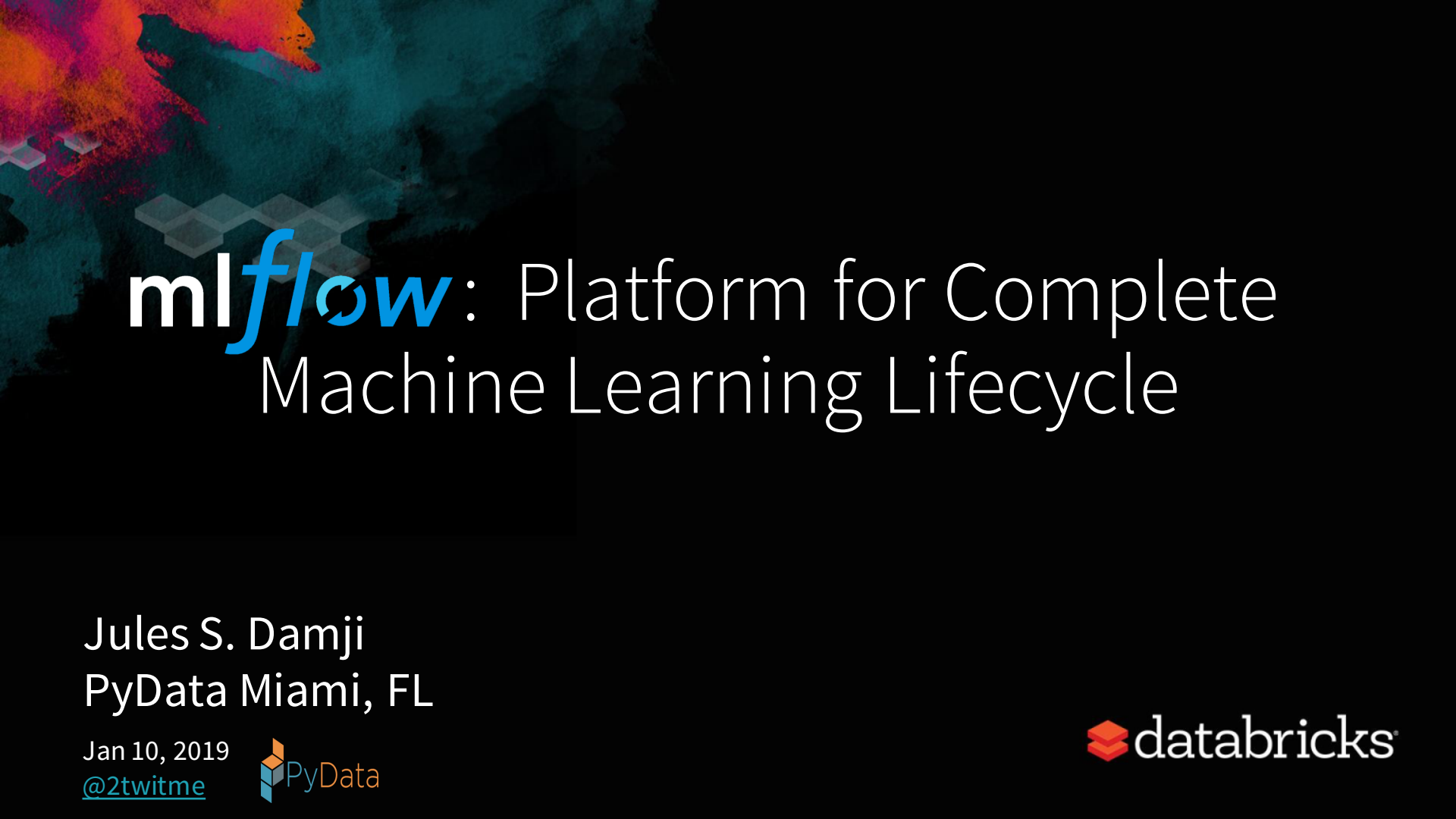


Jules S. Damji
PyData Miami, FL

Jan 10, 2019
@2twitme

PyData

databricks

Outline

Overview of ML development challenges

How MLflow tackles these

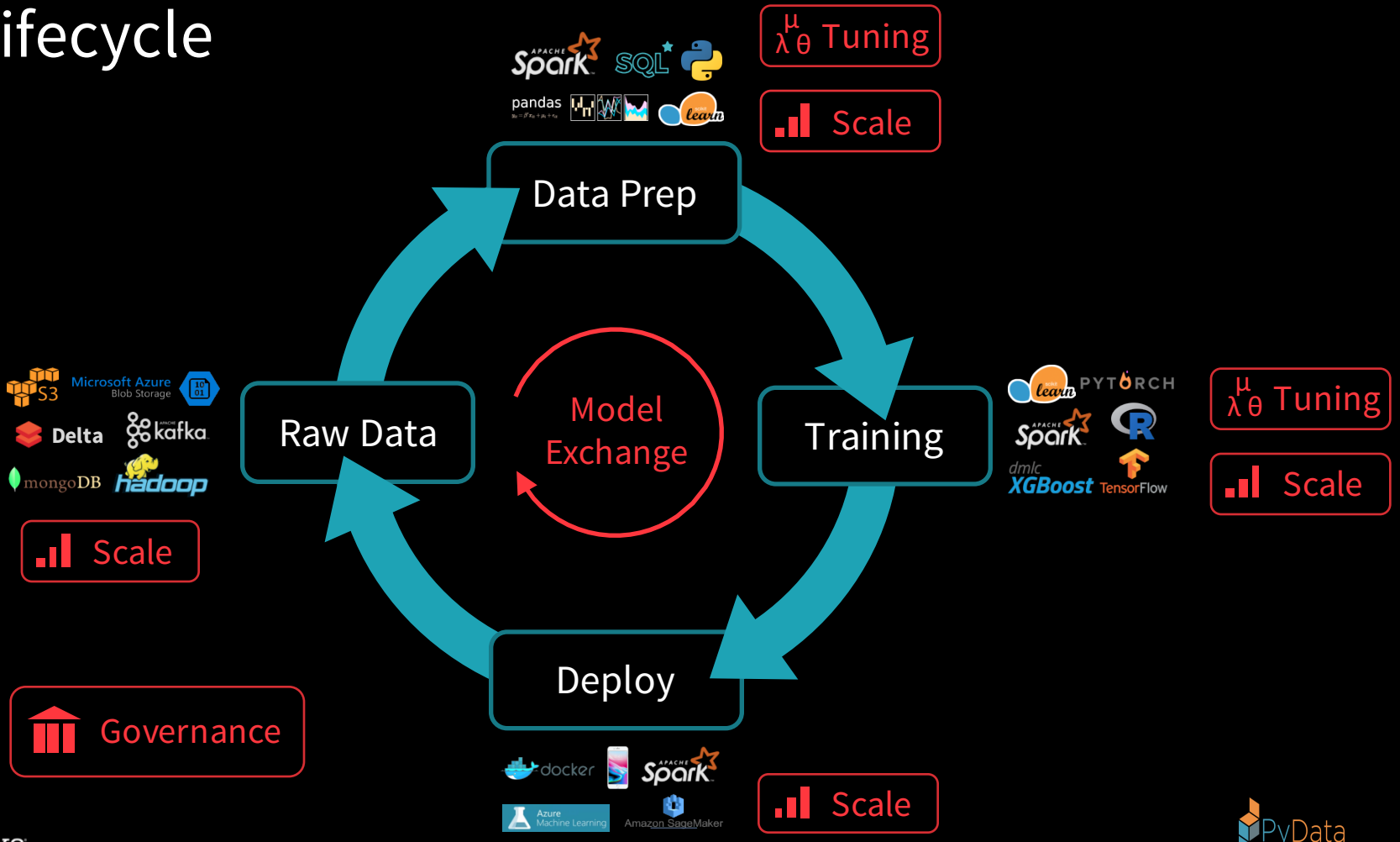
MLflow Components

Developer Experience & Demo

Ongoing Roadmap

Machine Learning Development is Complex

ML Lifecycle



databricks

PyData

Example

“I build 100s of models/day to lift revenue, using any library: MLlib, PyTorch, R, etc. There’s no easy way to see what data went in a model from a week ago, tune it and rebuild it.”

-- Chief scientist at ad tech firm

Custom ML Platforms

Facebook FBLearner, Uber Michelangelo, Google TFX

- + **Standardize the data prep / training / deploy loop: if you work with the platform, you get these!**
- – **Limited to a few algorithms or frameworks**
- – **Tied to one company's infrastructure**
- – **Out of luck if you left the company….**

Can we provide similar benefits in an open manner?

Introducing mlflow

Open machine learning platform

- Works with any ML library & language
- Runs the same way anywhere (e.g., any cloud)
- Designed to be useful for 1 or 1000+ person orgs
- Simple, Easy-to-use, Developer Experience, and get started!

MLflow Design Philosophy

1. "API-first", open platform

- Allow submitting runs, models, etc from any library & language
- Example: a "model" can just be a lambda function that MLflow can then deploy in many places (Docker, Azure ML, Spark UDF, …)

Key enabler: built around REST APIs and CLI

MLflow Design Philosophy

2. Modular design

- Let people use different components individually (e.g., use MLflow's project format but not its deployment tools)
- Not monolithic, Distinctive and Selective

Key enabler: distinct components (Tracking/Projects/Models)

MLflow Components

mlflow Tracking

Record and query experiments: code, configs, results, …etc

mlflow Projects

Packaging format for reproducible runs on any platform

mlflow Models

General model format that supports diverse deployment tools

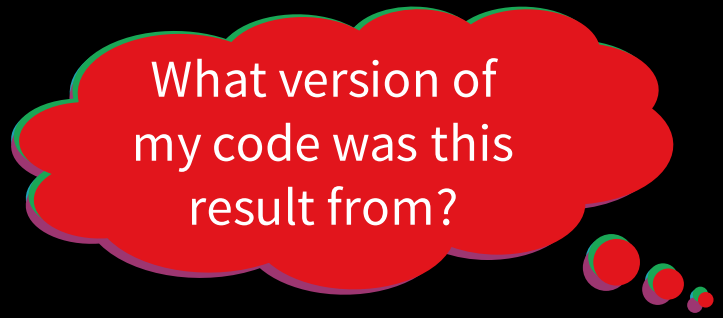
Model Development without MLflow

```
data   = load_text(file)
ngrams = extract_ngrams(data, N=n)
model  = train_model(ngrams,
             learning_rate=lr)
score  = compute_accuracy(model)

print("For n=%d, lr=%f: accuracy=%f"
      % (n, lr, score))

pickle.dump(model, open("model.pkl"))
```

```
For n=2, lr=0.1: accuracy=0.71
For n=2, lr=0.2: accuracy=0.79
For n=2, lr=0.5: accuracy=0.83
For n=2, lr=0.9: accuracy=0.79
For n=3, lr=0.1: accuracy=0.83
For n=3, lr=0.2: accuracy=0.82
For n=4, lr=0.5: accuracy=0.75
...
```



Key Concepts in Tracking

Parameters: key-value inputs to your code

Metrics: numeric values (can update over time)

Tags and Notes: information about a run

Artifacts: files, data and models

Source: what code ran?

Version: what of the code?

PyData

MLflow Tracking API: *Simple!*

mlflow
Tracking

Record and query experiments: code, configs, results, …etc

```
import mlflow

 # log model's tuning parameters

with mlflow.start_run():
  mlflow.log_param("layers", layers)
  mlflow.log_param("alpha", alpha)

  # log model's metrics
  mlflow.log_metric("mse", model.mse())
  mlflow.log_artifact("plot", model.plot(test_df))
  mlflow.tensorflow.log_model(model)
```

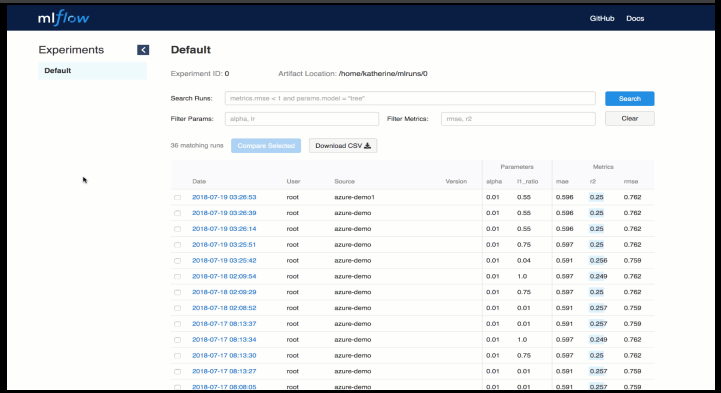
Model Development *with* MLflow is *Simple!*

```
data   = load_text(file)
ngrams = extract_ngrams(data, N=n)
model  = train_model(ngrams,
                     learning_rate=lr)
score  = compute_accuracy(model)

mlflow.log_param("data_file", file)
mlflow.log_param("n", n)
mlflow.log_param("learning_rate", lr)
mlflow.log_metric("score", score)

mlflow.sklearn.log_model(model)
```

```
$ mlflow ui
```



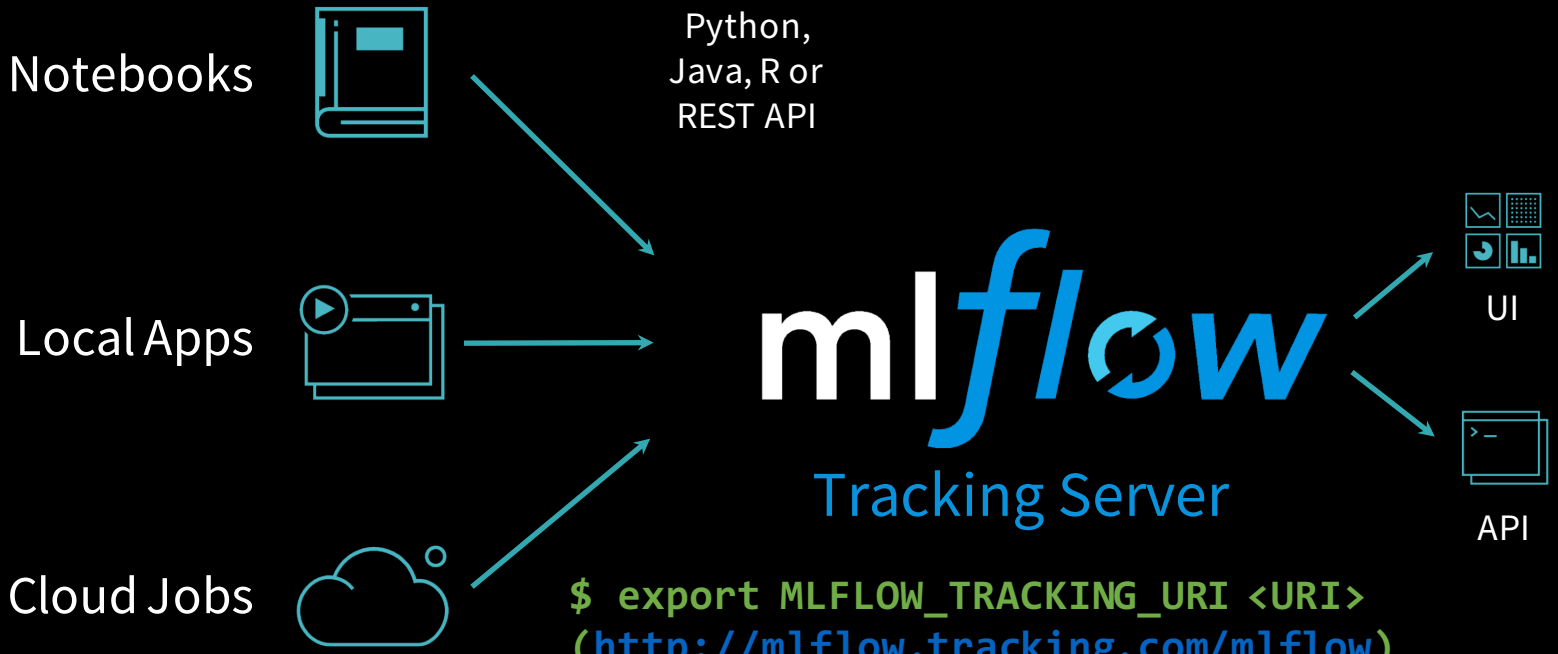
Track parameters, metrics, output files & code version

Search using UI or API

databricks

PyData

MLflow Tracking



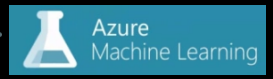
PyData

MLflow Projects Motivation

Diverse set of tools


Diverse set of environments



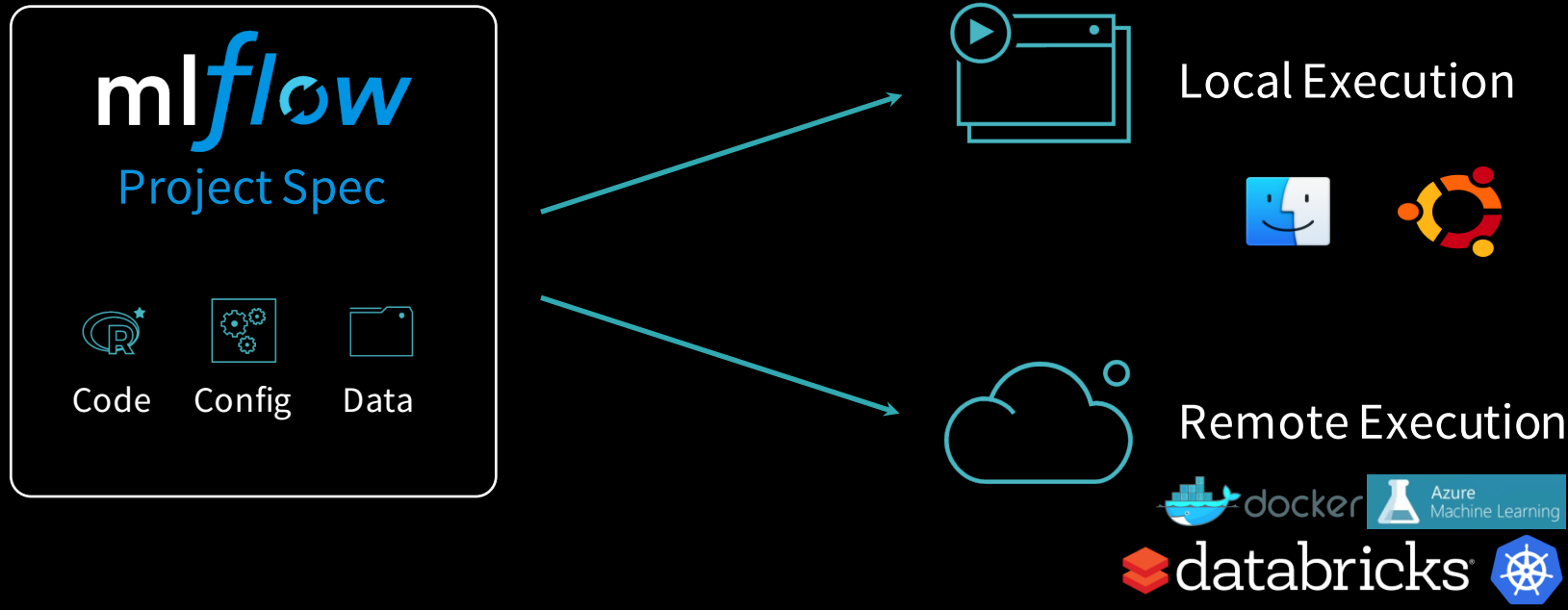


mlflow
Projects

Packaging format
for reproducible
runs
on any platform

Result: It is difficult to productionize and share.

MLflow Projects



Example MLflow Project

```
my_project/
├── MLproject
│
│
│
│
│
│
├── conda.yaml
├── main.py
└── model.py
    ...
```

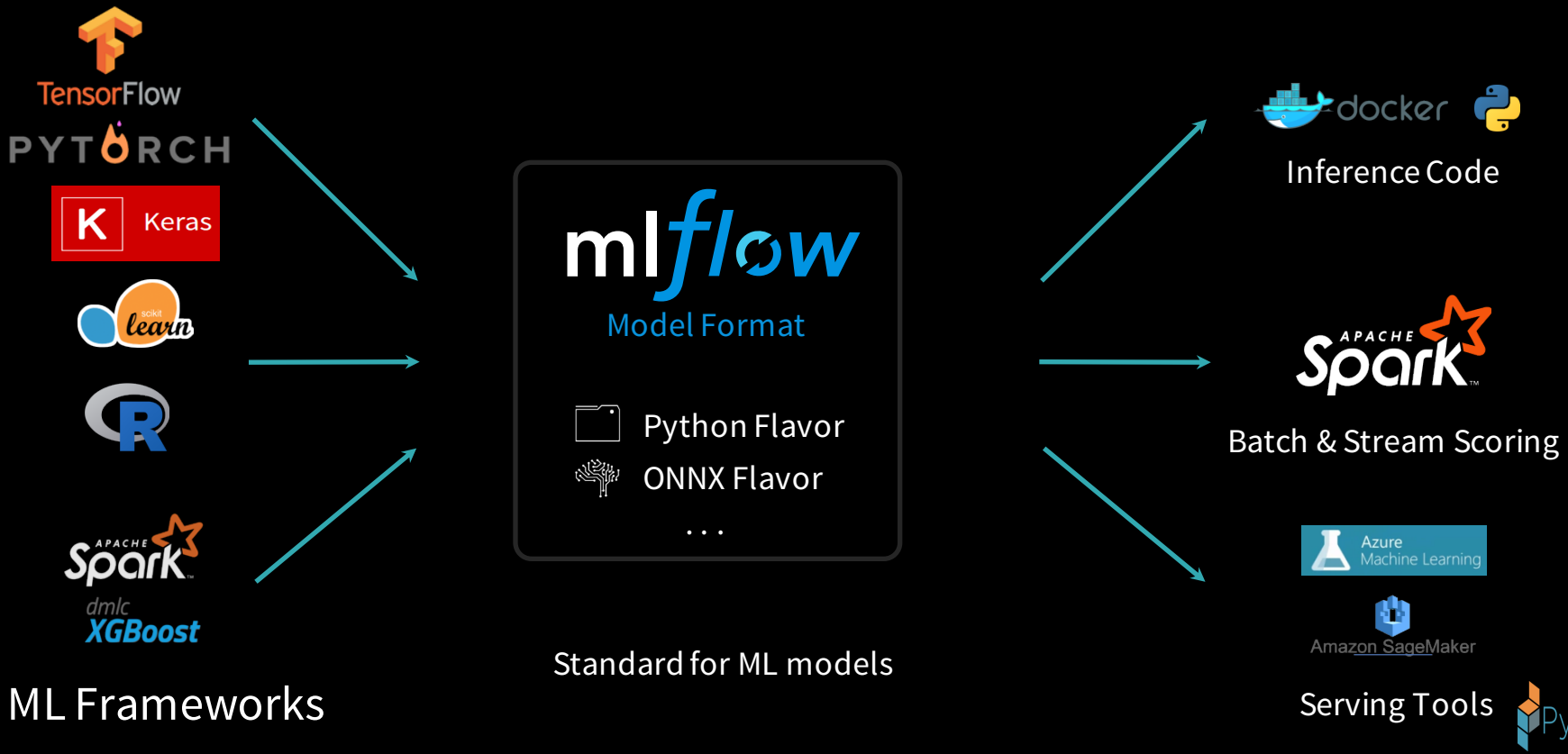
```
conda_env: conda.yaml

entry_points:
  main:
    parameters:
      training_data: path
      lambda: {type: float, default: 0.1}
    command: python main.py {training_data} {lambda}
```

```
$ mlflow run git://<my_project>
```

```
mlflow.run("git://<my_project>", ...)
```

MLflow Models



Example MLflow Model

```
my_model/
├── MLmodel
└── estimator/
    ├── saved_model.pb
    └── variables/
        ...
```

```
run_id: 769915006efd4c4bbd662461
time_created: 2018-06-28T12:34
flavors:
  tensorflow:
    saved_model_dir: estimator
    signature_def_key: predict
  python_function:
    loader_module: mlflow.tensorflow
```

Usable by tools that understand TensorFlow model format

Usable by any tool that can run Python (Docker, Spark, etc!)

```
>>> mlflow.tensorflow.log_model(...)
```

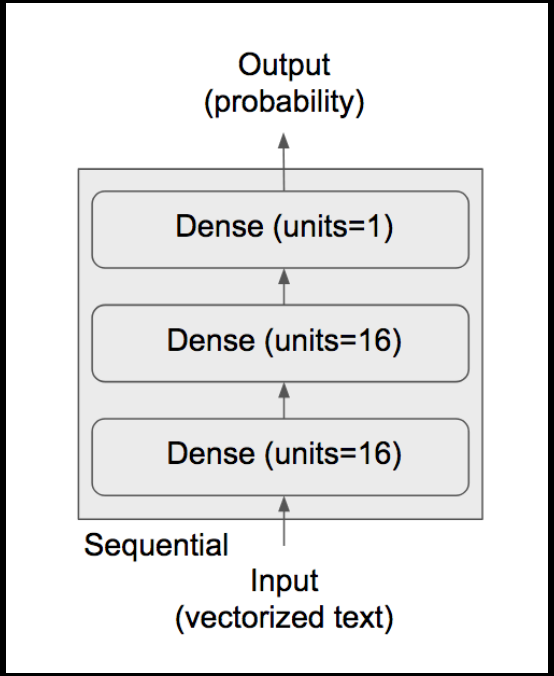
Developer Experience & Demo

Binary Classification of
IMDB Movie Reviews Using
Keras Neural Network Model

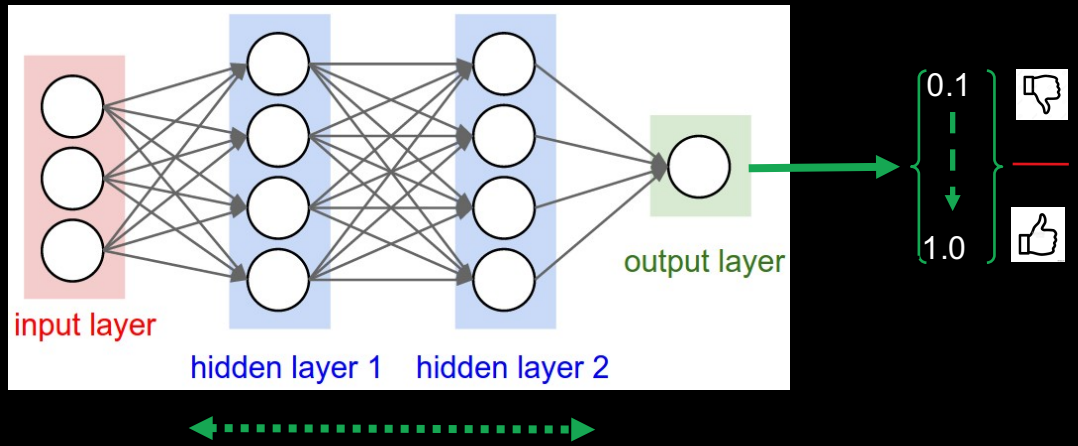
https://dbricks.co/keras_imdb
https://github.com/dmatrix/jsd-mlflow-examples/

MLflow Baseline & Experiment Models

Model	Units	Epochs	Loss Function	Hidden Layers
Base	16	20	binary_crosstropy	1
Experiment-1	32	30	binary_crosstropy	3
Experiment-2	32	20	mse	3



MLflow Baseline & Experiment Models



Ongoing MLflow Roadmap

- TensorFlow, Keras, PyTorch, H2O, MLlib integrations ✔
- Java and R MLflow Client language APIs ✔
- Multi-step workflows ✔
- Hyperparameter tuning ✔
- Integration with Databricks Tracking Server ✔
- Support for other Data Store (e.g., MySQL) ✔
- Data source API based on Spark data sources
- Model metadata & management

Just released v8.0.1

- Faster & Improved UI
- Extended Python Model as Spark UDF
- Persist model dependencies as Conda Environment

Learning More About MLflow

`pip install mlflow` to get started

Find docs & examples at **mlflow.org**

tinyurl.com/mlflow-slack

What Did We Talk About? mlflow

Workflow tools can greatly simplify the ML lifecycle

- Improve usability among data scientists and engineers
- Simplify lifecycle development
- Lightweight, open platform that integrates easily
- Available APIs: Python, Java & R
- Easy to install and use
- Develop locally and track locally or remotely

databricks

15% Discount Code: PyDataMiami

PyData

Thank You ☺

jules@databricks.com
@2twitme
https://www.linkedin.com/in/dmatrix/