



Data + AI  
Online Meetup Group

# mlflow

Platform for Complete Machine Learning Lifecycle

Jules S. Damji  
@2twitme

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

# 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

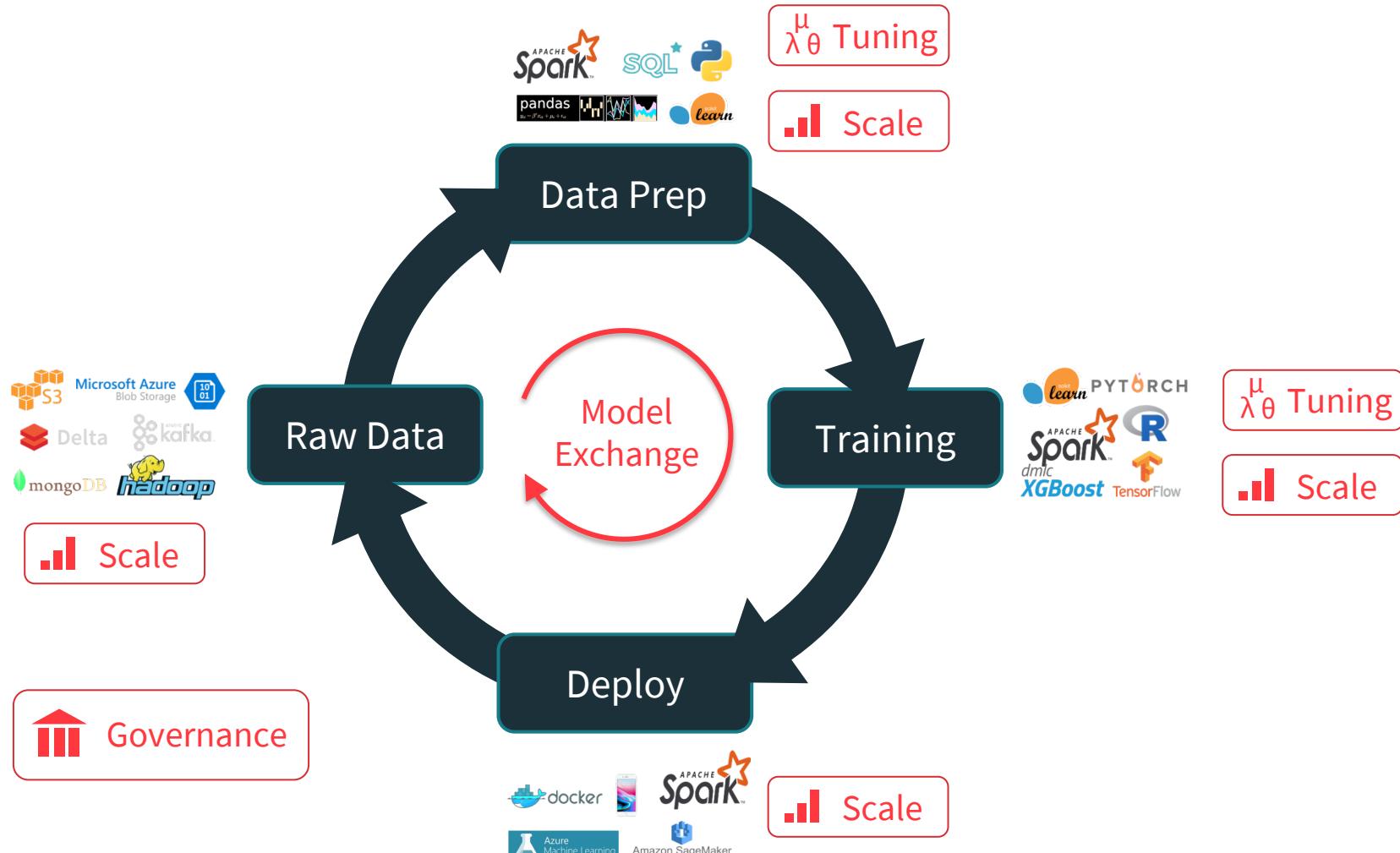
- Goal: Meet a functional specification
- Quality depends only on code
- Typically pick one software stack w/ fewer libraries and tools

## 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



# Machine Learning Lifecycle



# 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 any platform

## mlflow Models

Deploy machine learning models in diverse serving environments environments

new

## mlflow Model Registry

Store, annotate and manage models in a central repository

[databricks.com  
/mlflow](https://databricks.com/mlflow)



[mlflow.org](https://mlflow.org)



[github.com/mlflow](https://github.com/mlflow)



[twitter.com/MLflow](https://twitter.com/MLflow)

# 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?

# 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(
    % (n, lr, score))

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

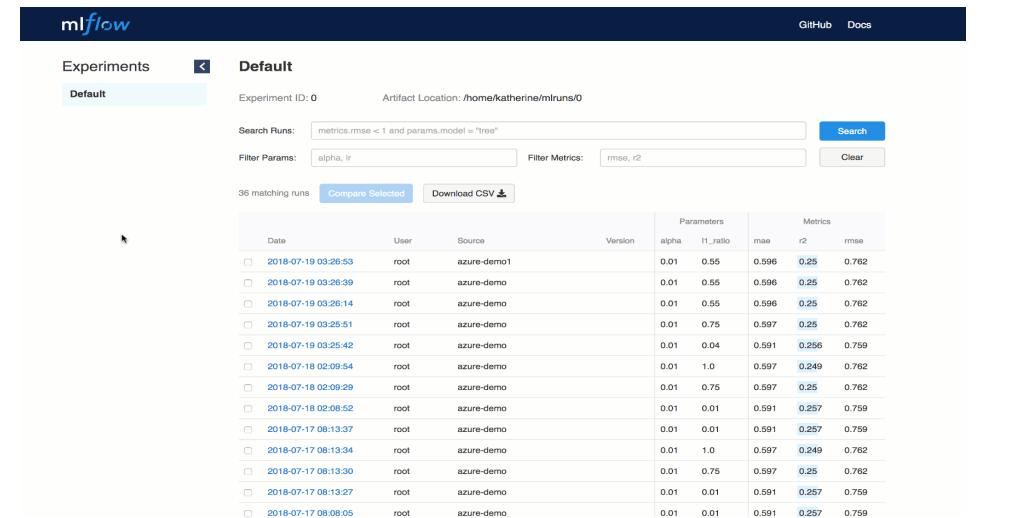
```
For n=2, lr=0.1: accuracy=0.71
For n=2, lr=0.2: accuracy=0.79
For n=2, lr=0.5: accuracy=0.83
For n=2, lr=0.9: accuracy=0.79
For n=3, lr=0.1: accuracy=0.83
For n=3, lr=0.2: accuracy=0.82
For n=4, lr=0.5: accuracy=0.75
...
```

What version of  
my code was this  
result from?

# 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)
with mlflow.start_run() as 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)
```

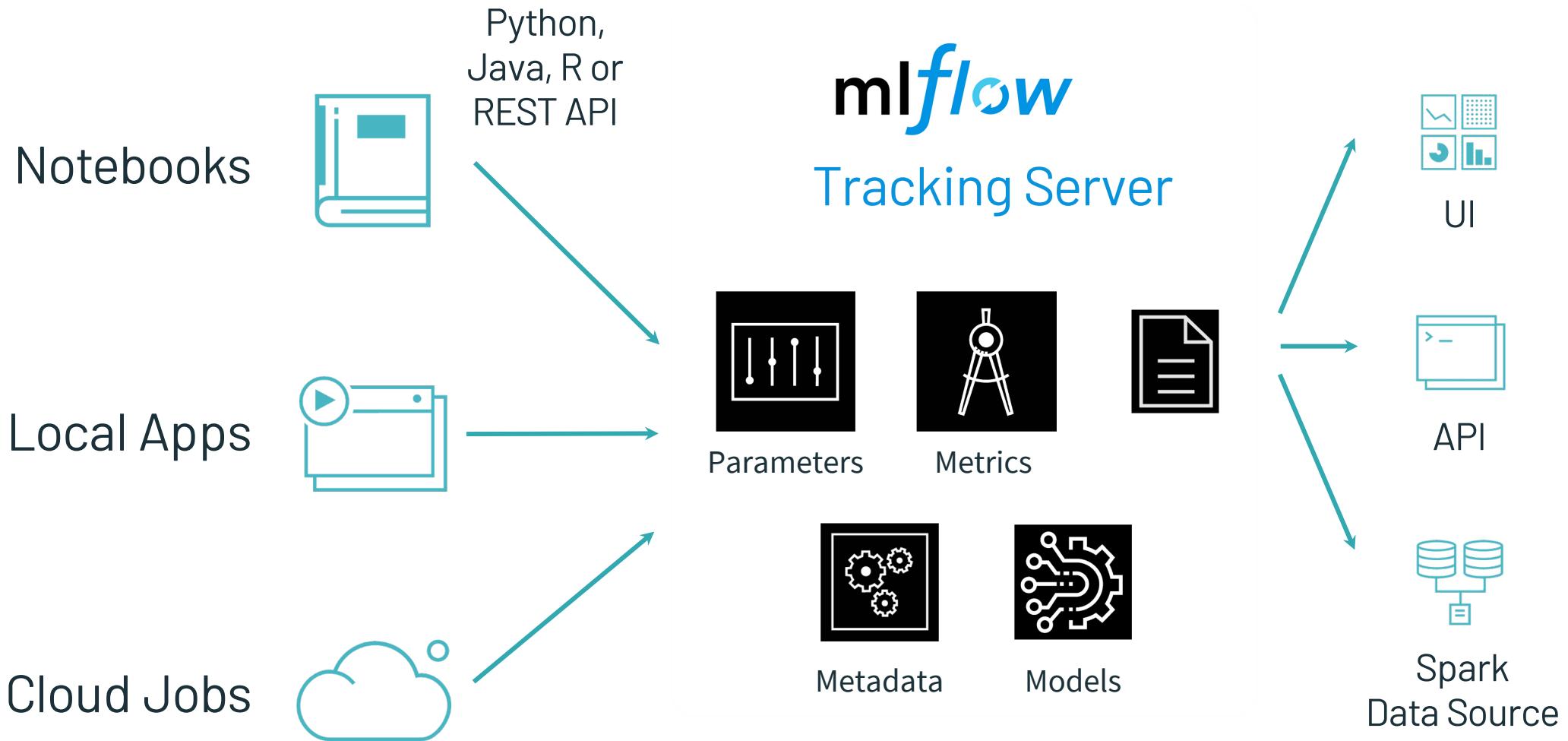
```
$ mlflow ui
```



The screenshot shows the MLflow UI interface. At the top, there's a search bar with the query "metrics.rmse < 1 and params.model = 'tree'". Below the search bar, there are two filter inputs: "Filter Params: alpha, lr" and "Filter Metrics: rmse, r2". A "Clear" button is also present. The main area displays a table of 36 matching runs. The columns in the table are Date, User, Source, Version, Parameters, and Metrics. The Parameters column includes alpha and l1\_ratio, while the Metrics column includes rmse, r2, and rmse.

| Date                | User | Source      | Version | Parameters   | Metrics                 |
|---------------------|------|-------------|---------|--------------|-------------------------|
| 2018-07-19 03:26:53 | root | azure-demo1 | 0.01    | 0.55<br>0.01 | 0.596<br>0.25<br>0.762  |
| 2018-07-19 03:26:39 | root | azure-demo  | 0.01    | 0.55         | 0.596<br>0.25<br>0.762  |
| 2018-07-19 03:26:14 | root | azure-demo  | 0.01    | 0.55         | 0.596<br>0.25<br>0.762  |
| 2018-07-19 03:25:51 | root | azure-demo  | 0.01    | 0.75         | 0.597<br>0.25<br>0.762  |
| 2018-07-19 03:25:42 | root | azure-demo  | 0.01    | 0.04         | 0.591<br>0.256<br>0.759 |
| 2018-07-18 02:09:54 | root | azure-demo  | 0.01    | 1.0          | 0.597<br>0.249<br>0.762 |
| 2018-07-18 02:09:29 | root | azure-demo  | 0.01    | 0.75         | 0.597<br>0.25<br>0.762  |
| 2018-07-18 02:08:52 | root | azure-demo  | 0.01    | 0.01         | 0.591<br>0.257<br>0.759 |
| 2018-07-17 08:13:37 | root | azure-demo  | 0.01    | 0.01         | 0.591<br>0.257<br>0.759 |
| 2018-07-17 08:13:34 | root | azure-demo  | 0.01    | 1.0          | 0.597<br>0.249<br>0.762 |
| 2018-07-17 08:13:30 | root | azure-demo  | 0.01    | 0.75         | 0.597<br>0.25<br>0.762  |
| 2018-07-17 08:13:27 | root | azure-demo  | 0.01    | 0.01         | 0.591<br>0.257<br>0.759 |
| 2018-07-17 08:08:05 | root | azure-demo  | 0.01    | 0.01         | 0.591<br>0.257<br>0.759 |

# MLflow Tracking



# MLflow Tracking Backend Stores

## Entity (Metadata) Store

- FileStore (local filesystem)
- SQLStore (via SQLAlchemy)
  - PostgreSQL, MySQL, SQLite

## Artifact Store

- S3 backed store
- Azure Blob storage
- Google Cloud Storage
- DBFS artifact repo

# What Did We Talk About?

Modular Components greatly simplify the ML Lifecycle

- Open machine learning platform
- Available APIs: Python, Java and R  
(Soon Scala)

Developer Experience

- API-First and Modular design
- Simple. Easy-to-use
- Offers positive developer experience to get started

# Learning More About MLflow

- pip install mlflow to get started
- Find docs & examples at [mlflow.org](https://mlflow.org)
- Peruse code at [MLflow Github](https://github.com/mlflow/mlflow)
- Join the [Slack channel](#)
- More [MLflow tutorials](#)

# MLflow Tracking Tutorials

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

# Thank you! 😊

## Q & A

[jules@databricks.com](mailto:jules@databricks.com)  
@2twitme

<https://www.linkedin.com/in/dmatrix/>