Microsoft

Reference Architecture

Real-Time Event Processing with Microsoft Azure Stream Analytics

Abstract:

The Reference Architecture for real-time event processing with Microsoft Azure Stream Analytics provides a framework for designing and deploying event based data processing solutions on Microsoft Azure. The intended audience for this paper includes Business Decision Maker (BDM) and Information Technology Decision Maker (ITDM) resources who are interested in the benefits and business value of real-time data insights, as well as architects and developers who are evaluating real-time data solutions based on Azure Stream Analytics and supporting Azure services.

Author:
Charles Feddersen, Solution Architect, Data Insights Center of Excellence
charlesf@microsoft.com

Reviewers:
Santosh Balasubramanian, Senior Program Manager, Azure Stream Analytics
Delbert Murphy, Azure Technical Specialist

Publication Date: January, 2015

Revision: 1.0
Please feel free to submit any feedback comments or suggestions either directly to the author of this document or via the comments section on the download page.

Table of Contents

1. Executive Summary

Companies across every industry vertical have an opportunity to benefit from faster data insights and decision making. As businesses look for new competitive advantages in their respective industry, one of the opportunities often identified is the ability to derive actionable insights from information faster than had previously been possible. Businesses that can make better, faster decisions in response to their customers or operations stand to gain market share by delivering higher levels of customer satisfaction, driving repeat business, and ultimately obtaining larger wallet share. The ability to exercise rapid response rates to business events in seconds, rather than minutes or hours can yield significant revenue upside to company operations.

The Microsoft portfolio of data platform products and cloud services provides a robust and sophisticated set of capabilities that range from large scale storage and information orchestration to event processing, rich interactive visualizations and even machine learning to support predictive analytics. By leveraging proven patterns and practices for deploying these capabilities, either separately or in combination, customers can rapidly realize new value from their data whilst minimizing the risks that are often associated with projects of this nature.

The reference architecture for real-time event processing with Microsoft Azure Stream Analytics provides a layered model that describes how supporting Azure services such as ingestion and storage can be leveraged to provide a robust, end-to-end solution for event driven analytics in the cloud.

This architecture is designed to deliver a repeatable implementation pattern for real-time analytics that can be applied across a variety of horizontal solution domains, such as customer analytics or operational intelligence. Examples of scenarios within these solution domains include web analytics, predictive maintenance, fraud detection, and recommendation engines to name a few. This architecture adheres to a set of key architectural concerns to help ensure that the deployed solution is enterprise ready and will support the highest operational performance and reliability demands.

The combination of Platform as a Service (PaaS) and Software as a Service (SaaS) cloud services that support this architecture ensures that the Azure consumption cost structure for operationalization is tightly aligned to the volume of data consumed and processed, and therefore can be forecast with a high degree of accuracy. This is opposed to the cost structure that is typically associated with Infrastructure as a Service (IaaS) or private cloud deployments whereby the cost is driven by uptime regardless of the actual infrastructure utilization. The

inherent elasticity of matching cost with actual consumption helps reduce the effort to manually manage cloud resources based on potentially unpredictable or cyclical demand spikes.

2. Introduction to Real-Time Analytics

2.1 Traditional Analytics Approaches

Traditionally the majority of business intelligence and analytics solutions have been designed around the concept of batch operations that move data between different persisted data stores, such as relational databases or analytics cubes. Users would then issue a query against the data at rest to support scenarios such as ad-hoc analysis, dashboards or scorecards. While this approach has proven to be exceedingly successful for a number of years, and still remains a highly relevant solution to business operations today, the sheer volume and velocity of data being generated by modern applications or devices is putting pressure on this existing paradigm. The following diagram illustrates this approach.

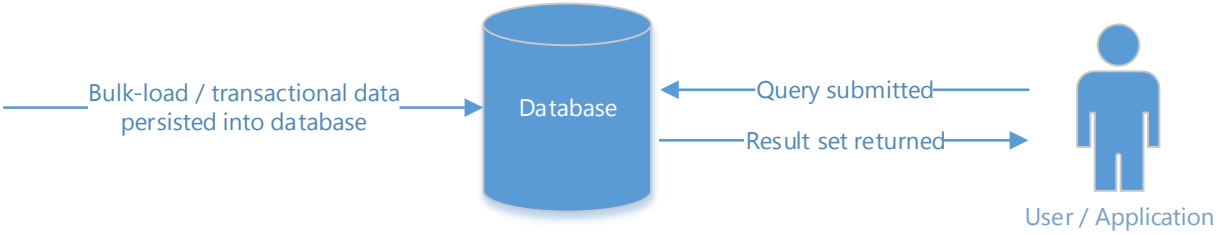

Figure 1: High-Level approach for analytics query submission / response for persisted data stores

To circumvent some of the challenges related to timeliness of information being made available to decision support systems, companies have often looked to optimize their existing architectures using techniques such as accelerating data loading schedules, or manually scaling out data transformation processes. In some instances this has proven to be a successful remediation for making data available faster, however data availability is generally still limited to minutes at best. There is a limit as to frequently that an extract / load process can be instantiated and completed against a source system prior to being restarted.

Alternatively, some companies have been able to accelerate decision making by leveraging technologies such as database replication that can asynchronously transfer data between transactional operational databases and analytics repositories. This approach has proven to fulfil the requirement of near real-time data availability for analytics, however the need for continual, up to the second insights across large datasets does create significant pressure on the analytics repository due to the continual submission of queries. Ultimately, different types of questions require different approaches to answering them in the most efficient manner.

2.2 Shifting to a Streaming Data Solution

Enabling a real-time analytics solution based on streaming data provides a solution to a number of the aforementioned challenges of real-time data at scale. The model for the implementation represents a significant shift by moving from point queries against stationary data, to a standing temporal query that consumes moving data. Fundamentally, we enable insight on the data before it is stored in the analytics repository. As such, companies gain the benefits of real-time insights on data as business events occur, but also the ability to store this information in a robust repository for historical analysis at a later time. The following diagram illustrates the approach to real-time analytics, demonstrating the contrast of Figure 1 to show how real-time analytics is applied.

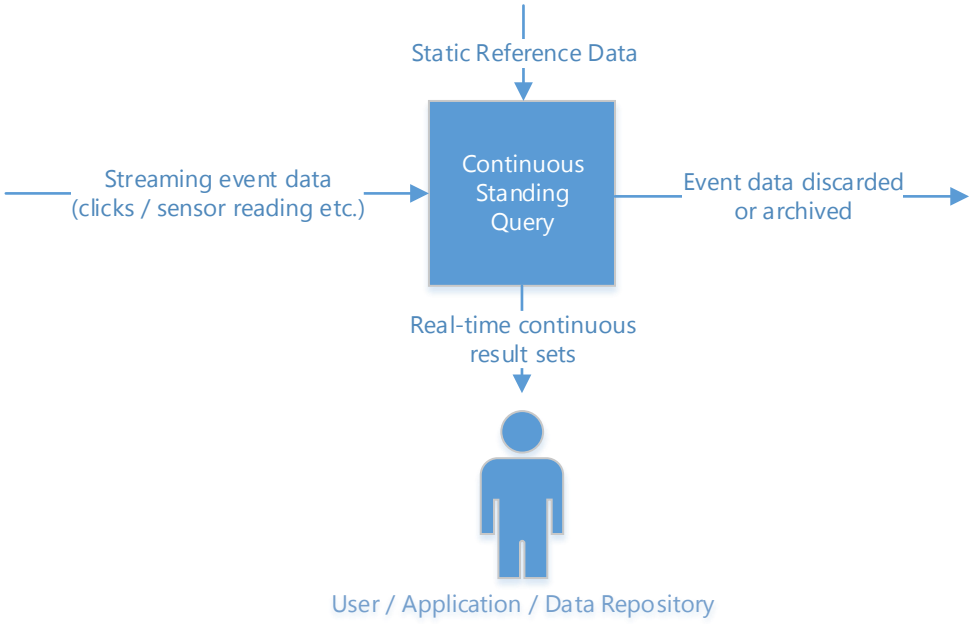

Figure 2: High-Level approach for standing query to support moving data in-flight

Streaming data solutions are not a new introduction to the marketplace. Microsoft has had offerings in this space for a number of years now. Additionally, some industries, in particular financial services have been deploying large scale streaming data solutions for many years to support the highly time sensitive nature of their business. By enabling a scalable PaaS streaming data service in Azure that interoperates seamlessly with a set of other data services in Azure such as event ingestion technologies, databases, and visualizations, Microsoft can help lower the barrier of entry for deploying modern real-time decision support systems across a wider set of industry verticals.

2.3 Complement Existing Analytics with Streaming Data

Whilst enabling analysis over moving data should be an approach considered for a number of existing business information challenges, it is not intended to be a replacement for the more traditional batch oriented analytics architectures. There remain many scenarios where streaming analytics does not fulfil the solution requirements. For example, there are many 3rd party providers that deliver data to companies periodically through flat files. Alternatively, many business system data sources, such as relational transactional databases, are dependent on ETL processes that execute on a fixed schedule so as to reconcile the period before it is made available for analysis and to provide a clean cut-off date for reporting.

Additionally, modern data warehousing solutions can now natively scale out through architectures such as massive parallel processing (MPP) that provides near linear scale across a database cluster through a shared nothing hardware design. The rise of Hadoop as a scale out open-source platform for analytics, combined with the on-going reduction in storage costs, is enabling companies to shift from the traditional ETL approaches to an ELT solution where all data can be stored and analyzed on demand. This approach is particularly important for businesses looking to develop and deploy advanced predictive analytics capabilities to support automated decision support or forecasting.

The reference architecture presented in this document will make reference to other data services such as those mentioned above, however the focus of the content will be towards the critical path of an end-to-end solution for real-time event processing for analytics.

2.4 Multiple Stream Processing Offerings

Microsoft currently provides three event stream processing products that customers can deploy for their solutions. StreamInsight ships as part of SQL Server on-premises product and is a .NET framework-based development platform. Solutions developed with StreamInsight can be deployed within customer datacenters on traditional server infrastructure, or alternatively in the cloud using IaaS as the platform.

One of the challenges that customers frequently face when architecting cloud solutions is about making decisions between multiple services that offer similar capabilities. The Microsoft Azure platform offers a vast set of data services, and while it's a luxury to have such a broad array of capabilities to select from, it can also present a challenge. Designing a solution requires that customers evaluate which offerings are best suited to their requirements as part of the planning and design phases of a project.

There are a number of instances where Azure provides similar platforms for a given task. For example, both MongoDB and DocumentDB are database technologies designed to support

schema free JSON data storage for mobile and web applications. They are both designed to scale linearly with predictable performance, and provide indexing for rapid response rates. Customers intending to develop new applications that require this type of data platform should evaluate each of these Azure offerings, and trace their requirements against the platform capabilities before committing to one or the other.

A similar situation exists for Azure HDInsight Storm and Azure Stream Analytics. Each of these PaaS offerings provides event stream processing, however a closer look at the services reveals differences in both their deployment model and developer interfaces. Both of these platforms provide highly capable engines that are suitable for a range of solution deployments, however some of the differences will influence the decision of which platform is best suited to a project. Common areas of comparison include:

- Language support - Storm offers a more diversified set of languages whereas Azure Stream Analytics supports only a SQL language very similar to that provided with SQL Server.
- Deployment model – Storm runs on dedicated HDInsight clusters, whereas Azure Stream Analytics has built-in multi-tenancy support.
- Maintenance – Both platforms are fully managed services, no patching etc. required
- Interoperability – Azure Stream Analytics has first party click & configure support for Event Hub, Azure Blob Storage, and Azure SQL Database. Storm has ingestion from Azure Event Hub, Azure Service Bus, and Apache Kafka amongst others, as well as data egress to Apache Cassandra, HDFS and SQL Azure Database.

The above bullets serve only as a high level example of some of the differences, and are not intended to favor one offering or another. Different solutions will be better suited to each of these platforms. There is extensive information online for each of these platforms that can support making an informed decision on the most appropriate stream processing platform for a given set of requirements. The Microsoft Azure documentation center is a good place to start, located at http://azure.microsoft.com/documentation/ .

Given that Azure Stream Analytics and HDInsight Storm are both PaaS offerings in Azure, it is natural that customers and partners will evaluate each when designing a solution. While the focus of this paper is to provide a reference architecture for developing a solution with Azure Stream Analytics, section 6 of this document positions Storm within the layered architecture to show where it aligns to the surrounding services.

3. Value Proposition of Real-Time Data in Azure

By implementing automated data systems that can monitor and act on repetitive known business processes, or potentially detect outlier events as they occur, organizations can redeploy human resource away from lower value repetitive tasks and move their responsibility to high impact analytical tasks such as researching new target markets or process optimization.

As mentioned earlier in this document, the cost elasticity of PaaS applications is highly flexible, far more so than staff resources that typically have a largely fixed cost. By applying this type of model to applications that support varying workloads, organizations can significantly lower their operational costs for repetitive tasks, while increasing the value derived from their staff. Additionally, given that many operational systems require support on a 24/7 basis, the automation of supporting tasks can reduce the burden on employees to be available outside of standard working hours.

It's not uncommon for streaming data solutions to consume data from globally distributed systems, such as mobile applications or web servers that have been locally deployed to reduce request latency. Deploying cloud based services can help customers effectively deploy processing capabilities where the data is generated, thereby reducing the costs of regional datacenter deployment such as bandwidth and fixed CAPEX infrastructure costs, whilst benefiting from elastic consumption pricing and native scale-out capabilities that support unanticipated growth or decline in consumption.

4. Common Scenarios for Real-Time Analytics

Real-time event based analytics and decision engines can be applied to a variety of industry scenarios. The following sections are intended to provide a subset of examples of where a streaming data analytics solution can add value for business.

4.1 Inventory Management

Optimized inventory control can help businesses ensure that retail store shelves maintain a continual level of stock to support unpredictable purchasing habits. Understocking items can result in missed sales opportunities, and even worse, the potential loss of a customer to a competitor. Customers who are unable to purchase one item, particularly at the start of the shopping process, may be more inclined to not purchase any items and instead move directly to a similar retail store.

By implementing a real-time analytics system to correlate existing stock on the shelf against items being checked-out at the register, stock clerks can be informed in real-time about declining shelf volumes of any particular product line. This scenario becomes particularly valuable during periods of variable demand, such as seasonal shopping whereby a certain product (for example, Christmas lights) may see significantly higher demand from consumers, resulting in low shelf stock situations.

4.2 Demand Based Price Elasticity

For industries that maintain highly elastic pricing models depending on demand, the introduction of real-time analytics and decision making is a powerful solution for supporting scenarios of rapid demand growth whereby price adjustment must be automated.

Consider a hotel chain or perhaps an airline. Both of these industries constantly adjust their product pricing (hotel rooms, or base seat airfares) according to demand and seasonality. Sudden shifts in demand, such as a large conference in a city, can significantly impact travel reservations for a given destination. In this instance, real-time analytics can help detect a burst in the rate of bookings being taken. By leveraging temporal queries, such as the count of bookings within the past 30 minutes, recalculated every 1 minute moving forward, the system can very quickly detect an increase in the rate of bookings over time as it is happening, and subsequently either adjust pricing automatically or notify an operator that a surge of bookings is currently occurring.

4.3 Infrastructure Monitoring

Data centers are continuing to grow in response to the increasing demand for compute and storage resources. Large enterprises can maintain thousands of servers, whilst cloud hosting providers can support tens or possibly hundreds of thousands of servers. For operations to maintain this scale of infrastructure in a reliable and efficient manner, a traditional reactive method will not scale sufficiently to support these operations in a cost effective manner.

The scalability of Azure based event processing provides a robust solution for monitoring even the largest scale server and infrastructure environments in real-time. Enabling scaling to support high volumes of event throughput per second can help infrastructure teams deliver proactive support through either basic rule based outlier detection, or more advanced predictive analytics based on machine learning to support the largest datacenter deployments.

Infrastructure events such as above average CPU consumption or prolonged temperature increases can be identified and surfaced with prioritization through operational dashboards for attention. Additionally, machine learning models can be operationalized and invoked to predict shifting capacity requirements or potential equipment failures prior to the quality of service being degraded.

4.4 Device Telemetry

The number of connected devices is growing at an exponential rate. While some connected devices such as security badge readers are quite mature in their adoption, others such a vehicles or smart buildings are still very much in their infancy. By enabling real-time analytics for connected vehicles, manufactures enable new levels of visibility into the running performance of a given vehicle, rather than waiting to download telemetry data during scheduled vehicle service. This in turn can enable additional scenarios such as predictive maintenance whereby the health of the vehicle is consistently validated against proven machine learning models to detect anomalies in the vehicle.

Large commercial and industrial buildings depend on a number of systems to support day to day operations. One example is the air handlers that support climate control requirements within the premises. By correlating real-time airflow data from the air handlers with the change of temperature in the building, operators can identify in real-time whether the equipment is performing at optimum levels.

4.5 Website Analytics / Content Management

Websites can generate a vast amount of data based on users' interactions. By understanding metrics such as time spent on a page or click paths through the site, web designers can continue to refine layouts and content to suit the individual. Real-time web metrics can be very useful to understand the impact of advertising investment intended to drive people to a site. Additionally, by leveraging real-time telemetry and analysis of site actions, developers can perform highly interactive A/B testing of site layouts to continually iterate changes and optimizations. This can be particularly effective in sites that experience heavy usage.

4.6 Fraud Detection

The cost of fraudulent transactions for large scale financial or insurance institutions can run into the millions of dollars per year. Common scenarios include identity theft and stolen credit card details. Solving for these scenarios is a particularly time sensitive issue as the amount of time taken to identify a fraudulent transaction and automatically cancel a credit card can directly impact the bottom line of a card issuing company.

Real-time analytics and decision engines are particularly well-suited to supporting this scenario as each payment can be analyzed as the transactions occurs. This may include processes such as counting the rate of transactions that are occurring for a single credit card, or correlating the use of the same card across varying geographical distances. It would obviously look suspicious for the same card to be used at a retail outlet in Seattle and Sydney at the same time!

Furthermore, by leveraging a machine learning model based on historical spending patterns for the card, a decision engine can predict whether a seemingly legitimate purchase has a higher probability of being fraudulent for that given credit card. This enables personalized insights rather than a more generalized rules based approach.

5. Architecture and Components

It is important to understand the various layers of the reference architecture prior to drilling into the products and services and their associated capabilities for supporting each of these layers. Similar to traditional business analytics solutions, the architecture for real-time event processing is based on the pattern of data ingestion, processing, and ultimately consumption by either end users or other decision support applications. The following table provides a high level explanation of the end-to-end architecture and a summary of the roles that each of these layers fulfils.

Presentation / Consumption	• Rich interactive visualizations for real-time data analysis • Application integration for process automation • Descriptive analytics using traditional SQL like languages • Dashboards and reporting
Data Storage	• Store aggregated / filtered events in relational storage • Consolidate and store raw events into files for historical analysis
Real-Time Analytics	• Temporal engine for analyzing data across time-series windows • Consume reference data to compare current and historical data • Aggregate / filter / join incoming event streams • Predictive analytics based on models leveraging incoming events and reference data
Data-Integration	• Collection and persistence inbound events • Persisted reference data
Data Sources	• Internal data already stored in the cloud • Internal data stored in on-premise repositories • External data such a devices or applications

Table 1: High level view of reference architecture layers

The real-time event processing reference architecture is dependent on a number of different Azure PaaS capabilities to support end-to-end analytics scenarios. The following diagram and subsequent sections describe the various stages of the data flow lifecycle within this architecture, as well as the components required to support each of these stages, and the interaction between each of these components.

It should be noted that, given that this document is intended to act as a reference for a number of different streaming scenarios, it is unlikely that all of the components prescribed will be relevant to any specific implementation, such as web analytics or fraud detection.

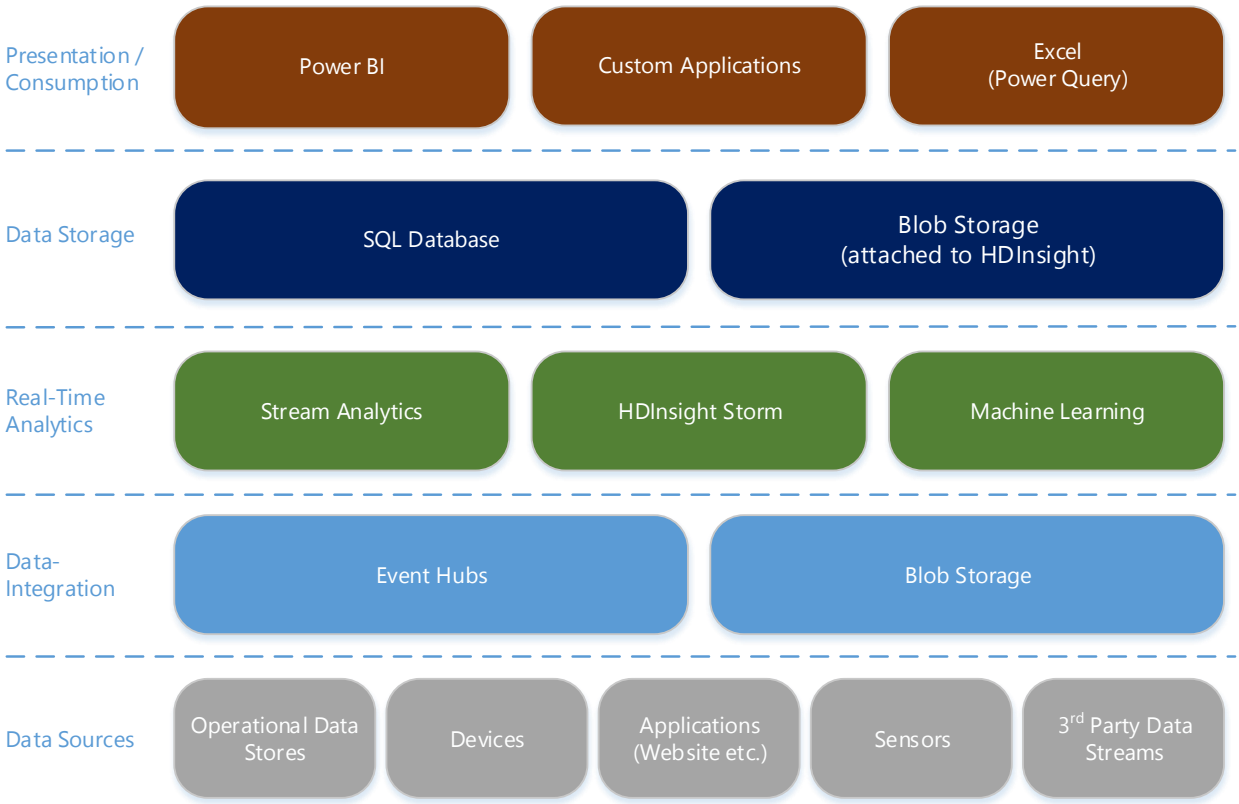

Figure 3: Microsoft Azure capabilities aligned to real-time analytics solution layers

The roles that each of the services illustrated above can support tends to blur across the layer boundaries. For example, SQL Database provides both a storage capability as well as a query engine for returning records to a user. Also, Event Hubs provides storage for a pre-defined period of time. This diagram is intended to illustrate each of the services being mapped to their primary role within this architecture.

5.1 Data Sources

The Data Source layers represents the spectrum of systems that can generate events for processing in the analytics engine. The scope of applications and systems that could fulfil the data source layer is incredibly vast and as such a detailed examination of all possible sources is beyond the scope of this document.

Data may be generated from systems already in the cloud, or from devices local to a person or piece of equipment. Additionally, the events from these systems may be delivered on time as they occur, or batched with timestamps to be delivered when connectivity it available. Finally, data sources may generate their own timestamp for when an event occurred, or require one to be assigned on the Azure side of the data transmission. This is a particularly important aspect to consider given the time sensitivity of temporal event processing solutions.

Another consideration for external connectivity is whether the device itself connects directly to the Event Hub, or whether a gateway is deployed to enable communications between device and cloud. A detailed examination of connectivity methods into Azure is also beyond the scope of this document. For the purpose of providing the event processing architecture, we make the assumption that events being generated are already being securely sent to the Event Hubs ingestion point.

5.2 Data-Integration Layer

The Data-Integration layer provides a set of capabilities for ingesting data of varying formats and velocity from either external sources or existing cloud storage. Both Event Hubs and Blob Storage are foundational storage components of Azure that often support additional solution scenarios outside of real-time analytics. As such, they create a natural data-integration point for additional workloads such as archival storage, batch analytics, or supporting self-service business intelligence. The following diagram illustrates a mapping of source systems into their respective integration layer service, based on the type of data that is expected.

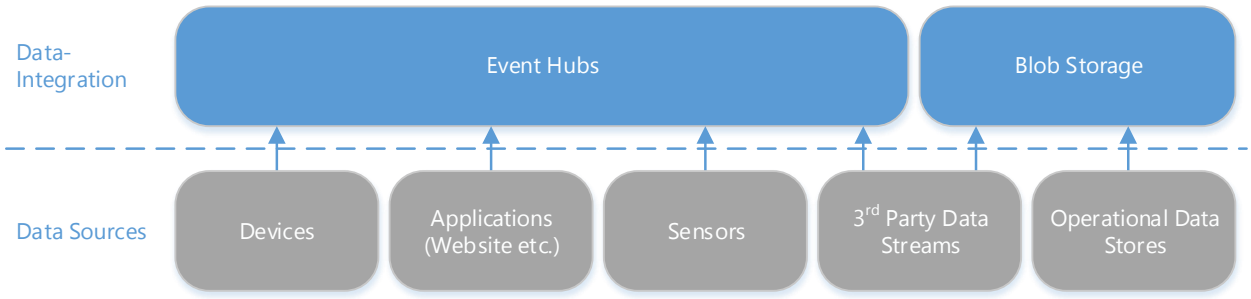

Figure 4: Sample Data Sources mapped against data-integration layer ingestion platforms

5.2.1 Azure Event Hubs

Event Hubs is a highly scalable publish-subscribe data integrator capable of consuming large volumes of events per second, enabling Azure to process vast amounts of data from connected applications or devices. It provides a unified collection point for a broad array of platforms and devices, and as such, abstracts the complexity of ingesting multiple different input streams directly into the streaming analytics engine.

In this streaming analytics scenario the Event Hubs can fulfil two separate roles. The primary role of Event Hubs in this architecture is to provide a robust ingestion capability for incoming data that is being generated at data source endpoints such as devices or sensors. Once these events are loaded, the real-time event processing engine can then consume these events from Event Hubs for performing temporal analytics.

Figure 5: Azure Event Hubs supporting Stream Analytics as a real-time event input

Events are persisted in the Event Hubs for a period of time that is set through a configurable value. This setting can be modified after Event Hubs is deployed also. The persistence of events in Event Hubs is important for two reasons. Firstly, it allows a stream to be re-run over a set of events by designating a historical start time for the streaming engine to run from. Azure Stream Analytics can consume events based on either the streaming job start time (which is the default setting) or a historical point in time as defined for that job. When consuming based on job start time, only events that land in Event Hubs after the stream is started will be consumed. By setting a historical start time, the streaming engine effectively enables a "replay" of events up to the current point in time in which the stream will continue to consume events as they arrive.

The second reason for which persisting events in Event Hubs is important is that it allows multiple Stream Analytics jobs with different purposes to consume the same event for processing. The following diagram illustrates a generic view of this.

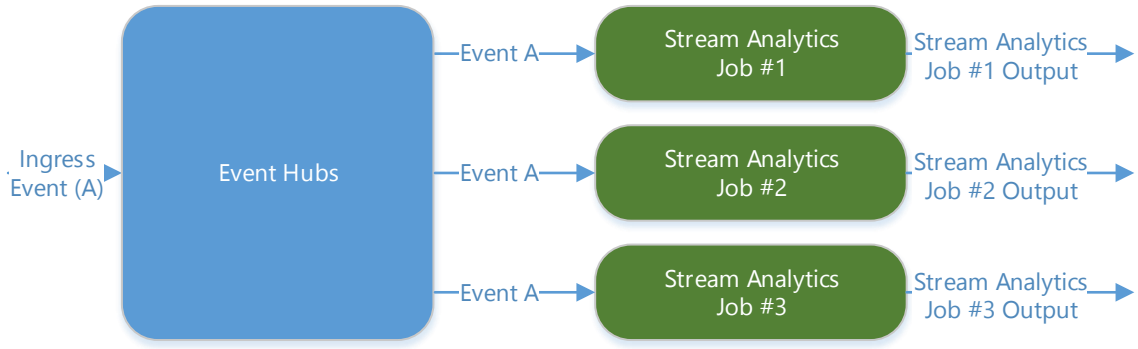

Figure 6: A single event ingested to Event Hubs can be reused across multiple Stream Analytics jobs

An example of deploying multiple streaming jobs against a single event hub could be where a separate aggregation and filter operation is required to deliver data to different outputs. One stream may be tasked with aggregating events across a hopping window and outputting this to Azure SQL Database, whereas the second stream is responsible for filtering for a given event based on a tumbling window and outputting this to a different Azure SQL Database, or possibly Blob Storage.

Aside from providing a data-integration point to consume externally generated data, the Event Hubs also supports the secondary role of consuming data that is output from the stream processing engine itself. This enables post processing data-integration with applications, or alternatively the event can be consumed into a secondary stream for additional processing. There are a number of scenarios where this secondary use case can be of value for a solution. Two examples of these include consolidating streams running across multiple Azure regions into

a single unified stream, or integrating static non-stream data to enrich a dataset prior to calling an Azure Machine Learning web service. The following diagram describes how a secondary Event Hub can be positioned within the solution to support additional downstream data processing.

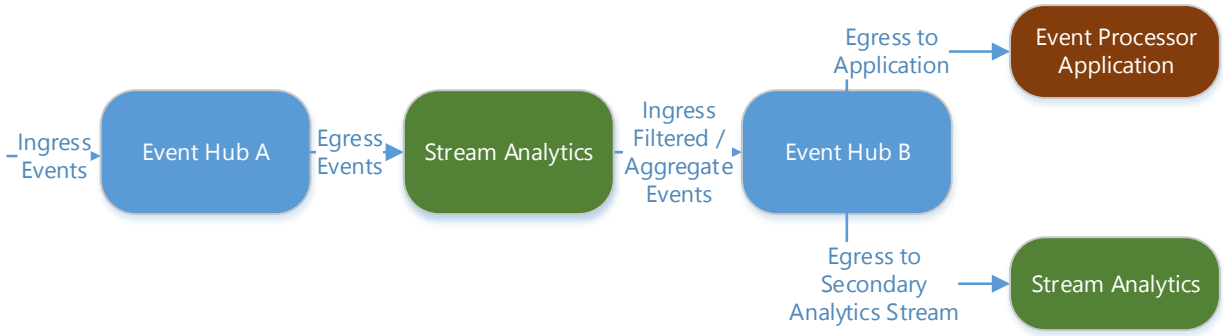

Figure 7: Example of deploying Event Hubs on either side of Stream Analytics processing

The ability of the Event Hubs service to scale consumption elastically, and by association to also scale cost elastically, is particularly important for real-time ingestion where the rate of incoming data streams can vary dramatically over a given time period. For example, some applications may provide a continual stream based on a pre-defined time period of 5 seconds, whereas other applications may only send an event based on a particular triggering event or action. Event throughput based on actions is much more difficult to predict. Additionally, connected devices can suffer from intermediate connectivity in which case a set of data will be held on the device that is waiting for a connection to be re-established before the burst is transmitted, and so even if throughput of triggered events can be forecast with reasonable accuracy, forecasting intermittent device connectivity reliably is a lot more difficult.

By deploying Event Hubs as the intermediate ingestion point of events between the data sources and the event processing engine, this architecture helps ensure that data events being generated from various source systems will be robustly consumed and persisted. Azure Stream Analytics has an Event Hubs adapter built into the offering. By providing these adapters, rather than requiring custom development, customers can accelerate their solution deployment with significantly less custom coding overhead.

5.2.2 Azure Blob Storage

Azure Storage is a scalable, durable, and highly available storage service that can support the massive volumes of data required for a variety of analytics solutions. Azure Storage accounts supports three types of storage: Blob, Table, and Queue. Only Blob Storage is applied to this real-time analytics reference architecture.

Similar to the Event Hubs, Azure Blob Storage supports capabilities that are applied at multiple layers of this architecture. Files stored in Blob Storage can be consumed as either an incoming

stream based off static data, or as reference data that will leveraged in the stream processing engine to perform joins against the incoming event stream on the fly. Additionally, Blob Storage can be used as the target location for the output data stream. Additional detail for the stream output scenario is described in a later section this document. The following diagram illustrates the input data integration with the Azure Stream Analytics engine.

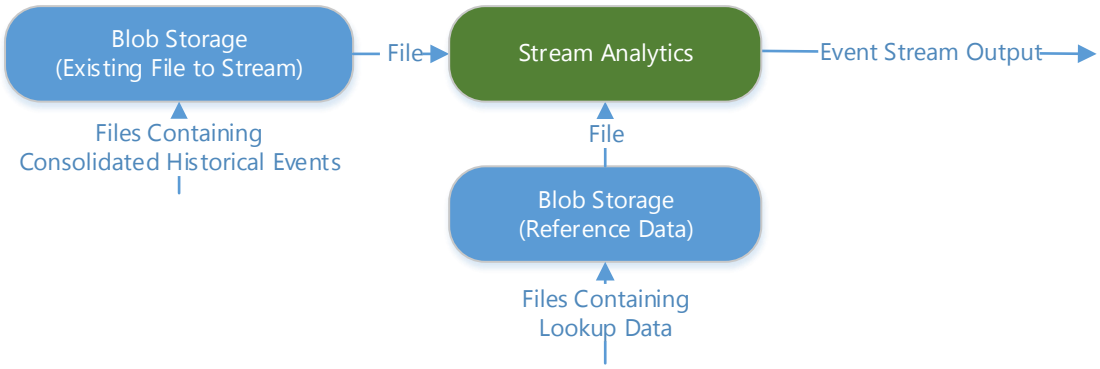

Figure 8: Applying Blob Storage as both event stream input and reference data input

Typical reference data may include geographic attributes such as country / state / city that might be joined in an incoming location identifier, or perhaps equipment part attributes that could be joined to a unique incoming part identifier. It is quite rare that all data required for real-time analytics is contained within the actual incoming stream as applications deployed to devices or sensors are generally built to be very lightweight and intentionally designed to produce minimal network traffic.

Reference data is typically quite static and will tend to be updated on a fixed schedule. One way to think of reference data is to compare it to a traditional star schema data warehouse. The reference data represents the dimensions that are typically wide tables with a number of attributes to describe the numeric fact data. They typically support aggregations, and have a one-to-many relationship with the fact tables. As opposed to the reference data, the streaming data being ingested through Event Hubs is comparable to the fact tables that are typically time based point transactions such that the transaction represents a product purchase or perhaps a single sensor reading.

The following diagram provides a high level comparison of the typical relationship between dimension and fact tables in a star schema, and how this can be compared to the relationship between reference and stream data in a real-time event processing solution.

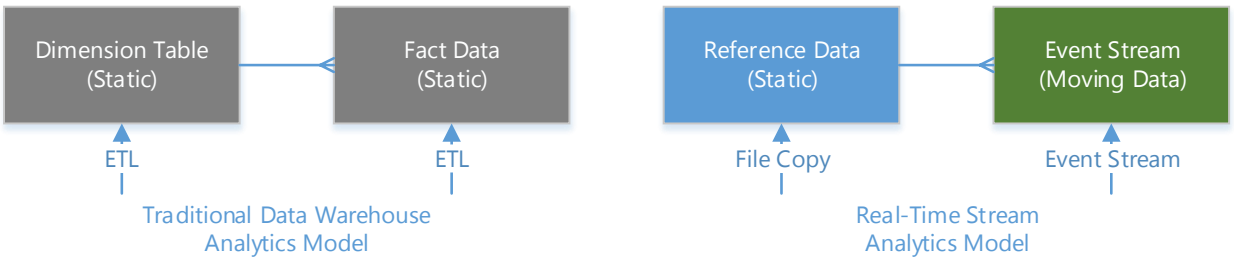

Figure 9: Comparison of traditional data warehouse star schema relationship with streaming event processing reference data relationship

To take this comparison a step further, the types of data modifications that are often applied to reference data are well aligned to the principals of slowly changing dimensions (SCD's). The majority of SCD behavior fall into 3 categories, commonly known as type 0, 1 and 2 whereby the records in a dimensions either remain unchanged except for amending new rows, have existing records updated in place, or add new records to represent an update to an existing record, but maintain the existing record to ensure accuracy in historical reporting. Changes to reference data most commonly reflect type 0 and 1, as stream analysis is intended for the analysis of current events, and not historical events. Given the similarities in the implementation approach, resources that have experience in developing traditional data warehouses or data marts are able to translate these skills and domain expertise across to streaming data solutions in Azure with very little re-training or upskill effort required.

5.2.3 Data-Integration Best Practices

Whilst it is technically possible to apply reference data stored in any Azure region to a given Stream Analytics query, it is recommended that the data is stored within the same region that the query will be executed so as to avoid the costs associated with moving data between different Azure regions. In a scenario where the solution is deployed across multiple Azure regions, the process that refreshes reference data files should be designed in a manner to ensure that similar reference files across all regions (such as a product catalog) are updated simultaneously, ensuring a consistent state across all reference files upon refresh completion.

The combination of Event Hubs to robustly consume and store ingested events at massive scale, combined with Blob Storage to support reference data typically consumed from operational systems, provides the foundation on which to develop Stream Analytics queries to support real-time analytics over moving data. The following section will outline the event processing layer and describe the downstream alternatives for outputting the stream results.

5.3 Real-Time Analytics Layer

There are two technologies that support the real-time analytics layer of this event processing reference architecture. Azure Stream Analytics acts as the linchpin of this reference architecture by providing the engine capable of temporal analytics over moving data streams. The second technology in the real-time analytics layer is Azure Machine Learning, which is a predictive analytics service capable of consuming either a single record made up of multiple columns (sometimes referred to in the context of machine learning as "features") via request / response API, or consuming a file for asynchronous batch scoring. Given the real-time scope of this document, the role of Azure Machine Learning will be based on the request / response method only.

Additionally, Apache Storm is positioned in this diagram as a reference as to where it aligns to Azure Stream Analytics. Each of these platforms support similar event processing scenarios, however there are a number of differences in their supported languages and interoperation capabilities.

5.3.1 Azure Stream Analytics

Similar to Azure Event Hubs that was described in the previous section, Azure Stream Analytics can also scale to support the processing of large volumes of events. However, while the primary function of Event Hubs is to provide a single storage entry point to robustly ingest data for short term storage, the role of Stream Analytics is to consume these events to provide temporal queries for analytics against the moving event stream for filter, aggregate, or join transformations that are then written to an output destination.

As described earlier in this document, Azure Stream Analytics ships with a pre-built adapter for Event Hubs, which significantly reduces the time to market for delivering solutions since developers do not need to develop their own custom adapters for the streaming engine. Additionally, and perhaps more significant for enabling accelerated solution deployment, Azure Stream Analytics provides a SQL-like language that allows database developers with existing SQL skills to transition to this platform very quickly. This language is very similar to T-SQL, which is the primary database language for SQL Server, however it contains a superset of functions that support temporal operations such as applying sliding, hopping or tumbling time windows to the event stream.

Azure Stream Analytics supports two different types of inputs, either stream data or reference data, and two different input data sources, either Event Hubs or files from Blob Storage. Only Blob Storage is supported for consuming reference data into the stream. Event streams can be consumed from either Blob Storage (in which case all events already exist in a file) or from Event Hubs whereby data is arriving in real-time. In the instance that data is being consumed from

Blob Storage, the file being referenced must contain a timestamp record in a supported time format for enabling Stream Analytics to perform temporal analysis. Whilst it is possible to stream a file without a timestamp field that would not enable temporal queries, the value of the solution is somewhat diminished unless there is a specific scenario that requires this functionality. The following diagram represents the different data ingestion options for Stream Analytics.

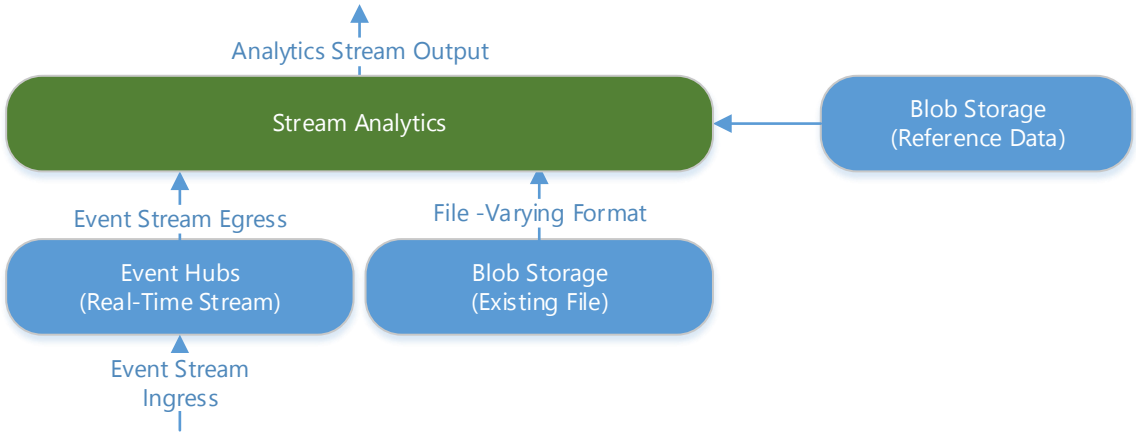

Figure 10: Input data source options for Azure Stream Analytics

Azure Stream Analytics is also capable of performing join operations between multiple streams that are sourced from multiple Event Hubs. For example, in a real-time web analytics scenario there may be two Event Hubs deployed where one captures impression events, and the other captures click events. The engine would be required to perform a join between the two incoming streams. One stream representing the impression would be consumed into the engine, and the second click stream would also be consumed and joined on one or more key values. To enable this, Azure Stream Analytics allows the join to be defined as valid for a time period so as to allow for late arriving events. For example, the above scenario may be configured to be valid for a period of 300 seconds after the impression was served. The following diagram represents multiple Event Hubs supporting a single analytics solution.

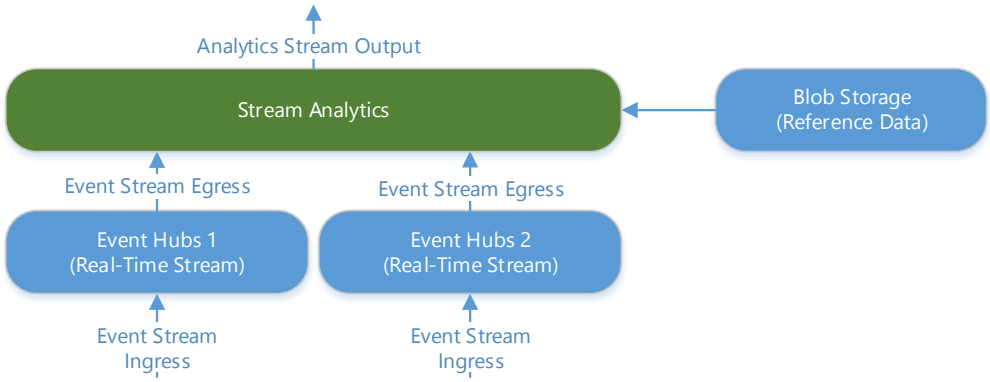

Figure 11: Example architecture for enabling multi-stream join

Deploying solutions that join streams is quite common however the complexity of the query does increase to account for the arrival of independent events that may suffer from varying latency. It is important for developers to understand the implementation scenario that is generating these events to ensure that the queries deployed can accommodate different circumstances of late arriving or out of order events.

5.3.2 Azure Machine Learning

Azure Machine Learning provides a cloud based platform for mining data to identify trends and patterns across large scale information sets. By "learning" from historical events and trends, machine learning can publish a model that can be leveraged for determining predictions in real-time based on incoming event data. Combining the real-time rules based processing capabilities of Azure Stream Analytics with the real-time predictive analytics capabilities of Azure Machine Learning can help businesses rapidly deploy highly scalable data solutions to support complex information challenges.

Machine learning models that have been developed, trained and tested, are deployed as web services that can be called from applications. A request to a model is made asynchronously, either by submitting a single record containing a number of columns, or by submitting a file that contains a number of records to be scored. For the purposes of real-time analytics, the single record request / response method is used for submitting an incoming event to the machine learning model.

One of the considerations for constructing a single record feature set to submit to machine learning models is that the incoming event stream may not contain all of the data required for the model schema. For example, telemetry that is transmitted from equipment may contain data about the performance characteristics, but it will likely not contain all of the attributes of that equipment as it is repetitive and will bloat the data transmission. Equipment attributes, such as age may be highly influential features in a real-time machine learning model and therefore need to be appended to the record prior to calling the model.

The following diagram illustrates how Azure Machine Learning can make use of streaming data to enable real-time prediction.

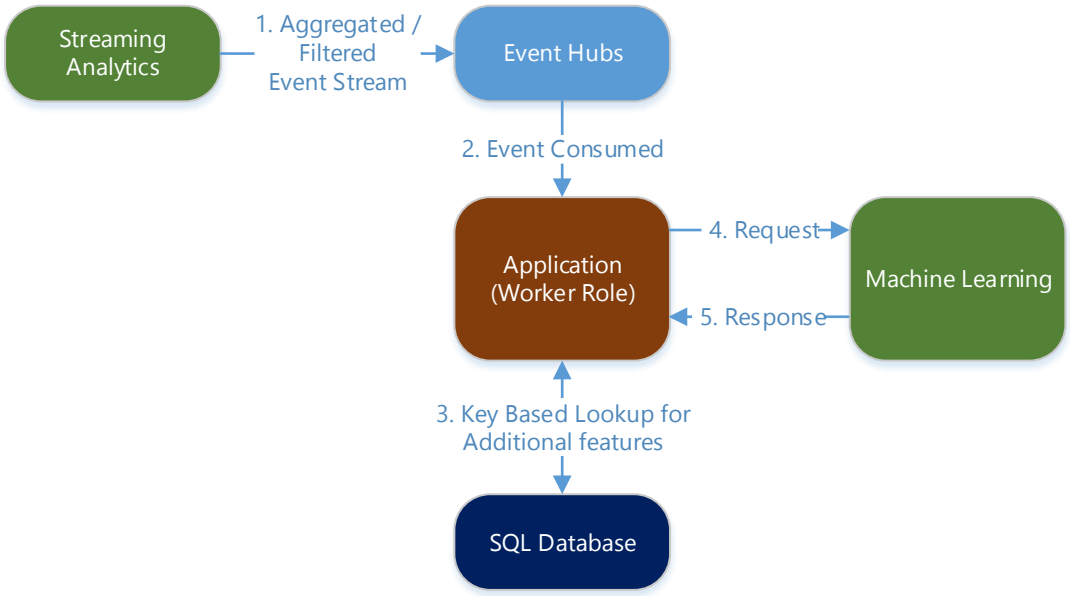

Figure 12: Architecture to support additional feature lookup on incoming data stream for machine learning request

The diagram above does not make an assumption on which action the application will perform once the prediction has been received back from the machine learning service. There are a number of possibilities here, one of which may be that the application calls a notification service (Azure notifications are beyond the scope of this document) to send an alert to a device. Other alternatives may include writing the prediction result to a SQL database for consumption into reporting analytics, or initiating a broader automated workflow.

A single solution may contain multiple models that need to be trained for predicting different outcomes based on the same or similar input data. Whilst this diagram depicts a single machine learning model, an actual implementation may comprise multiple models being called from a single orchestrating application, including scenarios where the response from one model may result in the triggering of another model request in a sequential manner.

5.4 Data Storage Layer

The role of the Data Storage Layer within this reference architecture is to provide persisted storage for the data that is output from the stream processing engine. The output of an event processing stream can be written either to Azure Blob Storage, Azure SQL Database, or even back to Event Hubs for subsequent consumption by an application or another processing stream. This section details the implementation of Blob Storage and SQL Database only as these are the primary endpoints for which a data analyst will consume the data. The implementation of Event Hubs as an output destination was described in a previous section of this document.

A real-time event processing solution is generally designed to support one or more of the following scenarios:

- Real-time ETL, aggregating to align with existing dimensions in a data model such as time, product, or customer
- Point event / pattern detection where events are filtered based on a set of one or more conditions. Events that do not meet this condition are discarded.
- Raw event capture and consolidation for historical analysis at a later time

Typically, solutions that require interactive query response rates across aggregate / filtered data will use Azure SQL Database as the destination data store, whereas solutions that capture all events for large scale analytics such as training machine learning models should leverage Azure Blob Storage as the store. HDInsight compute can then be bound to this storage on demand for processing the data.

5.4.1 Azure Blob Storage (with HDInsight)

In addition to acting as an input for event and reference data, Azure Blob Storage can also be designated as an output destination for the event stream. Blob Storage provides a massive scale data repository that can consume information at high transfer rates, and therefore is suitable to consuming raw data streams coming through the stream event processor. As events are written, they are appended to a single file within the Azure storage container. This is an important feature as it enables more efficient query processing by tools such as HDInsight which is Microsoft's distribution of Hadoop that runs in Azure.

Business users have a number of options for accessing the raw data that is stored in Azure Blob Storage. One common approach, as mentioned above, is to deploy an HDInsight cluster in Azure which effectively binds the dynamically created compute power to the persisted storage account. Hive tables can be created against the schema of the consolidated events that are output from Azure Stream Analytics. This is an efficient method of resource management as the

HDInsight compute can be shut down when analytics are complete whilst the storage account continues to consume and store events from the stream.

An alternative self-service method for consuming data from Blob Storage is through Power Query in Excel. With Power Query, users can connect directly to Azure Blob Storage using a built-in native connector and consume data to the in-memory engine within Excel. The following diagram illustrates Azure Blob Storage as a stream output.

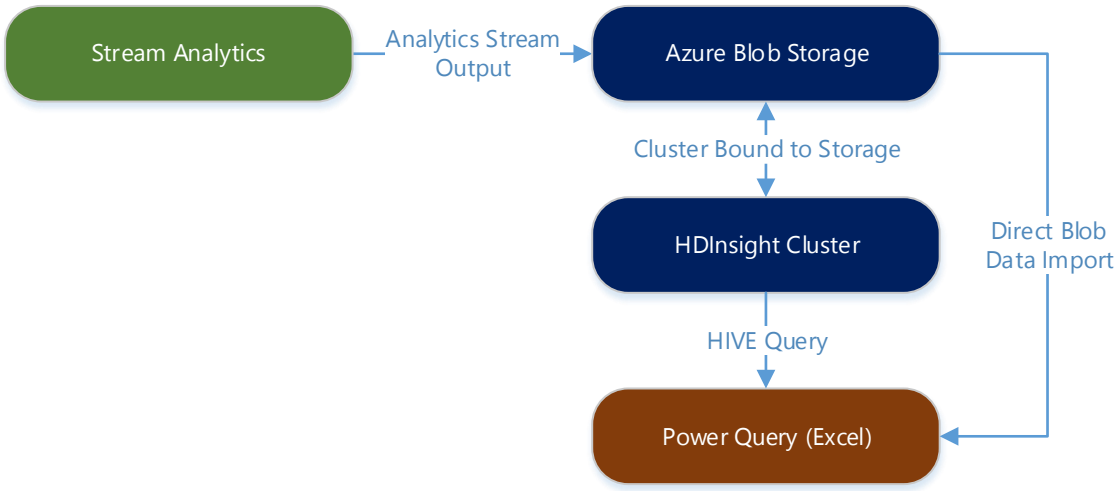

Figure 13: Architecture for large scale historical analytics on consolidated event data using HDInsight with Azure Blob Storage

5.4.2 Azure SQL Database

Azure SQL Database provides resilient, non-volatile PaaS relational databases for storing and querying data. Filtered or aggregated events flowing out of Azure Stream Analytics can be written to a database for consumption either through direct user queries or business analytics tools. Additional attributes relevant to the streaming event but not actually part of the stream can also be stored in a database to create a holistic model for analysis. Such examples of these attributes include additional time descriptions, or product attributes such as size, color or weight.

Azure Stream Analytics provides a native connector to Azure SQL Database for consuming events that are output from the stream. Business users (and business applications) can connect to a database directly to issue queries against real-time, inbound data using the same familiar T-SQL language that is part of SQL Server. Alternatively, developers can build out business intelligence solutions that either query this database directly such as reporting, or periodically repopulate cubes as an intermediate layer using the database as the source. Although the emphasis of this architecture is geared towards delivering real-time insights, and by populating

a cube we would add a degree of latency to the data, the scenario of deploying Azure Stream Analytics for the purpose of massive scale real-time ETL with slightly delayed visibility to the data remains relevant to a number of analytics applications.

The following diagram illustrates the positioning of Azure SQL Database within the real-time analytics reference architecture. Although there are many potential downstream destinations for this data such as Business Intelligence tools or operational applications, we abbreviate these as "Other Applications" to simplify the diagram. Business users are represented in this diagram as performing ad-hoc querying on the SQL database only. It is of course a valid scenario that business users would also access either Power BI or Other Applications.

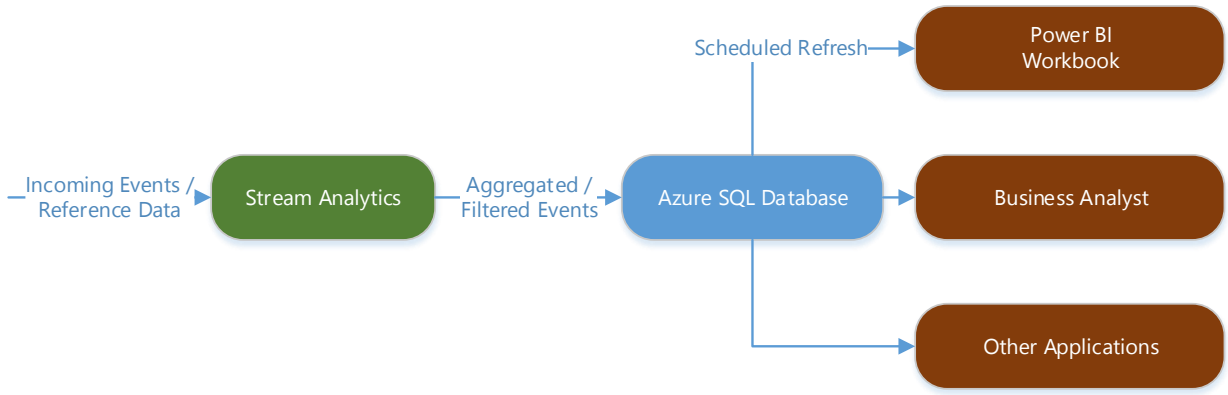

Figure 14: Integration points for Azure SQL Database to support storing and querying event data

Both the Azure SQL Database and Azure Blob Storage options in this section provide solutions for persisting event stream output, enabling users to query and / or refresh analytics models with new data as it arrives, however the requirements to issue a query for a point in time response still remain.

5.5 Presentation / Consumption Layer

The Presentation / Consumption layer supports two primary endpoints. Whereas a more traditional batch analytics solution comprised of ETL, Data Warehouses and Cubes will generally support end users only, real-time analytics solutions will commonly support both end users and/or applications that consume specific events and initiate additional automated action. The following sections describe the roles of Power BI and applications in fulfilling each of these architecture endpoints.

5.5.1 Power BI

Power BI provides a rich suite of data visualizations that enable users to dynamically interact with their information quickly and efficiently. Power BI enables data models to consume data from Azure SQL Database (amongst many other relational and non-relational data sources) and

store this data in a model hosted in Azure. This model can be scheduled to automatically refresh the data against the underlying data source on a periodic basis, or can be manually refreshed by users.

Visualizations representing data from the Azure SQL Database, in conjunction with other data sources outside of the scope of the real-time analytics solution can be centralized onto a dashboard to support a single view across all information, regardless of the underlying source. The following diagram demonstrates the connectivity method for enabling Power BI to connect with Azure SQL Database.

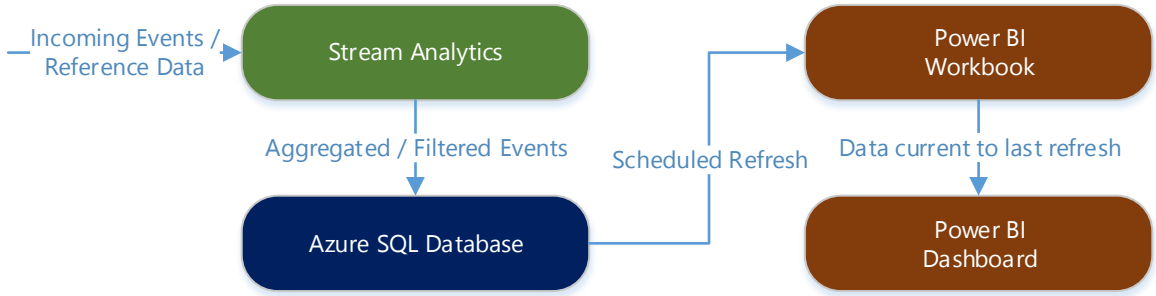

Figure 15: Applying Power BI for dashboards

While not reflected in the diagram above, Power BI also provides support for connecting Azure Blob Storage and HDInsight directly as described in a previous section of this document. This is a common analytics scenario, but not one that is generally part of an event based real-time processing solution.

5.5.2 Event Processor Application

By deploying Azure Event Hubs as an output destination for Stream Analytics events, business applications outside the scope of dashboards & reporting can be integrated into this architecture. Similar to the manner in which Stream Analytics consumes events that are ingested into input Event Hubs from external data sources, custom applications can consume events from secondary Event Hubs that are egressed from Stream Analytics. This enables a very powerful integration point for leveraging the real-time insights of Azure Stream Analytics for rapid automated process execution across an organization.

Solutions designed to detect situations that required a rapid response, such as bursts in activity, low inventory levels, fraudulent transactions or irregular data outliers can integrate into the existing business systems that support these processes to form part of a larger automated workflow. Similar to the Azure SQL Database output, the Event Hubs output could also support a diverse set of business applications. For the purposes of this reference architecture we bucket all of these potential application uses as “Downstream [Consumption/Processing]”. The following diagram positions the Event Processor application within the solution, labelled as “Custom Application”.

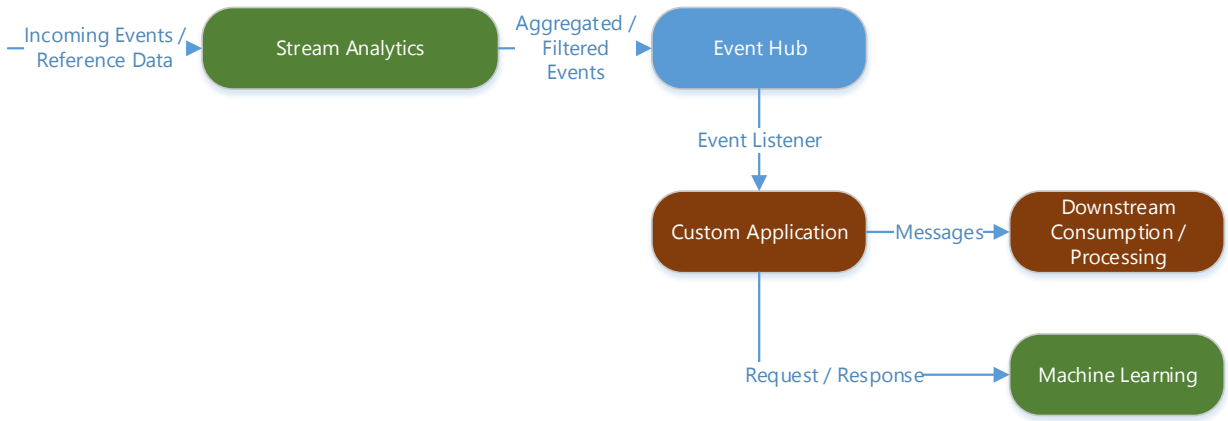

Figure 15: Consuming events to enable application consumption / machine learning

Defining the architecture for [consumption/processing] by downstream business applications is beyond the scope of this document. When enabling event [consumption/processing] with business applications, there are a number of additional factors to consider which may include how applications will communicate, and potential rework to existing applications for supporting new data.

5.6 Connected Architecture

The previous sections of this document have described the individual pieces of each layer in this reference architecture, the capabilities they provide, and the interactions between each of these components. This high level diagram provides a holistic view of the entire reference architecture end to end.

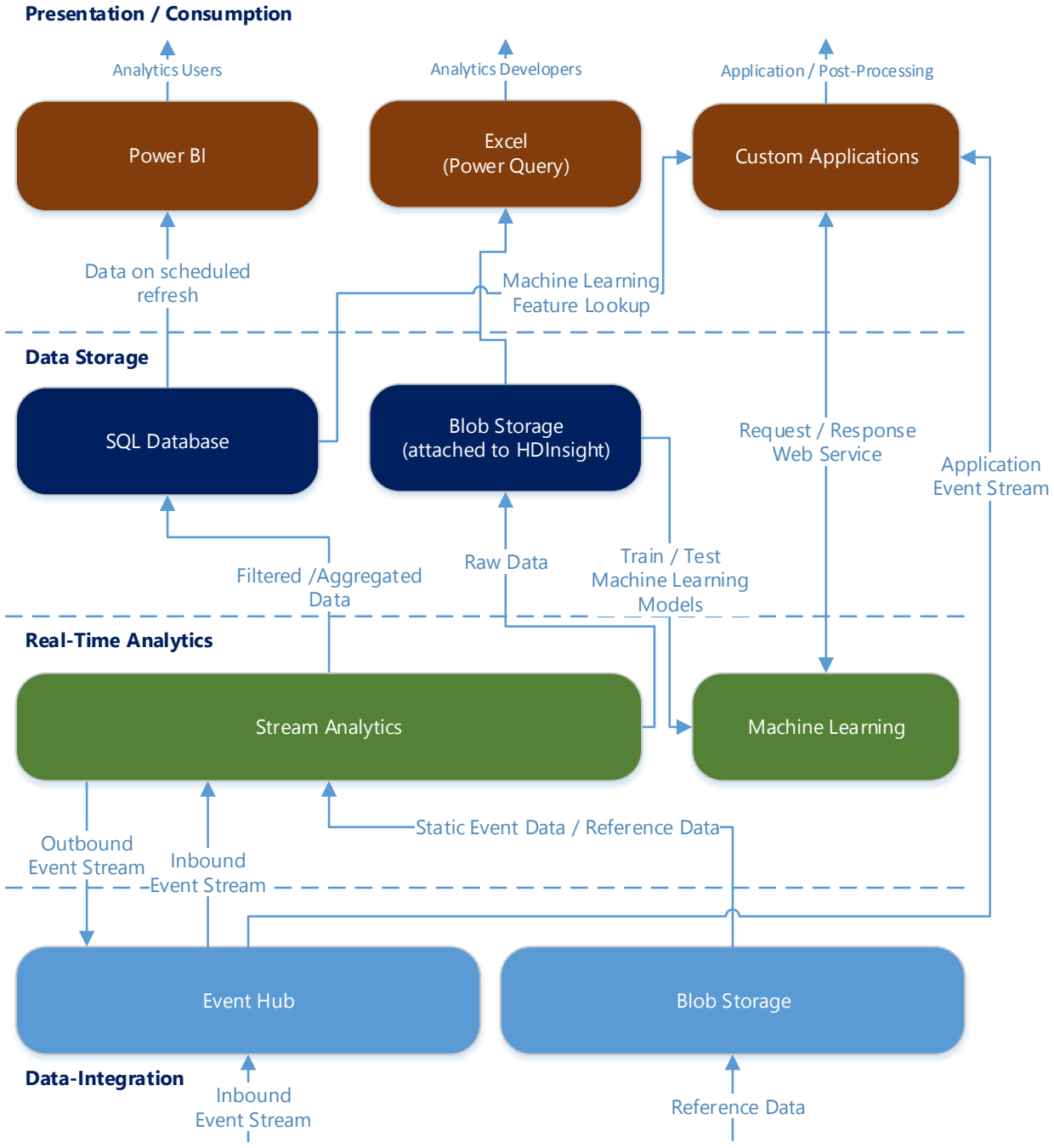

Figure 17: All-up architecture to support real-time event based analytics in Microsoft Azure

6. Conclusion

Businesses looking for an approach for accelerating their time to insight should consider event processing implementations as an ideal solution for their needs. By complementing traditional batch based analytics implementations with real-time analytics, and unifying the data visualizations experience into a single dynamic interface, companies can empower their staff to decide and act on information faster.

The reference architecture for real-time event based analytics is intended to serve as a blueprint for designing and deploying event processing solutions with Microsoft Azure. Customers can benefit from the cost and performance elasticity delivered by cloud based PaaS offerings to deploy flexible, efficient solutions that can scale over time based on business needs as well as changing market dynamics. Power BI as a SaaS offering not only supports the reporting and visualizations required of this architecture, but can be extended more broadly across an organization to support additional workloads.