MathWorks

MATLAB CONFERENCE 2018

Scaling up MATLAB Analytics with Kafka and Cloud Services

Branko Dijkstra

Agenda

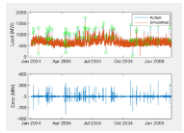
1. Access and Explore Data
2. Preprocess Data
3. Develop Predictive Models
4. Integrate with Production Systems
5. Visualize Results

Files

Databases

Sensors

Working with Messy Data



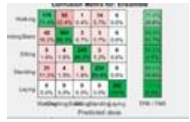
Data Reduction/ Transformation

Feature Extraction

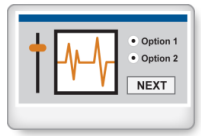
Model Creation e.g. Machine Learning

Parameter Optimization

Model Validation



Desktop Apps



Enterprise Scale Systems

kafka

AWS Kinesis

OSIsoft.

Embedded Devices and Hardware

3rd party dashboards

Spotfire® TIBCO Software

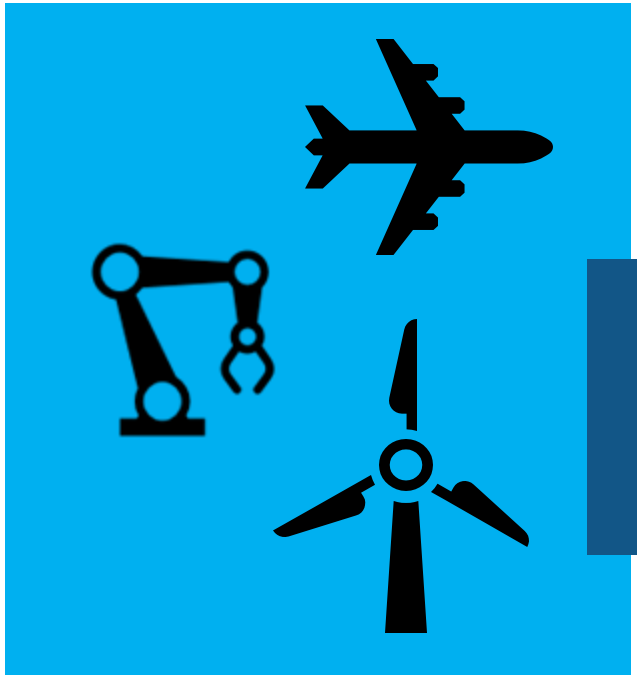
+ableau

Qlik

Power BI

Web apps

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

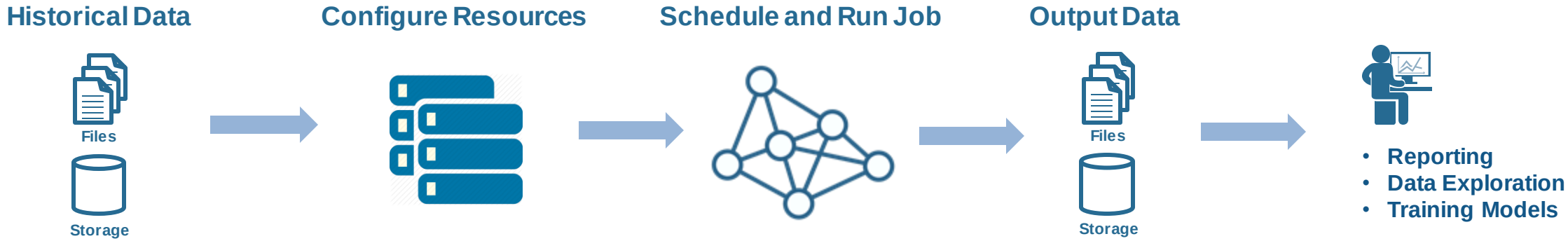
Safety, Maintenance

Advanced Driving Features

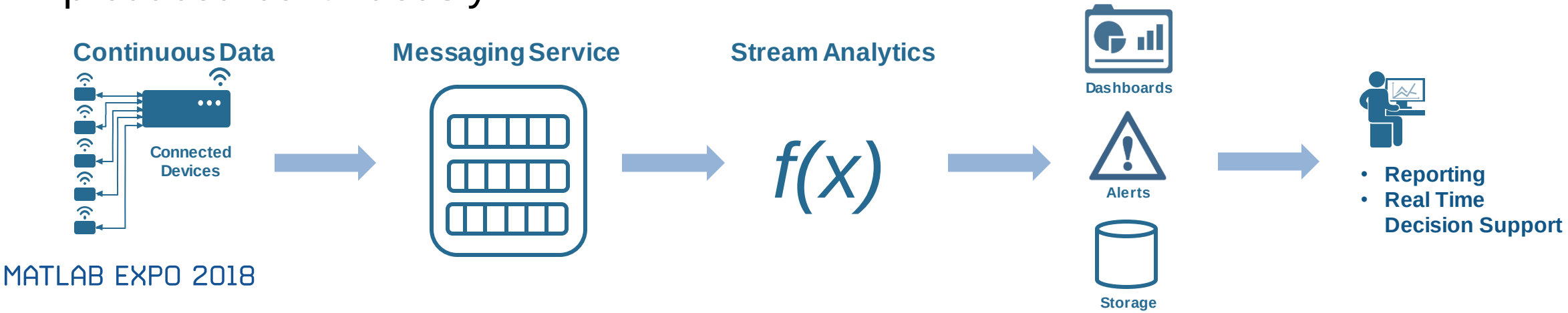
Car: ~25 GB per hour

A quick Intro to Stream Processing

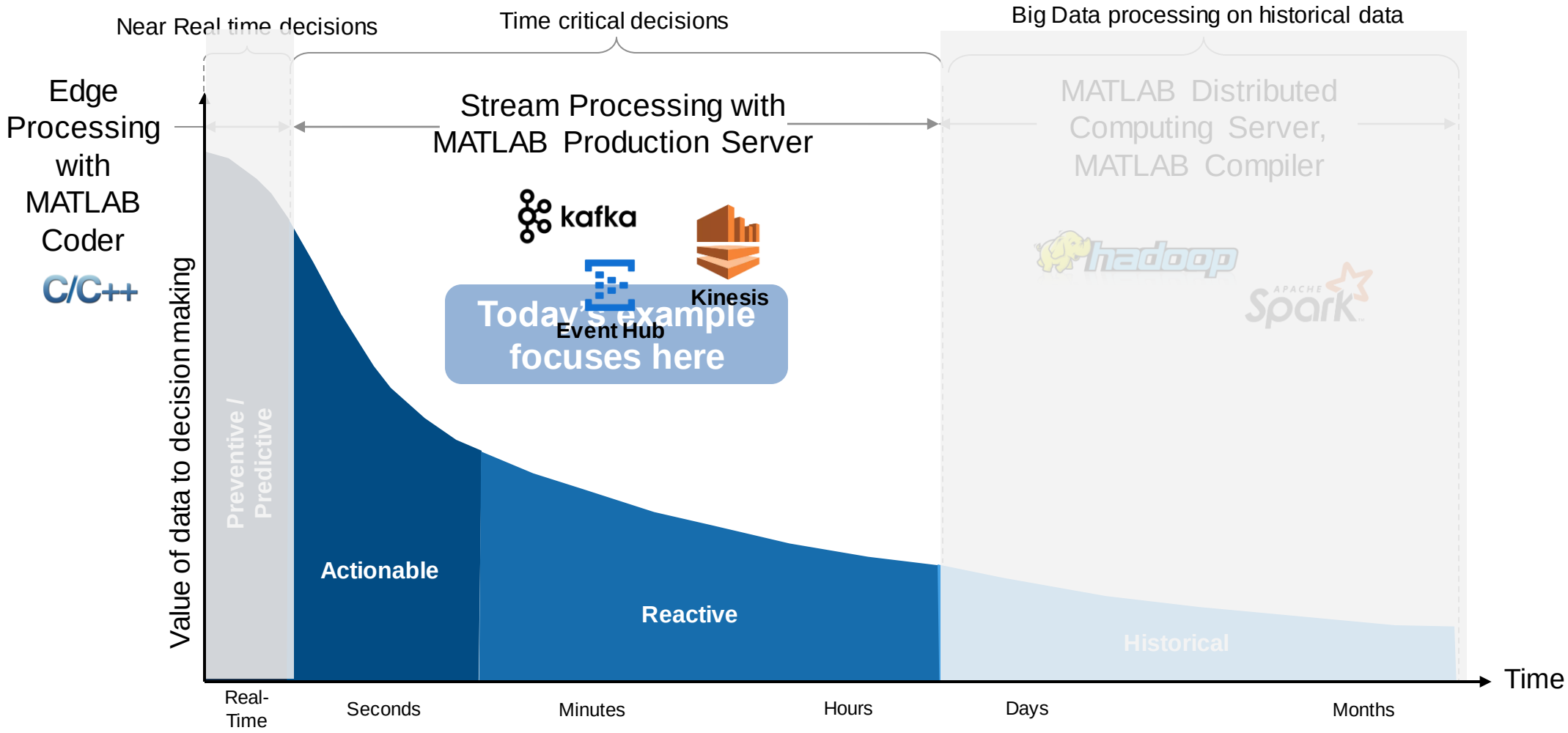
- ***Batch Processing*** applies computation to a finite sized historical data set that was acquired in the past



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



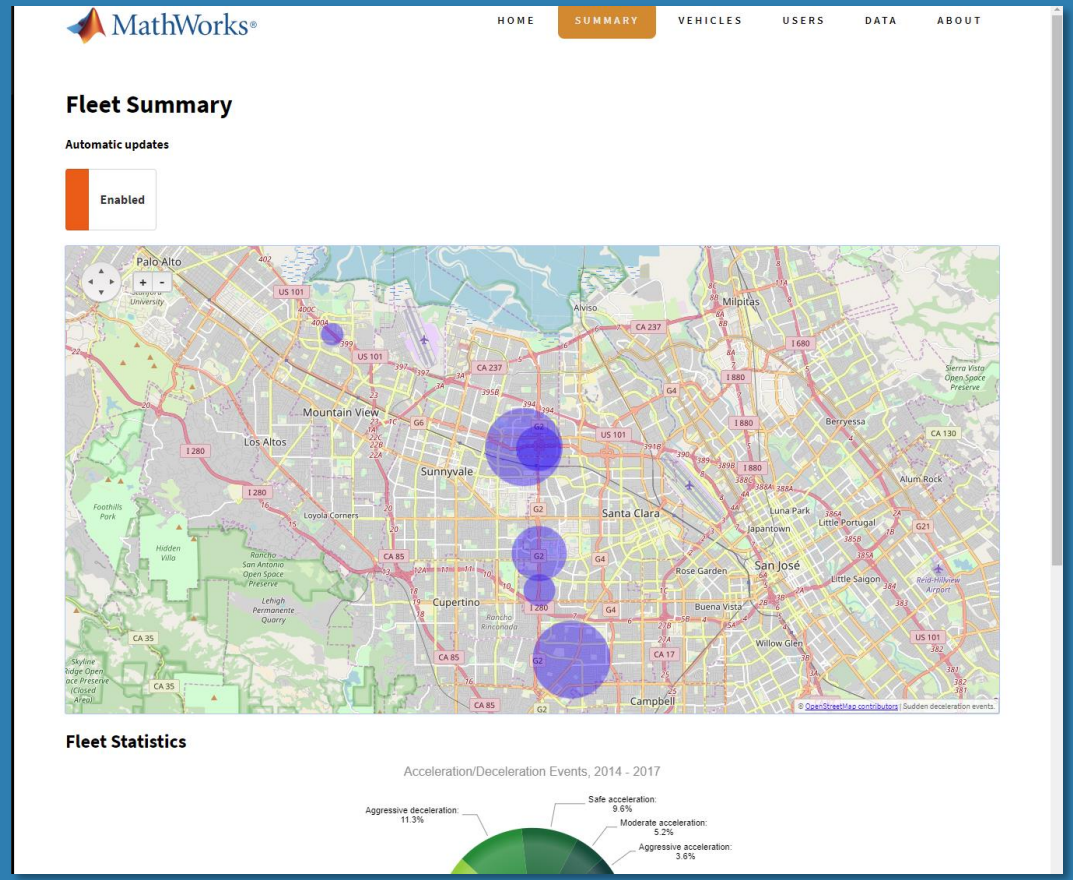
Why stream processing?

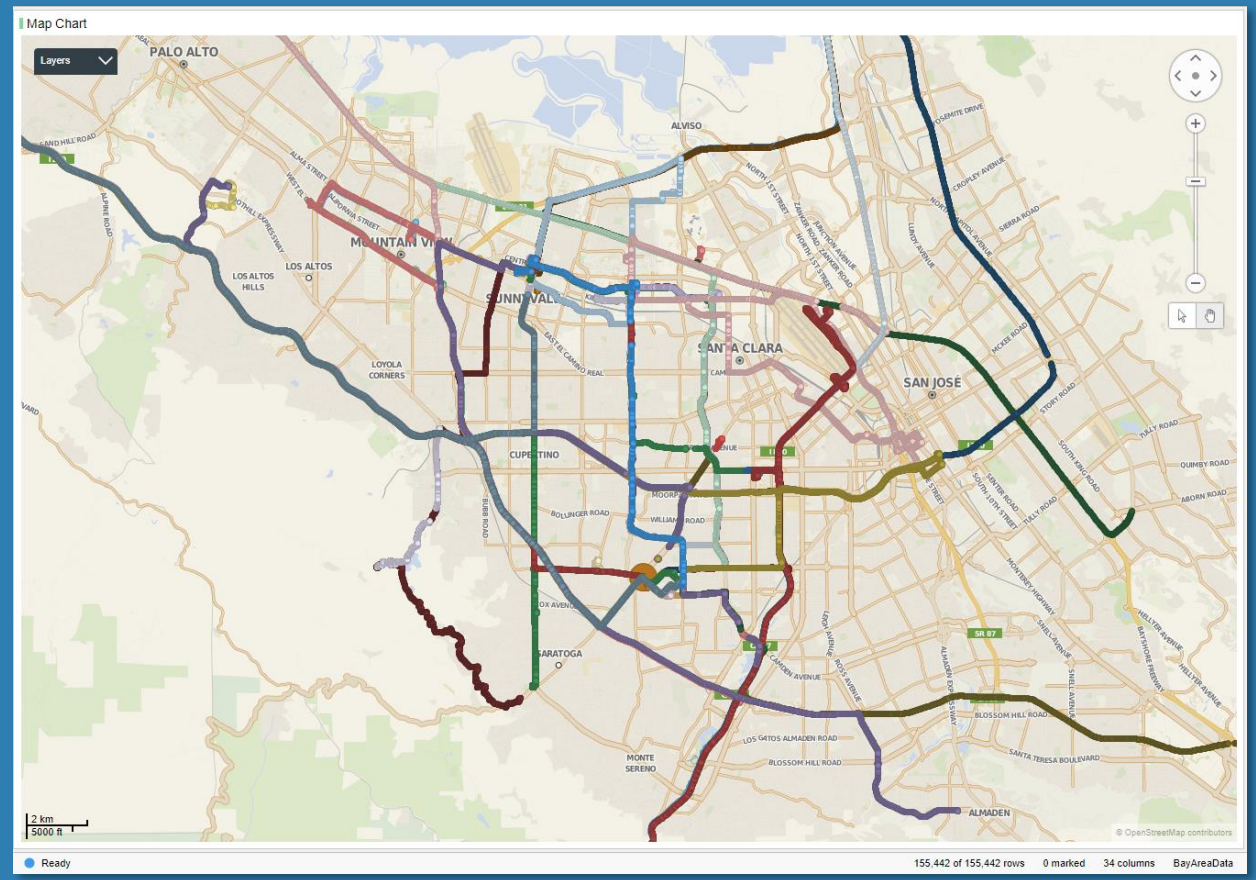


Example Problem – How's my driving?

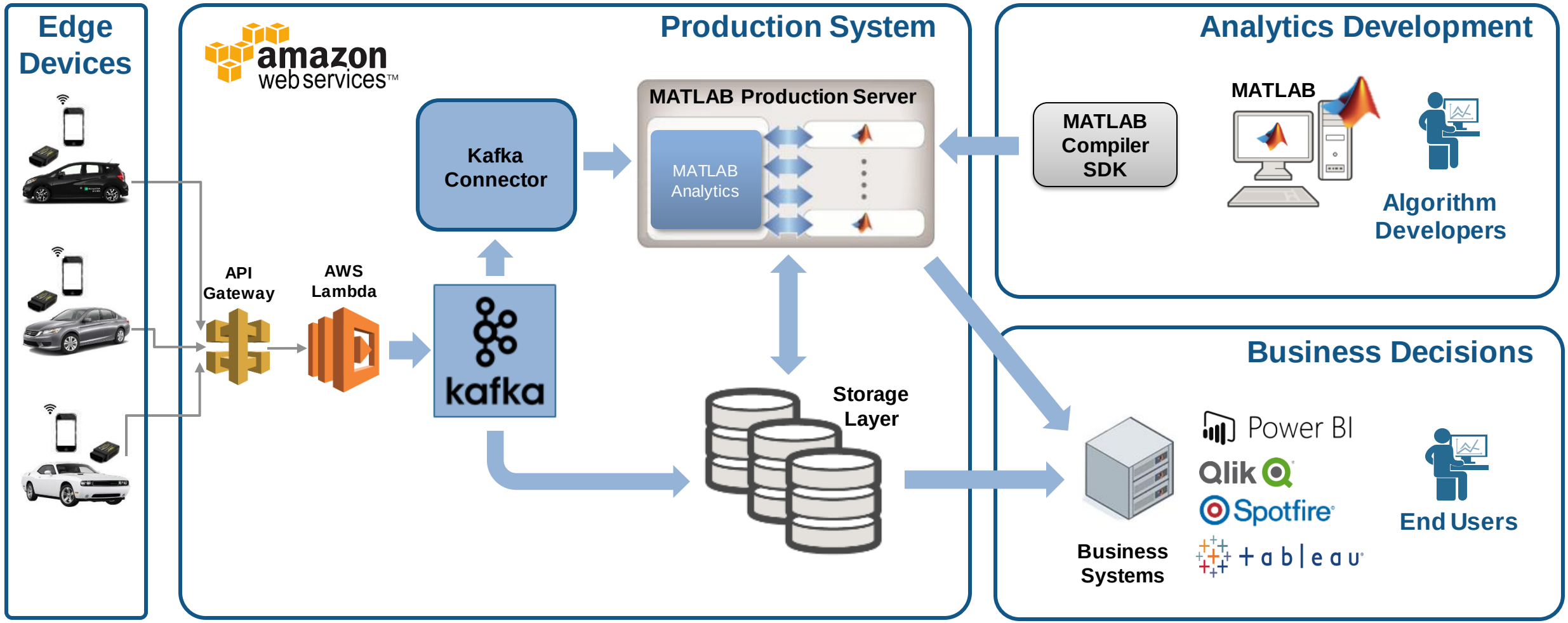
- A group of MathWorks employees installed an OBD dongle in their car that monitors the on-board systems
- Data is streamed to the cloud where it is aggregated and stored
- We would like to use this data to score the driving habits of participants

Example: Fleet Analytics with MATLAB



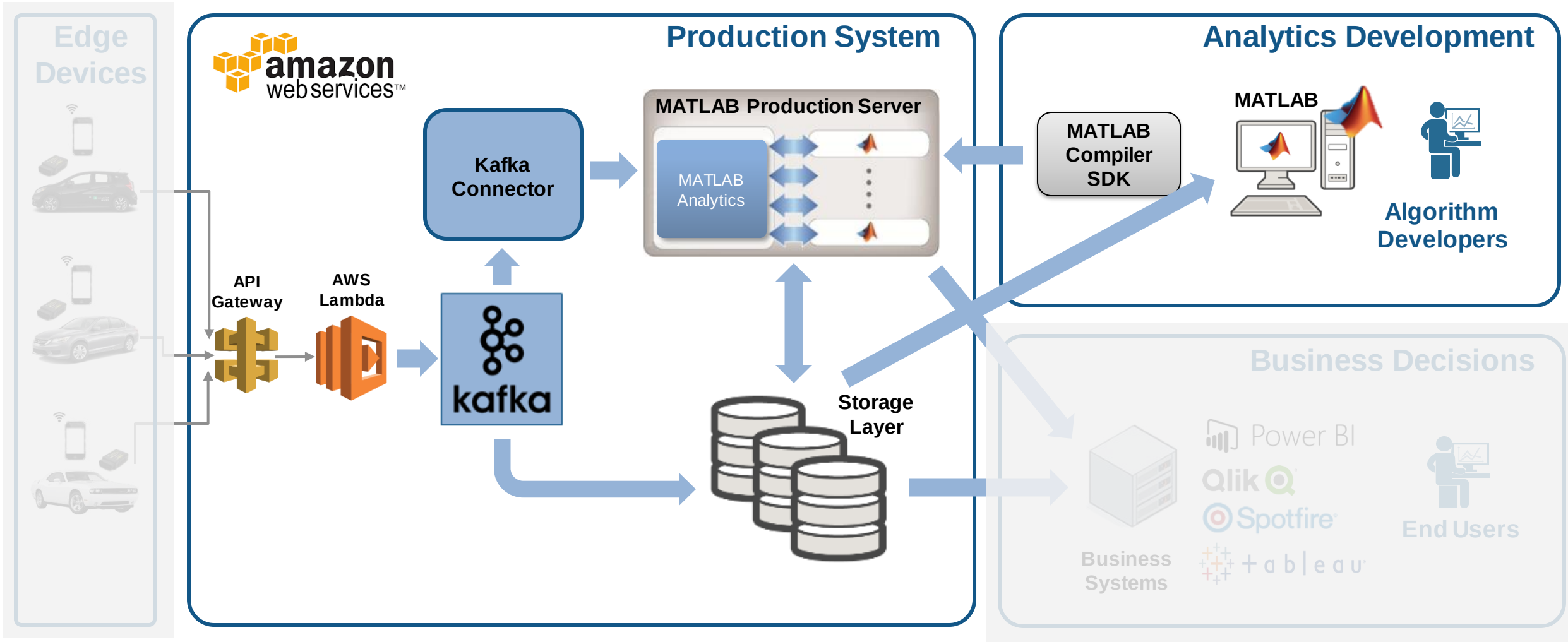


Fleet Analytics Architecture



1 Access and Explore Data

The first step is to clean up the incoming data



The Data: Timestamped messages with JSON encoding

```
{
    "vehicles_id": {"$oid":"55a3fd0069702d5b41000000"},
    "time" : {"$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,
      …
}
```

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'
2	15-Jan-2015 22:12:24	'{ "_id" : { "$oid" : "55a41cb069702d115b059ee1" }, "trip_id" : { "$oid"...	'55a41cb069702d115b059ede'
3	15-Jan-2015 22:12:25	'{ "_id" : { "$oid" : "55a41cb069702d115b059ee2" }, "trip_id" : { "$oid"...	'55a41cb069702d115b059ede'
4	15-Jan-2015 22:12:26	'{ "_id" : { "$oid" : "55a41cb069702d115b059ee3" }, "trip_id" : { "$oid"...	'55a41cb069702d115b059ede'

- ✓ **Decode JSON data**
- ✓ **Create Timetable**

Timetable

t = 4647×40 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

Develop a Preprocessing Function

Timetable

t = 4647×40 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	45.468	NaN	NaN	NaN	NaN	73.2006
7 Sun Jul 12 16:19:14 UTC 2015	55a3fe356...	55a3fe356...	56.7000	-84.9304	45.468					
8 Sun Jul 12 16:19:15 UTC 2015	55a3fe356...	55a3fe356...	57.6000	-84.9304	45.468					
9 Sun Jul 12 16:19:16 UTC 2015	55a3fe356...	55a3fe356...	56.7000	-84.9303	45.468					

- ✓ Clean up
- ✓ Enrich
- ✓ Restructure

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);
```

Ad Hoc Access to Data from MATLAB

athenaQuery.mlx

Access the data in S3

Bring up the AthenaClient

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

Create a query and submit

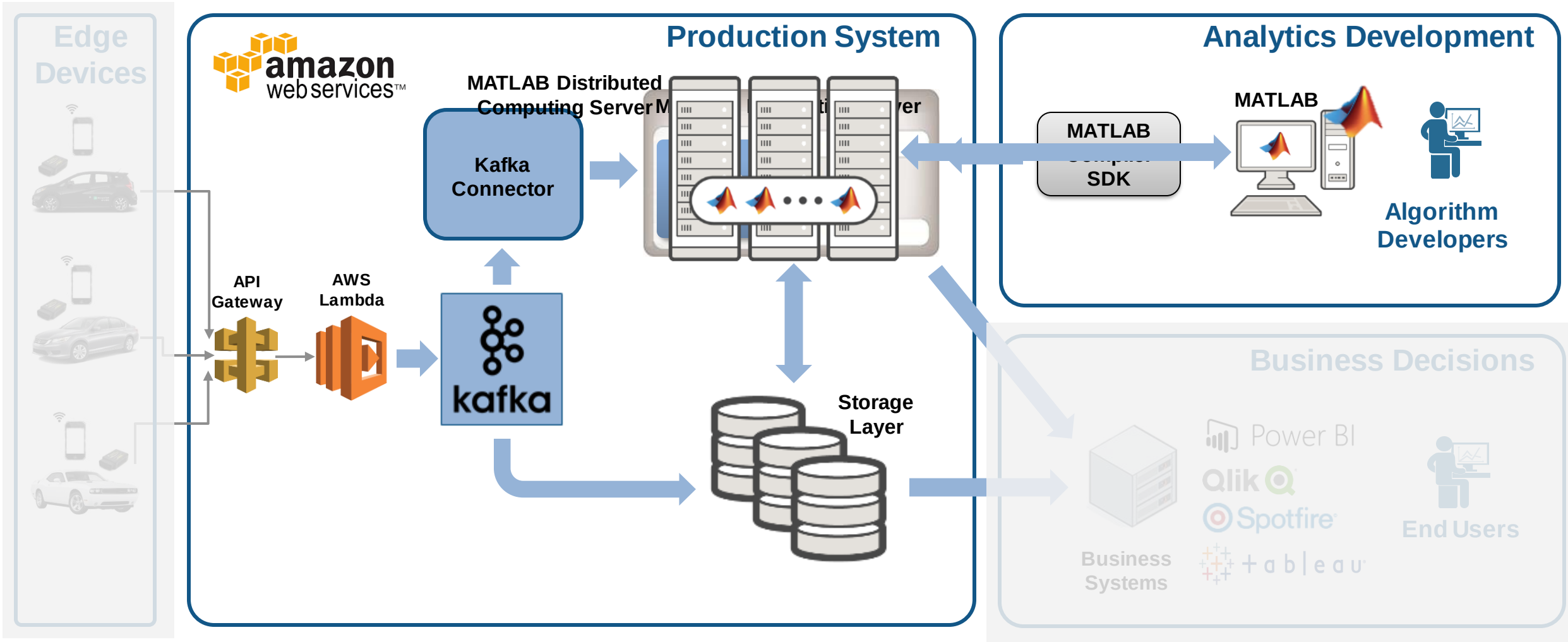
```
athenaClient.submitQuery('SELECT * FROM "trainingdata"."sampledata" limit 100','s3://fleettrainingdata')
```

Fetch data as a table for easy analysis

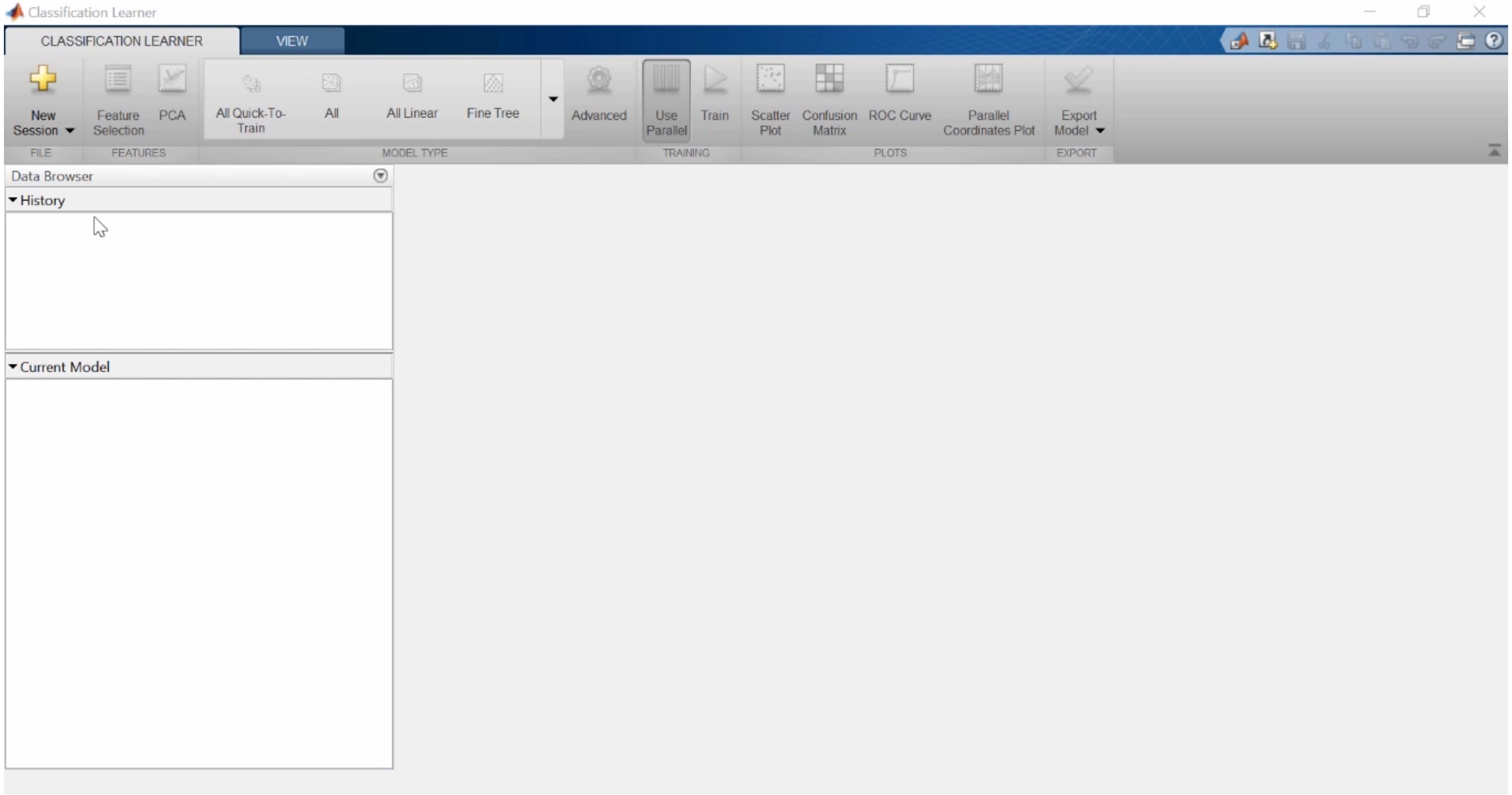
```
ds = datastore('s3://fleettrainingdata/*.csv');
ds.NumHeaderLines = 2;
data = table(ds);
```

Your usual MATLAB workflow goes here

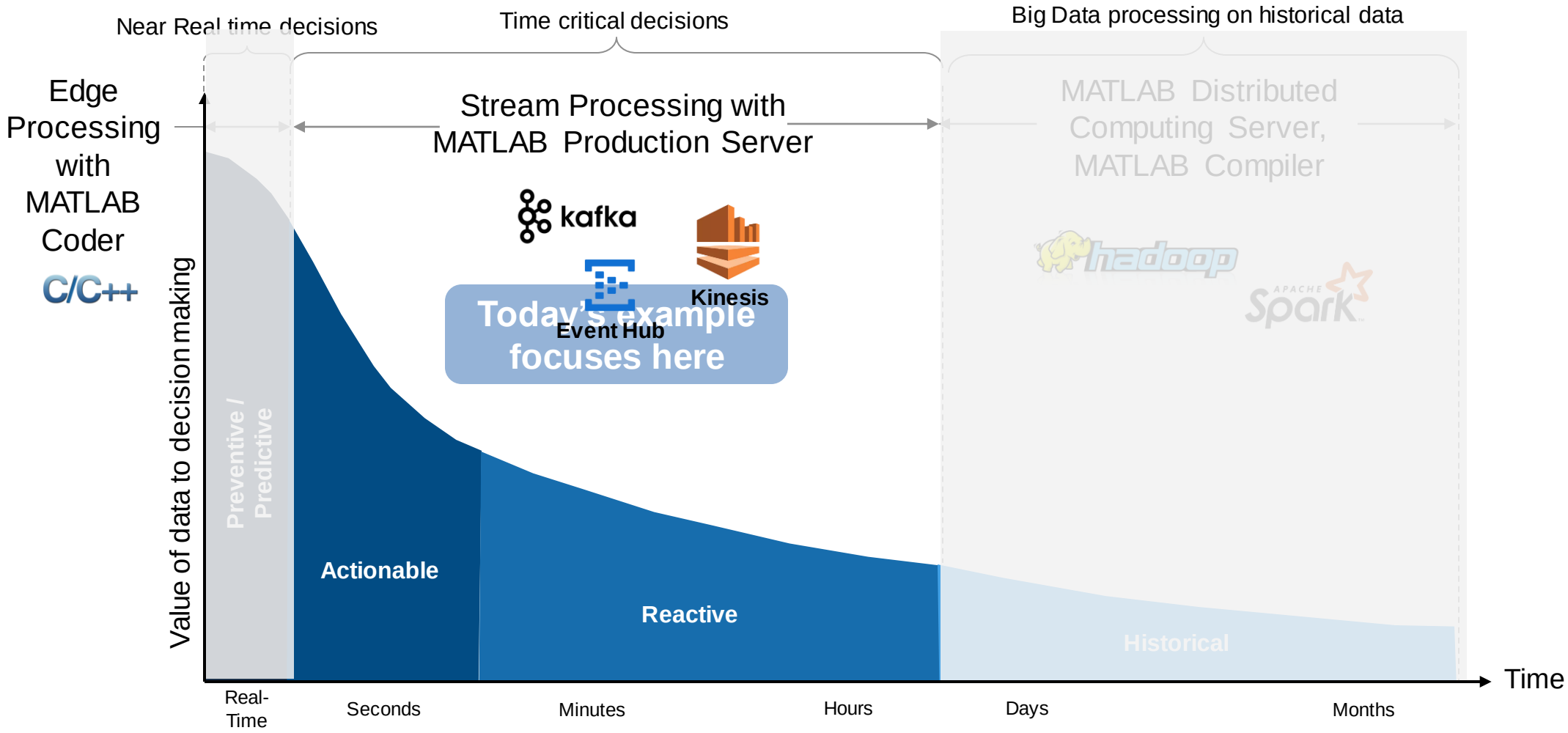
Develop a Predictive Model



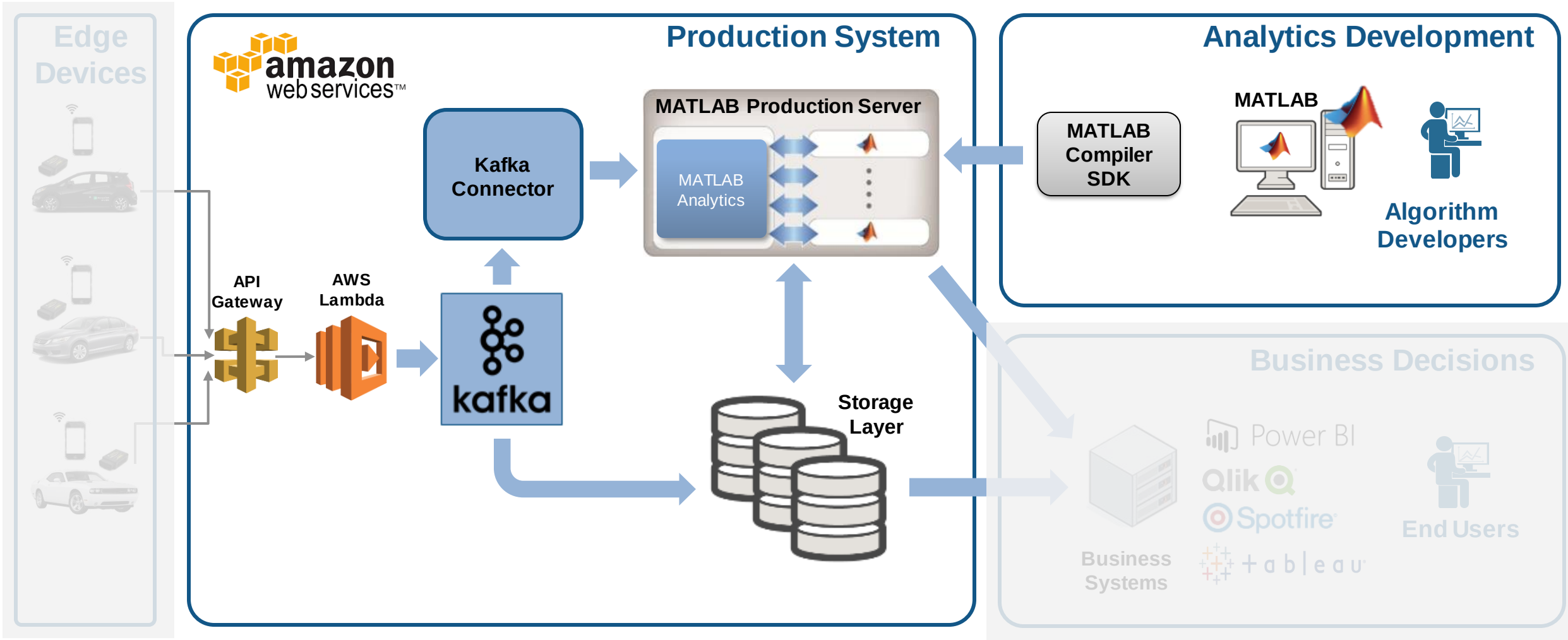
Develop a Predictive Model in MATLAB



Why stream processing?



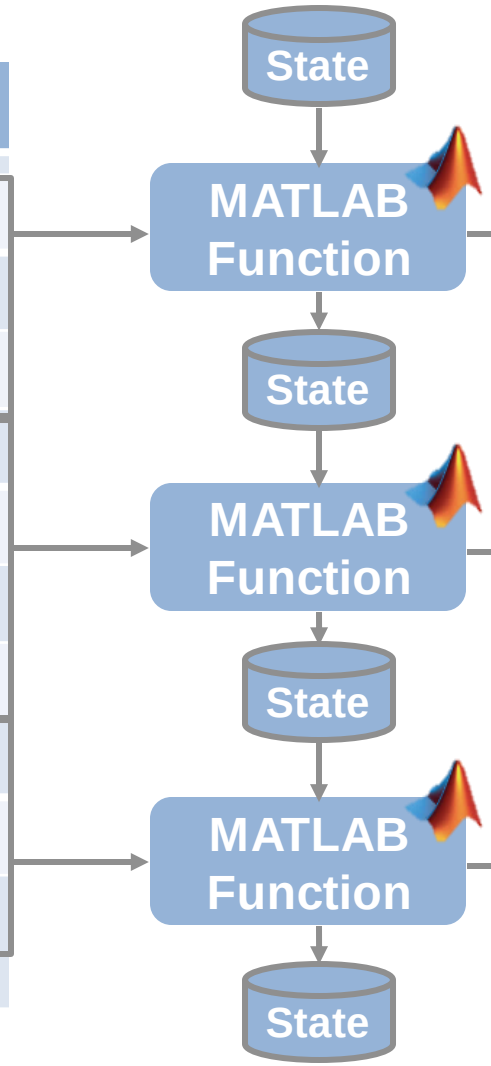
Integrate Analytics with Production Systems



Streaming data is treated as an unbounded Timetable

Input Table

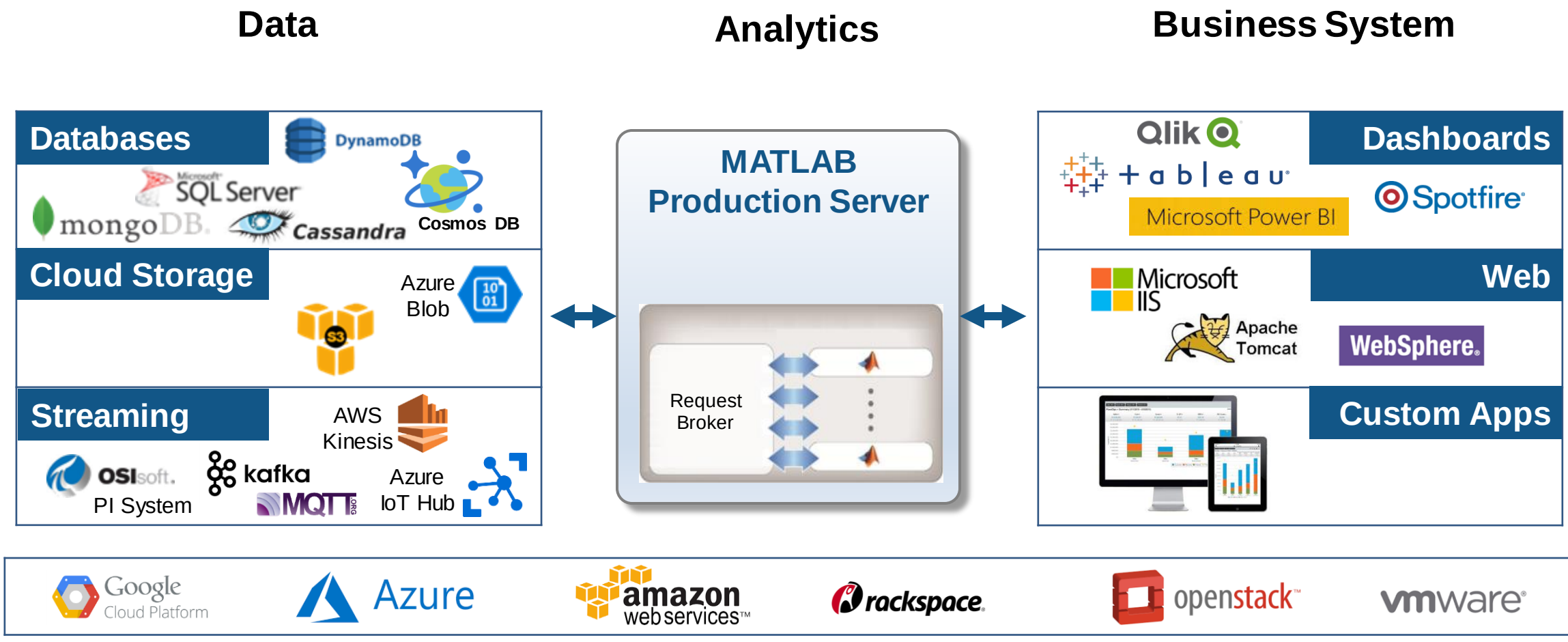
Event Time	Vehicle	RPM	Torque	Fuel Flow
18:01:10	55a3fd	1975	100	110
18:10:30	55a3fe	2000	109	115
18:05:20	55a3fd	1980	105	105
18:10:45	55a3fd	2100	110	100
18:30:10	55a419	2000	100	110
18:35:20	55a419	1960	103	105
18:20:40	55a3fe	1970	112	104
18:39:30	55a419	2100	105	110
18:30:00	55a3fe	1980	110	113
18:30:50	55a3fe	2000	100	110
…	…	…	…	…



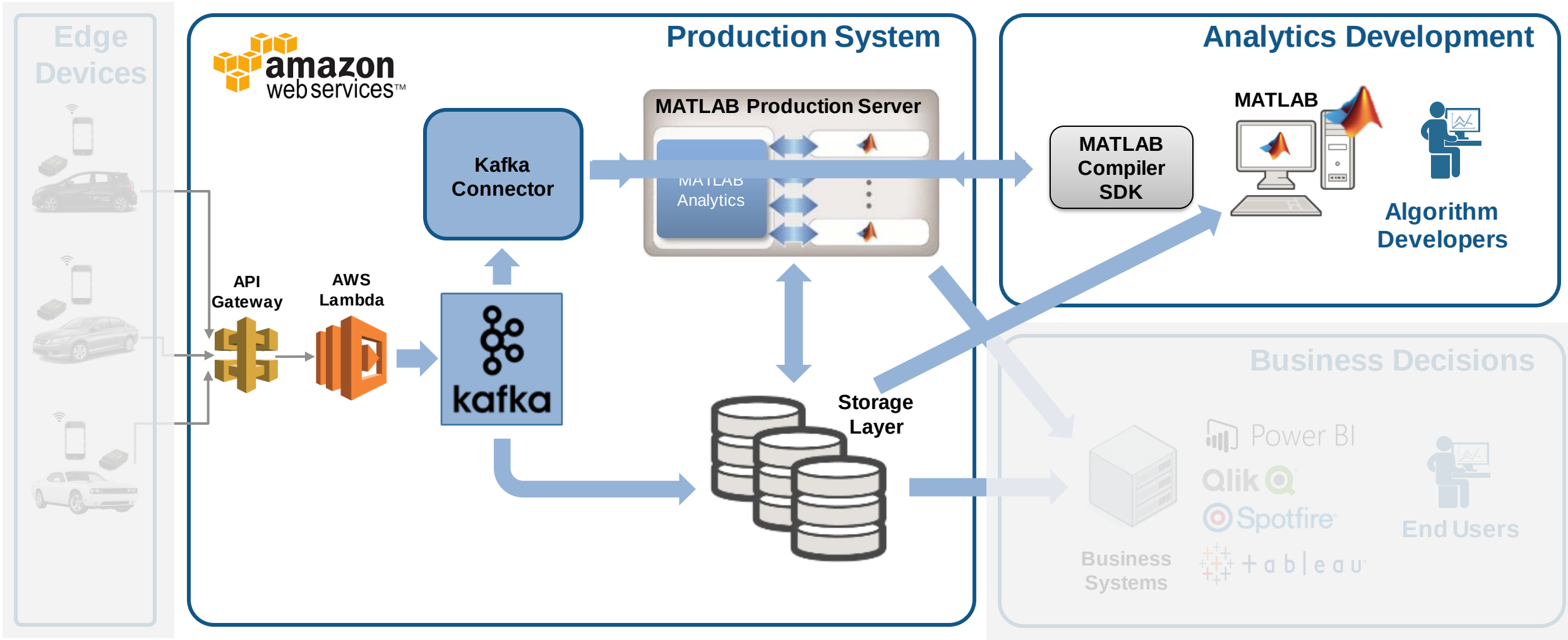
Output Table

Time window		Vehicle	Score
…		…	…
18:00:00	18:10:00	55a3fd	5
		55a3fe	…
		55a419	…
18:10:00	18:20:00	55a3fd	7
		55a3fe	3
		55a419	…
18:20:00	18:30:00	55a3fd	…
		55a3fe	4
		55a419	…
18:30:00	18:40:00	55a3fd	…
		55a3fe	5
		55a419	8

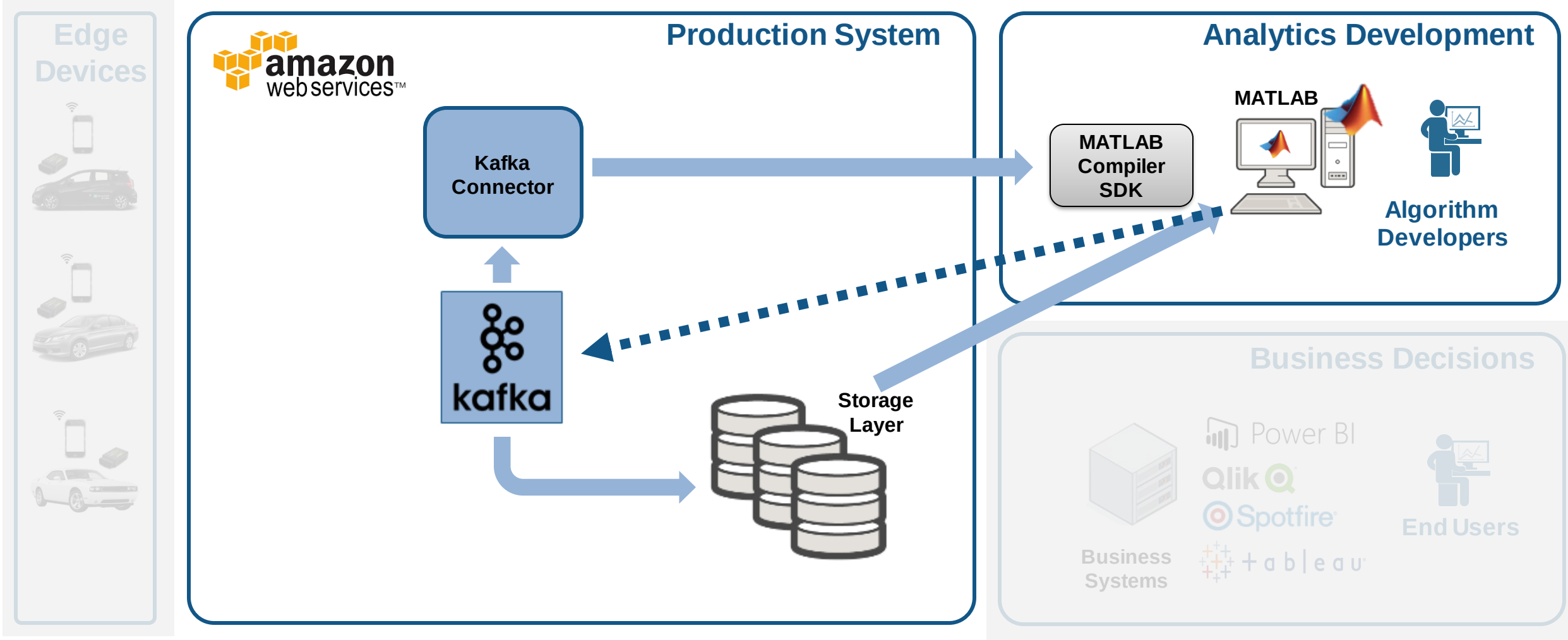
Introducing MATLAB Production Server



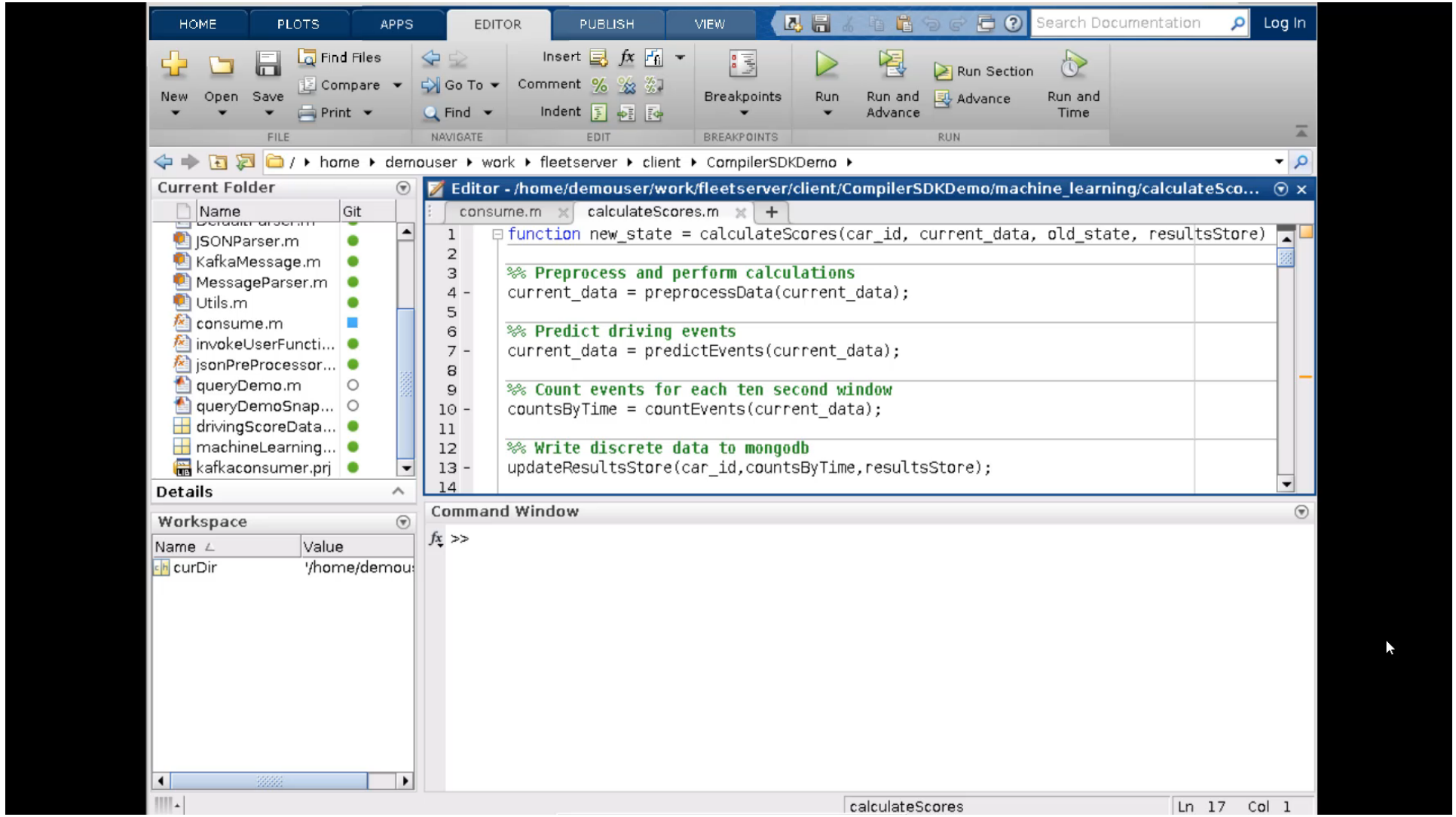
Develop and Deploy a Stream Processing Function



Debug a Stream Processing Function in MATLAB

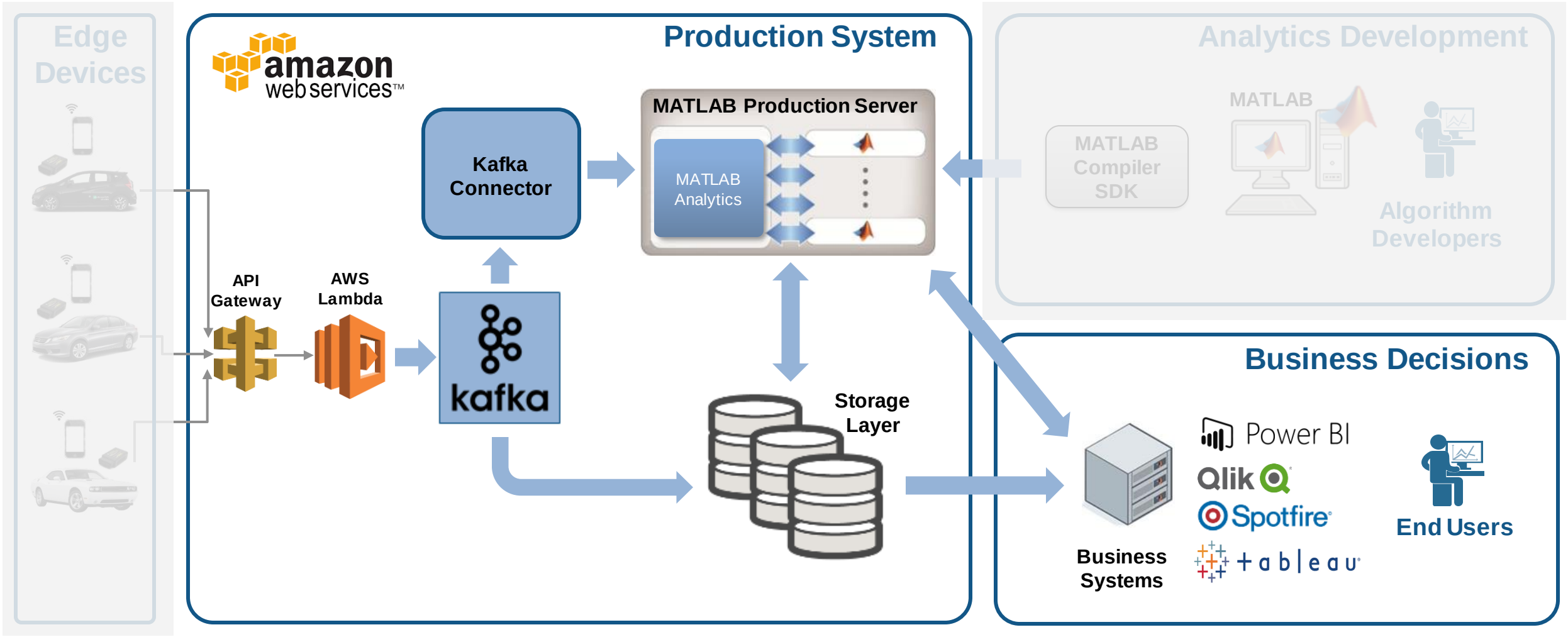


Debug a Stream Processing Function in MATLAB

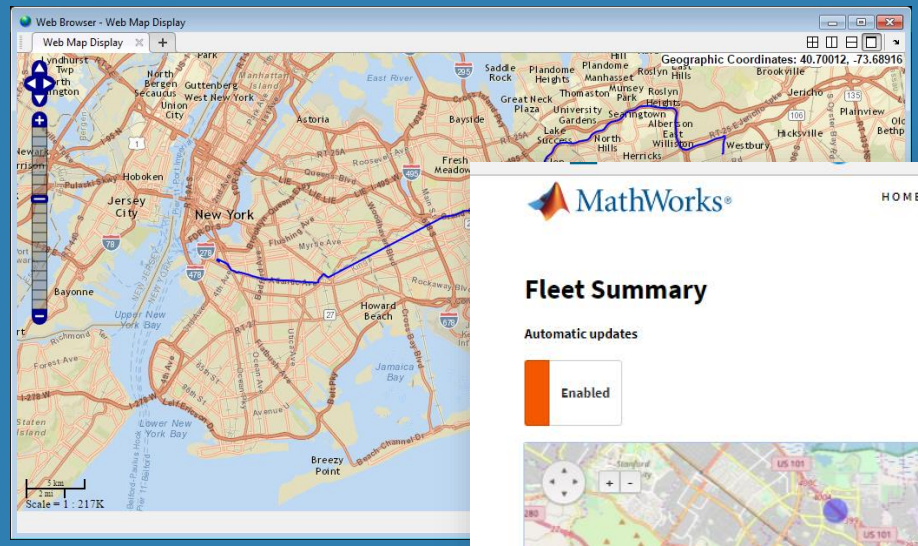


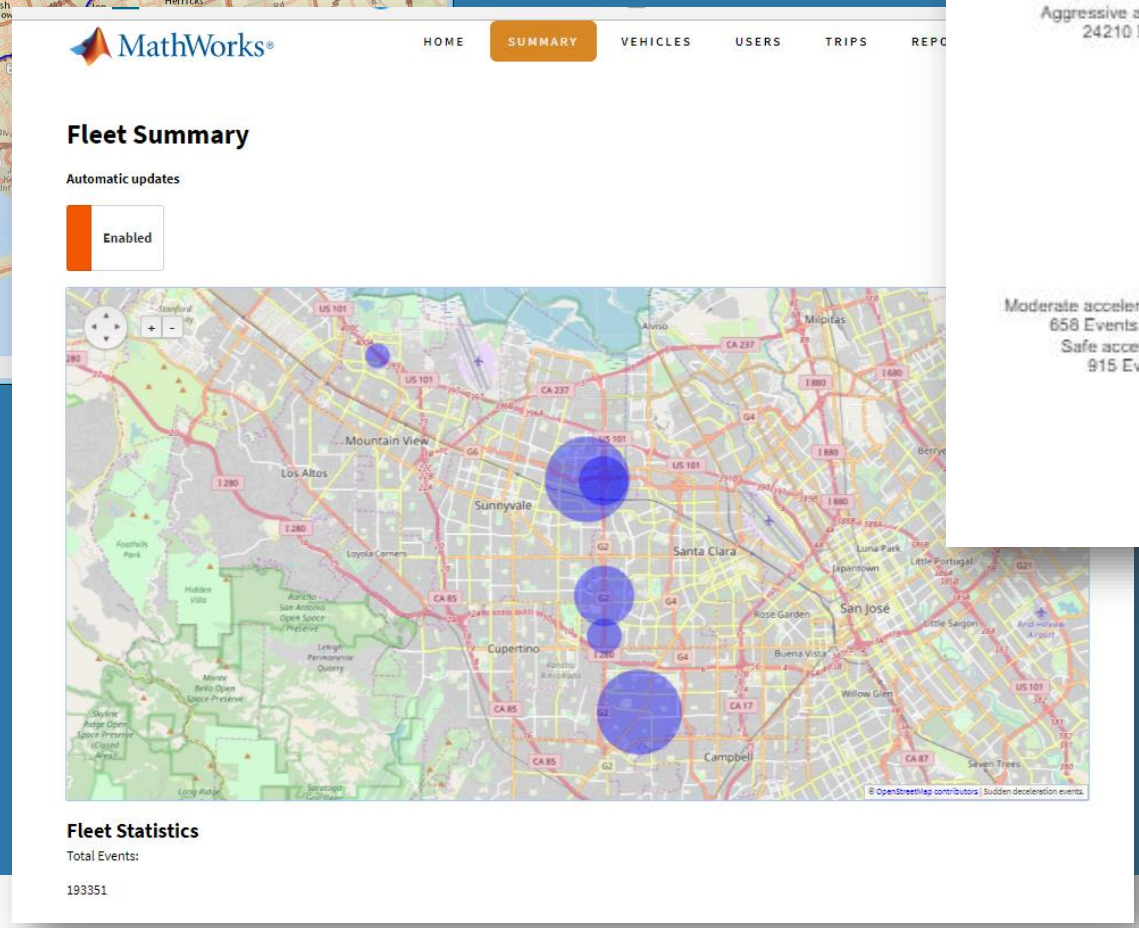
4 Integrate with Production Systems

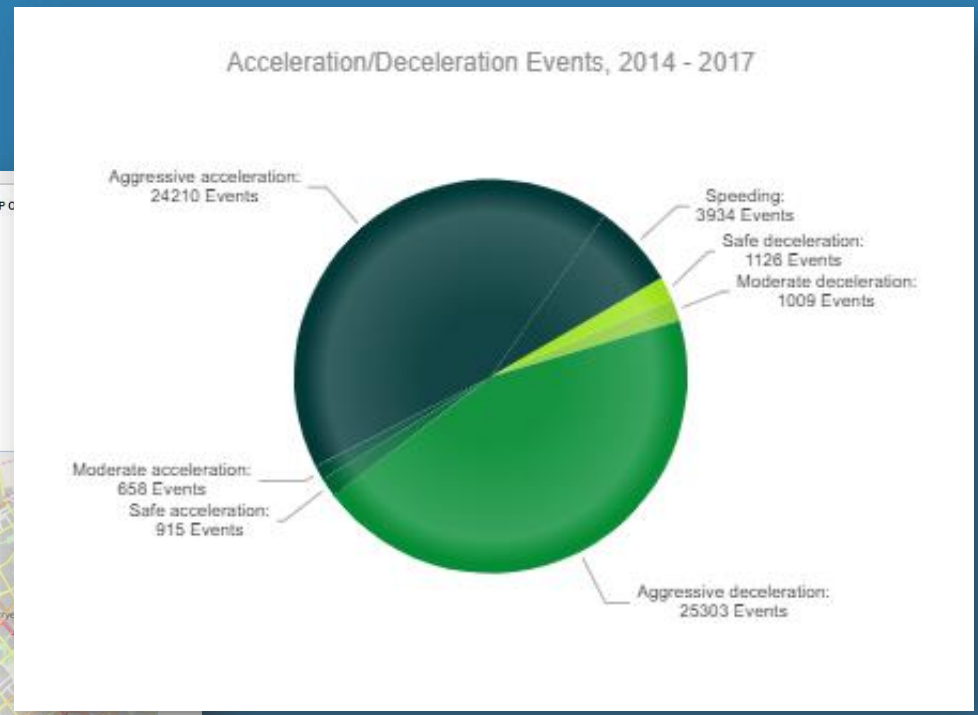
Tie in your Dashboard Application



Complete Your Application

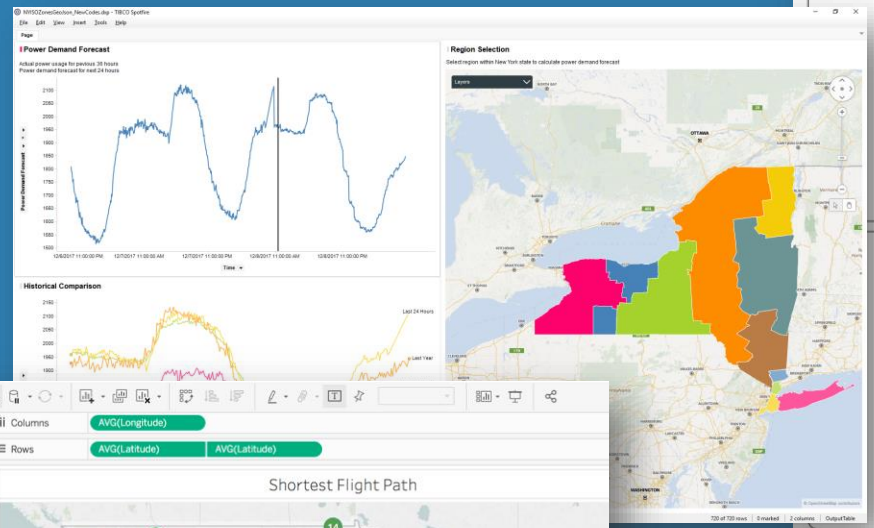




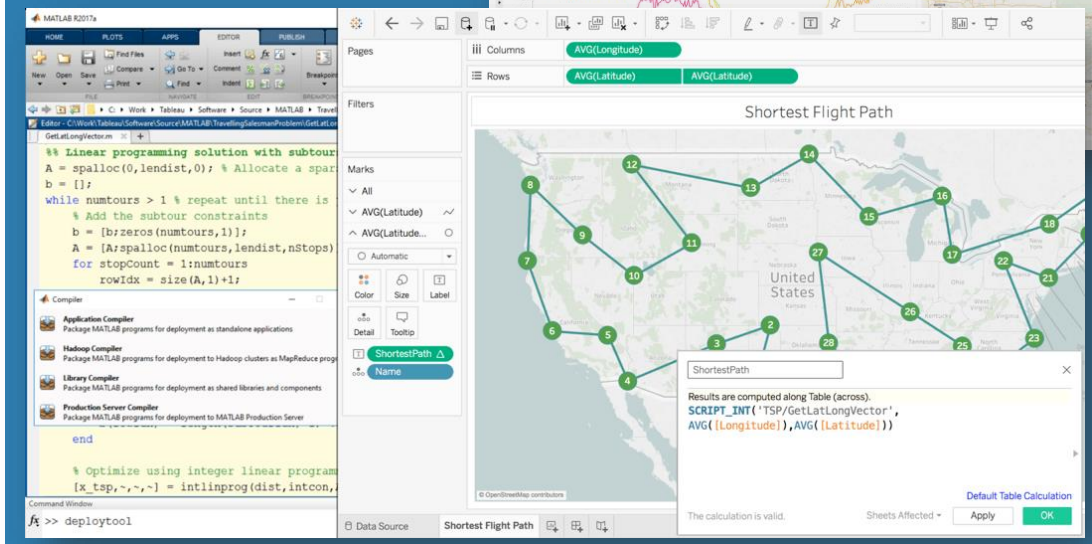


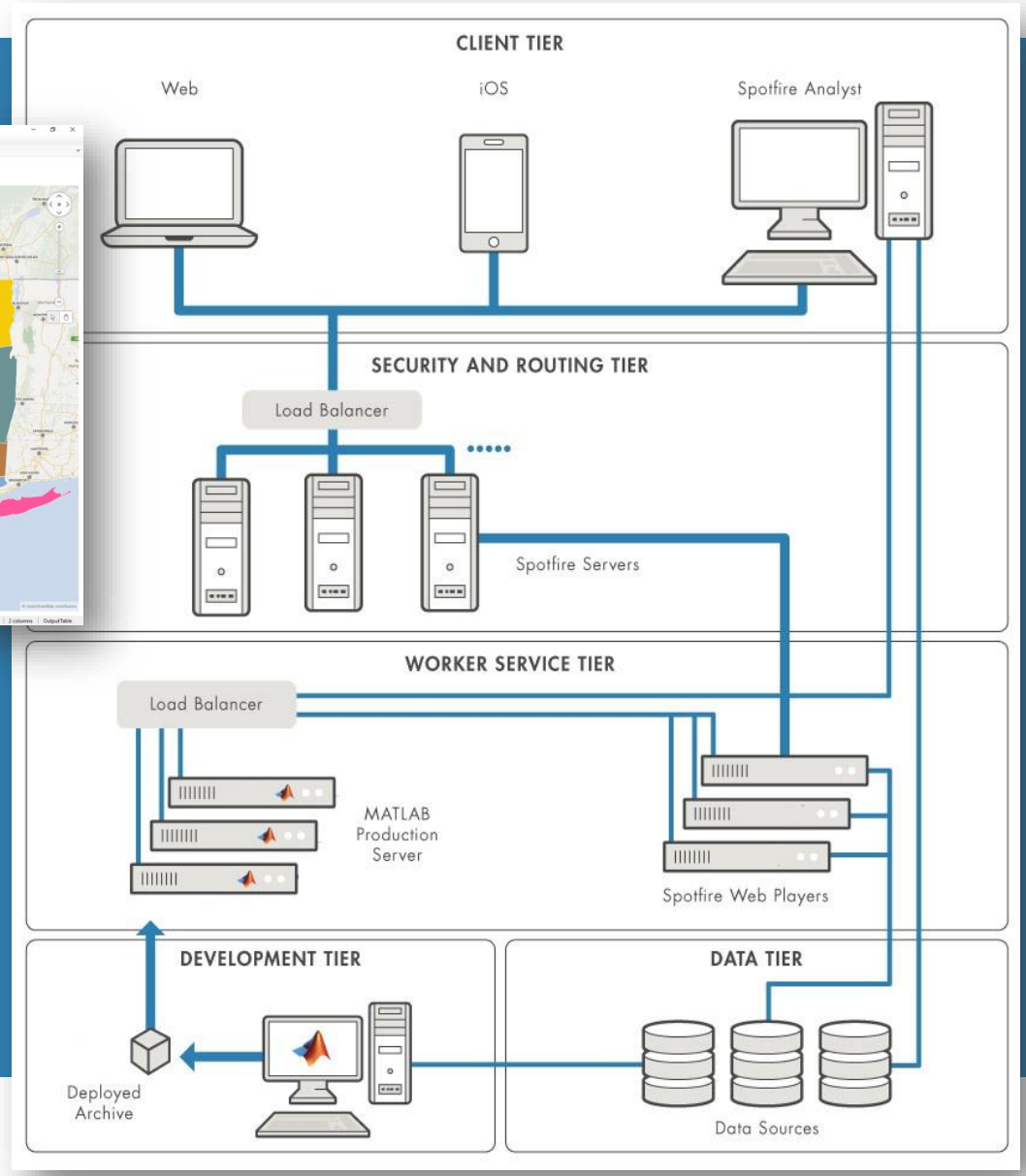
Scalable Analytics with Enterprise BI Tools

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