mlflow

A platform for the Complete Machine Learning Lifecycle

Corey Zumar

June 24th, 2019

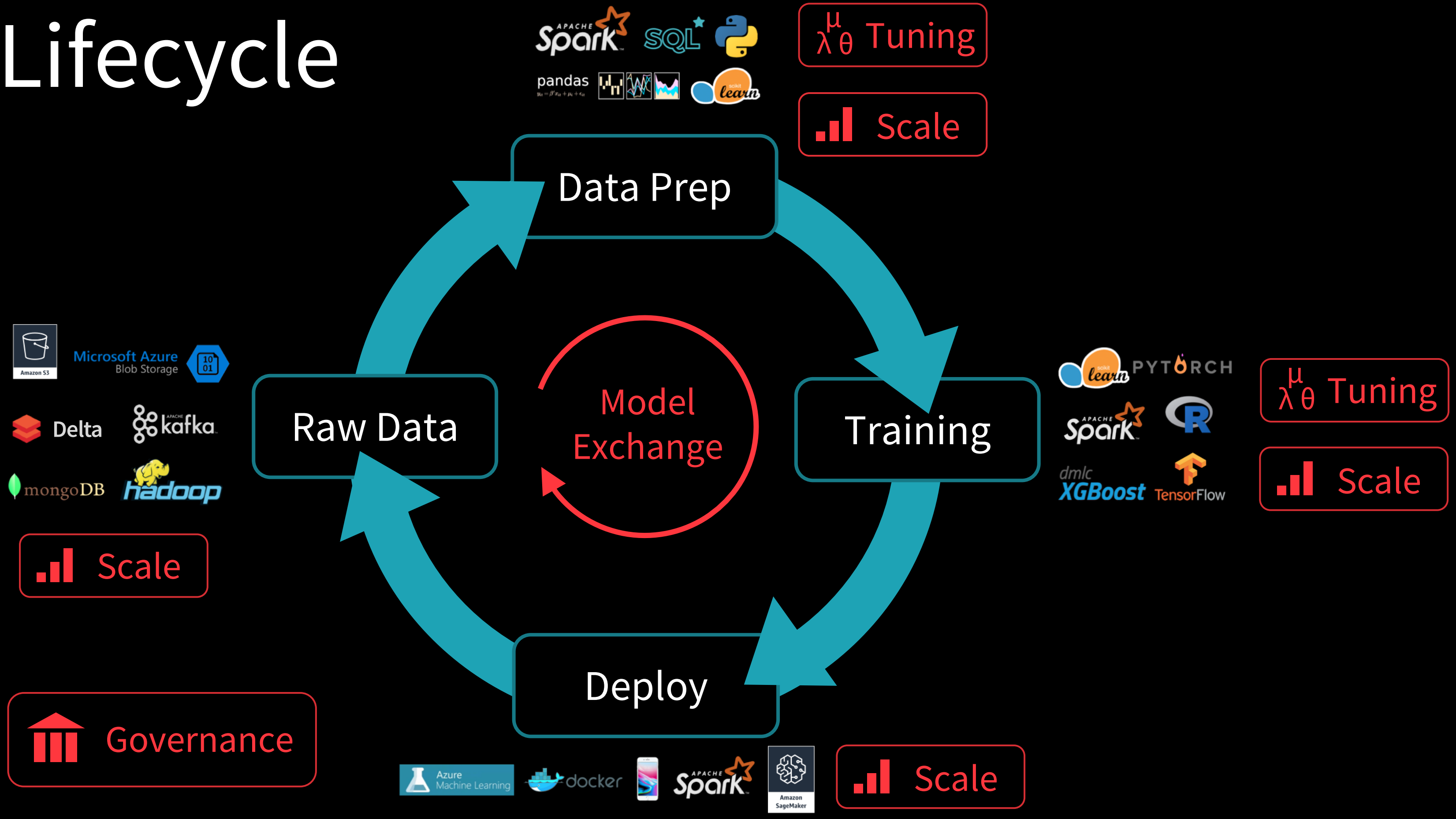
databricks

Outline

- Overview of ML development challenges
- How MLflow tackles these challenges
- MLflow components
- Demo
- How to get started

Machine Learning Development is Complex

ML Lifecycle



Custom ML Platforms

Facebook FBLearner, Uber Michelangelo, Google TFX

- + 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

Can we provide similar benefits in an open manner?

Introducing mlflow

Open machine learning platform

- Works with any ML library & language
- Runs the same way anywhere (e.g. any cloud)
- Designed to be useful for 1 or 1000+ person orgs

MLflow Components

mlflow
Tracking

Record and query experiments: code, configs, results, …etc

mlflow
Projects

Packaging format for reproducible runs on any platform

mlflow
Models

General model format that supports diverse deployment tools

Key Concepts in Tracking

Parameters: key-value inputs to your code

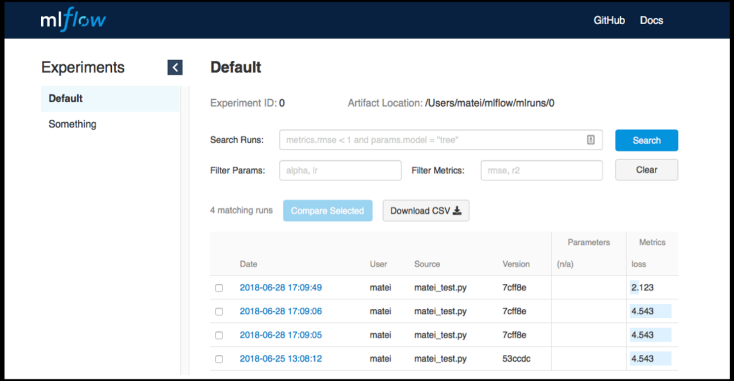
Metrics: numeric values (can update over time)

Source: training code that ran

Version: version of the training code

Artifacts: files, including data and models

Tags and Notes: any additional information



Default

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

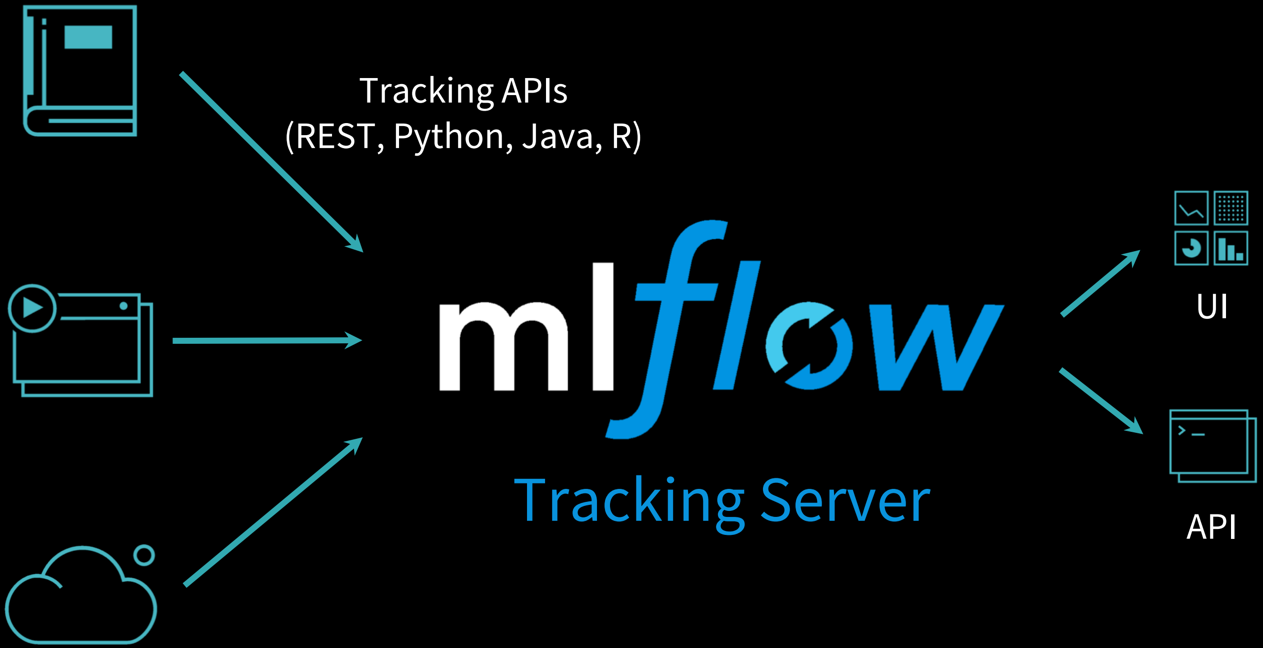
Search Runs: metrics.rmse < 1 and params.model = "tree"

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

4 matching runs

					Parameters	Metrics
	Date	User	Source	Version	(n/a)	loss
☐	2018-06-28 17:09:49	matei	matei_test.py	7cff8e		2.123
☐	2018-06-28 17:09:06	matei	matei_test.py	7cff8e		4.543
☐	2018-06-28 17:09:05	matei	matei_test.py	7cff8e		4.543
☐	2018-06-25 13:08:12	matei	matei_test.py	53ccdc		4.543

MLflow Tracking



MLflow Tracking

mlflow
Tracking

Record and query experiments: code, configs, results, …etc

```
import mlflow

with mlflow.start_run():
  mlflow.log_param("layers", layers)
  mlflow.log_param("alpha", alpha)

  # train model

  mlflow.log_metric("mse", model.mse())
  mlflow.log_artifact("plot", model.plot(test_df))
  mlflow.tensorflow.log_model(model)
```

Demo

Goal: Classify hand-drawn digits

1. Instrument Keras training code with MLflow tracking APIs
2. Run training code as an MLflow Project
3. Deploy an MLflow Model for real-time serving

MLflow backend stores

1. **Entity (Metadata) Store**
   - FileStore (local filesystem)
   - SQLStore (via SQLAlchemy)
   - REST Store

2. **Artifact Store**
   - S3 backed store
   - Azure Blob storage
   - Google Cloud storage
   - DBFS artifact repo

MLflow Components

mlflow
Tracking

Record and query experiments: code, configs, results, …etc

mlflow
Projects

Packaging format for reproducible runs on any platform

mlflow
Models

General model format that supports diverse deployment tools

MLflow Projects Motivation

Diverse set of training tools

PYTORCH TensorFlow scikit learn APACHE Spark dmlc XGBoost R

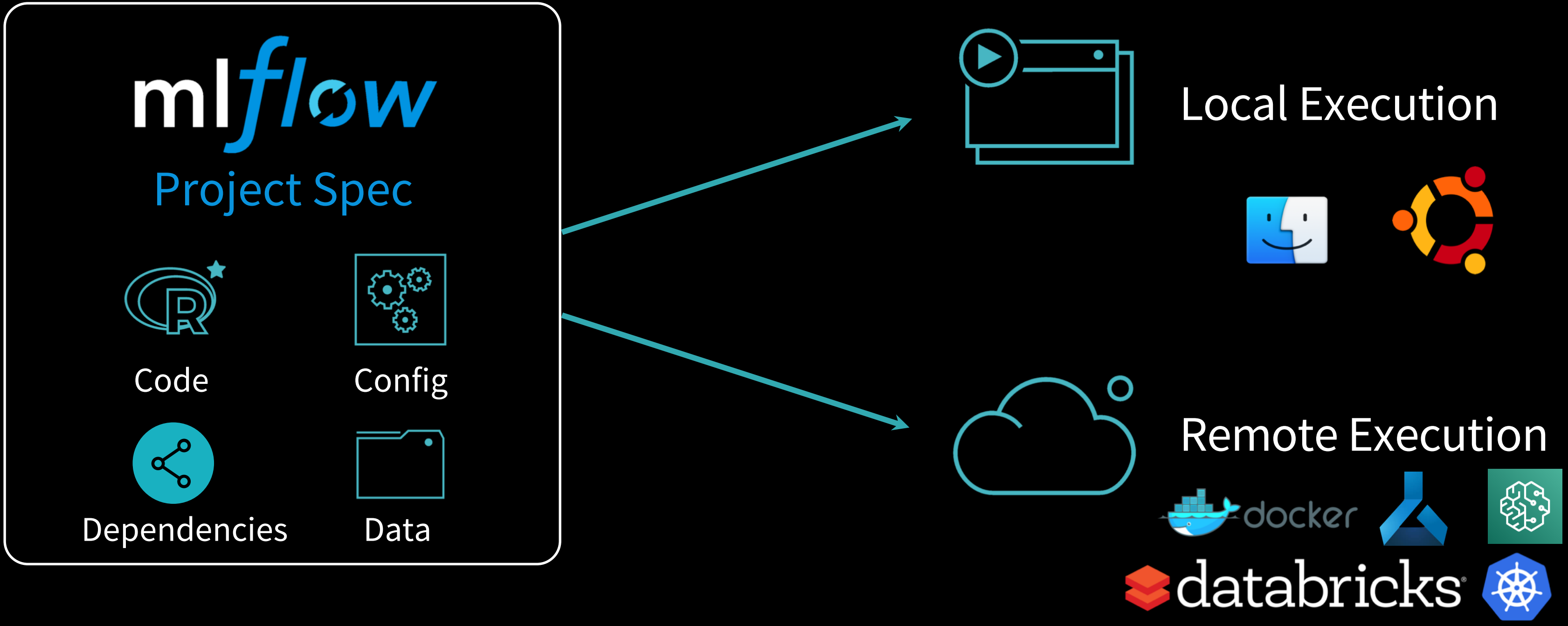
Diverse set of environments

docker databricks

Challenge:

ML results are difficult to reproduce.

MLflow Projects



MLflow Projects

Packaging format for reproducible ML runs

- Any code folder or GitHub repository
- Optional MLproject file with project configuration

Defines dependencies for reproducibility

- Conda (+ R, Docker, …) dependencies can be specified in MLproject
- Reproducible in (almost) any environment

Execution API for running projects

- CLI / Python / R / Java
- Supports local and 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>
```

Demo

Goal: Classify hand-drawn digits

1. Instrument Keras training code with MLflow tracking APIs
2. Run training code as an MLflow Project
3. Deploy an MLflow Model for real-time serving

MLflow Components

mlflow
Tracking

Record and query experiments: code, configs, results, …etc

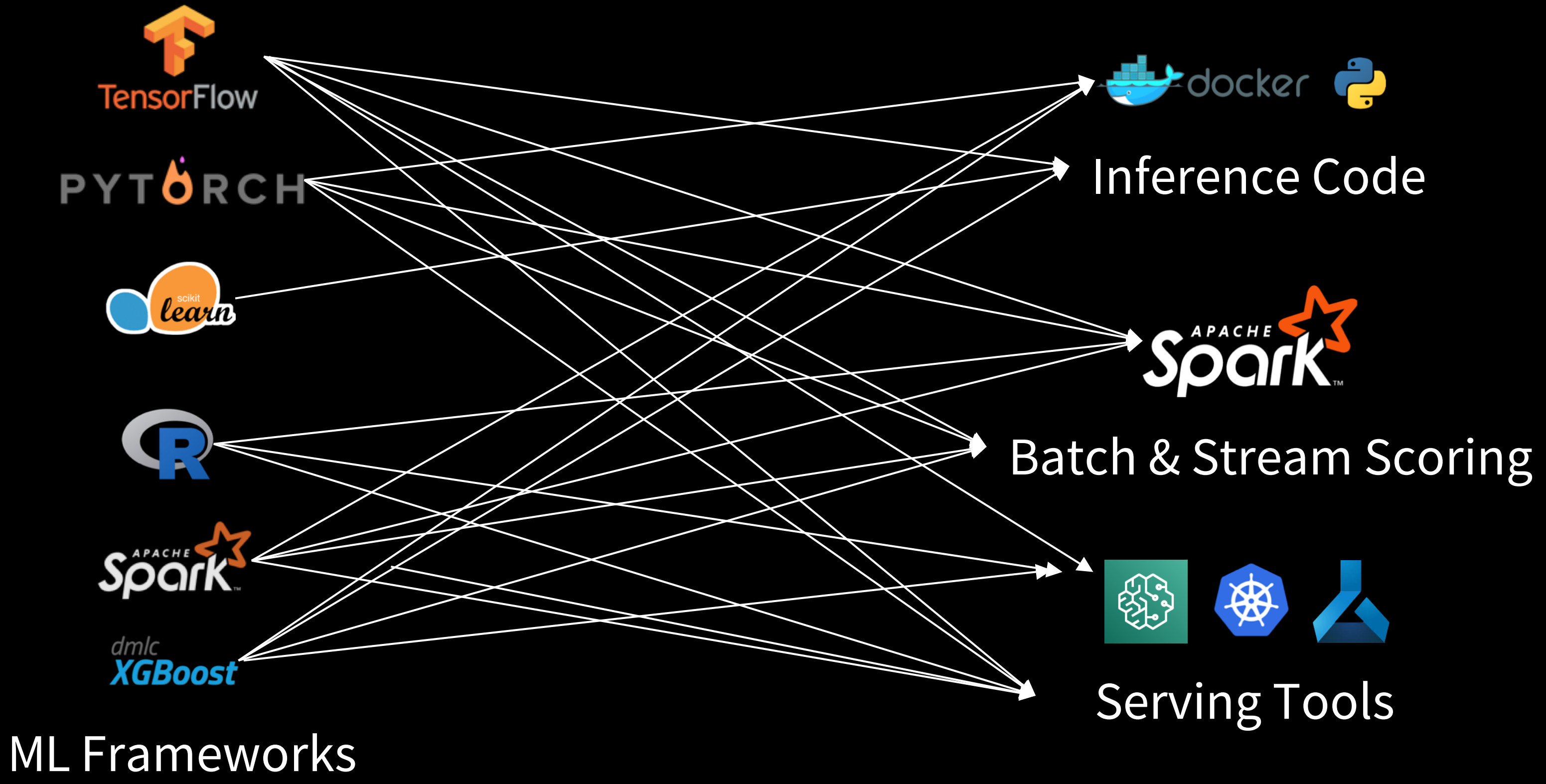
mlflow
Projects

Packaging format for reproducible runs on any platform

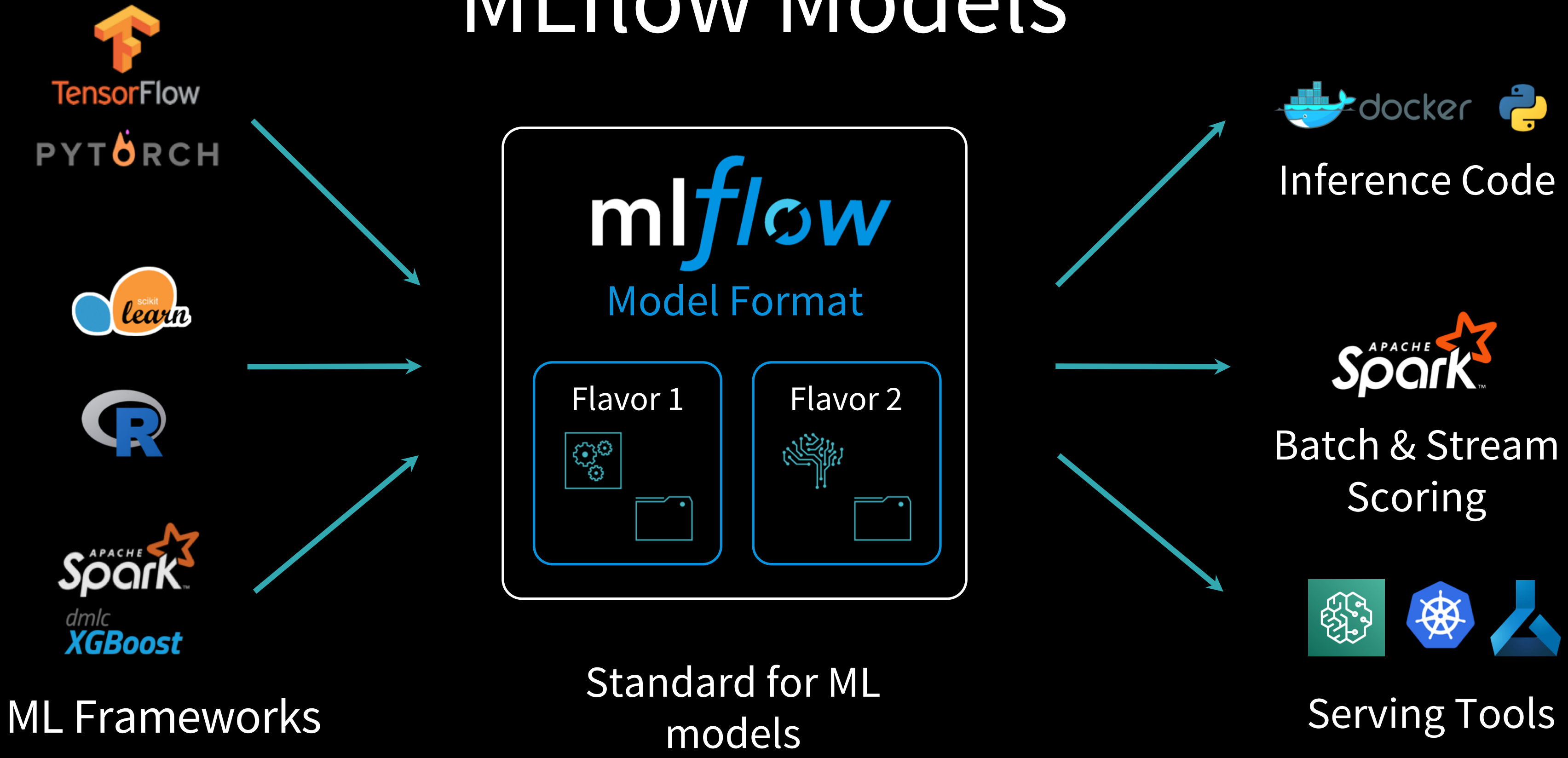
mlflow
Models

General model format that supports diverse deployment tools

Mlflow Models Motivation



MLflow Models



MLflow Models

Packaging format for ML Models

- Any directory with MLmodel file

Defines dependencies for reproducibility

- Conda environment can be specified in MLmodel configuration

Model creation utilities

- Save models from any framework in MLflow format

Deployment APIs

- CLI / Python / R / Java

Example MLflow Model

```
my_model/
├── MLmodel
│
│
│
│
│
└ estimator/
    ├─ saved_model.pb
    └─ variables/
       ...
```

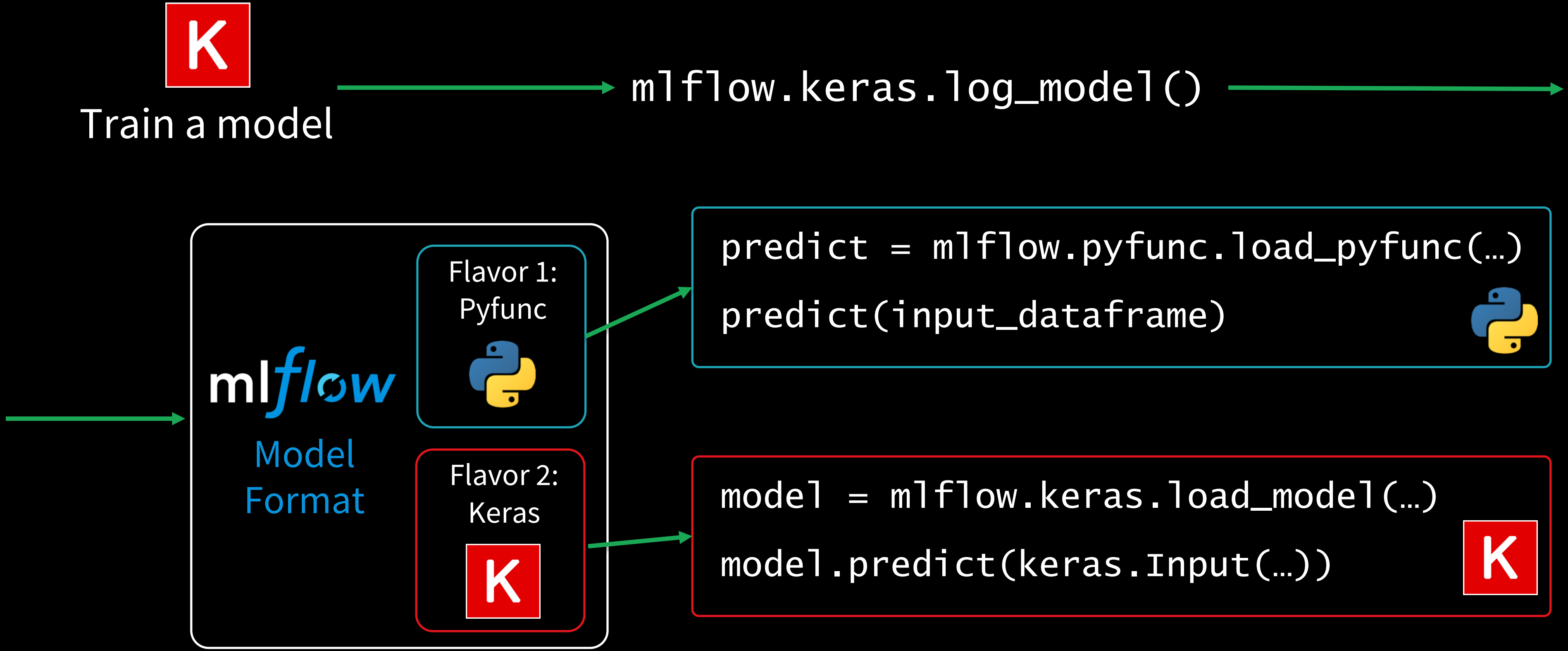
```
run_id: 769915006efd4c4bbd662461
time_created: 2018-06-28T12:34
flavors:
  tensorflow:
    saved_model_dir: estimator
    signature_def_key: predict
  python_function:
    loader_module:
       mlflow.tensorflow
```

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

Usable with Tensorflow tools / APIs

Usable with any Python tool

Model Flavors Example

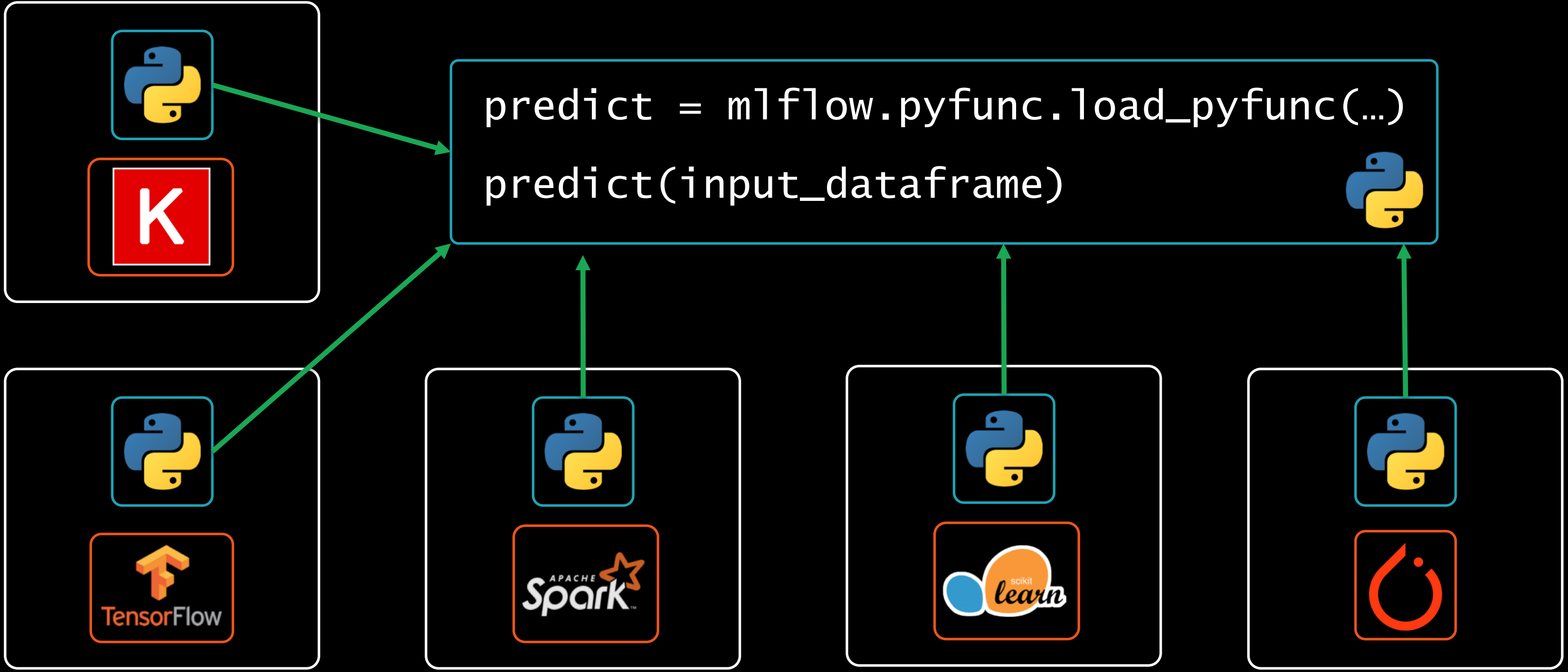


```
mlflow.keras.log_model()
```

```
predict = mlflow.pyfunc.load_pyfunc(…)
predict(input_dataframe)
```

```
model = mlflow.keras.load_model(…)
model.predict(keras.Input(…))
```

Model Flavors Example



```
predict = mlflow.pyfunc.load_pyfunc(…)

predict(input_dataframe)
```

Demo

Goal: Classify hand-drawn digits

1. Instrument Keras training code with MLflow tracking APIs

2. Run training code as an MLflow Project

3. Deploy an MLflow Model for real-time serving

1.0 Release

MLflow 1.0 was released recently! Major features:

- New metrics UI
- “Step” axis for metrics
- Improved search capabilities
- Package MLflow Models as Docker containers
- Support for ONNX models

Ongoing MLflow Roadmap

- New component: Model Registry for model management
- Multi-step project workflows
- Fluent Tracking API for Java and Scala
- Packaging projects with build steps
- Better environment isolation when loading models
- Improved model input/output schemas

Get started with MLflow

`pip install mlflow` to get started

Find docs & examples at **mlflow.org**

tinyurl.com/mlflow-slack

Thank you!