x
    - Supported by TensorFlow Serving
    - As of MLflow 1.15.0 SavedModel is the default MLflow format
- HD5 format
    - Default in Keras TF 1.x but is legacy in Keras TF 2.x

mlflow and TensorFlow Serving





TensorFlow Serving Example

Launch server in Docker container

```
docker run -t --rm --publish 8501 \
--volume
/opt/mlflow/mlruns/1/f48dafd70be044298f71488b0ae10df4/artifacts/tensorflow-model:/models/keras_wine\
--env MODEL_NAME=keras_wine \
tensorflow/serving
```

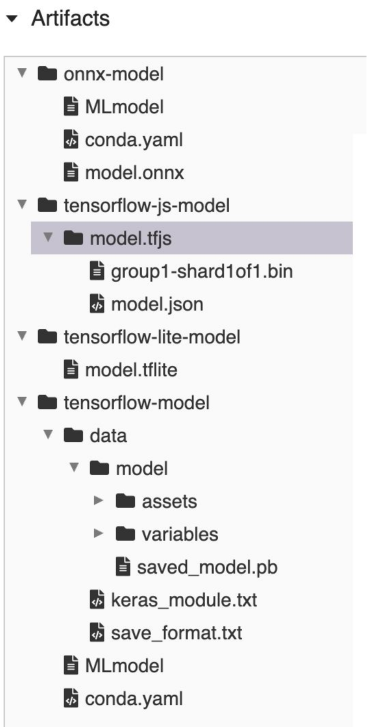
Request

```
curl -d '{"instances": [12.8, 0.03, 0.48, 0.98, 6.2, 29, 1.2, 0.4, 75 ] }' \
     -X POST http://localhost:8501/v1/models/keras_wine:predict
```

Response

```
 { "predictions": [[-0.70597136]] }
```

MLflow Keras/TensorFlow Run Models Example



MLflow Flavors - Managed Models

- Model flavors
  - tensorflow-model - standard SavedModel format (saved_model.pb)
  - onnx-model
- Can be served as Pyfunc models

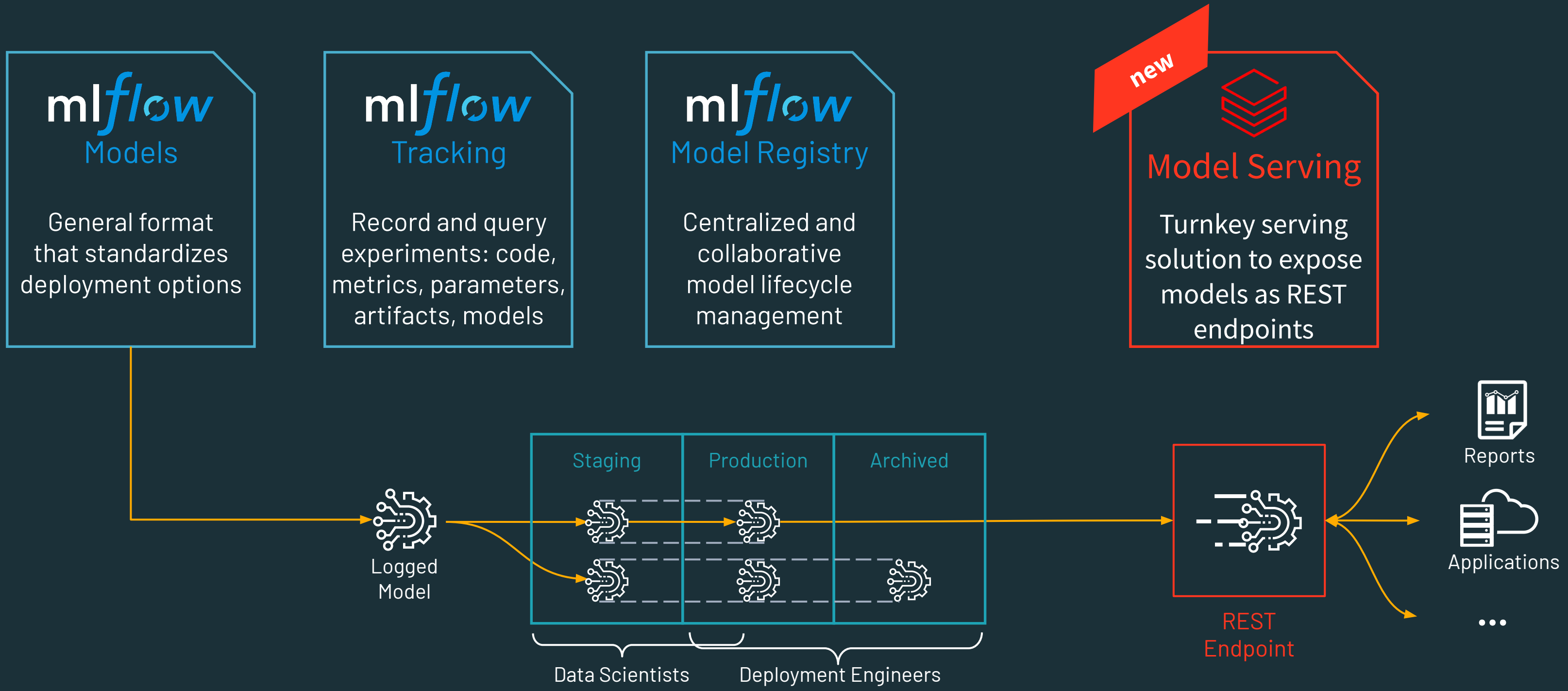
MLflow Unmanaged Models

- Models
  - tensorflow-model - Standard TF *SavedModel* format
  - tensorflow-lite-model
  - tensorflow.js model
- Load as raw artifacts and then serve manually
- Sample code: wine_train.py and wine_predict.py

ONNX - Open Neural Network Exchange

- Interoperable model format supported by:
  - MSFT, Facebook, AWS, Nvidia, Intel, etc.
- Train once, deploy anywhere (in theory) - depends on the robustness of ML framework conversion
- Save MLflow model as ONNX flavor
- Deploy ONNX model in standard MLflow scoring server
- ONNX runtime is embedded in MLflow Python model server

Model Serving on Databricks



databricks

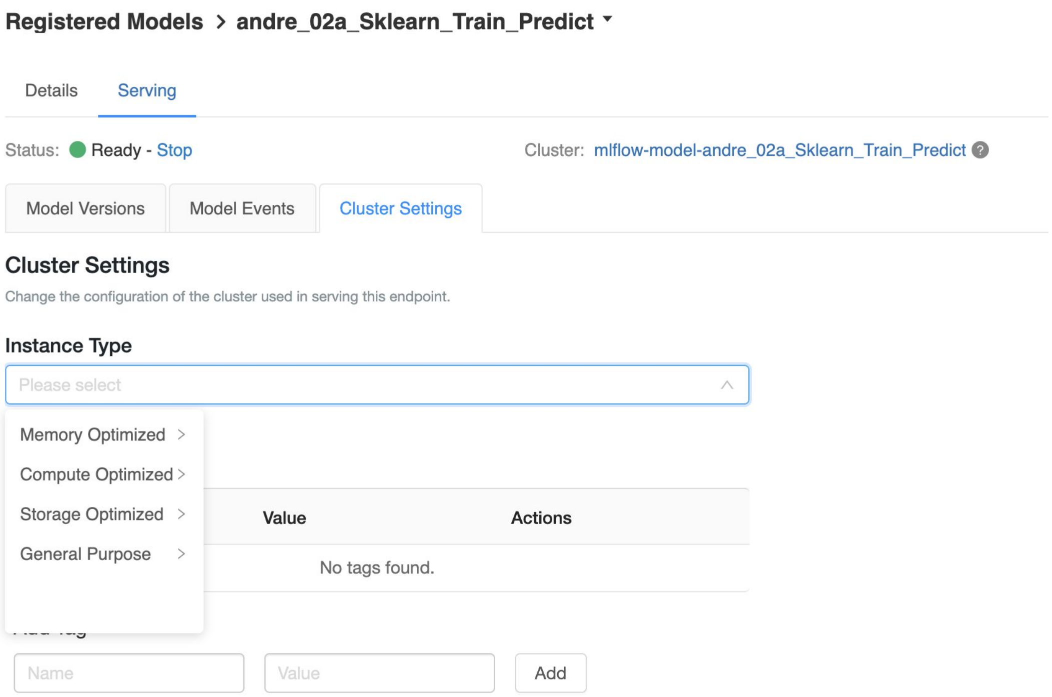
Current Databricks Model Serving - Public Preview

- HTTP endpoint publicly exposed using Databricks authentication
- One node cluster is automatically provisioned
- Can select instance type
- Limited production capability - for light loads and testing
- Hosts all active versions of a model
- Can score from UI or via HTTP (curl, Python Requests)

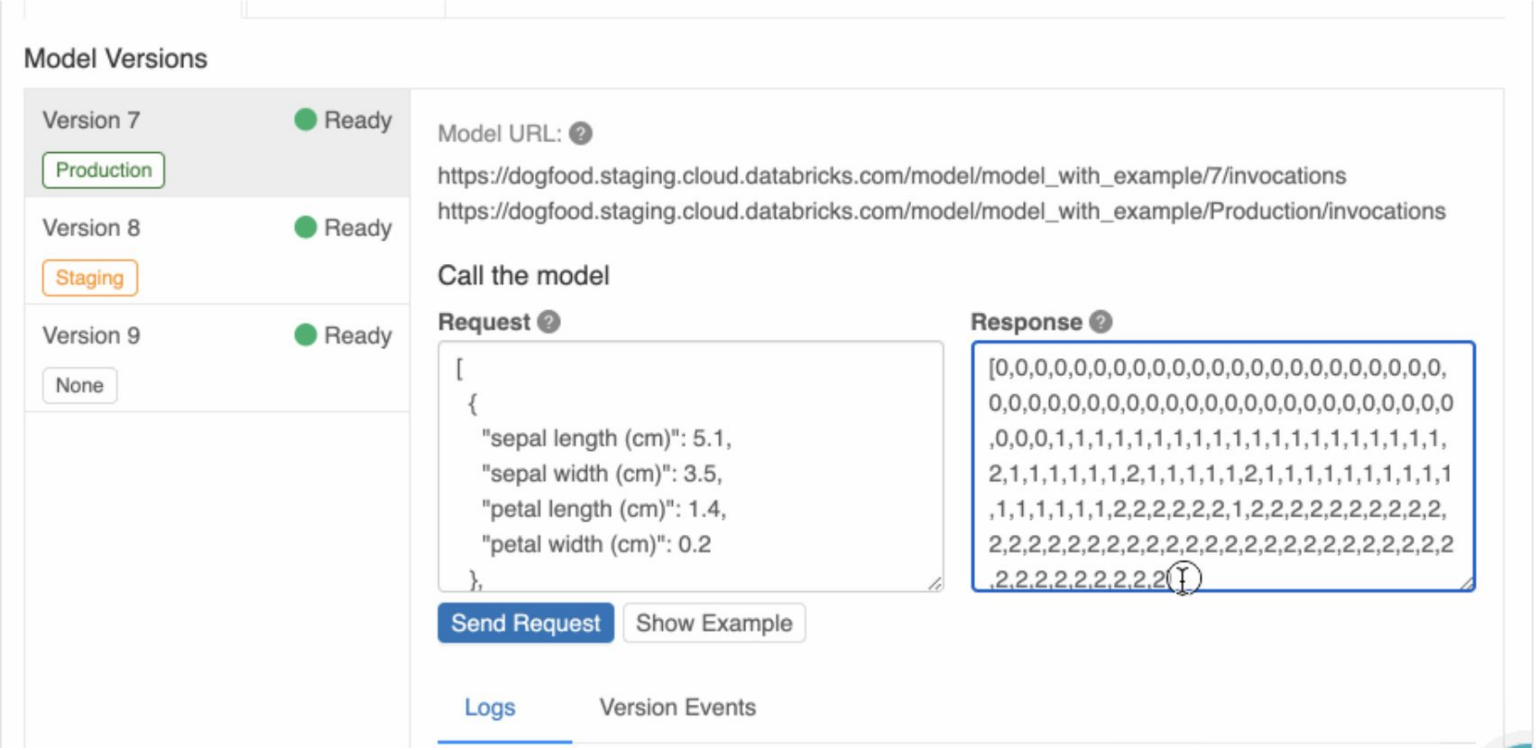
Databricks Production-grade Model Serving

- Production-grade serving is on the product roadmap
- Spin up a multi-node cluster
- Model monitoring, model drift detection and metrics
- Log prediction requests and responses
- Low latency, high availability and scalable
- GPU support

Databricks Model Serving Launch



Databricks Model Scoring



Score with curl

Request

```
curl \
  https://my_workspace.acme.com/my_model/Production/invocations \
  -H "Content-Type:application/json" \
  -d ' {
    "columns": ["sepal_length", "sepal_width", "petal_length", "petal_width"],
    "data": [
      [ 5.1, 3.5, 1.4, 0.2 ],
      [ 4.9, 3.0, 1.4, 0.2 ] ] }'
```

Response

```
[0, 0]
```

Databricks Model Serving Resources

- Blog posts
  - Quickly Deploy, Test, and Manage ML Models as REST Endpoints with MLflow Model Serving on Databricks - 2020-11-02
  - Announcing MLflow Model Serving on Databricks - 2020-06-25
- Documentation
  - MLflow Model Serving on Databricks

Feedback

Your feedback is important to us.

Don't forget to rate and review the sessions.

Thank You

Happy model serving!