

plain concepts



ABOUT US

 XAMARIN
PREMIUM
CONSULTING PARTNER

13 ★★★★★★★★★★
MICROSOFT
MOST VALUABLE
PROFESSIONAL

 **PREMIER**
DEVELOPER
PARTNER
LIVE Apps

ALM
PARTNER OF THE YEAR
FOR 7 CONSECUTIVE YEARS

AGILE
ALLIANCE
CORPORATE MEMBER

MICROSOFT
GOLD CLOUD
PLATFORM

MICROSOFT
GOLD APP
DEVELOPMENT

MICROSOFT
GOLD LIFECYCLE
MANAGEMENT
APPLICATION

MICROSOFT
SILVER APP
INTEGRATION

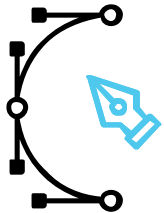
MICROSOFT **SILVER**
COLLABORATION
AND CONTENT

WINDOWS 8
APPLICATIONS
PARTNER OF THE YEAR

MICROSOFT
BEST CLOUD
APPLICATIONS **2016**

OUR SERVICES

UI/UX Design



Web & App
development



Demos &
Whitepapers



Marketing
Campaigns



Custom CMS



PROCESADO DE STREAMS

SOBRE HADOOP Y AZURE STREAM ANALYTICS

Francisco Martínez

Data Engineer at Plain Concepts

fmartinez@plainconcepts.com

@pacommiranda

LEARNING PATH

- Learning Path
 - Dos jornadas presenciales
 - Tres sesiones on-line



Sesiones Presenciales

12/19 Abril – Implementación de Hadoop en Azure. Despliegue y administración

13/20 Abril – Implementación de Hadoop en Azure. Desarrollo

Sesiones On-Line

28 Abril - Procesado de Streams sobre Hadoop y Azure Stream Analytics.

<https://goo.gl/mjBHRp>

5 Mayo - Machine Learning sobre Hadoop y Azure ML.

<https://goo.gl/IOQtIC>

12 Mayo - Visualización en Hadoop IaaS y Power BI.

<https://goo.gl/dOOUAi>

plain concepts

EN EL CAPÍTULO ANTERIOR



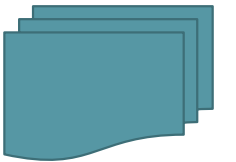
AGENDA

- Introducción al procesamiento de streams
- ¿Qué es Apache Storm?
 - Conceptos básicos de Apache Storm
 - Demo
- ¿Qué es Azure Stream Analytics?
 - Conceptos básicos de Azure Stream Analytics
 - Demo

INTRODUCCION

TRABAJANDO CON DATOS –TRANSFER

- Permiten obtener y proporcionar datos a herramientas de procesamiento o entre diferentes sistemas.
 - Sistemas de colas
 - Kafka, Rabbit, Message Bus, Flume
 - Hub
 - Event Hub, IoT Hub
 - Legacy
 - Online services
 - File Transfer



TRABAJANDO CON DATOS –STORAGE

- Nos permiten almacenar la información entre diferentes estados de procesamiento
 - NoSQL
 - HIVE, HBASE
 - MongoDB / Redis / CouchDB ..
 - SQL
 - SQL Server / PostgreSQL / Oracle ..
 - Files
 - HDFS



TRABAJANDO CON DATOS –PROCESSING

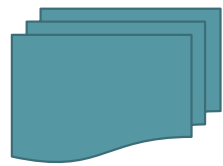
- Realizamos diversos cálculos que intentan extraer información, nueva inteligencia de negocio o simplemente nuevas visualizaciones de los mismos.
 - Hadoop
 - Tez / MR / para realizar procesos sobre grandes cantidades de datos
 - Spark
 - Opciones de visualización sobre un conjunto de datos, aprendizaje automático o procesamiento son algunas de las opciones de Spark.
 - Spark SQL / Mlib / GraphX.
 - Otros



ANALITICA TRADICIONAL

- Data At Rest
- Tenemos gran cantidad de productores de datos: sensores, dispositivos, aplicaciones...
- En un escenario BI tradicional, primero almacenamos los datos y despues los analizamos
- Esto no es suficiente para adaptarse a los escenarios emergentes
 - Redes sociales
 - IoT, Internet of Things
- Los datos offline no son suficientes

¿QUE NOS FALTA?



STREAMING

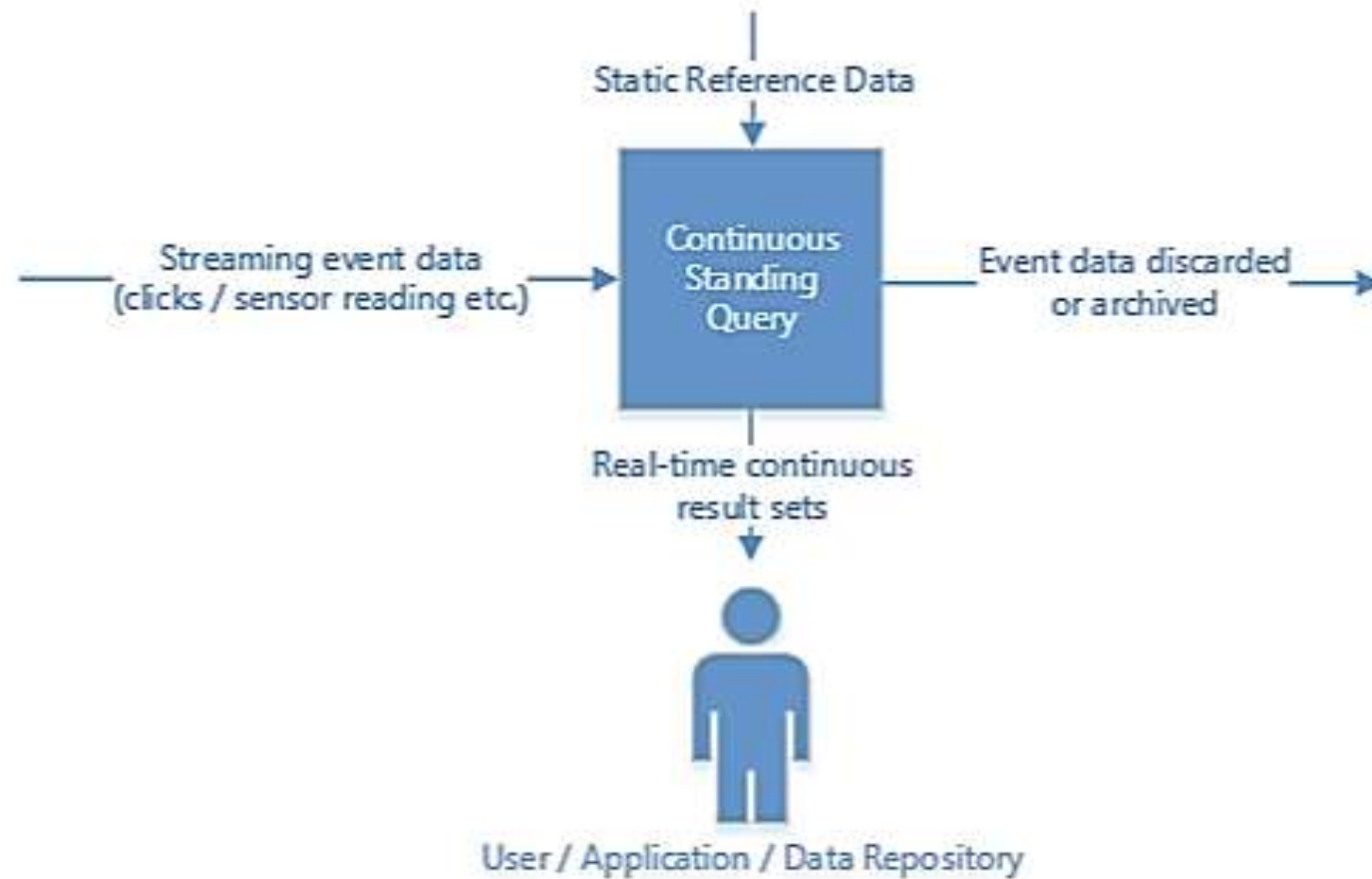
ANALITICA EN EL MUNDO MODERNO

- Data In Motion
- Trabajamos con datos en streaming
- Queremos monitorizar y analizar los datos en tiempo (casi) real
- No tenemos tiempo para recibir datos, almacenarlos y procesarlos antes del analisis
 - Necesitamos trabajar con streams

STREAM PROCESSING

- Stream vs Batch
 - Procesamos los datos según van entrando, no necesitamos hacer batch de grandes cantidades.
- Procesamiento por puntos de tiempo
 - Thresholds
 - Sensores
- Procesamiento por ventanas de tiempo
 - Trending
 - Alarmas
 - Agregados

STREAM PROCESSING





¿QUE ES APACHE STORM?

APACHE STORM

- Sistema distribuido de procesamiento de eventos
- Disponible en HDInsight
 - Clusters en Windows o Linux
- <http://storm.apache.org>

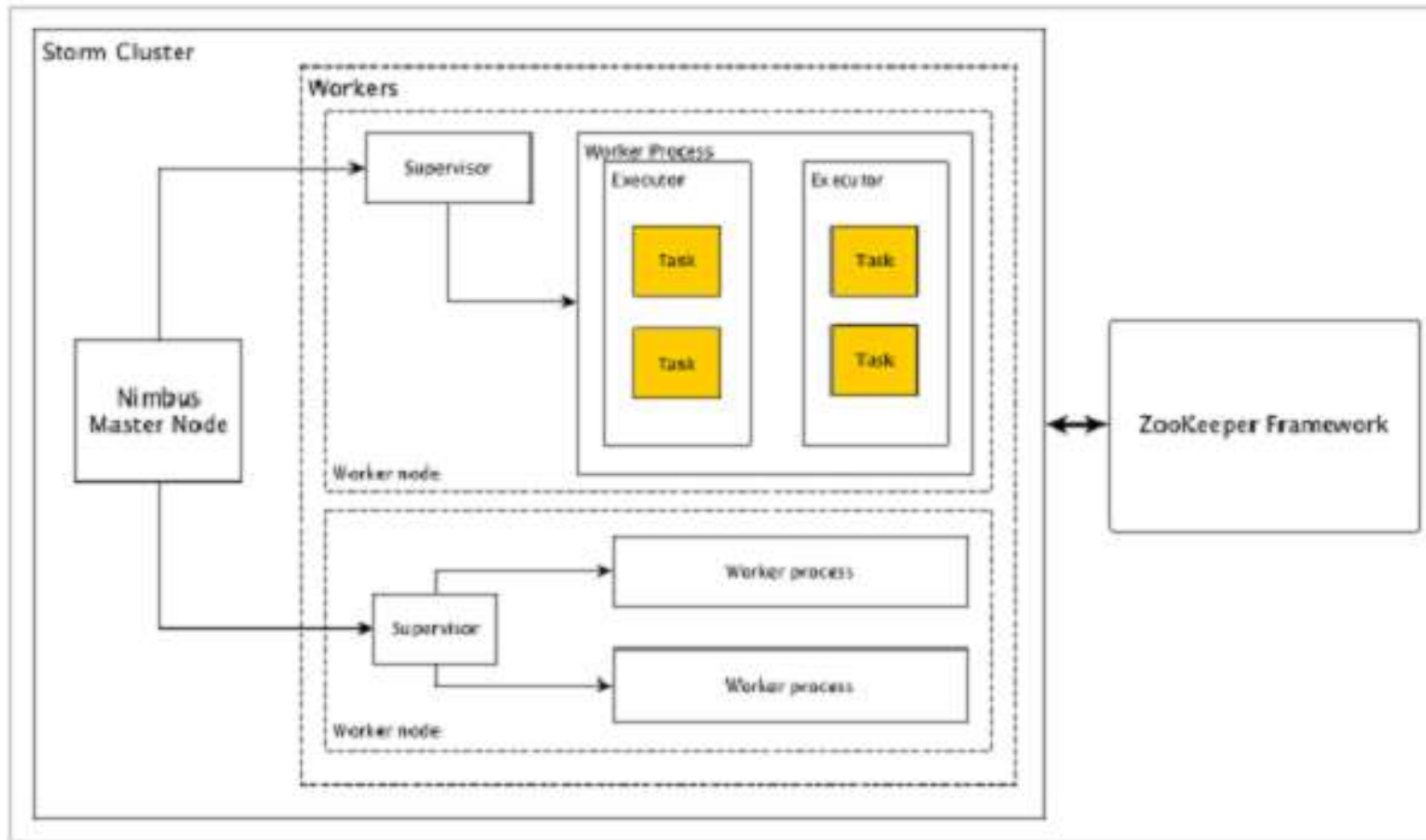
APACHE STORM

- Zookeeper
 - Como en cualquier otro clúster, para coordinación de los diferentes nodos
 - Ya existe en Hadoop
 - Si queremos failover debemos disponer de varios nodos de Zookeeper
- Nimbus
 - Master Node que se encarga de ejecutar nuestras topologías
 - Analiza y divide en tareas
 - Asigna tareas a “Supervisor(s)”

APACHE STORM

- Supervisor
 - Sigue las instrucciones del Nimbus
 - Dispone de múltiples “Worker Process”
 - Gobierna cada “Worker Process”
- Worker Process
 - Ejecuta tareas de una topología dentro de hilos llamados “Executors”
 - Un “Worker Process” tiene potencialmente muchos “Executors”

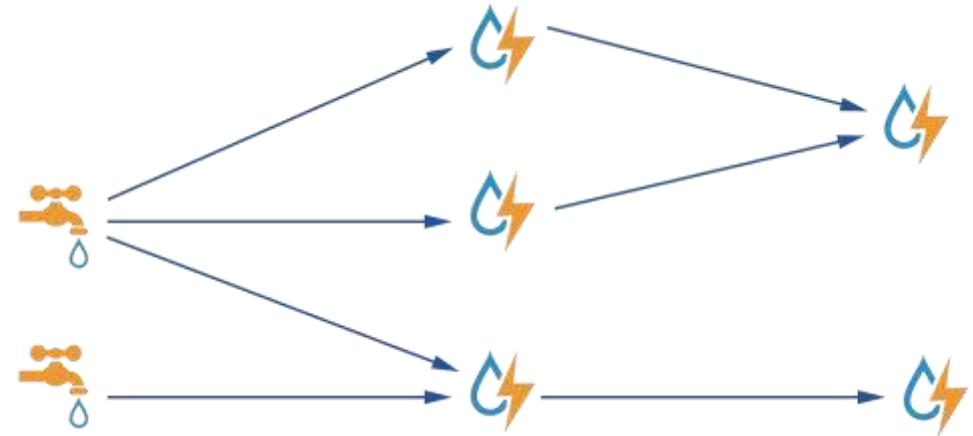
APACHE STORM



CONCEPTOS BÁSICOS DE APACHE STORM

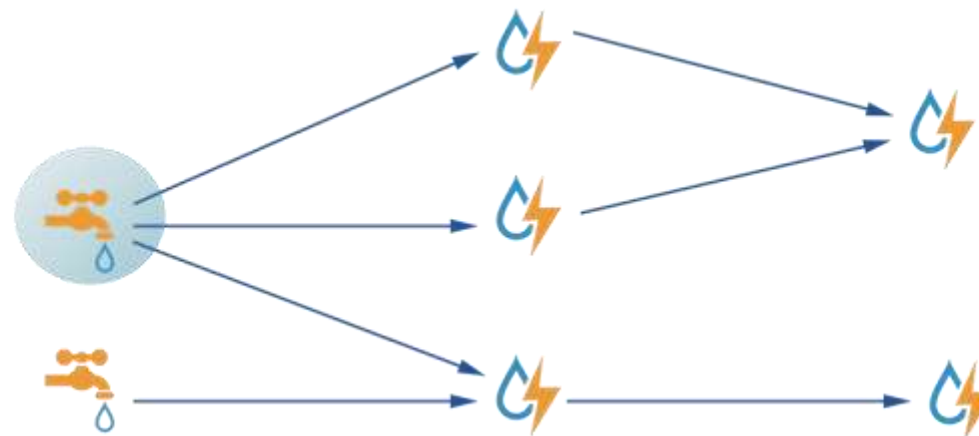
CONCEPTOS BASICOS - TOPOLOGIA

- Un workflow que marca el procesamiento de nuestros Streams
- Básicamente, una topología es una grafo dirigido donde los vértices son elementos de computación y las aristas son los stream de datos



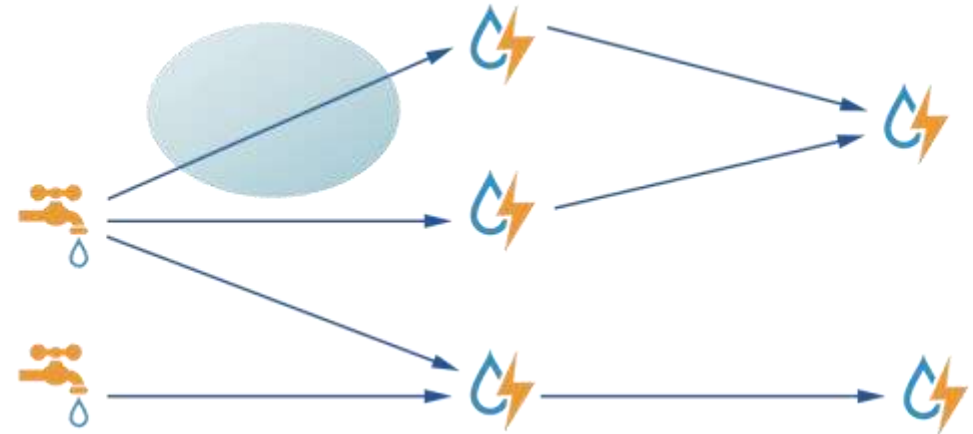
CONCEPTOS BASICOS - SPOUT

- El origen de los datos
- ISpout
 - Kafka
 - Kestrel
 - EventHub
 - ...
- Proporciona **tuplas** a la topología
 - Las tuplas son la estructura de datos básica.
 - Una lista de elementos ordenados



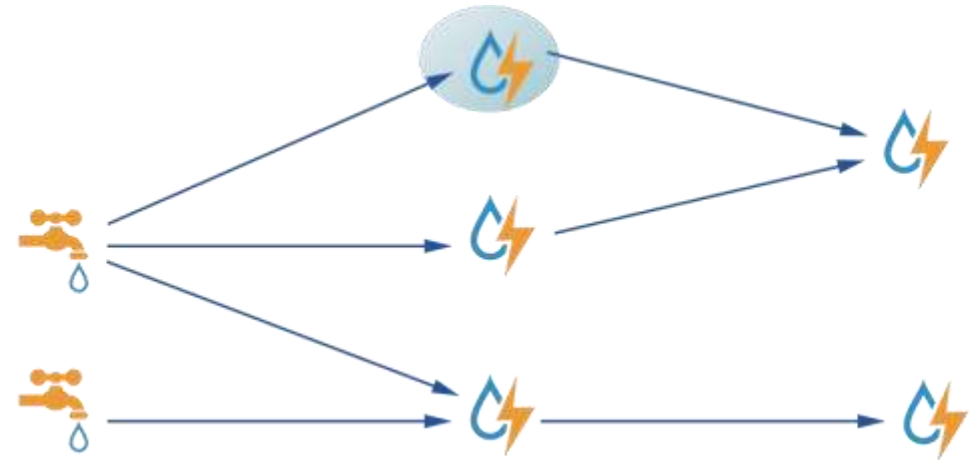
CONCEPTOS BASICOS - STREAM

- Una secuencia no ordenada de tuplas que salen de cada vértice
- Pueden estar en diferentes formatos
 - Json es habitual



CONCEPTOS BASICOS - BOLTS

- Unidades de procesamiento
- Pueden hacer operaciones simples o más complejas
 - Filtros, agregaciones, uniones..
 - Interactuar con datos referenciales como ficheros, bases de datos etc.
- IBolt
 - Lo habitual es desarrollar estas piezas aunque hay elementos reusables

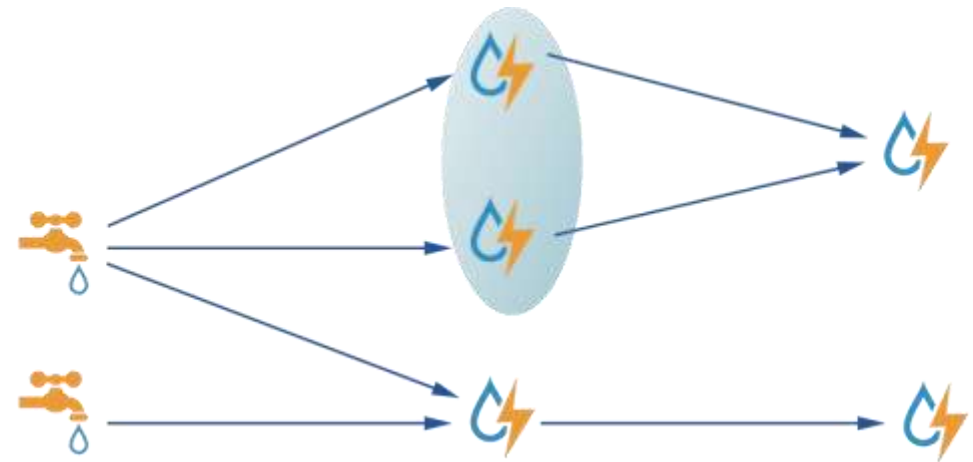


CONCEPTOS BASICOS – STREAM GROUPING

- Necesitamos estructurar la topología
- ¿Que streams recibe cada bolt?
- Podemos agrupar estas tuplas, según lo necesitemos, de varias formas diferentes:
 - Shuffle Grouping
 - Field Grouping
 - Global Grouping
 - All Grouping
 - Otros

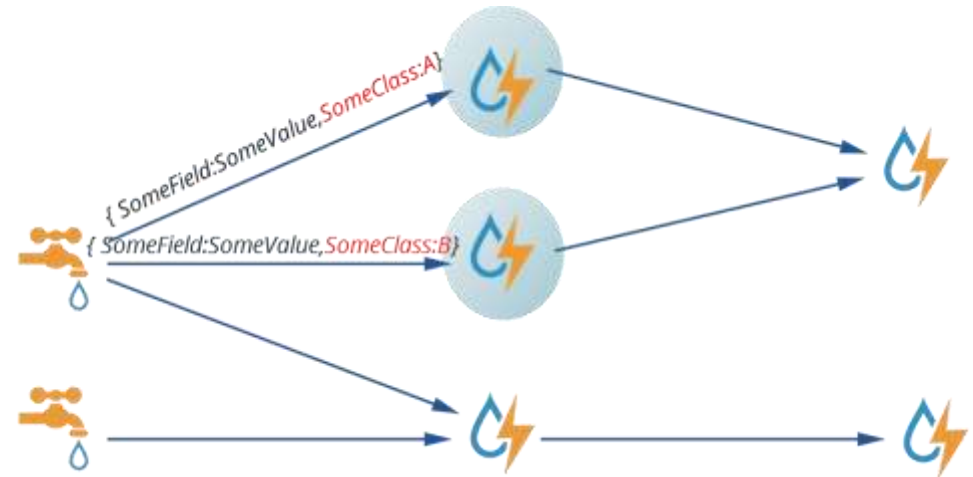
CONCEPTOS BASICOS - SHUFFLE GROUPING

- Las tuplas se dividen de forma aleatoria entre los diferentes bolts de destino.
- El numero de bolts para cada tarea es configurable



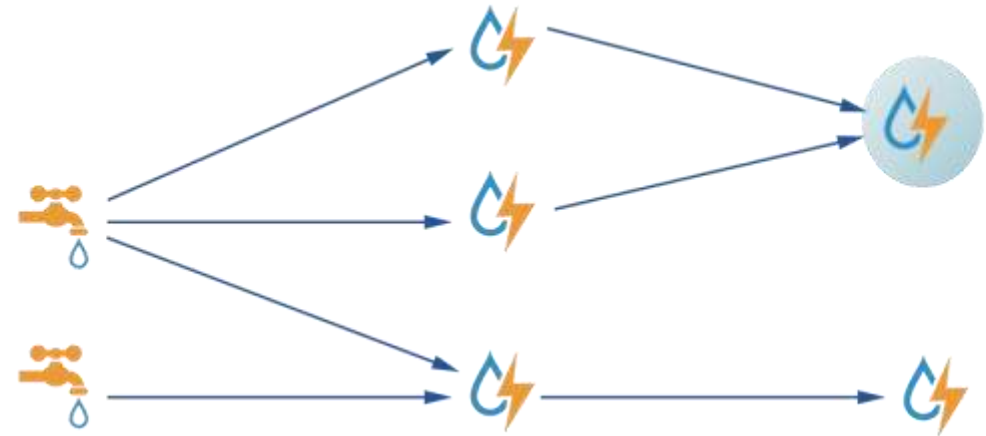
CONCEPTOS BASICOS – FIELD GROUPING

- La distribución depende de algún valor de los elementos de cada tupla
 - { *SomeField:SomeValue*, *SomeClass:A* }
 - { *SomeField:SomeValue*, *SomeClass:B* }



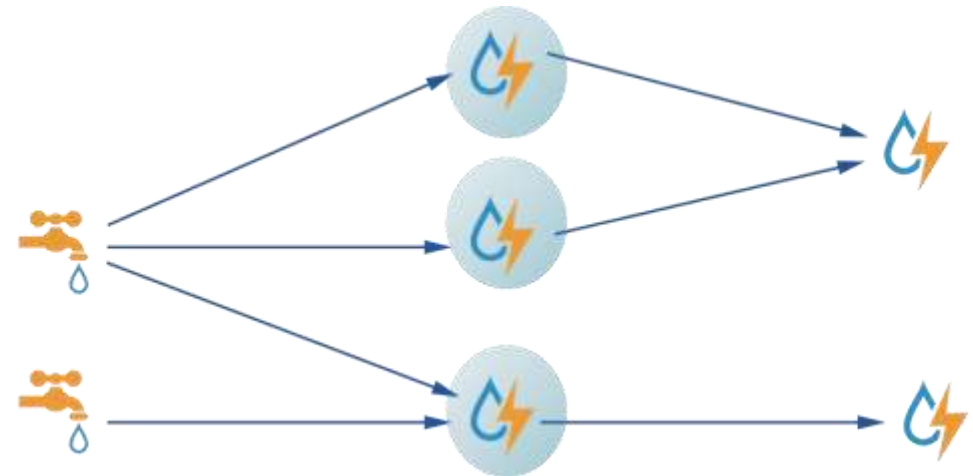
CONCEPTOS BASICOS – GLOBAL GROUPING

- Todos los streams son agrupados en un único bolt



CONCEPTOS BASICOS – ALL GROUPING

- Capa tupla es copiada y enviada a cada bolt



plain concepts

APACHE STORM



¿QUE ES AZURE STREAM ANALYTICS?

AZURE STREAM ANALYTICS

- Azure Stream Analytics es un motor de proceso de eventos
- Nos permite describir las transformaciones que queremos aplicar utilizando una sintaxis SQL
- Esta integrado con la infraestructura de Azure para gestion de colas de eventos de Azure Event Hubs

AZURE STREAM ANALYTICS

Analisis en
tiempo real

Facilidad de
escalado

Manejado
(PaaS)

Disponibilidad

Bajo coste

Desarrollo
rapido

ANALISIS EN TIEMPO REAL

- Ingesta de millones de eventos por Segundo
 - Hasta 1GB/s
- Nivel de carga variable
 - Escalado para acomodarse a cargas variables
 - Baja latencia en el procesamiento
- Transformaciones, enriquecimiento de los datos, correlacion...
 - Correlacion entre diferentes streams o con datos de referencia
 - Búsqueda de patrones en tiempo real

FACILIDAD DE ESCALADO

- Aprovecha las capacidades de Azure para el escalado
 - Aumento del numero de recursos
 - Escalado bajo demanda
 - Arquitectura distribuida

streaming unit pool

Scale settings can't be edited while a job is running.

STREAMING UNITS

A Stream Analytics job can be scaled through Streaming Units, which define the amount of processing power a job receives. Each Streaming Unit corresponds to roughly 1 MB/second of throughput.

MANEJADO

- Paradigma PaaS (Platform as a Service)
 - No es necesario tener experiencia en el despliegue
 - No es necesario mantener el software
 - No es necesario hacer ajustes para mejorar el rendimiento

DISPONIBILIDAD

- Entrega de eventos garantizada
 - No se pierden eventos
 - Los eventos se entregan una sola vez
- Disponibilidad garantizada
 - SLA con 99'9% de uptime
 - Auto recuperación en caso de error
 - Gestion del estado embebida, para simplificar y acelerar la recuperación en caso de error

BAJO COSTE

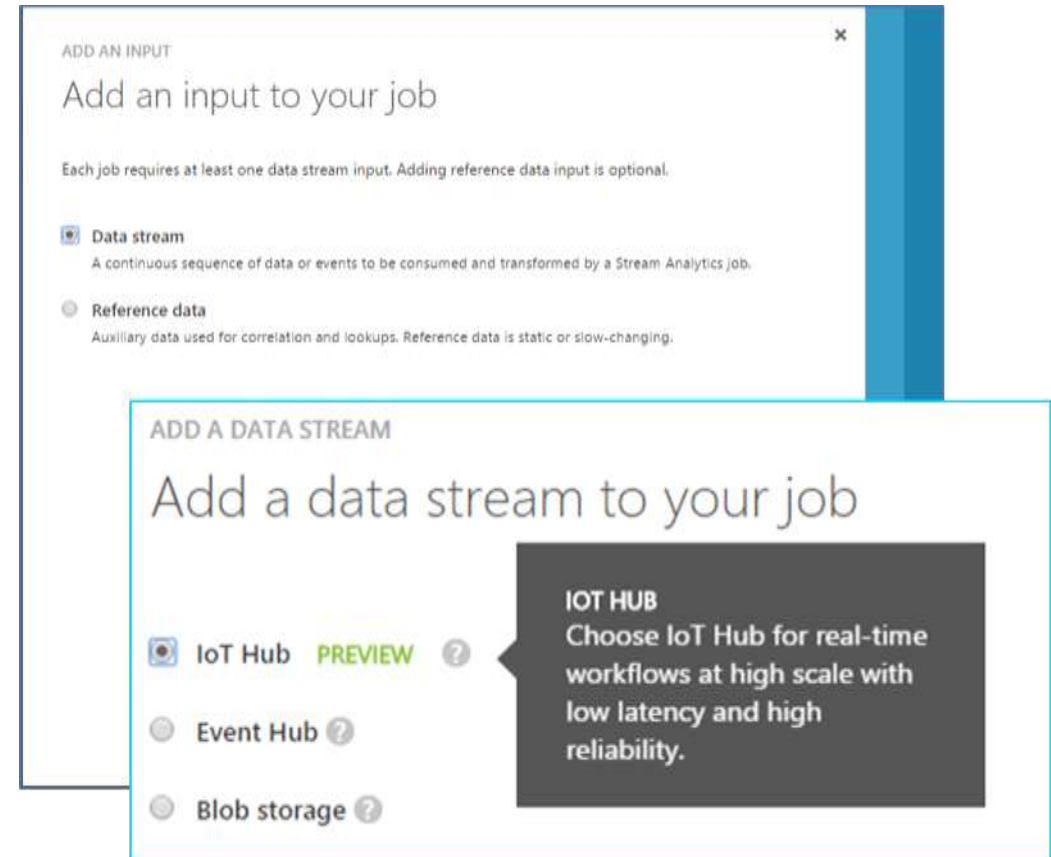
- Pago por uso
 - Si no estamos procesando, no pagamos
- Ventajas de Azure
 - Bajos costs iniciales
 - Posibilidad de escalar de forma incremental

DESARROLLO RAPIDO

- Lenguaje declarativo similar a SQL
 - De alto nivel
 - Conciso
 - Soporte de primera clase a los streams de eventos y los datos de referencia
- Semanticas temporales incorporadas
 - Windowing y joining
 - Configuracion sencilla para tratar con eventos retrasados y/o fuera de orden temporal

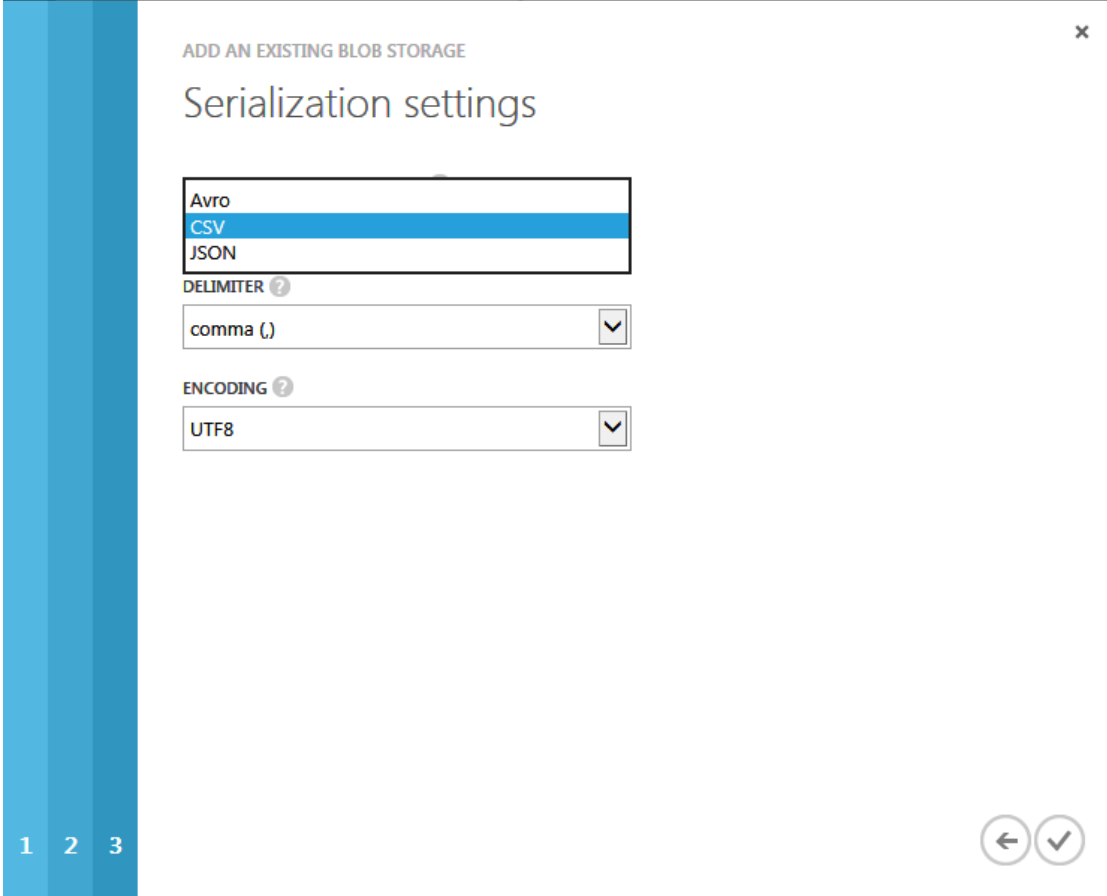
ENTRADAS

- Como entradas de datos de stream
 - Azure Event Hub
 - Azure IoT Hub
 - Azure Blob Storage
- Como entradas datos de referencia
 - Azure Blob Storage.
 - Cacheado para mejorar el rendimiento



DEFINIENDO UN ESQUEMA

- Debemos definir el formato y encoding de los datos de entrada
 - El formato puede ser CSV, JSON o AVRO
 - Para CSV disponemos de distintos delimitadores
 - En el caso de usar CSV o AVRO podemos definir el esquema de los datos



ADD AN EXISTING BLOB STORAGE

Serialization settings

Format selection: Avro, CSV (selected), JSON

DELIMITER ?
comma (,)

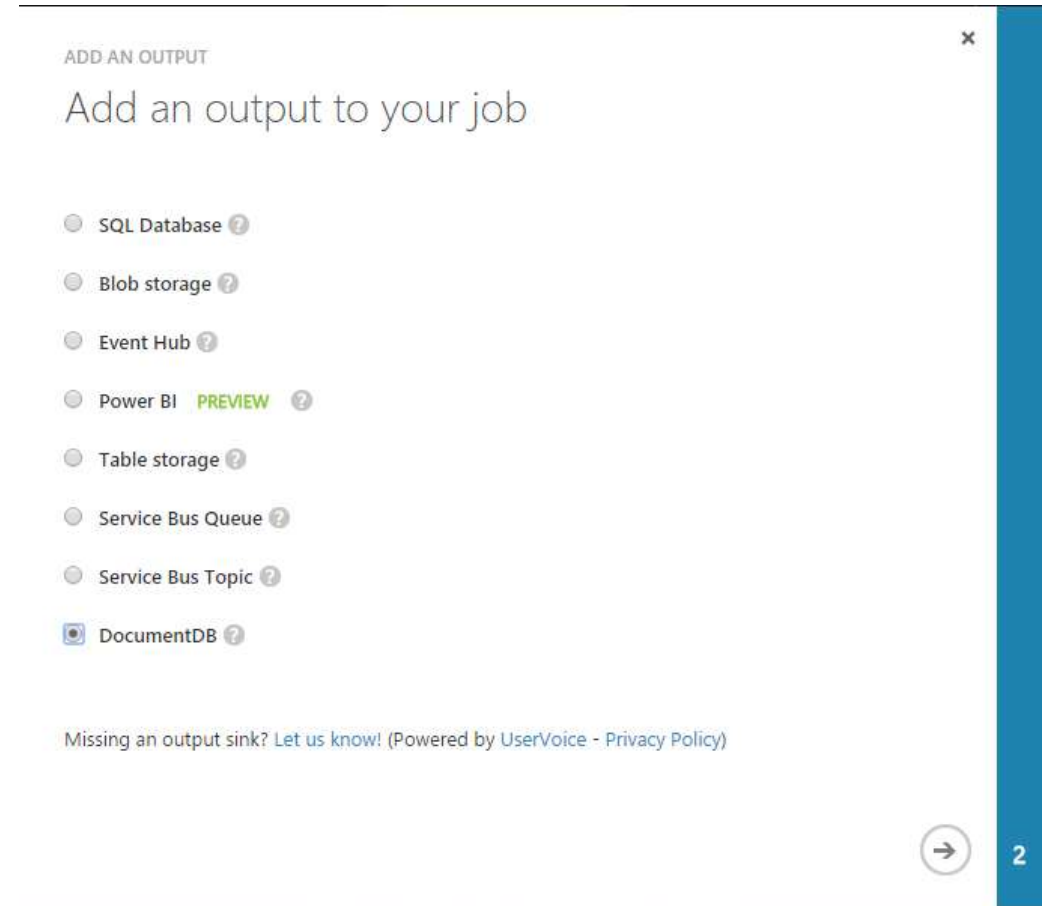
ENCODING ?
UTF8

Progress bar: 1 2 3

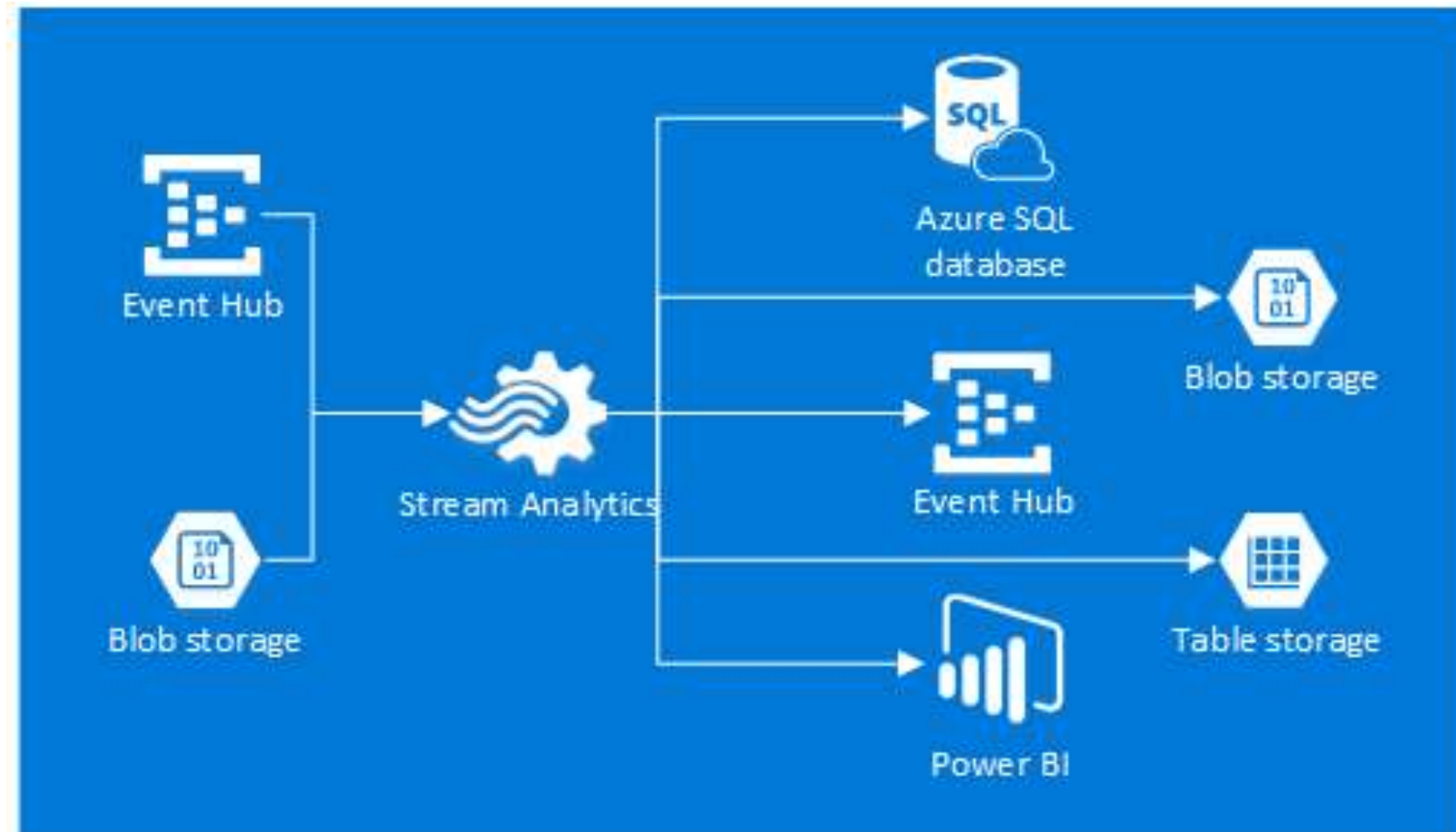
Navigation: back, confirm, close

SALIDAS

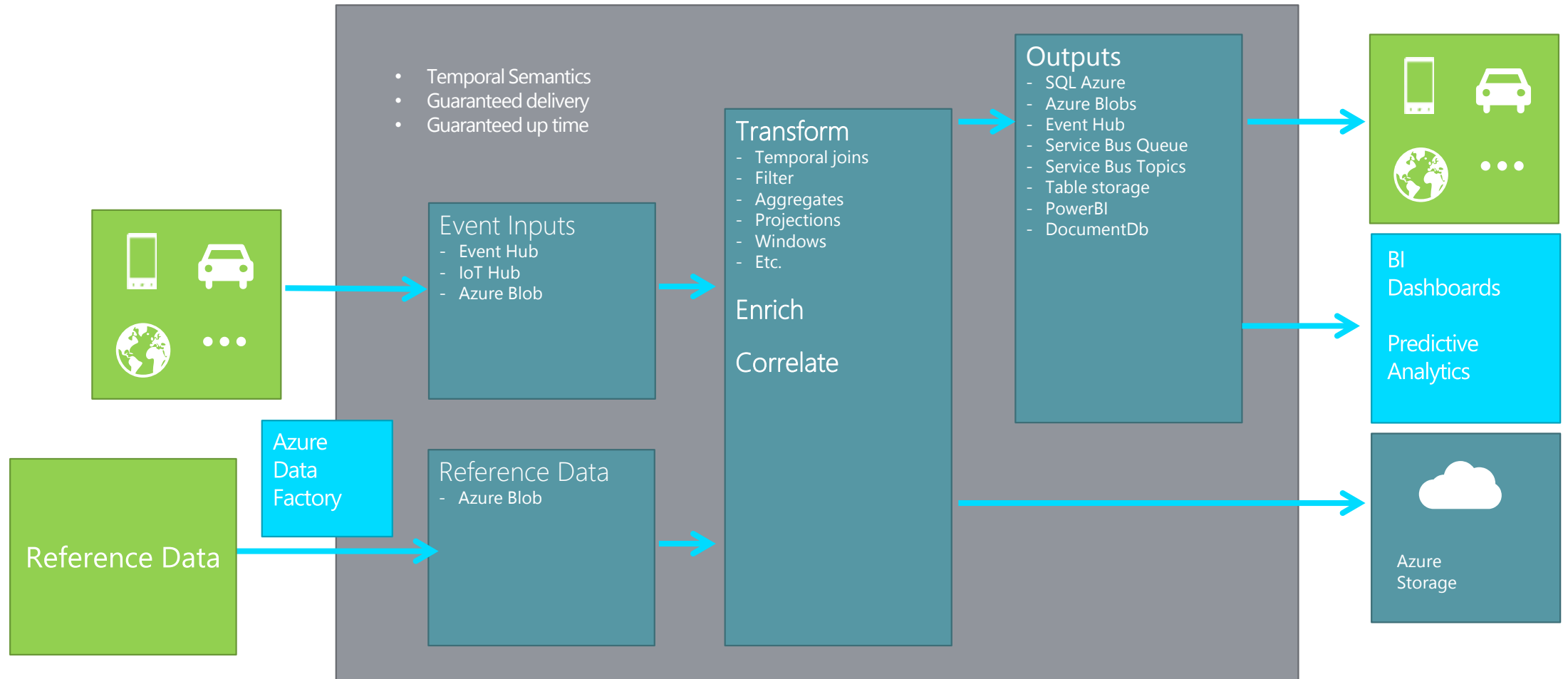
- Azure Blob storage
- Azure Table storage
- SQL database. Para reporting tradicional
- Event hub. Para alertas o notificaciones
- Service Bus Queue
- Service Bus Topicsconsumers
- PowerBI.com
- DocumentDb



ARQUITECTURA



ARQUITECTURA



CONCEPTOS BASICOS DE AZURE STREAM ANALYTICS

EL LENGUAJE

SAQL

DML

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

Scaling Extensions

- WITH
- PARTITION BY
- OVER

Date and Time Functions

- DateName
- DatePart
- Day
- Month
- Year
- DateTimeFromParts
- DateDiff
- DateAdd

Temporal Functions

- Lag, IsFirst
- CollectTop

Windowing Extensions

- TumblingWindow
- HoppingWindow
- SlidingWindow

Aggregate Functions

- Sum
- Count
- Avg
- Min
- Max
- StDev
- StDevP
- Var
- VarP

String Functions

- Len
- Concat
- CharIndex
- Substring
- PatIndex

TIPOS DE DATOS

- Los datos de entrada se castean a uno de estos tipos
- Podemos establecerlos mediante un CREATE TABLE
 - No estemos creando una tabla, simplemente un mapeo de tipo de datos

Tipo	Descripcion
bigint	Integers in the range -2^{63} (-9,223,372,036,854,775,808) to $2^{63}-1$ (9,223,372,036,854,775,807).
float	Floating point numbers in the range - 1.79E+308 to -2.23E-308, 0, and 2.23E-308 to 1.79E+308.
nvarchar(max)	Text values, comprised of Unicode characters. Note: A value other than max is not supported.
datetime	Defines a date that is combined with a time of day with fractional seconds that is based on a 24-hour clock and relative to UTC (time zone offset 0).

WHERE

- Especifica condiciones para las filas devueltas en un SELECT
- Nos permite establecer filtros
- No hay limite al numero de predicados en el WHERE

```
SELECT UserName, TimeZone  
FROM InputStream  
WHERE Topic = 'XBox'
```

INTO

- Canaliza datos entre una entrada y una salida
- Podemos tener multiples salidas
 - Utilizamos el INTO para llevar cada SELECT a su destino
 - Por ejemplo, algunos eventos se van al Blob Storage para analizarse posteriormente, otros van directamente a Event Hub

```
SELECT UserName, TimeZone  
INTO Output  
FROM InputStream  
WHERE Topic = 'XBox'
```

JOIN

- Nos permite combinar multiples streams, o un stream con datos de referencia
 - Podemos especificar la ventana temporal en la que lo aplicaremos, mediante DateDiff

query

```
1 SELECT events.id, events.content, events.tempo, heartbeat.level
2 FROM [events]
3 TIMESTAMP BY EventEnqueuedUtcTime
4 JOIN [Heartbeat]
5 TIMESTAMP BY MeasureDate
6 ON DateDiff(Minute, events, heartbeat) between 0 and 1
```

DATOS DE REFERENCIA

- Información accesoria al stream
 - Estatica o lentamente cambiante
 - Almacenada en ficheros CSV o JSON en Azure Blob Storage
- Podemos utilizarla junto con los datos del stream

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


UNION

- Combina los resultados de dos o mas queries en un único conjunto de resultados que incluye filas de ambas
- El número de columnas y su orden debe ser el mismo en todas las queries
- Los tipos de datos deben de ser compatibles
- Si no utilizamos UNION ALL los duplicados se eliminan

UNION

Tolld	EntryTime	LicensePlate	...	Tolld	ExitTime	LicensePlate
1	2014-09-10 12:01:00.000	JNB 7001	...	1	2009-06-25 12:03:00.000	JNB 7001
1	2014-09-10 12:02:00.000	YXZ 1001	...	1	2009-06-25 12:03:00.000	YXZ 1001
3	2014-09-10 12:02:00.000	ABC 1004	...	3	2009-06-25 12:04:00.000	ABC 1004

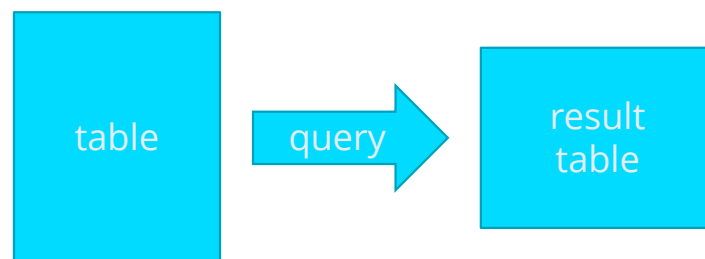
SELECT Tolld, ENTime AS Time , LicensePlate FROM EntryStream TIMESTAMP BY ENTime
UNION
SELECT Tolld, EXTime AS Time , LicensePlate FROM ExitStream TIMESTAMP BY EXTime

Tolld	Time	LicensePlate
1	2014-09-10 12:01:00.000	JNB 7001
1	2014-09-10 12:02:00.000	YXZ 1001
3	2014-09-10 12:02:00.000	ABC 1004
1	2009-06-25 12:03:00.000	JNB 7001
1	2009-06-25 12:03:00.000	YXZ 1001
3	2009-06-25 12:04:00.000	ABC 1004

LA GESTION TEMPORAL

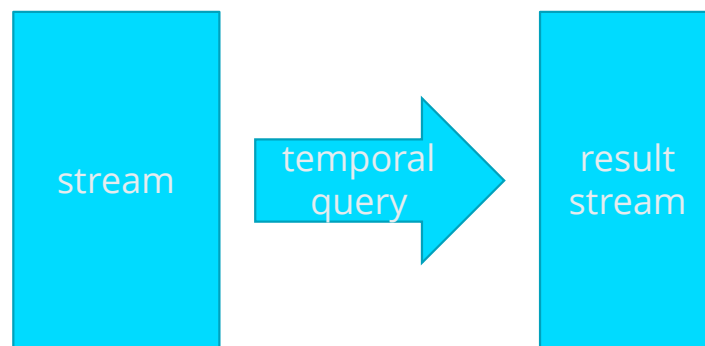
CONSULTAS TRADICIONALES

- Las consultas tradicionales asumen que los datos no cambian mientras las consultas
 - La consulta trabaja sobre un estado fijo
 - En el caso de que los datos varíen, trabajamos sobre un snapshot o una transacción
 - Consultando un estado fijo, la consulta termina en un tiempo finito



CONSULTAS TEMPORALES

- Cuando analizamos un stream de datos, trabajamos con una cantidad de datos potencialmente infinita
 - La consulta nunca terminaria
- Para solucionar esto trabajamos con ventanas temporales



ARRIVAL TIME VS APPLICATION TIME

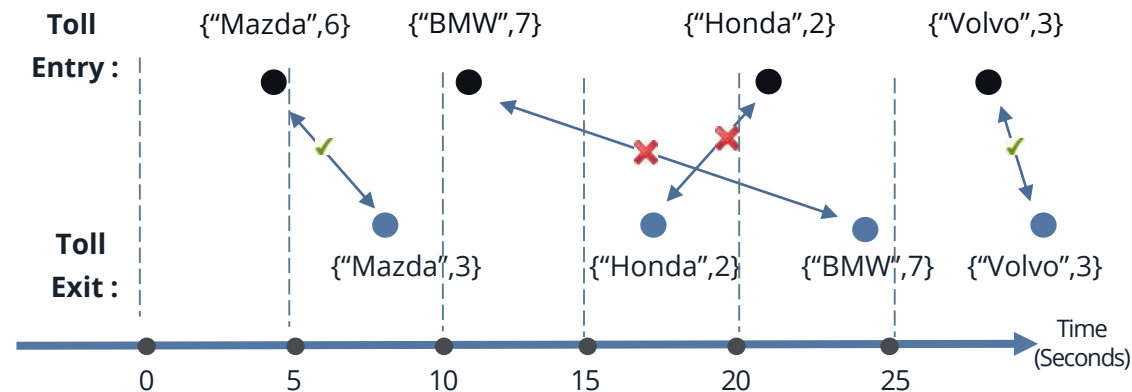
- Todos los eventos tienen un timestamp, accessible mediante el campo `System.Timestamp`
- Por defecto es el tiempo de entrada al sistema
 - Arrival time
 - Para eventos, el que le asigne el Event Hub en su entrada
 - Para datos en blob storage, el Last Modified del blob
- En muchas ocasiones este no es el timestamp que nos interesa
 - Application time
 - Generado mediante la expression `TIMESTAMP BY`

JOINS TEMPORALES

- Hemos visto como utilizar JOIN para combinar eventos de una o mas entradas
- En Stream Analytics, JOIN es temporal
 - Cada JOIN debe especificar limites temporales para la union
 - En la clausula ON, utilizando DATEDIFF

JOINS TEMPORALES

```
SELECT Make
FROM EntryStream ES TIMESTAMP BY EntryTime
JOIN ExitStream EX TIMESTAMP BY ExitTime
ON ES.Make= EX.Make
AND DATEDIFF(second,ES,EX) BETWEEN 0 AND 10
```



"Honda" – Not in result because event in Exit stream precedes event in Entry Stream

"BMW" – Not in result because Entry and Exit stream events > 10 seconds apart

Query Result = [Mazda, Volvo]

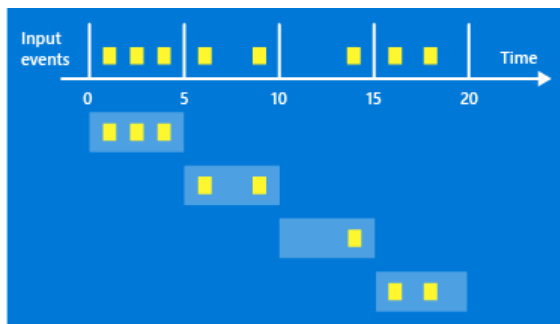
FUNCIONES DE VENTANA

- A la hora de procesar operaciones sobre conjuntos que llegan en tiempo real, un escenario comun es tener un subconjunto temporal
 - Suma de todos los datos en una hora determinada
- ¿Como definimos un subconjunto temporal en un stream?
 - Mediante funciones de ventana

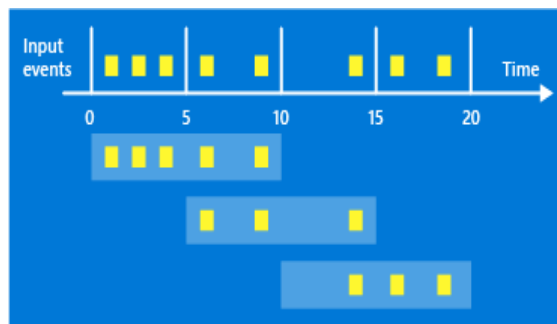
FUNCIONES DE VENTANA

- Una ventana contiene datos de eventos a lo largo de una escala de tiempo
- Cada operación de ventana genera un evento al final de la ventana
 - Con el TimeStamp de la ventana
- Todas tienen una longitud fija
- Se generan con la clausula GROUP BY

FUNCIONES DE VENTANA



Tumbling window
Aggregate per time interval



Hopping window
Schedule overlapping windows



Sliding window
Windows constant re-evaluated

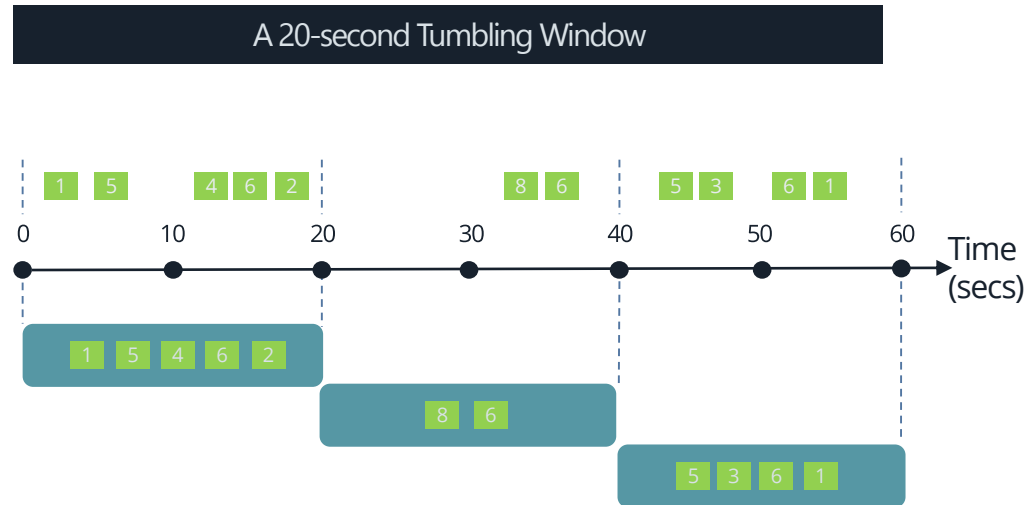
TUMBLING WINDOW

- Se repiten
- No se superponen
- Un evento pertenece a una sola ventana

Query: Count the total number of vehicles entering each toll booth every interval of 20 seconds.

```
SELECT TollId, COUNT(*)  
FROM EntryStream TIMESTAMP BY EntryTime  
GROUP BY TollId, TumblingWindow(second, 20)
```

TUMBLING WINDOW



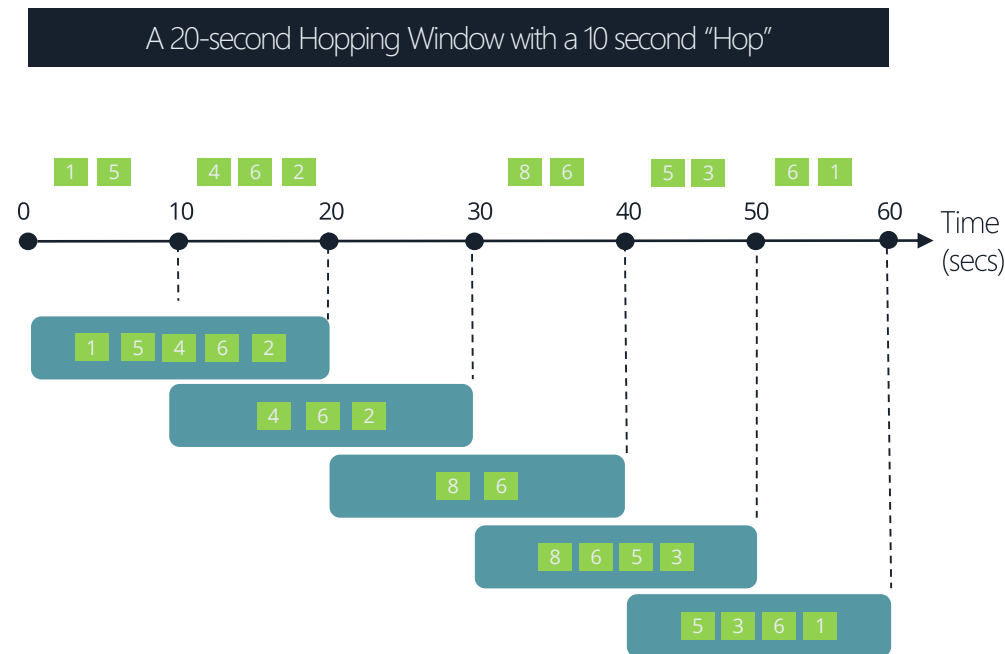
HOPPING WINDOW

- Se repiten
- Se pueden superponer
- Se mueven hacia adelante en saltos fijos
 - Si el tamaño del salto es igual al de la ventana, es igual a la anterior

QUERY: Count the number of vehicles entering each toll booth every interval of 20 seconds; update results every 10 seconds

```
SELECT COUNT(*), TollId
FROM EntryStream TIMESTAMP BY EntryTime
GROUP BY TollId, HoppingWindow (second, 20,10)
```

HOPPING WINDOW



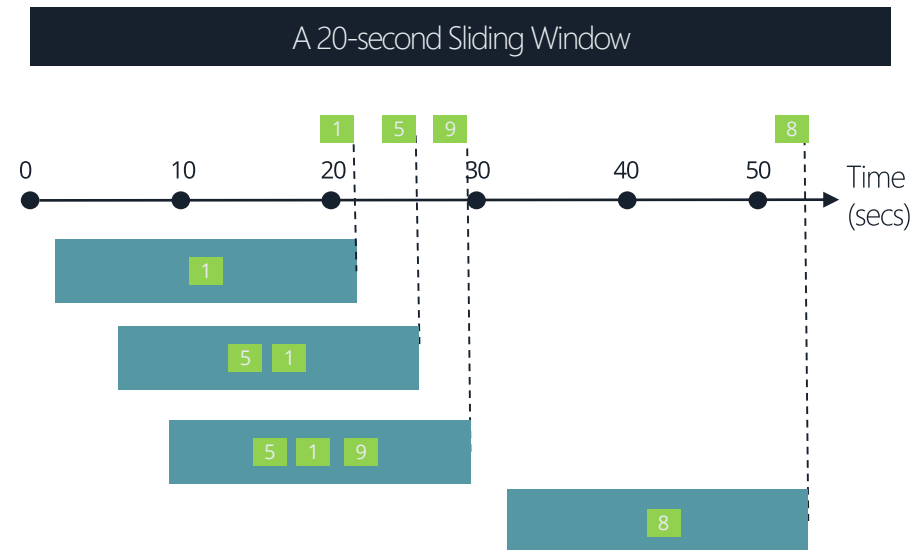
SLIDING WINDOW

- Se mueve continuamente hacia delante de ϵ (epsilon) en ϵ
 - $\epsilon = 1/100$ de nanosegundo
- Genera una salida solo durante la ocurrencia de un evento
- Cada ventana tiene al menos un evento
- Los eventos pueden pertenecer a mas de una ventana

Query: Find all the toll booths which have served more than 10 vehicles in the last 20 seconds

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


SLIDING WINDOW



ESCALANDO EL ANALISIS

STREAMING UNIT

- Medida de los recursos de computacion disponibles para procesar un trabajo
 - Una Streaming Unit procesa hasta 1Mb/Segundo
- Por defecto, cada trabajo consiste de una Streaming Unit
- El número total depende de
 - Ritmo de entrada de eventos
 - Complejidad de la consulta

SUBCONSULTAS

- Una consulta puede tener mas de un paso
 - Un paso es una subconsulta definida utilizando WITH
 - CTE, Common Table Expression
- La unica querie fuera del WITH es tambien un paso
- Nos permite desarrollar queries complejas al utilizar resultados intermedios
 - La salida de cada paso se puede enviar a multiples lugares utilizando INTO

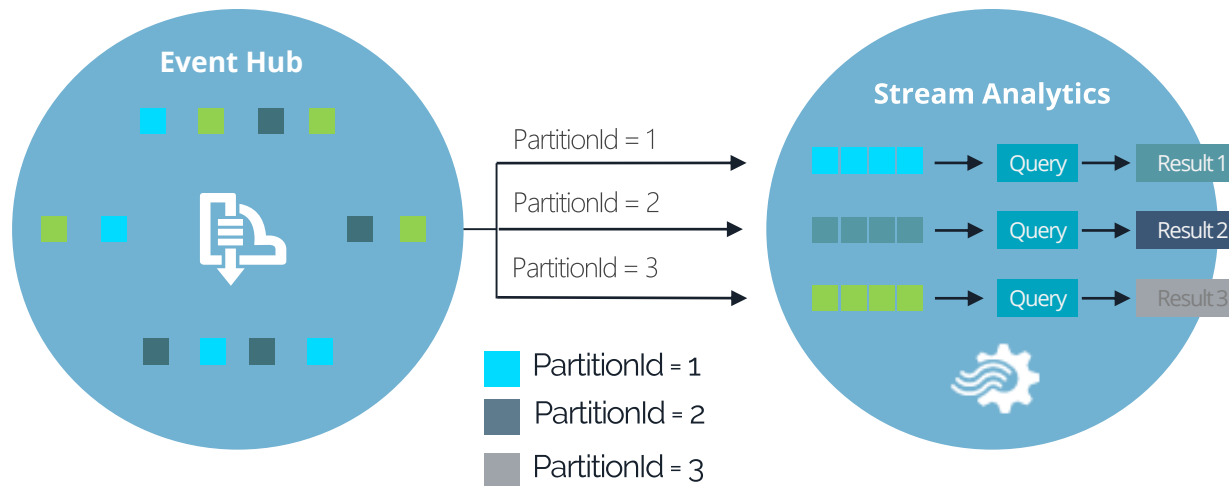
```
WITH Step1 AS (  
    SELECT Count(*) AS CountTweets, Topic  
    FROM TwitterStream PARTITION BY PartitionId  
    GROUP BY TumblingWindow(second, 3), Topic, PartitionId  
)  
Step2 AS (  
    SELECT Avg(CountTweets)  
    FROM Step1  
    GROUP BY TumblingWindow(minute, 3)  
)  
SELECT * INTO Output1 FROM Step1  
SELECT * INTO Output2 FROM Step2  
SELECT * INTO Output3 FROM Step2
```

PARTICIONADO

- Cuando una consulta esta particionada, los eventos de entrada se procesan y agregan en grupos separados
 - Se genera un evento de salida por cada partición
- La consulta debe utilizar la sintaxis PARTITION BY
- Si la entrada es un Event Hub particionado, podemos escribir consultas y subconsultas particionadas
- Nos permite maximizar el numero de Streaming Units a utilizar

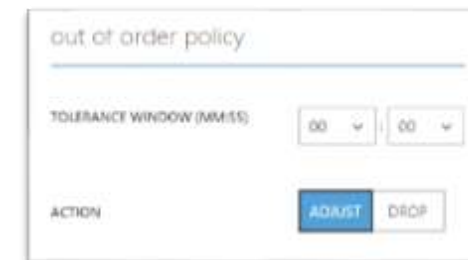
PARTICIONADO

```
SELECT Count(*) AS Count, Topic  
FROM TwitterStream PARTITION BY PartitionId  
GROUP BY TumblingWindow(minute, 3), Topic, PartitionId
```



ENTRADAS DESORDENADAS

- Podemos configurar una tolerancia en la Out of Order Policy para tratar con eventos desordenados
 - Por defecto es 0, lo que indica que esperamos que los eventos siempre lleguen en orden
- Utilizar un valor mayor que 0 permite a Azure Stream Analytics corregir el desorden
 - Habrá un buffer del tamaño especificado y dentro de ese buffer se reordenaran los eventos en base al TimeStamp seleccionado
 - Esto ocasiona un retraso en la salida



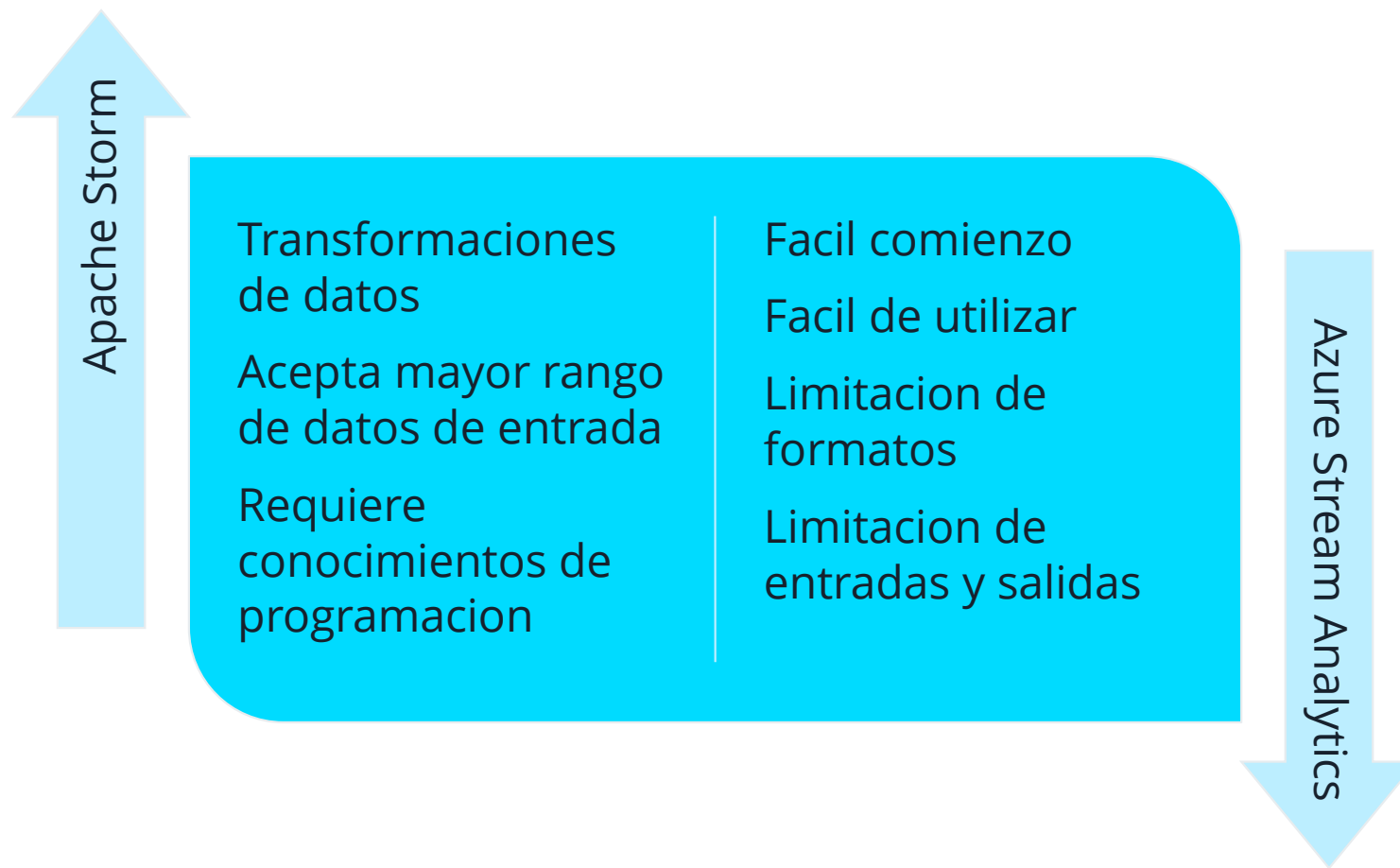
The screenshot shows a configuration window titled "out of order policy". It contains a "TOLERANCE WINDOW (MM:SS)" field with two dropdown menus, both set to "00". Below this is an "ACTION" section with two buttons: "ADJUST" (highlighted in blue) and "DROP".

plain concepts

AZURE STREAM ANALYTICS



STREAM ANALYTICS VS APACHE STORM





¿PREGUNTAS?



GRACIAS

Barcelona



Bilbao



Madrid



Sevilla



Dubai



London



Seattle