mlflow

and the Lifecycle of ML & AI

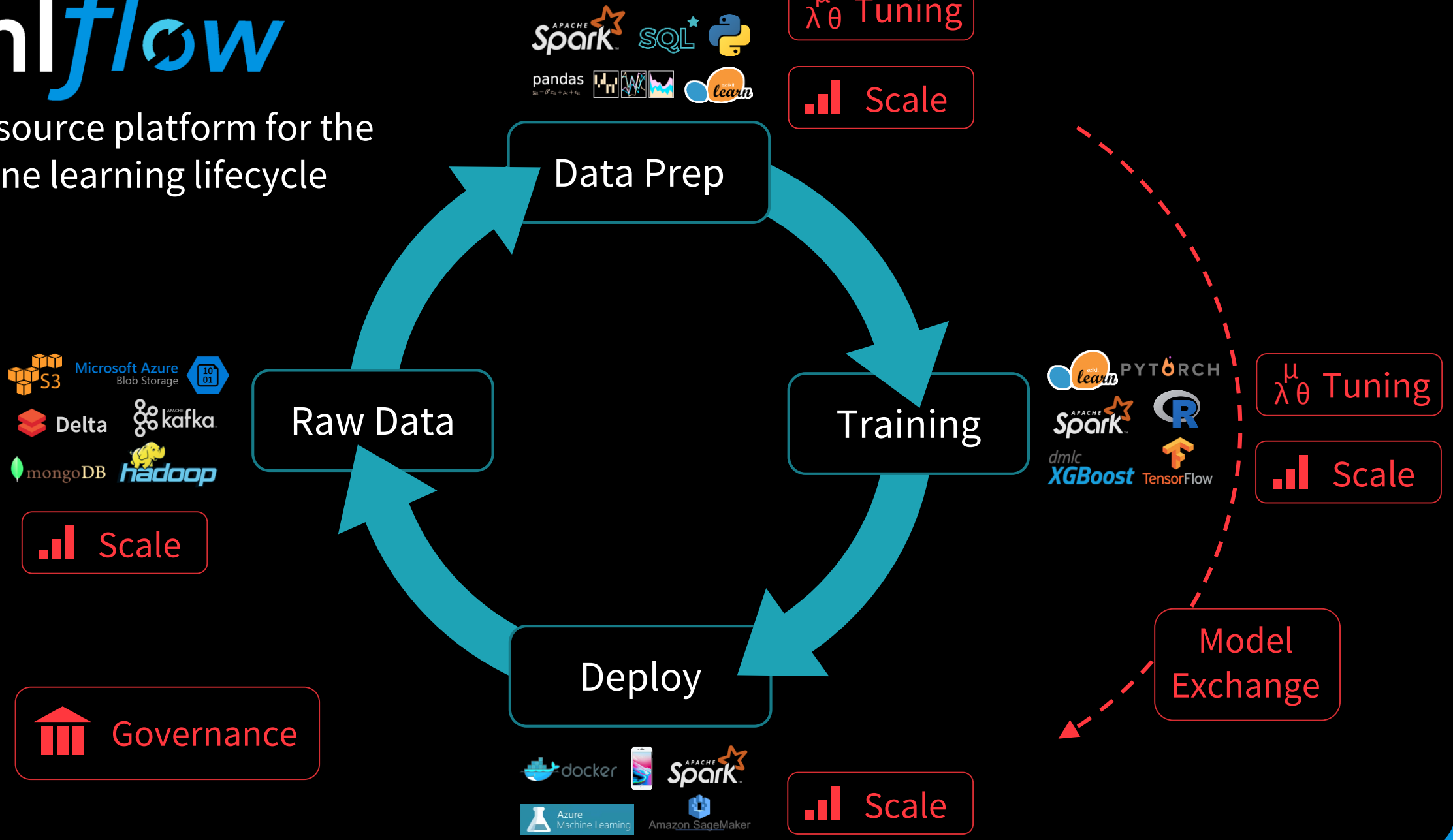
Ben Sadeghi
Databricks Solutions Architect

databricks

Machine Learning Development is Complex

mlflow

An open source platform for the machine learning lifecycle



What is mlflow

Open source project
Conventions, specs, tools
CLI, libraries, REST service
Community

 databricks.com/mlflow

 mlflow.org

 github.com/mlflow

 twitter.com/MLflow

Design Philosophy

API-first

Allow submitting runs, models, etc. from any library & language

E.g., “model” as lambda function that can be deployed anywhere

Key enabler: built around REST APIs,
language libs, CLI

Modular

E.g., use MLflow's project format but not its deployment tools

Easy to integrate into existing ML platforms & workflows

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

Easy

Runs the same way anywhere (local, cloud platforms)
Easy for a single dev to use locally, ***or*** very large teams

Key enabler: minimal, ubiquitous dependencies
Python, pip, Conda, that's it!

Open Source

Everyone is solving a similar problem

Lots of benefits in having a common API across orgs

- Can open source & share individual workflow steps
- ML tool developers can easily reach lots of users
  - E.g. a new ML library can use MLflow Models to reach many serving tools

MLflow Components

mlflow Tracking

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

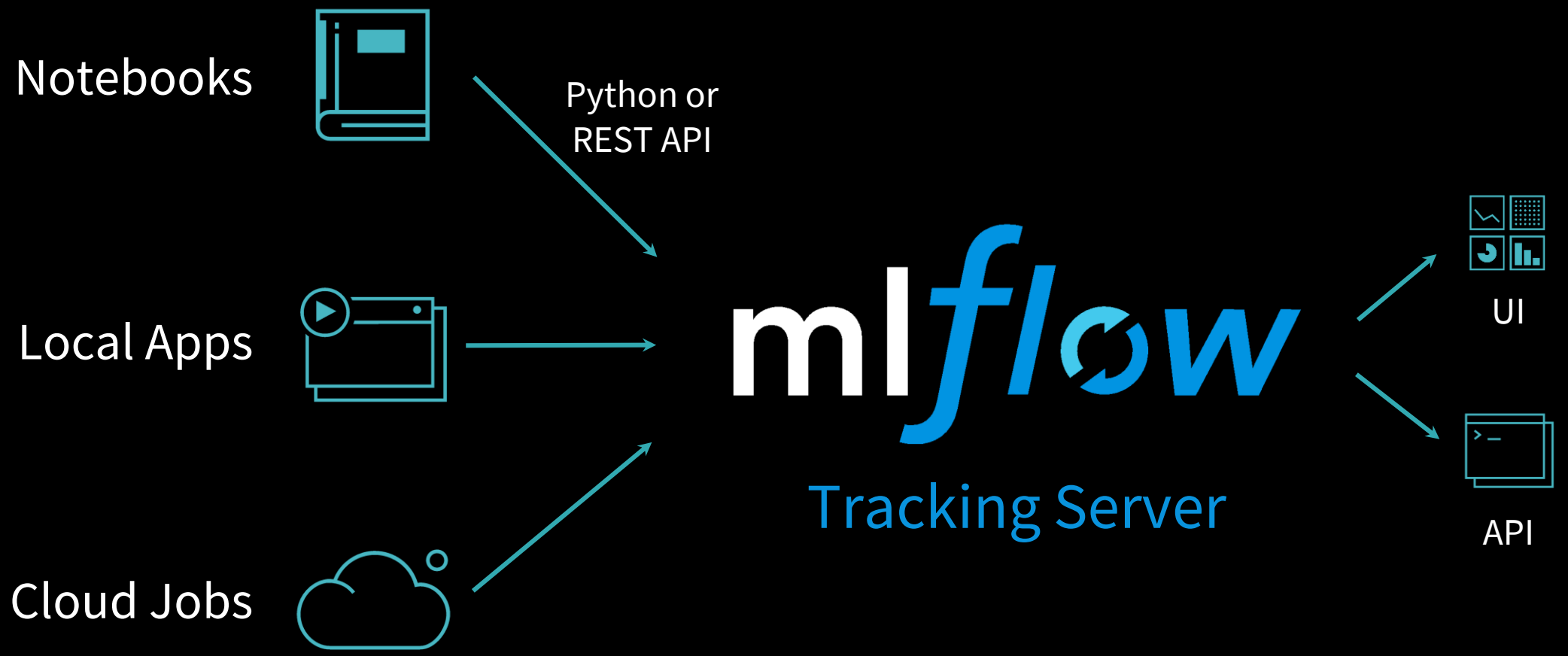
mlflow Projects

Packaging format for reproducible runs on any platform

mlflow Models

General model format that supports diverse deployment tools

MLflow Tracking



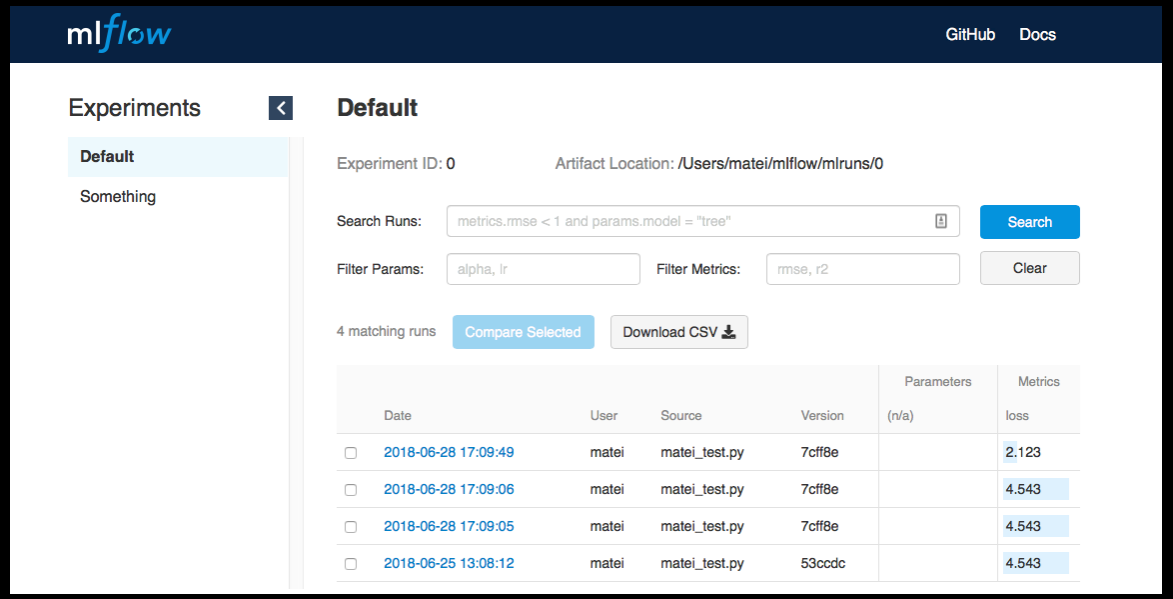
Key Concepts in Tracking

Parameters: key-value inputs to your code

Metrics: numeric values (can update over time)

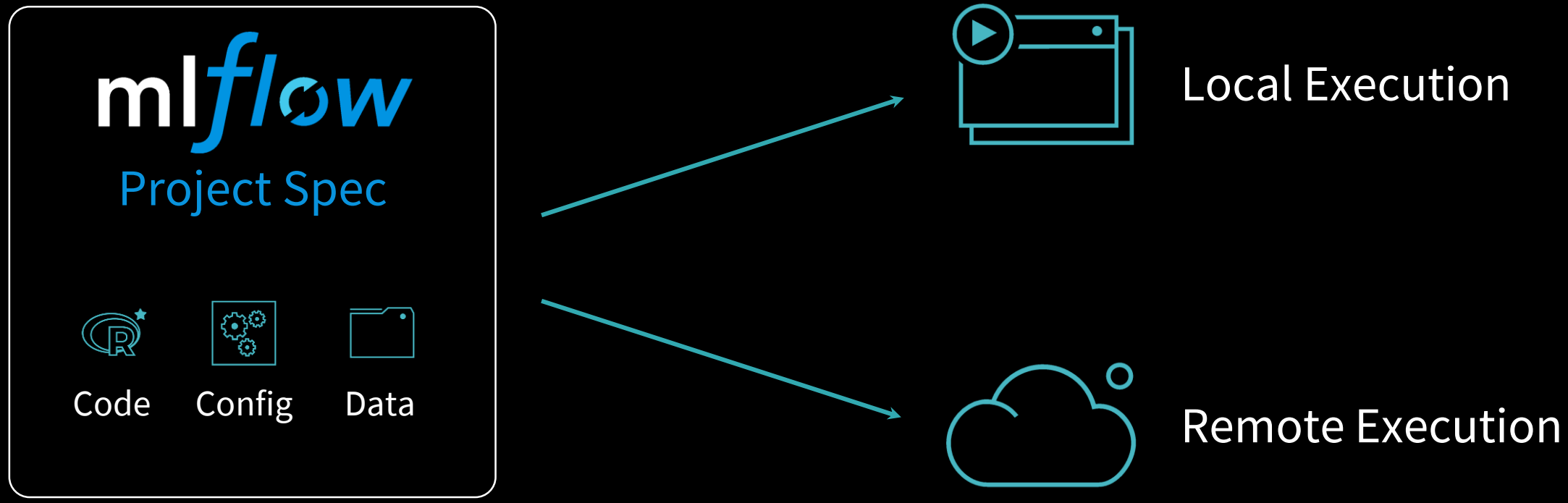
Artifacts: arbitrary files, including models

Source: what code ran?



databricks

MLflow Projects



Example MLflow Project

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

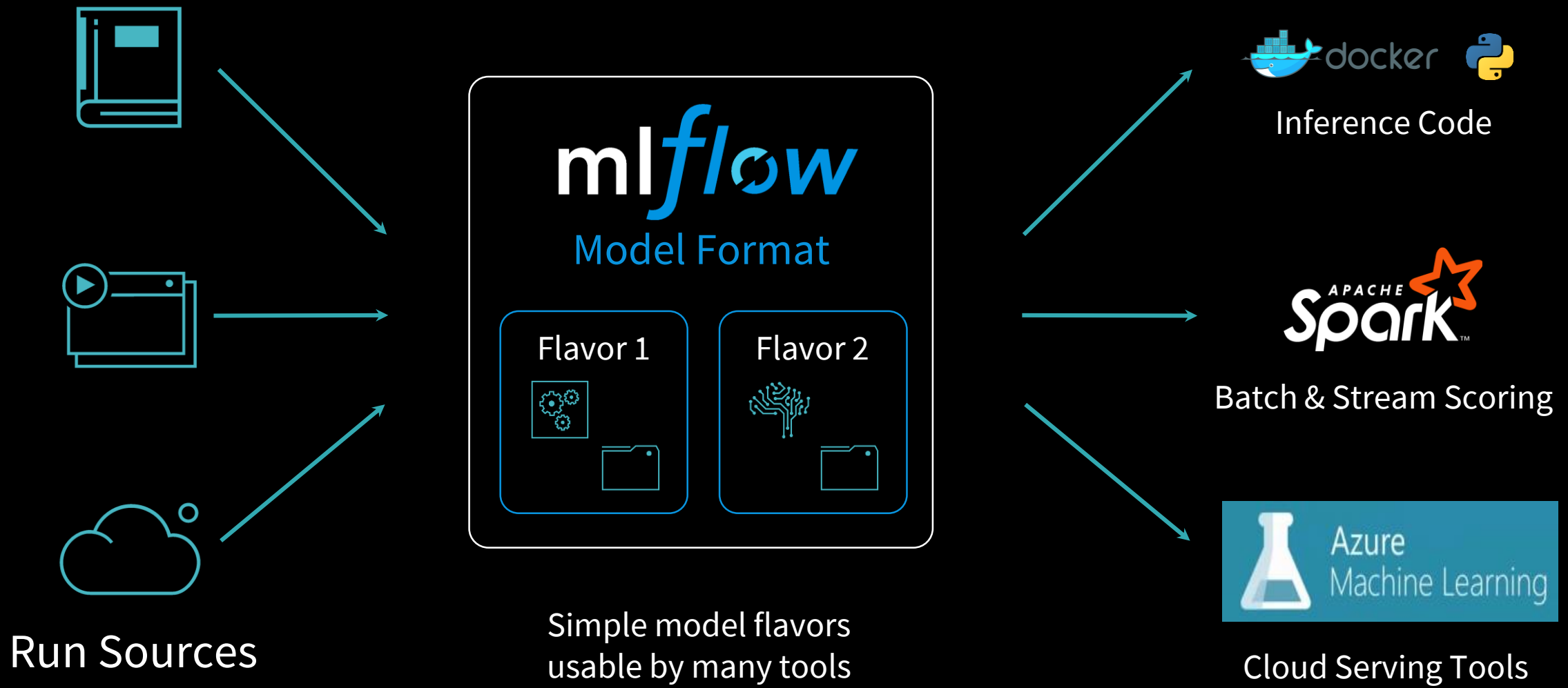
```
conda_env: conda.yaml

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>", ...)
```

databricks

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 Model Deployment Options

In-Line Code

Containers

Batch & Stream Scoring

OSS Inference Solutions

Cloud Inference Services

Conclusion

Powerful workflow tools can simplify the ML lifecycle

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

MLflow is a lightweight, open platform that integrates easily into existing workflows

mlflow

Managed MLflow in Databricks

Demo

Thank you!

/BenSadeghi

LinkedIn, GitHub, Twitter

databricks