

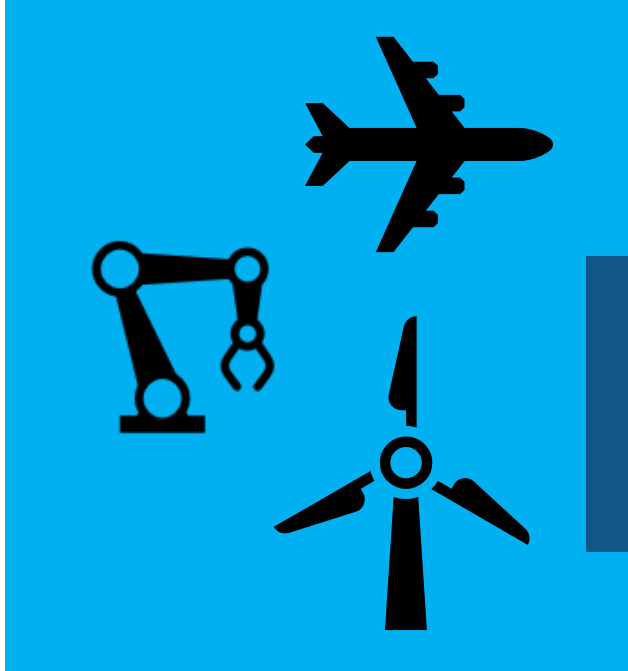
MATLAB EXPO 2018

**Deploying AI for Near
Real-Time Manufacturing
Decisions**

Arvind Hosagrahara
Heather Gorr, PhD



The Need for Large-Scale Streaming



Predictive Maintenance

Increase Operational Efficiency
Reduce Unplanned Downtime

**More applications require
near real-time analytics**

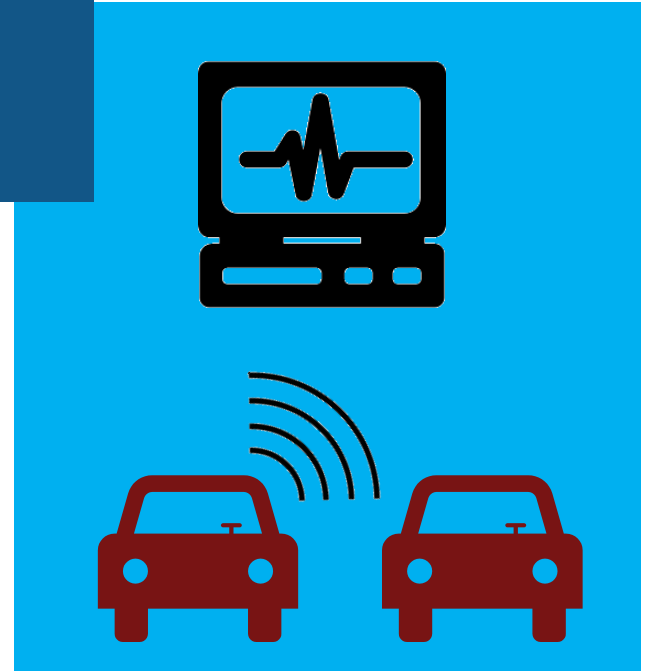
Jet engine: ~800TB per day
Turbine: ~ 2 TB per day

Medical Devices

Patient Safety
Better Treatment Outcomes

Connected Cars

Safety, Maintenance
Advanced Driving Features



Car: ~25 GB per hour

Example Problem: Develop and operationalize a machine learning model to predict failures in industrial pumps

The current system requires the operator to manually monitor the operational metrics for anomalous conditions. It is dependent on experience and expertise to detect and take preventative action



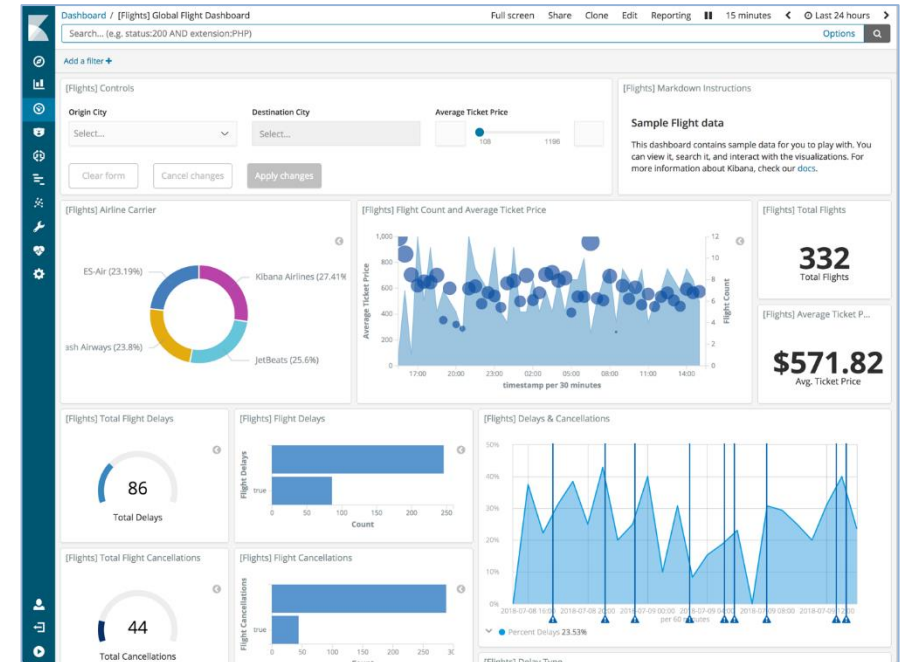
Process Engineer
Develops machine learning model in MATLAB

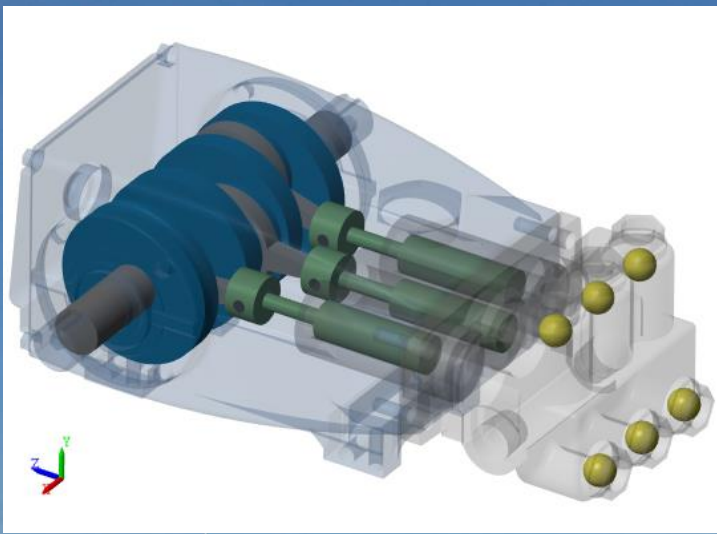


System Architect
Deploy and operationalize model on Azure cloud



Operator
Makes operational decisions based on model output





Baker Hughes Develops Predictive Maintenance Software for Gas and Oil Extraction Equipment Using Data Analytics and Machine Learning

Challenge

Develop a predictive maintenance system to reduce pump equipment costs and downtime

Solution

Use MATLAB to analyze nearly one terabyte of data and create a neural network that can predict machine failures before they occur

Results

- Savings of more than \$10 million projected
- Development time reduced tenfold
- Multiple types of data easily accessed



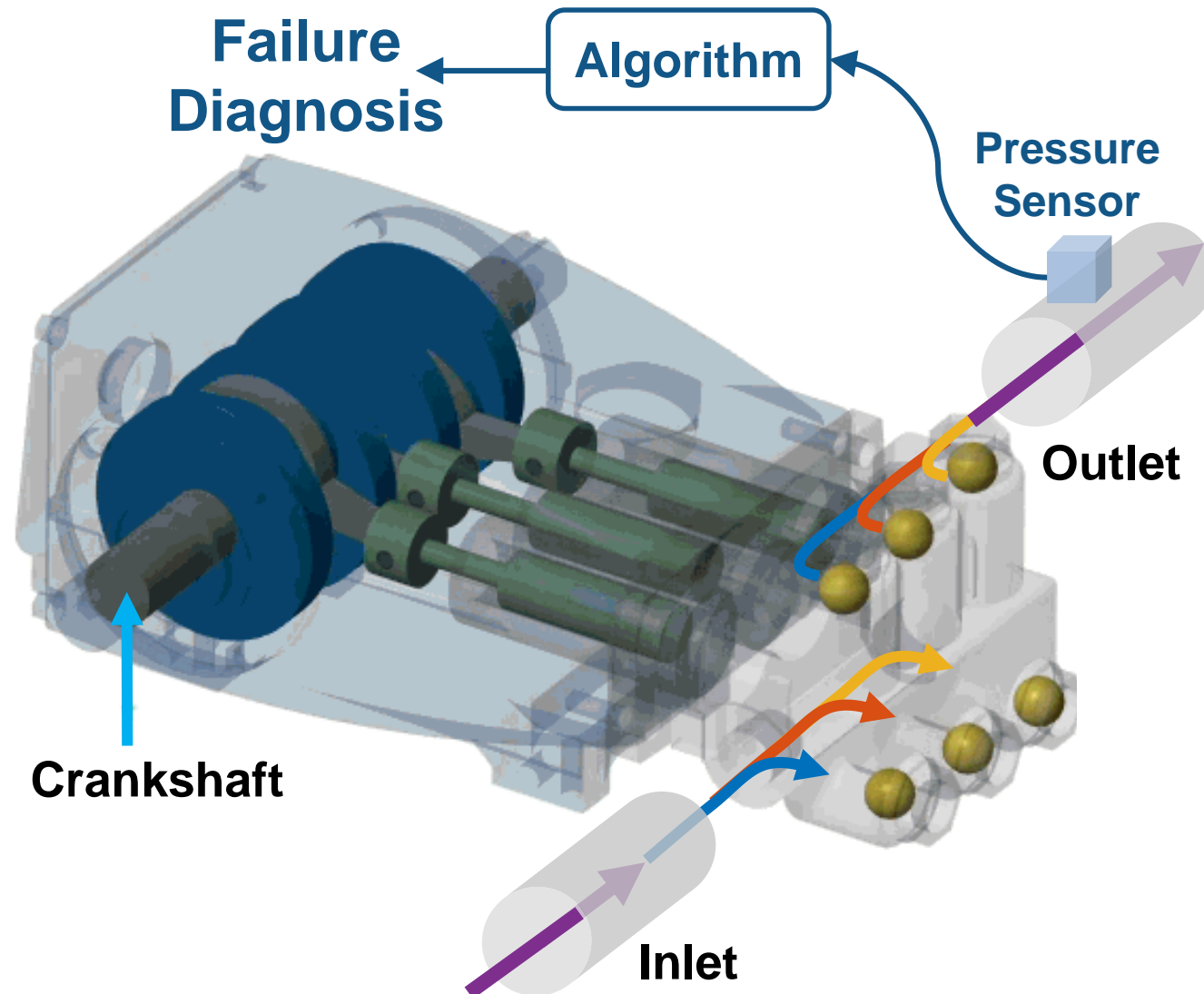
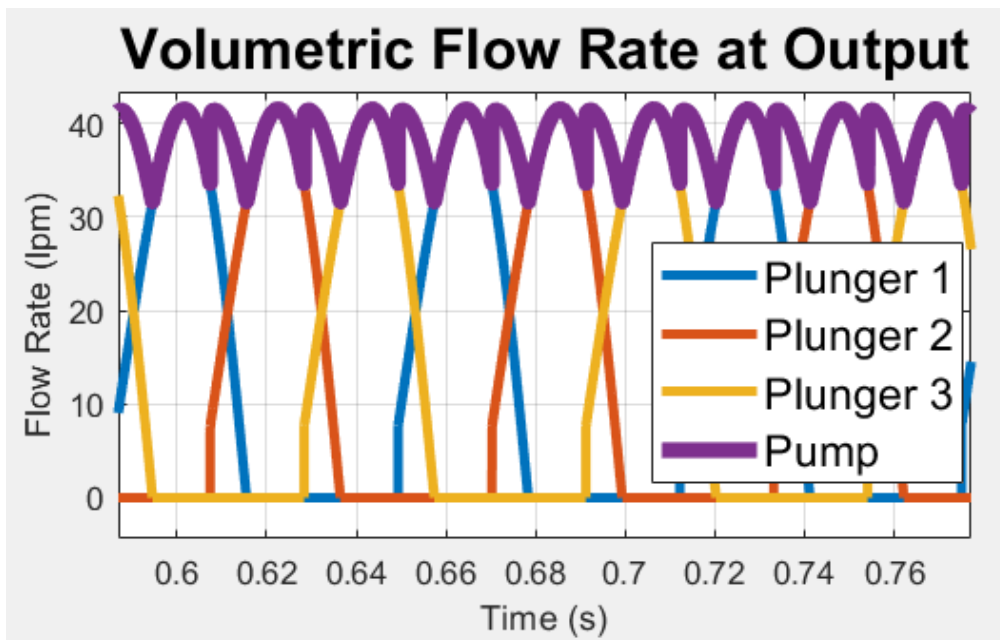
Truck with positive displacement pump.

“MATLAB gave us the ability to convert previously unreadable data into a usable format; automate filtering, spectral analysis, and transform steps for multiple trucks and regions; and ultimately, apply machine learning techniques in real time to predict the ideal time to perform maintenance.”

- Gulshan Singh, Baker Hughes

Triplex Pump

- Crankshaft drives three plungers
 - Each 120 degrees out of phase
 - One chamber always discharging
 - Smoother flow than single or duplex piston pumps



Project statement: Develop a full end-to-end predictive maintenance system and demo with the plant operator in one 3-4 week sprint



Plant
Operator

1. I would like to monitor the *flow*, *pressure*, and *current* of each pump in my plant so that I always know their *operational state*
2. I need to be *alerted* when any pump's operational parameters drift outside of an acceptable range so that I can take *immediate corrective action*
3. I would like to get a continuous estimate of each pump's *remaining useful life (RUL)* so that I can *schedule maintenance or replacement* of the asset

Project constraints and solutions



Process Engineer

We don't have a large set of failure data, and it's too costly to generate real failures in our plant for this project

Solution: Use an accurate physics-based software model for the pump to develop synthetic training sets



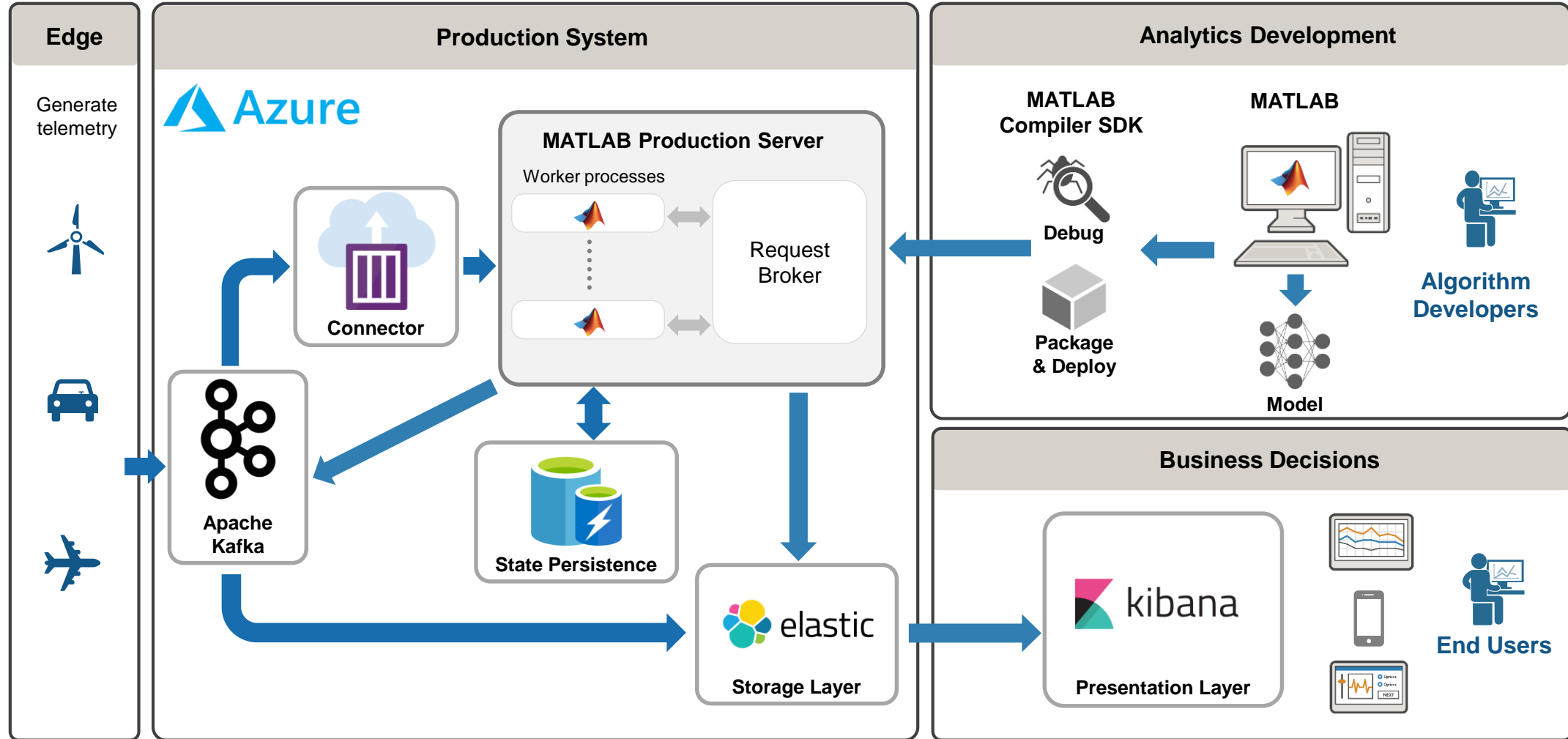
System Architect

We don't have a large IT/hardware budget, and we need to see results before committing to a particular platform or technology

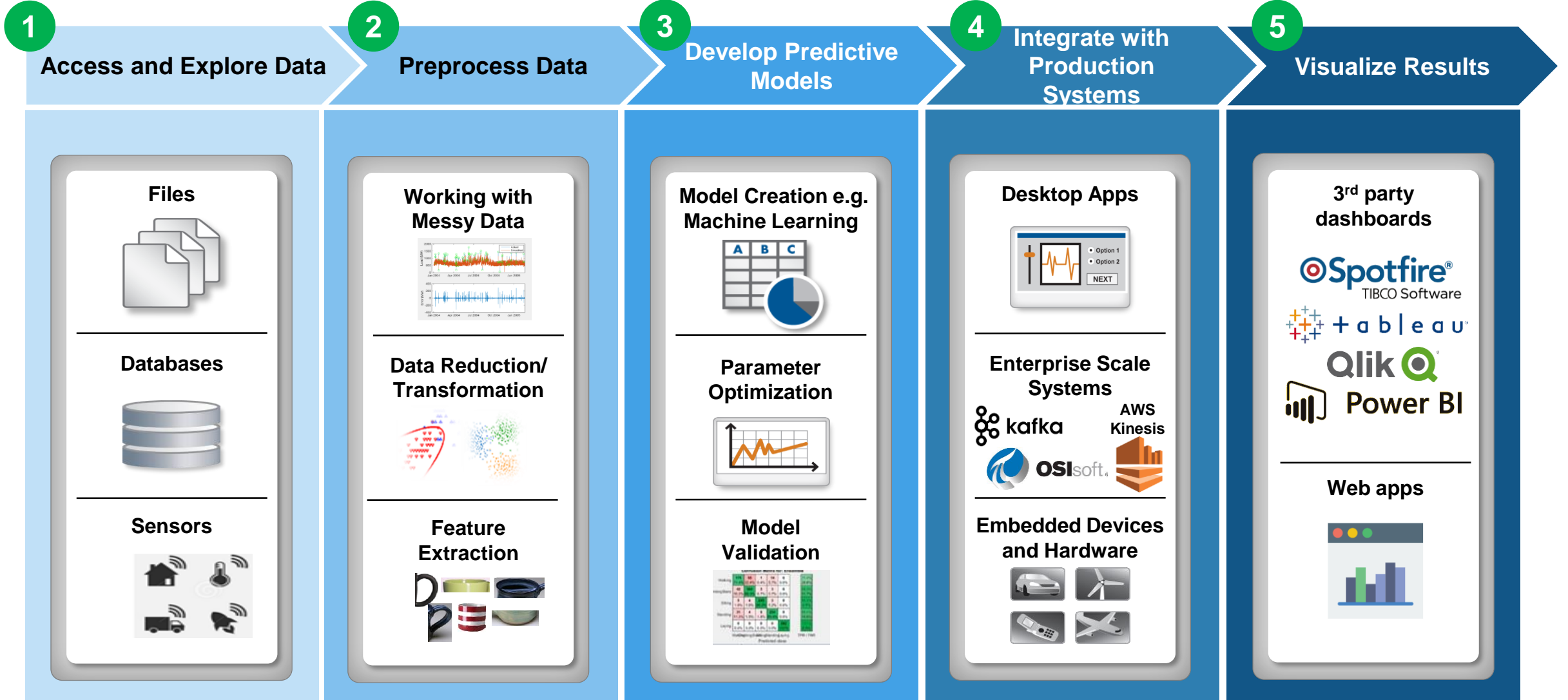
Solution: Leverage a cloud platform that enables us to quickly configure and provision the services we need to build the solution, while minimizing lock-in to a particular provider



Predictive Maintenance Architecture on Azure



Agenda



4 Integrate with Production Systems

Review model requirements

- Requirements from the Operator
 - Continuous predictions of:
 - Fault values
 - Leak cylinder area
 - Block in factor
 - Bearing friction
 - Type of fault (“Blocking”, “Leaking”, “All”)
 - Remaining Useful Life (RUL)
- Requirements from System Architect
 - Define window for streaming
 - Define format of results, intermediate values
 - Scalable code
 - Test code



Operator

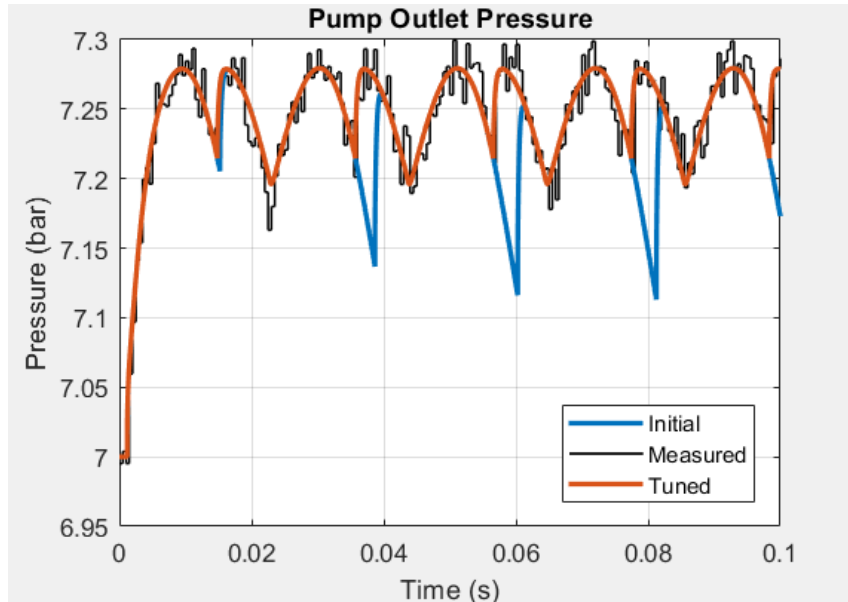


System Architect

1

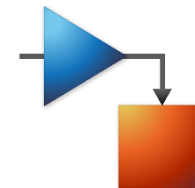
Access and Explore Data

Use sensor data from pump to identify levels of failure



Pump sensor data

The image shows a Simulink workspace with a pump model and a diagnostic display. The diagnostic display, titled "pMeas", has "Diagnostics: On" and shows four status indicators: "No Fault", "Blocked Inlet", "Seal Leak", and "Worn Bearing". The "Seal Leak" indicator is highlighted in yellow, indicating an active fault. The background shows a Simulink window titled "sm_pump_triplex" with a block diagram of a pump and a scope window titled "Pressure with Noise" showing a noisy signal.



Simulate failure data

1

Access and Explore Data

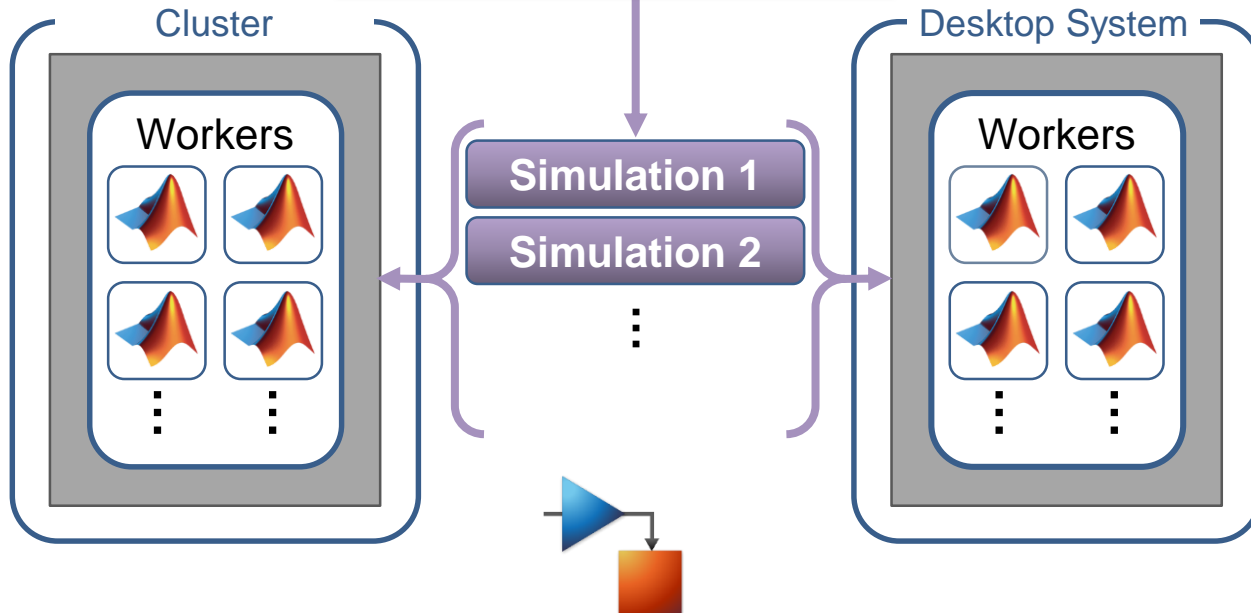
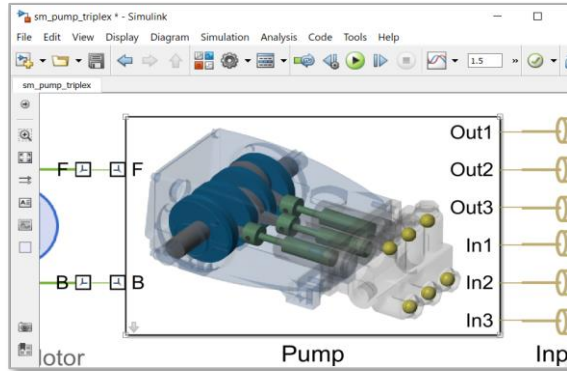
Build digital twin and generate sensor data

The image displays a Simulink model of a pump system. On the left, a 'Driver: Motor' block is connected to a 'Pump' block. The pump block is highlighted with a blue box and labeled with 'Out1', 'Out2', 'Out3', 'In1', 'In2', and 'In3'. A 3D model of the pump assembly is shown to the right of the motor block. Below the Simulink model, a 'Pressure with Noise' window is open, showing a graph of 'Virtual Sensor Data'. The graph plots pressure over time, with a legend indicating 'Sampled with Noise' (red line) and 'Simulation' (blue line). The y-axis ranges from 7.0 to 7.35, and the x-axis ranges from 0 to 1.5. Below the graph, a 'Diagnostics: On' section shows four blue circles representing different fault types: 'No Fault', 'Blocked Inlet', 'Seal Leak', and 'Worn Bearing'. A 'pMeas' label points to a blue circle, and a 'pMeas' label points to a blue circle. The status bar at the bottom shows 'Running', 'Sample based', 'T=0.006', '211%', 'T=0.012', '0%', and 'auto(ode23t)'.

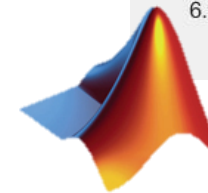
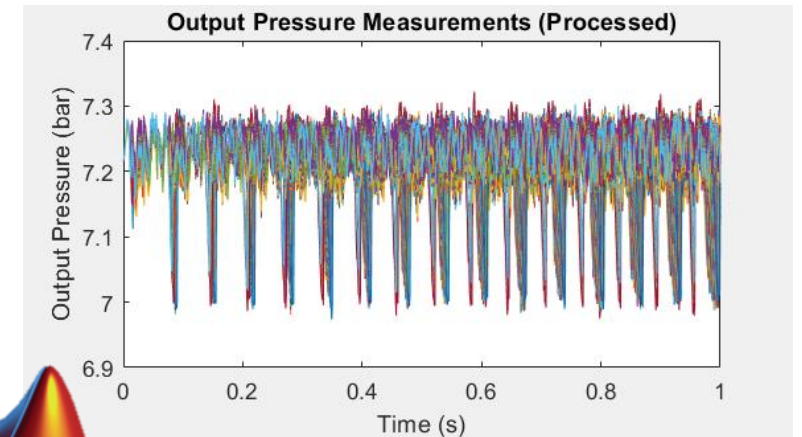
1

Access and Explore Data

Simulate data with many failure conditions



Run parallel simulations

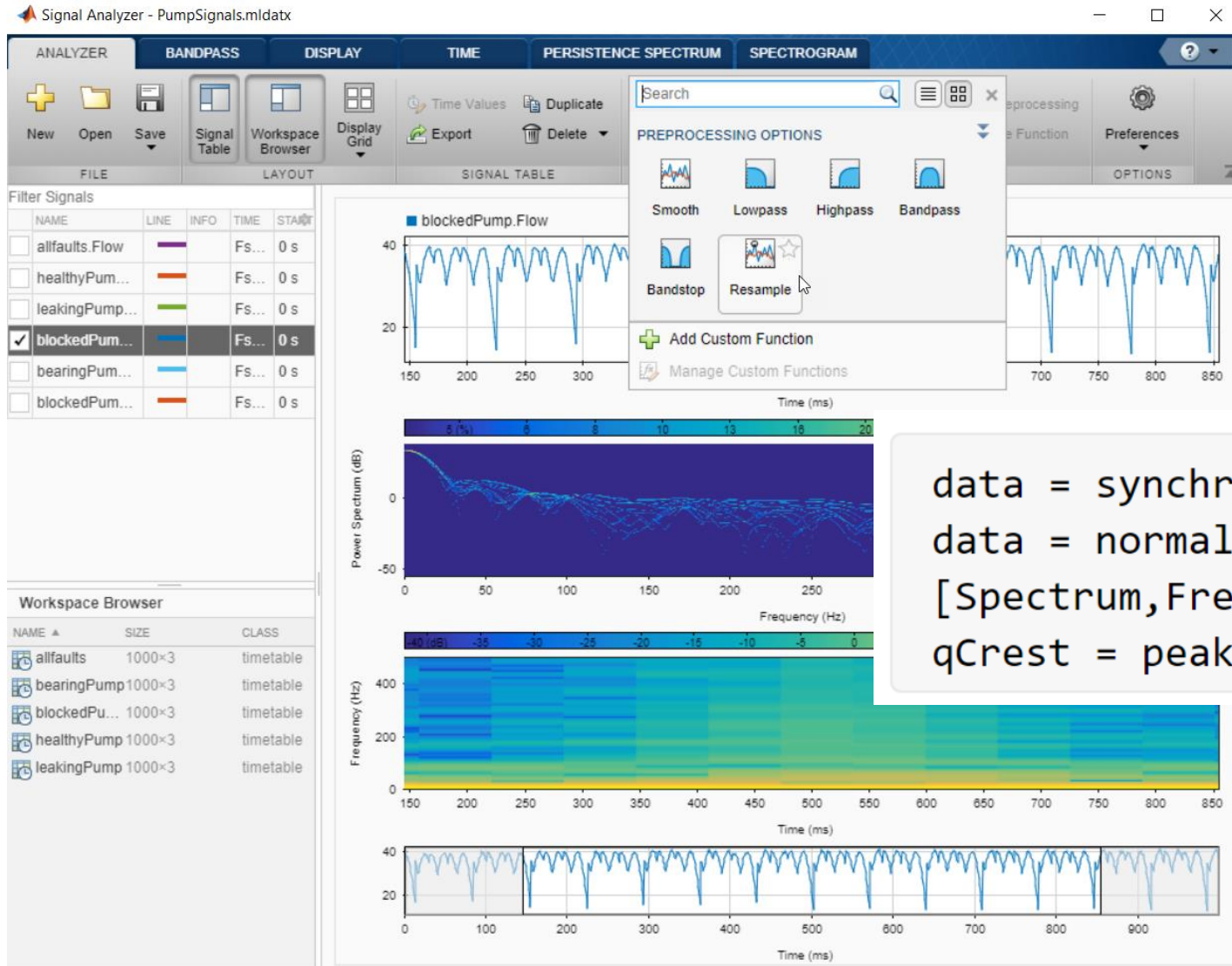


Store data on HDFS and leverage
MATLAB + Spark for machine learning

2

Preprocess Data

Preprocess data and represent signal information

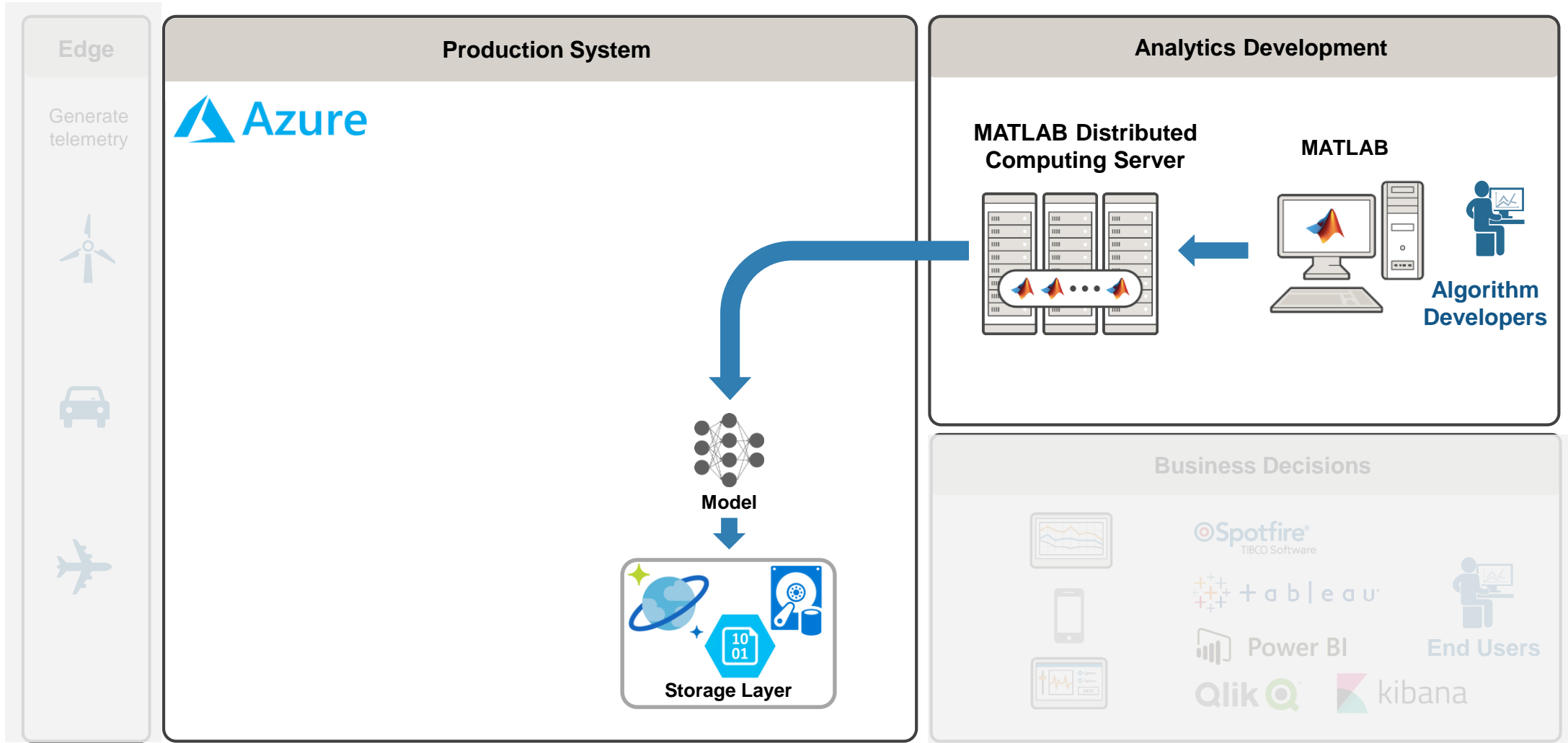


```
data = synchronize(Flow,Pressure,t,'linear');
data = normalize(data,'center');
[Spectrum,Frequencies] = pspectrum(data.Flow);
qCrest = peak2rms(data.Flow);
```

3

Develop Predictive Models

Develop a Predictive Model



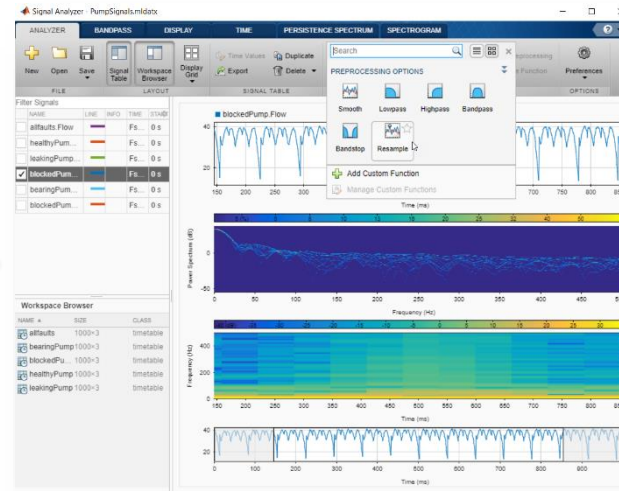
3

Develop Predictive Models

Develop a Predictive Model in MATLAB

time	1	2	3	4	5
Event	SpeedGPS	AccelerationSensorXAxis	AccelerationSensorYAxis	AccelerationSensorZAxis	
Mon May 11 04:03:15 UTC 2015	Hard Brake	10.8360	-0.6996	0.6014	0.205
Wed May 06 19:09:48 UTC 2015	Hard Brake	27.8280	0.1419	0.9035	-0.526
Sun May 17 17:09:19 UTC 2015	Hard Brake	6.5520	0.9986	-0.0761	-0.004
Fri Jan 16 20:38:37 UTC 2015	Hard Brake	39.6128	0.0999	0.8000	0.367
Sat May 02 14:00:37 UTC 2015	Hard Brake	61.1280	0.4006	-0.4022	0.663
Mon Apr 27 17:54:27 UTC 2015	Fast Accel	37.7640	0.1527	0.4666	0.857
Sun May 03 21:00:42 UTC 2015	Fast Accel	17.2440	1.0235	0.0815	0.304
Mon May 04 11:30:33 UTC 2015	Fast Accel	19.6560	0.1336	0.8932	-0.578
Wed May 20 10:20:55 UTC 2015	Hard Brake	23.4000	0.2050	0.0954	0.006

Label Faults



Represent Signals

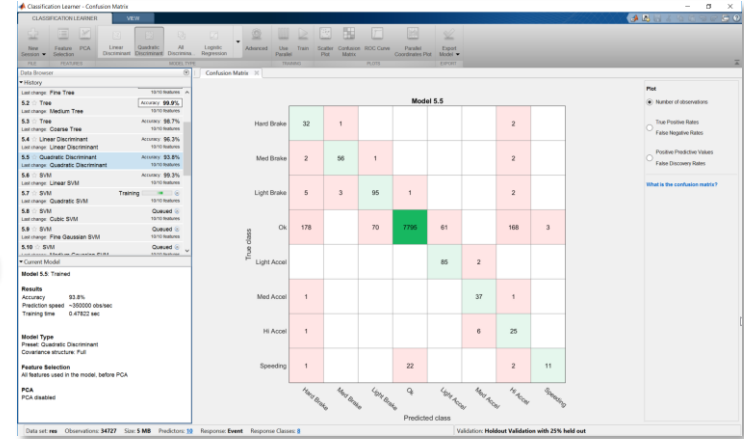
Evaluating tall expression using the Spark Cluster:
 - Pass 1 of 2: Completed in 11 sec
 - Pass 2 of 2: Completed in 2.3333 min
 Evaluation completed in 2.6167 min

Scale Up

```

Scale up
tt = tall(data); % test tall array
model = TreeBagger(50,tt,'Event');

Scale to out of memory data
tt = tall(ds);
tt = preprocessData(tt);
model = TreeBagger(50,tt,'Event');
save machineLearningModel model
    
```



Train Model

Validate Model

3

Develop Predictive
Models

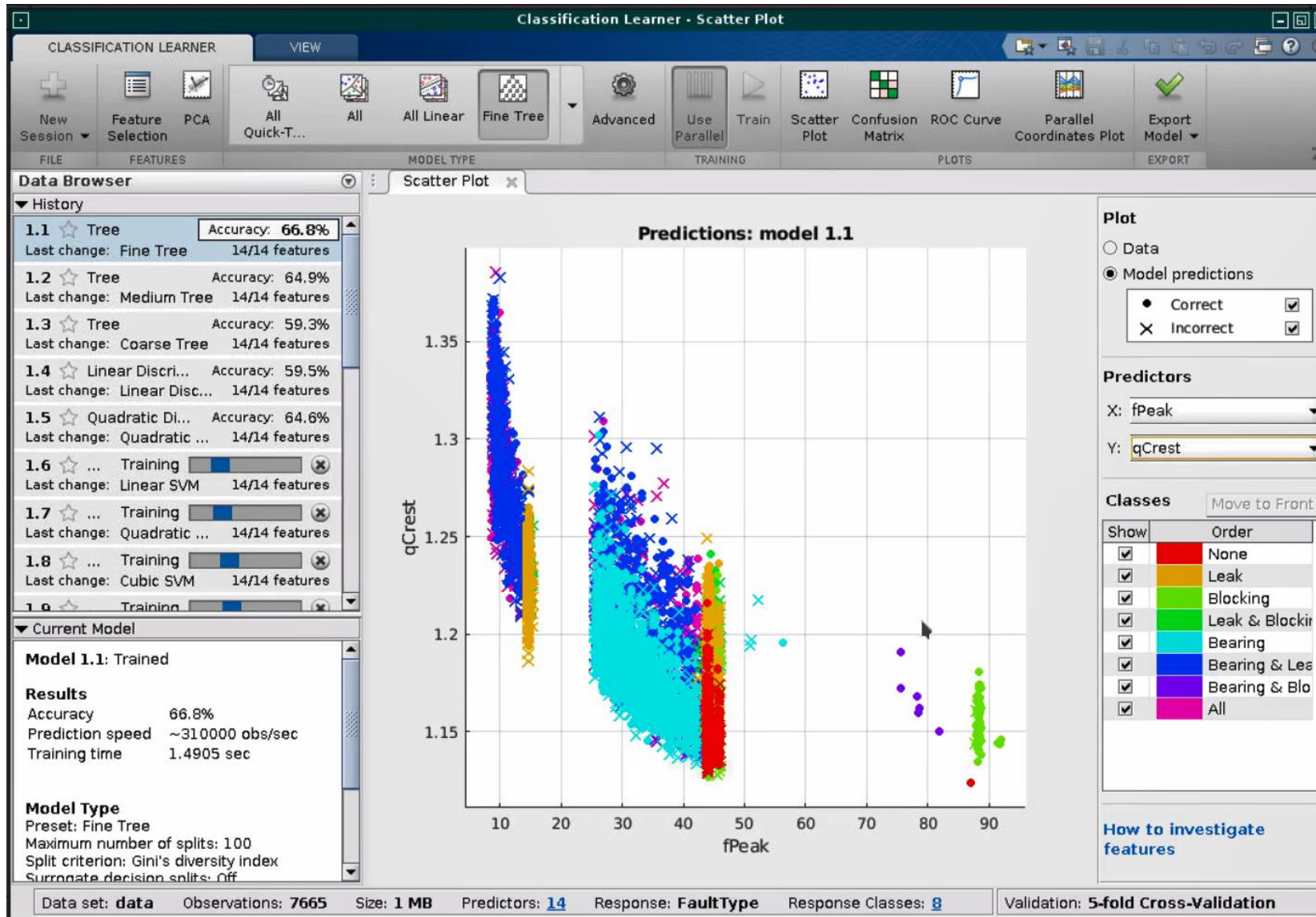
Develop a Predictive Model in MATLAB

- Predict fault values (Regression)
- Predict type of fault (Classification)
- Predict remaining useful life (Exponential degradation)
- Use 1 second of data for predictions

3

Develop Predictive Models

Develop a Predictive Model in MATLAB



4

Integrate with
Production
Systems

Develop a Stream Processing Function

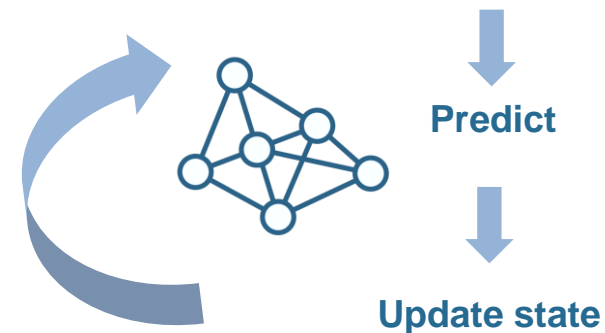
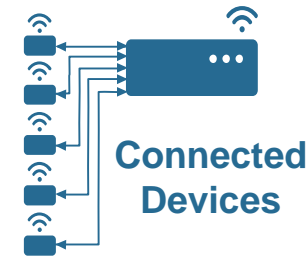
- Model is built and tested on historic data
- Model will be used on streaming data
- Retain model state and update



Files



Storage



4

Integrate with
Production
Systems

Develop a Stream Processing Function in MATLAB

Streaming Function

```
function new_state = streamingFunction(data,old_state)
```

Process each window of
data as it arrives

Preprocess signals

```
[data,features] = preprocessData(data);
```

Predict faults

```
[Leak,Blocking,Bearing] = predictFaultValues(features);  
FaultType = predictFault(features);  
[RUL,Model] = predictUpdateRUL(data.Timestamp,data.Flow,500);
```

Previous state

Current window of data to
be processed

Update state

```
new_state = updateState(data,old_state);
```

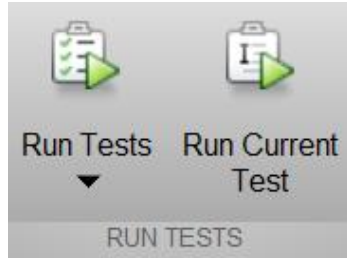
Write results

```
writeResults(Leak,Blocking,Bearing,FaultType,RUL,Model)  
end
```

4

Integrate with
Production
Systems

Test Stream Processing Function



```
results = runtests('predictFaults_tests')
```

```
Running predictFaults_tests
```

```
....
```

```
Done predictFaults_tests
```

```
results =
```

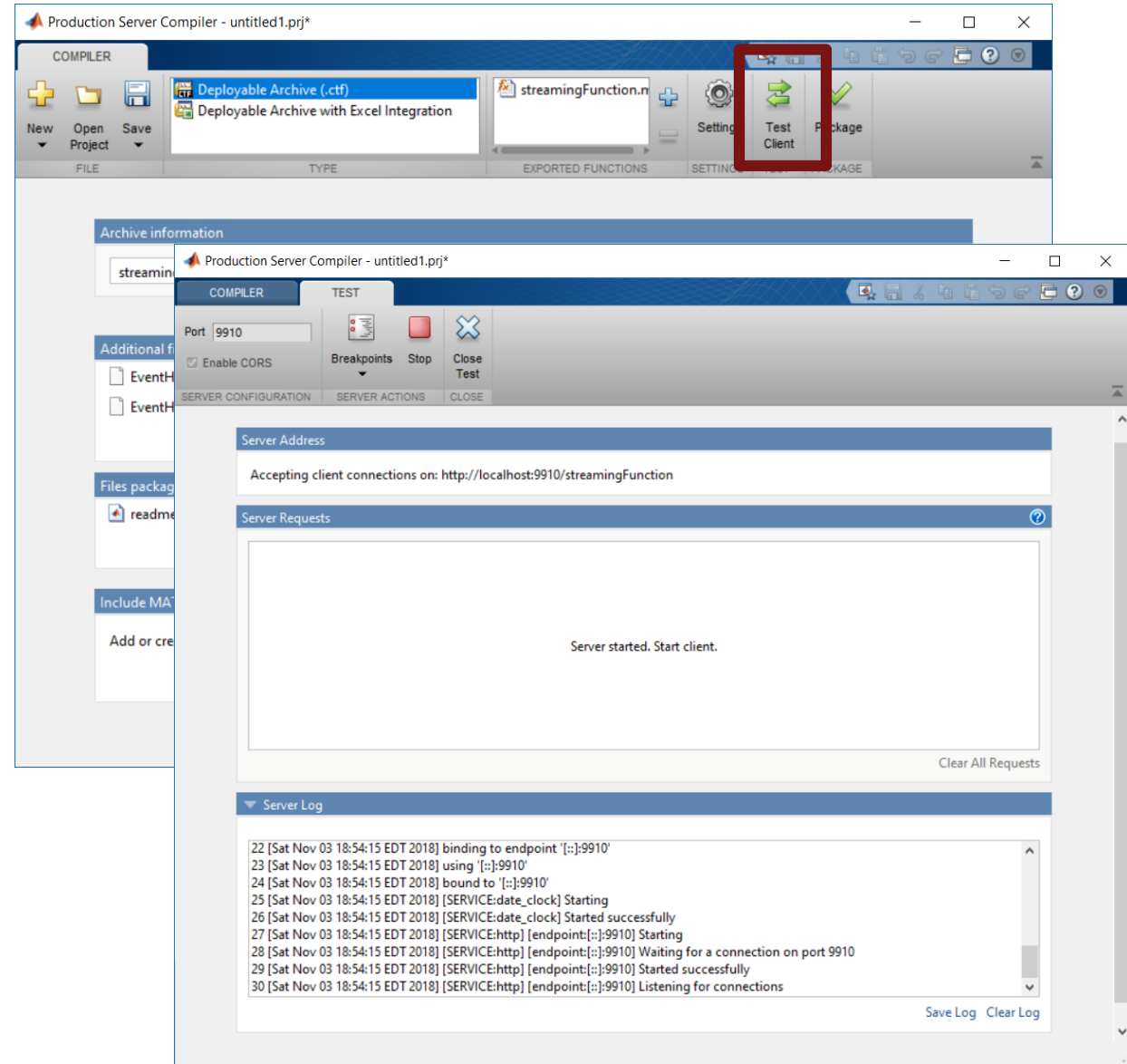
```
1x4 TestResult array with properties:
```

```
Name  
Passed  
Failed  
Incomplete  
Duration  
Details
```

```
Totals:
```

```
4 Passed, 0 Failed, 0 Incomplete.
```

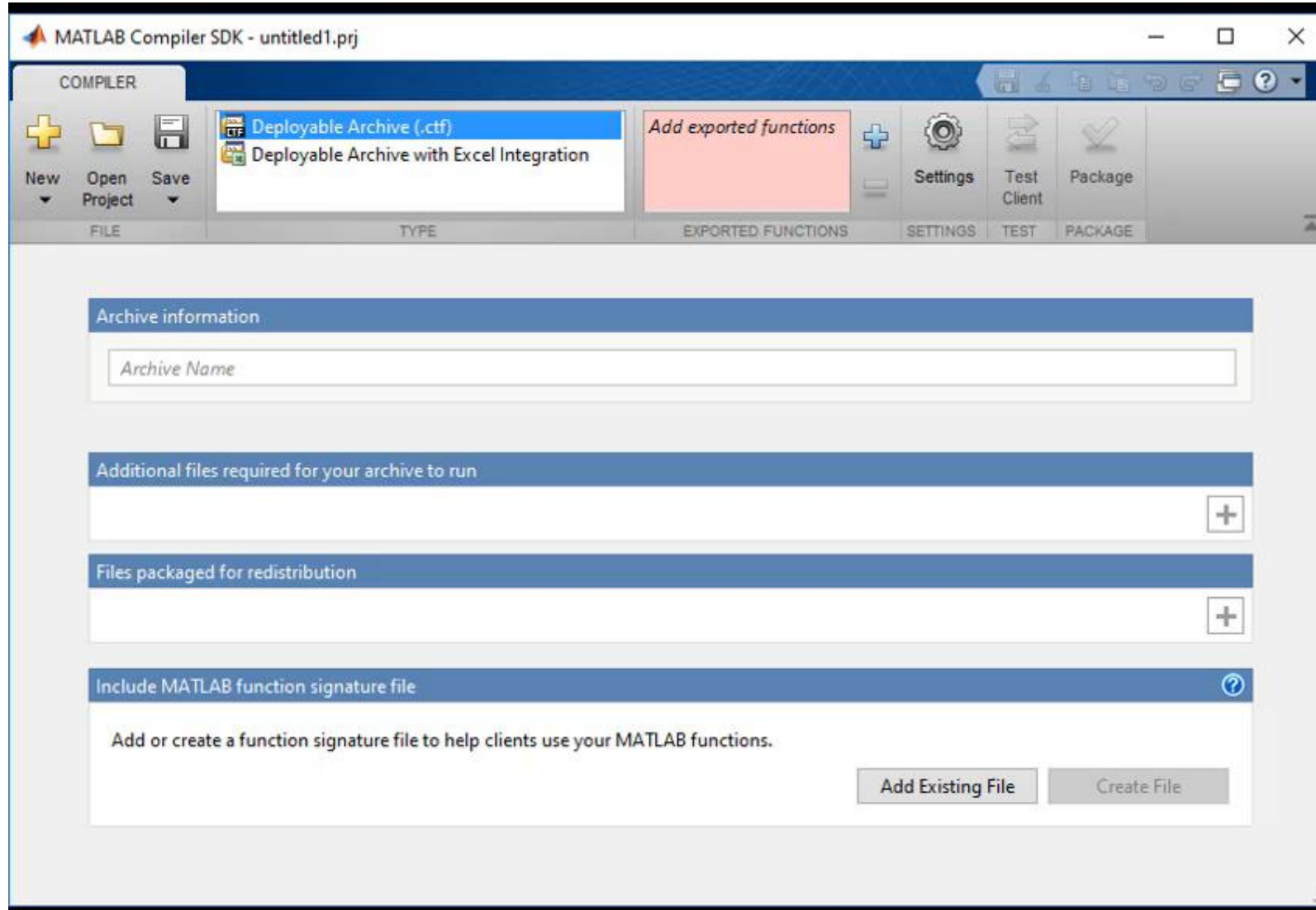
```
0.01614 seconds testing time.
```



4

Integrate with
Production
Systems

Package Stream Processing Function



4

Integrate with
Production
Systems

Share with the team

Review results with
Operator



Share code with
System Architect



Analyze and Compare Pump Data
Plot the simulated data to compare the signals based on the types of faults: leaking, blocking, and bearing friction.

```

1 stackedplot(healthyPump);
2 title('Healthy Pump');
3 stackedplot(leakingPump);
4 title('Leaking Pump');
5 stackedplot(blockedPump);
6 title('Blocked Pump');
7 stackedplot(bearingPump);
8 title('Bearing Friction');

```

From the visualizations, you can see the differences in the signal profiles for these different types of faults. The machine learning model will predict these faults based on these differences.

Significant data can be simulated with similar conditions to build these models. These are adjusted by selecting the parameters in the model:

The screenshot shows a MATLAB Live Editor window with a toolbar, a code editor containing MATLAB code for plotting pump data, and a workspace area with two plots: 'Healthy Pump' and 'Leaking Pump'. The plots show time-series data for Flow, Pressure, and Current. A schematic diagram of a pump system with two plungers is also visible.

pdf, html, LaTeX

Branches
Current Branch: master
Name: master
HEAD: 46495bc97f110c26706f9dbee71f2c2ea7c582e

Branch Browser
Branches: master

Message:
Updated port and signal names in Controllers, ControllerSensorManagement and ControllersWinglet to make common to enable root port mapping.

- Remove project_paths file from project and use 'fmacmill...
- Refactored designStudies code to make the top 'lgpacitti...
- Modified Design studies script to fix issues causegpacitti...

StreamingCode

- predictFault.m
- predictFaultValues.m
- predictUpdateRUL.m
- StreamingExample.mlx
- streamingFunction.m
- testData.mat
- test
- consume.m

Buttons: Create, Close

Source Control

Review system requirements

- Requirements from the Process Engineer
 - Every millisecond, each pump generates a time-stamped record of flow, pressure, and current
 - Model expects 1 sec. window of data per pump
 - Initially, 1's – 10's of devices, but quickly scale to 100's
- Requirements from the operator
 - Visual description of the device output for each pump
 - Alerts when parameters drift outside the expected ranges
 - Continuous estimating of RUL for each pump



Process Engineer

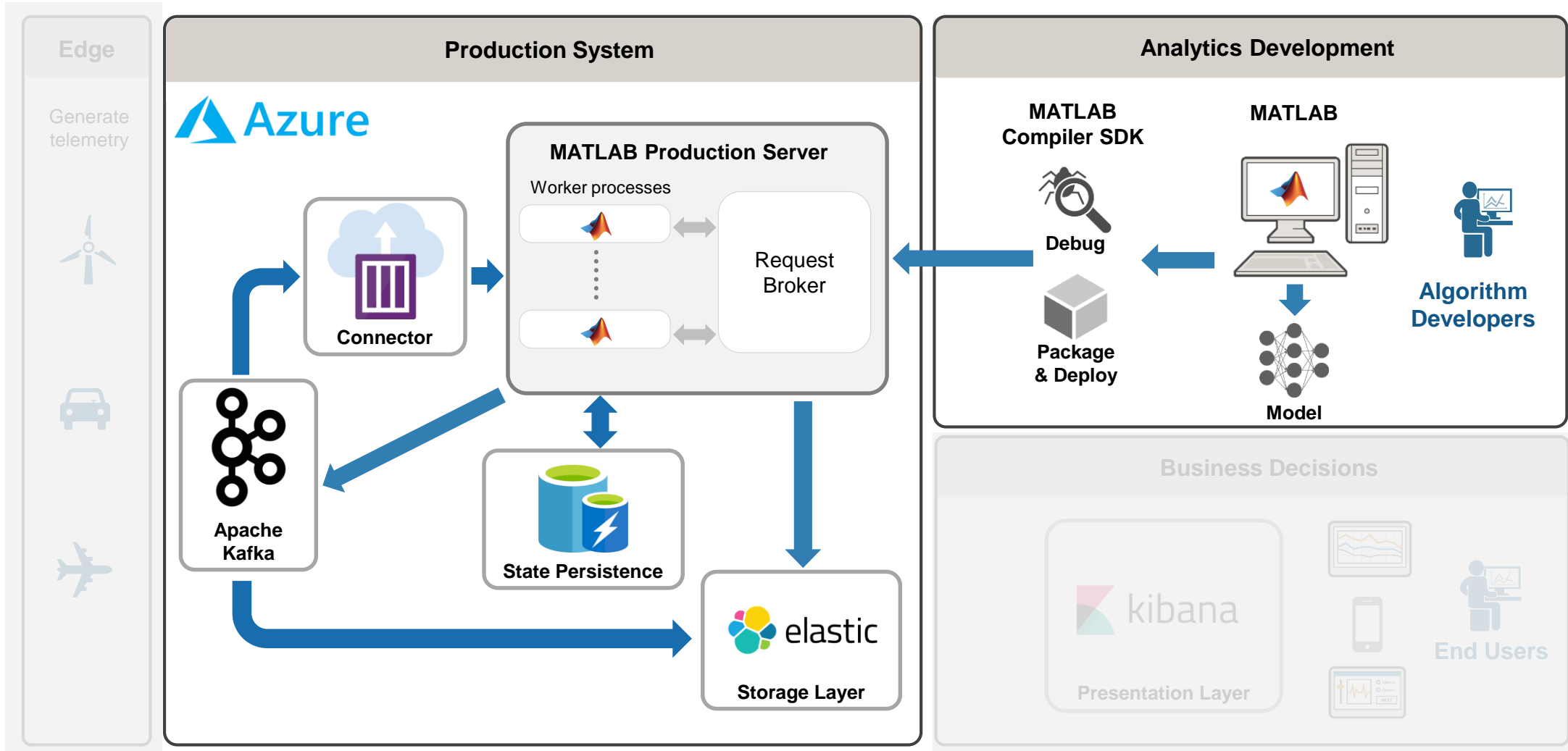


Operator

4

Integrate with
Production
Systems

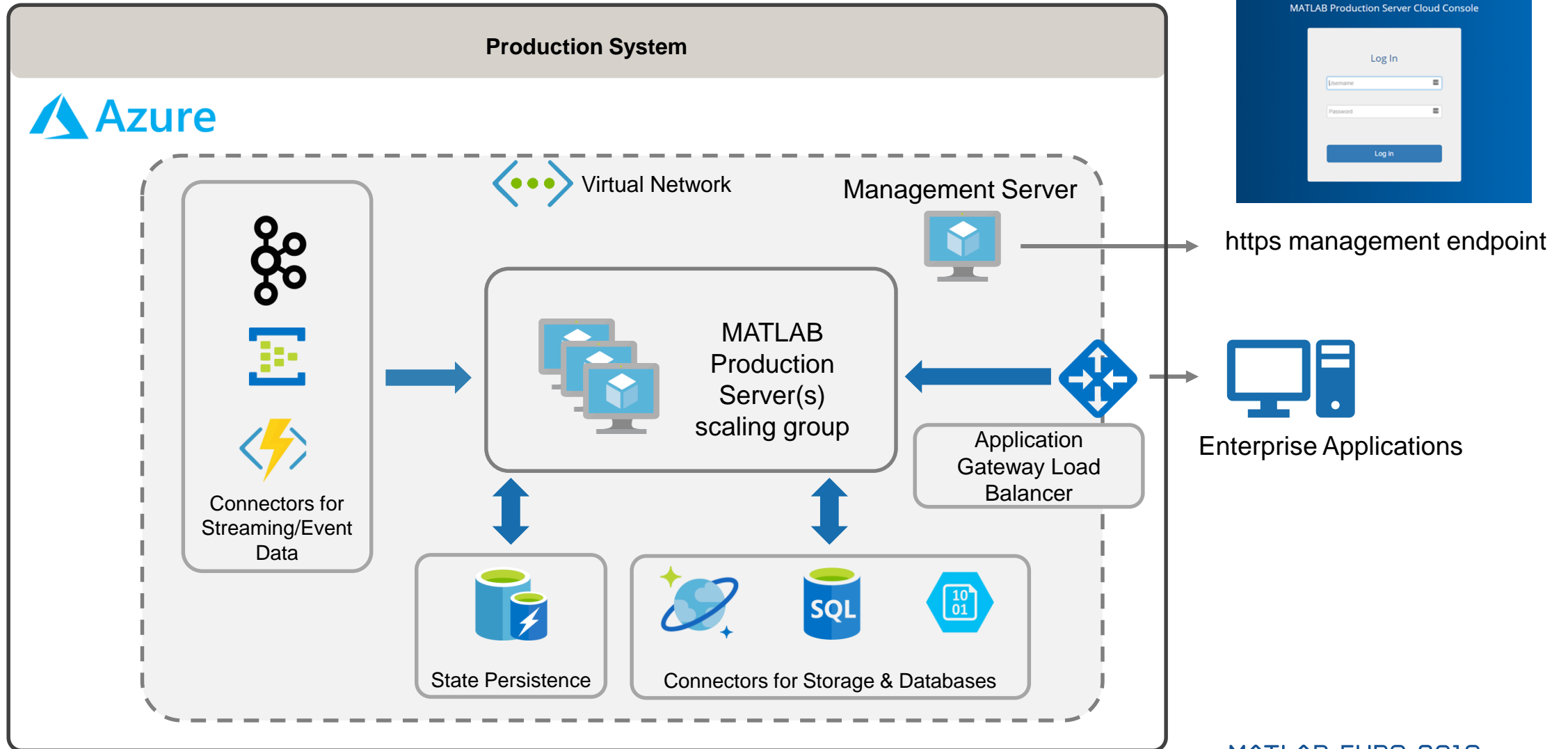
Integrate Analytics with Production Systems



4

Integrate with
Production
Systems

MATLAB Production Server on Azure



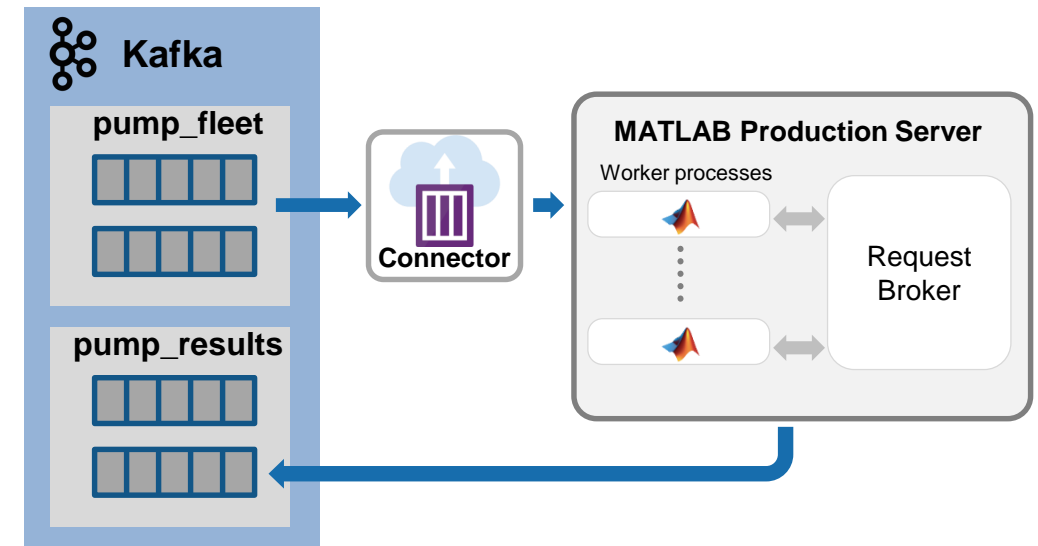
Connecting MATLAB Production Server to Kafka

Goals:

- Develop a micro-service that feeds a single Kafka topic to a MATLAB function
- Develop a publisher library for MATLAB for writing output to a results stream

- Basic Features:

- Group data into time windows and pass to MATLAB as a timetable
- Time-stamps are in “event time” and may arrive out of order
- Execute streaming function asynchronously and exploit parallel execution of Kafka partitions
- Use Kafka’s check-pointing (i.e. at-least-once)

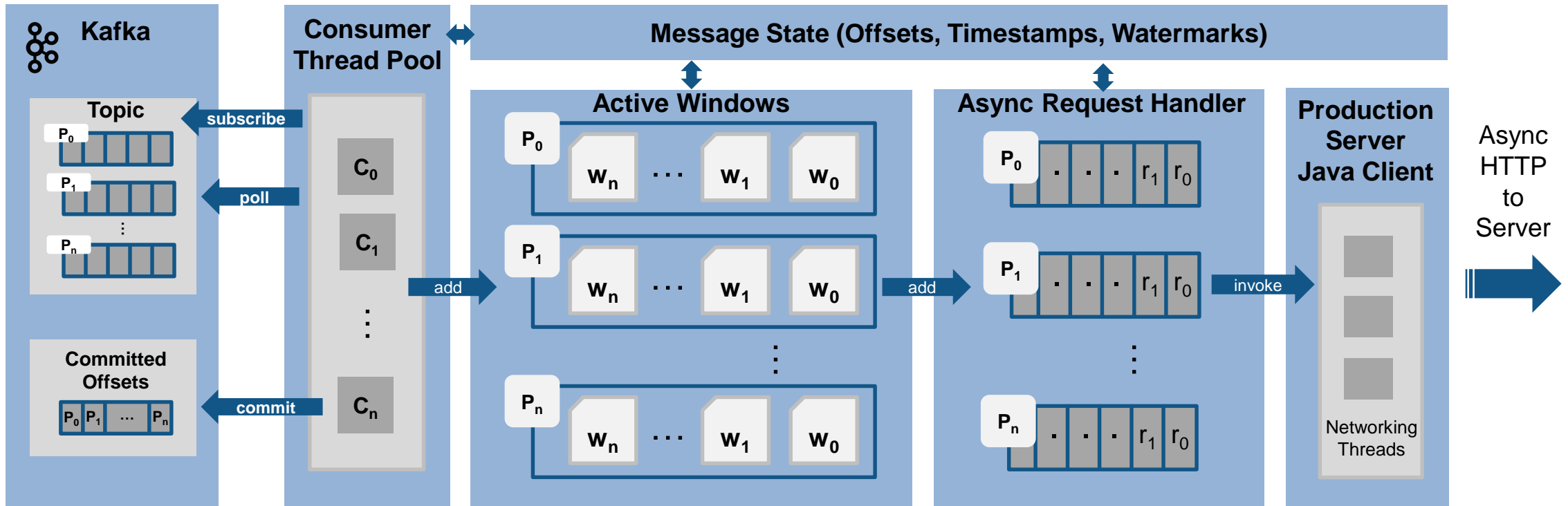


- Deployment model: Docker container suitable for Azure Container Instance

4

Integrate with
Production
Systems

Kafka connector architecture



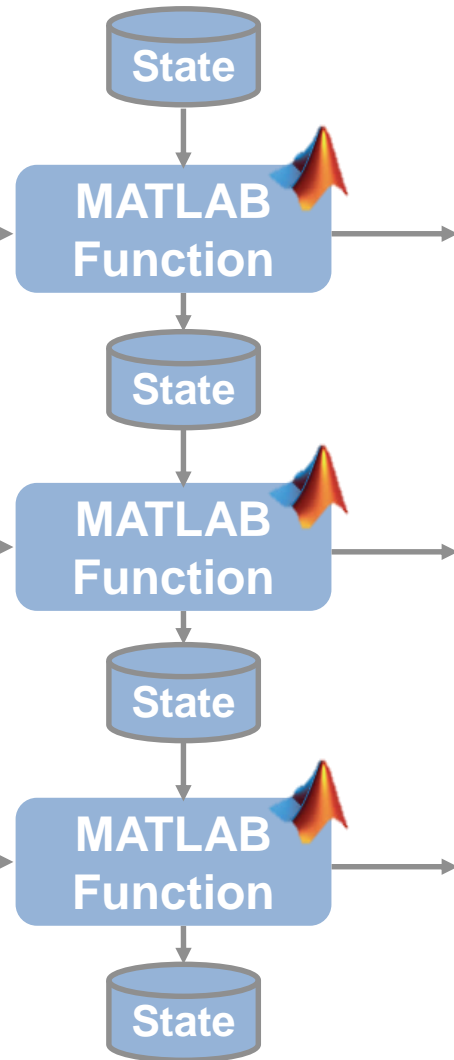
Streaming data is treated as an unbounded Timetable

Input Table

Event Time	Pump Id	Flow	Pressure	Current
18:01:10	Pump1	1975	100	110
18:10:30	Pump3	2000	109	115
18:05:20	Pump1	1980	105	105
18:10:45	Pump2	2100	110	100
18:30:10	Pump4	2000	100	110
18:35:20	Pump4	1960	103	105
18:20:40	Pump3	1970	112	104
18:39:30	Pump4	2100	105	110
18:30:00	Pump3	1980	110	113
18:30:50	Pump3	2000	100	110
...

Output Table

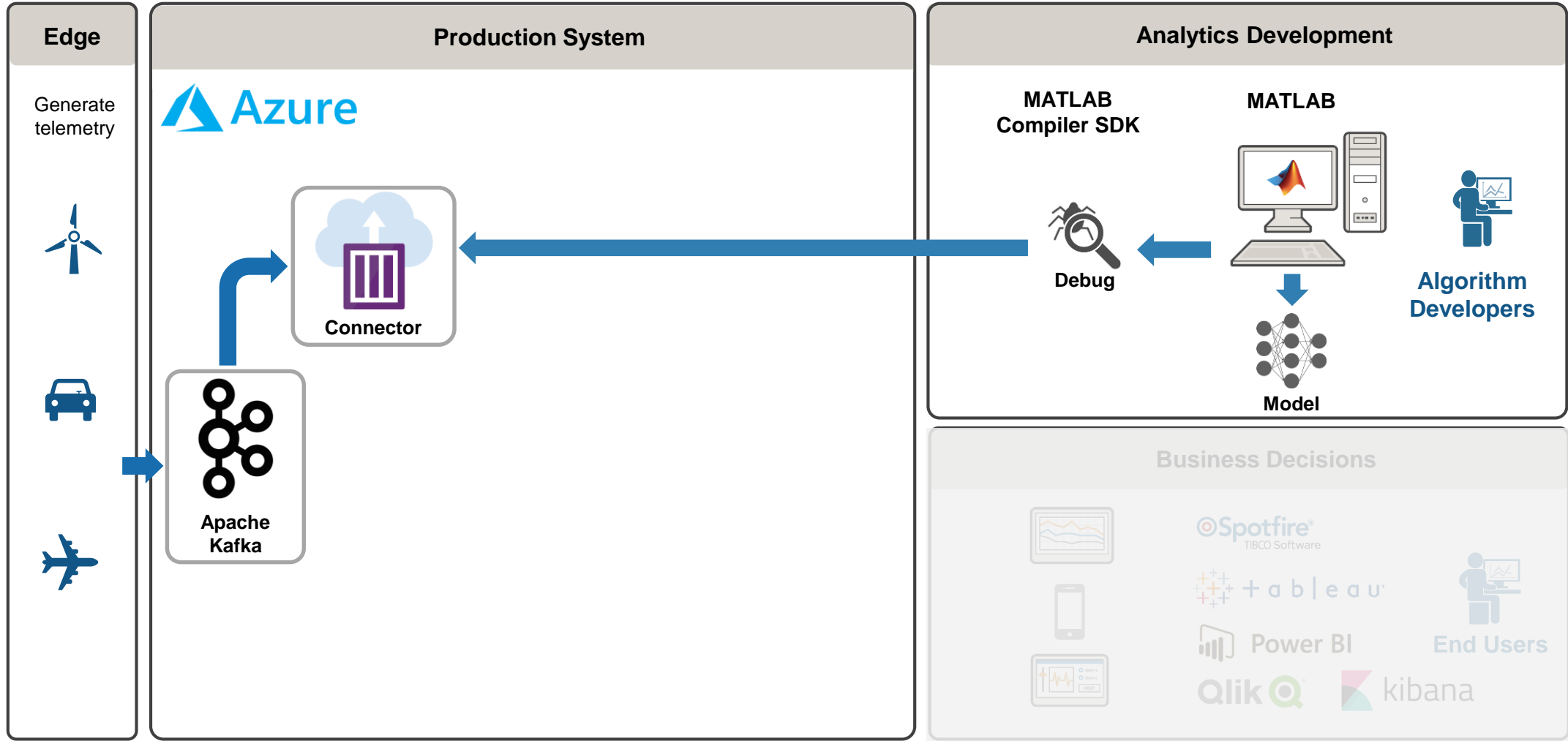
Time window	Pump Id	Bearing Friction	
...	
18:00:00	18:10:00	Pump1 Pump3 Pump4	5
18:10:00	18:20:00	Pump2 Pump3 Pump4	7 3 ...
18:20:00	18:30:00	Pump1 Pump3 Pump4	... 4 ...
18:30:00	18:40:00	Pump5 Pump3 Pump4	... 5 8



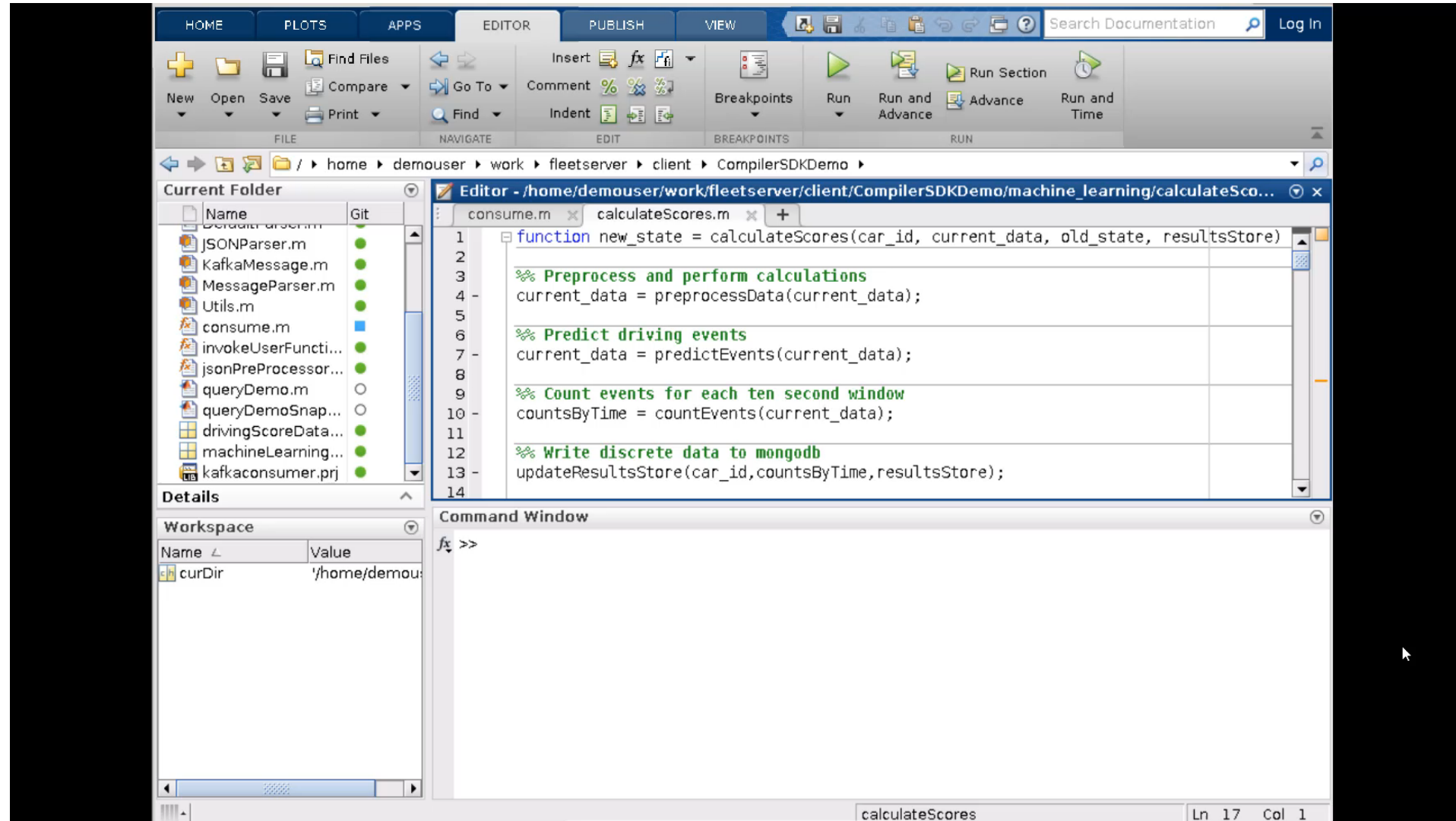
4

Integrate with
Production
Systems

Debug a Stream Processing Function in MATLAB



Debug a Stream Processing Function in MATLAB



The screenshot displays the MATLAB IDE interface. The top menu bar includes HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. The toolbar contains icons for file operations (New, Open, Save, Compare, Print), navigation (Go To, Find), editing (Insert, Comment, Indent), and execution (Run, Run and Advance, Run Section, Advance, Run and Time). The current folder is `/home/demouser/work/fleetserver/client/CompilerSDKDemo`. The workspace shows a table with columns 'Name' and 'Value', containing the entry `curDir` with value `'/home/demou...`. The editor window shows the file `calculateScores.m` with the following code:

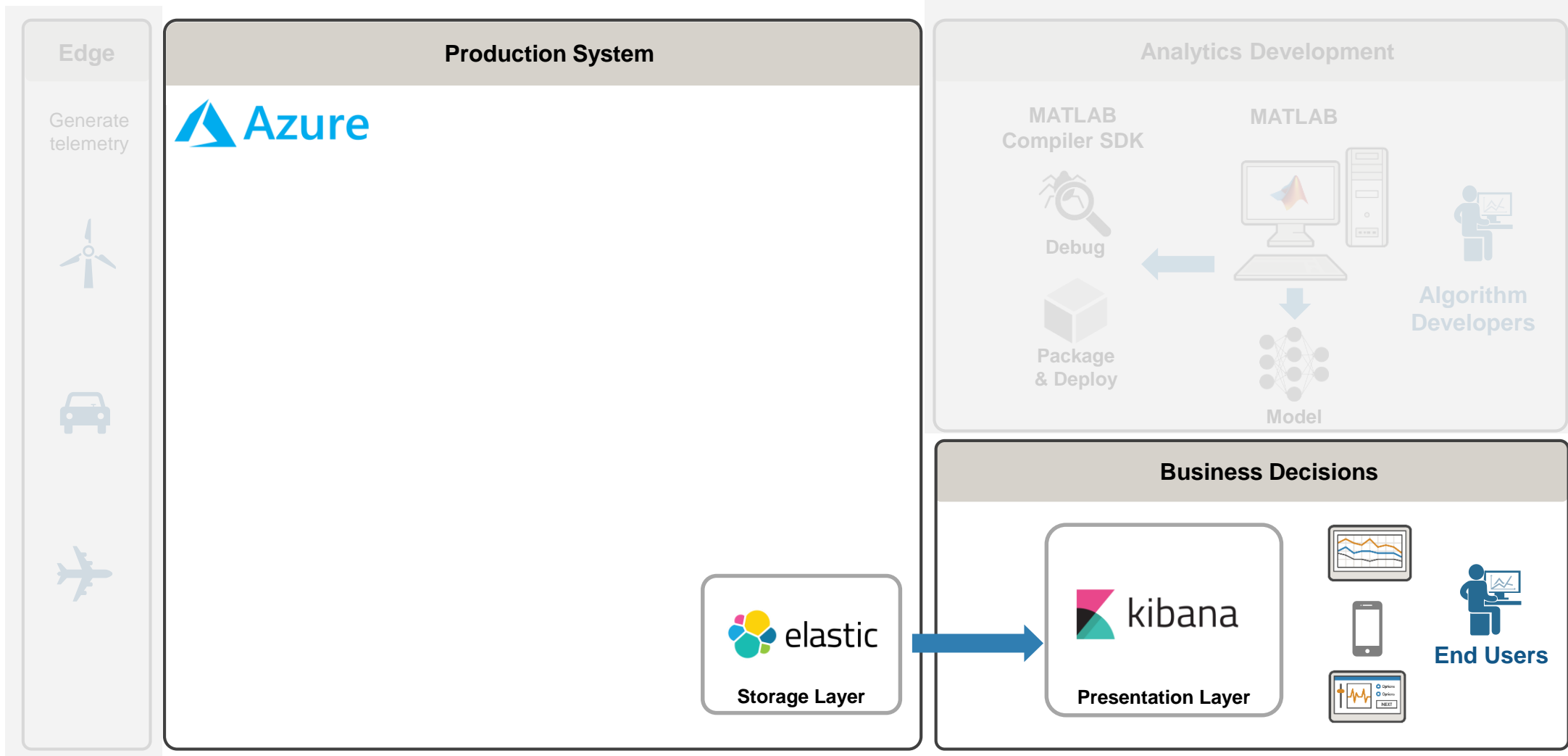
```
1 function new_state = calculateScores(car_id, current_data, old_state, resultsStore)
2
3 %% Preprocess and perform calculations
4 current_data = preprocessData(current_data);
5
6 %% Predict driving events
7 current_data = predictEvents(current_data);
8
9 %% Count events for each ten second window
10 countsByTime = countEvents(current_data);
11
12 %% Write discrete data to mongodb
13 updateResultsStore(car_id, countsByTime, resultsStore);
14
```

The Command Window shows the prompt `fx >>`. The status bar at the bottom indicates the current file is `calculateScores` at line 17, column 1.

5

Visualize Results

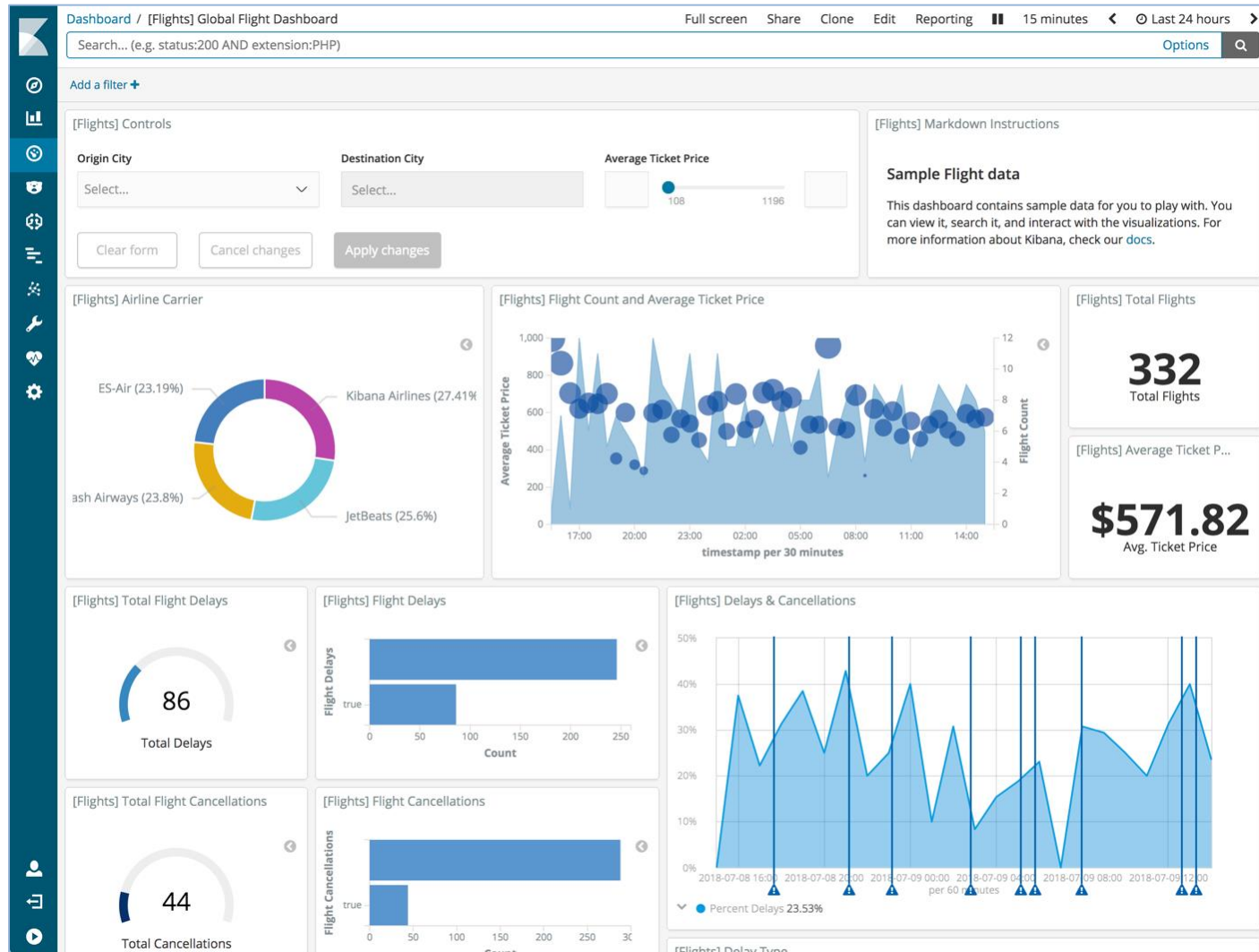
Complete Your Application



5

Visualize Results

Complete Your Application



Key Takeaways

- Rapidly import, generate, and preprocess your data with MATLAB and Simulink
- Spend less time preprocessing your data with pre-built MATLAB functions
- Focus on selecting and tuning your AI models with MATLAB apps instead of dealing with the mechanics of training
- Deploy your predictive models to the cloud, and integrate with third-party storage, applications, and visualization dashboards with MATLAB Production Server

Resources to learn and get started

- [Data Analytics with MATLAB](#)
- [Working with Enterprise IT Systems](#)
- [MATLAB Production Server](#)
- [MATLAB Compiler SDK](#)
- [Statistics and Machine Learning Toolbox](#)
- [Predictive Maintenance Toolbox](#)
- [Simulink \(related to pump model\)](#)

The screenshot shows the MathWorks website page for 'Enterprise and IT Systems'. The page features a navigation bar with links for Products, Solutions, Academia, Support, Community, and Events. A search bar is located in the top right corner. The main content area has a dark background with the text 'MATLAB Works with Your IT Systems'. Below this, there is a list of bullet points describing the benefits of MATLAB in production environments, such as secure deployment, integration with existing systems, and alignment with DevOps workflows. A quote from Manuel Arancibia and Cedric Kouam of Horizon Wind Energy is also included. At the bottom, there is a diagram titled 'Stable, Scalable Deployment' showing MATLAB running on Windows, Linux, and macOS, and being powered by AWS and Azure. A 'Learn More' link is provided at the bottom left.

Enterprise and IT Systems

MATLAB Works with Your IT Systems

MATLAB® code is production ready and can be securely deployed and integrated with enterprise IT systems, data sources, and operational technologies. IT can partner with engineering teams to:

- Run reliable, secure, and scalable production applications on Windows® and Linux®, either on-premise or on public clouds like AWS® and Microsoft® Azure®.
- Use industry-standard security mechanisms to authenticate, grant access to, and encrypt your data.
- Integrate directly to existing systems and data, including modern analytics systems like Tableau®, TIBCO® Spotfire®, and Power BI.
- Align with existing DevOps workflows and tools, and enable engineers to self-deploy their models, algorithms, and applications to production systems without having to recode.
- Leverage prebuilt, industry-specific MATLAB and Simulink toolboxes, so users can get started fast.

“By creating standalone operational programs using MATLAB Compiler and running them automatically, we can provide up-to-date forecasts and projections to Horizon analysts on a daily basis.... Our IT department set us up on the enterprise server, and we simply update the programs without any further help from them.”

— Manuel Arancibia and Cedric Kouam, Horizon Wind Energy

Learn why engineers and scientists choose MATLAB.

Stable, Scalable Deployment

MATLAB applications, algorithms, and models can run on Windows, Linux, and macOS either on-premise or on the public cloud. Leverage reference architectures for AWS and Azure to get up and running quickly. Use MATLAB Production Server™ to implement large-scale, high-availability systems. Scale up to many computers with MATLAB Distributed Computing Server™.

Learn More

- Using MATLAB in the Cloud