

# mlflow

A platform for the Complete Machine  
Learning Lifecycle

DATA+AI SUMMIT EUROPE

#DataTeams #DataAIStorytelling

# Traditional Software

Goal: Meet a functional specification

Quality depends only on code

Typically pick one software stack

# Machine Learning

Goal: Optimize a metric (e.g., accuracy)

- Constantly experiment to improve it

Quality depends on input data  
and tuning parameters

Compare + combine many libraries,  
models & algorithms for the same task

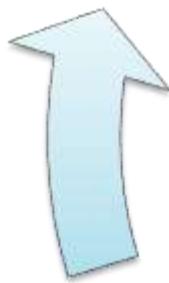
# Machine Learning Lifecycle



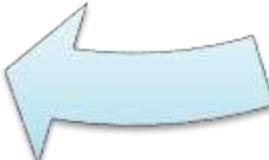
Raw  
data



Data  
prep



Deploy



Train



# Problems

---



- Reproduce experiments
- Compare experiments
- Fine tune previous experiments across teams
- Share data, parameters or metrics
- Deploy trained models

# It is difficult to productionize and share

---



# Are there any similar solutions in an open manner?

---



# MLflow: An Open Source ML Platform

Works with any ML library, programming language, deployment tool

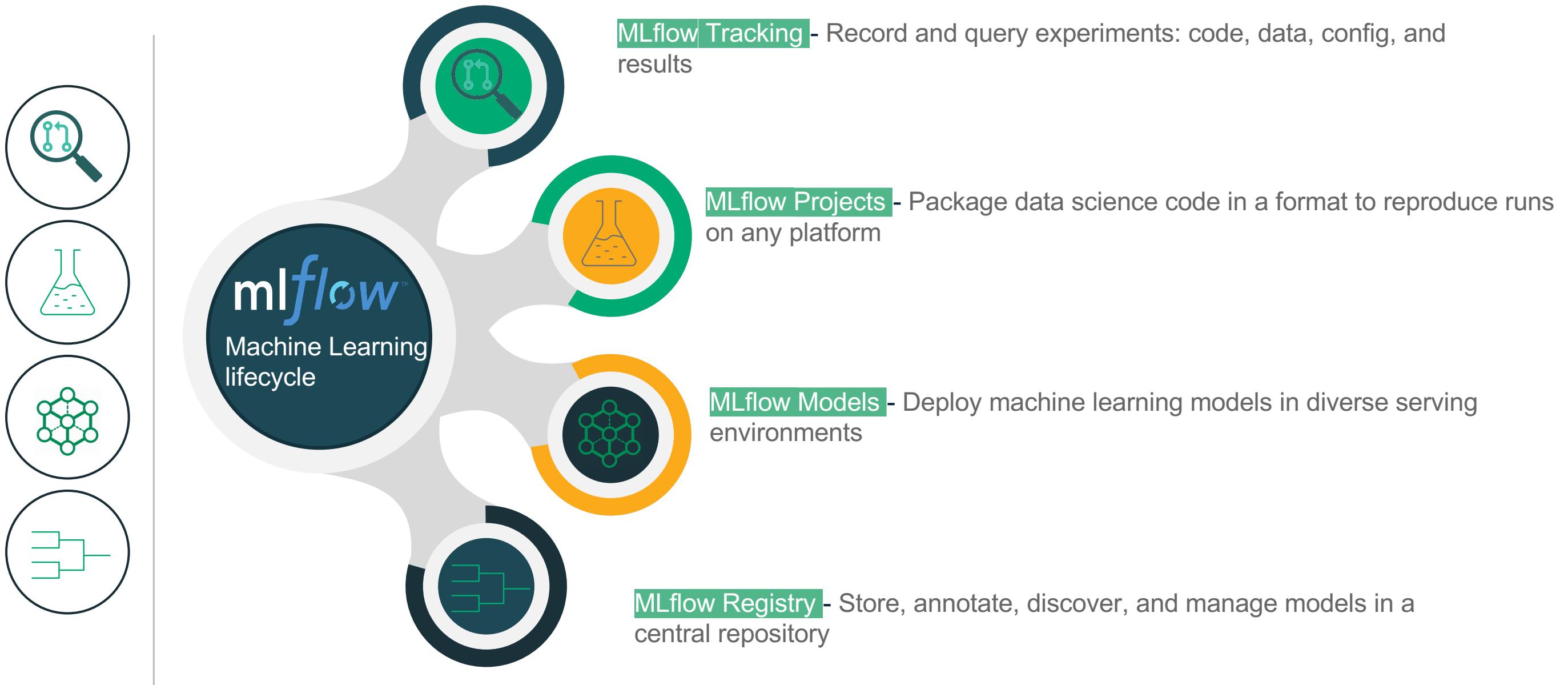


Three components:

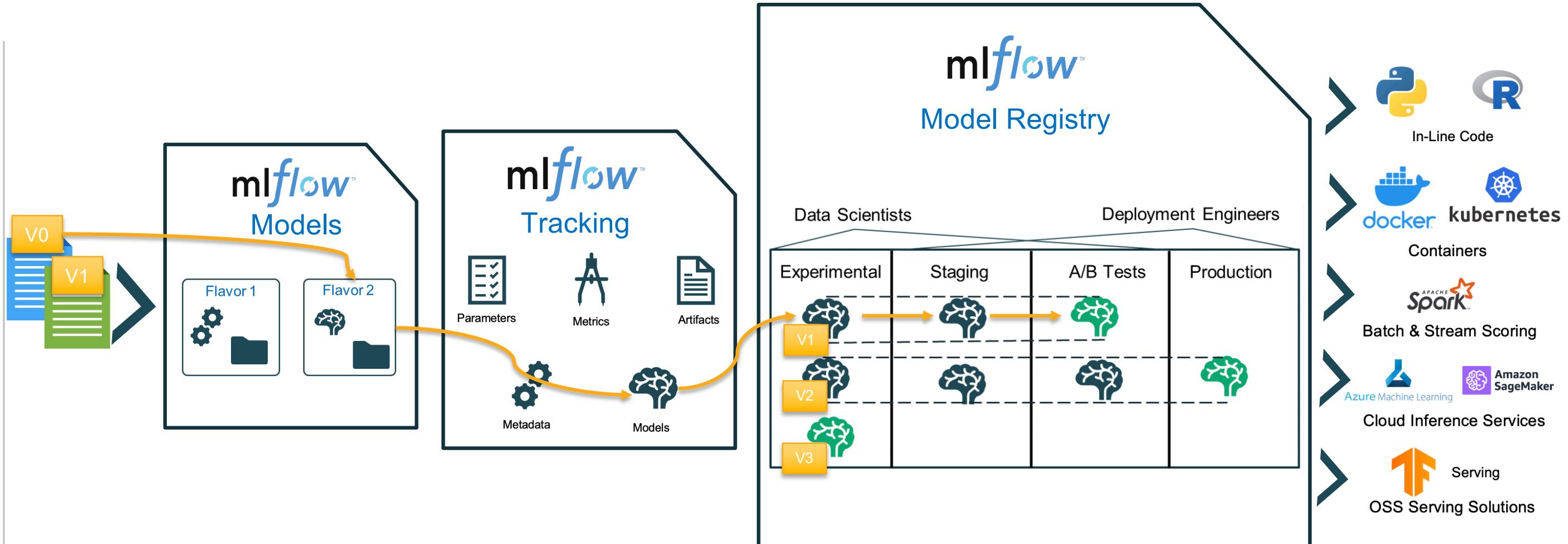
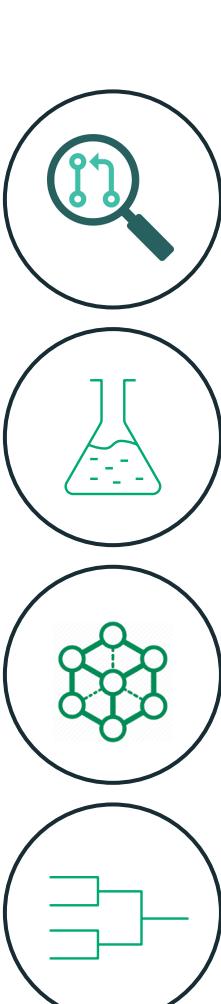
- **Tracking:** experiment tracking
- **Projects:** reproducible runs
- **Models:** model packaging

140 contributors, 800K downloads/month

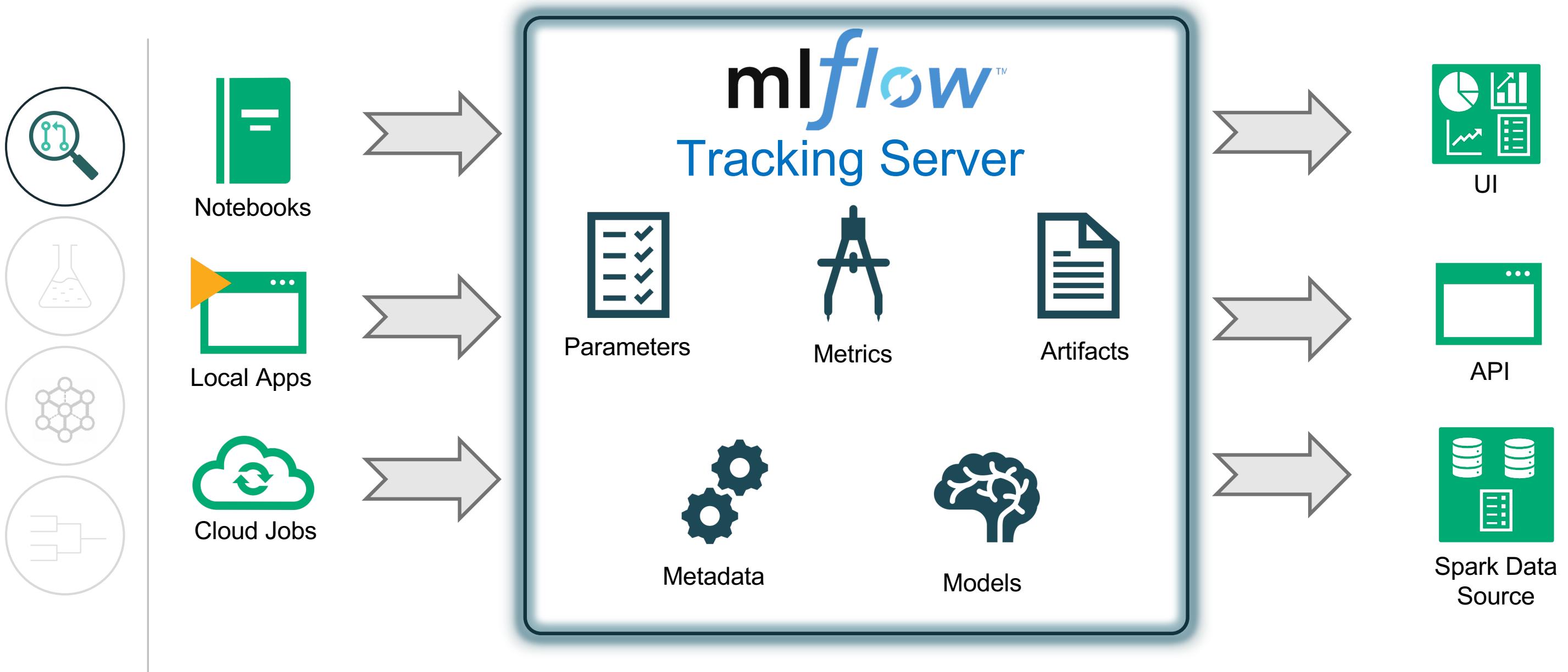
# MLFlow Components



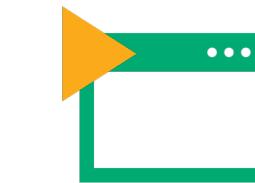
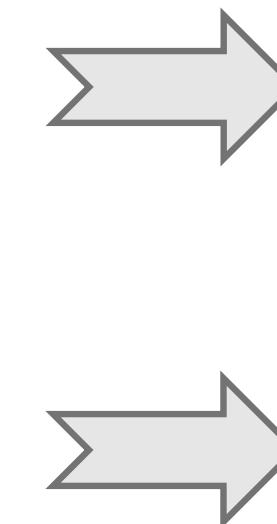
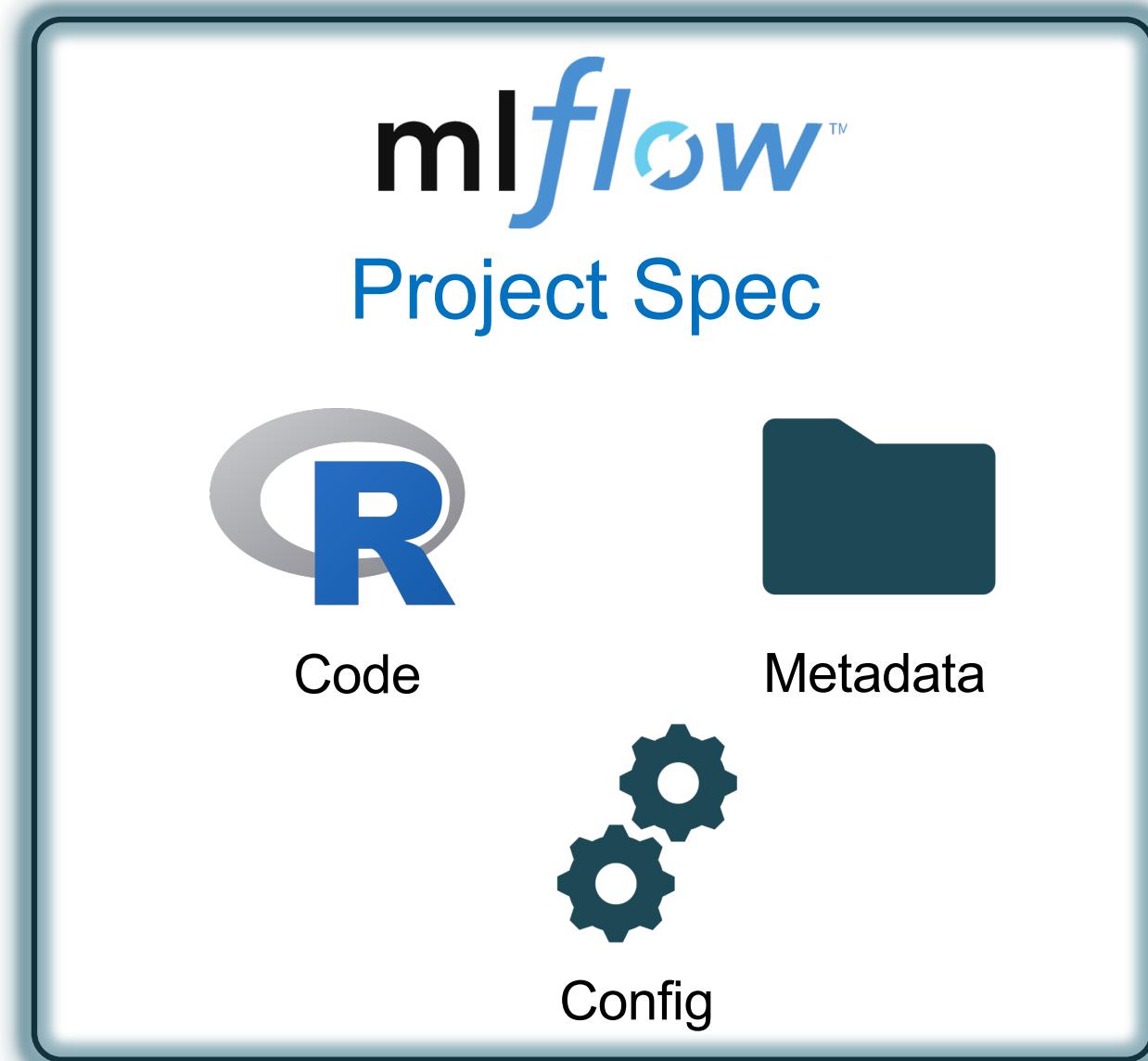
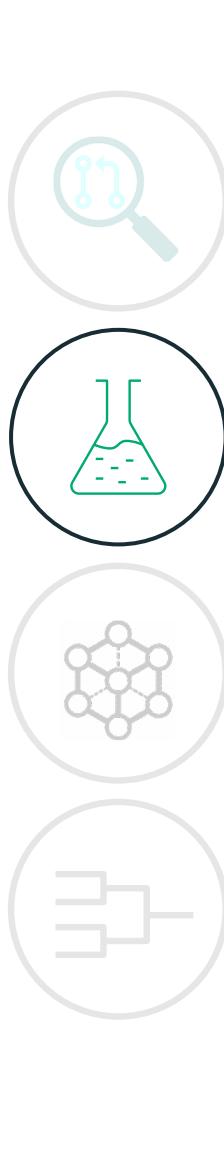
# MLflow Workflow



# MLflow Tracking



# MLflow Projects

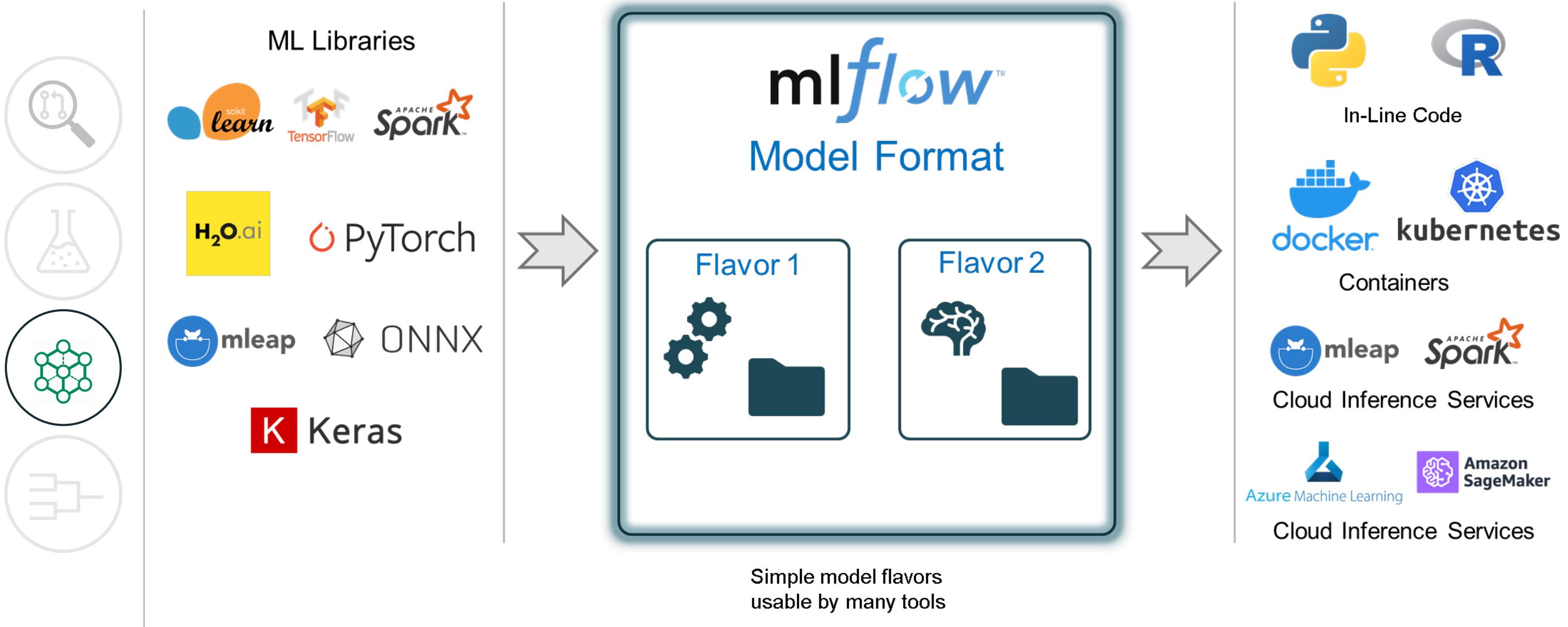


Local Execution

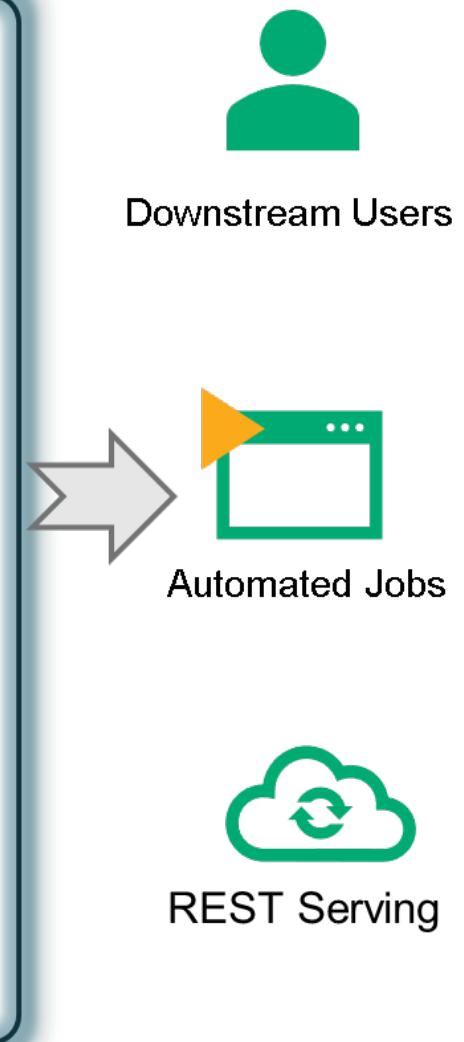
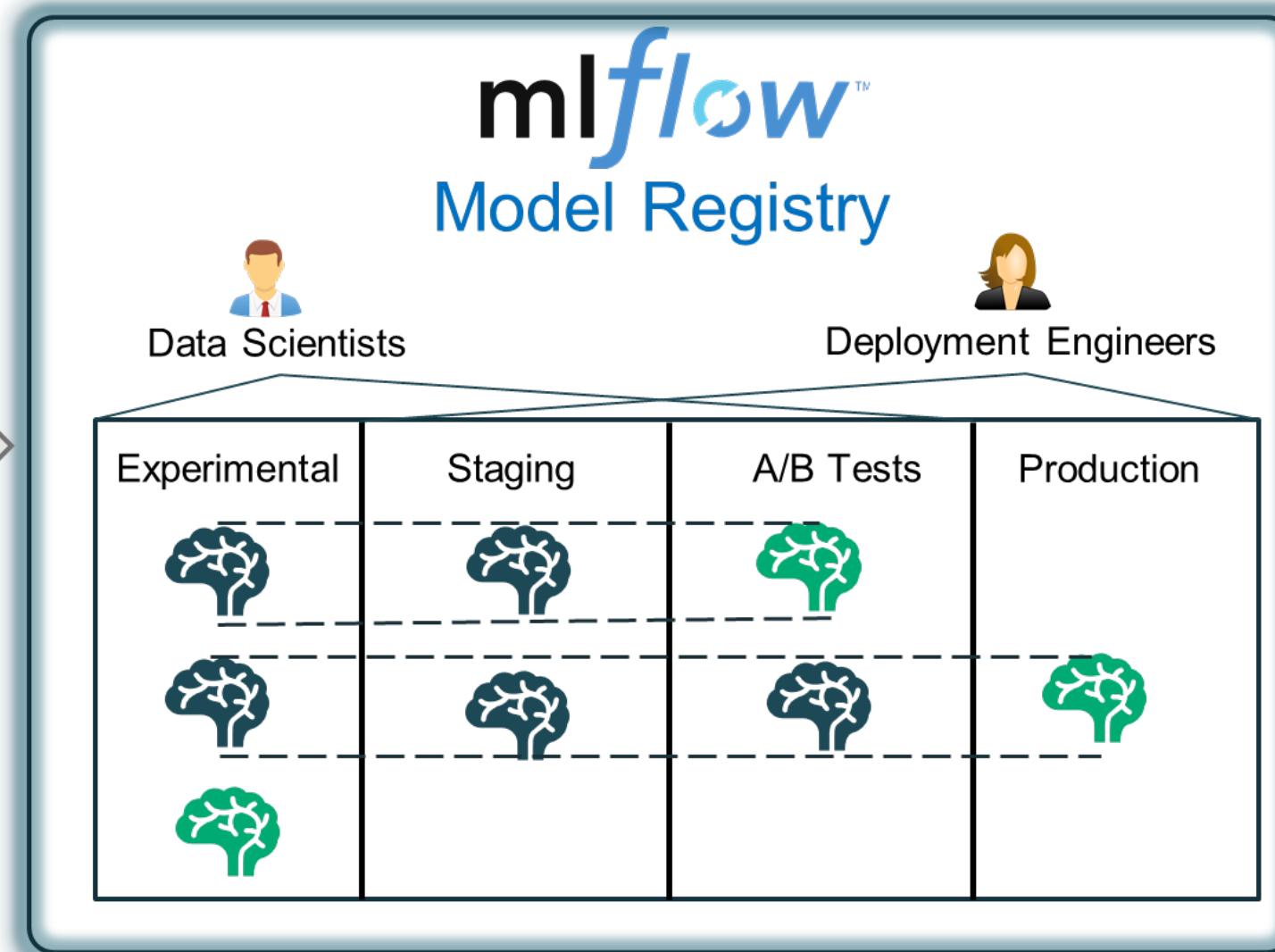
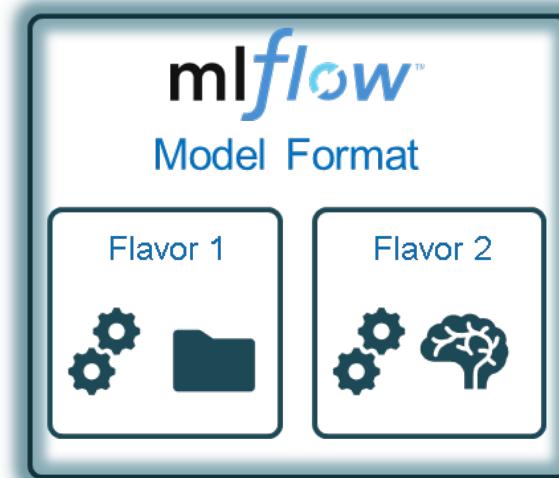


Remote Execution

# MLflow Models



# MLflow Registry



# MLFlow Tracking

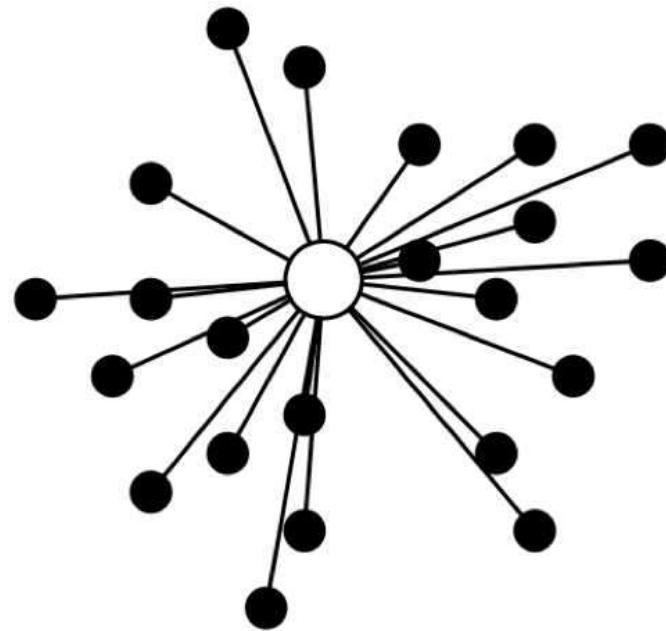
---



- Record and query experiments:
  - Data
  - Code
  - Model parameters
  - Results (performance metrics)
  - Model

# Motivation

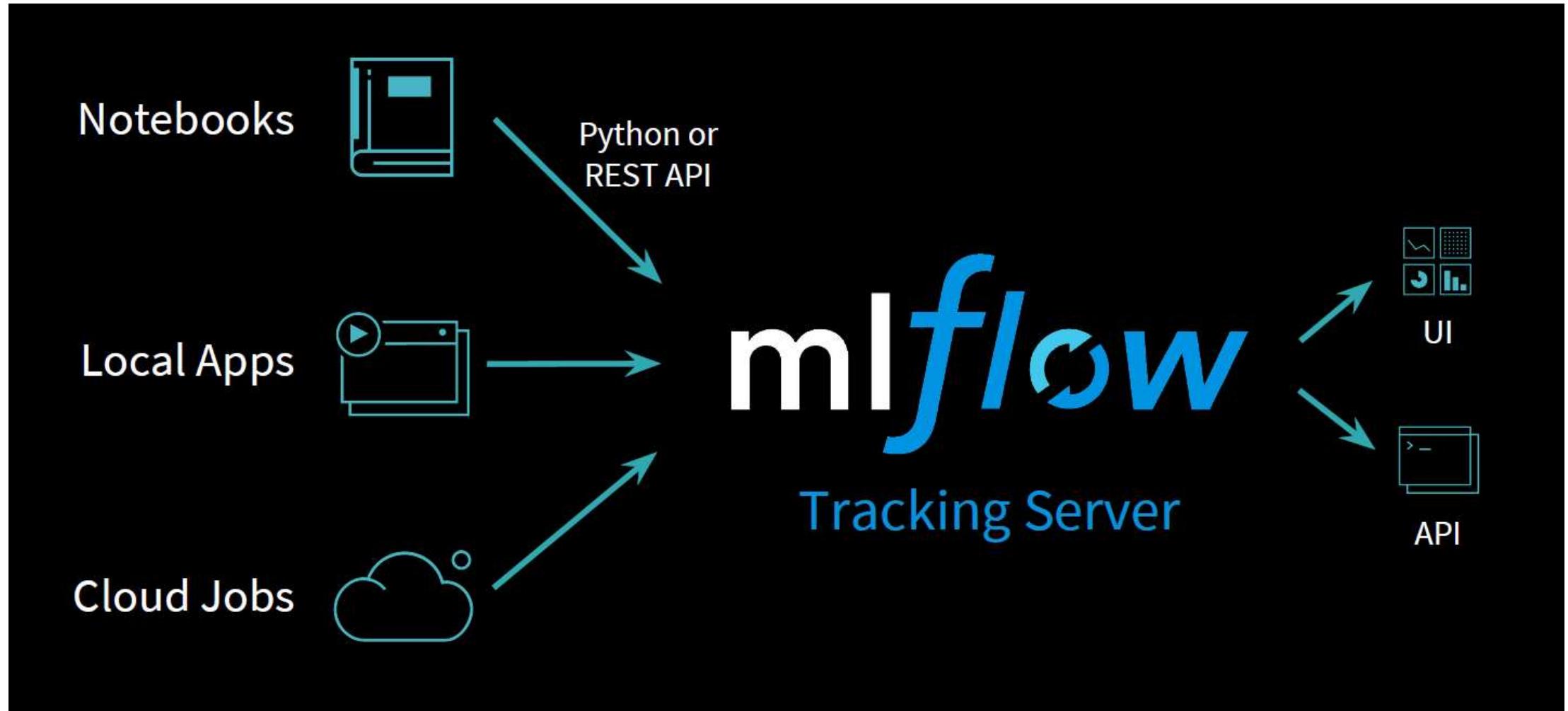
---



Centralised repository of useful information to analyse  
several runs of training.

# How it is working

---



# Key concepts

---



- Parameters
- Metrics
- Tags and notes
- Artifacts
- Source
- Version

# 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)

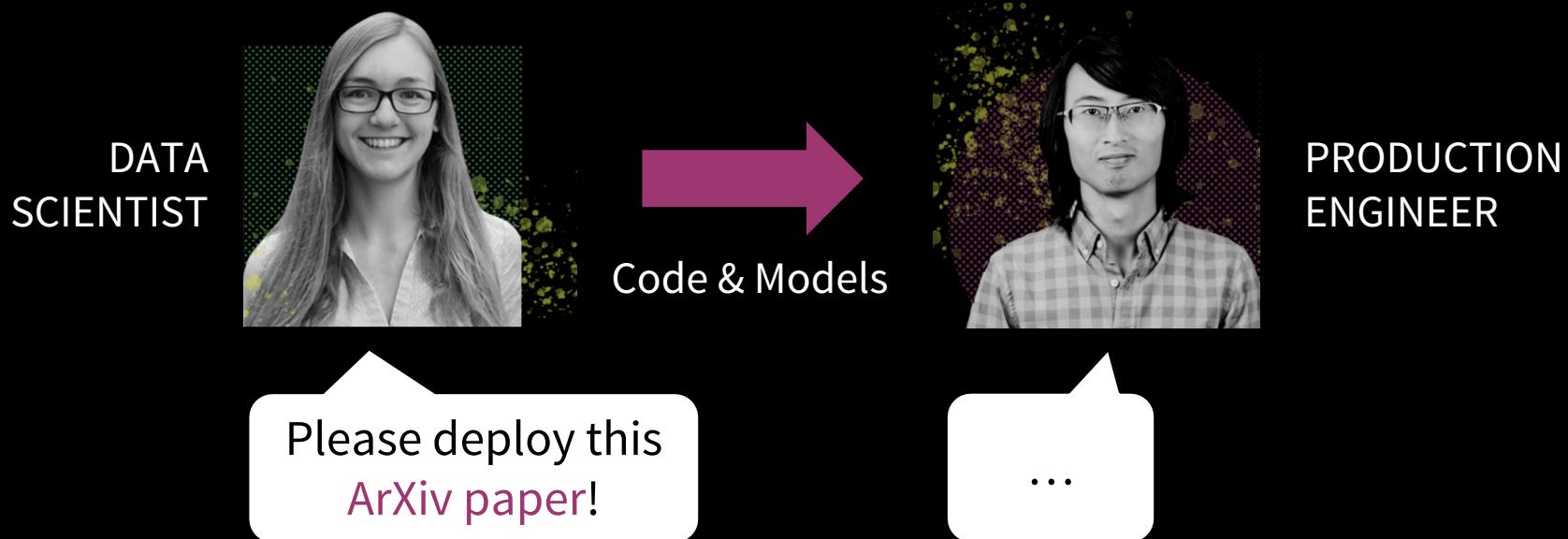
print("For n=%d, lr=%f: accuracy=%f"
      % (n, lr, score))

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
...
```

What version of  
my code was this  
result from?

# Model Deployment without MLflow



# Model Development with MLflow

```
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"))
```

# Model Development with MLflow

```
data    = load_text(file)
ngrams = extract_ngrams(data, N=n)
model   = train_model(ngrams,
                      learning_rate=lr)
score   = compute_accuracy(model)

mlflow.log_param("data_file", file)
mlflow.log_param("n", n)
mlflow.log_param("learning_rate", lr)
mlflow.log_metric("score", score)

mlflow.sklearn.log_model(model)
```

The screenshot shows the MLflow UI interface. At the top, there's a header with the MLflow logo, GitHub, and Docs links. Below the header, the title 'Language Model' is displayed. Underneath the title, it says 'Experiment ID: 0' and 'Artifact Location: /Users/matei/mlflow/mlruns/0'. There are search and filter controls: 'Search Runs:' with a dropdown for 'State: Active' and a 'Search' button; 'Filter Params:' with a dropdown for 'alpha\_lr' and a 'Filter Metrics:' dropdown with 'rmse, f1' selected, along with a 'Clear' button. Below these controls, it says '10 matching runs' with buttons for 'Compare', 'Delete', and 'Download CSV'. A table follows, with columns: Date, User, Source, Version, input\_file, lr, n, accuracy, and f1. The data rows are:

Date	User	Source	Version	input_file	lr	n	accuracy	f1
2018-10-02 21:53:57	matei	lang_model.py	e55d56	data.txt	2.0	1	0.77	0.704
2018-10-02 21:53:55	matei	lang_model.py	e55d56	data.txt	2.0	4	0.835	0.609
2018-10-02 21:53:48	matei	lang_model.py	e55d56	data.txt	2.0	2	0.66	0.476
2018-10-02 21:53:53	matei	lang_model.py	e55d56	data.txt	1.0	1	0.663	0.468
2018-10-02 21:53:49	matei	lang_model.py	e55d56	data.txt	1.0	4	0.902	0.461

Track parameters, metrics,  
output files & code version

Search using UI or API

# MLflow UI: Inspecting Runs

mlflow GitHub Docs

## Language Model

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

Search Runs: metrics.rmse < 1 and params.model = "tree" State: Active ▾ Search

Filter Params: alpha, lr      Filter Metrics: rmse, r2 Clear

10 matching runs Compare Delete Download CSV

	Date ▾	User	Source	Version	Parameters			Metrics	
					input_file	lr	n	accuracy	f1
<input type="checkbox"/>	2018-10-02 21:53:57	matei	lang_model.py	e55d56	data.txt	2.0	1	0.77	0.704
<input type="checkbox"/>	2018-10-02 21:53:56	matei	lang_model.py	e55d56	data.txt	1.0	2	0.254	0.222
<input type="checkbox"/>	2018-10-02 21:53:55	matei	lang_model.py	e55d56	data.txt	2.0	4	0.835	0.609
<input type="checkbox"/>	2018-10-02 21:53:53	matei	lang_model.py	e55d56	data.txt	1.0	1	0.663	0.468
<input type="checkbox"/>	2018-10-02 21:53:52	matei	lang_model.py	e55d56	data.txt	0.2	4	0.034	0.032
<input type="checkbox"/>	2018-10-02 21:53:51	matei	lang_model.py	e55d56	data.txt	0.1	4	0.177	0.16

databricks

# MLflow UI: Comparing Runs

mlflow GitHub Docs

## Language Model

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

Search Runs: metrics.rmse < 1 and params.model = "tree"      State: Active ▾      Search

Filter Params: alpha, lr      Filter Metrics: rmse, r2      Clear

10 matching runs      Compare      Delete      Download CSV

	Date ▾	User	Source	Version	Parameters			Metrics	
					input_file	lr	n	accuracy	f1
<input type="checkbox"/>	2018-10-02 21:53:57	matei	lang_model.py	e55d56	data.txt	2.0	1	0.77	0.704
<input type="checkbox"/>	2018-10-02 21:53:56	matei	lang_model.py	e55d56	data.txt	1.0	2	0.254	0.222
<input type="checkbox"/>	2018-10-02 21:53:55	matei	lang_model.py	e55d56	data.txt	2.0	4	0.835	0.609
<input type="checkbox"/>	2018-10-02 21:53:53	matei	lang_model.py	e55d56	data.txt	1.0	1	0.663	0.468
<input type="checkbox"/>	2018-10-02 21:53:52	matei	lang_model.py	e55d56	data.txt	0.2	4	0.034	0.032
<input type="checkbox"/>	2018-10-02 21:53:51	matei	lang_model.py	e55d56	data.txt	0.1	4	0.177	0.16

databricks

# Code example

---

```
1 import mlflow
2
3 with mlflow.start_run():
4     mlflow.log_param("layers", layers)
5     mlflow.log_param("alpha", alpha)
6
7     # train model
8     model = train_model(layers, alpha)
9
10    mlflow.log_metric("mse", model.mse())
11    mlflow.log_artifact("plot", model.plot(test_df))
12    mlflow.tensorflow.log_model(model
```

# Demo

```
# Enable MLflow Autologging
mlflow.keras.autolog()

X, y = get_training_data()opt = keras.optimizers.Adam(lr=params["learning_rate"],
                                                    beta_1=params["beta_1"],
                                                    beta_2=params["beta_2"],
                                                    epsilon=params["epsilon"])

model = Sequential()
model.add(Dense(int(params["units"])), ...)
model.add(Dense(1))
model.compile(loss="mse", optimizer=opt)

rest = model.fit(X, y, epochs=50, batch_size=64, validation_split=.2)
```

# MLflow Autologging

```
model = keras.models.Sequential()
model.add(layers.Dense(hidden_units, ...))
model.fit(X_train, y_train)
test_loss = model.evaluate(X_test, y_test)
```

# MLflow Autologging

```
with mlflow.start_run():
    model = keras.models.Sequential()
    model.add(layers.Dense(hidden_units, ...))
    model.fit(X_train, y_train)
    test_loss = model.evaluate(X_test, y_test)
    mlflow.log_param("hidden_units", hidden_units)
    mlflow.log_param("learning_rate", learning_rate)
    mlflow.log_metric("train_loss", train_loss)
    mlflow.log_metric("test_loss", test_loss)
    mlflow.keras.log_model(model)
```



```
mlflow.keras.autolog()
model = keras.models.Sequential()
model.add(layers.Dense(hidden_units, ...))
model.fit(X_train, y_train)
test_loss = model.evaluate(X_test, y_test)
```

# MLFlow Projects

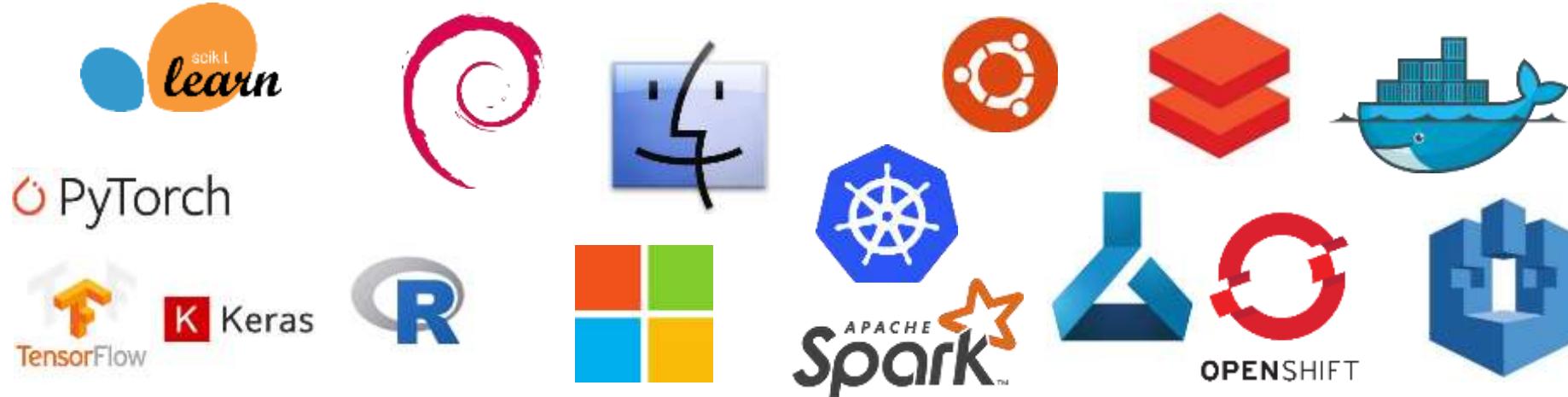
---



- Packaging format for reproducible ML runs
- Defines dependencies for reproducibility
- Execution API for running projects locally or remote

# Motivation

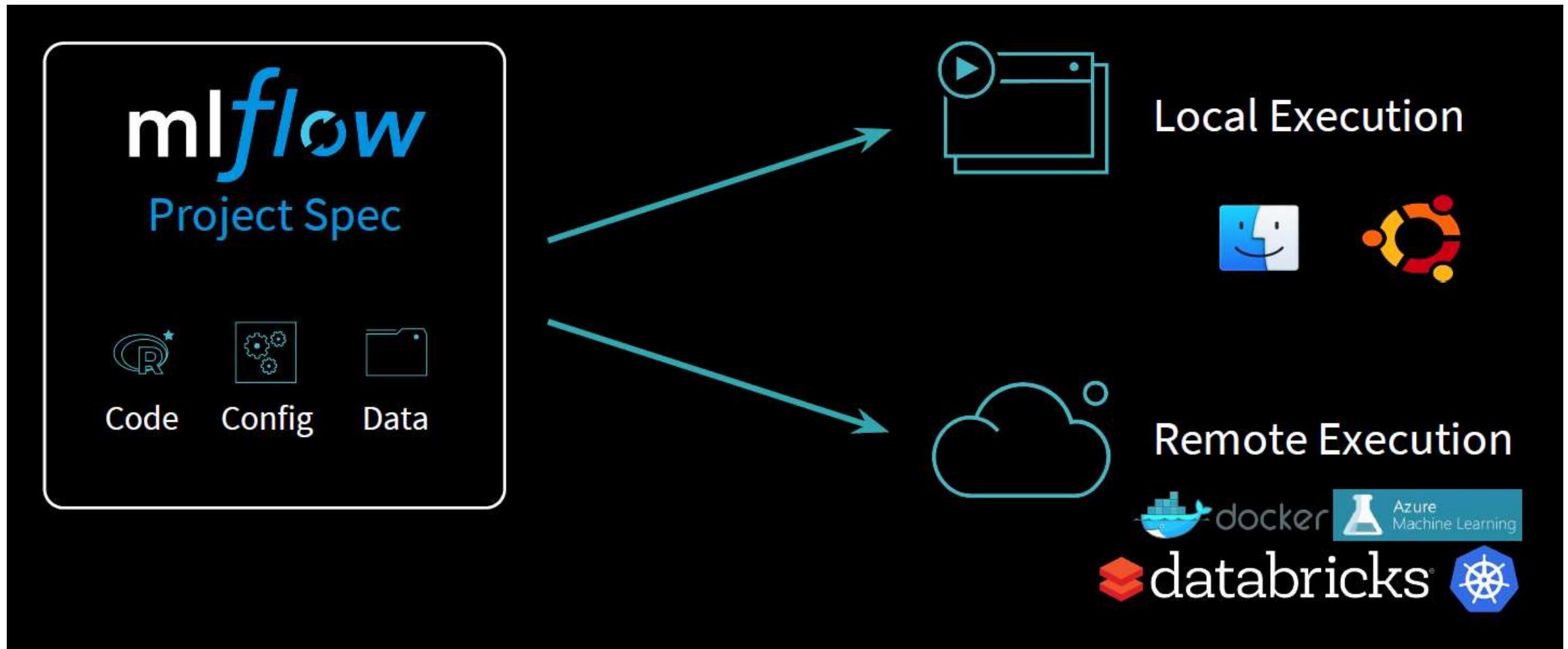
---



Diverse set of tools and environments involves difficulty  
to productionalize and share ML work

# How it is working

---



# Key concepts

---



- MLproject file
- Entry points
- Environments
  - Conda
  - Docker
  - System
- Run

# Code example

---

The diagram illustrates the relationship between a project directory structure and its corresponding MLflow configuration files. On the left, a tree view shows a project named 'my\_project' containing an 'MLproject' file and several Python source files ('main.py', 'model.py', etc.). A blue arrow points from the 'MLproject' file towards the configuration code on the right. On the right, the contents of the 'MLproject' file are displayed as a YAML document:

```
1 conda_env: conda.yaml
2
3 entry_points:
4     main:
5         parameters:
6             training_data: path
7             lambda: {type: float, default: 0.1}
8         command: python main.py {training_data} {lambda}
```

```
mlflow run git@github.com:mlflow/mlflow-example.git -P alpha=0.5
```

```
mlflow run <uri> -m databricks --cluster-spec <json-cluster-spec>
```

# Model Deployment *with* MLflow

DATA  
SCIENTIST



PRODUCTION  
ENGINEER



Please run this  
**MLflow Project**  
nightly for updates!

Don't even tell me  
what ArXiv paper  
that's from...

# MLFlow Models

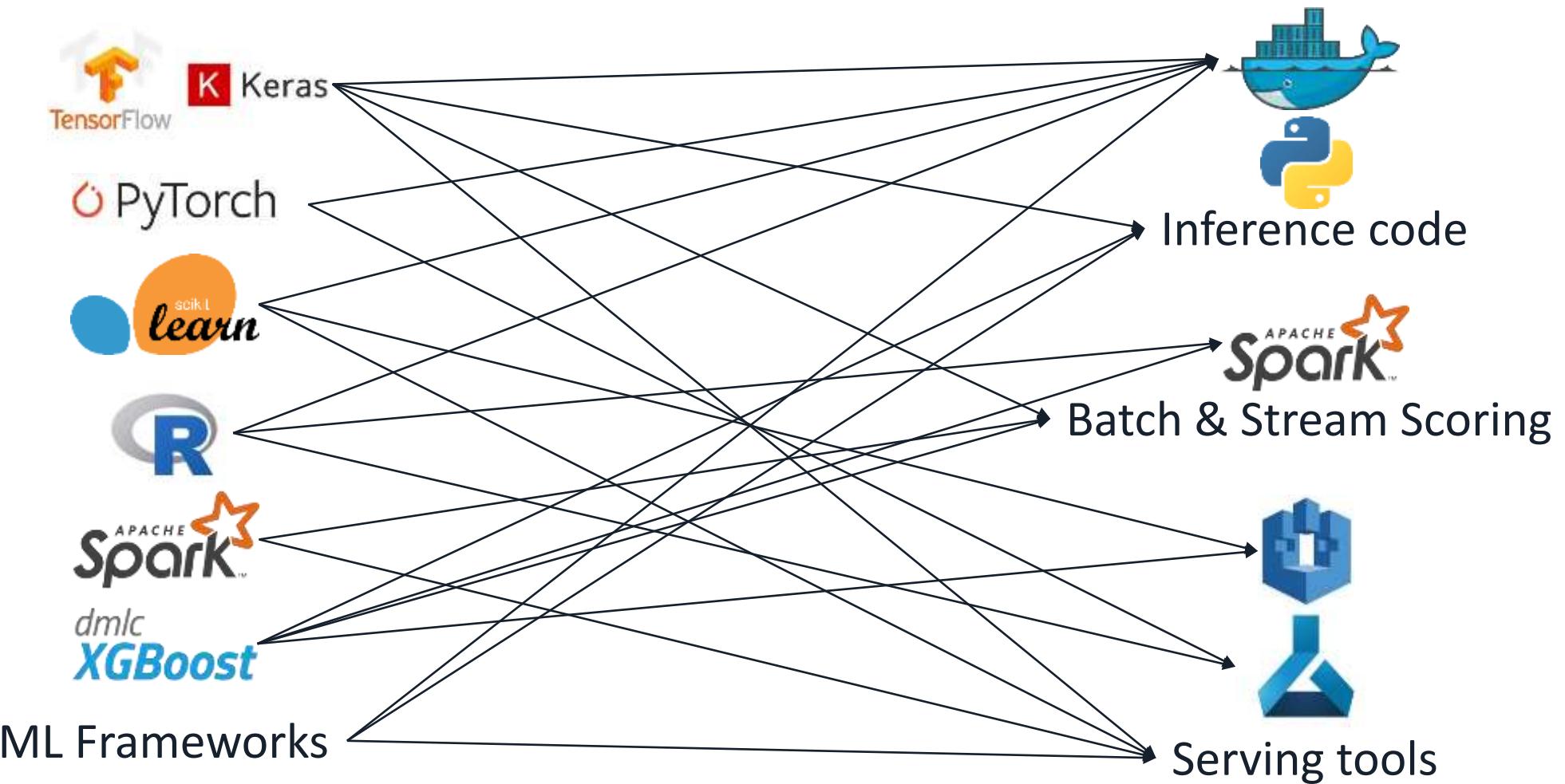
---



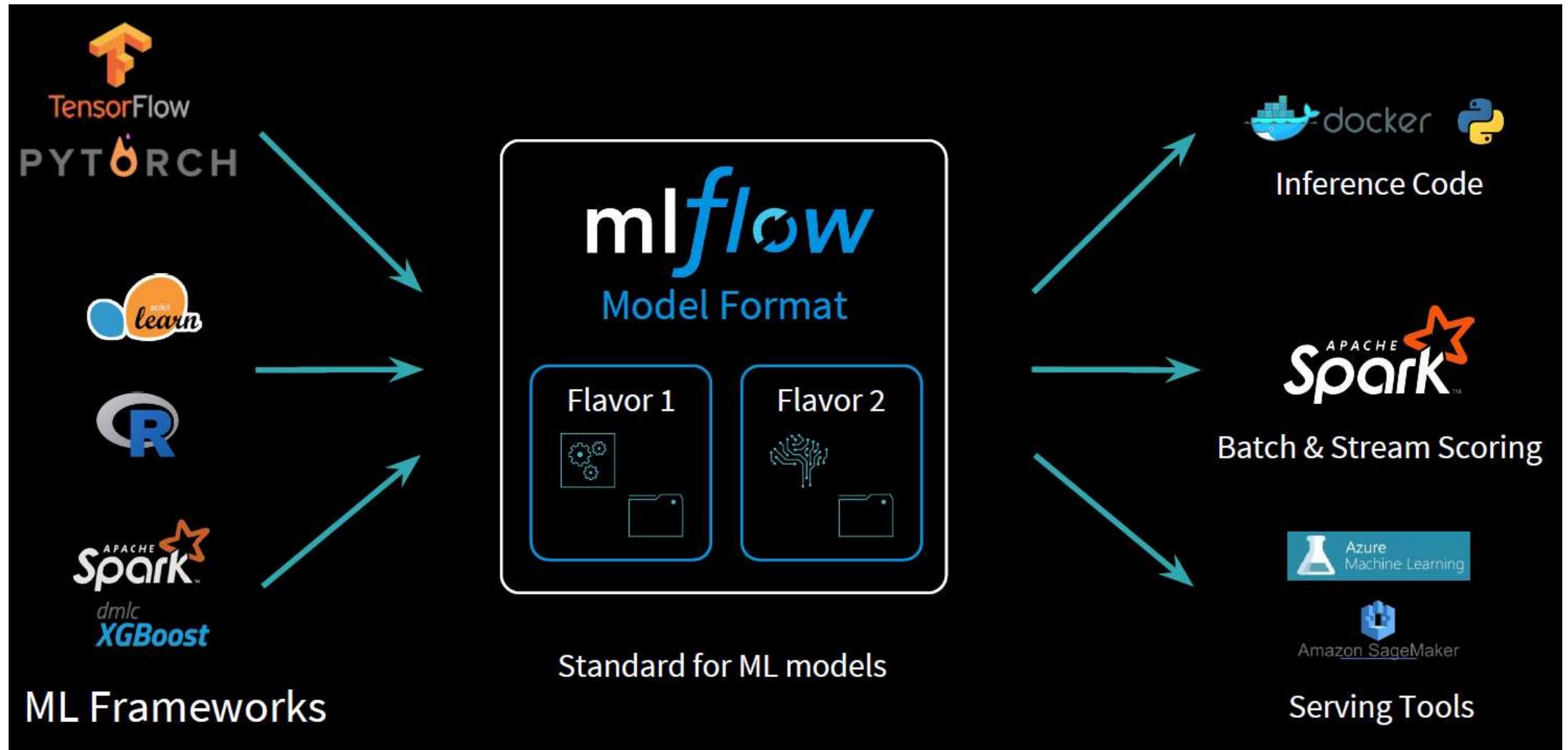
- Packaging format for ML Models
- Defines dependencies for reproducibility
- Deployment APIs

# Motivation

---



# How it is working



# Key concepts

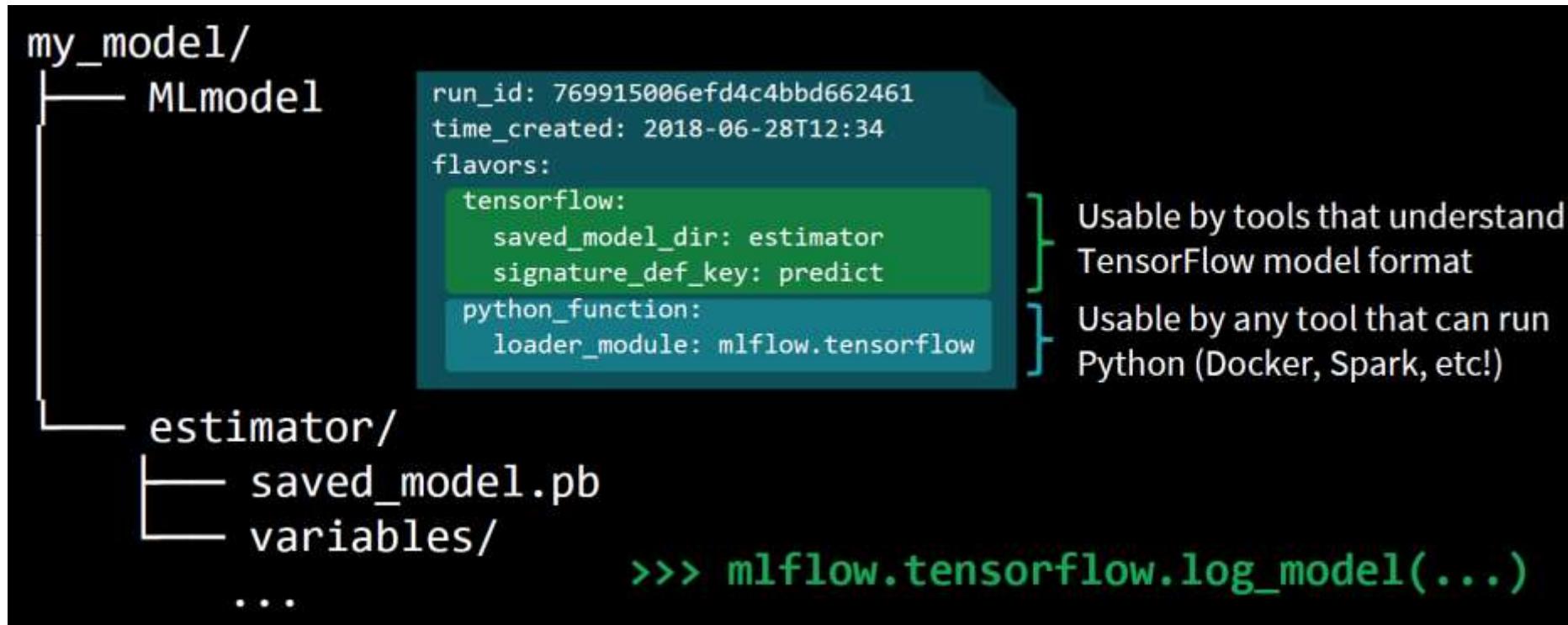
---



- MLmodel file
- Storage format
- Entry points
- Flavours
- Custom model

# Code example

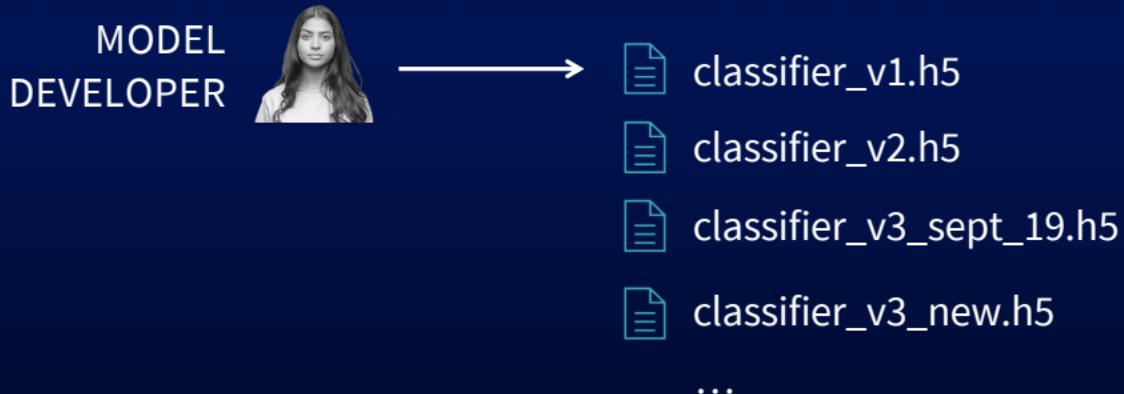
---



# MLflow's Next Goal: Model Management

# The Model Management Problem

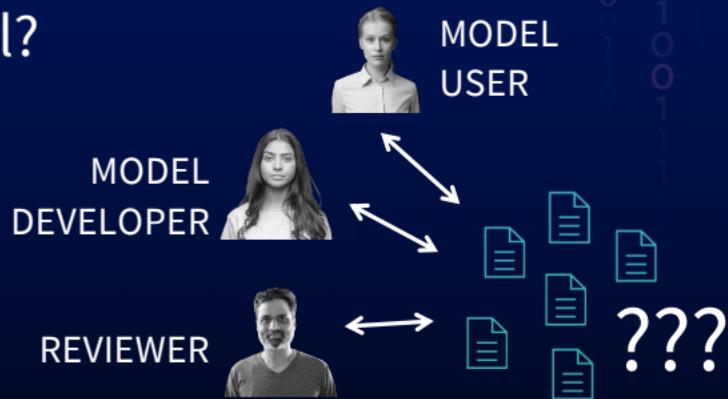
When you're working on one ML app alone, storing your models in files is manageable



# The Model Management Problem

When you work in a **large organization** with **many models**, 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

Repository of named, versioned models with comments & tags

Track each model's stage: dev, staging, production, archived

Easily load a specific version

Registered Models > Airline\_Delay\_SparkML ▾

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

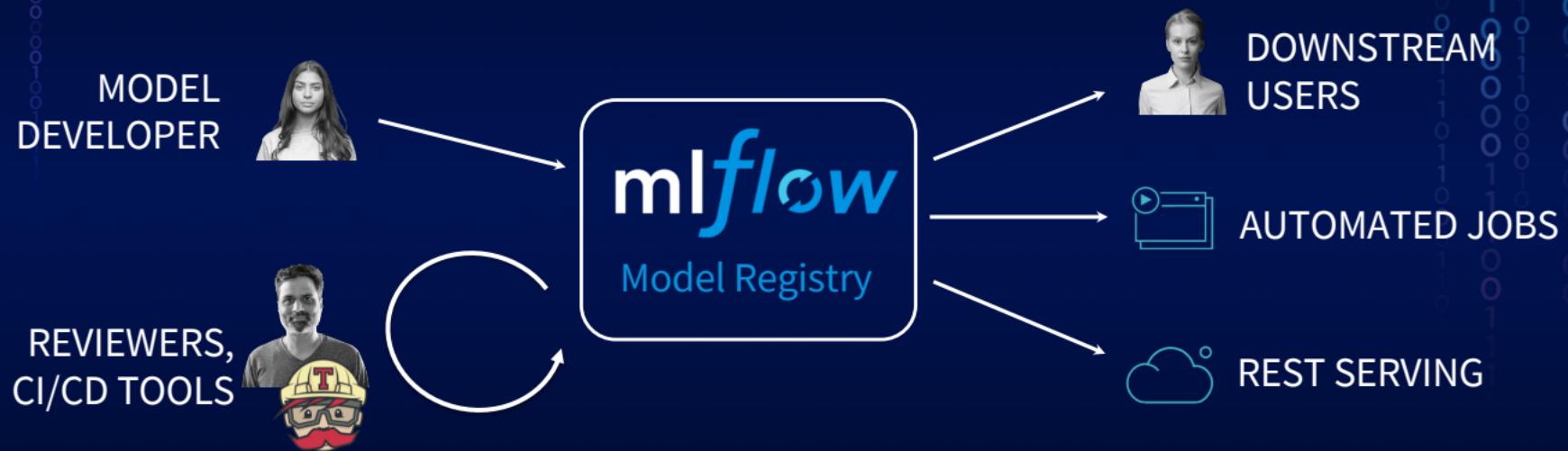
▼ Description

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

▼ Versions 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

# Model Registry Workflow



# MLFlow rocks!

---



- **Log** important parameters, metrics, and other data that is important to the machine learning model
- **Track** the environment a model is run on
- **Run any** machine learning codes on that environment
- **Deploy and export** models to various platforms with multiple packaging formats