



Data + AI
Online Meetup Group

mlflow

Platform for Complete Machine
Learning Lifecycle

Jules S. Damji
@2twitme

San Francisco | May 20, 2020: Part 3 of 3 Series

Outline – Introduction to MLflow: Model Registry Workflows Explained – Part 3

- Review & Recap Part 2: MLflow Projects & Models
- Concepts and Motivations
- MLFlow Component
 - MLflow Model Registry
 - Model Registry UI & API Workflow
 - Managed MLflow Model Registry Demo
 - Tutorials on local host
- Q & A

<https://dbricks.co/mlflow-part-3>

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



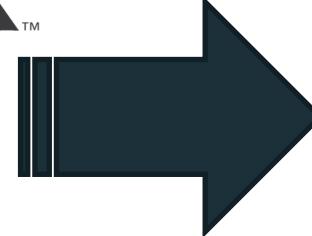
github.com/mlflow



twitter.com/MLflow

MLflow Projects Motivation

Diverse set of tools



Diverse set of environments

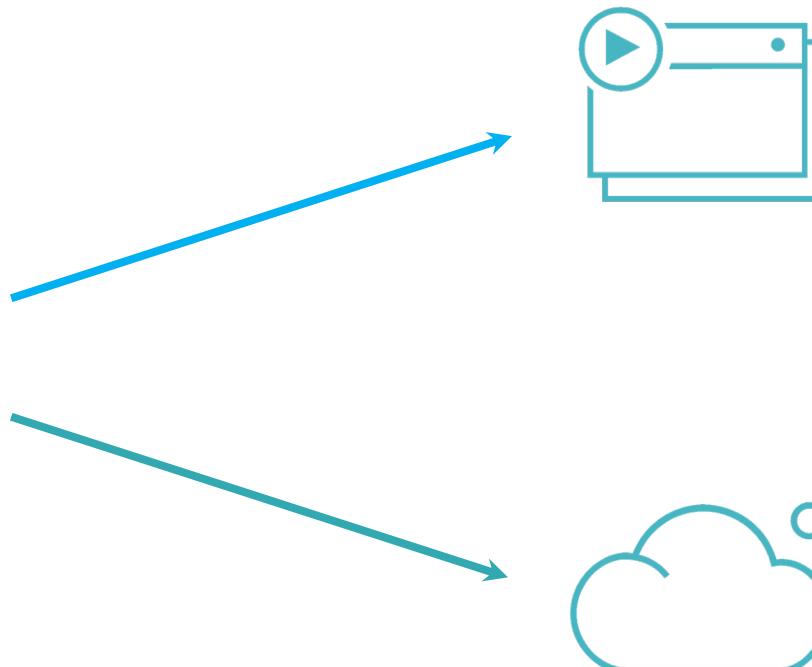
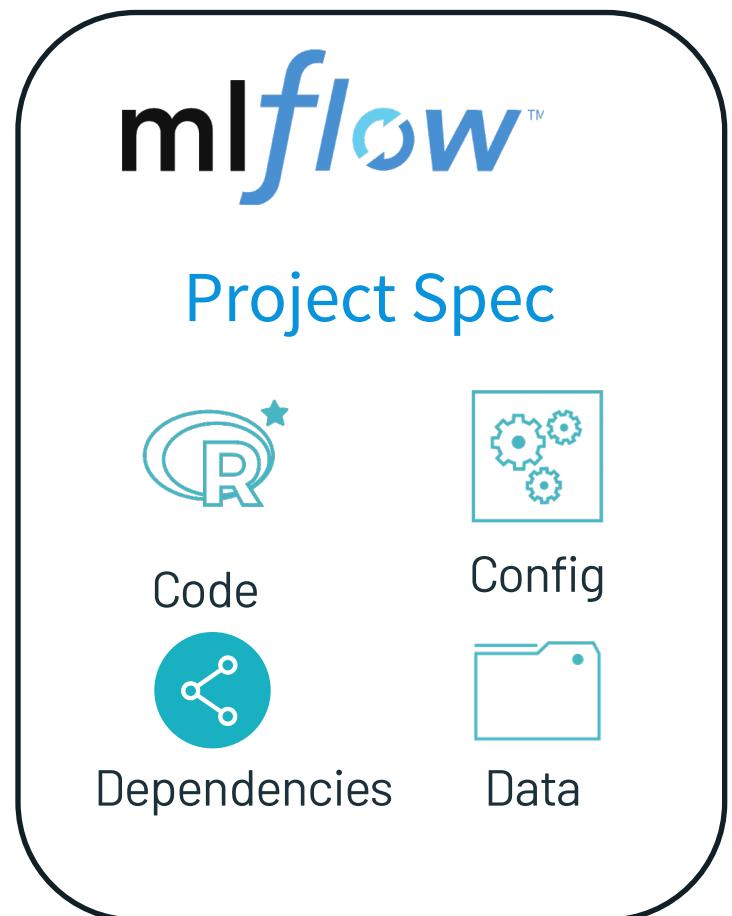


mlflow™
Projects

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

Challenge: ML results difficult to reproduce

MLflow Projects



Local Execution



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 run . -e main -P lambda=0.2
```

Example

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

```

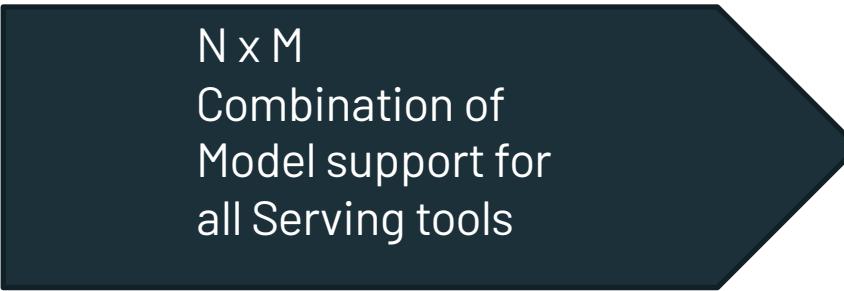
```
channels:
  - defaults
dependencies:
  - python=3.7.3
  - scikit-learn=0.20.3
  - pip:
    - mlflow
    - cloudpickle==0.8.0
name: mlflow-env
```

MLflow Model Motivations



dmlc
XGBoost

ML Frameworks



N x M
Combination of
Model support for
all Serving tools



Inference Code

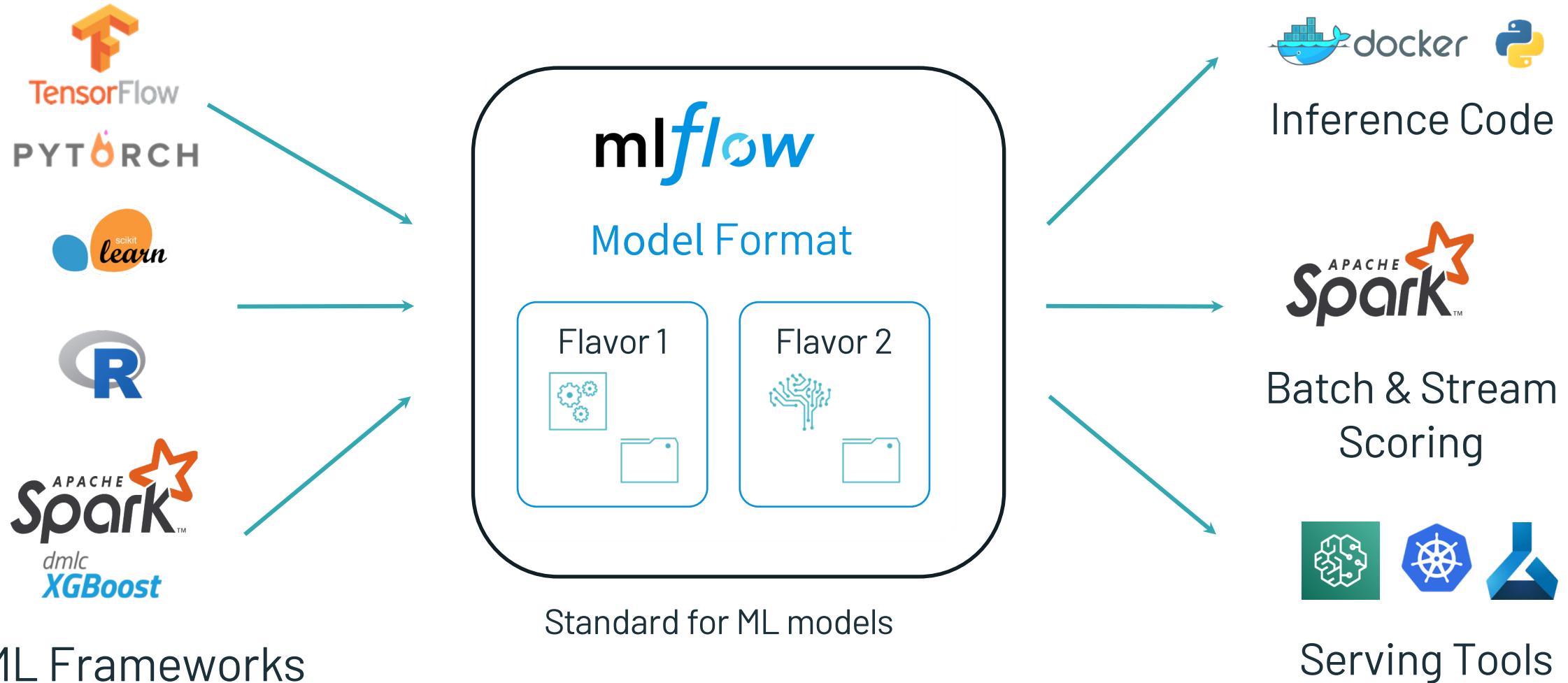


Batch & Stream Scoring



Serving Tools

MLflow Models



Example MLflow Model

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

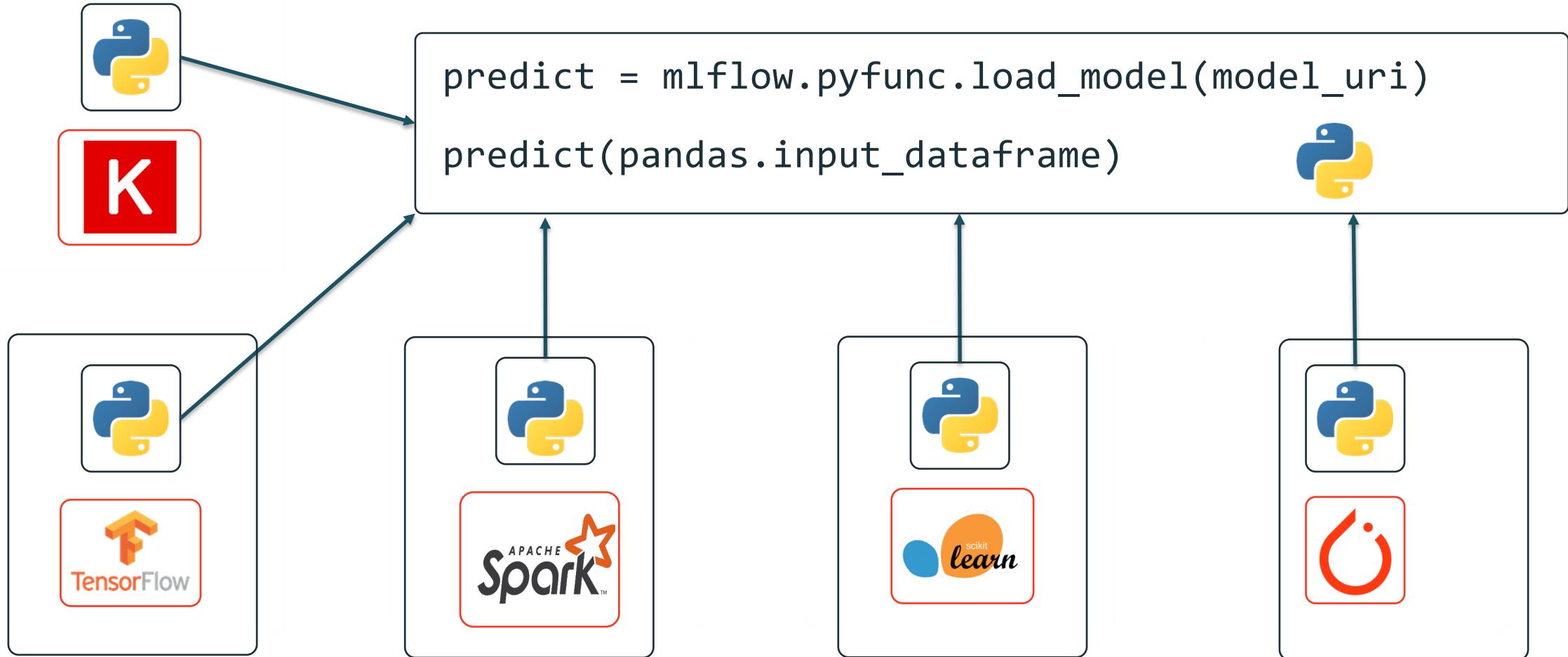
```
my_model/  
└── MLmodel
```

```
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
```

```
estimator/  
└── saved_model.pb  
variables/  
...
```

} Usable by tools that understand TensorFlow model format
} Usable by any tool that can run Python (Docker, Spark, etc!)

Model Flavors Example



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



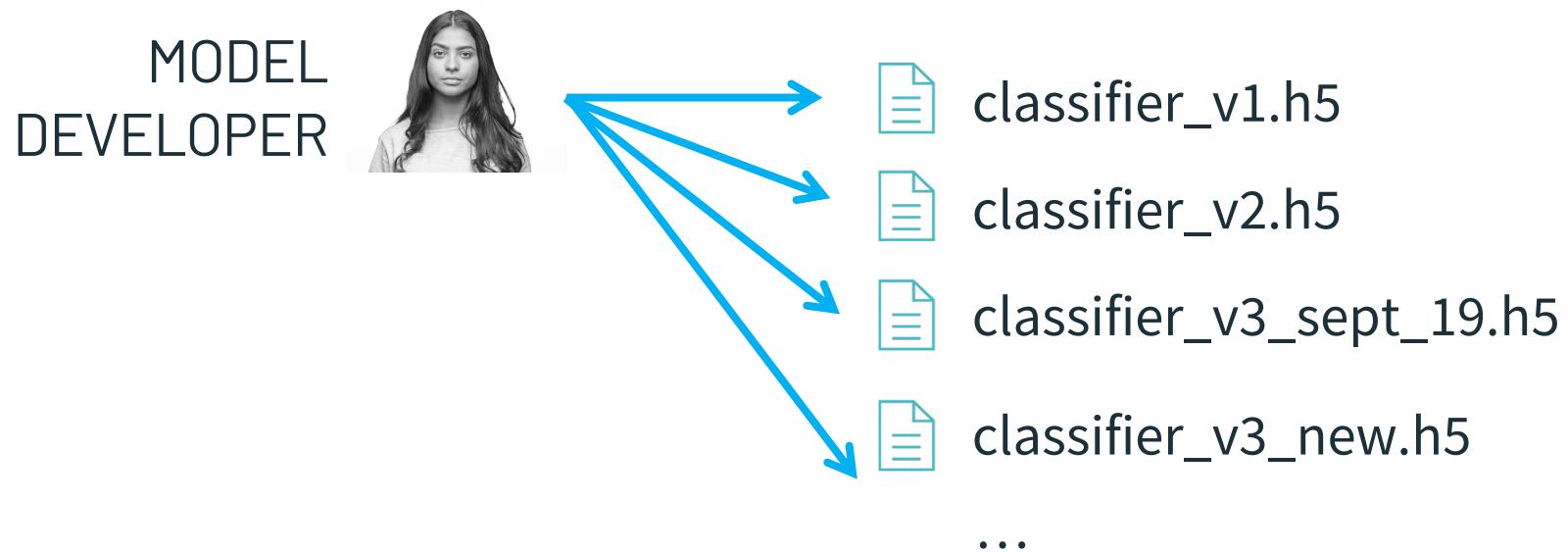
github.com/mlflow



twitter.com/MLflow

The Model Management Problem

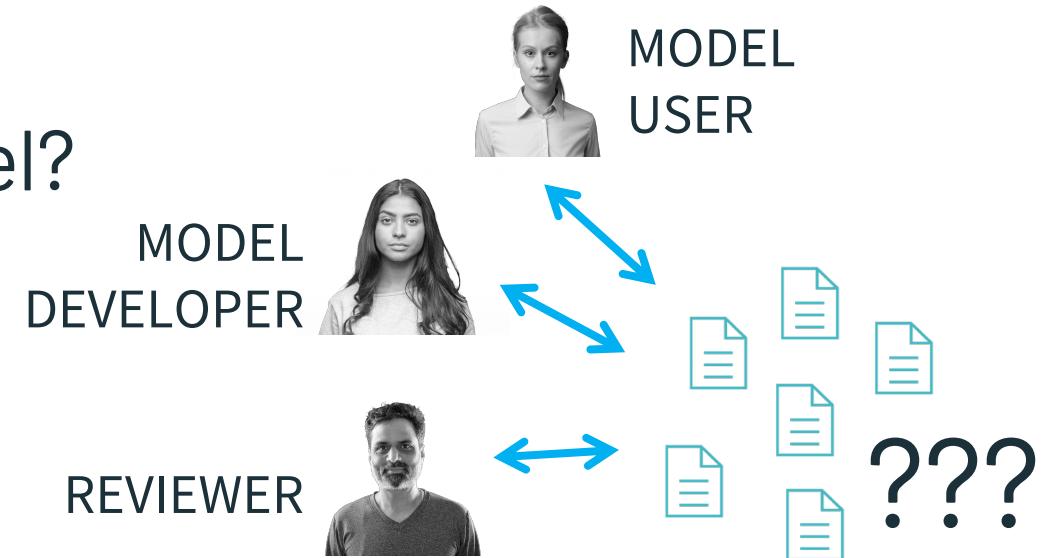
When you're working on one ML app alone, storing your models in files is manageable



The Model Management Problem

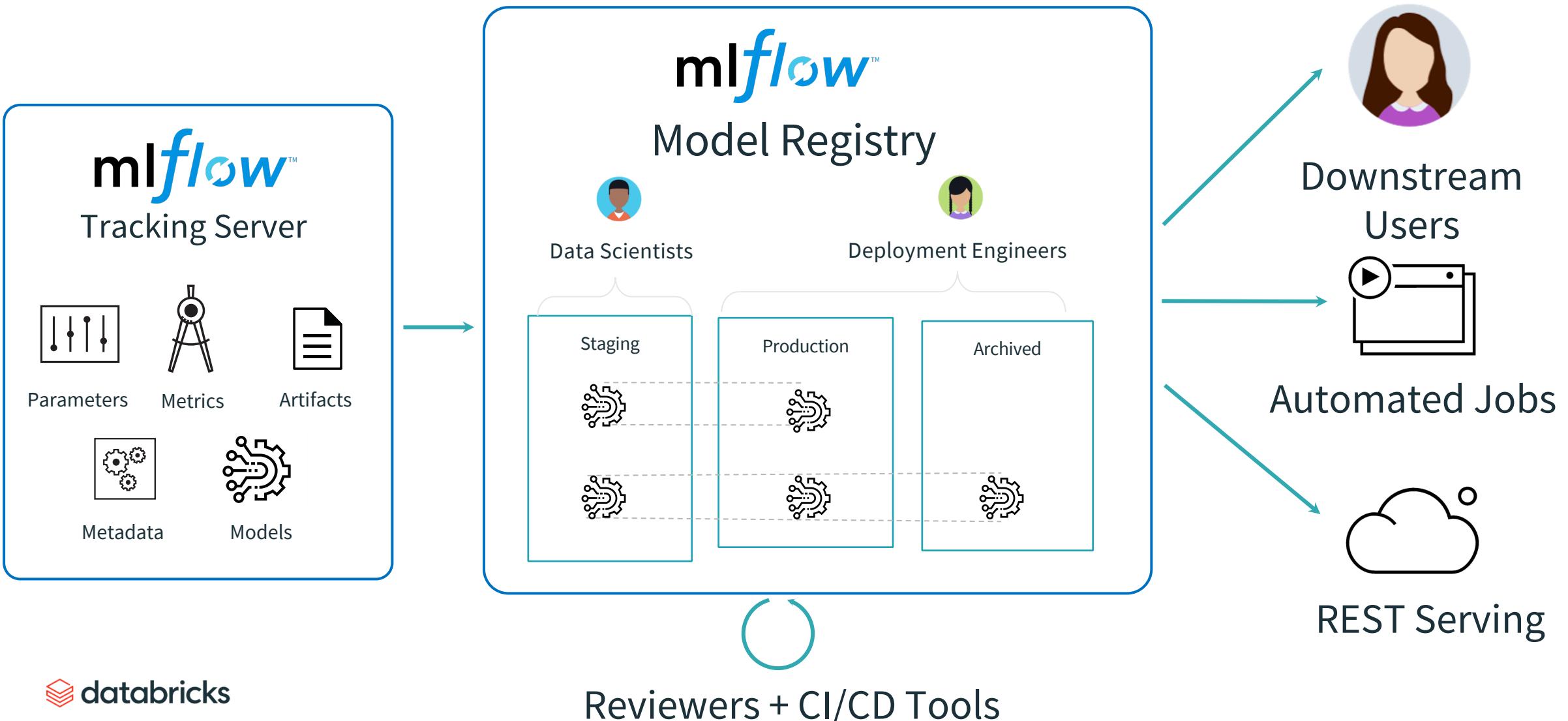
When you work in a large organization with many models, many data teams, management becomes a major challenge:

- Where can I find the best version of this model?
- How was this model trained?
- How can I track docs for each model?
- How can I review models?



mlflow™ Model Registry

VISION: Centralized and collaborative model lifecycle management



MLflow Model Registry

- Repository of named, versioned models with comments & tags
- Track each model's stage: none, staging, production, or archived
- Easily inspect a specific version and its run info
- Easily load a specific version
- Provides Model description, lineage & activities

Registered Models > Airline_Delay_SparkML ▾

Created Time: 2019-10-10 15:20:29 Last Modified: 2019-10-14 12:17:04

▼ Description 

Predicts airline delays (in minutes) using the best Spark RF model from the AutoML Toolkit.

▼ Versions  

| Version | Registered at | Created by | Stage |
|---|---------------------|------------------|------------|
|  Version 1 | 2019-10-10 15:20:30 | clemens@demo.com | Archived |
|  Version 2 | 2019-10-10 21:47:29 | clemens@demo.com | Archived |
|  Version 3 | 2019-10-10 23:39:43 | clemens@demo.com | Production |
|  Version 4 | 2019-10-11 09:55:29 | clemens@demo.com | None |
|  Version 5 | 2019-10-11 12:44:44 | matei@demo.com | Staging |

Model Registry Workflow UI

This screenshot shows the initial step of registering a model. On the left, there's a sidebar labeled "Artifacts" with a single folder named "sklearn-model" containing three files: "MLmodel", "conda.yaml", and "model.pkl". The main area displays a message: "Select a file to preview" with "Supported formats: image, text, html, geojson files". At the top right is a blue "Register Model" button. A large green arrow points from this screen to the next one.

This screenshot shows the "Register Model" dialog box open. It has two tabs: "Model" and "Tags". Under "Model", there's a dropdown menu set to "+ Create New Model" and a text input field where "SKLearnPowerForecast" is typed. Under "Tags", there's a table with columns "Name" and "Value", showing "No tags found." A small "Add Tag" button is visible. At the bottom right of the dialog are "Cancel" and "Register" buttons. A large green arrow points from the previous screen to this one.

This screenshot shows the "Register Model" dialog box open on a different screen. The interface is identical to the previous one, with the "Model" tab active and the "Model Name" field containing "SKLearnPowerForecast". A large green arrow points from the previous screen to this one.

This screenshot shows the final step where the registered model is listed. The "Artifacts" sidebar shows the "sklearn-model" folder with its contents. The main area displays the registered model details: "Full Path: ./mirluns/0/55eb2ad528114c68bd354a0568eca327/artifacts/sklearn-model", "Size: 0B", and "SKLearnPowerForecast, v1" with a timestamp "Registered on 2020/05/04". Below this, there's a "Select a file to preview" section with "Supported formats: image, text, html, geojson files". A large green arrow points from the previous screen to this one.

Model Registry Workflow UI

[mlflow](#) GitHub Docs

Registered Models > SKLearnPowerForecast > Version 1 ▾

Registered At: 2020-05-04 11:38:47 Creator: Stage: None ▾

Last Modified: 2020-05-04 11:38:47 Source Run: [Random Forest Regressor: Power Forecasting Model](#)

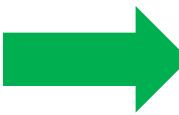
▼ Description

[Write](#) Preview [H](#) [B](#) [I](#) [S](#) [♂](#) [”](#) [↶](#) [☰](#) [≡](#) [☰](#) [☰](#)

This version forecasts the power

...

[Save](#) [Cancel](#)



mlflow

GitHub Docs

Registered Models > SKLearnPowerForecast > Version 1 ▾

| | | | | | | | | | | |
|--|--|-------------|--|--|---------------|--|---------------|--|---------------|--|
| Registered At: | 2020-05-04 11:38:47 | Creator: | Stage: | <input type="button" value="None"/> ▾ | | | | | | |
| Last Modified: | 2020-05-04 11:38:47 | Source Run: | Random Forest Regressor: Power Forecasting Model | <table><tbody><tr><td>Transition to</td><td>→ Staging</td></tr><tr><td>Transition to</td><td>→ Production</td></tr><tr><td>Transition to</td><td>→ Archived</td></tr></tbody></table> | Transition to | → Staging | Transition to | → Production | Transition to | → Archived |
| Transition to | → Staging | | | | | | | | | |
| Transition to | → Production | | | | | | | | | |
| Transition to | → Archived | | | | | | | | | |
| <p>▼ Description 🔗</p> <p>This version forecasts the power</p> | | | | | | | | | | |



[mlflow](#) GitHub Docs

Registered Models > SKLearnPowerForecast ▾

Created Time: 2020-05-04 11:38:47 Last Modified: 2020-05-04 11:42:54

▼ Description [🔗](#)

None

▼ Versions [All](#) Active(3) [Compare](#)

| <input type="checkbox"/> | Version | Registered at | Created by | Stage |
|--------------------------|--|---------------------|------------|------------|
| <input type="checkbox"/> | Version 1 ✓ | 2020-05-04 11:38:47 | | Staging |
| <input type="checkbox"/> | Version 2 ✓ | 2020-05-04 11:41:37 | | Staging |
| <input type="checkbox"/> | Version 3 ✓ | 2020-05-04 11:42:07 | | Production |

< 1 >



Model Registry Workflow API

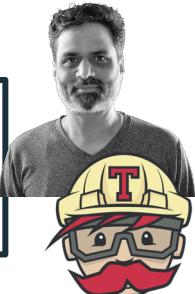
```
mlflow.register_model("runs:/e148278107274..832/artifacts/model",  
    "WeatherForecastModel")
```

```
model_uri= "models:/{{model_name}}/production".format(  
    model_name=" WeatherForecastModel")  
model_prod = mlflow.sklearn.load_model(model_uri)
```

MODEL
DEVELOPER



REVIEWERS,
CI/CD TOOLS



```
client = mlflow.tracking.MlflowClient()  
client.transition_model_version_stage(name="WeatherForecastModel",  
    version=5,  
    stage= "Production")
```



DOWNSTREAM
USERS

AUTOMATED JOBS

REST SERVING

MLflow Backend Registry Stores

Entity (Metadata) Store and Models

- SQLStore (via SQLAlchemy)
 - PostgreSQL, MySQL, SQLite
 - Default is mlruns.db file locally
- Set programmatically for locally
- `mlflow.set_tracking_uri("sqlite:///mlruns.db")`
- `sqlite3 ./mlruns.db` (on local host)

Artifact Store

- Local Filesystem
 - `mlruns` directory
- S3 backed store
- Azure Blob storage
- Google Cloud Storage
- DBFS artifact repo
- Managed MLflow on Databricks
 - MySQL on AWS and Azure

```
(tutorials) ➔ src git:(master) ✘ sqlite3 ./mlruns.db
SQLite version 3.30.1 2019-10-10 20:19:45
Enter ".help" for usage hints.
sqlite> .databases
main: /Users/julesdamji/gits/mlflow-workshop-part-3/src./mlruns.db
sqlite> .tables
alembic_version      latest_metrics      params          tags
experiment_tags       metrics            registered_models
experiments           model_versions     runs
sqlite> █
```

1,363 commits

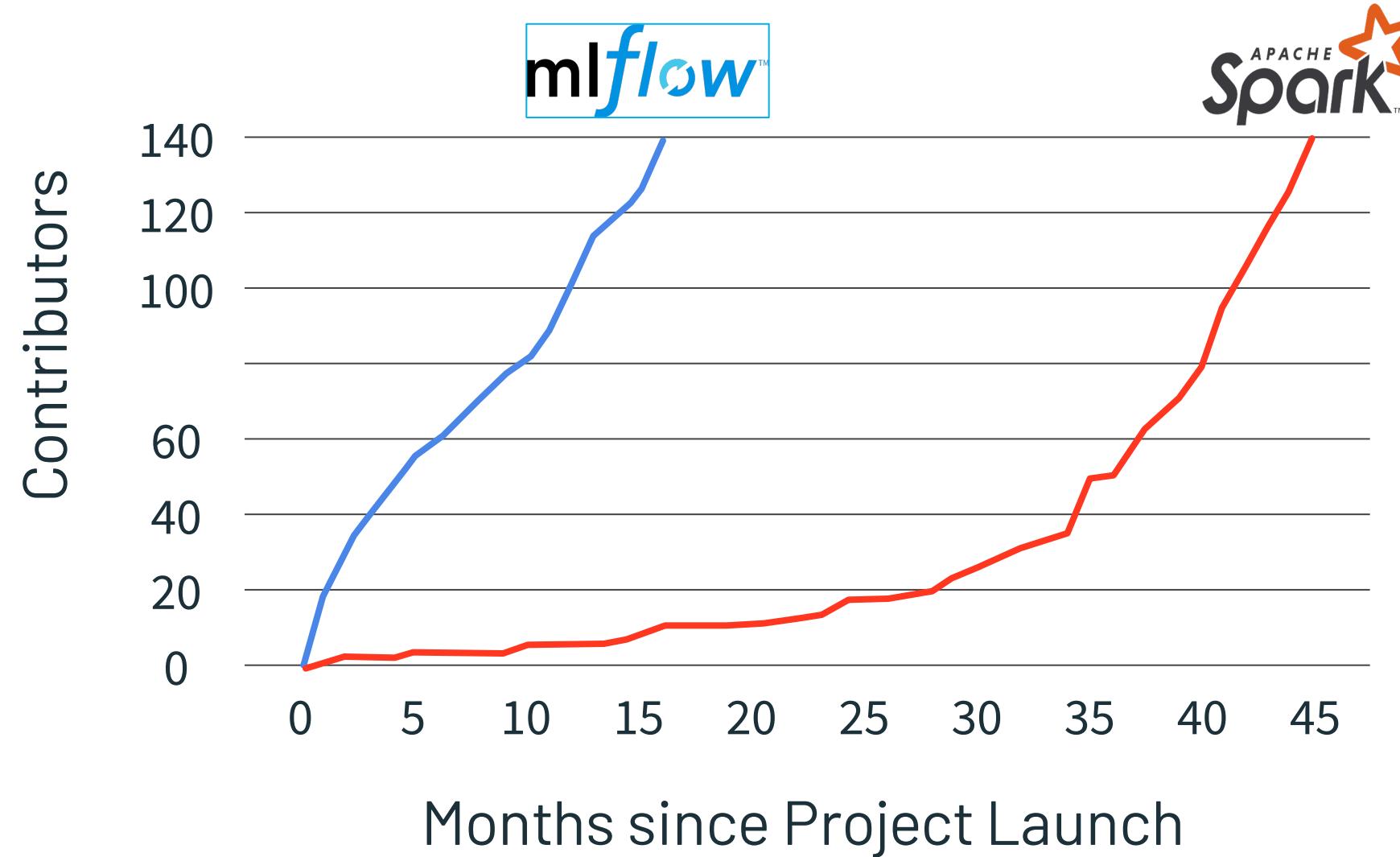
37 branches

0 packages

28 releases

200 contributors

Apache-2.0



What Did We Talk About? **mlflow**TM

- 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 manage model lifecycle

Learning More About MLflow

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

MLflow Model Registry

Tutorials: <https://github.com/dmatrix/mlflow-workshop-part-3>

Thank you! 😊

Q & A

jules@databricks.com
@2twitme

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