

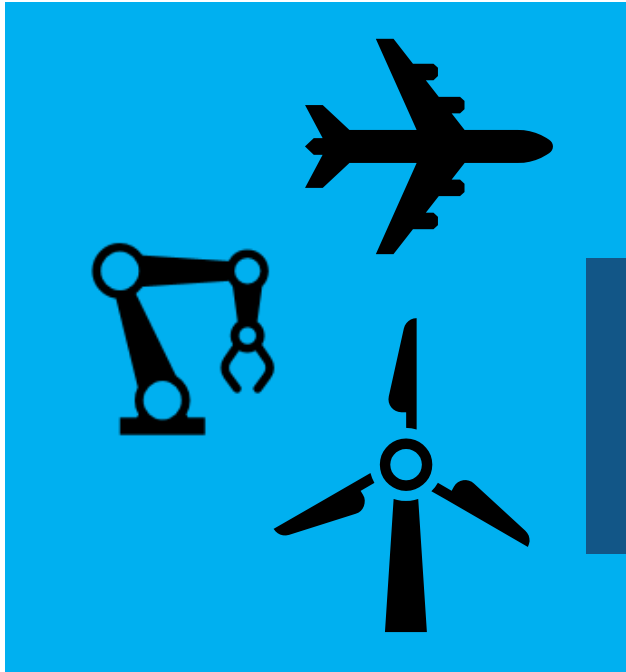
MATLAB EXPO 2019

Despliegue de Inteligencia Artificial
para decisiones de fabricación
cercanas al tiempo real

Lucas García



The Need for Large-Scale Streaming



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

Predictive Maintenance

Increase Operational Efficiency
Reduce Unplanned Downtime

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

Medical Devices

Patient Safety
Better Treatment Outcomes

Connected Cars

Safety, Maintenance
Advanced Driving Features



Car: ~25 GB per hour

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

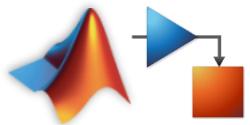
- We did this for the customer
- We wanted to go further:
 - Create a streaming application based on this real customer request
 - Develop application in a 3-4 week sprint
- We believe this represents a realistic customer situation

Our Project: Develop and operationalize a machine learning model to predict failures in industrial pumps



Process Engineer

Develops models in MATLAB and Simulink



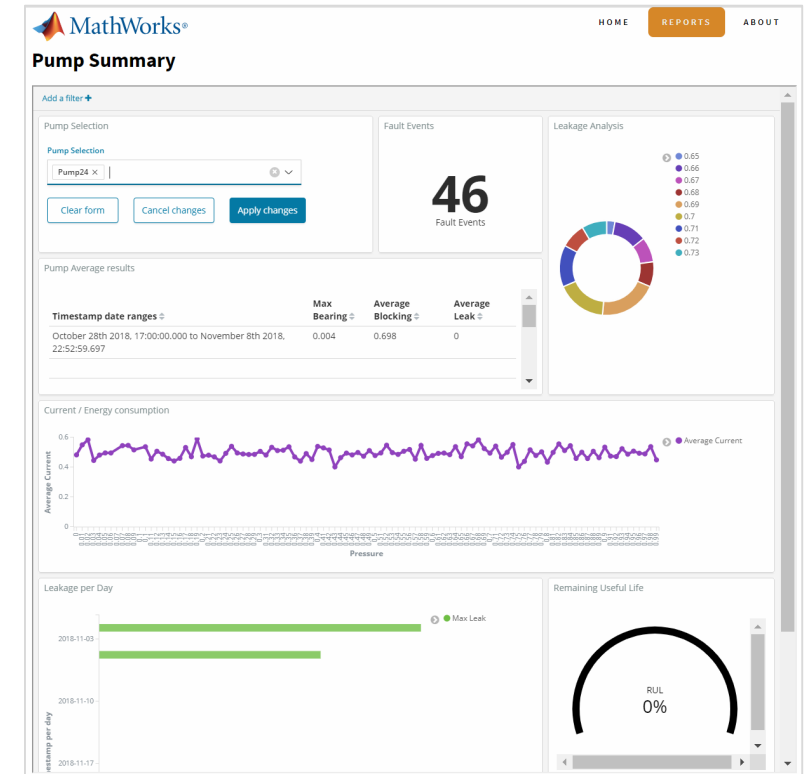
System Architect

Deploys and operationalizes model on Azure cloud



Operator

Makes operational decisions based on model output



Current system requires Operator to manually monitor operational metrics for anomalies. Their expertise is required to detect and take preventative action



Project statement: Develop end-to-end predictive maintenance system and demo in one 3-4 week sprint



Plant
Operator

1. Monitor *flow*, *pressure*, and *current* of each pump so I always know their *operational state*
2. Need *alert* when fault parameters drift outside an acceptable range so I can take *immediate corrective action*
3. Continuous estimate of each pump's *remaining useful life (RUL)* so I can *schedule maintenance or replace* the asset

Challenges of AI Deployment



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

Challenges of AI Deployment



**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 cloud platform to quickly configure and provision the services needed to build the solution, while minimizing lock-in to a particular provider

Challenges of AI Deployment



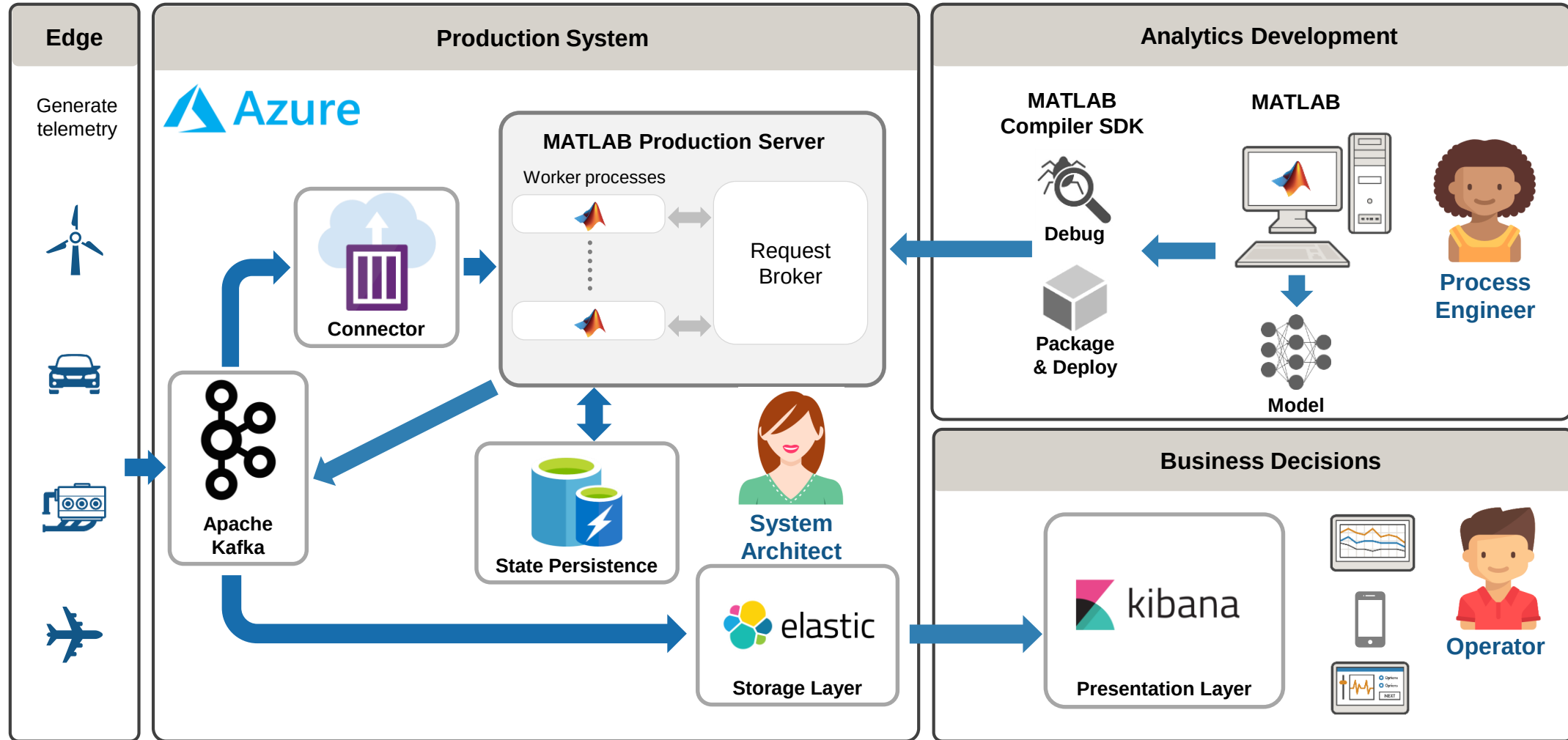
Process
Engineer

Need software for multidisciplinary problem across teams, plus integration w/ IT

Solution: Use MATLAB and integrate with OSS



Predictive Maintenance Architecture on Azure





Modeling approach

Process Engineer

1

Access and Explore Data

Files



Databases



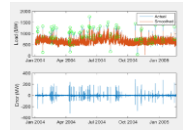
Sensors



2

Preprocess Data

Working with
Messy Data



Data Reduction/
Transformation



Feature
Extraction



3

Develop Predictive
Models

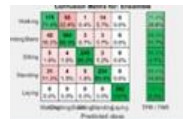
Model Creation e.g.
Machine Learning



Parameter
Optimization



Model
Validation



4

Integrate with
Production
Systems

Desktop Apps



Enterprise Scale
Systems



Embedded Devices
and Hardware



5

Visualize Results

3rd party
dashboards



Web apps





Process
Engineer

Review model requirements



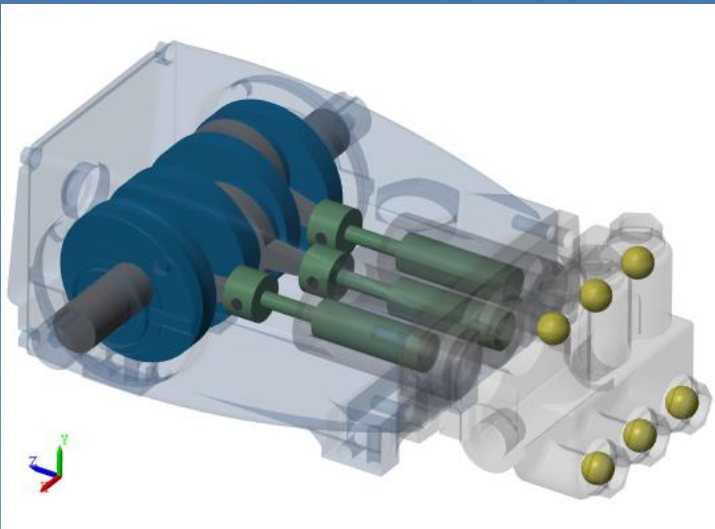
Requirements From Operator

- Continuous predictions of type of fault
 - “Blocking”
 - “Leaking”
 - “Bearing”
 - Combination of above
- Continuous predictions of Remaining Useful Life [RUL]



Requirements From System Architect

- Define window for streaming
- Define format of results, intermediate values
- Test code
- Scale code

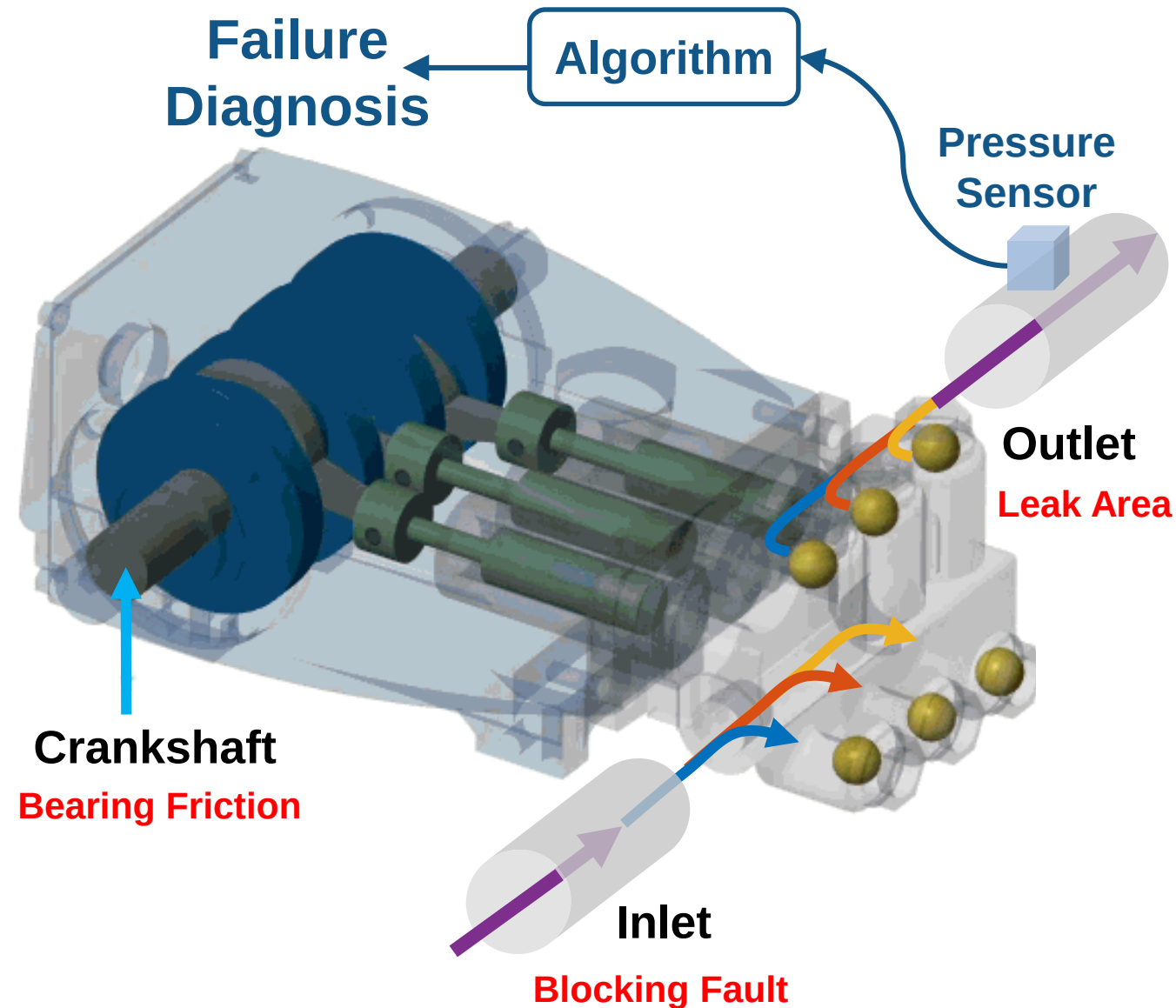
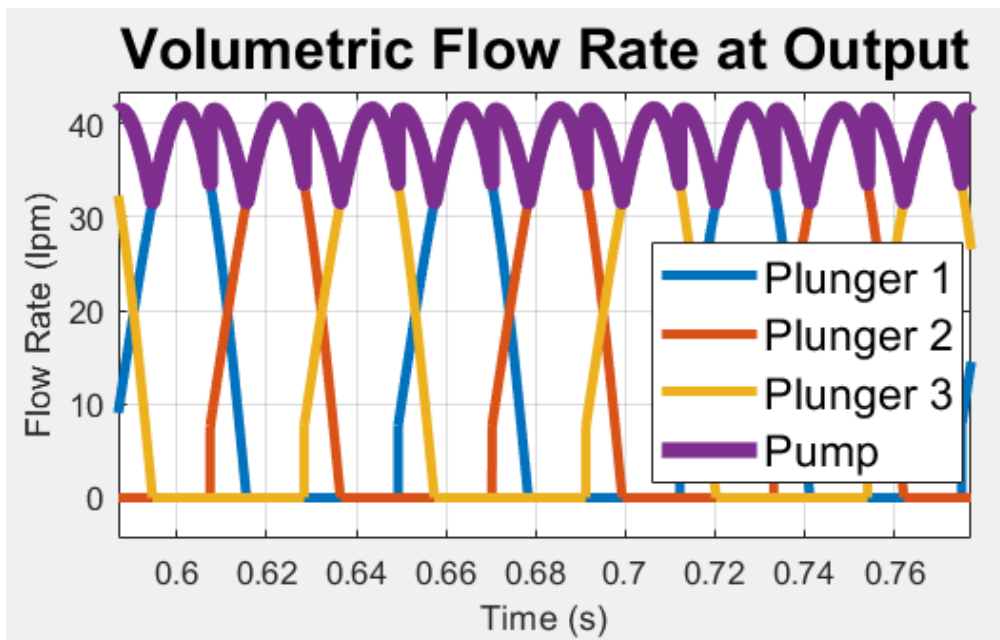




Process
Engineer

Physics of Triplex Pump

- Crankshaft drives three plungers
 - Each 120 degrees out of phase
 - One chamber always discharging
 - Three types of **failures**

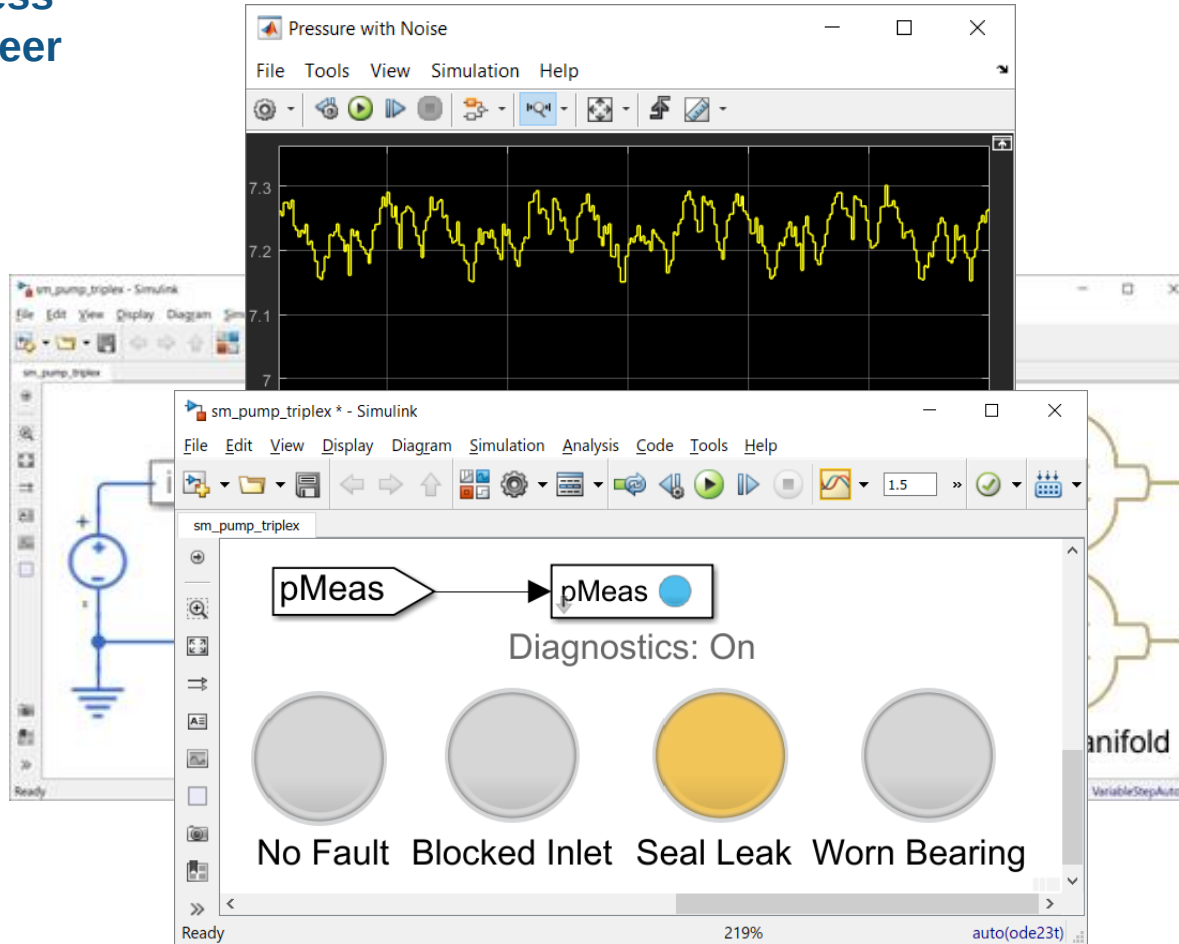




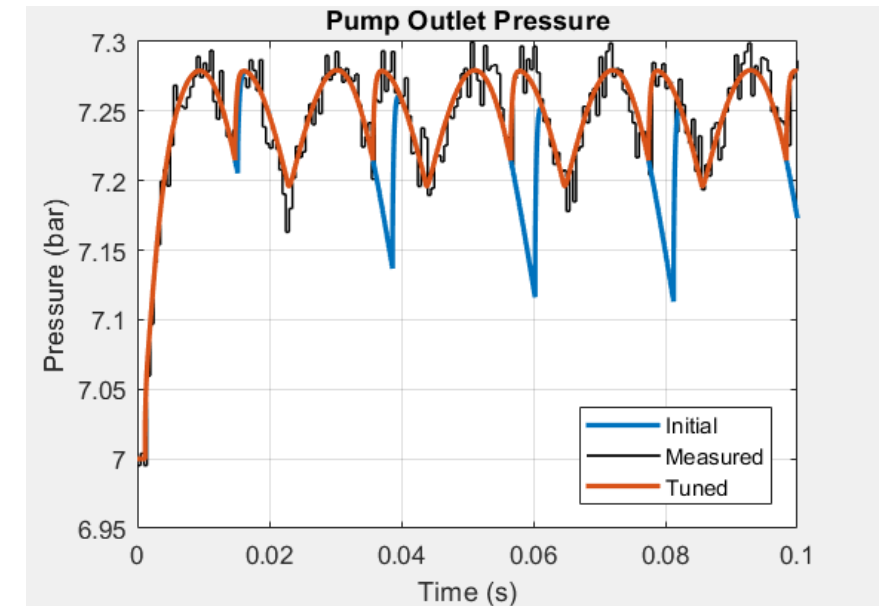
1

Access and Explore Data

Use sensor data from pump to identify levels of failure

Process
Engineer

Simulate faults



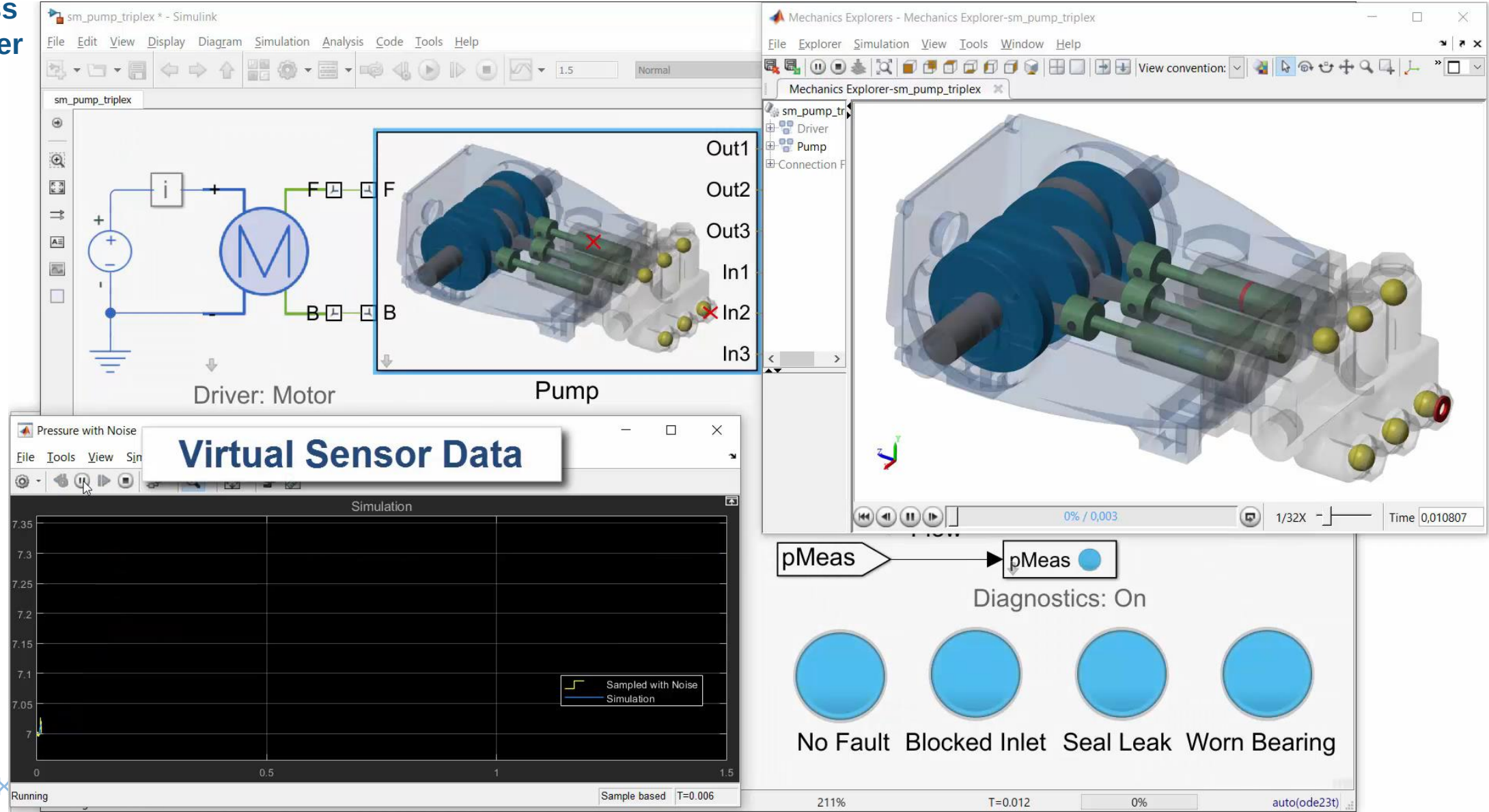
Pump sensor data



1

Access and Explore Data

Build digital twin and generate sensor data

Process
Engineer

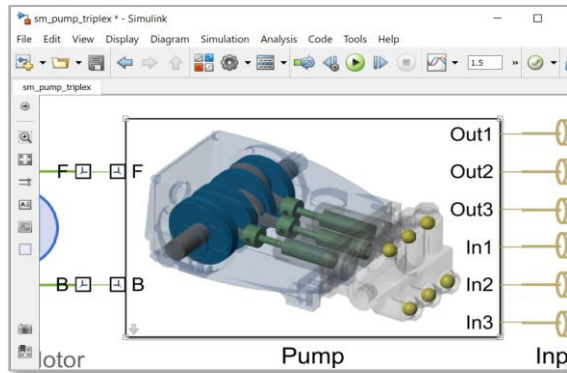


1

Access and Explore Data

Process
Engineer

Simulate data with many failure conditions

Leak Area = $[1e-9 \ 0.036]$ Bearing Friction = $[0 \ 6e-4]$ Blocking Fault = $[0.5 \ 0.8]$

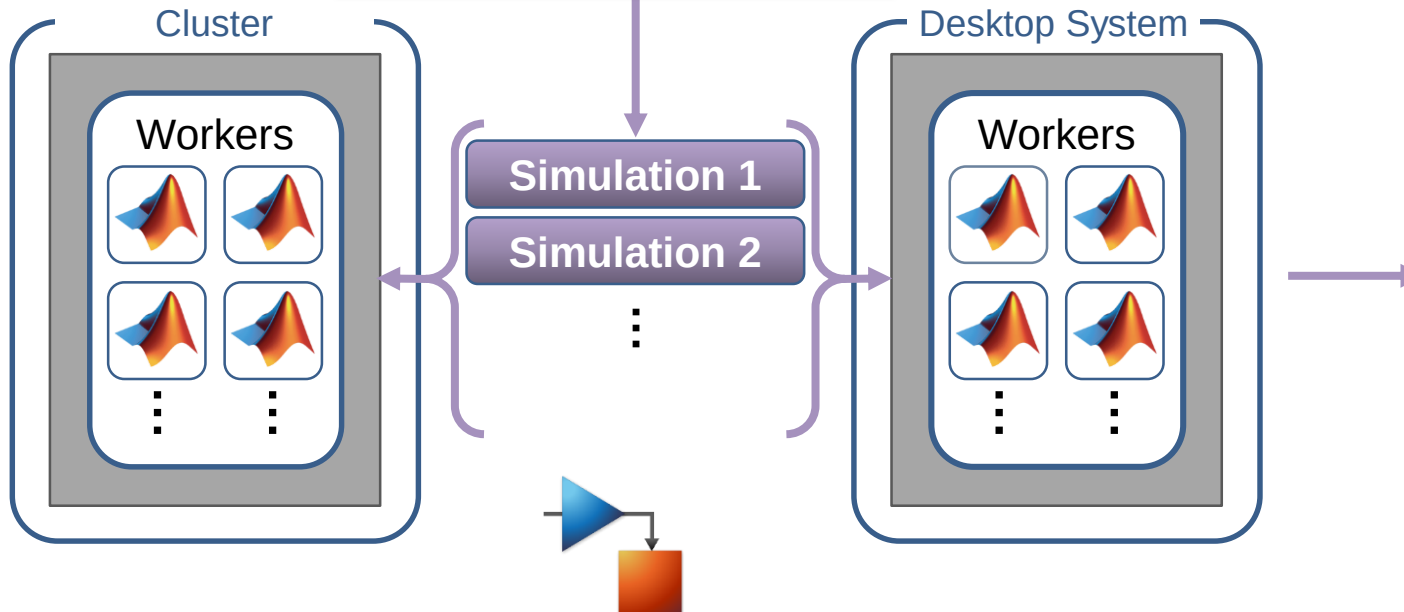
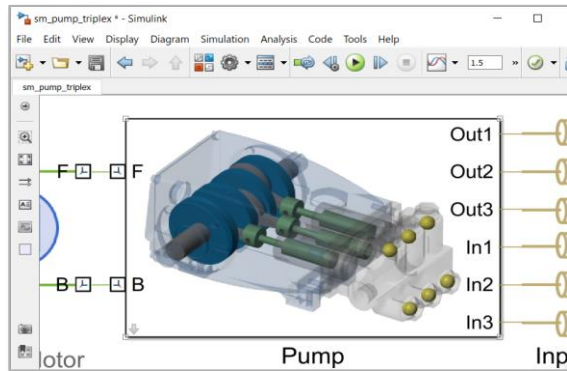


1

Access and Explore Data

Simulate data with many failure conditions

Process
Engineer



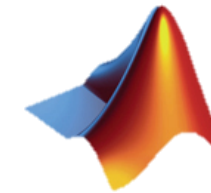
Access Data

```
ens = simulationEnsembleDatastore(location)
```

```
ens =
```

simulationEnsembleDatastore with properties:

```
DataVariables: [25x1 string]
IndependentVariables: [0x0 string]
ConditionVariables: [0x0 string]
SelectedVariables: [25x1 string]
ReadSize: 1
NumMembers: 702
LastMemberRead: [0x0 string]
Files: [702x1 string]
```



Run parallel simulations

Store data on HDFS



2

Preprocess Data

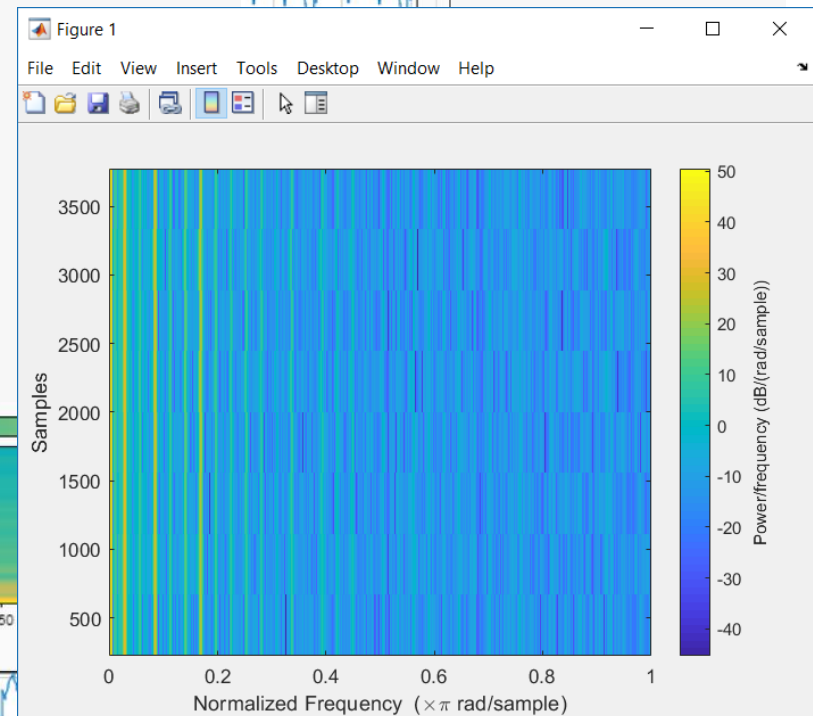
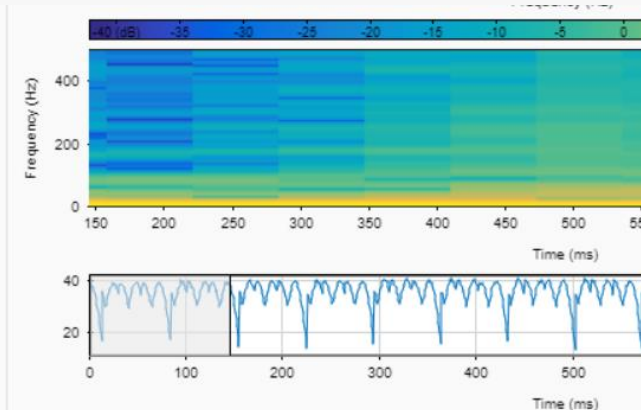
Represent signal information

Process
Engineer

Signal processing

```
[Spectrum,Frequencies] = pspectrum(data.Flow);
[pLow,pHigh] = bounds(Spectrum);
fPeak = Frequencies(Spectrum==pHigh);
qPeak2Peak = peak2peak(data.Flow);
qCrest = peak2rms(data.Flow);
qRMS = rms(data.Flow);
qMAD = mad(data.Flow);
```

NAME	SIZE	CLASS
allfaults	1000×3	timetable
bearingPump1000×3	1000×3	timetable
blockedPu... 1000×3	1000×3	timetable
healthyPump 1000×3	1000×3	timetable
leakingPump 1000×3	1000×3	timetable





3

Develop Predictive Models

Develop Predictive Models in MATLAB

Process Engineer

	Time	1 LeakFault	2 BlockingFault	3 BearingFault	4 FaultType
1	0 sec	2.8472	-0.1477	1.8000	All
2	0.001 sec	-0.1498	-0.4207	1.3103	Bearing & Blocking
3	0.002 sec	0.6511	1.6521	-0.5557	Leak
4	0.003 sec	0.1469	-0.2775	1.0074	All
5	0.004 sec	-0.6480	0.7065	-0.8878	Blocking
6	0.005 sec	-0.8165	-0.5434	-0.3079	Blocking
7	0.006 sec	-1.0061	1.2083	0.0661	Bearing
8	0.007 sec	1.0125	-1.9098	-0.7027	Leak & Blocking

Label Faults

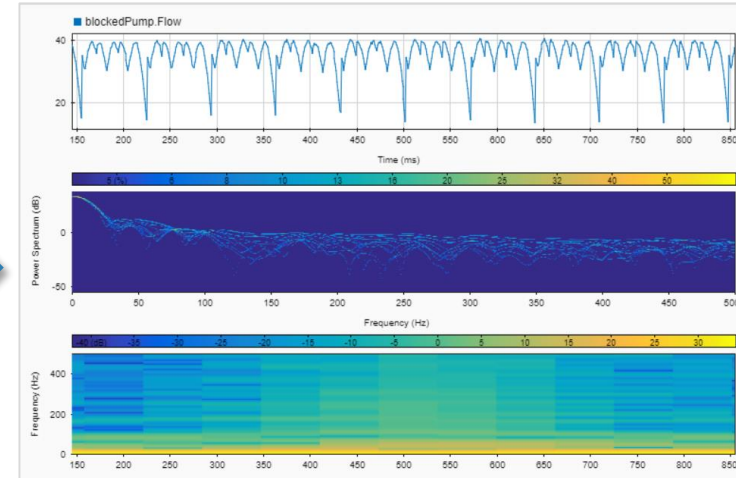
Scale

```
tt = tall(ds);
tt = preprocessData(tt);
model = TreeBagger(50,tt,'Event');
```

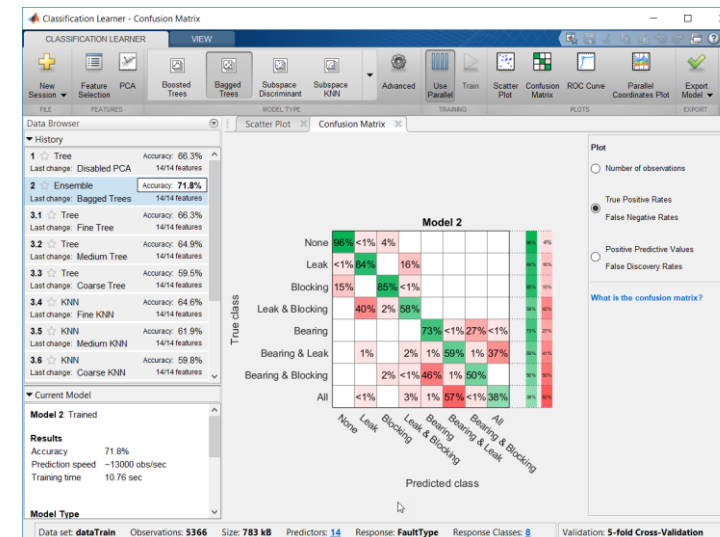
Evaluating tall expression using the Spark Cluster:

- Pass 1 of 2: Completed in 11 sec
- Pass 2 of 2: Completed in 2.3333 min

Evaluation completed in 2.6167 min



Represent Signals



Train Model

Validate Model

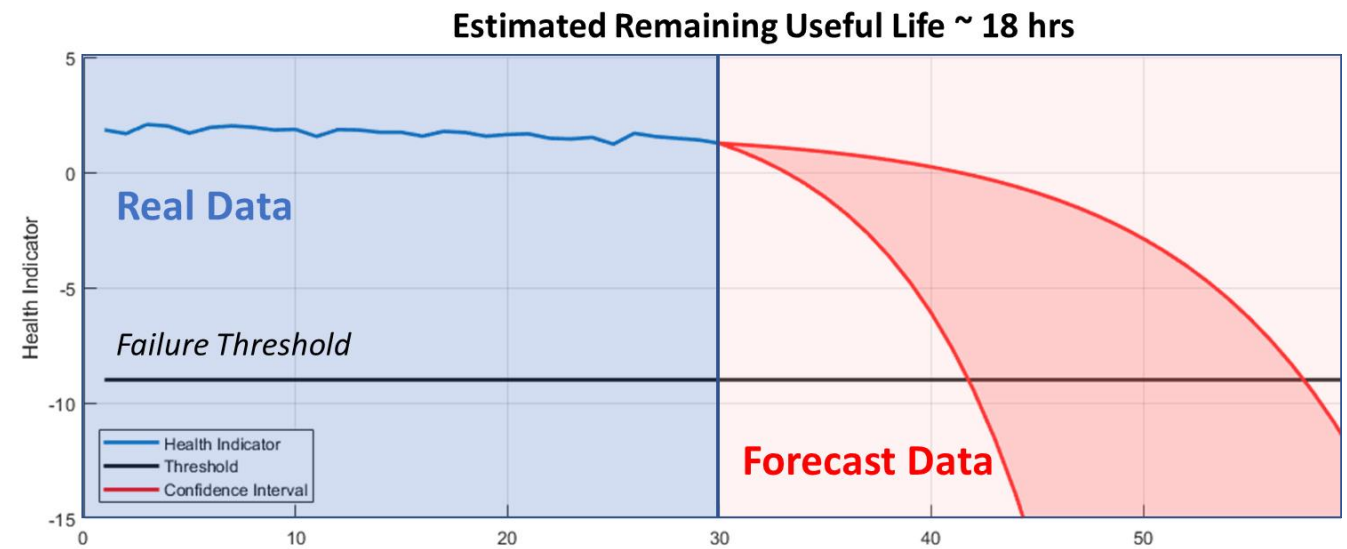
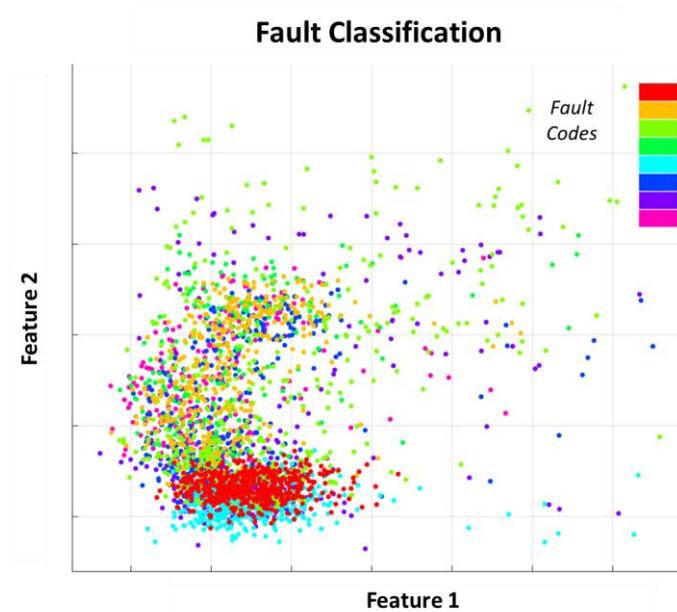


3

Develop Predictive Models

Process Engineer

Develop Predictive Models in MATLAB



**Type of Fault
(Classification)**

**Remaining Useful Life
(Regression)**



Plant Operator

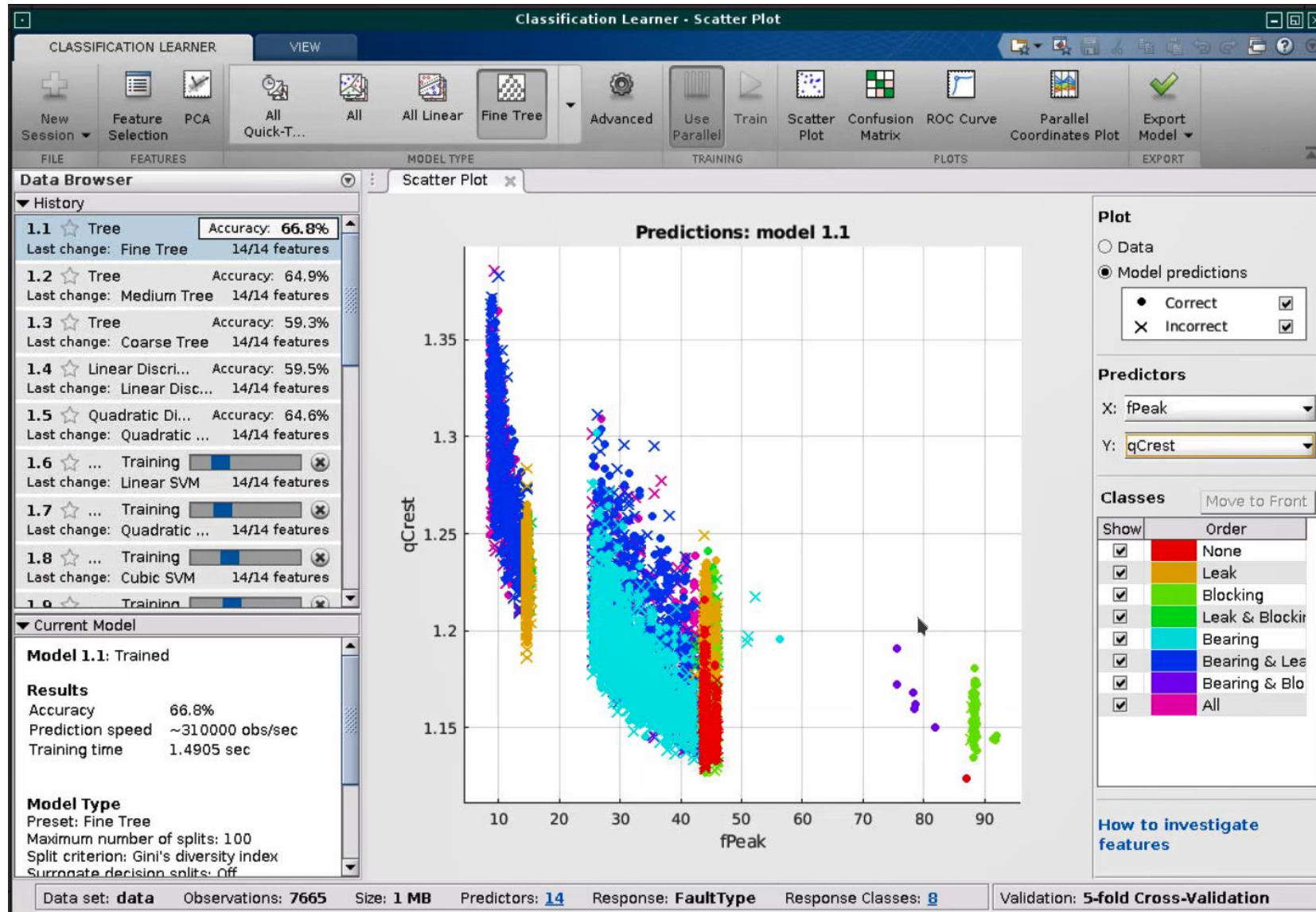


3

Develop Predictive Models

Process Engineer

Develop Machine Learning Models

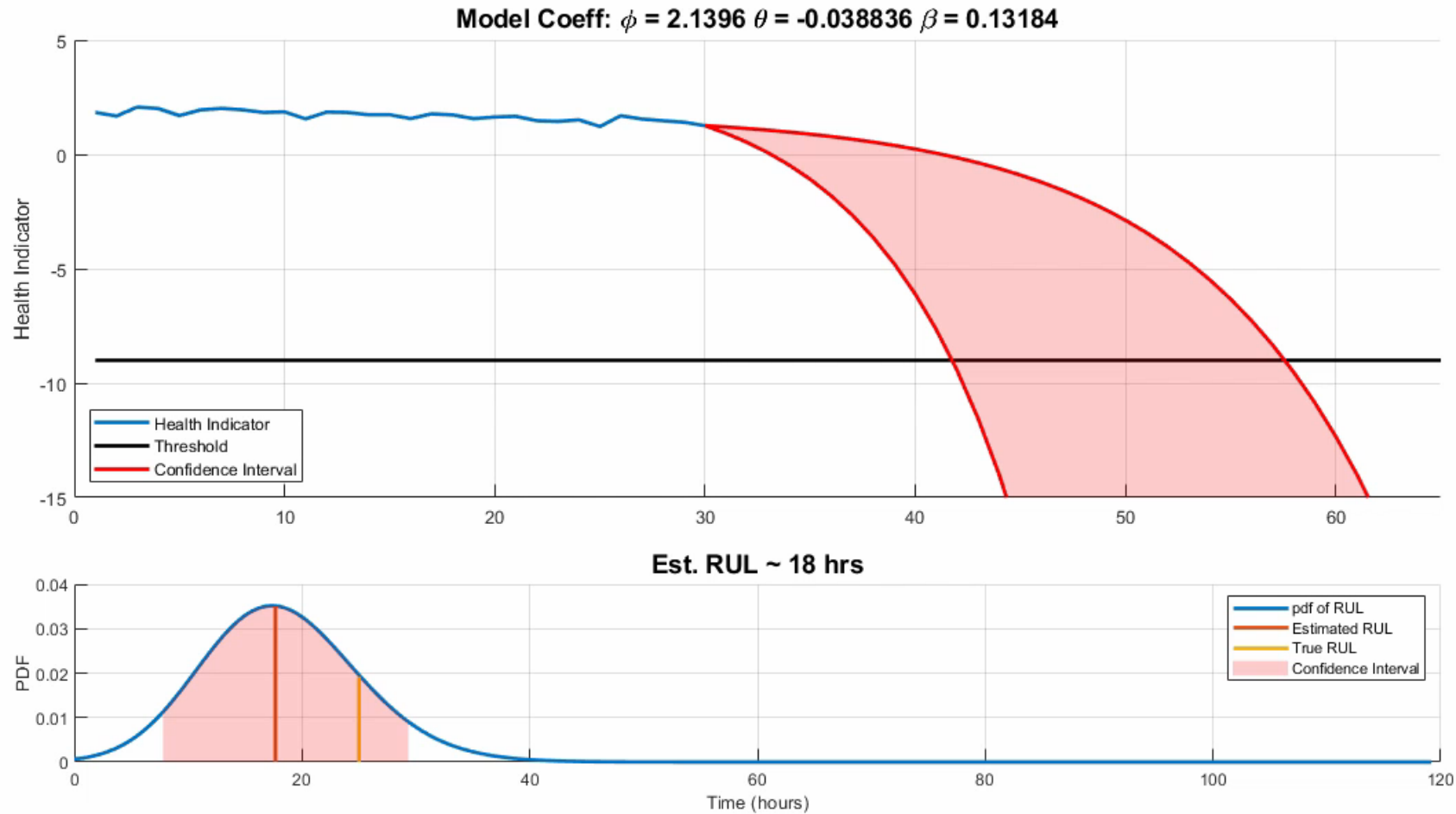




3

Develop Predictive
ModelsProcess
Engineer

Estimate Remaining Useful Life



$$S(t) = \phi + \theta(t) e^{(\beta(t)t + \epsilon(t) - \frac{\sigma}{2})}$$

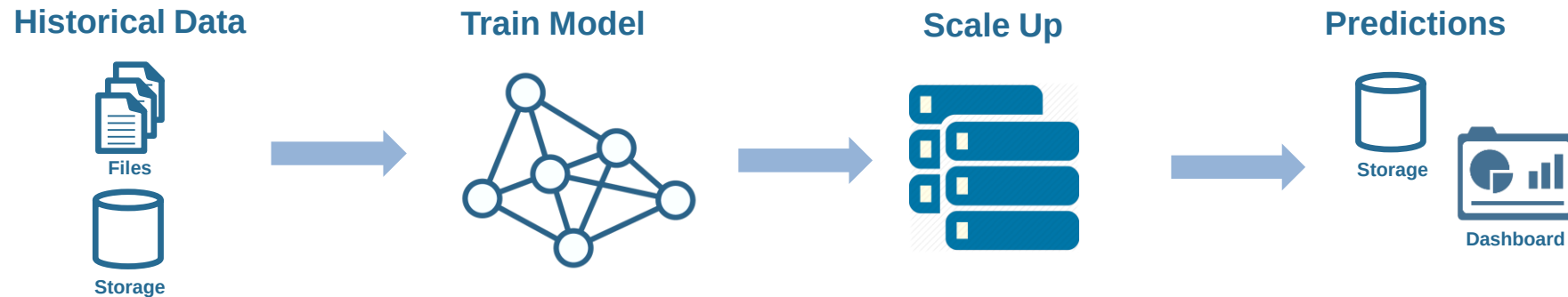


4

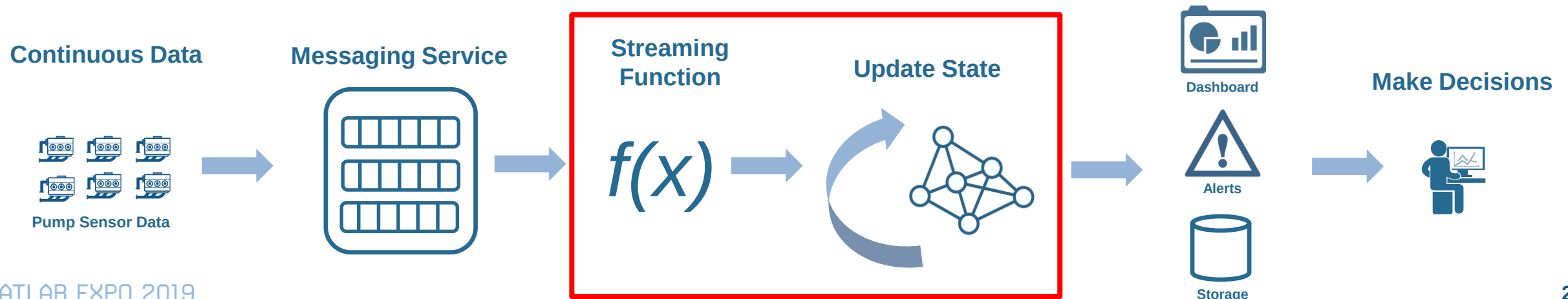
Integrate with
Production
SystemsProcess
Engineer

Develop a Stream Processing Function

- **Batch Processing:** Build and test model on simulated data



- **Stream Processing:** Apply model to sensor data in near real-time





4

**Integrate with
Production
Systems****Process
Engineer**

Develop a Stream Processing Function

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);
```

Update state

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

Write results

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

Previous state

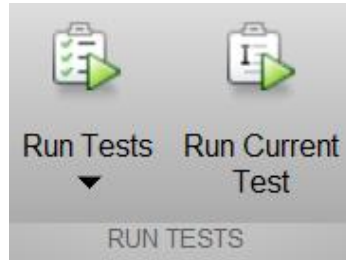
Current window of data to
be processed



4

Integrate with
Production
Systems

Process
Engineer



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

Running predictFaults_tests

....

Done predictFaults_tests

results =

1×4 **TestResult** array with properties:

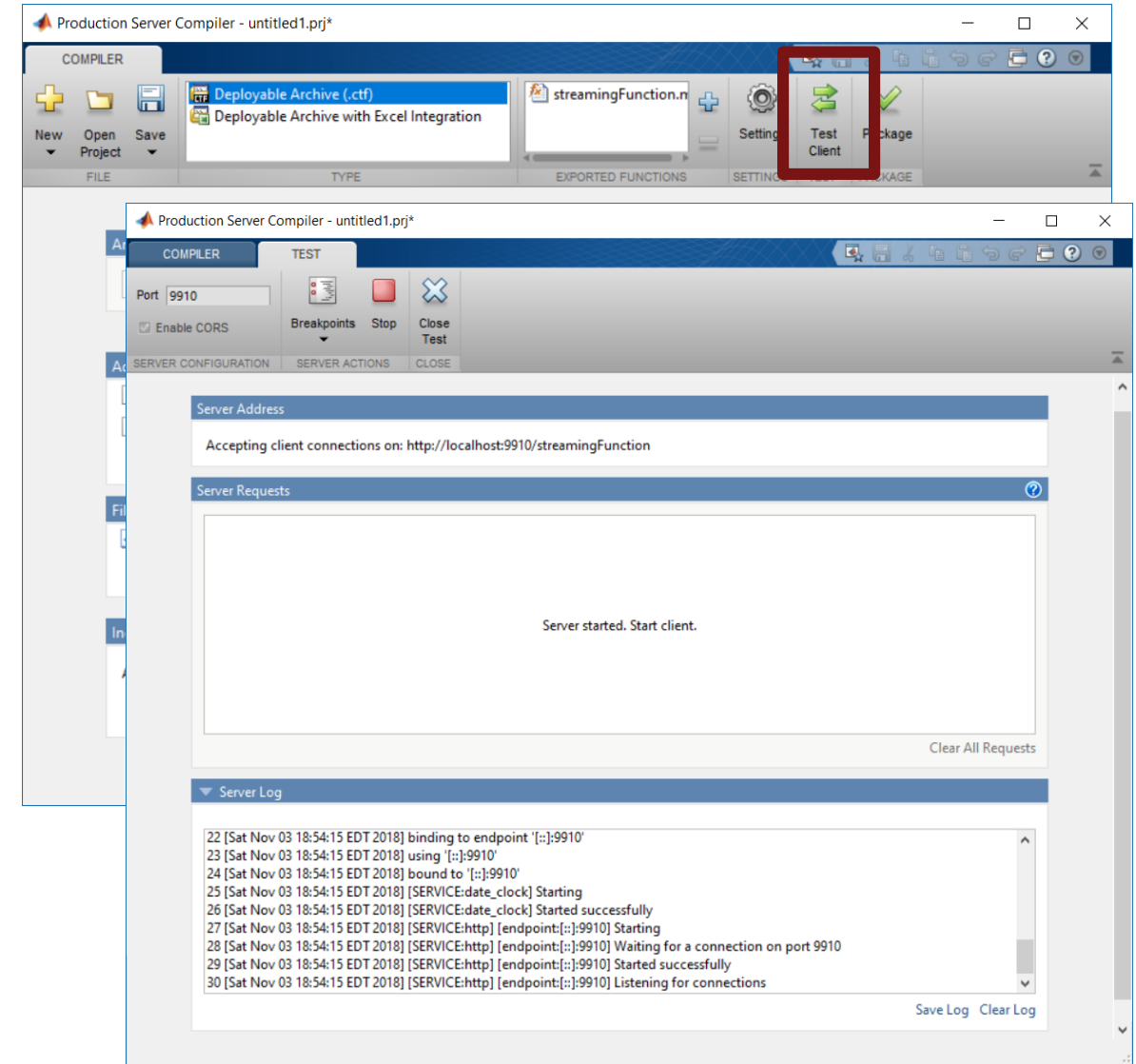
Name
Passed
Failed
Incomplete
Duration
Details

Totals:

4 Passed, 0 Failed, 0 Incomplete.

0.01614 seconds testing time.

Test Stream Processing Function





4

Integrate with
Production
Systems

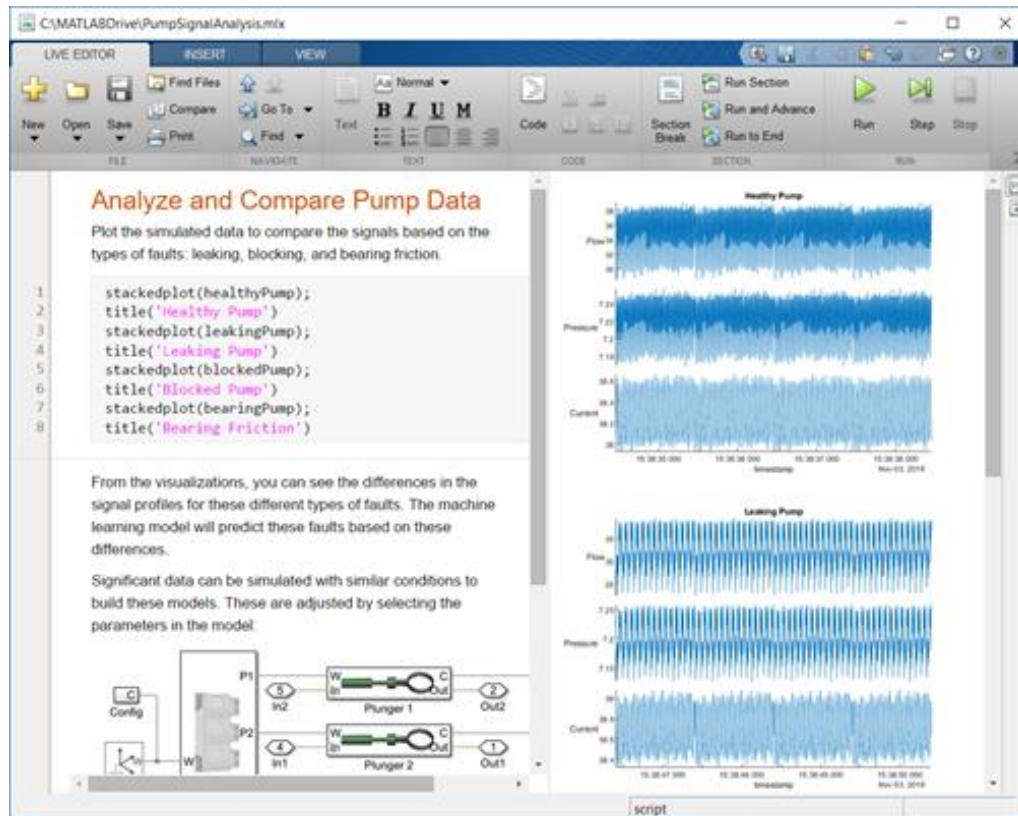
Share with the team

Process
Engineer

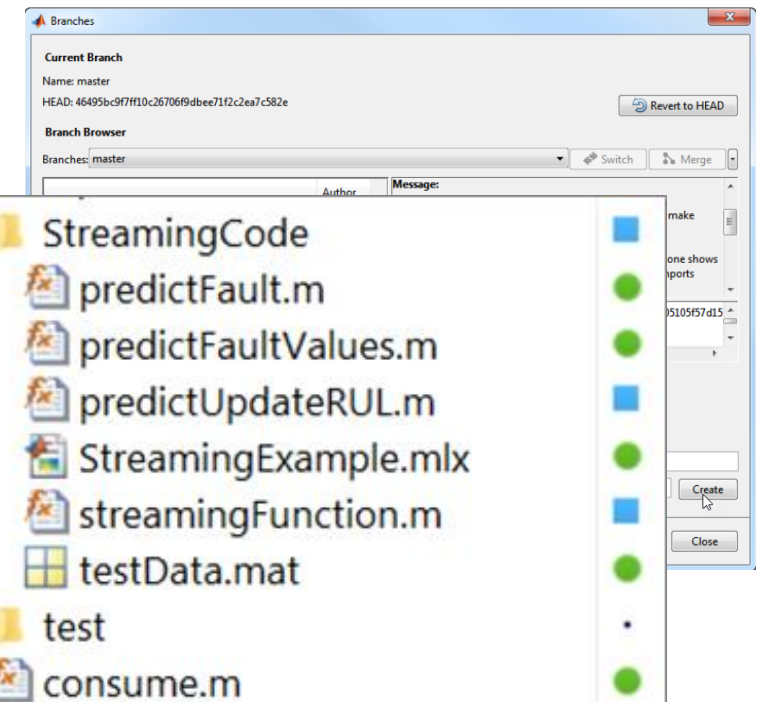
Review results with
Operator



Share code with
System Architect



.pdf, html, LaTeX



Source Control

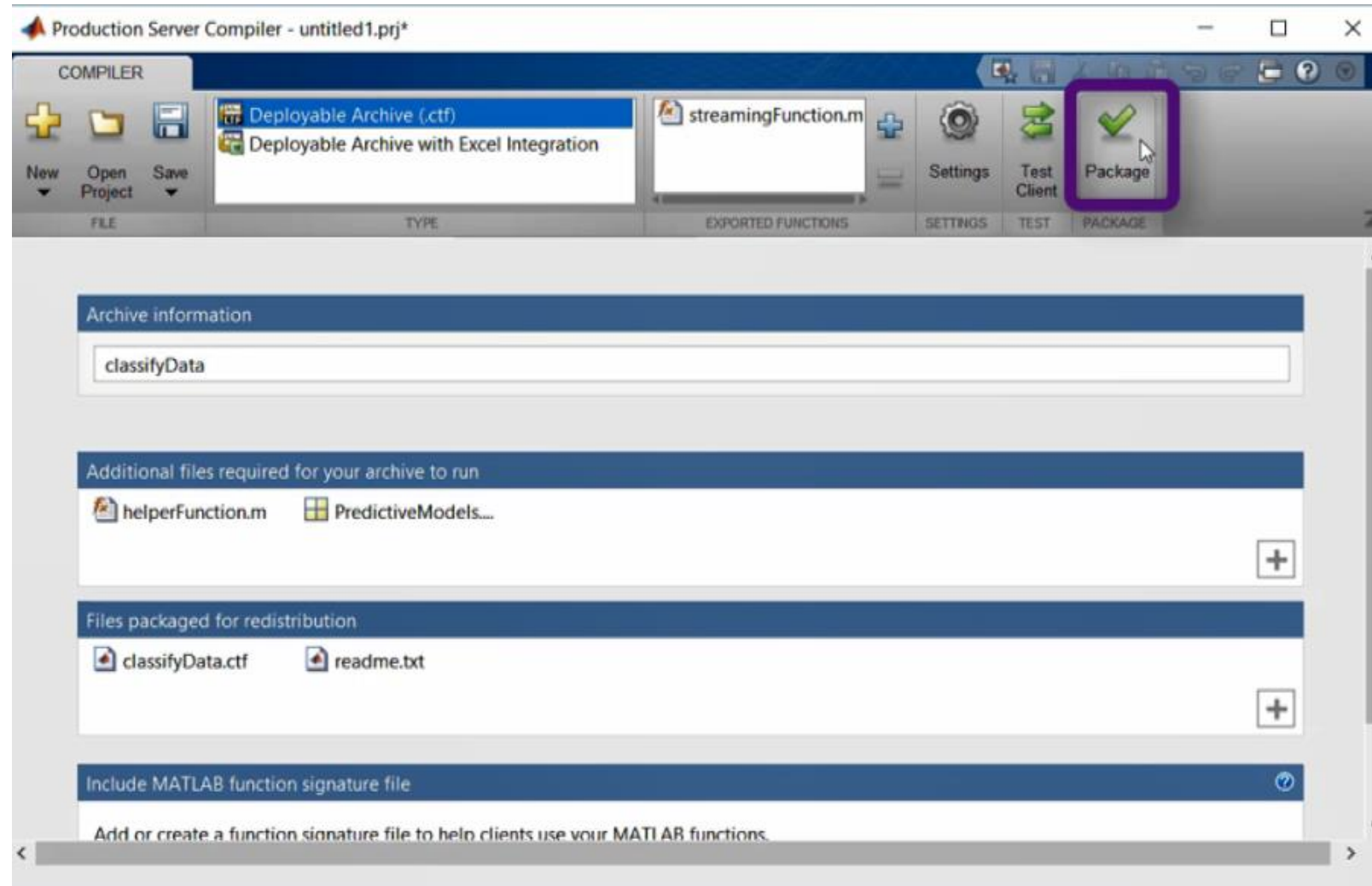


4

Integrate with
Production
Systems

Process
Engineer

Package Stream Processing Function





**System
Architect**

4

**Integrate with
Production
Systems**

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
 - Alerts when parameters drift outside the expected ranges
 - Continuous estimating of RUL for each pump



Process Engineer



Operator

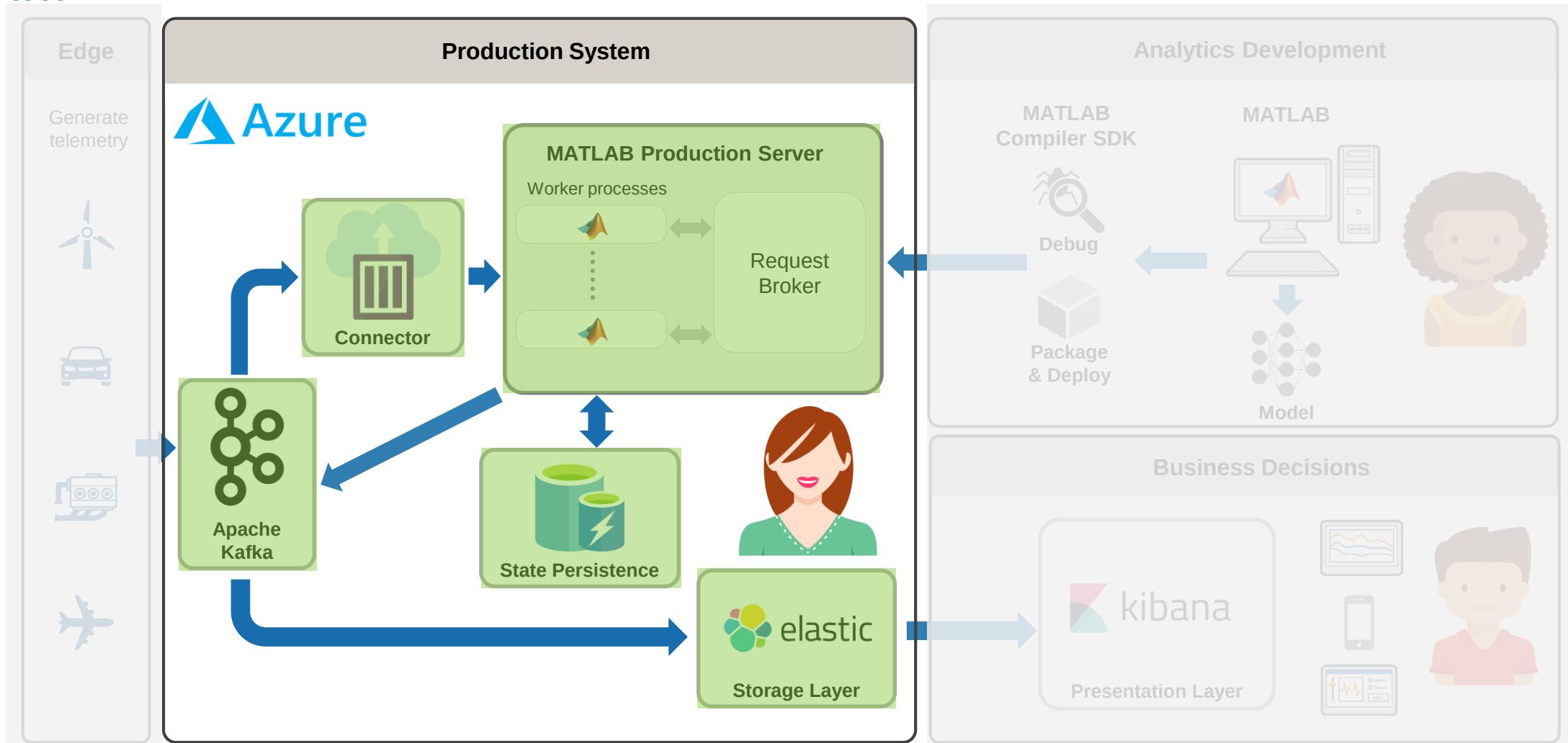


4

Integrate with
Production
Systems

System
Architect

Integrate Analytics with Production Systems

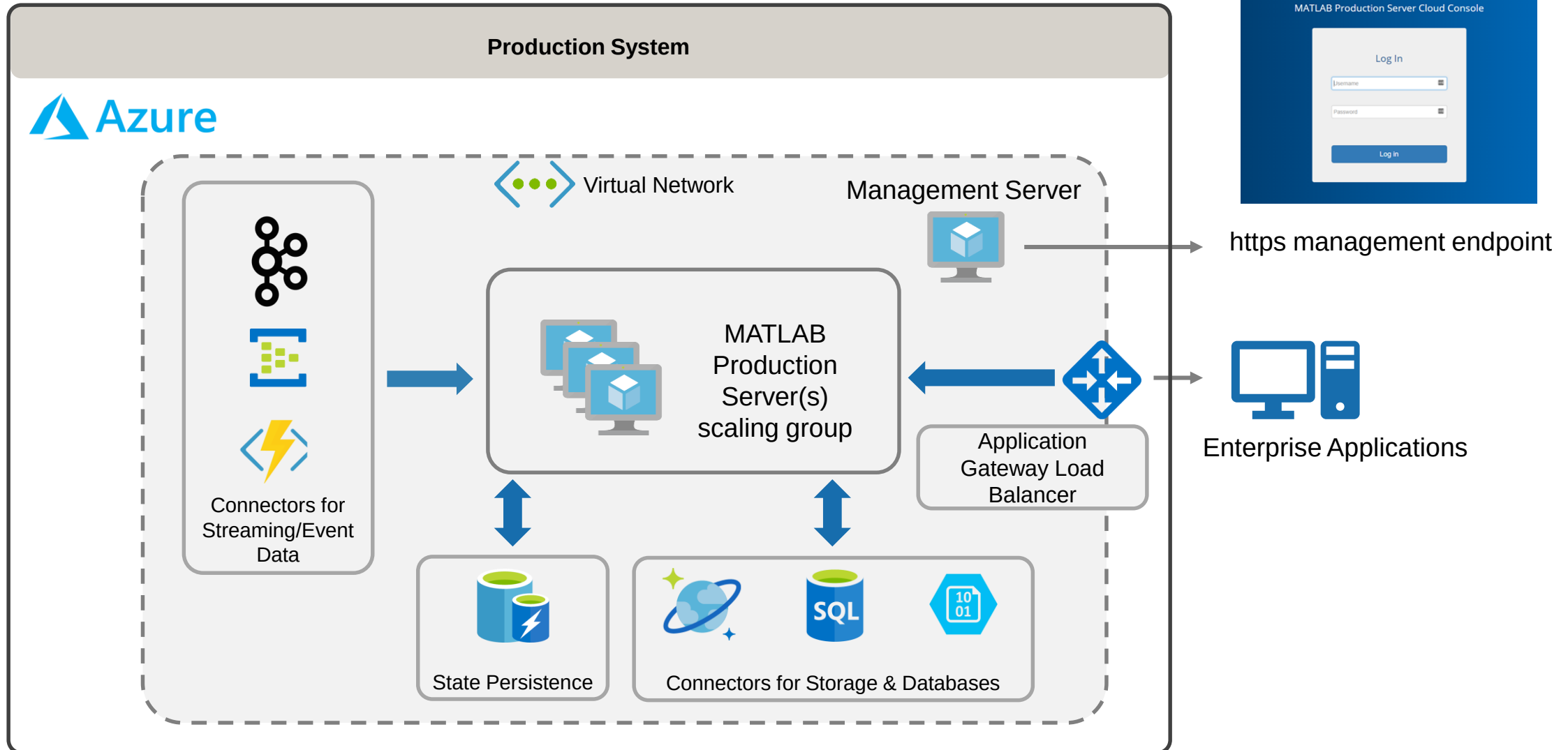




4

Integrate with
Production
Systems

MATLAB Production Server on Azure



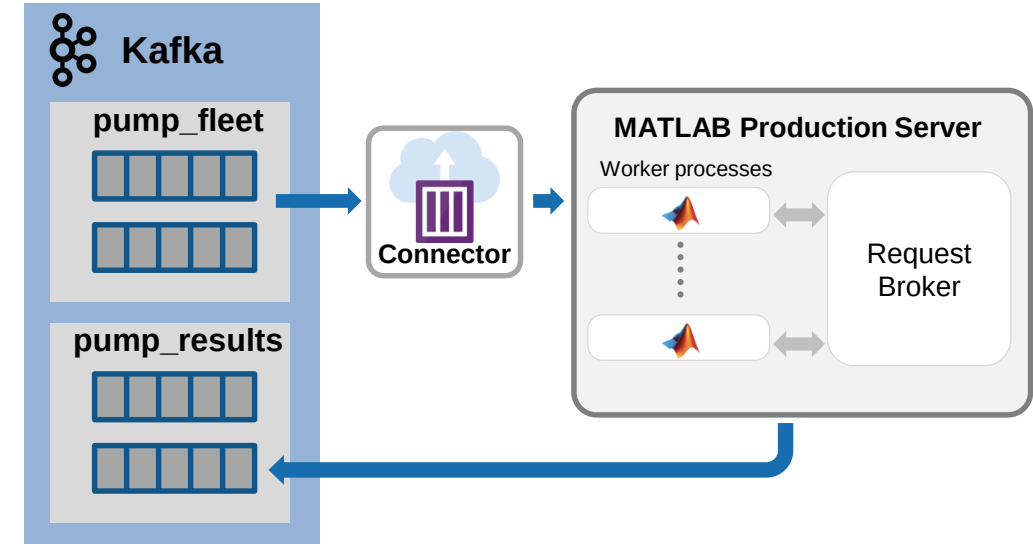


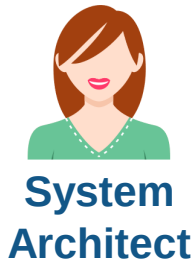
4

Integrate with
Production
Systems

Connecting MATLAB Production Server to Kafka

- Connector feeds single Kafka topic to a MATLAB function
- Publisher library for MATLAB for writing to a results stream
- Connector Features:
 - Deploy as a micro-service with Docker
 - Drive everything through config
 - Group data into time windows and pass to MATLAB as a timetable
 - Use Kafka's check-pointing (i.e. at-least-once)





4

Integrate with
Production
Systems

Messaging adapter for Production Server

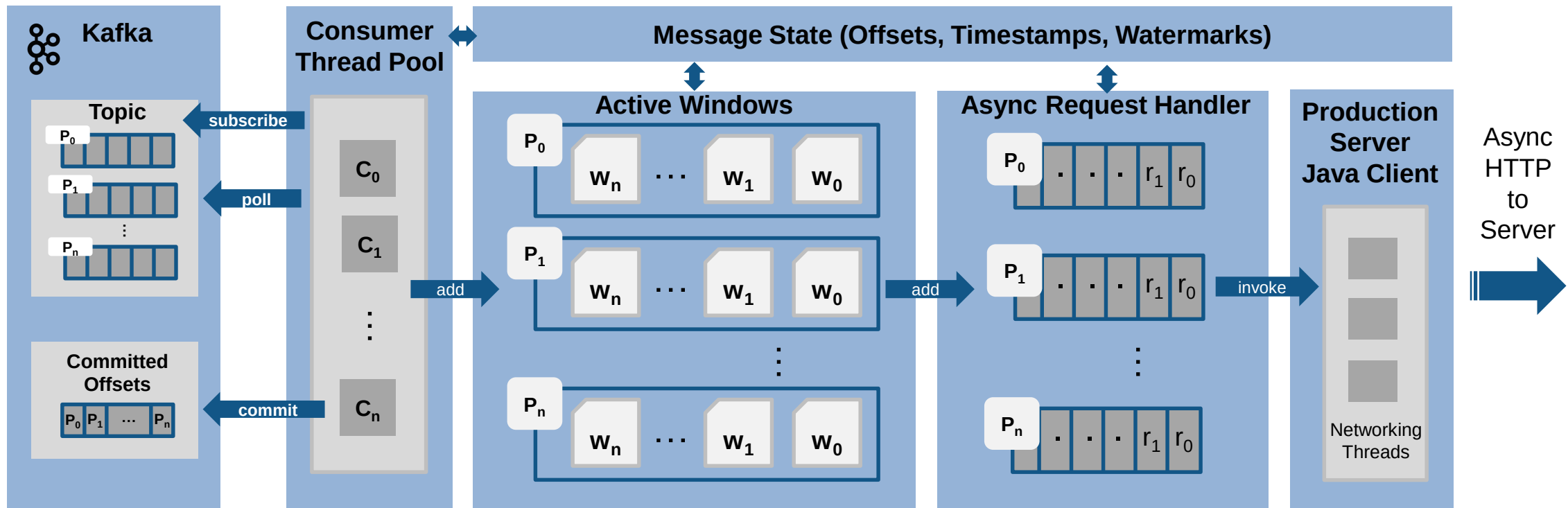
- Bridges streaming data and Production Server Async Java Client
- Batches incoming messages and sends them via HTTP request/response
 - Time windows, event time processing, and out-of-order data
- Uses Asynchronous pipeline model with back-pressure
 - Kafka consumers are automatically paused when server is busy
- Supports sequential (stateful) and unordered (stateless) processing
 - Provide unique stream ID/topic/partition info for persistence layer
- Pass data as MATLAB timetables
- Partition aware – enables full exploitation of partition-based parallelism



4

Integrate with
Production
Systems

Kafka connector architecture





4

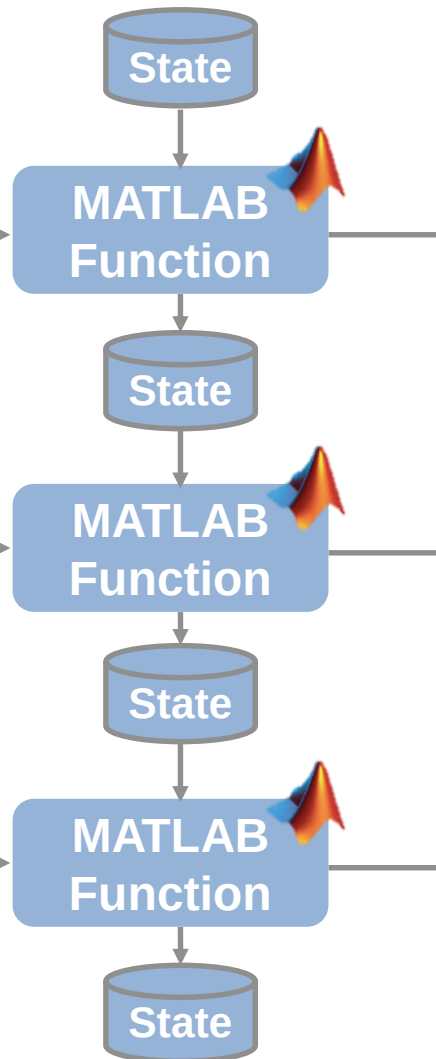
Integrate with
Production
Systems

System
Architect

Streaming data is treated as an unbounded Timetable

Input Stream

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 Stream

Time window	Pump Id	Bearing Friction
...	...	...
18:00:00	18:10:00	Pump1 Pump3 Pump4
18:10:00	18:20:00	Pump2 Pump3 Pump4
18:20:00	18:30:00	Pump1 Pump3 Pump4
18:30:00	18:40:00	Pump5 Pump3 Pump4

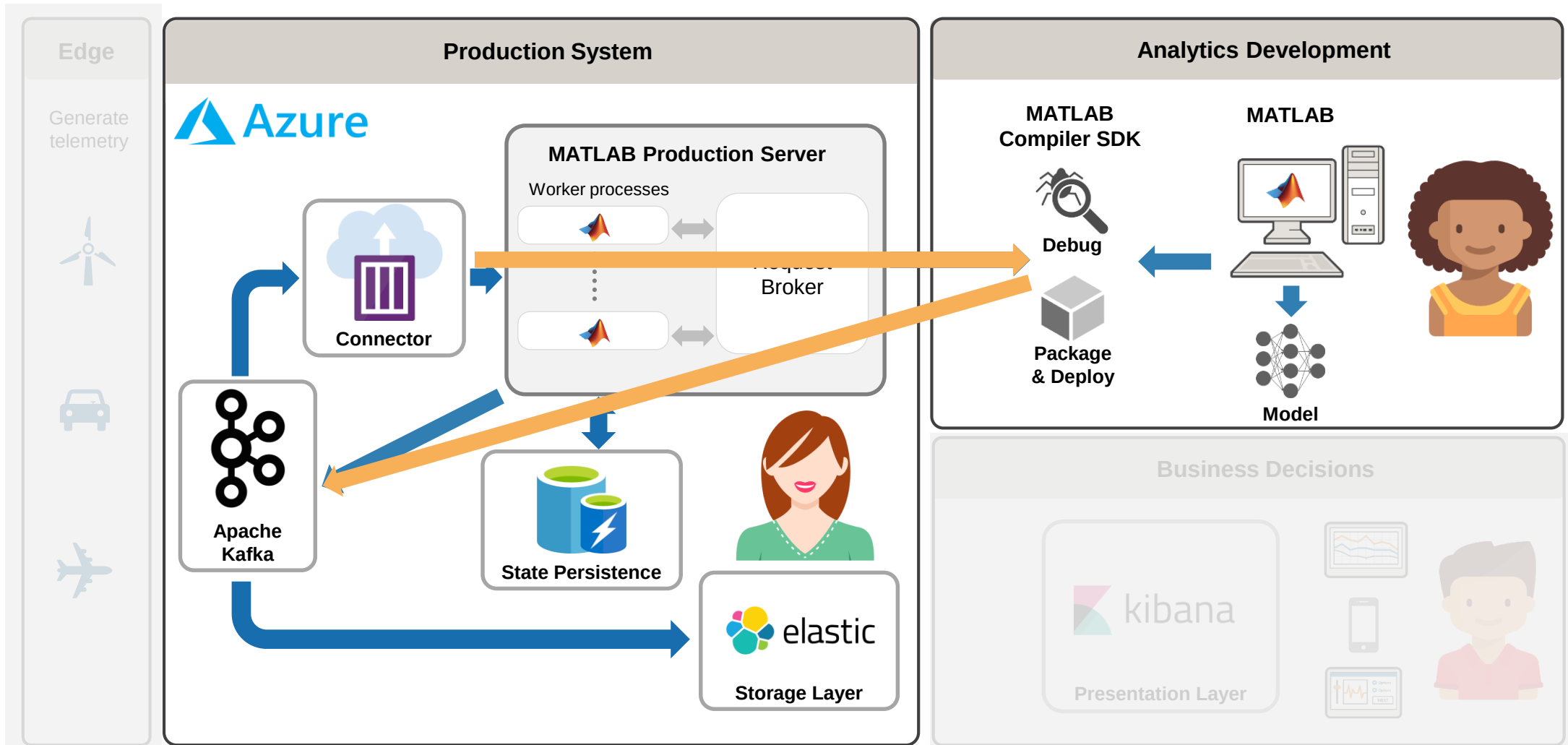


4

Integrate with
Production
Systems

System
Architect

Debug your streaming function on live data





4

Integrate with
Production
Systems

System
Architect

Debug a Stream Processing Function in MATLAB

The screenshot displays the MATLAB R2018b System Architect environment. The interface includes a top toolbar with tabs for HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. Below the toolbar is a ribbon with various icons for file operations, editing, and running code. The main workspace is divided into three panes:

- Current Folder:** Shows the project structure, including folders like '+mps', 'PumpFault_Linux', 'StreamingExample', and 'test', and files like 'consume.m', 'consumeResults.m', 'fakeResultsToKafka.m', 'make_cache_name.m', 'PumpFault_Linux.prj', and 'TheKnowledge.txt'.
- Editor:** Displays the MATLAB script 'streamingFunction.m'. The script contains the following code:


```

13 streamingFunction.m analyzeResults.m
14 %%function mlearnapp.internal.model.adapterlayer.TrainedClassificationEnsemble
15 %%function mlearnapp.internal.model.adapterlayer.TrainedLinearRegressionModel
16
17 if nargin < 2
18     old_state = [];
19 end
20
21 % Model doesn't work well with too few data points.
22 limit = 100;
23 if height(data) < limit
24     fprintf('Too few rows (%d < %d) to generate effective model.\n', ...
25         height(data), limit);
26     new_state = old_state;
27     return;
28 end
29
30 % Get the keys present in the data, use categorical for performance
31 data.key = categorical(data.key);
32 pumps = string(categories(data.key));
33
34 % Load models
35 persistent leakModel bearingModel blockingModel trainedModel
36
37 if isempty(leakModel)
38     x = load('MLModels.mat');
39     leakModel = x.leakModel;
40     bearingModel = x.bearingModel;
41     blockingModel = x.blockingModel;
42     trainedModel = x.trainedModel;
43 end
      
```
- Workspace:** Shows a single variable 'ctrl' with a value of '1x1 Contr...'.

At the bottom of the interface, there is a Command Window with a yellow banner that reads 'New to MATLAB? See resources for [Getting Started](#).' and a prompt 'fx>>'.

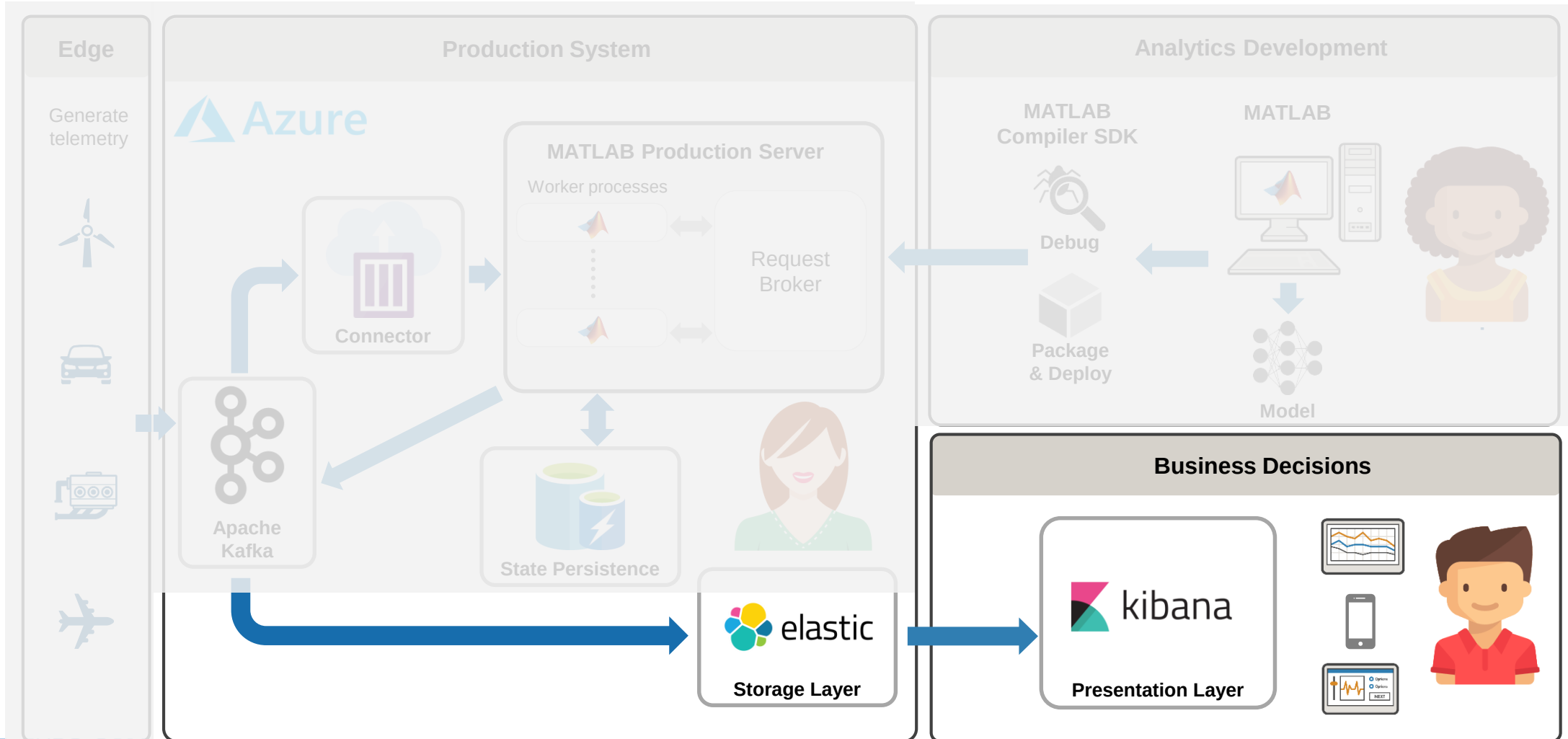


4

Integrate with
Production
Systems

System
Architect

Complete your application



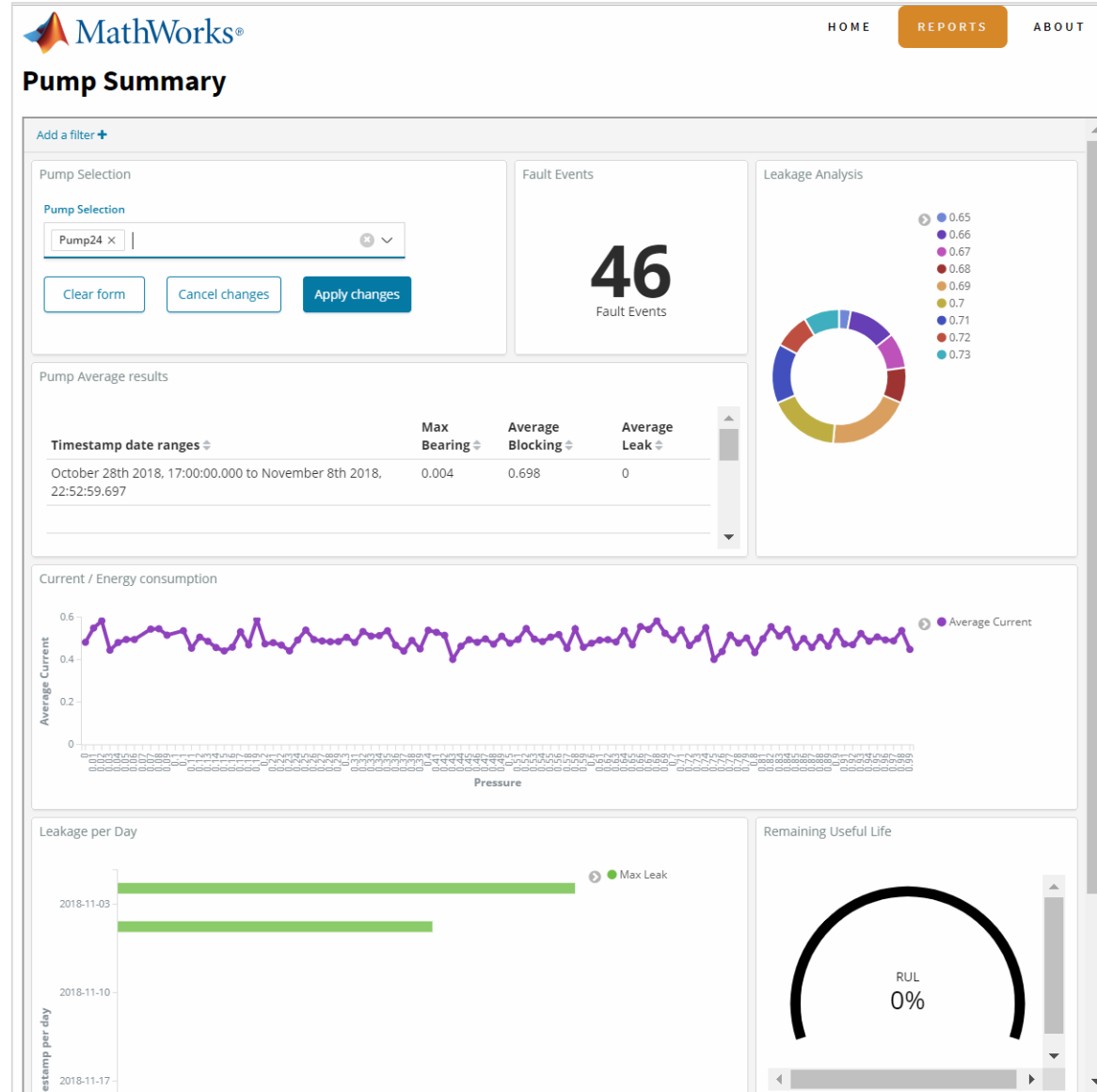


5

Visualize Results

Plant
Operator

Complete Your Application

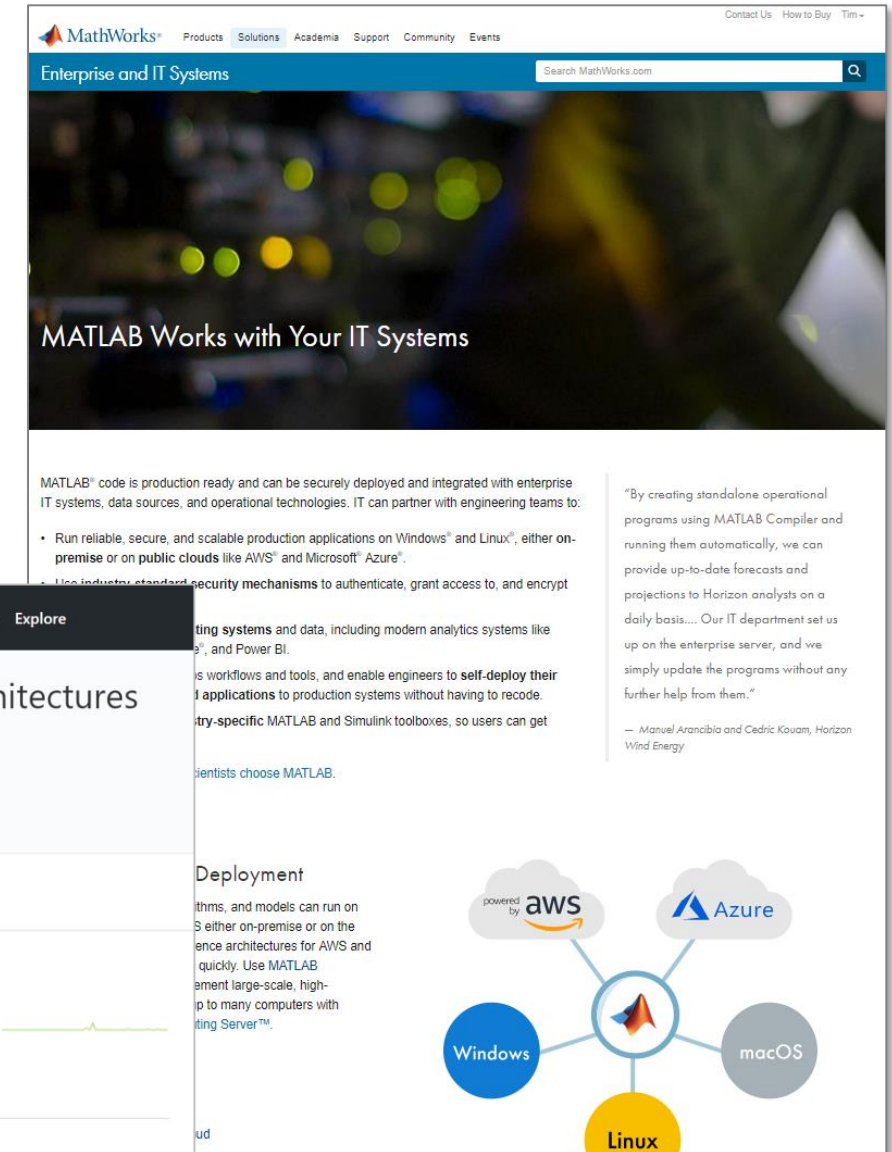


Team Retrospective

- Completed demo of full system in 3 week sprint
- Successfully used digital twin to generate faults and train models
- Fast prototyping of physical and AI models with MATLAB and Simulink.
Easy integration with OSS
- Cloud platform enabled faster IT setup
- Next steps:
 - Make model adjustments
 - Test against real pump
 - Customize dashboard for Operator's needs

Resources to learn and get started

- [GitHub: MathWorks Reference Architectures](#)
- [Working with Enterprise IT Systems](#)
- [Data Analytics with MATLAB](#)
- [Simulink](#)



The image shows two overlapping screenshots. The background screenshot is the MathWorks website, specifically the 'Enterprise and IT Systems' section. It features a dark header with the MathWorks logo and navigation links (Products, Solutions, Academia, Support, Community, Events). Below the header is a search bar and a large banner image with the text 'MATLAB Works with Your IT Systems'. The main content area discusses MATLAB's production readiness and integration with enterprise IT systems, mentioning its ability to run on Windows, Linux, and public clouds like AWS and Azure. A quote from Manuel Arancibia and Cedric Kouam of Horizon Wind Energy is also present.

The foreground screenshot is a GitHub repository page for 'MathWorks Reference Architectures'. It shows the repository name, a verified badge, and a link to the MathWorks cloud page. Below the repository name are tabs for 'Repositories' (6), 'People' (1), and 'Projects' (0). A search bar and filters for 'Type' and 'Language' are visible. Two repositories are listed: 'mdcs-on-azure' (Stand up a MATLAB Distributed Computing Server cluster using Azure Deployment) and 'mps-on-aws' (Stand up a MATLAB Production Server using CloudFormation).

At the bottom right of the foreground screenshot is a diagram titled 'Deployment' showing a central MATLAB logo connected to five cloud or OS icons: 'powered by aws', 'Azure', 'Windows', 'Linux', and 'macOS'.