



Get Started with Machine Learning on Databricks

We will start at 3 minutes past
the starting time...



Meet your instructor

Venkita Krishnan Mani, Technical Instructor



Now

- Technical Instructor

Then

- Big Data & Spark Consultant
 - JPMC / Citi Bank / Deutsche Bank / Zarantec
- Hadoop Practice Lead –
 - Nichetek / Collabera India / CavalierIT

Interests

- Consulting / Teaching & Mentoring

 [linkedin.com/in/venkitakrishnan](https://www.linkedin.com/in/venkitakrishnan)





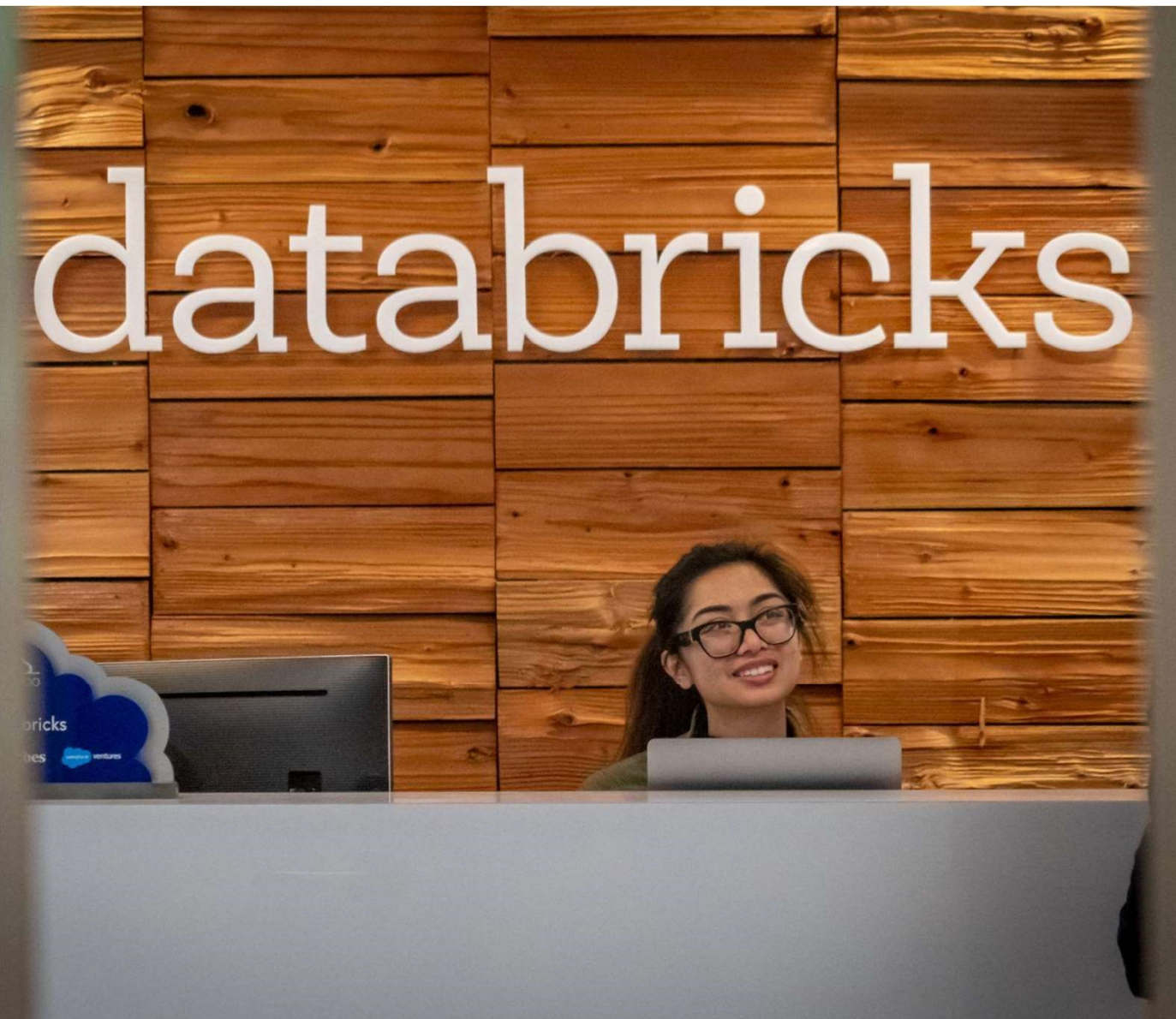
Get Started with Machine Learning on Databricks

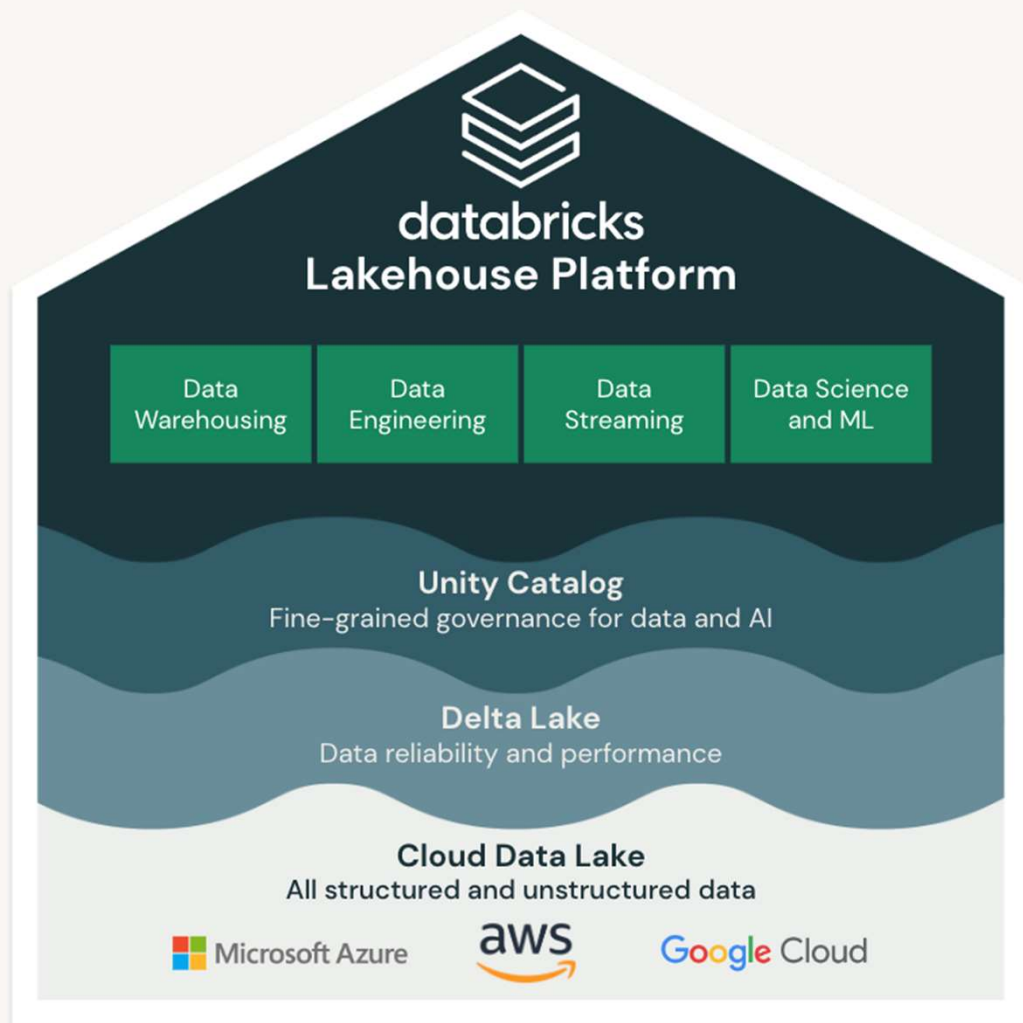


Session topics

- Databricks Lakehouse Platform overview
- Databricks Machine Learning feature dive/demos:
 - UI tour
 - Creating a table with Feature Store
 - Developing a baseline model with AutoML
 - Managing the ML lifecycle using Model Registry
 - Deploying a model for batch inference
 - Scheduling a model refresh with Databricks Workflows

Databricks Lakehouse Platform Overview





Databricks Lakehouse Platform

Simple

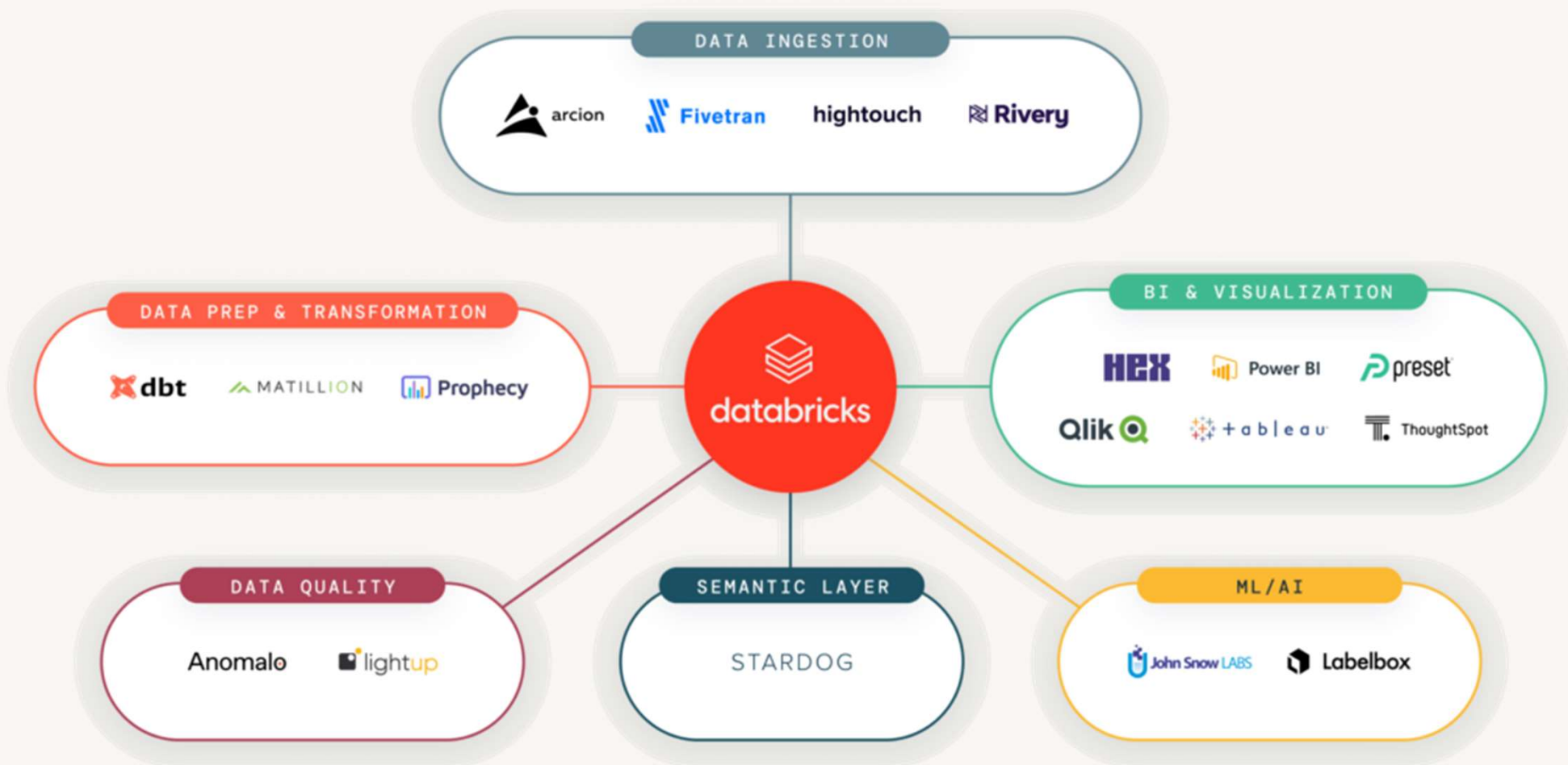
Unify your data warehousing and AI use cases on a single platform

Open

Built on open source and open standards

Multicloud

One consistent data platform across clouds



Introduction to Databricks Machine Learning



Data Science Workspace



AutoML



Data
Ingestion



Model
Training



Model
Tuning



Runtime and
Environment



Batch
Scoring



Online
Serving



Monitoring



Data
Versioning



Feature
Store



Batch (high
throughput)



Real Time
(low latency)



MLOps / Governance

Open Data LakeHouse Foundation with



Benefits of using Databricks ML

- Production machine learning depends on code and data.

Benefits of using Databricks ML

- Production machine learning depends on code and data.
- Supports git repositories for version control.

Benefits of using Databricks ML

- Production machine learning depends on code and data.
- Supports git repositories for version control.
- Full team collaboration support.

Benefits of using Databricks ML

- Production machine learning depends on code and data.
- Supports git repositories for version control.
- Full team collaboration support.
- Development and production workflow support.

Databricks Machine Learning Runtime



- Optimized and preconfigured ML Frameworks
- Turnkey distributed ML
- Built-in AutoML
- GPU support out of the box

Built-in **ML Frameworks** and
model explainability



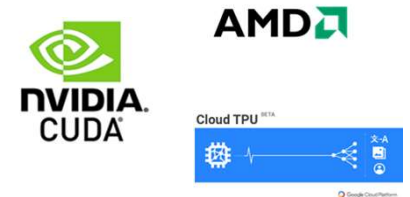
Built-in support for
distributed training



Built-in support for **AutoML**
and **hyperparameter tuning**

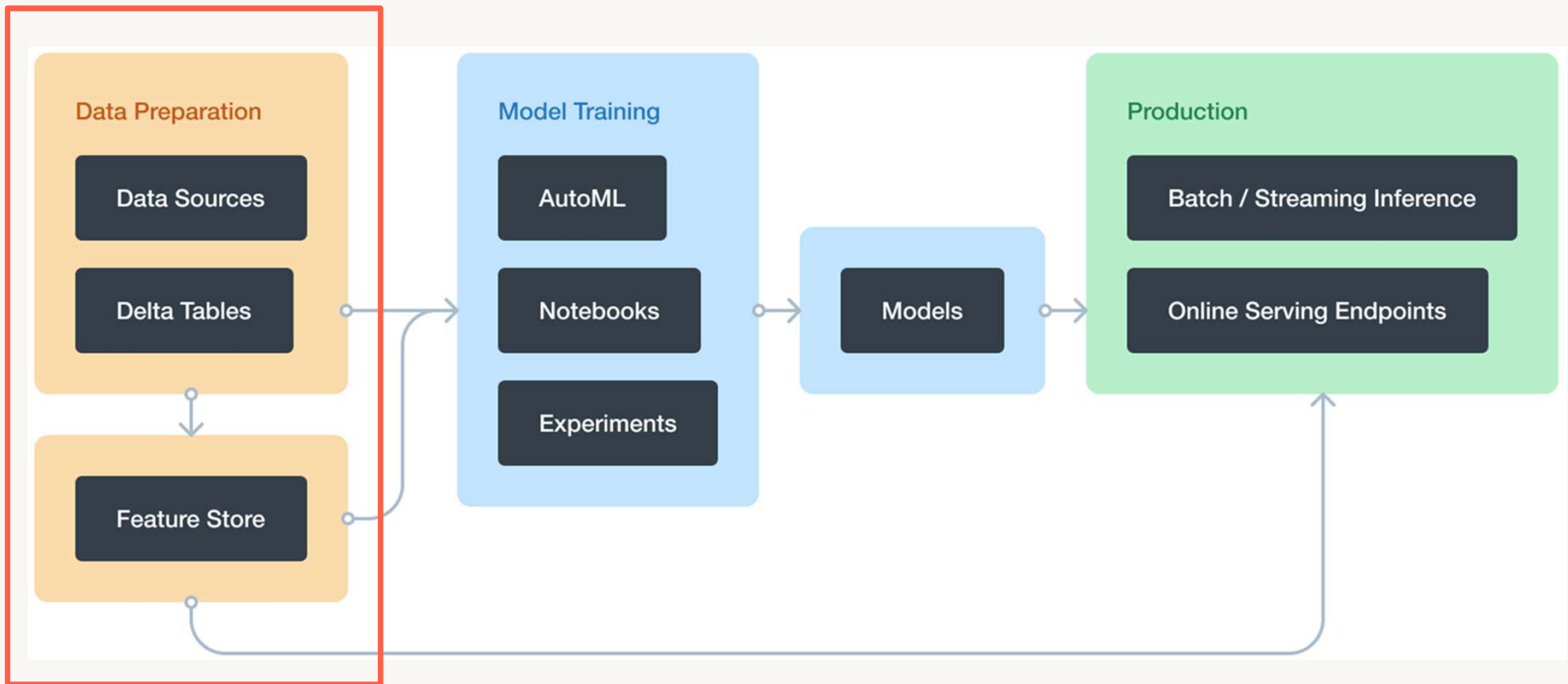


Built-in support for **hardware accelerators**



Tour of the Databricks ML user-interface

Databricks ML Data Preparation with Feature Store





Data Science Workspace



AutoML



Data Ingestion



Model Training



Model Tuning



Runtime and Environment



Batch Scoring



Online Serving



Monitoring



Data Versioning



Feature Store



Batch (high throughput)



Real Time (low latency)



MLOps / Governance

Open Data LakeHouse Foundation with



Benefits of the Feature Store

- Organization wide access and availability.

Benefits of the Feature Store

- Organization wide access and availability.
- Feature Registry for tracking and reuse.

Benefits of the Feature Store

- Organization wide access and availability.
- Feature Registry for tracking and reuse.
- Feature consistency between training and inference.

Delta Lake and MLflow



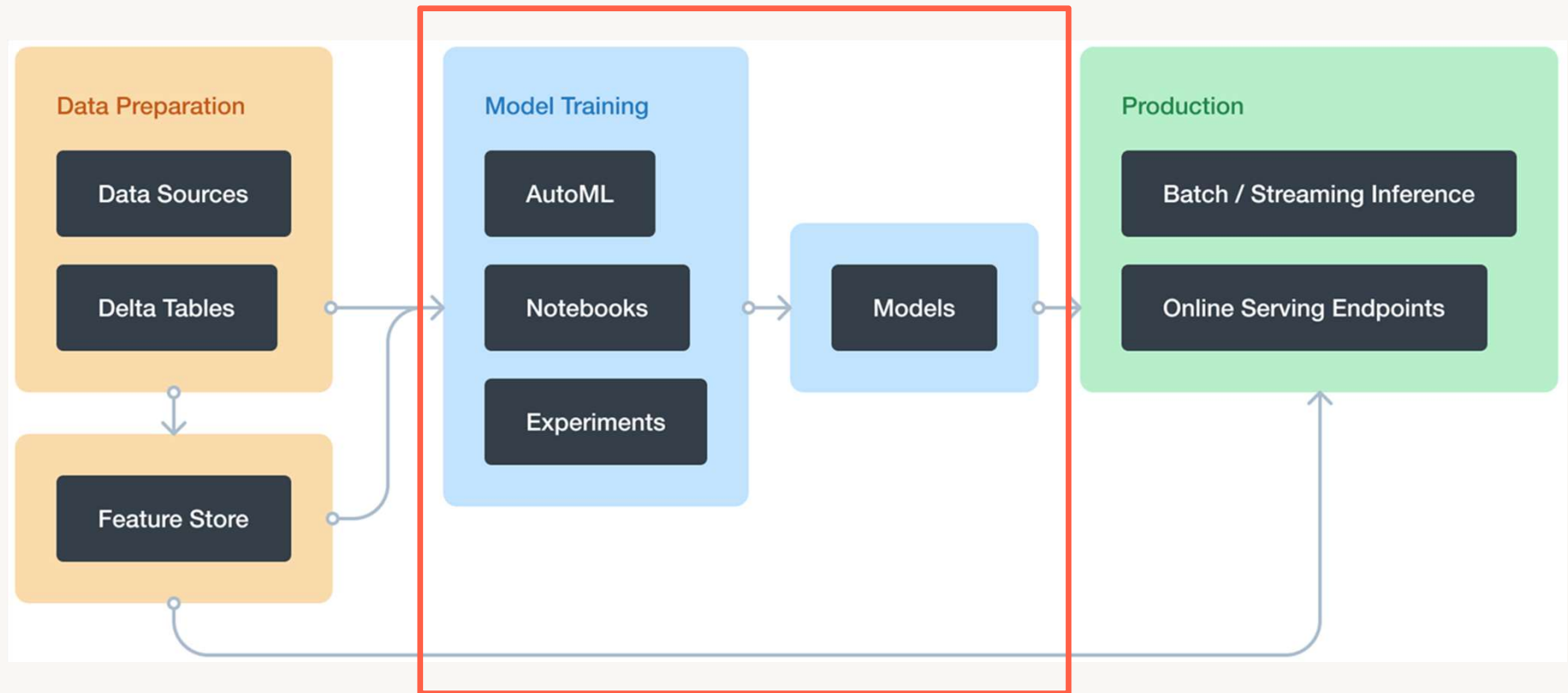
- Open storage format
- Built-in data versioning and governance
- Native access through PySpark, SQL, etc.



- Tracking and auto-logging
- Model Registry
- Model Serving

Demo: Use Feature Store

Model Training with Databricks ML





Data Science Workspace



AutoML



Data
Ingestion



Model
Training



Model
Tuning



Runtime and
Environment



Batch
Scoring



Online
Serving



Monitoring



Data
Versioning



Feature
Store



Batch (high
throughput)



Real Time
(low latency)



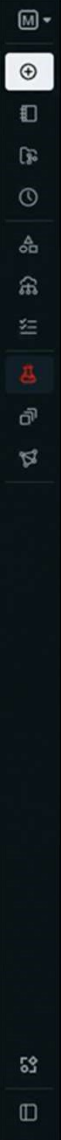
MLOps / Governance

Open Data LakeHouse Foundation with



DELTA LAKE





Experiments

Create AutoML Experiment ▼ Compare (0)

Owned by me Accessible by me

Search experiments Search

<input type="checkbox"/>	Name	Location	Last Modified	Created by	Notes	Actions
<input type="checkbox"/>	units_purchased_global_temp.automl_313a1986_d4f...	/Users/kevin.coyle@databricks.com/databricks_automl/units_purchased_global_tem...	2022-11-08 18:17:07 HST	kevin.coyle@databricks.com		⋮
<input type="checkbox"/>	DL 08 - Model Interpretability /	/Repos/Instructor-Led Source/deep-learning-with-databricks-source/Source/DL 08 - ...	2022-11-08 06:54:34 HST	jacob.parr@databricks.com		⋮
<input type="checkbox"/>	test-retail-prediction-kcoyle	/Users/kevin.coyle@databricks.com/databricks_automl/test-retail-prediction-kcoyle	2022-11-07 22:53:08 HST	kevin.coyle@databricks.com		⋮
<input type="checkbox"/>	02 - End to End with Model Serve /	/Repos/kevin.coyle@databricks.com/new-capability-model-serving/Source/02 - End ...	2022-11-06 20:37:55 HST	kevin.coyle@databricks.com		⋮
<input type="checkbox"/>	monitoring_prototype /	/Users/yinxi.zhang@databricks.com/monitoring/monitoring_prototype	2022-11-06 17:10:11 HST	yinxi.zhang@databricks.com		⋮
<input type="checkbox"/>	jacob.parr-k8e4-da-sml-459470233064954	/Curriculum/Test Results/jacob.parr-k8e4-da-sml-459470233064954	2022-11-03 09:19:56 HST	jacob.parr@databricks.com		⋮
<input type="checkbox"/>	jacob.parr-k8e4-da-sml-495646975864666	/Curriculum/Test Results/jacob.parr-k8e4-da-sml-495646975864666	2022-11-03 09:15:33 HST	jacob.parr@databricks.com		⋮
<input type="checkbox"/>	jacob.parr-k8e4-da-sml-877757080908534	/Curriculum/Test Results/jacob.parr-k8e4-da-sml-877757080908534	2022-11-03 09:14:31 HST	jacob.parr@databricks.com		⋮
<input type="checkbox"/>	price_global_temp.automl_ea5f6a08_6d98_44c7_83...	/Users/jacob.parr@databricks.com/databricks_automl/price_global_temp.automl_ea...	2022-11-03 09:14:15 HST	jacob.parr@databricks.com		⋮
<input type="checkbox"/>	jacob.parr-k8e4-da-sml-346127299743712	/Curriculum/Test Results/jacob.parr-k8e4-da-sml-346127299743712	2022-11-03 09:13:59 HST	jacob.parr@databricks.com		⋮

Required parameters:

- **dataset** - Input Spark or pandas DataFrame that contains training features and targets. If using a Spark DataFrame, it will convert it to a Pandas DataFrame under the hood by calling `.toPandas()` - just be careful you don't OOM!
- **target_col** - Column name of the target labels

We will also specify these optional parameters:

- **primary_metric** - Primary metric to select the best model. Each trial will compute several metrics, but this one determines which model is selected from all the trials. One of **r2** (default, R squared), **mse** (mean squared error), **rmse** (root mean squared error), **mae** (mean absolute error) for regression problems.
- **timeout_minutes** - The maximum time to wait for the AutoML trials to complete. **timeout_minutes=None** will run the trials without any timeout restrictions
- **max_trials** - The maximum number of trials to run. When **max_trials=None**, maximum number of trials will run to completion.

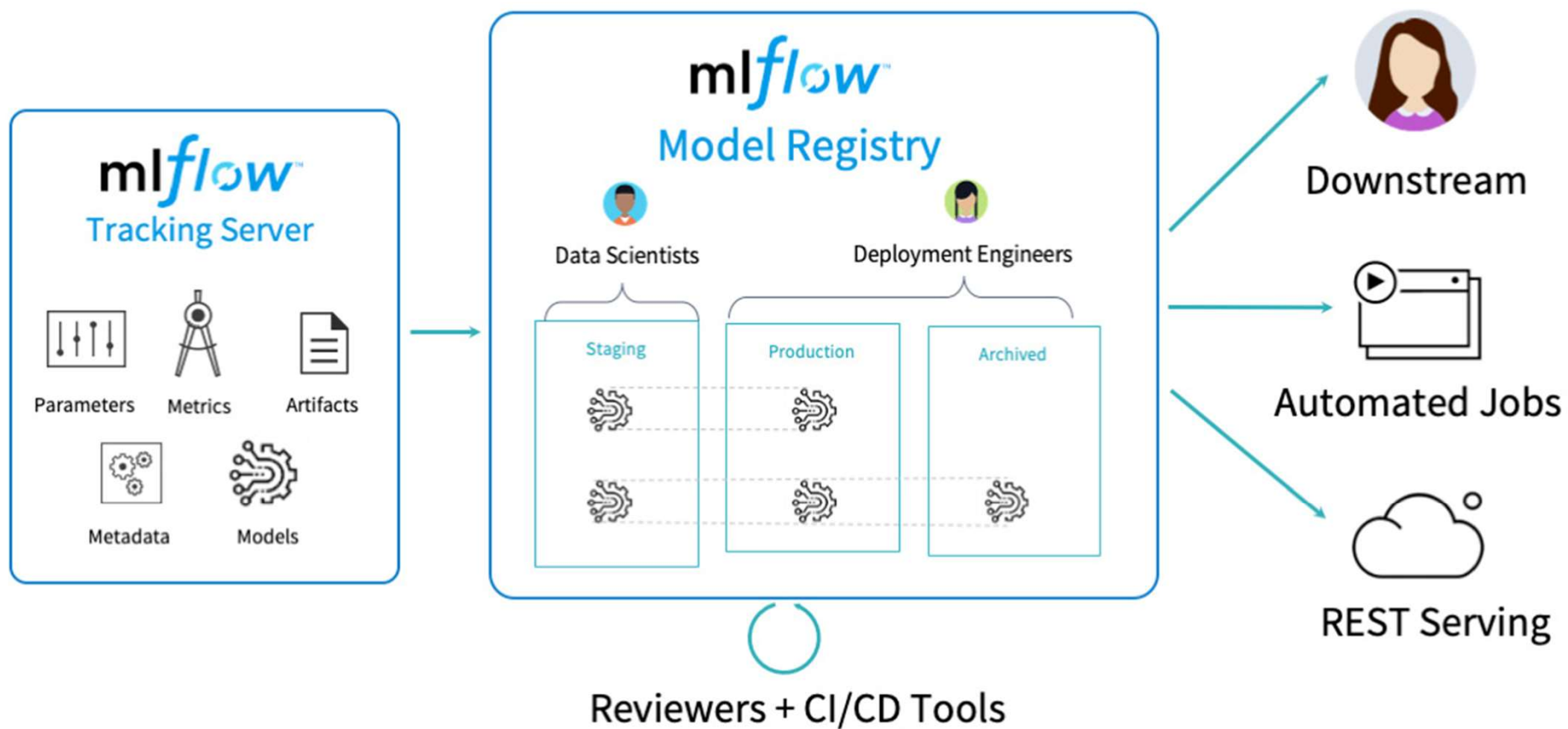
```
1 from databricks import automl
2
3 summary = automl.regress(train_df, target_col="price", primary_metric="rmse", timeout_minutes=5, max_trials=10)
```


Demo: Using AutoML

What is MLflow?



An open-source platform for
managing the end-to-end
machine learning lifecycle.



MLflow Tracking and Autologging

The screenshot displays the Databricks MLflow Tracking interface. The left sidebar shows the navigation menu with 'Experiments' selected. The main panel shows the details of an experiment named 'random_forest_regressor'. The experiment's Run ID is '8ccea8bc64504eb29bc0786b9d230d6e', and it was created on '2022-11-10 02:37:24'. The source is 'Notebook: RandomForestRegressor' and the user is 'kevin.coyle@databricks.com'. The experiment is in a 'FINISHED' state with a 'Lifecycle Stage' of 'active'. The duration is '10.8s'. The interface includes sections for 'Description', 'Parameters (78)', 'Metrics (15)', 'Tags (2)', and 'Artifacts'. The 'Artifacts' section is expanded, showing a list of files: 'MLmodel', 'conda.yaml', 'input_example.json', 'model.pkl', 'python_env.yaml', and 'requirements.txt'. The 'MLflow Model' section provides instructions on how to use the logged model, including a 'Model schema' table and 'Make Predictions' code snippets for Spark and Pandas DataFrames.

Experiments > /Users/kevin.coyle@databricks.com/databricks_automi/units_purchased_global_temp.automi_ef693b0a_7754_4161_8110_2430697c2db7_2022-35-10_07-35-16 >

random_forest_regressor [Provide Feedback](#)

[Reproduce Run](#)

Run ID: 8ccea8bc64504eb29bc0786b9d230d6e Date: 2022-11-10 02:37:24 Source: Notebook: RandomForestRegressor User: kevin.coyle@databricks.com

Duration: 10.8s Status: FINISHED Lifecycle Stage: active

> Description [Edit](#)

> Parameters (78)

> Metrics (15)

> Tags (2)

▼ Artifacts

model

- MLmodel
- conda.yaml
- input_example.json
- model.pkl
- python_env.yaml
- requirements.txt

Full Path: dbfs:/databricks/mlflow-tracking/3386347744162755/8ccea8bc64504eb29bc0786b9d230d6e/artifacts/model [Register Model](#)

MLflow Model

The code snippets below demonstrate how to make predictions using the logged model. You can also [register it to the model registry](#) to version control and deploy as a REST endpoint for real time serving.

Model schema

Input and output schema for your model. [Learn more](#)

Name	Type
Inputs (56)	
state_AK	double
state_AL	double
state_AR	double

Make Predictions

Predict on a Spark DataFrame:

```
import mlflow
from pyspark.sql.functions import struct, col
logged_model = 'runs:/8ccea8bc64504eb29bc0786b9d230d6e/model'

# Load model as a Spark UDF. Override result_type if the model does not return double value s.
loaded_model = mlflow.pyfunc.spark_udf(spark, model_uri=logged_model, result_type='double')

# Predict on a Spark DataFrame.
df.withColumn('predictions', loaded_model(struct(*map(col, df.columns))))
```

Predict on a Pandas DataFrame:

MLflow Models

MLflow Model

The code snippets below demonstrate how to make predictions using the logged model. You can also [register it to the model registry](#) to version control and deploy as a REST endpoint for [real time serving](#).

Model schema

Input and output schema for your model. [Learn more](#)

Name	Type
Inputs (56)	
state_AK	double
state_AL	double
state_AR	double
state_AZ	double
state_CA	double
Outputs (1)	
-	Tensor (dtype: float64, shape: ...)

Make Predictions

Predict on a Spark DataFrame:

```
import mlflow
from pyspark.sql.functions import struct, col
logged_model = 'runs:/8ccea8bc64504eb29bc0786b9d230d6e/model'

# Load model as a Spark UDF. Override result_type if the model does not return double values.
loaded_model = mlflow.pyfunc.spark_udf(spark, model_uri=logged_model, result_type='double')

# Predict on a Spark DataFrame.
df.withColumn('predictions', loaded_model(struct(*map(col, df.columns))))
```

Predict on a Pandas DataFrame:

```
import mlflow
logged_model = 'runs:/8ccea8bc64504eb29bc0786b9d230d6e/model'

# Load model as a PyFuncModel.
loaded_model = mlflow.pyfunc.load_model(logged_model)

# Predict on a Pandas DataFrame.
import pandas as pd
loaded_model.predict(pd.DataFrame(data))
```

MLflow Model Registry

The screenshot displays the Databricks MLflow Model Registry interface. The top navigation bar includes the Databricks logo, a search bar, and a user profile icon. The left sidebar shows the 'Machine Learning' section with various options: New, Workspace, Repos, Recents, Data, Compute, Workflows, Experiments, Feature Store, Models (highlighted), Partner Connect, and Menu options. The main content area is titled 'Registered Models' and features a 'Create Model' button. Below this, a list of registered models is shown, with the first model being '-rf-model_5ff8f5'. The detailed view of this model is displayed, showing its 'Details' and 'Serving' tabs. The 'Details' tab is active, showing the model's name, a 'Permissions' button, and a 'Use model for inference' button. The 'Serving' tab shows the model's status, including 'Created Time', 'Last Modified', and 'Creator'. The 'Versions' section lists two versions: 'Version 2' (created 2022-09-06 10:02:44) and 'Version 1' (created 2022-09-06 09:59:59). Both versions are in the 'Production' stage. The table below shows the details of the registered models.

Version	Registered at	Created by	Stage	Pending Requests	Description
Version 2	2022-09-06 10:02:44		Production	—	This model version is a randon...
Version 1	2022-09-06 09:59:59		Production	—	This model version was built ...

MLflow Model Serving

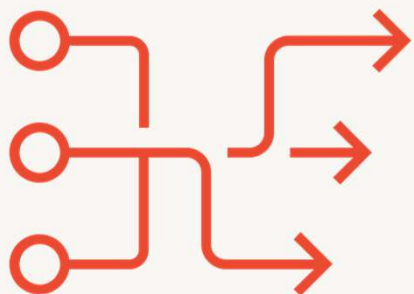
The screenshot shows the Databricks MLflow Model Serving interface. The top navigation bar includes the Databricks logo, a search bar, and a 'Curriculum Dev' dropdown. The left sidebar contains a 'Machine Learning' dropdown menu with options for 'New', 'Workspace', 'Repos', and 'Recents'. The main content area displays the 'Registered Models' page for a model named '-rf-model_5ff8f5'. The model name is partially obscured by a grey box. To the right of the model name are 'Permissions' and 'Use model for inference' buttons. Below the model name, there are tabs for 'Details' and 'Serving'. The 'Details' tab is active, showing a 'Notify me about' section with a dropdown menu set to 'Activity on versions I follow'. Below this, the 'Created Time' is '2022-09-06 09:59:59', the 'Last Modified' time is '2022-09-06 10:04:49', and the 'Creator' is listed as 'Curriculum Dev'.

The screenshot shows the Databricks MLflow Model Serving interface, specifically the 'Serving' tab for a model named '-model_5ff8f5'. The model name is partially obscured by a grey box. To the right of the model name are 'Permissions' and 'Use model for inference' buttons. Below the model name, there are tabs for 'Details' and 'Serving'. The 'Serving' tab is active, showing a message: 'Enable real-time model serving behind a REST API interface. This will launch a single-node cluster that will host all active versions of this model. [Learn more.](#)' Below this message is a blue button labeled 'Enable Serving'.

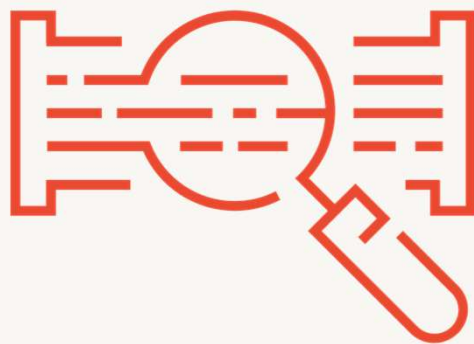
Deploy a model for batch inference

Deploying Models to Production

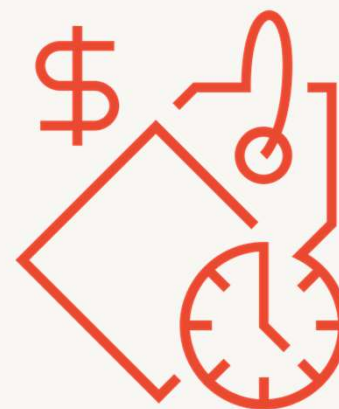
Batch



Streaming




Real-time serving





Questions?





To learn more about Databricks
Machine Learning, check out the
following resources:

Databricks Academy
Databricks Community