

mlflow

Platform for Machine Learning Lifecycle

Jules S. Damji

@2twitme

Outline – Introduction to MLflow: Model Registry Workflows Explained – Module 4

- Model Registry
- Concepts and Motivations
 - MLflow Model Registry
 - Model Registry UI & API Workflow
 - Tutorials on local host
 - Jupyter Lab
- Q & A

<https://github.com/dmatrix/olt-mlflow>

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

mlflow Model Registry

Store, annotate and manage models in a central repository

databricks.com/mlflow



mlflow.org



github.com/mlflow



twitter.com/MLflow

O'REILLY®

The Model Management Problem

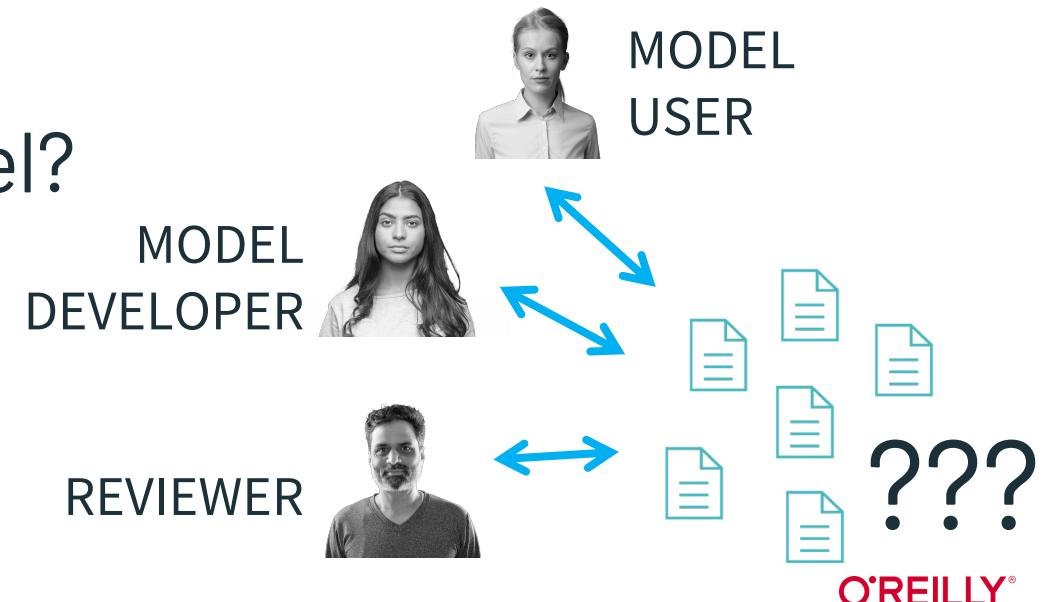
When you're working on one ML app alone, storing your models in files is manageable



The Model Management Problem

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

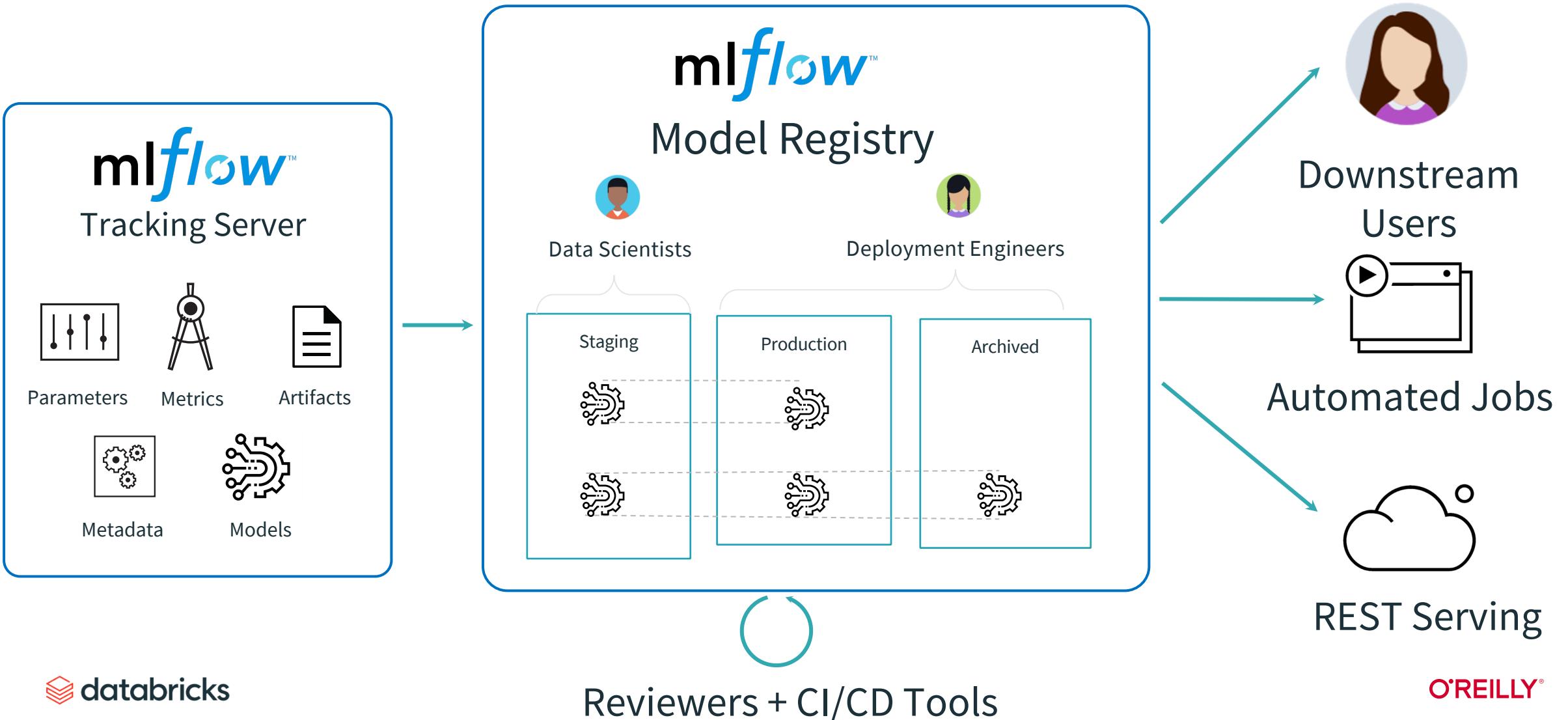
- Where can I find the best version of this model?
- How was this model trained?
- How can I track docs for each model?
- How can I review models?
- How can I integrate with CI/CD?





Model Registry

VISION: Centralized and collaborative model lifecycle management



MLflow Model Registry

- Repository of named, versioned models with controlled Access to Models
- 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 and activities

The screenshot shows the MLflow Model Registry interface for a registered model named "Airline_Delay_SparkML".

Header: 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: All Active(1)

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

MLflow Model Registry

The MLflow Model Registry component is a centralized model store, set of APIs, and UI, to collaboratively manage the full lifecycle of an MLflow Model. It provides model lineage (which MLflow experiment and run produced the model), model versioning, stage transitions (for example from staging to production), and annotations.

Table of Contents

- Concepts
- Model Registry Workflows
 - UI Workflow
 - Registering a Model
 - Using the Model Registry
 - API Workflow
 - Adding an MLflow Model to the Model Registry
 - Fetching an MLflow Model from the Model Registry
 - Serving an MLflow Model from Model Registry
 - Adding or Updating an MLflow Model Descriptions
 - Renaming an MLflow Model
 - Transitioning an MLflow Model's Stage
 - Listing and Searching MLflow Models
 - Archiving an MLflow Model
 - Deleting MLflow Models



`mlflow.register_model(model_uri, name)` [source]

`mlflow.get_registry_uri()` [source]

`mlflow.set_registry_uri(uri)` [source]

MLflow Model Registry

The MLflow Model Registry component is a centralized model store, set of APIs, and UI, to collaboratively manage the full lifecycle of an MLflow Model. It provides model lineage (which MLflow experiment and run produced the model), model versioning, stage transitions (for example from staging to production), and annotations.

Table of Contents

- [Concepts](#)
- [Model Registry Workflows](#)

- [UI Workflow](#)

- [Registering a Model](#)
- [Using the Model Registry](#)

- [API Workflow](#)

- [Adding an MLflow Model to the Model Registry](#)
- [Fetching an MLflow Model from the Model Registry](#)
- [Serving an MLflow Model from Model Registry](#)
- [Adding or Updating an MLflow Model Descriptions](#)
- [Renaming an MLflow Model](#)
- [Transitioning an MLflow Model's Stage](#)
- [Listing and Searching MLflow Models](#)
- [Archiving an MLflow Model](#)
- [Deleting MLflow Models](#)

Model Registry CRUD Operations MLflowClient()

`create_model_version(name, source, run_id, tags=None, run_link=None, description=None)` [\[source\]](#)

`create_registered_model(name, tags=None, description=None)` [\[source\]](#)

`delete_model_version(name, version)` [\[source\]](#)

`get_latest_versions(name, stages=None)` [\[source\]](#)

`transition_model_version_stage(name, version, stage, archive_existing_versions=False)` [\[source\]](#)

Model Registry Workflow UI

This screenshot shows the Model Registry interface from the developer's perspective. On the left, there is a tree view of artifacts under 'Artifacts'. A folder named 'sklearn-model' is selected, containing files 'MLmodel', 'conda.yaml', and 'model.pkl'. The 'Full Path' is listed as '/mirluns/0/55eb2ad528114c68bd354a0568eca327/artifacts/sklearn-model' and the 'Size' is '0B'. Below this, there is a section titled 'Select a file to preview' with the instruction 'Supported formats: image, text, html, geojson files'. A large green arrow points from this interface to the 'MODEL DEVELOPER' section.



MODEL
DEVELOPER

This screenshot shows the Model Registry interface from the developer's perspective. It features a 'Tags' section with a table for adding tags, an 'Artifacts' section showing the same 'sklearn-model' folder, and a 'Register Model' dialog box. The dialog box contains fields for 'Model' (set to '+ Create New Model') and 'Model Name' (set to 'SKLearnPowerForecast'). A large green arrow points from this interface to the 'MODEL DEVELOPER' section.

This screenshot shows the Model Registry interface from the developer's perspective. It includes a summary of metrics ('mse: 43186.3', 'rmse: 207.8'), a 'Tags' section, and a 'Register Model' dialog box. The dialog box has 'Model' set to '+ Create New Model' and 'Model Name' set to 'SKLearnPowerForecast'. A large green arrow points from this interface to the 'MODEL DEVELOPER' section.



This screenshot shows the Model Registry interface after the model has been registered. It displays the registered model details: 'SKLearnPowerForecast', 'v1', 'Registered on 2020/05/04', and a 'Tags' section. The 'Artifacts' section shows the 'sklearn-model' folder with files 'MLmodel', 'conda.yaml', and 'model.pkl'. A large green arrow points from this registered state back to the 'MODEL DEVELOPER' section.

Model Registry Workflow UI

This screenshot shows the mlflow Model Registry UI. It displays a model version named 'SKLearnPowerForecast > Version 1'. The page includes details such as 'Registered At: 2020-05-04 11:38:47', 'Creator:', 'Stage: None', and 'Source Run: Random Forest Regressor: Power Forecasting Model'. A large green arrow points from this screen to the 'MODEL REVIEWER' section.



MODEL
REVIEWER

This screenshot shows the mlflow Model Registry UI. It displays a model version named 'SKLearnPowerForecast > Version 1'. The 'Stage' dropdown is set to 'None'. Below it, there are three buttons for transitioning the model: 'Transition to → Staging' (orange), 'Transition to → Production' (green), and 'Transition to → Archived' (grey). A large green arrow points from the 'MODEL REVIEWER' section to this screen.



DOWNSTREAM
USERS



AUTOMATED JOBS



REST SERVING

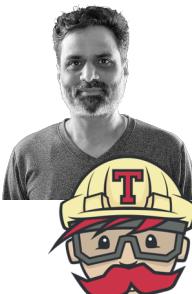
Model Registry Workflow API

```
mlflow.register_model(model_uri, "WeatherForecastModel")  
  
mlflow.sklearn.log_model(model,  
    artifact_path="sklearn_model",  
    registered_model_name= "WeatherForecastModel")
```

MODEL
DEVELOPER



REVIEWERS,
CI/CD TOOLS



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

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

DOWNSTREAM
USERS



AUTOMATED JOBS



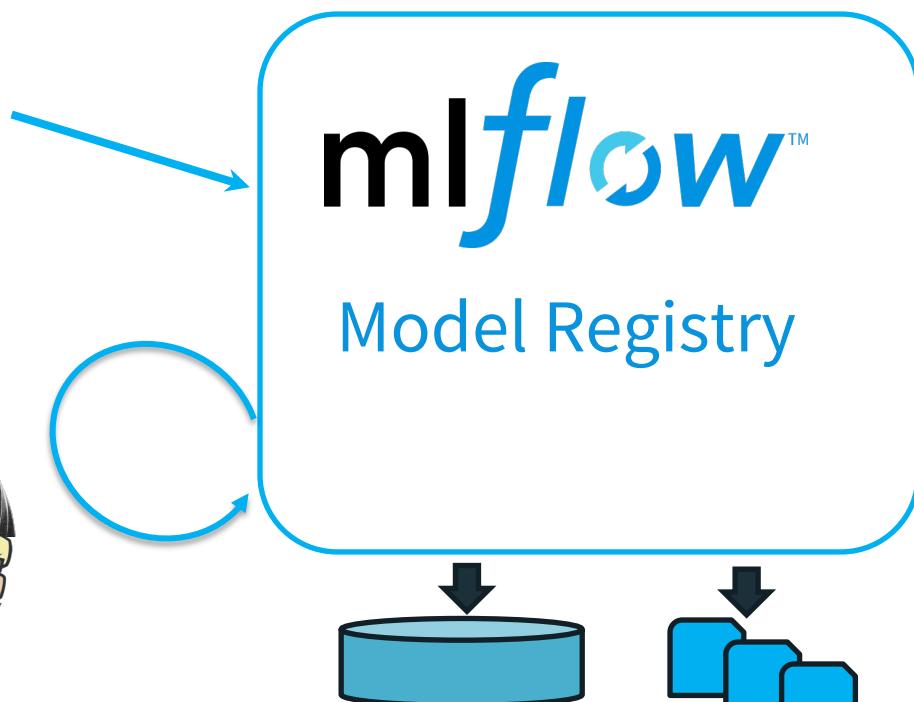
REST SERVING



Model Registry Workflow API

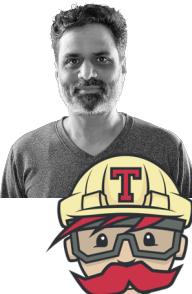
```
mlflow.register_model(model_uri, "WeatherForecastModel")  
  
mlflow.sklearn.log_model(model,  
    artifact_path="sklearn_model",  
    registered_model_name= "WeatherForecastModel")
```

MODEL
DEVELOPER

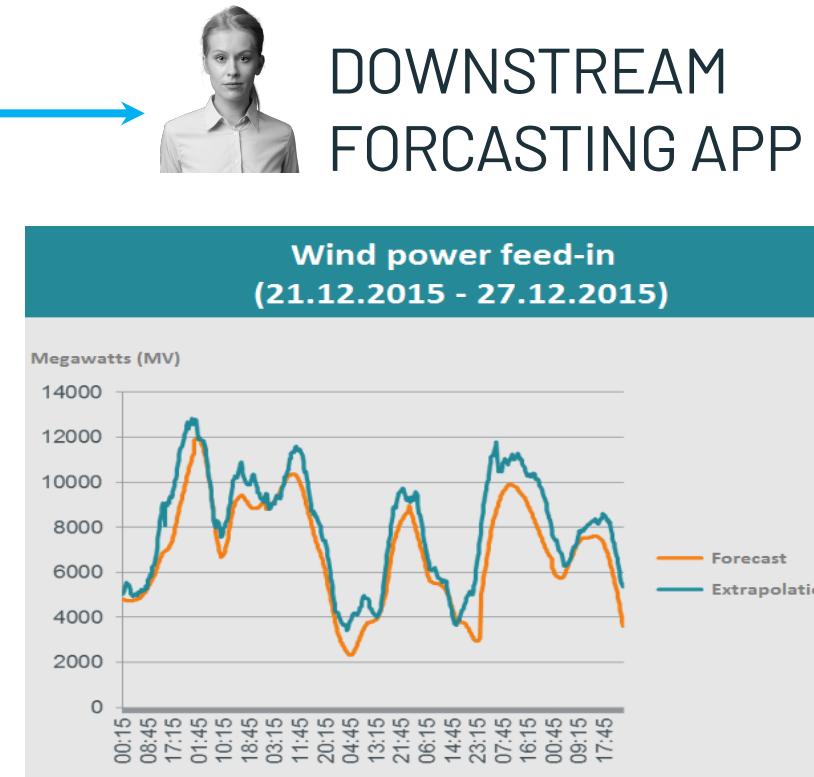


```
model_uri = "models:/{{model_name}}/production".format(  
    model_name="WeatherForecastModel")  
model_prod = mlflow.pyfunc.load_model(model_uri)  
model_prod.predict(data)
```

REVIEWERS,
CI/CD TOOLS



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

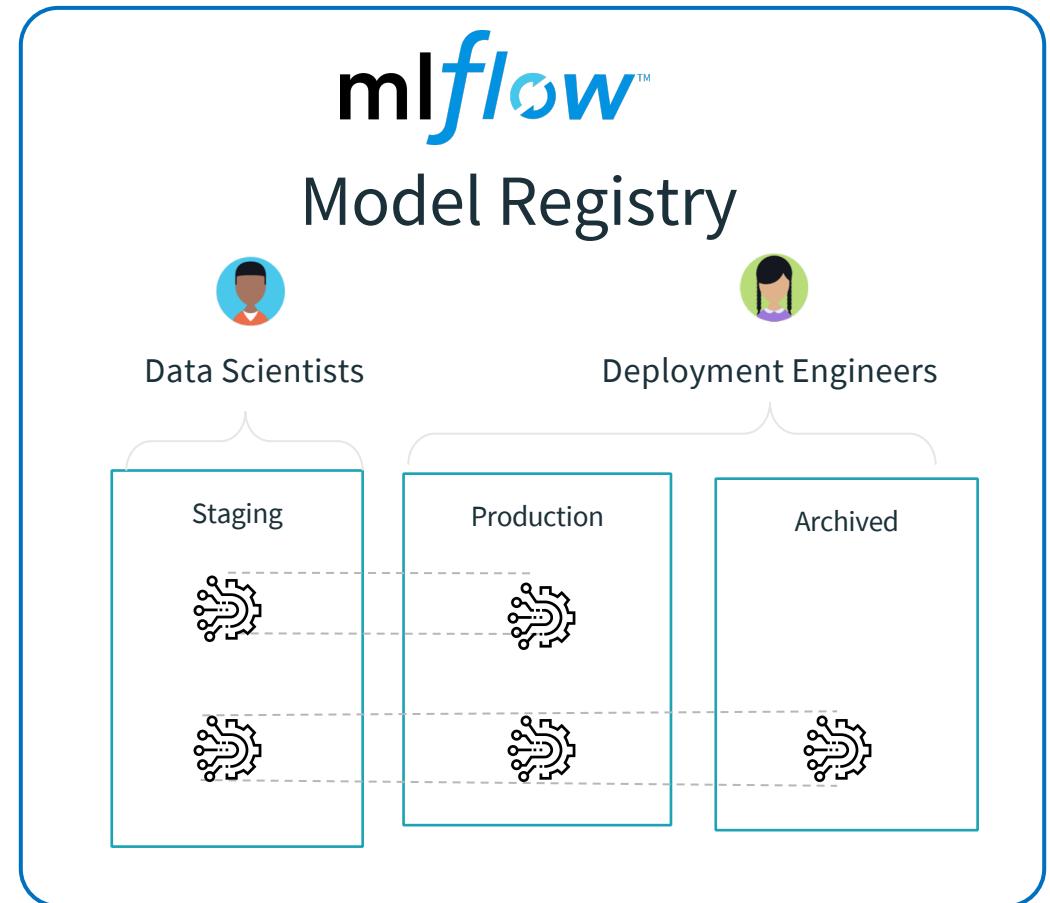


O'REILLY®

```
(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> █
```

MLflow Model Registry Recap

- **Central Repository:** Unique named registered models for discovery across data teams
- **Model Registry Workflow:** Provides UI and API for registry operations
- **Model Versioning:** Allow multiple versions of model in different stages
- **Model Stages:** Allow stage transition: none, staging, production, or archived
- **CI/CD Integration:** Easily load a specific version for testing and inspection
- **Model Lineage:** Provides model description, lineage and activities



mlflow Model Registry: Tag and Search APIs

Tags to track custom metadata for a model version, e.g., test results

Search API to automate model management and MLOps actions

The screenshot shows the MLflow Model Registry interface. At the top, there are navigation links for 'Experiments' and 'Models'. Below that, the path 'Models > KNN > Version 12' is displayed. On the right, there are details: 'Created at: 2018-12-04 17:11:06', 'User: test@example.com', 'Stage: Staging', 'Last Modified: 2018-12-04 17:11:06', and 'Source: Run 123'. A red arrow points to the delete icon in the 'Actions' column for the 'passed-gdpr-review' tag. The 'Tags' section contains two entries:

Name	Value	Actions
passed-gdpr-review	true	
passed-performance-test	true	

Below the tags is an 'Add Tag' button. The 'Activity' section shows two recent events:

- Alice requested a stage transition from None to Staging 10 hours ago.
- Carol approved a stage transition from None to Staging 10 hours ago.

Databricks Model Serving

Available in
Databricks

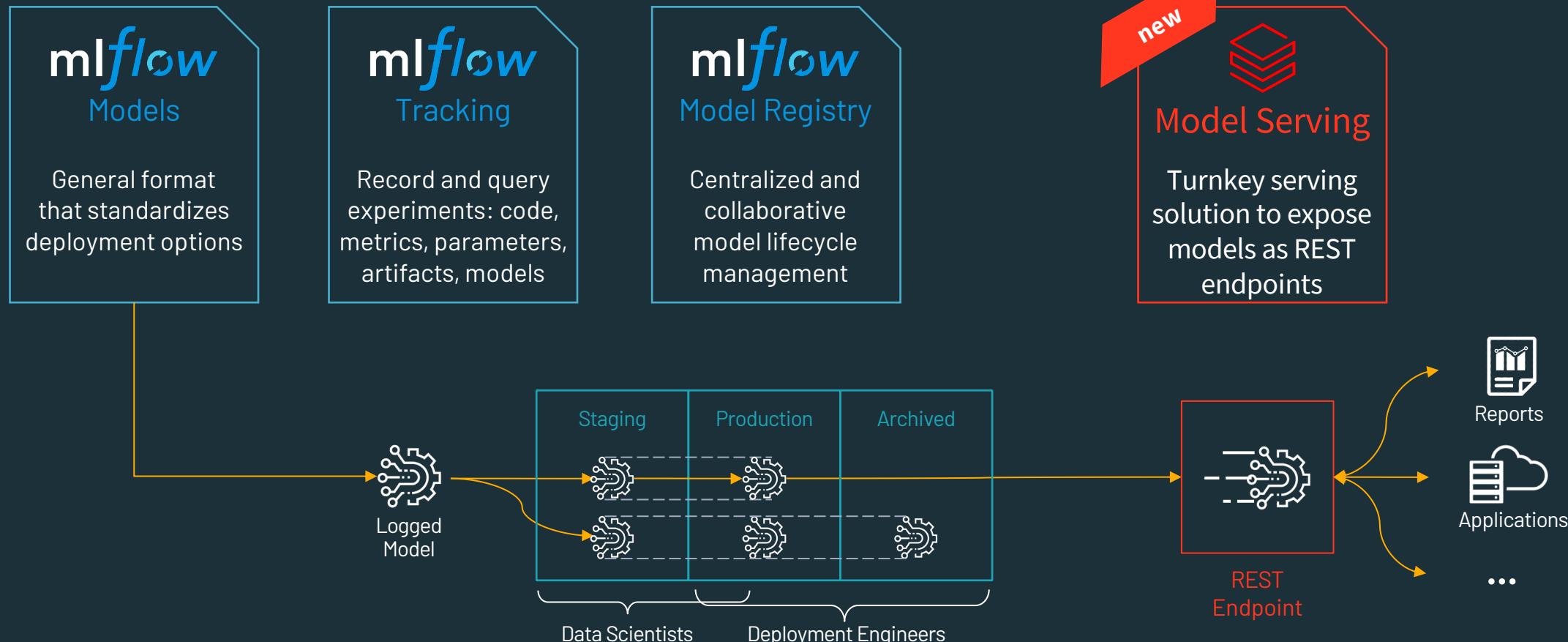
Databricks

Announcing MLflow Model Serving on Databricks - 2020-06-25

Databricks

MLflow Model Serving On Databricks

Model Serving on Databricks



Databricks Lightweight Model Serving

The screenshot shows the Databricks MLflow interface for a model named "classificationnxmodel". The sidebar on the left includes icons for MLflow, My folder, Workspace, Recent, Table, Cluster, Job, Models, and Search. The main area displays the model's details: Created at: 08/15/2018 17:49:18, Last modified: 08/30/2018 12:13:43, and a description field. A modal dialog titled "Enable RESTful Model Serving" is open, prompting the user to "Select Cluster" (set to "Use Existing Cluster") and "Cluster Name" (set to "Shared Autoscaling"). The "Register" button is highlighted with a cursor. In the background, there are tabs for "All" and "Active (4)" versions, a pending requests section (3 pending), and an activity log showing recent interactions between users Alice and Carol.

MLflow

Models > classificationnxmodel

Created at: 08/15/2018 17:49:18 Last modified: 08/30/2018 12:13:43

Description:

Versions All Active (4)

Version	Created
Version 13	08-2
Version 12	06-1
Version 9	12-1
Version 8	04-2

Activity

View 14 older

Alice added a description 2 days ago

Alice requested a stage transition [None] → Staging on version 12 12 hours ago

Carol approved a stage transition [None] → Staging on version 11 10 hours ago

Alice requested a stage transition Staging → Production on version 11 2 hours ago

Carol rejected a stage transition Staging → Production on version 11 2 hours ago

Enable RESTful Model Serving

Once enabled, the given cluster will be used to serve all versions of this model.

Select Cluster Use Existing Cluster

Cluster Name Shared Autoscaling

Pending Requests

Count
3
1
-
1

Cancel Register

Databricks Model Serving

Registered Models > andre_02_Sklearn_Train_Predict ▾

Details Serving

Status: ● Ready - Stop

Cluster: [mlflow-model-andre_02_Sklearn_Train_Predict](#) ?

Model Versions

Version 1

● Ready

Production

Model URL: ?

https://dogfood.staging.cloud.databricks.com/model/mlflow-model-andre_02_Sklearn_Train_Predict/1
https://dogfood.staging.cloud.databricks.com/model/mlflow-model-andre_02_Sklearn_Train_Predict/Production

Call the model

Request ?

```
[[ 7, 0.27, 0.36, 20.7, 0.045, 45, 170, 1.001, 3, 0.45, 8.8 ]]
```

Send Request

Response ?

```
5.600277264325324
```

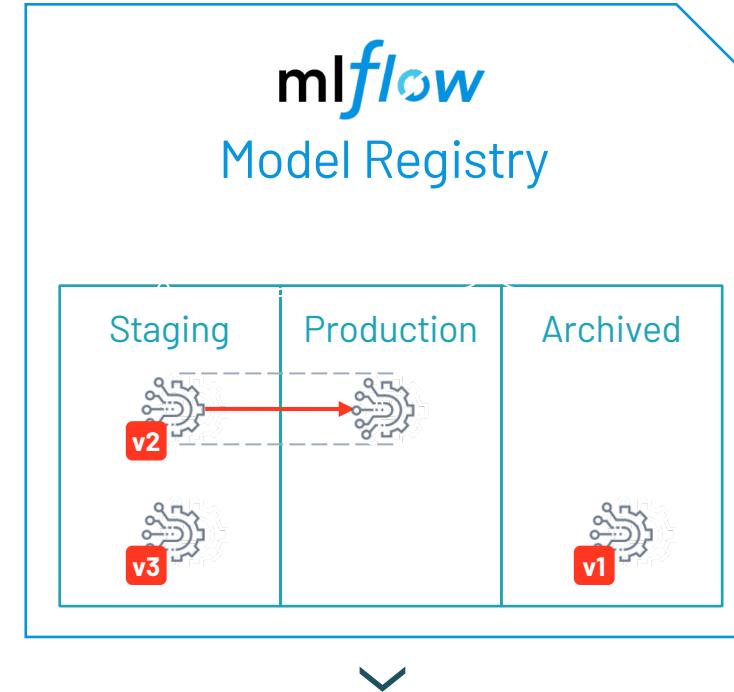
mlflow Model Registry: Webhooks

Databricks Webhooks allow setting callbacks on registry events like stage transitions to run CI/CD tools

MLflow Model Registry on Databricks
Simplifies MLOps With CI/CD Features



by Sue Ann Hong, Ankit Mathur, Jules Damji and Mani Parkhe
Posted in ENGINEERING BLOG | November 19, 2020



`VERSION_REGISTERED: MyModel, v2`

`TRANSITION_REQUEST: MyModel, v2, Staging→Production`

`TAG_ADDED: MyModel, v2, BacktestPassed`

just launched

Human
Reviewers

CI/CD Tools

Batch
Scoring

Real-time
Serving



mlflow Model Registry: Comments

just launched

Comments in the Databricks workspace can now be used to discuss changes on models

The screenshot shows a 'Activities' section with one item. It features a blue speech bubble icon, the email address 'ci-pipeline@databricks.com', the timestamp '2 hours ago (edited)', and the message 'Tests failed - see output [here](#)'. Below this is a placeholder 'Add a comment' input field.

MLflow Model Registry on Databricks Simplifies MLOps With CI/CD Features



by Sue Ann Hong, Ankit Mathur, Jules Damji and Mani Parkhe
Posted in ENGINEERING BLOG | November 19, 2020



Recap of all parts: What Did We Talk About?



- Modular Components greatly simplify the ML lifecycle
- Easy to install & Great Developer experience
- Develop & Deploy locally; track locally or remotely
- Use Available APIs: Python, Java & R (Soon Scala)
- REST APIs and CLI tools
- Visualize experiments and compare runs
- Centrally register and manage model lifecycle

Model Registry Tutorial

Tutorials: <https://github.com/dmatrix/olt-mlflow>

Thank you! 😊

Q & A

jules@databricks.com
@2twitme

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