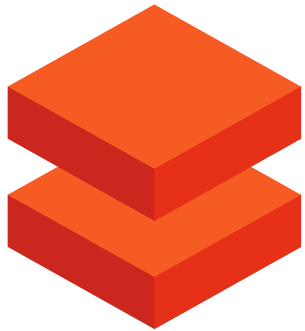
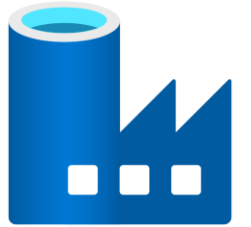


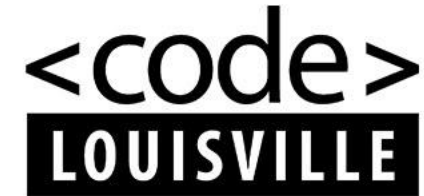
# Predicting Flights with Azure Databricks



Presented by Sarah Dutkiewicz  
Microsoft MVP, Developer Technologies  
Cleveland Tech Consulting, LLC



# Thank You to the Code PaLOUsa Sponsors



## Friends of Code PaLOUsa



Couchbase



mongoDB



redis



# Agenda

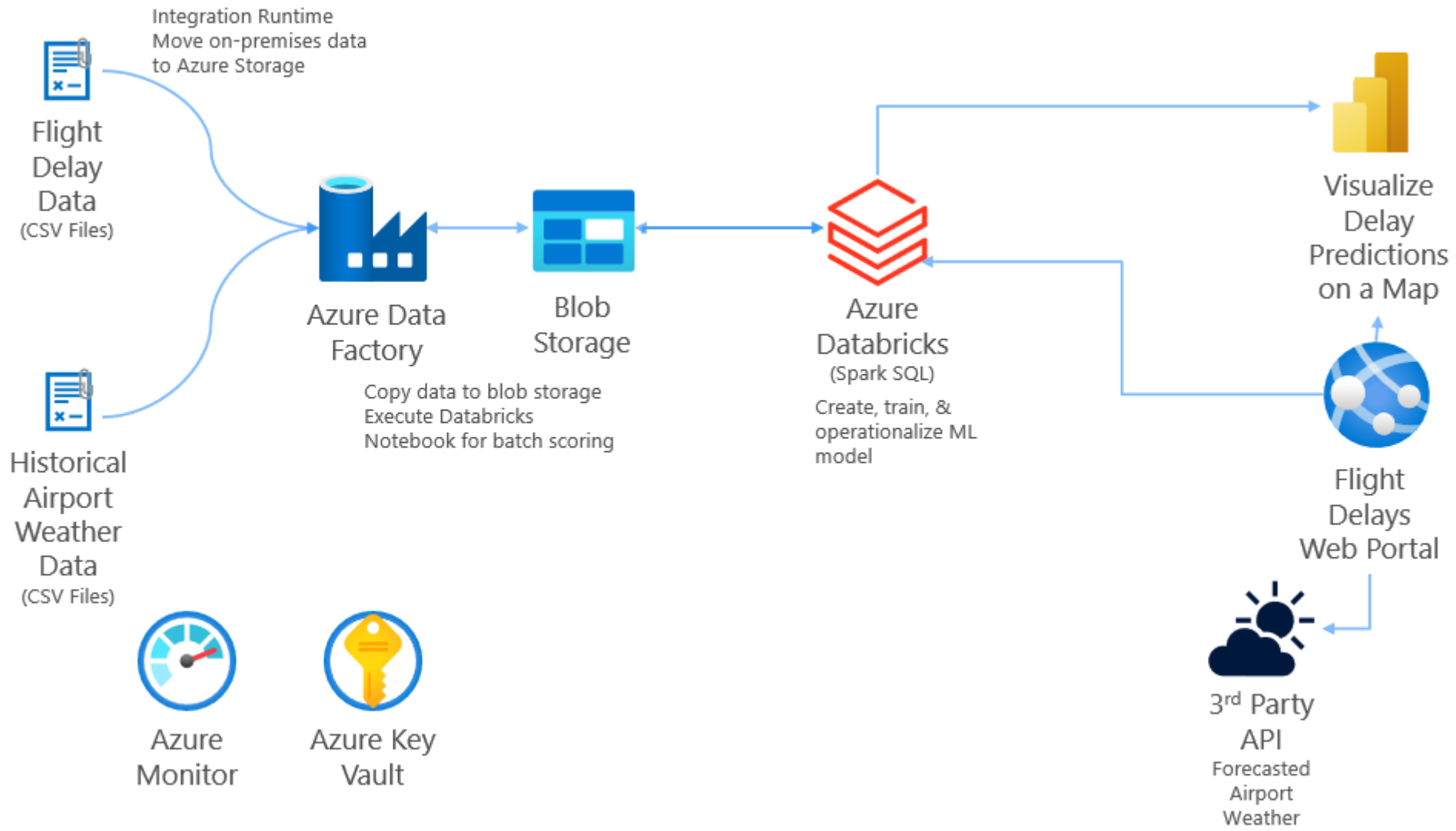
- Introducing the flight prediction scenario
- What is Databricks?
- Exploring Databricks while getting to the solution

# Scenario

- Margie's Travels – a *fictitious* concierge for business travelers - wants to enable their agents to **enter in the flight information** and **produce a prediction** as to **whether the departing flight will encounter a 15-minute or longer delay, considering the weather forecast** for the departure hour.
- Data details:
  - Sample data from 2013
  - 2.7 million flight delays with airport codes - **examples**
  - 20 columns – **features**

# Training the model

- Using Decision Tree algorithm (**binary classification**) from Spark MLlib
- Part of the historical data (2013) is used for training and another part for test
- Flights are delayed have **DepDel15** value of 1
  - [OST R | BTS | Transtats](#) – Departure Delay Indicator of 15 minutes or more
- Sample keeps all delayed and a downsample of 30% not delayed – **stratified sampling**
- One-Hot encoded categorical variables and use the Pipeline API
- 3-fold cross validation
- Save the model for use in other notebooks and saved in case cluster restarts



Source: [Microsoft Cloud Workshop: Big Data Analytics and Visualization](#), Hands-on Lab

# What is Databricks?

- Web-based analytics platform with 3 workloads:
  - **Databricks SQL** – for querying data lakes with SQL
  - **Databricks Data Science & Engineering** – for data engineers, data scientists, and ML engineers for data ingestion and analysis using the Apache Spark Ecosystem. This is the classic Databricks environment.
  - **Databricks Machine Learning** – experiment tracking, model training, feature development and management



Loading the Data



# Data to Load

Data	CSV Name	Databricks Table Name
Flight delays with airport codes	FlightDelaysWithAirportCodes.csv	flight_delays_with_airport_codes
Flight weather with airport code	FlightWeatherWithAirportCode.csv	flight_weather_with_airport_code
Airport code location lookup	AirportCodeLocationLookupClean.csv	airport_code_location_lookup_clean

# Creating a Cluster

- Clusters can be set for high concurrency, single node, or standard
- Runtimes include:
  - Standard or ML
  - ML runtimes include GPU or non-GPU
- Photon acceleration can be supported in cases
- Autoscaling is supported
- Terminate due to inactivity
- Workers and drivers support:
  - General usage
  - Compute optimized
  - Memory optimized
  - Storage optimized
  - GPU accelerated
- Integration with Azure Data Lake Storage
- Spark config and environment variables
- Init scripts
- ... and more

The screenshot shows the 'New Cluster' configuration page in the Microsoft Azure Databricks portal. The page is titled 'New Cluster' and includes a 'Cancel' button and a 'Create Cluster' button. The 'DBU / hour' is set to 1. The 'Databricks runtime version' is set to 'Runtime: 10.4 LTS ML (Scala 2.12, Spark 3.2.1)'. The 'Use Photon Acceleration' checkbox is unchecked. The 'Autopilot options' section includes 'Enable autoscaling' (unchecked) and 'Terminate after 120 minutes of inactivity' (checked). The 'Worker type' is set to 'Standard\_F4' with '8 GB Memory, 4 Cores'. The 'Workers' count is set to 1. The 'Spot instances' checkbox is unchecked. The 'Driver type' is set to 'Same as worker' with '8 GB Memory, 4 Cores'. The 'DBU / hour' for the driver is set to 1. The 'Advanced options' section includes 'Azure Data Lake Storage credential passthrough' (unchecked) and 'Enable credential passthrough for user-level data access' (unchecked). The 'Spark' tab is selected, showing 'Spark config' and 'Environment variables' sections. The 'Spark config' section contains a text area for Spark configuration options, with an example: 'spark.speculation true' and 'spark.kryo.registrator my.package.MyRegistrator'. The 'Environment variables' section contains a text area for environment variables, with examples: 'SPARK\_NICENESS=0', 'JAVA\_OPTS=-D... -D... -XX:MaxPermSize=256m ...', and 'JAVA\_OPTS=\$JAVA\_OPTS -D...'.

Microsoft Azure | Databricks Portal

Clusters / New Compute

New Cluster Cancel Create Cluster DBU / hour: 1 1 Worker: 8 GB Memory, 4 Cores  
1 Driver: 8 GB Memory, 4 Cores

UI | JSON

Databricks runtime version ②

Runtime: 10.4 LTS ML (Scala 2.12, Spark 3.2.1) | v

☐ Use Photon Acceleration ②

Autopilot options

☐ Enable autoscaling ②

☒ Terminate after 120 minutes of inactivity ②

Worker type ② Workers

Standard\_F4 8 GB Memory, 4 Cores | v 1 ☐ Spot instances ②

Driver type

Same as worker 8 GB Memory, 4 Cores | v

DBU / hour: 1 ② Standard\_F4

Advanced options

Azure Data Lake Storage credential passthrough ②

☐ Enable credential passthrough for user-level data access

Spark Tags Logging Init Scripts

Spark config ②

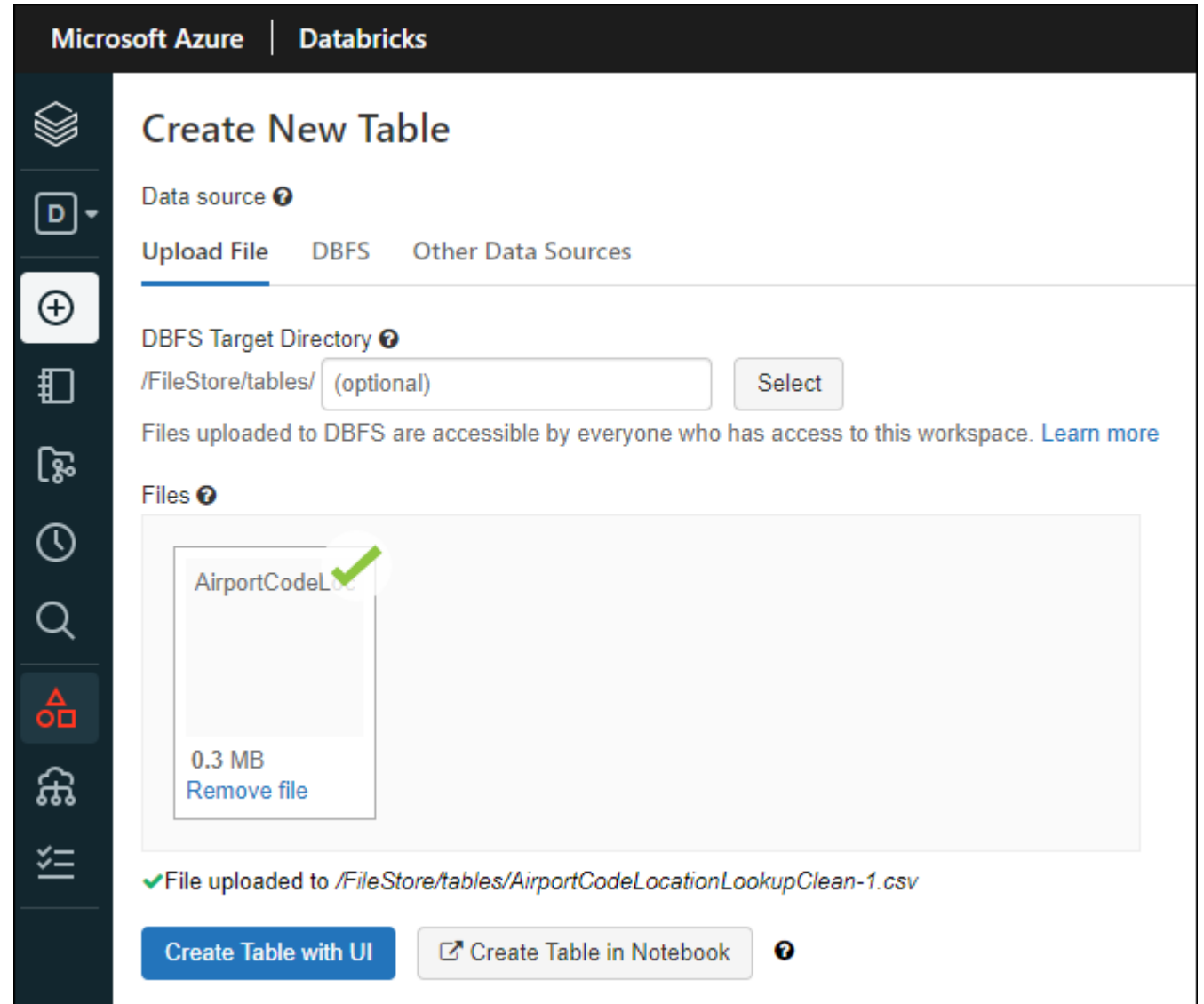
Enter your Spark configuration options here. Provide only one key-value pair per line.  
Example:  
spark.speculation true  
spark.kryo.registrator my.package.MyRegistrator

Environment variables ②

SPARK\_NICENESS=0  
JAVA\_OPTS=-D... -D... -XX:MaxPermSize=256m ...  
JAVA\_OPTS=\$JAVA\_OPTS -D...

# Loading the Data into Azure Databricks

- Done in the Data Science and Engineering load
- Uses a cluster for loading data into Azure Databricks
- Tables have 2 different types:
  - Global tables - accessible across all clusters
  - Local tables - available only within one cluster



The screenshot shows the 'Create New Table' interface in the Microsoft Azure Databricks portal. The interface is divided into a left sidebar with navigation icons and a main content area. The main content area has a dark header with 'Microsoft Azure | Databricks'. Below the header, the title 'Create New Table' is displayed. Underneath, there are tabs for 'Data source', 'Upload File', 'DBFS', and 'Other Data Sources'. The 'Upload File' tab is selected. Below the tabs, there is a section for 'DBFS Target Directory' with a text input field containing '/FileStore/tables/' and a 'Select' button. A note states: 'Files uploaded to DBFS are accessible by everyone who has access to this workspace. [Learn more](#)'. Below this, there is a 'Files' section with a file preview card for 'AirportCodeLoc...'. The card shows a green checkmark, the file name, and its size '0.3 MB', with a 'Remove file' link. At the bottom, a green message states: 'File uploaded to /FileStore/tables/AirportCodeLocationLookupClean-1.csv'. At the very bottom, there are two buttons: 'Create Table with UI' and 'Create Table in Notebook'.

Microsoft Azure | Databricks

## Create New Table

Data source ⓘ

Upload File DBFS Other Data Sources

DBFS Target Directory ⓘ

/FileStore/tables/ (optional) Select

Files uploaded to DBFS are accessible by everyone who has access to this workspace. [Learn more](#)

Files ⓘ

AirportCodeLoc... ✓

0.3 MB  
[Remove file](#)


✓ File uploaded to /FileStore/tables/AirportCodeLocationLookupClean-1.csv


Create Table with UI Create Table in Notebook ⓘ

# Creating a Table

### Select a Cluster to Preview the Table

Choose a cluster with which you will read and preview the data.


Cluster 

lab | 


Preview Table


### Specify Table Attributes


Specify the Table Name, Database and Schema to add this to the data UI for other users to access


Table Name 


airportcodeolocationlookupcl

Create in Database 


default 


File Type 


CSV 

Column Delimiter 

,

☒ First row is header 






☐ Infer schema 

☐ Multi-line 

Create Table

Create Table in Notebook

Table Preview

AIRPORT_ID	AIRPORT	DISPLAY_AIRPORT_NAM	LATITUDE	LONGITUDE
STRING 	STRING 	STRING 	STRING 	STRING 
10001	01A	Afognak Lake Airport	58.10944444	-152.9066667
10003	03A	Bear Creek Mining Strip	65.54805556	-161.0716667
10004	04A	Lik Mining Camp	68.08333333	-163.1666667
10005	05A	Little Squaw Airport	67.57	-148.1838889
10006	06A	Kizhuyak Bay	57.74527778	-152.8827778
10007	07A	Klawock Seaplane Base	55.55472222	-133.1016667
10008	08A	Elizabeth Island Airport	59.15694444	-151.8291667

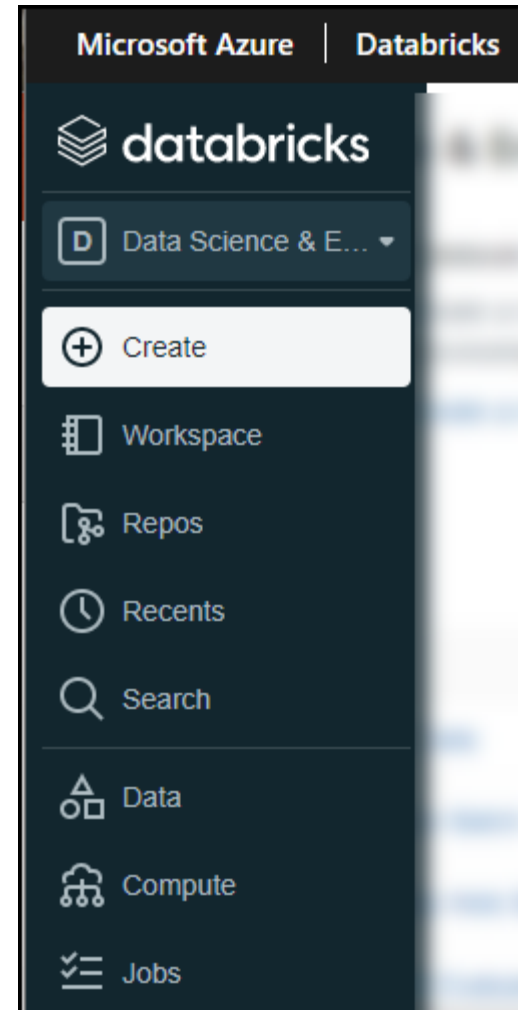
# Databricks

# Data Science & Engineering

Classic Databricks

# Databricks Data Science & Engineering

- Classic Databricks environment
- Backbone for the ML environment
- Key components:
  - Workspaces
  - Runtimes
  - Clusters
  - Notebooks
  - Jobs

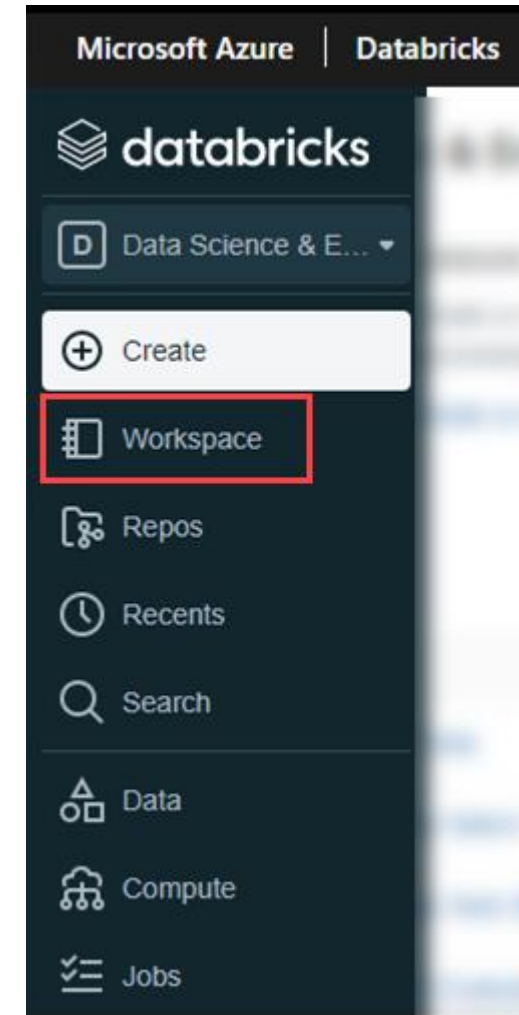


# Preparing our data

- This is all in a **notebook** labeled *01 Data Preparation*.
- Preparation steps:
  1. Explore the data.
  2. Munge the data for flight delays with airport codes with R.
  3. Export the prepared data to a **global table**.
  4. Prepare the weather data with Python.
  5. Join the flight and weather datasets using Spark SQL.
  6. Store the flight delays with weather in a **global table**.

# Workspaces

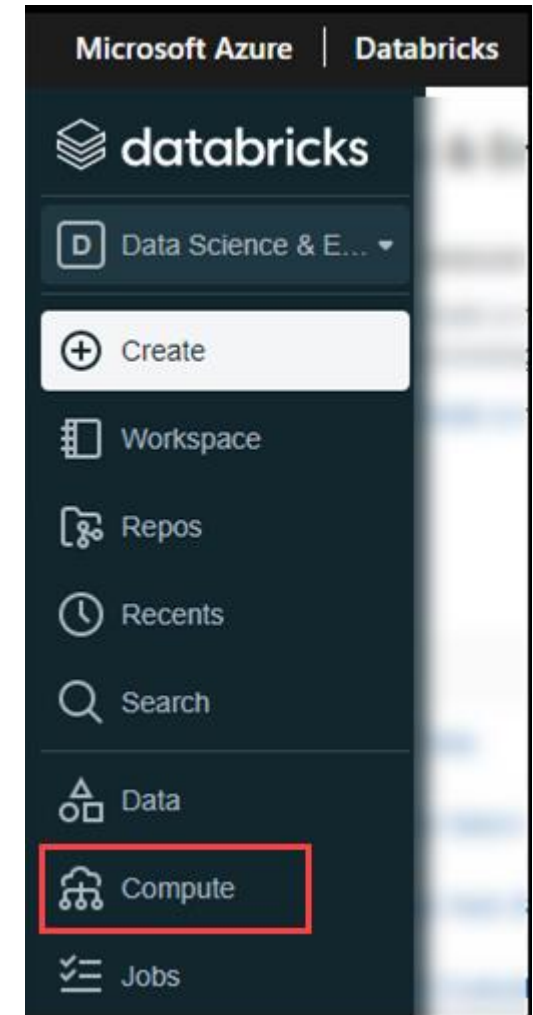
- Environment for all Azure Databricks assets
- Organizes notebooks, libraries, and experiments into folders
  - **Notebooks** – runnable code + markdown
  - **Libraries** – third party resources or locally built code accessible to clusters
  - **Experiments** – MLflow machine learning model activities
- Provides access to clusters and jobs
- Integrates with Git through Repos





# Clusters

- Powerhouse for Azure Databricks
  - Compute and configuration
  - Supports workloads for:
    - Data engineering
    - Data science
    - Data analytics
  - Takes time to start
- Two types
  - **All-purpose** – can be shared for collaborative works; manually managed
  - **Job clusters** – used for jobs, created and terminated by the Azure Databricks job scheduler; cannot be restarted





Clusters / lab

lab



Edit

Permissions

Start

Clone

Delete

Configuration

Notebooks (0)

Libraries

Event log

Spark UI

Driver logs

Metrics

Apps

Spark cluster UI - Master ▼

Policy ?

Unrestricted

Cluster mode ?

Standard ▼

Databricks Runtime Version

9.1 LTS ML (includes Apache Spark 3.1.2, Scala 2.12)

Autopilot options

☐ Enable autoscaling ?☒ Terminate after 120 minutes of inactivity ?

Worker type ?

Standard\_F4

8 GB Memory, 4 Cores

Workers

1

☐ Spot instances ?

Driver type

Standard\_F4

8 GB Memory, 4 Cores

DBU / hour: 1 ?

Standard\_F4

▼ Advanced options

Azure Data Lake Storage credential passthrough ?

☐ Enable credential passthrough for user-level data access

Spark

Tags

Logging

Init Scripts

JDBC/ODBC

Permissions

Spark config ?

spark.hadoop.fs.azure.account.key[redacted].blob.core.windows.net

Environment variables ?

No environment variables

UI | [JSON](#)

# Runtimes

- Assigned at the cluster-level
- Provides the engine for the platform, based on Apache Spark
- Includes Delta Lake for storage
- GPU-enabled support available
- Ubuntu and system libraries
- Supports the following languages:
  - Python
  - R
  - Java
  - Scala

# Special Runtimes

- Databricks Runtime for Machine Learning
- Databricks Light
- Photon-enabled runtimes
  - Uses a native vectorized query engine
  - Currently in Public Preview
  - Works in both Azure Databricks clusters and Databricks SQL endpoints

# Notebooks

- Can mix Markdown and languages to present data
- Example: Data Preparation notebook with:
  - Markdown
  - SQL
  - Python
  - R

Microsoft Azure | Databricks

01 Data Preparation Python

lab | File | Edit | View: Standard | Run All | Clear

Cmd 4

First, let's execute the below command to make sure all three tables were created. You should see an output like the following:

database	tableName	isTemporary
default	airport_code_loca...	false
default	flight_delays_wit...	false
default	flight_weather_wi...	false

Cmd 5

```
1 spark.sql("show tables").show()
```

```
+-----+-----+-----+
|database|      tableName|isTemporary|
+-----+-----+-----+
| default|airport_code_loca...|      false|
| default|flight_delays_wit...|      false|
| default|flight_weather_wi...|      false|
+-----+-----+-----+
```

Command took 0.17 seconds -- by sarah@cletechconsulting.com at 7/29/2022, 3:59:54 PM on lab

Cmd 6

Now execute a SQL query using the %sql magic to select all columns from flight\_delays\_with\_airport\_codes. By default, only the first 1,000 rows will be returned.

Cmd 7

```
1 %sql
2 select * from flight_delays_with_airport_codes
```

▶ (1) Spark Jobs

Table Data Profile

	Year	Month	DayofMonth	DayOfWeek	Carrier	CRSDepTime	DepDelay	DepDel15	CRSArrTime	ArrDelay	ArrDel15	Cancelled	OriginAirportCode
1	2013	4	19	5	DL	837	-3	0	1138	1	0	0	DTW
2	2013	4	19	5	DL	1705	0	0	2336	-8	0	0	SLC
3	2013	4	19	5	DL	600	-4	0	851	-15	0	0	PDX
4	2013	4	19	5	DL	1630	28	1	1903	24	1	0	STL
5	2013	4	19	5	DL	1615	-6	0	1805	-11	0	0	CVG
6	2013	4	19	5	DL	1726	-1	0	1818	-19	0	0	ATL
7	2013	4	19	5	DL	1900	0	0	2133	-1	0	0	STI

Truncated results, showing first 1000 rows.  
[Click to re-execute with maximum result limits.](#)

Command took 0.98 seconds -- by sarah@cletechconsulting.com at 7/29/2022, 3:59:55 PM on lab

# Notebook cells

- Command number helps for execution and navigation
- Execution details include:
  - Duration
  - User
  - Timestamp
  - Cluster name
- Example shows Python command and output

```
Cmd 5
1 spark.sql("show tables").show()

+-----+-----+-----+
|database|      tableName|isTemporary|
+-----+-----+-----+
| default|airport_code_loca...|      false|
| default|flight_delays_wit...|      false|
| default|flight_weather_wi...|      false|
+-----+-----+-----+

Command took 0.17 seconds -- by sarah@cletechconsulting.com at 7/29/2022, 3:59:54 PM on lab
```

# SQL notebook cell

- Using the magic commands – start with % - to indicate using SQL
- Other magic commands include:
  - %fs
  - %python
  - %md
  - %r
  - %scala
  - %sh

Cmd 7

```
1 %sql
2 select * from flight_delays_with_airport_codes
```

▶ (1) Spark Jobs

Table Data Profile

	Year	Month	DayofMonth	DayOfWeek	Carrier	CRSDepTime
1	2013	4	19	5	DL	837
2	2013	4	19	5	DL	1705
3	2013	4	19	5	DL	600
4	2013	4	19	5	DL	1630
5	2013	4	19	5	DL	1615
6	2013	4	19	5	DL	1726
7	2013	4	19	5	DL	1900

Truncated results, showing first 1000 rows.  
[Click to re-execute with maximum result limits.](#)

Command took 0.98 seconds -- by sarah@cletechconsulting.com at 7/29/2022, 3:59:55 PM on lab

# DataFrame schema

- dfFlightDelays is a DataFrame
- Python code, using pretty print pprint library

```
Cmd 19

1 pprint.pprint(dfFlightDelays.dtypes)

[('Year', 'string'),
 ('Month', 'string'),
 ('DayofMonth', 'string'),
 ('DayOfWeek', 'string'),
 ('Carrier', 'string'),
 ('CRSDepTime', 'string'),
 ('DepDelay', 'string'),
 ('DepDel15', 'string'),
 ('CRSArrTime', 'string'),
 ('ArrDelay', 'string'),
 ('ArrDel15', 'string'),
 ('Cancelled', 'string'),
 ('OriginAirportCode', 'string'),
 ('OriginAirportName', 'string'),
 ('OriginLatitude', 'string'),
 ('OriginLongitude', 'string'),
 ('DestAirportCode', 'string'),
 ('DestAirportName', 'string'),
 ('DestLatitude', 'string'),
 ('DestLongitude', 'string')]

Command took 0.04 seconds -- by sarah@cletechconsulting.com at 7/29/2022, 3:59:55 PM on lab
```



# Data munging

- Using SparkR to clean
- Yes... R in the same notebook as SQL... labeled as a Python notebook

Cmd 24

```
1 %r
2 library(SparkR)
3
4 # Select only the columns we need, casting CRSDepTime as long and DepDel15 as int, into a new DataFrame
5 dfflights <- sql("SELECT OriginAirportCode, OriginLatitude, OriginLongitude, Month, DayofMonth,
6 cast(CRSDepTime as long) CRSDepTime, DayOfWeek, Carrier, DestAirportCode, DestLatitude, DestLongitude,
7 cast(DepDel15 as int) DepDel15 from flight_delays_with_airport_codes")
8
9 # Delete rows containing missing values
10 dfflights <- na.omit(dfflights)
11
12 # Round departure times down to the nearest hour, and export the result as a new column named
13 "CRSDepHour"
14 dfflights$CRSDepHour <- floor(dfflights$CRSDepTime / 100)
15
16 # Trim the columns to only those we will use for the predictive model
17 dfflightsClean = dfflights[, c("OriginAirportCode", "OriginLatitude", "OriginLongitude", "Month",
18 "DayofMonth", "CRSDepHour", "DayOfWeek", "Carrier", "DestAirportCode", "DestLatitude", "DestLongitude",
19 "DepDel15")]
20
21 createOrReplaceTempView(dfflightsClean, "flight_delays_view")
```

Attaching package: 'SparkR'

The following object is masked \_by\_ '.GlobalEnv':

setLocalProperty

The following objects are masked from 'package:stats':

cov, filter, lag, na.omit, predict, sd, var, window

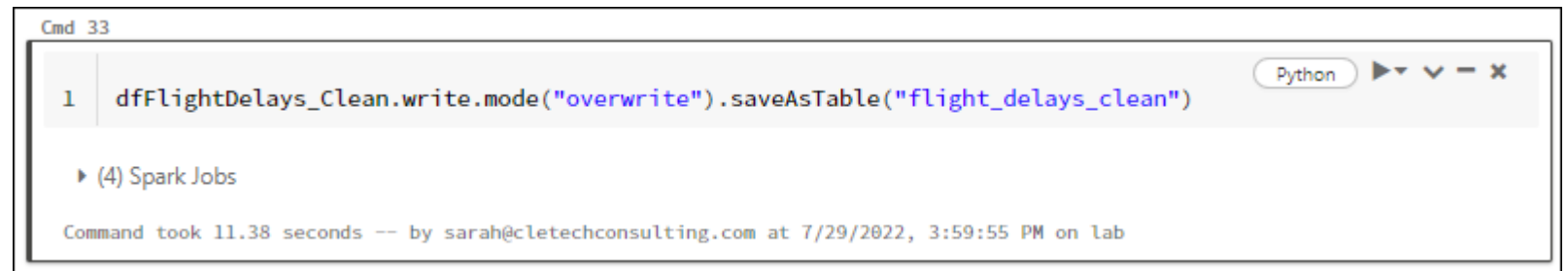
The following objects are masked from 'package:base':

as.data.frame, colnames, colnames<-, drop, endsWith, intersect,  
rank, rbind, sample, startsWith, subset, summary, transform, union

Command took 1.29 seconds -- by sarah@cletechconsulting.com at 7/29/2022, 3:59:55 PM on lab

# Export to a Databricks table

- Storing munged data into another table
- `saveAsTable()` – creates **global** table
- `createOrReplaceTempView()` and `registerTempTable()` create **local** tables

A screenshot of a Databricks Command Shell window titled 'Cmd 33'. The window has a light gray header bar with a 'Python' button and standard window controls (play, close, maximize, minimize). The main area shows a single line of Python code: `1 dfFlightDelays_Clean.write.mode("overwrite").saveAsTable("flight_delays_clean")`. Below the code, there is a status bar indicating '(4) Spark Jobs' and a message: 'Command took 11.38 seconds -- by sarah@cletechconsulting.com at 7/29/2022, 3:59:55 PM on lab'.

```
Cmd 33
1 dfFlightDelays_Clean.write.mode("overwrite").saveAsTable("flight_delays_clean")

▶ (4) Spark Jobs

Command took 11.38 seconds -- by sarah@cletechconsulting.com at 7/29/2022, 3:59:55 PM on lab
```

# Cleaning weather data with Python

- WindSpeed: Replace missing values with 0.0, and "M" values with 0.005
- HourlyPrecip: Replace missing values with 0.0, and "T" values with 0.005
- SeaLevelPressure: Replace "M" values with 29.92 (the average pressure)
- Convert WindSpeed, HourlyPrecip, and SeaLevelPressure to numeric columns
- Round "Time" column down to the nearest hour, and add value to a new column named "Hour"
- Eliminate unneeded columns from the dataset

Cmd 56

```
1  # Round Time down to the next hour, since that is the hour for which we want to use flight data. Then,
2  # add the rounded Time to a new column named "Hour", and append that column to the dfWeather DataFrame.
3  df = dfWeather.withColumn('Hour', F.floor(dfWeather['Time']/100))
4
5  # Replace any missing HourlyPrecip and WindSpeed values with 0.0
6  df = df.fillna('0.0', subset=['HourlyPrecip', 'WindSpeed'])
7
8  # Replace any WindSpeed values of "M" with 0.005
9  df = df.replace('M', '0.005', 'WindSpeed')
10
11 # Replace any SeaLevelPressure values of "M" with 29.92 (the average pressure)
12 df = df.replace('M', '29.92', 'SeaLevelPressure')
13
14 # Replace any HourlyPrecip values of "T" (trace) with 0.005
15 df = df.replace('T', '0.005', 'HourlyPrecip')
16
17 # Be sure to convert WindSpeed, SeaLevelPressure, and HourlyPrecip columns to float
18 # Define a new DataFrame that includes just the columns being used by the model, including the new Hour
19 # feature
20 dfWeather_Clean = df.select('AirportCode', 'Month', 'Day', 'Hour', df['WindSpeed'].cast('float'),
21                             df['SeaLevelPressure'].cast('float'), df['HourlyPrecip'].cast('float'))
```

df: pyspark.sql.dataframe.DataFrame = [AirportCode: string, Month: integer ... 6 more fields]

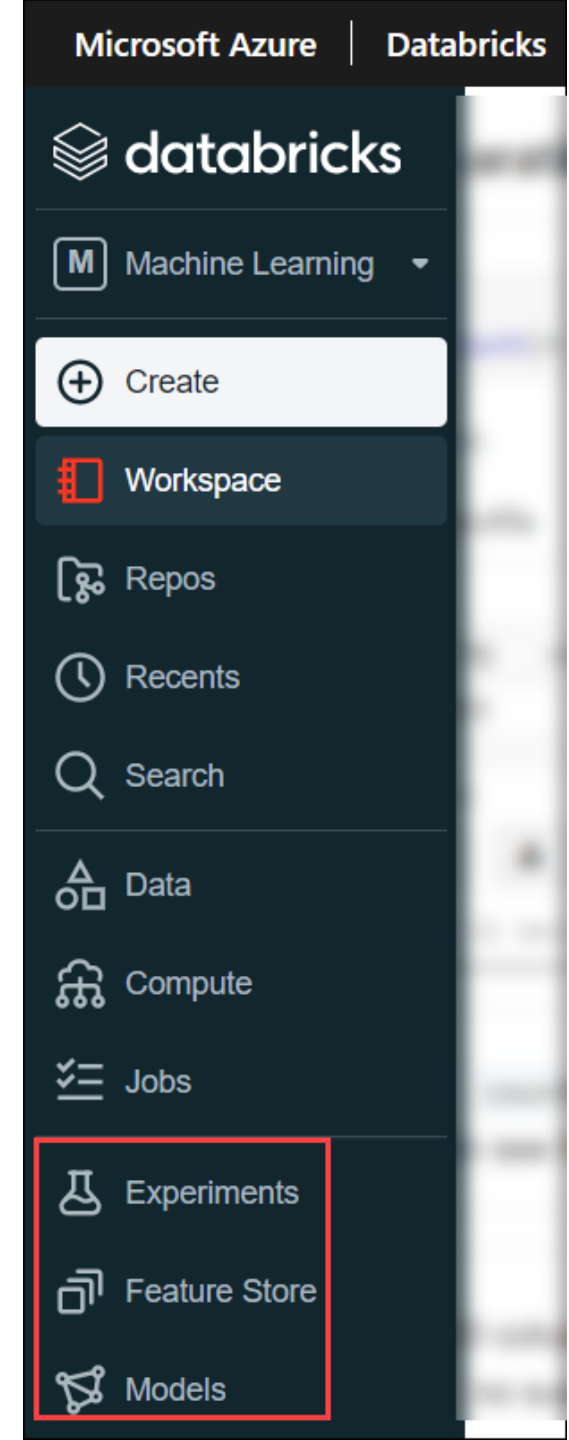
dfWeather\_Clean: pyspark.sql.dataframe.DataFrame = [AirportCode: string, Month: integer ... 5 more fields]

Command took 0.14 seconds -- by sarah@cletechconsulting.com at 7/29/2022, 3:59:56 PM on lab

Databricks Machine Learning

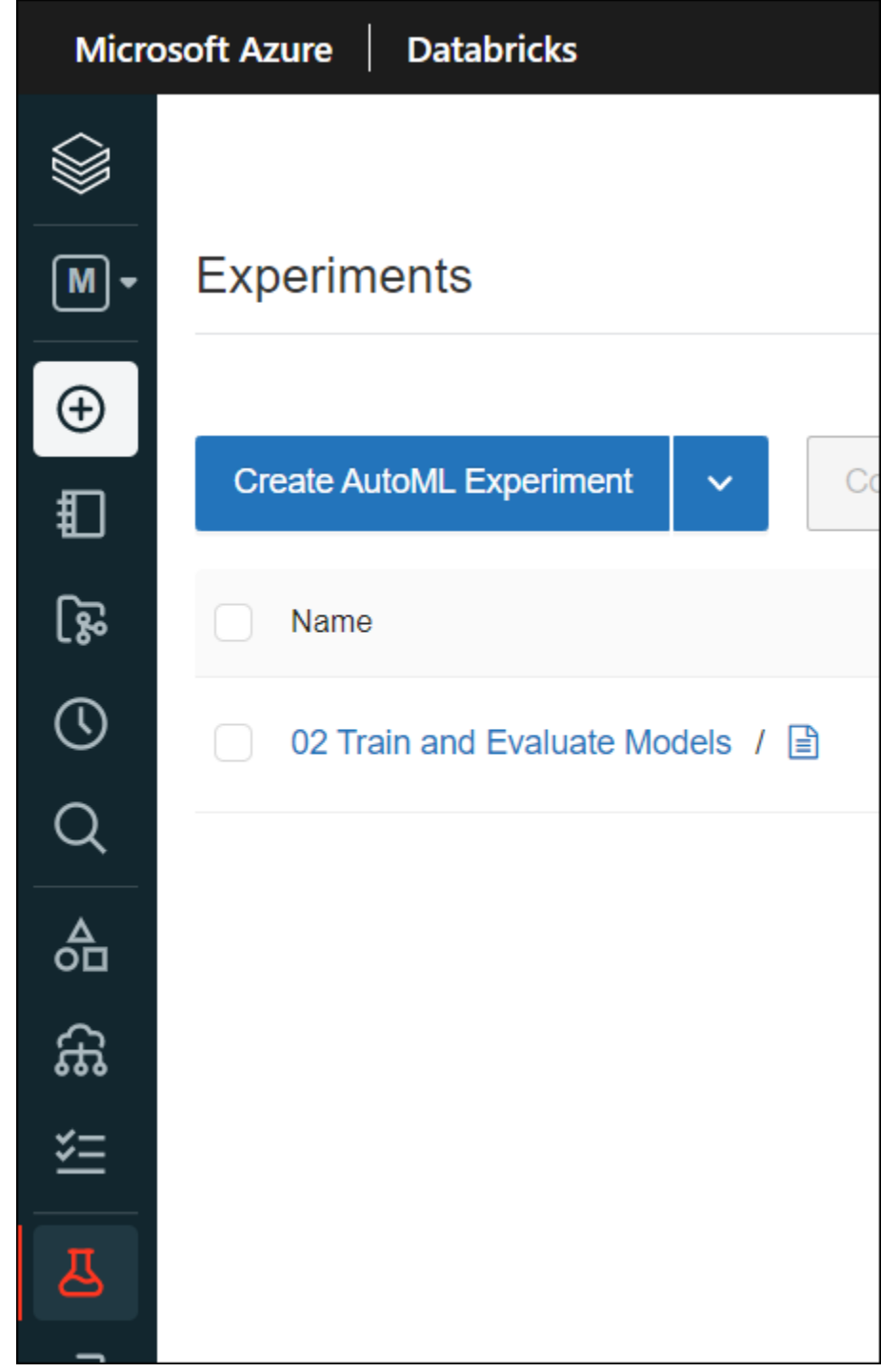
# Databricks Machine Learning

- Builds on top of Data Science & Engineering
- Same workspace components
- ML components:
  - Experiments
  - Feature Stores
  - Models



# Experiments

- In the 02 Train and Evaluate Models notebook, Cmd27
- Uses mlflow to trigger experiment via code
- Experiments can show MLflow experiments across an organization that you have access to

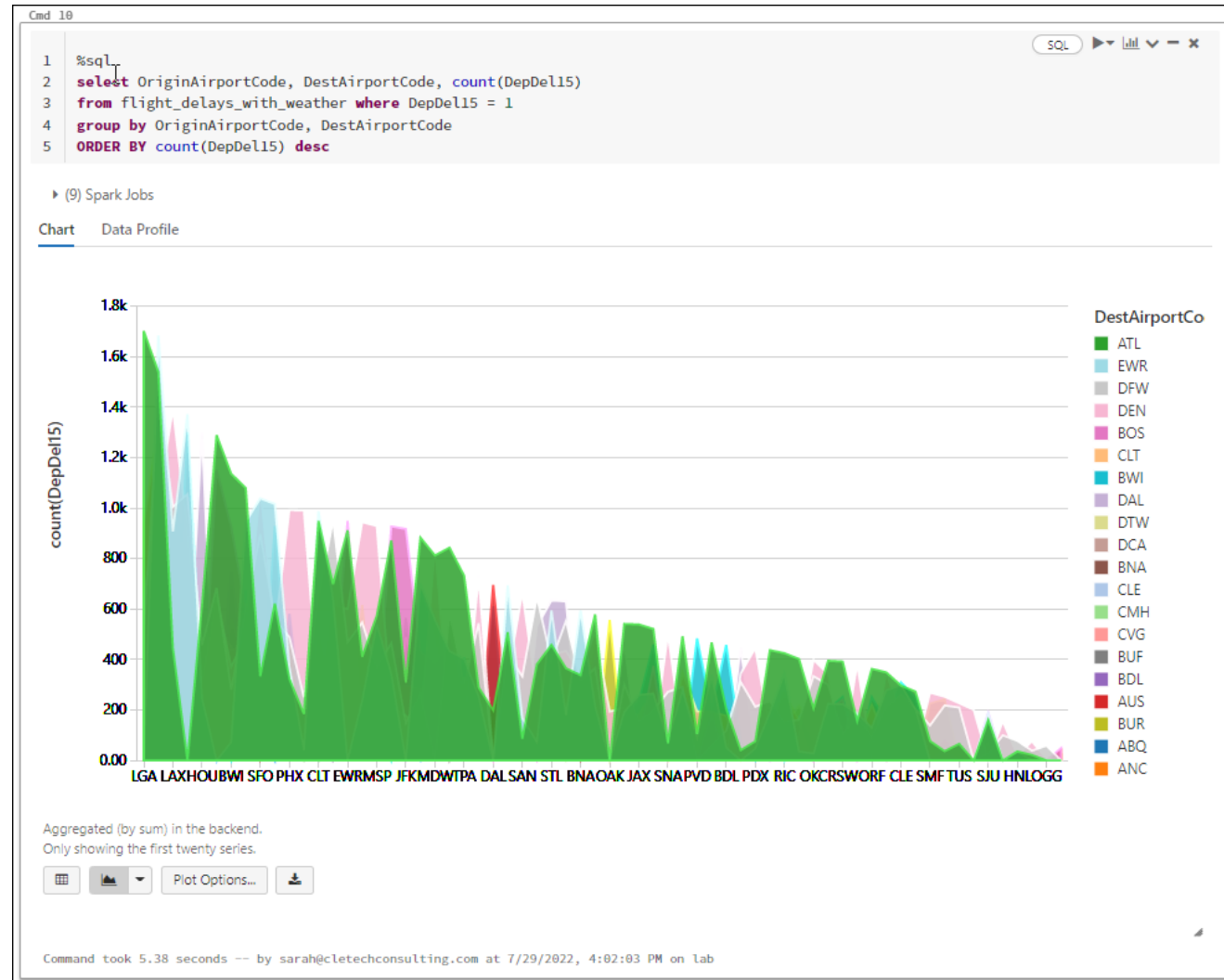


# Creating your own AutoML experiment

- Select a compute cluster
- ML Problem Type:
  - Classification
  - Regression
  - Forecasting – ML Runtimes 10.x and higher
- Evaluation metric (advanced configuration):
  - Classification
    - F1 score
    - Accuracy
    - Log loss
    - Precision
    - ROC/AUC
  - Regression
    - R-squared
    - Mean absolute error
    - Mean squared error
    - Root mean squared error

# Train and Evaluate Models

- Working with PySpark in Python
- Using stratified sampling with sampleBy() function
- Using **binary classification** – flight is either delayed or it is not
- Using the Decision Tree classifier from Spark MLlib
- Models will be saved for **batch scoring**





# Training and Evaluating Models

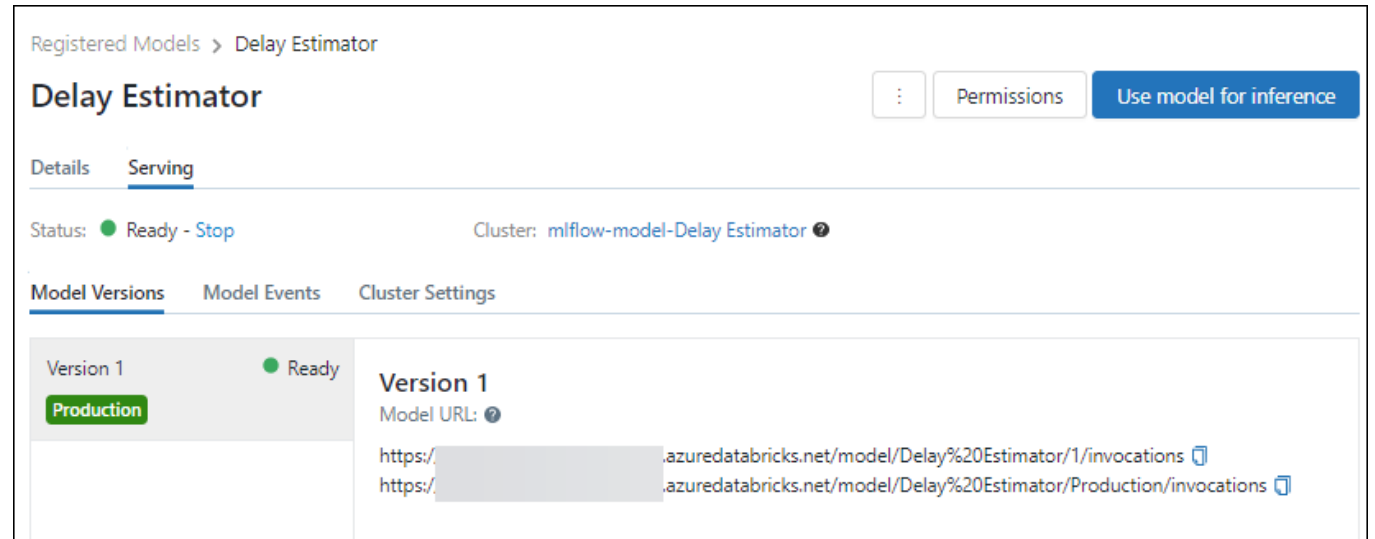
1. Load the cleaned flight delays with weather data.
2. Sample the data by the **DepDel15** field using `sampleBy()`.
  - We will pass a fraction for **stratified sampling**.
3. Select an algorithm and transform the features.
  - Flight delayed or not – binary classification => Decision Tree classifier from Spark MLlib.
  - Using `StringIndexer` and `OneHotEncoderEstimator` for categorical conversion
  - Using the Pipeline API for tying multiple stages together
4. Save the model for batch scoring.

# Batch Scoring

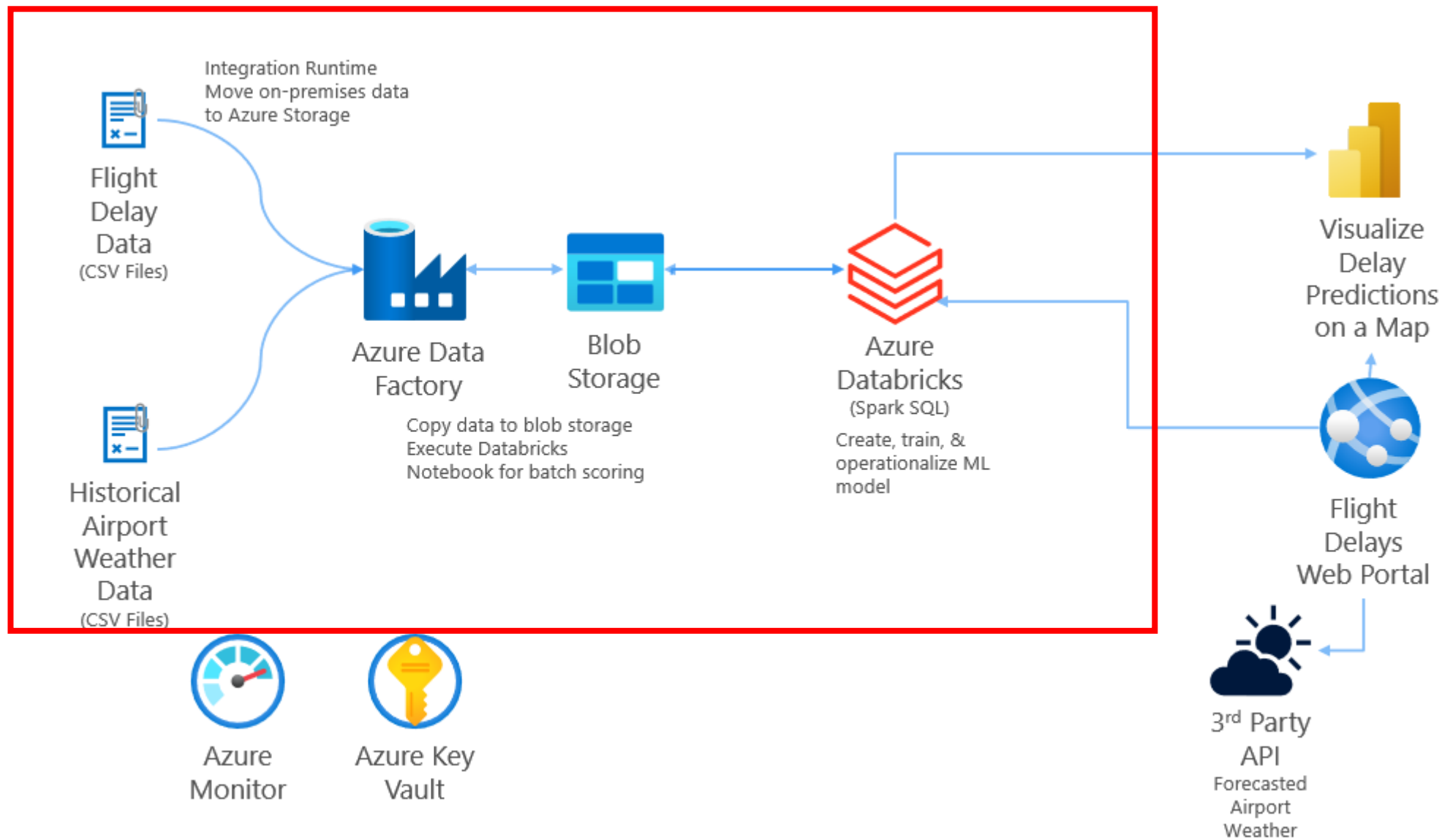
- Reads from Azure Data Storage
- Creates DataFrames for the CSVs
- Load the trained model
- Make predictions against the set
- Save the scored data in a global table **scoredflights**

# Deploy model for batch scoring

1. Register the model with MLflow.
2. Move the model to Production.
3. Set up the service for the model.
4. Test the service.
5. Once all is confirmed, set up the pipeline for batch scoring.



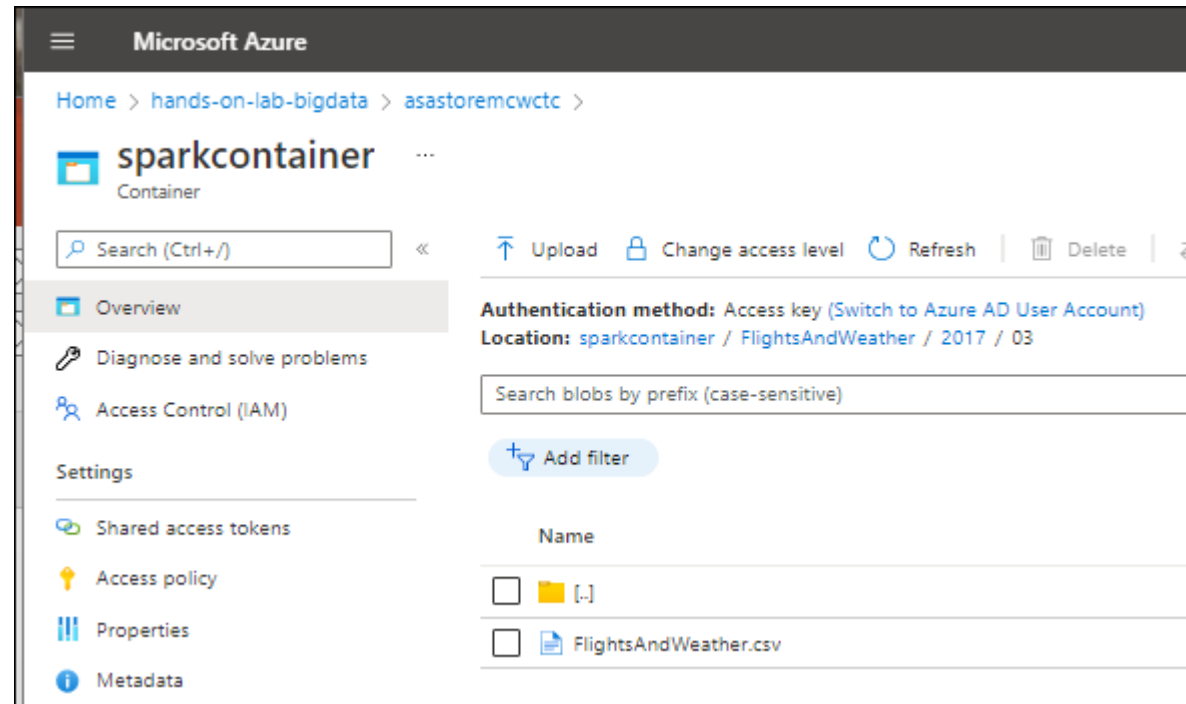
# Setting up the Azure Data Factory Pipeline with the Azure Databricks Notebook



Source: [Microsoft Cloud Workshop: Big Data Analytics and Visualization](#), Hands-on Lab

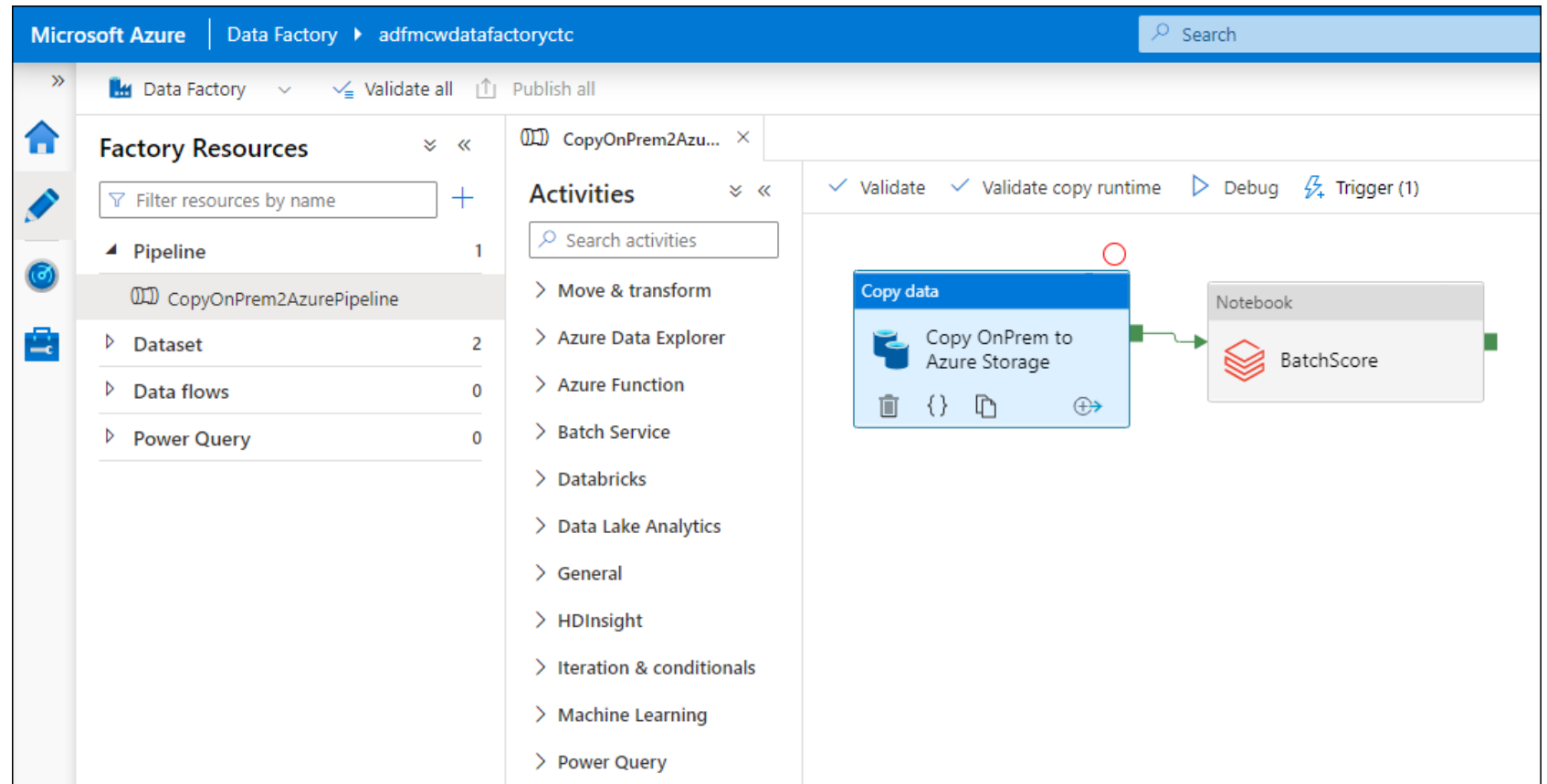
# Data Source: Data Lake

- Using Azure Storage
- CSVs will get migrated from an on-premises environment to Azure Storage using Azure Data Factory



# Azure Data Factory

- Azure Data Factory can migrate data from on-premises to Azure Storage.
- Once migrated, we can run a Databricks notebook to score the data.



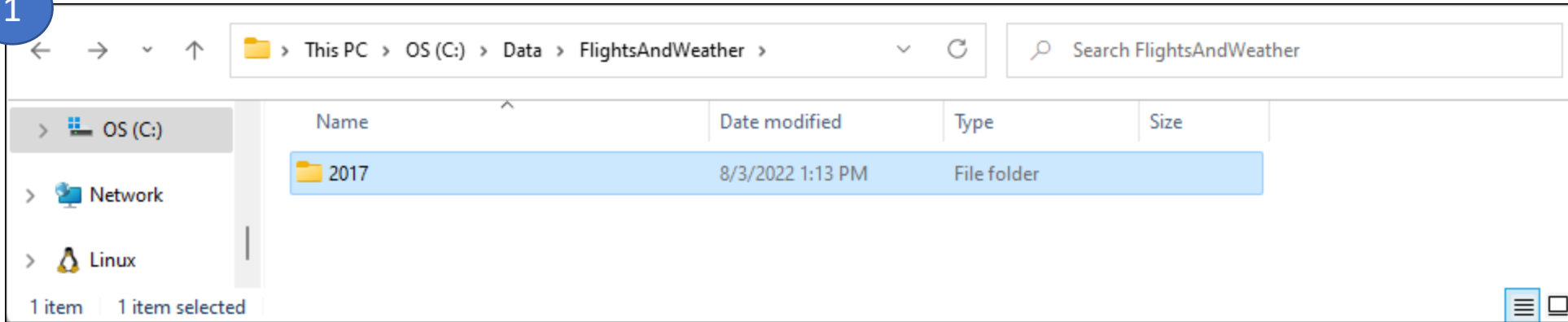
# Setting Up Azure Data Factory

1. Set up the integration runtime.
2. Create a pipeline that uses the integration runtime to move from local to Azure.
3. Add Azure Databricks notebook for batch scoring
4. Publish changes.
5. Trigger the pipeline.

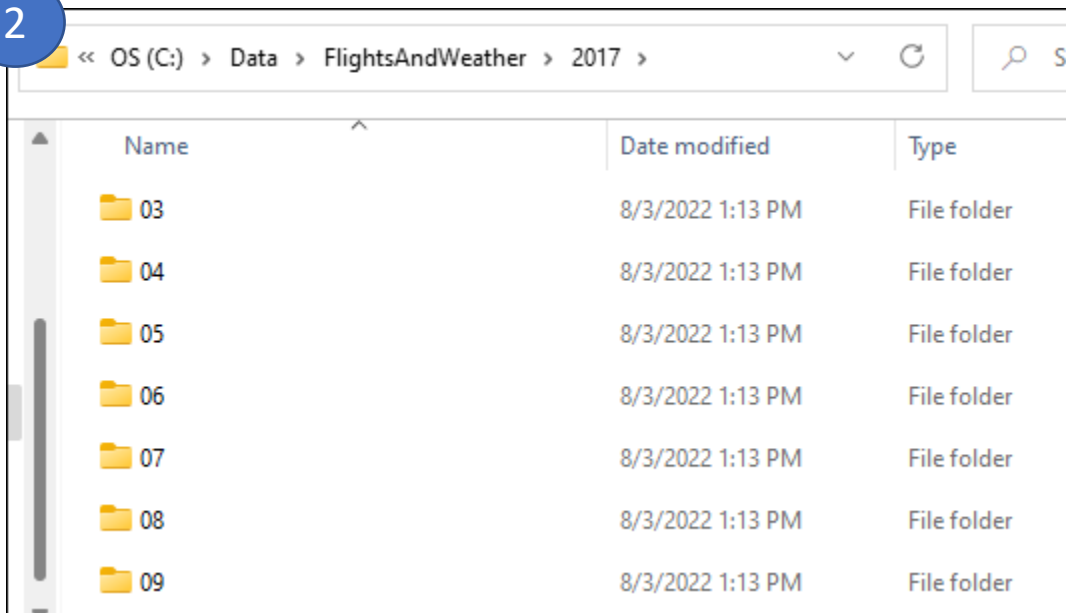


# Local Folder Setup

1



2



3



Put the files in a location where the integration runtime will have permission to access them.



# Azure Storage Setup

**Authentication method:** Access key ([Switch to Azure AD User Account](#))

**Location:** sparkcontainer / FlightsAndWeather / 2017 / 03

Search blobs by prefix (case-sensitive)  ☐ Show deleted blobs

+ Add filter

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state	
<input type="checkbox"/>  [..]							...
<input type="checkbox"/>  FlightsAndWeather.csv	8/3/2022, 1:33:05 PM	Hot (Inferred)		Block blob	43.35 MiB	Available	...

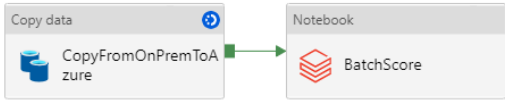
# Monitoring Pipeline Runs

[All pipeline runs](#) > CopyOnPrem2AzurePipeline - Activity runs

## CopyOnPrem2AzurePipeline

[List](#) [Gantt](#)

[Rerun](#) [Rerun from activity](#) [Rerun from failed activity](#) [Refresh](#) [Update pipeline](#)



+ - [100%] [ ]

### Activity runs

Pipeline run ID ca9525a6-e465-44e4-8532-b0fe999425d5

All status ▾

Showing 1 - 1 of 1 items

Activity name	Activity type	Run start ↑↓	Duration	Status	Error	Log
CopyFromOnPremToAzure	Copy data	Aug 3, 2022, 1:32:16 pm	00:00:15	In progress		

### Activity runs

Pipeline run ID ca9525a6-e465-44e4-8532-b0fe999425d5

All status ▾

Showing 1 - 2 of 2 items

Activity name	Activity type	Run start ↑↓	Duration	Status	Error	Log
BatchScore	Notebook	Aug 3, 2022, 1:33:08 pm	00:06:20	Succeeded		
CopyFromOnPremToAzure	Copy data	Aug 3, 2022, 1:32:16 pm	00:00:51	Succeeded		

# Batch Scoring Notebook

- Defines schema for expected CSV file
- Read in the CSV from the Azure Storage
- Use the saved pipeline model
- Make a prediction for the data within the CSV
- Write the prediction to a **global table** named scoredflights

# Review results

- Databricks SQL
- Databricks Notebook

Cmd 6

```
1 %sql
2 SELECT OriginAirportCode, Month, DayofMonth, CRSDepHour, Sum(prediction) NumDelays,
3        CONCAT(Latitude, ',', Longitude) OriginLatLong
4 FROM scoredflights s
5 INNER JOIN airport_code_location_lookup_clean a
6 ON s.OriginAirportCode = a.Airport
7 WHERE Month = 4
8 GROUP BY OriginAirportCode, OriginLatLong, Month, DayofMonth, CRSDepHour
9 HAVING Sum(prediction) > 1
10 ORDER BY NumDelays DESC
```

► (2) Spark Jobs

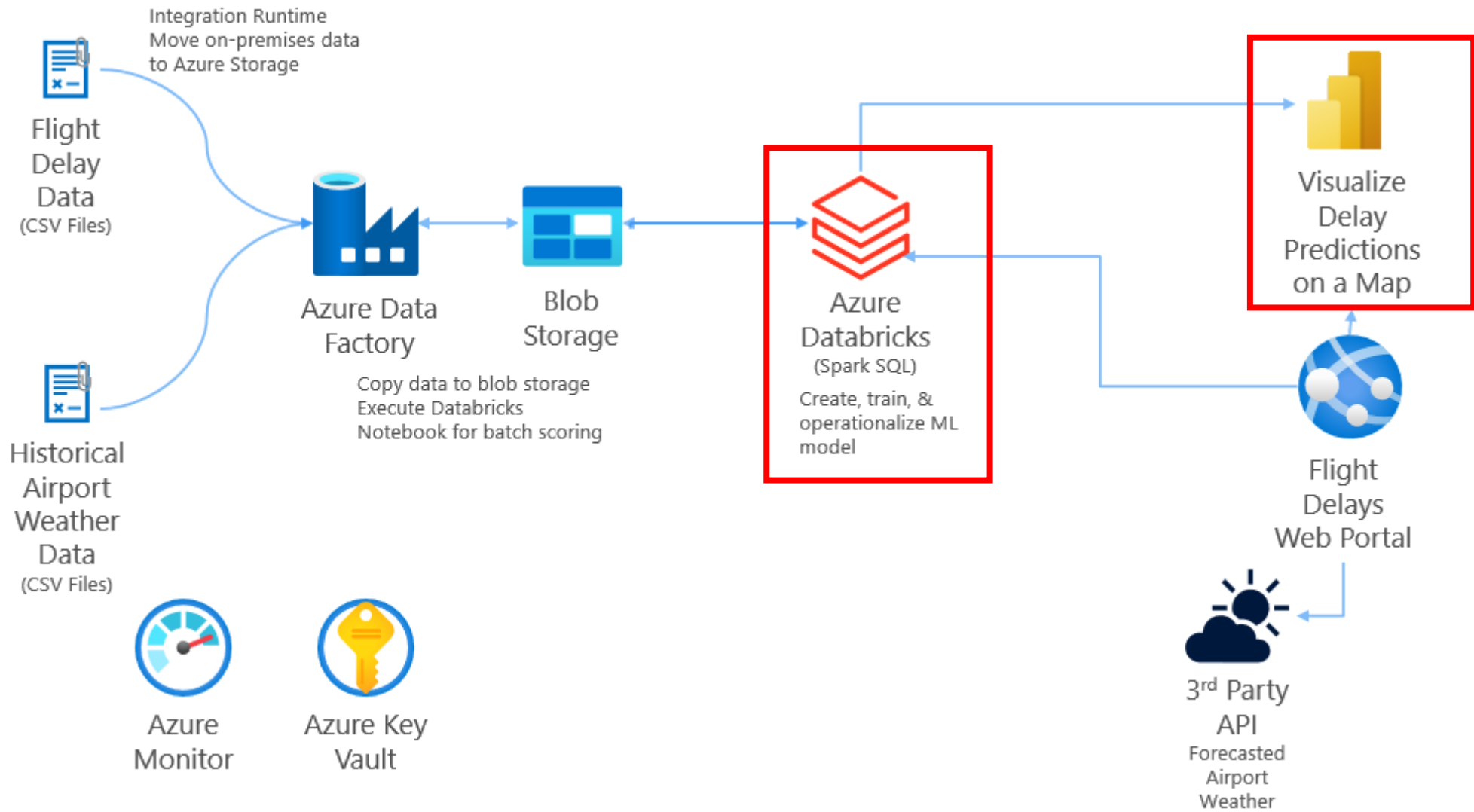
Table Data Profile

	OriginAirportCode ▲	Month ▲	DayofMonth ▲	CRSDepHour ▲	NumDelays ▲	OriginLatLong ▲
1	ATL	4	28	16	120	33.63666667,-84.42777778
2	MSP	4	12	19	88	44.88194444,-93.22166667
3	MSP	4	11	19	72	44.88194444,-93.22166667
4	MSP	4	21	19	72	44.88194444,-93.22166667
5	ATL	4	14	16	72	33.63666667,-84.42777778
6	ORD	4	17	12	66	41.97944444,-87.9075
7	MSP	4	5	17	64	44.88194444,-93.22166667

Truncated results, showing first 1000 rows.  
[Click to re-execute with maximum result limits.](#)

Command took 3.43 seconds -- by sarah@cletechconsulting.com at 8/3/2022, 2:35:08 PM on lab

# Databricks as a Data Source



Source: [Microsoft Cloud Workshop: Big Data Analytics and Visualization](#), Hands-on Lab

## Get Data

azure databricks



All

Azure

All



Azure Databricks

Certified Connectors

Template Apps

Connect

Cancel

## Azure Databricks

Server Hostname ⓘ

Example: example.azuredatabricks.net

HTTP Path ⓘ

Example: sql/protocolv1/o/1814582234607533/7508-187377-agent704

Advanced Options (optional)

Default catalog (optional) ⓘ

Example: abc

Database (optional) ⓘ

Example: abc

Fast Evaluation (optional) ⓘ

Data Connectivity mode ⓘ

☐ Import

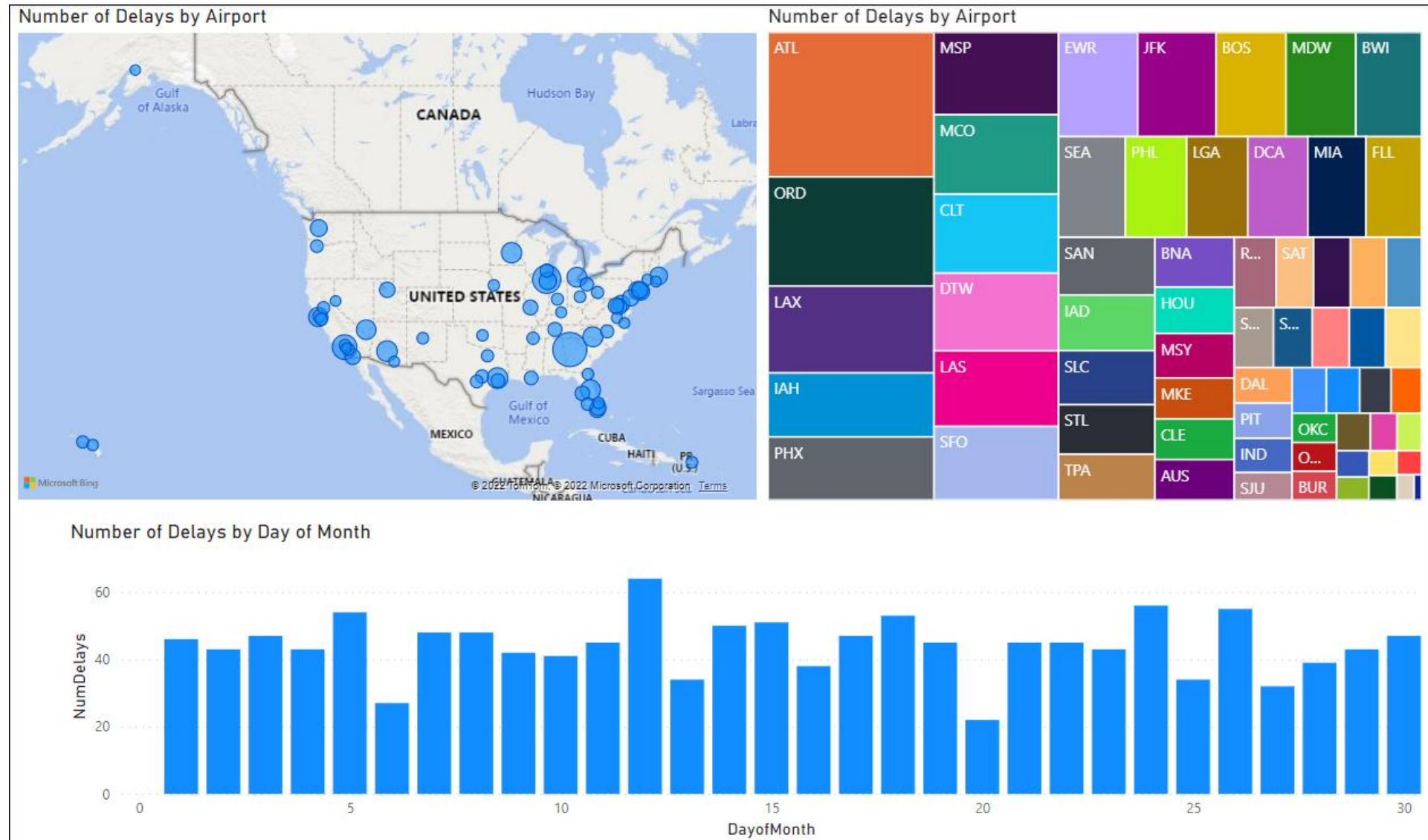
☒ DirectQuery

OK

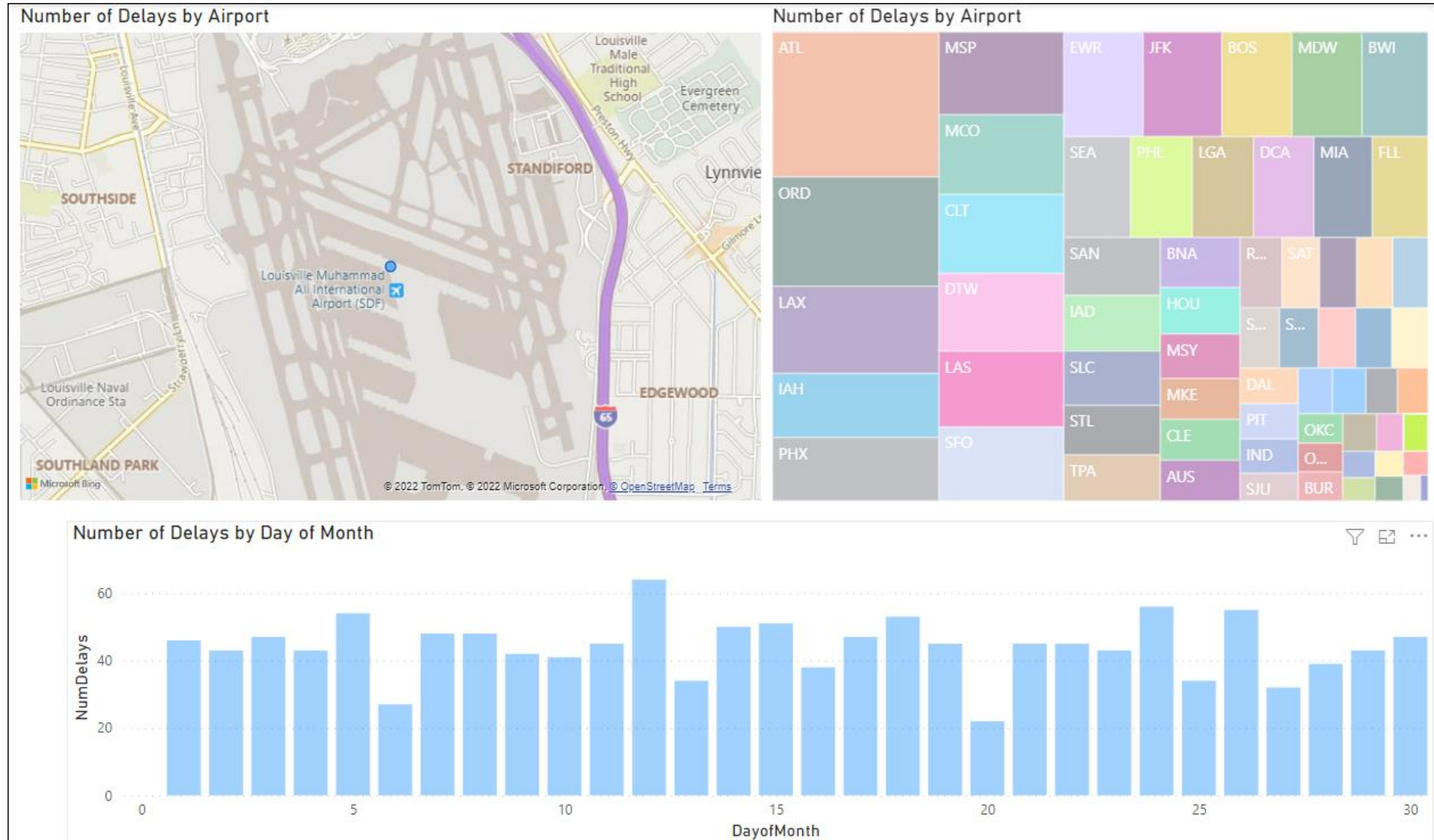
Cancel



# Summary data in Power BI



# Looking at SDF's track record



# Additional Resources

- User guides
  - [Databricks SQL user guide - Azure Databricks - Databricks SQL | Microsoft Docs](#)
  - [Databricks Data Science & Engineering guide - Azure Databricks | Microsoft Docs](#)
  - [Databricks Machine Learning guide - Azure Databricks | Microsoft Docs](#)
- Microsoft Learn pathways
  - [Build and operate machine learning solutions with Azure Databricks - Learn | Microsoft Docs](#)
  - [Data engineering with Azure Databricks - Learn | Microsoft Docs](#)
  - [Perform data science with Azure Databricks - Learn | Microsoft Docs](#)

# Any Questions?



Twitter: [@sadukie](https://twitter.com/sadukie)

LinkedIn:

<https://linkedin.com/in/sadukie>