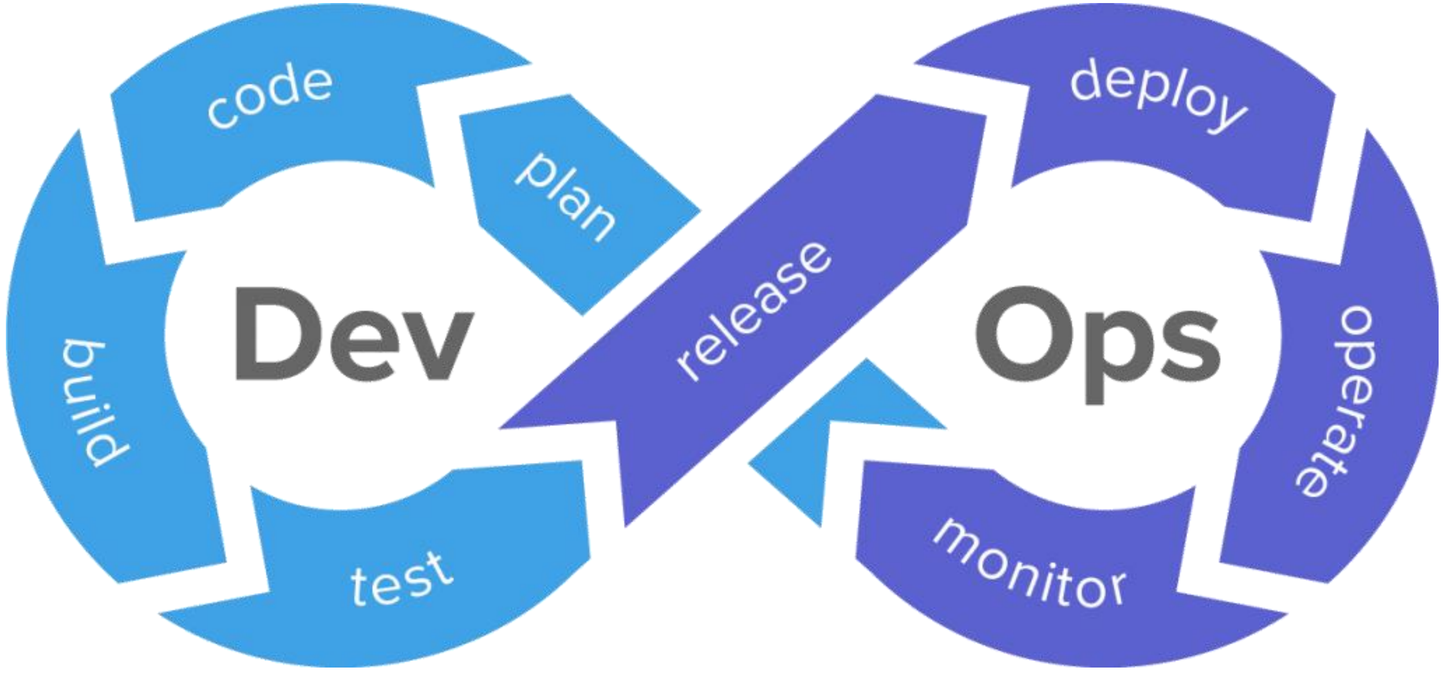
MLOps

Or how not to die to put models in production

What is MLOps?

DevOps

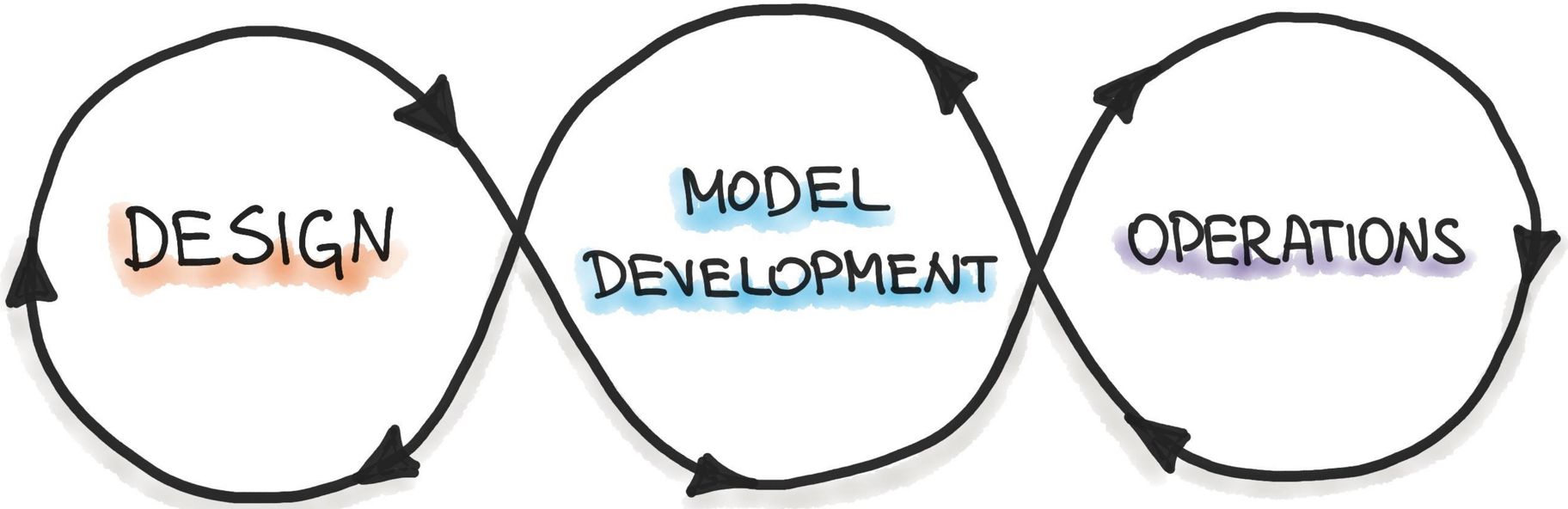
“DevOps is the union of people, process and products to enable continuos delivery of value to your end user”



ML + DEV + OPS

- Experiment
  - Data Acquisition
  - Model training & exploration
  - Evaluation
- Develop
  - Code, test & build
- Deploy
  - Release, inference & monitor

MLOps



- Requirements Engineering
- ML Use-Cases Priorization
- Data Availability Check

- Data Engineering
- ML Model Engineering
- Model Testing & Validation

- ML Model Deployment
- CI/CD Pipelines
- Monitoring & Triggering

Who am I?

@ematde

eduardo.matallanas@plainconcepts.com

Eduardo Matallanas

Principal Software Engineer@ Plain Concepts

Knowmad interested in:

- Changing live through AI
- Passionate in robotics
- Data lover
- Films & series
- Bike enthusiast & martial artist practioner

Things you will learn today. Hopefully!!

What is Mlflow

How to use it

Tracking and logging experiments

How to share your research

ATTENTION: Code examples!

Things you will not learn today.

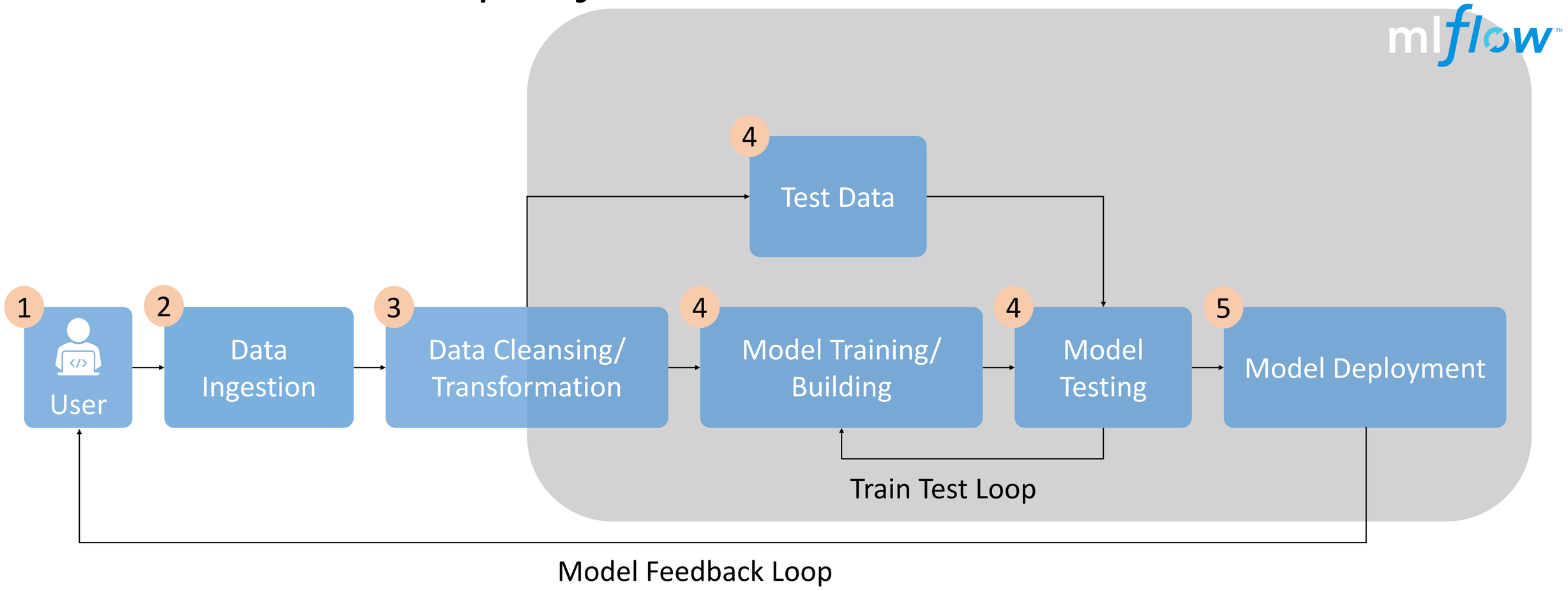
What is AI/ML/DL

How to code in Python

Maths and statistics

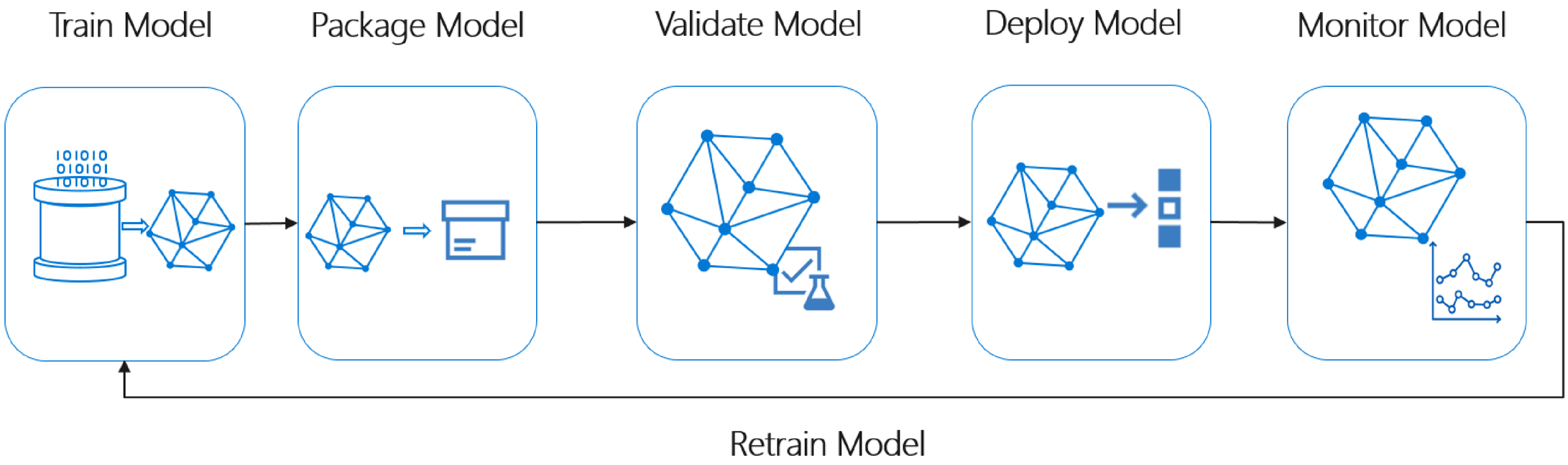
Where do I start?

How is an ML project?

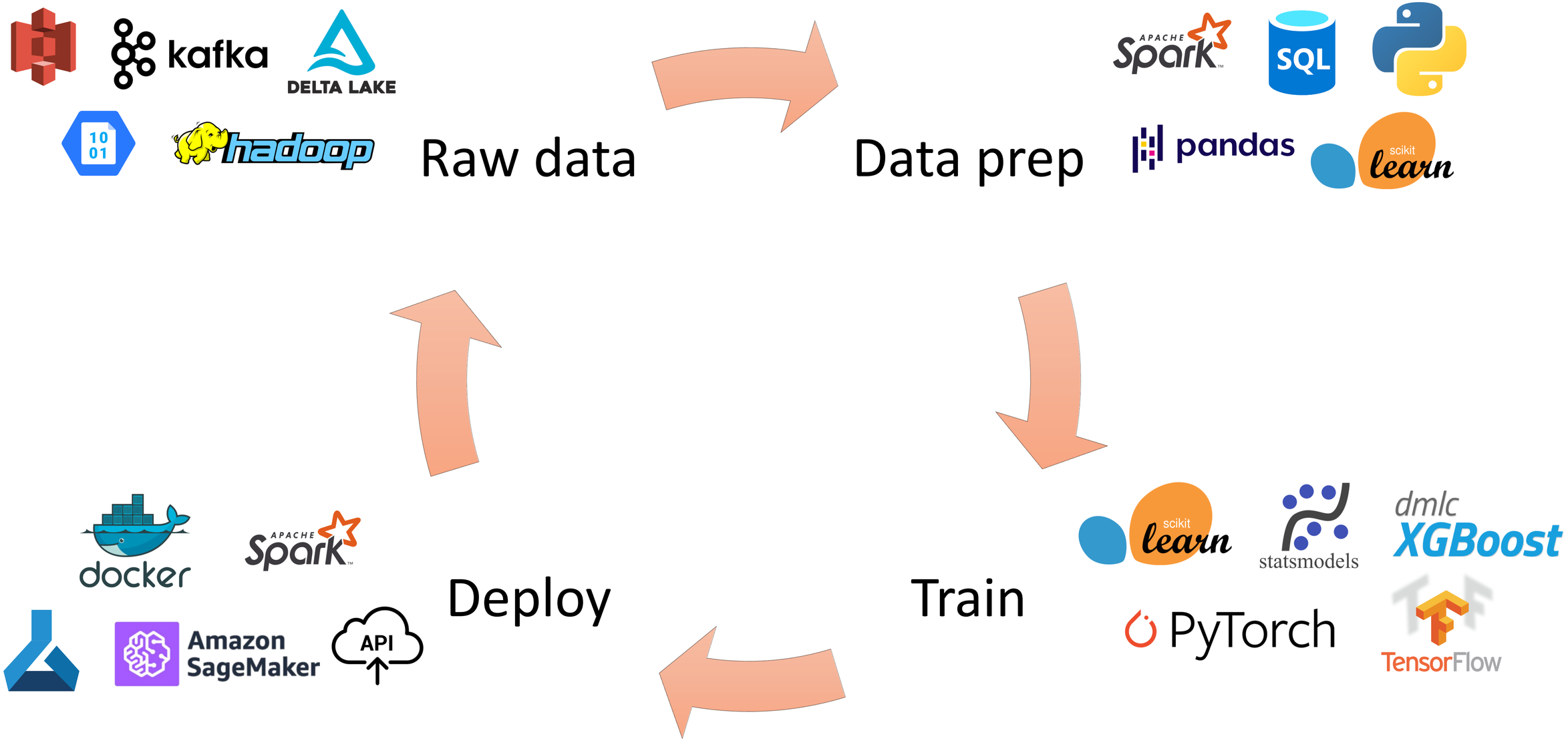


What happened when finished?

- We are ready to be exposed
- How I expose it to the world?
- Skew on the resultant exit?
- Retrain and redeploy best model?

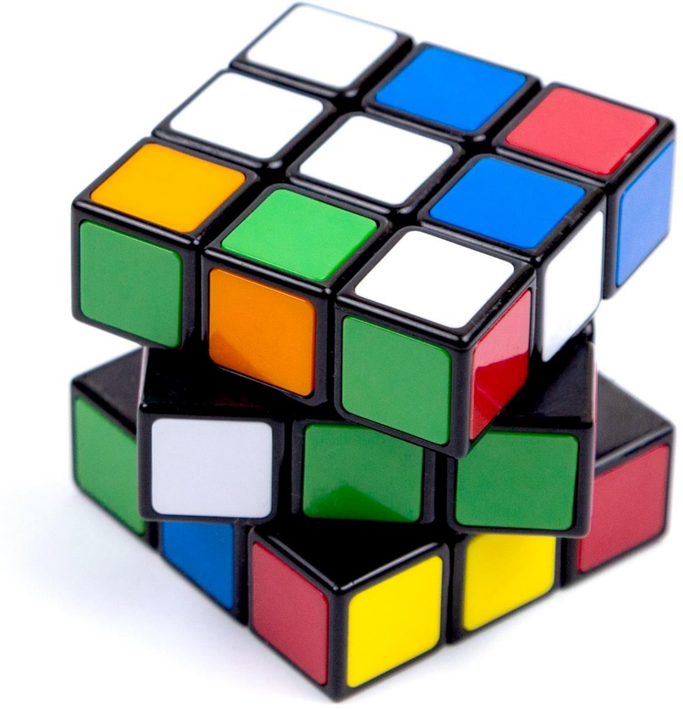


ML lifecycle and its tools

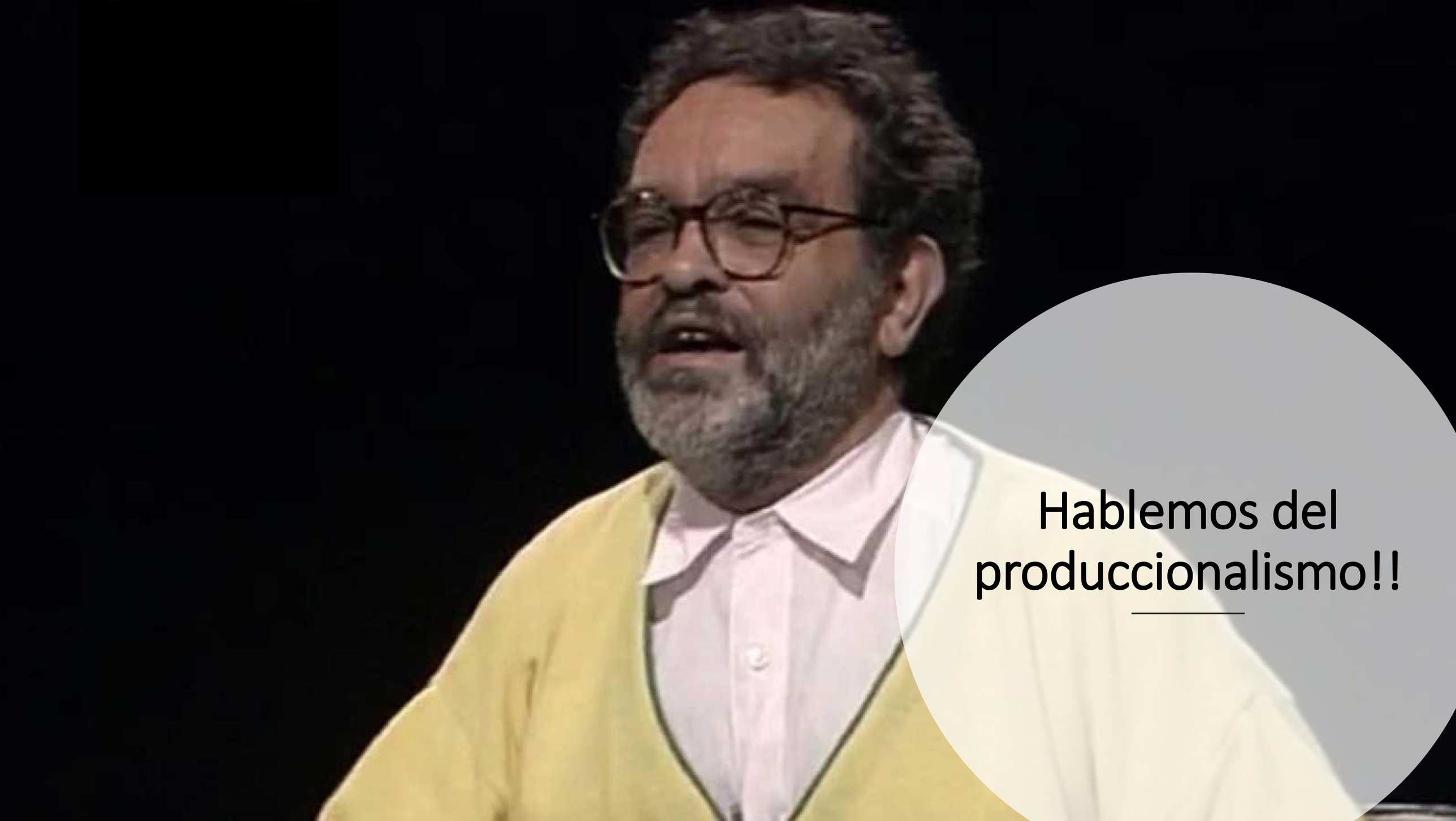


Problems to solve

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



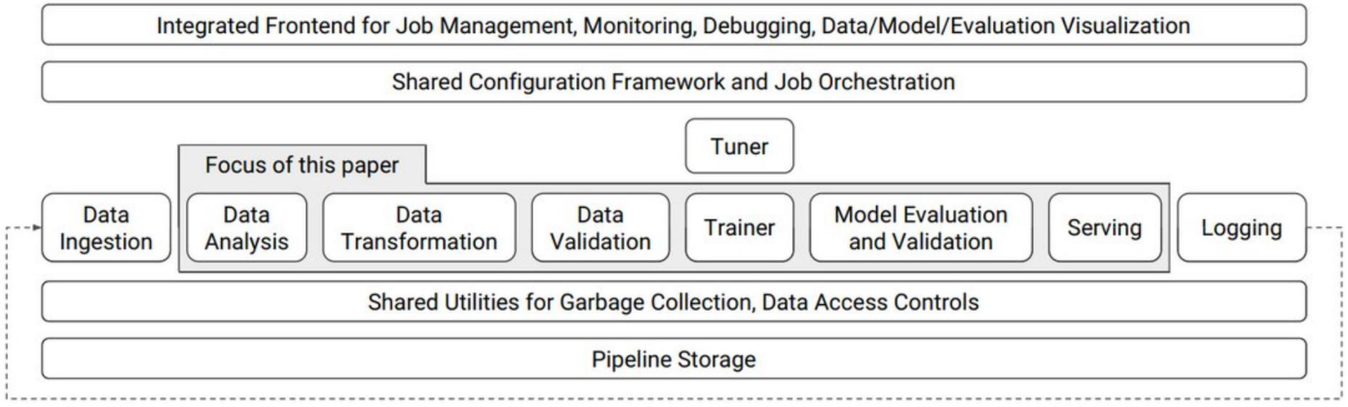
It is difficult to productionize and share



Hablemos del produccionalismo!!

Model Serving
Long time ago…

TFX: A TensorFlow-Based Production-Scale Machine Learning Platform



TFX: A TensorFlow-Based Production-Scale Machine Learning Platform

Denis Baylor, Eric Breck, Heng-Tze Cheng, Noah Fiedel, Chuan Yu Foo, Zakaria Haque, Salem Haykal, Mustafa Ispir, Vihan Jain, Levent Koc, Chiu Yuen Koo, Lukasz Lew, Clemens Mewald, Akshay Naresh Modi, Neoklis Polyzotis, Sukriti Ramesh, Sudip Roy, Steven Euijong Whang, Martin Wicke, Jarek Wilkiewicz, Xin Zhang, Martin Zinkevich
Google Inc.*

ABSTRACT

Creating and maintaining a platform for reliably producing and deploying machine learning models requires careful orchestration of many components—a learner for generating models based on training data, modules for analyzing and validating both data as well as models, and finally infrastructure for serving models in production. This becomes particularly challenging when data changes over time and fresh models need to be produced continuously. Unfortunately, such orchestration is often done ad hoc using glue code and custom scripts developed by individual teams for specific use cases, leading to duplicated effort and fragile systems with high technical debt.

We present TensorFlow Extended (TFX), a TensorFlow-based general-purpose machine learning platform implemented at Google. By integrating the aforementioned components into one platform, we were able to standardize the components, simplify the platform configuration, and reduce the time to production from the order of months to weeks, while providing platform stability that minimizes disruptions.

We present the case study of one deployment of TFX in the Google Play app store, where the machine learning models are refreshed continuously as new data arrive. Deploying TFX led to reduced custom code, faster experiment cycles, and a 2% increase in app installs resulting from improved data and model analysis.

KEYWORDS

large-scale machine learning; end-to-end platform; continuous training

1 INTRODUCTION

It is hard to overemphasize the importance of machine learning in modern computing. More and more organizations adopt machine learning as a tool to gain knowledge from data across a broad spectrum of use cases and products, ranging from recommender systems [6, 7], to clickthrough rate prediction for advertising [13, 15], and even the protection of endangered species [5].

The conceptual workflow of applying machine learning to a specific use case is simple: at the training phase, a *learner* takes a dataset as input and emits a learned model; at the inference phase, the *model* takes features as input and emits predictions. However, the actual workflow becomes more complex when machine learning needs to be deployed in production. In this case, additional components are required that, together with the learner and model, comprise a *machine learning platform*. The components provide automation to deal with a diverse range of failures that can happen in production and to ensure that model training and serving happen reliably. Building this type of automation is non-trivial, and it becomes even more challenging when we consider the following complications:

- Building *one* machine learning platform for *many* different learning tasks: Products can have substantially different needs in terms of data representation, storage infrastructure, and machine learning tasks. The machine learning platform must be generic enough to handle the most common set of learning tasks as well as be extensible to support one-off atypical use-cases.
- *Continuous training and serving*: The platform has to support the case of training a single model over fixed data, but also the case of generating and serving up-to-date models through continuous training over evolving data (e.g., a moving window over the latest n days of a log stream).
- *Human-in-the-loop*: The machine learning platform needs to expose simple user interfaces to make it easy for engineers to deploy and monitor the platform with minimal configuration. Furthermore, it also needs to help users with various levels of machine-learning expertise understand and analyze their data and models.
- *Production-level reliability and scalability*: The platform needs to be resilient to disruptions from inconsistent data, software, user configurations, and failures in the underlying execution environment. In addition, the platform must scale gracefully to the high data volume that is common in training, and also to increases in the production traffic to the serving system.

*Corresponding authors: Heng-Tze Cheng, Clemens Mewald, Neoklis Polyzotis, and Steven Euijong Whang: {hengtze,clemensm,npolyzotis,swhang}@google.com

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
KDD'17, August 13–17, 2017, Halifax, NS, Canada.
© 2017 Copyright held by the owner/author(s). 978-1-4503-4887-4/17/08.
DOI: http://dx.doi.org/10.1145/3097983.3098021

Custom ML platforms

- Facebook FBLearner, Google TFX, Uber Michelangelo
- Advantages:
  - Standarise the ML loop
- Disadvantages:
  - Limited to a few algorithms or frameworks
  - Tied to the company's infrastructure

Real Production Tools

H2O.ai

Azure Machine Learning

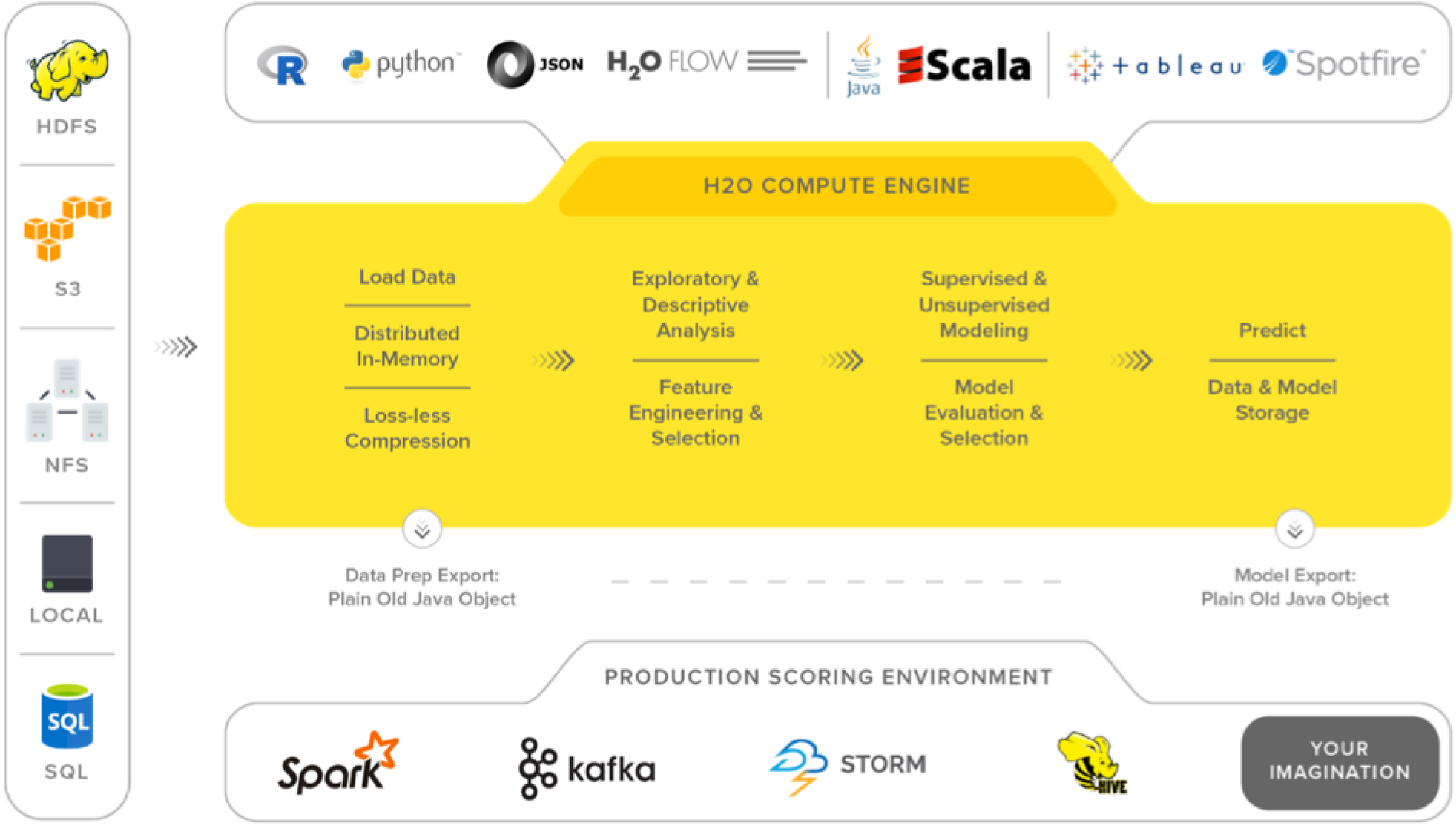
rapidminer

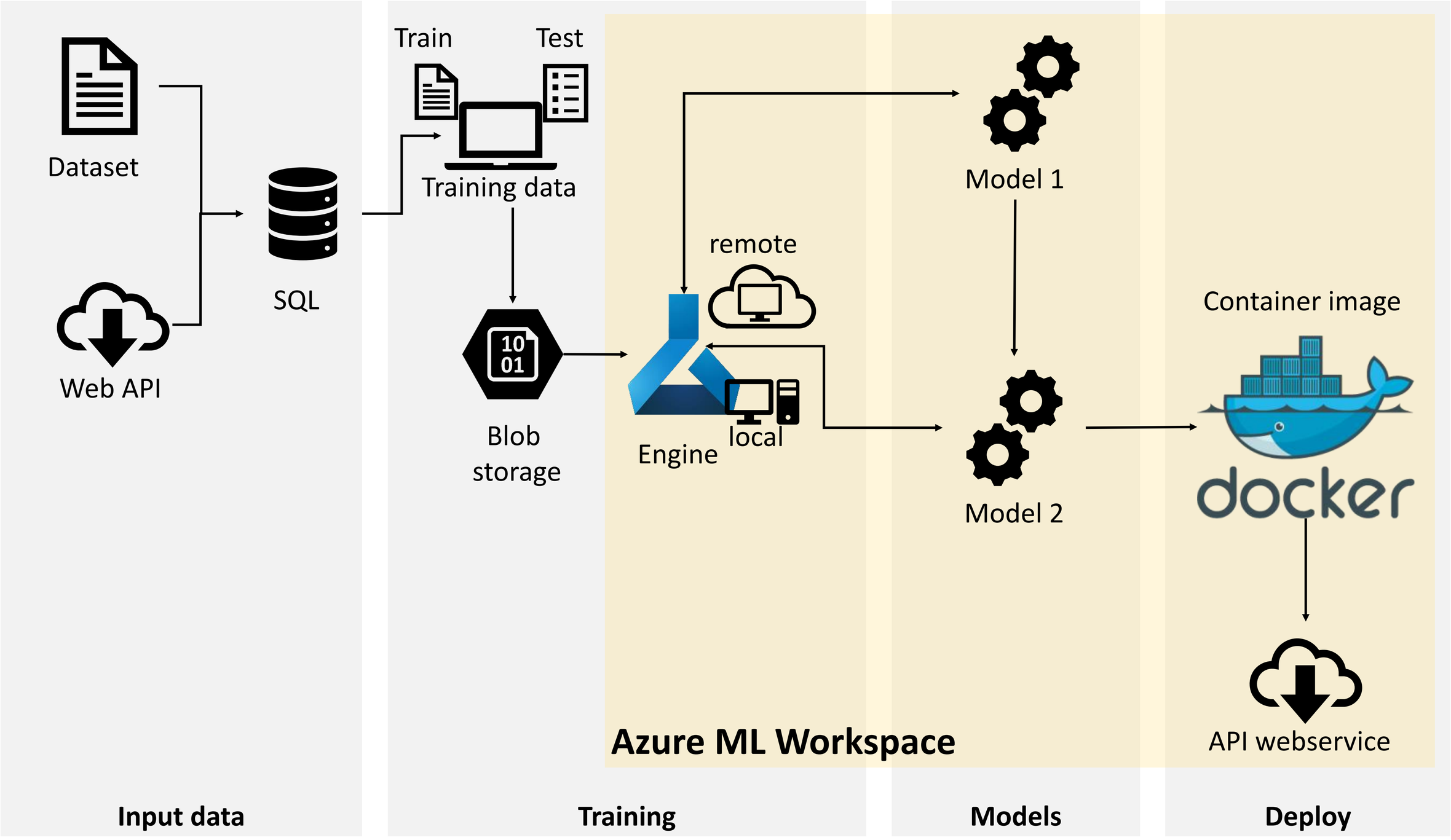
vertex.ai

databricks
&
mlflow

Amazon SageMaker

H2O


HDFS
S3
NFS
LOCAL
SQL
SQL
R
python
JSON
H2O FLOW
Java
Scala
tableau
Spotfire
H2O COMPUTE ENGINE
Load Data
Distributed In-Memory
Loss-less Compression
Exploratory & Descriptive Analysis
Feature Engineering & Selection
Supervised & Unsupervised Modeling
Model Evaluation & Selection
Predict
Data & Model Storage
Data Prep Export: Plain Old Java Object
Model Export: Plain Old Java Object
PRODUCTION SCORING ENVIRONMENT
Spark
kafka
STORM
HIVE
YOUR IMAGINATION


Dataset
Web API
SQL
Train
Test
Training data
Blob storage
remote
Engine
local
Azure ML Workspace
Model 1
Model 2
Container image
docker
API webservice
Input data
Training
Models
Deploy

classifier | All Runs Report

Add Chart Filter Get Link Refresh

PREPARING

0

RUNNING

0

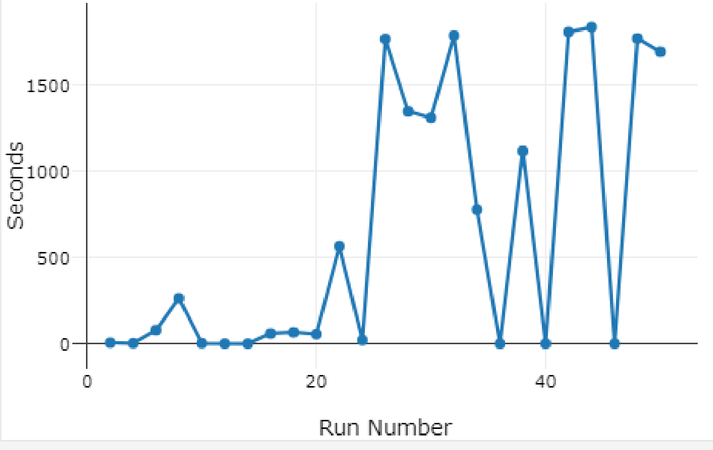
COMPLETED

7

FAILED

18

DURATION (SEC)



RUN NUMBER	STATUS	CREATED TIME (UTC)	DURATION (SECONDS)	ACCURACY	TAGS
42	Completed	10/30/2018, 12:21:37 PM	1810.439		
44	Failed	10/31/2018, 2:40:02 PM	1838.426		
46	Failed	10/31/2018, 3:40:15 PM	0.127		
48	Completed	10/31/2018, 3:42:05 PM	1772.63		
50	Completed	11/7/2018, 12:24:12 PM	1694.762		

ATTRIBUTES	
Status	Completed
Created	11/6/2018, 4:00:00 PM
Duration	0:25:43
Target	devindicationsvm
Run Id	embedding_1541516393251
Run Number	70
Script Name	train.py

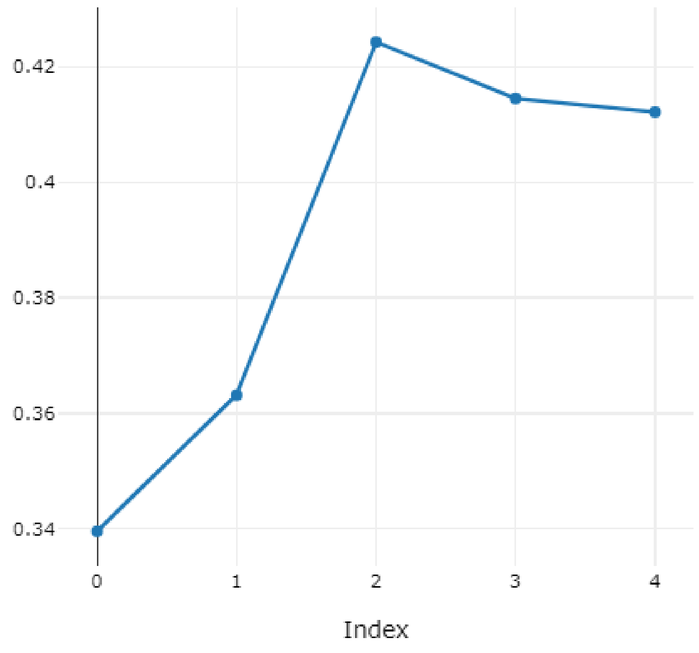
TRACKED METRICS	
Learning Rate	0.05
Epochs	5
Embedding Type	skipgram
Dimension	300
Pretrained Vectors	wiki.es.vec

Training Metrics

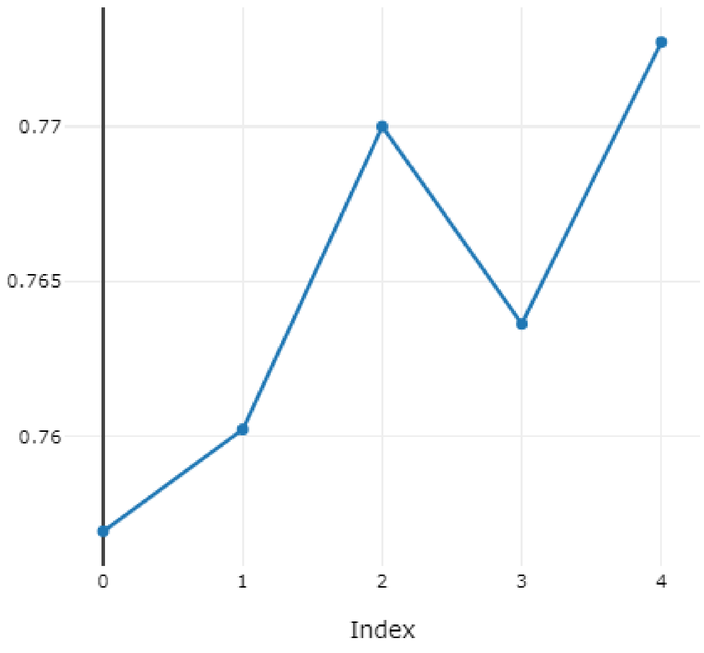
Performance

CHARTS

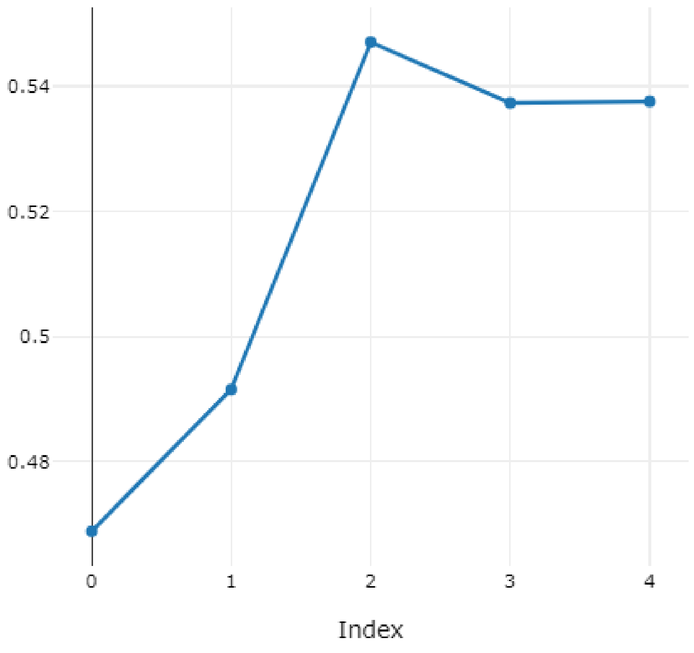
Recall



Precision



F1 score



mlflow

- Open source platform
- Run the same way anywhere
- Works with any ML library & language
- Designed to be useful both for 1 person, small teams or big teams
- Can publish to different flavours
  - AzML, Sagemaker, local python, H_2O, etc.

MLflow

127.0.0.1:5000/#/

mlflow

GitHub Docs

Experiments

Default

Default

Experiment ID: 0 Artifact Location: /Users/jules/jsd-mlflow-examples/keras/binary_classifier_nn/mlruns/0

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

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

9 matching runs Compare Selected Download CSV

					Parameters				Metrics					
Date	User	Source	Version	epochs	hidden_layers	loss_function	output	average_acc	average_loss	binary_acc	binary_loss	validation_acc	validation_loss
2018-08-13 15:13:54	jules	main_nn.py	abcd1f	20	3	mse	32	0.878	0.09	0.977	0.025	0.885	0.025
2018-08-13 14:34:43	jules	main_nn.py	abcd1f	30	3	binary_crossentropy	32	0.866	0.441	0.992	0.035	0.879	0.035
2018-08-13 09:12:03	jules	main_nn.py	abcd1f	20	1	binary_crossentropy	16	0.883	0.304	0.937	0.212	0.89	0.212
2018-08-10 15:11:24	jules	main_nn.py	abcd1f	20	1	binary_crossentropy	16	0.883	0.304	0.937	0.212	0.89	0.212
2018-08-10 15:09:02	jules	main_nn.py	abcd1f	20	1	binary_crossentropy	16	0.883	0.304	0.937	0.212	0.89	0.212
2018-08-10 14:41:29	jules	main_nn.py	abcd1f	20	1	binary_crossentropy	16	0.883	0.304	0.937	0.212	0.89	0.212
2018-08-10 14:39:25	jules	main_nn.py	abcd1f	20	1	binary_crossentropy	16	0.883	0.304	0.937	0.212	0.89	0.212
2018-08-09 14:54:40	jules	main_nn.py	e3c9ae	15	3	mse	32	0.881	0.089	0.96	0.04	0.887	0.04
2018-08-09 14:53:49	jules	main_nn.py	e3c9ae	20	1	binary_crossentropy	16	0.883	0.304	0.937	0.212	0.89	0.212

MLFlow 1.0 community

600k
monthly downloads

100+
code contributors

40
contributing organizations

Supported Integrations

Programming Languages

python

R

Java

ML libraries

Apache Spark

TensorFlow

Keras

scikit learn

PyTorch

XGBoost

LightGBM

spaCy

mleap

H2O.ai

CatBoost

⋮

Storage backend

SQL

hadoop

SFTP

NFS

Deployment

Apache Spark

docker

Amazon SageMaker

Azure Machine Learning

kubernetes

mleap

ONNX

mlflow Components

mlflow Tracking

Record and query experiments: code, data, config, and results

mlflow Projects

Package data science code in a format to reproduce runs on any platform

mlflow Models

Deploy machine learning models in diverse serving environments

mlflow Model Registry

Store, annotate, discover, and manage models in a central repository

mlflow Plugins

Customize the behavior of the client to integrate any third-party tool

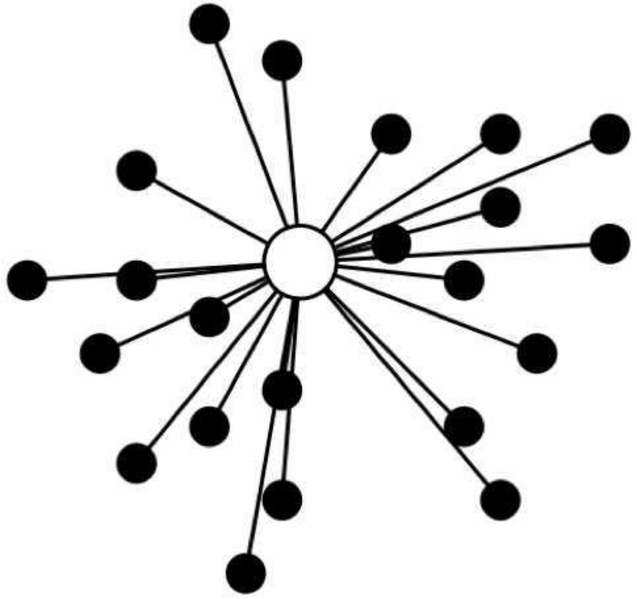
mlflow Tracking

mlflow

Tracking

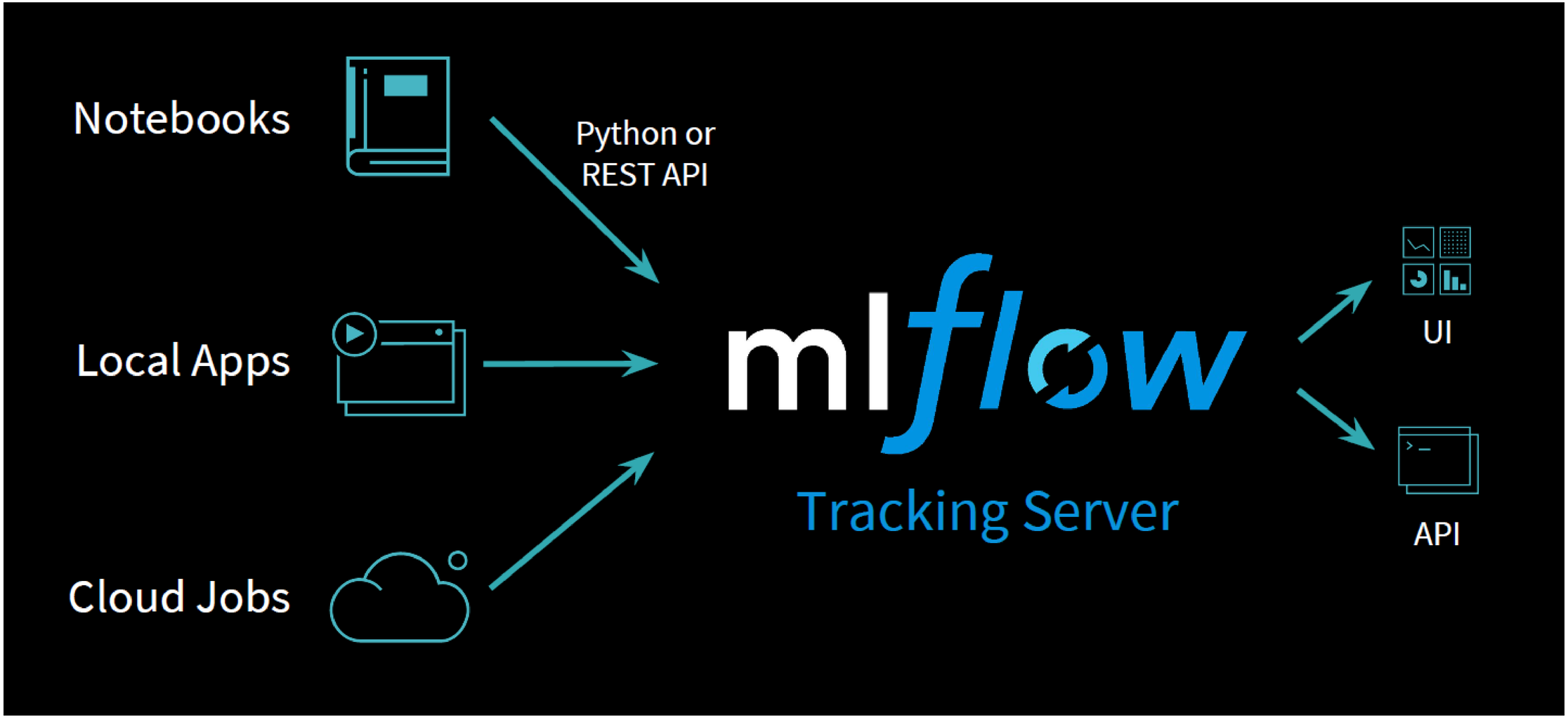
- Record and query experiments:
  - Data
  - Code
  - Model parameters
  - Results (performance metrics)
  - Model
- Autolog for most common ML libraries

Motivation



Centralized repository of useful information to analyze several runs of training.

How it is working



Key concepts

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

Code example

```
import mlflow

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

    # train model
    model = train_model(layers, alpha)

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

Show me
the code

mlflow Projects

mlflow

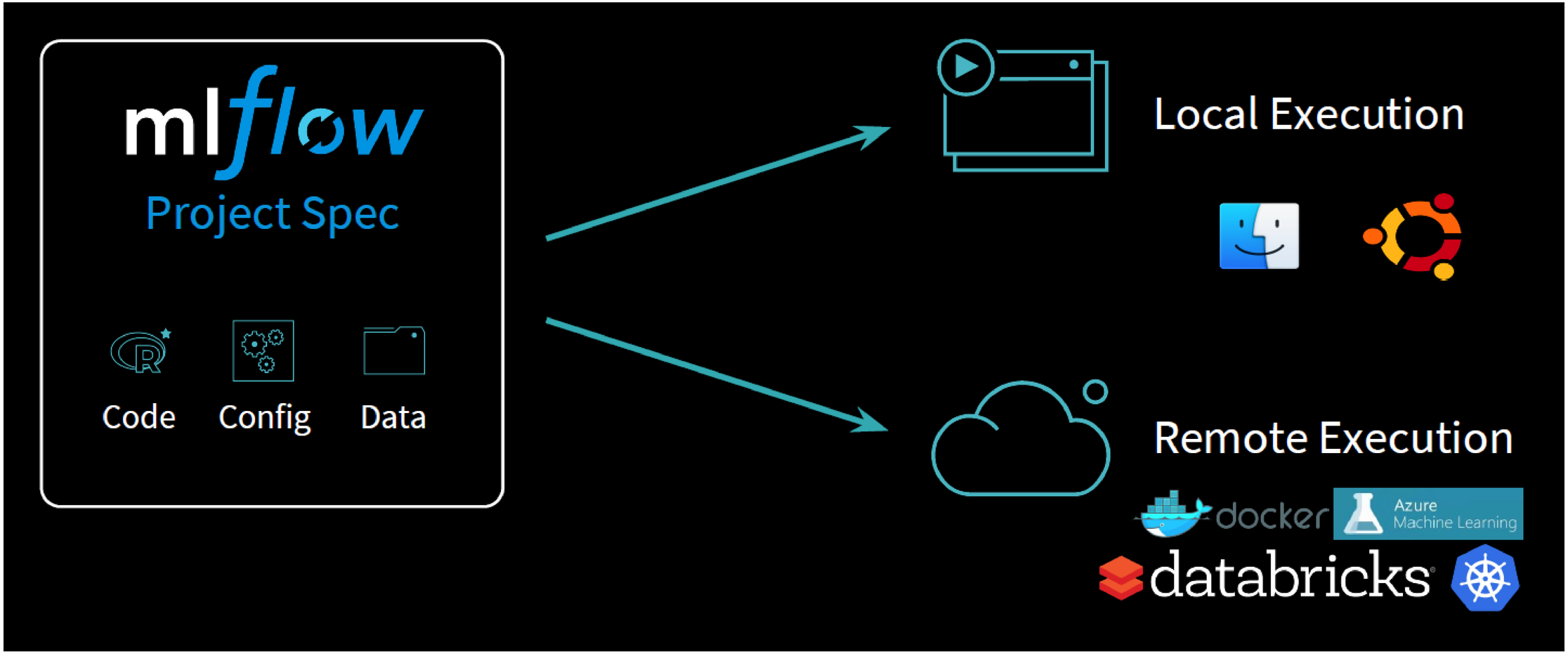
Projects

- 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

```
my_project/
  |- MLproject
  |
  |
  |
  |
  |
  |- conda.yaml
  |- main.py
  |- model.py
  ...
```

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

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

Show me
the code

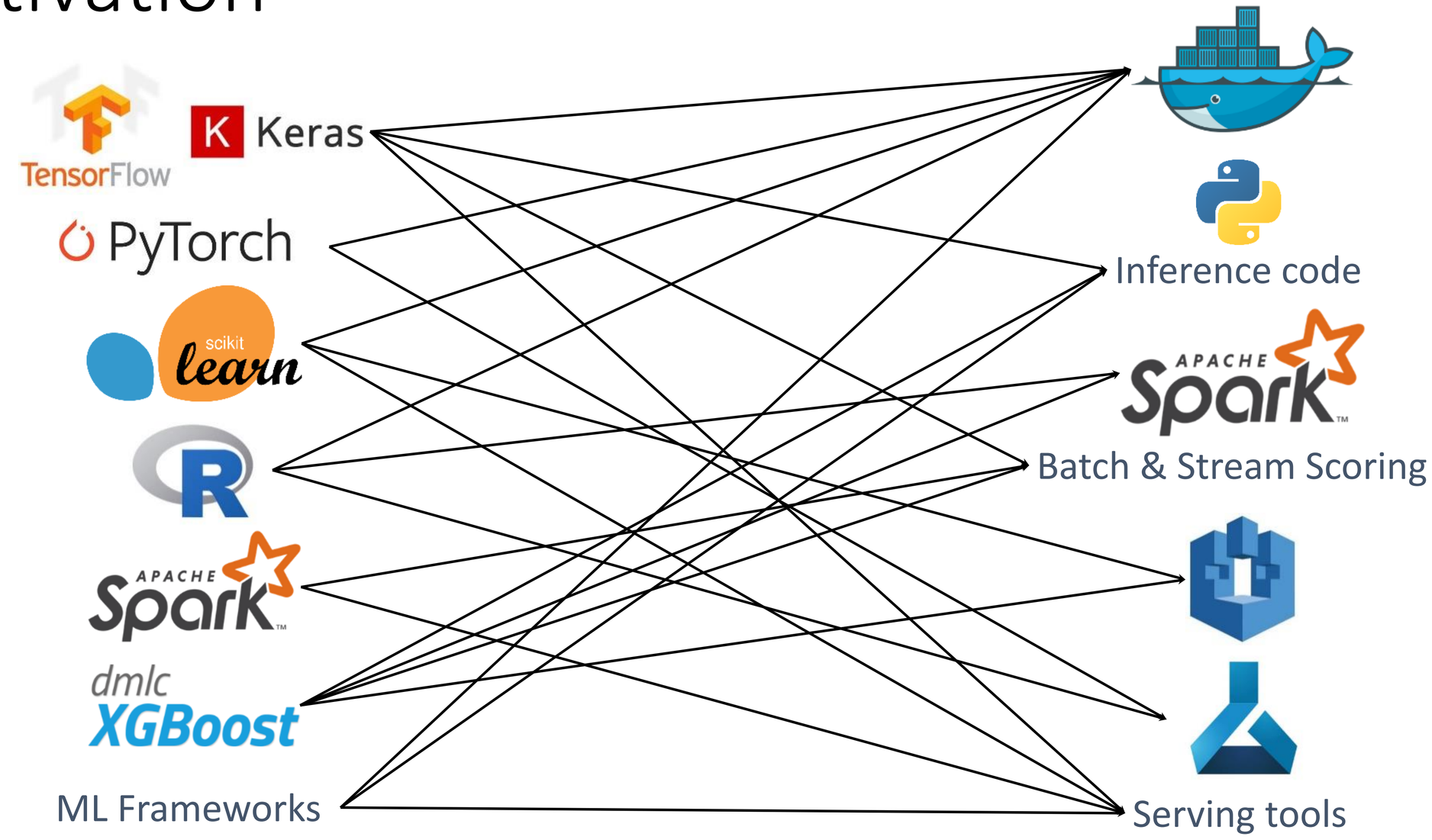
mlflow Models

mlflow

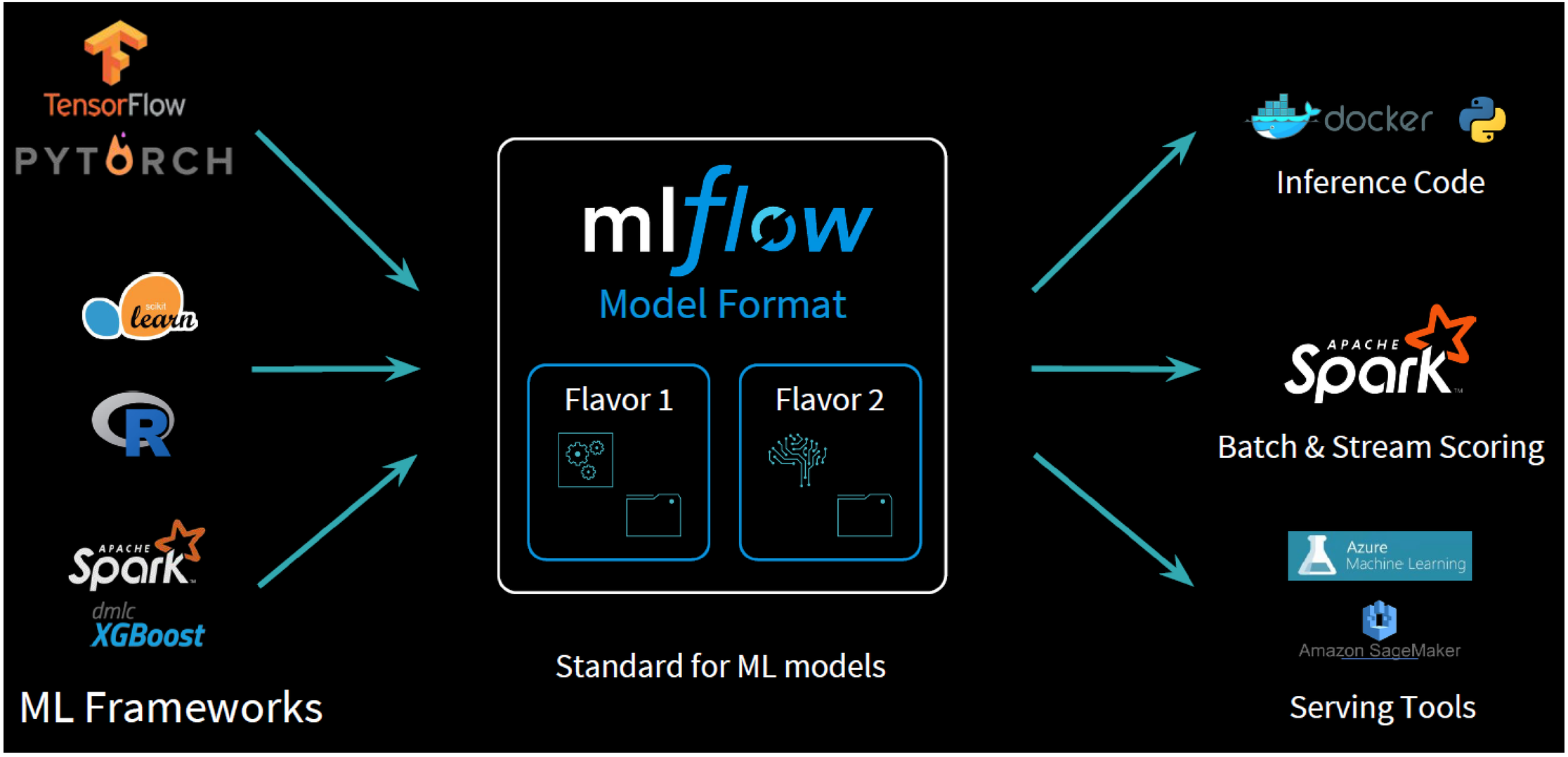
Models

- Packaging format for ML Models
- Defines dependencies for reproducibility
- Deployment APIs
- Custom flavors for each of them

Motivation


TensorFlow
Keras
PyTorch
scikit learn
R
APACHE Spark
dmlc XGBoost
ML Frameworks
Inference code
APACHE Spark
Batch & Stream Scoring
Serving tools

How it is working



Key concepts

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

Code example

```
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
```

Usable by tools that understand TensorFlow model format

Usable by any tool that can run Python (Docker, Spark, etc!)

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

Show me
the code

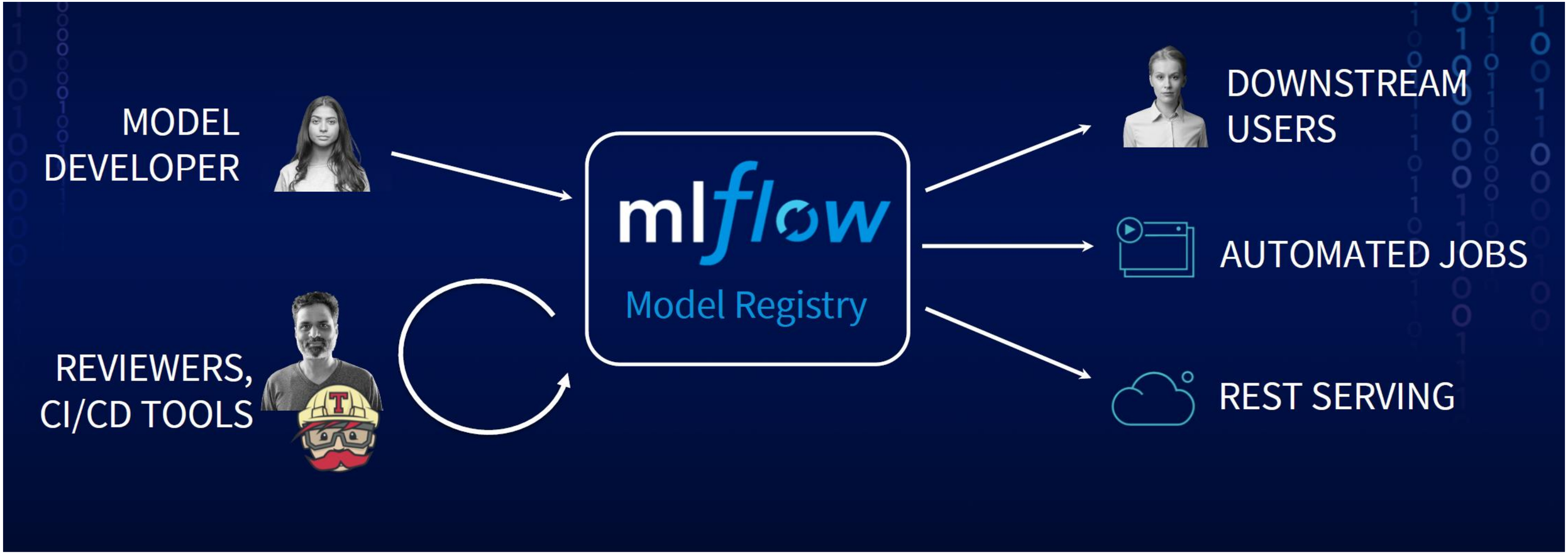
mlflow Model Registry

mlflow

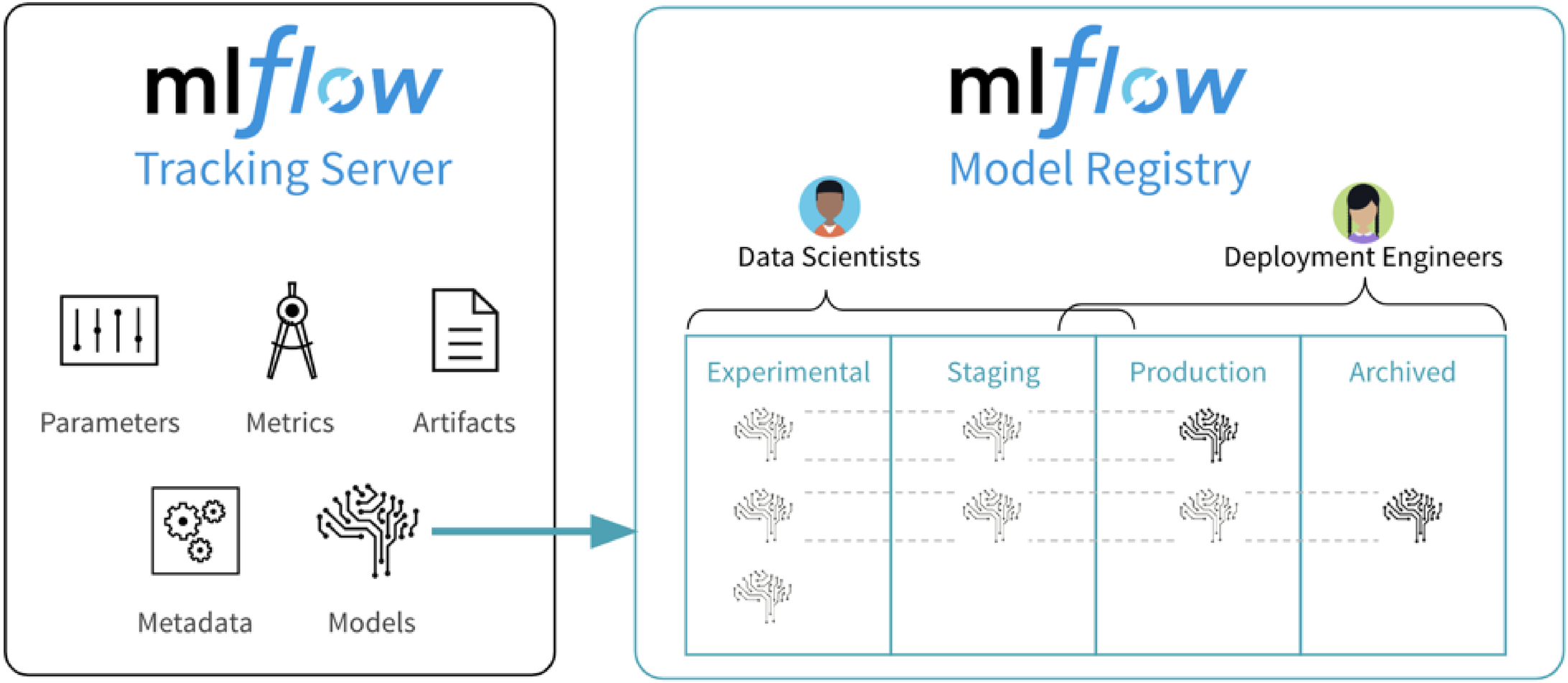
Model Registry

- Centralized model store to manage the full model lifecycle
  - Model lineage
  - Model versioning
  - Stage transitions
  - Annotations
- Manage models collaboratively

Motivation


MODEL DEVELOPER
REVIEWERS, CI/CD TOOLS
mlflow
Model Registry
DOWNSTREAM USERS
AUTOMATED JOBS
REST SERVING

How it is working



Key concepts

- Registered models
- Specific model versions
- Different stages per model versions
- Annotations and descriptions

Code example

One way to register a model

```
result = mlflow.register_model(
    "runs:/d16076a3ec534311817565e6527539c0/artifacts/sklearn-model",
    "sk-learn-random-forest-reg"
)
```

Another way to register the model

```
from mlflow.tracking import MlflowClient

client = MlflowClient()
client.create_registered_model("sk-learn-random-forest-reg-model")
result = client.create_model_version(
    name="sk-learn-random-forest-reg-model",
    source="mlruns/0/d16076a3ec534311817565e6527539c0/artifacts/sklearn-model",
    run_id="d16076a3ec534311817565e6527539c0"
)
```

Show me
the code

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

THANK YOU