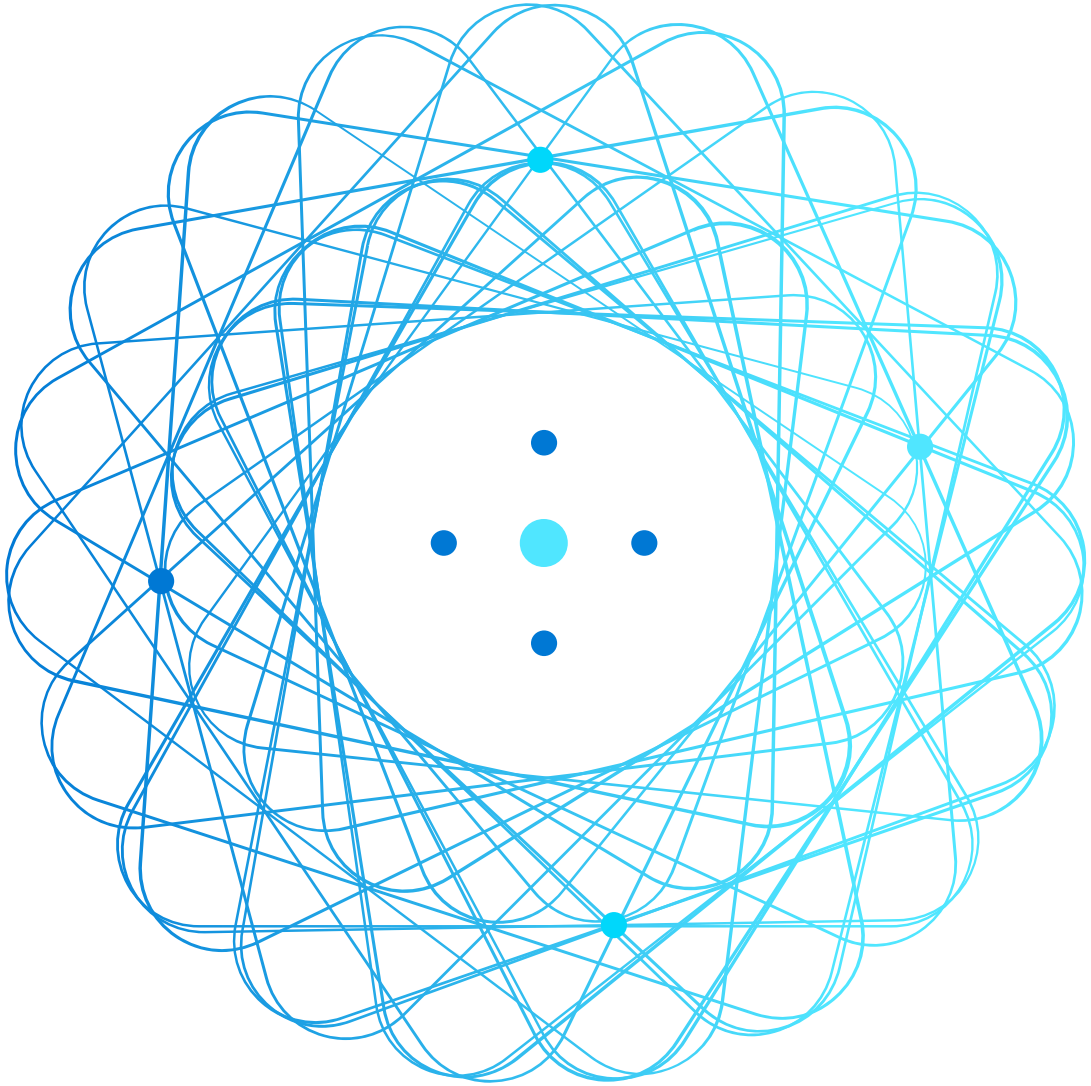
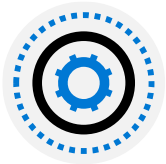
Microsoft Azure

AZ-220T01
Module 04: Message processing and analytics

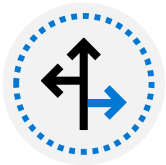


Lesson 01: Learning objectives

Module 4 – Learning objectives



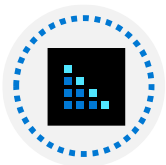
Configure message and event routing



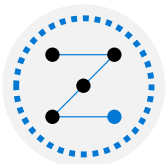
Route data to the built-in and custom endpoints

Implement message enrichment

Implement Azure Stream Analytics Inputs, Queries, and Outputs

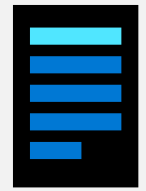


Store message data in a warm storage for archival purposes and additional analysis

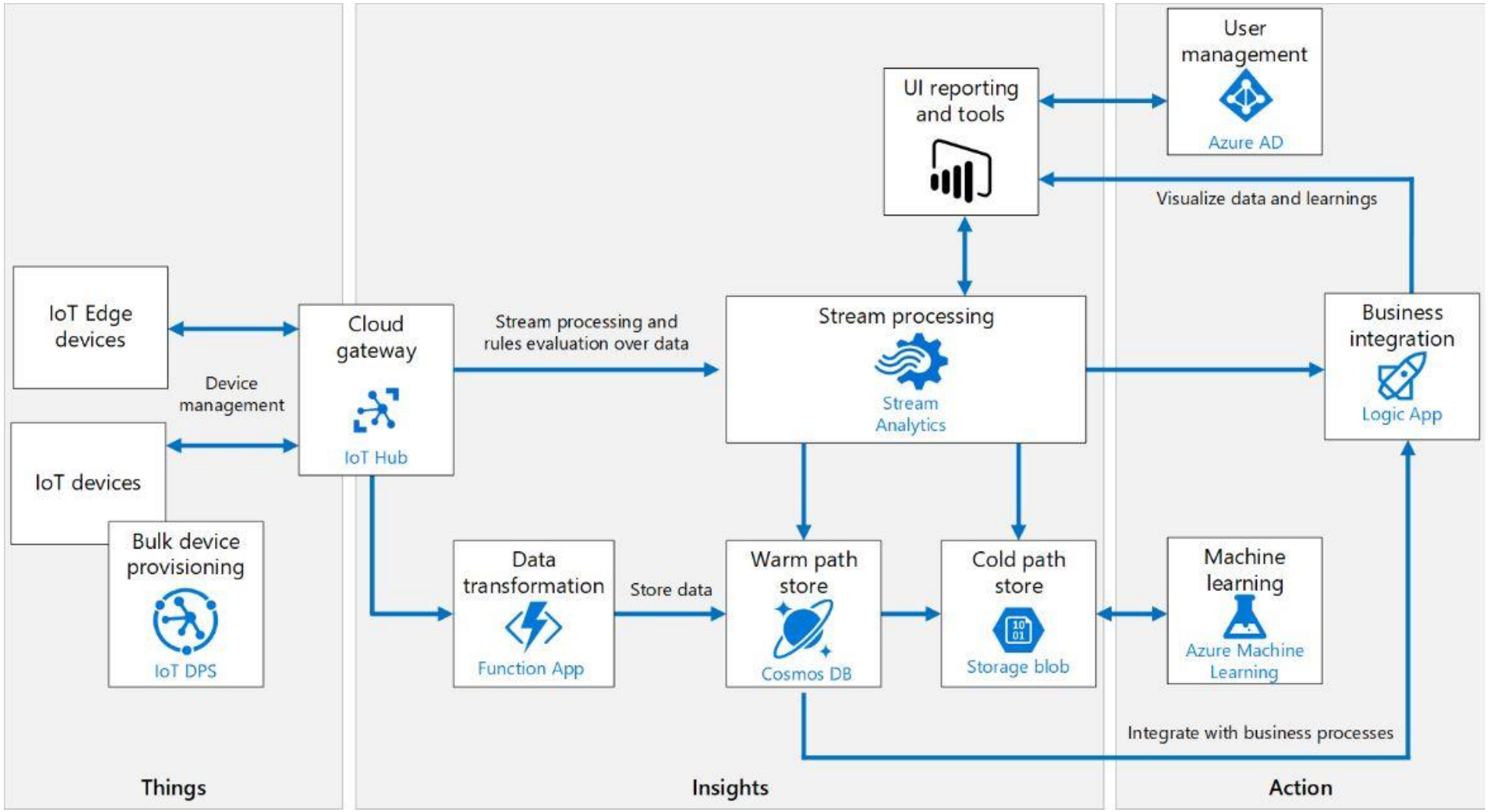


Use an Azure Function within a message processing and analytics solution

Lesson 02: Messages and message processing



What is message processing?



IoT Hub common message format

Purpose

Characteristics:

Retained up to 7 days

256 KB size limit

Partitioned based on deviceId

Supports message enrichment

Components:

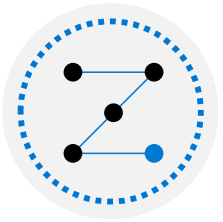
System properties

Application properties – do not require deserialization

Message body – Opaque

```
{
  "message": {
    "systemProperties": {
      "contentType": "application/json",
      "contentEncoding": "UTF-8",
      "iothub-message-source": "deviceMessages",
      "iothub-enqueuedtime": "2017-05-08T18:55:31.8514657Z"
    },
    "appProperties": {
      "processingPath": "{cold | warm | hot}",
      "verbose": "{true, false}",
      "severity": 1-5,
      "testDevice": "{true | false}"
    },
    "body": "{\"Weather\":{\"Temperature\":50}}"
  }
}
```

Introduction to message routing



Endpoints:

Built-in

Custom

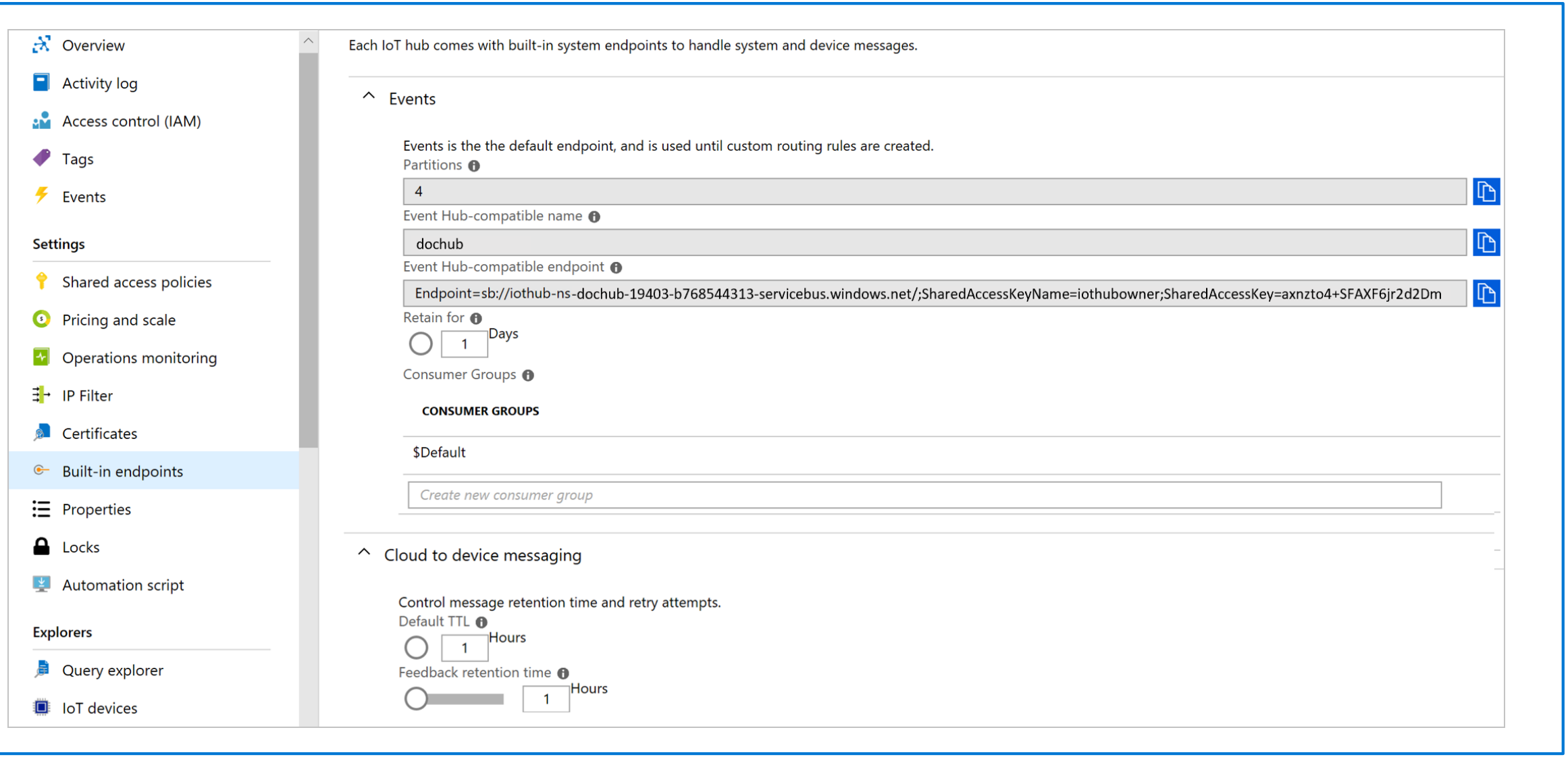
Queries:

Source

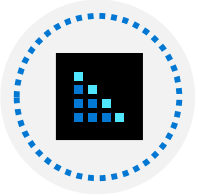
Condition

Endpoint

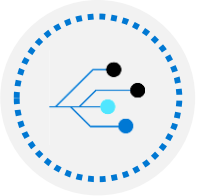
IoT Hub built-in endpoint



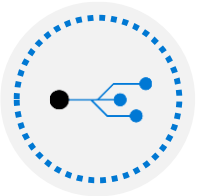
IoT Hub custom endpoints



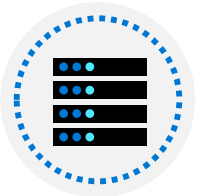
Azure Storage containers



Event Hubs



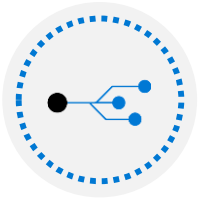
Service Bus queues



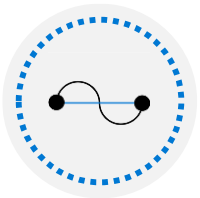
Service Bus topics

Message routing to multiple endpoints

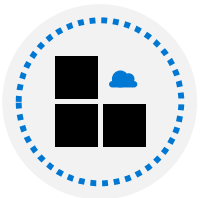
Configuration



Endpoint options



Fallback route



Non-telemetry events

Message routing query syntax

Sample message

```
{
 "message": {
  "systemProperties": {
    "contentType": "application/json",
    "contentEncoding": "UTF-8",
    "iothub-message-source": "deviceMessages",
    "iothub-enqueuedtime": "2017-05-08T18:55:31.8514657Z"
  },
  "appProperties": {
   "processingPath": "{cold | warm | hot}",
   "verbose": "{true, false}",
   "severity": 1-5,
   "testDevice": "{true | false}"
  },
  "body": "{\"Weather\":{\"Temperature\":50}}"
 }
}
```

Message property query example

```
$contentEncoding = 'UTF-8' AND processingPath = 'hot'
```

Message body query example

```
$body.Weather.Temperature = 50 AND processingPath =
'hot'
```

Message routing query syntax

Device twin sample

```
"tags": {
  "deploymentLocation": {
    "building": "43",
    "floor": "1"
  } },
"properties": {
  "desired": {
    "telemetryConfig": {
      "sendFrequency": "5m"
    },
    "$metadata" : {...},
    "$version": 1
  },
  "reported": {
    "telemetryConfig": {
      "sendFrequency": "5m",
      "status": "success"
    },
    "batteryLevel": 55,
    "$metadata" : {...},
    "$version": 4
  } }
```

Device twin sample

```
$twin.properties.desired.telemetryConfig.sendFrequency = '5m’

$body.Weather.Temperature = 50 AND
$twin.properties.desired.telemetryConfig.sendFrequency = '5m'
```

Lesson 3: Additional considerations for IoT hub messaging

Message routing or Event Grid: Differences

Feature	IoT Hub message routing	IoT Hub integration with Event Grid
Device messages and events	Yes, message routing can be used for telemetry data, report device twin changes, device lifecycle events, and digital twin change events	Yes, Event Grid can be used for telemetry data but can also report when devices are created, deleted, connected, and disconnected from IoT Hub
Ordering	Yes, ordering of events is maintained	No, order of events is not guaranteed
Filtering	Rich filtering on message application properties, message system properties, message body, device twin tags, and device twin properties. Filtering isn't applied to digital twin change events.	Filtering based on event type, subject type and attributes in each event. For examples, see Understand filtering events in Event Grid Subscriptions. When subscribing to telemetry events, you can apply additional filters on the data to filter on message properties, message body and device twin in your IoT Hub, before publishing to Event Grid.
Endpoints	• Event Hubs • Azure Blob Storage • Service Bus queue • Service Bus topics	• Azure Functions • Azure Automation • Event Hubs • Logic Apps • Storage Blob • Custom Topics • Queue Storage • Microsoft Flow • Third-party services through WebHooks
Cost	There is no separate charge for message routing. Only ingress of telemetry into IoT Hub is charged. For example, if you have a message routed to three different endpoints, you are billed for only one message	There is no charge from IoT Hub. Event Grid offers the first 100,000 operations per month for free, and then $0.60 per million operations afterwards

Message routing or Event Grid: Similarities

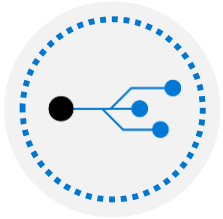
Feature	IoT Hub message routing	IoT Hub integration with Event Grid
Maximum message size	256 KB, device-to-cloud	256 KB, device-to-cloud
Reliability	High: Delivers each message to the endpoint at least once for each route. Expires all messages that are not delivered within one hour	High: Delivers each message to the webhook at least once for each subscription. Expires all events that are not delivered within 24 hours
Scalability	High: Optimized to support millions of simultaneously connected devices sending billions of messages	High: Capable of routing 10,000,000 events per second per region
Latency	Low: Near-real time	Low: Near-real time
Send to multiple endpoints	Yes, send a single message to multiple endpoints	Yes, send a single message to multiple endpoints
Security	Iot Hub provides per-device identity and revocable access control.	Event Grid provides validation at three points: event subscriptions, event publishing, and webhook event delivery.

Message routing or Event Grid: How to choose

What kind of data are you sending to the endpoints?

IoT Hub message routing to send telemetry data to other services

The IoT Hub integration with Event Grid works with events that occur in the IoT Hub service



What endpoints need to receive this information?

IoT Hub message routing supports limited number of unique endpoints and endpoint types, but you can build connectors to reroute the data and events to additional endpoints

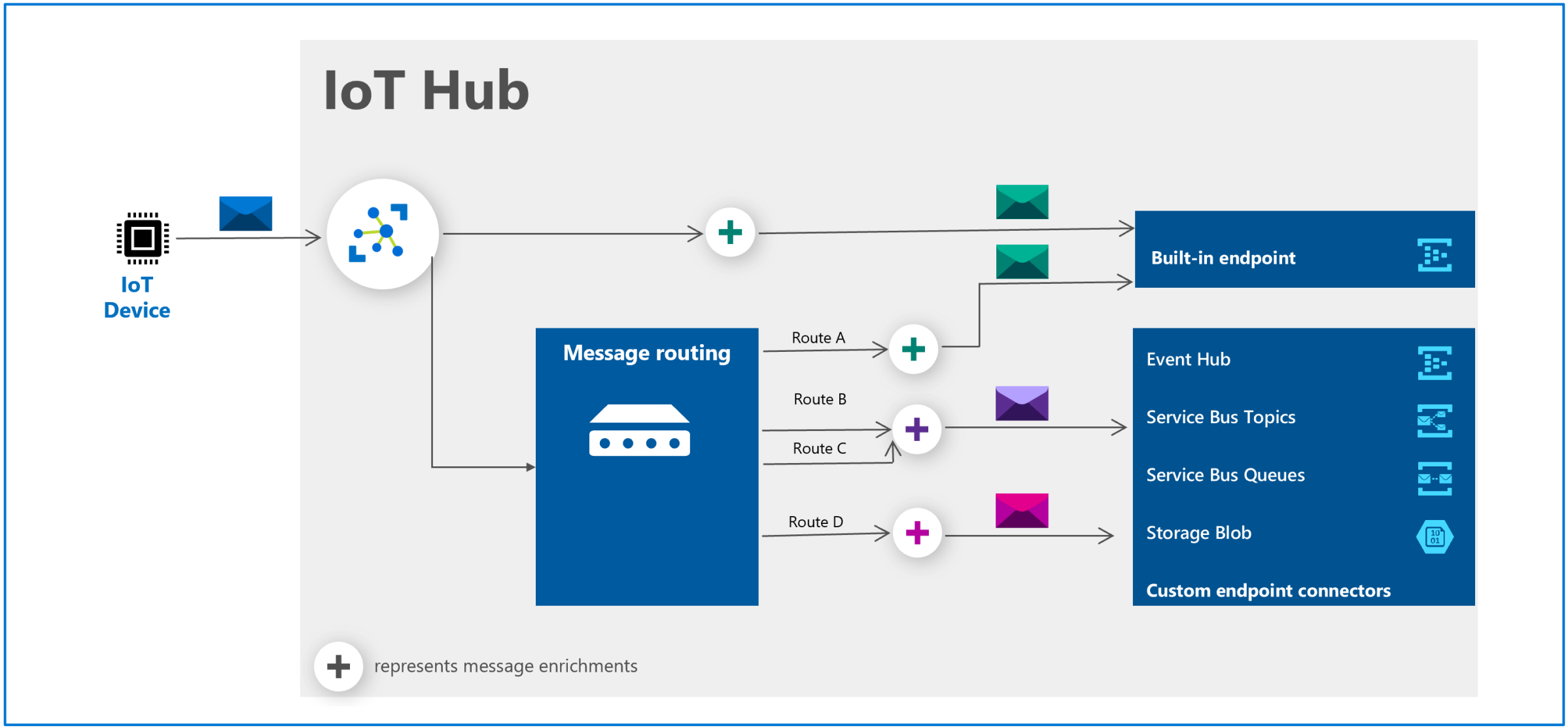
The IoT Hub integration with Event Grid supports a larger variety of endpoint types

Does it matter if your data arrives in order?

IoT Hub message routing maintains the order in which messages are sent

Event Grid does not guarantee that endpoints will receive events in the same order that they occurred

Message enrichments for D2C messages: Overview



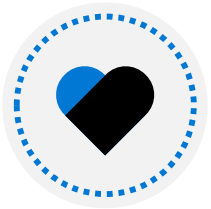
Message enrichments for D2C messages: Details

Three key elements:

Enrichment name or key

A value

One or more endpoints for which the enrichment should be applied



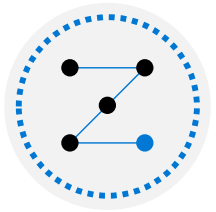
Value possibilities

Any static string

The name of the IoT hub sending the message, *$iothubname*.

Information from the device twin, such as its path, tags such as *$twin.tags.field,* or twin properties such as *$twin.properties.reported.fanspeed*

IoT Hub quotas and throttling



Operation throttles:

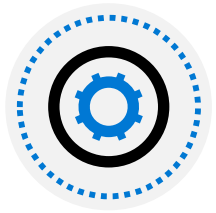
Throttling Details – tied to the number of units in the specific tier, based on number of 4 KB messages (0.5 KB for the Free tier)

Traffic shaping – IoT Hub will allow a certain amount of burst above the quota, then will queue, then finally will reject messages with a 429 error

Identity registry operations throttle – export and import are available for batch operations

Device connections throttle – 100 per second per unit for new connections
(existing connections don't count)

Other limits

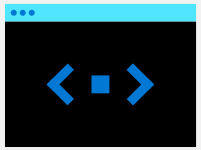


Latency:

Depends on network

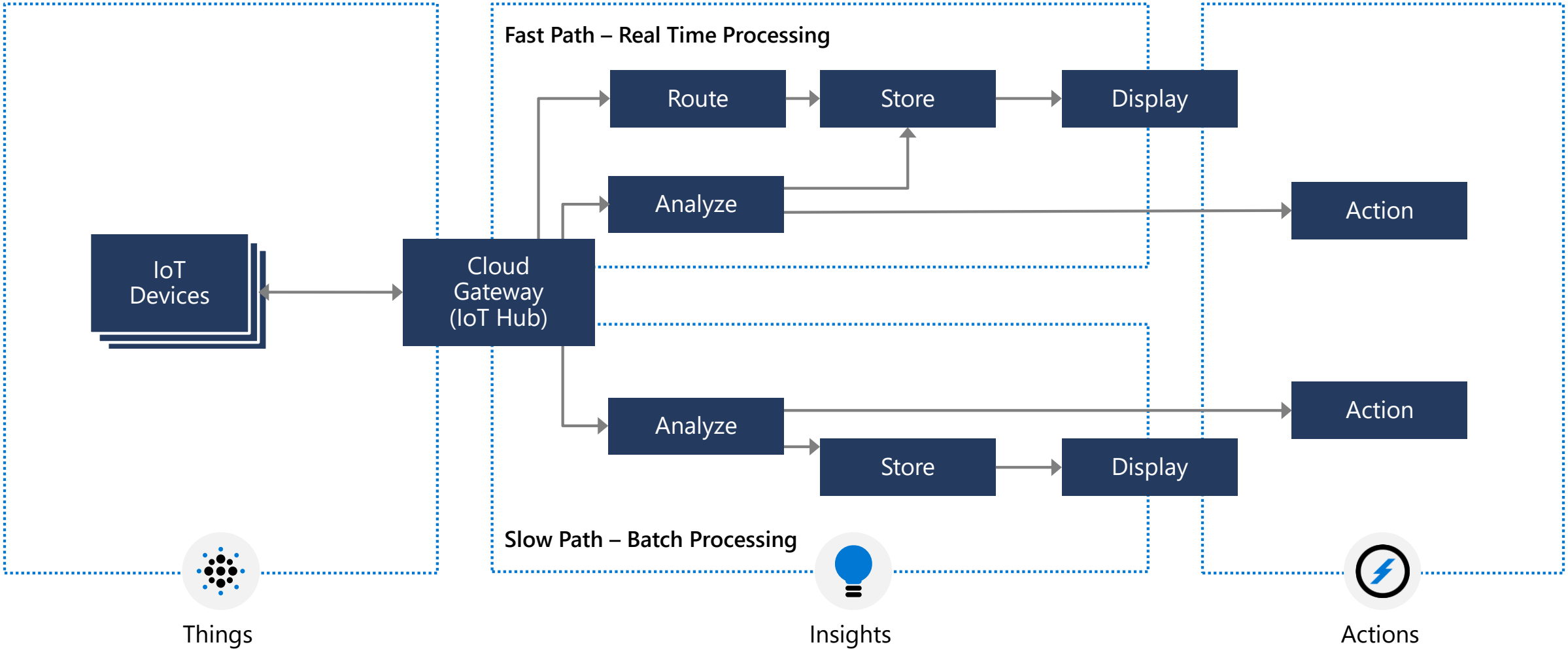
Impacted by throttling

Lesson 4: Data storage and the lambda architecture

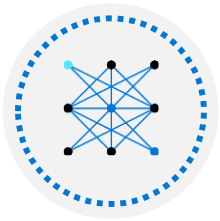


Introduction to lambda architecture

Recommended data flow – Lambda architecture



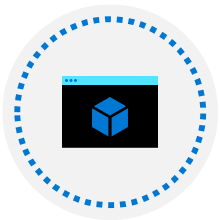
Common storage options for Azure IoT solutions



Azure Storage Accounts

Azure Data Lake Gen2

Azure Cosmos DB



Azure SQL Database

Azure storage accounts

Storage services:

Blob

Files

Queues

Tables

Types of storage accounts:

General-purpose v1 and v2, BlockBlobStorage, FileStorage, BlobStorage

Securing access to storage accounts:

Azure AD, Shared Key, Shared Access Signature, Anonymous

Azure Data Lake Gen 2

Designed for big data analytics:

Performance is optimized because it's built for analysis

Management is easier because data is organized in files and directories

Security is enforceable because you can define POSIX permissions on directories or individual files

Cost effective because Data Lake Storage Gen2 is built on top of the low-cost Azure Blob storage

Key features of Data Lake Storage Gen2:

Hadoop compatible access

A superset of POSIX permissions

Cost effective

Optimized ABFS driver

Azure Cosmos DB

Natively partitions data for 99.99% guarantees for availability, throughput, low latency, and consistency on all single-region accounts and all multi-region accounts with relaxed consistency, and 99.999% read availability on all multi-region database accounts

SSD backed storage with low-latency order-of-millisecond response

Five levels of selectable consistency

Flexible data-friendly pricing model that meters storage and throughput independently.

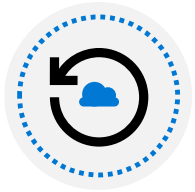
A reserved throughput model allows you to think in terms of number of reads/writes instead of underlying hardware.

Massive scalability on the order of trillions of requests per day

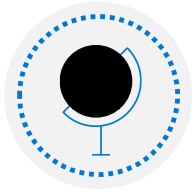
Azure SQL Database

Business continuity: Azure SQL Database will handle the disruptive events that might happen in the cloud environment and keep your applications and business processes running

High availability: 99.99% uptime



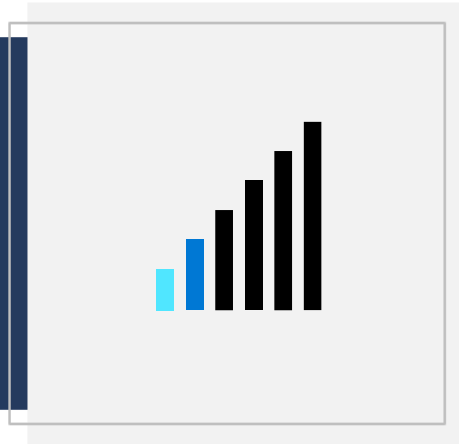
Automated backups: automatic backups that are kept between 7 and 35 days, and uses Azure read-access geo-redundant storage (RA-GRS) to ensure that they are preserved even if the data center is unavailable. *Long-term retention* also available



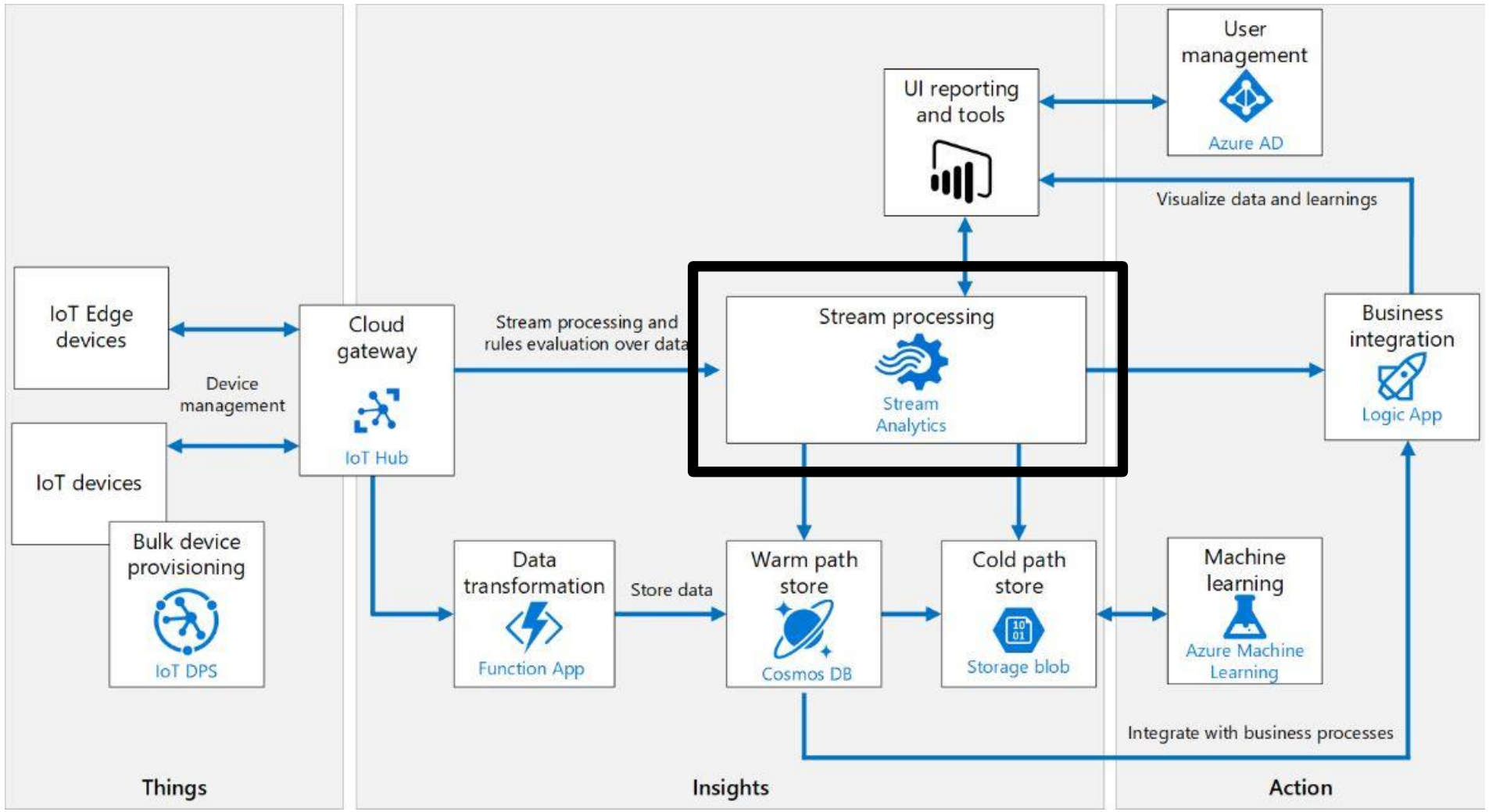
Geo-replication: Auto-failover groups across regions

Scale resources: Dynamically add more resources to your database with minimal downtime

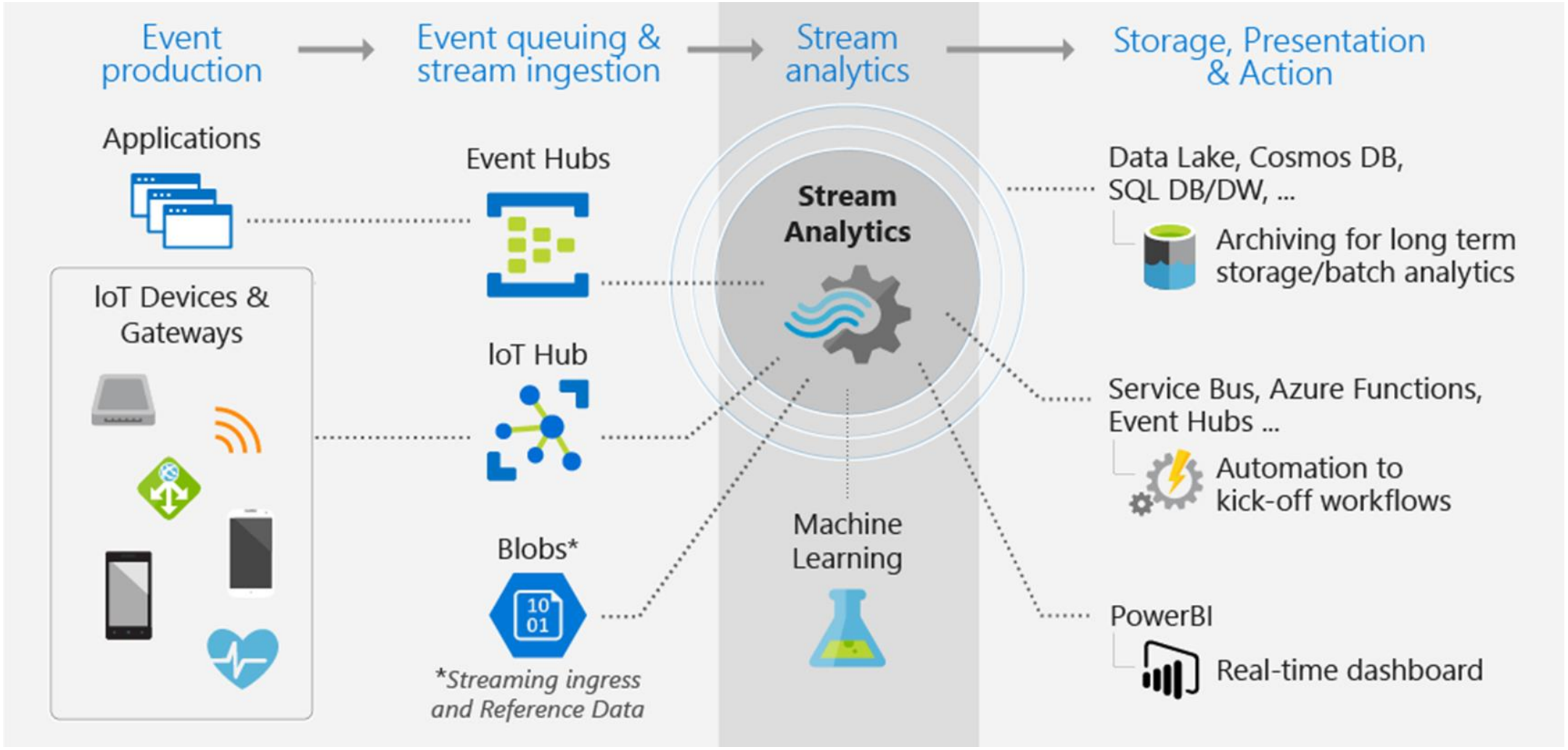
Lesson 05: Azure Functions and Stream Analytics



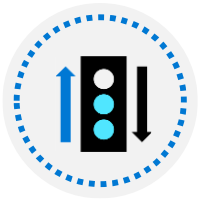
What is message processing?



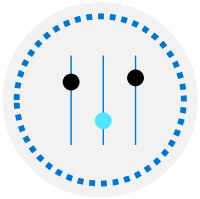
What is Azure Stream Analytics (ASA)?



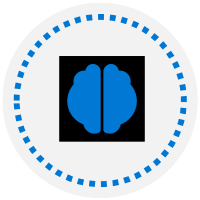
Azure Stream Analytics data flow



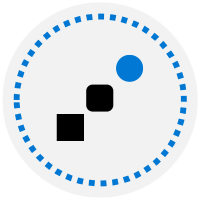
An Azure Stream Analytics *job* consists of an *input*, *query*, and an *output*



Input – ASA can ingest data from Azure Event Hubs, Azure IoT Hub, or Azure Blob Storage. ASA can parse CSV, JSON, and Avro

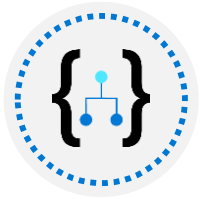


Query – ASA uses a SQL-like query language that includes support for filtering, sorting, aggregating, joining, and user-defined functions



Output – ASA can output to many targets

ASA and other stream processing technologies



When to use Azure Stream Analytics:

Recommended tool for analytics on Azure

Dashboards for data visualization

Real-time alerts from temporal and spatial patterns or anomalies

Basic Extract, Transform, and Load (ETL) work

When to use other technologies:

Apache Kafka connectivity (Azure Event Hubs are a good choice here)

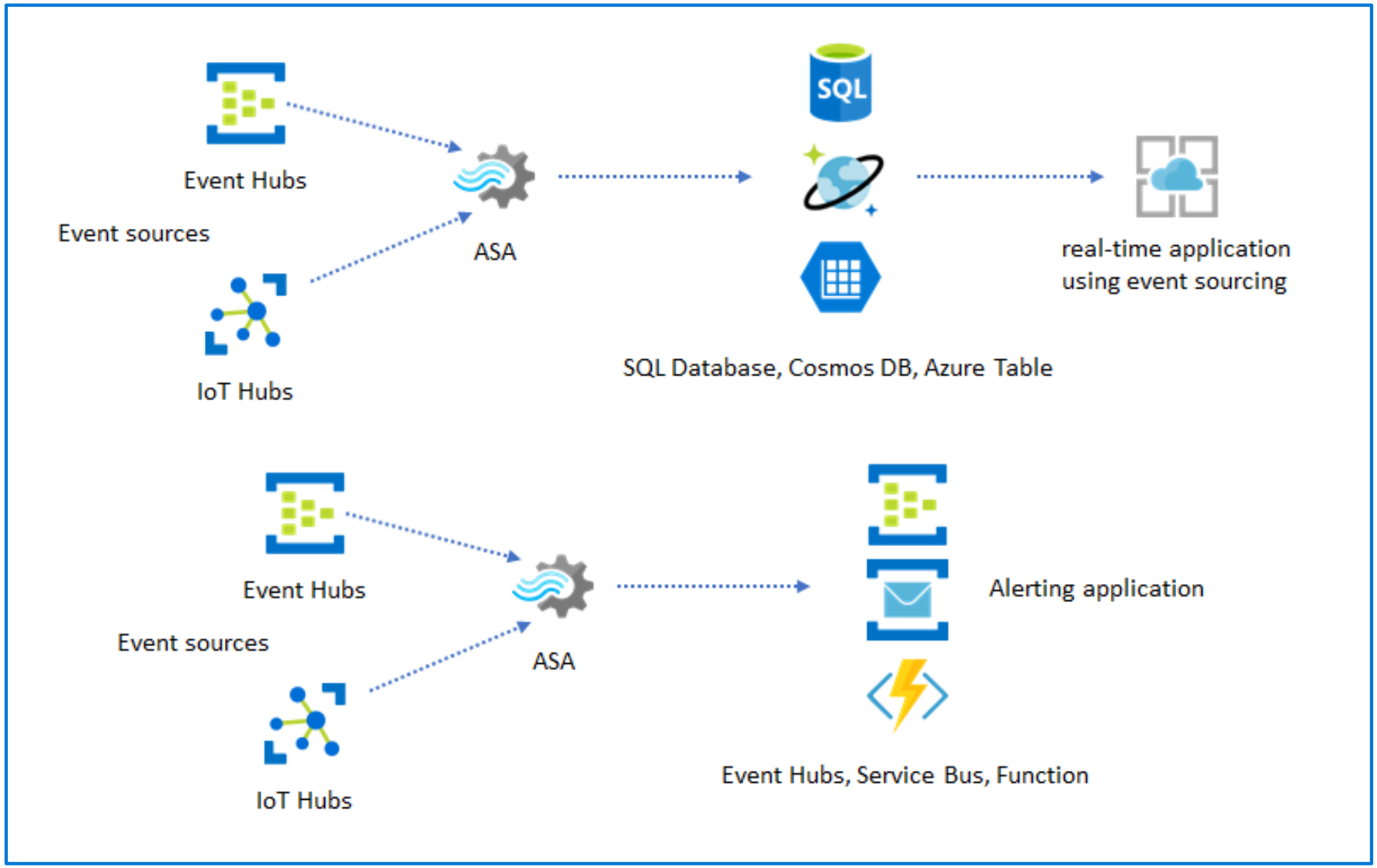
In-line query handling in a language besides C# or JavaScript (e. g. Java; Spark Structured Streaming or specialized Azure Event Hubs implementation are good choices here)

Multi-cloud support (Azure Stream Analytics is Azure-specific; Spark Structured Streaming, Storm, etc. will work here)

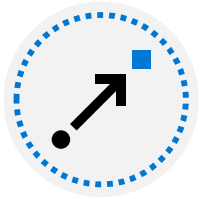
Some common ASA patterns and tools

Real-time insights through data stores

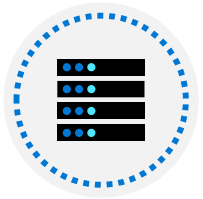
Real-time insights with event messaging



ASA input types



Data stream input – an unbounded sequence of events over time



Reference data input – static (or slowly changing) data used for lookups and correlation

ASA streaming data input

Stream data from Event Hubs

```
SELECT
 EventProcessedUtcTime,
 EventEnqueuedUtcTime,
 PartitionId
FROM Input
```

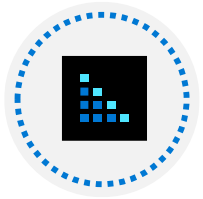
Stream data from IoT Hub

```
SELECT *
FROM Input
HAVING Temperature > 27
```

Stream data from Blob storage

```
SELECT
 BlobName,
 EventProcessedUtcTime,
 BlobLastModifiedUtcTime
FROM Input
```

ASA reference data input



Azure Blob storage:

Can be static or scheduled

Static reference data – immutable, with no changes over time

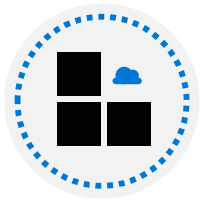
Generate reference data on a schedule – configured to reference a date and time in the blob path (can be from Azure Data Factory!)

Azure SQL Database:

SQL Query used as a source

Stored in memory and in a storage account (configured as part of setup)

Refreshed on a regular (configurable) basis



Size limitation:

Based on number of units configured for the IoT Hub – approximately 50 MB per unit

ASA query sample: Convert data types pattern

Query Solution

```
SELECT
  Make,
  SUM(CAST(Weight AS BIGINT)) AS Weight
FROM
  Input TIMESTAMP BY Time
GROUP BY
  Make,
  TumblingWindow(Second, 10)
```

Input

Make	Time	Weight
Honda	2015-01-01T00:00:01.0000000Z	"1000"
Honda	2015-01-01T00:00:02.0000000Z	"2000"

Output

Make	Weight
Honda	3000

ASA Query Sample: Multiple Outputs

Input

Make	Time
Honda	2015-01-01T00:00:01.0000000Z
Honda	2015-01-01T00:00:02.0000000Z
Toyota	2015-01-01T00:00:01.0000000Z
Toyota	2015-01-01T00:00:02.0000000Z
Toyota	2015-01-01T00:00:03.0000000Z

Query Solution

```
SELECT
  *
INTO
  ArchiveOutput
FROM
  Input TIMESTAMP BY Time
SELECT
  Make,
  System.TimeStamp() AS
AsaTime,
  Count(*) AS [Count]
INTO
  AlertOutput
FROM
  Input TIMESTAMP BY Time
GROUP BY
  Make,
  TumblingWindow(Second, 10)
HAVING
  [Count] >= 3
```

Output 1

Make	Time
Honda	2015-01-01T00:00:01.0000000Z
Honda	2015-01-01T00:00:02.0000000Z
Toyota	2015-01-01T00:00:01.0000000Z
Toyota	2015-01-01T00:00:02.0000000Z
Toyota	2015-01-01T00:00:03.0000000Z

Output 2

Make	Time	Count
Toyota	2015-01-01T00:00:10.0000000Z	3

ASA queries – Parse complex data types

Record data type example:

```
{
    "DeviceId" : "12345",
    "Location" :
    {
        "Lat": 47,
        "Long": 122
    },
    "SensorReadings" :
    {
        "Temperature" : 80,
        "Humidity" : 70,
        "CustomSensor01" : 5,
        "CustomSensor02" : 99,
        "SensorMetadata" :
        {
        "Manufacturer":"ABC",
        "Version":"1.2.45"
        }
    }
}
```

Query examples:

```
SELECT
    DeviceID,
    Location.Lat,
    Location.Long,
    SensorReadings.SensorMetadata.Version
FROM input
```

```
SELECT input.Location.*
FROM input
```

ASA queries: Time handling terms

Background time concepts:

Event time – The time the original event happened, such as a car arrives at a toll booth

Processing time – When the event is observed, such as when the computer system sees the sensor data indicating the car's arrival

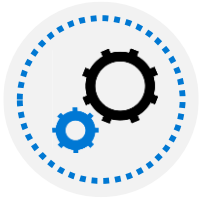
Watermark – A time marker indicating up to what point events have been processed

ASA lets you choose what the applicable time for an event is:

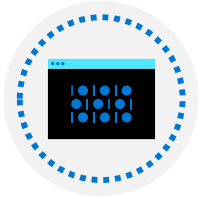
Arrival time – When the event reached the input event source, after the processing time; best for when the exact time of arrival isn't important

Application time (or *event time*) – When the event was originally generated

ASA queries: How time progresses in ASA



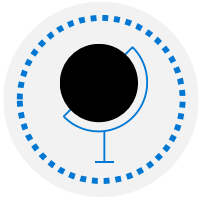
Stream processing requires relevant watermarks for data processing to be meaningful



ASA generates watermarks with a specific algorithm when you've selected event time (application time):

When there's an incoming event, the watermark is the largest selected time value seen so far, minus any configured out-of-order tolerance

When there's no incoming event, the watermark is an estimated time based on the last time seen so far, plus time progression in the ASA environment, minus any out-of-order tolerance



Watermarks in most systems and by default in ASA are partition-level (effectively global), but can be per-device through *substreams*

ASA queries: Late and early arrivals

Late arriving events **– Arrive with a time "sooner" than the watermark, meaning outside of configured tolerance:**

Can be dropped

Can be forced to the watermark time

Early arriving events **– Arrive with an *input* arrival time "sooner" than the *output* start time as specified in the query:**

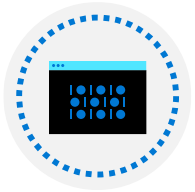
Corrects the inherent discrepancy in the concept of output time (as specified in the query) and the concept of input arrival time

Fixed tolerance of five (5) minutes, meaning the system looks at arrival times five minutes back from the requested start time to ensure no events are "lost" in that window

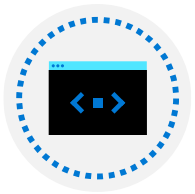
Allows repeatable output no matter where you start the query from

ASA queries: Output windows

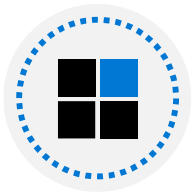
Temporal windows are very common



ASA has native windowing functions



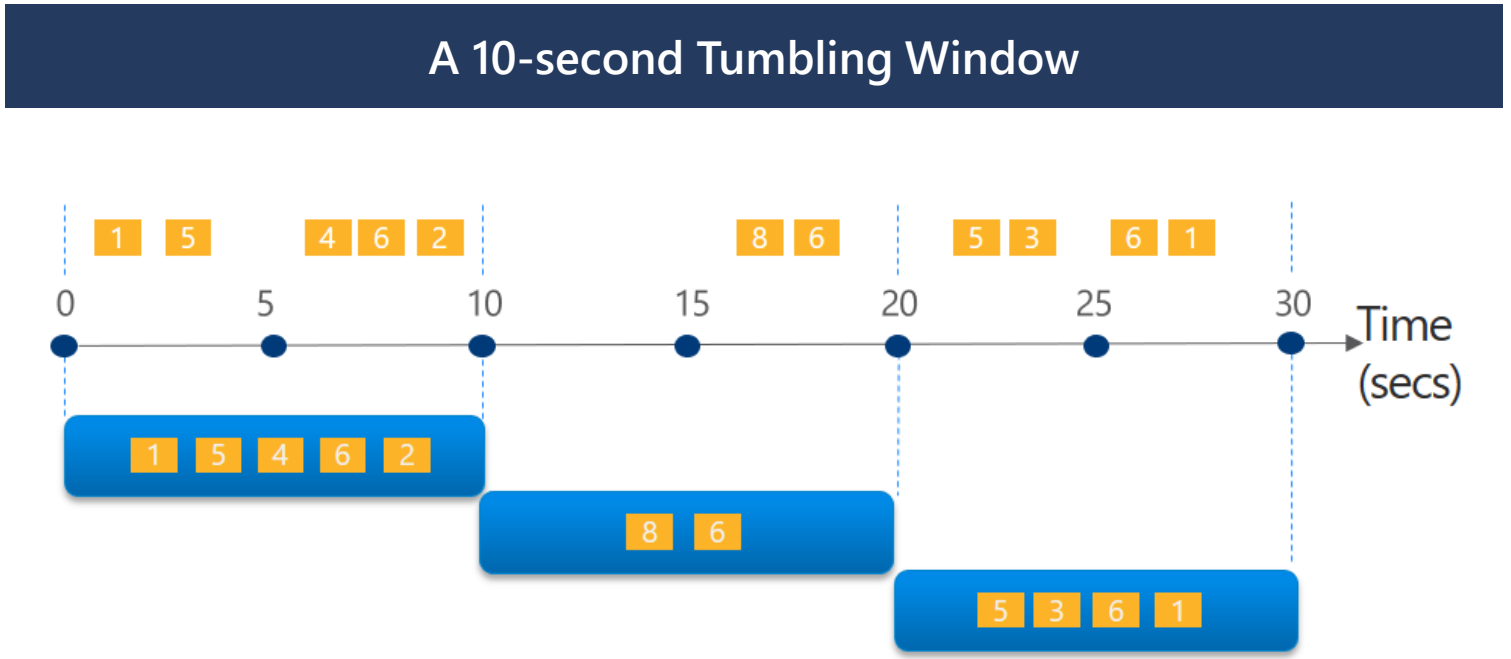
Windows are always collected at the end of the window, to a single event, based on the aggregate function used



There are four window types, covered on the next slides…

ASA queries: Tumbling window

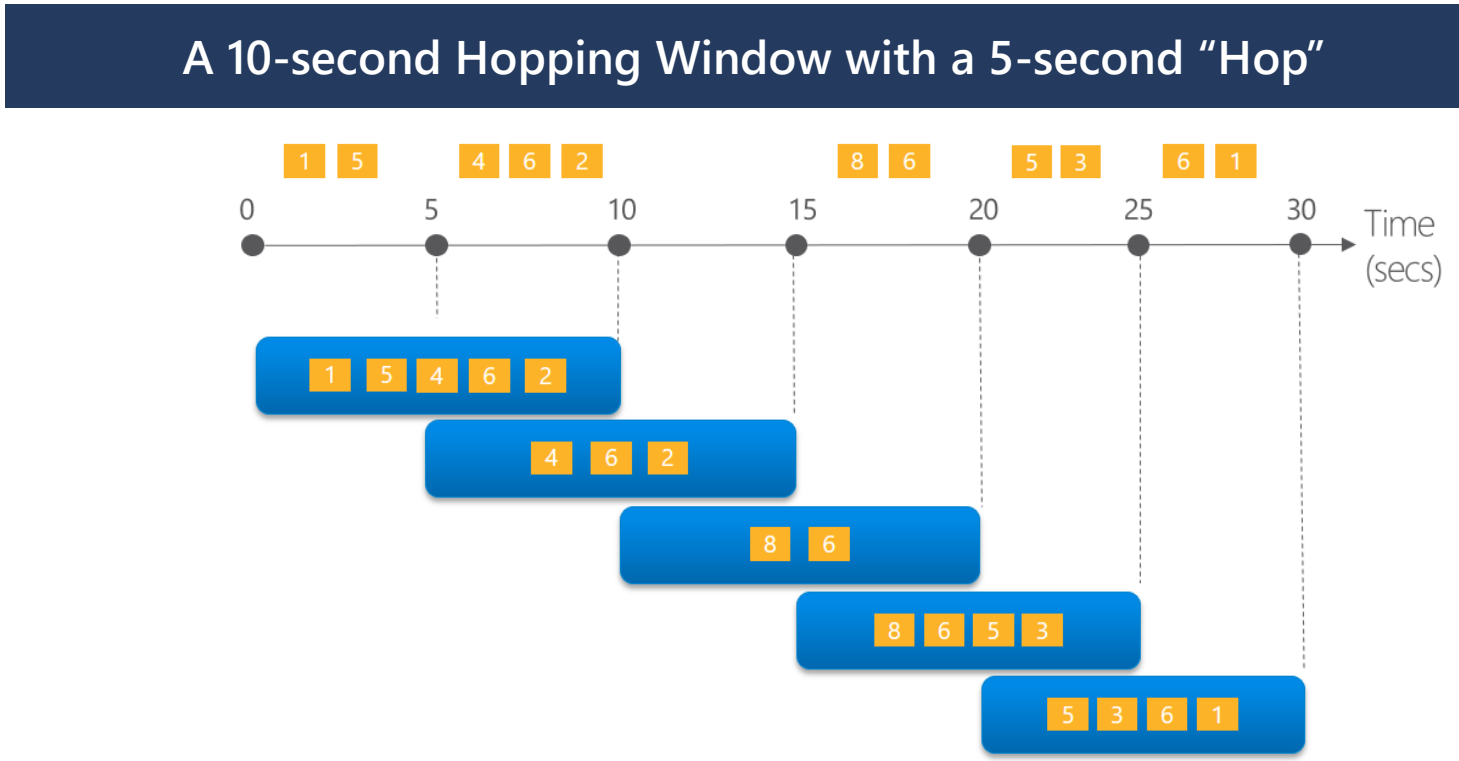
Tell me the count of tweets per time zone every 10 seconds




```
SELECT TimeZone, COUNT(*) AS Count
FROM TwitterStream TIMESTAMP BY CreatedAt
GROUP BY TimeZone, TumblingWindow(second,10)
```

ASA queries: Hopping window

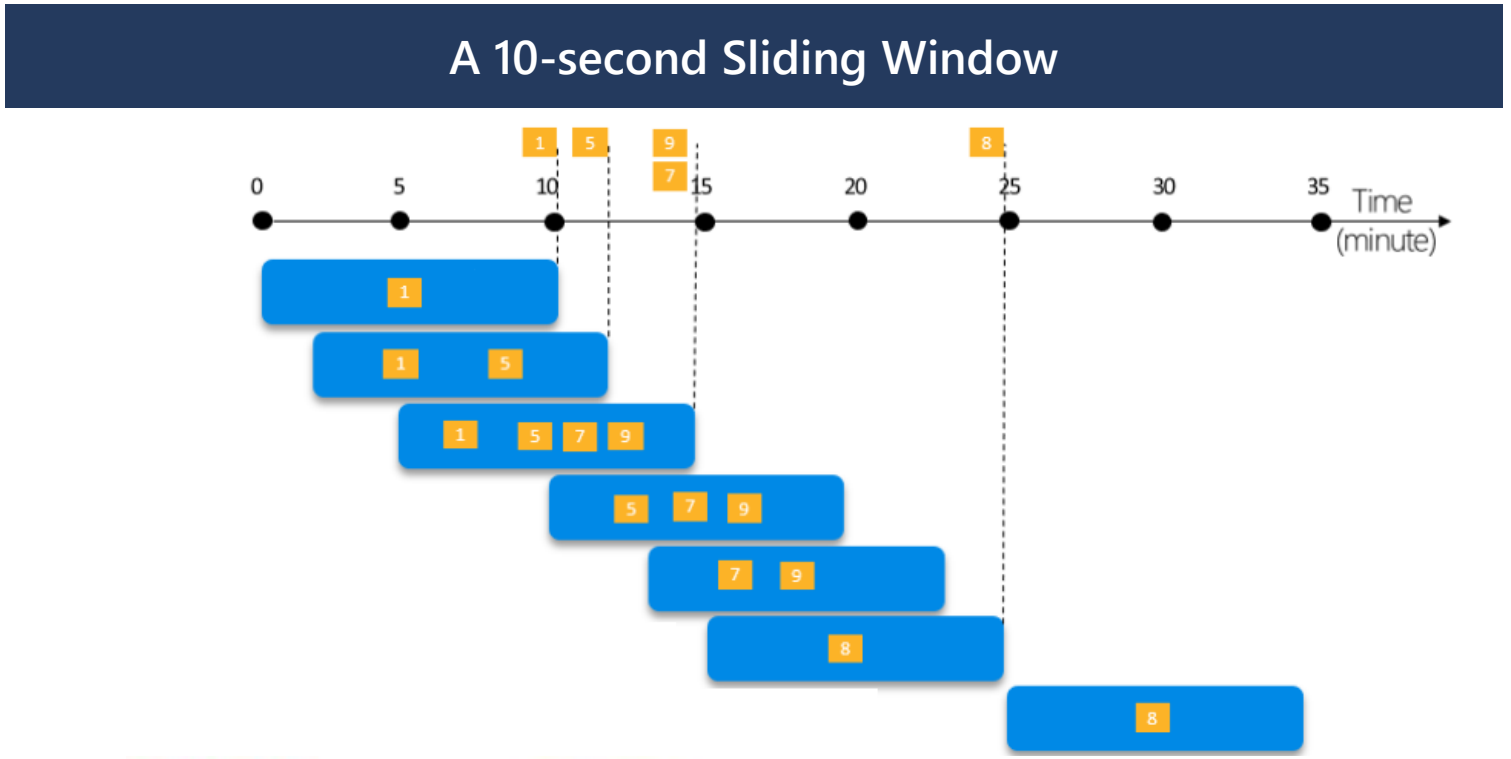
Every 5 seconds give me the count of tweets over the last 10 seconds




```
SELECT Topic, COUNT(*) AS TotalTweets
FROM TwitterStream TIMESTAMP BY CreatedAt
GROUP BY Topic, HoppingWindow(second, 10 , 5)
```

ASA queries: Sliding window

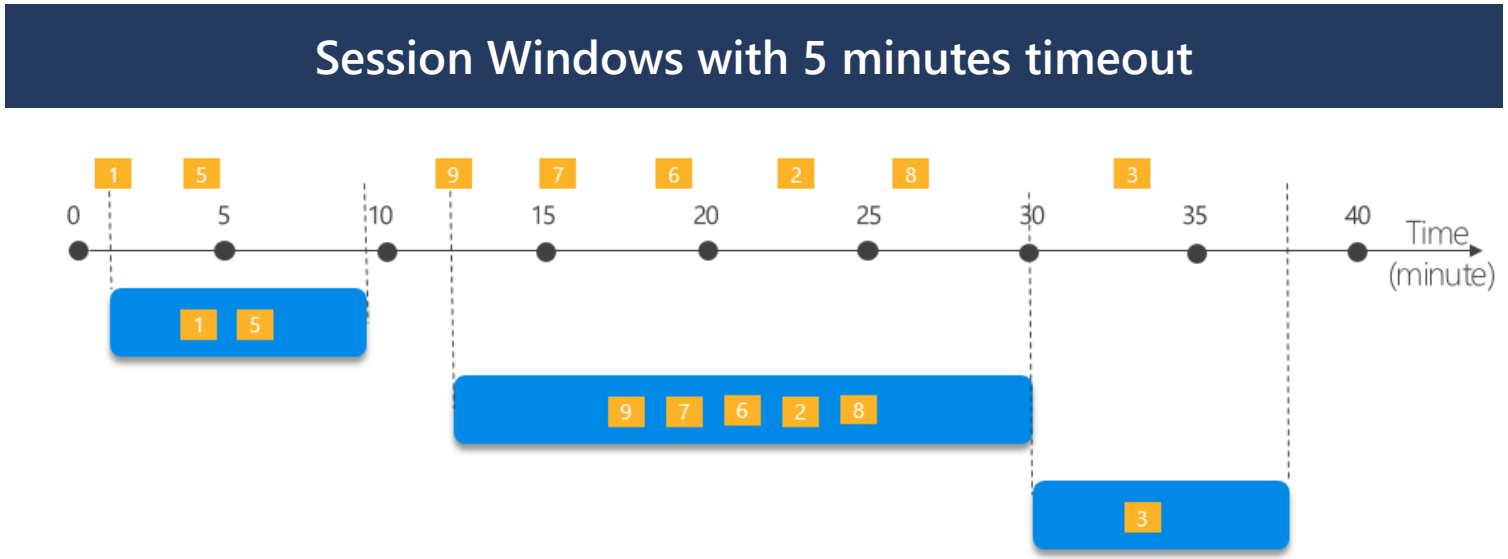
Give me the count of tweets for all topics which are tweeted more than 10 times in the last 10 seconds




```
SELECT Topic, COUNT(*)
FROM TwitterStream TIMESTAMP BY CreatedAt
GROUP BY Topic, SlidingWindow(second, 10)
HAVING COUNT(*) > 10
```

ASA queries: Session window

Tell me the count of tweets that occur within 5 minutes to each other

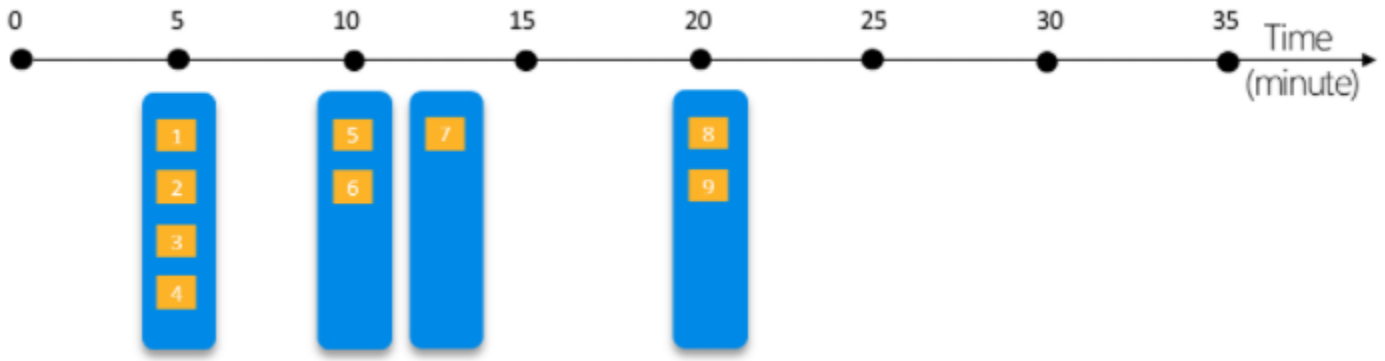



```
SELECT Topic, COUNT(*)
FROM TwitterStream TIMESTAMP BY CreatedAt
GROUP BY Topic, SessionWindow(minute, 5, 10)
```

ASA queries: Snapshot window

Give me the count of tweets with the same topic type that occur at exactly the same time

Snapshot Windows group events with the same timestamp




```
SELECT Topic, COUNT(*)
FROM TwitterStream TIMESTAMP BY CreatedAt
GROUP BY Topic, System.Timestamp()
```

ASA output options


Azure Data Lake Storage Gen 1 – Original Azure “big data” processing storage target


SQL Database – (As previously discussed)


Blob Storage (with Azure Data Lake Storage Gen 2) – (As previously discussed)


Event Hubs – (As previously discussed with more detail coming)


Power BI – Desktop visualization, as will be seen in the lab

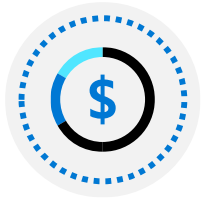

Table storage – (As previously discussed)


Service Bus queues or topics – One-to-one or one-to-many messaging

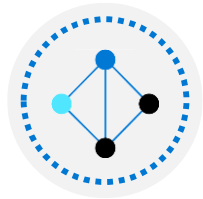

Azure Cosmos DB – (As previously discussed)

Azure Functions: Features

Choice of language

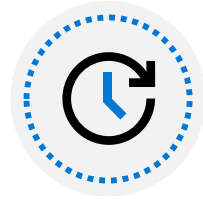


Pay-per-use pricing option



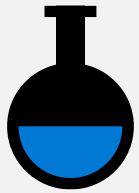
Easy integration with other Azure services and with Oauth providers

Flexible development – Portal, CI/CD, etc.



Open source, cross-platform runtime

Lesson 6: Module 4 labs



Module 4 labs

Lab 7: Device Message Routing:

You will write code to generate vibration telemetry

You will create an IoT Hub message route to Azure Blob storage

You will create a Stream Analytics job that outputs logging messages to Blob storage

Lesson 7: Module 4 review questions

Module review: Question 4.1

Within an IoT solution, under what conditions is the fallback route used?

Answer A:
When multiple routes would have directed a message to the same custom endpoint, the fallback route is used specify which route delivers the message.

Answer B:
When multiple routes would have directed a message to multiple endpoints, the fallback route is used specify which endpoint the message should be delivered to.

Answer C:
When multiple routes are defined, but a message doesn't match any query conditions, the fallback route sends the message to the built-in endpoint.

Module review: Question 4.2

An Azure IoT solution uses a common message format.

Which of the following statements about the common message format is accurate?

Answer A:
The common message format enables IoT hub to interpret device-to-cloud messages that arrive unformatted.

Answer B:
Versions of the common message format are optimized for each of the messaging protocols supported by IoT hub.

Answer C:
The common message format is used across protocols for both device-to-cloud and cloud-to-device messages.

Module review: Question 4.3

What message size is used to calculate IoT hub's daily message quota for the Basic and Standard tiers?

Answer A:
4 KB

Answer B:
0.5 KB

Answer C:
256 KB

Module review: Question 4.4

Within an IoT solution, what is message enrichment?

Answer A:
An IoT hub feature that enables users to direct device messages to a service endpoint.

Answer B:
A real-time analytics and complex event-processing engine that accepts data from multiple sources simultaneously.

Answer C:
The ability of IoT hub to stamp messages with additional information before the message is sent a designated endpoint.

Module review: Question 4.5

What are the names of the two storage paths in an IoT lambda architecture?

Answer A:
Hot and cold

Answer B:
High and low

Answer C:
Large and small

Module review: Question 4.6

A developer needs to store time series data on a hot storage path. High availability and performance are critical, but they need the solution to be cost effective as well. The company must be able to use ASA to query the stored data as unstructured JSON.

Which storage option should the developer select?

Answer A:
Azure Blob Storage

Answer B:
Azure Data Lake Gen 2

Answer C:
Azure Cosmos DB

Module review: Question 4.7

A developer needs to set up an Azure Stream Analytics Windowing function that will output results every 10 seconds. The results should be based on data from previous 30 seconds.

Which one of the Windowing function types should the developer use to achieve the desired results?

Answer A:
Hopping window.

Answer B:
Tumbling window.

Answer C:
Sliding window.

Module review: Question 4.8

A developer is setting up their first Azure Stream Analytics job.

What are the components of an Azure Stream Analytics job?

Answer A:
FROM, WITH, and INTO

Answer B:
Input, Action, and Output

Answer C:
Input, Query, and Output