

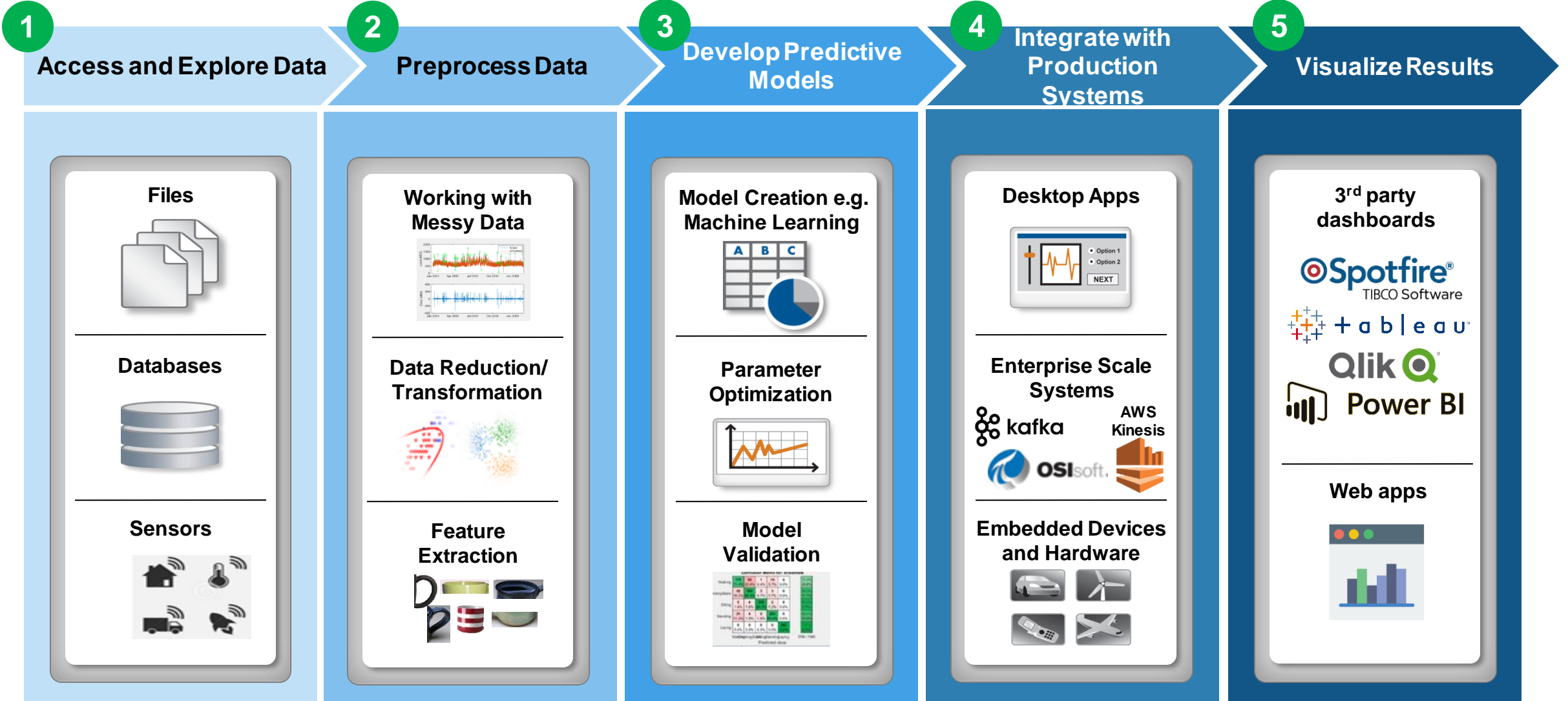
MathWorks
MATLAB
CONFERENCE 2018

Scaling up MATLAB
Analytics with Kafka and
Cloud Services

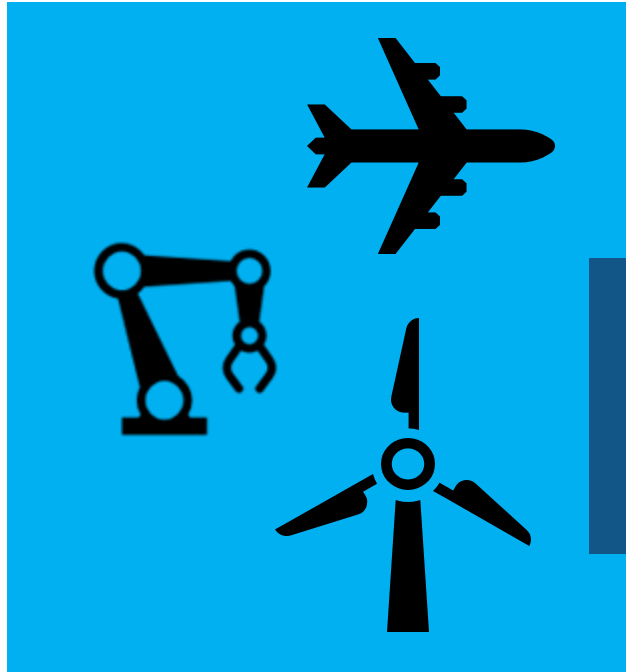
Branko Dijkstra



Agenda



The Need for Large-Scale Streaming



Predictive Maintenance

Increase Operational Efficiency
Reduce Unplanned Downtime

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

More applications require near real-time analytics

Medical Devices

Patient Safety
Better Treatment Outcomes

Connected Cars

Safety, Maintenance
Advanced Driving Features



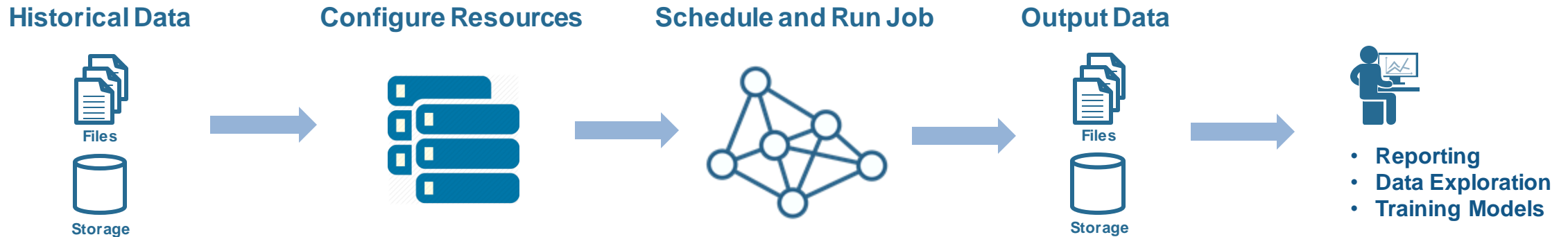
Car: ~25 GB per hour

4

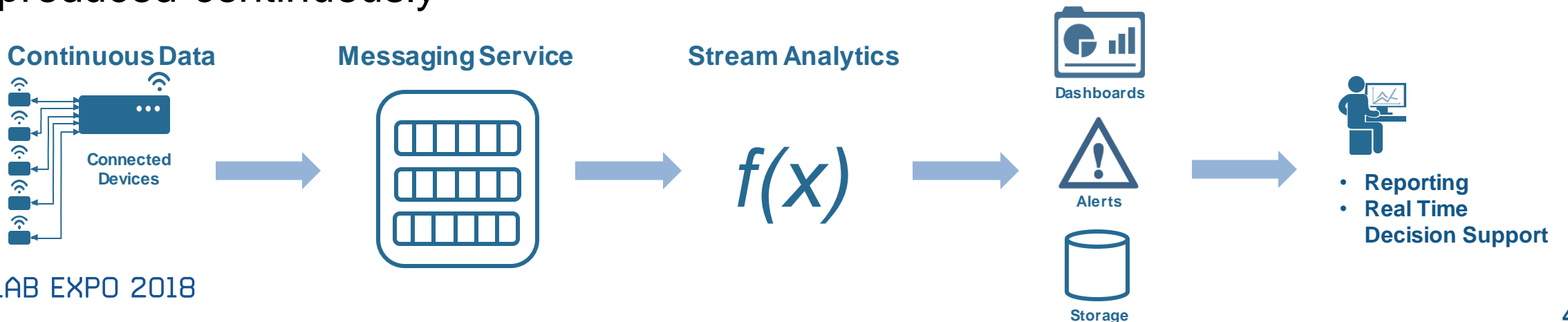
Integrate with
Production
Systems

A quick Intro to Stream Processing

- **Batch Processing** applies computation to a finite sized historical data set that was acquired in the past



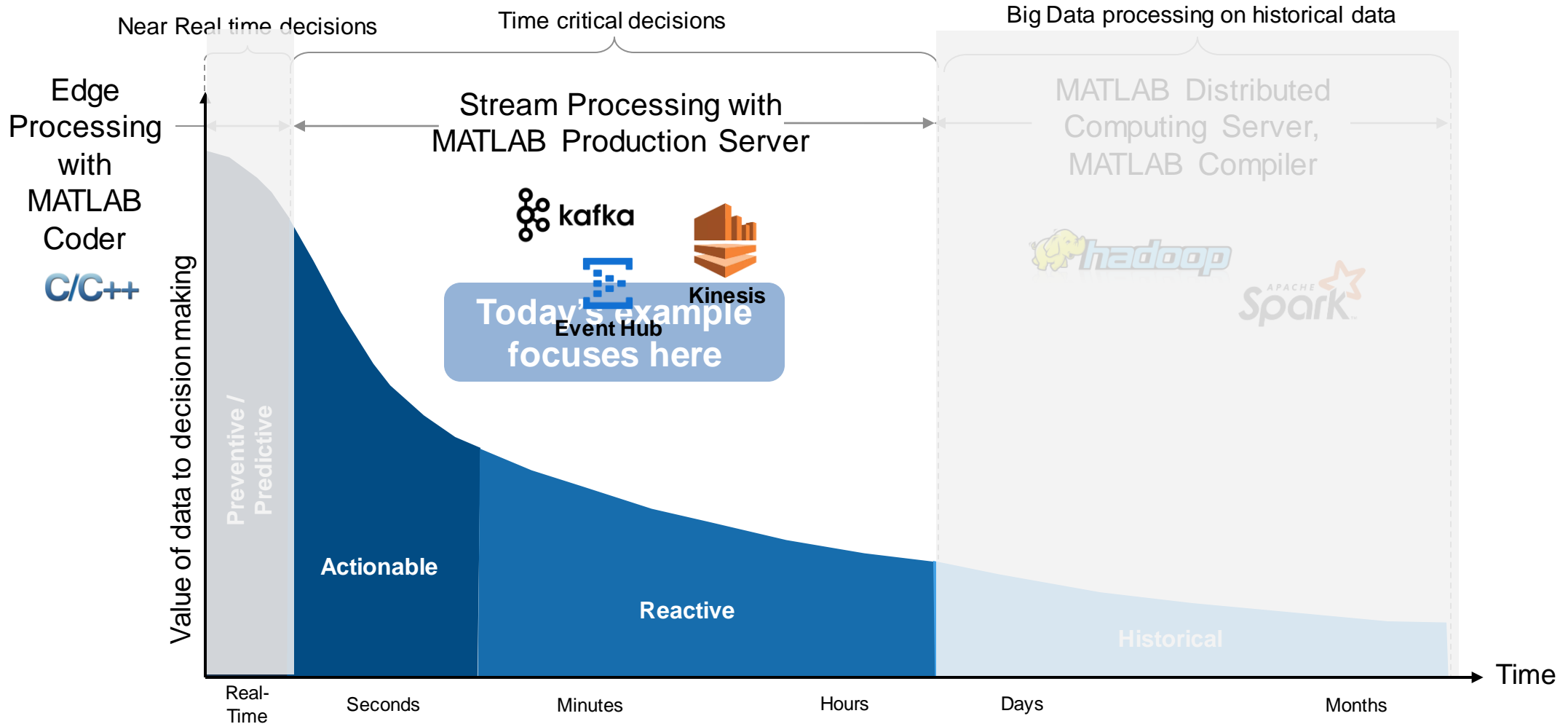
- **Stream Processing** applies computation to an unbounded data set that is produced continuously



4

Integrate with
Production
Systems

Why stream processing?

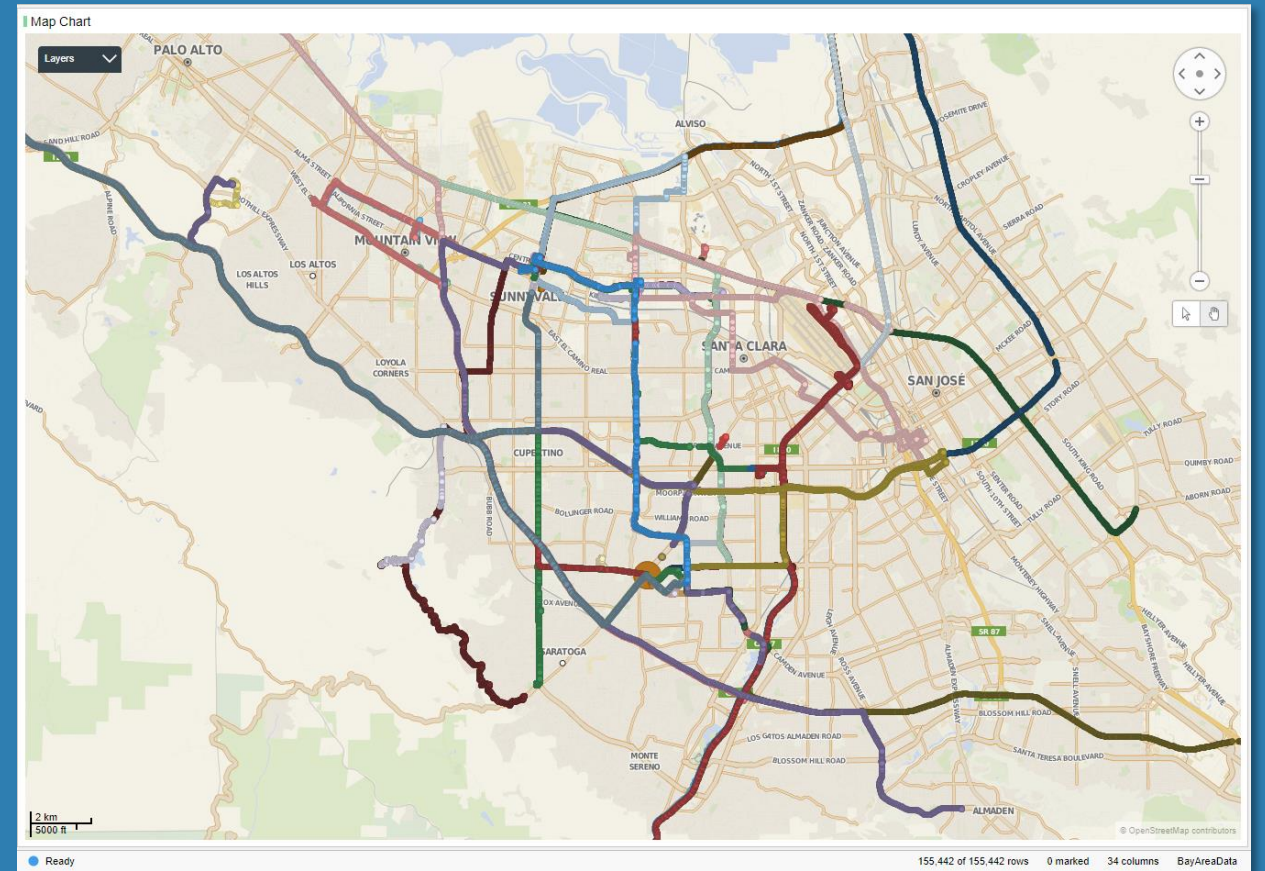
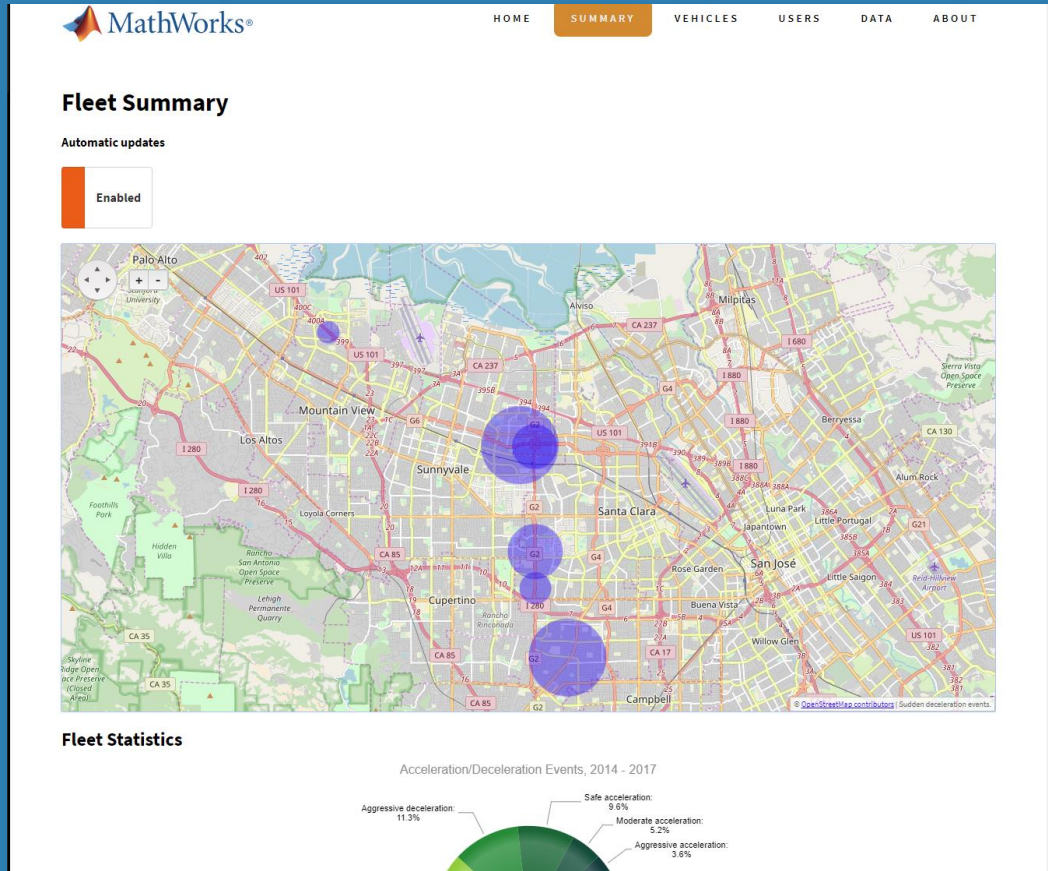


Example Problem – How's my driving?

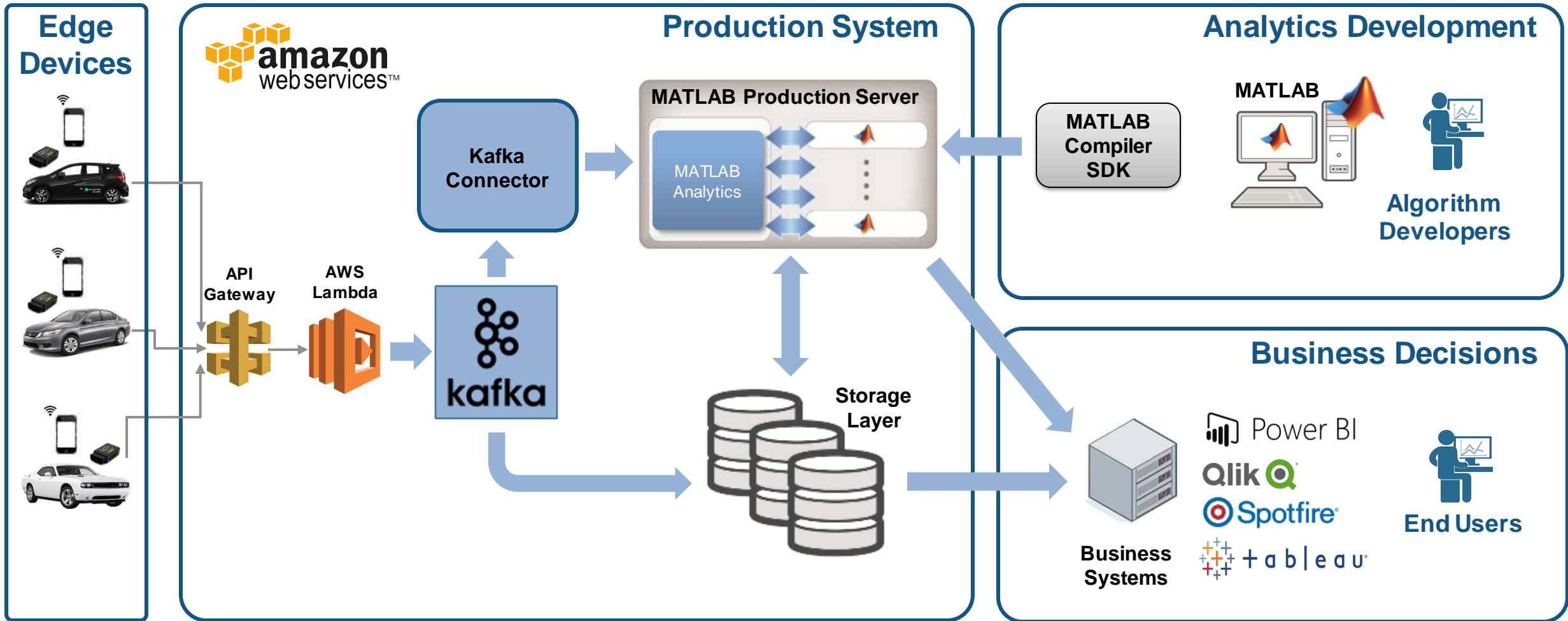
- A group of MathWorks employees installed an OBD dongle in their car that monitors the on-board systems
- Data is streamed to the cloud where it is aggregated and stored
- We would like to use this data to score the driving habits of participants



Example: Fleet Analytics with MATLAB



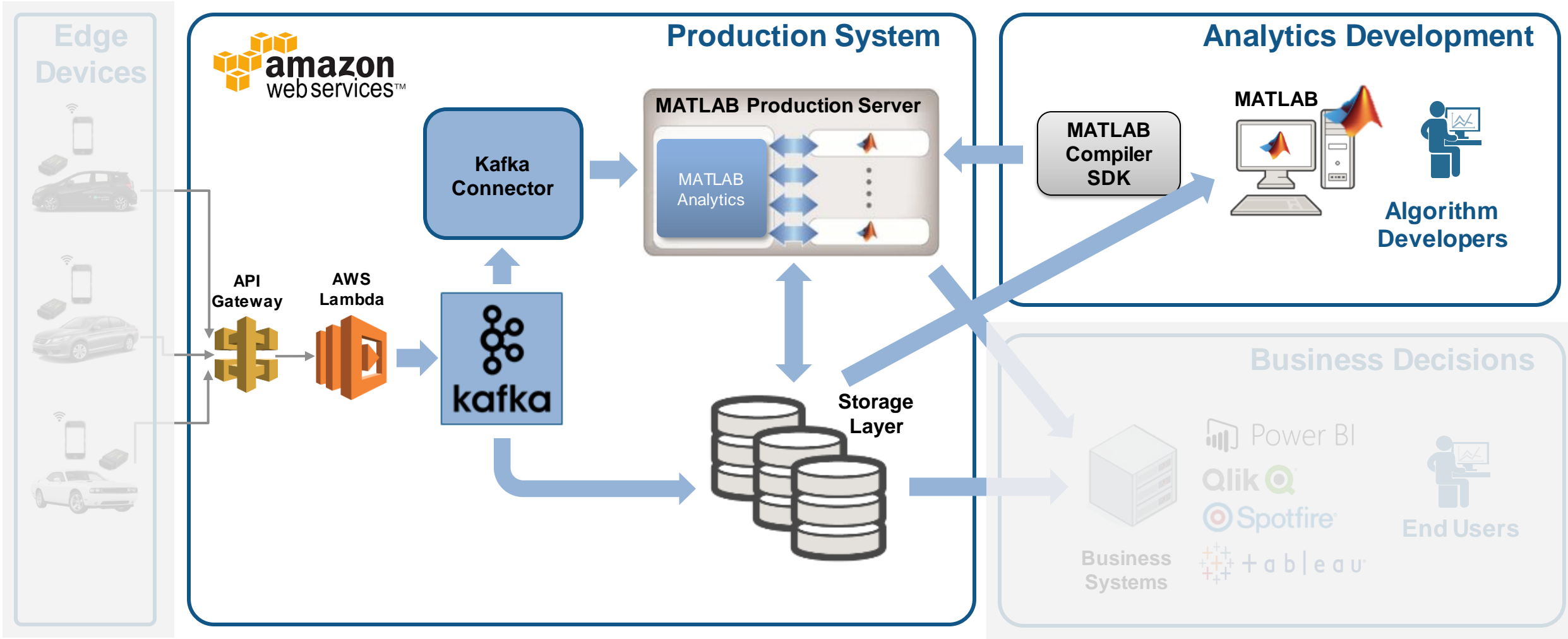
Fleet Analytics Architecture



1

Access and Explore Data

The first step is to clean up the incoming data



1

Access and Explore Data

The Data: Timestamped messages with JSON encoding



```

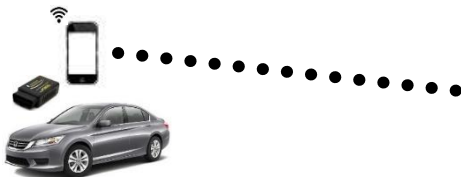
{
  "vehicles id": {"$oid":"55a3fd0069702d5b4100000"},
  "time" : {"$date":"2015-07-13T18:01:35.000Z"},
  "kc" : 1975.0, "kff1225" : 100.65293, "kff125a" : 110.36619, ...
}

```

Key

Timestamp

Values



```

{
  "vehicles_id": {"$oid":"55a3fe3569702d5c5c000020"},
  "time":{"$date":"2015-07-13T18:01:53.000Z"},
  "kc" : 2000.0, "kff1225" : 109.65293, "kff125a" : 115.36619,
  ...
}

```



```

{
  "vehicles_id": {"$oid":"55a4193569702d115b000001"},
  "time":{"$date":"2015-07-12T19:04:04.000Z"},
  "kc":2200.0, "kff1225" : 112.65293, "kff125a" : 112.36619,
  ...
}

```

1

Access and Explore Data

Access a Sample of Data

Raw Data

	timestamp	1 value	2 key
1	15-Jan-2015 22:12:23	'{"_id": {"\$oid": "55a41cb069702d115b059ee0"}, "trip_id": {"\$oid": "55a41cb069702d115b059ede"}}	'55a41cb069702d115b059ede'
2	15-Jan-2015 22:12:24	'{"_id": {"\$oid": "55a41cb069702d115b059ee1"}, "trip_id": {"\$oid": "55a41cb069702d115b059ede"}}	'55a41cb069702d115b059ede'
3	15-Jan-2015 22:12:25	'{"_id": {"\$oid": "55a41cb069702d115b059ee2"}, "trip_id": {"\$oid": "55a41cb069702d115b059ede"}}	'55a41cb069702d115b059ede'
4	15-Jan-2015 22:12:26	'{"_id": {"\$oid": "55a41cb069702d115b059ee3"}, "trip_id": {"\$oid": "55a41cb069702d115b059ede"}}	'55a41cb069702d115b059ede'

- ✓ Decode JSON data
- ✓ Create Timetable



Timetable

t = 4647x40 timetable

	trip_id	VIN	kff1001	kff1005	kff1006	kff1220	kff1221	kff1222	kff1223	kff125a
1 Sun Jul 12 16:18:41 UTC 2015	55a3fe356...	55a3fe356...	17.1000	-84.9323	45.4704	NaN	NaN	NaN	NaN	59.0434
2 Sun Jul 12 16:18:42 UTC 2015	55a3fe356...	55a3fe356...	17.1000	-84.9322	45.4704	NaN	NaN	NaN	NaN	57.8609
3 Sun Jul 12 16:18:43 UTC 2015	55a3fe356...	55a3fe356...	18.9000	-84.9322	45.4705	NaN	NaN	NaN	NaN	52.7147
4 Sun Jul 12 16:18:44 UTC 2015	55a3fe356...	55a3fe356...	18.9000	-84.9322	45.4705	NaN	NaN	NaN	NaN	51.1983
5 Sun Jul 12 16:18:45 UTC 2015	55a3fe356...	55a3fe356...	18.0000	-84.9321	45.4706	NaN	NaN	NaN	NaN	49.1095
6 Sun Jul 12 16:19:13 UTC 2015	55a3fe356...	55a3fe356...	58.5000	-84.9305	45.4686	NaN	NaN	NaN	NaN	73.2005
7 Sun Jul 12 16:19:14 UTC 2015	55a3fe356...	55a3fe356...	56.7000	-84.9304	45.4685	NaN	NaN	NaN	NaN	75.3612
8 Sun Jul 12 16:19:15 UTC 2015	55a3fe356...	55a3fe356...	57.6000	-84.9304	45.4683	NaN	NaN	NaN	NaN	70.7542
9 Sun Jul 12 16:19:16 UTC 2015	55a3fe356...	55a3fe356...	56.7000	-84.9303	45.4682	NaN	NaN	NaN	NaN	62.8340

2

Preprocess Data

Develop a Preprocessing Function

Timetable

t = 4647x40 timetable

	trip_id	VIN	kff1001	kff1005	kff1006	kff1220	kff1221	kff1222	kff1223	kff125a	
1	Sun Jul 12 16:18:41 UTC 2015	55a3fe356...	55a3fe356...	17.1000	-84.9323	45.4704	NaN	NaN	NaN	NaN	59.0434
2	Sun Jul 12 16:18:42 UTC 2015	55a3fe356...	55a3fe356...	17.1000	-84.9322	45.4704	NaN	NaN	NaN	NaN	57.8609
3	Sun Jul 12 16:18:43 UTC 2015	55a3fe356...	55a3fe356...	18.9000	-84.9322	45.4705	NaN	NaN	NaN	NaN	52.7147
4	Sun Jul 12 16:18:44 UTC 2015	55a3fe356...	55a3fe356...	18.9000	-84.9322	45.4705	NaN	NaN	NaN	NaN	51.1983
5	Sun Jul 12 16:18:45 UTC 2015	55a3fe356...	55a3fe356...	18.0000	-84.9321	45.4706	NaN	NaN	NaN	NaN	49.1095
6	Sun Jul 12 16:19:13 UTC 2015	55a3fe356...	55a3fe356...	58.5000	-84.9305	45.4686	NaN	NaN	NaN	NaN	73.2005
7	Sun Jul 12 16:19:14 UTC 2015	55a3fe356...	55a3fe356...	56.7000	-84.9304	45.4686	NaN	NaN	NaN	NaN	73.2005
8	Sun Jul 12 16:19:15 UTC 2015	55a3fe356...	55a3fe356...	57.6000	-84.9304	45.4686	NaN	NaN	NaN	NaN	73.2005
9	Sun Jul 12 16:19:16 UTC 2015	55a3fe356...	55a3fe356...	56.7000	-84.9303	45.4686	NaN	NaN	NaN	NaN	73.2005



Preprocess data

```
t = sortrows(t);
t = rmmissing(t, 'MinNumMissing', width(t)-2);
```

Perform windowed calculations

```
t.Speed = movmedian(t.SpeedGPS, 3);
t.D1 = [0; diff(t.SpeedGPS)];
```

```
[tmin, tmax] = bounds(t.time);
tnew = tmin:seconds(10):tmax;
countsByTime = retime(t(:, 'Event'), tnew, @histcounts);
```

- ✓ Clean up
- ✓ Enrich
- ✓ Restructure

1

Access and Explore Data

Ad Hoc Access to Data from MATLAB

```
athenaQuery.mlx x +
```

Access the data in S3

Bring up the AthenaClient

```
athenaClient = aws.athena.Client();  
athenaClient.Database = 'trainingdata';  
athenaClient.initialize();
```

Create a query and submit

```
athenaClient.submitQuery('SELECT * FROM "trainingdata"."sampledata" limit 100', 's3://fleettrainingdata')
```

Fetch data as a table for easy analysis

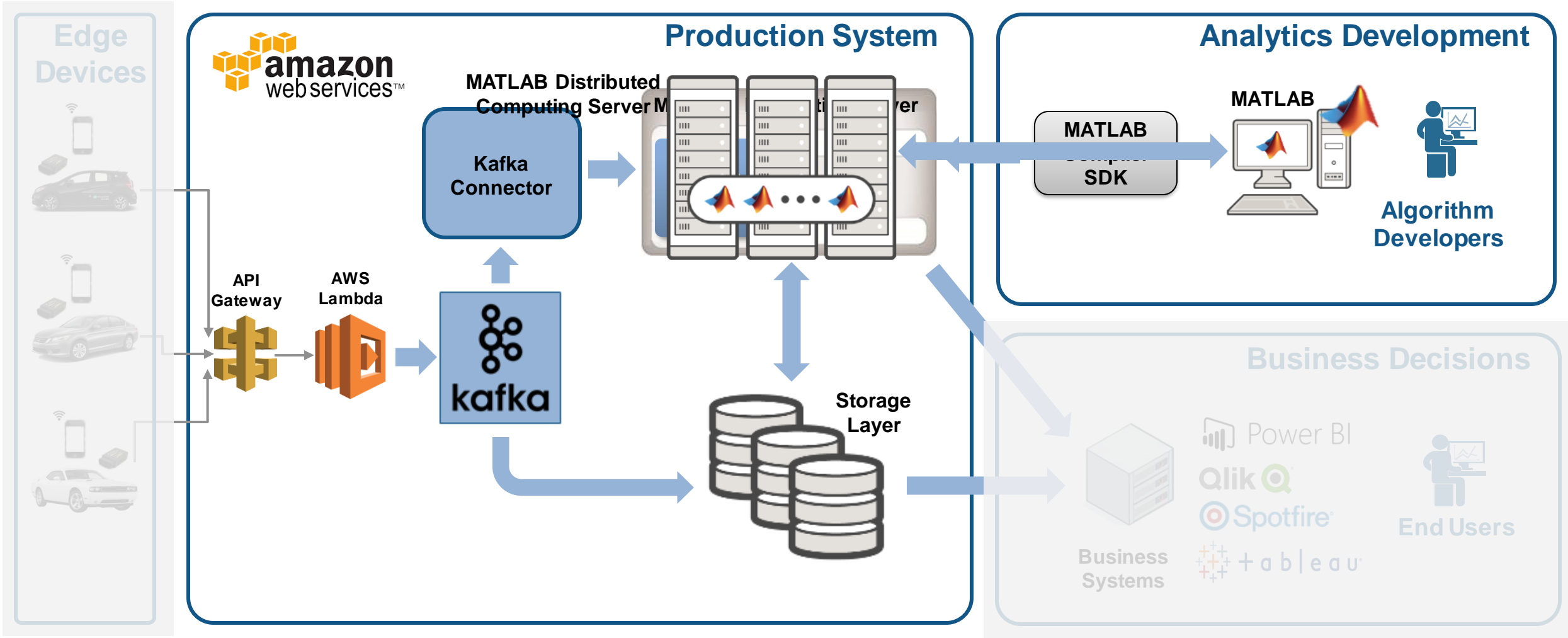
```
ds = datastore('s3://fleettrainingdata/*.csv');  
ds.NumHeaderLines = 2;  
data = table(ds);
```

Your usual MATLAB workflow goes here

3

Develop Predictive Models

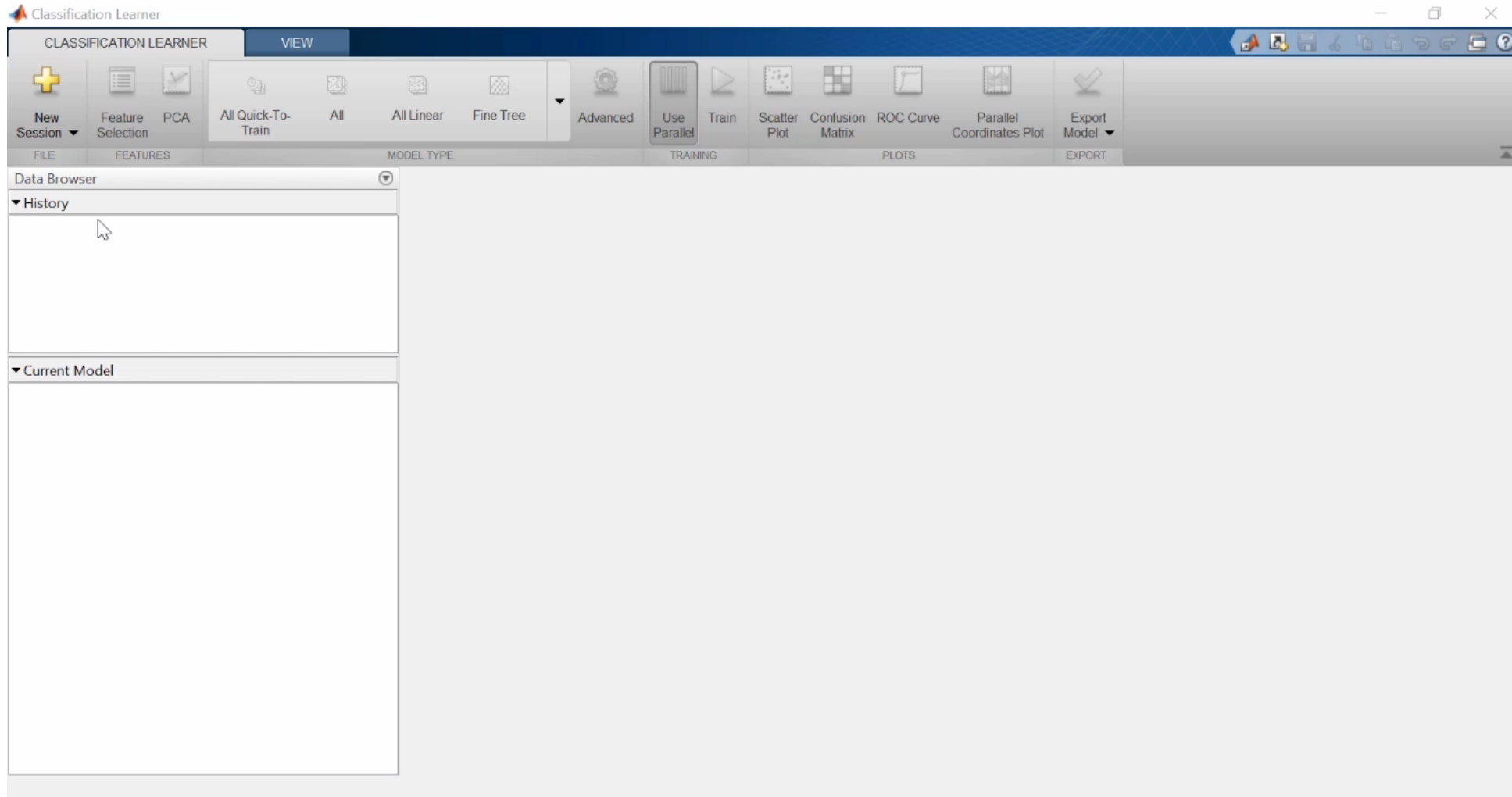
Develop a Predictive Model



3

Develop Predictive Models

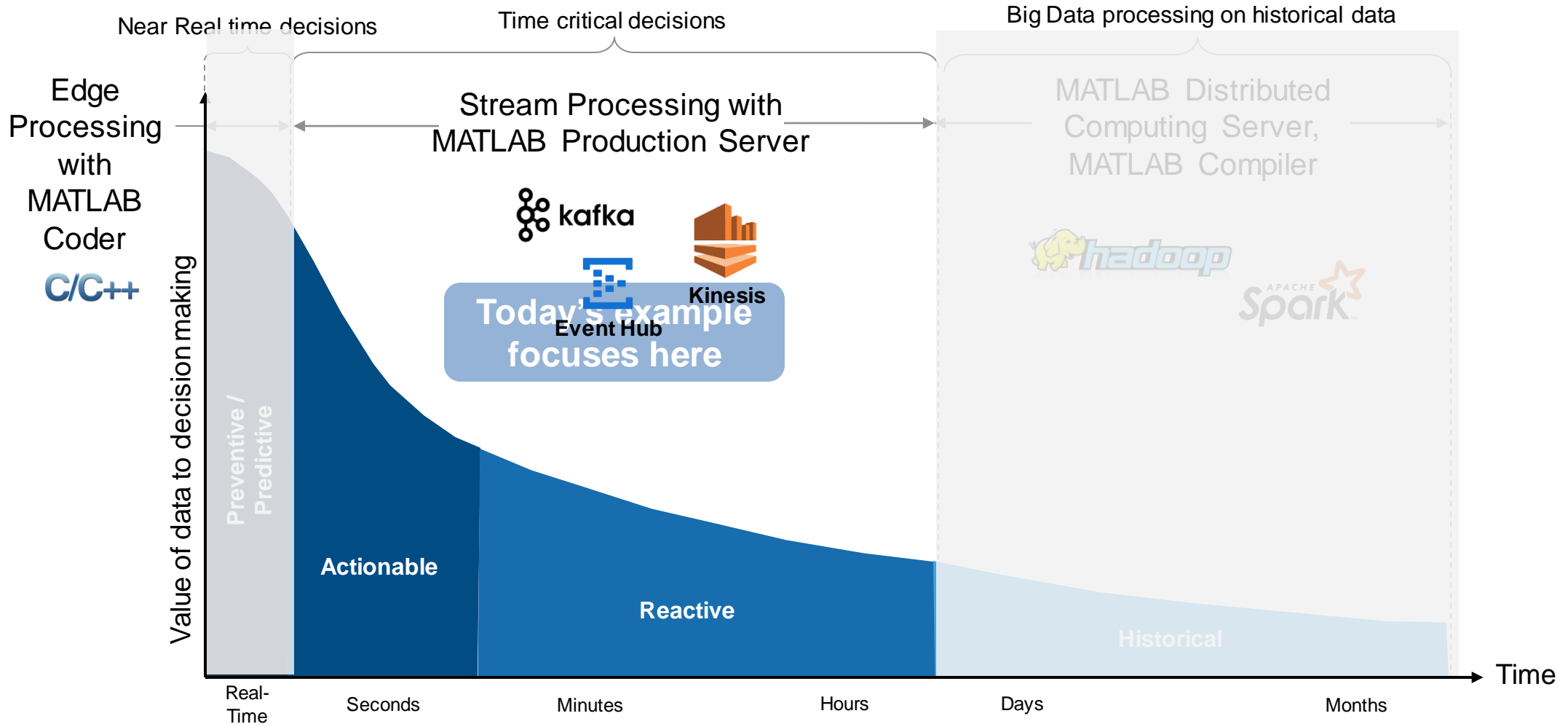
Develop a Predictive Model in MATLAB



4

Integrate with
Production
Systems

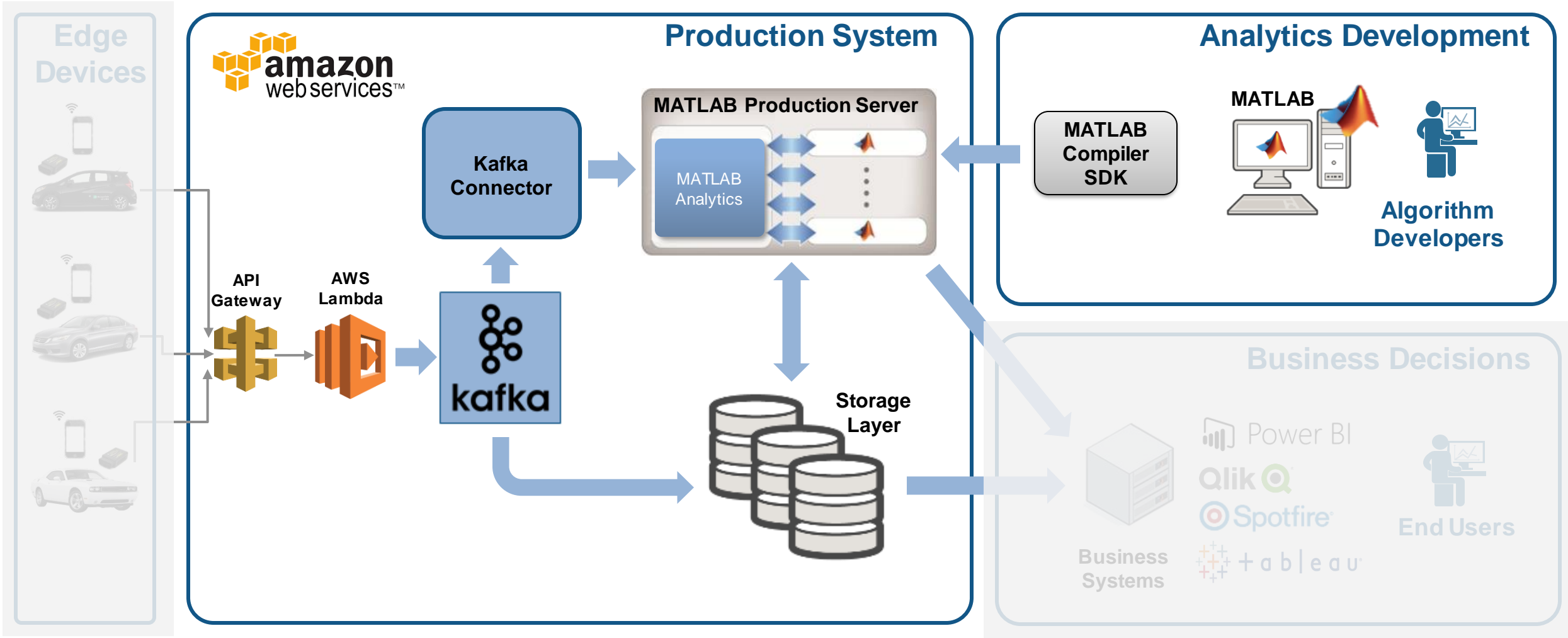
Why stream processing?



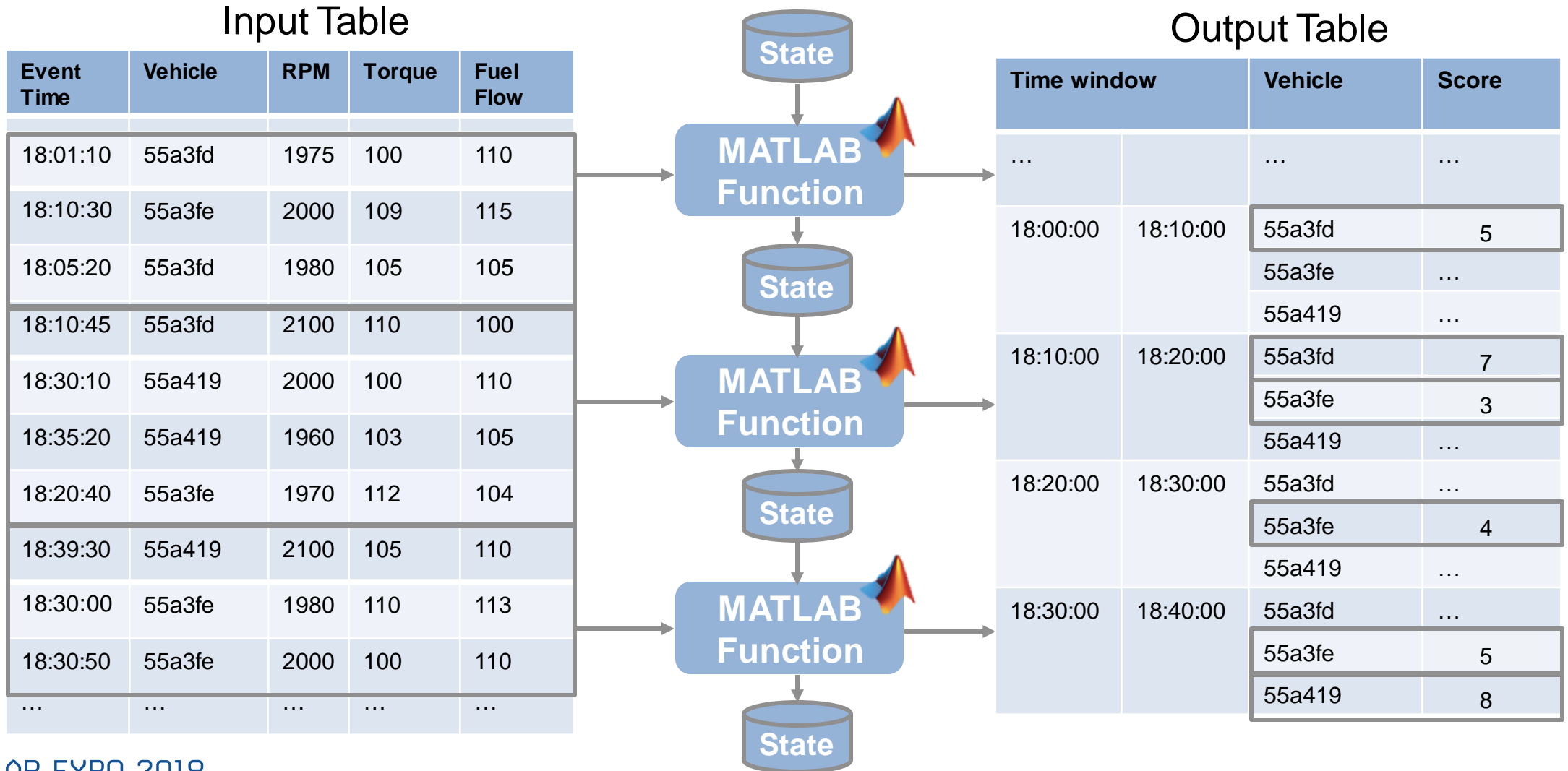
4

Integrate with
Production
Systems

Integrate Analytics with Production Systems



Streaming data is treated as an unbounded Timetable



4

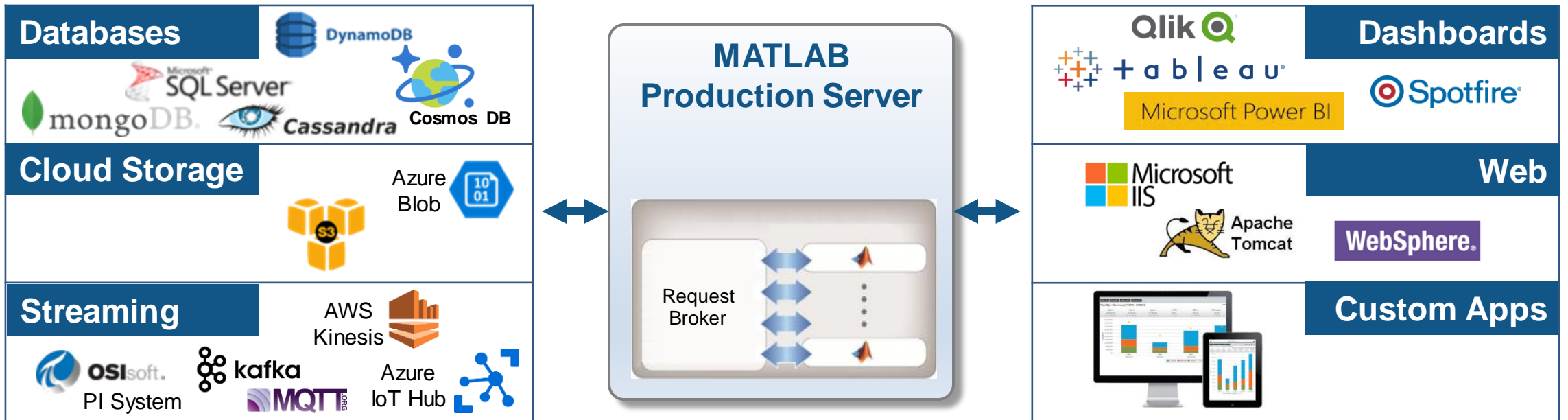
Integrate with
Production
Systems

Introducing MATLAB Production Server

Data

Analytics

Business System

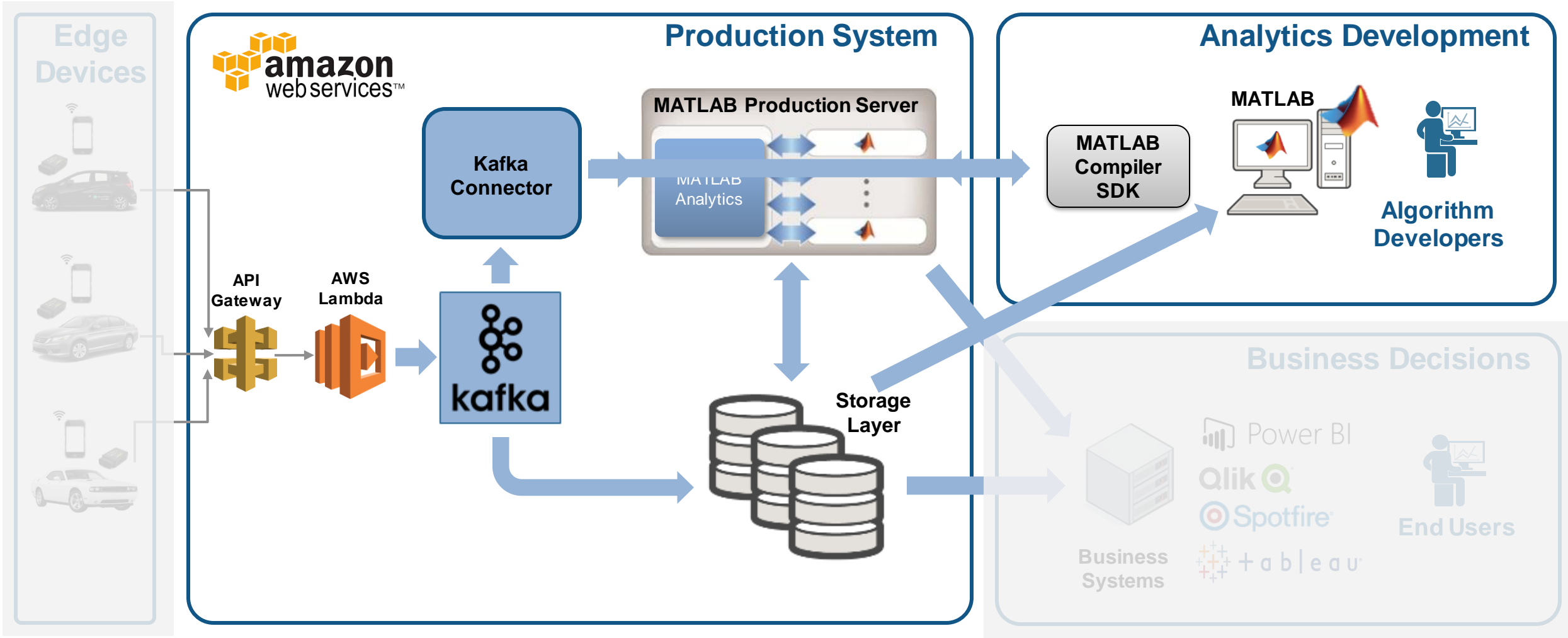


Platform

4

Integrate with
Production
Systems

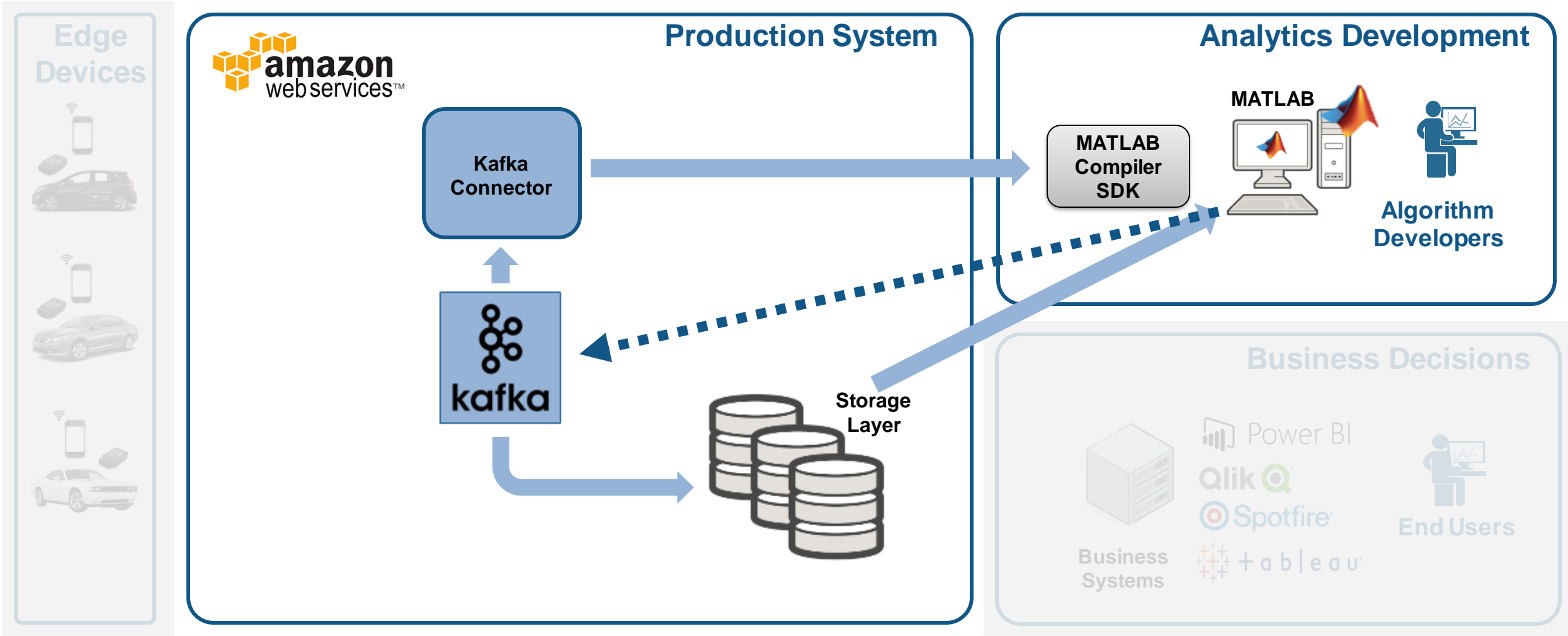
Develop and Deploy a Stream Processing Function



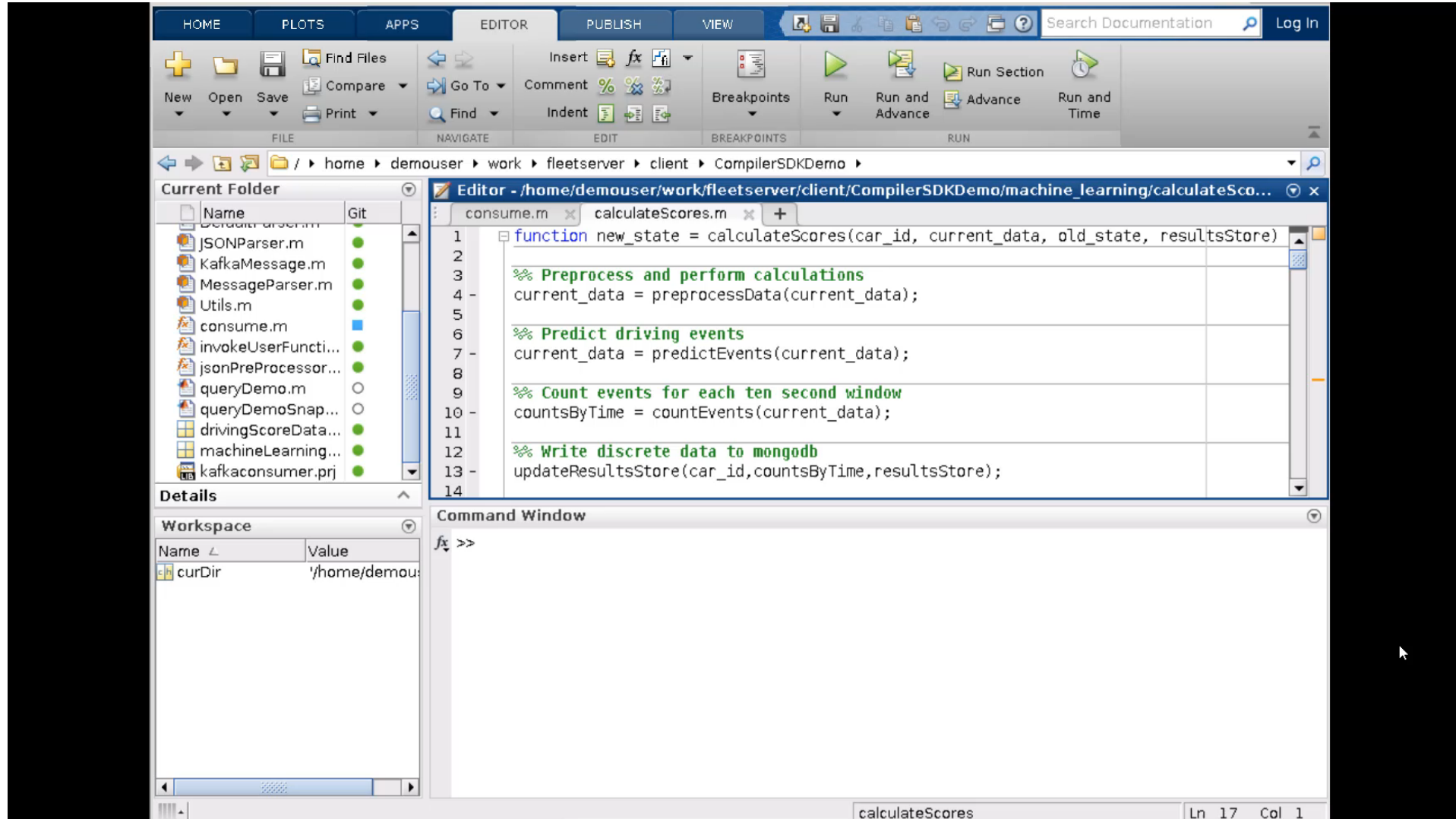
4

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, Print), navigation (Go To, Find), editing (Insert, Comment, Indent), and execution (Run, Run and Advance, Run and Time). The current folder is `/home/demouser/work/fleetserver/client/CompilerSDKDemo`. The file explorer on the left shows a list of files including `JSONParser.m`, `KafkaMessage.m`, `MessageParser.m`, `Utils.m`, `consume.m`, `invokeUserFuncti...`, `jsonPreProcessor...`, `queryDemo.m`, `queryDemoSnap...`, `drivingScoreData...`, `machineLearning...`, and `kafkaconsumer.prj`. The workspace shows a variable `curDir` with the value `'/home/demou...`. The editor window shows the `calculateScores.m` file with the following code:

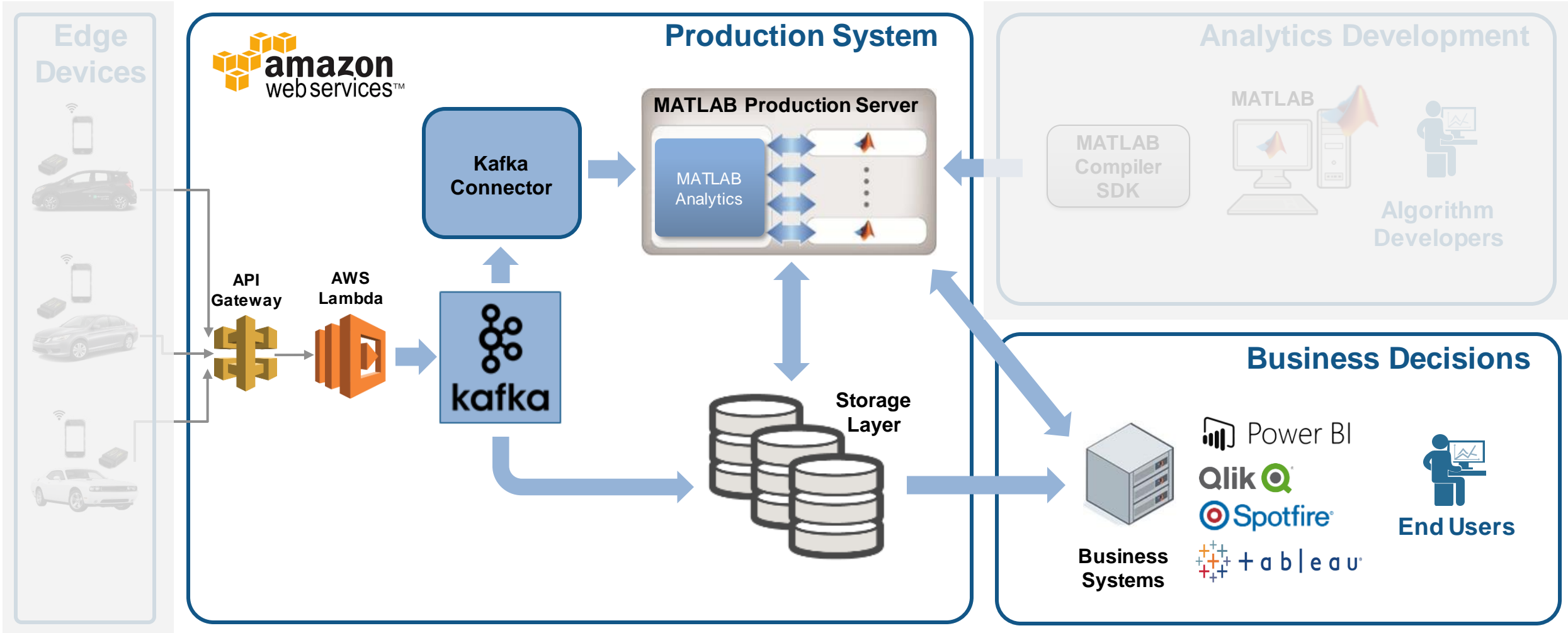
```
1 function new_state = calculateScores(car_id, current_data, old_state, resultsStore)
2
3 % Preprocess and perform calculations
4 current_data = preprocessData(current_data);
5
6 % Predict driving events
7 current_data = predictEvents(current_data);
8
9 % Count events for each ten second window
10 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.

4

Integrate with
Production
Systems

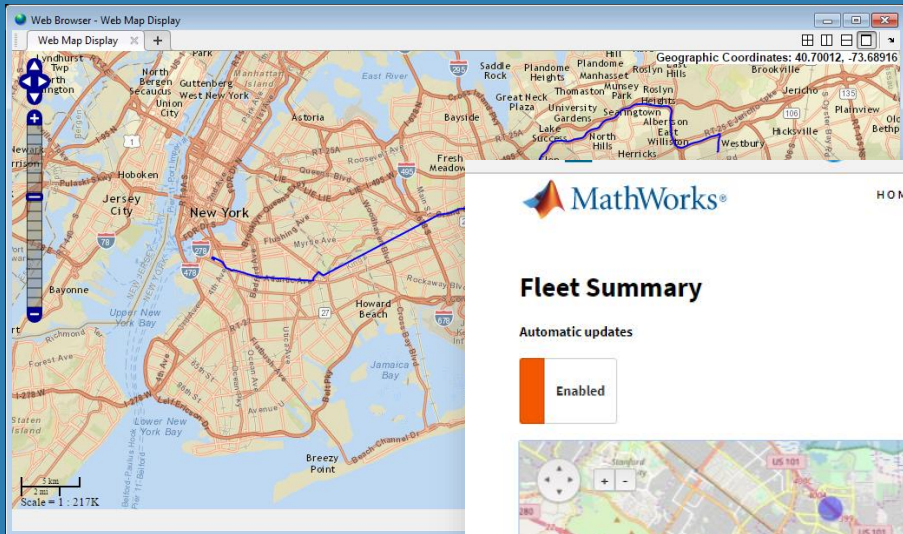
Tie in your Dashboard Application



5

Visualize Results

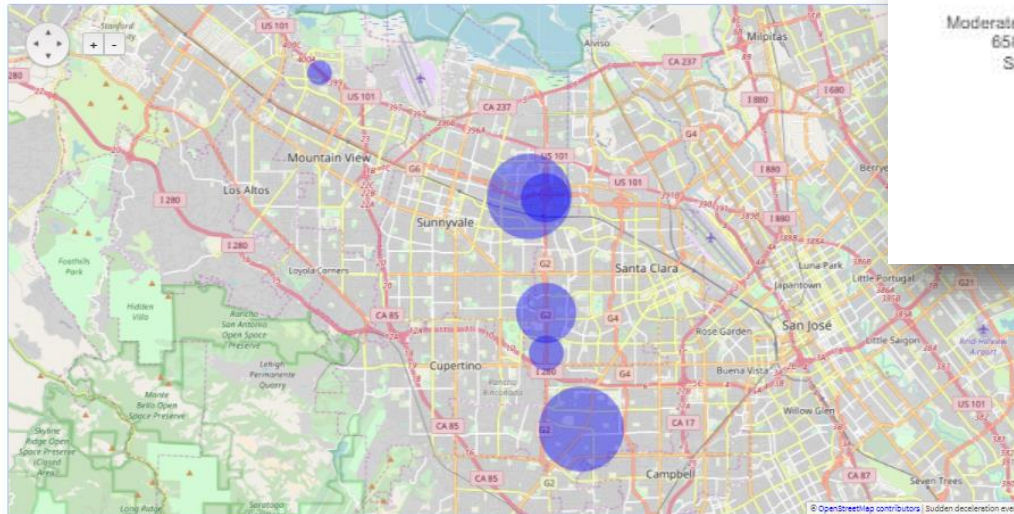
Complete Your Application



HOME SUMMARY VEHICLES USERS TRIPS REPORT

Fleet Summary

Automatic updates

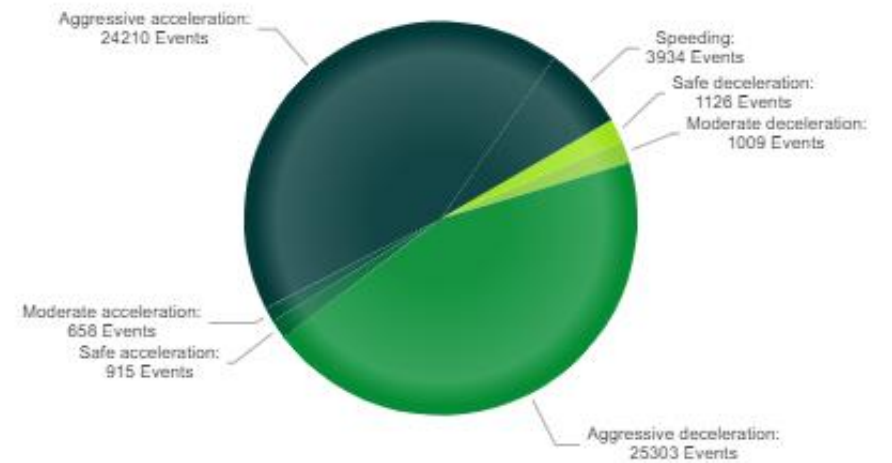


Fleet Statistics

Total Events:

193351

Acceleration/Deceleration Events, 2014 - 2017

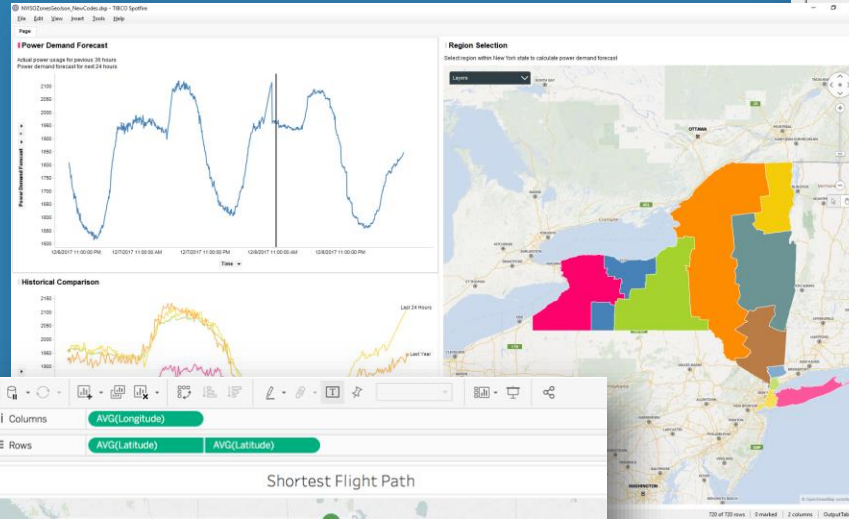


5

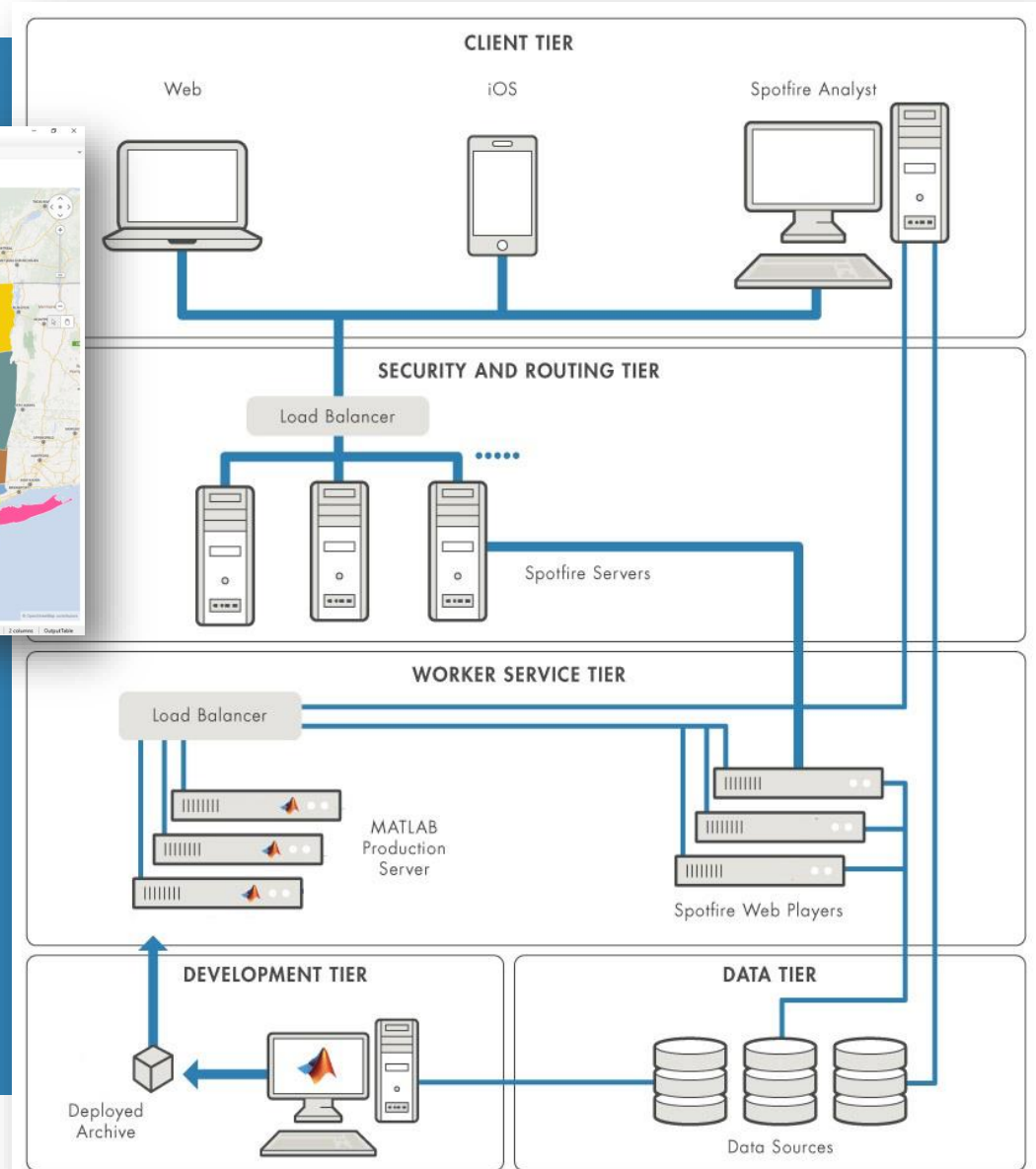
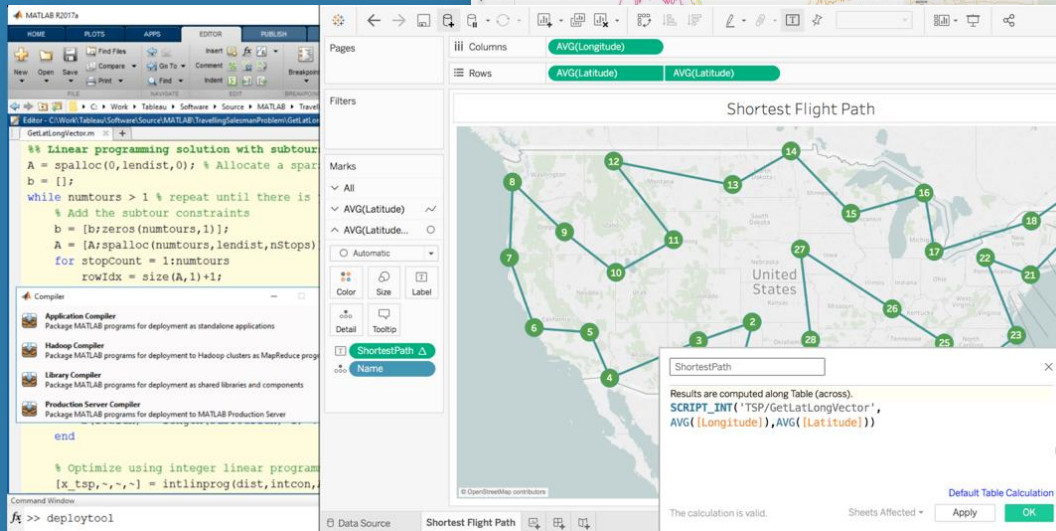
Visualize Results

Scalable Analytics with Enterprise BI Tools

TIBCO Spotfire



Tableau



Key Takeaways

- MATLAB connects directly to your data so you can quickly design and validate algorithms
- The MATLAB language and apps enable fast design iterations
- MATLAB Production Server enables easy integration of your MATLAB algorithms with enterprise production systems
- You to spend your time understanding the data and designing algorithms

Resources to learn and get started

- [Data Analytics with MATLAB](#)
- [MATLAB Production Server](#)
- [MATLAB Compiler SDK](#)
- [Statistics and Machine Learning Toolbox](#)
- [Database Toolbox](#)
- [Mapping Toolbox](#)
- [MATLAB with TIBCO Spotfire](#)
- [MATLAB with Tableau](#)
- [MATLAB with MongoDB](#)

The screenshot shows a MathWorks webpage titled "Reference Architecture" for "Scalable Analytics with TIBCO Spotfire and MATLAB Production Server". The page includes a navigation bar with links for Products, Solutions, Academia, Support, Community, and Events. A search bar is present in the top right. The main content area features the title and a brief description: "Resources and Spotfire extension to scale MATLAB analytics for use with Spotfire applications". Below this, there is a text block explaining the MATLAB Production Server™ Interface for TIBCO® Spotfire® Software as a Spotfire extension that provides a link to a robust and scalable MATLAB® analytics engine. Two call-to-action buttons are provided: "Request the Extension & Getting Started Guide" (with a "Submit request" button) and "Download the Free Technical Brief" (with a "Download now" button). On the right side, a diagram illustrates the architecture. It shows three client types: MOBILE, WEB, and DESKTOP (Mathworks MPSExtension). These clients connect through a Load Balancer to a TIBCO SPOTFIRE WEB PLAYER (Mathworks MPSExtension). This player then connects through another Load Balancer to the TIBCO SPOTFIRE SERVER, which in turn connects to the MATLAB PRODUCTION SERVER (MATLAB Analytics).