Microsoft

Azure Stream Analytics

Manjunath Suryanarayana

Unlocking Real-time Insights

Time to Insight is Critical

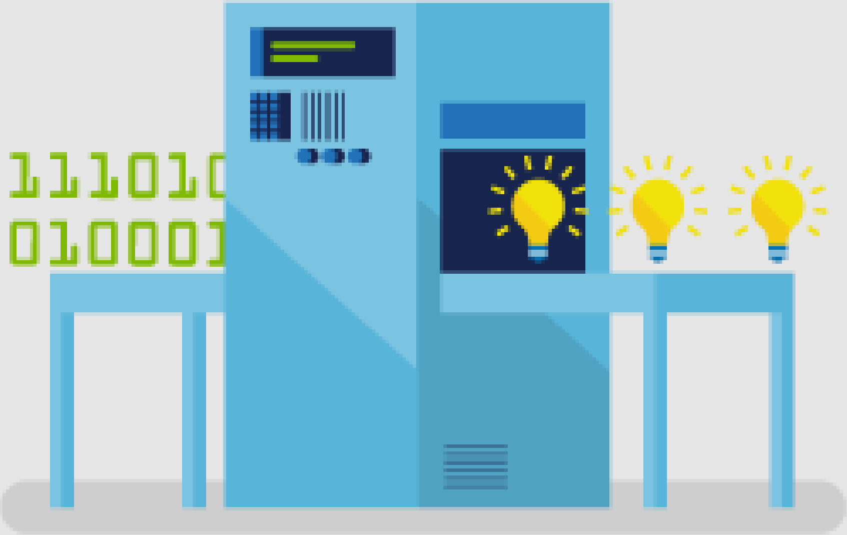
Reducing decision latency can unlock business value

Insights are Perishable

Window of opportunity for insights to be actionable

Ask Questions to Data in Motion

Can't wait for data to get to rest before running computation



Real-time Stream Processing

Simple Event Processing

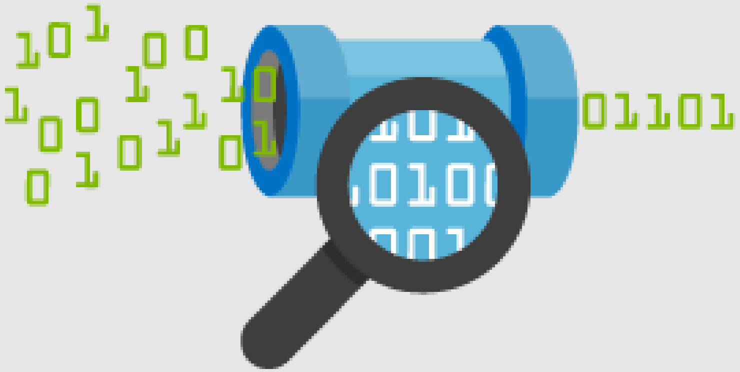
Filter
Transform
Enrich
Split
Route

Event Stream Processing

[Simple event processing] +
Aggregate
Rules

Complex Event Processing

[Event Stream Processing] +
Pattern detection
Time windows
Joins & correlations



Scenarios

Scenario Types

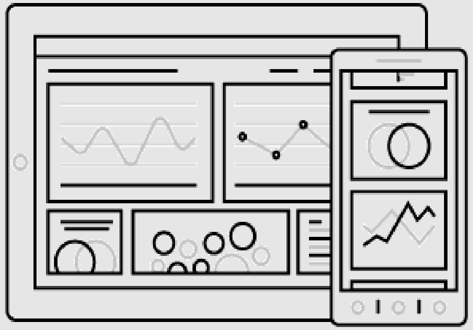
Actions by Human Actors

- "See and seize" insights
- Live visualization
- Alerts and alarms
- Dynamic aggregation

Machine to Machine Interactions

- Data movement with enrichment
- Kick-off workflows for automation

Dashboarding



Automation

Enriched Data Movement

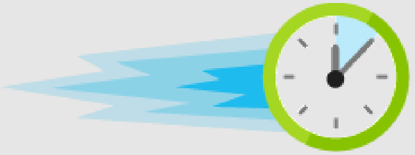
Scenario Examples

Real-time Fraud Detection

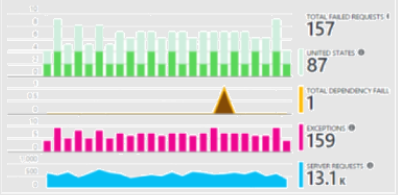
Streaming ETL

Predictive Maintenance

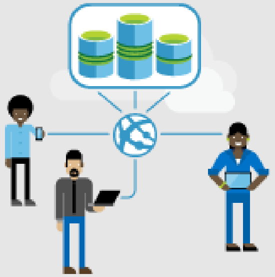
Call Center Analytics



IT Infrastructure and Network Monitoring



Customer Behavior Prediction



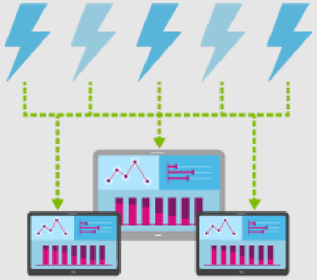
Log Analytics

Real-time Cross Sell Offers

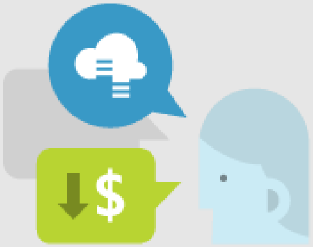
Fleet monitoring and Connected Cars

Real-time Patient Monitoring

Smart Grid



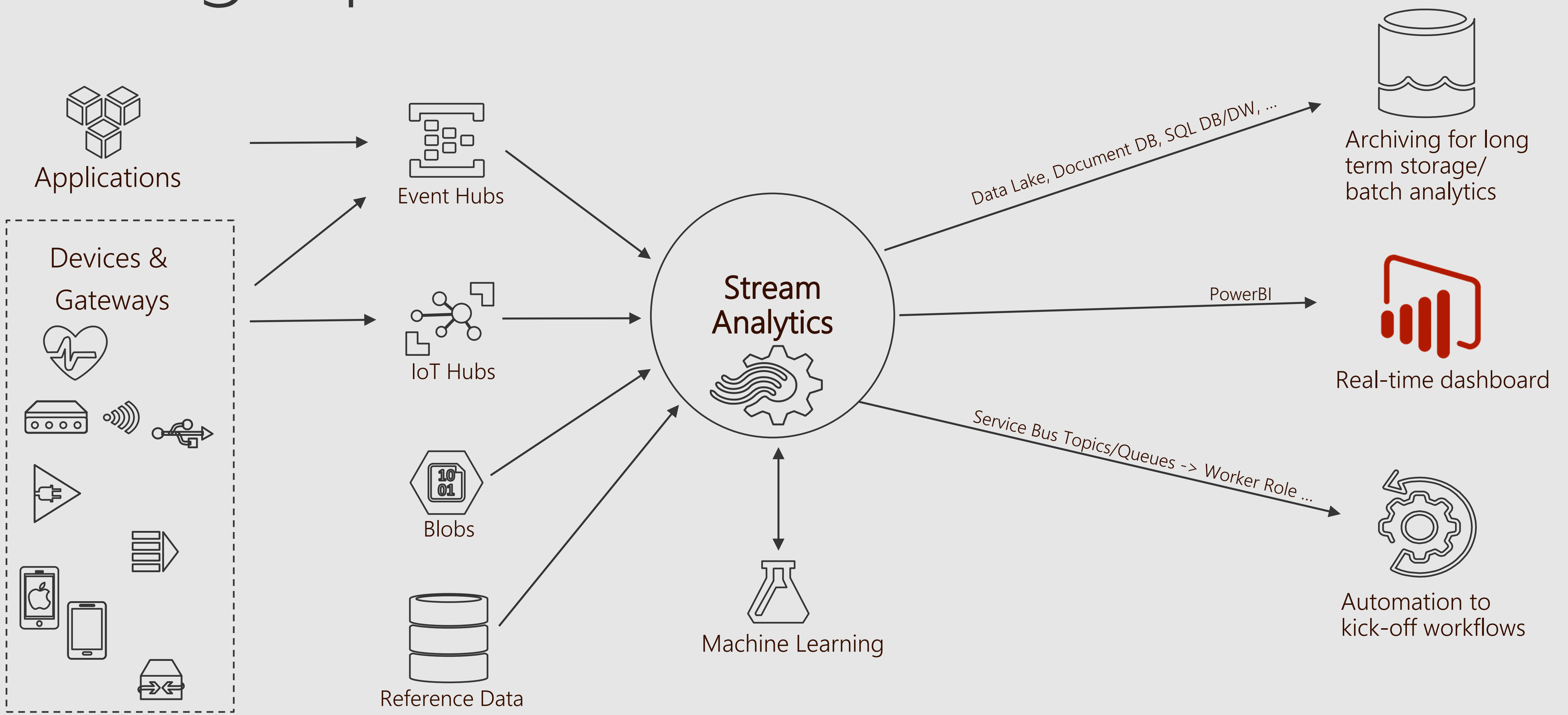
Real-time Marketing



and many more...

Streaming Analytics Pipeline

Streaming Pipeline



Applications

Devices & Gateways

Event Hubs

IoT Hubs

Blobs

Reference Data

Stream Analytics

Machine Learning

Data Lake, Document DB, SQL DB/DW, ...

Archiving for long term storage/ batch analytics

PowerBI

Real-time dashboard

Service Bus Topics/Queues -> Worker Role ...

Automation to kick-off workflows

Event production ⇨ Event Queuing & Stream Ingestion ⇨ Stream Analytics ⇔ Storage & Batch Analysis ⇨ Presentation & Action

Differentiated Value Proposition

Azure Stream Analytics

Programmer Productivity

Declarative SQL like query language

Ease o
st

Source/sink integrations

Job Service
No cluster provisioning

Pay as you go

Mission critical reliability

Enterprise grade SLA

Differentiators

Programmer Productivity

Declarative SQL like language

Built-in temporal semantics

Ease of Getting Started

Integrations with sources, sinks, & ML

Build real-time dashboards in minutes

Lowest Total Cost of Ownership(TCO)

Fully managed service

No cluster topology management required

Seamless scalability

Usage based pricing

1,915 lines of code with Apache Storm

```
@ApplicationAnnotation(name="WordCountDemo")
public class Application implements StreamingApplication
{
      protected String fileName =
      "com/datatorrent/demos/wordcount/samplefile.txt";
                  private Locality locality = null;

      @Override  public void populateDAG(DAG dag, Configuration
      conf)
      {
            locality = Locality.CONTAINER_LOCAL;
            WordCountInputOperator input =
            dag.addOperator("wordinput", new
            WordCountInputOperator());
            input.setFileName(fileName);
            UniqueCounter<String> wordCount =
            dag.addOperator("count", new
            UniqueCounter<String>());
            dag.addStream("wordinput-count", input.outputPort,
            wordCount.data).setLocality(locality);
            ConsoleOutputOperator consoleOperator =
            dag.addOperator("console", new
            ConsoleOutputOperator());
            dag.addStream("count-console",wordCount.count,
            consoleOperator.input);
       }
}
```

3 lines of SQL in Azure Stream Analytics

```
SELECT Avg(Purchase), ScoreTollId, Count(*)
FROM GameDataStream
GROUP BY TumblingWindows(5, Minute), Score
```

Stream Analytics Query Language (SAQL)

Declarative SQL like language to describe transformations

- Filters (“Where”)
- Projections (“Select”)
- Time-window and property-based aggregates (“Group By”)
- Time-shifted joins (specifying time bounds within which the joining events must occur)
- and all combinations thereof

Data Manipulation

```
SELECT
FROM
WHERE
HAVING
GROUP BY
CASE WHEN THEN ELSE
INNER/LEFT OUTER JOIN
UNION
CROSS/OUTER APPLY
CAST INTO
ORDER BY ASC, DSC
```

Aggregation

```
SUM
COUNT
AVG
MIN
MAX
STDEV
STDEVP
VAR
VARP
TopOne
```

Date and Time

```
DateName
DatePart Day, Month, Year
DateDiff
DateTimeFromParts
DateAdd
```

Temporal

```
Lag
IsFirst
Last
CollectTop
```

Windowing Extensions

```
TumblingWindow
HoppingWindow
SlidingWindow
```

Scaling Extensions

```
WITH
PARTITION BY
OVER
```

String

```
Len
Concat
CharIndex
Substring
Lower, Upper
PatIndex
```

Mathematical

```
ABS
CEILING
EXP
FLOOR
POWER
SIGN
SQUARE
SQRT
```

Geospatial (preview)

```
CreatePoint
CreatePolygon
CreateLineString
ST_DISTANCE
ST_WITHIN
ST_OVERLAPS
ST_INTERSECTS
```

Mission Critical Reliability

Enterprise Grade SLA

At least three 9s of availability

Business Continuity During Failures

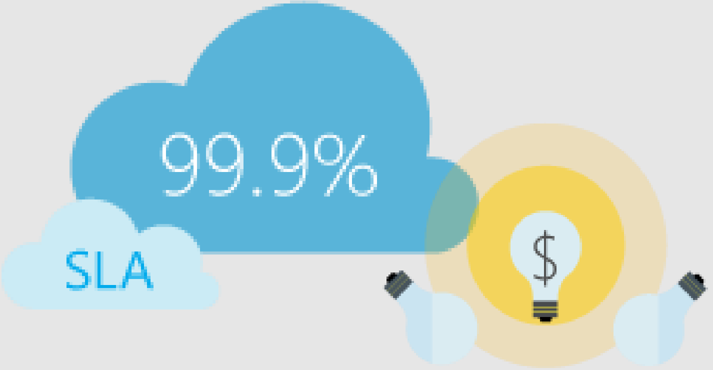
Automatic checkpoint-recovery

Fast restarts

Guaranteed Event Delivery

At-least-once event delivery semantics

No data loss



Global Availability Footprint

Currently available in 19 Azure regions including China and Germany

Current list includes:

Central US, East US, East US2, North Central US, South, Central US, West US, North Europe, West Europe, East Asia, Southeast Asia, Japan West, Japan East, Brazil South, Australia East, Central India.

In China: Stream Analytics is made available through a unique partnership between Microsoft and 21Vianet.

In Germany: Stream Analytics is available via a new data trustee model whereby customer data remains in Germany under control of T-Systems, a Deutsche Telekom company, acting as the German data trustee.

Satisfies Major Global Compliance Requirements

Current list includes:

ISO 27001
ISO 27018
SOC 1 Type 2
SOC 2 Type 2
SOC 3 Type 2
HIIPAA/HITECH
PCI DSS Level 1
European Union Model Clauses
China GB 18030

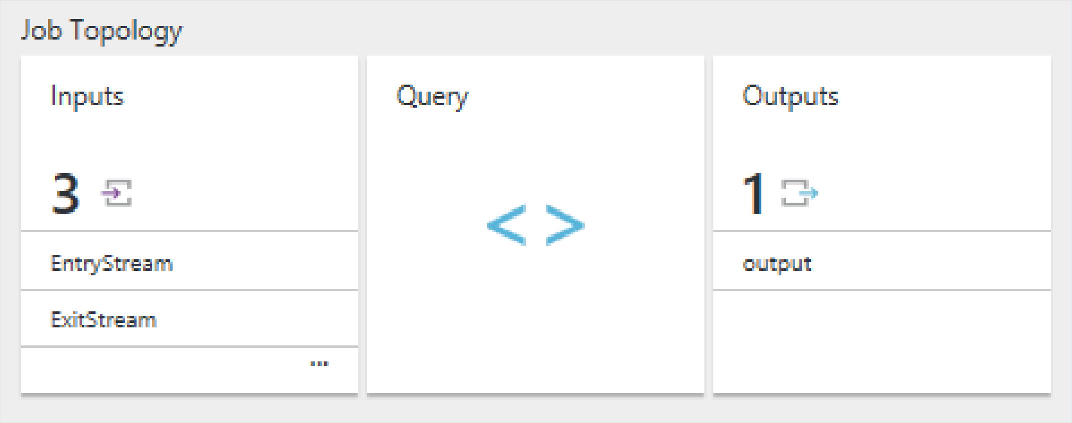
Capabilities

Stream Analytics Job

Users construct and deploy jobs to ASA

Job definition includes inputs, a query, and output

- **Inputs** are from where the job reads the data stream
- **Query** runs for perpetuity unless explicitly stopped and transforms the input stream
- **Output** is where the job sends the job results to



Job Topology

Inputs

3

EntryStream

ExitStream

...

Query

<>

Outputs

1

output

Scenario – Tolling Station

Tolling stations have multiple booths (identified by TollId)

Each booth has two sensors: Entry and Exit that send out EntryStream and ExitStream respectively

EntryStream – Data stream from the Entry sensor data on vehicles entering toll booths

TollId	EntryTime	License Plate	State	Make	Model	Type	Weight
1	2014-10-25T19:33:30.0000000Z	JNB 7001	NY	Honda	CRV	1	3010
1	2014-10-25T19:33:31.0000000Z	YXZ 1001	NY	Toyota	Camry	2	3020
3	2014-10-25T19:33:32.0000000Z	ABC 1004	CT	Ford	Taurus	2	3800
2	2014-10-25T19:33:33.0000000Z	XYZ 1003	CT	Toyota	Corolla	2	2900
…	…	…	…	…	…	…	…

ExitStream - Data stream from the Exit sensor on vehicles exiting toll booths

TollId	ExitTime	LicensePlate
1	2014-10-25T19:33:40.0000000Z	JNB 7001
1	2014-10-25T19:33:41.0000000Z	YXZ 1001
3	2014-10-25T19:33:42.0000000Z	ABC 1004
2	2014-10-25T19:33:43.0000000Z	XYZ 1003
…	…	…

ReferenceData - Commercial vehicle registration data

LicensePlate	RegistartionId	Expired
SVT 6023	285429838	1
XLZ 3463	362715656	0
QMZ 1273	876133137	1
RIV 8632	992711956	0
…	…	….

Events and Time

Every event that flows through the system has a timestamp

ASA supports:

- Arrival Time - Event timestamps based on arrival time (input adapter clock, e.g., Event Hubs)
- App Time - Event timestamps based on a timestamp field in the actual event tuple

User can pick up App Time from the payload

```
SELECT * FROM EntryStream TIMESTAMP BY EntryTime
```

System can assign timestamps automatically based on the event arrival time

```
SELECT * FROM EntryStream
```

Filters and Projections

From the incoming stream find only vehicles that:

- Are from either WA and CA state
- Have a weight less than 3000 lbs
- Have License plate number end in 999
- Have a make that starts with a “M”

Display:

“Passenger” if type = 1

“Commercial” if Type = 2

“Other” for all other types

Display time as ‘Mins’, ‘Seconds’, ‘Milliseconds’

```
SELECT VehicleCategory =
        Case Type
                WHEN 1 THEN ‘Passenger’
                WHEN 2 THEN ‘Commercial’
                ELSE THEN ‘Other’
        END,
TollId, State LicensePlate, State, Make,
Model, Weight,
DATEPART(mi,EntryTime) AS ‘Mins’,
DATEPART(ss,EntryTime) AS ‘Seconds’
DATEPART(ms,EntryTime) AS ‘Milleseconds’
FROM EntryStream TIMESTAMP BY EntryTime
WHERE (State = ‘CA’ OR State = ‘WA’)
AND Weight < 3000
AND CHARINDEX (‘M’,  model) = 0
AND PATINDEX(‘%999’, LicensePlate) = 5
```

Temporal Joins

Report the time in seconds required for vehicles to pass the toll booth

```
SELECT ES.TollId, ES.EntryTime, EX.ExitTime, ES.EntryTime, ES.LicensePlate
DATEDIFF(minute, ES.EntryTime, EX.ExitTime )
FROM EntryStream ES TIMESTAMP BY EntryTime

JOIN   ExitStream EX TIMESTAMP BY ExitTime
ON     (EX.TollId= ES.TollId and ES.LicensePlate = EX.LicensePlate)
AND   DATEDIFF(minute, ES, EX) BETWEEN 0 AND 15
```

Temporal Left Outer Join to Detect Patterns

Reports all cars that have entered the toll booth but have not exited within 5 minutes

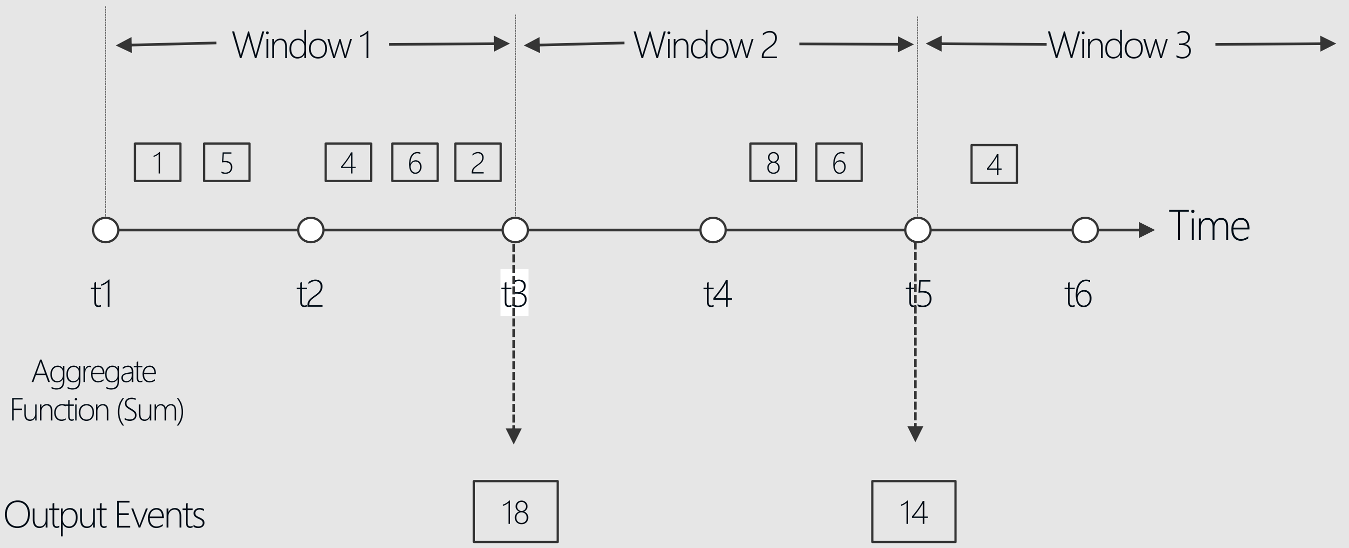
```
SELECT ES.TollId, ES.EntryTime, ES.LicensePlate
FROM EntryStream EN TIMESTAMP BY EntryTime

LEFT OUTER JOIN ExitStream EX TIMESTAMP BY ExitTime
ON (EN.TollId= EX.TollId AND EN.LicensePlate = EX.LicensePlate)
AND DATEDIFF(minute, EN, EX) BETWEEN 0 AND 5
WHERE EX.ExitTime IS NULL
```

Windowing Concepts

Output at the end of each window

Windows are fixed length

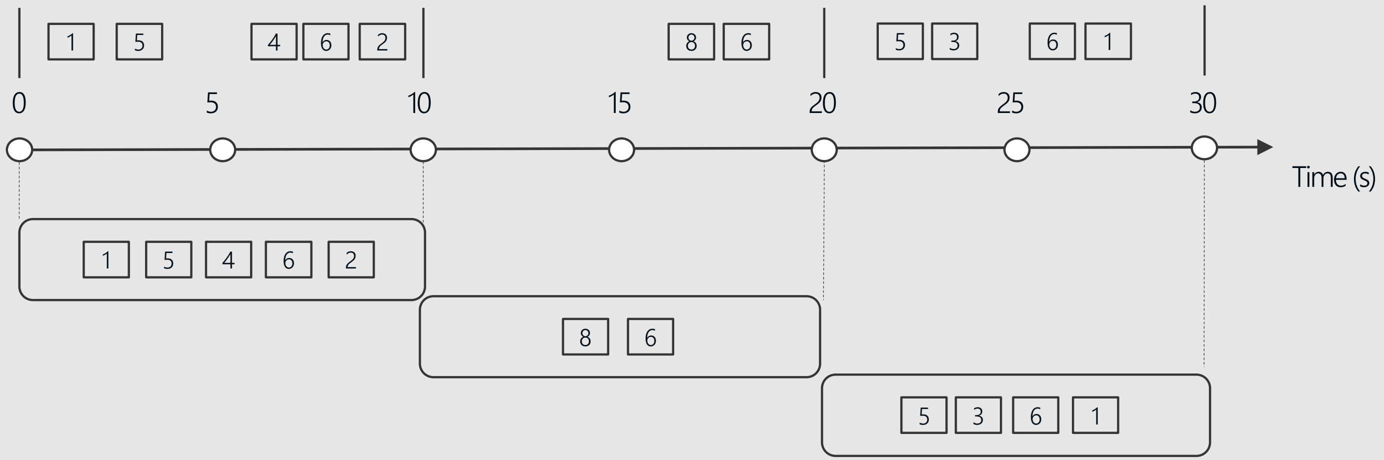
Used in a GROUP BY clause



Tumbling Windows

Every 10 seconds give me the count of vehicles entering each toll booth over the last 10 seconds

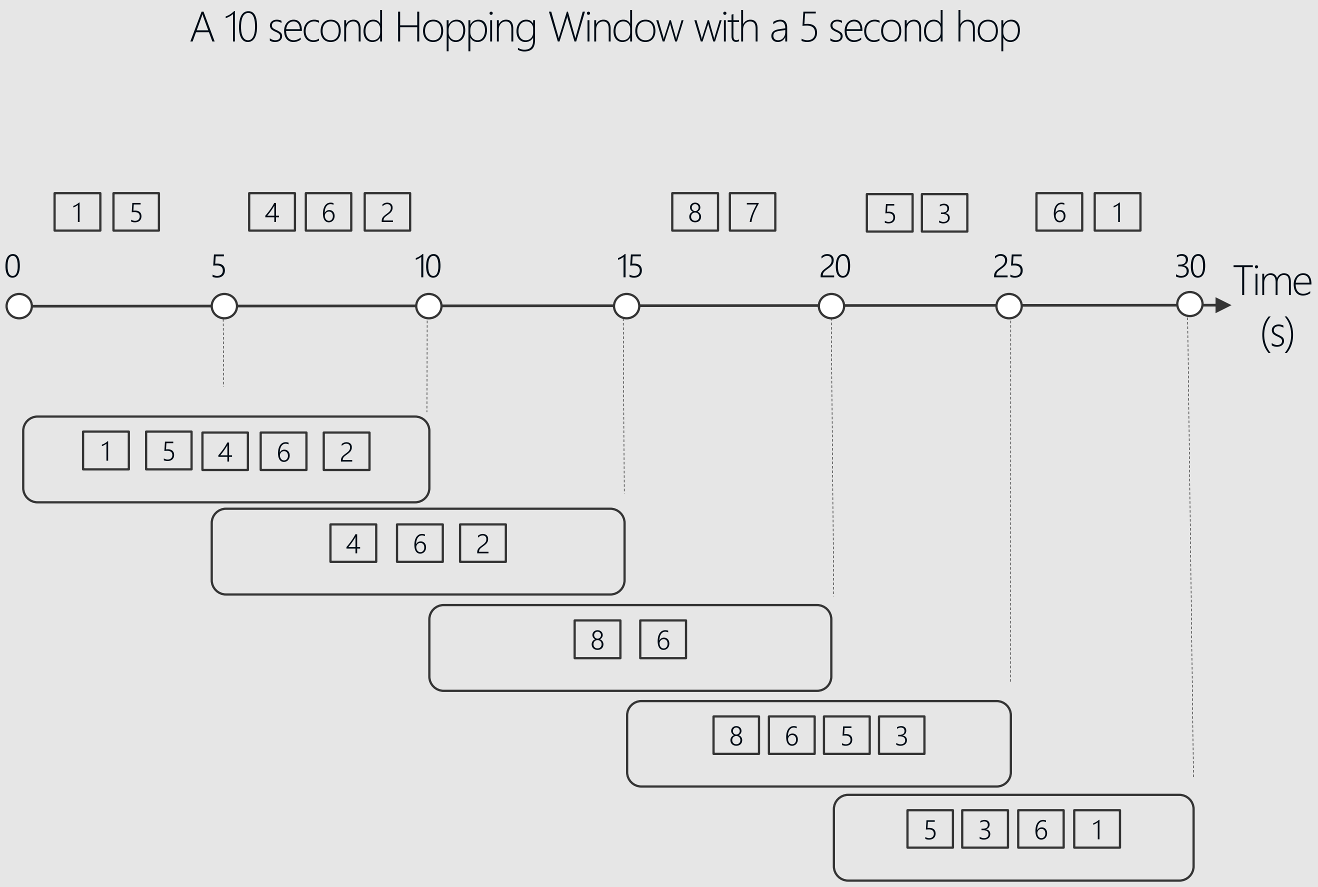
A 10-second Tumbling Window



```
SELECT TollId, Count(*)
FROM EntryStream TIMESTAMP BY EntryTime
GROUP BY TollId, TumblingWindow(second, 10)
```

Hopping Windows

Every 5 seconds give me the count of vehicles entering each toll booth over the last 10 seconds

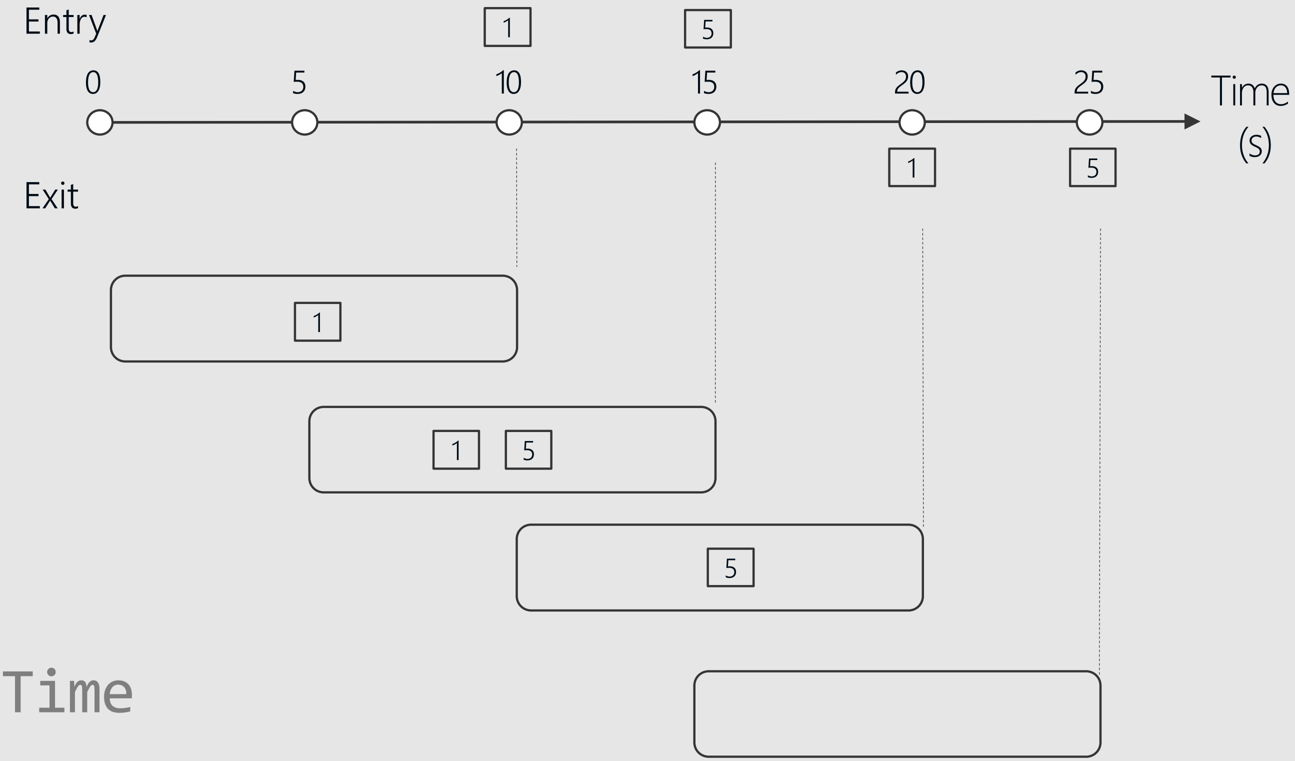


```
SELECT TollId, Count(*)
FROM EntryStream TIMESTAMP BY EntryTime
GROUP BY TollId, HoppingWindow(second, 10, 5)
```

Sliding Windows

Find all toll booths that have served more than 10 vehicles in the last 20 seconds

A 10-second Sliding Window



```
SELECT TollId, Count(*)
FROM EntryStream TIMESTAMP BY EntryTime
GROUP BY TollId, SlidingWindow(second, 20)
HAVING Count(*) > 10
```

An output is generated whenever an event either enters/leaves the system

Advanced Query Examples

Determine if a Value has Changed

```
SELECT
    Make,
    Time
FROM
    Input TIMESTAMP BY Time
WHERE
    LAG(Make, 1) OVER (LIMIT DURATION(minute, 1)) <>
Make
```

LAG is used to peek into the input stream one event back and get the Make value. Then compare it to the Make on the current event and output the event if they are different

Find First Event in a Window

Find first car in every 10 minute interval

```
SELECT
    LicensePlate,
    Make,
    Time
FROM
    Input TIMESTAMP BY Time
WHERE
    IsFirst(minute, 10) = 1
```

Find Last Event in a Window

Find last car in every 10 minute interval

```
WITH LastInWindow AS
(
    SELECT
        MAX(Time) AS LastEventTime
    FROM
        Input TIMESTAMP BY Time
    GROUP BY
        TumblingWindow(minute, 10)
)
SELECT
    Input.LicensePlate,
    Input.Make,
    Input.Time
FROM
    Input TIMESTAMP BY Time
    INNER JOIN LastInWindow
    ON DATEDIFF(minute, Input, LastInWindow) BETWEEN 0 AND 10
    AND Input.Time = LastInWindow.LastEventTime
```

There are two steps in the query – the first one finds latest timestamp in 10 minute windows; the second joins results of the first query with original stream to find events matching last timestamps in each window

Detect Duration of a Condition

Find out how long a condition occurred for. For example, suppose that a bug that resulted in all cars having an incorrect weight (above 20,000 pounds) – we want to compute the duration of the bug.

```
WITH SelectPreviousEvent AS
    (
    SELECT
    *,
        LAG([time]) OVER (LIMIT DURATION(hour, 24)) as previousTime,
        LAG([weight]) OVER (LIMIT DURATION(hour, 24)) as previousWeight
    FROM input TIMESTAMP BY [time]
    )

    SELECT
        LAG(time) OVER (LIMIT DURATION(hour, 24) WHEN previousWeight < 20000
) [StartFault],
        previousTime [EndFault]
    FROM SelectPreviousEvent
    WHERE
        [weight] < 20000
        AND previousWeight > 20000
```

SELECT System.Timestamp AS windowEnd, TopOne() OVER (ORDER BY t DESC) AS lastEvent FROM input TIMESTAMP BY t GROUP BY HOPPINGWINDOW(second, 300, 5)

Fill Missing Values

For the stream of events that have missing values, produce a stream of events with regular intervals. For example, generate event every 5 seconds that will report the most recently seen data point.

```
SELECT System.Timestamp AS windowEndSELECT
    System.Timestamp AS windowEnd,
    TopOne() OVER (ORDER BY t DESC) AS lastEvent
FROM
    input TIMESTAMP BY t
GROUP BY HOPPINGWINDOW(second, 300, 5), TopOne() OVER (ORDER BY t DESC) AS
lastEvent FROM input TIMESTAMP BY t GROUP BY HOPPINGWINDOW(second, 300, 5)
```

This query will generate events every 5 second and will output the last event that was received before. Hopping Window duration determines how far back the query will look to find the latest event (300 seconds in this example).

Detect Duration Between Events

Find the duration of a given event. For example, given a web clickstream determine time spent on a feature.

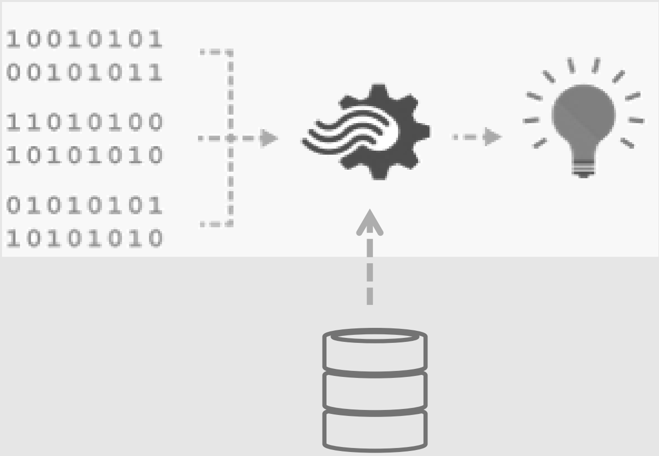
```
SELECT

[user], feature, DATEDIFF(second, LAST(Time) OVER (PARTITION BY [user], feature
LIMIT DURATION(hour, 1) WHEN Event = 'start'), Time) as duration

FROM input TIMESTAMP BY Time

WHERE
        Event = 'end'
```

Reference Data

Correlation of Event Streams with Reference Data

- Static or slowly-changing data stored in blobs
  - Scanned for changes on a settable cadence
- Joins between streams and reference data sources for correlations
  - Reference data appears like another input in the query

```
SELECT myRefData.Name, myStream.Value
FROM myStream
JOIN myRefData
ON myStream.myKey = myRefData.myKey
```



Reference Data Example

Reports all vehicles that entered the toll with expired licenses

```
SELECT ES.EntryTime, ES.LicensePlate, ES.TollId,
RD.RegistrationId
FROM EntryStream ES TIMESTAMP BY EntryTime

JOIN RegistrationData RD
ON ES.LicensePlate = RD.LicensePlate
WHERE RD.Expired = 1
```

Custom Code Support

JavaScript User Defined Functions (UDFs)

Custom Code is Supported Using JavaScript UDFs

- Stateless
- Side-effect-free
- Implementers do not need to concern themselves with sharding, resilience, or resumption

Use Cases

- ASA supports declarative representation of the logically difficult parts of the streaming computations
- Custom-code extensibility is meant for logically simple but technically difficult logic:
  - String parsing and manipulation [e.g. Regexp_Replace() and Regexp_Extract()]
  - Array operations (e.g. sorting, joining, find, fill)
  - Regular Expressions
  - Mathematics operations
  - Date operations

Restrictions

- Callouts to external REST endpoints
- Pulling reference data from an external source
- Custom event format serialization/deserialization on inputs/outputs.
- Custom aggregation functions

JavaScript UDF Example

```
SELECT
    time,
    udf.hex2Int(offset) AS IntOffset
INTO Output
FROM InputStream
```

Local definition file for the JavaScript UDF:

```
{
  "properties": {
    "type": "Scalar",  //Function type. Scalar is the only supported value
    "properties": {
      "inputs": [ // Function input parameter(s).
        {
          "dataType": "any", // Input data type
        }
      ],
      "output": { // Output
        "dataType": "any" // Output data type
      },
      "binding": {
        "type": "Microsoft.StreamAnalytics/JavascriptUdf",
        "properties": { // Function definition
          "script": "function hex2Int(hexValue) {return parseInt(hexValue,
16);}",
        }
      }
    }
  }
}
```

Machine Learning Integration

Azure Machine Learning Callouts in public preview

Perform real-time scoring on streaming data

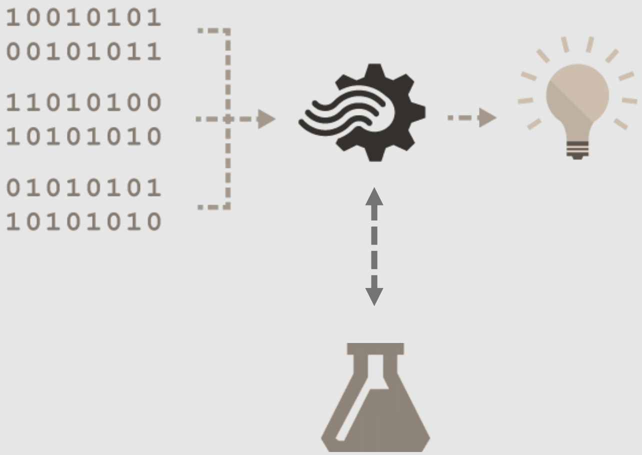
Anomaly Detection and Sentiment Analysis are common use cases

Function calls from the query

Azure ML can publish web endpoints for operationalized ML models

Azure Stream Analytics binds custom function names to such web endpoints

```
SELECT text, sentiment(text) AS score
FROM myStream
```



Geospatial Capabilities

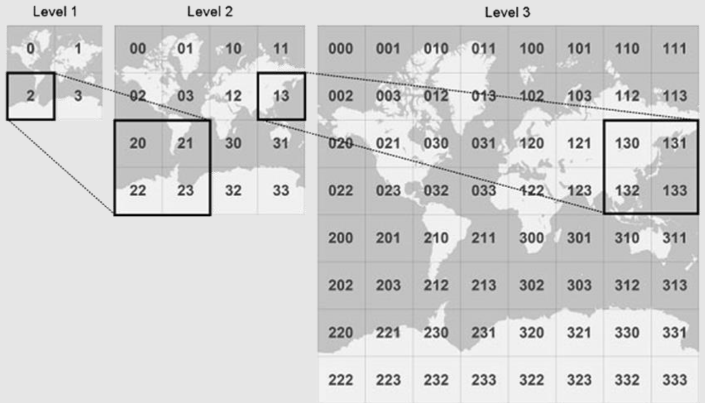
Real-time Geospatial Analytics Scenarios

- Phone Tracking Across Cell Sites
- Connected Car - Remote Management & Diagnostics
- Asset Tracking
- Fleet Management
- Facilities Management
- Personnel Tracking & Crowd Control
- Ride Sharing
- Geofencing
- Racecar Telemetry
- Connected Manufacturing

and many more…

Geospatial Functions

```
CreatePoint
CreatePolygon
CreateLineString

ST_DISTANCE
ST_WITHIN
ST_OVERLAPS
ST_INTERSECTS
```



Level 1

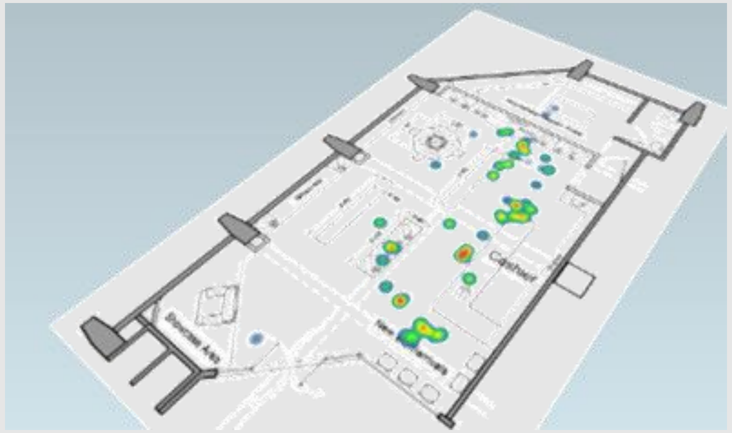
0	1
2	3

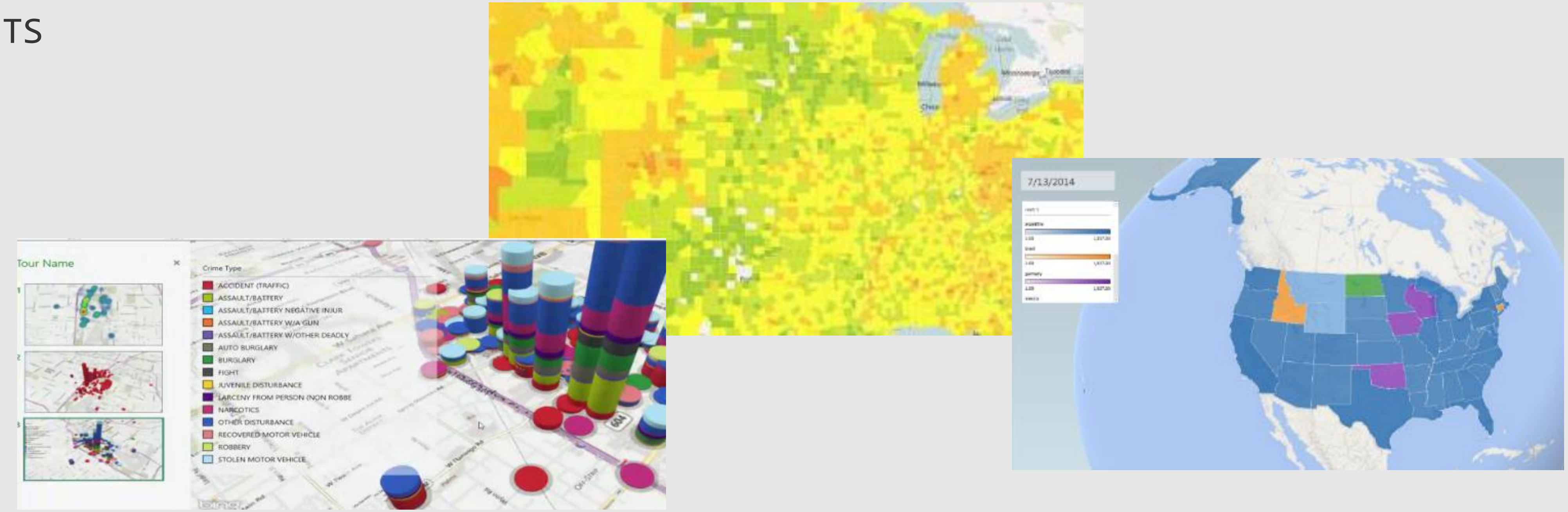
Level 2

00	01	10	11
02	03	12	13
20	21	30	31
22	23	32	33

Level 3

000	001	010	011	100	101	110	111
002	003	012	013	102	103	112	113
020	021	030	031	120	121	130	131
022	023	032	033	122	123	132	133
200	201	210	211	300	301	310	311
202	203	212	213	302	303	312	313
220	221	230	231	320	321	330	331
222	223	232	233	322	323	332	333





Geospatial Examples

Generate an event when gas is less than 50 km from the car

```
SELECT Cars.Location, Station.Location

FROM Cars c

JOIN Station s ON ST_DISTANCE(c.Location, s.Location) < 50 * 1000
```

Generate an event when fuel level is lower than 50%, a gas station is in promotion and course of car is pointing to gas station

```
SELECT Cars.gas, Cars.Location, Cars.Course, Station.Location,
Station.Promotion

FROM Cars c

JOIN Station s ON Cars.gas < 0.5 AND Station.Promotion AND
ST_OVERLAPS(c.Location, c.course)
```

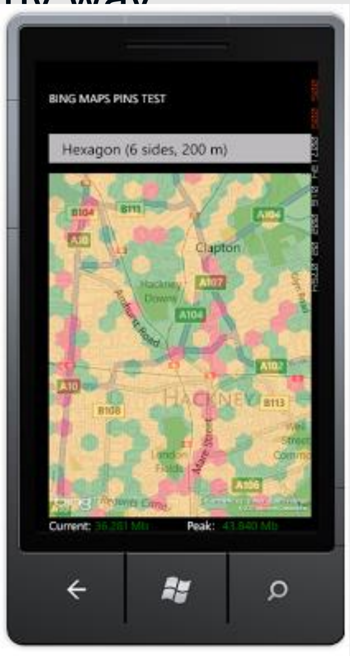
Generate an event when a store is within a possible flooding zone

```
SELECT Store.Polygon, Flooding.Polygon

FROM Cars c

JOIN Flooding f ON ST_OVERLAPS(s.Polygon, f.Polygon)
```

Generate an event when a storm is heading my way

```
SELECT Cars.Location, Storm.Course

FROM Cars c

JOIN Storm s ON ST_OVERLAPS(c.Location, s.Cours
```

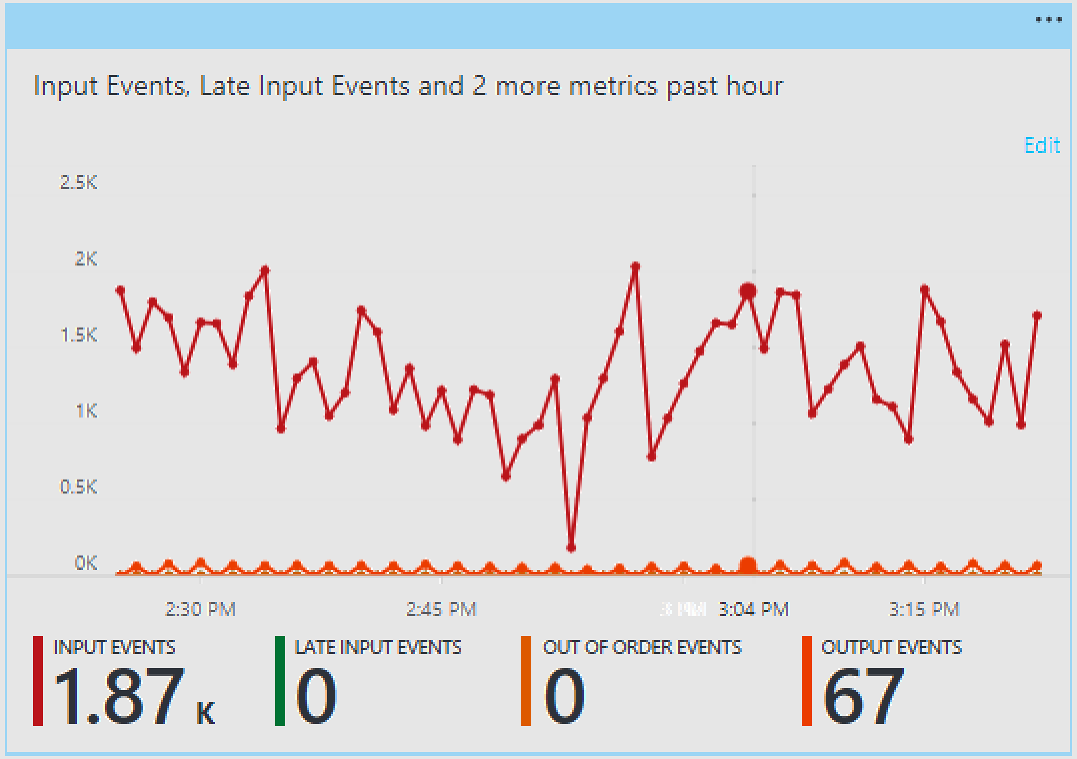
Millwall Outer Dock
Mudchute Park
London
Average prices
£40k
£70k
£120k
£170k
£250k
£500k
Microsoft Corporatio



Combination of clustering and heat maps. Clusters are represented using color coded geometric shapes that fit together evenly

Monitoring and Troubleshooting

Job Level Monitoring



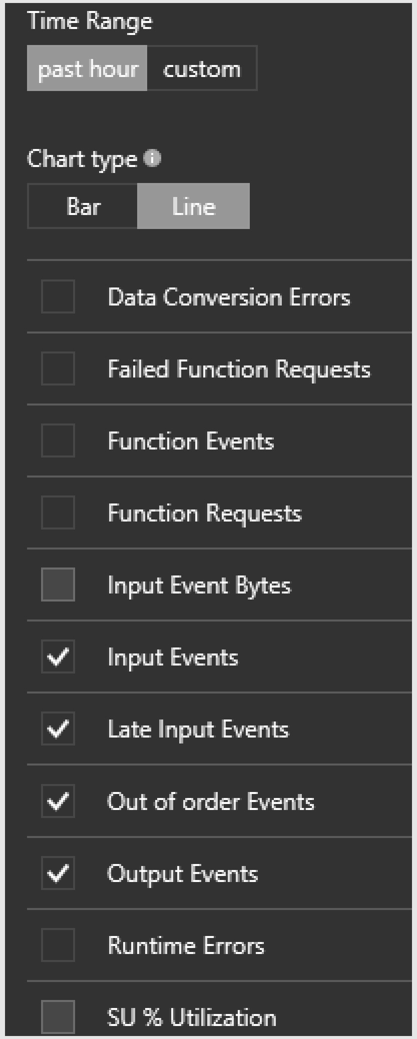
Input Events, Late Input Events and 2 more metrics past hour

INPUT EVENTS 1.87 K

LATE INPUT EVENTS 0

OUT OF ORDER EVENTS 0

OUTPUT EVENTS 67



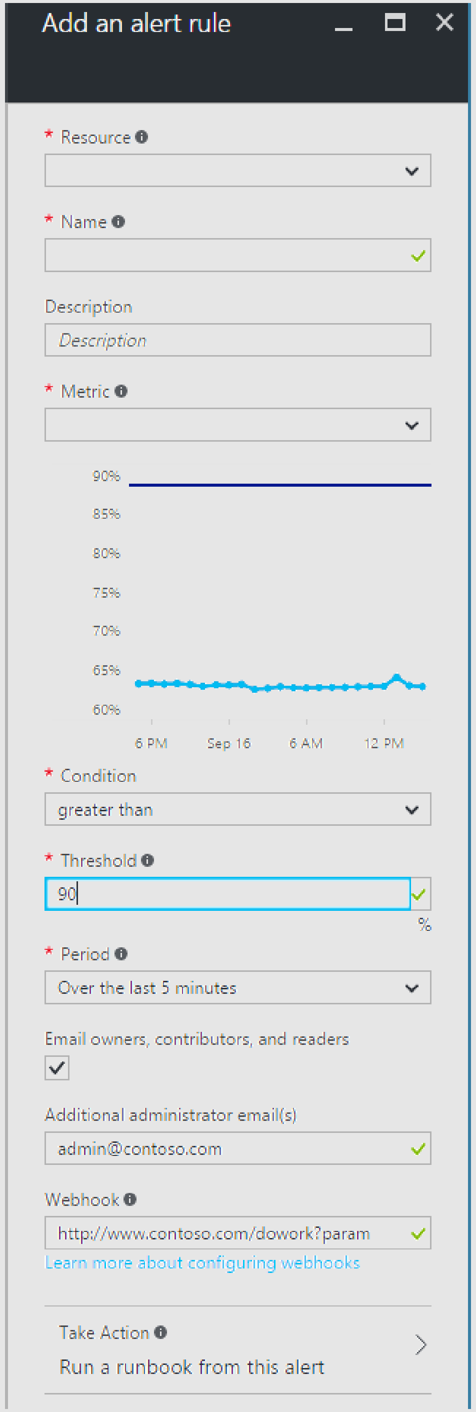
Time Range

[✓ past hour] custom

Chart type

Bar [✓ Line]

- □ Data Conversion Errors
- □ Failed Function Requests
- □ Function Events
- □ Function Requests
- □ Input Event Bytes
- ☑ Input Events
- ☑ Late Input Events
- ☑ Out of order Events
- ☑ Output Events
- □ Runtime Errors
- □ SU % Utilization



Add an alert rule

* Resource

* Name

Description

Description

* Metric

* Condition

greater than

* Threshold

90

%

* Period

Over the last 5 minutes

Email owners, contributors, and readers

☑

Additional administrator email(s)

admin@contoso.com

Webhook

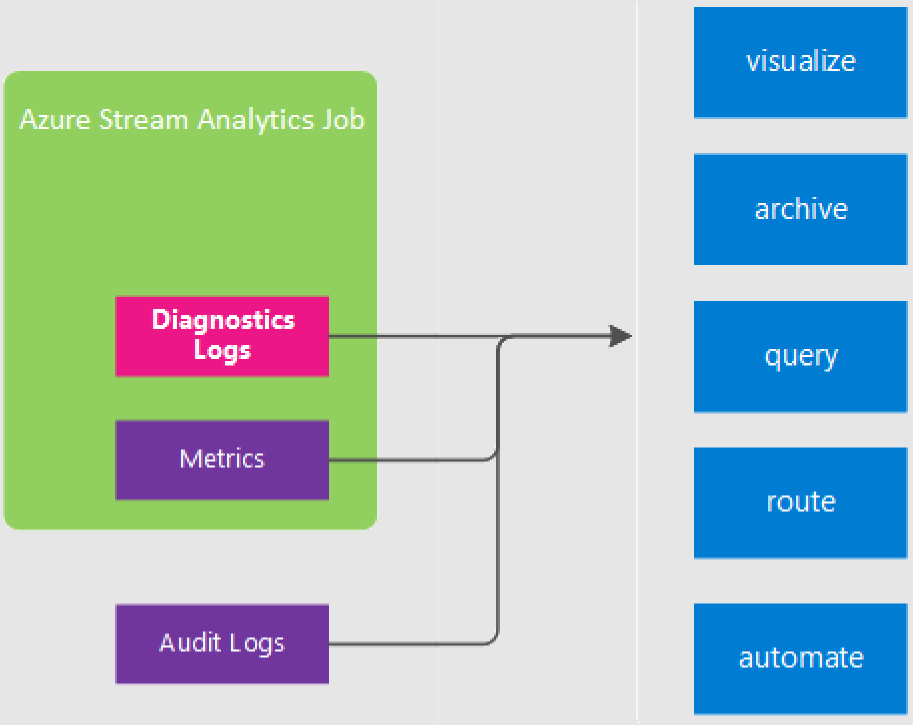
http://www.contoso.com/dowork?param

Learn more about configuring webhooks

Take Action

Run a runbook from this alert

Diagnostic Logs

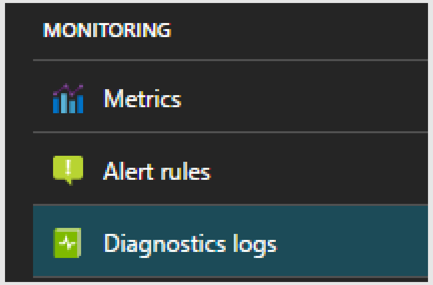


“like stderr for ASA”

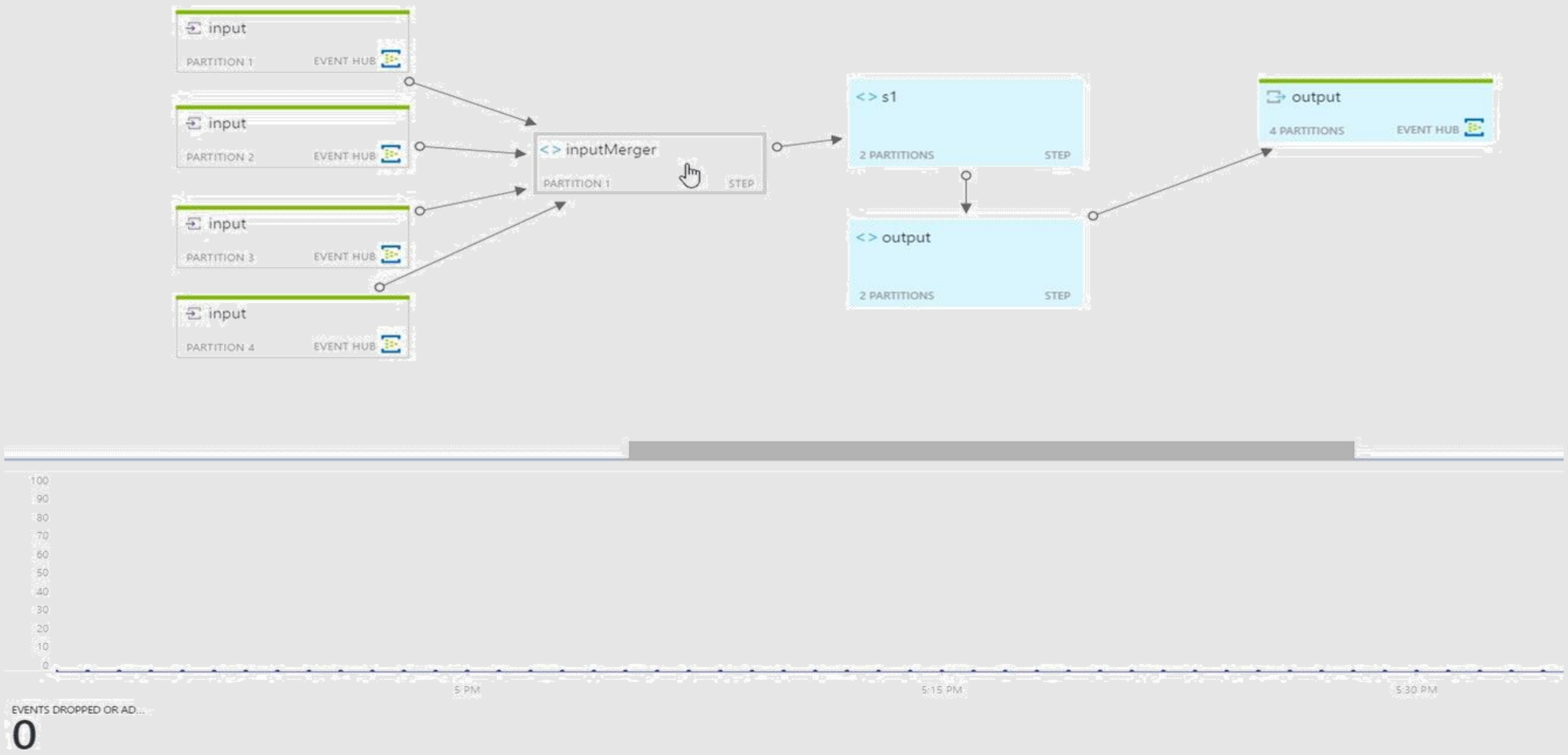
Event Hub: Monitor an ASA job with another ASA job

Storage account: Can potentially include PII data

Log Analytics

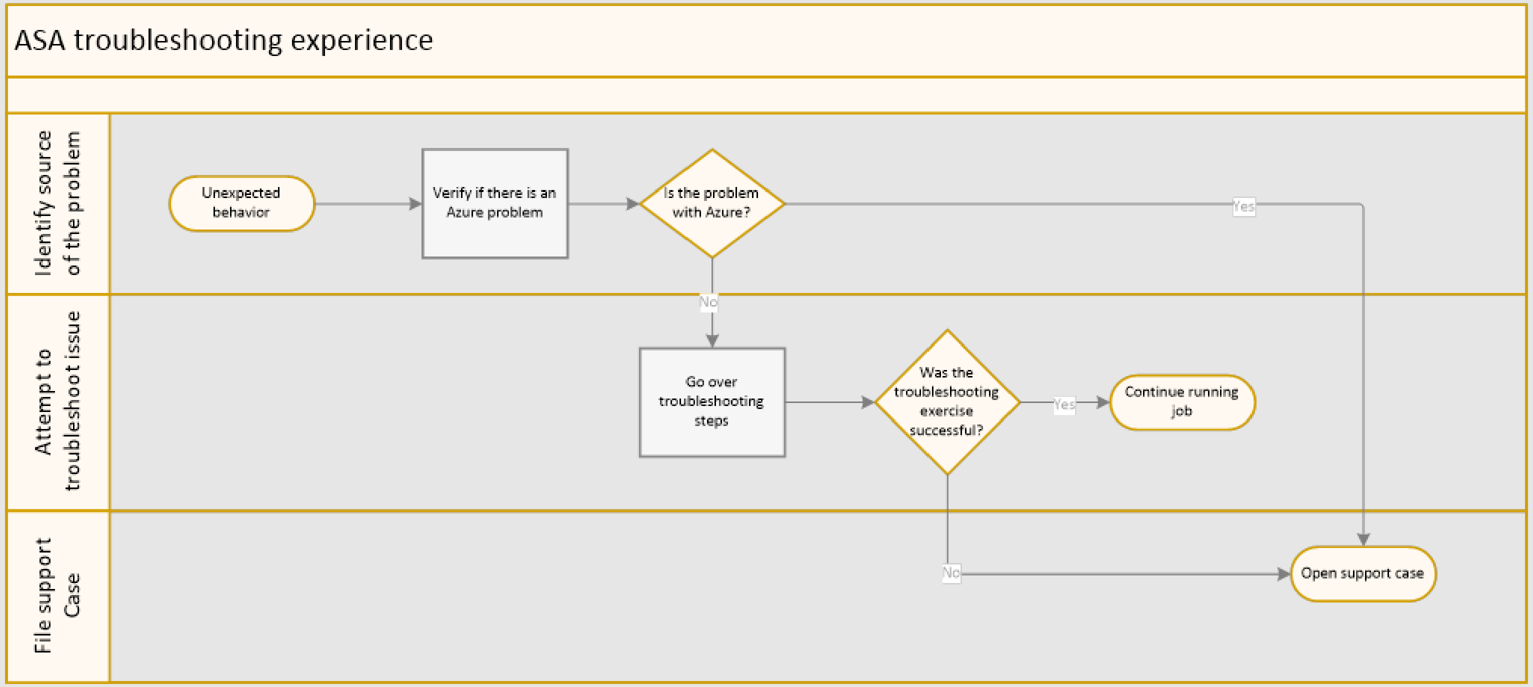


Job Diagram with Metrics



Resource Health

“is there a problem with Azure Stream Analytics that is affecting my job”



ASA troubleshooting experience

Identify source of the problem

Attempt to troubleshoot issue

File support Case

Unexpected behavior

Verify if there is an Azure problem

Is the problem with Azure?

Yes

No

Go over troubleshooting steps

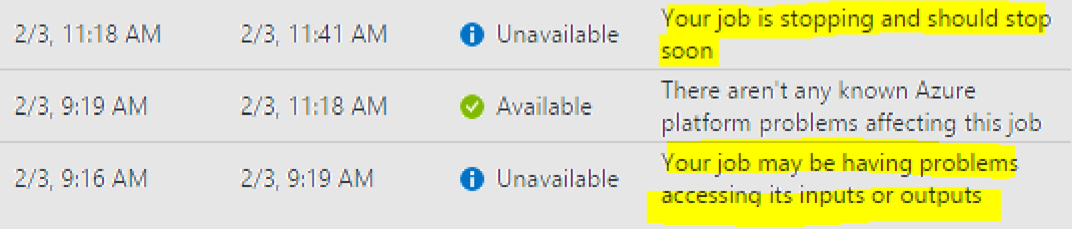
Was the troubleshooting exercise successful?

Yes

Continue running job

No

Open support case



2/3, 11:18 AM	2/3, 11:41 AM	ℹ Unavailable	Your job is stopping and should stop soon
2/3, 9:19 AM	2/3, 11:18 AM	✓ Available	There aren't any known Azure platform problems affecting this job
2/3, 9:16 AM	2/3, 9:19 AM	ℹ Unavailable	Your job may be having problems accessing its inputs or outputs

Seamless Scalability

Streaming Units

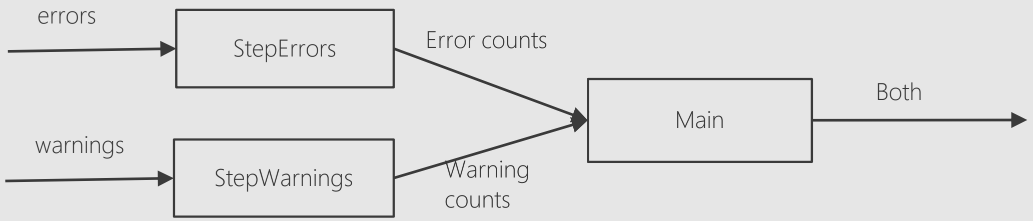
Represents the computing resources footprint of a Azure Stream Analytics job

Seamlessly add/remove Streaming Units

Vertical Partitioning of the Query

Vertically partition into logical steps

Using a named WITH-AS in SAQL



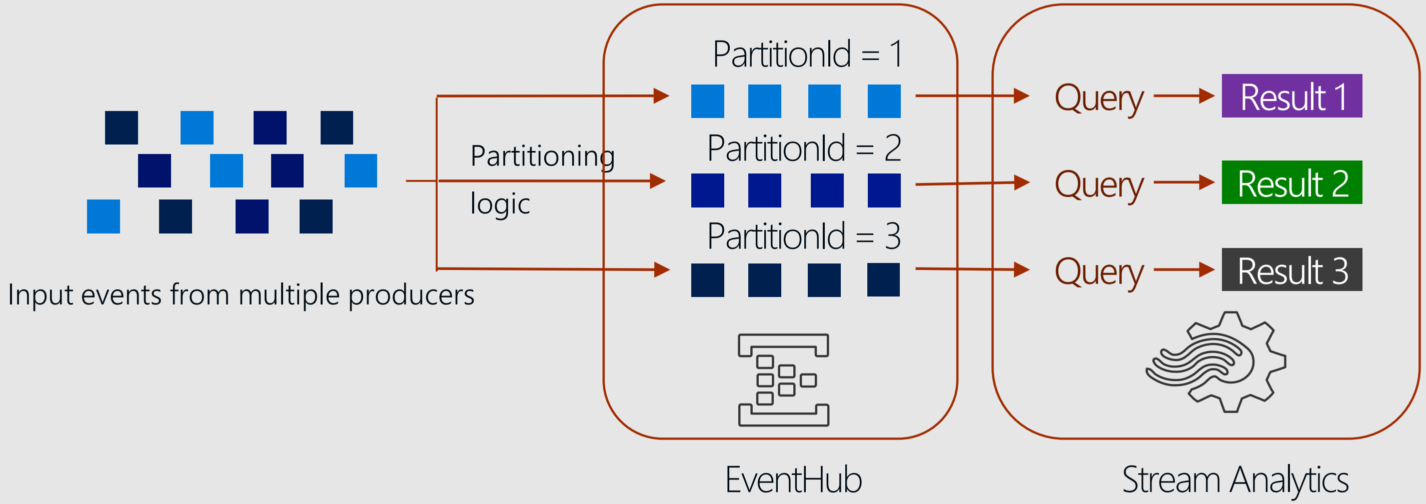
```
WITH
StepErrors AS
(
      SELECT machine, count(*)
      FROM Errors
      GROUP BY machine
),
StepWarnings AS
(
      SELECT machine, count(*)
      FROM Warnings
      GROUP BY machine
)
SELECT machine, e.count as numErrors,
w.count as numWarnings
FROM StepErrors e JOIN StepWarnings w
ON e.machine = w.machine
WHERE e.count < w.count * 2
```

Horizontal Partitioning for Scale

Horizontal sharding into data-parallel execution over scaled-out resources

User specifies partitioning keys for each stage

- Partitioning is "safe" if the partitioning key is also a grouping key
- Otherwise, incomplete groupings are computed per partition
  - Explicit global grouping step is required to reconcile



User specifies bound on the number of SUs

Partitioning keys are assigned to shards using a built-in hash function

```
SELECT COUNT(*) AS Count, TollBoothId
FROM EntryStream PARTITION BY PartitionId
GROUP BY TumblingWindow (minute, 3), TollBoothId
```

Out-of-order and Late-arriving Events

Each individual stream is always in-order of time

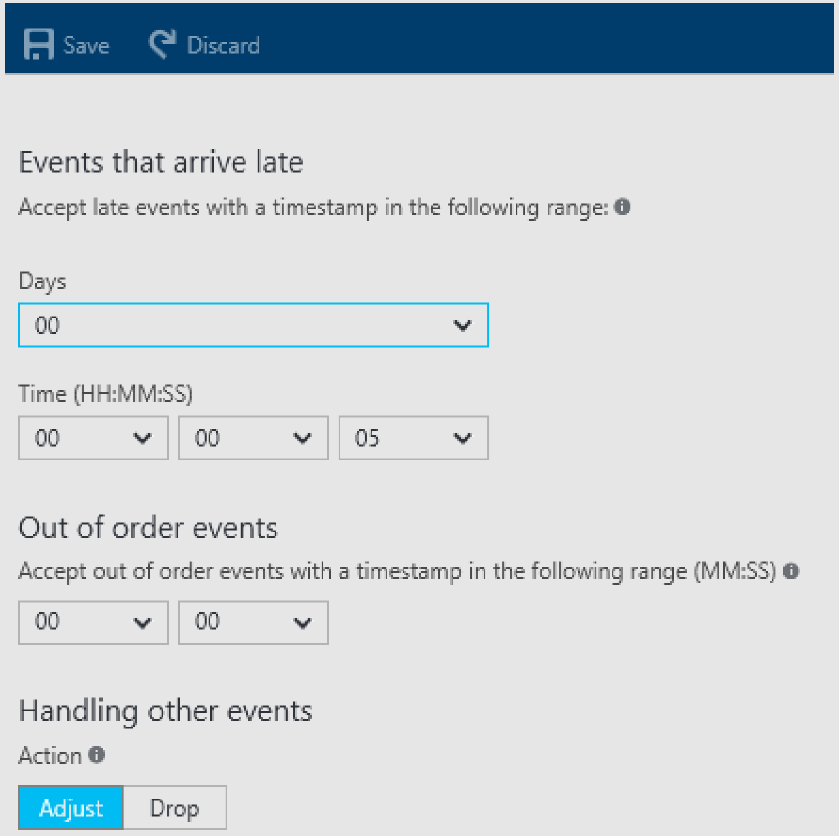
- Input streams that are not in-order are either:
  - Sorted (and therefore delayed!)
  - Adjusted by the system, as per a user-specified policy

Additional "punctuation" events

- Advance the time in the absence of event arrivals
  - "Notify me when no logins occur for 3 minutes"

Tolerance for lateness

- Events arrive out of order but within the tolerance: Re-ordered by timestamp
- Events arrive later than tolerance: Dropped or Adjusted
  - Adjust - Adjusted to appear to have arrived at the latest still acceptable time
  - Drop - Discarded



Events that arrive late

Accept late events with a timestamp in the following range:

Days: 00

Time (HH:MM:SS): 00 00 05

Out of order events

Accept out of order events with a timestamp in the following range (MM:SS): 00 00

Handling other events

Action: [✓ Adjust] Drop

Interacting with Stream Analytics

Modalities of Interaction & Programmability

Azure Portal

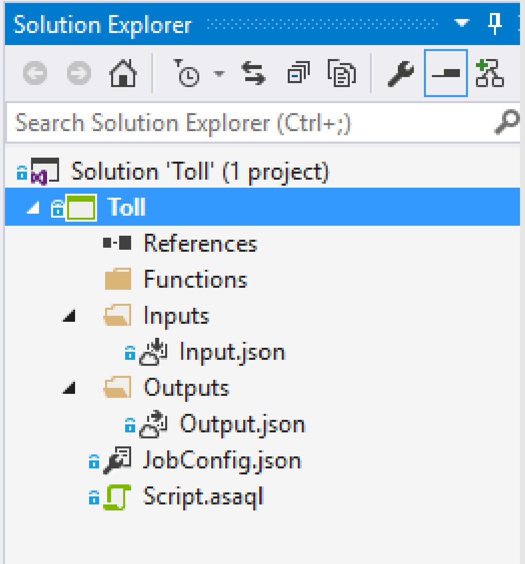
Visual Studio

PowerShell

.NET SDK

REST APIs

Visual Studio Support



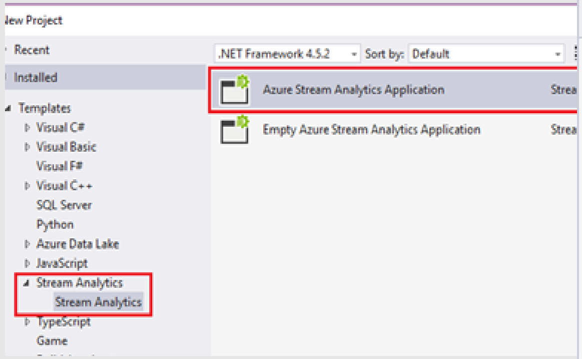
Version Control

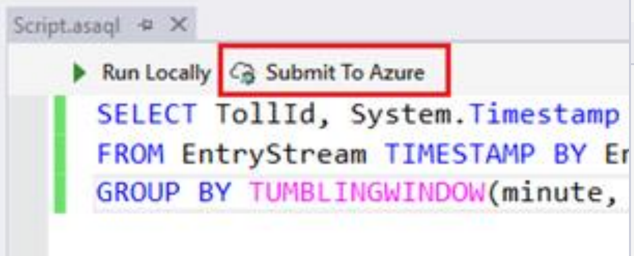
Author with Intellisense

Test Locally

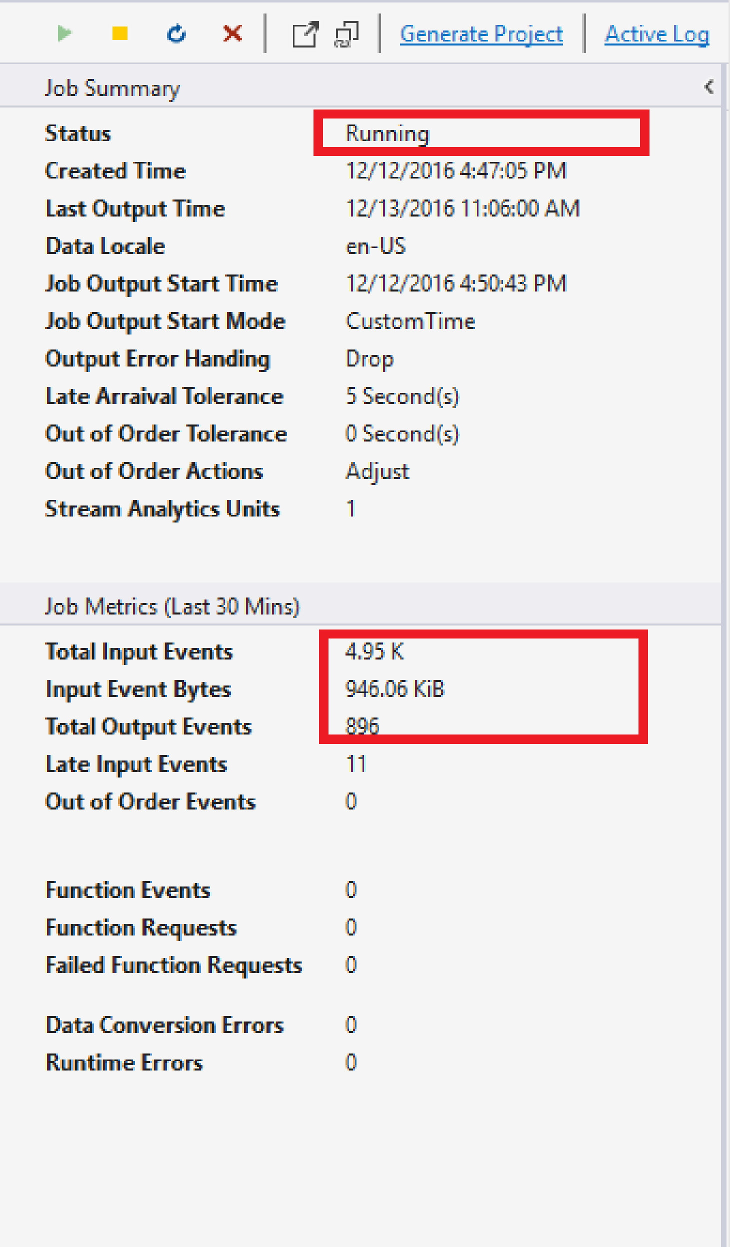
Deploy to Azure

Monitor





```
SELECT TollId, System.Timestamp
FROM EntryStream TIMESTAMP BY Er
GROUP BY TUMBLINGWINDOW(minute,
```

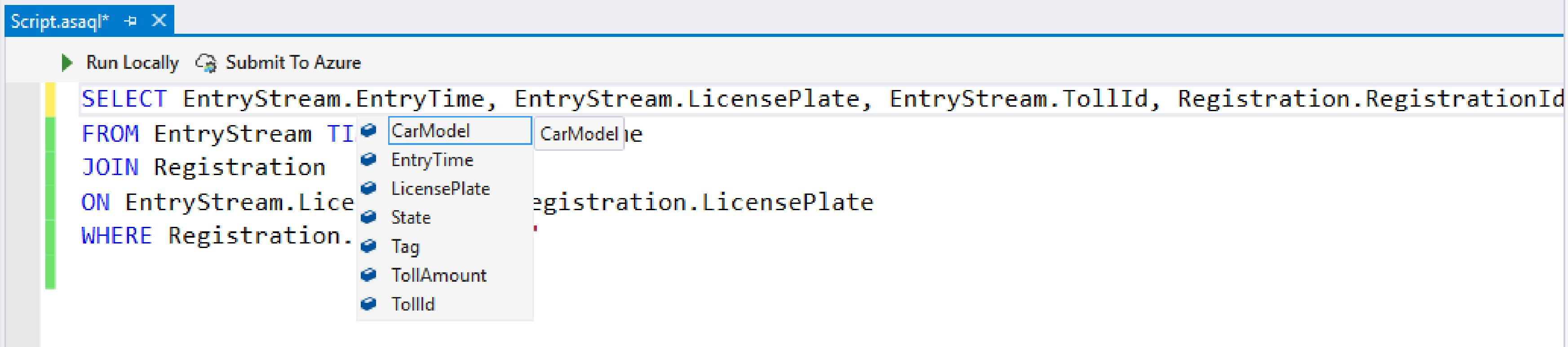


Job Summary

Status	Running
Created Time	12/12/2016 4:47:05 PM
Last Output Time	12/13/2016 11:06:00 AM
Data Locale	en-US
Job Output Start Time	12/12/2016 4:50:43 PM
Job Output Start Mode	CustomTime
Output Error Handing	Drop
Late Arraival Tolerance	5 Second(s)
Out of Order Tolerance	0 Second(s)
Out of Order Actions	Adjust
Stream Analytics Units	1

Job Metrics (Last 30 Mins)

Total Input Events	4.95 K
Input Event Bytes	946.06 KiB
Total Output Events	896
Late Input Events	11
Out of Order Events	0
Function Events	0
Function Requests	0
Failed Function Requests	0
Data Conversion Errors	0
Runtime Errors	0



```
SELECT EntryStream.EntryTime, EntryStream.LicensePlate, EntryStream.TollId, Registration.RegistrationId
FROM EntryStream TI                        e
JOIN Registration
ON EntryStream.Lice                      egistration.LicensePlate
WHERE Registration.
```

Edge Analytics

Edge Analytics

Local Execution

Stream analytics runs on 'edge devices'

Unlock the Value of Untapped data

Only ~5% of data in industrial processes is sent to the cloud

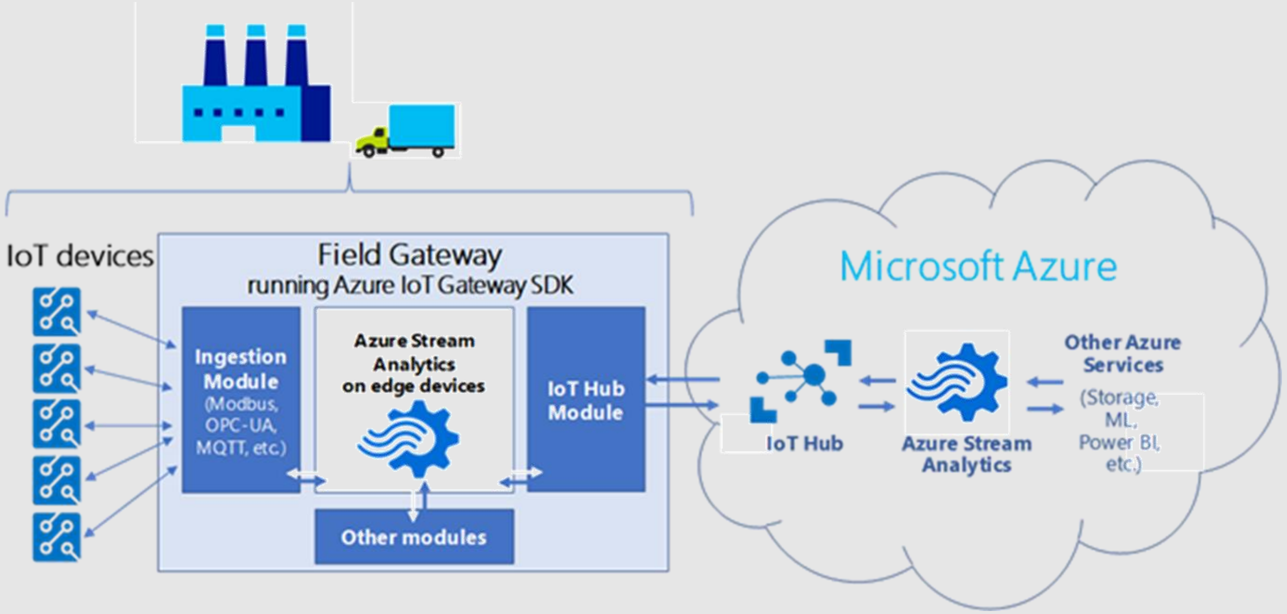
Deploy intelligence near the data to unlock the full value of data

Seamless development and operations

Stream analytics jobs run in the cloud and on edge devices

Intelligent actions

Deploy situational awareness, custom code, ML models on the edge



Edge Analytics Scenarios

Low Latency

Resiliency

Efficient Use of Bandwidth

Compliance

Management at IoT-Scale

Canonical Use Cases

Reduction

When you are interested in only parts or significant changes in your operational data

Aggregation

When business operations need an aggregate view

Batching

When connectivity is intermittent and cost is high

Transformation

Converting messages from legacy industrial automation to modern applications formats

Edge Intelligence

Machine Learning models on the edge

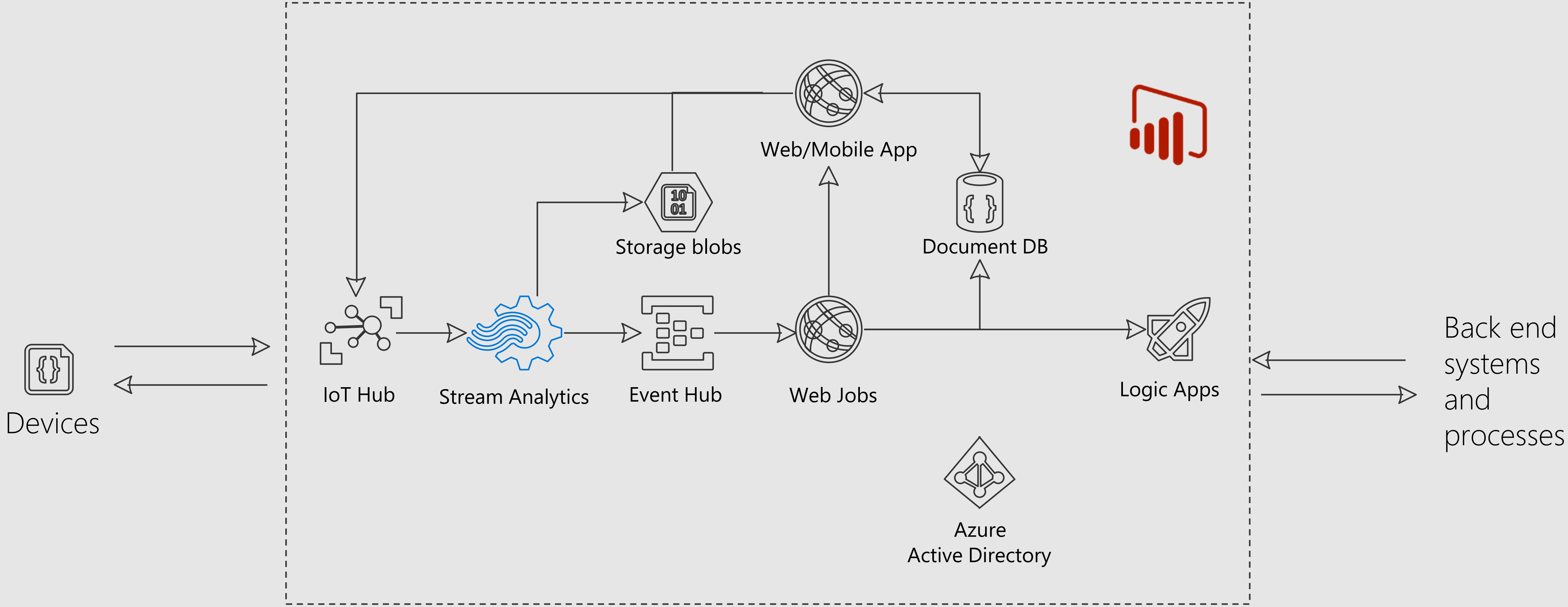
Reference data

Edge Analytics Platform Requirements

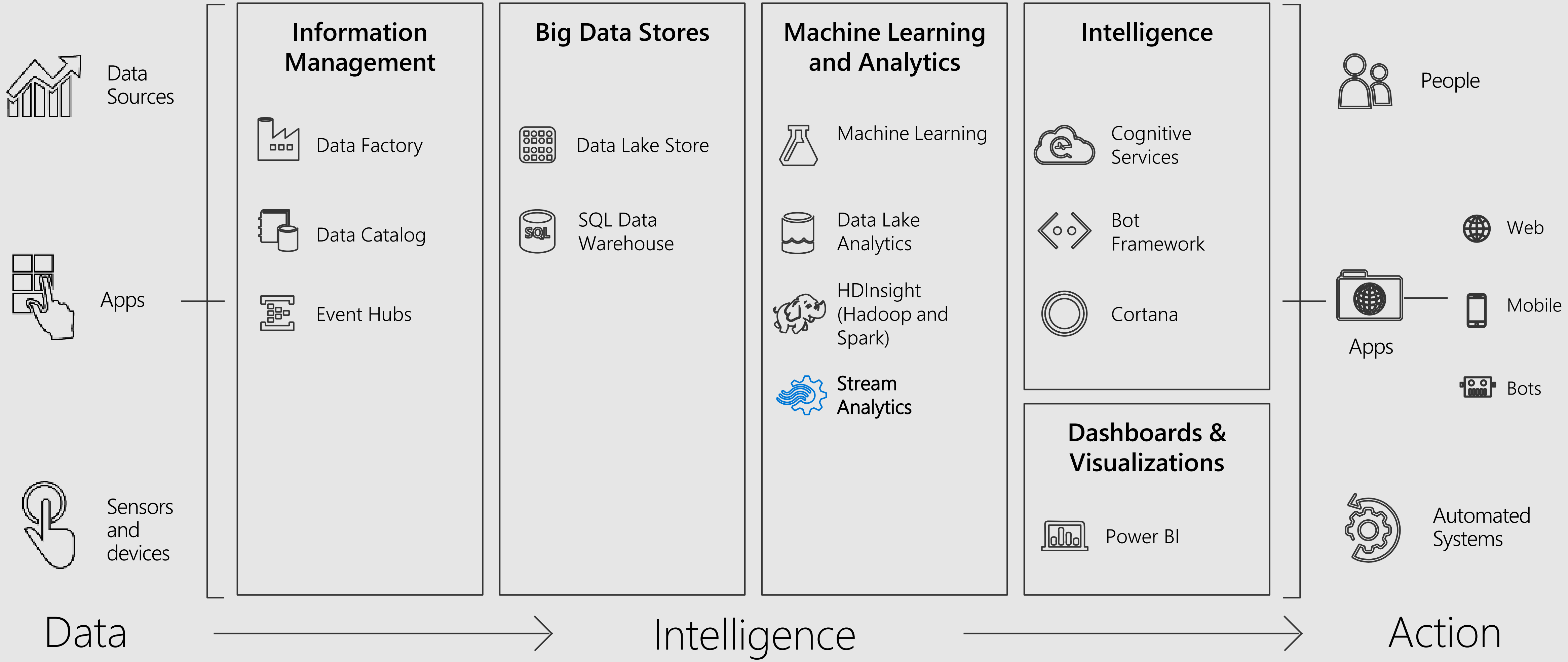
- Windows or Linux
- Azure IoT Gateway
- Memory
  - Minified stream analytics engine needs ~2MB of main memory
  - Storage and additional memory based on the amount of data
- Node.js

Stream Analytics is a Key Part of Multiple Strategic Bets

Microsoft IoT Suite



Microsoft Cortana Intelligence Suite



Data Sources

Apps

Sensors and devices

Information Management

Data Factory

Data Catalog

Event Hubs

Big Data Stores

Data Lake Store

SQL Data Warehouse

Machine Learning and Analytics

Machine Learning

Data Lake Analytics

HDInsight (Hadoop and Spark)

Stream Analytics

Intelligence

Cognitive Services

Bot Framework

Cortana

Dashboards & Visualizations

Power BI

People

Apps

Web

Mobile

Bots

Automated Systems

Data → Intelligence → Action

Streaming Options on Azure

Streaming Options with Microsoft

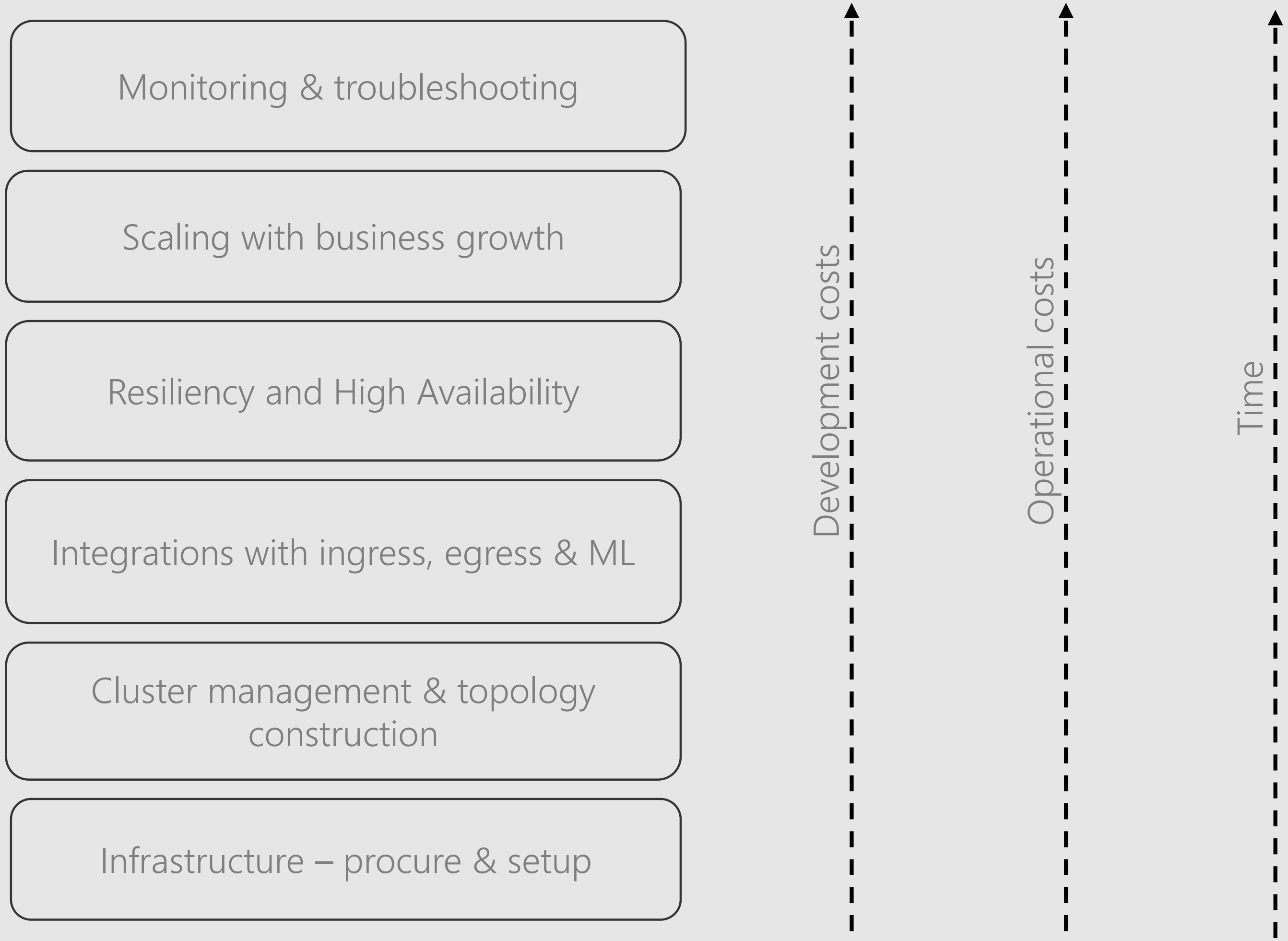
Azure Stream Analytics

Storm on Azure HDInsight

Spark Streaming on Azure HDInsight

StreamInsight in SQL Server

Streaming Stack



Monitoring & troubleshooting

Scaling with business growth

Resiliency and High Availability

Integrations with ingress, egress & ML

Cluster management & topology construction

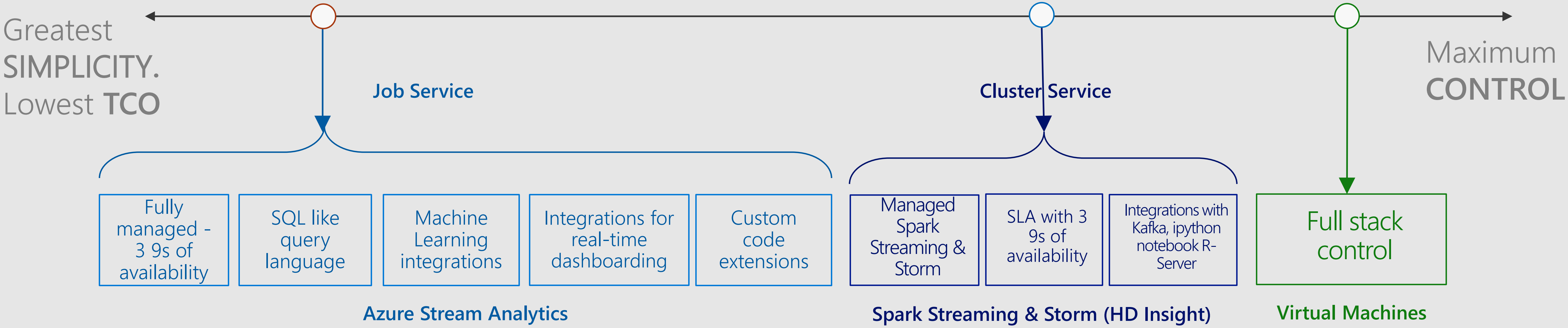
Infrastructure – procure & setup

Development costs

Operational costs

Time

Tradeoff Spectrum



Greatest **SIMPLICITY.**
Lowest **TCO**

Job Service

- Fully managed - 3 9s of availability
- SQL like query language
- Machine Learning integrations
- Integrations for real-time dashboarding
- Custom code extensions

Azure Stream Analytics

Cluster Service

- Managed Spark Streaming & Storm
- SLA with 3 9s of availability
- Integrations with Kafka, ipython notebook R-Server

Spark Streaming & Storm (HD Insight)

Full stack control

Virtual Machines

Maximum
CONTROL

Apache Storm on Azure HDInsight

Open source Apache Storm as a managed service

- SLA of 99.9% up time

Built in Scale-up & Scale-down capability

- Customers can scale up and scale down a running cluster; with no impact to a running topology

Deep integration with Event hub

Program your Storm Topology in Java or C#

Rich Visual Studio experience

- Debug, monitor, and troubleshoot
- Manage and deploy topologies

Fully integrated Azure portal experience

Write your data into SQL Azure, DocumentDB, PowerBI, …

Spark on Azure HDInsight

Managed Service

Fully supported by Microsoft and Hortonworks

Latest open source Spark 2.0 with 100+ stability fixes (available later this week on 9/30)

Enterprise Readiness

99.9% uptime SLA

Compliance: PCI, ISO 27018, SOC, HIPAA, EU-MC

Azure Integrations

Optimized for data exploration, experimentation and development

Jupyter Notebooks (Scala, Python, Automatic data visualizations)

IntelliJ and Eclipse plugins (job submission, remote debugging)

ODBC connectors: Power BI, Tableau, Qlik, SAP, Excel, etc.

Technical Comparison

	Storm on HDInsight	Spark Streaming on HD Insight	Azure Stream Analytics
Strictest Guarantee	At-least-once (exactly once with Trident)	Exactly once	At-least-once
Processing Model	Event-at-a-time (micro batching with Trident)	Micro batching	Adaptive batching (event-at-a-time)
Programing Language Support	Java, C#	Scala, Python, Java	SAQL: SQL like query language, JavaScript UDFs
Open Source	Yes	Yes	No