

# Overview of Lakehouse AI

INTRODUCTION TO DATABRICKS



**Kevin Barlow**  
Data Practitioner

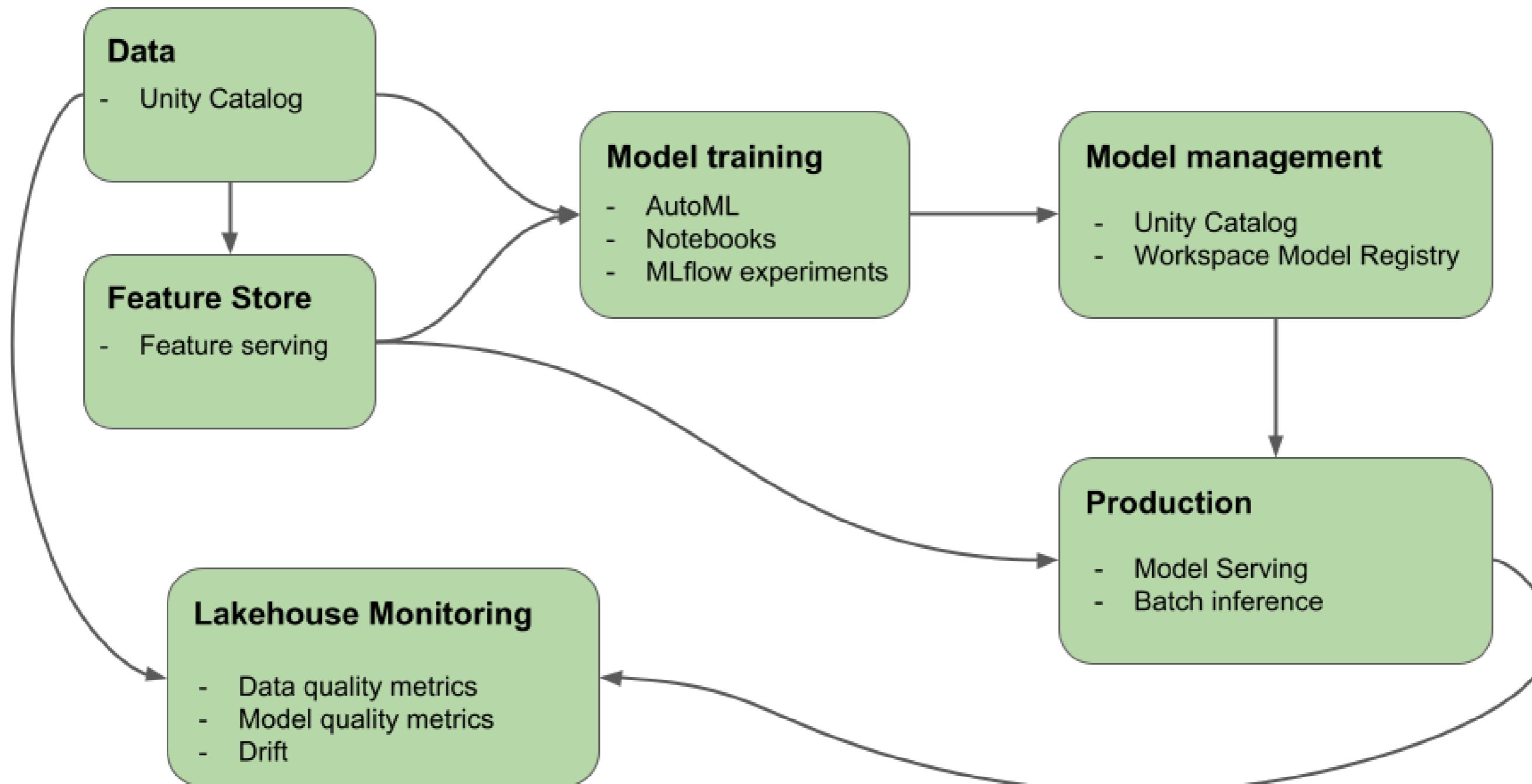
# Lakehouse AI



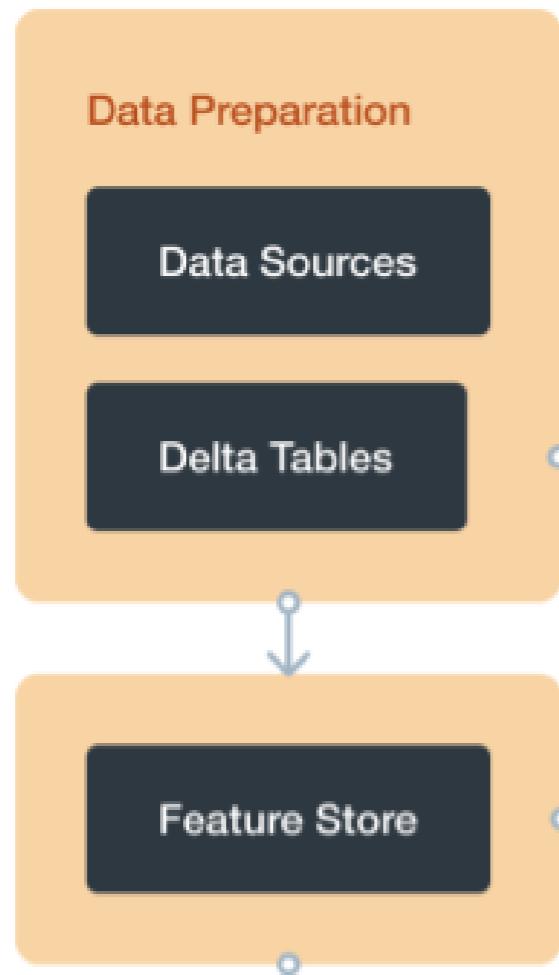
## Why the Lakehouse for AI / ML?

1. Reliable data and files in the Delta lake
2. Highly scalable compute
3. Open standards, libraries, frameworks
4. Unification with other data teams

# MLOps Lifecycle



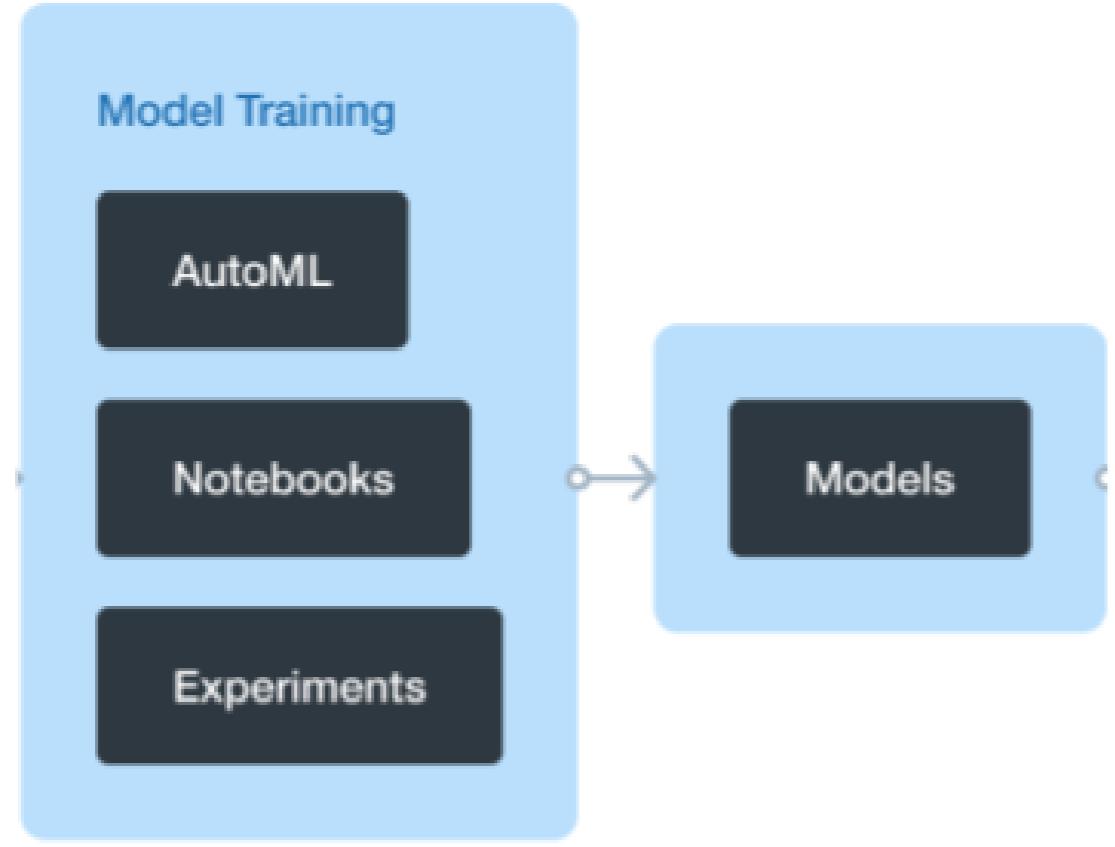
# MLOps in the Lakehouse



## DataOps

- Integrating data across different sources (*AutoLoader*)
- Transforming data into a usable, clean format (*Delta Live Tables*)
- Creating useful features for models (*Feature Store*)

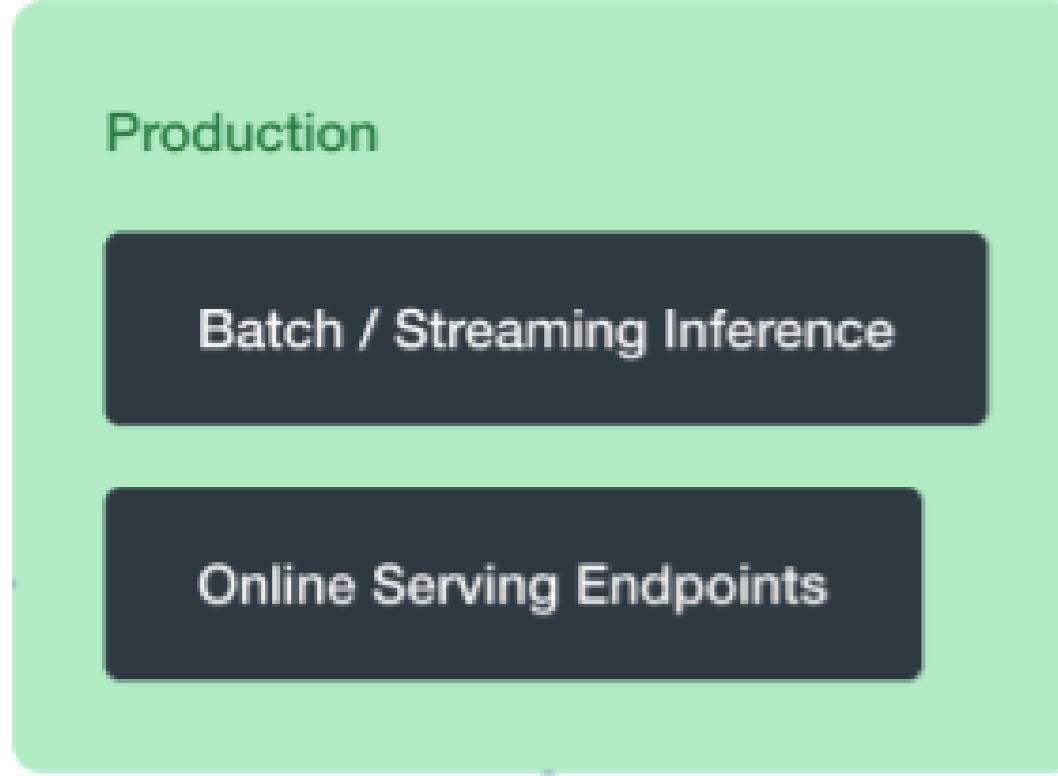
# MLOps in the Lakehouse



## ModelOps

- Develop and train different models (*Notebooks*)
- Machine learning templates and automation (*AutoML*)
- Track parameters, metrics, and trials (*MLFlow*)
- Centralize and consume models (*Model Registry*)

# MLOps in the Lakehouse



## DevOps

- Govern access to different models (*Unity Catalog*)
- Continuous Integration and Continuous Deployment (CI/CD) for model versions (*Model Registry*)
- Deploy models for consumption (*Serving Endpoints*)

# **Let's review!**

**INTRODUCTION TO DATABRICKS**

# Using Databricks for machine learning

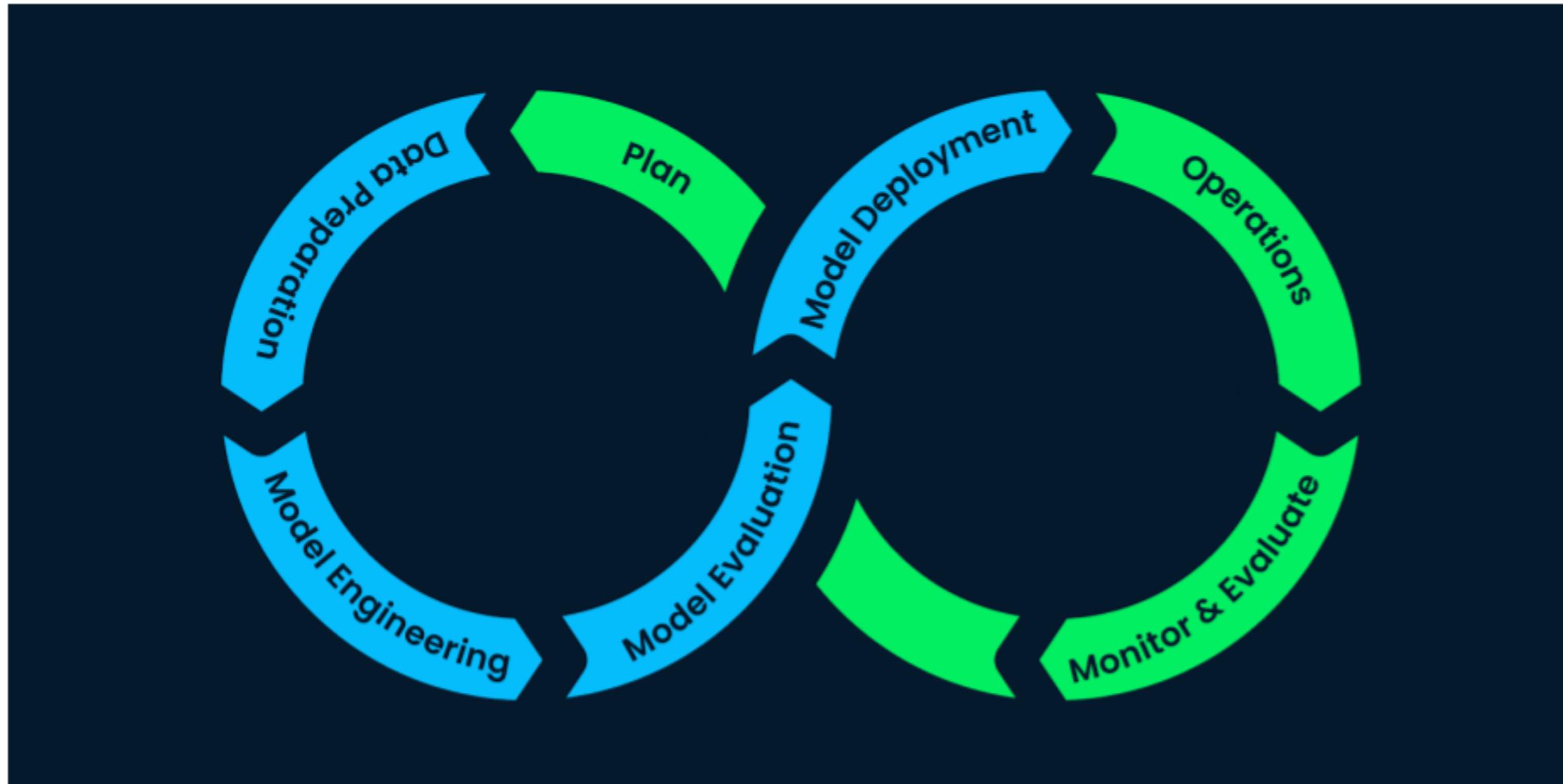
INTRODUCTION TO DATABRICKS



**Kevin Barlow**

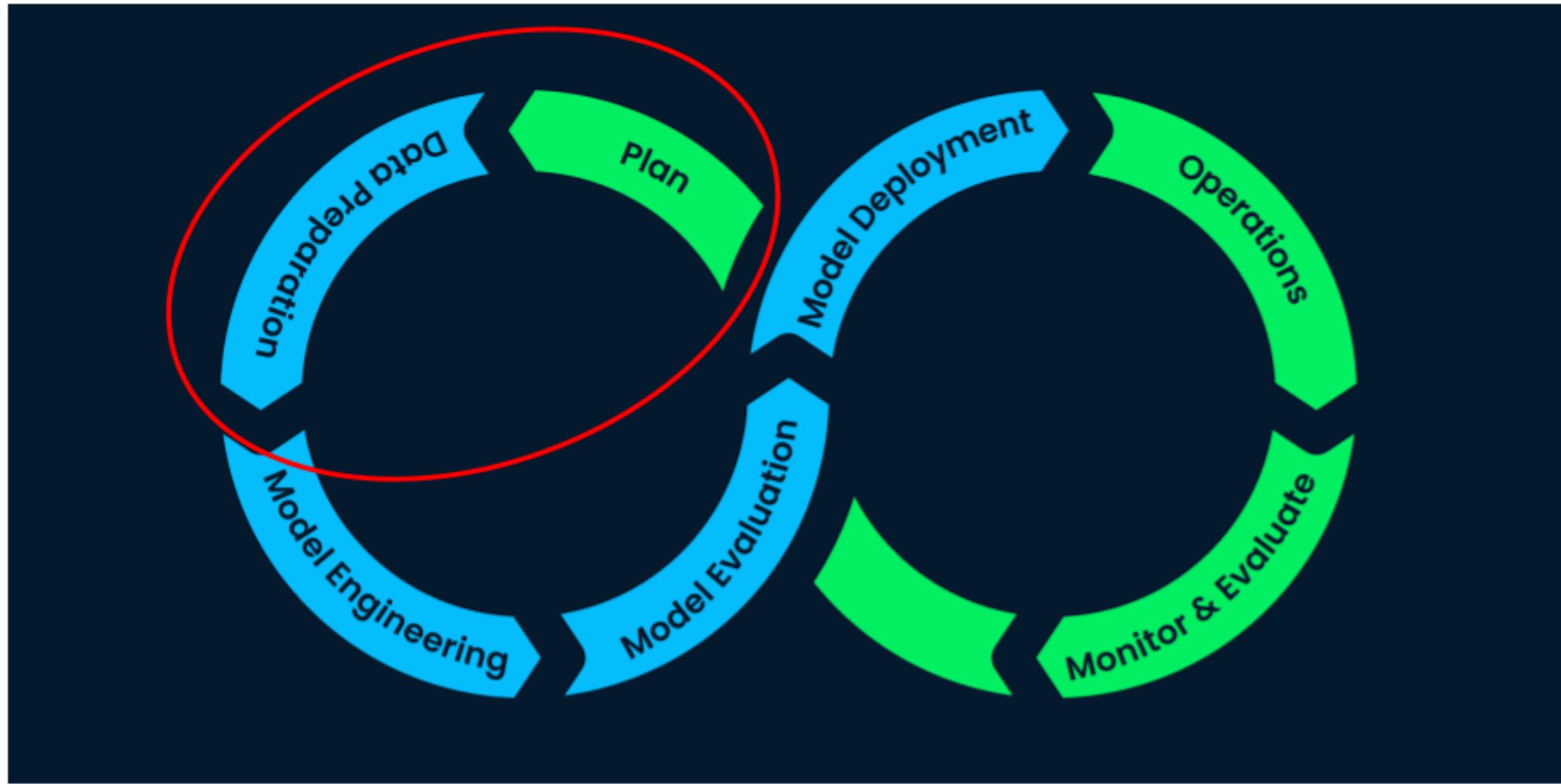
Data Practitioner

# Machine Learning Lifecycle



<sup>1</sup> <https://www.datacamp.com/blog/machine-learning-lifecycle-explained>

# Planning and preparation



# Planning for machine learning

What do I have?

1. Data availability
2. Business requirements
3. Data scientists/data analysts



What do I want?

1. Use cases
2. Legal and security compliance
3. Business outcomes



# ML Runtime

- Extension of Databricks compute
- Optimized for machine learning applications
- Contains most common libraries and frameworks
  - scikit-learn , SparkML , TensorFlow
  - MLFlow
- Works with cluster library management

Databricks runtime version 

Runtime: 13.3 LTS (Scala 2.12, Spark 3.4.1)		
Standard	>	14.0 ML Beta includes GPU, Scala 2.12
ML	>	14.0 ML Beta includes Scala 2.12
Uncategorized	>	13.3 LTS ML GPU, Scala 2.12, Spark 3.4.1
		13.3 LTS ML Scala 2.12, Spark 3.4.1
		13.2 ML GPU, Scala 2.12, Spark 3.4.0
		13.2 ML Scala 2.12, Spark 3.4.0
		13.1 ML GPU, Scala 2.12, Spark 3.4.0
		13.1 ML Scala 2.12, Spark 3.4.0
		13.0 ML GPU, Scala 2.12, Spark 3.4.0
		13.0 ML Scala 2.12, Spark 3.4.0
		12.2 LTS ML GPU, Scala 2.12, Spark 3.3.2
		12.2 LTS ML Scala 2.12, Spark 3.3.2

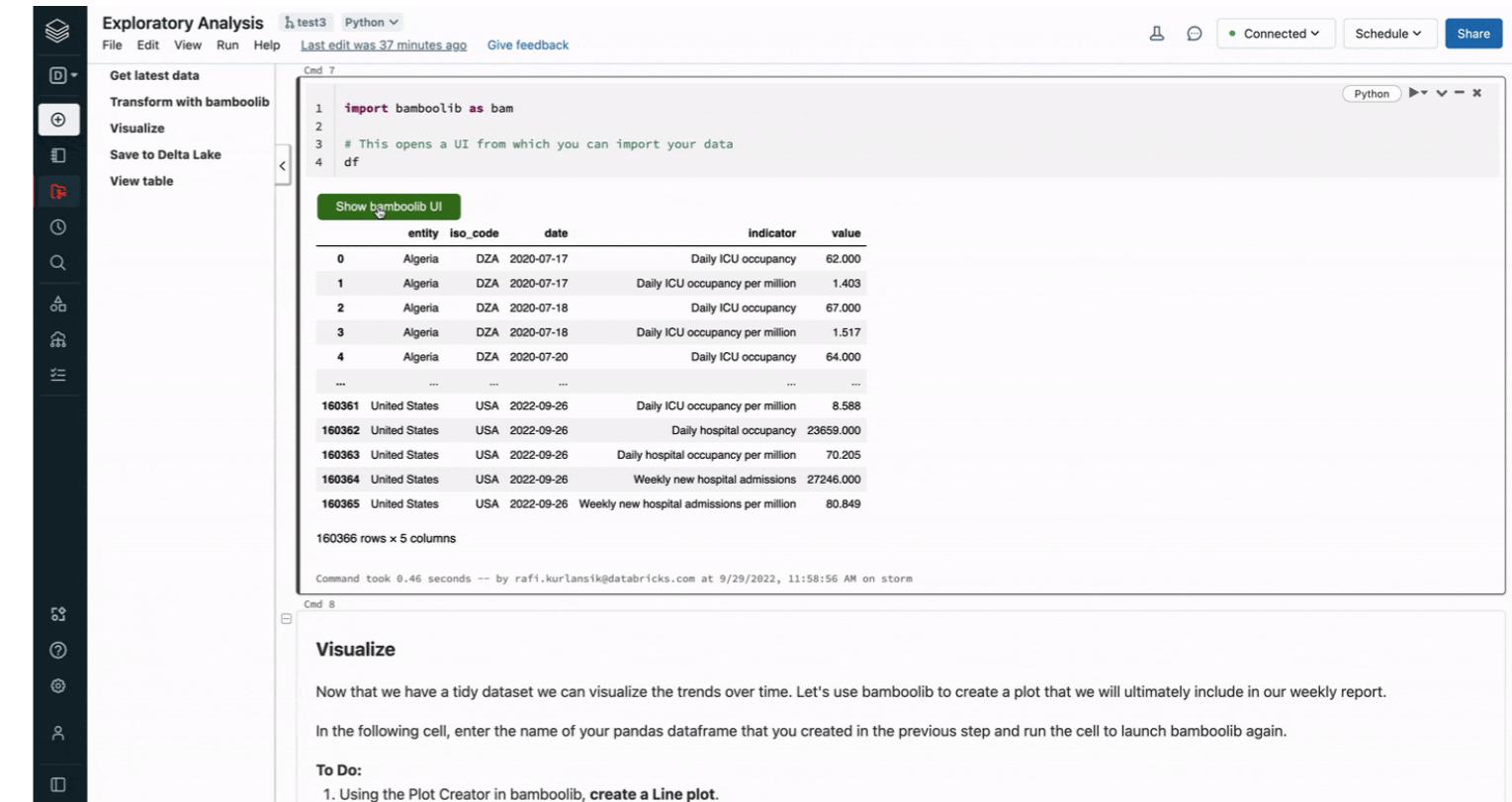
# Exploratory Data Analysis

```
import pandas as pd  
  
pd.describe(df)
```

```
# Spark DF  
  
df.summary()
```

```
dbutils.data.summarize()
```

```
import bamboolib as bam  
  
df
```



The screenshot shows the Databricks Exploratory Analysis interface. On the left, there's a sidebar with various icons for data management. The main area has two code cells and a visualization section.

**Code Cell 7:**

```
import bamboolib as bam  
# This opens a UI from which you can import your data  
df
```

**Code Cell 8:**

**Show bamboolib UI**

entity	Iso_code	date	indicator	value
Algeria	DZA	2020-07-17	Daily ICU occupancy	62.000
Algeria	DZA	2020-07-17	Daily ICU occupancy per million	1.403
Algeria	DZA	2020-07-18	Daily ICU occupancy	67.000
Algeria	DZA	2020-07-18	Daily ICU occupancy per million	1.517
Algeria	DZA	2020-07-20	Daily ICU occupancy	64.000
...	...	...	...	...
160361	United States	USA	Daily ICU occupancy per million	8.588
160362	United States	USA	Daily hospital occupancy	23659.000
160363	United States	USA	Daily hospital occupancy per million	70.205
160364	United States	USA	Weekly new hospital admissions	27246.000
160365	United States	USA	Weekly new hospital admissions per million	80.849

160366 rows x 5 columns

Command took 0.46 seconds -- by rafi.kurlansik@databricks.com at 9/29/2022, 11:58:56 AM on storm

**Visualize**

Now that we have a tidy dataset we can visualize the trends over time. Let's use bamboolib to create a plot that we will ultimately include in our weekly report.

In the following cell, enter the name of your pandas dataframe that you created in the previous step and run the cell to launch bamboolib again.

**To Do:**

1. Using the Plot Creator in bamboolib, create a Line plot.

# Feature tables and feature stores

Raw Data

count	category	price	shelf_loc	rating
4	horror	12.50	end	3
6	romance	13.99	top	4.5
12	sci-fi	16.50	bottom	5
31	romance	9.99	bottom	3.5
23	fantasy	24.99	top	4
18	horror	19.99	end	2.5
19	cooking	17.50	end	4.5
7	fantasy	12.99	top	3
37	sci-fi	14.99	bottom	5

Feature table

count	category	price	shelf_loc	rating
4	1	12.50	1	3
6	2	13.99	2	4.5
12	3	16.50	3	5
31	2	9.99	3	3.5
23	4	24.99	2	4
18	1	19.99	1	2.5
19	5	17.50	1	4.5
7	4	12.99	2	3
37	3	14.99	3	5

# Databricks Feature Store

- Centralized storage for featurized datasets
- Easily discover and re-use features for machine learning models
- Upstream and downstream lineage

The screenshot shows the Databricks Feature Store interface. On the left is a sidebar with options like Machine Learning, Create, Workspace, Repos, Recents, Search, Data, Compute, Jobs, Experiments, Feature Store (which is selected), and Models. The main area displays the details for the 'prod.customer\_features' dataset. It includes information such as creation date (2021-05-07), last written date (2022-04-22 18:48:40), last modified date (2022-04-22 18:48:40), primary keys ('customer\_id, yyyy\_mm\_dd'), and partition keys ('yyyy\_mm\_dd'). It also lists data sources: 'customers.page\_view\_events' and 'customers.receipts'. Below this, there's a 'Producers' section showing a single job named 'feature\_creator job' with a schedule of 'At 06:00 PM (US/Pacific)', status 'Running', last run at '2022-04-22 18:50:22', and last written at '2022-04-22 18:48:40'. The final section is 'Features (7)', which lists four features: 'page\_visits\_7d', 'page\_visits\_daily', 'total\_purchases\_30d', and 'total\_purchases\_6m'. Each feature has its data type (FLOAT), consumers (models like 'product\_recommendation\_model/1' and endpoints like 'product\_recommendation\_model/1'), and associated jobs and notebooks.

```
from databricks import feature_store  
  
fs = feature_store.FeatureStoreClient()  
  
fs.create_table(  
    name=table_name,  
    primary_keys=["wine_id"],  
    df=features_df,  
    schema=features_df.schema,  
    description="wine features"  
)
```

# **Let's practice!**

## INTRODUCTION TO DATABRICKS

# Model training with MLFlow in Databricks

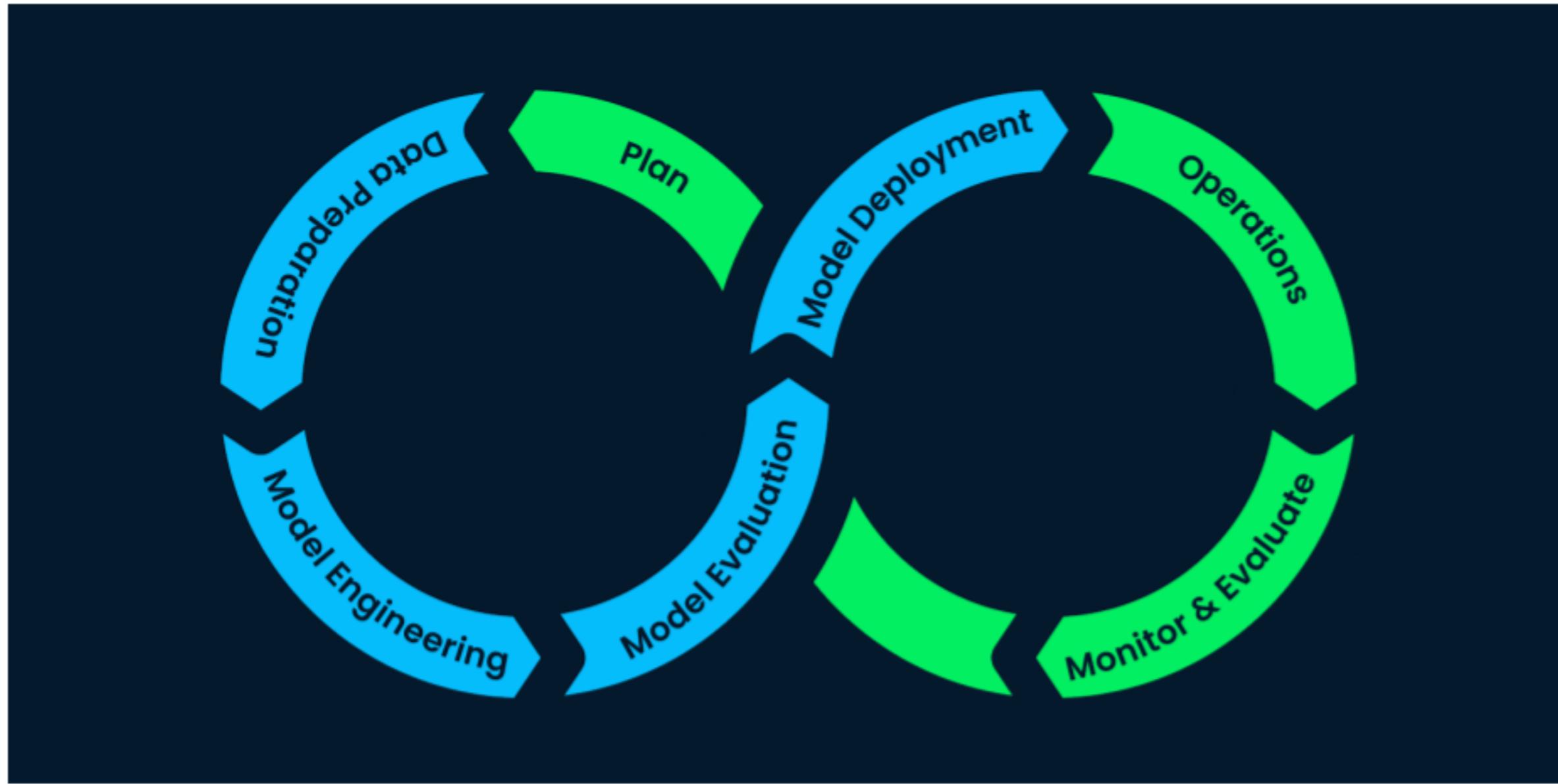
INTRODUCTION TO DATABRICKS



**Kevin Barlow**

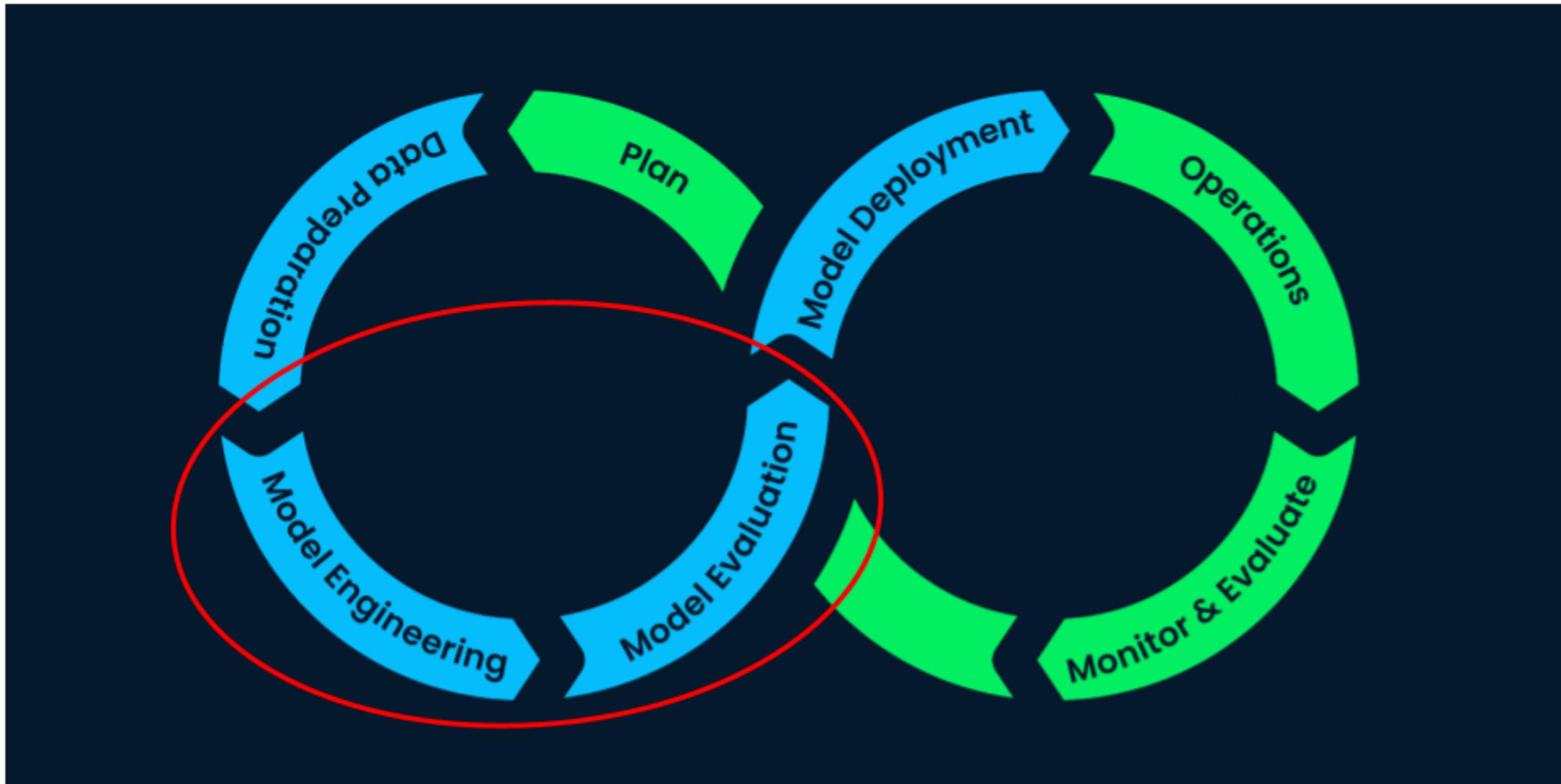
Data Practitioner

# Machine Learning Lifecycle



<sup>1</sup> <https://www.datacamp.com/blog/machine-learning-lifecycle-explained>

# Model training and development



# Single-node vs. Multi-node

## Single-node machine learning

- Great for experimenting and starting
- Easier initial setup
- Hard to implement in production



## Multi-node machine learning

- Great for production workloads
- Easier maintenance long-term
- Highly scalable



# AutoML

- "Glass box" approach to automated machine learning
- Leverages open-source libraries
- Creates models based on data and targeted prediction
- Provides notebook with generated code for further

The screenshot shows the 'Configure AutoML experiment' page in the Databricks interface. On the left is a vertical toolbar with icons for various data and ML operations. The main area has a header 'Configure AutoML experiment' with tabs for 'Preview' and 'Provide Feedback'. Below the header is a breadcrumb 'Experiments > Configure AutoML experiment'. A progress bar at the top indicates steps 1 (Configure), 2 (Train), and 3 (Evaluate). The 'Configure' step is expanded, showing the 'AutoML Experiment Configuration' section. It includes fields for 'Compute' (set to 'ML shard (8.x)'), 'ML problem type' (set to 'Classification'), 'Training data' (set to 'ibm\_telco\_churn.bronze\_customers'), and 'Target column' (set to 'churnString'). To the right of the configuration is a 'Schema:' section with a table showing the dataset columns:

	col_name	data_type	comment
1	customerID	string	
2	gender	string	
3	seniorCitizen	double	
4	partner	string	
5	dependents	string	
6	tenure	double	
7	phoneService	string	

Below the table, it says 'Showing all 24 rows.'

# MLFlow

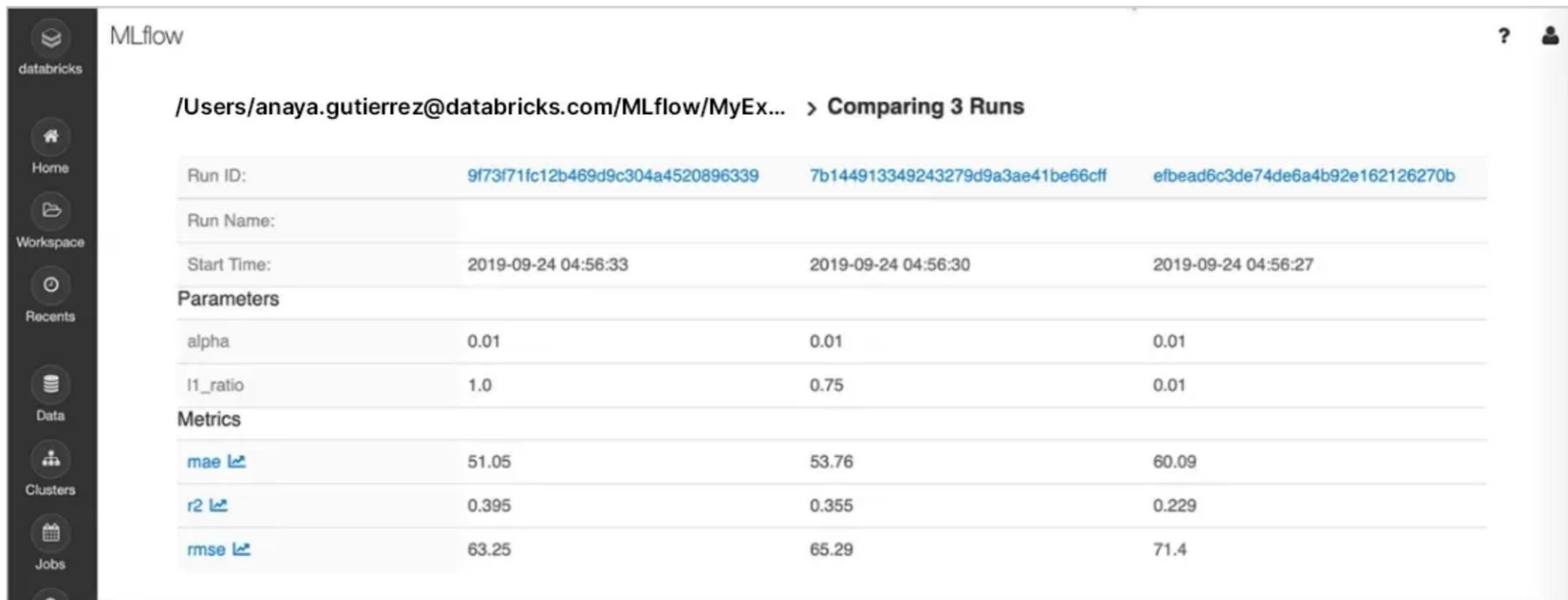
- Open-source framework
- End-to-end machine learning lifecycle management
- Track, evaluate, manage, and deploy
- Pre-installed on ML Runtime!

The logo for MLflow, featuring the word "mlflow" in a lowercase sans-serif font. The letters are primarily blue, except for the "m" which is black. A small trademark symbol (TM) is located at the top right of the "W".

```
import mlflow  
  
with mlflow.start_run() as run:  
    # machine learning training  
  
    mlflow.autolog()  
  
    mlflow.log_metric('accuracy', acc)  
  
    mlflow.log_param('k', kNum)
```

# MLFlow Experiments

- Collect information across multiple runs in a single location
- Sort and compare model runs
- Find and promote the best model



The screenshot shows the Databricks MLflow interface comparing three machine learning runs. The left sidebar includes icons for Home, Workspace, Recents, Data, Clusters, and Jobs. The main area displays the following data:

Run ID:	9f73f71fc12b469d9c304a4520896339	7b144913349243279d9a3ae41be66cff	efbead6c3de74de6a4b92e162126270b
Run Name:			
Start Time:	2019-09-24 04:56:33	2019-09-24 04:56:30	2019-09-24 04:56:27
Parameters			
alpha	0.01	0.01	0.01
l1_ratio	1.0	0.75	0.01
Metrics			
mae ↗	51.05	53.76	60.09
r2 ↗	0.395	0.355	0.229
rmse ↗	63.25	65.29	71.4

# **Let's practice!**

## **INTRODUCTION TO DATABRICKS**

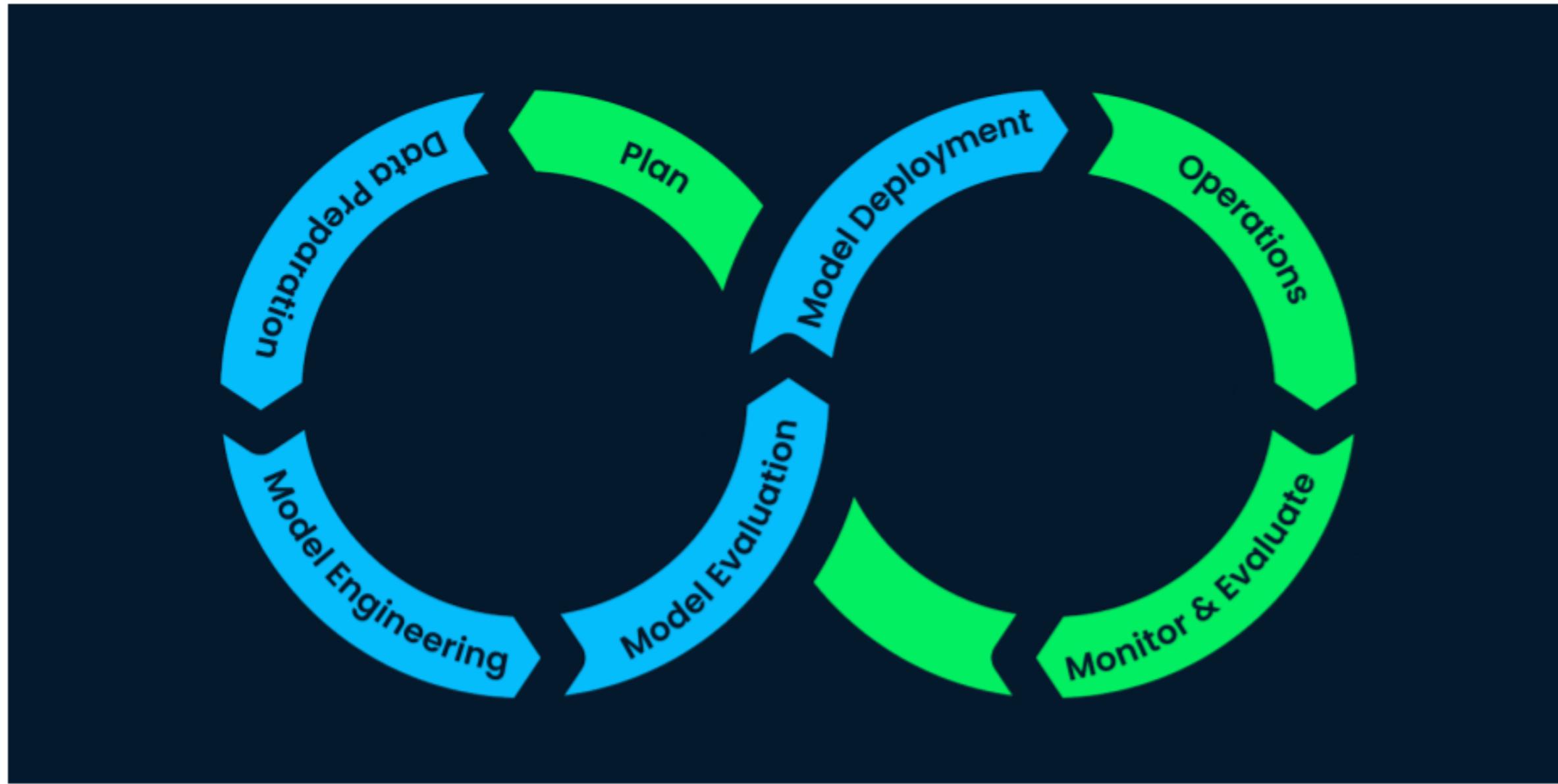
# Deploying a model in Databricks

INTRODUCTION TO DATABRICKS



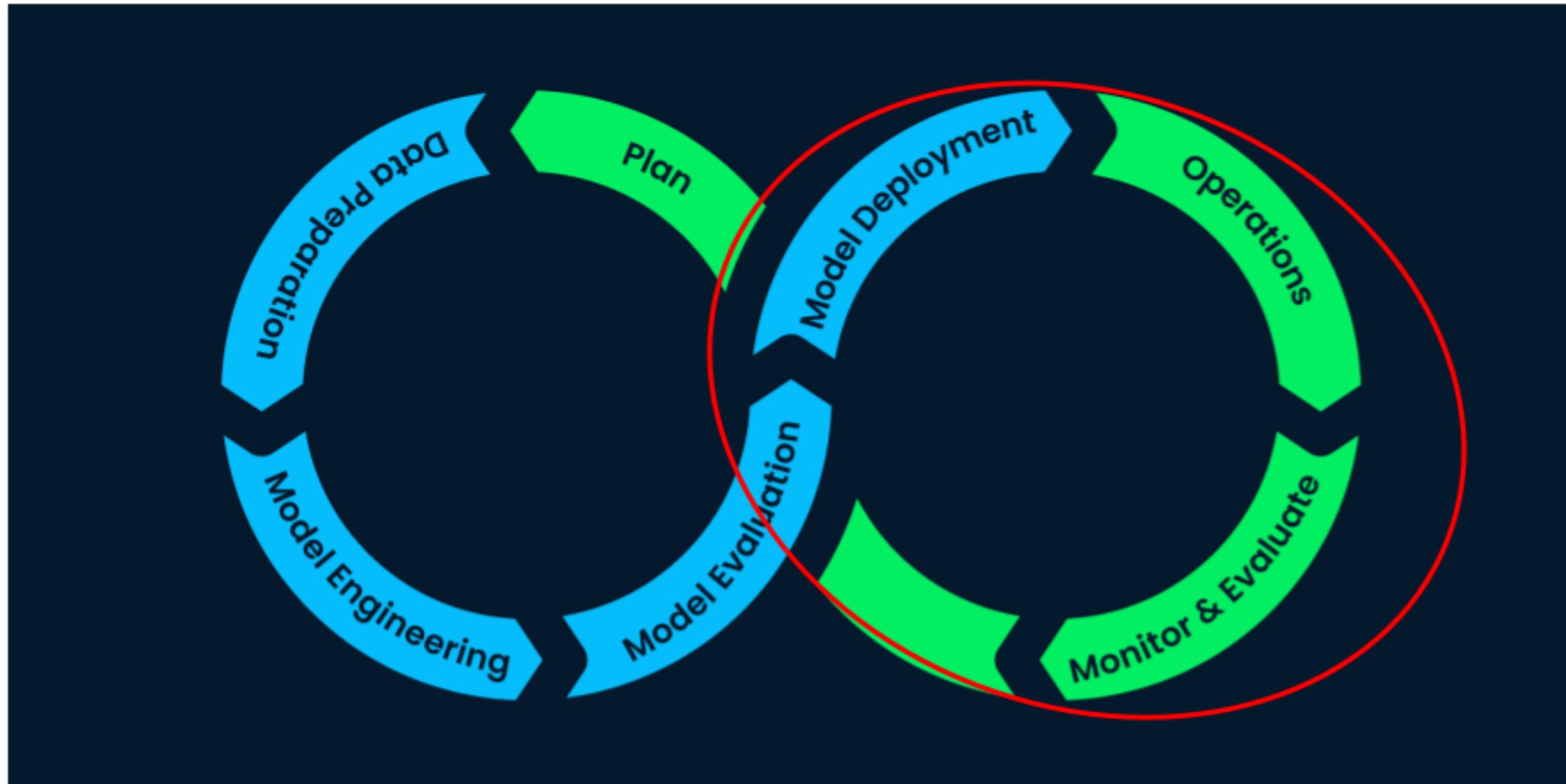
**Kevin Barlow**  
Data Practitioner

# Machine Learning Lifecycle



<sup>1</sup> <https://www.datacamp.com/blog/machine-learning-lifecycle-explained>

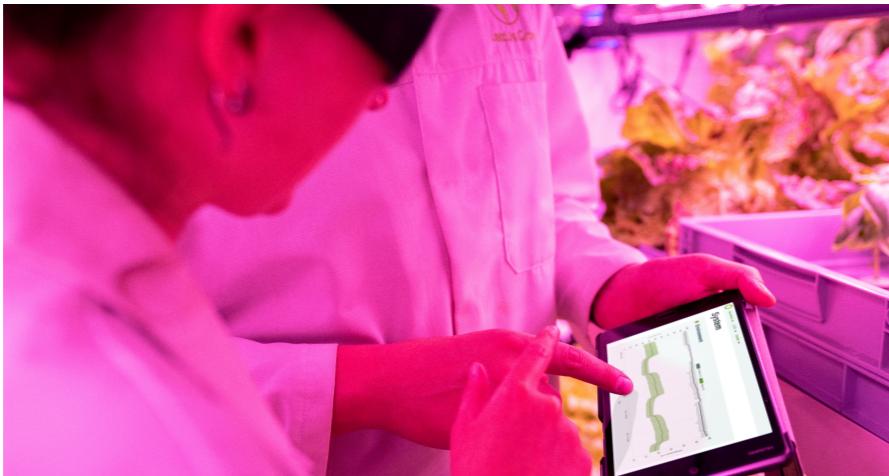
# Model Deployment and Operations



# Concerns with deploying models

## Availability

- How will my end users or application use the model?
- Where do I need to put my model to access it?
- Will the model be easy to understand or use?

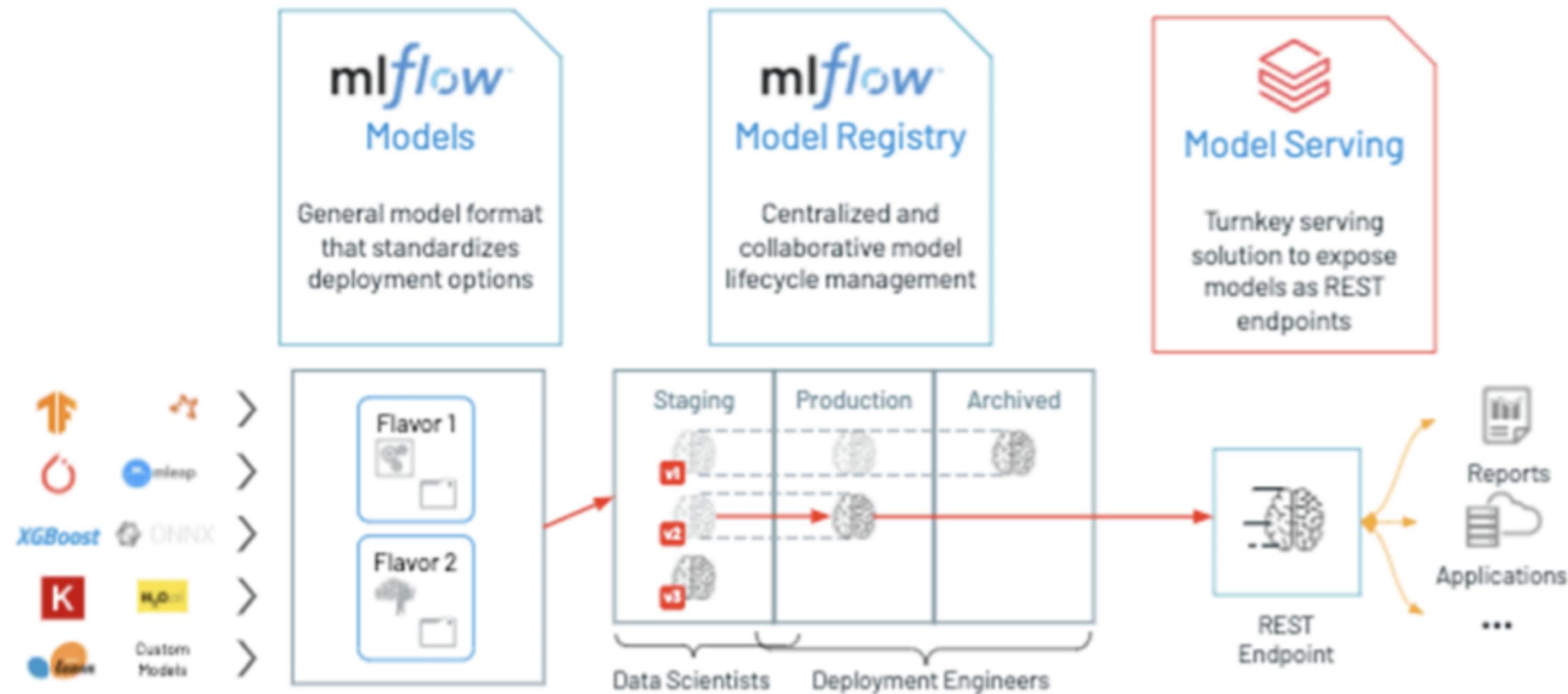


## Evaluation

- Are my users *actually* using my model?
- Is my model still performing well?
- Do I need to retrain my model?
- Do I need a new model that is better?



# Model Deployment Process



# Model Flavors

- MLFlow Models can store a model from any machine learning framework
- Models are stored alongside different configurations and artifacts
- Models can be "translated" into another kind of model based on needs. For example:
  - scikit-learn
  - pyfunc
  - spark
  - tensorflow



# Model Registry

The screenshot shows the Databricks Model Registry interface. On the left is a dark sidebar with navigation icons for Home, Workspace, Recents, Data, Clusters, Jobs, Models, and Search. The main area is titled "Registered Models" and contains a table with the following data:

Name	Latest Version	Staging	Production	Last Modified
Item_Recommender	Version 5	Version 5	Version 4	2019-10-11 15:30:02
Airline_Delay_Scikit	Version 3	—	Version 1	2019-10-11 12:41:43
Airline_Delay_SparkML	Version 5	Version 5	Version 3	2019-10-11 12:45:15
Transaction_Fraud_Classifier	Version 1	—	—	2019-10-11 15:18:05
Icon_GAN	Version 1	—	—	2019-10-12 08:20:12
Power_Forecasting_Model	Version 1	—	Version 1	2019-10-07 15:38:27
Product_Image_Classifier	Version 6	—	Version 5	2019-10-12 00:38:56
Comment_Summarizer	Version 3	Version 2	Version 3	2019-10-12 00:39:40
Movie_Recommender	Version 5	Version 5	Version 3	2019-10-10 14:07:07
Translation_Alpha	—	—	—	2019-10-11 16:45:01

At the bottom right, there are navigation arrows and page numbers (1, 2, 3, >). A search bar at the top right contains the placeholder "search model name".

# Model Registry

The screenshot shows the Databricks Model Registry interface. On the left is a dark sidebar with navigation icons: Home, Workspace, Recents, Data, Clusters, Jobs, Models (selected), and Search. The main area is titled "Registered Models" and contains a table of registered models. A red box highlights the first column of the table. The table has columns: Name, Latest Version, Staging, Production, and Last Modified. The "Production" column is highlighted in green. The table data is as follows:

Name	Latest Version	Staging	Production	Last Modified
Item_Recommender	Version 5	Version 5	Version 4	2019-10-11 15:30:02
Airline_Delay_Scikit	Version 3	—	Version 1	2019-10-11 12:41:43
Airline_Delay_SparkML	Version 5	Version 5	Version 3	2019-10-11 12:45:15
Transaction_Fraud_Classifier	Version 1	—	—	2019-10-11 15:18:05
Icon_GAN	Version 1	—	—	2019-10-12 08:20:12
Power_Forecasting_Model	Version 1	—	Version 1	2019-10-07 15:38:27
Product_Image_Classifier	Version 6	—	Version 5	2019-10-12 00:38:56
Comment_Summarizer	Version 3	Version 2	Version 3	2019-10-12 00:39:40
Movie_Recommender	Version 5	Version 5	Version 3	2019-10-10 14:07:07
Translation_Alpha	—	—	—	2019-10-11 16:45:01

Pagination controls at the bottom right show page 1 of 3.

# Model Registry

The screenshot shows the Databricks Model Registry interface. On the left is a dark sidebar with navigation icons for Home, Workspace, Recents, Data, Clusters, Jobs, Models, and Search. The main area is titled "Registered Models" and contains a table with the following data:

Name	Latest Version	Staging	Production	Last Modified
Item_Recommender	Version 5	Version 5	Version 4	2019-10-11 15:30:02
Airline_Delay_Scikit	Version 3	-	Version 1	2019-10-11 12:41:43
Airline_Delay_SparkML	Version 5	Version 5	Version 3	2019-10-11 12:45:15
Transaction_Fraud_Classifier	Version 1	-	-	2019-10-11 15:18:05
Icon_GAN	Version 1	-	-	2019-10-12 08:20:12
Power_Forecasting_Model	Version 1	-	Version 1	2019-10-07 15:38:27
Product_Image_Classifier	Version 6	-	Version 5	2019-10-12 00:38:56
Comment_Summarizer	Version 3	Version 2	Version 3	2019-10-12 00:39:40
Movie_Recommender	Version 5	Version 5	Version 3	2019-10-10 14:07:07
Translation_Alpha	-	-	-	2019-10-11 16:45:01

A red box highlights the "Latest Version" column header. A search bar at the top right says "search model name". The bottom right shows page navigation with "1" highlighted.

# Model Registry

The screenshot shows the Databricks Model Registry interface. On the left is a dark sidebar with navigation icons for Home, Workspace, Recents, Data, Clusters, Jobs, Models, and Search. The main area is titled "Registered Models" and contains a table with the following data:

Name	Latest Version	Staging	Production	Last Modified
Item_Recommender	Version 5	Version 5	Version 4	2019-10-11 15:30:02
Airline_Delay_Scikit	Version 3	-	Version 1	2019-10-11 12:41:43
Airline_Delay_SparkML	Version 5	Version 5	Version 3	2019-10-11 12:45:15
Transaction_Fraud_Classifier	Version 1	-	-	2019-10-11 15:18:05
Icon_GAN	Version 1	-	-	2019-10-12 08:20:12
Power_Forecasting_Model	Version 1	-	Version 1	2019-10-07 15:38:27
Product_Image_Classifier	Version 6	-	Version 5	2019-10-12 00:38:56
Comment_Summarizer	Version 3	Version 2	Version 3	2019-10-12 00:39:40
Movie_Recommender	Version 5	Version 5	Version 3	2019-10-10 14:07:07
Translation_Alpha	-	-	-	2019-10-11 16:45:01

A red box highlights the "Staging" and "Production" columns. A search bar at the top right contains the placeholder "search model name". Navigation links at the bottom right show page 1 of 3.

# Model Serving

The screenshot shows the Databricks Model Serving interface. At the top, it displays the endpoint name: `churn_prediction_test`. The endpoint state is listed as `Ready`, and the URL is provided: <https://e2-dogfood.staging.cloud.databricks.com/model-endpoint/charlie/1/invocations>. Below this, the "Served models" section lists a single model entry:

Model	Version	Name	State	Compute
churn prediction	Version 1	churn-prediction-1	Ready	Large 16-64 (8 provisioned)

At the bottom, there is a "Call endpoint" section with tabs for "Metrics", "Logs", and "Events". Under "Call endpoint", there are three buttons: "Browser", "Curl", and "Python". Below these buttons are two large text input areas labeled "Request" and "Response", each with a "Send Request" button at the bottom.

# Model Serving

The screenshot shows the Databricks Model Serving interface for a endpoint named "churn\_prediction\_test". The endpoint state is "Ready" and the URL is <https://e2-dogfood.staging.cloud.databricks.com/model-endpoint/charlie/1/invocations>. A red box highlights the "Compute" section which shows "Large 16-64 (8 provisioned)".

Endpoints > churn\_prediction\_test

Endpoint state: Ready

URL: <https://e2-dogfood.staging.cloud.databricks.com/model-endpoint/charlie/1/invocations>

Served models

Model	Version	Name	State	Compute
churn prediction	Version 1	churn-prediction-1	Ready	Large 16-64 (8 provisioned)

Call endpoint Metrics Logs Events

Request Response

Send Request Show Example

# Model Serving

The screenshot shows the Databricks Model Serving interface for a endpoint named "churn\_prediction\_test". The "Served models" table is highlighted with a red box. It contains one row:

Model	Version	Name	State	Compute
churn prediction	Version 1	churn-prediction-1	Ready	Large 16-64 (8 provisioned)

Below the table is a "Call endpoint" section with tabs for "Browser", "Curl", and "Python". It includes "Request" and "Response" fields with "Send Request" and "Show Example" buttons.

# Model Serving



# **Let's practice!**

## **INTRODUCTION TO DATABRICKS**

# Example end-to-end machine learning pipeline

INTRODUCTION TO DATABRICKS



**Kevin Barlow**  
Data Practitioner

# **Let's practice!**

## INTRODUCTION TO DATABRICKS

# Wrap Up

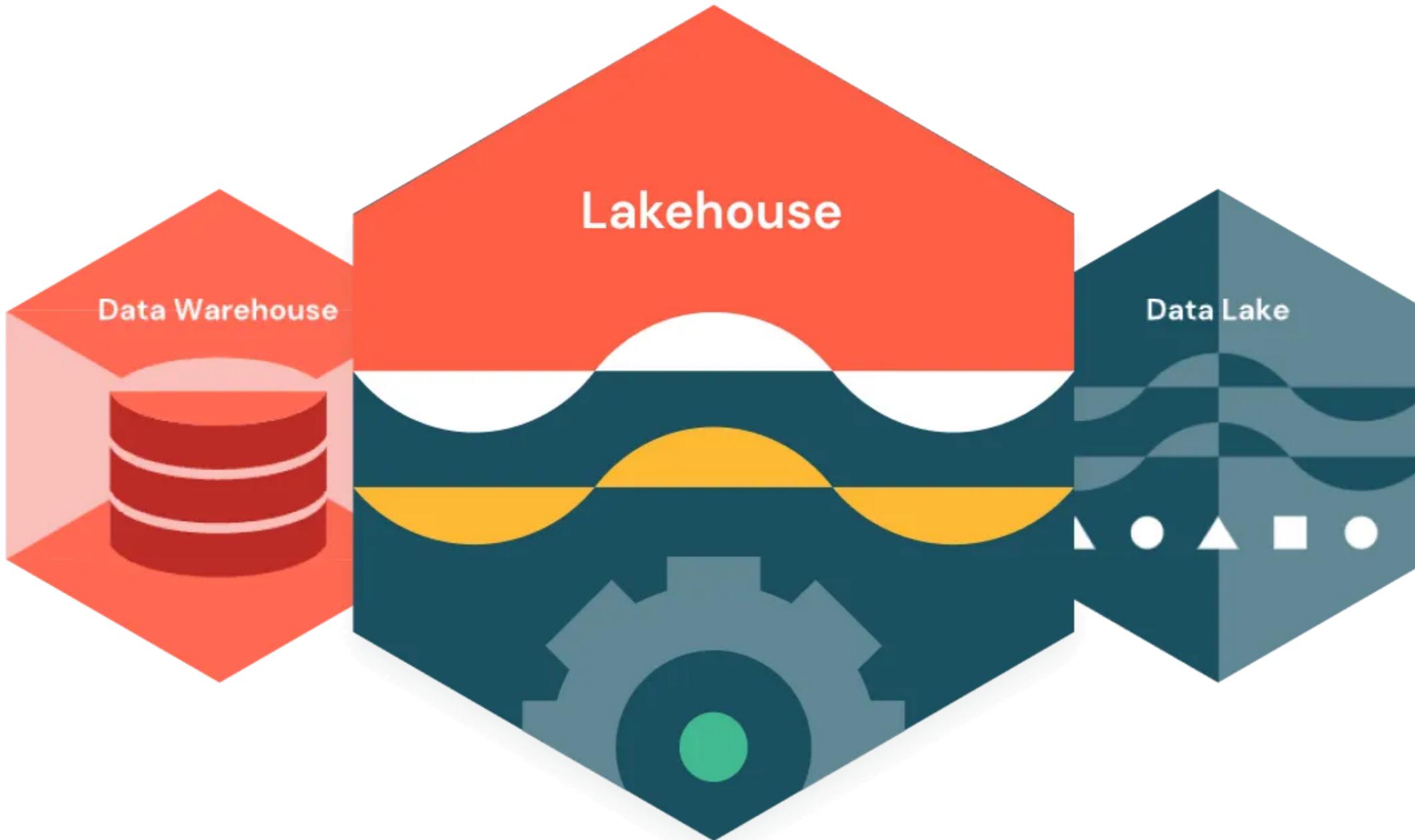
INTRODUCTION TO DATABRICKS



**Kevin Barlow**

Data Practitioner

# Why the Lakehouse?



# Databricks for data engineering

Apache Spark

Delta

Delta Live Tables

Auto Loader

Structured Streaming

Workflows



# Databricks for data warehousing

SparkSQL

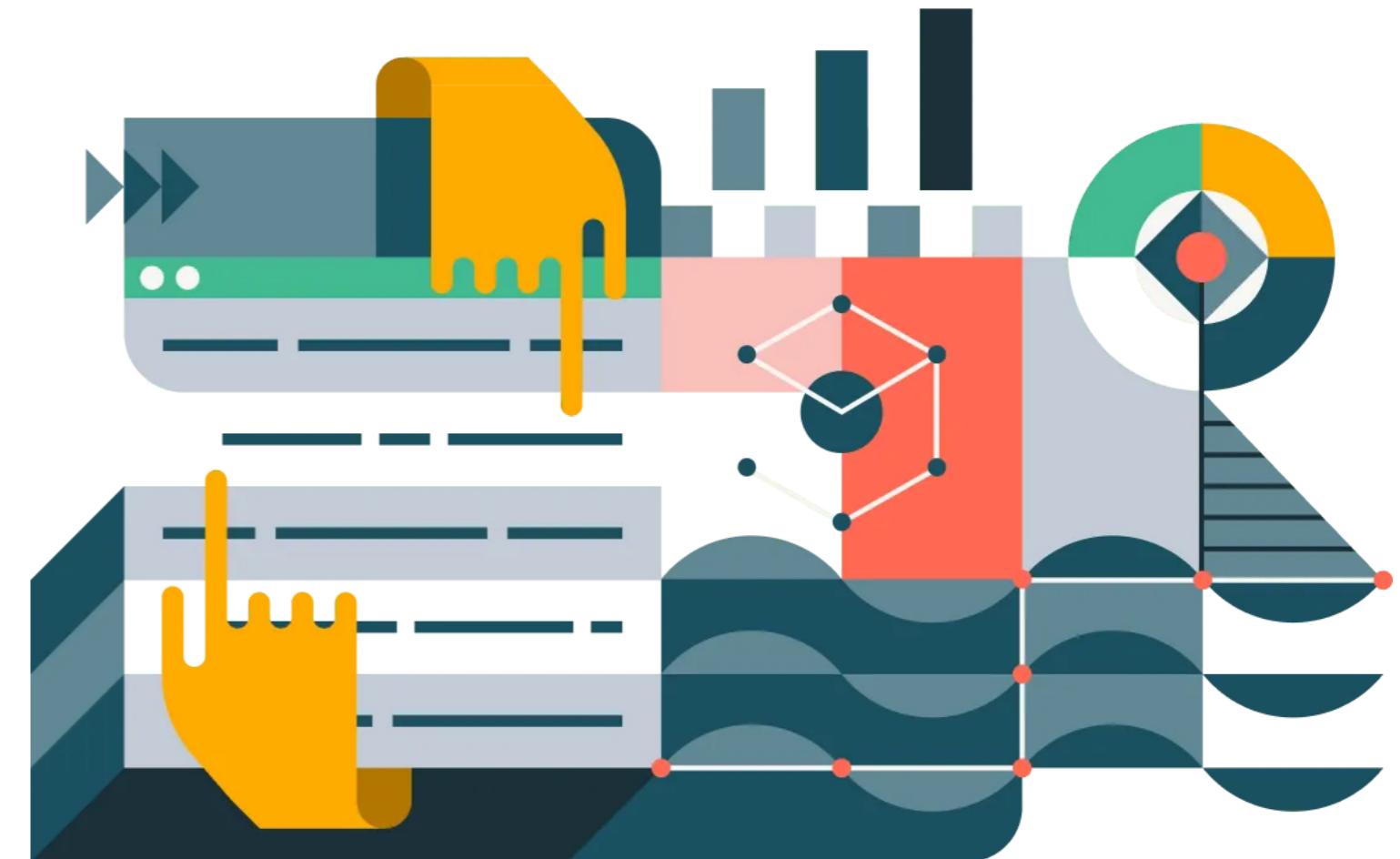
ANSI SQL

SQL Warehouses

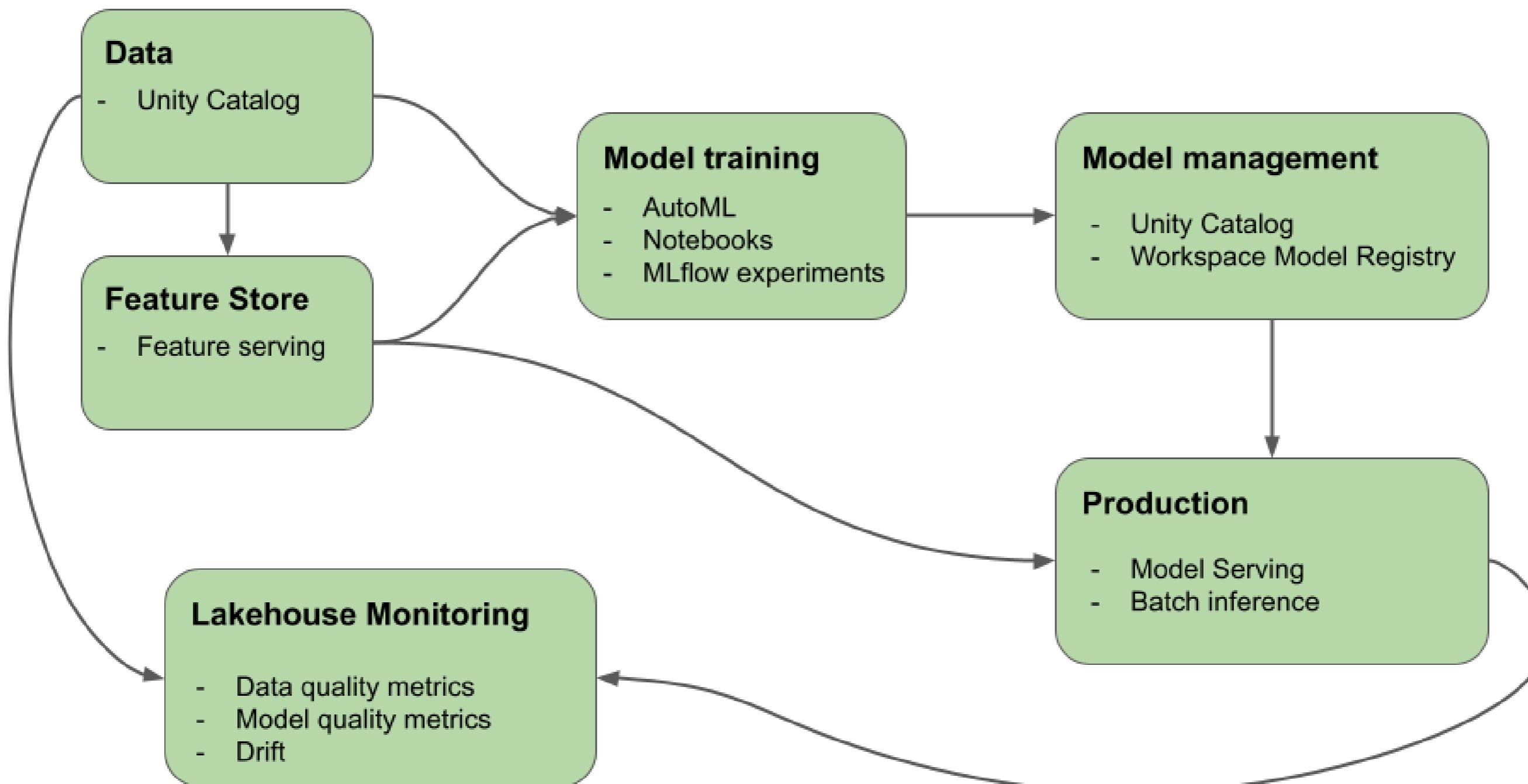
Queries

Visualizations

Dashboards



# Databricks for machine learning



# Congratulations!

INTRODUCTION TO DATABRICKS