mlflow : Platform for Complete Machine Learning Lifecycle

Mani Parkhe & Tomas Nykodym

Oct 4, 2018

databricks

Outline

Overview of ML development challenges

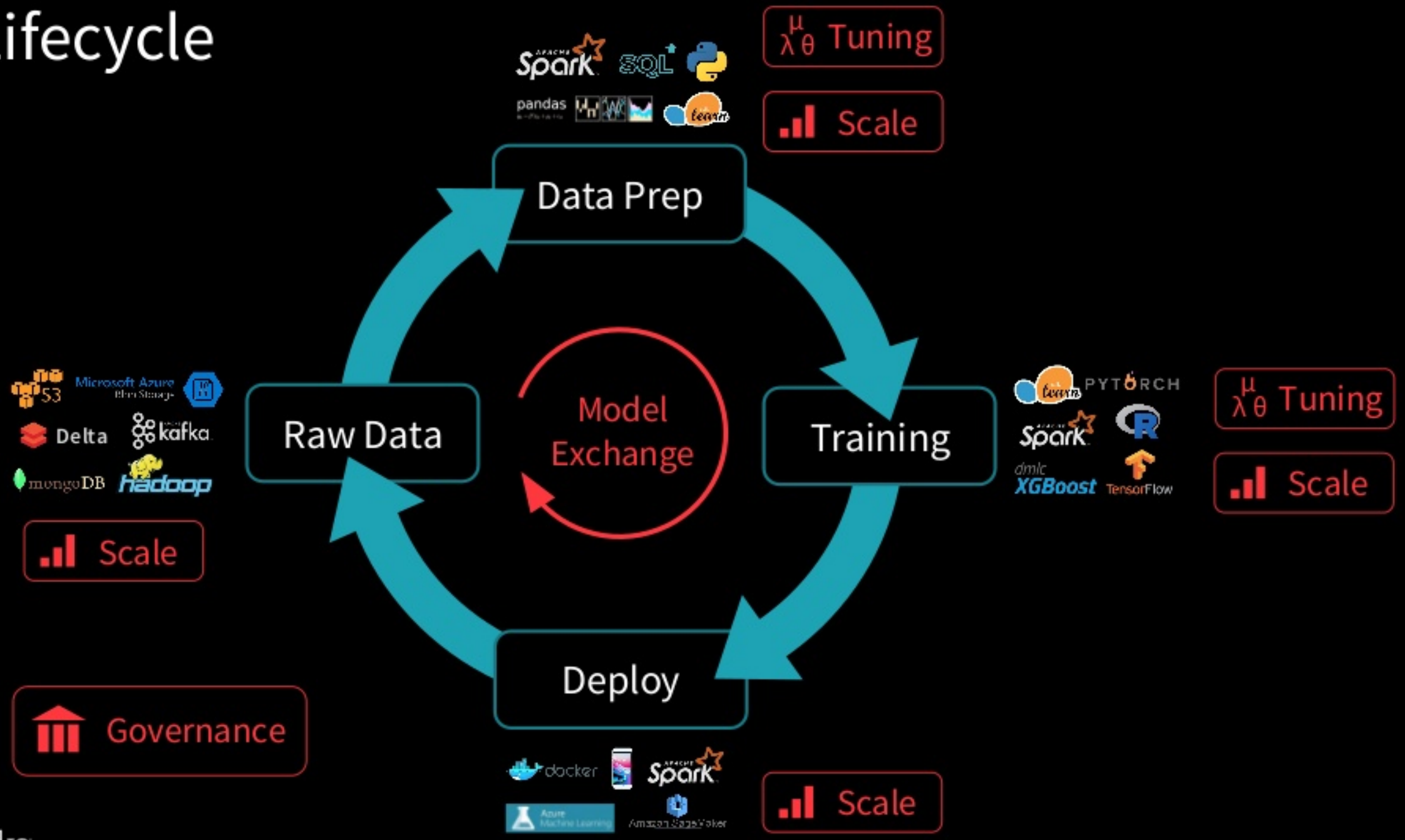
How MLflow tackles these

MLflow components

How to get started

Machine Learning Development is Complex

ML Lifecycle



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

Example

“Our company has 100 teams using ML worldwide. We can’t share work across them: when a new team tries to run some code, it often doesn’t even give the same result.”

-- Large consumer electronics 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

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

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 Tracking

mlflow Tracking

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

```
import mlflow

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

  # train model

  mlflow.log_metric("mse", model.mse())
  mlflow.log_artifact("plot", model.plot(test_df))
  mlflow.tensorflow.log_model(model)
```

Demo

Goal: Predict Price of Airbnb Listings

listing attributes

bathrooms: 1
bedrooms: 2
accommodates: 4
total_reviews: 45
cleanliness_rating: 9
location_rating: 10
checkin_rating: 10
zip_code: 94105

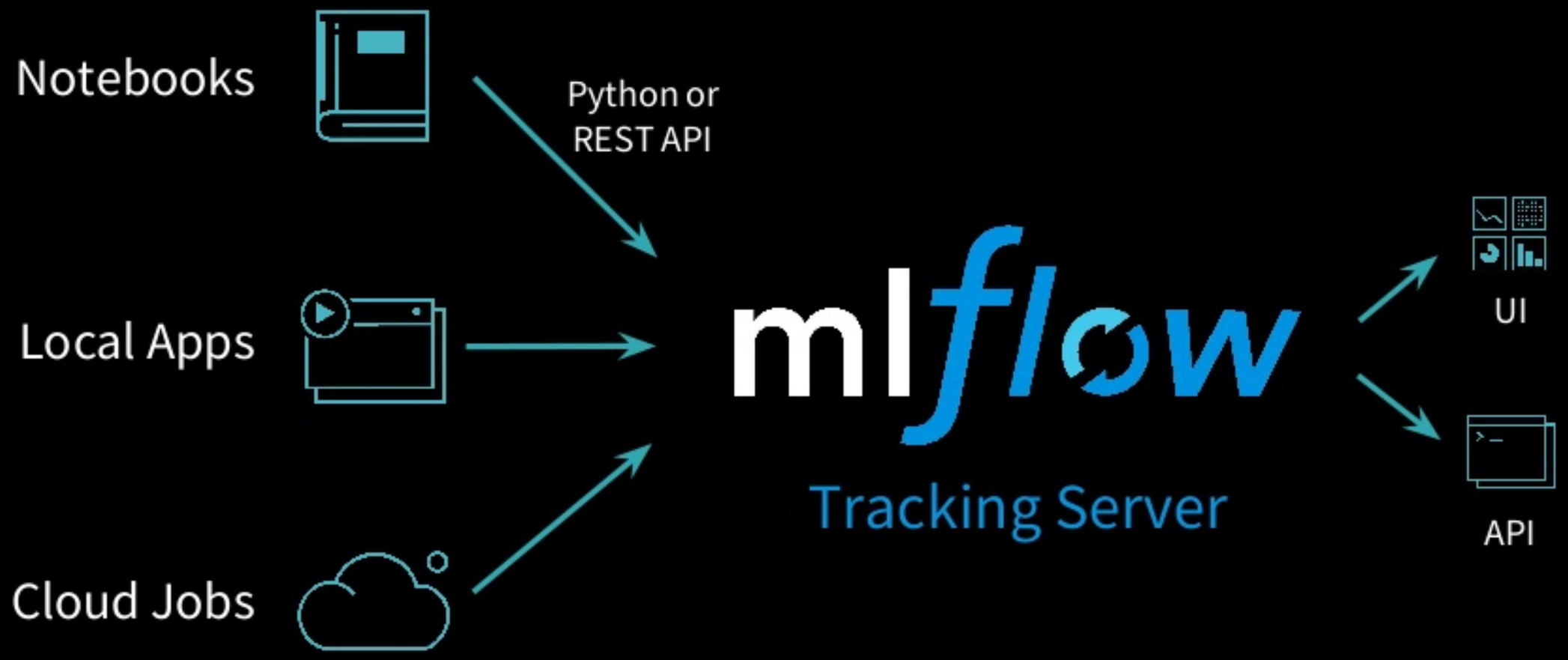
$f(x)$

Model

→ **price:** 150

based on data from insideairbnb.com

MLflow Tracking



Key Concepts in Tracking

Parameters: key-value inputs to your code

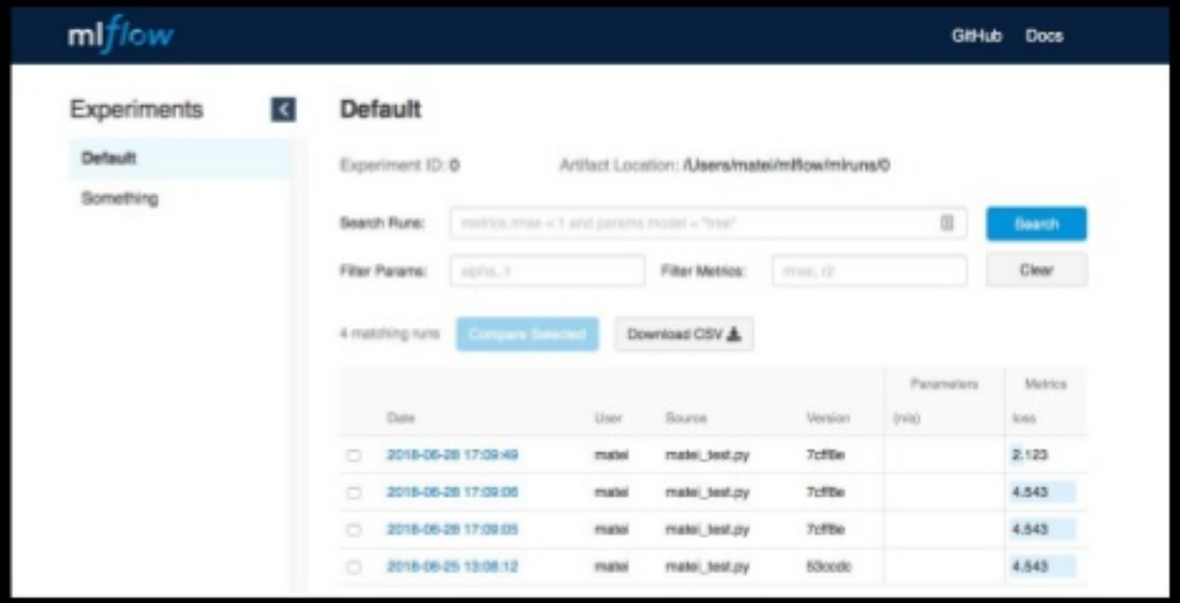
Metrics: numeric values (can update over time)

Tags and Notes: additional information about a run

Artifacts: arbitrary files, including data and models

Source: what code ran?

Version: what of the code?



MLflow backend stores

1. **Entity Store**
   - FileStore (local and REST)
   - Database backed (coming soon)

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

Learning More About MLflow

`pip install mlflow` to get started

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

tinyurl.com/mlflow-slack

Break

MLflow Deep Dive Part #2

Closer look at MLflow Projects and Models

mlflow Tracking

Record and query experiments: code, data, config, results

mlflow Projects

Packaging format for reproducible runs on any platform

mlflow Models

General model format that supports diverse deployment tools

MLflow Deep Dive Part #2

Outline

2.1 MLflow Projects

- Motivation
- Quick Overview
- Examples
- Demo

2.2 MLflow Models

- Motivation
- Quick Overview
- About Flavors
- Examples
- Demo

2.1 MLflow Projects

- Motivation
- Quick Overview
- Examples
- Demo

MLflow Projects Motivation

Diverse set of tools

PYTORCH

TensorFlow

learn

APACHE Spark

dmlc XGBoost

R

Diverse set of environments

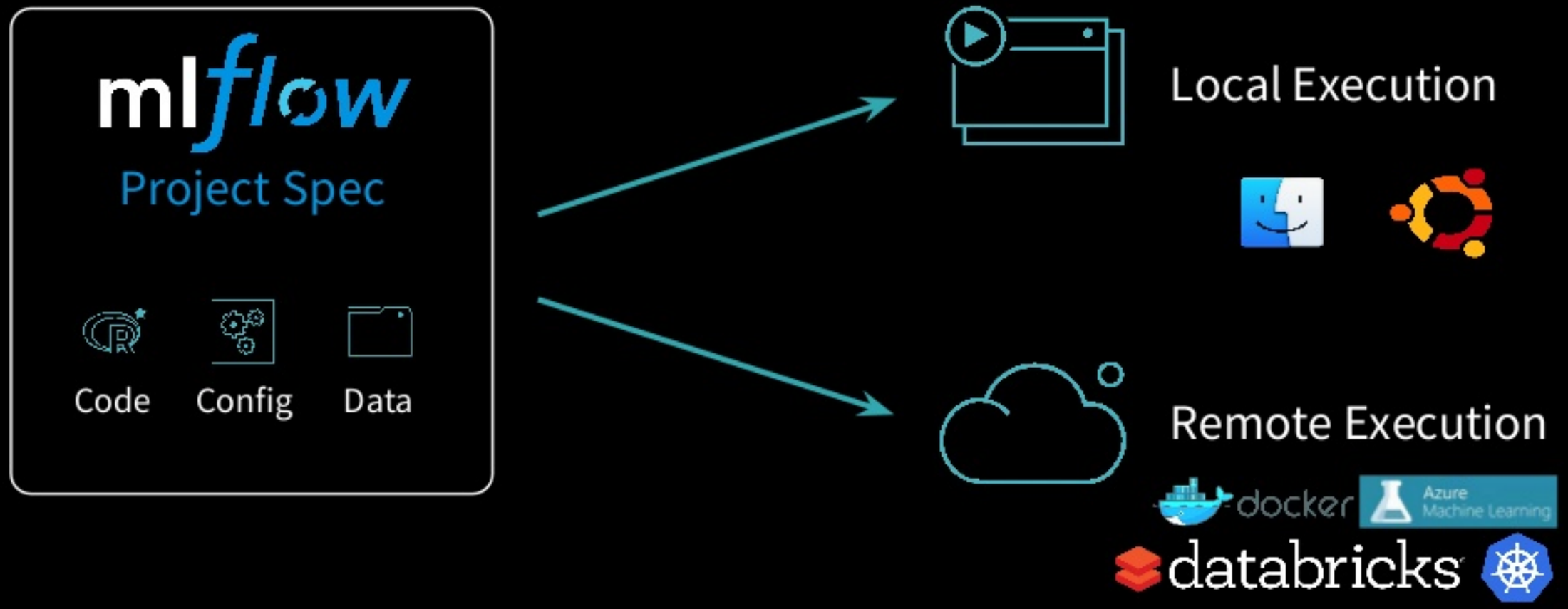
docker

Azure Machine Learning

databricks

Result: It is difficult to productionalize 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 locally or remote

- CLI / Python / R / Java
  - `mlflow run . -e train …`
  - `mlflow.projects.run(., “train”, ...)`

Example MLflow Project

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

```yaml
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>", ...)
```

Airbnb Example Revisited

We have 2 other data scientists working on the project

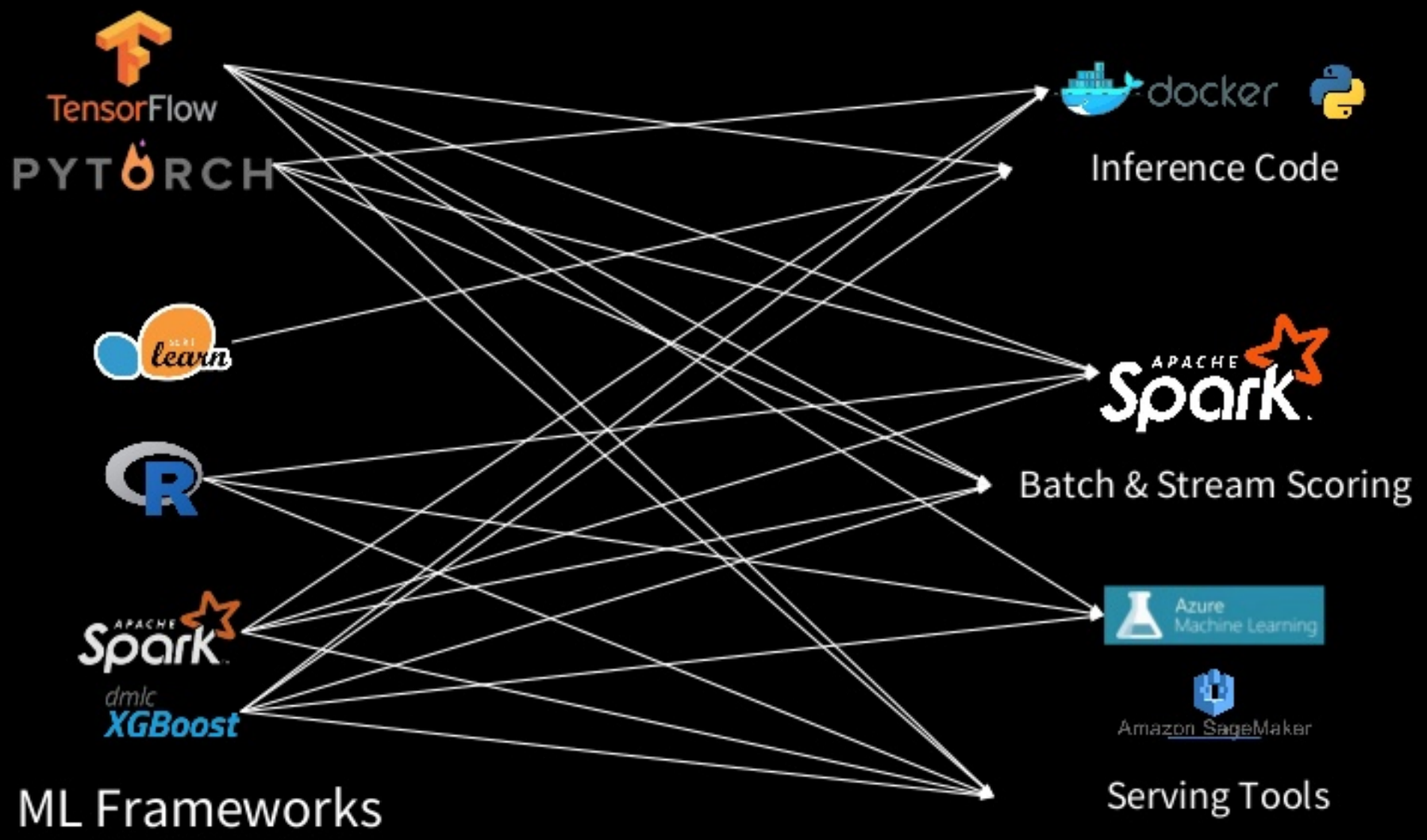
- Aaron used XGBoost in python
- Amy used Keras in R
- They provided their code as MLflow project on github

Let's test their code!

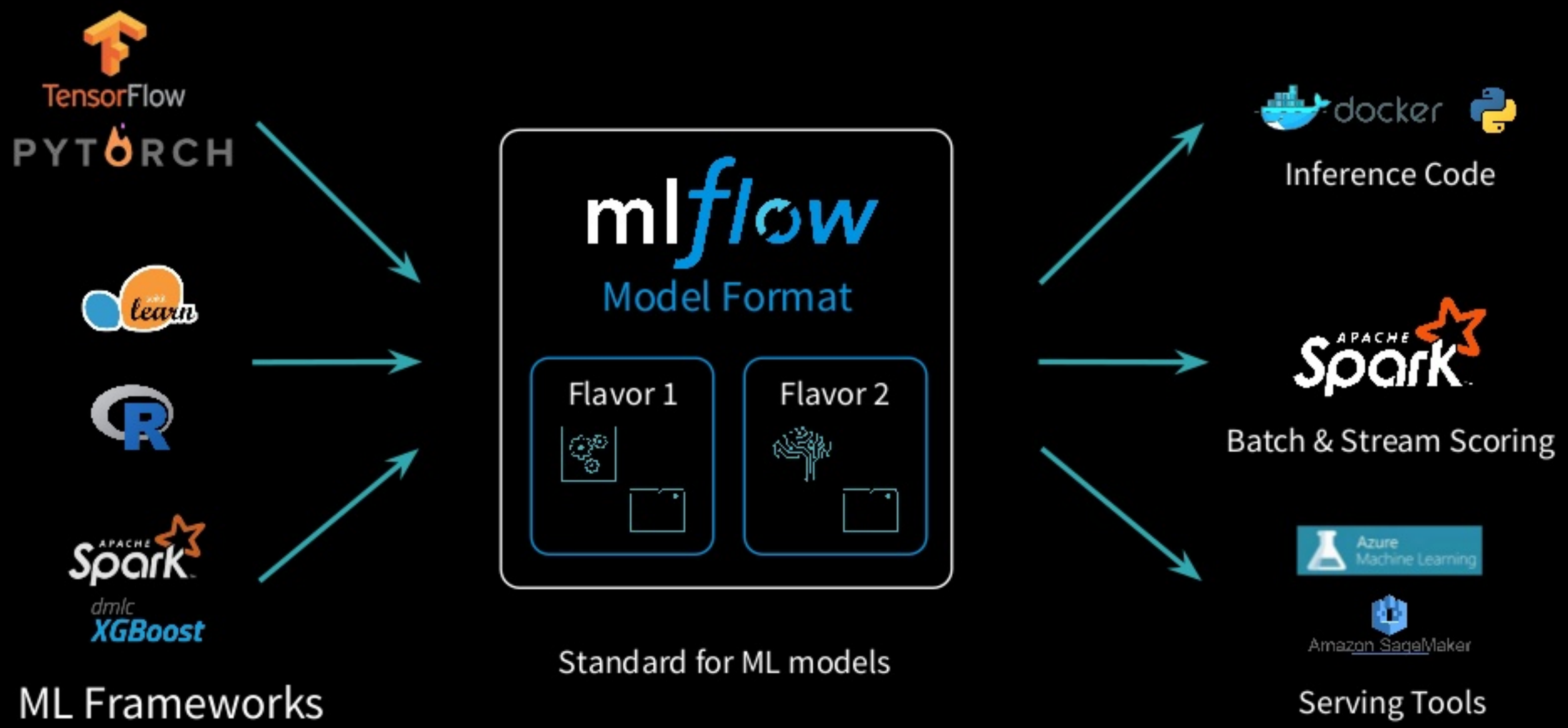
2.2 MLflow Models

- Motivation
- Quick Overview
- About Flavors
- Examples
- Demo

MLflow Models Motivation



MLflow Models



MLflow Models

Packaging format for ML Models

- Any directory with MLmodel file
- Different flavors for different models

Defines dependencies for reproducibility

- Conda (+ R, Docker, … ?) dependencies can be specified in MLproject
- Deployable (almost) anywhere

Deployment APIs

- CLI / Python / R / Java
  - `mlflow pyfunc serve …`
  - `mlflow.pyfunc.load(...).predict()`

Example MLflow Model

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

```yaml
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(...)
```

About Those Flavors

Models can generate several flavors

- Spark ML model generates spark, mleap and pyfunc flavors

Generic flavors provide abstraction layer

- pyfunc can be served locally, as spark_udf, on Azure ML, …
- By generating pyfunc flavor we get all above

Different Flavors Can be Loaded By Different Languages

- pyfunc in python, mleap in Java, crate in R, Keras in python and R

PyFunc - Generic Python Model

PyFunc is saved as “just a directory”:

```
./model_dir/
    ./MLmodel: configuration
    <code>: code packaged with the model (specified in the MLmodel file)
    <data>: data packaged with the model (specified in the MLmodel file)
    <env>: Conda environment definition (specified in the MLmodel file)
```

Model loader specified in MLmodel file:

Arbitrary python code loaded dynamically at runtime

Loaded Pyfunc is “just an object” with a predict method:

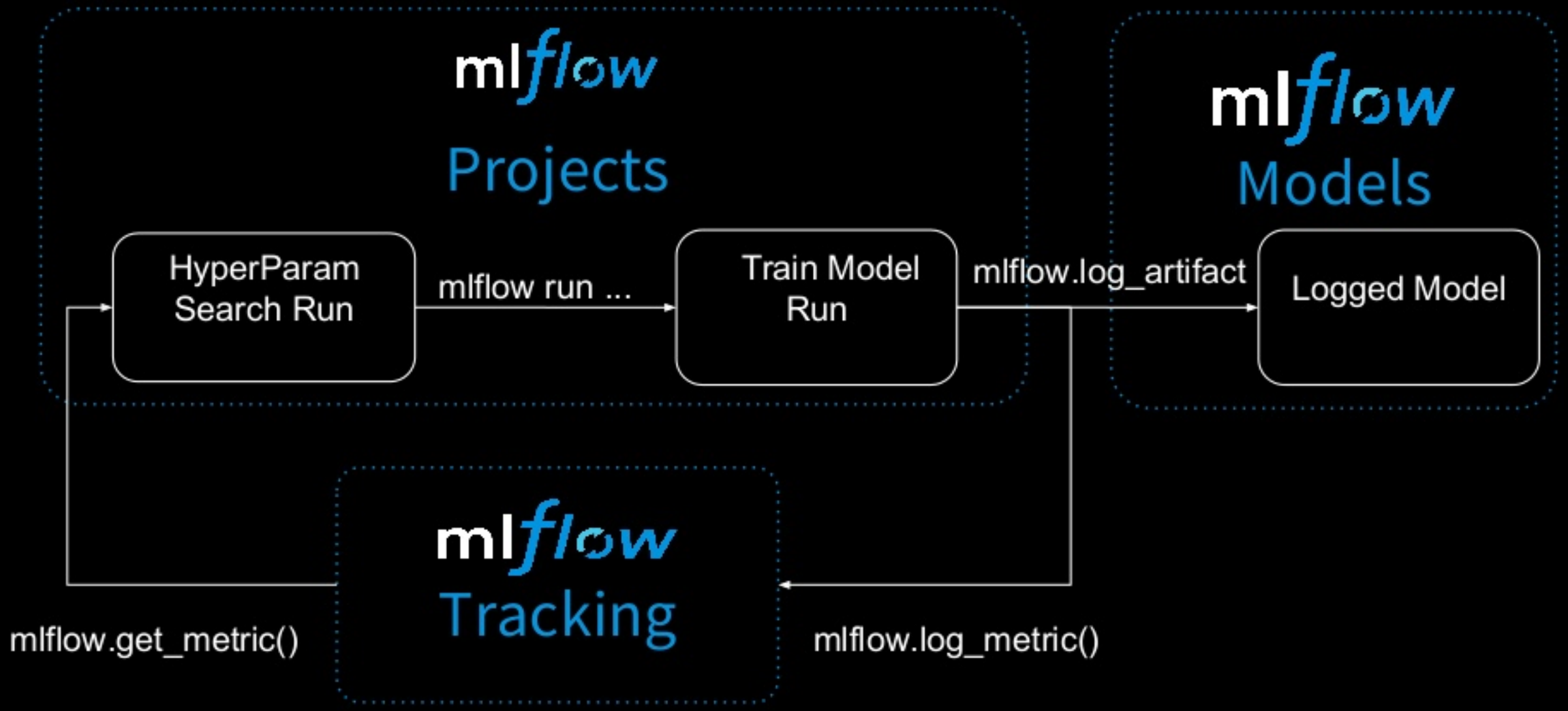
```
predict(pandas.DataFrame) -> pandas.DataFrame | numpy.array
```

Airbnb Example Revisited (Again)

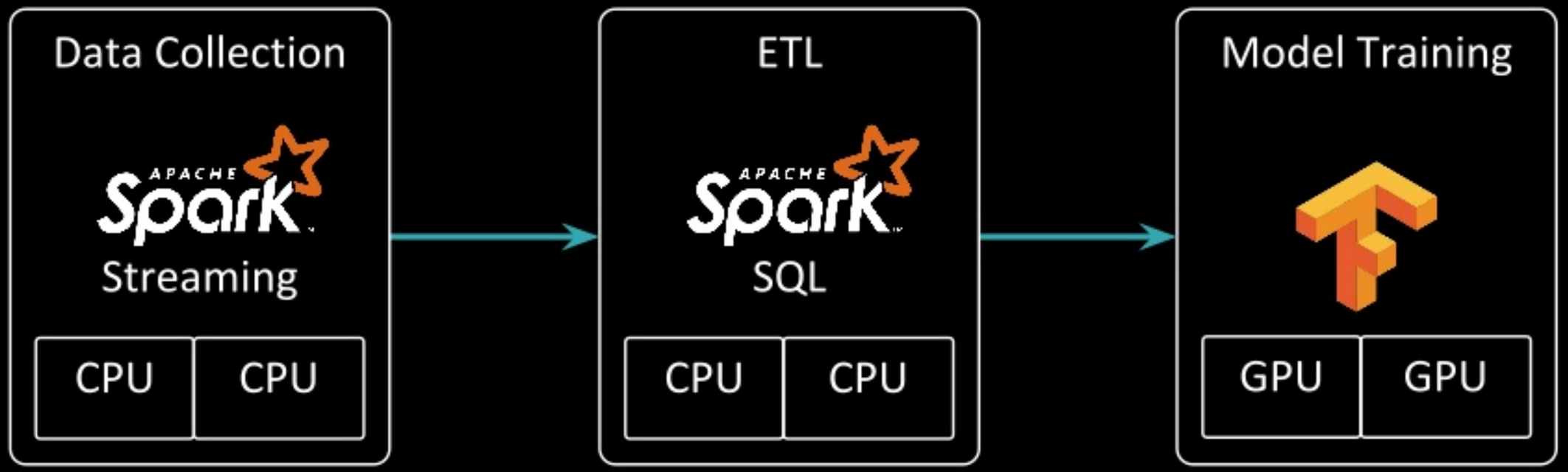
Let's put our models to production!

- Predict from cli
- Set up a real time REST API prediction server
- Score big data in Spark

Advanced MLFlow - HyperParameters



Advanced MLFlow - Multistep Workflow



Get started with MLflow

`pip install mlflow` to get started

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

tinyurl.com/mlflow-slack

Ongoing MLflow Roadmap

- TensorFlow, Keras, PyTorch, H2O, MLlib integrations ✔
- Java and R language APIs ✔
- Multi-step workflows
- Hyperparameter tuning
- Data source API based on Spark data sources
- Model metadata & management

Conclusion

Workflow tools can greatly simplify the ML lifecycle

- Improve usability for *both* data scientists and engineers
- Same way software dev lifecycle tools simplify development

Learn more about MLflow at mlflow.org

mlflow

Thank you!

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)
- Easy to integrate into existing ML platforms & workflows

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