EBOOK

Modern Analytics With Azure Databricks

databricks

Microsoft Azure

Contents

Introduction

Organizations are turning to cloud platforms like Microsoft Azure to take advantage of scalable services to simplify and accelerate data ingestion, storage, processing, exploration, visualization, machine learning (ML) model preparation, model training and model serving. This eBook discusses the end-to-end technology stack and the role Azure Databricks plays in the architecture for performing modern analytics.

The maturity of data analytics has come a long way from descriptive data analysis, where simple relational databases, historical data and spreadsheet expertise delivered all the insight necessary to a business. These are no longer sufficient to deliver on competitive business opportunities. In order to answer critical business questions, a modern approach to data analytics is required. Modern analytics goes beyond lowering the total cost of storage and compute resources. An even larger opportunity takes the form of innovative and competitive new business growth. At right are the specific challenges and required capabilities:

CHALLENGE	REQUIRED CAPABILITY
Architectural complexity causes confusion Organizations need a single source of truth for their analytics, BI and ML use cases. Building a solution using numerous overlapping cloud services can be confusing and complex, even for common use cases.	**Common architecture for all your data workloads, from data to AI** A shared data lake, an open storage format and a single, optimized compute service enable a simplified architecture to address all your data engineering, data science and analytics workloads.
Slow performance increases total cost of ownership (TCO) As data volume and velocity dramatically increase, processing performance must cost-effectively scale to meet business demands for fresh and clean data. Processing delays result in tangible and intangible costs related to human capital, missed service level agreements, impaired decision-making and lost sales.	**Faster performance to lower your overall cost** Performance innovations must deliver timely results on larger and fresher data sets than analysts have traditionally been able to access. The combination of data warehouse performance and data lake economics enables your organization to tap into vast amounts of data in near real time to uncover predictive insights, enable faster innovation and boost customer lifetime value.
Reduced data team collaboration due to lack of data science and ML integrations Using separate tools for data engineering and data science slows communication and leads to data silos and friction between teams. The end result is reduced productivity and slower business results. Duplicated work lowers team productivity and increases the risk of errors.	**Collaborative environments to boost productivity** Data science and ML tool integration increase data team productivity. Persona-based experiences improve your data team's collaborative workflow. A single source of truth improves communication and ensures consistency across your organization.

Modern analytics enables you to unify all your data, analytics and AI workloads at any scale, and to get insights for all your users through analytics dashboards, operational reports or advanced analytics. Modern analytics with Azure Databricks is simple, open and collaborative, and seamlessly integrated with Azure services such as Azure Data Lake Storage, Azure Data Factory, Azure Synapse Analytics and Power BI.

SIMPLE

Simplify your data architecture by unifying analytics, data science and machine learning.

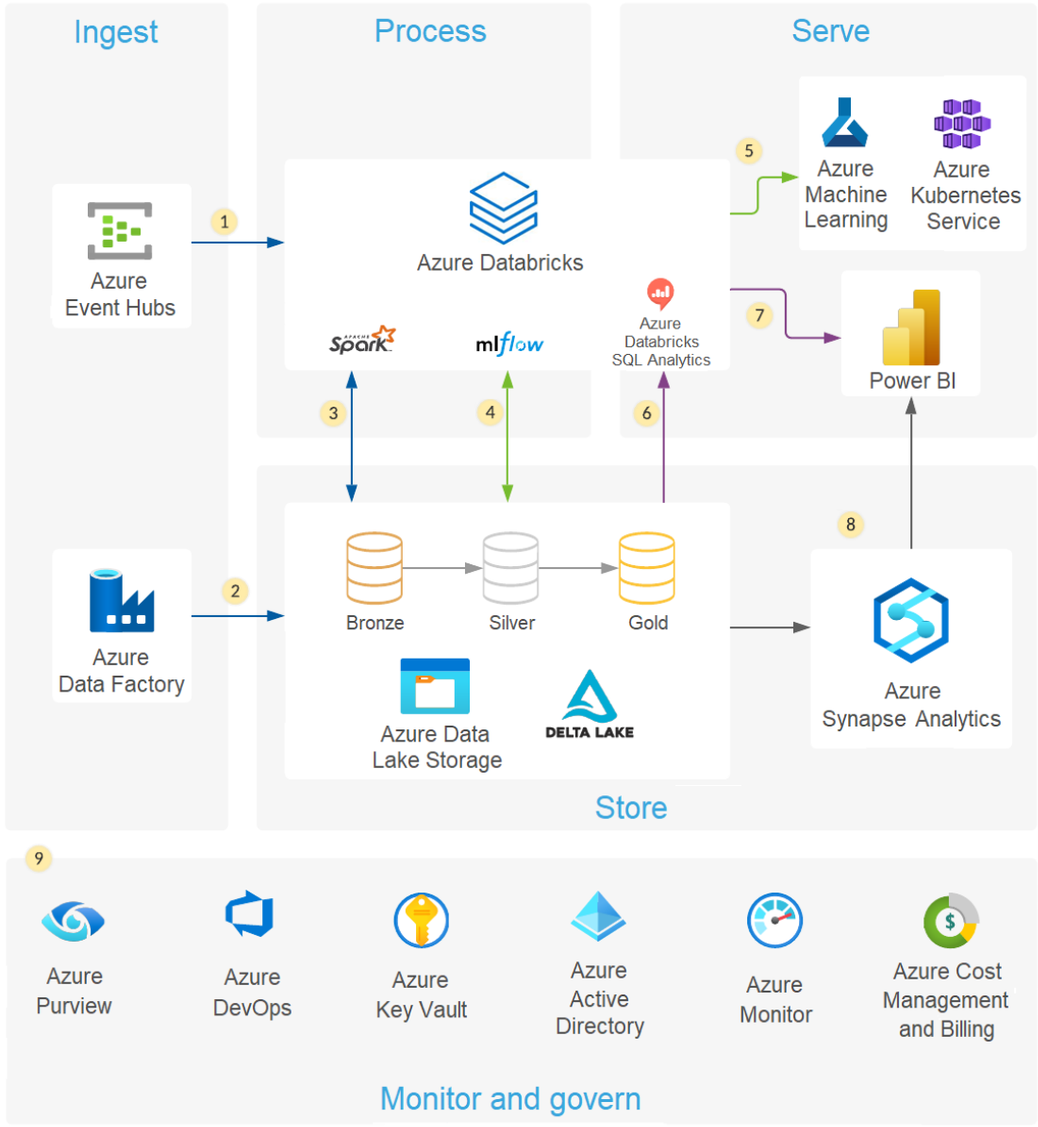
OPEN

Support for open source, open standards and open frameworks helps future-proof your architecture. Work seamlessly with your favorite integrated development environments (IDEs), libraries and programming languages. Integrate with a broad ecosystem of other services through native connectors and APIs.

COLLABORATIVE

Data teams can work together using their favorite tools to collaborate on the same underlying data. Data engineers, data scientists and analysts can leverage a common data lake using collaborative notebooks, IDEs and dashboards.

Architecture



databricks

Data Flow

This reference architecture is inspired by systems built by customers for streaming data analytics and machine learning across numerous industry verticals, including the energy sector, retail and e-commerce, banking and finance, healthcare and medicine.

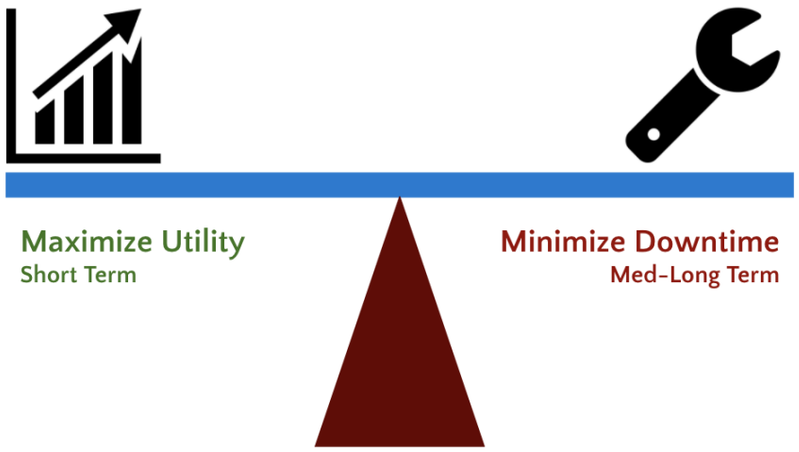
1. Ingests raw streaming data from Azure Event Hubs.
2. Ingests raw batch data from Azure Data Factory into Azure Data Lake Storage.
3. For both streaming and batch data, Azure Databricks combines all your structured, unstructured, semi-structured and streaming data (logs, files and media) using the medallion model with Delta Lake on Azure Data Lake Storage.
4. Data scientists perform data prep, data exploration, model preparation and model training using managed MLflow for parameter, metric and model tracking. Users can leverage the language of their choice, including SQL, Python, R and Scala, and leverage popular open source libraries and frameworks such as Koalas, pandas and scikit-learn, which are pre-installed and optimized. Practitioners can optimize for performance and cost with single- and multi-node compute options.
5. Models can be served natively for batch, streaming and REST APIs in Azure Databricks using the MLflow Model Repository. Optionally, models can be deployed to Azure Machine Learning web services and/or Azure Kubernetes Service (AKS).
6. Users can perform ad hoc SQL queries on the data lake with Azure Databricks SQL Analytics, which provides a query editor and catalog, query history, basic dashboarding and alerting, and integrated security, including row- and column-level permissions, all accelerated with Photon for up to 20x faster performance.
7. Data lake exploration, reports and dashboards in Power BI use a native connector and optimized JDBC/ODBC drivers.
8. Optionally, if a data warehouse is needed for business-ready data and aggregates, Gold data sets can be exported out of the data lake into Azure Synapse via the optimized Synapse connector.
9. Leverage Azure platform services for collaboration, performance, reliability, governance and security:

    - **Azure Purview:** Data discovery and governance insights, data classification and sensitivity insights across your entire data estate.
    - **Azure DevOps:** Continuous integration and continuous deployment (CI/CD). Configure Azure DevOps as your Git provider and take advantage of the integrated version control features.
    - **Azure Key Vault:** Securely manage your secrets such as keys and passwords.
    - **Azure Active Directory:** Use single sign-on (SSO) to Azure Databricks. Azure Databricks also supports automated user provisioning with Azure AD to create new users, give them the proper level of access and remove users to deprovision access.
    - **Azure Monitor:** Collect, analyze and act on telemetry information of your Azure resources to proactively identify problems and maximize performance and reliability.
    - **Azure Cost Management:** Financial governance over your Azure workloads.

Each service connects to the same underlying data to ensure consistency. The architecture leverages a shared data lake leveraging the open Delta Lake format. The analytical platform ingests the data from the disparate batch and streaming sources to form a unified data platform, which can be used to serve analytical reports, serve historical reports for end users and train ML models for a recommendation engine.

Example Use Case — Wind Turbine Optimization

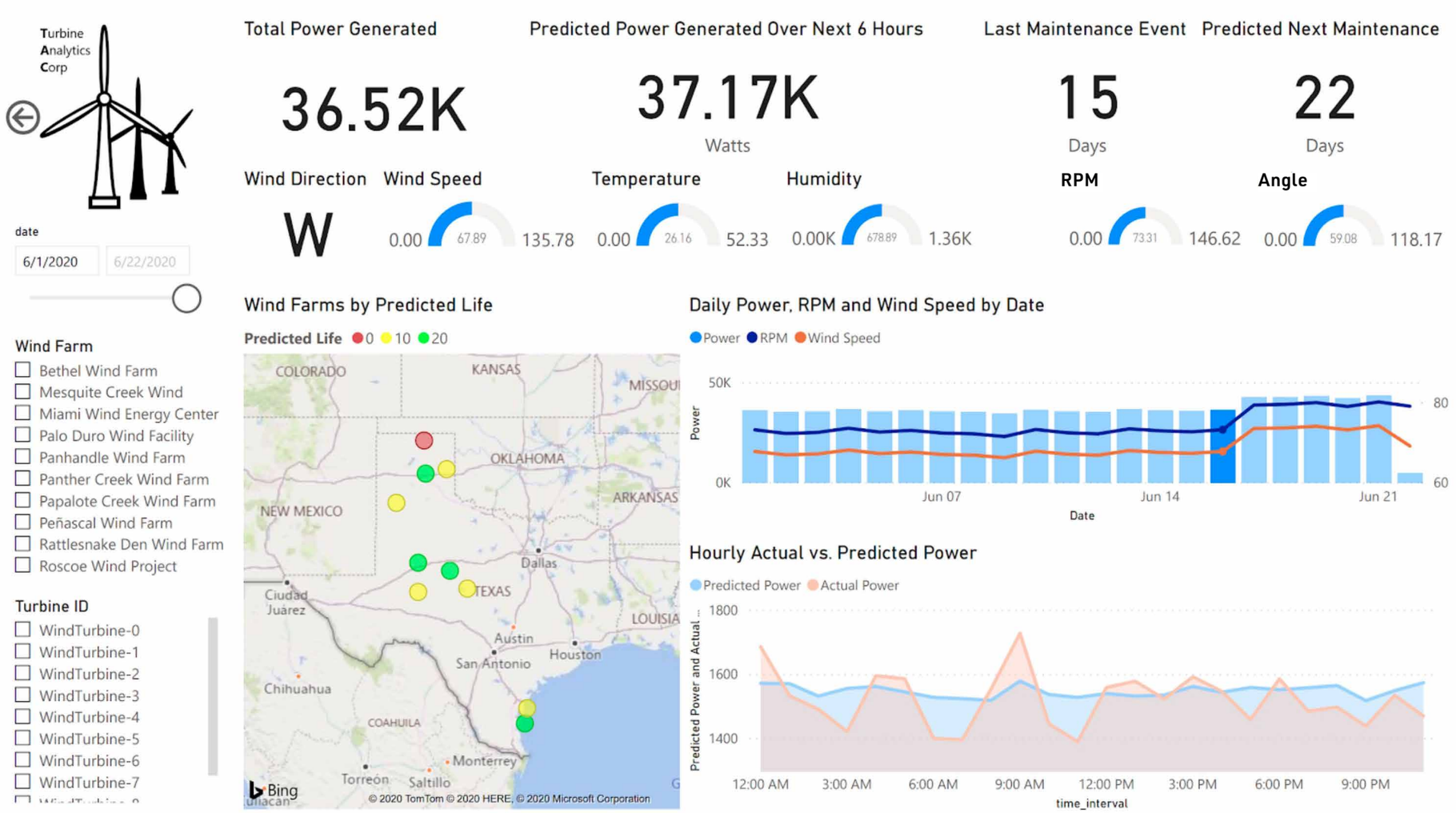
Most Industrial IoT (IIoT) analytics projects are designed to maximize the short-term utilization of an industrial asset while minimizing its long-term maintenance costs. In this example, we focus on a hypothetical energy provider trying to optimize its wind turbines. The ultimate goal is to identify the set of optimal turbine operating parameters that maximize each turbine's power output while minimizing its time to failure.



The goal of IIoT is to maximize utility in the short term while minimizing downtime over the long term.

The final artifacts of this project are:

1. An automated data ingestion and processing pipeline that streams data to all end users
2. A predictive model that estimates the power output of each turbine given current weather and operating conditions
3. A predictive model that estimates the remaining life of each turbine given current weather and operating conditions
4. An optimization model that determines the optimal operating conditions to maximize power output and minimize maintenance costs, thereby maximizing total profit
5. A real-time analytics dashboard for executives to visualize the current and future state of their wind farms, as shown on the next page

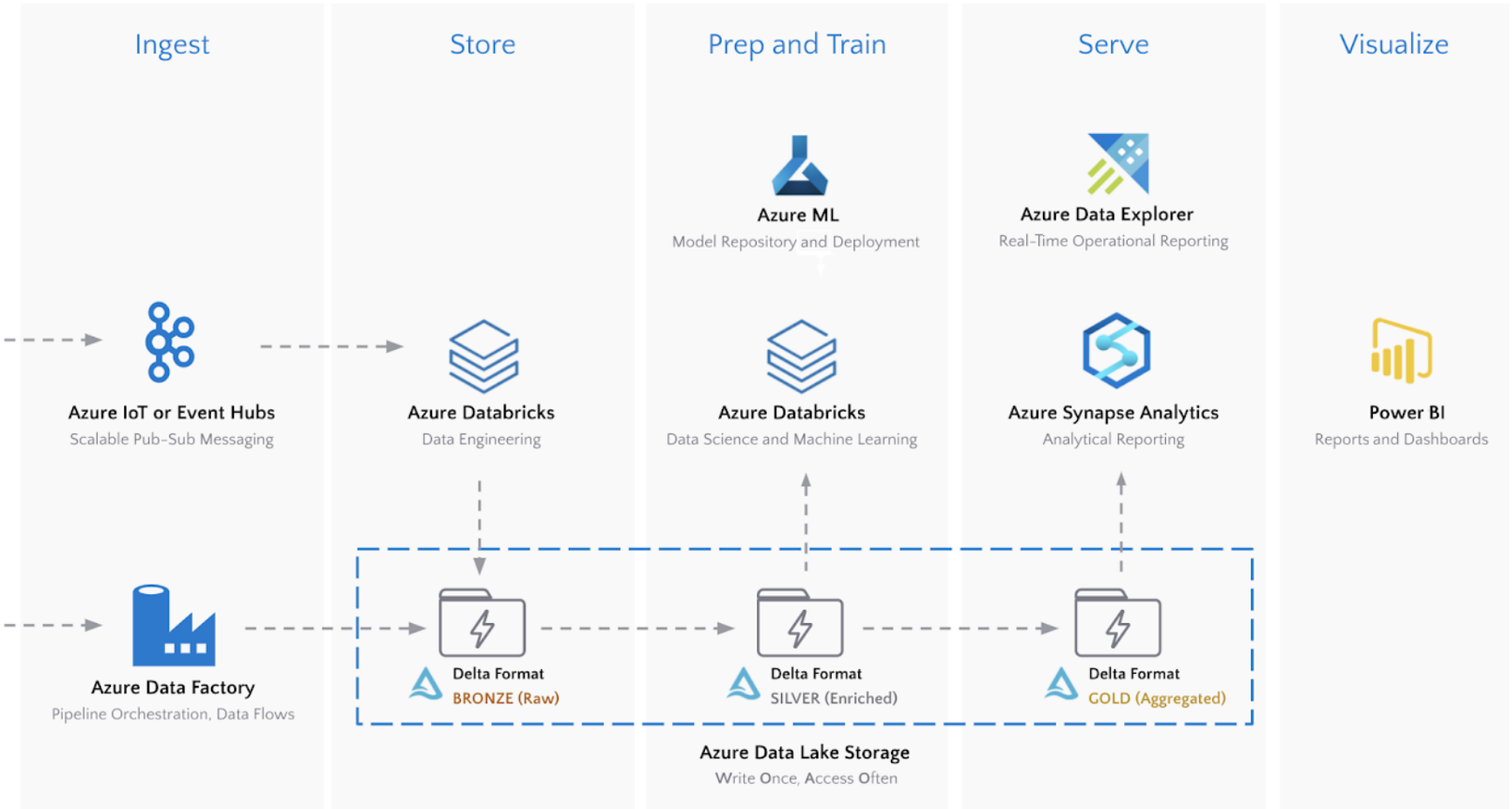


An IIoT analytics dashboard can help business executives visualize, for example, the current and future state of an industrial asset, such as a wind farm.

databricks

The Architecture — Ingest, Store, Prep, Train, Serve, Visualize

The architecture below illustrates a modern, best-of-breed platform used by many organizations that leverages all that Azure has to offer for IIoT analytics.



The IIoT data analytics architecture featuring the Azure Data Lake Storage and Delta storage format offers data teams the optimal platform for handling time-series streaming data.

databricks

A key component of this architecture is Azure Data Lake Storage (ADLS), which enables the write-once, access-often analytics pattern in Azure. However, data lakes alone do not solve challenges that come with time-series streaming data. The Delta storage format provides a layer of resiliency and performance on all data sources stored in ADLS. Specifically for time-series data, Delta provides the following advantages over other storage formats on ADLS:

REQUIRED CAPABILITY	OTHER FORMATS ON ADLS GEN 2	DELTA FORMAT ON ADLS GEN 2
Unified batch and streaming	Data lakes are often used in conjunction with a streaming store like Cosmos DB, resulting in a complex architecture.	ACID-compliant transactions enable data engineers to perform streaming ingest and historically batch loads into the same locations on ADLS.
Schema enforcement and evolution	Data lakes do not enforce schema, requiring all data to be pushed into a relational database for reliability.	Schema is enforced by default. As new IoT devices are added to the data stream, schemas can be evolved safely so downstream applications don't fail.
Efficient upserts	Data lakes do not support in-line updates and merges, requiring deletion and insertions of entire partitions to perform updates.	MERGE commands are effective for situations handling delayed IoT readings, modified dimension tables used for real-time enrichment or if data needs to be reprocessed.
File compaction	Streaming time-series data into data lakes generates hundreds or even thousands of tiny files.	Auto-compaction in Delta optimizes the file sizes to increase throughput and parallelism.
Multidimensional clustering	Data lakes provide push-down filtering on partitions only.	Z-ORDERing time-series on fields like timestamp or sensor ID allows Databricks to filter and join on those columns up to 100x faster than simple partitioning techniques.

The Deployment

We use Azure's Raspberry Pi IoT simulator to simulate real-time machine-to-machine sensor readings and send them to Azure IoT Hub.

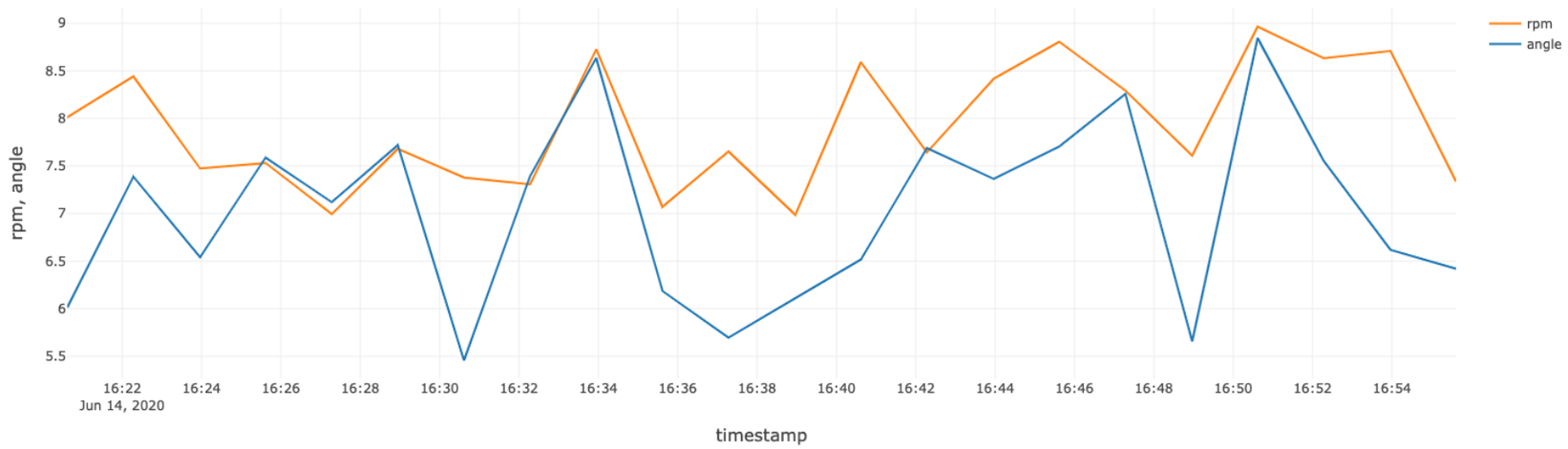
Data Ingest: Azure IoT Hub to Data Lake

Our deployment sends sensor readings for weather (wind speed and direction, temperature, humidity) and wind turbine telematics (angle and RPM) to an IoT cloud computing hub. Azure Databricks can natively stream data from IoT Hub directly into a Delta table on ADLS and display the input rate vs. the processing rate of the data.

```
# Read directly from IoT Hubs using the EventHubs library for Azure Databricks
iot_stream = (
   spark.readStream.format("eventhubs") # Read from IoT Hubs directly
   .options(**ehConf) # Use the Event-Hub-enabled connect string
   .load() # Load the data
   .withColumn('reading', F.from_json(F.col('body').cast('string'), schema)) # Extract the payload
   .select('reading.*', F.to_date('reading.timestamp').alias('date')) # Create a "date" field
)

# Split our IoT Hubs stream into separate streams and write them both into their own Delta locations
write_turbine_to_delta = (
   iot_stream.filter('temperature is null') # Filter out turbine telemetry data
   .select('date','timestamp','deviceId','rpm','angle') # Extract the fields of interest
   .writeStream.format('delta') # Write our stream to the Delta format
   .partitionBy('date') # Partition our data by Date for performance
   .option("checkpointLocation", ROOT_PATH + "/bronze/cp/turbine") # Checkpoint
   .start(ROOT_PATH + "/bronze/data/turbine_raw") # Stream the data into an ADLS path
)
```

Delta allows our IoT data to be queried within seconds of it being captured in IoT Hub.

```
%sql
-- We can query the data directly from storage immediately as it streams into Delta
SELECT * FROM delta.`/tmp/iiot/bronze/data/turbine_raw` WHERE deviceid = 'WindTurbine-1'
```



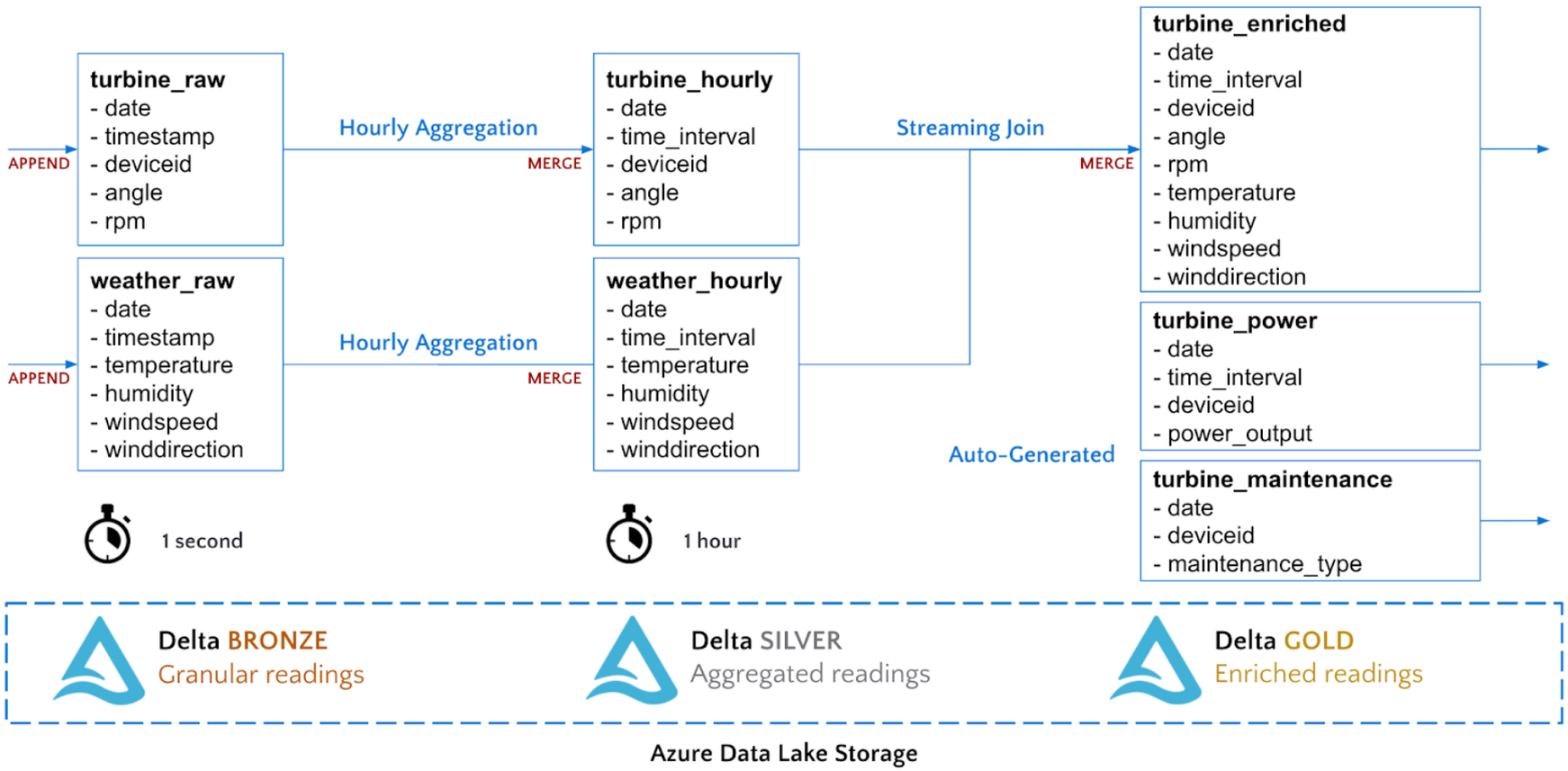
With the Delta storage format, IIoT data can be queried within seconds of capture for use with downstream analytics, such as a time-series data visualization.

We can now build a downstream pipeline that enriches and aggregates our IIoT applications data for data analytics.

databricks

Data Storage and Processing: Azure Databricks and Delta Lake

Delta Lake supports a multi-hop pipeline approach to data engineering, where data quality and aggregation increase as the data streams through the pipeline. Our time-series data will flow through the following Bronze, Silver and Gold data levels.



Delta Lake supports a multi-hop pipeline approach to data engineering, where data quality and aggregation increase as the data streams through the pipeline.

databricks

Our pipeline from Bronze to Silver will simply aggregate our turbine sensor data to one-hour intervals. We will perform a streaming MERGE command to upsert the aggregated records into our Silver Delta tables.

```
# Create functions to merge turbine and weather data into their target Delta tables
def merge_records(incremental, target_path):
   incremental.createOrReplaceTempView("incremental")

# MERGE consists of a target table, a source table (incremental),
# a join key to identify matches (deviceid, time_interval), and operations to perform
# (UPDATE, INSERT, DELETE) when a match occurs or not
incremental._jdf.sparkSession().sql(f"""
   MERGE INTO turbine_hourly t
   USING incremental i
   ON i.date=t.date AND i.deviceId = t.deviceid AND i.time_interval = t.time_interval
   WHEN MATCHED THEN UPDATE SET *
   WHEN NOT MATCHED THEN INSERT *
""")
# Perform the streaming merge into our data stream
turbine_stream = (
   # Read data as a stream from our Bronze Delta
   spark.readStream.format('delta').table('turbine_raw')
   # Aggregate readings to hourly intervals
   .groupBy('deviceId','date',F.window('timestamp','1 hour'))
   .agg({"rpm":"avg","angle":"avg"})
   .writeStream
   .foreachBatch(merge_records) # Pass each micro-batch to a function
   .outputMode("update") # Merge works with update mod
   .start()
)
```

Our pipeline from Silver to Gold will join the two streams into a single table for hourly weather and turbine measurements.

```
# Read streams from Delta Silver tables
turbine_hourly = spark.readStream.format('delta').option("ignoreChanges", True).table("turbine_
hourly")
weather_hourly = spark.readStream.format('delta').option("ignoreChanges", True).table("weather_
hourly")

# Perform a streaming join to enrich the data
turbine_enriched = turbine_hourly.join(weather_hourly, ['date','time_interval'])

# Perform a streaming merge into our Gold data stream
merge_gold_stream = (
   turbine_enriched.writeStream
   .foreachBatch(merge_records)
   .start()
)
```

We can query our Gold Delta table immediately.

Cmd 21

```
1 %sql SELECT * FROM turbine_enriched WHERE deviceid='WindTurbine-1'
```

▸ (3) Spark Jobs

deviceId	date	time_interval	angle	rpm	temperature	humidity	windspeed	winddirection
WindTurbine-1	2020-06-14	2020-06-14T16:00:00.000+0000	7.1520467	7.983291	26.130173	70.060844	7.006084	W
WindTurbine-1	2020-06-14	2020-06-14T17:00:00.000+0000	7.3464413	8.395933	26.089893	70.747955	7.0747952	NW

With Delta Lake, you can query your enriched, AI-ready data immediately, for use with IIoT data science predictive models to optimize asset utilization.

The notebook also contains a cell that will generate historical hourly power readings and daily maintenance logs that will be used for model training. Running that cell will:

1. Backfill historical readings for one year in the turbine_enriched table
2. Generate historical power readings for each turbine in the power_output table
3. Generate historical maintenance logs for each turbine in the turbine_maintenance table

We now have enriched, artificial intelligence (AI)-ready data in a performant, reliable format on Azure Data Lake Storage that can be fed into our data science modeling to optimize asset utilization.

```
%sql
-- Query all 3 tables together
CREATE OR REPLACE VIEW gold_readings AS
SELECT r.*,
   p.power,
   m.maintenance as maintenance
FROM turbine_enriched r
   JOIN turbine_power p ON (r.date=p.date AND r.time_interval=p.time_interval AND r.deviceid=p.
   deviceid)
   LEFT JOIN turbine_maintenance m ON (r.date=m.date AND r.deviceid=m.deviceid);

SELECT * FROM gold_readings
```

date	time_interval	deviceId	rpm	angle	temperature	humidity	windspeed	winddirection	power	maintenance
2020-06-12	2020-06-12T05:00:00.000+0000	WindTurbine-8	6.667731921441555	5.374035574520114	23.79682872841723	61.74833192837229	6.174833192837228	E	124.62010486492946	true
2020-06-12	2020-06-12T06:00:00.000+0000	WindTurbine-8	6.368534099393156	5.132889145951731	22.729005454913175	58.97752970570339	5.897752970570339	N	113.68700051216648	true
2020-06-12	2020-06-12T07:00:00.000+0000	WindTurbine-8	6.277602604888083	5.059600493667568	22.404475130921323	58.13543404685028	5.813543404685028	E	110.46367583934948	true
2020-06-12	2020-06-12T08:00:00.000+0000	WindTurbine-8	9.12659422889947	7.355820935541138	32.572395275892696	84.51929028661552	8.451929028661551	SW	233.47981228981843	true
2020-06-12	2020-06-12T09:00:00.000+0000	WindTurbine-8	7.436996407447541	5.994044711467739	26.54229831777958	68.87231342346766	6.887231342346765	SW	155.03401512015512	true
2020-06-12	2020-06-12T10:00:00.000+0000	WindTurbine-8	7.4902038161716185	6.03692863521874	26.73219311370233	69.3650549994116	6.93650549994116	S	157.2603081067385	true
2020-06-12	2020-06-12T11:00:00.000+0000	WindTurbine-8	7.198240497875406	5.801613047030012	25.690189451537517	66.66124985225359	6.666124985225358	SE	145.23943691974017	true
2020-06-12	2020-06-12T12:00:00.000+0000	WindTurbine-8	6.7642942185052775	5.451862521627735	24.14145512790678	62.642573154790355	6.2642573154790355	NE	128.2557456603227	true
2020-06-12	2020-06-12T13:00:00.000+0000	WindTurbine-8	6.8405200689427	5.513206820207074	24.413527272743726	63.24857583252042	6.324857583252041	S	131.16200676266283	true

Showing the first 1000 rows.

Our data engineering pipeline is complete! Data is now flowing from IoT Hub to Bronze (raw) to Silver (aggregated) to Gold (enriched). It is time to perform some analytics on our data.

Machine Learning: Power Output and Remaining Life Optimization

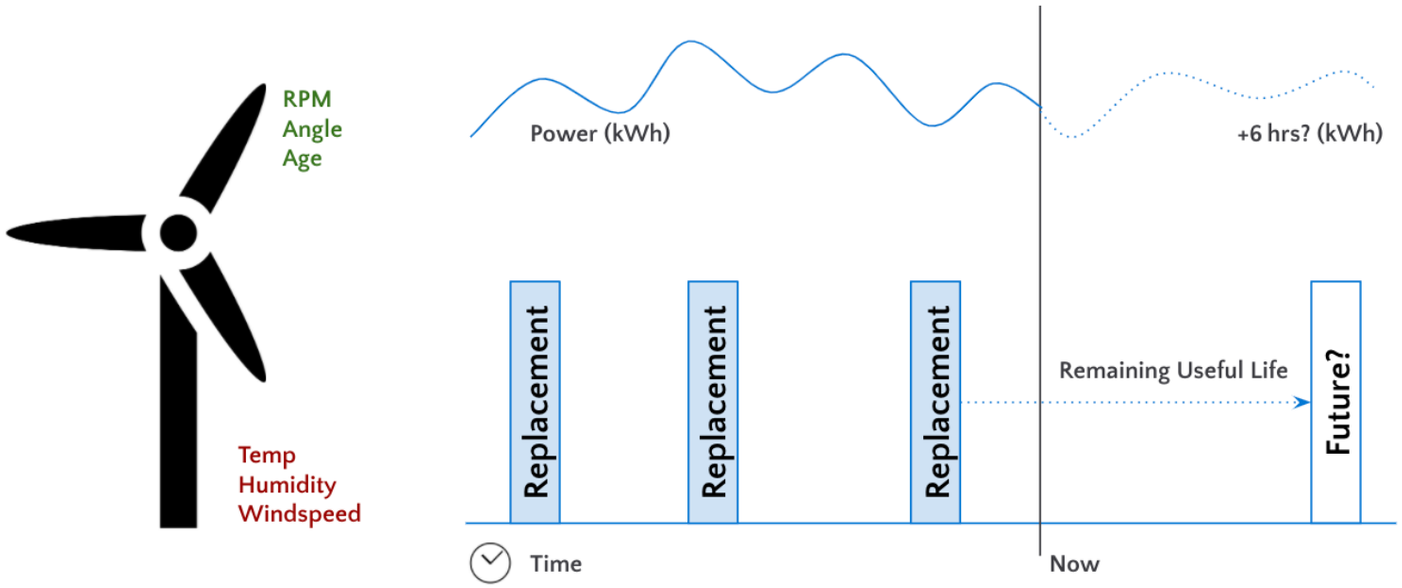
Optimizing the utility, lifetime and operational efficiency of industrial assets like wind turbines has numerous revenue and cost benefits. The real-world challenge we explore in this scenario is to maximize the revenue of a wind turbine while minimizing the opportunity cost of downtime, thereby maximizing our net profit.

Net profit = power generation revenue – cost of added strain on equipment

If we push a turbine to a higher RPM, it will generate more energy and therefore more revenue. However, the added strain on the turbine will cause it to fail more often, introducing cost.

To solve this optimization problem, we will create two models:

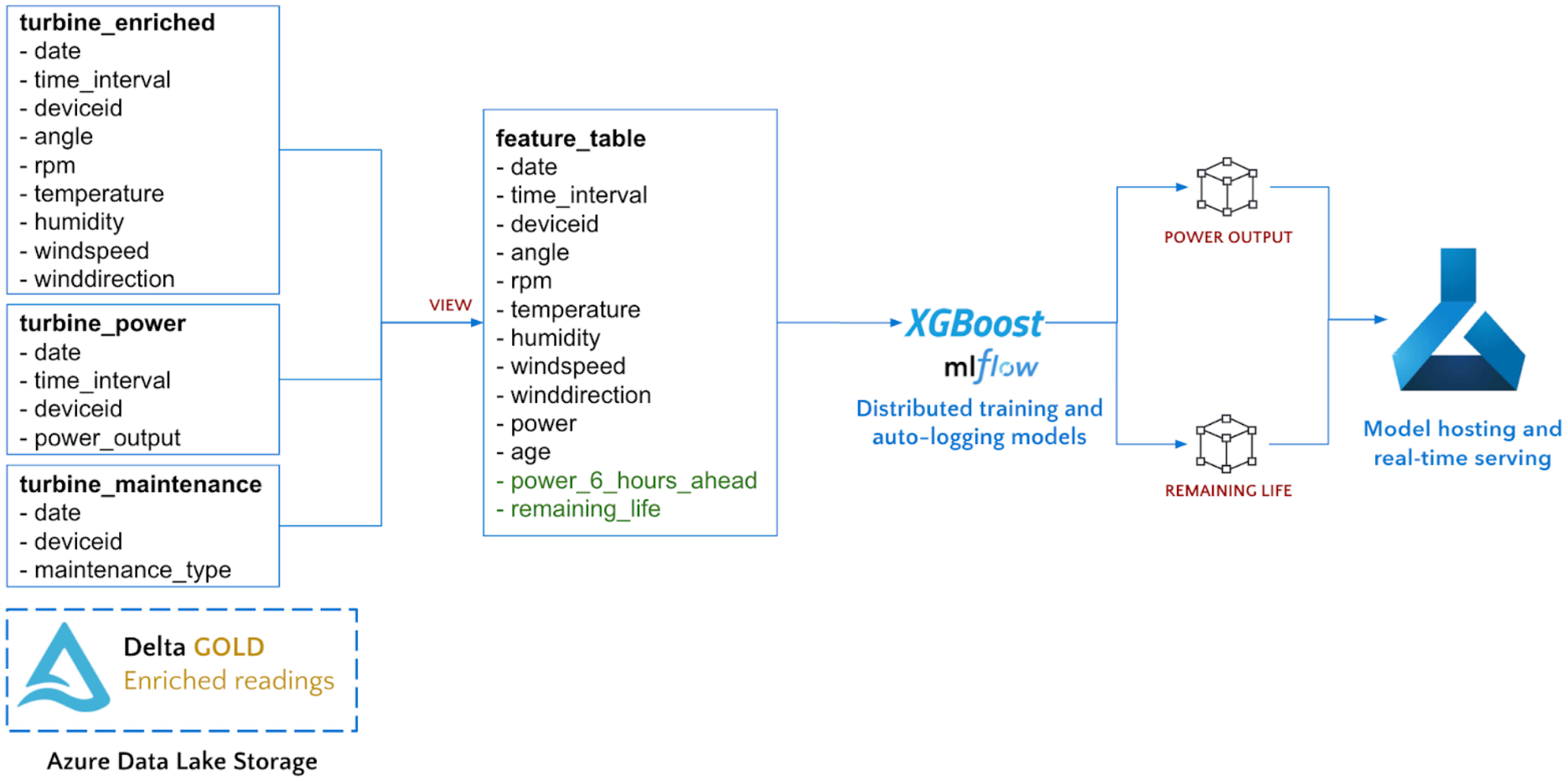
1. Predict the power generated by a turbine given a set of operating conditions
2. Predict the remaining life of a turbine given a set of operating conditions



Using Azure Databricks IIoT data analytics to predict the remaining life of a wind turbine.

We can then produce a profit curve to identify the optimal operating conditions that maximize power revenue while minimizing costs.

Using Azure Databricks with our Gold Delta tables, we will perform feature engineering to extract the fields of interest, train the two models and finally deploy the models to Azure Machine Learning for hosting.



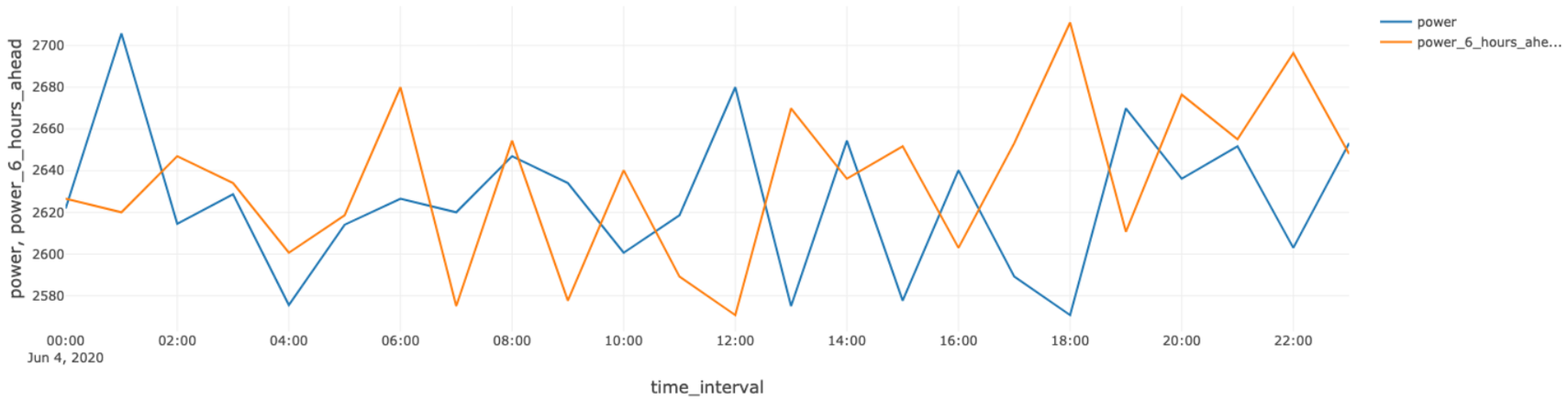
The Azure Databricks machine learning model lifecycle for an IIoT data analytics use case.

To calculate the remaining useful lifetime of each wind turbine, we can use our maintenance records, which indicate when each asset should be replaced.

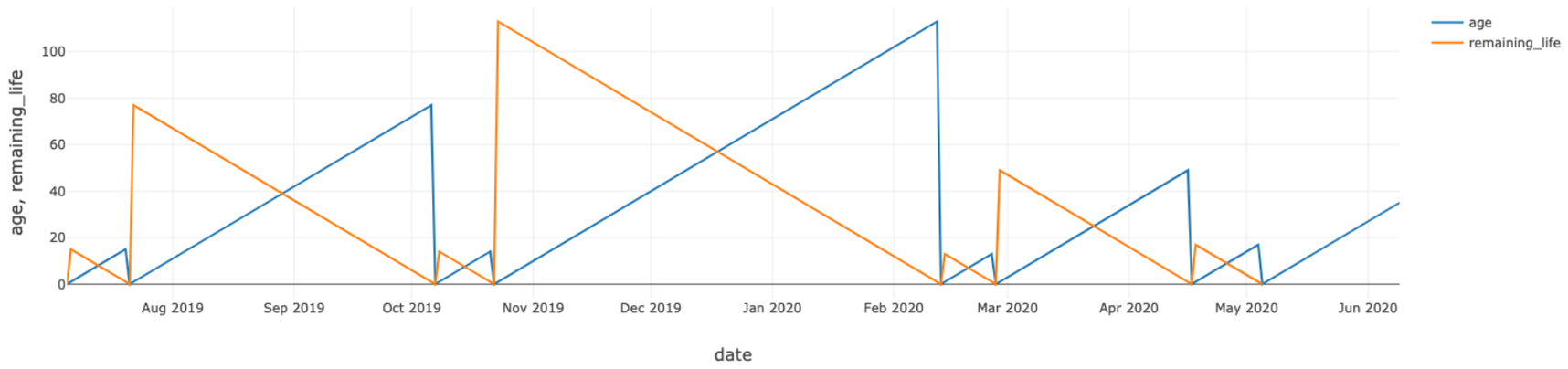
```
%sql
-- Calculate the age of each turbine and the remaining life in days
CREATE OR REPLACE VIEW turbine_age AS
WITH reading_dates AS (SELECT distinct date, deviceid FROM turbine_power),
   maintenance_dates AS (
   SELECT d.*, datediff(nm.date, d.date) as datediff_next, datediff(d.date, lm.date) as
   datediff_last
   FROM reading_dates d LEFT JOIN turbine_maintenance nm ON (d.deviceid=nm.deviceid AND
   d.date<=nm.date)
   LEFT JOIN turbine_maintenance lm ON (d.deviceid=lm.deviceid AND d.date>=lm.date ))
SELECT date, deviceid, min(datediff_last) AS age, min(datediff_next) AS remaining_life
FROM maintenance_dates
GROUP BY deviceid, date;
```

To predict power output at a six-hour time horizon, we calculate time-series shifts using Apache Spark™ window functions.

```
CREATE OR REPLACE VIEW feature_table AS
SELECT r.*, age, remaining_life,
   -- Calculate the power 6 hours ahead using Spark Windowing and build a feature_table to
   feed into  our ML models
   LEAD(power, 6, power) OVER (PARTITION BY r.deviceid ORDER BY time_interval) as
   power_6_hours_ahead
FROM gold_readings r
JOIN turbine_age a ON (r.date=a.date AND r.deviceid=a.deviceid)
WHERE r.date < CURRENT_DATE(); -- Only train on historical data
```



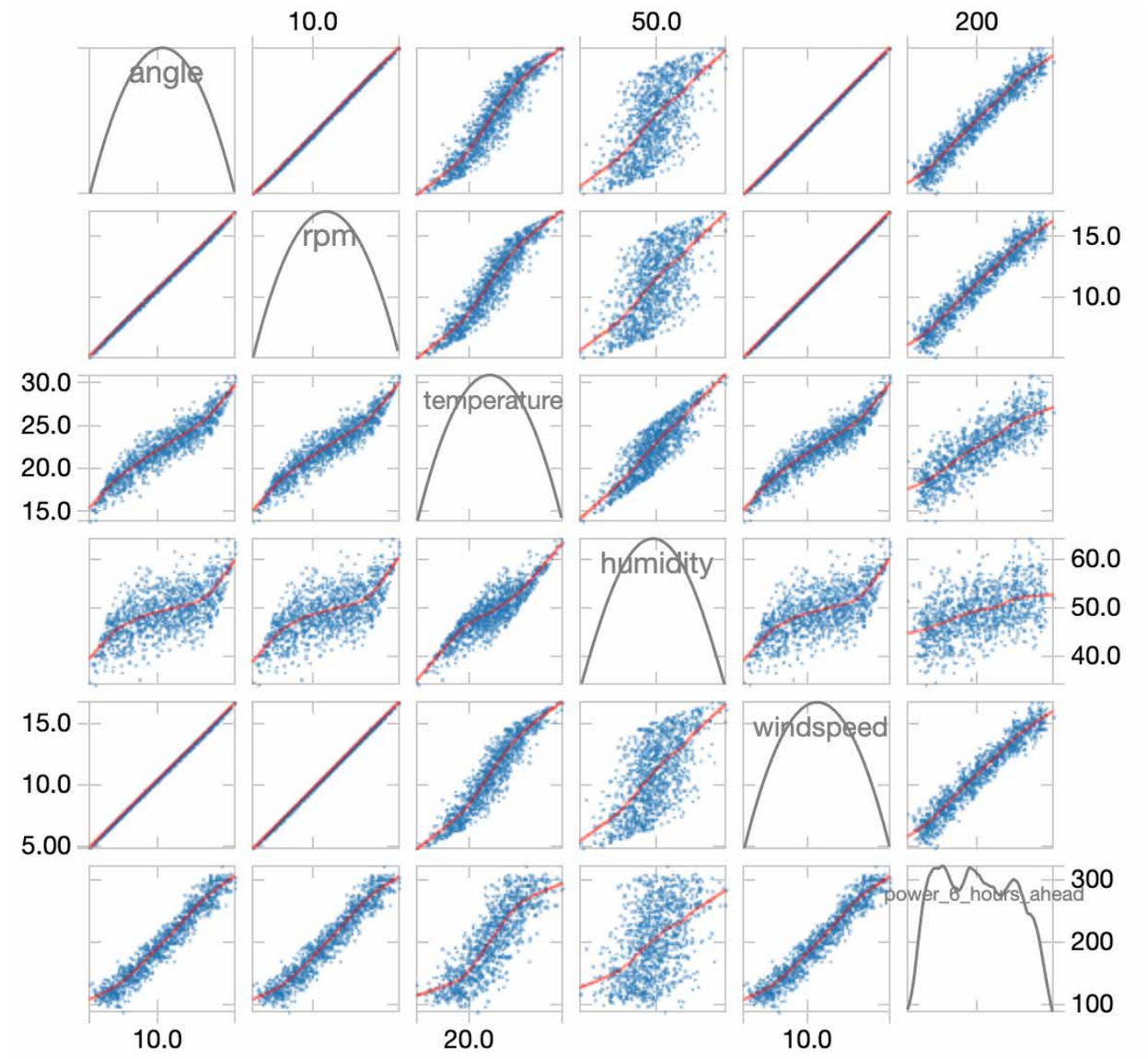
With Azure Databricks, you can calculate time-series shifts using Spark window functions to predict, for example, the power output of a wind farm at a six-hour time horizon.



This shows how Azure Databricks IIoT data analytics lets you calculate the remaining useful lifetime of a wind turbine, using maintenance records that indicate when each asset has been replaced.

databricks

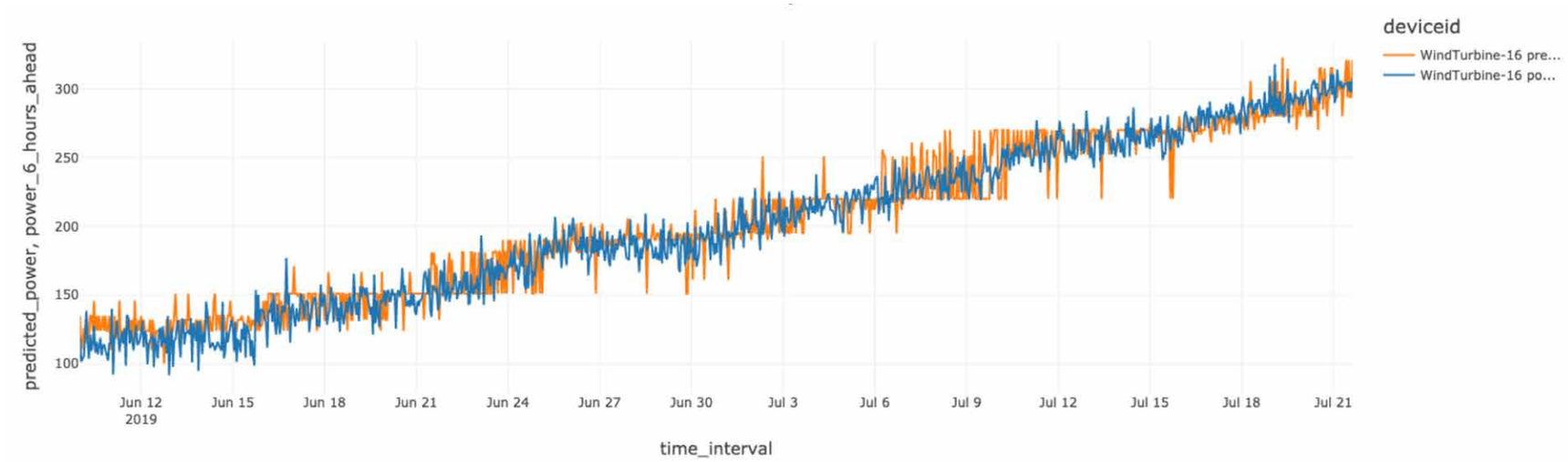
There are strong correlations of turbine operating parameters (RPM and angle) and weather conditions with the power generated six hours from now.



With Azure Databricks IIoT data analytics, you can uncover, for example, the strong correlations of turbine operating parameters (RPM and angle) and weather conditions with future power generated.

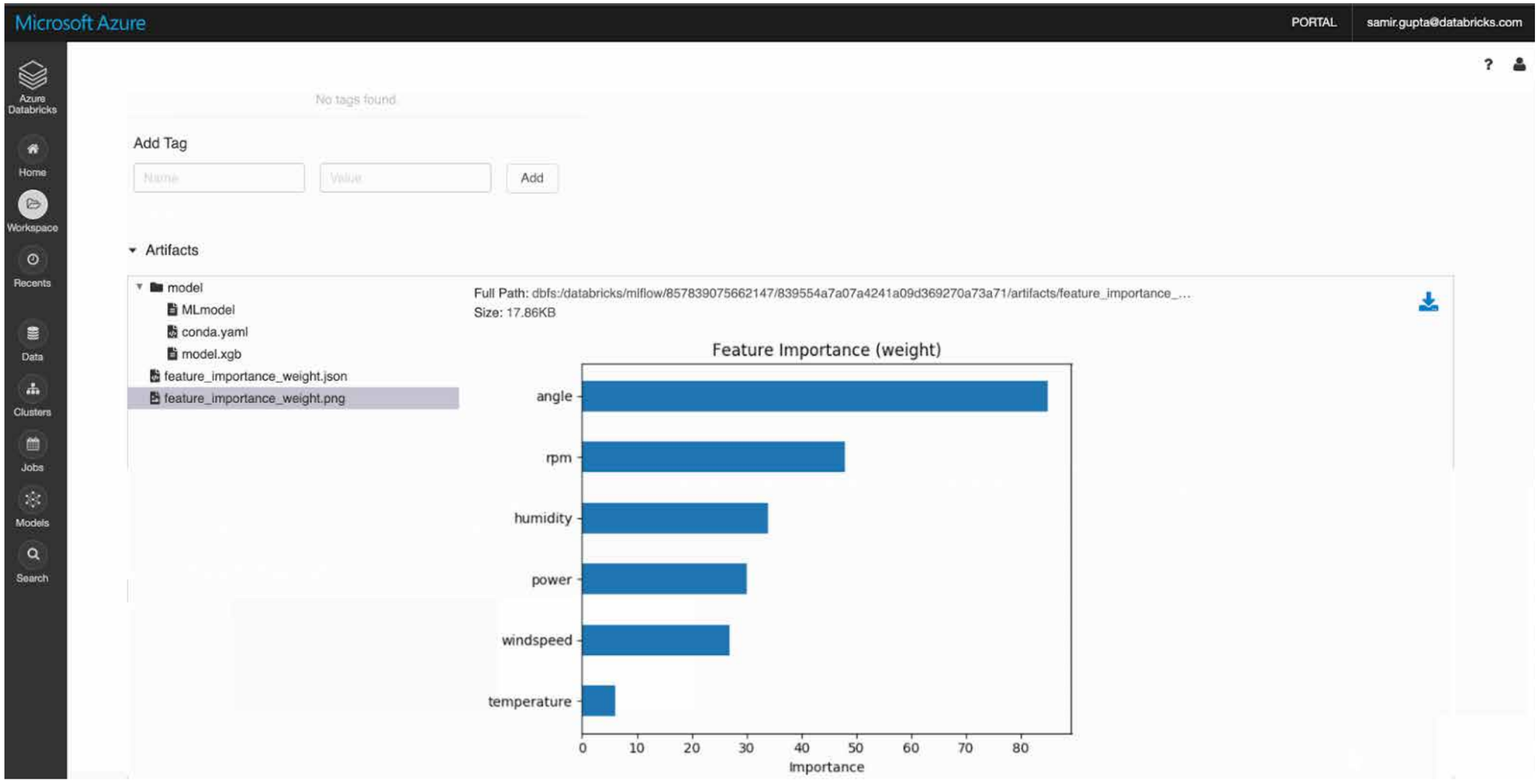
We can now train an XGBoost regressor model to use our feature columns (weather, sensor and power readings) to predict our label (power reading six hours ahead). We can train a model for each wind turbine in parallel using a pandas UDF, which distributes our XGBoost model training code to all the available nodes in the Azure Databricks cluster.

```
# Create a Spark Dataframe that contains the features and labels we need
feature_cols = ['angle','rpm','temperature','humidity','windspeed','power','age']
label_col = 'power_6_hours_ahead'

# Read in our feature table and select the columns of interest
feature_df = spark.table('feature_table')

# Create a Pandas UDF to train a XGBoost Regressor on each turbine's data
@pandas_udf(feature_df.schema, PandasUDFType.GROUPED_MAP)
def train_power_model(readings_pd):
   mlflow.xgboost.autolog() # Auto-Log the XGB parameters, metrics, model and artifacts
   with mlflow.start_run():
   # Train an XGBRegressor on the data for this Turbine
   alg = xgb.XGBRegressor()
   train_dmatrix =
   xgb.DMatrix(data=readings_pd[feature_cols].astype('float'),label=readings_pd[label_col])
   model = xgb.train(dtrain=train_dmatrix, evals=[(train_dmatrix, 'train')])
   return readings_pd

# Run the Pandas UDF against our feature dataset
power_predictions = feature_df.groupBy('deviceid').apply(train_power_model)
```



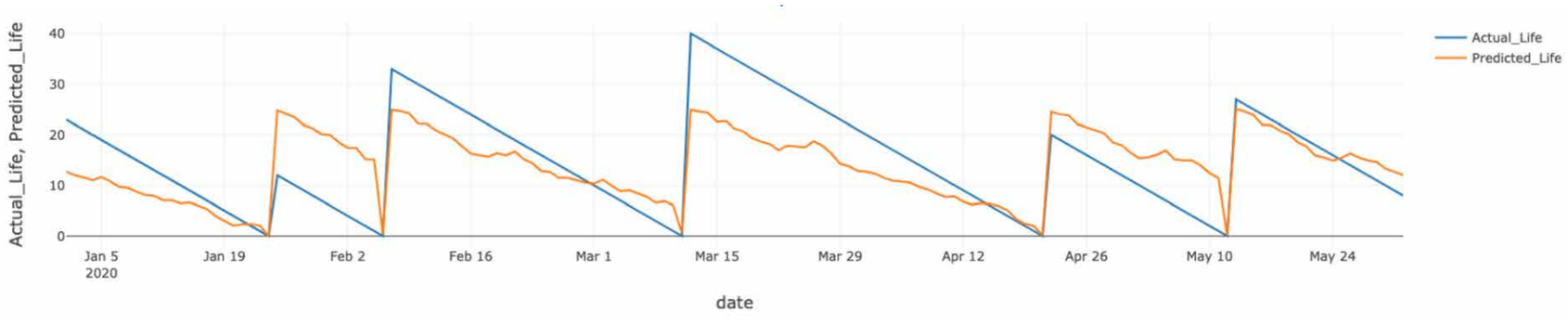
Azure Databricks IIoT data analytics lets you predict the power output of a specific wind turbine and display the results in a time-series visualization, as shown here.

Azure Databricks will automatically track each model training run with a hosted MLflow experiment. For XGBoost regression, MLflow will track any parameters passed into the params argument, the root-mean-square error (RMSE) metric, the turbine this model was trained on and the resulting model itself. For example, the RMSE for predicting power on deviceid WindTurbine-18 is 45.79.



With Azure Databricks IIoT data analytics, you can automatically track each model training run with a hosted MLflow experiment. For XGBoost regression, MLflow will track any parameters passed into the params argument to predict, for example, the power output of a wind turbine by specific deviceid.

We can train a similar model to predict the remaining life of the wind turbine. The actual life vs. the predicted life of one of the turbines is shown below.



With Azure Databricks IIoT data analytics, you can predict, for example, the remaining life span of a wind turbine and generate a time-series visualization comparing the prediction against the actuality.

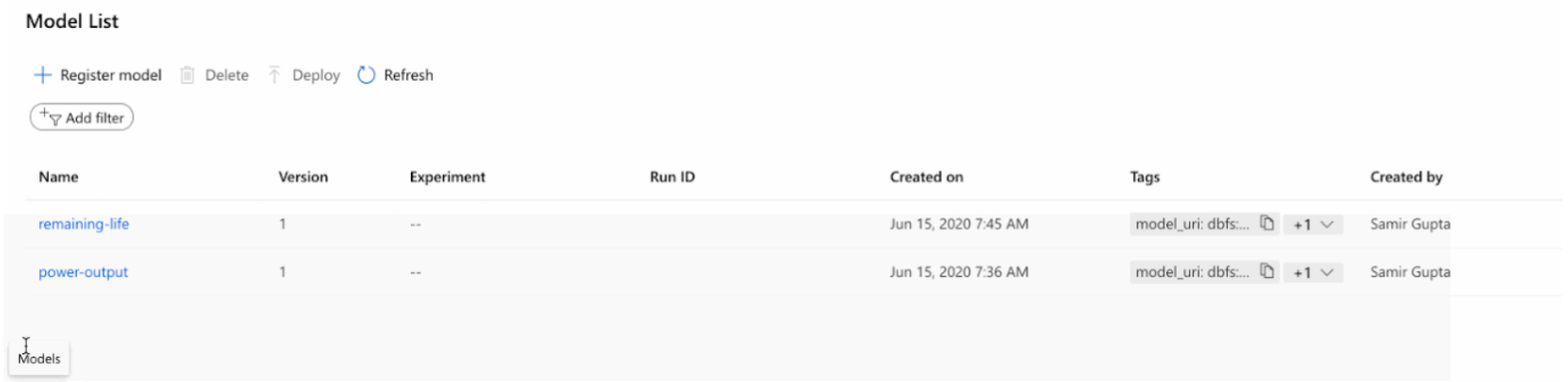
Model Deployment and Hosting

Azure Databricks is integrated with Azure Machine Learning for model deployment and scoring. Using the Azure ML APIs directly inside of Databricks, we can automatically deploy an image for each model to be hosted in a fast, scalable container service (ACI or AKS) by Azure ML.

```
# Create a model image inside of AzureML
model_image, azure_model = mlflow.azureml.build_image(model_uri=path,
   workspace=workspace,
   model_name=model,
```

```
    image_name=model,
    description="XGBoost model to predict power output",
    synchronous=False)

# Deploy a web service to host the model as a REST API
dev_webservice_deployment_config = AciWebservice.deploy_configuration()
dev_webservice = Webservice.deploy_from_image(name=dev_webservice_name,
    image=model_image,
    workspace=workspace)
```

Once the model is deployed, it will show up inside the Azure ML studio, and we can make REST API calls to score data interactively.



Once the model is deployed, it will show up inside the Azure ML studio. You can make REST API calls to score data interactively.

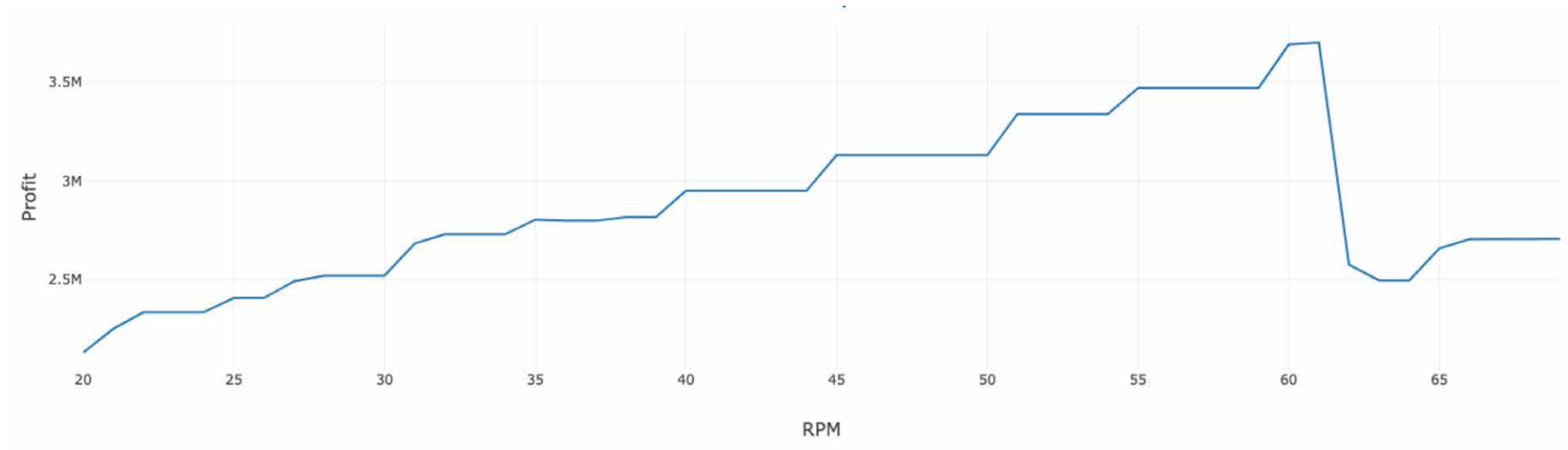
```
# Construct a payload to send with the request
payload = {
   'angle':12,
   'rpm':10,
   'temperature':25,
   'humidity':50,
   'windspeed':10,
   'power':200,
   'age':10
}

def score_data(uri, payload):
 rest_payload = json.dumps({"data": [list(payload.values())]})
 response = requests.post(uri, data=rest_payload, headers={"Content-Type": "application/json"})
 return json.loads(response.text)

print(f'Predicted power (in kwh) from model: {score_data(power_uri, payload)}')
print(f'Predicted remaining life (in days) from model: {score_data(life_uri, payload)}')
```

Now that both the power optimization and the RUL models are deployed as prediction services, we can utilize both in optimizing net profit from each wind turbine.

Assuming $1 per kWh, annual revenue can simply be calculated by multiplying the expected hourly power by 24 hours and 365 days.

The annual cost can be calculated by multiplying the daily revenue by the number of times the turbine needs to be maintained in a year (365 days / remaining life).

We can iteratively score various operating parameters simply by making multiple calls to our models hosted in Azure ML. By visualizing the expected profit cost for various operating parameters, we can identify the optimal RPM to maximize profit.



With Azure Databricks IIoT data analytics, you can iteratively score various operating parameters by calling the models hosted in Azure ML. The resulting visual analytic of the expected profit cost for various parameters — like the one shown here — can help to identify the optimal RPM to maximize profit.

Data Serving: Azure Data Explorer and Azure Synapse Analytics

Operational Reporting in ADX

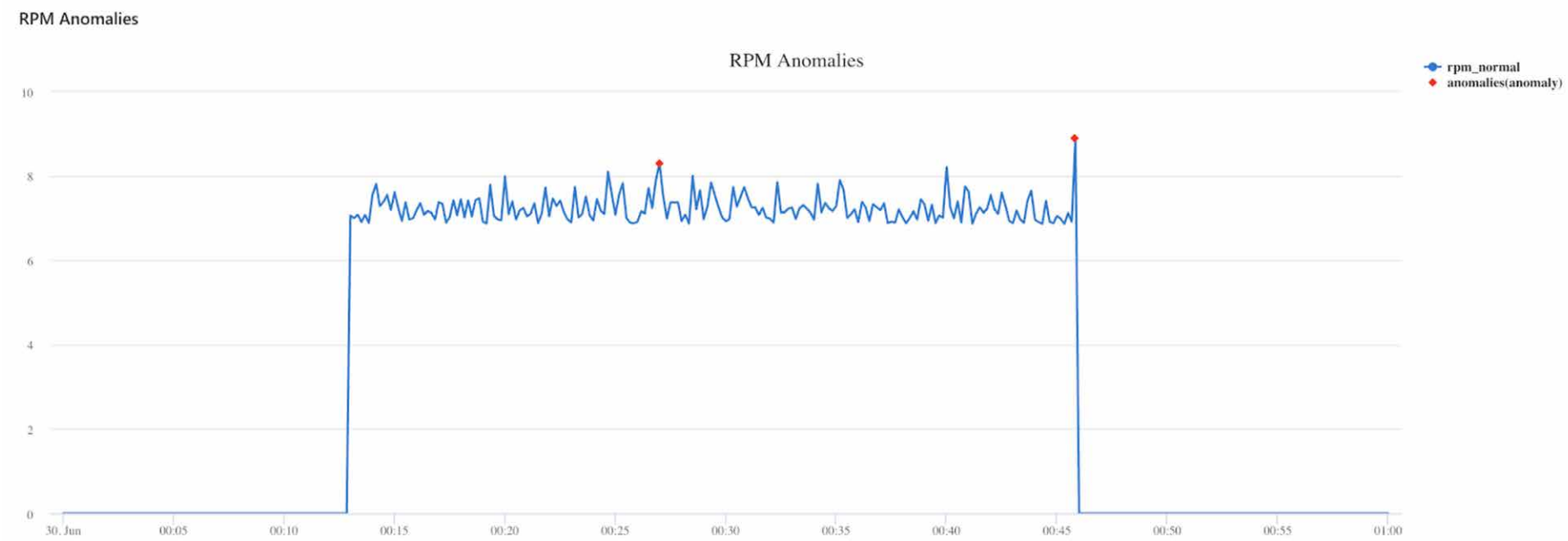
Azure Data Explorer (ADX) provides real-time operational analytics on streaming time-series data. Data can be streamed directly into ADX from IoT Hub or pushed from Azure Databricks using the Kusto Spark Connector from Microsoft, as shown on the next page.

```
stream_to_adx = (
   spark.readStream.format('delta').option('ignoreChanges',True).table('turbine_enriched')
   .writeStream.format("com.microsoft.kusto.spark.datasink.KustoSinkProvider")
   .option("kustoCluster",kustoOptions["kustoCluster"])
   .option("kustoDatabase",kustoOptions["kustoDatabase"])
   .option("kustoTable", kustoOptions["kustoTable"])
   .option("kustoAadAppId",kustoOptions["kustoAadAppId"])
   .option("kustoAadAppSecret",kustoOptions["kustoAadAppSecret"])
   .option("kustoAadAuthorityID",kustoOptions["kustoAadAuthorityID"])
)
```

Power BI can then be connected to the Kusto table to create a true, real-time, operational dashboard for turbine engineers.

ADX also contains native time-series analysis functions such as forecasting and anomaly detection. For example, the Kusto code below finds anomalous points for RPM readings in the data stream.

```
turbine_raw
| where rpm > 0
| make-series rpm_normal = avg(rpm) default=0 on todatetime(timestamp) in
range(datetime(2020-06-30  00:00:00), datetime(2020-06-30 01:00:00), 10s)
| extend anomalies = series_decompose_anomalies(rpm_normal, 0.5)
| render anomalychart with(anomalycolumns=anomalies, title="RPM Anomalies")
```

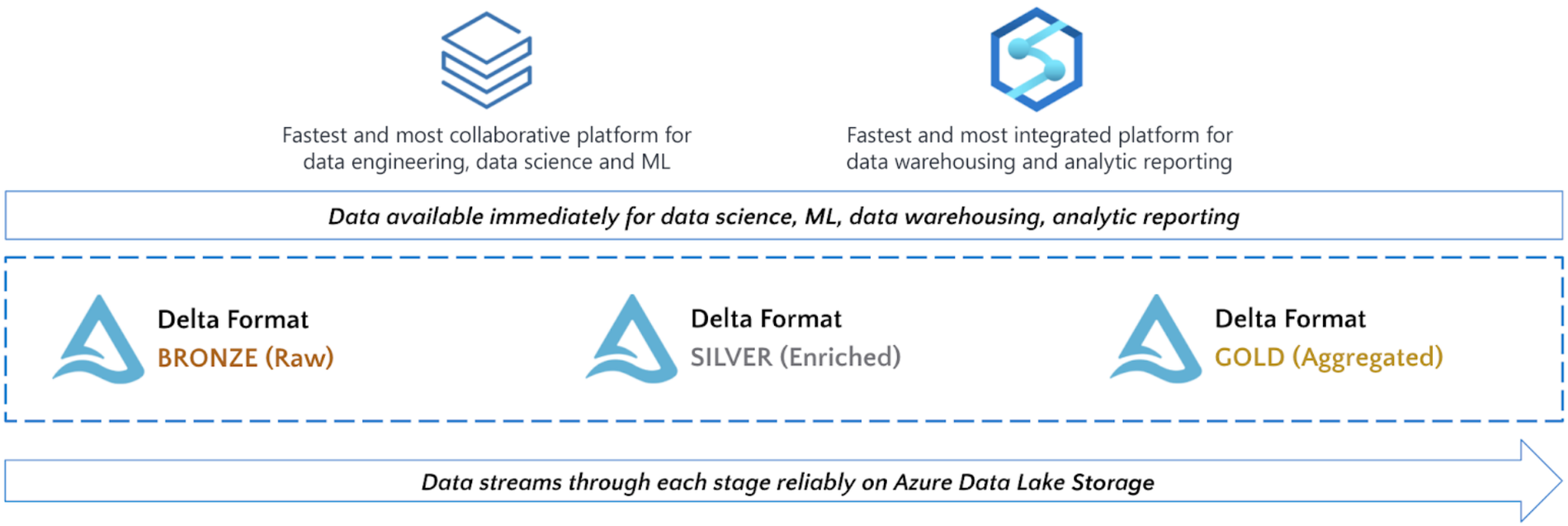


With Azure Databricks IIoT data analytics, data can be streamed directly into ADX from IoT Hub or pushed from Azure Databricks to generate real-time operational analytics on streaming time-series data.

databricks

Analytical Reporting in Azure Synapse Analytics

Azure Synapse Analytics is the next-generation data warehouse from Azure that natively leverages ADLS Gen 2 and integrates with Azure Databricks to enable seamless data sharing between these services.



While leveraging Azure Databricks and Azure Synapse, use the best tool for the job given your team's requirements.

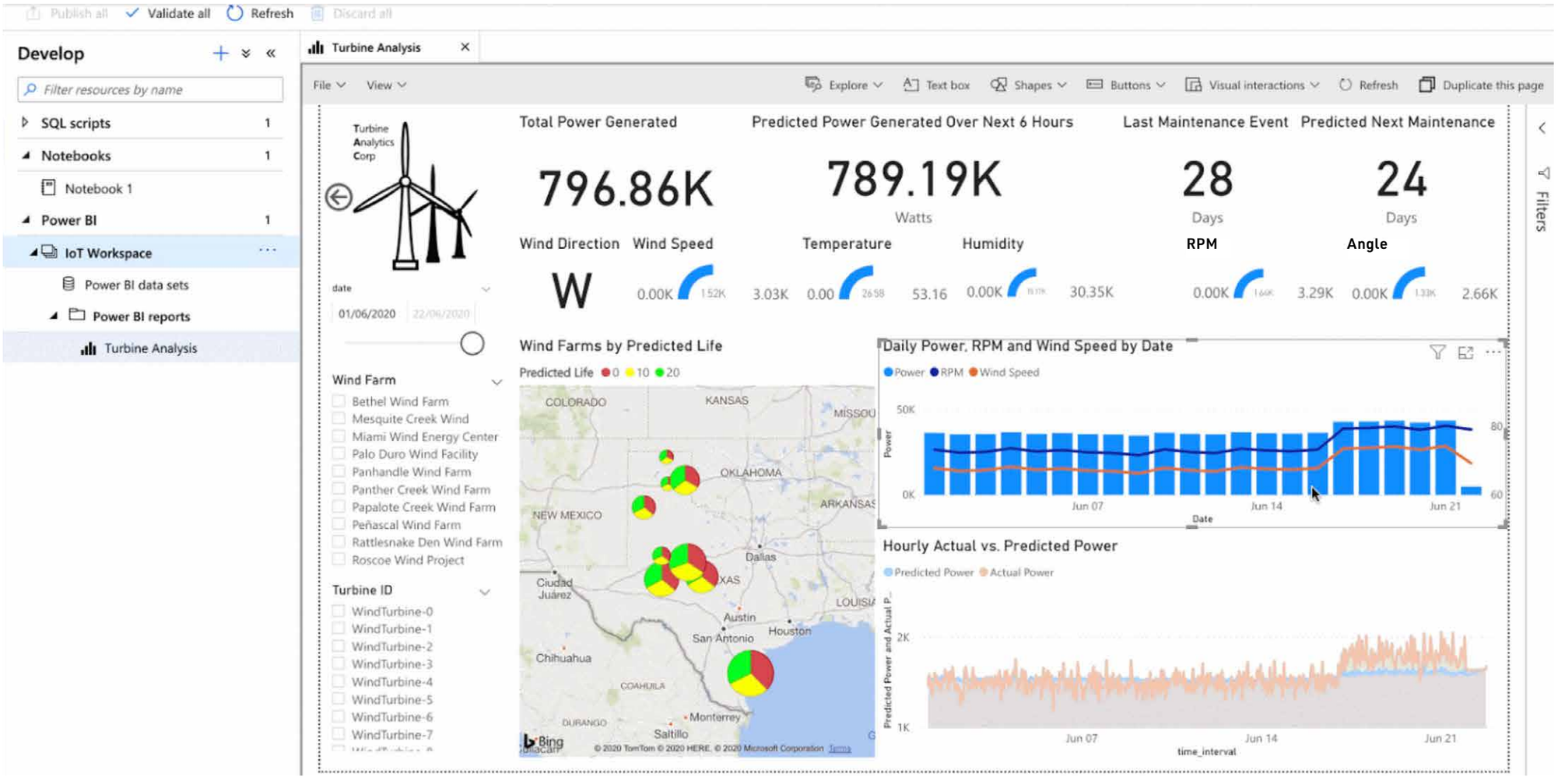
While leveraging the capabilities of Synapse and Azure Databricks, the recommended approach is to use the best tool for the job given your team's requirements and the user personas accessing the data. For example, data engineers that need the performance benefits of Delta, as well as data scientists that need a collaborative, rich and flexible workspace, will gravitate toward Azure Databricks. Analysts that need a low-code or data warehouse-based SQL environment to ingest, process and visualize data will gravitate toward Synapse.

The Synapse streaming connector for Azure Databricks allows us to stream the Gold turbine readings directly into a Synapse SQL Pool for reporting.

```
spark.conf.set("spark.databricks.sqldw.writeSemantics", "copy") # Use COPY INTO for faster loads

write_to_synapse = (
    # Read in Gold turbine readings
    spark.readStream.format('delta').option('ignoreChanges',True).table('turbine_enriched')
    .writeStream.format("com.databricks.spark.sqldw") # Write to Synapse
    .option("url",dbutils.secrets.get("iot","synapse_cs")) # SQL Pool JDBC (SQL Auth)
    .option("tempDir", SYNAPSE_PATH) # Temporary ADLS path
    .option("forwardSparkAzureStorageCredentials", "true")
    .option("dbTable", "turbine_enriched") # Table in Synapse to write to
    .option("checkpointLocation", CHECKPOINT_PATH+"synapse")
    .start()
)
```

Alternatively, Azure Data Factory can be used to read data from the Delta format and write it into Synapse SQL Pools. More documentation can be found here.

Now that the data is clean, processed and available to data analysts for reporting, we can build a live Power BI dashboard against the data, as shown below.



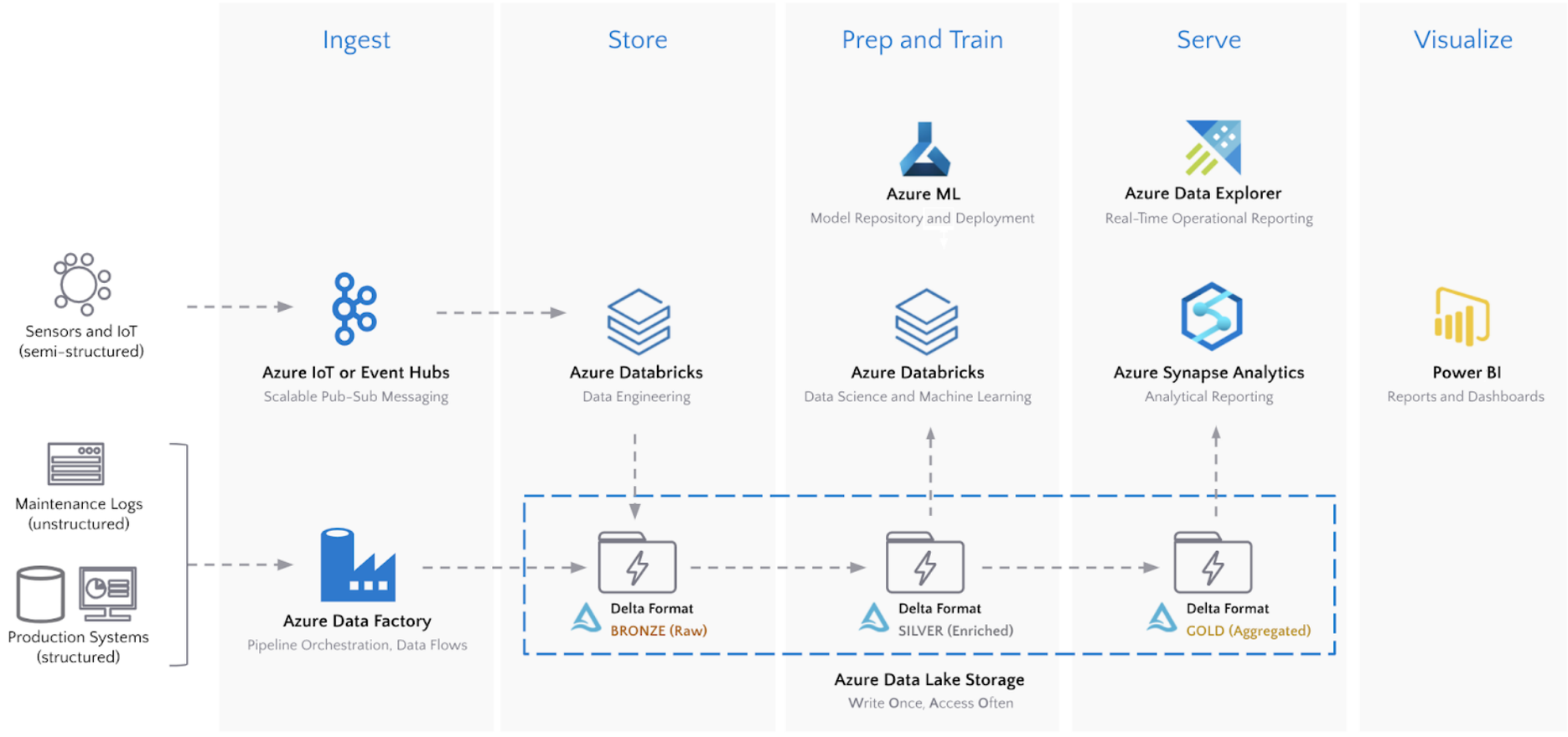
With Azure Databricks IIoT data analytics, you can use the output to generate powerful real-time BI dashboards.

databricks

Summary

To summarize, we have successfully:

- Ingested real-time IIoT data from field devices into Azure
- Performed complex time-series processing directly on a data lake
- Trained and deployed ML models to optimize the utilization of our wind turbine assets
- Served the data to engineers for operational reporting and data analysts for analytical reporting



The IIoT data analytic architecture featuring the Azure Data Lake Storage and Delta storage format offers data teams the optimal platform for handling time-series streaming data.

The key big data technology that ties everything together is Delta Lake. Delta on ADLS provides reliable streaming data pipelines and highly performant data science and analytics queries on massive volumes of time-series data. Lastly, it enables organizations to adopt a truly modern analytics pattern by bringing best-of-breed Azure tools to a write-once, access-often data store.

databricks

What's Next?

Learn more about Azure Databricks with this three-part training series, see how to create modern data architectures on Azure by attending Azure Databricks events, then try out the data engineering notebook and machine learning notebooks.

Databricks is the data and AI company. More than 5,000 organizations worldwide — including Comcast, Condé Nast, H&M and over 40% of the Fortune 500 — rely on the Databricks Lakehouse Platform to unify their data, analytics and AI. Databricks is headquartered in San Francisco, with offices around the globe. Founded by the original creators of Apache Spark™, Delta Lake and MLflow, Databricks is on a mission to help data teams solve the world's toughest problems.

START YOUR FREE TRIAL

To learn more, visit us at:
databricks.com/azure

databricks