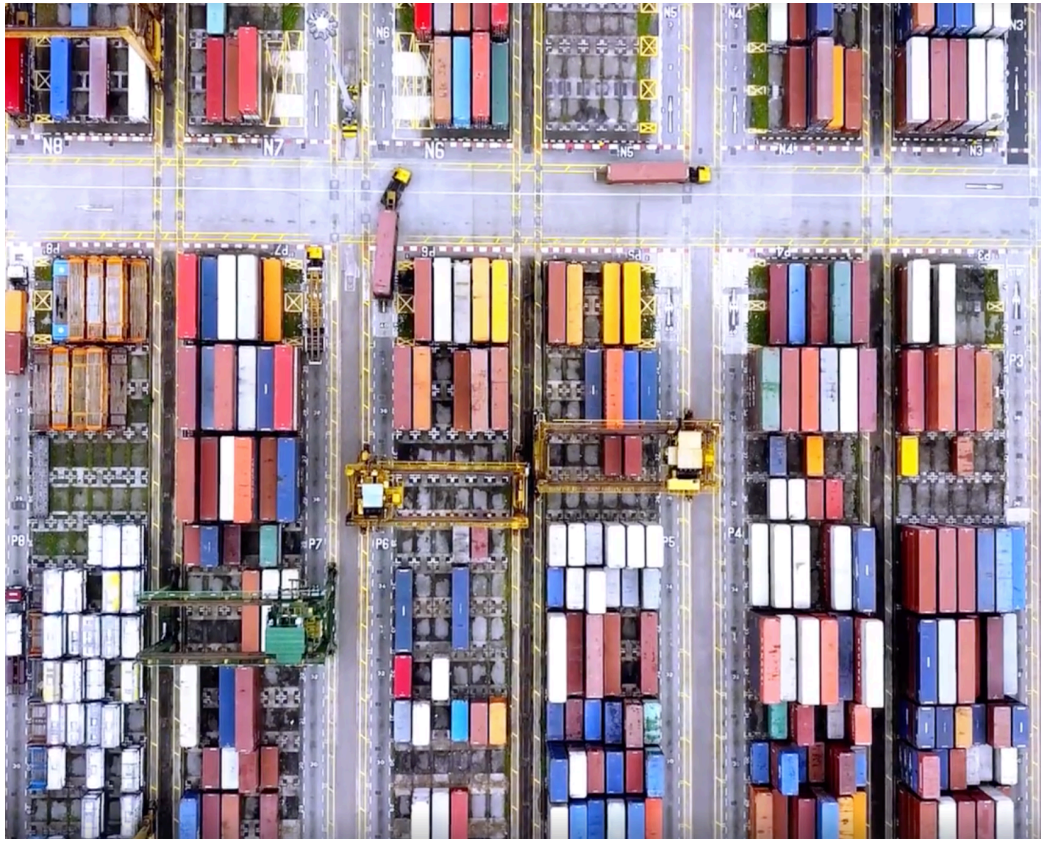
Mini Workshop w/ MLflow

https://qconsf.com/sf2019/presentation/ml-mini-workshop

Hien Luu

Agenda

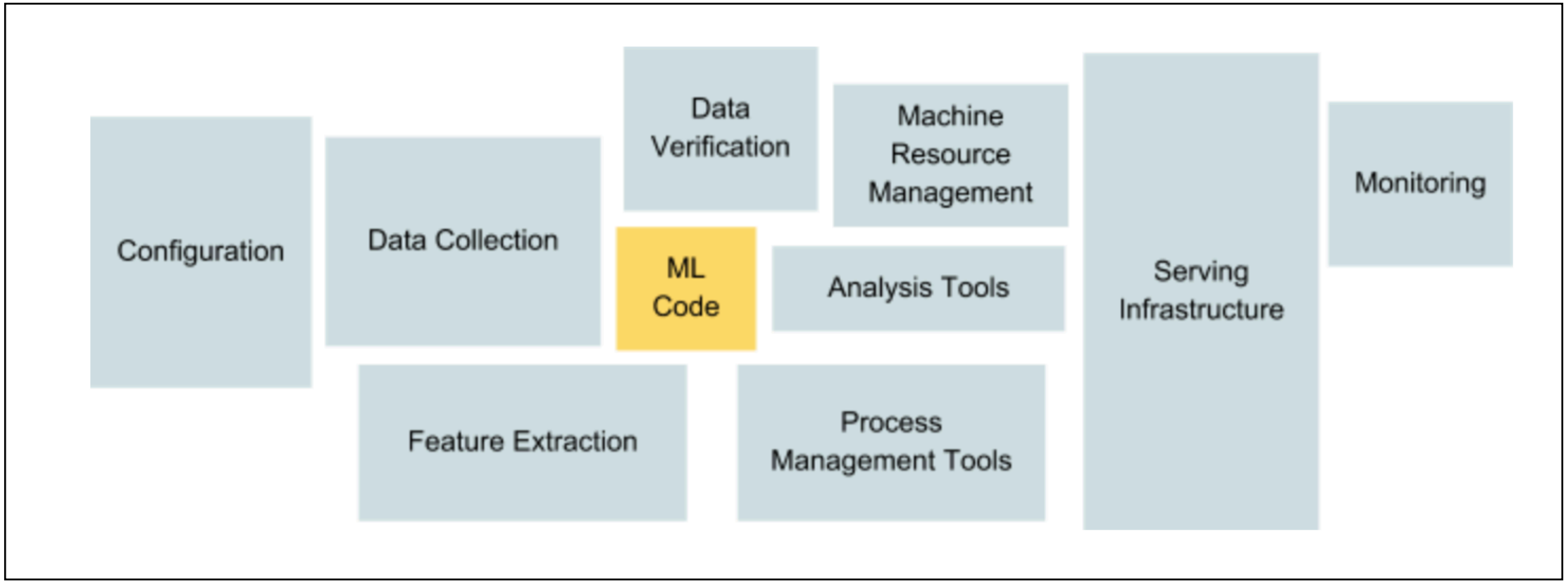
- Preparation Steps
- ML Development Process
- Introduction to MLflow
- Tutorial



https://qconsf.com/sf2019/presentation/ml-mini-workshop

ML Development Process Overview

Hidden Technical Debt in Machine Learning System (paper from Google)



The required surrounding infrastructure is vast & complex

ML Development Process Overview

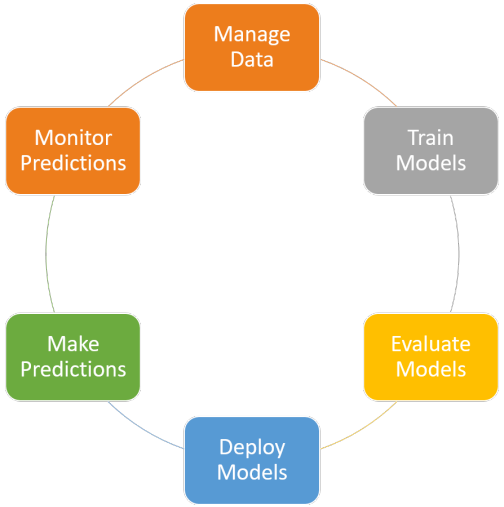
ML development is more challenging than traditional software development

ML development has unique challenges

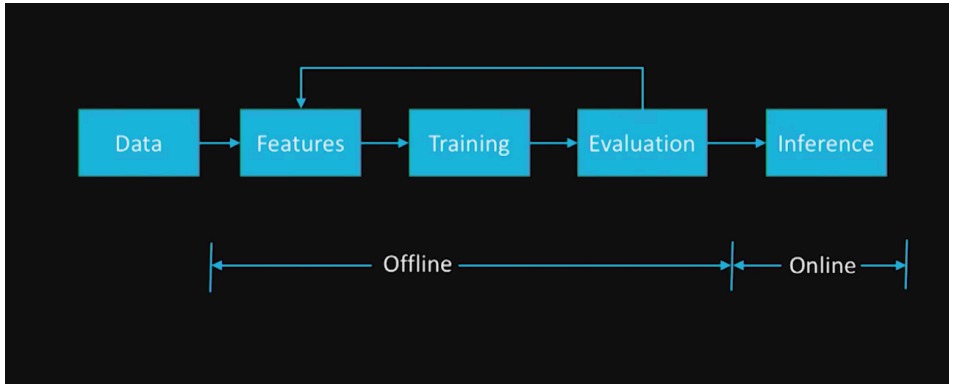
ML Development Process Overview

Machine Learning Development Dimensions

Iterative



Environment



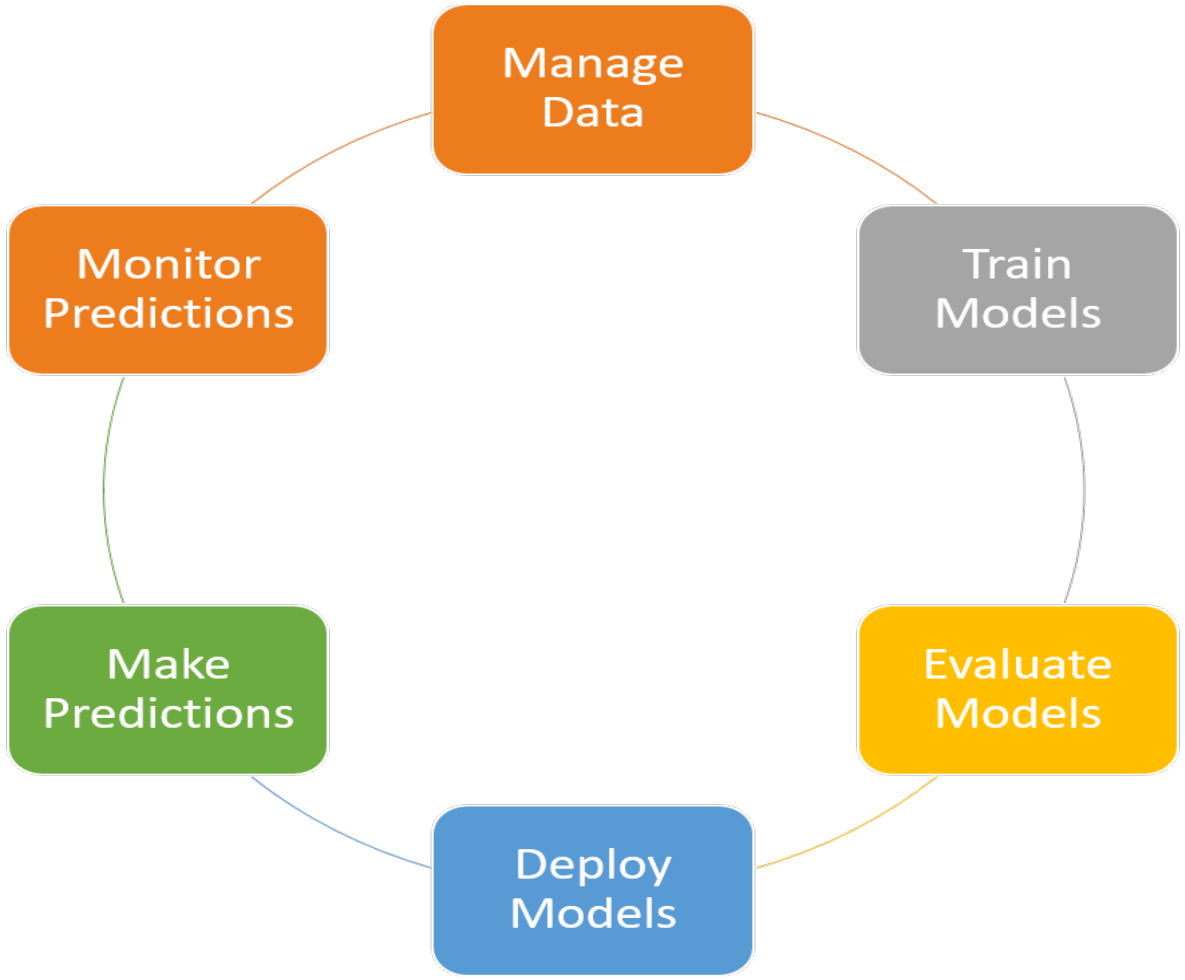
Big Data

ML Development Process Overview

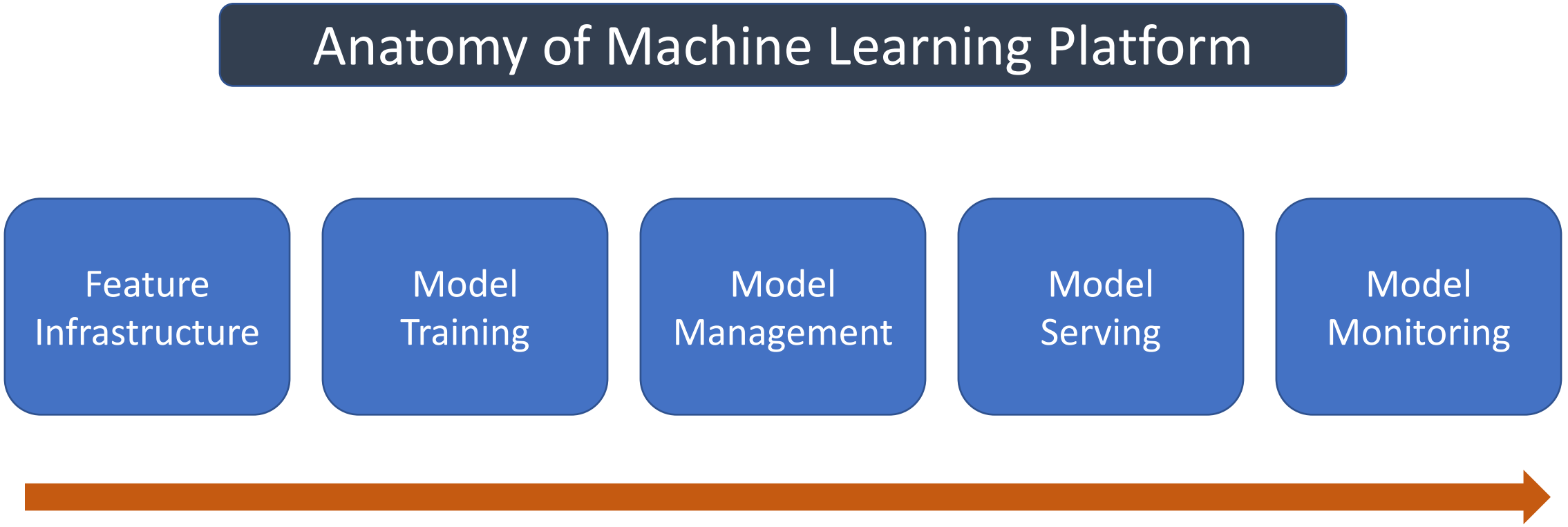
Iterative Dimension

Model Performance

- Data
- Algorithm
- Parameters

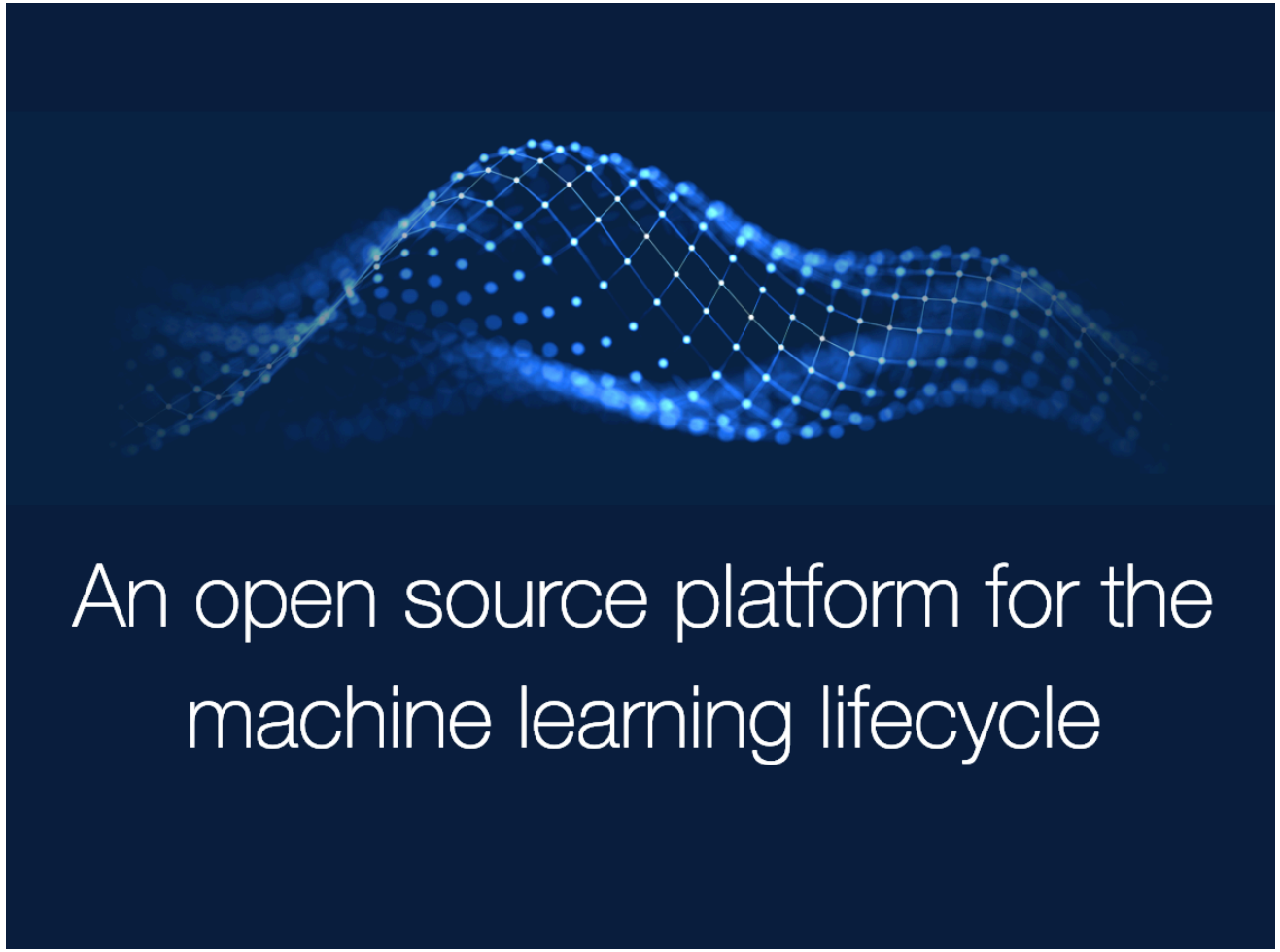


Machine Learning Platform Tour



Minimize incidental complexity in Machine Learning

MLflow – Open ML Platform



Principles
• Open • Ease of use • Extensible • Scalable

Manage the ML lifecycle, including experimentation, reproducibility and deployment

MLflow – Open ML Platform

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

Model Registry

Manage model lifecycle

MLflow – Tracking

MLflow – Tracking

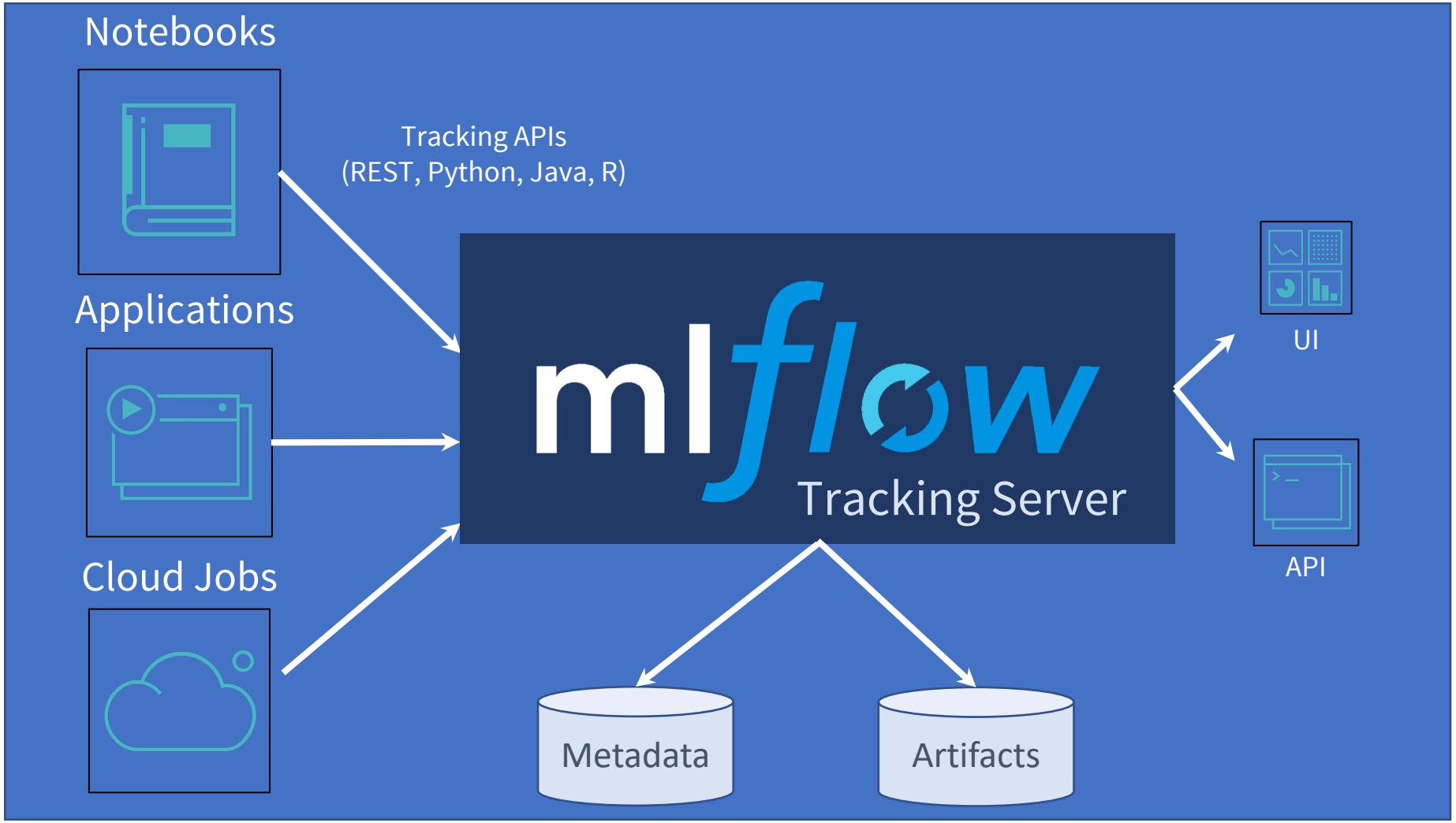
Tracking Experiments

11	Dataset	Split (trian/dev/test)	0.7/0.2/0.1	0.7/0.2/0.1	0.7/0.2/0.1	0.7/0.15/0.15	0.7/0.15/0.15
12		Class ratio (train/dev/test)	0.42/0.42/0.42	0.42/0.42/0.42	0.42/0.42/0.42	/0.3/0.3	/0.3/0.3
13		train/dev/test size	4871/1392/696	4871/1392/696	5315/1518/760	5315/1139/1139	5315/1139/1139
14	Training hyperparameters	Learning rate	1.00E-05	1.00E-05	1.00E-05	1.00E-05	1.00E-05
15		epoch	3	2	1	5	6
16		batch size	32	32	32	32	32
17	Results	accuracy	0.88304595	0.8650862069	0.8687747	0.86997364	0.65
18		f1	0.82495437	0.8108753316	0.82383946	0.81827954	0.44
19		precision	0.878865	0.7848381601	0.8407407	0.8556561	0.56
20		recall	0.7780239	0.8389705882	0.8076923	0.78442625	0.36
21		tp	1398	1402	1460	1334	1130
22		tn	1692	1663	1707	1543	1504
23		fp	1113	1142	1161	1108	1148
24		fn	1189	1185	1190	1154	1357
25		loss	0.59637538	0.594134	0.594134	0.6037084	0.594134
26	Test results	accuracy	0.90747	0.90747	0.88026	0.88314	0.75847
27		f1	0.85636	0.85636	0.83108	0.83469	0.5915
28		precision	0.90934	0.90934	0.86689	0.87027	0.77626
29		recall	0.8099	0.8099	0.79846	0.80226	0.48604
30							

MLflow – Tracking

Tracking

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



MLflow – Tracking

Tracking

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

Experiment -> Runs

```
import mlflow

mlflow.set_tracking_uri("http://remote-server:5555")
mlflow.set_experiment("learing-mlflow")

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

  # do the model training
  mlflow.log_metrics("RMSE", model.rmse())
  mlflow.log_artifact("plot", model.plot(df))
  mlflow.tensorlfow.log_model(model)
```

Key Concepts

- K-V parameters
- Metrics
- Artifacts
- Source code
- Version (git)
- Tags & notes

Python, R, Java, REST

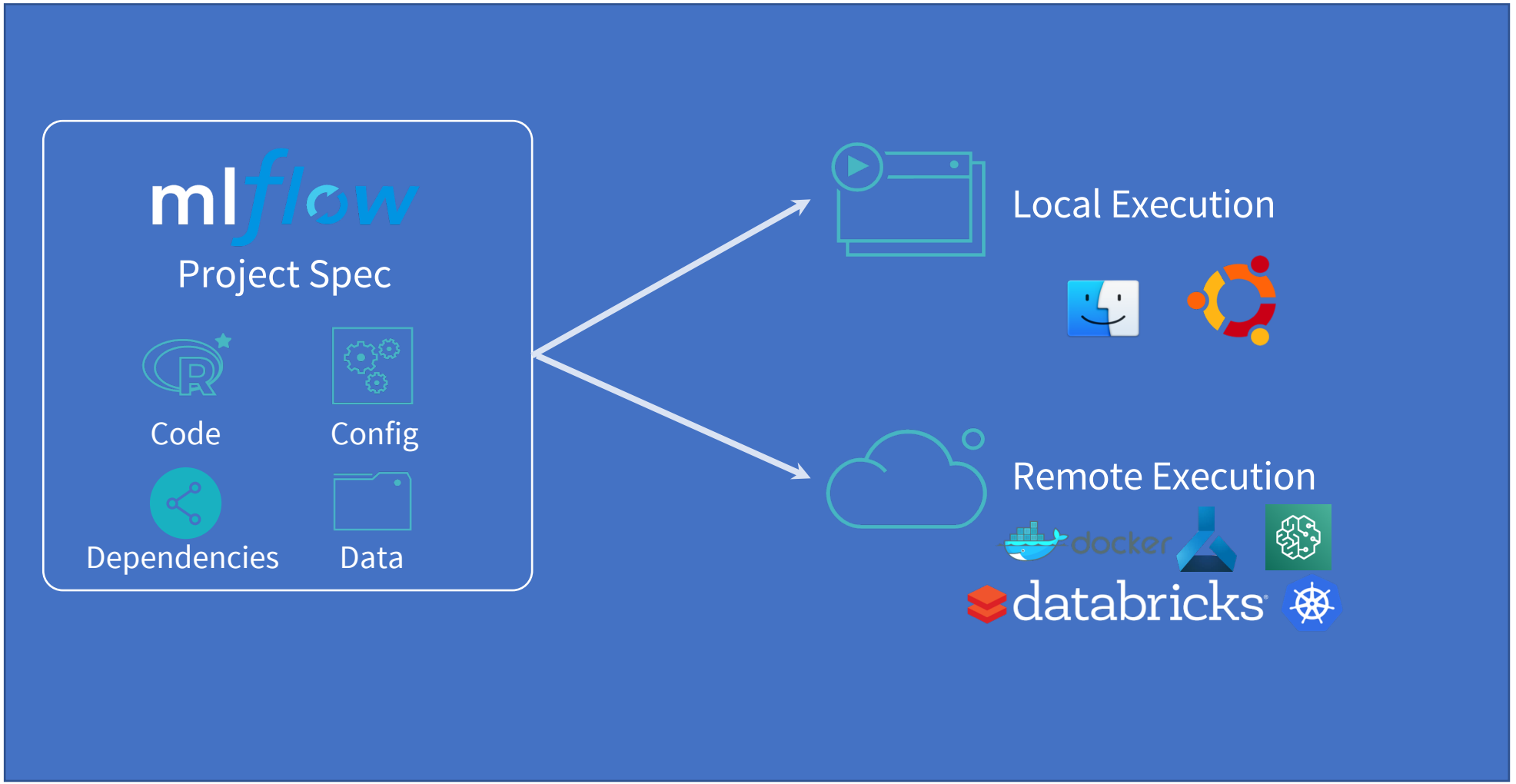
MLflow – Projects

Reusability, reproducibility and versatility

MLflow – Open ML Platform

Projects

Packaging format for reproducible runs on any platform



Reproducibility, Sharing, Productionalization

MLflow – Projects

Projects

Packaging format for reproducible runs on any platform

```
my_project/
├── MLproject
│
│
│
│
│
│
│
├── conda.yaml
├── main.py
└── model.py
    ...
```

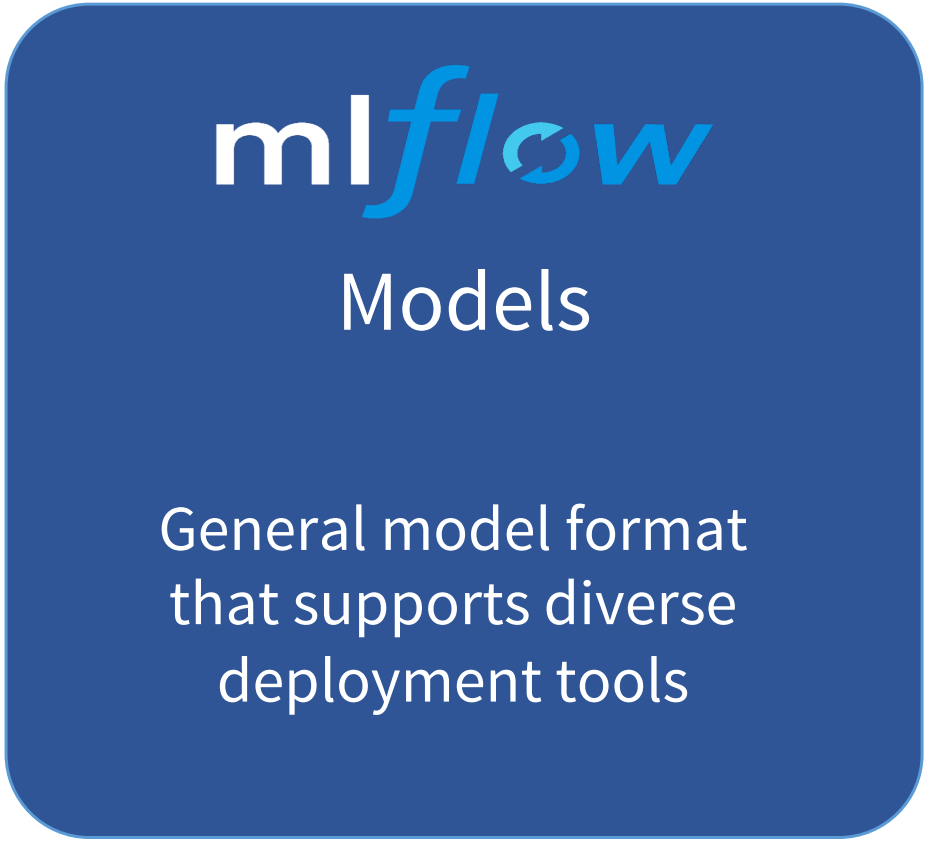
MLproject file

```
conda_env: conda.yaml (or)
docker_env:
  image: mlflow-image

entry_points:
  main:
    parameters:
      training_data: path
      lambda: {type: float, default: 0.1}
    command: python main.py {training_data}
                    {lambda}
```

```
$ mlflow run <directory> or git://<my_project>
mlflow.run("<directory> or git://<my_project>")
```

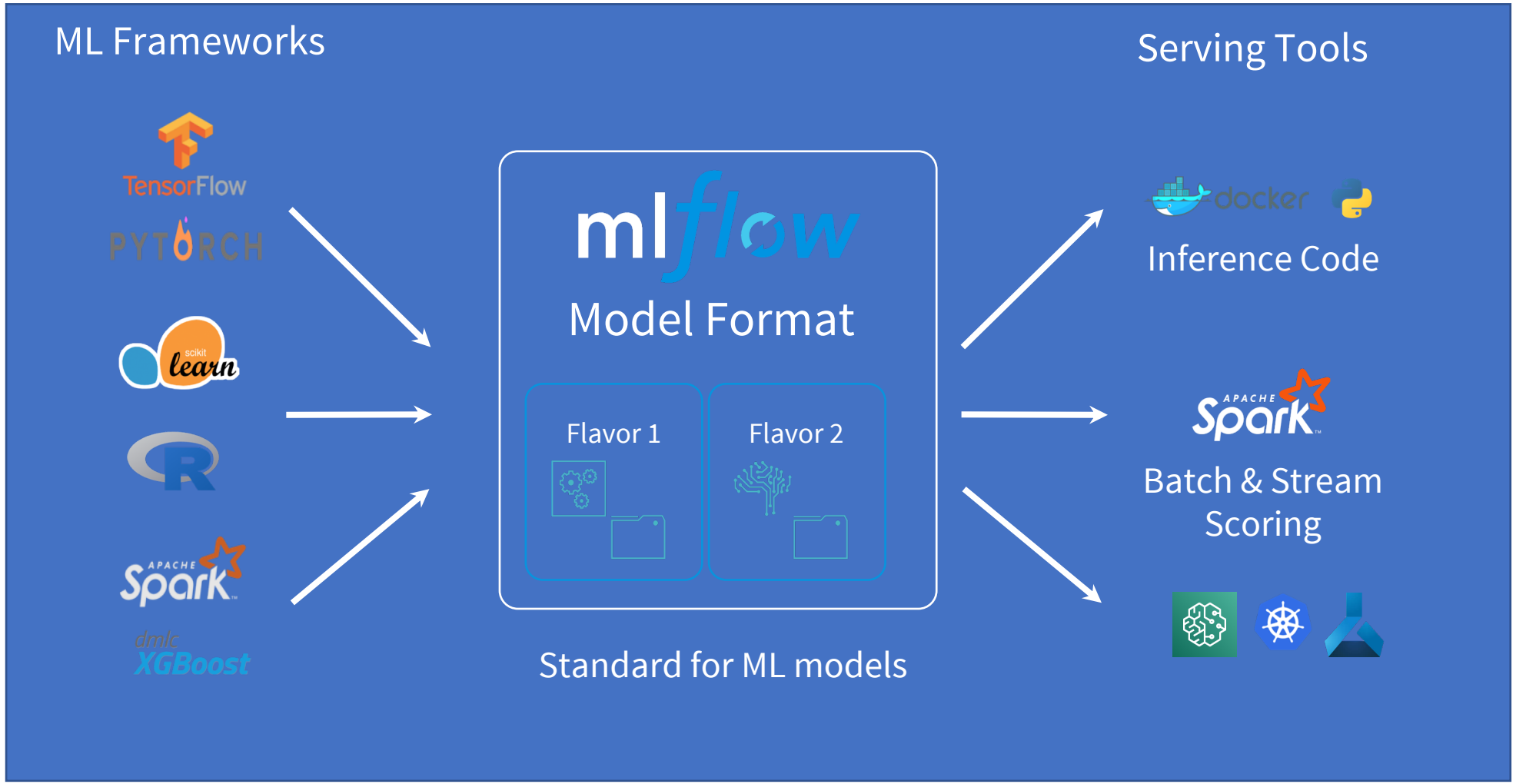
MLflow – Models



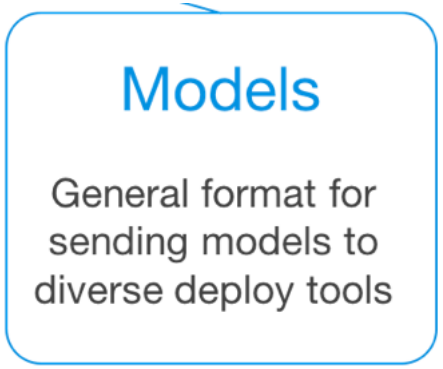
MLflow – Models

Models

General format for sending models to diverse deploy tools



MLflow – Models



- Packaging format for ML Model
  - Directory structure
  - Different flavors
    - (keras, pytorch, sklearn, spark, tensorflow)
- Define dependencies
  - Easy reproducibility & deployment
- Model creation utilities
  - To save model in MLFlow format
- Deployment APIs
  - CLI – Python,R,Java

MLflow – Models

Models

General format for sending models to diverse deploy tools

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

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

```
run_id: 869915006abd4c4bbd662450
time_created: 2019-06-20T08:11
flavors:
  tensorflow:
    saved_model_dir: estimator
    signature_def_key: predict
  python_function:
    loader_module:
            mlflow.tensorflow
```

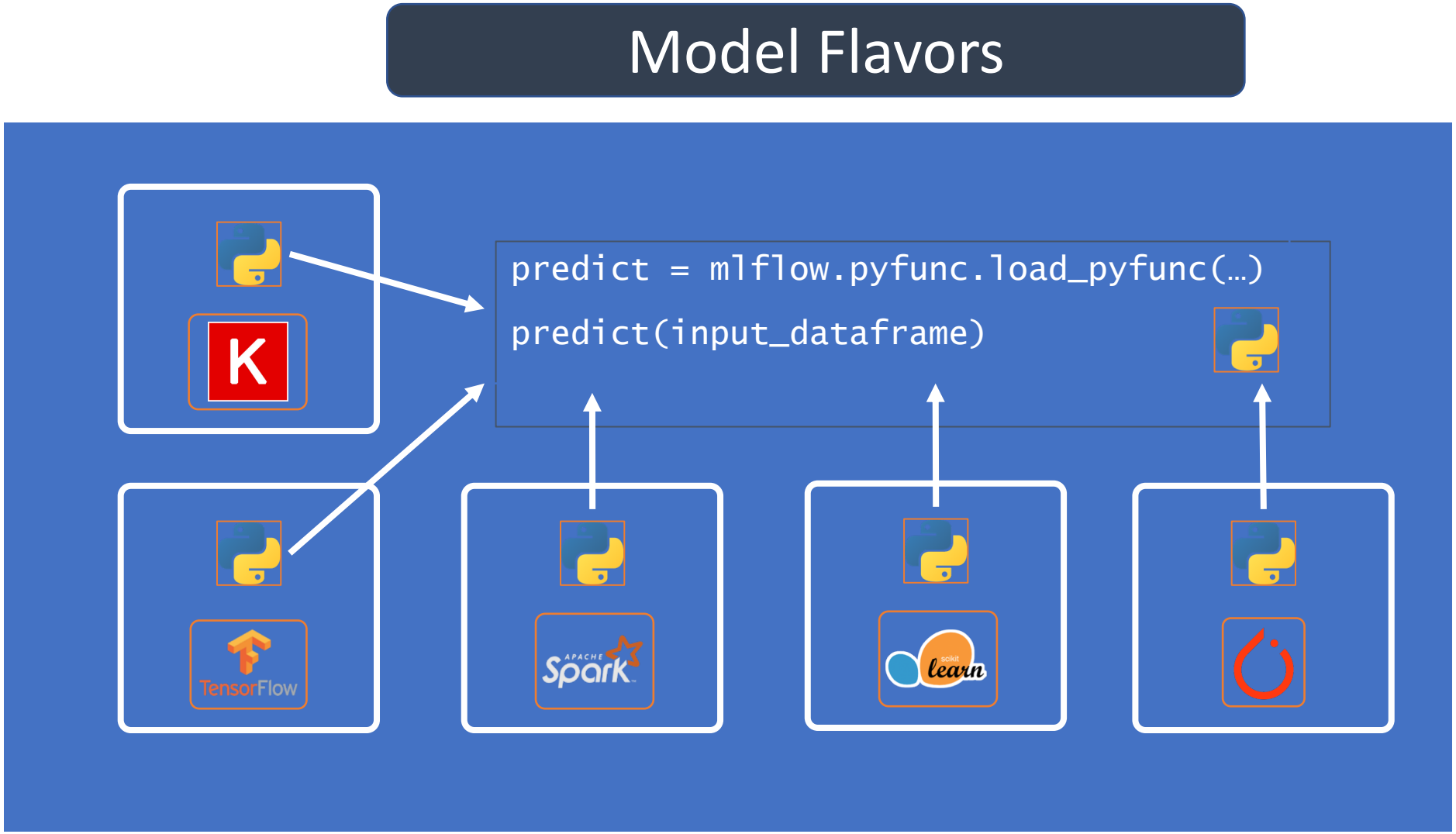
MLflow – Models

Models

General format for sending models to diverse deploy tools

Built-In Flavors

- Python Function
- R,H_2O
- Keras
- Mleap
- PyTorch
- Skiki-learn
- Spark Mllib
- TensorFlow
- ONNX



MLflow – Models

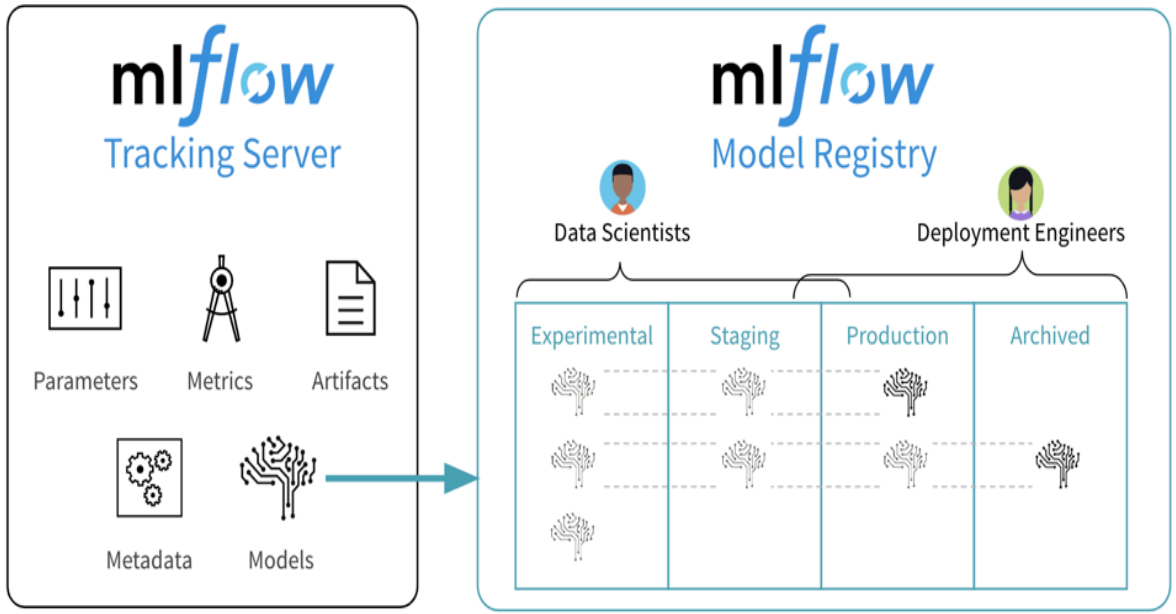
- Model Deployment
  - Local machine
  - Cloud providers – Azure, AWS
- Model Deployment Tools
  - Command line
  - APIs
- Model Serving
  - As REST API endpoint
  - Package as Docker image with REST API endpoint

MLflow – Models

MLflow – Model Registry

Managing Models Collaboratively

- Model lineage
  - From experiment and run
- Model administration and review
  - Sharing, versioning, approval workflow
- Integrate with MLflow tracking
- Centralized activity logs and comments
- Integration with CI/CD



Q&A

Let’s have fun

https://github.com/hluu/qcon-miniworkshop