Data/AI Projects Methodology

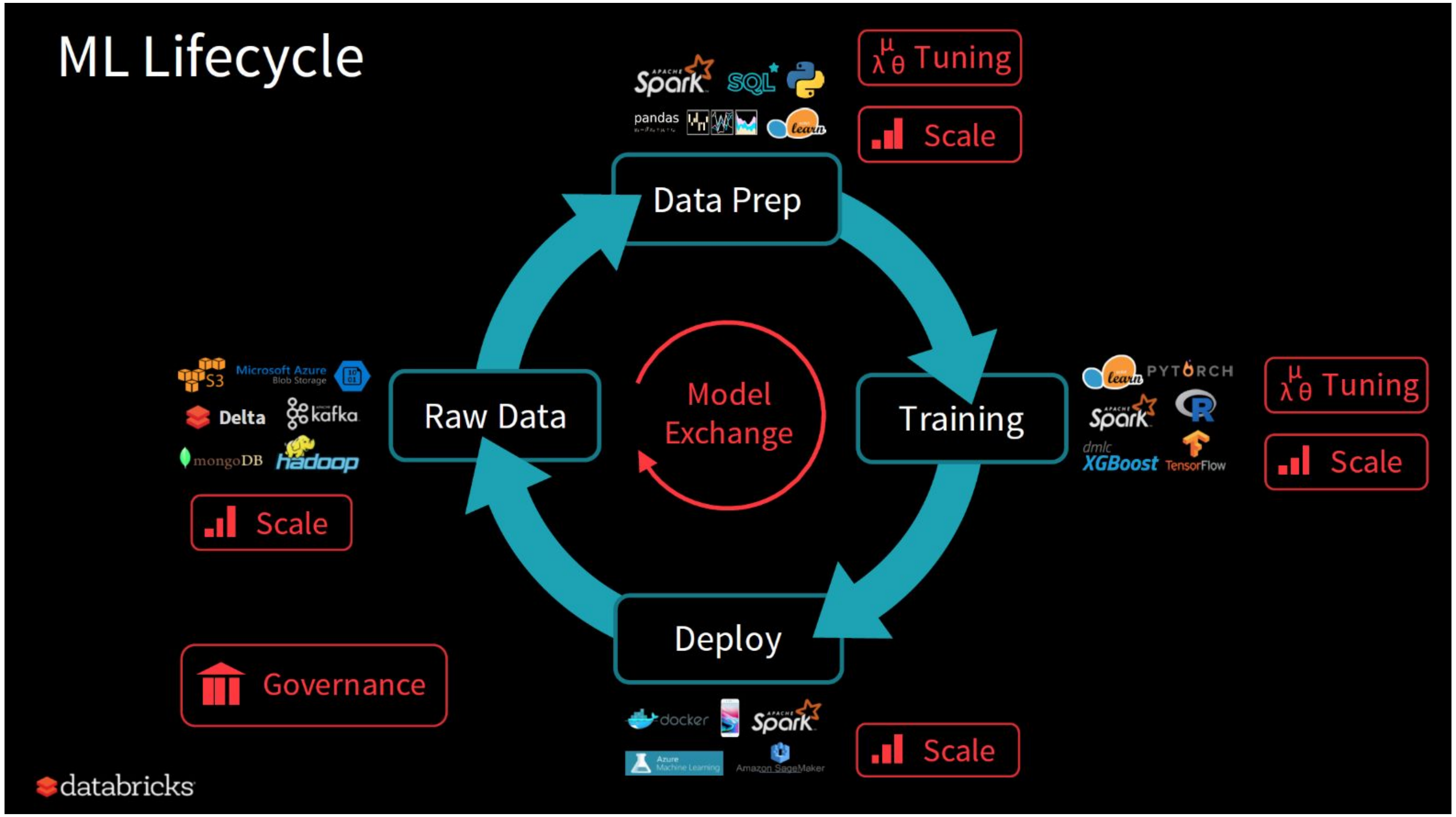
Chapter 2 : Lifecycle Management with MLflow

Medina HADJEM

2021-2022

Outline

ML Development Lifecycle



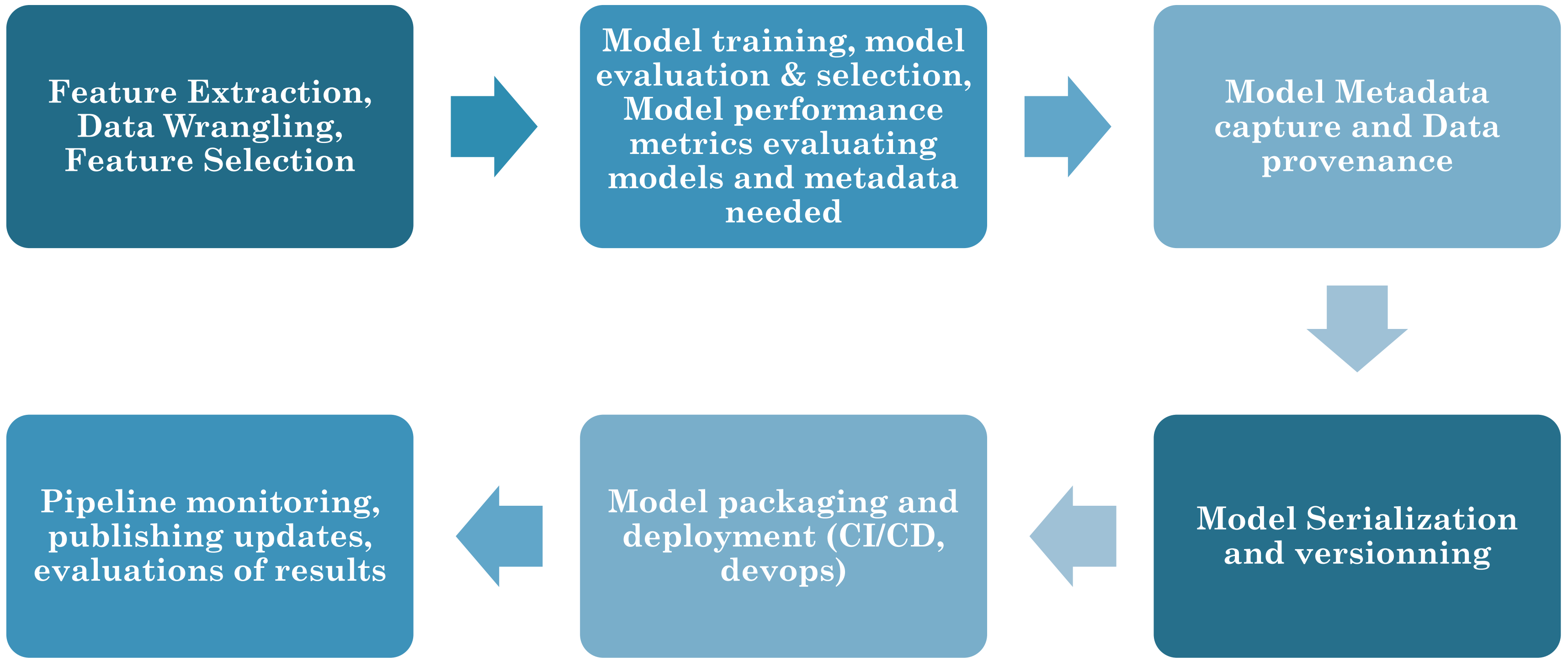
ML Development Challenges

- ML projects development process is complex :

Traditional Software	ML Software
Answer a functional specification	Optimize a metric
Quality depends on code	Quality depends on input data and tuning parameters
Generally use one software stack	Compare/combine many libraries, models & algorithms

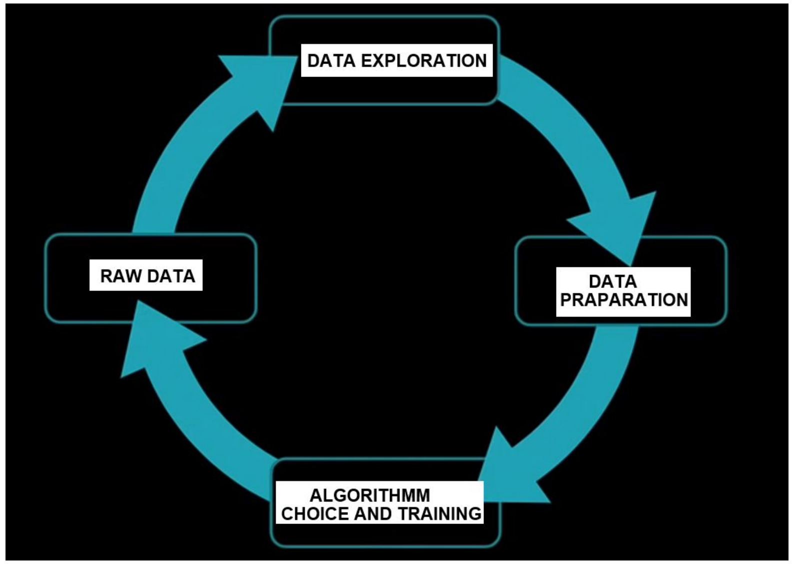
ML Production Steps

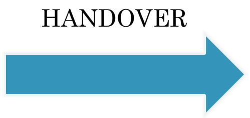
- Main ML project industrialization steps :

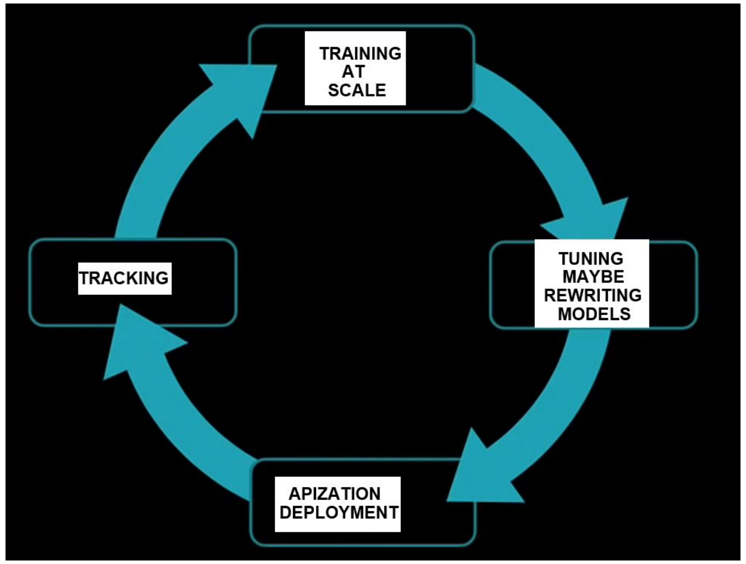

Feature Extraction, Data Wrangling, Feature Selection
Model training, model evaluation & selection, Model performance metrics evaluating models and metadata needed
Model Metadata capture and Data provenance
Model Serialization and versionning
Model packaging and deployment (CI/CD, devops)
Pipeline monitoring, publishing updates, evaluations of results

ML Production Challenges

- ML in production is even harder than ML Development :
  - Different people involved in the process (Data Engineer, ML Engineer, Software Engineer, Ops..)
  - Many technical steps (buiding, deploying, packaging, scheduling, monitoring…)







- DATALAB DATASCIENTISTS PART
- INDUSTRIALIZATION DATA ENGINEERS PART

ML Production challenges

- Handoffs from data scientists to data engineers is generally not easy :
  - Data access in production (specially in big volumes)
  - Rewriting models in another languages (ability o industrialize many languages)
  - Packaging, scheduling and continuous models deployment CI/CD Devops needs
  - Different ML frameworks and libraries in production
  - Scaling Up

- ML Experiments/models tracking (In Dev/Lab & production)
- Managing ML models portfolio (Model catalog)
- Models monitoring in production
- Lack of tools needed for ML in production

ML Production Challenges

- Results reproducibility
- Auditability and Compliance
- Standard documentation
- AB testing

Custom ML Plateforms

- Examples :
  - AWS SageMaker Pipelines
  - Google TFX (Tensor Flow Extended)
  - Facebook FBLearner
- Advantages / Drawbacks

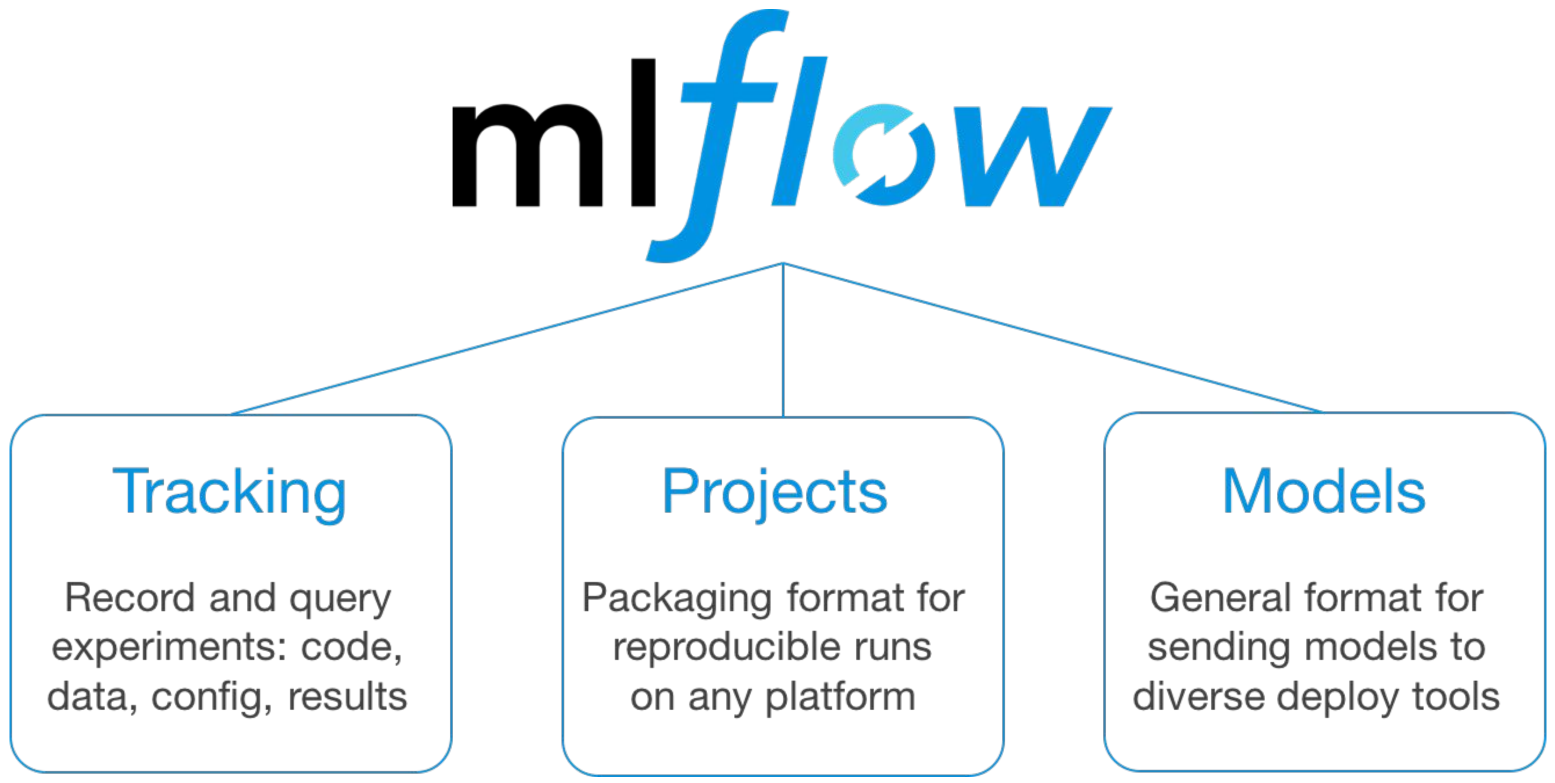
  + Standardize the data prep / training / deploy loop

  – Limited to a few algorithms or frameworks

  – Tied to one company's infrastructure
- In general :
  - There is not yet a standard way for ML projects industrialization
  - Some companies try to provide custom templates to be implemented by data scientists.

What is MLflow ?

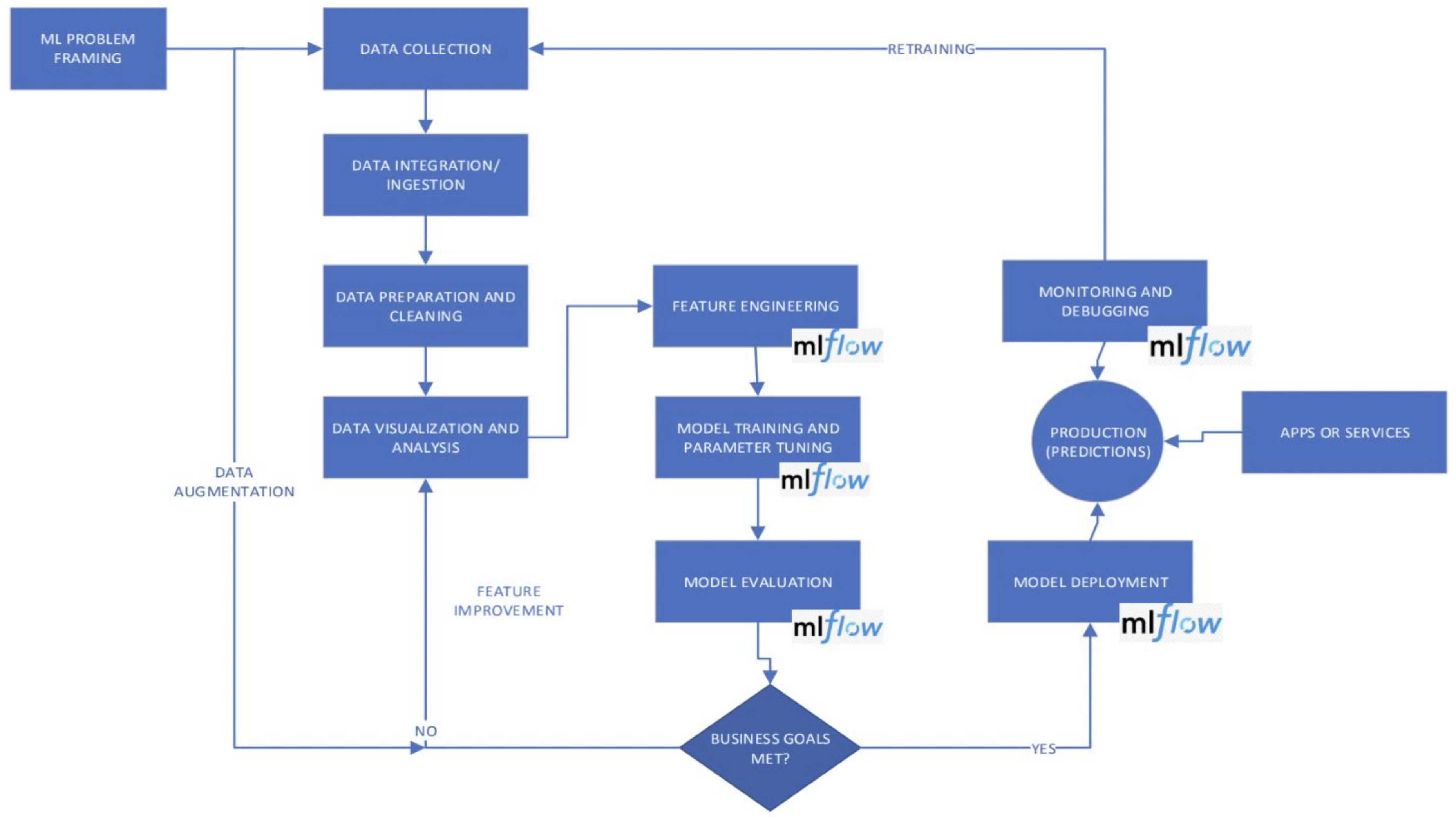
- MLflow is an **open source library** to manage the ML lifecycle, including experimentation, reproducibility and deployment.
- It currently offers three main components:


mlflow
Tracking
Record and query experiments: code, data, config, results
Projects
Packaging format for reproducible runs on any platform
Models
General format for sending models to diverse deploy tools

Why MLflow can be useful ?

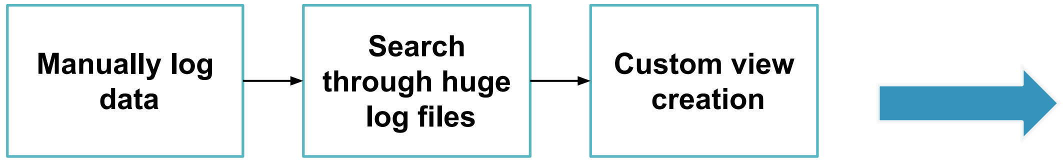
- Open Source Library
- Works with any ML library and language
- Runs the same way everywhere (On premise, Cloud, ..)
- Scales to Big data with Apache spark
- Provides out of the box web application
- Provides integrations (like with Scality)

MLFlow in ML process



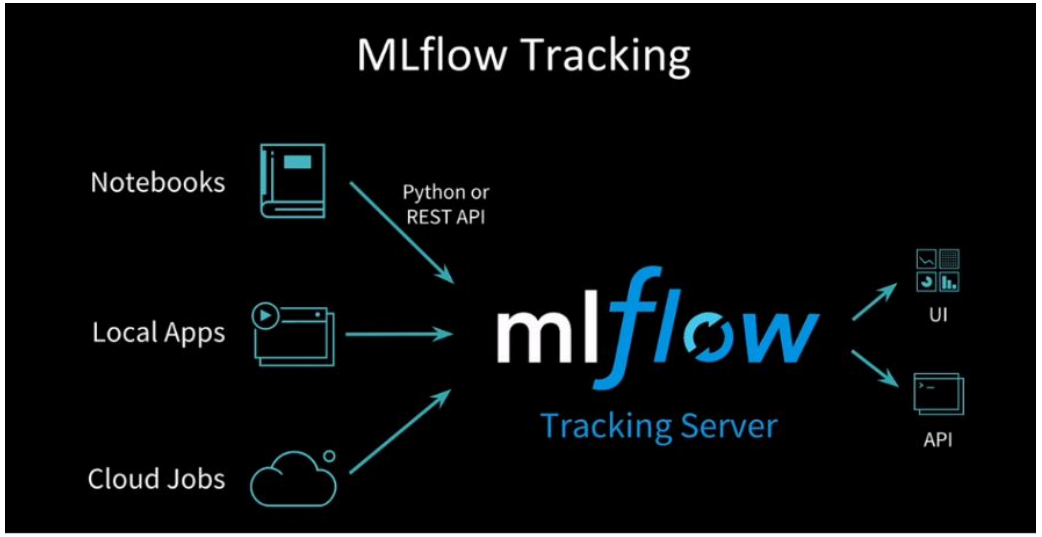
MLFlow Tracking

- **Without MLFlow**



- Log data using custom loggers
- Web app for viewing log/tracking data not available and if needed must be developed and maintained
- Difficult to track model versions, metadata like parameters used in training and other useful data if searched in log files

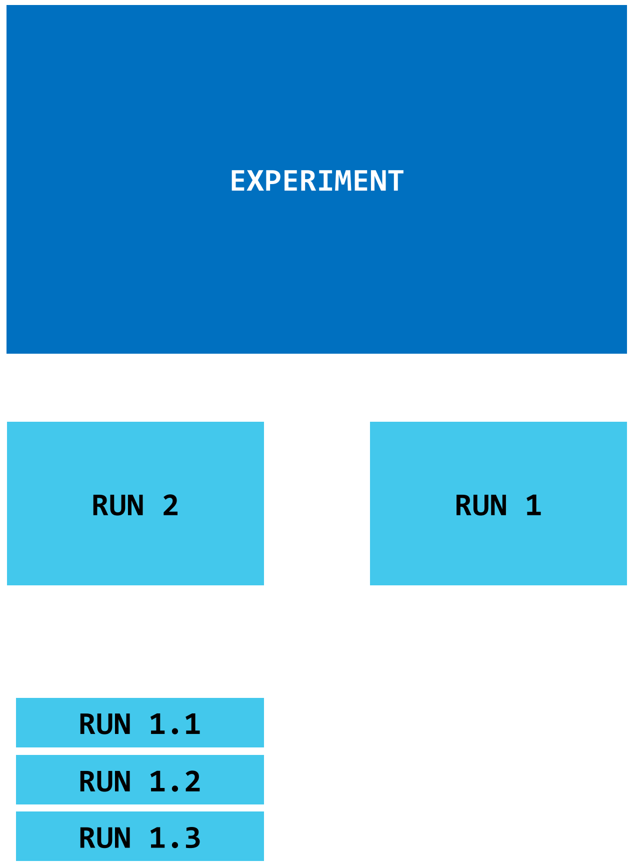
- **With MLFlow**



- Log data using ML flow API
- Web app for viewing log/tracking data available out of the box
- Easy to track model versions, metadata like parameters used in training and other useful data

ML Flow Tracking

- The plotted information is organized in runs (optionally nested). For example, 1 workout = 1 run.
- The run contains the traced information (metrics, parameters, tags, artifacts), but also information such as the source of the code that ran to generate these metrics. E.g. a path in a local FS or the address of a GIT repo with the hash of the associated commit.
- Several runs can be part of an experiment. E.g. 1 project = 1 experiment, according to the team agreement.
- The traced data can be accessed through the available APIs or through the MLFlow UI.

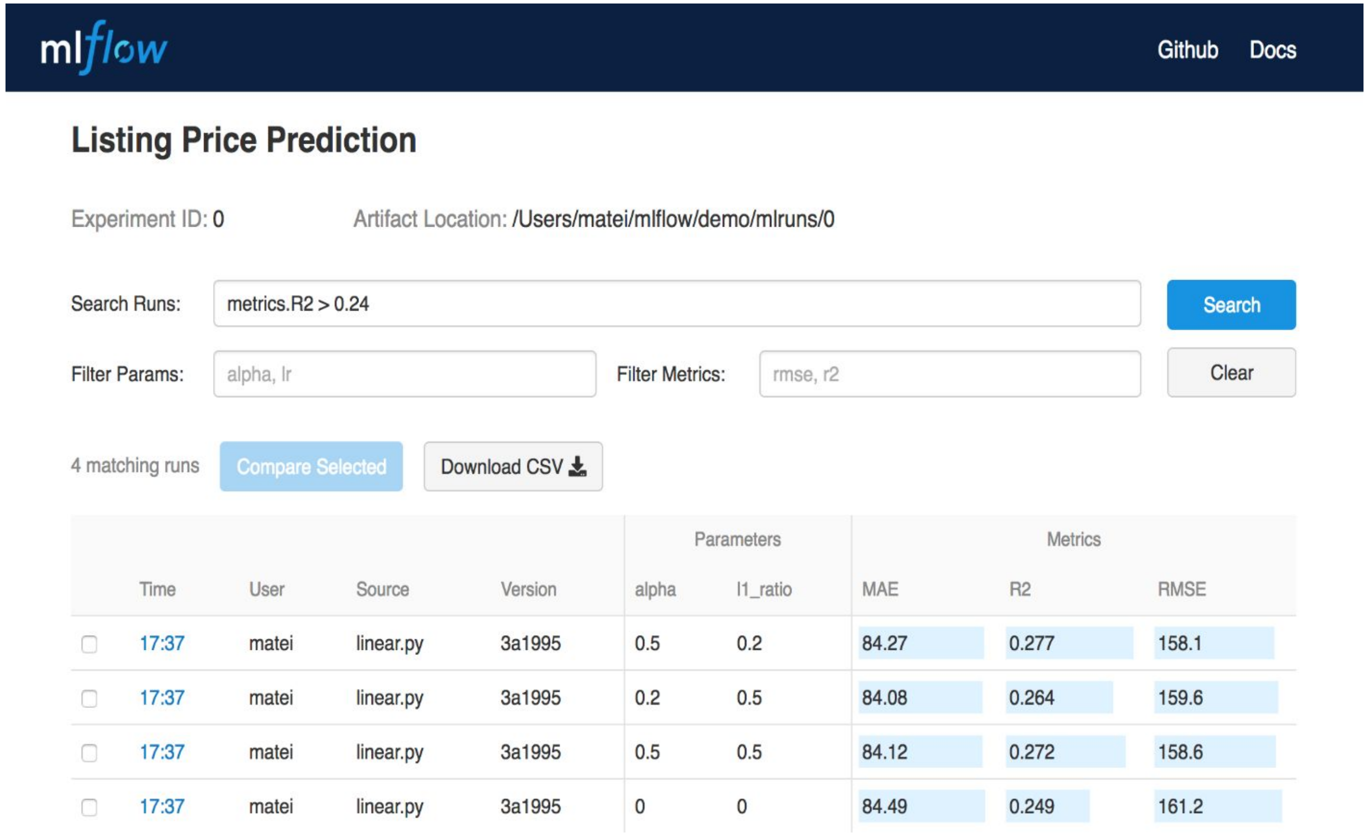


ML Flow Tracking

- The traced information and artifacts are contained locally by default (./mlruns folder).
- MLFLOW offers a tracking server to track information (metrics, parameters, tags) in a centralized way. Its backend can be a files tree or a database.
- Artifacts have their own storage area depending on the experiment. This can be a local folder, but also on HDFS, Cloud and others.

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?



Listing Price Prediction

Experiment ID: 0   Artifact Location: /Users/matei/mlflow/demo/mlruns/0

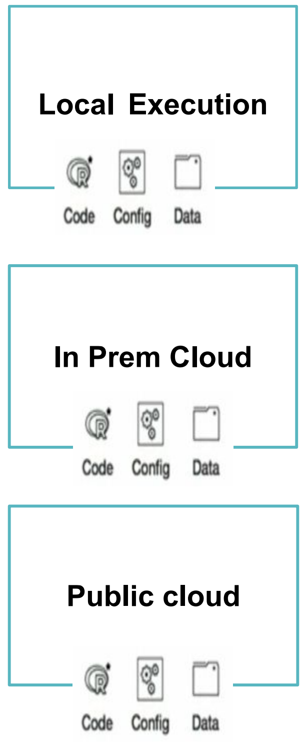
Search Runs: metrics.R2 > 0.24

4 matching runs

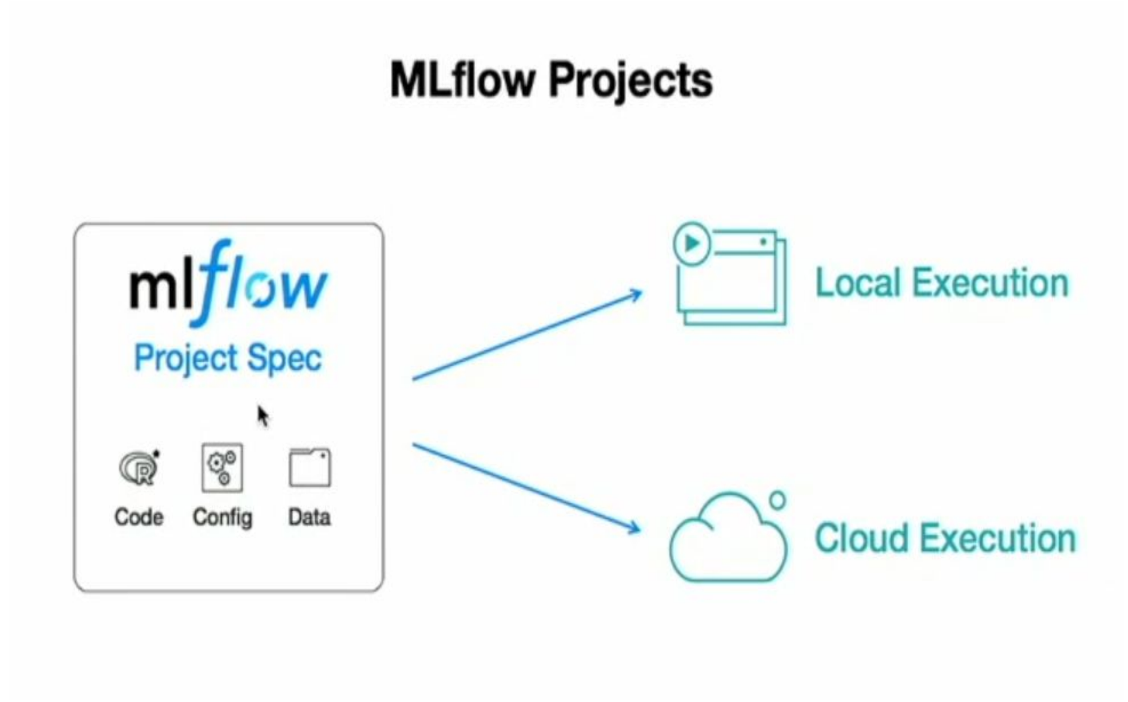
				Parameters		Metrics		
Time	User	Source	Version	alpha	l1_ratio	MAE	R2	RMSE
17:37	matei	linear.py	3a1995	0.5	0.2	84.27	0.277	158.1
17:37	matei	linear.py	3a1995	0.2	0.5	84.08	0.264	159.6
17:37	matei	linear.py	3a1995	0.5	0.5	84.12	0.272	158.6
17:37	matei	linear.py	3a1995	0	0	84.49	0.249	161.2

MLFlow Projects

- Without MLFlow



- With MLFlow

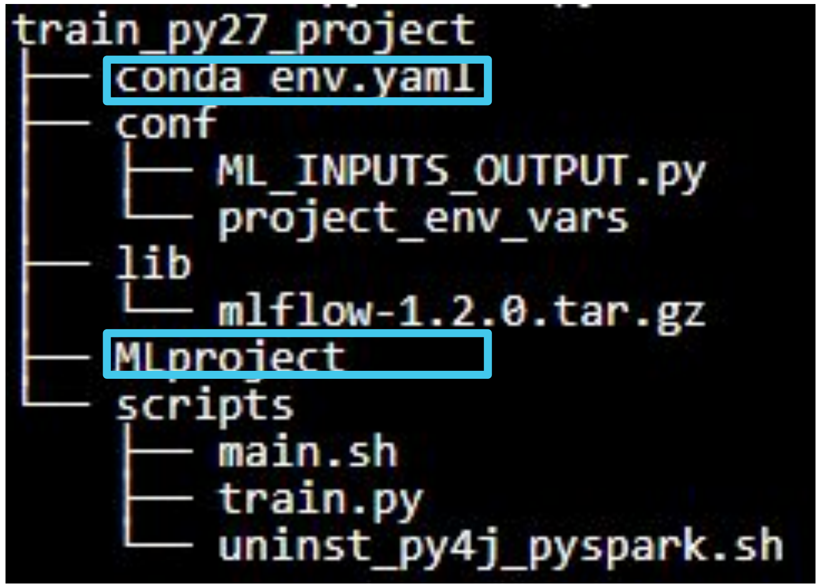


- Difficulty in packaging
- Different way of running in all environments

- Ease of packaging
- Uniform way of running in all environments

MLFlow Projects

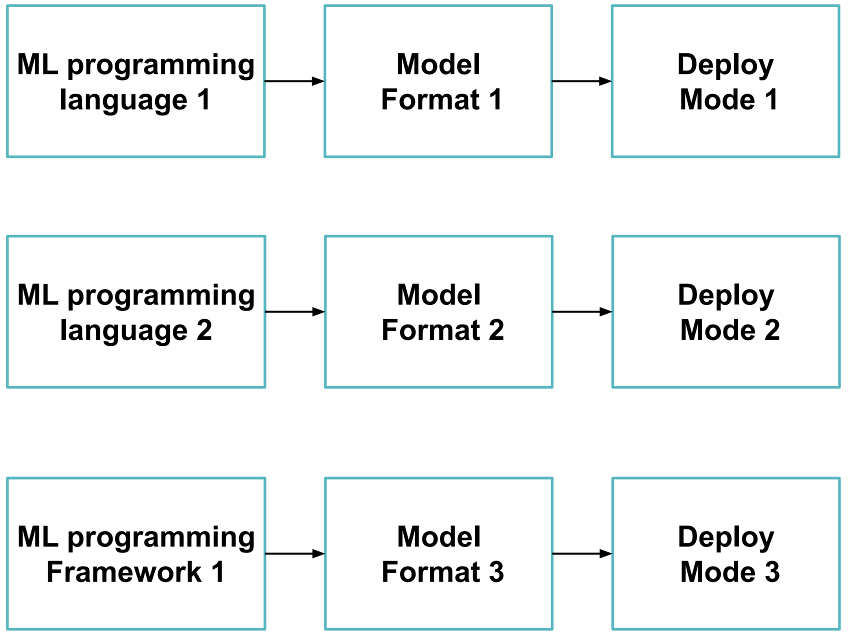
- MLFLOW Project establishes an organization convention for the project code to make it easily reusable.
- This convention allows MLFLOW to generate the necessary environment and to execute the code under the same conditions as intended by its developer, so that the execution is reproducible.
- An "`MLproject`" file located at the root of the project folder is used to configure the project, by specifying:
  - The runtime environment (.yaml file to generate a conda environment, or path to a docker image)
  - The entry points of the project with, optionally, parameters. For example, several entry points can be useful if there are several training scripts



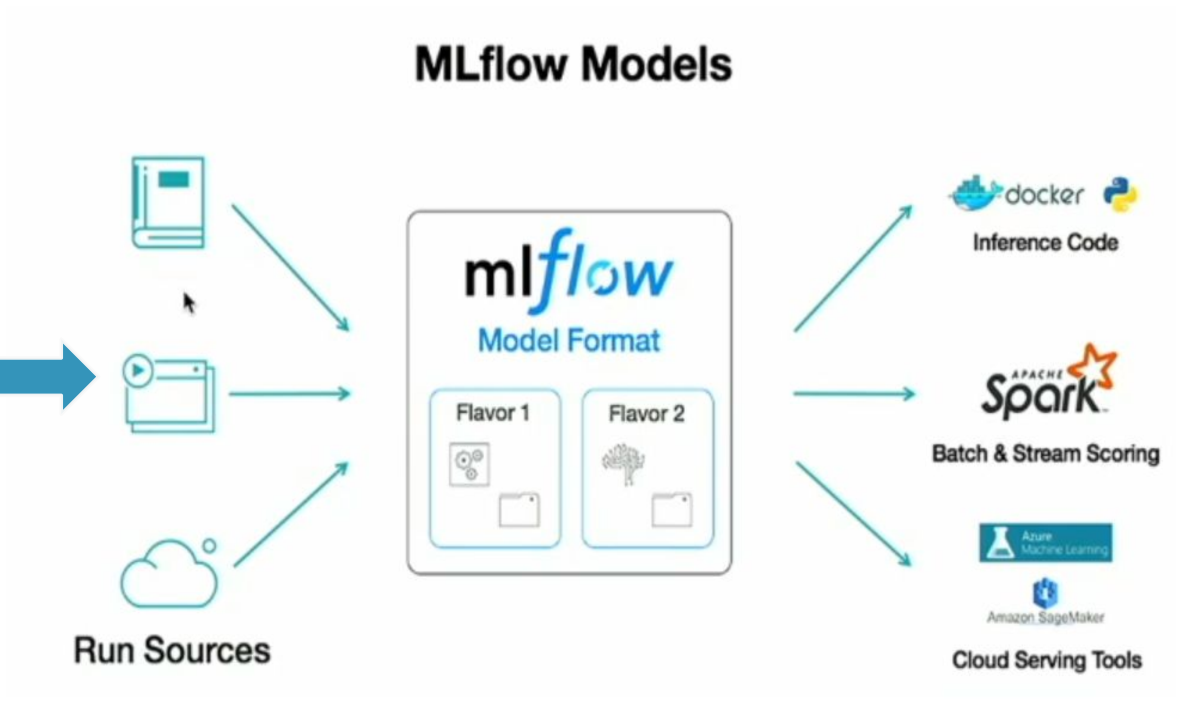
```
train_py27_project
├── conda_env.yaml
├── conf
│   ├── ML_INPUTS_OUTPUT.py
│   └── project_env_vars
├── lib
│   └── mlflow-1.2.0.tar.gz
├── MLproject
└── scripts
    ├── main.sh
    ├── train.py
    └── uninst_py4j_pyspark.sh
```

MLFlow Models

- **Without MLFlow**



- **With MLFlow**



- Different model format while using different languages and frameworks
- Maybe deployment pipeline different for each model format
- Possible impact in model performance because of change in model format

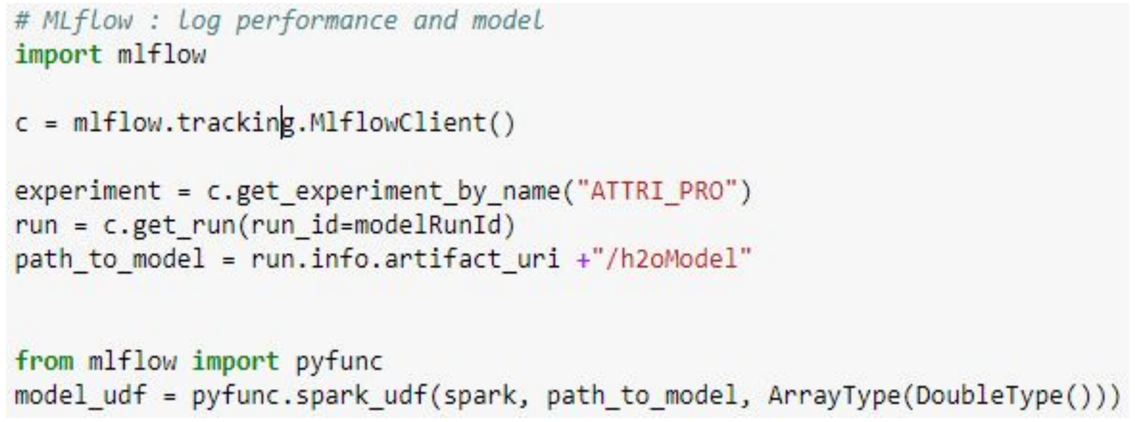
- Same model format while using different languages and frameworks
- Same deployment pipeline for same model format
- No impact in model performance because of change in model format

ML Flow Models

- MLFLOW Model allows you to export models by saving them in the experiment's artifact store (in the sub-folder associated with the model's run). It supports several libraries, among which H2O, scikit-learn, xgboost, tensorflow, etc.

- The models can then be imported by MLFLOW to make predictions.

- Models can be exported and imported with multiple flavors. Among these, the `python_function` flavor is available for several libraries and presents a generic way to use models. Among other things, it allows you to load models and pass them to Spark UDFs.

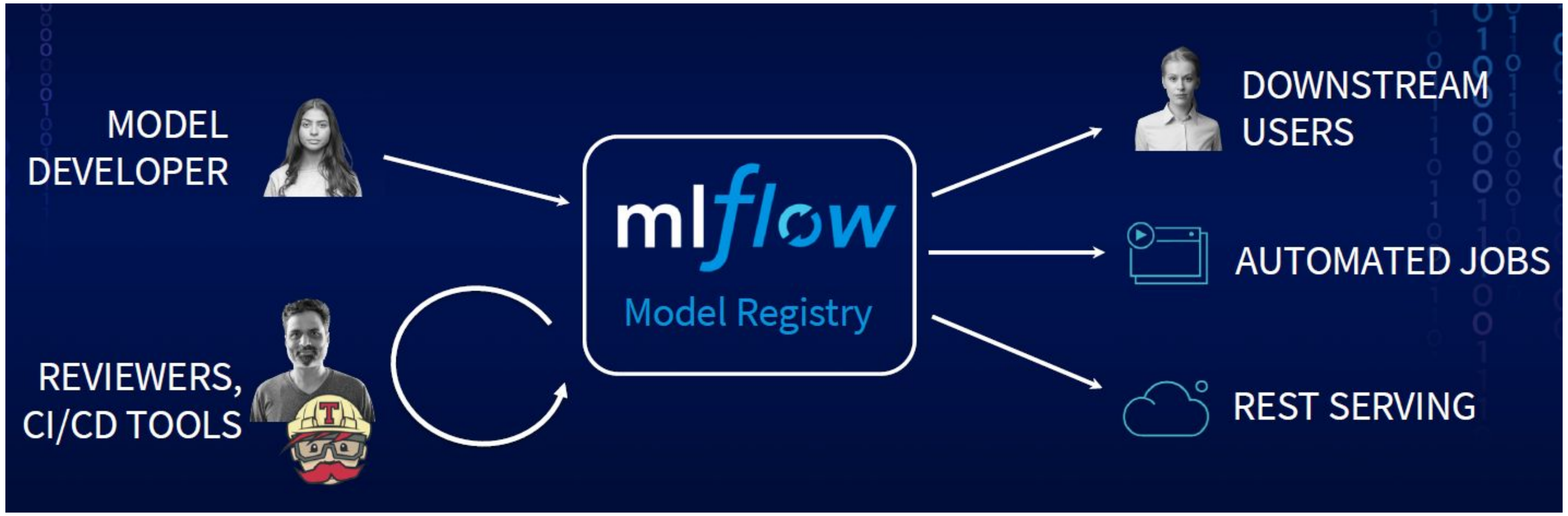


```
# MLflow : log performance and model
import mlflow

c = mlflow.tracking.MlflowClient()

experiment = c.get_experiment_by_name("ATTRI_PRO")
run = c.get_run(run_id=modelRunId)
path_to_model = run.info.artifact_uri +"/h2oModel"


from mlflow import pyfunc
model_udf = pyfunc.spark_udf(spark, path_to_model, ArrayType(DoubleType()))
```

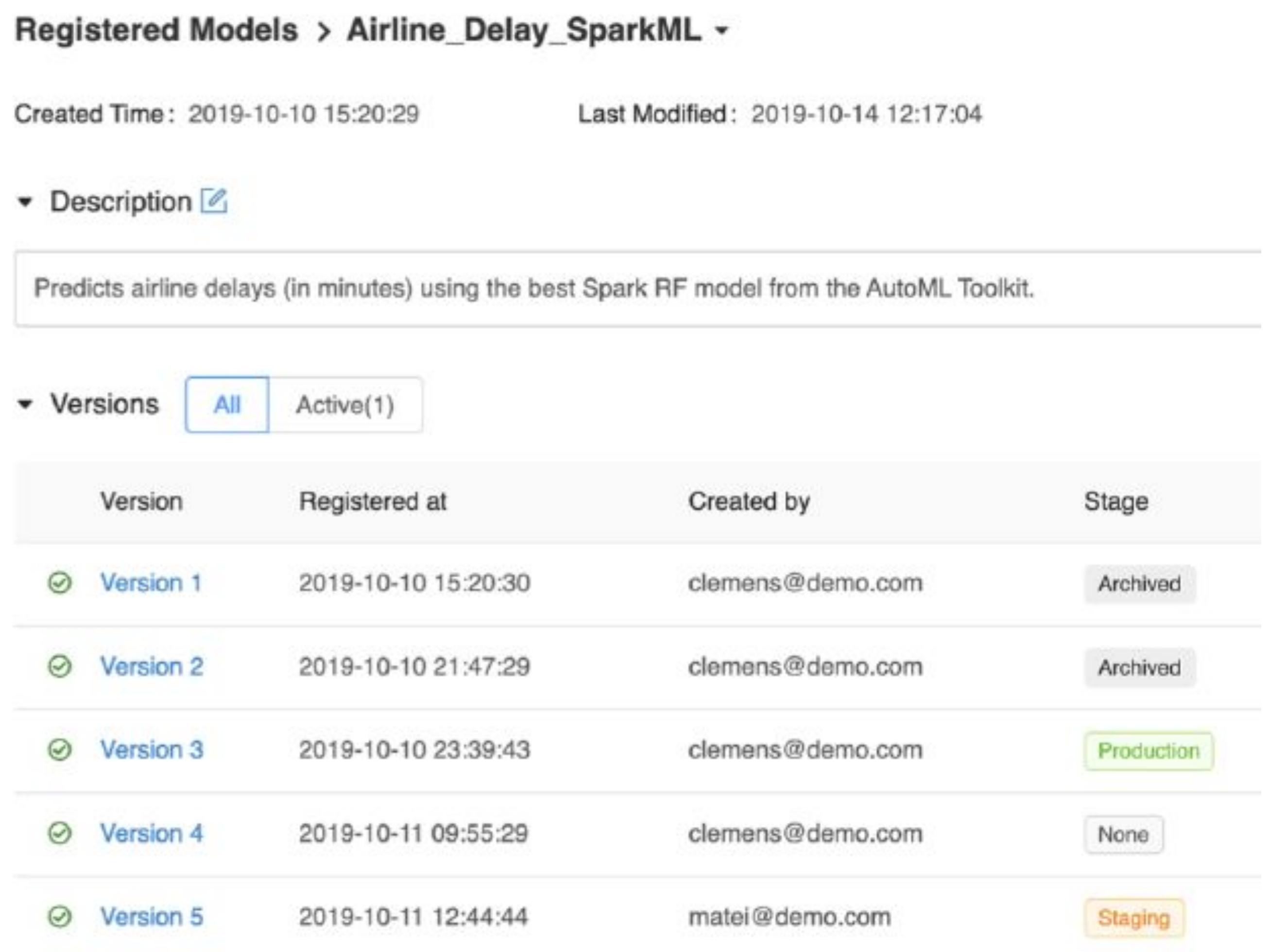
Using the Python API (or other), it is possible to request the tracking server to find out the artifact store of a specific run, then to read the model from the artifact with MLFLOW Model. Thus, there is no need to include the model in the project "bundle".

MLFlow Model Registery

- Repository of named, versioned models with comments & tags
- Track each model's stage: dev, staging, production, archived
- Easily load a specific version



MLFlow Model Registery



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.

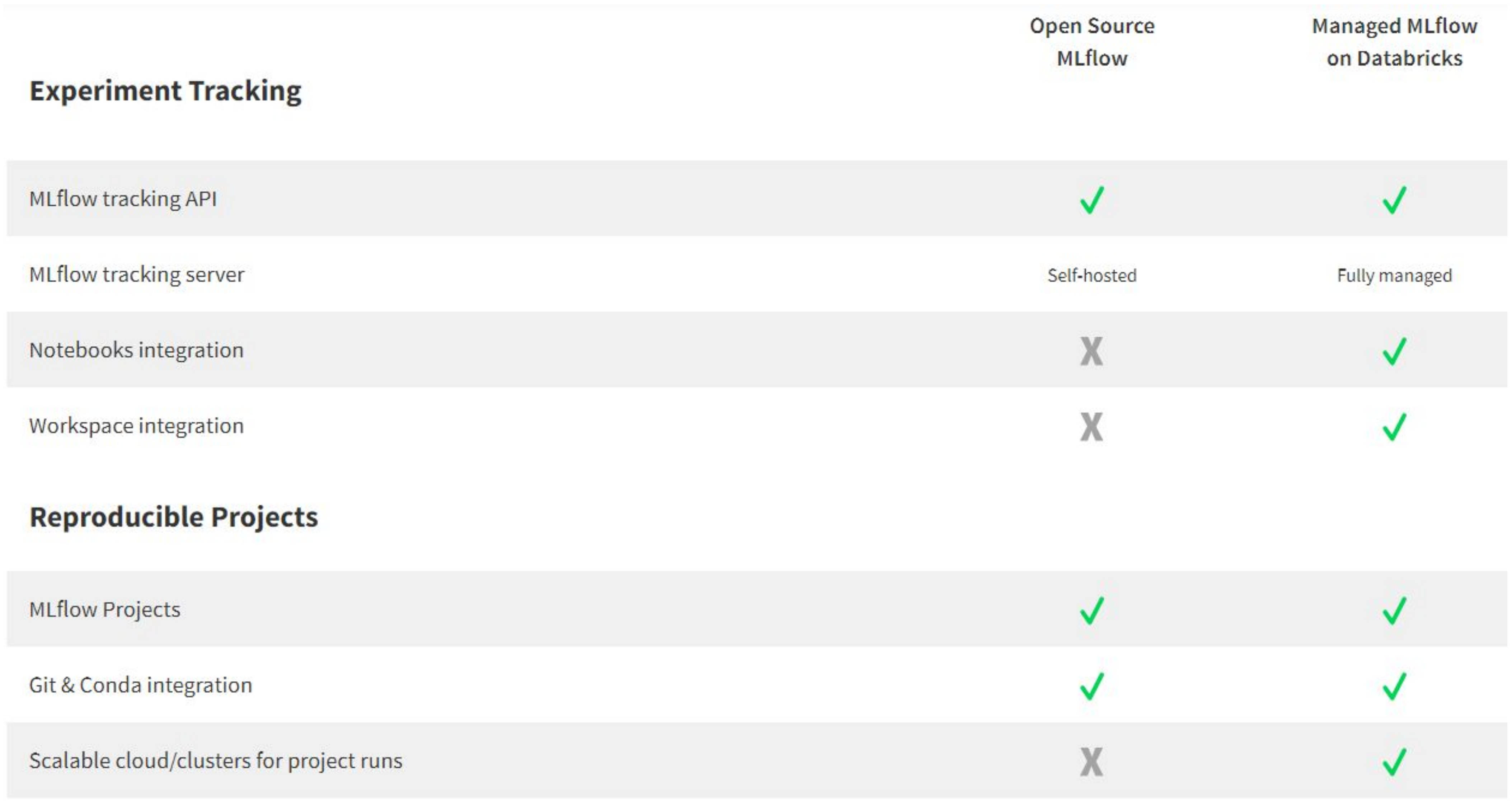
Versions All Active(1)

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

Managed MLFlow on Databricks

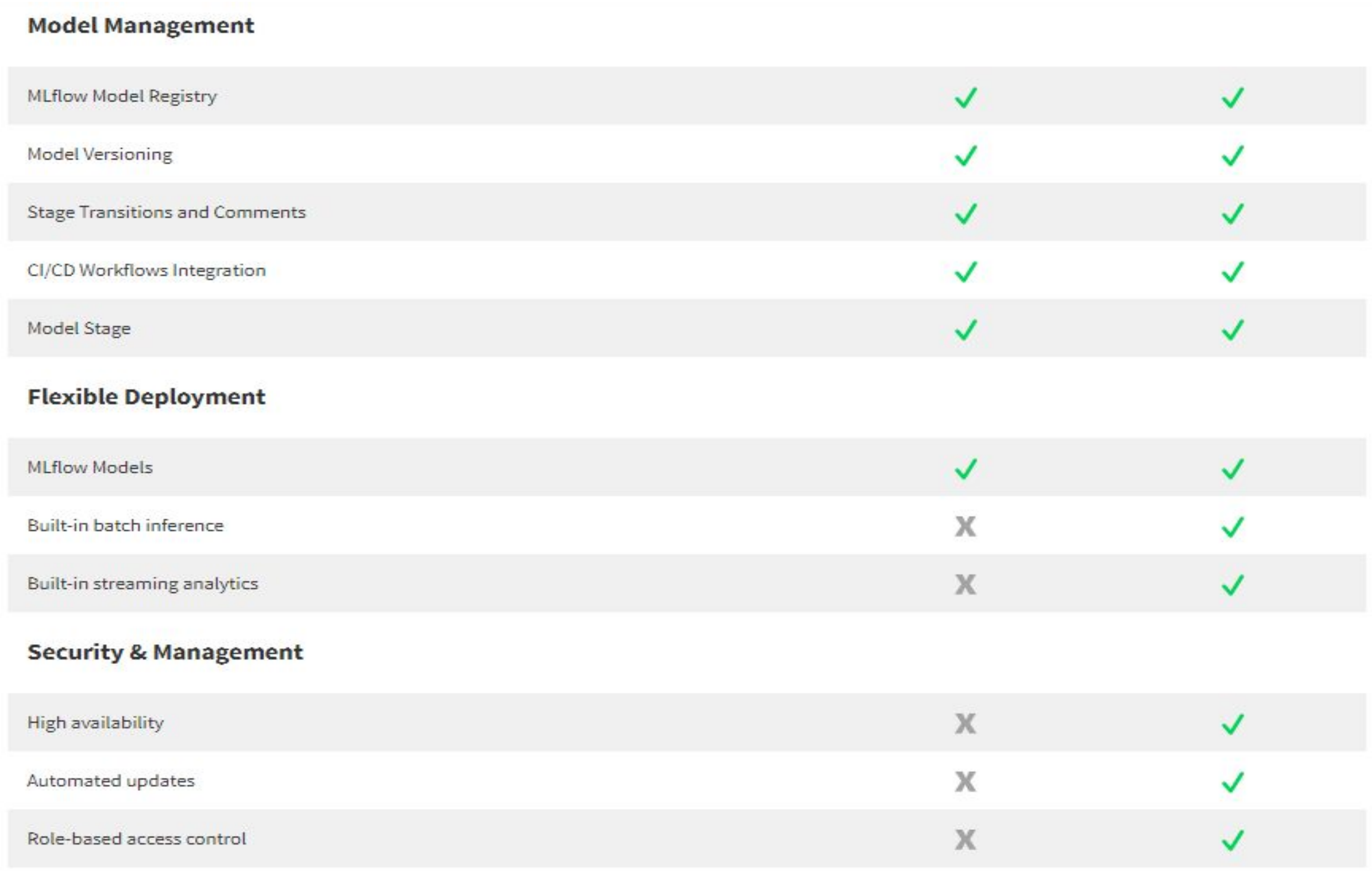
- Hosted MLflow on Databricks
- MLflow on Databricks integrates with the complete Databricks Unified Analytics Platform, including :
  - Notebooks,
  - Jobs,
  - Databricks Delta,
  - Databricks security model.
- Enabling running existing MLflow jobs at scale in a secure, production-ready manner

Managed MLFlow on Databricks



Experiment Tracking	Open Source MLflow	Managed MLflow on Databricks
MLflow tracking API	✓	✓
MLflow tracking server	Self-hosted	Fully managed
Notebooks integration	X	✓
Workspace integration	X	✓
Reproducible Projects		
MLflow Projects	✓	✓
Git & Conda integration	✓	✓
Scalable cloud/clusters for project runs	X	✓

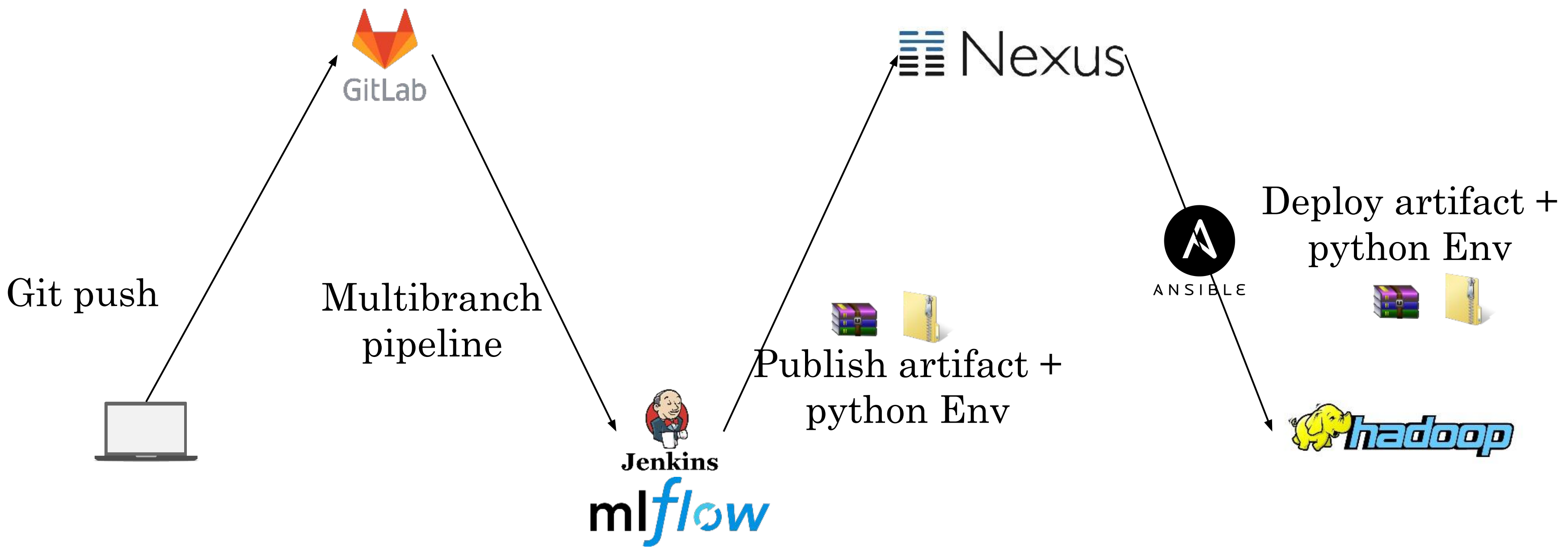
Managed MLFlow on Databricks



Model Management		
MLflow Model Registry	✓	✓
Model Versioning	✓	✓
Stage Transitions and Comments	✓	✓
CI/CD Workflows Integration	✓	✓
Model Stage	✓	✓
Flexible Deployment		
MLflow Models	✓	✓
Built-in batch inference	X	✓
Built-in streaming analytics	X	✓
Security & Management		
High availability	X	✓
Automated updates	X	✓
Role-based access control	X	✓

MLFlow with CI/CD

- Deploying python libraries in advance by the infrastructure team for all the projects is hard to manage. And it often takes a very long time.
- MLFlow offers an alternative by allowing each project to bring its necessary libraries to the infrastructure automatically in a conda environment. It improves the agility of dev teams and reduces the time to production.



Related Tools

- **KUBRFLOW :** Mlops on Kubernetes
- **AIRFLOW**: Scheduler that would offer the possibility of writing its scheduled workflow in Python.

https://github.com/apache/airflow

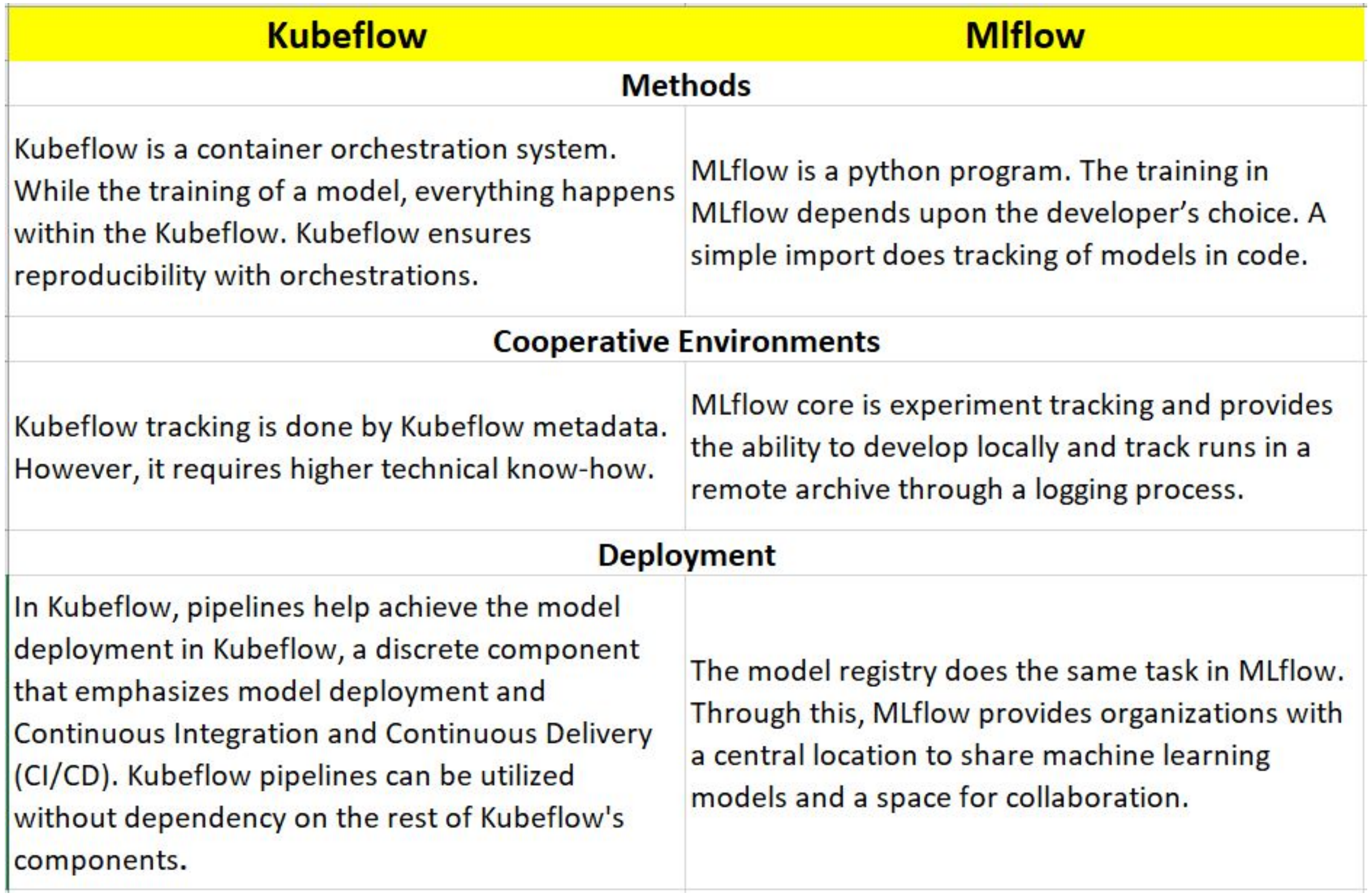
- **KEDRO:** Framework for building pipelines

Kedro

https://github.com/quantumblacklabs/kedroc

- **And also**
  - Luigi
  - Gokart
  - MetaFlow

KUBEFLOW vs MLFLOW



Kubeflow	Mlflow
Methods	
Kubeflow is a container orchestration system. While the training of a model, everything happens within the Kubeflow. Kubeflow ensures reproducibility with orchestrations.	MLflow is a python program. The training in MLflow depends upon the developer's choice. A simple import does tracking of models in code.
Cooperative Environments	
Kubeflow tracking is done by Kubeflow metadata. However, it requires higher technical know-how.	MLflow core is experiment tracking and provides the ability to develop locally and track runs in a remote archive through a logging process.
Deployment	
In Kubeflow, pipelines help achieve the model deployment in Kubeflow, a discrete component that emphasizes model deployment and Continuous Integration and Continuous Delivery (CI/CD). Kubeflow pipelines can be utilized without dependency on the rest of Kubeflow's components.	The model registry does the same task in MLflow. Through this, MLflow provides organizations with a central location to share machine learning models and a space for collaboration.

source :
https://royalcyberinc.medium.com/kubeflow-vs-mlflow-an-mlops-comparison-36db04a665d8

MLFlow references

- Library
  - https://github.com/mlflow/mlflow
- Documentation
  - https://mlflow.org/docs/latest/index.html
- Debugging
  - https://stackoverflow.com/questions/tagged/mlflow
  - https://github.com/mlflow/mlflow/issues?q=is%3Aissue+is%3Aopen
- Managed MLFlow (by Databricks)
  - https://databricks.com/product/managed-mlflow