mlflow: Platform for Complete Machine Learning Lifecycle

https://dbricks.co/mlflow_strata_nyc

Jules S. Damji
Strata NYC
Sept 24 2019
@2twitme

databricks

Guest WiFi:
password:

I have used ML Frameworks Before…

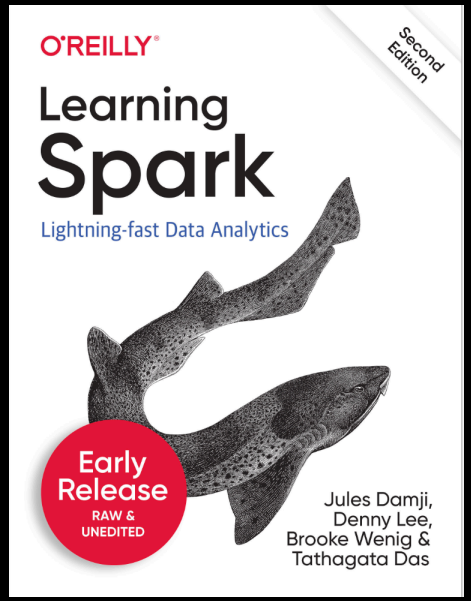
Apache Spark Developer & Community Advocate @ Databricks

Developer Advocate @ Hortonworks

Software engineering @ Sun Microsystems, Netscape, @Home, Excite@Home, VeriSign, Scalix, Centrify, LoudCloud/Opsware, ProQuest

Program Chair Spark + AI Summit

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



databricks

Outline

Overview of ML development challenges

How MLflow tackles these

MLflow Components

Ongoing Roadmap

Q & A

Break

Machine Learning Development is Complex

ML Development Challenges

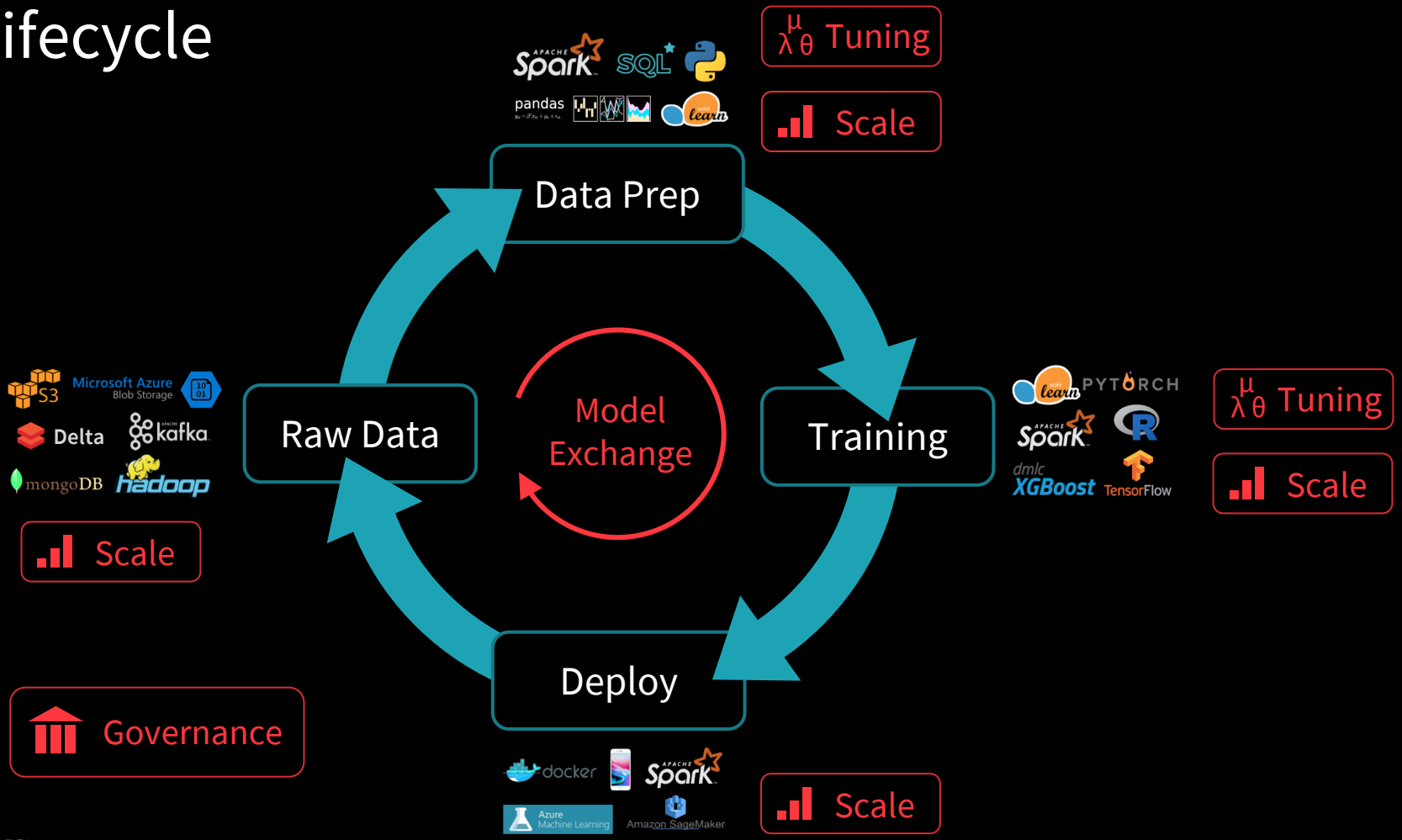
Zoo of software tools

Tracking & reproducing results

Productionizing models

Scaling

ML Lifecycle



databricks

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

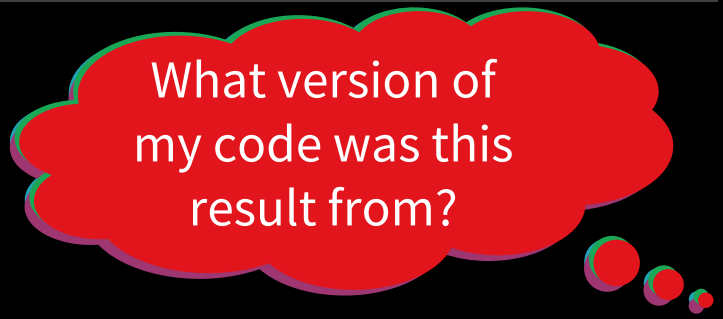
Model Development without MLflow

```
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?

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)
```

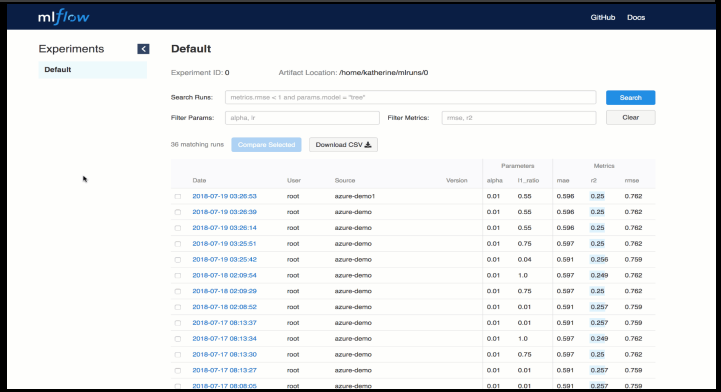
Model Development *with* MLflow is *Simple!*

```
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
```

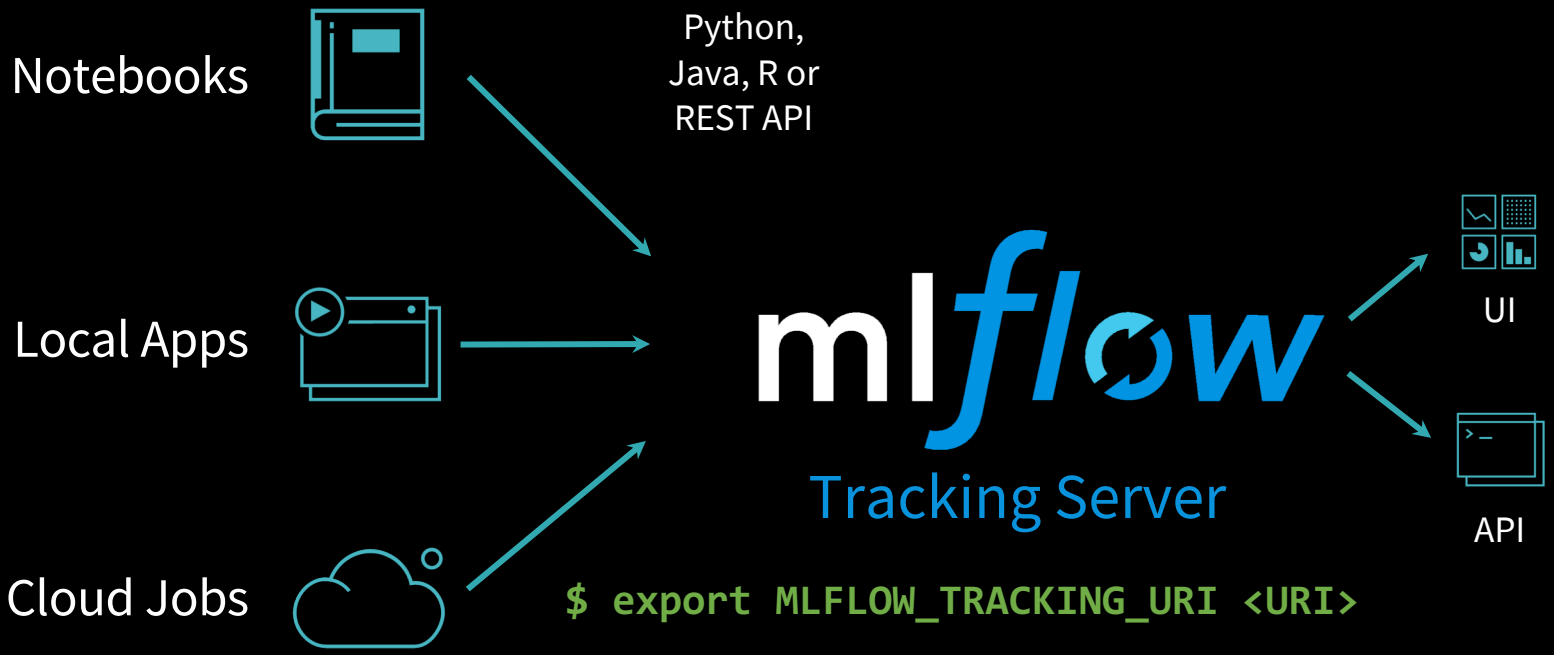


Track parameters, metrics, output files & code version

Search using UI or API

databricks

MLflow Tracking



MLflow Tracking Backend Stores

1. **Entity (Metadata) Store**
   - FileStore (local filesystem)
   - SQLStore (via SQLAlchemy)
   - REST Store

2. **Artifact Store**
   - S3 backed store
   - Azure Blob storage
   - Google Cloud storage
   - DBFS artifact repo

MLflow Components

mlflow Tracking

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

mlflow Projects

Packaging format for reproducible runs on any platform

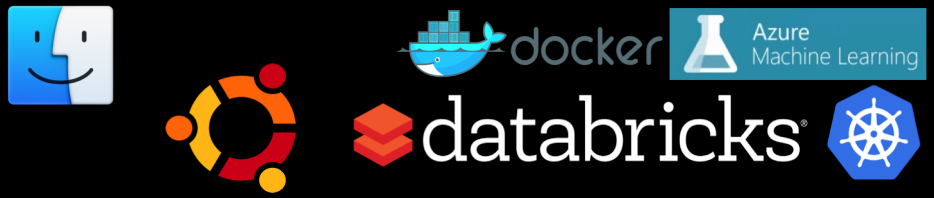
mlflow Models

General model format that supports diverse deployment tools

MLflow Projects Motivation

Diverse set of tools

Diverse set of environments

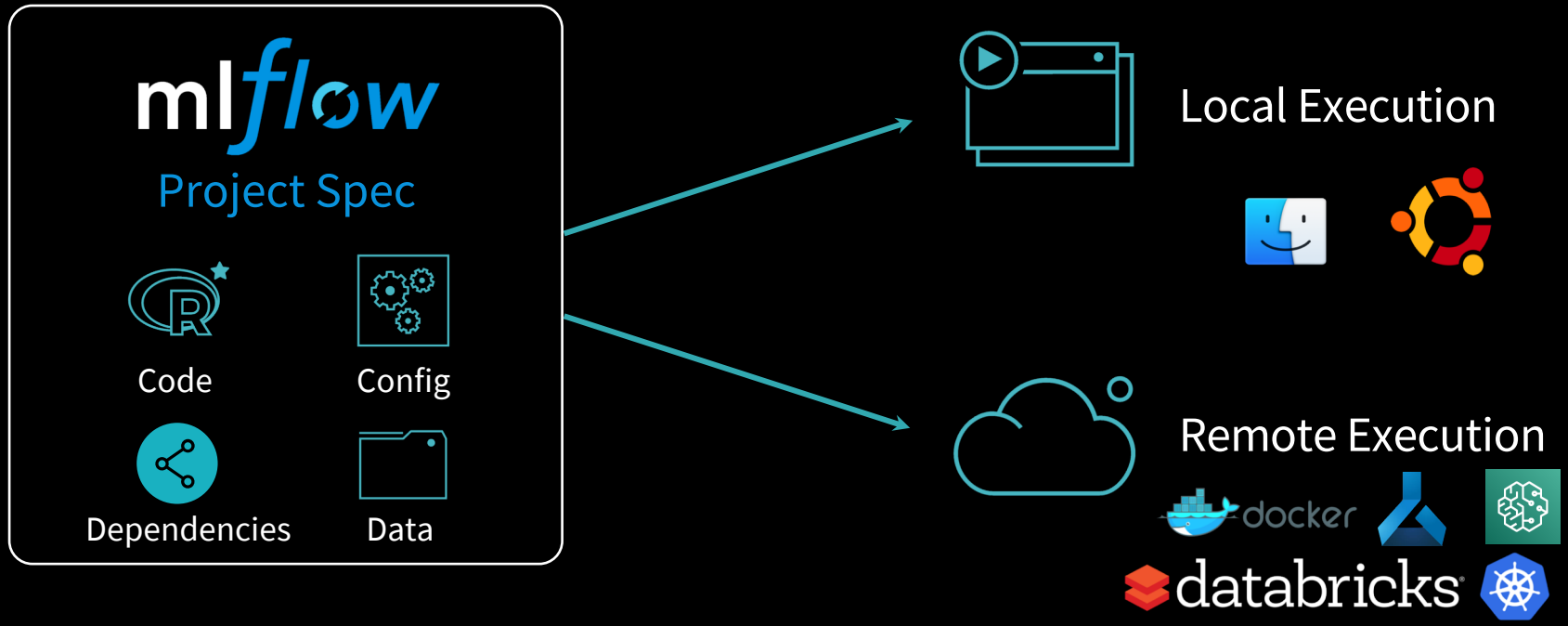


mlflow Projects

Packaging format
for reproducible
runs
on any platform

Result: It is difficult to productionize and share.

MLflow Projects



MLflow Projects

Packaging format for reproducible ML runs

- Any code folder or GitHub repository
- Optional MLproject file with project configuration

Defines dependencies for reproducibility

- Conda (+ R, Docker, …) dependencies can be specified in MLproject
- Reproducible in (almost) any environment

Execution API for running projects

- CLI / Python / R / Java
- Supports local and remote execution

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> -P lambda=0.2
```

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

MLflow Components

mlflow Tracking

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

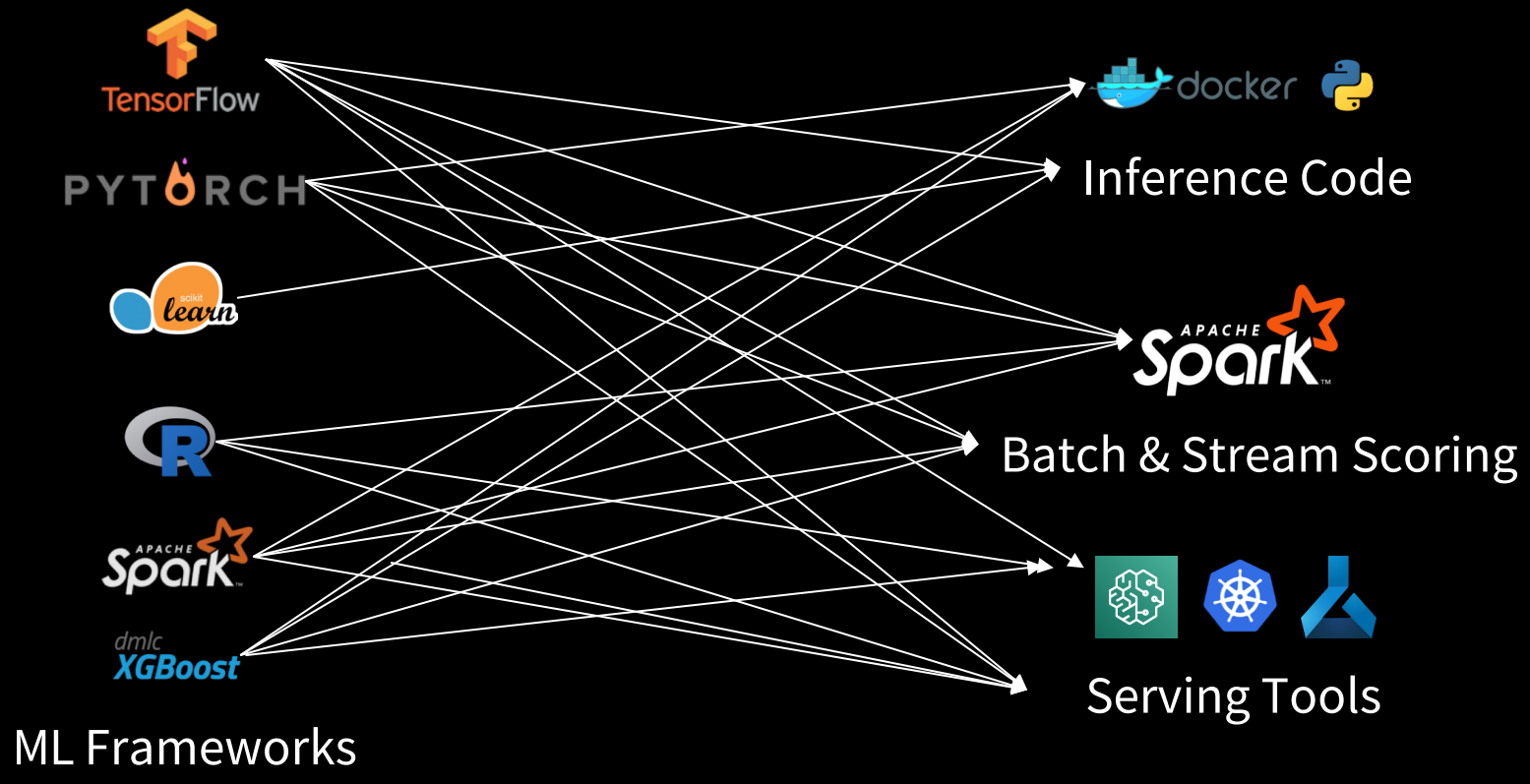
mlflow Projects

Packaging format for reproducible runs on any platform

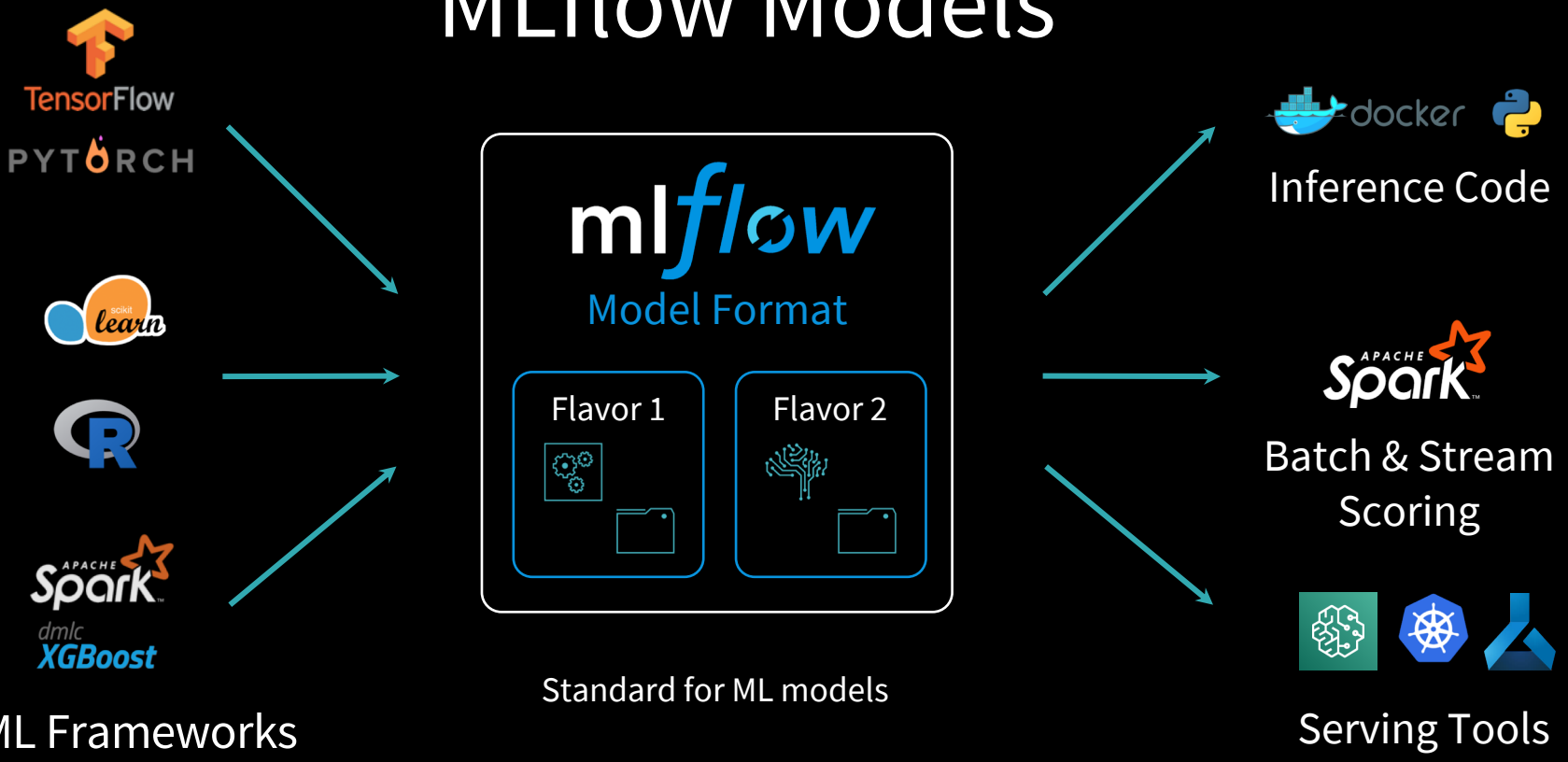
mlflow Models

General model format that supports diverse deployment tools

Mlflow Models Motivation



MLflow Models



Example MLflow Model

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

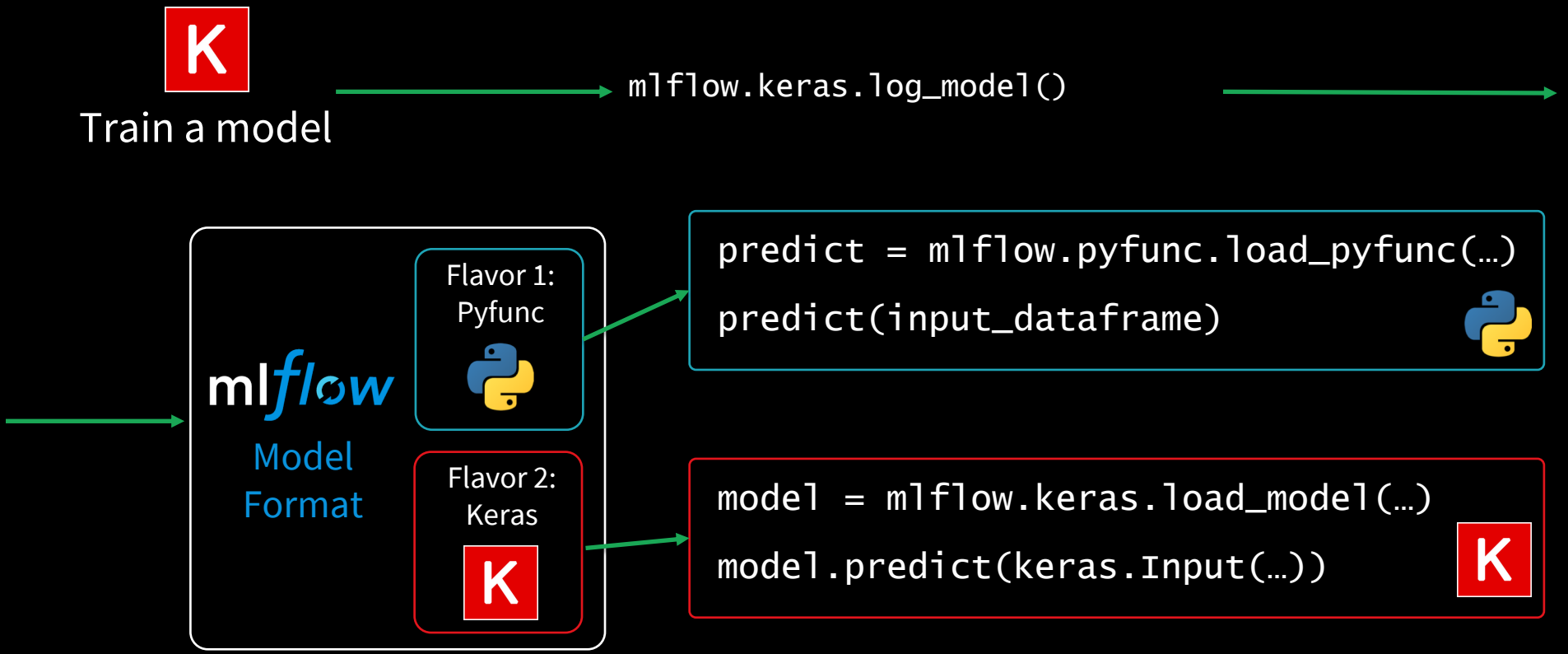
```
run_id: 769915006efd4c4bbd662461
time_created: 2018-06-28T12:34
flavors:
  tensorflow:
    saved_model_dir: estimator
    signature_def_key: predict
  python_function:
    loader_module: mlflow.tensorflow
```

Usable by tools that understand TensorFlow model format

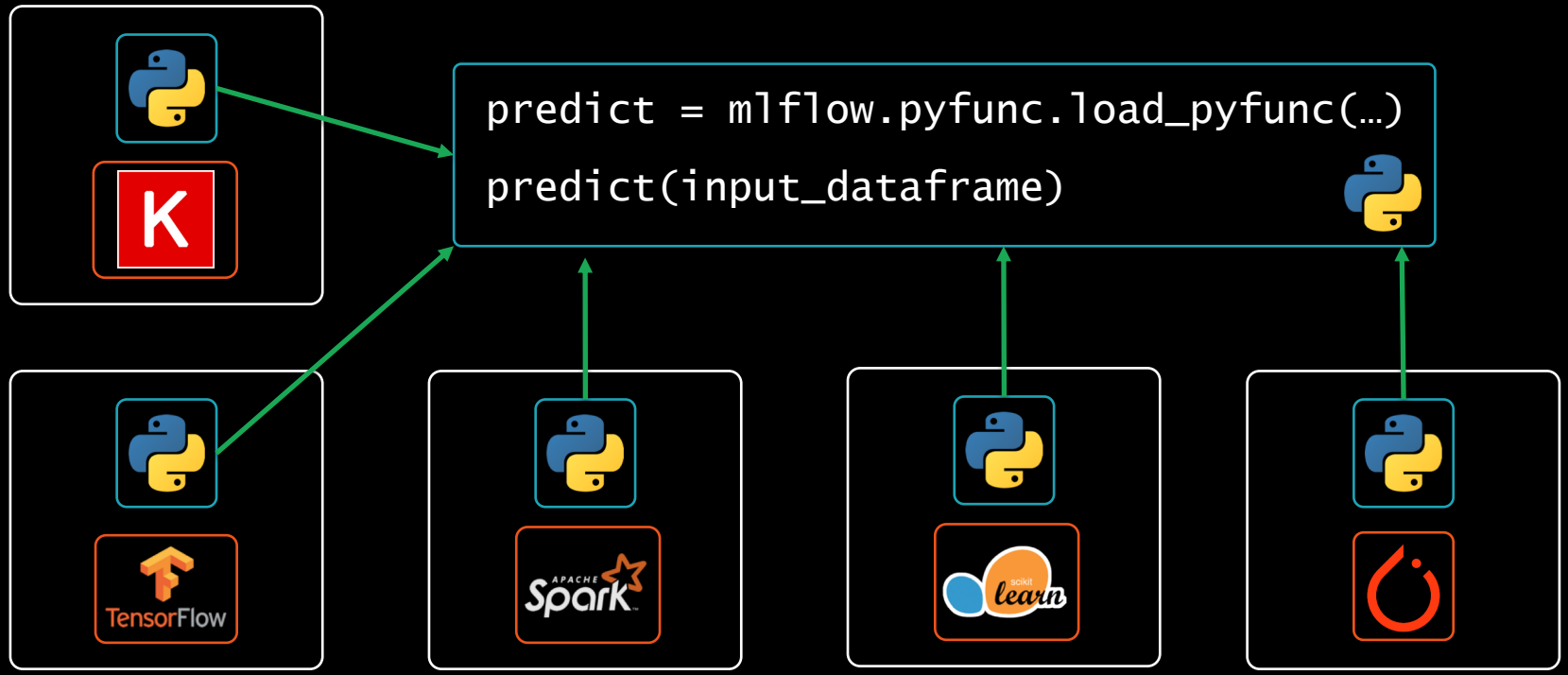
Usable by any tool that can run Python (Docker, Spark, etc!)

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

Model Flavors Example



Model Flavors Example



MLflow Roadmap 1.2 & Beyond

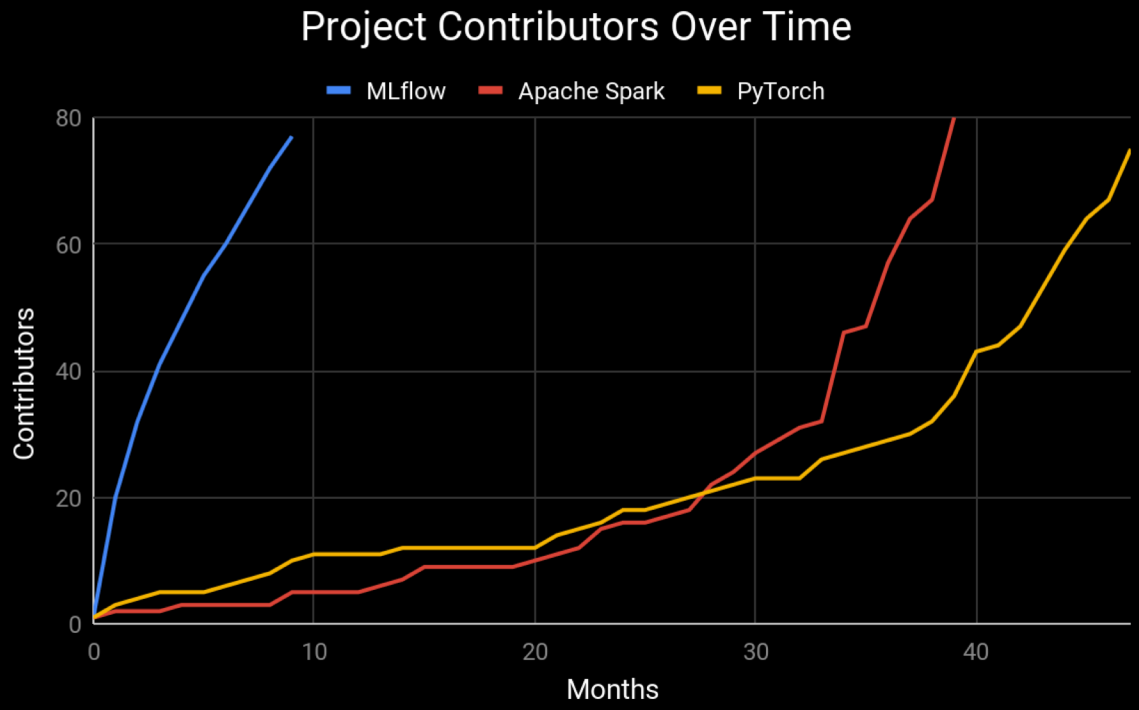
MLflow 1.2 was released recently! Major features:

- New metrics UI
- Model metadata & management & registry
- Multi-step workflows
- “Step” axis for metrics
- Improved search capabilities
- Support for ONNX models

Project & Community Growth

Rapid Community Adoption

- 77 code contributors from >40 companies in the past 8 months



Learning More About MLflow

- pip install mlflow to get started
- Find docs & examples at mlflow.org
- https://github.com/mlflow/mlflow
- tinyurl.com/mlflow-slack

What Did We Talk About? mlflow

Workflow tools can greatly simplify the ML lifecycle

- 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
- Deploy locally, cloud, on premise…
- Visualize experiments

Thank You ☺

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