mlflow Platform for Complete Machine Learning Lifecycle

Jules S. Damji
@2twitme

San Francisco| May 6, 2020: Part 1 of 3 Series

databricks

Outline – Introduction to MLflow: How to Use MLflow Tracking - Part 1

- Overview of ML development challenges
- Concepts and Motivations
- How MLflow tackles these
- MLFlow Components
  - MLflow Tracking
  - How to use MLflow Tracking APIs
  - Use Databricks Community Edition
  - Explore MLflow UI
  - Tutorials
- Q & A

https://dbricks.co/mlflow-part-1

Machine Learning Development is Complex

Traditional Software vs. Machine Learning

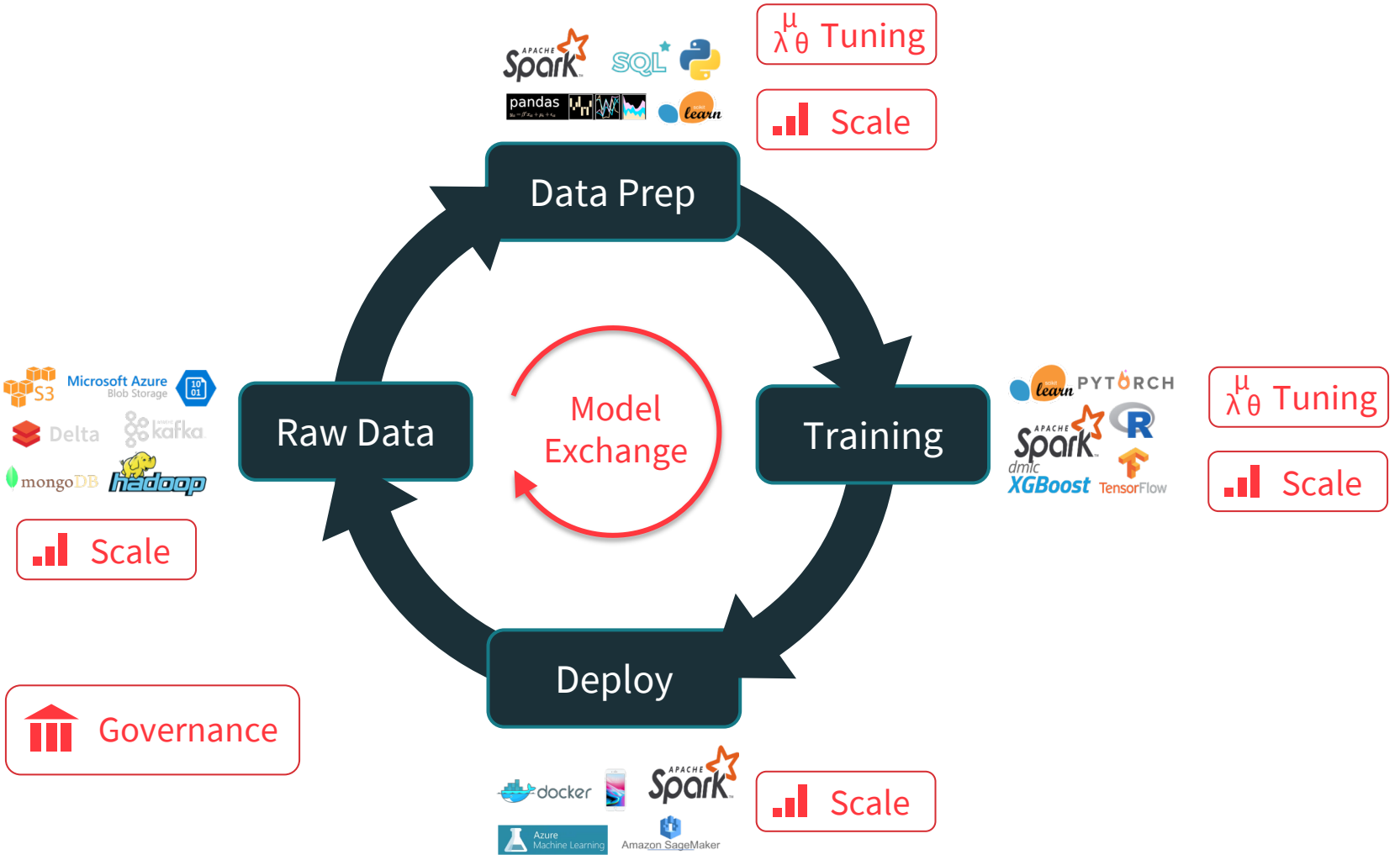
Traditional Software

- Goal: Meet a functional specification
- Quality depends only on code
- Typically pick one software stack w/ fewer libraries and tools
- Limited deployment environments

Machine Learning

- Goal: Optimize metric(e.g., accuracy. Constantly experiment to improve it
- Quality depends on input data and tuning parameters 
- Compare + combine many libraries, model
- Diverse deployment environments

Machine Learning Lifecycle



databricks

Custom Machine Learning Platforms

Some Big Data Companies

+ 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 popular ML library & language
- Runs the same way anywhere (e.g., any cloud or locally)
- Designed to be useful for 1 or 1000+ person orgs
- Simple. Modular. Easy-to-use.
- Offers positive developer experience to get started!

MLflow Design Philosophy

API-First

- Submit runs, log models, metrics, etc. from popular library & language
- Abstract “model” lambda function that MLflow can then deploy in many places (Docker, Azure ML, Spark UDF)
- Open interface allows easy integration from the community

Key enabler: built around Programmatic APIs, REST APIs & CLI

Modular Design

- Allow different components individually (e.g., use MLflow’s project format but not its deployment tools
- Not monolithic
- But Distinctive and Selective

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

MLflow Components

mlflow

Tracking

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

mlflow

Projects

Package data science code in a format that enables reproducible runs on many platform

mlflow

Models

Deploy machine learning models in diverse serving environments

new

mlflow

Model Registry

Store, annotate and manage models in a central repository

databricks.com/mlflow

mlflow.org

github.com/mlflow

twitter.com/MLflow

databricks

Key Concepts in MLflow 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?

Run: an instance of code that runs by MLflow

Experiment: {Run, ... Run}

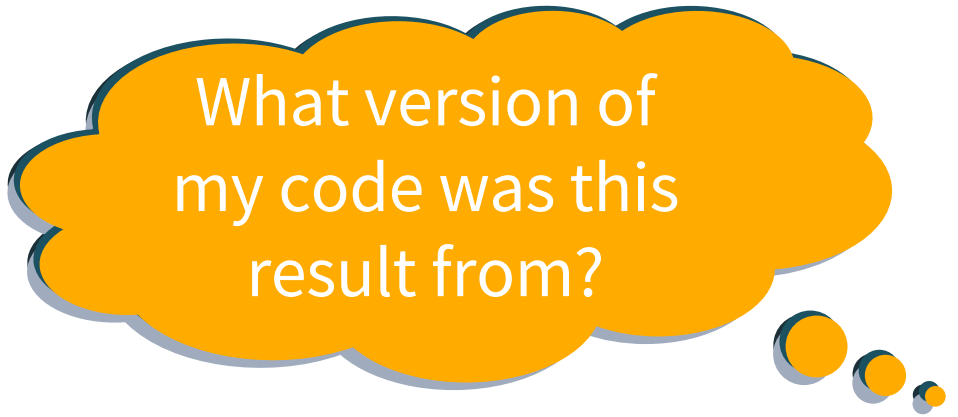
Model Development without MLflow Tracking

```
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"))
```

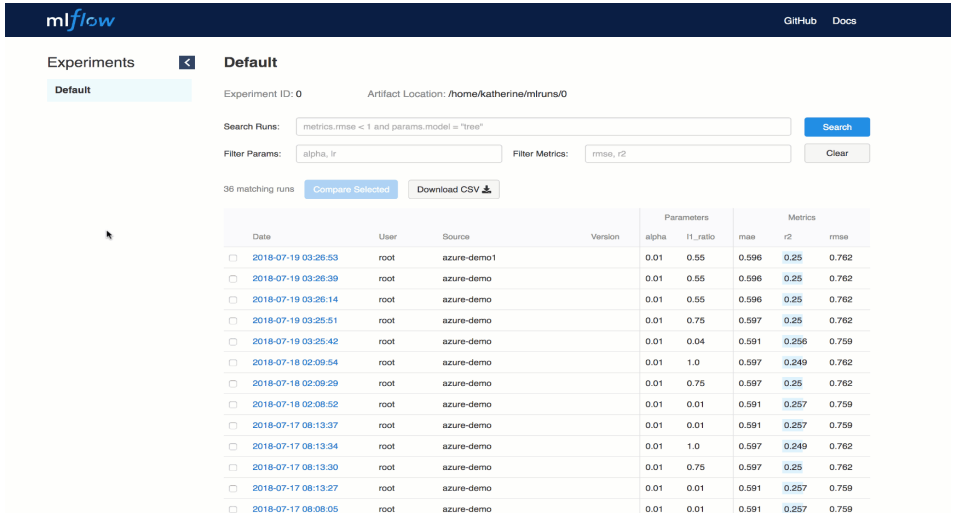
```
For n=2, lr=0.1: accuracy=0.71
For n=2, lr=0.2: accuracy=0.79
For n=2, lr=0.5: accuracy=0.83
For n=2, lr=0.9: accuracy=0.79
For n=3, lr=0.1: accuracy=0.83
For n=3, lr=0.2: accuracy=0.82
For n=4, lr=0.5: accuracy=0.75
...
```



Model Development *with* MLflow is *Simple!*

```
import mlflow
data    = load_text(file)
ngrams  = extract_ngrams(data, N=n)
model   = train_model(ngrams,
                      learning_rate=lr)
score   = compute_accuracy(model)
with mlflow.start_run():
  mlflow.log_param("data_file", file)
  mlflow.log_param("n", n)
  mlflow.log_param("learn_rate", lr)
  mlflow.log_metric("score", score)
  mlflow.sklearn.log_model(model)
```

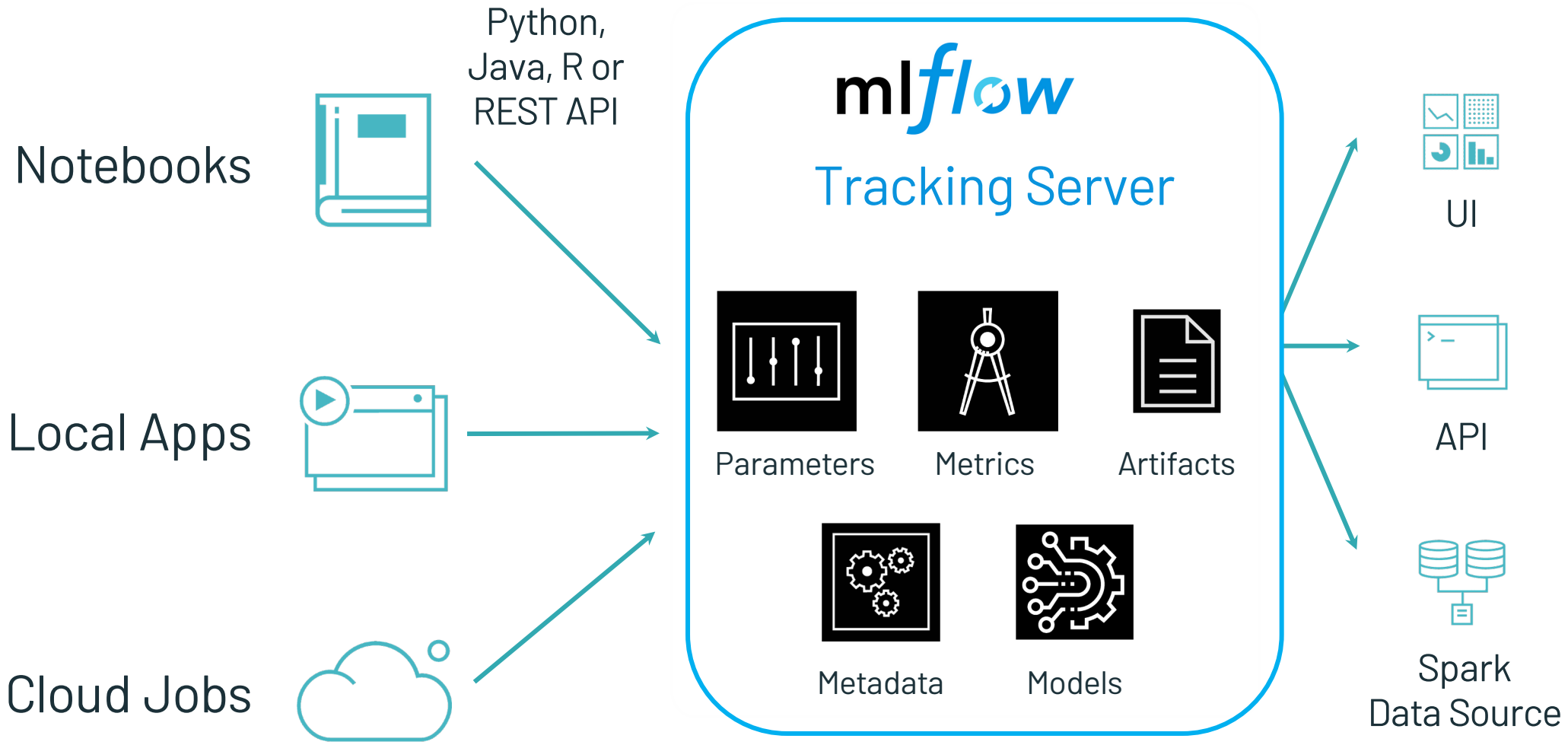
```
$ mlflow ui
```



Track parameters, metrics, output files & code version

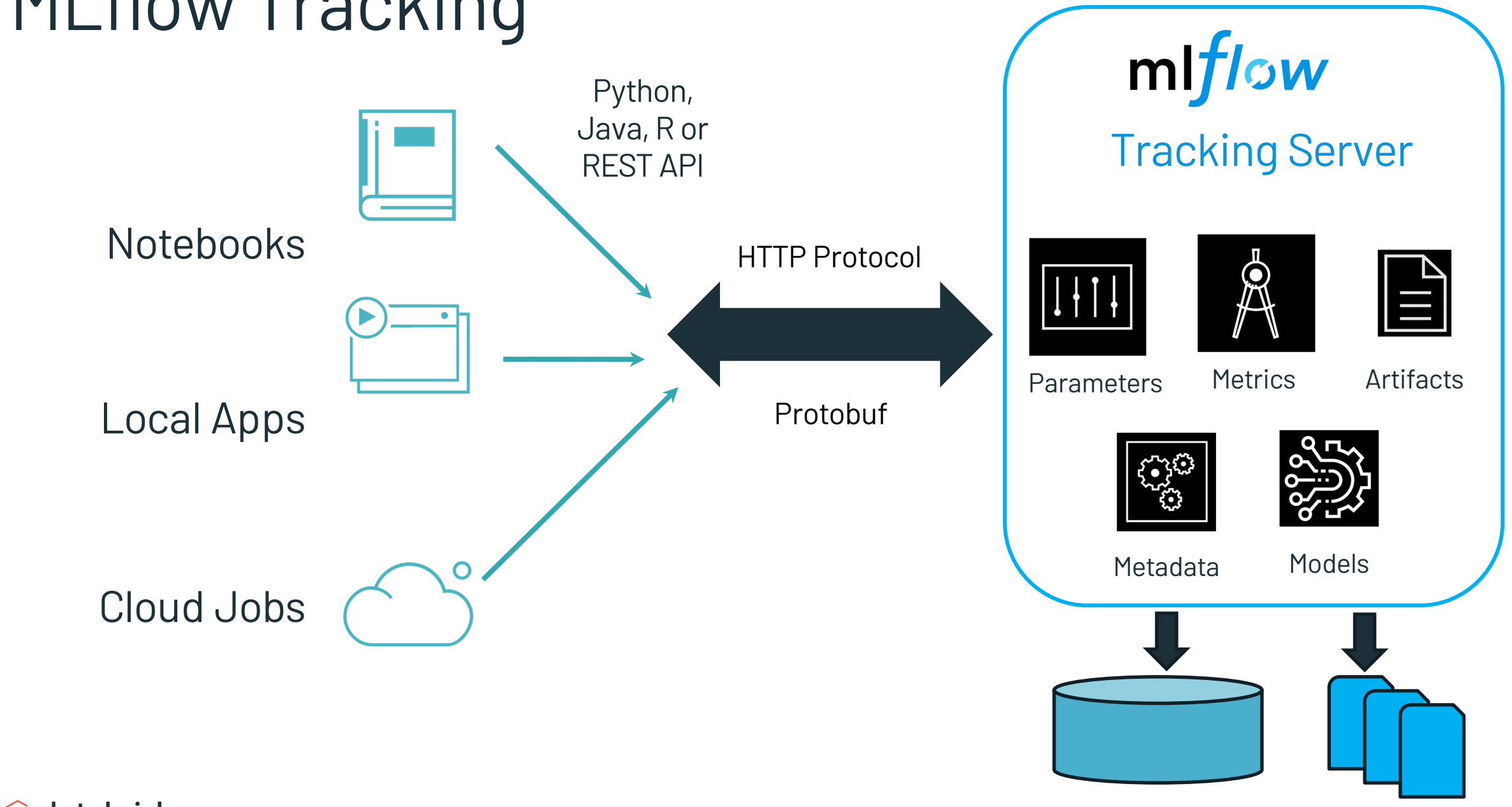
databricks

MLflow Tracking




```
$ export MLFLOW_TRACKING_URI <URI>
mlflow.set_tracking_uri(URI)
```

MLflow Tracking



MLflow Tracking Backend Stores

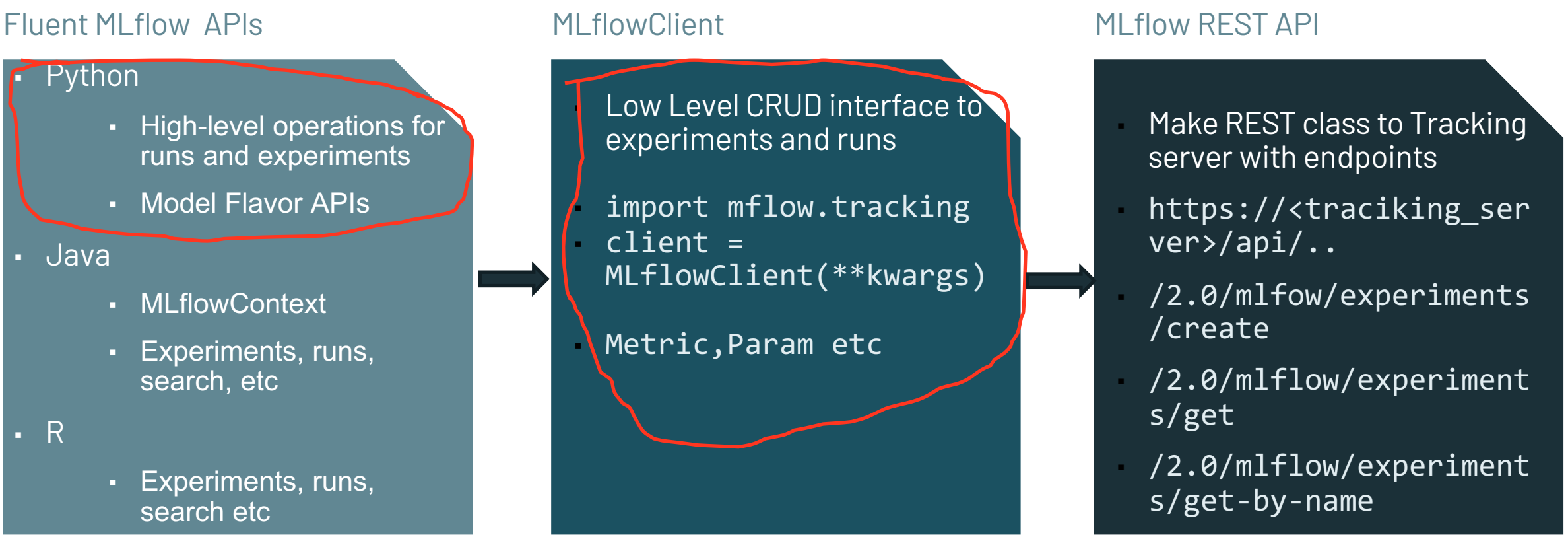
Entity (Metadata) Store

- FileStore (local filesystem)
  - ***mlruns*** directory by default
- SQLStore (via SQLAlchemy)
  - PostgreSQL, MySQL, SQLite
- MLflow Plugins Scheme
  - Customized Entity Metastore
- Managed MLflow on Databricks
  - MySQL on AWS and Azure

Artifact Store

- Local Filesystem
  - ***mlruns*** directory
- S3 backed store
- Azure Blob storage
- Google Cloud Storage
- DBFS artifact repo

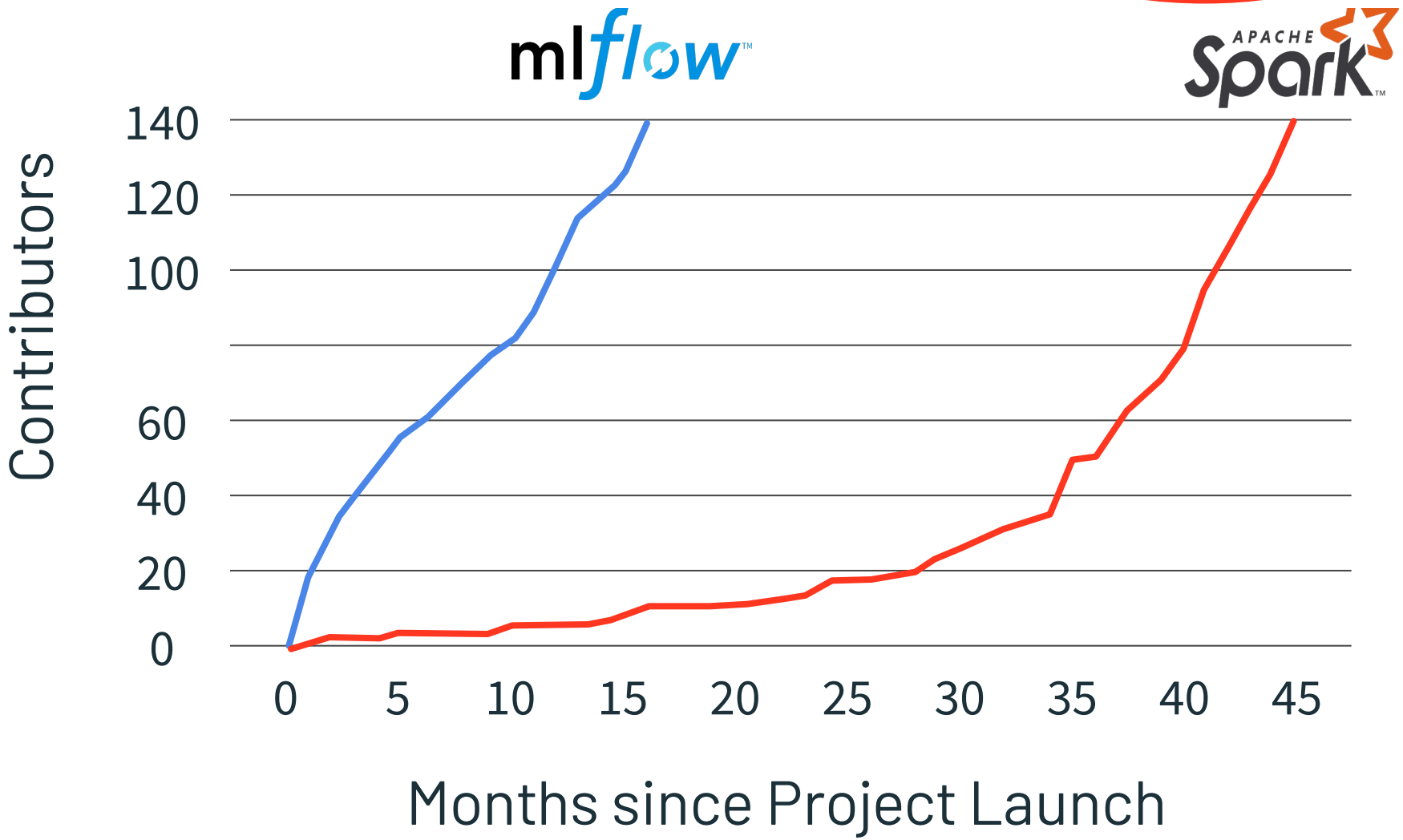
MLFlow set of APIs



What Did We Talk About? mlflow™

- Modular Components greatly simplify the ML lifecycle
- Easy to install and use & Great Developer Experience
- Develop & Deploy locally and track locally or remotely
- Available APIs: Python, Java & R (Soon Scala)
- Visualize experiments and compare runs
- Centrally register and mange model lifecycle

1,363 commits | 37 branches | 0 packages | 28 releases | 200 contributors | Apache-2.0



Learning More About MLflow & Get Involved!

- pip install mlflow–to get started
- Find docs & examples at mlflow.org
- Peruse code and contribute at MLflow Github
- Join the Slack channel
- More MLflow tutorials

MLflow Tracking Tutorials

https://github.com/dmatrix/mlflow-workshop-part-1

Short URL: https://dbricks.co/mlflow-part-1

Thank you! ☺

Q & A

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